UNIVERSIDAD CARLOS III DE MADRID

Tesis Doctoral

Algoritmos de Geometría Diferencial para la Locomoción y Navegación Bípedas de Robots Humanoides
Aplicación al robot RH0

Autor:
José Manuel Pardos Gotor

Director:
Prof. Dr. Carlos Balaguer Bernaldo de Quirós

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Leganés, Septiembre de 2005

TESIS DOCTORAL

Algoritmos de Geometría Diferencial para la Locomoción y Navegación Bípedas de Robots Humanoides Aplicación al Robot RH0.

Autor: José Manuel Pardos Gotor

Director: Prof. Dr. Carlos Balaguer Bernaldo de Quirós

Firma del Tribunal Calificador:

Firma

Presidente:

Vocal:

Vocal:

Vocal:

Secretario:

Calificación:

Leganés, de de

Agradecimientos

Al director de esta tesis, el Profesor D. Carlos Balaguer Belnaldo de Quirós, por sus aportaciones, comentarios, orientación, por haber puesto a mi disposición los medios necesarios y sobre todo, por haberme concedido la libertad y el tiempo necesarios para explorar nuevos caminos.

Por su ayuda y comprensión, a los miembros del proyecto "Robot Humanoide RH0": Mario Arbulu, Daniel Gutierrez, Dimitry Kaynov, Ignacio Prieto, Luis Cabas, Pavel Staroverov, Ramiro Diez, Carlos Pérez, Carlos Martin y Francisco Jesús Prieto.

A los miembros del Área de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid, por todo lo que allí he aprendido.

A los profesores R.M. Murray y J.A. Sethian, por sus extraordinarios trabajos que han sido la principal inspiración de esta tesis.

A Guillermo y Aitana por su tiempo. A Nuria, por todo.

Resumen

"El hombre es la medida de todas las cosas..." – Protágoras (S. V a.C.) –

Los humanos crean entornos adecuados para ser habitados por ellos mismos, por lo que un robot humanoide es un instrumento muy bien adaptado para proporcionar muchos servicios a las personas. Sin embargo, todavía nos encontramos lejos de una producción comercial masiva de humanoides fiables y útiles para la sociedad. Una de las principales razones que justifican la situación actual es el formidable desafío computacional que presentan estos sistemas mecánicos, debido a la complejidad dada por el gran número de restricciones y grados de libertad.

Cuando la complejidad es grande, la necesidad de formulaciones matemáticas elegantes se convierte en un asunto de extrema importancia, porque nos permite construir soluciones eficaces. Por ello, este trabajo aborda la investigación en robótica utilizando técnicas de **Geometría Diferencial**, basadas en la teoría matemática de **Grupos y Álgebras de Lie** y herramientas de Geometría Computacional para el análisis de interfaces en evolución. Estas formulaciones conducen a aplicaciones con soluciones cerradas y completas, numéricamente estables y con una clara interpretación geométrica.

Esta tesis pionera en el campo de la investigación con robots, tiene como objetivo fundamental la resolución completa del problema de **Locomoción y Navegación Bípeda de Robots Humanoides**. Para ello, desarrolla nuevos modelos y algoritmos geométricos de propósito general, no presentados anteriormente en la literatura. Estas nuevas soluciones son potentes, flexibles y válidas para aplicaciones en tiempo real. **El nuevo algoritmo "Un Paso Adelante" (UPA)**, resuelve la locomoción bípeda de un humanoide, basándose en **el nuevo modelo "División Cinemática Sagital" (DCS)**, que da soluciones cerradas al problema cinemático inverso del robot. **El nuevo algoritmo "Método Modificado de Marcha Rápida" (M3R)** proporciona trayectorias libres de colisiones para resolver problemas de planificación, sea cual fuere la estructura del entorno de trabajo. Para la navegación del robot humanoide, introducimos **el nuevo modelo "Trayectoria Corporal Global" (TCG)**. Se ha creado un **nuevo Simulador de Realidad Virtual (RobManSim)** para robots, que permite desarrollar las teorías presentadas. Los nuevos modelos y algoritmos introducidos en esta tesis, se han probado con éxito en experimentos reales con el humanoide **RH0** de la **Universidad Carlos III de Madrid**.

Sinceramente, creemos que los mejores diseños y aplicaciones son concebidos con elegancia de pensamiento. Esta es la idea que ha inspirado los trabajos de esta tesis, para acercar siquiera en algo, ese futuro de humanoides socialmente útiles, diseñados a la medida del hombre.

Abstract

"Man is the measure of all things..." – Protagoras (Vth century B.C.) –

The humankind creates environments suited to be inhabited by them, therefore a humanoid robot is a tool very well adapted to provide a lot of services to people. Nevertheless, we are still far from a commercial production of reliable humanoids really useful to our society. One of the main reasons which justify the current situation is the formidable computational challenge presented by these mechanical systems, mainly because of the complexity given by the high number of restrictions and degrees of freedom.

When the complexity is huge, the need for some elegant mathematical formulations becomes a paramount issue, because it allows us to build up efficient solutions. Therefore, this work explores the research on robotics using some **Differential Geometry** techniques based on the mathematical theory of **Lie Groups and Algebras**, and some Computational Geometry tools from the analysis of evolving interfaces. These formulations lead to applications with closed and complete solutions, which are numerically stable, and with a very clear geometrical interpretation.

This pioneering thesis on the field of research with robots has a fundamental goal; that is to obtain the complete solution for ***the Humanoid Robot Bipedal Locomotion and Navigation problem***. For doing so, it develops new geometric models and algorithms of general purpose, which have not been presented in the literature before. These new solutions are powerful, flexible and valid to real time applications. **The new algorithm "One Step Goal" (OSG),** solves the bipedal locomotion based upon **the new humanoid model called "Sagittal Kinematics Division" (SKD)**, which provides closed solutions for the robot inverse kinematics problem. **The new algorithm "Fast Marching Method Modified" (FM3)** delivers collision-free trajectories to solve the path planning problems, whatever the structure of the working environment. For the humanoid robot navigation problem, **the new model "Whole Body Trajectory" (WBT)** is introduced. **A new Virtual Reality Simulator (RobManSim)** for humanoid robots has been created, in order to develop the herein presented theories. The new models and algorithms introduced by this thesis have been successfully tested through real experiments with the **humanoid RH0** of the **University Carlos III of Madrid**.

Sincerely, we believe that the best designs and applications are conceived with elegance o mind and this is the inspirational idea for the new developments of this thesis, in order to bring a little bit closer, a future of socially useful humanoids made up to the measure of the mankind.

Índice

Índice de figuras

Nomenclaturas

Por orden de aparición en el texto.

SE(3) Grupo de Lie Especial Euclídeo tridimensional – *Special Euclidean Group*
se(3) Álgebra de Lie del Grupo Especial Euclídeo tridimensional
SO(3) Grupo de Lie Especial Ortogonal tridimensional – *Special Orthogonal Group*
so(3) Álgebra de Lie del Grupo Especial Ortogonal tridimensional
$\mathbf{R^n}$ Espacio n-dimensional de números reales
Ev Espacio vectorial
$\mathbf{R^3}$ Espacio tridimensional de números reales
g Matriz de transformación homogénea
d Vector de Traslación o Desplazamiento 3x1
R Matriz de Rotación 3x3
I Matriz Identidad
$\boldsymbol{\xi}^\wedge$ *Twist* – Movimiento representado como matriz 4x4
$\boldsymbol{\upsilon}, \boldsymbol{\upsilon}_i$ Vector traslacional del movimiento - Componente del *twist*
$\boldsymbol{\omega}^\wedge$ Álgebra de **so(3)** – Transformación del producto vectorial en matricial
[,] *Lie Bracket* - Operación definida entre dos *twists* de **SE(3)**
$\boldsymbol{\xi}, \boldsymbol{\xi}_i$ *Twist* – Movimiento representado como vector 6x1
$\boldsymbol{\omega}, \boldsymbol{\omega}_i$ Vector rotacional del movimiento - Componente del *twist*
$[\,]^\vee$ *Vee* – Operación que transforma $\xi^\wedge$ en ξ
$[\,]^\wedge$ *Wedge* – Operación que transforma ξ en $\xi^\wedge$
S *Spatial System* – Sistema de referencia Espacial
B *Body System* – Sistema de referencia Operacional
$\mathbf{Ad_g}$ Transformación Adjunta
$\boldsymbol{\theta}, \boldsymbol{\theta}_i$ Coordenadas generalizadas – Valor de los GDL – Ángulos en rotaciones y desplazamientos en traslaciones
$\mathbf{h}_\xi$ *Pitch* de un *twist*
$\mathbf{l}_\xi$ Eje de un *twist*
$\mathbf{M}_\xi$ Magnitud de un *twist*
$\boldsymbol{\lambda}$ Valor real constante
p, p' Punto – Vector que representa un punto cualquiera en $\mathbf{R^3}$

$\mathbf{r, r', r_i}$	Punto – Vector que representa un punto cualquiera en $\mathbf{R^3}$
$\boldsymbol{\varphi}^\wedge$	*Wrench* – Esfuerzos representados como matriz 4x4
$\boldsymbol{f, f_i}$	Vector lineal de fuerzas - Componente del *wrench*
$\boldsymbol{\varphi, \varphi_i}$	*Wrench* – Esfuerzos representados como vector 6x1
$\boldsymbol{\tau, \tau_i}$	Vector rotacional de pares - Componente del *wrench*
$\mathbf{h_\varphi}$	*Pitch* de un *wrench*
$\mathbf{l_\varphi}$	Eje de un *wrench*
$\mathbf{M_\varphi}$	Magnitud de un *wrench*
n	Número de **GDL**, de articulaciones o de dimensiones
$\mathbf{g(\theta)}$	Fórmula del **POE** para representar una cinemática directa
$\mathbf{g(0)}$	Posición de referencia para un sistema coordenado
$\mathbf{J^s}$	Manipulador Jacobiano Espacial
$\mathbf{V^s}$	Velocidades en el sistema de referencia espacial **S**
$\mathbf{J^b}$	Manipulador Jacobiano Operacional
$\mathbf{V^b}$	Velocidades en el sistema de referencia operacional **B**
H	Sistema de referencia de la Herramienta del robot
$\mathbf{g_{sh}(\theta)}$	Configuración de **H** con respecto a **S** en función $\boldsymbol{\theta}$ - cinemática directa
$\mathbf{g_{sh}(0)}$	Configuración de **H** con respecto a **S** en la referencia $\boldsymbol{\theta=0}$
k	Punto – Vector que representa un punto cualquiera en $\mathbf{R^3}$
u, v	Vectores auxiliares para formulación de problemas canónicos geométricos
u', v'	Vectores auxiliares proyecciones de **u** y **v** sobre un plano
c	Punto en $\mathbf{R^3}$ donde se cruzan los ejes de dos *twists*
$\boldsymbol{\alpha, \beta, \gamma}$	Coeficientes geométricos para la obtención del punto **c** en Paden-Kahan-Dos
$\boldsymbol{\delta}$	Distancia - Valor real constante
L	Lagrangiano
K	Energía Cinética
V	Energía Potencial
$\boldsymbol{\Gamma}$	Fuerzas Generalizadas
$\mathbf{M(\theta)}$	Matriz de Inercia del Manipulador
$\boldsymbol{\mu_i}$	Matriz de Inercia Generalizada para un eslabón
$\mathbf{m_i}$	Masa de un eslabón
$\boldsymbol{\Psi_i}$	Tensor de Inercia de un eslabón
C	Matriz de Coriolis
$\mathbf{C_{ij}}$	Elementos de la Matriz de Coriolis
N	Matriz de Potencial
$\boldsymbol{\Gamma_{ijk}}$	Símbolos de Christoffel
$\mathbf{M_{ij}}$	Términos de la matriz $\mathbf{M(\theta)}$
$\mathbf{A_{ij}}$	Transformación adjunta del manipulador
$\boldsymbol{\mu_i^*}$	Matriz de Inercia Transformada para un eslabón
$\mathbf{l_i}$	Eslabones de la cadena cinemática del manipulador
$\mathbf{g_{sli}(0)}$	Configuración de los eslabones $\mathbf{l_i}$ con respecto a **S** en la referencia $\boldsymbol{\theta=0}$
$\boldsymbol{\theta_d}$	Trayectoria de referencia en las coordenadas de las articulaciones
e	Error de posición
$\mathbf{K_v}$	Matriz de ganancia constante de Velocidad
$\mathbf{K_p}$	Matriz de ganancia constante de Posición
$\mathbf{x_d}$	Trayectoria de referencia en las coordenadas de la herramienta
M~	Matriz Efectiva de Inercia del Manipulador (en el espacio Operacional)
C~	Matriz Efectiva de Coriolis (en el espacio Operacional)
N~	Matriz Efectiva de Potencial (en el espacio Operacional)

$\Gamma\sim$	Fuerzas Efectiva Generalizadas (en el espacio Operacional)
$\mathbf{J}$	Matriz Jacobiana
$\mathbf{A_{j,j+1}}$	Matriz de Transformación Relativa de un eslabón al siguiente de **D-H**
θ_j	Parámetro de **D-H** para giro en el eje de la articulación Y.
$\mathbf{d_j}$	Parámetro de **D-H** para traslación giro en el eje de la articulación Y.
$\mathbf{a_j}$	Parámetro de **D-H** para traslación en el eje X.
α_j	Parámetro de **D-H** para giro en el eje X.
$\mathbf{A_s}$	Área de soporte de la locomoción bípeda
Ca	Cadencia – Número de pasos por unidad de tiempo
β	Factor de Función - % tiempo que una pierna se encuentra en etapa de soporte
Fr	Número de Froude
$\mathbf{S_D}$	Tiempo de Discretización de los datos de un paso.
$\mathbf{H_p}$	Altura de Paso
$\mathbf{L_z}$	Longitud de zancada
$\mathbf{L_p}$	Longitud de Paso
$\mathbf{W_p}$	Anchura de Paso
$\mathbf{H_{CM}}$	Altura del Centro de Masas
$\mathbf{Po_{CM}}$	Potencial para generación de la Trayectoria del Centro de Masas en plano XZ
$\mathbf{K_{CM}}$	Coeficiente coseonoidal para generación de la Trayectoria del CM en plano XY
$\mathbf{P_{CM}}$	Proyección del Centro de Masas
$\mathbf{G_{CM}}$	Giro máximo que puede realizar el Centro de Masas en un solo movimiento
$\mathbf{RA_{as}}$	Radio del área de soporte del pie fijo.
$\mathbf{P_{CP}}$	Proyección del Centro de Presión
$\mathbf{S_S}$	Sistema de referencia del Suelo – *Spatial System*
$\mathbf{S_H}$	Sistema de referencia del Tronco – Sistema de la Herramienta
$\mathbf{V_l}$	Velocidad de Locomoción
$\mathbf{V_{rp}}$	Velocidad Relativa del Pie
$\theta\mathbf{v}$	Grados de Libertad virtuales o calculados
PM	Pie Móvil
PF	Pie Fijo
$\theta\mathbf{v_{PMi}}$	Un **GDL** virtual del **PM**
$\theta\mathbf{v_{CMi}}$	Un **GDL** virtual del **CM**
$\mathbf{Q_{rPM}}$	Camino factible del Pie Móvil **PM**
$\mathbf{Q_{rCM}}$	Camino factible del Centro de Masas **CM**
$\mathbf{q_{PMi}}$	Una configuración del **PM**
$\mathbf{q_{CMi}}$	Una configuración del **CM**
$\mathbf{A_{sPM}}$	Área de Soporte del Pie Móvil **PM**
$\mathbf{A_{sPF}}$	Área de Soporte del Pie Móvil **PF**
Pd	Pasillo de dirección para la locomoción bípeda
υ_l	Eje longitudinal
υ_d	Eje de dirección
υ_{PM}	Eje de avance del **PM**
$\mathbf{q_{ol}}$	Configuración de Objetivo Local
$\mathbf{q_h}$	Configuración del Humanoide
$\mathbf{q_o}$	Configuración de los Obstáculos
$\mathbf{Q_h}$	Espacio total de configuraciones del Humanoide
$\mathbf{Q_i(q_h)}$	Subconjunto de $\mathbf{Q_h}$ ocupado por una configuración específica del humanoide
$\mathbf{O_i(q_o)}$	Conjunto de configuraciones ocupadas por obstáculos
$\mathbf{Q_{hl}}$	Espacio libre de configuraciones para el Humanoide

$\mathbf{Q_r}$	Camino factible
$\mathbf{q_s}$	Configuración Inicial - *Start*
$\mathbf{q_f}$	Configuración Final
$\mathbf{q_i}$	Configuración i-ésima
$\boldsymbol{\gamma}$	Camino - Parametrización de una Trayectoria
$\mathbf{q_n}$	Configuración alcanzable
G	Valor de discretización de una dimensión – Número de celdas
F	Velocidad de avance de un frente
Lo	Factores y Propiedades locales
Gl	Factores y Propiedades Globales
In	Factores y Propiedades Independientes
T	Función de Llegada – Tiempo de Alcance de un frente en expansión
$\boldsymbol{\Lambda}$	Frente circular
HJ	Ecuación de Hamilton Jacobi
$\mathbf{D_T}$	Derivadas parciales de la función **T**
t	Tiempo
$\mathbf{D^{\theta i}T}$	Derivadas espaciales de **T** con respecto a las dimensiones $\boldsymbol{\theta}$ del espacio
z	Posición z-ésima de una estructura de datos
A, B	Puntos – Vectores que representan un puntos cualesquiera en $\mathbf{R^2}$
kd, md	Puntos – Vectores que representan puntos en $\mathbf{R^3}$ de la pierna derecha
pd, rd	Puntos – Vectores que representan puntos en $\mathbf{R^3}$ de la pierna derecha
kz, mz	Puntos – Vectores que representan puntos en $\mathbf{R^3}$ de la pierna izquierda
pz, rz	Puntos – Vectores que representan puntos en $\mathbf{R^3}$ de la pierna izquierda
θvzi	Grados de Libertad virtuales para el pie izquierdo del **RH0**
θvdi	Grados de Libertad virtuales para el pie derecho del **RH0**
$\mathbf{Q_{rZ}}$	Camino factible para las configuraciones del pie izquierdo
$\mathbf{Q_{rD}}$	Camino factible para las configuraciones del pie derecho
$\mathbf{q_{Zi}}$	Una configuración del pie izquierdo
$\mathbf{q_{Di}}$	Una configuración del pie derecho
MD	Manipulador derecho del **RH0** según su **DCS**
MZ	Manipulador izquierdo del **RH0** según su **DCS**
$\boldsymbol{\Gamma_D}$	Fuerzas Generalizadas de las articulaciones del **MD**
$\mathbf{M_D}$	Matriz de Inercia del **MD**
$\mathbf{C_D}$	Matriz de Coriolis del **MD**
$\mathbf{N_D}$	Matriz de Potencial del **MD**
$\boldsymbol{\theta_D}$	Trayectorias de las articulaciones del **MD**
$\boldsymbol{\Gamma_Z}$	Fuerzas Generalizadas de las articulaciones del **MZ**
$\mathbf{M_Z}$	Matriz de Inercia del **MZ**
$\mathbf{C_Z}$	Matriz de Coriolis del **MZ**
$\mathbf{N_Z}$	Matriz de Potencial del **MZ**
$\boldsymbol{\theta_Z}$	Trayectorias de las articulaciones del **MZ**
$\boldsymbol{\theta_{dD}}$	Trayectoria de referencia en las coordenadas del **MD**
$\mathbf{e_D}$	Error de posición del **MD**
$\mathbf{K_{vD}}$	Matriz de ganancia constante de Velocidad del **MD**
$\mathbf{K_{pD}}$	Matriz de ganancia constante de Posición del **MD**
$\boldsymbol{\theta_{dZ}}$	Trayectoria de referencia en las coordenadas del **MZ**
$\mathbf{e_Z}$	Error de posición del **MZ**
$\mathbf{K_{vZ}}$	Matriz de ganancia constante de Velocidad del **MZ**
$\mathbf{K_{pZ}}$	Matriz de ganancia constante de Posición del **MZ**

Acrónimos

CM	Centro de Masas
CP	Centro de Presión
DCS	División Cinemática Sagital
D-H	Denavit Hartenberg (Parámetros)
FMM	*Fast Marchig Methods* – Métodos de Marcha Rápida
GDL	Grados De Libertad
MATLAB®	MAtrix LABoratory – Programa de Software Comercial
M3R	Método Modificado de Marcha Rápida
P	Proporcional (tipo de Control)
PD	Proporcional Derivado (tipo de Control)
PID	Proporcional Integral Derivado (tipo de Control)
PM	Planificación de Movimientos
PEA	Posición Extrema Anterior
PEP	Posición Extrema Posterior
POE	*Product Of Exponentials* – Producto de Exponenciales
RH0	Robot Humanoide modelo-0 (Universidad Carlos III de Madrid)
TCG	Trayectoria Corporal Global
UPA	Un Paso Adelante
VRML	*Virtual Reality Modelling Language*
X3D	*Extensible 3D (Three Dimensional) Graphics*
ZMP	*Zero Moment Point* – Punto de Momento Cero

1 Introducción

El análisis mecánico de un robot Humanoide representa un esfuerzo computacional enorme, especialmente para aplicaciones en tiempo real. Esto es debido al gran número de ***GDL*** *y restricciones, los complejos modelos cinemáticos y los requisitos de equilibrio de la locomoción bípeda. Cuando la complejidad de las ecuaciones de movimiento es grande, es cuando se hacen necesarias formulaciones matemáticas elegantes que nos permitan desarrollar soluciones eficaces. Es por ello, por lo que esta tesis investiga la mecánica y control de robots utilizando técnicas de Geometría Diferencial. Se estudia la Locomoción Bípeda del humanoide basándose en la teoría matemática de* ***Grupos y Álgebras de Lie*** *y la Navegación del robot usando herramientas de* ***Geometría Computacional para el análisis de interfaces*** *en evolución. Introducimos en este capítulo las motivaciones que nos empujan a desarrollar la tesis en línea con estas investigaciones. También presentamos los trabajos, tanto históricos como del estado del arte, relativos a este campo de la ciencia, que son el fundamento de las nuevas ideas introducidas en esta tesis. Para finalizar, este capítulo repasa los objetivos y contenido de la tesis, que serán discutidos en capítulos posteriores.*

1.1 Motivación

El estudio de sistemas mecánicos que interactúan entre sí, ha recibido el interés de los científicos durante cientos de años. Sin embargo, los sistemas mecánicos muy complejos, como lo son los robots humanoides, padecen lo que se conoce como maldición dimensional, esto es, el incremento de la dificultad del problema de forma exponencial con el número de **GDL**, haciendo que los problemas con un espacio dimensional elevado sean muy difíciles de manejar.

Cualquiera que haya experimentado la necesidad de obtener las ecuaciones del movimiento de un robot, se habrá dado cuenta de la enorme complejidad que encierra la resolución de las mismas. Y esto, a pesar de que desde el punto de vista teórico de la mecánica clásica, conseguir las ecuaciones de movimiento de un conjunto de sólidos rígidos acoplados puede parecer relativamente sencillo, una vez que tenemos un sistema de coordenadas de referencia y aplicamos las ecuaciones de **Newton-Euler** o las de **Lagrange**, para obtener las ecuaciones diferenciales correspondientes.

Cuando el sistema es un humanoide, como por ejemplo el **RH0** desarrollado en la **Universidad Carlos III de Madrid** (ver Figura 1-1), que vamos a utilizar para experimentar los desarrollos de esta tesis, la complejidad del análisis mecánico crece aún más. En primer lugar porque el número de grados de libertad es muy grande (e.g. 21 **GDL** en el caso del **RH0**), en segundo lugar porque aparecen los problemas de locomoción bípeda estable y en tercer lugar porque necesitamos resolver importantes problemas de planificación de trayectorias libres de colisiones, tanto para el cuerpo del humanoide como para sus articulaciones.

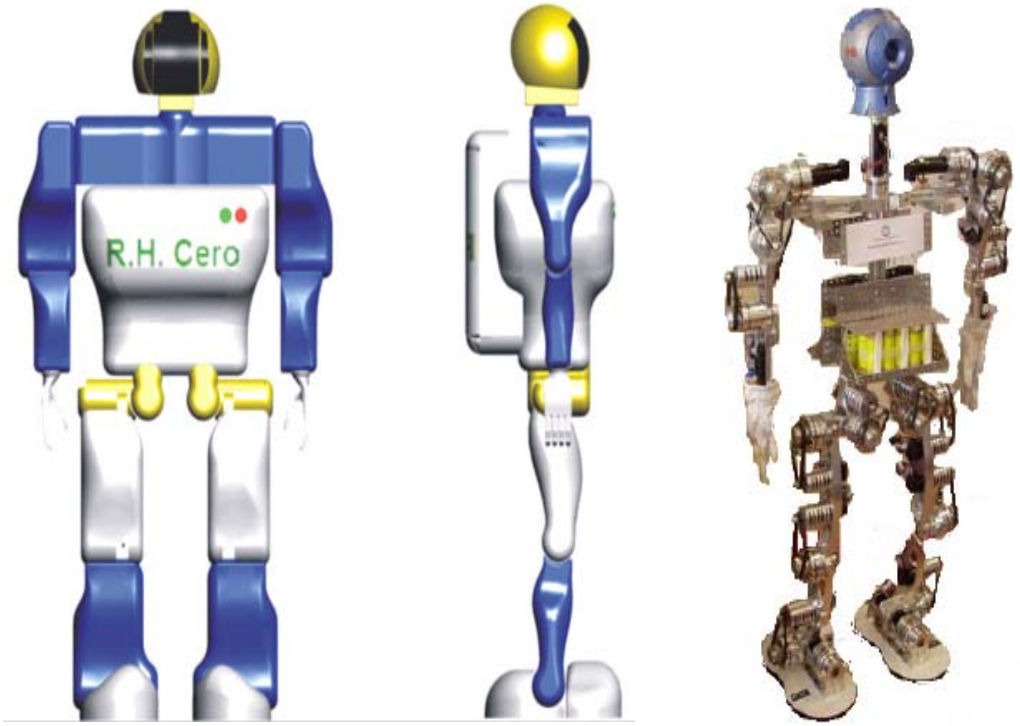


Figura 1-1: Robot Humanoide RH0 de la Universidad Carlos III de Madrid.

La necesidad de formulaciones sencillas para las ecuaciones de movimiento, se convierte en un asunto de vital importancia conforme aumenta la complejidad de los robots y de sus métodos de control. Es deseable tener una representación explícita de estas ecuaciones que pueda ser manipulada a alto nivel y donde los parámetros cinemáticos y dinámicos del sistema mecánico puedan ser representados de una forma transparente. Muchas aplicaciones de planificación y control de robots necesitan ecuaciones que puedan ser derivadas explícitamente con respecto a ciertos parámetros de interés. Además, los algoritmos utilizados deben ser independientes de los sistemas de coordenadas elegidos (i.e., algoritmos no ligados a ningún sistema de referencia local), para que resulten flexibles en los análisis cinemáticos y dinámicos. Para el control de humanoides en tiempo real, en términos prácticos existe la necesidad de algoritmos con solución cerrada, frente a otro tipo de implementaciones cuya convergencia no está garantizada y que consumen mucho tiempo de computación.

En esta tesis se crean, presentan y experimentan nuevos métodos analíticos que exhiben muchas de las deseables propiedades mencionadas en el párrafo anterior. Usando técnicas y herramientas de **Geometría Diferencial,** de la teoría matemática de **Grupos y Álgebras de Lie** para la mecánica del sólido rígido y de **Geometría Computacional** para el seguimiento de **interfaces en evolución**, se desarrollan algoritmos geométricos que son soluciones cerradas numéricamente estables.

En la búsqueda de una ***solución completa*** para **el problema de la Locomoción y Navegación Bípedas de Robots Humanoides**, este trabajo desarrolla nuevos algoritmos geométricos eficaces, no introducidos en trabajos anteriores, que son válidos para aplicaciones en tiempo real y por su generalidad pueden usarse como base para otras aplicaciones y con otros robots. Una de las principales contribuciones de la tesis es **el nuevo algoritmo "Un Paso Adelante" (UPA)**, que resuelve de forma genérica la Locomoción Bípeda de un humanoide, teniendo como núcleo **el nuevo modelo "División Cinemática Sagital" (DCS)**, que da soluciones cerradas al problema cinemático inverso completo del robot. Otra contribución fundamental de la tesis es **el nuevo algoritmo "Método Modificado de Marcha Rápida" (M3R)**, que soluciona de forma analítica la planificación de trayectorias libres de colisiones para, sea cual fuere la naturaleza de los obstáculos (i.e.: cóncavos, convexos, interconectados, irregulares o incluso mal definidos). Para la Navegación Bípeda del humanoide introducimos **el nuevo modelo "Trayectoria Corporal Global" (TCG)**, que se basa en la aplicación del **M3R** para calcular las trayectorias tanto del cuerpo como de los miembros del humanoide. A lo largo de la tesis aparecen otras ideas y aportaciones de menor nivel, pero también originales, como es **el nuevo problema canónico de Pardos-Uno**, que permite utilizar toda la mecánica de Álgebras y Grupos de Lie en la resolución de problemas de robots con articulaciones prismáticas. Son originales las **formulaciones naturales para las trayectorias del centro de masas y de lo pies del humanoide**, basadas en el algoritmo **UPA**. La bondad de las novedades presentadas en esta tesis se ha comprobado mediante **un nuevo Simulador de Realidad Virtual (RobManSim)** y experimentos con el humanoide **RH0** de la **Universidad Carlos III de Madrid**.

La línea de investigación seguida en esta tesis se aparta un tanto de los enfoques estándar para robótica, no obstante, se apreciará que cuenta ya con una dilatada historia llena de trabajos excelentes y grandes resultados. Para mayor información nos remitimos a las numerosas referencias incluidas a lo largo del texto.

1.2 Grupos y Álgebras de Lie para Robots Humanoides.

La creación de un robot humanoide está muy lejos de ser un proyecto mecánico trivial. Las ecuaciones cinemáticas y dinámicas no son lineales y cambian con la configuración del apoyo de las piernas. La posición del centro de masas no es conocida con precisión. El entorno puede ser desconocido y dinámico, con presencia de obstáculos. La superficie de apoyo puede tener diversas características: rígida, elástica, pegajosa, blanda o irregular.

Las ecuaciones del movimiento de un sistema mecánico complejo (e.g., humanoide), se pueden obtener aplicando las ecuaciones de Lagrange o alternativamente las ecuaciones de Newton-Euler. Tradicionalmente, el enfoque Lagrangiano proporciona una formulación de alto nivel muy interesante para la representación de los parámetros de importancia, mientras que la formulación de Newton-Euler se utiliza para desarrollar algoritmos recursivos de gran eficiencia computacional.

En los pasados años se han realizado importantes investigaciones en el desarrollo de soluciones para la mecánica de cadenas de sólidos rígidos con diferentes topologías, así como avances en el análisis de la mecánica de la locomoción bípeda, que pueden ser aplicados al problema mecánico del robot humanoide. Presentamos a continuación una revisión de algunos de estos trabajos, parte de los cuales son utilizados en esta tesis y son base para los nuevos desarrollos que se presentan.

1.2.1 Matemática de Lie en Robótica y en el Movimiento del Sólido Rígido.

En los últimos años se han realizado muchos trabajos de investigación en el área de la mecánica de conjuntos complejos de sólidos rígidos y sistemas robotizados. Numerosos investigadores han aplicado métodos de geometría diferencial para el estudio de la mecánica de sólidos rígidos, siendo la principal herramienta matemática la teoría de **Grupos y Álgebras de Lie**. Seguidamente hacemos un repaso a los principales trabajos en este campo.

Murray et al en [80], presentan una muy buena introducción a la teoría matemática de los grupos de Lie y especialmente al Grupo Especial Euclídeo **SE(3)** y su Álgebra **se(3)**. Además, muestran el significado geométrico de estas teorías al relacionarlo con la teoría de *screws* presentada por Ball [13]. Brockett [19] conecta la teoría de Grupos de Lie con la cinemática de robots mediante la introducción del Producto de Exponenciales (**Product Of Exponentials – POE)**. Paden y Sastry [89] usan la teoría de Lie para investigar las propiedades de los manipuladores. Park, Bobrow y Ploen [92] utilizan la teoría de Lie para una formulación de la dinámica de robots. Brockett, Stokes y Park [18] derivan las ecuaciones del movimiento de cadenas abiertas de sólidos rígidos, usando la teoría de Lie y las ecuaciones de Lagrange. Murray, Li y Sastry [80] desarrollan las ecuaciones de Lagrange para el movimiento de robots. Selig [103] deriva versiones recursivas de las formulaciones de Newton-Euler y Lagrange para resolver la dinámica de robots, basadas en la teoría de Lie. WendLandt y Sastry [125] desarrollan una versión recursiva del algoritmo de Newton-Euler para el control de sistemas en el espacio de trabajo. En [74], Martín y Bobrow desarrollan una solución de esfuerzo mínimo, para el movimiento de cadenas de sólidos rígidos. La formulación

en el espacio operacional fue introducida por Khatib [58] y es muy útil para aplicaciones centradas en el control fuerza/posición de las herramientas de los robots. Lilly [68], usando la notación de Featherstone, desarrolla algoritmos para la simulación dinámica de mecanismos robotizados. En [61], Kreutz-Delgado, Jain y Rodríguez desarrollan un algoritmo recursivo O(n) para el control en el espacio operacional, usando el álgebra de operador espacial. Fijany [32] desarrolla algoritmos paralelos para computar la dinámica directa de cadenas de sólidos rígidos. Ploen [94] desarrolla versiones recursivas para las ecuaciones del movimiento de conjuntos de sólidos rígidos, resultando en unos algoritmos independientes del sistema de coordenadas, que son una versión geométrica de los presentados por Featherstone [31].

1.2.2 Cinemática, Dinámica y Control de Robots.

La cinemática y dinámica de un solo sólido rígido no presenta problemas analíticos importantes, al ser un sistema muy sencillo. El problema aparece cuando unimos varios de estos sólidos rígidos, convirtiendo el conjunto de sistemas sencillos en un sistema mecánico que puede resultar realmente complejo. Los problemas básicos a resolver son los cinemáticos y dinámicos, tanto directos como inversos, que en la práctica necesitan algún tipo de control para ser implementados.

En la investigación de la mecánica de cadenas abiertas de sólidos rígidos, la mayor parte de los trabajos se basaron en desarrollos recursivos de las ecuaciones de Newton-Euler, aunque como no podía ser de otra manera, formulaciones basadas en las ecuaciones de Lagrange son totalmente equivalentes, como ya demostró Silver en [106]. Nos remitimos a Goldstein [38] y Meirovitch [78] para más detalles.

Vukobratovic et al [123], introdujeron soluciones unificadas para el modelado dinámico de robots manipuladores. Luh, Walker, y Paul [71] mejoraron la implementación de los algoritmos al expresar las ecuaciones en términos de los sistemas de referencia de las articulaciones. Balafoutis y Patel [10] reformularon los algoritmos recursivos mediante tensores. A destacar el gran trabajo de Featherstone [31] sobre algoritmos para la dinámica de robots, que podemos considerar pionero para muchos de los mejores desarrollos posteriores, al introducir la notación espacial. Lilly [68] usa la notación espacial de Featherstone para mejorar la eficiencia de algoritmos en la simulación de mecanismos. En [98], Rodríguez estudia la mecánica de cadenas de sólidos utilizando la notación espacial con técnicas de la teoría de filtros de Kalman. Este trabajo es ampliado posteriormente por Rodríguez, Jain y Kreutz-Delgado [99] para conseguir un álgebra del operador espacial, equivalente al trabajo de Featherstone, que permite analizar la dinámica de robots, donde el elemento clave es el cálculo de la matriz de inercias del cuerpo articulado. Jain y Rodríguez [49] demuestran que los operadores espaciales pueden ser diferenciados a alto nivel en base la formulación de Lagrange. En [32], Fijany deriva un algoritmo recursivo para el cálculo del problema dinámico directo, llamado “constraint force algorithm”, que aunque es menos eficaz que el de Rodríguez, puede ser procesado en paralelo, por lo que se puede incrementar la eficiencia computacional. Siguiendo en esta evolución de trabajos, Ploen [94] establece formulaciones tanto de Lagrange como de Newton-Euler, usando las conexiones entre la física mecánica y la geometría diferencial con las investigaciones previas acerca de la dinámica y control de robots. Hardt et al [44] introducen un tratamiento explícito de la gravedad en las formulaciones anteriores.

En cuanto al control de cadenas de sólidos rígidos, los robots necesitan un sistema que permita resolver el problema dinámico inverso, incluso ante la presencia de perturbaciones y errores de modelado. Existen dos paradigmas básicos en robótica: el *control en el Espacio de las Articulaciones* y el *control en el Espacio de Trabajo*. Slotine y Li [108] demostraron que la estabilidad de un sistema de control para robots depende críticamente de la propiedad antisimétrica de las ecuaciones de movimiento, y por lo tanto depende de la elección que hagamos para representar la matriz de Coriolis, que no es única. Murray et al [80] aplican la teoría de los Grupos de Lie para desarrollar varios tipos de control dinámico para robots, aplicando las ecuaciones de Lagrange.

Muchos sistemas robotizados están formados por un conjunto de cadenas abiertas de sólidos rígidos acopladas mediante un cuerpo común, que normalmente es una base considerada como sistema de referencia inercial. Estos son los que se conocen como *sistemas con topología de árbol*. Featherstone [31] extendió su algoritmo "Articulated Body Inertia" para cadenas abiertas, a los sistemas con topología en árbol, definiendo la conectividad del sistema mediante unos índices que especifican para cada eslabón cual es su precedente. En [99], Rodríguez, Jain y Kreutz-Delgado amplían el álgebra del operador espacial para obtener las ecuaciones de movimiento de los sistemas con topología de árbol. Vukobratovic et al [123] utilizan la teoría de *screws* para obtener las ecuaciones de movimiento, modelando la conectividad del sistema con topología de árbol mediante teoría de grafos. En [80], Murray, Li y Sastry analizan la manipulación en robótica, que no es sino una cooperación de varios sistemas en cadena abierta conectados, con un planteamiento matemático basado en la teoría de Grupos de Lie. En [94], también Ploen extiende sus formulaciones tanto de Lagrange como de Newton-Euler para los sistemas topológicos ramificados. Hardt [44] introduce el tratamiento de restricciones provocadas por el contacto de las cadenas cinemáticas con el entorno. García de Jalón y Bayo [35] usan las coordenadas cartesianas de las articulaciones de los robots, para obtener las ecuaciones del movimiento automáticamente. También tenemos que destacar los trabajos en el **RoboticsLab** de la **Universidad Carlos III de Madrid**, donde C. Balaguer [5][21][90][91] et al han realizado trabajos muy interesantes para el análisis mecánico de robots, destacando las aportaciones de S. de la Torre [117] para la resolución de la dinámica inversa de robots humanoides.

En esta tesis se usan los fundamentos matemáticos desarrollados en los trabajos anteriores para extenderlos al análisis de robots humanoides, aunque éstos tienen particularidades mecánicas que los alejan del concepto más tradicional de sistema mecánico con topología de árbol, principalmente porque la base de las cadenas abiertas de sólidos rígidos no es inercial, esto es, la base (i.e., el tronco del humanoide) es libre, lo que introduce restricciones de equilibrio importantes.

Para el análisis de Robots Humanoides, tenemos que tener en cuenta otros conceptos como son: los estudios de *Acomodación*, los tipos de *Controladores*, la necesaria *Coordinación* para un funcionamiento armónico, los problemas de *Deslizamiento*, la necesidad de un *Modelado* mecánico eficaz para la Locomoción Bípeda y los problemas de *Planificación de Movimientos* y *Navegación* libre de colisiones en entornos de trabajo reales. Como veremos a lo largo de esta tesis, estos conceptos se abordarán desde la perspectiva de Geometría Diferencial que se introducirá en el siguiente capítulo (2) mediante el uso de los Grupos y Álgebras de Lie, extendiendo los conceptos y desarrollando nuevos modelos y algoritmos.

1.3 Generación de Movimientos y Locomoción para Robots Humanoides.

La investigación sobre robots humanoides bípedos es actualmente uno de las materias más excitantes de la robótica y hay muchos proyectos desarrollándose en este campo [55][60][93]. Como la resolución del problema dinámico inverso para la locomoción bípeda del robot humanoide resulta ser muy complejo, se han propuesto en la literatura diferentes modelos para simplificar los cálculos de estabilidad necesarios para que un humanoide camine: Vukobratovic [122] fue el primero en proponer el concepto de **ZMP** (Zero Moment Point), Yoneda [128] propuso el llamado Criterio de Estabilidad de Vuelco, Goswami [41] propuso el Indicador de Rotación del Pie.

La pérdida de la estabilidad podría resultar potencialmente desastrosa para el robot humanoide, por lo que resulta necesario hacer un seguimiento y control de la estabilidad del robot en cada instante, y especialmente cuando éste se encuentra sometido a perturbaciones externas. La tarea más importante para la locomoción bípeda de un robot humanoide es preservar el equilibrio en relación a la superficie de apoyo. Los pies realizan su función de soporte mediante la fuerzas de fricción y normal de reacción, pero no pueden ser controlados de forma directa, por lo que el mejor indicador del comportamiento del mecanismo en su conjunto, es el punto donde la influencia de todas las fuerzas que actúan sobre el mecanismo pueden ser reemplazadas por una única fuerza, este punto es el **ZMP**. De este modo, podremos hacer una planificación del paso del humanoide que garantiza la estabilidad, si conseguimos que el **ZMP** se encuentre en todo momento dentro del área de soporte.

El proceso de Locomoción Bípeda es un fenómeno periódico (o casi). Un ciclo completo de locomoción se compone de dos etapas: una etapa de doble soporte, cuando ambos pies están en contacto con el suelo, y una etapa de soporte simple, cuando sólo un pie se encuentra de forma estacionaria sobre el suelo (o superficie de soporte), mientras que el otro pie se encuentra en vuelo. Este proceso de locomoción puede ser generado con estabilidad usando el principio de equilibrio con el **ZMP**.

En resumen, como veremos seguidamente, la investigación debe resolver dos problemas bien diferenciados, estos son, la Generación del Movimiento de Locomoción Bípeda o Síntesis del Paso y su Control.

1.3.1 Locomoción Bípeda – Síntesis del Paso.

La mecánica de la locomoción bípeda es un problema complejo, que salvo simplificaciones importantes, nos lleva a sistemas algebraicos diferenciales, variables en el tiempo. Numerosos trabajos han tratado de simplificar el problema para hacerlo manejable y durante la última década, la investigación sobre la locomoción bípeda de robots ha recibido un gran impulso, que es creciente en nuestros días, y que se va a proyectar de forma creciente hacia el futuro.

Todd [116] da una introducción interesante a la historia de las máquinas que caminan. Song y Waldron [109] hacen una buena revisión de la locomoción con equilibrio estático. En [33], Furusho y Sano revisan las investigaciones realizadas sobre robots con dos piernas. Kato et al [55], fueron los primeros en crear un robot bípedo con un

control cuasi-dinámico de la locomoción, esto es, durante pequeños períodos de tiempo la proyección del **CM** era desplazada fuera de la zona de apoyo para permitir al robot ganar aceleración horizontal. Miura y Shimoyama [81] abandonan el control estático y usan un control dinámico de péndulo invertido para el posicionamiento de los pies. Otro enfoque fue presentado por Raibert [96], quien desarrolló con gran éxito un robot saltarín, que a pesar de no ser bípedo, inició las investigaciones en el área de vuelo balístico para la locomoción con extremidades. Kajita et al [52], trataron de reducir la complejidad de las ecuaciones dinámicas al restringir el movimiento del **CM** a un plano horizontal. Takanishi et al [115] y Yamaguchi et al [127], implementan sistemas de control sobre el punto de momento nulo **ZMP**, para alcanzar la estabilidad dinámica. Wyeth et al [126] proponen algo tan curioso como adaptar para humanoides un bien conocido paso de robots de tipo insecto. En [77], McGeer demostró que un sistema mecánico puede caminar por una pendiente descendente sin ninguna motorización en función de su dinámica y la gravedad, lo que inició mejoras en el diseño mecánico de los robots. Kun y Miller [63] usan redes neuronales para generar el paso de locomoción bípeda. Nicholls [85] estudia la locomoción bípeda desde un punto de vista intuitivo, sobre la base del análisis gráfico de la locomoción humana. Roussel [102] y Chevallereau [23] investigan el movimiento de locomoción bípeda con energía mínima. Chestnutt et al [22] desarrollan el paso de locomoción bípeda como un problema de navegación. Nakanishi et al [83] proponen una generación de la locomoción bípeda a través de un proceso de sucesivas demostraciones y adaptaciones. Este mismo año 2005, Nakaoka et al [84] analizan y experimentan con detalle el paso de los humanoides, imitando el de un ser humano bailando.

Proyectos muy importantes en el desarrollo de la investigación de la locomoción bípeda, han sido los desarrollados en la universidad de Waseda en Japón, con el desarrollo de robots antropomórficos como el humanoide WABIAN. Así mismo, hay que destacar los proyectos de Honda para la construcción de robots bípedos que se asemejan a la estructura humana, que nos introducen Hirai et al [46], y que ha supuesto todo un hito en este campo de investigación. Los robots de Honda pueden caminar, empujar objetos e incluso subir escaleras, utilizando un sistema de control que es una combinación de trayectorias predefinidas con un control dinámico para una posición de equilibrio del **ZMP**. También destacables son los proyectos de Sony para desarrollar mascotas robotizadas y varios robots humanoides realmente avanzados que parecen estar cerca de lograr un éxito comercial importante. Recientemente Sugihara y Nakamura [114] presentaron un magnífico trabajo sobre la planificación del paso on-line para robots humanoides.

1.3.2 Locomoción Bípeda – Control del Paso.

La locomoción bípeda necesita un sistema de control para garantizar su estabilidad. Para el control de un robot humanoide, se deben desarrollar reguladores para las cadenas sólidos rígidos que lo forman (como ya hemos visto en el apartado 1.2.2). Se deben analizar muchos otros aspectos: objetivos, elasticidad, coordinación, navegación, controladores, deslizamientos y otros. Otro problema importante es el de modelado del entorno (especialmente la superficie de apoyo), de los actuadores y de los sensores.

En [97], Ridderstrom consolida de forma práctica diversos trabajos en relación con el control de la locomoción con extremidades, para varios tipos de robots. Vukobratovic y Stepanenko [122] analizan la estabilidad de los sistemas antropomórficos. En [96],

Raibert analiza el control estático y dinámico de la locomoción bípeda. Takanishi et al [115] estudian la importancia que tiene para la estabilidad del humanoide el control de la posición del tronco. Fujimoto y Kawamura [34] estudian el control de humanoides con interacción de fuerzas externas. Liu e Iba [69] proponen una interesante arquitectura de control por capas para robots humanoides.

Los métodos estándar para la implementación técnica del control de humanoides son los basados en las tradicionales técnicas de regulación (e.g., control realimentados de posición, control de fuerza o control de impedancia), aunque hay variaciones más sofisticadas. Tzafestas et al [119] comparan los controles de modo deslizamiento con el de par computado, para desarrollar un control robusto de locomoción bípeda. En [88], Osuka obtiene algunos resultados muy interesantes para el control de estabilidad, basados en el teorema de Lyapunov. Sorao et al [111], presentan resultados de un control bípedo mediante la generación y seguimiento de una trayectoria para el **ZMP**. Genot y Espiau [36] discuten un método de control que tiene en cuenta los posibles deslizamientos de los pies.

Algunos investigaciones utilizan algoritmos para distribuciones de fuerzas rígidas, pero otros consideran la elasticidad del sistema, lo que es computacionalmente muy costoso, y además nos abre el problema de la elección de las funciones de optimización. Por ejemplo, en [72], Marhefka y Orin introducen una optimización para un método de control de distribución de fuerzas.

En aplicaciones para humanoides se hace muchas veces necesario un Control de Alto Nivel que dirija las estrategia, que será alcanzada posteriormente con las acciones tácticas de algunos de los controles mencionados anteriormente. El control de alto nivel clasifica situaciones y selecciona restricciones, por ejemplo puede detectar un patinaje y lanzar un evento de comportamiento reflejo, o puede generar diversos patrones de movimiento, como muestran Oka et al [86]. Por su propia naturaleza, el control de alto nivel está muy relacionado con la planificación de trayectorias, que introduciremos en el apartado siguiente (1.3) En esta línea de trabajo, Yin et al [130] presentan una estrategia de control para el mantenimiento de la estabilidad, mediante la optimización de varios comportamientos del robot (e.g. alargar el paso, balancear el cuerpo), en función de la posición del **FZMP** (Fictitious Zero Moment Point).

El mantenimiento de la estabilidad también es un asunto muy importante en el proceso de control del paso. Se pueden desarrollar soluciones de optimización de movimientos básicos del robot (i.e.: inclinación del tronco del robot, aceleración del movimiento, alargamiento de la zancada), que ayuden a compensar el efecto de perturbaciones. Los diferentes métodos de optimización de la estabilidad, necesitan disponer de unas medidas representativas de la situación dinámica del conjunto mecánico, que normalmente son el **ZMP** o el **FZMP**.

Es difícil encontrar trabajos puramente analíticos, puesto que los trabajos de simulación se hacen casi imprescindibles. Kanehiro et al [53], así como Kuffner et al [62], han desarrollado herramientas y plataformas técnicas de propósito general muy importantes para la simulación de robots humanoides. En esta tesis, seguimos un enfoque parecido, para lo que hemos creado un Simulador (**RobManSim**) con el lenguaje de programación **VRML** (ver libro de Ames et al [4]), que nos permite desarrollar y comprobar las soluciones algorítmicas antes de pasar a los ensayos reales con el robot humanoide **RH0**.

1.4 Planificación de Movimientos y Navegación para Robots Humanoides.

Esta investigación está motivada principalmente por la necesidad de desarrollar aplicaciones para robots humanoides en entornos desconocidos. Es esencial un cierto grado de autonomía para realizar un trabajo con seguridad donde no tenemos disponibles modelos exactos del entorno (e.g., un humanoide que se mueva por habitaciones llenas de mobiliario y objetos de la vida diaria.) La planificación de movimientos y trayectorias es la disciplina que se encarga de tratar estos problemas.

Vamos a distinguir dos tipos de planificación: la primera es la planificación de movimientos en el espacio de configuraciones del humanoide, que se ocupa de la generación de la locomoción bípeda y del movimiento de las articulaciones, y la segunda es la planificación de trayectorias en el espacio físico del robot, que se ocupa básicamente de la generación de caminos libres de colisiones, esto es, de la navegación.

La planificación de movimientos para generar referencias adecuadas a la locomoción bípeda (que pueden ser periódicas, óptimas o simplemente factibles), ha sido estudiada por numerosos investigadores, tales como Goswami et al [40], o Rostami et al [101]. Kawaji et al [57] utilizan algoritmos genéticos para diseñar movimientos rítmicos de referencia para la cadera y la pierna en etapa de transferencia. En [113], Stitt y Zhen usan redes neuronales entrenadas con medidas de estabilidad procedentes del modelo dinámico del robot. Shih [105] optimiza los movimientos cambiando la posición del tronco, de forma que el **ZMP** quede en el centro del área de soporte. Goodwine y Burdick [39] proponen una planificación de movimientos para las articulaciones sin realizar una planificación explícita de las posiciones de apoyo de los pies.

Podemos dividir la planificación de movimientos en dos clases: problemas holonómicos, en los que los grados de libertad son independientes y problemas no holonómicos en los que existe una relación entre al menos alguno de ellos, que derivan en un incremento de la complejidad.

Se define la Navegación como la metodología que permite guiar el curso de un robot de una forma segura. Existen varios enfoques para la solución del problema de navegación, como veremos en 1.4.1, pero los podemos agrupar en dos tipos básicos:

- Navegación ACTIVA, que se caracteriza por las siguientes tareas:
    - Modelado del entorno conocido a priori.
    - Planificación global de trayectorias, normalmente off-line.
    - Control y seguimiento del camino dado por la trayectoria global.
- Navegación REACTIVA, que se caracteriza por las siguientes tareas:
    - Modelado del entorno que es desconocido a priori, por integración sensorial en tiempo real.
    - Planificación de trayectorias locales, normalmente on-line.
    - Control y seguimiento del camino local en el entorno del robot.

La complejidad del problema de navegación crece exponencialmente con el número de grados de libertad del robot y el número de obstáculos. Se han buscado varias aproximaciones completas a la solución del problema, que normalmente consumen mucho tiempo de computación. También se han propuesto soluciones heurísticas, que no garantizan un orden de convergencia máximo. Otro desafío para la navegación es el tipo de obstáculos, que son convexos o de geometría específica (polígonos, elipsoides), en casi todos los trabajos publicados, lo que claramente contrasta con el mundo real.

1.4.1 Diversos Enfoques para la Navegación.

La planificación de trayectorias se ocupa de generar caminos libres de colisiones (evitando obstáculos), tanto para el cuerpo del robot (es lo que llamaremos Navegación), como para las extremidades. En este campo se han realizado numeras investigaciones basadas en diferentes disciplinas (e.g., geometría, control, análisis matemático), como presentan Hwang y Ahuja [48].

Una buena introducción a diferentes métodos de planificación es la realizada por Latombe [64]. El inspirador trabajo de Lozano-Perez [70], introdujo el concepto de **C-Space**, en el que el robot es tratado como un punto en su espacio de configuraciones multidimensional. Cuando el número de grados de libertad es muy grande, el tratamiento explícito se convierte en un problema computacional complejísimo, por lo que aparecieron otros enfoques con reducción del **C-Space**, como el de Balaguer et al [12]. Muñoz [79], introduce una planificación basada en curvas spline para la obtención de un camino dinámicamente factible. Sethian [104] implementa una solución moderna de diagramas de Voronoi. En [20], Brooks introduce el algoritmo del modelado del espacio libre. Khatib [59] utiliza algoritmos de campos de potencial, que tienen gran éxito, aunque presentan el problema de la aparición de mínimos locales. En el trabajo de Hsu et al [47], podemos ver una solución con mapas de caminos probabilísticos. Vallejo et al [120], también desarrollan con éxito algoritmos probabilísticos aleatorios, para problemas con muchos **GDL**. Yu y Gupta [131], introducen el concepto de entropía del **C-Space**, para el desarrollo de planificación de trayectorias locales probabilísticas. En [6], Arkin presenta diversos métodos basados en comportamiento, extendiendo los trabajos de Brooks [20] sobre algoritmos de inteligencia artificial de amplia difusión. En [120], Versino y Gambardella diseñan experimentos con redes neuronales. Walsh et al [124] utilizan una optimización sobre Grupos de Lie para la planificación de caminos. Otra técnica que ha obtenido muy buenos resultados es la **SLAM** (Simultaneous Localisation and Mapping) y sus diferentes variantes, ver Davison et al [28] [29], que realizan un modelado on-line de la trayectoria, basando la navegación en un movimiento reactivo del robot humanoide. Kanehiro et al [54] desarrollan este mismo año 2005, modelos para planificación del movimiento completo del cuerpo de un humanoide, teniendo en cuenta un mapa tridimensional del espacio.

El profesor Sethian [104] desarrolló los algoritmos **FMM** (Fast Marching Methods) para analizar desde una perspectiva Euleriana la evolución de diversos frentes. Esta teoría se aplica a un gran número de áreas de la física y entre ellos, la podemos aplicar a la planificación de trayectorias en espacios con varios grados de libertad. Estas técnicas de geometría computacional tienen ventajas, que nos ayudarán a desarrollar algoritmos geométricos que soslayan muchos de los inconvenientes de otras técnicas de navegación, constituyendo uno de los trabajos importantes de esta tesis que desarrollaremos con detalle en el capítulo 4.

1.5 Objetivos y alcance de la investigación.

En los últimos años hemos visto un resurgir de la robótica con humanoides, tanto en términos científicos como sociales, debido al impacto causado por algunos desarrollos. Sin embargo, la investigación con humanoides continúa siendo un gran desafío técnico, debido a que son sistemas complejos con muchos grados de libertad y restricciones.

Las ecuaciones del movimiento de un robot humanoide son mucho más difíciles de resolver que las de un robot manipulador, por lo que la utilización de herramientas matemáticas estándar (e.g., Parámetros de **Denavit Hartenberg D-H**), conduce a enfoques muy particulares, poco eficientes y demasiado confusos. Para obtener soluciones de propósito general que sean aplicables en tiempo real, necesitamos formulaciones simples de las ecuaciones del movimiento, que se puedan manipular a alto nivel y que tengan parámetros expresados de forma transparente.

El objetivo fundamental de esta tesis pionera en el campo de la investigación robótica, es la búsqueda de una Solución Completa para resolver en tiempo real los problemas de Locomoción y Navegación Bípedas de Robots Humanoides, mediante la introducción de nuevos algoritmos de Geometría Diferencial, que son formulaciones elegantes y eficaces que no han sido presentadas anteriormente en la literatura. Este objetivo fundamental se construye con la integración (ver la Figura 1-2) de los desarrollos realizados para los siguientes objetivos parciales:

- ***Diseñar modelos mecánicos para humanoides que simplifiquen el análisis cinemático y dinámico***: Veremos que aquí presentaremos **el nuevo modelo "División Cinemática Sagital" (DCS)**, que posibilita las soluciones cerradas al problema cinemático inverso completo del robot y facilita el dinámico inverso. En términos prácticos, el modelo **DCS** nos permitirá resolver problemas como el planteado por la pregunta *¿Cómo mover las piernas del robot humanoide?*

- ***Diseñar y desarrollar algoritmos que tengan soluciones cinemáticas al problema de Locomoción Bípeda de robots humanoides***: Se introducirá **el nuevo algoritmo "Un Paso Adelante" (UPA)**, que resuelve el problema de forma geométrica genérica y es aplicable para muchos tipos de humanoides. En la práctica, el algoritmo **UPA** solucionará el problema generado por la pregunta *¿Cómo hacer que el robot humanoide ejecute un único paso hacia un objetivo?*

- ***Diseñar y desarrollar algoritmos con solución analítica para la planificación de movimientos y Navegación***: Se mostrará que **el nuevo algoritmo "Método Modificado de Marcha Rápida" (M3R)**, construye de forma geométrica caminos en un entorno con obstáculos de cualquier tipo. El nuevo algoritmo **M3R** nos permitirá responder de una manera eficaz a cuestiones prácticas como *¿Cuál es la trayectoria libre de colisiones para la navegación dentro de un espacio?*

- ***Diseñar modelos eficaces de Navegación Bípeda Global y Local para humanoides***: Veremos que se presentará **el nuevo modelo "Trayectoria Corporal Global" (TCG)** con soluciones geométricas para ambos. En términos prácticos, el modelo **TCG** resolverá preguntas tales como *¿Cuál es la trayectoria corporal del robot humanoide en pasos consecutivos?*

- *Demostrar la bondad de todos los nuevos modelos y algoritmos de la tesis*: Para ello nos planteamos como objetivos la creación de un **nuevo Simulador para Robots Humanoides con Realidad Virtual (RobManSim)** y la experimentación con el robot **RH0** de la **Universidad Carlos III de Madrid**.

Debemos señalar que quedan fuera del alcance de esta tesis, la planificación de movimientos en todo el espacio de configuraciones y la formulación completa de las soluciones dinámicas para el control de humanoides, que quedan para trabajos futuros.

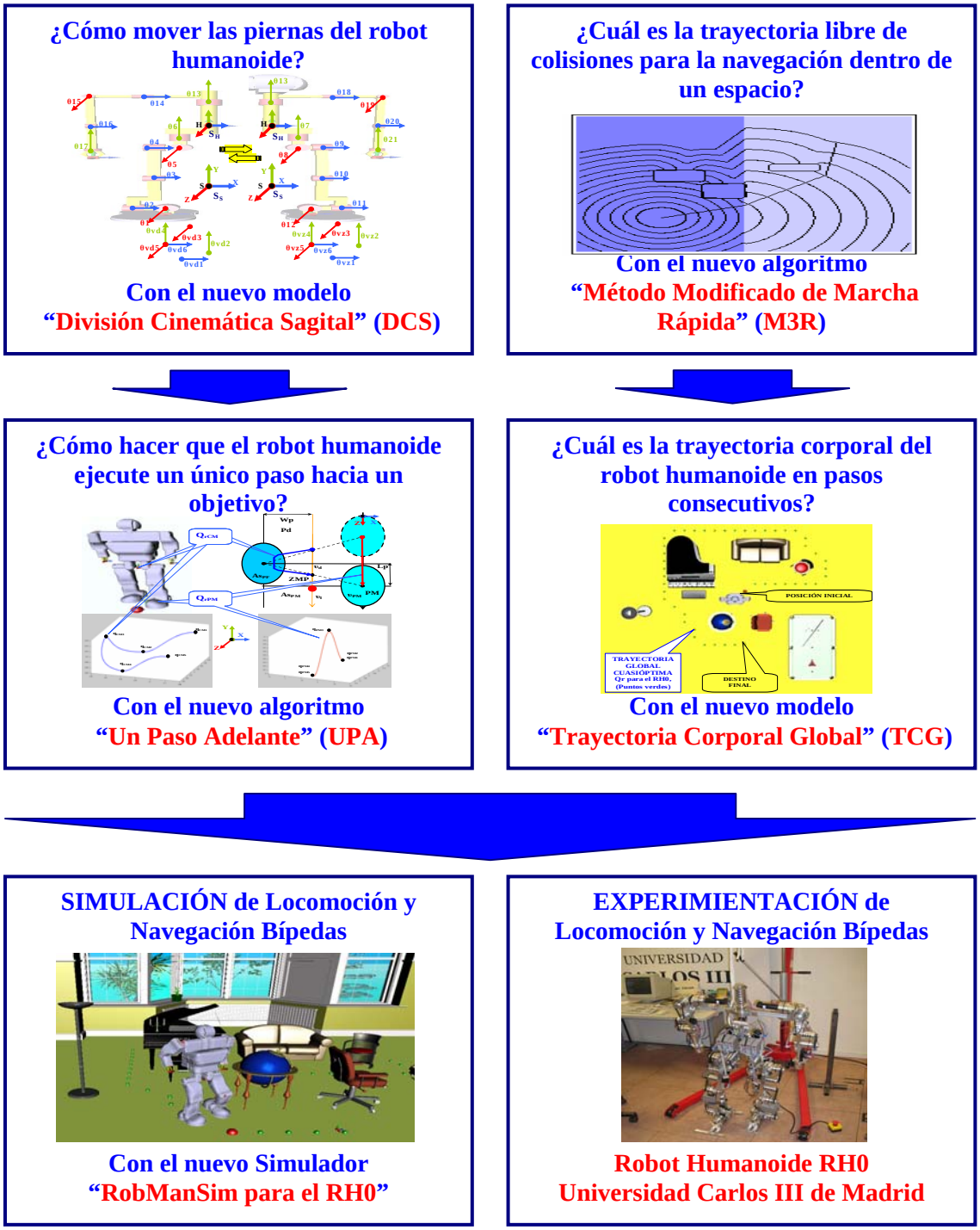


Figura 1-2: Objetivos de la Tesis – Descripción e Integración.

1.6 Contenido de la tesis.

En esta tesis se presenta una formulación geométrica unificada, potente, flexible y eficaz para las ecuaciones del movimiento, obteniendo una ***solución completa*** para **los problemas de Locomoción y Navegación Bípedas de robots humanoides**. Las soluciones y algoritmos desarrollados pueden ser empleados en aplicaciones de tiempo real y tienen un carácter genérico, por lo que se pueden usar para varios tipos de humanoides. La implementación de los trabajos se ha realizado con un nuevo **Simulador de Realidad Virtual** y en experimentos reales con el robot humanoide **RH0** de la **Universidad Carlos III de Madrid**.

Estos trabajos basados en técnicas matemáticas de **Geometría Diferencial**, que no han sido presentado anteriormente en la literatura, se fundamenta en las piedras angulares de la teoría matemática de **Grupos y Álgebras de Lie**, que nos proporciona el grupo especial Euclídeo **SE(3)** para la representación del movimiento de sólidos rígidos, y la teoría matemática de las **Leyes de Conservación Hiperbólica**, que nos proporciona los algoritmos **FMM** para la planificación de trayectorias. Para la mecánica de Locomoción Bípeda y su control, introducimos formulaciones de geometría diferencial. Para la Navegación Bípeda, presentamos herramientas de geometría computacional para el análisis de interfaces. Ambos enfoques nos servirán para crear algoritmos geométricos numéricamente estables, con soluciones cerradas y completas.

Estos nuevos desarrollos se presentan en esta tesis, estructurados según el siguiente contenido para los capítulos y apéndices:

- ***El capítulo 1*** *es el de* ***Introducción***. En él se establecen las motivaciones para el desarrollo de esta tesis. Se realiza como introducción una revisión al estado del arte de los antecedentes matemáticos para: Los Grupos y Álgebras de Lie en robótica, la Generación de Movimientos y Locomoción de humanoides y la Planificación de Movimientos y Navegación de robots. También se explicitan el objetivo fundamental, los objetivos parciales y el alcance de la investigación llevada a cabo. Finalmente, se resumen los contenidos de la tesis.

- ***El capítulo 2*** *está dedicado a los* ***Grupos y Álgebras de Lie para Robots Humanoides***. Comenzamos por una presentación de la matemática de Lie en Robótica, con la introducción del grupo **SE(3)**, su álgebra **se(3)**, la geometría de movimientos y fuerzas generalizadas, la fórmula del **POE** y los manipuladores Jacobianos geométricos. Aplicaremos esos conceptos a la descripción mecánica del sólido rígido. En base a lo anterior, analizamos los problemas cinemáticos y dinámicos de la robótica, e introducimos para la resolución de las cinemáticas inversas, los problemas canónicos de **Paden-Kahan y** el problema canónico original de **Pardos-uno** para robots con articulaciones prismáticas. Se aplican los mismos fundamentos para el análisis de robots humanoides y su control. Por último, se realiza una comparación entre el **POE** del **Álgebra de Lie** vs los **Parámetros de D-H**, con un ejemplo de la aplicación de ambas formulaciones a un robot Polar de 3 **GDL**. Para una mejor comprensión de las fórmulas matemáticas que se presentan en este capítulo, se indican referencias a la nueva librería de software abierto creada para esta tesis "RobotMan" (ver D).

- ***El capítulo 3*** *trata la* ***Generación de Movimientos y Locomoción para Robots Humanoides***. Se desarrolla la mecánica de la locomoción bípeda. Se presenta **el nuevo modelo mecánico "División Cinemática Sagital" (DCS)**, que permite obtener soluciones cerradas del problema cinemático inverso completo del robot humanoide. Se formaliza **el nuevo algoritmo "Un Paso Adelante" (UPA)** que resuelve el problema de la Locomoción Bípeda del humanoide de forma genérica. Se aplica el nuevo modelo mecánico **DCS** para el **RH0**, tanto en su versión cinemática como dinámica, presentando una solución geométrica cerrada para la cinemática inversa completa del humanoide, planteando bajo la misma perspectiva la solución del problema dinámico inverso. Se formaliza y aplica el nuevo algoritmo **UPA** para el **RHO**. Se presentan esquemas geométricos de los pasos básicos del **RHO** (i.e.: salida, zancada, entrada, giro), **introduciendo nuevas trayectorias naturales tanto para el centro de masas como para los pies del RHO**. Estos esquemas geométricos constituyen un grupo canónico de soluciones para un paso del robot **RH0**, a partir de los que se puede construir por conjugación de los mismos, movimientos realmente complejos para el humanoide.

- ***El capítulo 4*** *trata la* ***Planificación de Movimientos y Navegación para Robots Humanoides***. Se comienza formalizando la planificación de trayectorias y la Navegación en robótica y revisando diferentes métodos de planificación, con sus ventajas e inconvenientes. La parte más importante del capítulo se dedica a desarrollar **el nuevo algoritmo "Método Modificado de Marcha Rápida" (M3R)**, para la planificación de trayectorias en entornos reales tridimensionales con cualquier tipo de obstáculos. Este desarrollo se basa en las leyes de conservación hiperbólica y presenta muchas ventajas en comparación con otros métodos, como se ejemplifica en la comparación con los métodos de Campo Potencial. Finalmente, se aplican esos resultados a la Navegación Bípeda de humanoides, con la introducción del **nuevo modelo "Trayectoria Corporal Global" (TCG)** que permite obtener soluciones geométricas para ese problema.

- ***El capítulo 5*** *está dedicado al* ***SIMULADOR para Robots Humanoides con Realidad Virtual (RobManSim).*** En este capítulo se presenta un Simulador creado *ad hoc* para esta tesis, que es una Plataforma de Realidad Virtual que reproduce el comportamiento mecánico del **RH0** en un entorno de trabajo que replica lo más exactamente posible al del robot real. Esta plataforma de simulación nos permita ganar productividad y reducir los riesgos inherentes a la experimentación real con el humanoide. Se detallan la creación de modelos de Realidad Virtual, el Entorno de Simulación Integrado y el Interfaz Gráfico de Usuario. El Simulador nos permite estudiar y desarrollar nuevas ideas, mostrando los diferentes comportamientos del **RH0** bajo la aplicación de todos los nuevos algoritmos y desarrollos introducidos en esta tesis. Se analizan con detalle **simulaciones de Locomoción Bípeda** del Humanoide **RH0** caminando en línea recta y girando sobre su propio eje. Así mismo, se analiza el resultado de **simulaciones de la Navegación** del Humanoide **RH0**, tanto global como local. Para finalizar, se simula una tarea compleja para el Humanoide **RH0** que incluye de forma simultánea problemas tanto de Locomoción como de Navegación Bípedas, que necesitan de la ejecución de todos los algoritmos introducidos como novedades en esta tesis.

- ***El capítulo 6** presenta la **EXPERIMENTACIÓN con el Robot Humanoide RH0.*** Se hace una presentación de la plataforma de experimentación, constituida por el robot **RH0** y su arquitectura de control. Se analizan varias experimentaciones reales con el humanoide **RH0** de la **Universidad Carlos III de Madrid**, que tienen el objetivo de demostrar que todos los nuevos algoritmos introducidos funcionan de forma satisfactoria. Se analizan con detalle un **experimento de Locomoción Bípeda** del Humanoide **RH0** caminando en línea recta, donde se comparan los valores de diferentes magnitudes dados por la referencia obtenida del Simulador de Realidad Virtual, con los valores reales de las articulaciones del robot. Se presenta otro experimento con el robot girando sobre su propio eje. De igual modo se analiza el resultado de un experimento de navegación del humanoide. Amén de los experimentos detallados en este capítulo, en el soporte informático de la tesis se adjuntan otros muchos.

- ***El capítulo 7** es el de **Conclusiones***. Contiene un sumario la tesis, más una revisión de posibles futuros trabajos de ampliación en línea con los objetivos ya alcanzados, aprovechando los avances en el montaje del humanoide **RH0**, que no estaba completado en el tiempo de desarrollo de esta tesis.

- ***A - Apéndice: Resolución de Mecánicas usando Álgebras de Lie*.** Introducción a las técnicas matemáticas de **Lie** utilizadas en la tesis, especialmente para los no iniciados. Se desarrollan varios ejemplos completos, como son la resolución de la cinemática inversa para robots de tipo **STANFORD** y **PUMA de 6 GDL**.

- ***B - Apéndice: Descripción Mecánica del Humanoide RH0*.** Presenta los detalles de diseño, construcción y montaje mecánicos del humanoide **RH0**, que son necesarios para entender las implementaciones finales y los detalles de programación de los nuevos algoritmos presentados en esta tesis. Se incluyen apartados para las medidas de la estructura y los pesos e inercias del robot.

- ***C - Apéndice: Glosario*.** Es un glosario de términos relativos al mundo de la robótica de humanoides, centrado en los temas de locomoción y navegación.

- ***D - Apéndice: Librería de Software ROBOTMAN*.** Contiene un fruto interesante de los trabajos de esta tesis, como es la librería de software **RobotMan** (como una *Toolbox* de **MATLAB**, cuyo nombre es abreviatura de *Robot Manipulator*). El apéndice incluye el código documentado (resulta bastante educativo) de todas las funciones de esta librería para el análisis mecánico de robots mediante la teoría de **Grupos de Lie**. Con las funciones de esta librería se han construido todos los nuevos algoritmos de esta tesis y los controles del Simulador de Realidad Virtual. La librería puede constituirse en base fundamental para una ampliación de la misma o implementación de nuevas aplicaciones.

- ***Bibliografía***. Contiene numerosas referencias importantes, interesantes y aún imprescindibles para los interesados en las líneas de investigación seguidas en esta tesis. A destacar que los nuevos desarrollos que se presentan en la tesis han sido inspirados principalmente por la gran introducción matemática a la robótica de Murray [80], que extenderemos a robots humanoides y también por los trabajos de Sethian [104] sobre geometría computacional, que aplicaremos a la planificación de trayectorias libres de colisiones.

2 Grupos y Álgebras de Lie para Robots Humanoides.

La mecánica y control de robots humanoides necesita de formulaciones elegantes para las ecuaciones del movimiento, que puedan ser derivadas de forma fácil con respecto a los parámetros cinemáticos y dinámicos de importancia y que sirvan para representaciones de alto nivel. Es por ello, que esta tesis presenta y utiliza las técnicas de geometría diferencial de la teoría matemática de Grupos y Álgebras de Lie. Con la introducción del grupo ***SE(3)****, su álgebra* ***se(3)*** *y la fórmula del* ***POE****, podemos representar de forma adecuada la mecánica del sólido rígido, con extensión a la descripción cinemática y dinámica de robots. Introducimos para la resolución de las cinemáticas inversas los problemas canónicos de* ***Paden-Kahan*** *y el problema canónico original de* ***Pardos-uno*** *para robots con articulaciones prismáticas. Se realiza una comparación entre* ***el POE del Álgebra de Lie*** *vs* ***los Parámetros de D-H*** *con un ejemplo de la aplicación de ambas formulaciones a un robot Polar de 3 GDL. Para una mejor comprensión de las fórmulas matemáticas que se presentan en este capítulo, se indican referencias a la nueva librería de software abierto creada para esta tesis "RobotMan" (ver Apéndice-D).*

2.1 La Matemática de Lie en Robótica.

Los Grupos de Lie son muy importantes para el análisis matemático, para la física y para la geometría, porque sirven para describir la simetría de las estructuras analíticas. Fueron desarrollados por el matemático noruego **Marius Sophus Lie** en 1870 para estudiar las simetrías de ecuaciones diferenciales. En este apartado se introduce muy brevemente la teoría de grupos de Lie, para luego centrarnos en las aplicaciones geométricas de los mismos en el análisis mecánico del movimiento de sólidos rígidos, con especial énfasis en su utilización para la mecánica de robots.

Definiciones importantes:

- **Una Variedad (Manifold)**: Es un espacio topológico que localmente se parece a un espacio Euclídeo ordinario y además es un espacio de Hausdorff (i.e., sii cualquier pareja de puntos, pueden ser siempre separados mediante una vecindad.) Un ejemplo es la superficie de una esfera, como el planeta Tierra, que no es un plano, pero pequeñas porciones de esa superficie son homeomórficas (i.e., topológicamente equivalentes) a porciones de un plano Euclídeo. Para que una Variedad sea diferenciable, los mapas locales deben ser compatibles en cierto sentido; para poder hablar de direcciones, espacios tangentes y funciones diferenciales. Las Variedades diferenciales se usan en matemáticas para describir objetos geométricos y son la herramienta más general y natural para el estudio de la diferenciabilidad.

- **Un Grupo de Lie**: Es una variedad analítica que es también un grupo (i.e., sii tiene un operador para el que cumple las propiedades: asociativa, identidad, inversa y grupo cerrado), y además ambas estructuras son infinitamente diferenciables. Un ejemplo de grupo real de Lie es el espacio Euclídeo $\mathbf{R}^n$, con la suma ordinaria de vectores como operación de grupo. Grupos de Lie son los grupos de matrices invertibles con la operación producto de matrices, por ejemplo el grupo especial ortogonal **SO(3)** de todas las rotaciones en el espacio tridimensional.

- **Un Álgebra de Lie**: Es un espacio vectorial **Ev** sobre un campo (típicamente el de los números reales o complejos), junto con una operación binaria denominada *Lie bracket* (i.e., **[·, ·] : Ev × Ev -> Ev**), que satisface las siguientes propiedades: bilinealidad, identidad de Jacobi y antisimetría. Un ejemplo de álgebra de Lie es el espacio Euclídeo $\mathbf{R}^3$ con la operación Lie bracket dada por el producto vectorial. A cada Grupo de Lie, se le puede asociar un Álgebra de Lie que capture completamente la estructura local del grupo.

Por ser de especial interés en el desarrollo de la mecánica del sólido rígido, tenemos que introducir la existencia de una Transformación Exponencial, que es la principal conexión entre el álgebra de Lie y su correspondiente grupo de Lie, como una generalización de la función exponencial para números reales (i.e., **R** es el álgebra de Lie del Grupo de Lie dado por los números reales positivos con la operación de multiplicación.) En los grupos de matrices, la transformación exponencial corresponde a la exponencial ordinaria de matrices, (i.e., sí la matriz A es un elemento de un álgebra de Lie, entonces expA es un elemento del grupo de Lie correspondiente.)

2.1.1 Grupo de Lie Especial Euclídeo SE(3) y Álgebra se(3).

El Grupo Especial Euclídeo **SE(3)**, es el Grupo de Lie de dimensión seis, correspondiente a las transformaciones en $\mathbf{R}^3$. La representación del movimiento de un sólido rígido en el espacio Euclídeo tridimensional, esto es, el espacio de matrices 4x4 de transformación homogénea **g**, pertenece al Grupo de Lie **SE(3).** Entonces, la posición y orientación de un sólido rígido queda descrita por un elemento del **SE(3)**, consistente en una matriz de la forma **g** "(2-1)", donde **d** representa el vector de translación y **R** la matriz de rotación espacial.

$$g = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \in SE(3) / SE(3) = \left\{ (d, R) : d \in \Re^3, R \in SO(3) \right\} \in \Re^{4x4} \quad \textbf{(2-1)}$$

El grupo **SE(3)** tienen asociada el Álgebra de Lie **se(3)**, que puede ser identificada con la matrices ξ^ llamadas *twists*, dadas por "(2-2)", junto con la operación *Lie Bracket* dada por "(2-3)" {RobotMan-f(37):***twist*bracket**}.

$$\xi\wedge = \begin{bmatrix} \omega\wedge & \upsilon \\ 0 & 0 \end{bmatrix} \in se(3) / se(3) = \left\{ (\upsilon, \omega^{\wedge}) : \upsilon \in \Re^3, \omega^{\wedge} \in so(3) \right\} \in \Re^{4x4} \quad \textbf{(2-2)}$$

$$\left[\xi_1^{\wedge}, \xi_2^{\wedge} \right] = \xi_1^{\wedge} \xi_2^{\wedge} - \xi_2^{\wedge} \xi_1^{\wedge} \quad \textbf{(2-3)}$$

Donde las matrices antisimétricas de la forma **ω^** "(2-4)", son el álgebra de Lie **so(3)**, correspondiente al Grupo Ortogonal Especial de Lie **SO(3)**, que representa todas las rotaciones en el espacio tridimensional. Se observa que la matrices de **so(3)**, no son sino una transformación que convierte el producto vectorial en $\mathbf{R}^3$ en un producto matricial "(2-4)", por lo que dado un vector **ω** es trivial la operación para convertirlo en matriz **ω^** {RobotMan-f(31):**skew**}. La operación inversa consiste en dada una matriz **ω^**, obtener el vector asociado **ω** {RobotMan-f(43):**unskew**}.

$$\omega\wedge = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} / \forall \omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \wedge \upsilon = \begin{bmatrix} \upsilon_1 \\ \upsilon_2 \\ \upsilon_3 \end{bmatrix} \Rightarrow \quad \omega \times \upsilon = \omega\wedge \cdot \upsilon \quad \textbf{(2-4)}$$

Podemos extraer del álgebra de Lie (i.e., del *twist* ξ^ que es una matriz 4x4), el vector de seis dimensiones que parametriza el *twist*, este es ξ. Lo haremos con el sencillo operador "(2-5)" {RobotMan-f(45):**vee**}. La operación inversa se ejecuta con el operador "(2-6)" {RobotMan-f(46):**wedge**}.

$$\left(\xi^{\wedge}\right)^{\vee} = \begin{bmatrix} \omega\wedge & \upsilon \\ 0 & 0 \end{bmatrix}^{\vee} = \begin{bmatrix} \upsilon \\ \omega \end{bmatrix} = \xi \quad \textbf{(2-5)}$$

$$\left(\xi\right)^{\wedge} = \begin{bmatrix} \upsilon \\ \omega \end{bmatrix}^{\wedge} = \begin{bmatrix} \omega\wedge & \upsilon \\ 0 & 0 \end{bmatrix} = \xi^{\wedge} \quad \textbf{(2-6)}$$

La transformación exponencial, como ya establecimos anteriormente, es la principal conexión entre el grupo de Lie **SE(3)** y su correspondiente álgebra **se(3)** (i.e., por cada **g** perteneciente a **SE(3)**, existe un **ξ^** perteneciente a **se(3)**, tal que **exp(ξ^)=g**). Dado un determinado *twist* (i.e., **ξ=(υ, ω)**), y una determinada magnitud **θ** del movimiento, la fórmula exponencial vendrá dada por "(2-7)" {RobotMan-f(38):***twist***exp**}. De esta forma, la transformación exponencial para un *twist*, equivale a una transformación homogénea y representa el movimiento de un sólido rígido.

$$e^{\hat{\xi}\theta} = \begin{bmatrix} e^{\hat{\omega}\theta} & (I - e^{\hat{\omega}\theta})(\omega \times \upsilon) + \omega\omega^T \upsilon\theta \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \in SE(3); \omega \neq 0$$
$$e^{\hat{\xi}\theta} = \begin{bmatrix} I & \upsilon\theta \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \in SE(3); \omega = 0 \qquad \textbf{(2-7)}$$

Se puede invertir la operación exponencial para una matriz de transformación homogénea, aplicando la operación logaritmo de matrices "(2-8)", para obtener el *twist* **ξ^** {RobotMan-f(24):**rigid*twist***}, y la magnitud **θ** {RobotMan-f(21):**rigidangle**} del movimiento correspondiente.

$$\ln\left(e^{\hat{\xi}\theta}\right) = \ln\begin{bmatrix} e^{\hat{\omega}\theta} & (I - e^{\hat{\omega}\theta})(\omega \times \upsilon) + \omega\omega^T \upsilon\theta \\ 0 & 1 \end{bmatrix} = \ln\begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} = \hat{\xi}\theta \qquad \textbf{(2-8)}$$

En las fórmulas anteriores, la matriz de rotación **R** se calcula como la exponencial de una matriz **ω^** de **so(3)**, que no es sino la representación de la rotación de un cuerpo en el espacio $\mathbf{R^3}$ alrededor de un eje **ω** una cierta magnitud (i.e., ángulo) **θ**, que podemos calcular de una forma eficiente con la conocida fórmula de **Rodrigues** "(2-9)" {RobotMan-f(32):**skewexp**}.

$$e^{\hat{\omega}\theta} = I + \hat{\omega}\sin\theta + \hat{\omega}^2(1 - \cos\theta) \qquad \textbf{(2-9)}$$

Otra transformación importante es la representación adjunta del **SE(3)** y de **se(3)**. Un elemento de **SE(3)** puede ser identificado como un mapa lineal desde el álgebra **se(3)** correspondiente, mediante la representación adjunta. Físicamente, la transformación adjunta describe un cambio en el sistema de referencia. Es por ello, por lo que resulta tan interesante para analizar el movimiento de sólidos rígidos, puesto que normalmente, en estos análisis estamos interesados en conocer las magnitudes del movimiento en varios sistemas de referencia; al menos en un sistema de referencia inercial que denominaremos Sistema Espacial **S** (*Spatial*) y en un sistema de referencia móvil que llamaremos Sistema Operacional **B** (*Body*). Dada una matriz de transformación homogénea **g**, diremos que la transformación adjunta asociada correspondiente $\mathbf{Ad_g}$, queda definida por "(2-10)" {RobotMan-f(20):**rigidadjoint**}.

$$g = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \Rightarrow \quad Ad_g = \begin{bmatrix} R & \hat{d}R \\ 0 & R \end{bmatrix} \qquad \textbf{(2-10)}$$

2.1.2 Geometría de los Movimientos generalizados (*Twists*) ξ.

En esta sección, exploramos algunos de los atributos asociados con un *twist* ξ, que nos dan una visión del modo en que éstos pueden parametrizar de una forma geométrica el movimiento de un sólido rígido. Introducimos un movimiento consistente en una rotación alrededor de un eje, seguida de una translación a lo largo de ese eje. Se llama al movimiento así definido *screw*, por la reminiscencia con el movimiento de un tornillo mecánico. Mostraremos que un *twist* está naturalmente asociado con esa geometría, puesto un *twist* no es sino la versión infinitesimal de un *screw*.

Un *screw* representa la rotación de una magnitud $\mathbf{M}_\xi$ (i.e., **θ** radianes), alrededor de un eje $\mathbf{l}_\xi$, seguida por una traslación a lo largo del mismo eje de valor $\mathbf{h}_\xi$* $\mathbf{M}_\xi$, donde $\mathbf{h}_\xi$ es el valor que llamaremos *pitch* (ver Figura 2-1). Se observa que la transformación relaciona puntos en sus coordenadas iniciales con esos mismos puntos en sus coordenadas finales, por lo que una transformación de tipo *screw* es siempre relativa.

Dado un *screw* con eje $\mathbf{l}_\xi$, *pitch* $\mathbf{h}_\xi$ y magnitud $\mathbf{M}_\xi$, existe un *twist* **ξ=[υ,ω]** de magnitud unitaria, tal que el movimiento asociado por el *screw* se puede generar mediante el *twist* $\mathbf{M}_\xi$*ξ. De modo que se pueden definir las coordenadas *screw* de un determinado *twist* de la siguiente forma:

- ***Pitch* - $\mathbf{h}_\xi$**: Es el ratio entre movimiento traslacional y el rotacional "(2-11)" {RobotMan-f(41):***twistpitch***}.

$$h_\xi = \left\{ \omega^T \upsilon / \|\omega\|^2 \right\} \quad \omega \neq 0$$
$$h_\xi = \{\infty\} \quad \omega = 0 \qquad \textbf{(2-11)}$$

- **Eje – $\mathbf{l}_\xi$**: Línea de giro y traslation "(2-12)" {RobotMan-f(34):***twist***axis}.

$$l_\xi = \left\{ \lambda\omega + \omega \times \upsilon / \|\omega\|^2 : \lambda \in \Re \right\} \quad \omega \neq 0 \qquad \textbf{(2-12)}$$
$$l_\xi = \{ \lambda\omega + 0 : \lambda \in \Re \} \quad \omega = 0$$

- **Magnitud – $\mathbf{M}_\xi$**: Es la rotación neta si existe componente de rotación, en caso contrario es la traslación neta "(2-13)" {RobotMan-f(40):***twist***magnitude}.

$$M_\xi = \{\|\omega\|\} \quad \omega \neq 0$$
$$M_\xi = \{\|\upsilon\|\} \quad \omega = 0 \qquad \textbf{(2-13)}$$

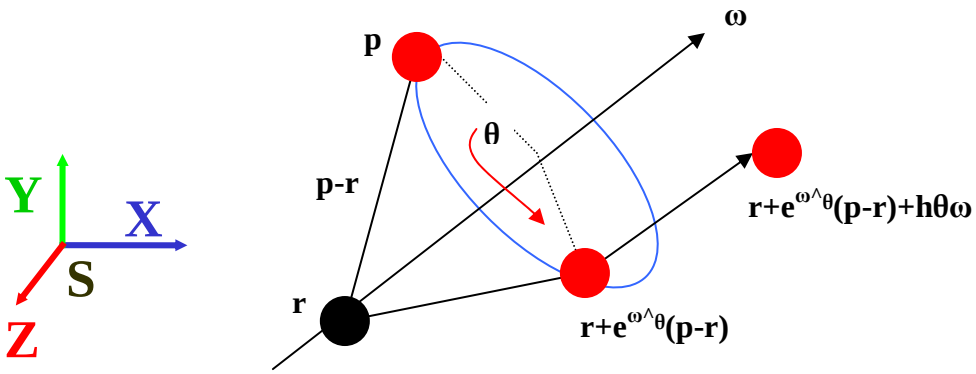


Figura 2-1: Movimiento *Screw* generalizado con rotación no nula.

Esta representación geométrica de un movimiento de tipo *screw* y su representación equivalente en modo de exponencial de un *twist*, tienen unas propiedades geométricas muy interesantes, que serán de utilidad para despejar incógnitas de las ecuaciones cinemáticas en desarrollos matemáticos posteriores.

- **Propiedad 1 – No Afectación del giro sobre su propio eje**: Si el movimiento *screw* es sólo rotativo, entonces la aplicación del mismo a un punto **r** situado en el eje del *twist*, no produce ningún efecto sobre ese punto (lo que es bastante elemental.) Lo que matemáticamente podemos expresar según la ecuación "(2-14)"; el producto de la exponencial de un twist de rotación sobre un punto de su propio eje es igual a ese mismo punto.

$$e^{\hat{\xi}\theta} \cdot r = r \quad \forall \quad r \in l_{\xi} \qquad \textbf{(2-14)}$$

- **Propiedad 2 – Conservación de la norma**: Si el movimiento *screw* es sólo rotativo, entonces se conserva la distancia (i.e., el módulo) entre el punto dado como resultado de la aplicación del *screw* a un punto cualquiera **p** y un punto **r** situado en el eje del *twist*. Lo que matemáticamente podemos expresar según la ecuación "(2-15)"; la norma de la diferencia entre el producto de la exponencial de un *twist* por un punto cualquiera y un punto de su propio eje, es igual a la norma de la diferencia entre esos dos mismos puntos.

$$\left\| e^{\hat{\xi}\theta} \cdot p - r \right\| = \left\| p - r \right\| \qquad \textbf{(2-15)}$$

Se señala que estas dos propiedades son igualmente válidas para una sucesión de **POE**, siempre que todas ellas tengan las propiedades que se establecen en los antecedentes de los enunciados.

2.1.3 Geometría de las Fuerzas generalizadas – (*Wrenches*) φ.

En esta sección se describe cómo los esfuerzos generalizados (fuerzas y momentos) son elementos del espacio dual al del álgebra de Lie **se(3)**. Uno naturalmente asocia fuerzas con velocidades lineales y momentos con velocidades angulares, cuando piensa en las posibles formulaciones estándar del Trabajo, y como resultado, aparece clara la dualidad espacial entre fuerzas y velocidades. De este modo, como una representación dual del vector de seis dimensiones que parametriza un *twist* **ξ=[υ,ω]** "(2-5)", podemos expresar la parametrización de un conjunto de fuerzas generalizadas, por otro vector de seis dimensiones que llamamos *wrench* **φ =[*f*,τ]** "(2-16)", donde ***f*** es la componente lineal de esfuerzos y **τ** la componente rotacional.

$$\varphi = \begin{bmatrix} f \\ \tau \end{bmatrix} \qquad \textbf{(2-16)}$$

Mostraremos que un *wrench* también está asociado con la geometría de un *screw*, al poder representar éste la aplicación de un fuerza de magnitud $\mathbf{M}_{\varphi}$ a lo largo de la línea definida por un eje $\mathbf{l}_{\varphi}$, seguida por un par de magnitud $\mathbf{h}_{\varphi}\mathbf{*M}_{\varphi}$ alrededor del mismo eje,

(donde $\mathbf{h}_\varphi$ es el valor del *pitch*). Dado un *screw* con eje $\mathbf{l}_\varphi$, *pitch* $\mathbf{h}_\varphi$ y magnitud $\mathbf{M}_\varphi$, existe un *wrench* $\boldsymbol{\varphi} = [\boldsymbol{f}, \boldsymbol{\tau}]$ de magnitud unitaria, tal que el esfuerzo asociado con el *screw* se puede generar mediante el *wrench* $\mathbf{M}_\varphi * \boldsymbol{\varphi}$. De modo que se pueden definir las coordenadas *screw* de un determinado *wrench* de la siguiente forma:

- ***Pitch* - $\mathbf{h}_\varphi$**: El *pitch* de un *wrench* es el ratio entre el par angular y la fuerza lineal {RobotMan-f(51):***wrenchpitch***}.

$$\begin{aligned} h_\varphi &= \left\{ f^T \tau / \| f \|^2 \right\} \quad f \neq 0 \\ h_\varphi &= \{ \infty \} \quad f = 0 \end{aligned} \qquad \textbf{(2-17)}$$

- **Eje – $\mathbf{l}_\varphi$**: La línea de aplicación del par y la fuerza {RobotMan-f(47):***wrench***axis**}.

$$\begin{aligned} l_\varphi &= \left\{ \lambda f + f \times \tau / \| f \|^2 : \lambda \in \Re \right\} \quad f \neq 0 \\ l_\varphi &= \left\{ \lambda \tau + 0 : \lambda \in \Re \right\} \quad f = 0 \end{aligned} \qquad \textbf{(2-18)}$$

- **Magnitud – $\mathbf{M}_\varphi$**: Es el valor neto de la fuerza lineal si existe componente lineal, en caso contrario es el par neto {RobotMan-f(50):***wrench***magnitude**}.

$$\begin{aligned} M_\varphi &= \{ \| f \| \} \quad f \neq 0 \\ M_\varphi &= \{ \| \tau \| \} \quad f = 0 \end{aligned} \qquad \textbf{(2-19)}$$

2.1.4 La Fórmula del Producto de Exponenciales POE.

En este apartado se presenta la fórmula del Producto de Exponenciales **POE** que nos servirá para describir de una forma geométrica la cinemática de un sólido rígido. El **POE** es el corazón de las formulaciones a los problemas cinemáticos, tanto directos como inversos, que se desarrollan en esta tesis.

Como ya vimos en 2.1.1, la transformación exponencial puede ser interpretada como un operador que relaciona puntos de un sólido rígido, desde sus coordenadas iniciales a sus nuevas coordenadas en el sistema de referencia, después de que el sólido haya sufrido un movimiento de tipo *screw*. Como una generalización de la transformación exponencial, el **POE** representa el sistema de referencia tras una concatenación de *screws* que han afectado al movimiento del sólido rígido.

La formula del **POE** cuya ecuación **g(θ)** "(2-20)" es una descripción para la cinemática directa de un sólido rígido que sufre una concatenación **n** de movimientos *screws* (caracterizados por sus correspondientes *twists* **ξ** y magnitudes **θ**), una vez que se elige una posición de referencia **g(0)** para un sistema coordenado.

$$g(\theta) = \prod_{i=1}^{n} e^{\xi_i^{\wedge} \theta_i} \cdot g(0) \qquad \textbf{(2-20)}$$

2.1.5 Los Manipuladores Jacobianos Geométricos.

Tradicionalmente, se describe el Jacobiano de un determinado movimiento de un sólido rígido, mediante la diferenciación de la cinemática directa de ese movimiento. Sin embargo, este Jacobiano no es una cantidad natural, puesto que la descripción sólo es válida de forma local. Para corregir este problema, se define un nuevo concepto en términos de *twists*, que llamaremos Manipulador Jacobiano de una cinemática directa. Veremos que el **POE** conduce a descripciones naturales y explícitas del Manipulador Jacobiano, que destacan la geometría de un mecanismo y no tiene ninguna de las desventajas de la representación local. Las dos representaciones fundamentales son:

- **El Manipulador Jacobiano Espacial - J^s**: Relaciona las velocidades correspondientes a la concatenación de movimientos *screws* (caracterizados por sus *twists* ξ y magnitudes derivadas de $\boldsymbol{\theta}$) aplicados a un sólido rígido, con las velocidades de ese sólido $\mathbf{V^s}$ en el sistema de referencia inercial o espacial **S** "(2-21)". El Manipulador Jacobiano viene dado por "(2-22)" {RobotMan-f(33):**spatialjacobian**}. De esta forma el Manipulador Jacobiano presenta un verdadero significado geométrico natural, puesto que cada columna que lo compone corresponde a un *twist* transformado por todos los anteriores, esto es, podemos con un poco de práctica, calcular el Manipulador Jacobiano por inspección, ¡sin necesidad de efectuar ninguna derivada!.

$$V^s = J^s(\theta)\dot{\theta} \qquad \textbf{(2-21)}$$

$$J^s(\theta) = \begin{bmatrix} \xi'_1 & \xi'_2 & \cdots & \xi'_n \end{bmatrix} \quad \wedge \quad \xi'_i = Ad_{\left(\prod_{i=1}^{i-1} e^{\xi_i^{\wedge}\theta_i}\right)}\xi_i \qquad \textbf{(2-22)}$$

- **El Manipulador Jacobiano Operacional - J^b**: Relaciona las velocidades correspondientes a la concatenación de movimientos *screws* (caracterizados por sus *twists* ξ y magnitudes derivadas de $\boldsymbol{\theta}$) aplicados a un sólido rígido, con las velocidades $\mathbf{V^b}$ en el sistema de referencia móvil del sólido (body) **B** "(2-23)". El Jacobiano viene dado por "(2-24)" {RobotMan-f(1):**bodyjacobian**}.

$$V^b = J^b(\theta)\dot{\theta} \qquad \textbf{(2-23)}$$

$$J^b(\theta) = \begin{bmatrix} \xi''_1 & \cdots & \xi''_{n-1} & \xi''_n \end{bmatrix} \quad \wedge \quad \xi''_i = Ad^{-1}_{\left(\prod_{i}^{n} e^{\xi_i^{\wedge}\theta_i}\cdot g(0)\right)}\xi_i \qquad \textbf{(2-24)}$$

El Manipulador Jacobiano también relaciona el esfuerzo final en un sólido rígido, con todos los esfuerzos (*wrenches*) aplicados en una concatenación de *screws*. Lo que es una relación fundamental para avanzar en la dinámica y control de robots.

El significado geométrico del Manipulador Jacobiano, además presenta muchas ventajas para el análisis mecánico, como es la disminución de las frecuentes singularidades que se presentan sólo por el hecho de realizar otros tipos de parametrización, llevando a falsas conclusiones sobre las configuraciones y velocidades alcanzables.

2.2 Matemática de Lie en el Movimiento del Sólido Rígido.

El tratamiento del movimiento que utilizamos en esta tesis se basa en la matemática de Grupos de Lie y la teoría de *srews* presentadas en el punto anterior (ver 2.1). Se desvía del estándar utilizado en la mayoría de trabajos, que siguen prefiriendo la formulación de parámetros de **Denavit-Hartenberg (D-H)** [30].

El teorema de Chasles establece que cualquier movimiento de un sólido rígido puede ser realizado mediante una rotación alrededor de un eje más una traslación paralela a ese mismo eje, esto es, mediante un movimiento de tipo *screw*. Pero resulta que la versión infinitesimal de ese movimiento es un elemento del álgebra de Lie **se(3)**, esto es, un *twist* $\xi^\wedge$ (ver 2.1.2).

El teorema de Poinsot establece que cualquier colección de esfuerzos aplicados a un sólido rígido son equivalentes a una fuerza aplicada en la dirección de un eje fijo, más un par aplicado a ese mismo eje, esto es, mediante un esfuerzo de tipo *screw*. Pero resulta que la versión infinitesimal de ese esfuerzo, es la representación dual de un elemento del álgebra de Lie **se(3)**, esto es, el elemento que llamamos *wrench* (ver 2.1.3).

Las ventajas de la teoría mecánica de *screws* y la matemática de Grupos de Lie asociada son las siguientes:

- Permiten una descripción realmente geométrica del movimiento que facilita el análisis mecánico.
- Permiten una descripción sin singularidades debidas al uso de coordenadas locales (como sucede por el contrario con las representaciones de ángulos de Euler y las de **D-H**), ya que es posible el uso de sólo dos sistemas coordenados de referencia, el de la base **S** y el de la herramienta **H**.
- Permite usar la misma representación matemática para diferentes tipos de movimientos, esto es, tanto para traslaciones como para rotaciones.
- Proporciona una descripción explícita realmente natural del Manipulador Jacobiano, que no tiene las desventajas de la representación local del tradicional Jacobiano (ver 2.1.5).

Para representar el movimiento de un sólido rígido utilizaremos la transformación exponencial de un *twist* (ver 2.1.1), que proporciona el movimiento relativo del mismo. La interpretación de esta transformación no es la relación entre los puntos de un sistema de coordenadas a otro, sino la relación entre las coordenadas iniciales de los puntos con sus coordenadas finales, tras aplicar un movimiento al sólido rígido.

Puesto que las cadenas abiertas de sólidos rígidos (e.g., robots manipuladores) están constituidas mediante la conexión de diferentes articulaciones (típicamente de revolución o prismáticas), usando eslabones rígidos, el movimiento queda restringido a un subgrupo de **SE(3)**, haciendo de la exponencial de un *twist* la representación natural para el análisis mecánico de estos sistemas.

En numerosas aplicaciones es muy importante la habilidad para derivar las ecuaciones del movimiento con respecto a los parámetros cinemáticos y dinámicos de interés. Las ecuaciones del movimiento resultantes de la formulación con Grupos de Lie pueden ser derivadas de una forma muy directa a alto nivel, como consecuencia de que la primitiva matemática básica es la exponencial de matrices.

Para otras aplicaciones (e.g., control fuerza-posición o control de trayectorias libres de colisiones), es conveniente expresar las ecuaciones del movimiento en términos de un conjunto de coordenadas generalizadas para un sistema de referencia asociado al extremo del robot, que es lo que llamaremos coordenadas en el espacio operacional, definido como el espacio de configuraciones de la herramienta. Los Grupos de Lie, permiten trabajar con sólo dos sistemas de referencia (i.e., uno unido a la base **S** y otro unido al último eslabón **H**), sin necesidad de sistemas de referencia locales.

Los grupos y álgebras de Lie junto con el **POE** constituyen las herramientas matemáticas ideales para resolver los problemas mecánicos de cadenas abiertas de sólidos rígidos. Los problemas típicos son los de cinemática y dinámica, tanto directa como inversa. El problema cinemático directo determina las posiciones finales del sistema dados los valores de los **GDL** del mismo, mientras que el problema cinemático inverso determina los valores de los **GDL** que llevan al sistema hasta la posición final deseada. El problema dinámico directo determina las aceleraciones del sistema cuando aplicamos determinadas fuerzas al mismo, conociendo las posiciones y velocidades iniciales, en tanto que el problema dinámico inverso determina los pares y fuerzas necesarias que deben aplicarse para producir el movimiento deseado.

Para cadenas de sólidos rígidos no es necesario posicionar sistemas de referencia locales a cada uno de los eslabones, debido a que todas las matrices del **POE** se computan con respecto a un único sistema de referencia fijo. Otra ventaja en el modelado con la fórmula del **POE**, es el hecho de que las articulaciones típicas de un sistema robotizado (i.e., prismáticas y de revolución) son tratadas de un modo matemáticamente uniforme, mientras que esto no es así al usar otros sistemas de representación como los parámetros de **D-H** [30].

2.2.1 Sistemas Mecánicos con Topología en Árbol.

Un sistema mecánico con topología de árbol es aquel formado por una colección de cadenas abiertas de sólidos rígidos, que llamaremos ramas, acopladas mediante un eslabón común que normalmente está unido al sistema de referencia de la base. Estos sistemas son conjuntos de cadenas cinemáticas, usaremos las mismas herramientas geométricas introducidas anteriormene para obtener las ecuaciones de movimiento. Como ya presentaron Featherstone [31] y Ploen [94], los algoritmos recursivos de Newton-Euler pueden ser modificados para describir sistemas con topología de árbol.

El estudio de los sistemas con topología en árbol es importante puesto que un robot humanoide puede ser considerado como un sistema mecánico similar, sólo que el eslabón común (i.e., el tronco) de las cadenas de sólidos rígidos (i.e., los brazos y piernas), no está asociado a una base con sistema de referencia inercial, sino que se mueve. Además, en la operación de robots humanoides se producen bucles cinemáticos cerrados, estructura mecánica que también ha sido tratada en el estudio de los sistemas con topología en árbol por Nakamura et al [82].

2.3 Cinemática de Robots.

La cinemática de un robot describe la relación entre el movimiento de las articulaciones del robot y el movimiento resultante de los cuerpos rígidos que lo constituyen.

2.3.1 Problema Cinemático Directo.

El problema cinemático directo de un robot formado por una cadena abierta de sólidos rígidos con **n** grados de libertad (**GDL**), consiste en determinar la configuración (i.e., $\mathbf{g_{sh}(\theta)}$ posición y rotación) del elemento extremo final del robot (i.e., el sistema de referencia **H** de la herramienta), dados los valores **θ** de todos los **GDL** del sistema, esto es, el valor de los giros para las articulaciones de revolución y el valor de los desplazamientos para las articulaciones prismáticas.

Para el desarrollo cinemático definimos los siguientes conceptos:

- **S**: Es el sistema de referencia (inercial) unido a la base del manipulador.
- **H**: Es el sistema de referencia (móvil) unido a la herramienta (i.e., último eslabón).
- **θ**: Es el valor de todos los **GDL** correspondientes a las articulaciones del robot.
- $\mathbf{g_{sh}(0)}$: Es la transformación entre **H** y **S** en la configuración de referencia del manipulador (esta es la correspondiente a **θ=0**).
- **ω**: Es un vector unitario en la dirección que define el giro de una articulación de revolución.
- **r**: Es cualquier punto en el eje de una articulación de revolución **ω**.
- **υ**: Es un vector unitario señalando en la dirección de translación de una articulación prismática.

A cada articulación de revolución le corresponde un *twist* de la forma "(2-25)", que define su movimiento.

$$\xi_i = \begin{bmatrix} \upsilon \\ \omega \end{bmatrix} = \begin{bmatrix} -\omega_i \times r_i \\ \omega_i \end{bmatrix} \qquad \textbf{(2-25)}$$

A cada articulación prismática le corresponde un *twist* de la forma "(2-26)", que define su movimiento.

$$\xi_i = \begin{bmatrix} \upsilon \\ \omega \end{bmatrix} = \begin{bmatrix} \upsilon_i \\ 0 \end{bmatrix} \qquad \textbf{(2-26)}$$

Mediante la fórmula del **POE** es posible generalizar la cinemática directa del robot manipulador, que viene dada por $\mathbf{g_{sh}(\theta)}$ "(2-27)".

$$g_{sh}(\theta) = e^{\xi_1^{\wedge}\theta_1} \cdot e^{\xi_2^{\wedge}\theta_2} \cdots e^{\xi_n^{\wedge}\theta_n} \cdot g_{sh}(0) \qquad \textbf{(2-27)}$$

2.3.2 Problema Cinemático Inverso.

Dada una configuración deseada (i.e., $\mathbf{g_{sh}(\theta)}$ posición y rotación) para el sistema de referencia de la herramienta **H** de un robot, el problema cinemático inverso consiste en encontrar valores $\boldsymbol{\theta}$ para los n **GDL** (i.e., valores de los giros para las articulaciones de revolución y valores de los desplazamientos para las articulaciones prismáticas), que una vez aplicados llevan a la herramienta hasta la configuración deseada. Este problema puede tener múltiples soluciones, una única solución o ninguna (es un problema complejo). Por ejemplo, un mecanismo en cadena abierta de seis grados de libertad puede tener hasta 16 posibles soluciones, para un problema cinemático inverso que tiene 6 variables y 12 ecuaciones.

Tradicionalmente, las soluciones a la cinemática inversa se separan en dos clases:

- **Soluciones cerradas**: Se basan en identidades geométricas, muchas veces son soluciones empíricas o de aprendizaje. Permiten cálculos rápidos y eficientes.
- **Soluciones numéricas**: Se basan en algoritmos iterativos para simplificar un sistema de ecuaciones, que se aplican hasta encontrar la necesaria convergencia. Se aplican a sistemas con muchos **GDL**, puesto que para este tipo de mecanismos las soluciones geométricas cerradas son muy complicadas. Estos métodos se suelen convertir en procedimientos de "fuerza bruta" muy poco adecuados para problemas en tiempo real.

Basándonos en la matemática de grupos de Lie y el **POE** (ver 2.3.3) se puede retomar la idea de obtener soluciones geométricas cerradas para la cinemática inversa de robots con muchos **GDL**, como se demuestra en los ejemplos presentados con detalle en el Apéndice-A, que puede servir, por otra parte, como una introducción a la aplicación de estas técnicas matemáticas para aquellos que sean más novicios en estos temas.

Todo ello evidentemente proporciona muchas ventajas para el desarrollo de algoritmos que sean rápidos, eficientes, sin problemas de convergencia y aplicables en tiempo real. Además, en esta tesis se extenderán estos desarrollos para su aplicación a la cinemática de sistemas mecánicos móviles con múltiples **GDL** y robots humanoides (ver 3.4.2).

2.3.3 El POE y los Problemas Canónicos de Paden-Kahan.

Usando el **POE** es posible desarrollar algoritmos geométricos para resolver el problema cinemático inverso. Este método fue originalmente presentado por Paden [89] y construido sobre diversos trabajos no publicados de Kahan [51].

La idea de este método es reducir la complejidad del problema cinemático inverso mediante la división del mismo en subproblemas más sencillos, que ocurren de forma frecuente en la mecánica de robots, y cuyas soluciones geométricas son conocidas. Estos subproblemas cinemáticos inversos canónicos, tienen un claro significado geométrico y su solución es numéricamente estable. Vamos a presentar tres problemas canónicos a los que llamaremos problemas de Paden-Kahan (Uno, Dos y Tres) que nos sirven para solucionar casi todos los problemas planteables para robots industriales con articulaciones de rotación.

Siguiendo el mismo enfoque teórico, es posible desarrollar nuevos problemas canónicos para dar solución a problemas más complejos que no sean resolubles por

reducción a los tres ya mencionados que vamos a ver a continuación en detalle. Un ejemplo es el nuevo problema canónico de Pardos-Uno que veremos en este capítulo.

En esta tesis haremos uso intensivo de los tres problemas canónicos que presentamos a continuación, extendiendo su aplicación a la mecánica de robots humanoides. Para entender en detalle la utilización práctica de los mismos, se recomienda revisar los ejemplos A.1 y A.2. del Apéndice-A.

Problema de PADEN-KAHAN-UNO - Rotación alrededor de un eje: Este ejercicio geométrico canónico resuelve de una forma directa y cerrada el problema cinemático inverso que se nos plantea cuando tenemos un movimiento deseado de giro alrededor de un eje, y queremos conocer cuál debe de ser el ángulo necesario para obtenerlo.

La formalización del problema de Paden-Kahan-Uno consiste en encontrar el valor del ángulo **θ** a girar alrededor de un eje **ω** para mover un punto **p** hasta llevarlo a una posición dada por **k** (ver Figura 2-2).

El problema se expresa en forma de Lie según la ecuación "(2-28)", esto es, la exponencial de un *twist* correspondiente a un giro de valor **θ** aplicado al punto **p** es igual al punto **k**. Se demuestra sin muchas dificultades que existe una única solución geométrica cerrada **θ**, ya que nunca consideramos la solución trivial igual a **2π-θ**. La solución viene dada por "(2-29)" {RobotMan-f(13):**padenkahanone**}.

$$e^{\xi^{\wedge}\theta}\cdot p = k \quad \wedge \quad \xi = \begin{bmatrix} \upsilon \\ \omega \end{bmatrix} = \begin{bmatrix} -\omega \times r \\ \omega \end{bmatrix} \qquad \textbf{(2-28)}$$

$$\theta = a\tan 2\left[\omega^T\left(u'\times v'\right), u'^T\cdot v'\right] \quad \wedge \quad \begin{cases} u' = u - \omega\omega^T u \\ v' = v - \omega\omega^T v \end{cases} \qquad \textbf{(2-29)}$$

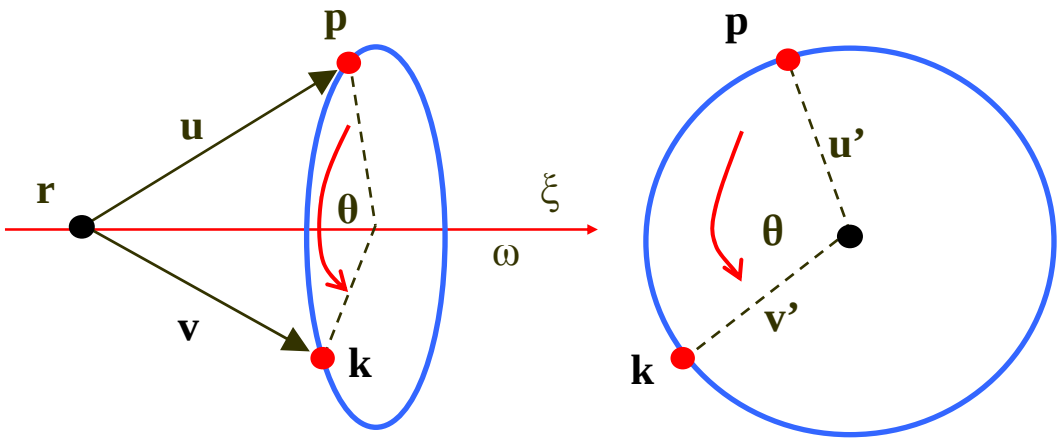


Figura 2-2: Problema de Paden-Kahan Uno – Rotación alrededor de un eje.

Problema de PADEN-KAHAN-DOS - Rotación alrededor de dos ejes consecutivos: Este ejercicio canónico resuelve de una forma geométrica directa y completa, el problema cinemático inverso consistente en tener dos movimientos de giros consecutivos alrededor de dos ejes que se cruzan, queriendo obtener cuales deben de ser los ángulos que hacen posible ese movimiento deseado. La formalización del problema de Paden-Kahan-Dos en términos de matemática de Lie, consiste en encontrar los valores de los ángulos que es necesario girar alrededor de dos ejes ω_2 y ω_1 de forma consecutiva, para mover un punto **p** hasta llevarlo al punto **k** (ver Figura 2-3).

El problema se formula con la ecuación de Lie "(2-30)". Se demuestra de forma geométrica que existen una solución geométrica cerrada doble, consistente en general en dos parejas de valores θ_2 y θ_1, aunque puede existir una única solución como pareja de valores θ_2 y θ_1 (en realidad son dos soluciones iguales), e incluso puede no tener solución, como se infiere fácilmente de la Figura 2-3. No se consideran aquí las soluciones triviales dadas por $2\pi - \theta_1$ o $2\pi - \theta_2$. Para obtener las soluciones, aplicaremos dos veces el problema de Paden-Kahan-Uno para ξ_2 y ξ_1, una vez obtenido el punto doble **c** según "(2-31)" {RobotMan-f(15):**padenkahantwo**}.

$$e^{\hat{\xi_1}\theta_1}\cdot e^{\hat{\xi_2}\theta_2}\cdot p = e^{\hat{\xi_1}\theta_1}\cdot c = k \quad \wedge \quad \xi_1 = \begin{bmatrix} -\omega_1 \times r \\ \omega_1 \end{bmatrix} \wedge \xi_2 = \begin{bmatrix} -\omega_2 \times r \\ \omega_2 \end{bmatrix} \qquad \textbf{(2-30)}$$

$$c = r + \alpha\omega_1 + \beta\omega_2 \pm \gamma(\omega_1 \times \omega_2) \quad \wedge \quad \begin{cases} \alpha = \dfrac{(\omega_1^T\omega_2)\omega_2^T u - \omega_1^T v}{(\omega_1^T\omega_2)^2 - 1} \\ \beta = \dfrac{(\omega_1^T\omega_2)\omega_1^T v - \omega_2^T u}{(\omega_1^T\omega_2)^2 - 1} \\ \gamma^2 = \dfrac{\|u\|^2 - \alpha^2 - \beta^2 - 2\alpha\beta\omega_1^T\omega_2}{\|\omega_1 \times \omega_2\|^2} \end{cases} \qquad \textbf{(2-31)}$$

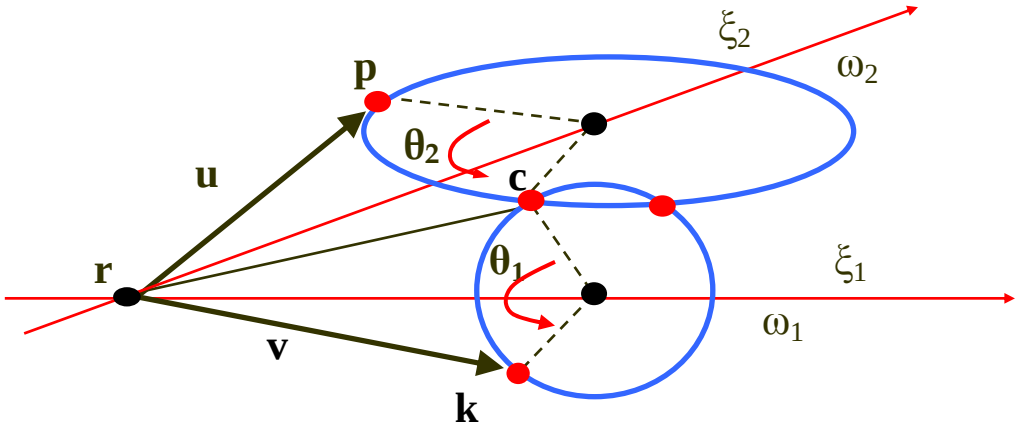


Figura 2-3: Problema de Paden-Kahan Dos – Rotación alrededor de dos ejes consecutivos.

Problema de PADEN-KAHAN-TRES - Rotación hasta una cierta distancia: Este ejercicio canónico resuelve de una forma geométrica cerrada y completa, el problema cinemático inverso que se nos plantea cuando queremos tener un movimiento de giro alrededor de un eje, que aplicado a un punto (o sólido rígido) lo lleva hasta una distancia deseada de otro punto en el espacio tridimensional, debiendo conocer cuál es el ángulo que tenemos que aplicar para obtener ese movimiento.

La formalización del problema de Paden-Kahan-Tres consiste en encontrar el valor del ángulo **θ** a girar alrededor de un eje **ω**, para mover un punto **p** hasta llevarlo a una posición que se encuentra a una distancia **δ** de un punto **k** conocido (ver Figura 2-4).

El problema se expresa en forma de Lie según la ecuación "(2-32)", esto es, la norma de la diferencia entre el punto resultante de aplicar un movimiento *screw* de giro al punto **p**, y el punto **k**, es igual a la constante **δ**. Se demuestra geométricamente que existe una solución **θ** doble (no consideramos las soluciones triviales **2π-θ**), que viene dada por la ecuación "(2-33)" {RobotMan-f(14):**padenkahanthree**}.

$$\left\| e^{\hat{\xi}\theta} \cdot p - k \right\| = \delta \quad \wedge \quad \xi = \begin{bmatrix} -\omega \times r \\ \omega \end{bmatrix} \qquad \textbf{(2-32)}$$

$$\theta = \theta_0 \pm \cos^{-1}\left(\frac{\|u'\|^2 + \|v'\|^2 - \delta'^2}{2\|u'\|\|v'\|} \right) \wedge \begin{cases} \theta_0 = a\tan 2\left[\omega^T (u' \times v'), u'^T \cdot v'\right] \\ \delta'^2 = \delta^2 - \left|\omega^T (p-k)\right|^2 \\ u' = u - \omega\omega^T u \\ v' = v - \omega\omega^T v \end{cases} \qquad \textbf{(2-33)}$$

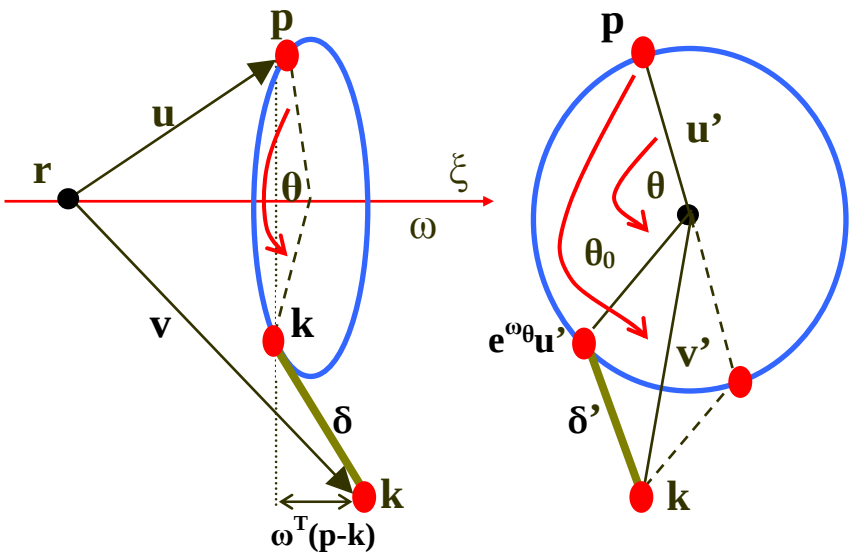


Figura 2-4: Problema de Paden-Kahan Tres – Rotación hasta una cierta distancia.

2.3.4 El POE y el Problema Canónico de Pardos-Uno.

Presentamos un nuevo problema canónico que llamaremos de **Pardos-Uno**. Este problema es un pequeño desarrollo teórico original que permite resolver de forma geométrica, con la matemática de Lie y el **POE**, la cinemática inversa de robots con articulaciones de traslación. La utilidad de este problema canónico puede no verse clara en una primera introducción, pero se puede apreciar su potencial en la resolución cerrada y completa del problema cinemático inverso de un robot de tipo STANFORD de 6 **GDL** en el apéndice A.1 de esta tesis.

Problema de PARDOS-UNO – Traslación hasta una cierta distancia: El ejercicio de cinemática inversa resuelve de una forma cerrada y geométrica, el problema planteado al querer realizar un movimiento de traslación a lo largo de un eje, que aplicado a un punto en el espacio tridimensional, lo lleva hasta una distancia deseada de otro punto dado, por lo que necesitamos conocer cuál debe de ser el valor del desplazamiento que hay que aplicar para obtener ese movimiento.

La formalización del problema de Pardos-Uno consiste en encontrar el valor del desplazamiento **θ** a lo largo de un eje **υ**, necesario para llevar un punto **p** a una posición que se encuentre a una distancia **δ** de un punto **k** conocido (ver Figura 2-5).

En la matemática de Lie, el problema se expresa según la ecuación "(2-34)", esto es, la norma de la diferencia entre el punto resultante de aplicar un movimiento *screw* de traslación al punto **p**, y un punto **k,** es igual a la constante **δ**. Se demuestra geométricamente que existe una solución **θ** doble, que viene dada por la ecuación "(2-35)" {RobotMan-f(16):**pardosone**}.

$$\left\| e^{\hat{\xi}\theta} \cdot p - k \right\| = \delta \quad \wedge \quad \xi = \begin{bmatrix} \upsilon \\ 0 \end{bmatrix} \qquad \textbf{(2-34)}$$

$$\theta = sign\theta \cdot \upsilon^T (p-k) \pm I \quad \wedge \quad \begin{cases} sign\theta = sign\left[\upsilon^T (k-r) - \upsilon^T (p-r)\right] \\ I^2 = \delta^2 - |p-k|^2 + \left|\upsilon^T (p-k)\right|^2 \end{cases} \qquad \textbf{(2-35)}$$

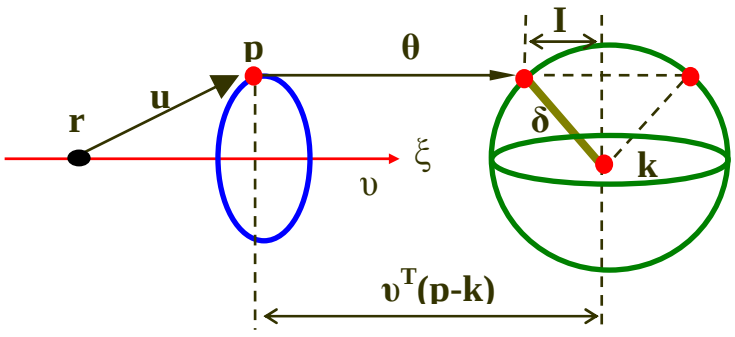


Figura 2-5: Problema de Pardos Uno – Traslación hasta una cierta distancia.

2.4 Dinámica de Robots.

En este apartado presentamos la dinámica de robots manipuladores haciendo uso explícito de los *twist*s de la matemática de grupos de Lie y explorando el rol que la cinemática 2.3 juega en las ecuaciones del movimiento, como fundamento de posibles extensiones de esta formulación a la dinámica de robots humanoides.

La dinámica de un manipulador describe el modo en que el robot se mueve, en respuesta a las fuerzas de los actuadores que actúan sobre él. Esta descripción dinámica utiliza un conjunto no lineal de ecuaciones diferenciales ordinarias de segundo orden, que dependen de las propiedades cinemáticas e inerciales del robot.

Las ecuaciones del movimiento pueden ser generadas por la aplicación de las ecuaciones de Newton-Euler a cada uno de los eslabones de la cadena de sólidos rígidos, esto es, mediante la suma de todas las fuerzas que actúan sobre el robot.

Sin embargo, y aunque sean equivalentes a las ecuaciones de Newton-Euler, nosotros derivaremos las ecuaciones dinámicas mediante las ecuaciones de Lagrange, porque son una formulación que permite reducir el número de ecuaciones necesarias para describir el movimiento de un sistema, desde un número igual al de partículas (i.e. número de elementos mecánicos del robot), hasta un número igual al de coordenadas generalizadas. Tienen la ventaja de que sólo se necesita computar las energías potencial y cinética del sistema, por lo que tienden a ser menos propensas a errores que la integración de las fuerzas inerciales, Coriolis, centrífugas y otras, que actúan sobre los eslabones del robot. Además, las ecuaciones de Lagrange permiten determinar y explotar las propiedades estructurales de la dinámica del robot, lo que es útil para realizar análisis y controles de alto nivel.

2.4.1 Ecuaciones de Lagrange.

En el marco Lagrangiano, una vez que un conjunto apropiado de coordenadas generalizadas **θ** ha sido elegido (e.g., los ángulos de las articulaciones de revolución y los desplazamientos lineales para las juntas prismáticas), se pueden generar las ecuaciones del movimiento mediante las ecuaciones de Lagrange, expresando las fuerzas aplicadas al sistema en términos de componentes aplicados según las coordenadas generalizadas; estas componentes son las Fuerzas Generalizadas.

Para escribir las ecuaciones del movimiento, definimos el Lagrangiano **L**, como la diferencia entre la energía cinética **K** y potencial **V** del sistema mecánico "(2-36)".

$$L\left(\theta,\dot{\theta}\right)= K\left(\theta,\dot{\theta}\right)- V\left(\theta\right) \qquad \textbf{(2-36)}$$

Las ecuaciones de Lagrange para el movimiento de un sistema mecánico con coordinadas generalizadas **θ** y Lagrangiano **L**, vienen dadas por la ecuación. "(2-37)". Donde **Γ** son las Fuerzas Generalizadas que actúan sobre el mecanismo.

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}}-\frac{\partial L}{\partial \theta}=\Gamma \qquad \textbf{(2-37)}$$

2.4.2 Problema Dinámico Inverso.

Consiste en encontrar las fuerzas de los actuadores de un robot que consiguen generar el movimiento deseado para el mecanismo. Una vez que conocemos las ecuaciones del movimiento de un robot, si el modelo dinámico fuese perfecto, podríamos encontrar los pares y fuerzas necesarios directamente mediante las ecuaciones de Lagrange. En la práctica, debido a errores en el modelo, presencia de ruido o errores en las condiciones iniciales, es necesario aplicar un sistema de control 2.5.

Utilizaremos la matemática de grupos de Lie para definir el Lagrangiano "(2-36)" en términos de las posiciones y velocidades de las articulaciones de un robot con **n** eslabones, llegando a la formulación "(2-38)", que incluye dos nuevos conceptos:

- **La Matriz de Inercia del Manipulador - M(θ):** viene dada por la fórmula "(2-39)" {RobotMan-f(8):**linkinertiasum**}, donde $\mathbf{J^b}$ es el manipulador Jacobiano operacional (2.1.5).
- **La Matriz de Inercia Generalizada - μ_i**: se define para cada eslabón y es necesaria para desarrollar **M(θ),** y viene dada por "(2-40)", donde $\mathbf{m_i}$ es la masa del eslabón y $\mathbf{\Psi_i}$ su tensor de inercia correspondiente.

$$L\left(\theta,\dot{\theta}\right)=\frac{1}{2}\dot{\theta}^{T}M\left(\theta\right)\dot{\theta}-V\left(\theta\right) \qquad \textbf{(2-38)}$$

$$M\left(\theta\right)=\sum_{i=1}^{n}J_{i}^{b^{T}}\left(\theta\right)\cdot\mu_{i}\cdot J_{i}^{b}\left(\theta\right) \qquad \textbf{(2-39)}$$

$$\mu_{i}=\begin{bmatrix} m_{i}I & 0 \\ 0 & \Psi_{i} \end{bmatrix} \qquad \textbf{(2-40)}$$

Sustituyendo en las ecuaciones de Lagrange "(2-37)" la nueva definición de Lagrangiano "(2-38)", llegamos a la formulación para las ecuaciones de movimiento del robot "(2-41)", que directamente nos resuelve el problema dinámico inverso, puesto que nos da el valor de los pares actuadores necesarios **Γ** para conseguir las posiciones, velocidades y aceleraciones en las articulaciones del robot. En la ecuación "(2-41)" el primer término recoge **M(θ)** tal como la definimos en "(2-39)", el segundo término incluye la matriz **C** que tiene en cuenta los términos centrífugos y de Coriolis, y el tercer término incluye la Matriz de Potencial **N** que tiene en cuenta las fuerzas externas de gravedad y fricción.

$$M\left(\theta\right)\ddot{\theta}+C\left(\theta,\dot{\theta}\right)\dot{\theta}+N\left(\theta,\dot{\theta}\right)=\Gamma \qquad \textbf{(2-41)}$$

Puede que en este punto no se aprecie la elegancia de la solución dinámica vectorial dada por "(2-41)", pero a continuación vamos a ver que la matemática de grupos de Lie nos va a permitir expresar de forma explícita la matriz de Coriolis, que es quizá el término más complicado de obtener, de modo que el vector de pares o fuerzas **Γ** que es solución al problema dinámico inverso quedará definido en forma casi geométrica, mediante un conjunto de operaciones matriciales que son básicamente **POE**.

La Matriz C de Coriolis se puede definir de muchas maneras, pero nosotros elegimos una que convierte la siguiente expresión en matriz antisimétrica, lo que hace que se mantenga la **propiedad de pasividad**, que implica la conservación de la energía neta del robot, lo que resulta una propiedad muy útil para aplicar leyes de control.

$$\dot{M} - 2C$$

Para definir la matriz de Coriolis en términos de álgebra de Lie, vamos a formularla en función de los llamados **Símbolos de Christoffel Γ_{ijk}** que corresponden a la Matriz de Inercia, quedando la expresión "(2-42)" {RobotMan-f(11):**manipulatorcoriolis**}.

$$C_{ij}\left(\theta,\dot{\theta}\right)=\sum_{k=1}^{n}\Gamma_{ijk}\dot{\theta}_k=\frac{1}{2}\sum_{k=1}^{n}\left(\frac{\partial M_{ij}}{\partial\theta_k}+\frac{\partial M_{ik}}{\partial\theta_j}-\frac{\partial M_{kj}}{\partial\theta_i}\right)\dot{\theta}_k \qquad \textbf{(2-42)}$$

Puede que uno se pregunte por la solución explícita matricial, puesto en la expresión "(2-42)" sólo se aprecian derivadas parciales para Γ_{ijk}. En realidad, estos términos quedan resueltos con un desarrollo matricial de Lie, según la expresión "(2-43)".

$$\frac{\partial M_{ij}}{\partial\theta_k}=\sum_{l=\max(i,j)}^{n}\left(\left[A_{ki}\xi_i,\xi_k\right]^T A_{lk}{}^T\mu_l^* A_{lj}\xi_j+\xi_i^T A_{li}{}^T\mu_l^* A_{lk}\left[A_{kj}\xi_j,\xi_k\right]\right) \qquad \textbf{(2-43)}$$

Para lo que se define una **Transformación Adjunta del Manipulador A_{ij}**, como la expresión siguiente "(2-44)" {RobotMan-f(10):**manipulatoradjoint**}, y la **Matriz de Inercia Transformada μ_i^*** "(2-45)" {RobotMan-f(9):**linkinertiatrans**}, que representa la inercia del eslabón i-ésimo en el sistema de referencia de la base. **$g_{sli}(0)$** son los sistemas de referencia de los eslabones y por supuesto **μ_i** es la Matriz de Inercia Generalizada de cada eslabón, tal como se definió en "(2-40)".

$$A_{ij}=\begin{cases}Ad_{\left(e^{\xi_{j+1}\theta_{j+1}}\cdots e^{\xi_i\theta_i}\right)}^{-1} & ; \quad i>j\\ \mathrm{I} & ; \quad i=j\\ 0 & ; \quad i<j\end{cases} \qquad \textbf{(2-44)}$$

$$\mu_i^*=Ad_{g_{sli}^{-1}(0)}{}^T\cdot\mu_i\cdot Ad_{g_{sli}^{-1}(0)} \qquad \textbf{(2-45)}$$

Los atributos dinámicos del robot (i.e., matrices **M** y **C**), pueden determinarse con sólo conocer la geometría de los *twists* **ξ_i** de las articulaciones y los tensores de inercia de los eslabones **Ψ_i** (ver implementación práctica en el Apéndice-D con la librería RobotMan.) Nos quedaría por desarrollar una representación similar para **N**, aunque se suelen despreciar las fricciones y nos queda una matriz gravitacional que es fácil de derivar.

En conclusión, hemos presentado una formulación geométrica para el problema dinámico de un robot, ya que las ecuaciones del movimiento quedan expresadas como productos matriciales (i.e., conjuntos de **POE** que no son sino transformaciones geométricas), con la gran ventaja computacional y de control que ello supone.

2.5 Control de Robots.

Los robots necesitan un sistema de control que permita regular las fuerzas de los actuadores que generan el movimiento del manipulador para seguir una trayectoria de referencia. Para ello, se debe diseñar una ley de control realimentado robusta que corrija las fuerzas aplicadas en respuesta a las desviaciones de la trayectoria deseada.

Existen varios enfoques para diseñar leyes de control estables. Usando las propiedades estructurales de la dinámica de robots, podemos probar la estabilidad de leyes de control generales para todos los robots que tengan esas propiedades. De esta forma se demuestra que la estabilidad de algunos algoritmos de control requiere que las ecuaciones del movimiento cumplan la **propiedad de pasividad** (como es el caso de la formulación presentada en esta tesis 2.4.2). Por supuesto, el rendimiento de un sistema de control depende en gran medida de cada robot, por lo que las leyes que se presentan en este apartado deben ser entendidas como un punto de partida para sintetizar un regulador adecuado a cada caso.

Hay dos caminos básicos en la resolución del problema de control. El primero, es al que nos referiremos por "**Control en el Espacio de las Articulaciones**", que convierte una tarea dada en un camino deseado para las articulaciones del robot, de forma que la ley de control determina los pares necesarios para que las articulaciones sigan esas trayectorias. El segundo enfoque, al que definimos como "**Control en el Espacio de Trabajo**", transforma la dinámica y el control en términos del espacio de la tarea, de forma que la ley de control se escribe en términos de posiciones y orientaciones de la herramienta del robot. Un procedimiento común en el control de robots es el denominado linealización por realimentación, que produce una dinámica en el espacio de las articulaciones lineal y desacoplada. Sin embargo, para aplicaciones tales como el control de la herramienta, resulta también útil desarrollar el mismo tipo de linealización por realimentación pero en las coordenadas del espacio de trabajo.

Si la herramienta del robot está en contacto con el entorno el problema dinámico es más difícil, puesto que debemos controlar no sólo la posición sino también la fuerza que se ejerce. El modelo se complica con la introducción de restricciones, como en el caso que nos ocupará para el control dinámico de humanoides.

2.5.1 Control en el Espacio de las Articulaciones.

La descripción del problema de control comienza con una formulación de la dinámica del robot según la fórmula presentada anteriormente "(2-41)", junto con una trayectoria deseada para las coordenadas generalizadas del robot $\boldsymbol{\theta_d}$ (i.e., $\boldsymbol{\theta_d}$ son los caminos que deben recorrer las articulaciones del robot), que pueden obtenerse por ejemplo resolviendo el problema cinemático inverso (ver 2.3.2). Entonces dada la posición y velocidad presentes del robot, el conjunto de pares necesarios para las articulaciones vendrá dado por "(2-46)".

$$M\left(\theta_d\right)\ddot{\theta}_d + C\left(\theta_d, \dot{\theta}_d\right)\dot{\theta}_d + N\left(\theta_d, \dot{\theta}_d\right) = \Gamma \qquad \textbf{(2-46)}$$

Como necesitamos una ley de control para corregir perturbaciones y errores, propondremos una solución como la "**Ley de Control de Par Computado**", que añade una realimentación de estado a la ecuación "(2-46)" para obtener la ecuación de control "(2-47)", donde **e** es el error de posición de las articulaciones (i.e., $\mathbf{e=\theta-\theta_d}$), $\mathbf{K_v}$ es la **matriz de ganancia constante de velocidad** y $\mathbf{K_p}$ la **matriz de ganancia constante de posición**. Es un ejemplo de linealización por realimentación, permitiendo el uso de las numerosas herramientas para síntesis de controles lineales. Existen muchos otros tipos de reguladores (e.g., **PD**), pero la Ley de Control de Par Computado, aunque pueda ser una técnica computacionalmente costosa, ha mostrado resultados experimentales con gran rendimiento en muchos trabajos de investigación.

$$M(\theta)\left(\ddot{\theta}_d - K_v\dot{e} - K_p e\right) + C\left(\theta,\dot{\theta}\right)\dot{\theta} + N\left(\theta,\dot{\theta}\right) = \Gamma \qquad \textbf{(2-47)}$$

2.5.2 Control en el Espacio de Trabajo.

Existen varias dificultades para el control en el espacio de las articulaciones, como son el tiempo de computación excesivo para resolver la cinemática inversa o la difícil elección de las constantes de realimentación, debido a que la tarea está definida en términos de la trayectoria de la herramienta $\mathbf{x_d}$ y no en términos de las trayectorias de las articulaciones. Es por esto, por lo que para muchas aplicaciones nos interesa definir las ecuaciones dinámicas del movimiento en función del espacio de trabajo del robot, quedando formuladas como "(2-48)". Donde los **Parámetros Efectivos del sistema** quedan definidos por las ecuaciones "(2-49)", en función de los parámetros dinámicos (i.e., **M** "(2-39)", **C** "(2-42)" y **N** "(2-41)") y el Jacobiano **J**, que no es el Manipulador Jacobiano que hemos utilizado hasta ahora.

$$\tilde{M}(\theta)\ddot{x} + \tilde{C}\left(\theta,\dot{\theta}\right)\dot{x} + \tilde{N}\left(\theta,\dot{\theta}\right) = \tilde{\Gamma} \qquad \textbf{(2-48)}$$

$$\left.\begin{aligned} \tilde{M} &= J^{-T} M J^{-1} \\ \tilde{C} &= J^{-T}\left(CJ^{-1} + M\frac{d}{dt}\left(J^{-1}\right)\right) \\ \tilde{N} &= J^{-T} N \\ \tilde{\Gamma} &= J^{-T}\Gamma \end{aligned}\right\} \qquad \textbf{(2-49)}$$

Una vez definido así el sistema, podemos aplicar también una ley de control para corregir perturbaciones y errores, como la Ley de Control de Par Computado, para obtener la ecuación de control "(2-50)", donde **e** es el error de posición de la trayectoria de la herramienta en el espacio de trabajo (i.e., $\mathbf{e=x-x_d}$), $\mathbf{K_v}$ es la **matriz de ganancia constante de velocidad** y $\mathbf{K_p}$ la **matriz de ganancia constante de posición**. La ventaja de esta formulación es que la elección de las ganancias (i.e., $\mathbf{K_v}$ y $\mathbf{K_p}$) para una correcta resolución de la tarea, es mucho más sencilla y además, no es necesario resolver la cinemática inversa a cada paso. La desventaja, por supuesto, es que necesitamos ir resolviendo el Jacobiano en cada momento.

$$\tilde{M}(\theta)\left(\ddot{x}_d - K_v\dot{e} - K_p e\right) + \tilde{C}\left(\theta,\dot{\theta}\right)\dot{x} + \tilde{N}\left(\theta,\dot{\theta}\right) = \tilde{\Gamma} \qquad \textbf{(2-50)}$$

2.6 Análisis de Robots Humanoides.

Para el análisis del control de un robot humanoide, en el apartado (2.5.2) hemos tratado el control de una cadena sólidos rígidos, pero debemos considerar otros aspectos que presentamos a continuación:

- **Acomodación**: existe siempre cierto nivel de acomodación en el conjunto formado por el robot (e.g., flexibilidad de los eslabones), el entorno (e.g., superficies de soporte blandas) y el controlador. Estos fenómenos pueden ser negativos para el control (e.g., reducción del ancho de banda, resonancia), pero también aportan ventajas (e.g., acomodan errores de posición, suavizan los impactos en los apoyos de los pies.) Muchas veces este aspecto se ignora en los proyectos debido a la falta de datos, pero puede ser muy importante para la estabilidad global y rendimiento del sistema.
- **Controladores**: se clasifican fundamentalmente en función de la causa que determina el movimiento de la máquina:
    - **Controladores Cinemáticos:** para ellos, el movimiento del tronco del humanoide es causado por el movimiento de las extremidades. Las posiciones de referencia de las extremidades se calculan y planifican para producir el deseado movimiento del tronco, resolviendo el problema cinemático inverso. Otro enfoque utiliza la cinemática inversa para garantizar que el **ZMP** se encuentra en el área de soporte.
    - **Controladores Dinámicos:** El movimiento del tronco del humanoide es causado por las fuerzas ejercidas sobre él por las extremidades.
- **Coordinación:** es el funcionamiento armónico de las diferentes partes del humanoide para conseguir resultados efectivos de movimiento. Necesitamos abordar la coordinación desde distintos puntos de vista:
    - **Coordinación del Tronco:** la necesaria con respecto a las extremidades, para seguir caminos evitando obstáculos. El movimiento del tronco afecta al movimiento de las extremidades y viceversa.
    - **Coordinación Intra-Miembros**: la necesaria entre las diversas articulaciones de una misma extremidad.
    - **Coordinación Inter-Miembros:** la necesaria entre diferentes extremidades para la ejecución de movimientos.
    - **Coordinación de las piernas:** la necesaria para alcanzar correctos apoyos de los pies, sin rebotes ni deslizamientos. Existen requisitos temporales y espaciales para el posicionado de los pies.
- **Deslizamiento:** provoca efectos negativos (no siempre) como son: movimientos no deseados, estrés mecánico o dificultades de navegación.
- **Modelado:** la forma de modelar las diferentes partes del sistema tiene una influencia decisiva en la simulación y control. En especial:

  - **Modelo del Entorno:** se refiere sobre todo a la superficie de apoyo, cuya geometría se considera (normalmente) como un plano. También es común asumir que el suelo aplica fuerzas sobre los pies. El suelo se puede considerar rígido, aunque es más real pensar que las fuerzas que aplica están en función de una posición y velocidad de penetración, esto es, que el modelo de suelo es el de un amortiguador.
  - **Modelo de los Actuadores:** se puede ir desde la consideración de actuadores que son ideales generadores de par, hasta modelos mucho más complejos y reales que tienen en cuenta la dinámica del motor, la viscosidad o la influencia de la temperatura.
  - **Modelo de los Sensores:** los robots tienen sensores para medir los ángulos de las articulaciones, amperímetros para la corriente de los motores y sensores de orientación (e.g., inclinómetros, giroscopios, acelerómetros.) El modelado y utilización de todos ellos es una tarea compleja en la que se usan todo tipo de herramientas matemáticas (e.g., filtros de Kalman, en el trabajo de Zhou [133] de este mismo año).

- **Navegación:** la planificación de trayectorias libres de colisiones debe ser parte integrante del control del robot humanoide.
- **Objetivos:** deben ser tenidos en cuenta para valorar el rendimiento. Algunas veces los objetivos son contradictorios (e.g., realizar un movimiento de la forma más rápida posible y de la forma más segura posible.) Los objetivos más frecuentes son: seguridad, rendimiento, tolerancia y cumplimiento de misiones.
- **Reflejos:** algunos sistemas utilizan el control reactivo de comportamiento (i.e., los reflejos) para mejorar los resultados. Normalmente se implementan como controles de bajo nivel que desencadenan ciertas acciones "reflejas" como respuesta a determinados eventos (e.g., levantar la pierna si el pie choca con un obstáculo.) La utilización de controles reflejos incrementa la robustez global.

La implementación técnica del control de humanoides se basa en variados desarrollos de regulación automática (e.g., control proporcional (**P**), proporcional-integral (**PI**), proporcional-integral-derivativo (**PID**)), pero en resumen los podríamos clasificar en:

- **Control de Posición:** denota normalmente cualquier método de seguimiento de una posición o trayectoria de referencia. En algunos casos, se incluye también el seguimiento de velocidades.
- **Control de Fuerza:** es el usado cuando la interacción del robot con el entorno es importante, esto es, cuando las herramientas del robot en la realización de una tarea reciben reacciones por parte del entorno, que hacen necesario un control con realimentación sensorial de fuerza.
- **Control de Impedancia:** utilizado para regular la relación entre la velocidad y la fuerza, en otras palabras, la impedancia mecánica del robot. Es una generalización de control de rigidez y amortiguamiento.

Los conceptos aquí mencionados que se encuentran dentro del alcance de esta tesis, se abordarán desde la perspectiva de Geometría Diferencial introducida en este capítulo (2) mediante el uso de los Grupos y Álgebras de Lie.

2.7 El POE del Álgebra de Lie vs los Parámetros de Denavit-Hartenberg.

El tratamiento de la cinemática que proponemos en esta tesis se basa en el **Álgebra** de **Lie** y es en cierto modo una desviación del estándar de facto usado en la mayoría de los trabajos de robótica, que utilizan la formulación de **D-H**. Una ventaja fundamental del formalismo del **POE** de **Lie** es que proporciona una formulación elegante de varios problemas canónicos (ver 2.3.3 y 2.3.4) para resolver de forma geométrica y cerrada complejos problemas de cinemática inversa, como probamos con los ejemplos del apéndice A. Por el contrario, con los parámetros de **D-H** resulta muy difícil encontrar soluciones cerradas para problemas cinemáticas inversos con más de 3 **GDL** (salvo simplificaciones como el desacoplo cinemático), por lo que se debe recurrir a soluciones numéricas de los sistemas de ecuaciones involucrados, que son implementaciones muy lentas y con desventajas provocadas por la necesidad de computar matrices mal condicionadas. Otra ventaja importante del **POE** es la descripción realmente geométrica del movimiento del sólido rígido, que simplifica enormemente el análisis de los mecanismos.

En la mayor parte de los problemas de robótica es mucho más fácil construir la cinemática basándose en las técnicas de **Lie** porque sólo son necesarios dos sistemas de referencia, estos son, el de la base y el del eslabón cinemático de interés, que normalmente es el sistema de referencia de la herramienta. Por el contrario, con **D-H** debemos construir los sistemas de referencia de todos los eslabones de las cadenas cinemáticas del mecanismo.

Una ventaja más del **POE** es la facilidad para obtener el manipulador Jacobiano y las velocidades del robot sin necesidad de calcular derivadas, sino con una caracterización geométrica que permite describir las singularidades de una forma sencilla. Además, la representación de la cinemática del sólido rígido basada en la teoría de **Lie** no sufre de singularidades provocadas por el uso de coordenadas locales, cosa que es inevitable si usamos algún otro tipo de representación para las rotaciones (e.g. ángulos de Euler). La descripción tradicional del Jacobiano 2.1.5, que es la utilizada por **D-H**, se obtiene por diferenciación del mapa cinemático directo, pero resulta que esta magnitud no es natural, mientras que el **POE** conduce a una representación explícita del manipulador Jacobiano que es realmente natural y que resalta las propiedades geométricas del mecanismo, sin tener las desventajas de las representaciones locales.

También hay que destacar que para el planteamiento y resolución de los problemas dinámicos, el **Álgebra** de **Lie** nos proporciona herramientas para un cálculo sistemático geométrico, como hemos visto con las funciones descritas en 2.4 y contenidas en la librería de software (Apéndice D) creada en esta tesis.

Por todo lo explicado anteriormente, en general (aunque existen casos particulares), el formalismo del **POE** de **Lie** es una alternativa superior al de los **Parámetros** de **D-H**.

Para un mejor entendimiento de las diferencias entre la técnica de **Lie** y la de **D-H**, vamos a desarrollar un ejemplo detallado en los siguientes apartados para un robot **Polar** de **3 GDL**, con resultados numéricos que comparan los algoritmos basados en las dos formulaciones matemáticas. En la comparación práctica entre los formalismos

matemáticos, hallaremos las soluciones cinemáticas y dinámicas de un robot sencillo de tres GDL de tipo Polar, que tiene dos articulaciones de rotación (**θ1**, **θ2**) y una de traslación **θ3**. Se puede ver el robot Polar en el gráfico de la Figura 2-6, junto con un esquema cinemático para mayor claridad de análisis.

2.7.1 Solución de Lie para un Robot Polar de 3 GDL.

En primer lugar veremos que utilizando el **POE** de álgebras de Lie es muy sencillo resolver tanto la cinemática directa como inversa, con tan sólo conocer el esquema cinemática del robot polar, esto es, los ejes de actuación de los **GDL**, la posición inicial de la herramienta **H** y el sistema de referencia inercial **S**. El problema cinemático inverso requiere utilizar los problemas canónicos (ver 2.3.3) y un poco de cálculo algebraico. El método puede parecer un poco confuso la primera vez que se utiliza, pero tras unos ejemplos (ver el Apéndice A para descripciones mucho más detalladas), se aprecia la sencillez, potencia y elegancia de este planteamiento matemático.

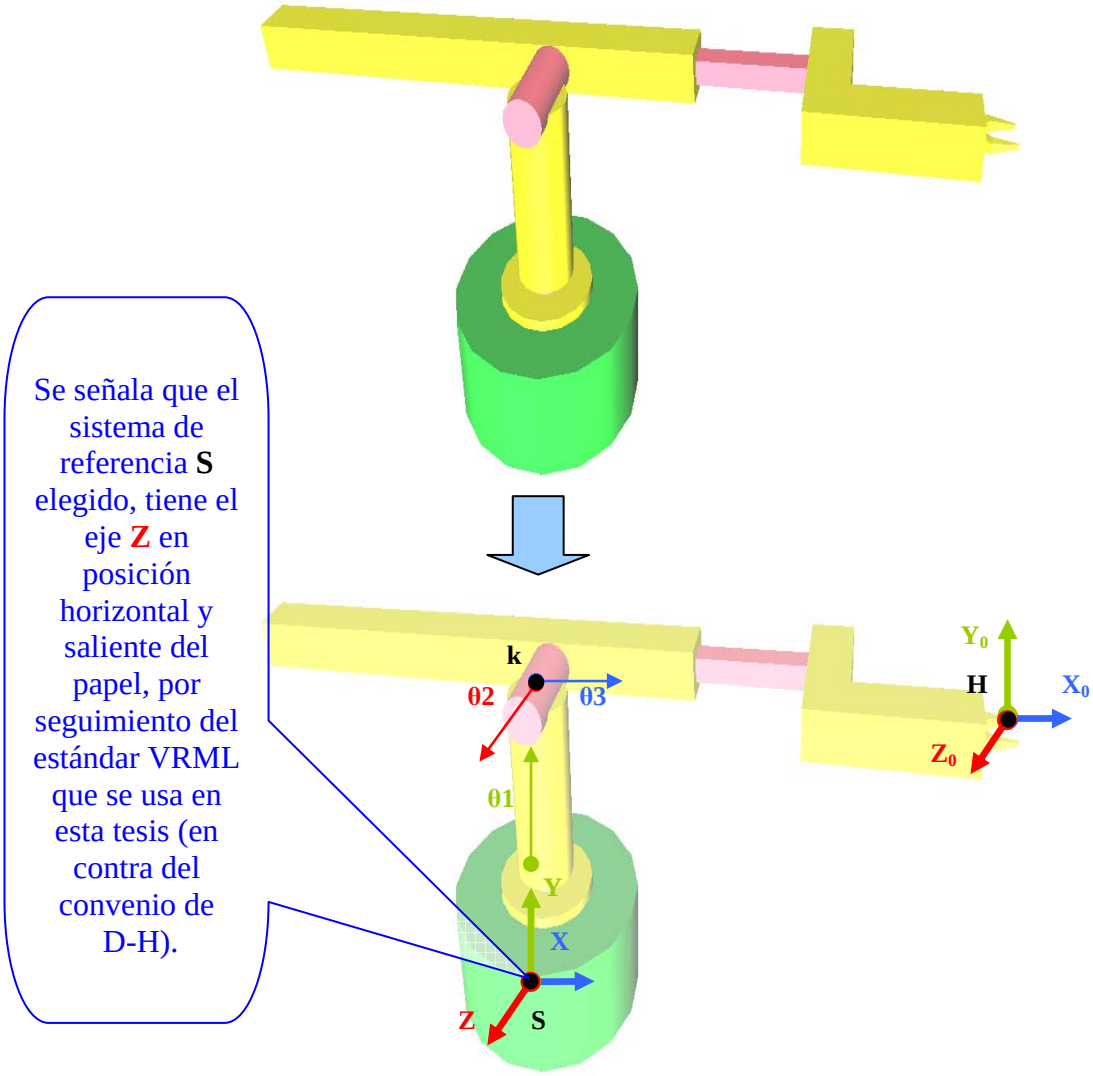


Figura 2-6: Esquema Cinemático del Robot POLAR (3 GDL).

Los ejes de aplicación de los tres **GDL** vienen dados por "(2-51)".

$$\omega_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}; \omega_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; \upsilon_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \textbf{(2-51)}$$

Entonces, los valores de los *twists* ξ para cada una de las articulaciones quedan formulados de una forma sencilla por "(2-52)". {RobotMan-f(18):**prismatictwist**} para la de translación y {RobotMan-f(19):**revolute*twist***} para las de revolución.

$$\xi_1 = \begin{bmatrix} \upsilon_1 \\ \omega_1 \end{bmatrix} = \begin{bmatrix} -\omega_1 \times k \\ \omega_1 \end{bmatrix}; \xi_2 = \begin{bmatrix} \upsilon_2 \\ \omega_2 \end{bmatrix} = \begin{bmatrix} -\omega_2 \times k \\ \omega_2 \end{bmatrix}; \xi_3 = \begin{bmatrix} \upsilon_3 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} \upsilon_3 \\ 0 \end{bmatrix} \qquad \textbf{(2-52)}$$

Siendo la fórmula para la transformación entre **H** y **S** en la configuración de referencia del manipulador $\mathbf{g_{sh}(0)}$, muy sencilla de definir "(2-53)".

$$g_{sh}(0) = \begin{bmatrix} 1 & 0 & 0 & H-k \\ 0 & 1 & 0 & k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{(2-53)}$$

El problema cinemático directo se soluciona con la aplicación del **POE "**(2-54)" {RobotMan-f(3):**forwardkinematics**}, que nos da el valor de $\mathbf{g_{sh}(\theta)}$. Nótese que el tratamiento matemático de la articulación de translación $\mathbf{\theta_3}$ es el mismo que el de las articulaciones de rotación.

$$g_{sh}(\theta) = e^{\xi_1^{\wedge}\theta_1} \cdot e^{\xi_2^{\wedge}\theta_2} \cdot e^{\xi_3^{\wedge}\theta_3} \cdot g_{sh}(0) \qquad \textbf{(2-54)}$$

Para resolver el problema cinemático inverso mediante álgebra de Lie, aplicamos los problemas canónicos (ver 2.3.3) de una forma sistemática, operando sobre la ecuación "(2-54)" con dos pasos sencillos.

Primer paso - Obtención de la variable $\mathbf{\theta_3}$: Para ello, pasamos $\mathbf{g_{sh}(0)}$ al otro lado de la ecuación "(2-54)" y aplicamos ambos lados de esta ecuación al punto **H**, para posteriormente hallar el módulo de la diferencia de esos términos con el punto **k**, con lo que obtendremos la ecuación "(2-55)".

$$\left\| g_{sh}(\theta) \cdot g_{sh}(0)^{-1} \cdot H - k \right\| = \left\| e^{\xi_1^{\wedge}\theta_1} \cdot e^{\xi_2^{\wedge}\theta_2} \cdot e^{\xi_3^{\wedge}\theta_3} \cdot H - k \right\| \qquad \textbf{(2-55)}$$

Hemos operado de este modo para obtener la ecuación "(2-55)" en la que las exponenciales de las variables $\mathbf{\theta_1}$, y $\mathbf{\theta_2}$ no tienen ningún efecto (ver el Apéndice A para una justificación detallada) y se pueden eliminar. Por todo ello, la ecuación "(2-55)" tiene el término izquierdo conocido (valor **δ**) y un término derecho que sólo se ve afectado por el tercer **GDL**, por lo que queda transformada en la ecuación "(2-56)", que no es sino la formulación del **problema canónico Pardos-Uno**, por lo que obtenemos directamente los dos valores posibles para $\mathbf{\theta_3}$ con la función "(2-35)".

$$\delta = \left\| e^{\xi_3^{\wedge}\theta_3} \cdot H - k \right\| \xrightarrow{Pardos\ -UNO} \theta_3 \quad Doble \qquad \textbf{(2-56)}$$

Segundo paso – Obtención de las variables θ_1, y θ_2: Para ello pasamos $\mathbf{g_{sh}(0)}$ al otro lado de la ecuación "(2-54)" y aplicamos ambos lados de esta ecuación al punto **H**, para obtener la ecuación "(2-57)", que tiene el término izquierdo conocido (igual a un punto cualquiera que llamaremos k') y un término derecho con el producto de la exponencial de $\boldsymbol{\theta_3}$ por el punto **H**, que será también un valor conocido (igual a un punto que llamaremos p'), al haber obtenido $\boldsymbol{\theta_3}$ en el primer paso. De esta forma, la ecuación "(2-57)" queda transformada en "(2-58)", que no es sino la formulación del **problema canónico PadenKahan-Dos**, por lo que por aplicación geométrica de éste "(2-31)" obtenemos los dos valores posibles para la pareja de variables $\boldsymbol{\theta_2}$ y $\boldsymbol{\theta_1}$.

$$g_{sh}(\theta)\cdot g_{sh}(0)^{-1}\cdot H = e^{\xi_1^{\wedge}\theta_1}\cdot e^{\xi_2^{\wedge}\theta_2}\cdot e^{\xi_3^{\wedge}\theta_3}\cdot H \qquad \textbf{(2-57)}$$

$$k' = e^{\xi_1^{\wedge}\theta_1}\cdot e^{\xi_2^{\wedge}\theta_2}\cdot p' \xrightarrow{PadenKahan-DOS} \theta_1,\theta_2 \quad Doble \qquad \textbf{(2-58)}$$

De esta forma queda resuelto de forma geométrica, determinista, cerrada y completa el problema cinemático inverso. Lo que es más, no sólo se ha encontrado una solución para el problema (i.e., un conjunto de valores $\boldsymbol{\theta_1}$, $\boldsymbol{\theta_2}$, $\boldsymbol{\theta_3}$), sino que de existir, se han resuelto las cuatro posibles soluciones en una sola formulación. Obsérvese para ello las combinaciones de las posibles soluciones en "(2-59)".

$$N\_Soluciones = \theta_3 Doble \times \theta_2\theta_1 Doble = 4 \qquad \textbf{(2-59)}$$

Una vez resuelta la cinemática inversa de una forma geométrica, avanzaremos un poco más en el análisis del robot Polar, estudiando las velocidades de la herramienta y su relación con la velocidades de las articulaciones, que viene dada por el Manipulador Jacobiano que se introdujo en 2.1.5, y que en este caso está formulada según "(2-60)". El Manipulador Jacobiano presenta un verdadero significado geométrico natural, puesto que cada columna que lo compone corresponde a un *twist* transformado por los anteriores, esto es, podemos calcularlo por inspección "(2-61)" {RobotMan-f(33):**spatialjacobian**} ¡sin necesidad de efectuar ninguna derivada!.

$$V^s = J^s(\theta)\dot{\theta} \qquad \textbf{(2-60)}$$

$$J^s(\theta) = \begin{bmatrix}\xi_1' & \xi_2' & \xi_3'\end{bmatrix} \quad \wedge \quad \xi_i' = Ad_{\left(\prod_{i=1}^{i-1} e^{\xi_i^{\wedge}\theta_i}\right)}\xi_i \qquad \textbf{(2-61)}$$

En cuanto a la dinámica del robot Polar, podemos sustituir en las ecuaciones de Lagrange "(2-37)" para llegar a la formulación para las ecuaciones de movimiento del robot "(2-62)", que hemos explicado en 2.4. La matemática de grupos de Lie nos va a permitir expresar de forma explícita la matriz de Coriolis, que es quizá el término más complicado de obtener, de modo que el vector de pares o fuerzas **Γ** que es solución al problema dinámico inverso quedará definido mediante un conjunto de operaciones matriciales que son básicamente **POE**.

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + N(\theta,\dot{\theta}) = \Gamma \qquad \textbf{(2-62)}$$

2.7.2 Solución de D-H para un Robot Polar de 3 GDL.

Analizamos el mismo robot Polar que en el apartado anterior (ver Figura 2-6), pero ahora utilizando los Parámetros de Denavit-Hartenberg (**D-H**). Por no ser objeto de esta tesis y estar ampliamente difundido en la literatura (ver por ejemplo [14]), no vamos a explicar aquí el detalle de los dieciséis pasos que son necesarios según el algoritmo de **D-H** para obtener los parámetros correspondientes a un robot que permiten resolver su problema cinemático directo, pero cualquiera que conozca la técnica convendrá en que resulta mucho más trabajosa, oscura y con menos sentido físico que la presentada anteriormente en este mismo apartado "(2-54)" mediante técnicas de **Lie**. Siguiendo el procedimiento habitual de **D-H**, con la particularidad de que en nuestro ejemplo realizamos un cambio de ejes (i.e. el Z por el Y) para adaptar el algoritmo de **D-H** a la definición estándar de ejes adoptada en esta tesis (ver Figura 2-6), que es la de objetos representados con realidad virtual, obtenemos la tabla de Parámetros de **D-H** para el robot Polar, que podemos ver en la Figura 2-7.

Estos parámetros de **D-H** son cuatro para cada articulación, dependen de las características geométricas de cada eslabón y de las articulaciones que le unen con el anterior y el siguiente. Su significado físico es:

- $\boldsymbol{\theta_j}$: Es el ángulo que forman los ejes $\mathbf{x_{j-1}}$ y $\mathbf{x_j}$ medido en un plano perpendicular al eje $\mathbf{y_{j-1}}$ utilizando la regla de la mano derecha, esto es, la rotación alrededor del eje $\mathbf{y_{j-1}}$.
- $\mathbf{d_j}$: Es la distancia a lo largo del eje $\mathbf{y_{j-1}}$ desde el origen del sistema de coordenadas anterior hasta la intersección del eje $\mathbf{y_{j-1}}$ con el $\mathbf{x_j}$, esto es, la traslación a lo largo de $\mathbf{y_{j-1}}$.
- $\mathbf{a_j}$: Es la distancia a lo largo del eje $\mathbf{x_j}$ que va desde la intersección del eje $\mathbf{y_{j-1}}$ con el eje $\mathbf{x_j}$ hasta el origen del sistema de coordenadas siguiente, en el caso de las articulaciones giratorias. En el caso de las articulaciones prismáticas, se calcula como la distancia más corta entre los ejes $\mathbf{y_{j-1}}$ con el $\mathbf{y_j}$, esto es, la traslación a lo largo del eje $\mathbf{x_j}$.
- $\boldsymbol{\alpha_j}$: Es el ángulo de separación del eje $\mathbf{y_{j-1}}$ y el eje $\mathbf{y_j}$ medido en un plano perpendicular al eje $\mathbf{x_j}$ utilizando la regla de la mano derecha, esto es, la rotación alrededor del eje $\mathbf{x_j}$.

Una vez obtenidos esos parámetros, podemos calcular la matriz de transformación homogénea $\mathbf{A_{j,j+1}}$ que relaciona el sistema de referencia de un eslabón de la cadena cinemática con el siguiente, según la ecuación "(2-63)".

Articulación - j	$\boldsymbol{\theta_j}$	$\mathbf{d_j}$	$\mathbf{a_j}$	$\boldsymbol{\alpha_j}$
1	θ_1	k - S	0	90º
2	θ_2 - 90º	0	0	-90º
3	0	θ_3 + H - k	0	0

Figura 2-7: Parámetros de D-H para el Robot tipo POLAR.

$$A_{j,j+1}=\begin{bmatrix}\cos\theta_j & 0 & \sin\theta_j & 0\\ 0 & 1 & 0 & 0\\ -\sin\theta_j & 0 & \cos\theta_j & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & d_j\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}1 & 0 & 0 & a_j\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha_j & -\sin\alpha_j & 0\\ 0 & \sin\alpha_j & \cos\alpha_j & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \quad \textbf{(2-63)}$$

Siendo $\mathbf{g_{sh}(0)}$, en este caso formulada según "(2-64)", la transformación homogénea que consigue girar el sistema de referencia de la herramienta **H** dado por el formalismo de **D-H** (i.e. en nuestro caso tiene el eje Y sobre el eje de articulación), de forma que el sistema de referencia de **H** queda como en la Figura 2-6, para que los resultados numéricos de posición y rotación sean comparables con los obtenidos por el formalismo de **Lie**.

$$g_{sh}(0)=\begin{bmatrix}\cos\frac{\pi}{2} & -\sin\frac{\pi}{2} & 0 & 0\\ \sin\frac{\pi}{2} & \cos\frac{\pi}{2} & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \quad \textbf{(2-64)}$$

Como resultado de las anteriores propuestas, el problema cinemático directo del robot polar queda formulado usando el formalismo de **D-H** como se indica en "(2-65)", donde $\mathbf{g_{sh}(\theta)}$ es la matriz de transformación homogénea que relaciona el sistema de referencia de la base **S** con el de la herramienta **H**, tal y como aparece en la Figura 2-6.

$$g_{sh}(\theta)=A_{0,1}\cdot A_{1,2}\cdot A_{2,3}\cdot g_{sh}(0) \quad \textbf{(2-65)}$$

Conocido el objetivo de la herramienta $\mathbf{g_{sh}(\theta)}$, para resolver el problema cinemático inverso del robot Polar mediante el uso del formalismo de **D-H**, tenemos que trabajar el sistema representado en "(2-65)" de doce ecuaciones con tres incógnitas para hallar las cuatro posibles soluciones. Los procedimientos utilizados para resolver este sistema se basan en conocidas técnicas numéricas para la resolución de sistemas de ecuaciones. En la literatura, sin embargo, tras un desarrollo matemático no precisamente breve [14] podemos extraer las soluciones geométricas directas del problema cinemático inverso de este robot "(2-66)".

$$\begin{aligned}\theta_1 &= A\tan2\left(-g_{sh}(3,4),g_{sh}(1,4)\right)\\ \theta_2 &= A\tan2\left(\sqrt{g_{sh}(3,4)^2+g_{sh}(1,4)^2},k-S-g_{sh}(2,4)\right)\\ \theta_3 &= \sin\theta_2\cdot\sqrt{g_{sh}(3,4)^2+g_{sh}(1,4)^2}-\cos\theta_2\cdot\left(g_{sh}(2,4)-k+S\right)-H+k\end{aligned} \quad \textbf{(2-66)}$$

Hemos elegido el robot Polar como ilustración de este capítulo porque al tener sólo 3 **GDL** permite obtener sus soluciones geométricas de una forma analítica, que podremos comparar de alguna forma con las soluciones sistemáticas de la formulación

de **Lie**. Sin embargo, esta comparación se hace casi imposible para robots con un número de **GDL** más elevado, puesto que mientras podremos resolver su cinemática inversa con **Lie**, resultan inaplicables en términos prácticos los parámetros de **D-H**.

Una vez resuelta la cinemática inversa, analizamos las velocidades de la herramienta y su relación con las velocidades de las articulaciones del robot Polar "(2-67)". Aquí se deben calcular las derivadas para resolver el Jacobiano "(2-68)", aunque no existe relación geométrica entre este Jacobiano y la realidad física del robot. Se supone que ya se conocen previamente las ecuaciones que resuelven el problema cinemático directo.

$$V^s = J(\theta)\dot{\theta} \qquad \textbf{(2-67)}$$

$$J(\theta) = \begin{bmatrix} \frac{\partial f_{dx}(\theta_1,\theta_2,\theta_3)}{\partial \theta_1} & \cdots & \frac{\partial f_{dx}(\theta_1,\theta_2,\theta_3)}{\partial \theta_3} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{Rz}(\theta_1,\theta_2,\theta_3)}{\partial \theta_1} & \cdots & \frac{\partial f_{Rz}(\theta_1,\theta_2,\theta_3)}{\partial \theta_3} \end{bmatrix} \neg \begin{cases} d \in g_{sh}(\theta) \mapsto (dx, dy, dz) \\ R \in g_{sh}(\theta) \mapsto (Rx, Ry, Rz) \end{cases} \qquad \textbf{(2-68)}$$

En cuanto a la dinámica del robot Polar, podemos plantear la solución de Lagrange para la formulación para las ecuaciones de movimiento del robot "(2-69)", pero hay que tener en cuenta que aquí no tenemos herramientas para una resolución matricial, sino que tenemos que aplicar un análisis convencional y complejo de la ecuación.

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + N(\theta,\dot{\theta}) = \Gamma \qquad \textbf{(2-69)}$$

2.7.3 Comparación Computacional de Lie vs D-H para Robot POLAR de 3 GDL.

Una vez resuelto el problema del robot Polar utilizando los formalismos del **POE** de **Lie** y los Parámetros de **D-H**, vamos a presentar en la Figura 2-8 una tabla comparativa de las implementaciones prácticas de estas soluciones en un computador. Los valores expresados son medias de tiempo, tras realizar una serie de mil ejecuciones.

Formalismo Utilizado para Robot POLAR	Problema Cinemático Directo	Problema Cinemático Inverso	Jacobiano y Velocidades	Problema Dinámico
POE de Lie	**0,4ms**	**2,5ms**	Solución SIN Derivar	Solución Sistemática
Parámetros de D-H	**0,2ms**	**25ms**	Es necesario Derivar	Solución no Sistemática

Figura 2-8: Tabla Comparativa POE de Lie vs Parámetros de D-H para el Robot POLAR.

Es necesario analizar con sumo cuidado el resultado de la comparativa que se resume en la Figura 2-8 puesto que es difícil extraer una conclusión contundente, ya que si el **POE** de **Lie** presenta una solución más elegante, mucho más veloz para resolver el problema cinemático inverso, sin necesidad de derivadas para el Jacobiano y con un cálculo sistemático de la dinámica, los **Parámetros** de **D-H** proporcionan soluciones más rápidas para el problema cinemático directo.

En defensa de la utilización del formalismo de **Lie** para la resolución del problema cinemático directo, podemos decir que el **POE** presenta matrices recurrentes que hacen que los algoritmos sean más rápidos conforme crece el número de **GDL**. No obstante, en este ejemplo también queda justificada de forma explícita la razón por la que para muchos robots (i.e. especialmente robots sencillos con pocos GDL) los parámetros de **D-H** siguen siendo un estándar muy utilizado en robótica, sobre todo para la resolución de problemas cinemáticos directos.

Para robots con un número reducido de **GDL** la elección más eficaz del formalismo a utilizar depende de la aplicación que vayamos a desarrollar. Es para ilustrar esta competencia entre los formalismos de **Lie** y **D-H** en robots de dos o tres **GDL**, por lo que elegimos precisamente este ejemplo del robot Polar.

Un caso muy diferente es el que se produce con mecanismos más complejos, como sucede en los ejemplos del Apéndice A con robots de seis **GDL**, para los que la resolución del problema cinemático directo con el **POE** de **Lie** se iguala en velocidad a los **Parámetros** de **D-H**, mientras que el problema cinemático inverso que con técnicas de **Lie** consigue resolver el sistema de doce ecuaciones con seis incógnitas, obteniendo las ocho posibles soluciones de una forma cerrada, sistemática y eficaz, tiene una solución de **D-H** muchísimo más compleja y lenta. De hecho, en la literatura se encuentran muy pocas soluciones cerradas de **D-H** para ese tipo de problemas y la mayor parte de éstas son aproximaciones, puesto que se basan en simplificaciones tales como utilizar el desacoplo cinemático.

Para ilustrar la comparación entre el **POE** de **Lie** con los **Parámetros** de **D-H**, haremos un símil utilizando la fórmula de Euler "(2-70)". Mientras que es posible trabajar con aritmética de cosenos y senos para manejar números complejos, resulta evidente que utilizar la expresión canónica exponencial nos permite acceder a herramientas de cálculo mucho más potentes a la hora de manipular complejos. Así de forma semejante, el uso de la matemática de **Lie** y su expresión canónica del **POE**, nos permite abordar problemas más complejos de cadenas de sólidos rígidos.

$$e^{ix} = \cos x + i \cdot senx \qquad \textbf{(2-70)}$$

En conclusión, la utilización de las herramientas matemáticas procedentes de los **Grupos y Álgebras de Lie**, como el formalismo canónico del **POE**, nos permite abordar la creación de algoritmos para el control cinemático y dinámico de robots complejos (i.e. con muchos GDL), con muchas ventajas frente al uso de los parámetros de **D-H**. Por todo ello, adoptamos la matemática de **Lie** en los nuevos desarrollos que se presentan en esta tesis para el Robot Humanoide **RH0** que tiene 21 **GDL**.

3 Generación de Movimientos y Locomoción para Robots Humanoides.

*En este capítulo se desarrolla la mecánica de la Locomoción Bípeda para robots humanoides. Se presenta el nuevo modelo mecánico "**División Cinemática Sagital**" (**DCS**), que permite obtener soluciones cerradas del problema cinemático inverso completo del humanoide. Se formaliza el nuevo algoritmo "**Un Paso Adelante**" (**UPA**) que resuelve el problema de la Locomoción Bípeda del de forma genérica. Se aplica el nuevo modelo mecánico **DCS** para el **RH0**, tanto en su versión cinemática como dinámica. Se formaliza y aplica el nuevo algoritmo **UPA** para el **RHO**. Se presentan esquemas geométricos de los pasos básicos del **RHO** (i.e.: salida, zancada, entrada, giro), introduciendo nuevas trayectorias naturales tanto para el centro de masas como para los pies. Estos esquemas geométricos constituyen un grupo canónico de soluciones, a partir de los que se puede construir por conjugación de los mismos, movimientos realmente complejos para el robots humanoides.*

3.1 Mecánica de la Locomoción Bípeda.

Las ecuaciones del movimiento para la locomoción bípeda de un humanoide son muy complejas, puesto que a las dificultades expuestas en apartados anteriores para un mecanismo con topología en árbol de múltiples grados de libertad, tenemos que añadir las propias del movimiento de la base libre del mecanismo (i.e., el tronco del humanoide), las que se derivan del mantenimiento de la estabilidad necesaria durante la locomoción y las dificultades que introduce el contacto de los pies con el suelo, que producen cadenas cinemáticas cerradas (i.e., restricciones.) El resultado es un sistema algebraico diferencial variable en el tiempo.

Para que el humanoide pueda caminar es necesario desarrollar un modelo mecánico de locomoción, esto es, un ***patrón de locomoción*** que el robot bípedo pueda seguir de una forma estable. Existen muchas formas diferentes de caminar, en función de la velocidad, los cambios de dirección, los modos de iniciar y finalizar el movimiento, las clases de superficie y los tipos de reacción del suelo. No obstante, si la acción de locomoción no cambia, el movimiento se hace periódico, con un desplazamiento rítmico de las distintas partes del cuerpo. Aunque en numerosos casos, el concepto de periodicidad que se utiliza tiene un significado de repetición de ciertas características del movimiento, más que un concepto de función matemática periódica. Para este tipo de movimientos periódicos, el ciclo de cada paso queda descrito, de forma clásica, por dos etapas principales que se alternan en cada pierna, a saber:

- **Etapa de Soporte o de Postura**: Esta etapa existe cuando los dos pies están en contacto con el suelo, de modo que la base de soporte es más grande, puesto que se produce un ***doble apoyo*** que genera un área de soporte dada por la envolvente del polígono formado por el doble apoyo de los dos pies, como podemos ver en el ejemplo de la parte izquierda de la Figura 3-1.
- **Etapa de Transferencia o de Balanceo**: Esta etapa existe cuando sólo un pie está en contacto con el suelo, mientras que el otro pie se encuentra en vuelo, por lo que se dispone sólo de un ***apoyo simple*** formado por el área de soporte del pie de apoyo, como podemos ver en la parte derecha de la Figura 3-1. El **CM** del robot rota alrededor del pie que se encuentra apoyado, de una forma similar a la de un péndulo invertido. En esta etapa de balanceo, se producen dos momentos críticos por sus implicaciones dinámicas en el control de la locomoción, que llamaremos transiciones, y que son:
    - **Transición de Elevación:** La que se produce al comienzo de la etapa de transferencia, esto es, cuando el pie impulsa el cuerpo de forma que la pierna correspondiente pierde contacto con el suelo.
    - **Transición de Colisión:** La que se produce al final de la etapa de transferencia, esto es, cuando el pie que se encontraba en vuelo hace contacto con el suelo.
- ***Etapa de Soporte o de Postura – Repite posición inicial – Nuevo Ciclo:*** Tras la transición de colisión de la etapa de balanceo, el humanoide se encuentra de nuevo en la etapa de soporte dispuesto para iniciar otro ciclo de locomoción.

Para las dos etapas del ciclo tenemos que resolver el problema algebraico diferencial, aunque la primera tiene mayores restricciones. Otra característica que aparece es la saturación de las variables de estado y de las fuerzas aplicadas. Hay que considerar igualmente las discontinuidades en las velocidades, que se producen entre etapas cuando el pie que se encontraba en vuelo se apoya en el suelo.

Las ecuaciones de estado de un robot bípedo tienen igual forma que las presentadas en los apartados anteriores para cadenas de sólidos rígidos (2.2) y sistemas mecánicos con topología en árbol, pero tenemos que añadir las fuerzas de contacto que se producen entre los pies y el suelo. Debido a la dificultad matemática del problema se recurre muchas veces a herramientas informáticas que utilizan algoritmos simbólicos.

Existen dos paradigmas básicos en la locomoción bípeda, estos son, el de "**equilibrio estático**" y el de "**equilibrio dinámico**", que se desarrollan más formalmente en el punto siguiente (3.1.1). El equilibrio estático lleva a movimientos lentos y menos naturales, mientras que para ganar eficiencia y velocidad tenemos que conseguir un equilibrio dinámico. Ambos tipos de equilibrio se basan en la implementación de leyes de control que utilizan el concepto de Área de Soporte $\mathbf{A_s}$, que viene dada por los polígonos de apoyo de las diferentes etapas de locomoción (ver Figura 3-1). Para el equilibrio estático la variable que hay que controlar para mantener en todo momento dentro del $\mathbf{A_s}$ es la proyección vertical del **CM**, mientras que para el equilibrio dinámico la variable a controlar para mantener dentro del $\mathbf{A_s}$ es el **ZMP**. Si bien queda claro que el equilibrio dinámico es deseable, éste supone un control de muy difícil implementación para mecanismos tan complejos como los robots humanoides, por lo que para muchas investigaciones resulta más práctico y suficiente un control estático.

El concepto de estabilidad dinámica también lo podemos analizar desde el punto de vista de la teoría de control. La definición de equilibrio dinámico se puede hacer en función de la respuesta del sistema mecánico ante las perturbaciones. En este sentido, vamos a utilizar dos tipos de equilibrio: la estabilidad dinámica de la postura y la estabilidad dinámica de la trayectoria. Si se desea que un robot bípedo consiga caminar de una forma estable, se necesitará un sistema de control para garantizar el mantenimiento de la condición de estabilidad para ambas dos.

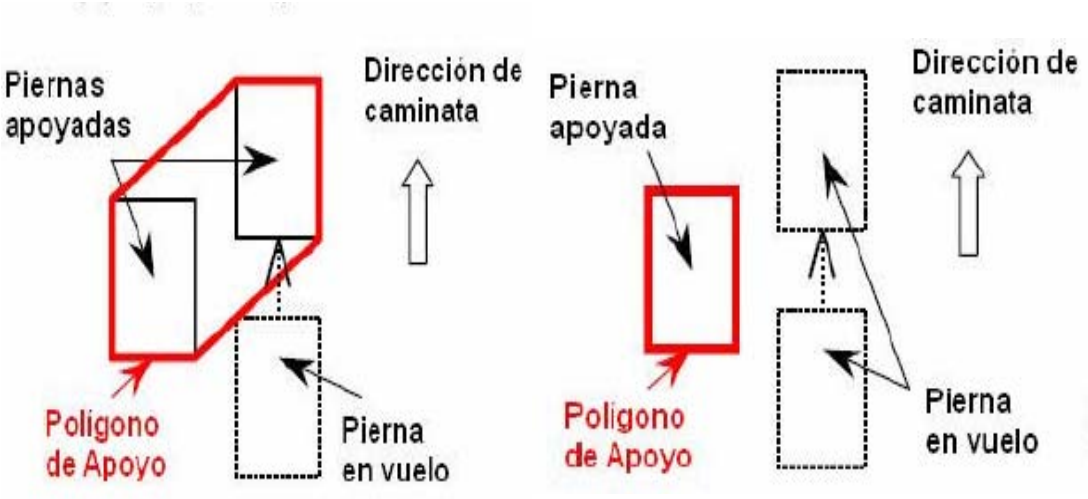


Figura 3-1: Polígonos de Apoyo doble y simple, para la Locomoción Bípeda.

3.1.1 Formalización del Equilibrio Estático y Dinámico.

El **equilibrio estático** se basa en la estabilidad del **CM** del robot, aunque por simplificación muchas veces se analiza en su lugar el **CM** del tronco del humanoide.

Una máquina con locomoción bípeda es estáticamente estable si para un determinado tiempo las velocidades generalizadas del cuerpo se anulan. En el equilibrio estático, el **CM** se encuentra siempre proyectado verticalmente sobre el área de soporte, como se ve en la parte izquierda de la Figura 3-2. En otras palabras, una locomoción bípeda es estáticamente estable si el margen de estabilidad estática es positivo en todo momento, esto es, si la ecuación "(3-1)" es satisfecha.

$$P_{CM}(t) \in A_s(t) \forall\, t \qquad \textbf{(3-1)}$$

Podemos formular la velocidad de locomoción "(3-2)", en función de la velocidad relativa del pie $\mathbf{V_{rp}}$ y el necesario factor de función para el equilibrio $\boldsymbol{\beta}$. Se aprecia que al tener el equilibrio estático factores de función relativamente grandes, la velocidad de locomoción $\mathbf{V_l}$ se reduce notablemente.

$$V_l \leq \frac{1-\beta}{\beta} \cdot V_{rp} \qquad \textbf{(3-2)}$$

Aclarar que el equilibrio no es sinónimo de locomoción segura. Por ejemplo, si un humanoide se encuentra realizando una locomoción con cierta velocidad y repentinamente se bloquean todos los motores de las articulaciones, el robot puede sufrir un momento de vuelco y caer debido a las fuerzas de inercia. El concepto de **locomoción segura** es precisamente el control que evita que eso suceda.

El **equilibrio dinámico** permite que la proyección del **CM** abandone el área de soporte por períodos de tiempo breves, consiguiendo una aceleración horizontal, siempre que estos períodos de tiempo sean pequeños para no provocar inestabilidad, como se ve en la parte derecha de la Figura 3-2.

Muchas veces se define la locomoción dinámica como aquella que no es estática para todo el tiempo, esto es, aquella que cumple la ecuación "(3-3)". El concepto tiene que ver con los efectos de la velocidad, esto es, una locomoción es dinámica cuando al reducir su velocidad hasta un determinado punto el robot perderá su equilibrio como resultado de la pérdida de los efectos dinámicos sobre el mecanismo.

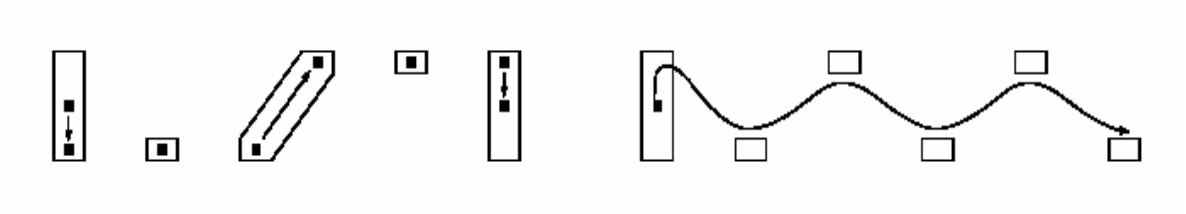


Figura 3-2: Posición del CM en Locomoción bípeda estática (izquierda) y dinámica (derecha).

Con el equilibrio estático, cualquier reducción de la velocidad de locomoción sólo supone movimiento más lento, pero no una caída del robot.

$$\exists t \neg P_{CM}(t) \notin A_s(t) \qquad (3\text{-}3)$$

Una locomoción bípeda es "dinámicamente estable", si la proyección del centro de presión $\mathbf{P_{cp}}$ se encuentra incluida en el área de soporte, esto es si el margen de estabilidad dinámico es positivo, es decir, si la ecuación "(3-4)" es satisfecha.

$$P_{CP}(t) \in A_s(t) \forall t \qquad (3\text{-}4)$$

De todos modos, el criterio más utilizado en la locomoción bípeda para conseguir un **movimiento dinámicamente estable**, es garantizar que el **ZMP** se encuentre incluido en el área de soporte en todo momento "(3-5)".

$$ZMP \in A_s(t) \forall t \qquad (3\text{-}5)$$

Los pies no pueden ser controlados de forma directa, por lo que el mejor indicador del comportamiento dinámico del mecanismo es el punto donde la influencia de todas las fuerzas que actúan sobre el humanoide se puede reemplazar por una única fuerza, este punto es el **ZMP**. Como se aprecia en la Figura 3-3 que representa una locomoción en equilibrio dinámico para un humanoide, aunque la proyección del **CM** sale fuera del $\mathbf{A_s}$, el **ZMP** se mantiene dentro del mismo, ya que si no se consigue esto se produce un momento de vuelco que desestabiliza el movimiento y provoca la caída del robot.

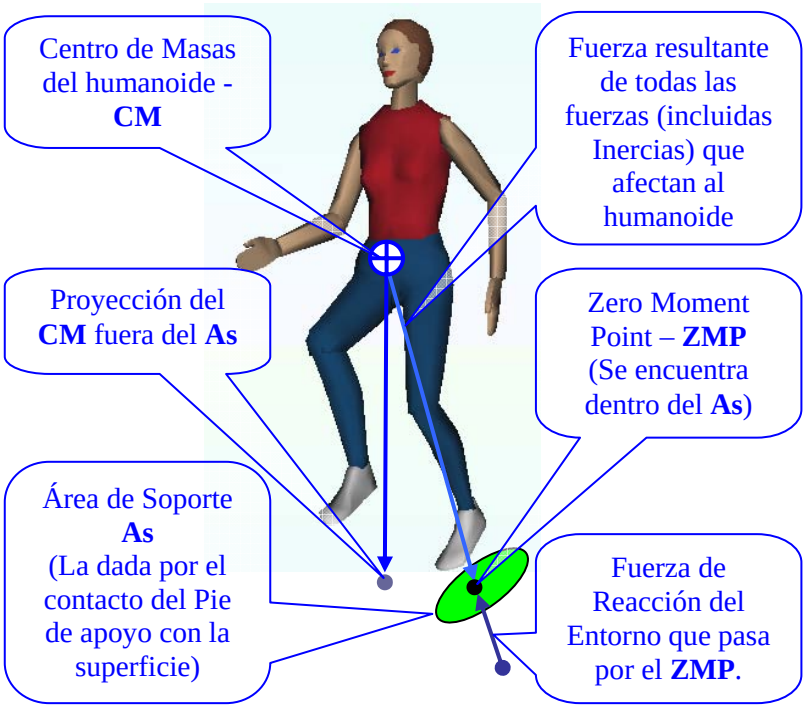


Figura 3-3: Locomoción Bípeda Dinámica con proyección del CM y situación del ZMP.

3.2 NUEVO MODELO MECÁNICO – División Cinemática Sagital (DCS).

Un robot humanoide tiene típicamente un número de **GDL** muy elevado, por lo que puede haber muchas soluciones para alcanzar un determinado movimiento, haciendo que los problemas de cinemática 2.3 y dinámica 2.4 sean ya muy complejos en principio, y tanto más cuanto mayor el número de **GDL**.

En muchas investigaciones y desarrollos se considera al robot humanoide como un sistema mecánico con topología en árbol 2.2.1, sólo que a diferencia de los mecanismos más comunes de este tipo, aquí el eslabón común (i.e., el tronco del humanoide) de las cadenas de sólidos rígidos (i.e., los brazos y piernas) se mueve y no queda asociado a una base con sistema de referencia inercial, lo que dificulta mucho más la solución del problema. Además, al considerar el sistema con topología en árbol se producen bucles cinemáticos cerrados cuando los dos pies del humanoide están en contacto con el suelo o cuando las manos u otra parte del robot están en contacto con el entorno, lo que convierte al problema inicial en otro mucho más difícil, puesto que el modelo se complica con la introducción de restricciones.

Por si lo comentado anteriormente fuese poco, tenemos que resolver el problema mecánico fundamental que constituye la locomoción bípeda del humanoide 3.1., que introduce la necesidad de equilibrio tanto estático como dinámico.

Es por todas estas consideraciones por lo que se han estado buscando simplificaciones al problema. En esta línea, esta tesis introduce la idea de un nuevo modelo mecánico para el estudio de humanoides, este es el que denominamos modelo por "**División Cinemática Sagital**" (**DCS**). La idea que subyace detrás del nuevo modelo **DCS** es que podamos analizar y controlar el humanoide considerando separadamente las dos mitades (izquierda y derecha) de su cuerpo, a semejanza del control que realizan los hemisferios cerebrales para la locomoción del cuerpo humano.

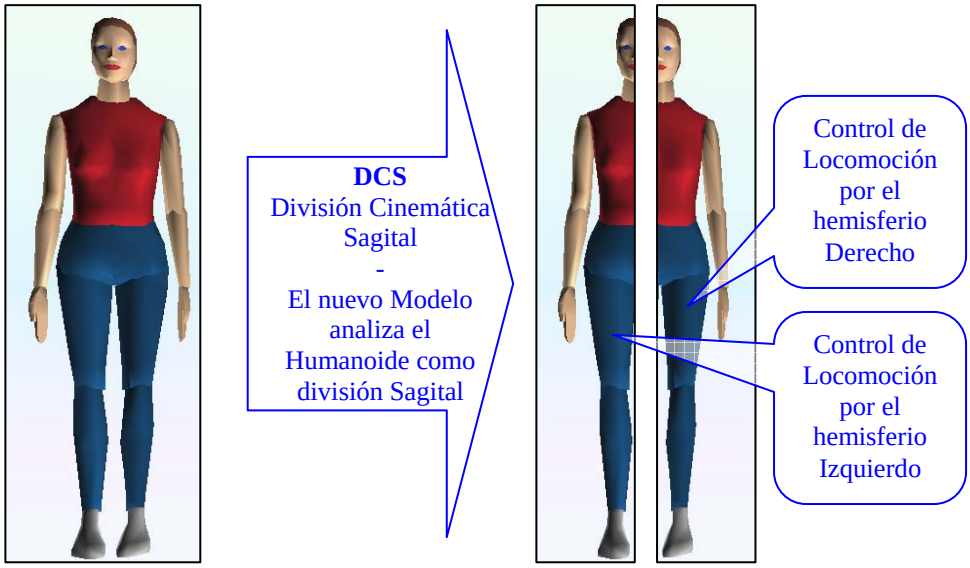


Figura 3-4: Idea del DCS a semejanza del control sagital del cerebro para el cuerpo humano.

Con esa idea, el **DCS** considera desde el punto de vista del análisis mecánico y de control, que el robot humanoide se encuentra dividido mecánicamente por su plano sagital, esto es, queda formado por dos robos manipuladores (izquierdo y derecho) acoplados por las partes comunes (normalmente la pelvis y el tronco del humanoide).

Algunas soluciones ampliamente utilizadas cambian de modelo mecánico conforme cambia la etapa de transferencia de la locomoción, "rompiendo" el bucle cinemático para solucionar las etapas de soporte. El **DCS** resulta ser mucho más sencillo, puesto que para él no existen bucles cinemáticos, derivandose las consecuencias prácticas que utilizamos en 3.4 para resolver el problema cinemático inverso del humanoide.

El **DCS** es un planteamiento de "divide y vencerás" que nos permite resolver el problema mecánico del humanoide de una forma mucho más sencilla mediante la resolución del problema mecánico de dos manipuladores, puesto que ya hemos visto cómo resolver de forma geométrica, mediante las técnicas de Lie, la cinemática de un manipulador (ver 2.3, A.1 y A.2) y también la dinámica en 2.4. El **DCS** introduce restricciones de sincronización de las partes comunes, pero esto es mucho más sencillo de resolver que todos los bucles cinemáticos producidos por los apoyos de los miembros del humanoide, especialmente los producidos por los pies al caminar.

Otro concepto que introduce el **DCS** es la distinción entre dos tipos de **GDL** dentro del modelo mecánico:

- **GDL Reales - θ**: Son los mecánicos, los que tiene el mecanismo del humanoide debido a las articulaciones mecánicas que mueven físicamente sus miembros.
- **GDL Virtuales - θv**: Son los derivados por la actuación conjunta del sistema mecánico. Pueden ser obtenidos por cinemática directa o ser obtenidos por algún tipo de planificación. Para la locomoción bípeda nos interesarán los **GDL** virtuales que definen las posiciones y orientaciones de los pies, además de los **GDL** virtuales del **CM** que definen el movimiento global del humanoide.

Todo esto nos va a permitir estudiar la mecánica de los dos robots manipuladores de una forma sencilla, puesto que cualquiera que sea la posición real del humanoide y de sus extremidades, siempre podremos considerar que los manipuladores quedan unidos al sistema de referencia inercial mediante los **GDL** virtuales de los pies.

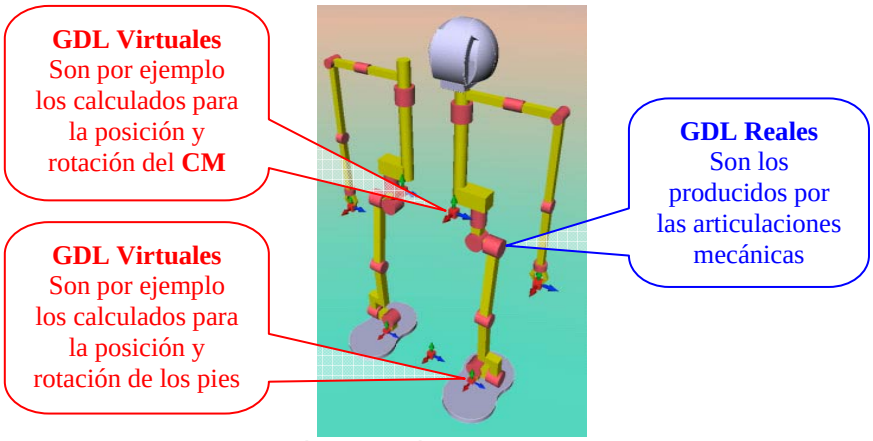


Figura 3-5: División Cinemática Sagital DCS para un humanoide con GDL Virtuales y Reales.

3.3 NUEVO ALGORITMO de Locomoción Bípeda – Un Paso Adelante (UPA.)

Abordamos ahora el problema de la **navegación local o planificación de movimientos para la locomoción bípeda de un humanoide**. Como hemos visto en 3.2 y a lo largo de este capítulo, son muchas las dificultades que nos encontramos: el elevado número de **GDL**, los bucles cinemáticos del modelo, la coordinación de los miembros y el equilibrio del sistema.

Para cualquier aplicación práctica, el humanoide debe desarrollar una locomoción bípeda compleja (i.e., compuesta por varios pasos, junto con probables cambios de dirección), que viene dirigida por algún método de planificación de trayectorias o navegación global (en esta tesis, la información correspondiente al camino de navegación global se obtiene con el modelo **TCG** 4.4, que utiliza el nuevo algoritmo **M3R** 4.3. Obtener una solución cerrada para un movimiento tan complejo resulta muy difícil, y lo que es más, de obtenerla no sería una solución muy práctica, puesto que supone un movimiento de tiempo prolongado dentro de entorno perfectamente conocido, cuando en realidad los mapas basados en modelado del entorno son raramente perfectos y además pueden ser dinámicos, con cambios en el número y estructura de los obstáculos. Por lo tanto, resultaría más útil desarrollar un movimiento elemental de locomoción bípeda (i.e. un paso), basado en una planificación local, incluso on-line, que pueda considerar información sensorial para poder planificar actos reflejos en reacción a cambios del entorno.

Como hemos estudiado en 3.1 la mecánica de la locomoción bípeda tiene muchas veces una estructura periódica y casi siempre simétrica. Esto nos puede permitir restringir el problema a la resolución del problema mecánico correspondiente a un único ciclo del paso, esto es, a la resolución de una sola etapa de soporte seguida de una sola etapa de transición. Así, la solución para una locomoción bípeda compleja puede desarrollarse con soluciones encadenadas de un único paso, siempre que la solución sea general y contemple todos los posibles casos, como son diferentes longitudes anchuras y altitudes de paso, o cambios en la dirección del movimiento.

Por todo ello, proponemos un nuevo algoritmo al que llamamos "**Un Paso Adelante**" (**UPA**) para el movimiento de locomoción bípeda de robots humanoides, cuya idea subyacente es desarrollar una solución de propósito general para el problema mecánico consistente en mover el humanoide un único paso adelante hacia un objetivo dado. El **UPA** recibe como entradas el objetivo local hacia el que se debe caminar (obtenido de la navegación), el modelo del entorno local (que puede ser mejorado mediante integración sensorial) y las características típicas de la locomoción bípeda (i.e., longitud, anchura y altura de paso, altura del **CM**). Los resultados obtenidos en la ejecución del algoritmo **UPA**, son los datos correspondientes a las trayectorias de los **GDL** virtuales $\boldsymbol{\theta v_{PM}}$ del pie móvil (**PM**) durante el ciclo del paso (i.e., trayectoria de la posición y rotación del **PM**) y a las trayectorias de los **GDL** virtuales $\boldsymbol{\theta v_{CM}}$ del **CM** del humanoide durante el ciclo del paso (i.e., trayectoria de la posición y rotación del **CM** que define el movimiento global del humanoide), que por supuesto respetará las restricciones dadas para el equilibrio estático o dinámico 3.1.1 de la locomoción bípeda, esto es, que $\mathbf{P_{CM}}$ "(3-1)"o el **ZMP** "(3-5)"se encuentren en todo momento dentro del área de soporte.

3.3.1 Una Formalización Genérica del algoritmo UPA.

Como acabamos de explicar en 3.3, la idea fundamental para desarrollar el nuevo algoritmo "Un Paso Adelante" (**UPA**) es que debe proporcionar una solución general para el problema mecánico consistente en mover el humanoide un único paso adelante hacia un objetivo dado, cualquiera que sea la configuración mecánica del robot. Con esta premisa se pueden obtener diferentes formulaciones para el algoritmo **UPA**, que se adapten de forma más precisa a las características mecánicas del robot real con el que trabajemos, como haremos en 3.5 para el humanoide **RH0**. Sin embargo, en este apartado lo que nos ocupa es presentar una formalización genérica del algoritmo **UPA**, que aún no siendo única, se pueda implementar directamente para una amplia variedad de robots (e.g. humanoides ya construidos con mecánicas conocidas), sin perjuicio de que puedan refinarse más las implementaciones particulares.

Esta implementación genérica del algoritmo **UPA** ha sido exitosamente probada con varios humanoides en el simulador **RobManSim** que presentaremos en el Capítulo-5, se caracteriza por su simplicidad de concepción y desarrollo. Para obtener los resultados buscados (i.e., los caminos factibles para el **PM** $\mathbf{Q_{rPM}}$, y para el **CM** $\mathbf{Q_{rCM}}$), el algoritmo **UPA** desarrolla el ciclo del paso teórico (descrito en 3.1) para una locomoción bípeda, mediante la definición de **cinco fases** fundamentales, de la siguiente forma:

- *Etapa de Soporte o de Postura*: Esta etapa existe cuando los dos pies están en contacto con el suelo se produce un ***doble apoyo***. La posición del **ZMP** se sitúa dentro del área de soporte dada por la envolvente del polígono formado por el doble apoyo de los dos pies, pero fuera de la superficie de soporte del pie de apoyo del anterior ciclo de locomoción.
    - **FASE-1 Orientar:** Sin mover los pies, se orienta el cuerpo del robot humanoide hasta alinearlo lo más posible con la dirección del objetivo. Es parte de la etapa de soporte, cuando los dos pies están en contacto con el suelo. Para garantizar la estabilidad, se mantiene el **ZMP** dentro del $\mathbf{A_s}$ dada por el patrón de soporte formado por los dos pies.
    - **FASE-2 Inclinar:** Sin mover los pies, se inclina el cuerpo del humanoide para que el **ZMP** se mueva hasta alcanzar una posición interna al patrón de soporte dado por el pie que quedará como apoyo.
- *Etapa de Transferencia o de Balanceo*: Esta etapa existe cuando sólo un pie está en contacto con el suelo, mientras que el otro pie se encuentra en vuelo, por lo que se dispone sólo de un ***apoyo simple***. Se mantiene hasta que la posición del **ZMP** abandona el área de soporte dada por el pie de apoyo, esto es, aún cuando se produce la transición de colisión que sitúa a ambos pies sobre la superficie.
    - **FASE-3 Elevar:** Constituye la Transición de Elevación, mediante la que se eleva el **PM**, pasando desde su "Posición Extrema Posterior" (**PEP**) hasta la posición dada por la altura de paso y de ahí hasta la "Posición Extrema Anterior" (**PEA**) dada por la longitud de paso.
    - **FASE-4 Apoyar:** Es la Transición de Colisión, en la que aterriza el **PM** tocando el suelo. El **ZMP** sigue dentro del patrón de soporte dado por el pie de apoyo, puesto que estamos aún en la parte final de la etapa de transferencia de Balanceo.

- *Etapa de Soporte o de Postura – Repite Posición Inicial – Nuevo Ciclo*: Esta etapa existe cuando los dos pies están en contacto con el suelo y se produce un ***doble apoyo***. Es el comienzo del nuevo ciclo de locomoción, pero no se podrá considerar la repetición periódica del movimiento hasta no completar la última fase de balanceo.

    - **FASE-5 Balancear:** Sin mover los pies, se balancea el cuerpo del humanoide para que el **ZMP** se mueva hasta alcanzar la posición central del patrón de soporte dado por los dos pies. De esta forma, al final de la fase cinco nos encontramos en condiciones de repetir otro paso, con la ejecución periódica del algoritmo **UPA** desde el comienzo.

Mediante estas cinco fases, (ver 3.3.2) vamos a **obtener de forma geométrica** los cinco puntos principales que forman cada uno de los caminos factibles $\mathbf{Q_{rPM}}$, y $\mathbf{Q_{rCM}}$, que son el resultado buscado por algoritmo **UPA**. Para las cinco fases, y por lo tanto para los cinco puntos obtenidos en cada camino solución, por supuesto se respetan las restricciones dadas para el equilibrio 3.1.1 de la locomoción bípeda, es decir, el $\mathbf{P_{CM}}$ "(3-1)"o el **ZMP** "(3-5)"se encuentren en todo momento dentro del $\mathbf{A_s}$. Teniendo los cinco valores principales, posteriormente los caminos factibles se pueden completar por interpolación hasta el valor de granularidad que sea necesario, según las trayectorias que favorezcan el comportamiento mecánico de cada robot específico para el que se implemente el algoritmo (ver 3.5 para el **RH0**).

Definimos por tanto las expresiones que presentan los resultados del algoritmo **UPA**:

- $\mathbf{Q_{rCM}}$ **- Camino Factible del CM**: Trayectoria compuesta por configuraciones alcanzables "(4-6)" pertenecientes todas ellas al espacio libre "(4-3)", que presentan continuidad "(4-5)". Donde $\mathbf{q_{CM}}$ es una configuración del **CM** del humanoide, dada por los **GDL** ficticios $\boldsymbol{\theta}\mathbf{v_{CM}}$ del **CM** (i.e., $\boldsymbol{\theta}\mathbf{v_{CM1}}...\boldsymbol{\theta}\mathbf{v_{CM6}}$) para las tres traslaciones y tres rotaciones generalizadas del **CM** en $\mathbf{R^3}$. El $\mathbf{Q_{rCM}}$ viene definido según "(3-6)" por cinco puntos principales (i.e., $\mathbf{q_{CM1}}...\mathbf{q_{CM5}}$) que son soluciones geométricas a las cinco fases del algoritmo **UPA**.

$$Q_{rCM} = \{q_{CM1}...q_{CM2}...q_{CM3}...q_{CM4}...q_{CM5}\} \qquad \textbf{(3-6)}$$

- $\mathbf{Q_{rPM}}$ **- Camino Factible del PM**: Trayectoria de configuraciones alcanzables "(4-6)" pertenecientes al espacio libre "(4-3)" con continuidad "(4-5)". Definimos $\mathbf{q_{PM}}$ como una configuración del **PM**, dada por los **GDL** ficticios $\boldsymbol{\theta}\mathbf{v_{PM}}$ del **PM** (i.e., $\boldsymbol{\theta}\mathbf{v_{PM1}}...\boldsymbol{\theta}\mathbf{v_{PM6}}$) para los movimientos generalizados del **PM** en $\mathbf{R^3}$. El $\mathbf{Q_{rPM}}$ viene definido según "(3-7)" por cinco puntos principales (i.e. $\mathbf{q_{PM1}}...\mathbf{q_{PM5}}$) que son soluciones geométricas de las fases del algoritmo **UPA**.

$$Q_{rPM} = \{q_{PM1}...q_{PM2}...q_{PM3}...q_{PM4}...q_{PM5}\} \qquad \textbf{(3-7)}$$

Una vez obtenidos los resultados del algoritmo **UPA** (i.e, $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$), podremos generar el movimiento de locomoción bípeda correspondiente a un paso del robot humanoide genérico, resolviendo el problema cinemático inverso (ver 2.3), para los dos manipuladores que constituyen el humanoide según **DCS** (ver 3.2). Antes de pasar a desarrollar un esquema geométrico eficaz para el **UPA** en 3.3.2, podemos ver en la Figura 3-6 una imagen de su implementación para que un humanoide ejecute cuatro pasos consecutivos hacia el objetivo marcado por una semiesfera roja.

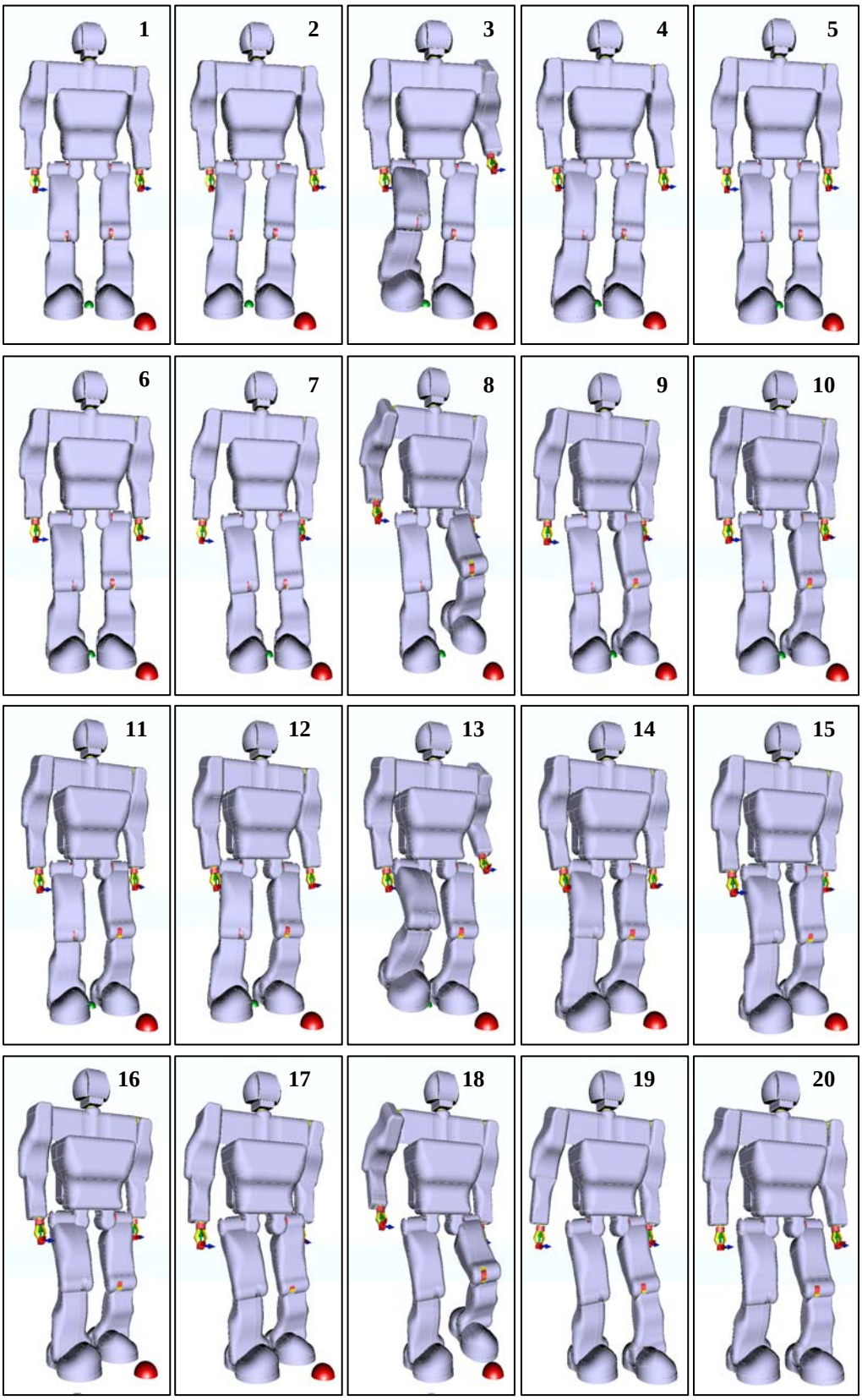


Figura 3-6: 4 pasos del UPA: 1-Orientar, 2-Inclinar, 3-Elevar, 4-Apoyar y 5-Balancear.

3.3.2 Esquema Geométrico para el algoritmo UPA Genérico.

Para una mejor compresión del funcionamiento del algoritmo **UPA** en su formalización genérica para cualquier tipo de humanoide, describiremos en detalle las soluciones geométricas para las cinco fases que lo componen. En el ejemplo que vamos a utilizar, el humanoide ejecuta un paso girando hacia un objetivo que se encuentra situado a su izquierda, para que se vea el funcionamiento de propósito general del algoritmo **UPA** que no sólo sirve para una locomoción en línea recta. Para ello, desarrollaremos diagramas de paso para cada una de las fases. La implementación particular del **UPA** para el **RH0** se verá con detalle en el apartado 3.5 y difiere en cierta medida de lo que aquí presentamos, puesto que está optimizada para ese robot.

Para entender los diagramas de paso de las descripciones geométricas, necesitamos repasar algunas definiciones que se pueden encontrar detalladas en el glosario de la tesis (Apéndice C): Área de sopote del pie móvil $\mathbf{As_{PM}}$, área de soporte del pie fijo $\mathbf{As_{PF}}$, pasillo de dirección **Pd** que viene definido por la anchura de paso $\mathbf{W_p}$, eje longitudinal $\mathbf{\upsilon_l}$, eje de dirección $\mathbf{\upsilon_d}$, eje de avance del pie móvil $\mathbf{\upsilon_{PM}}$ (las figuras permiten reconocer estos conceptos con facilidad). Se representarán en los diagramas de paso el **ZMP**, que coincide con la proyección de la configuración del centro de masas $\mathbf{q_{CM}}$, si queremos realizar un control de equilibrio estático, y la configuración del pie móvil $\mathbf{q_{PM}}$.

En primer lugar, describimos las condiciones iniciales que vienen dadas por la presencia de un robot humanoide en posición de equilibrio estático, en una etapa de soporte del paso (ver Figura 3-7) con doble apoyo. Se puede apreciar un objetivo local $\mathbf{q_{ol}}$ representado por una semiesfera roja, que indica la dirección y sentido para el movimiento de locomoción del robot. El objetivo local para la planificación de movimientos suele venir dado como resultado de la navegación global del humanoide (e.g., con el modelo **TCG** o el algoritmo **M3R**), aunque puede definirse con cualquier otro algoritmo, como pueden ser modelos reactivos de respuesta refleja o señales de interacción con el hombre, siempre que señalen un objetivo local.

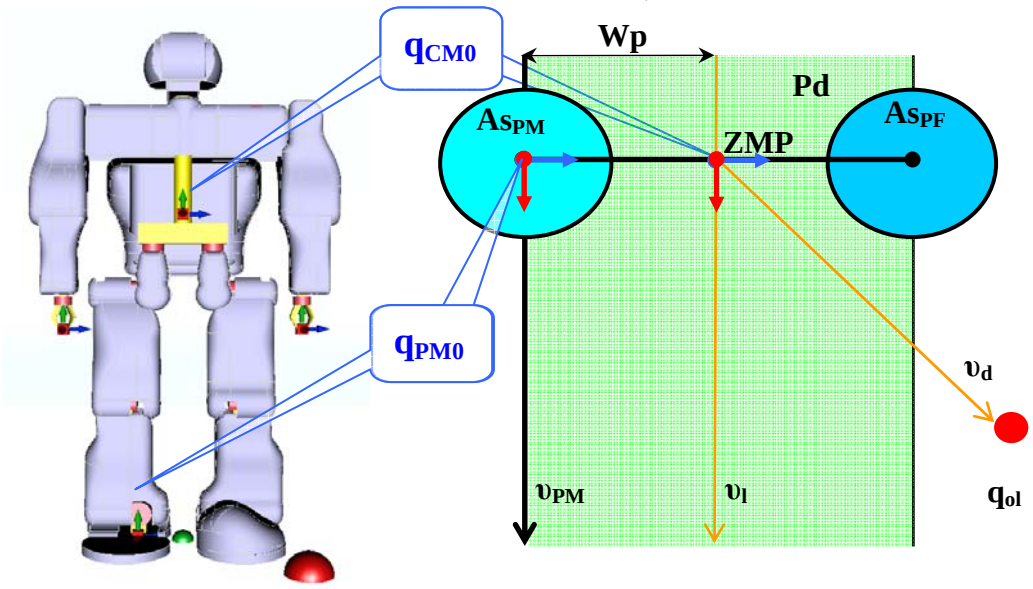


Figura 3-7: Condiciones Iniciales del UPA – Humanoide en etapa de soporte estático.

Fase-1 del algoritmo UPA – ORIENTAR: En la Figura 3-8, el humanoide continúa con la posición firme de los dos pies sobre la superficie de soporte, por lo que la configuración del pie móvil $\mathbf{q}_{PM1}$ es igual a $\mathbf{q}_{PM0}$. La configuración del centro de masas **CM** pasa a ser $\mathbf{q}_{CM1}$, que no es sino la inicial $\mathbf{q}_{CM0}$ orientada lo mejor posible con la dirección del objetivo $\mathbf{q}_{ol}$, esto es, hay que orientar el tronco del robot lo más posible hacia el eje de dirección υ_d. Si el robot se encuentra en su posición erguida de máxima altura la orientación no sería posible, como tampoco la extensión de las piernas para avanzar un paso, por lo que en este movimiento se incluye el descenso del **CM** hasta una altura que permita a los **GDL** de las piernas ejecutar tanto el giro de caderas como el avance del paso deseado. Estos movimientos vendrán limitados por las restricciones mecánicas del robot concreto al que apliquemos el algoritmo.

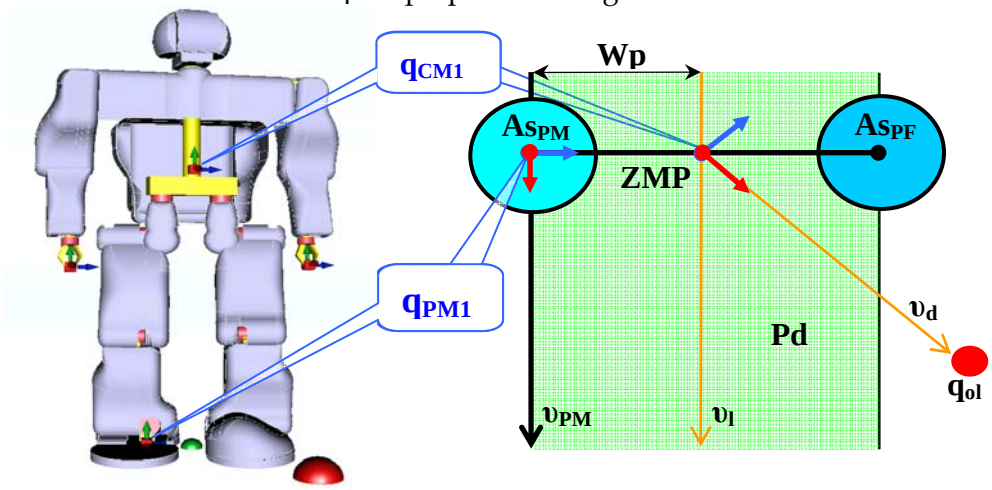


Figura 3-8: Fase-1 del algoritmo UPA - ORIENTAR.

Fase-2 del algoritmo UPA – INCLINAR: El robot sigue con los dos pies sobre la superficie de soporte (ver Figura 3-9), pero se inclina el cuerpo del humanoide para que el **ZMP** se mueva hasta alcanzar el área de soporte del pie fijo $\mathbf{As}_{PF}$. La configuración del pie móvil $\mathbf{q}_{PM2}$ no cambia y es igual a $\mathbf{q}_{PM1}$. La configuración del centro de masas pasa a ser $\mathbf{q}_{CM2}$, que no es sino la anterior $\mathbf{q}_{CM1}$ trasladada en la dirección marcada por el eje que une los dos pies hasta alcanzar el patrón de soporte del **PF**. Resulta evidente que conociendo el radio del patrón de soporte del **PF** y $\mathbf{q}_{CM1}$, $\mathbf{q}_{CM2}$ queda definida de forma geométrica {RobotMan-f(28):**SphereLineIntersection**}.

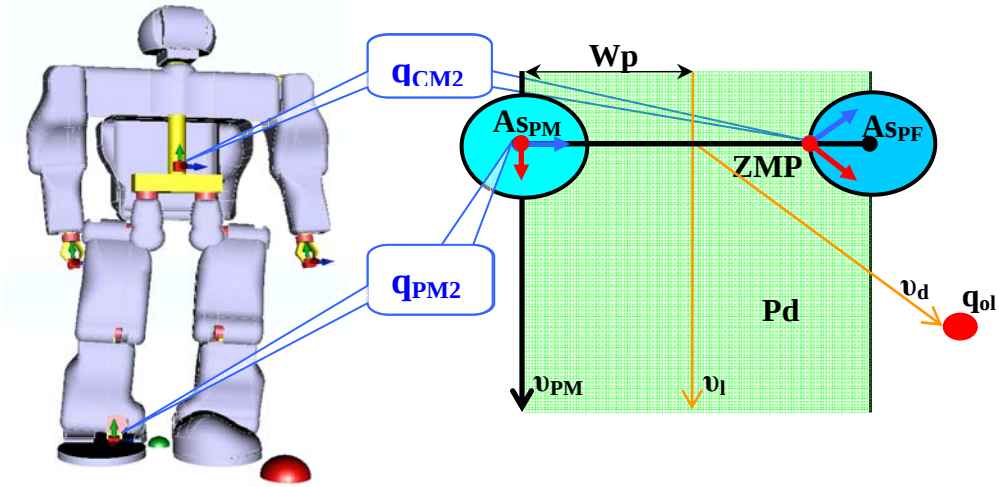


Figura 3-9: Fase-2 del algoritmo UPA - INCLINAR.

Fase-3 del algoritmo UPA – ELEVAR: Durante esta fase (ver Figura 3-10), el humanoide se encuentra siempre en equilibrio al moverse el **ZMP** sobre el patrón de soporte del **PF** (i.e., el límite circular del $\mathbf{As_{PF}}$). El robot eleva el **PM** hasta alcanzar la altura de paso **Hp** en el punto del plano dado por el diagrama de paso. Con esos datos y la orientación del $\mathbf{\upsilon_d}$ podemos definir la configuración $\mathbf{q_{PM3}}$. La configuración del **CM** pasa a ser $\mathbf{q_{CM3}}$, que no es sino la anterior $\mathbf{q_{CM2}}$ trasladada sobre el patrón de soporte circular del **PF**. Conociendo el radio de $\mathbf{As_{PF}}$ y el eje que une los pies, $\mathbf{q_{CM3}}$ queda definida de forma geométrica {RobotMan-f(28):**SphereLineIntersection**}.

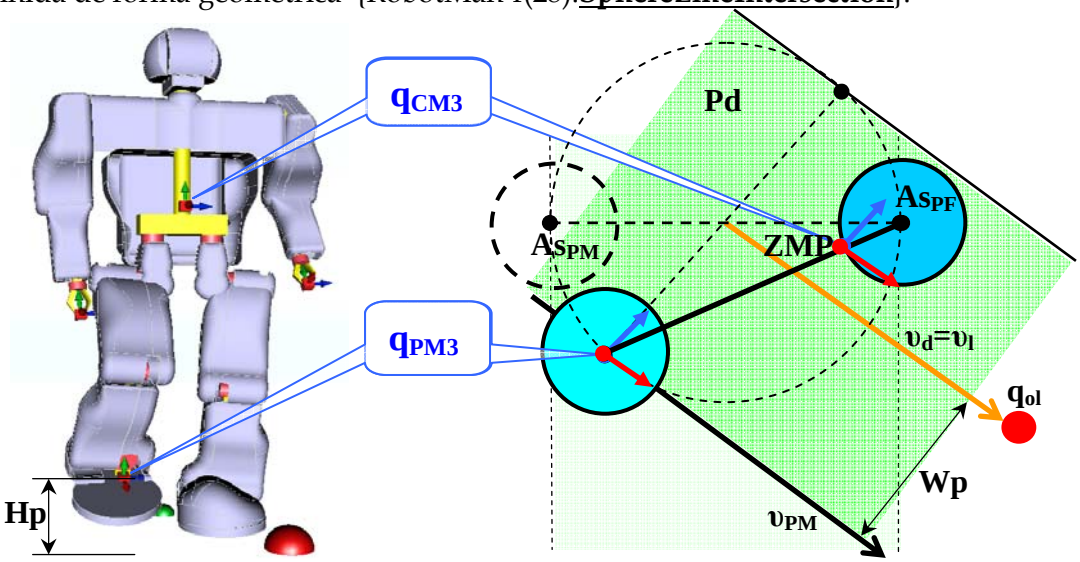


Figura 3-10: Fase-3 del algoritmo UPA - ELEVAR.

Fase-4 del algoritmo UPA – APOYAR: En esta fase (verFigura 3-11), el humanoide continúa en equilibrio sobre $\mathbf{As_{PF}}$ al moverse el **ZMP** sobre ese patrón de soporte. El robot apoya el **PM** aterrizándolo en el suelo, en la posición dada por **Lp** y la geometría propuesta en el diagrama de paso. Conociendo estos datos y $\mathbf{q_{PM3}}$ podemos definir la nueva configuración del **PM** $\mathbf{q_{PM4}}$. La nueva configuración del **CM** es $\mathbf{q_{CM4}}$ como resultado de continuar el movimiento de $\mathbf{q_{CM3}}$ sobre el patrón de soporte circular del **PF**. Conociendo el radio de $\mathbf{As_{PF}}$ y el eje que une los pies, $\mathbf{q_{CM4}}$ se define trivialmente de forma geométrica {RobotMan-f(28):**SphereLineIntersection**}.

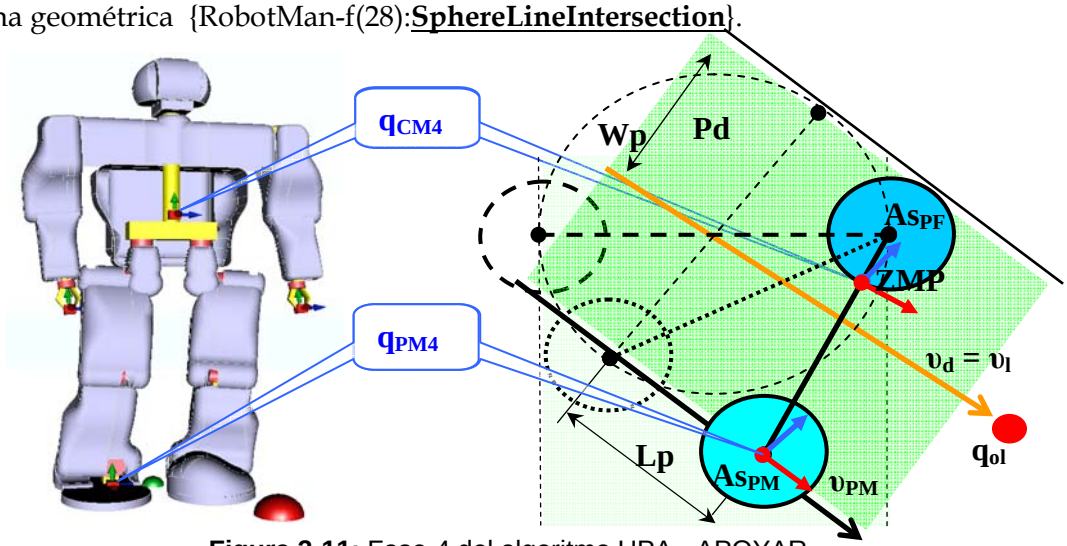


Figura 3-11: Fase-4 del algoritmo UPA - APOYAR.

Fase-5 del algoritmo UPA – BALANCEAR: Sin mover los pies, se balancea el cuerpo del humanoide para que el **ZMP** se mueva hasta la posición central del $\mathbf{A_s}$ dada por el patrón de soporte conjunto de los dos pies (ver Figura 3-12). La configuración del pie móvil $\mathbf{q_{PM5}}$ no cambia y es igual a $\mathbf{q_{PM4}}$. La configuración del **CM** pasa a ser $\mathbf{q_{CM5}}$, que no es sino la anterior $\mathbf{q_{CM4}}$ trasladada sobre el eje que une los dos pies, una distancia igual a la mitad de ese eje. De este modo todo se define de forma geométrica.

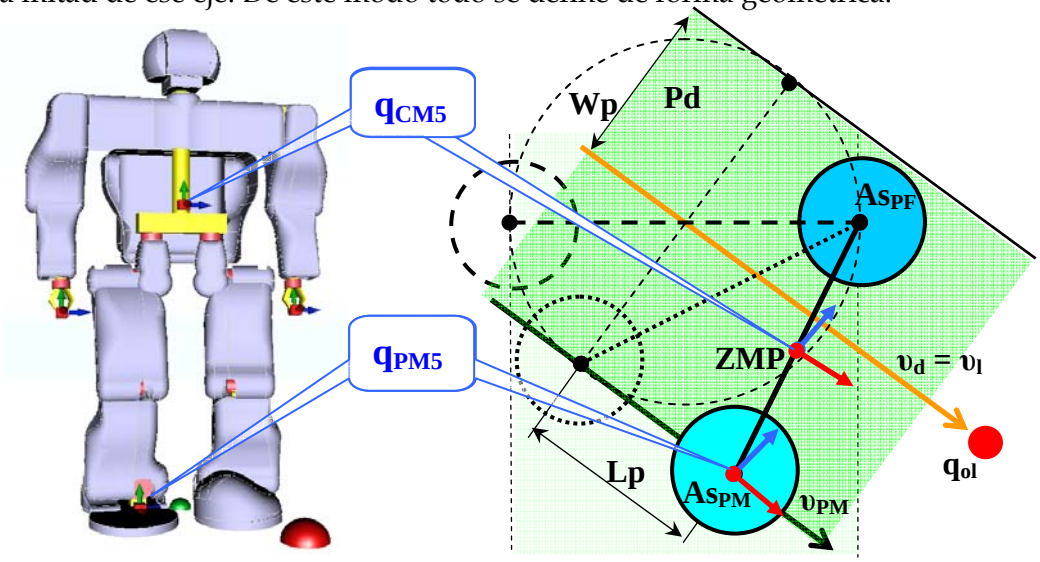


Figura 3-12: Fase-5 del algoritmo UPA – BALANCEAR.

Tras las cinco fases, hemos obtenido de forma geométrica cinco configuraciones para ambos, el **CM** (i.e., $\mathbf{q_{CM1}}$...$\mathbf{q_{CM5}}$) y el **PM** (i.e., $\mathbf{q_{PM1}}$...$\mathbf{q_{PM5}}$), que definen fundamentalmente los resultados del algoritmo **UPA,** que son los caminos del **CM** y del **PM** (i.e, $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$). Para completar los caminos se puede interpolar sin dificultad entre los cinco puntos calculados, por ejemplo, para la trayectoria en vuelo del **PM**, se interpola una función (e.g., senoidal, parabólica u otras) entre las configuraciones $\mathbf{q_{PM2}}$, $\mathbf{q_{PM3}}$, y $\mathbf{q_{PM5}}$.

Al final de la fase cinco, nos encontramos en condiciones de repetir otro paso con la ejecución periódica del algoritmo **UPA** desde el comienzo. Se hace notar, que para una segunda ejecución, el eje de dirección $\mathbf{\upsilon_d}$ se encuentra dentro del $\mathbf{P_d}$, por lo que el humanoide no necesita repetir otro paso de giro, sino que realizará un movimiento más natural de paso adelante. Como el algoritmo **UPA** funciona con generalidad para cualquier circunstancia, la construcción de movimientos mucho más complejos del humanoide se conseguirá sencillamente aplicando sucesivos objetivos de una planificación global como entradas al algoritmo.

Otro hecho a destacar es que el algoritmo **UPA** se puede integrar con otros para la planificación de trayectorias locales libres de colisiones (i.e., **M3R**) sin ninguna modificación y con total generalidad, puesto que la posición de apoyo del **PM** durante la fase 4 de apoyo $\mathbf{q_{PM4}}$ puede ser modificada para evitar un obstáculo (sujeto a limitaciones cinemáticas) sin que se tenga que modificar la idea y ejecución de la fase 5.

Una vez obtenidos los caminos $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$, podremos generar el movimiento de locomoción bípeda en equilibrio correspondiente a un paso de un robot humanoide, resolviendo con técnicas de Lie el problema cinemático inverso (ver 2.3) para los dos manipuladores que constituyen el humanoide según el modelo **DCS** (ver 3.2), como veremos en detalle en 3.4 para la aplicación en el humanoide **RH0.**

3.4 APLICACIÓN del Nuevo Modelo Mecánico DCS al Humanoide RH0.

El nuevo modelo mecánico **DCS** considera que un humanoide genérico se encuentra, desde el punto de vista teórico, dividido mecánicamente por su plano sagital, para de esta forma poder analizar de una forma mucho más sencilla el problema global como análisis de dos robos manipuladores (izquierdo y derecho) acoplados por las partes comunes (i.e., pelvis y el tronco). Ahora aplicaremos el concepto **DCS** al **RH0**.

3.4.1 Modelo Cinemático DCS del RH0.

El modelo cinemático **DCS** del **RH0** queda constituido por dos robots manipuladores según la Figura 3-13. Cada uno de ellos cuenta con seis **GDL** virtuales que definen la posición y orientación de los pies en relación con el sistema de referencia inercial $\mathbf{S_S}$ (i.e., **θvz1** a **θvz6** para el pie izquierdo y **θvd1** a **θvd6** para el derecho), más **11 GDL** reales **θi** correspondientes a los actuadores mecánicos del humanoide. El **GDL** correspondiente al movimiento de la columna **θ13** es común a los dos manipuladores.

El problema cinemático directo del **RH0** queda definido según las fórmulas del **POE**, correspondientes al manipulador izquierdo "(3-8)" y derecho "(3-9)".

$$g(\theta) = e^{\xi_{vz1}\hat{}\theta_{vz1}} \cdots e^{\xi_{vz6}\hat{}\theta_{vz6}} \cdot e^{\xi_{12}\hat{}\theta_{12}} \cdots e^{\xi_{7}\hat{}\theta_{7}} \cdot e^{\xi_{13}\hat{}\theta_{13}} \cdot e^{\xi_{18}\hat{}\theta_{18}} \cdots e^{\xi_{21}\hat{}\theta_{21}} \cdot g(0) \quad \textbf{(3-8)}$$

$$g(\theta) = e^{\xi_{vd1}\hat{}\theta_{vd1}} \cdots e^{\xi_{vd2}\hat{}\theta_{vd6}} \cdot e^{\xi_{1}\hat{}\theta_{1}} \cdots e^{\xi_{6}\hat{}\theta_{6}} \cdot e^{\xi_{13}\hat{}\theta_{13}} \cdots e^{\xi_{17}\hat{}\theta_{17}} \cdot g(0) \quad \textbf{(3-9)}$$

El **CM** es un punto muy importante, común a ambos manipuladores, donde se localiza el sistema de referencia del tronco $\mathbf{S_H}$, que nos sirve para definir $\mathbf{q_{CM}}$ (i.e., $\mathbf{g_{sh}(\theta)}$, la configuración de $\mathbf{S_H}$ con respecto a $\mathbf{S_S}$ {RobotMan-f(2):**BodyGenCoor2homogeneous**}), que debe recorrer el camino factible $\mathbf{Q_{rCM}}$ obtenido por el algoritmo **UPA**.

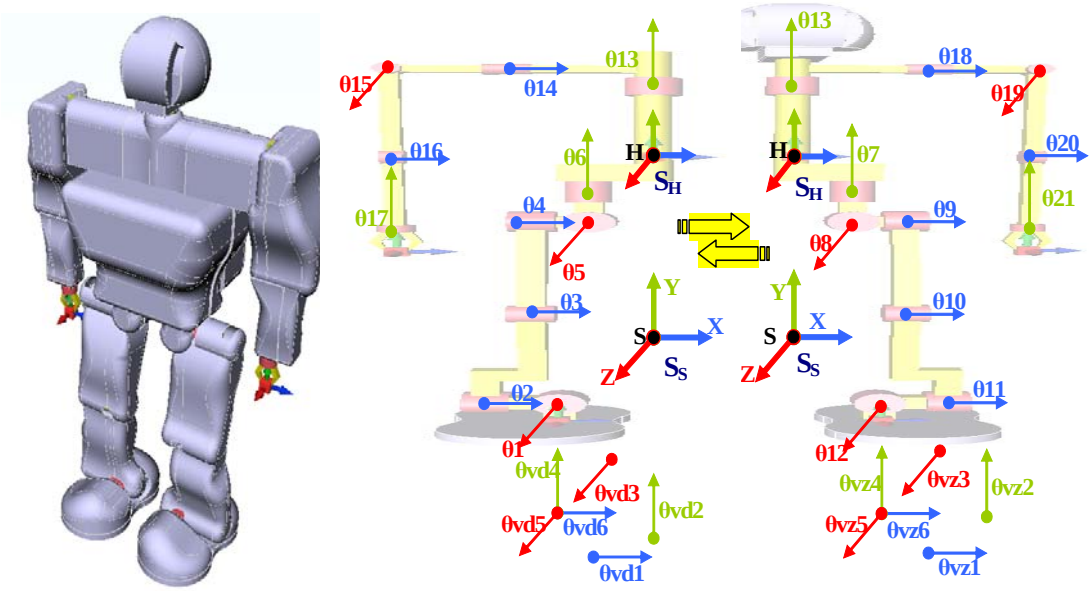


Figura 3-13: Humanoide RH0 y su Modelo cinemático DCS.

3.4.2 Solución Geométrica de la Cinemática Inversa del RH0 con DCS.

La construcción de movimientos complejos de locomoción bípeda para el **RH0** se conseguirá aplicando objetivos sucesivos que forman parte del camino de navegación global $\mathbf{Q_r}$ (obtenido con el **TCG**, ver 4.4), como entradas al algoritmo **UPA** de planificación local del paso (ver 3.5). Una vez obtenidos los resultados del **UPA** que serán caminos factibles de equilibrio para las configuraciones del **CM** y los pies del humanoide (i.e, $\mathbf{Q_{rCM}}$, $\mathbf{Q_{rZ}}$ y $\mathbf{Q_{rD}}$), podremos generar el movimiento completo del humanoide resolviendo con las técnicas de Lie (ver 2.3) el problema cinemático inverso de los dos manipuladores que constituyen el **RH0** (según **DCS** 3.4) para cada una de las configuraciones $\mathbf{q_{CMi}}$ que constituyen el camino factible $\mathbf{Q_{rCM}}$.

El problema queda reducido a determinar de una forma general una solución del problema cinemático inverso del humanoide **RH0** para una configuración del **CM** $\mathbf{q_{CM}}$ (i.e., matriz homogénea $\mathbf{g_{sh}(\theta)}$ {RobotMan-f(2):**BodyGenCoor2homogeneous**}), puesto que teniendo esa solución general el problema global se resolverá como una sucesión de soluciones generales para cada una de las configuraciones $\mathbf{g_{sh}(\theta)}$ que constituyen el camino factible del centro de masas $\mathbf{Q_{rCM}}$.

En los ejemplos A.1 y A.2, mostramos con detalle cómo con el **POE** de álgebras de Lie (ver 2.3.3) es sencillo resolver la mecánica de robots manipuladores. Aplicando las mismas técnicas, resolvemos el problema cinemático inverso del **RH0**, solucionando los problemas del manipulador izquierdo y derecho que lo modelan según **DCS**. Los valores de medidas reales del **RH0** pueden encontrarse en el apéndice B.1.

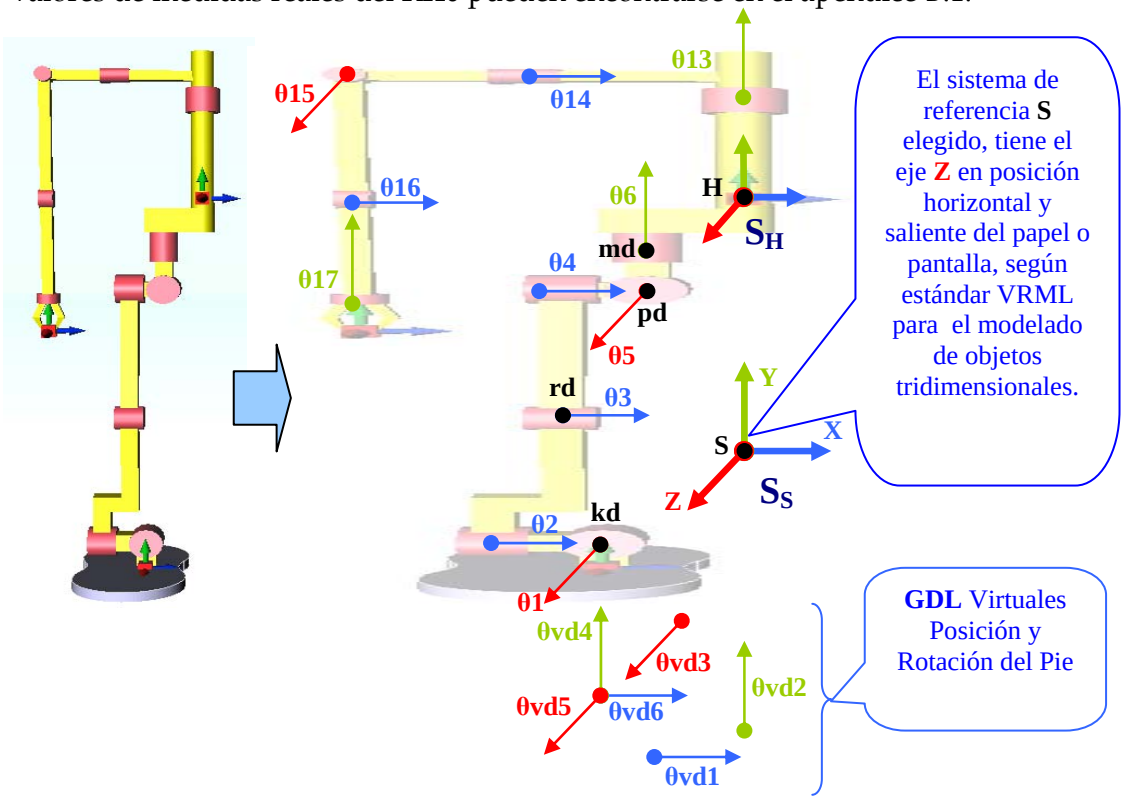


Figura 3-14: Esquema cinemático del manipulador derecho MD según modelo del RH0.

Comenzaremos con el **MANIPULADOR DERECHO (MD)** del **RH0**. La única información necesaria es el esquema cinemático del robot (ver Figura 3-14), que está formado por: seis **GDL** virtuales (**θvd1** a **θvd3** de traslación y **θvd4** a **θvd6** de rotación) correspondientes a la posición y orientación del pie, seis **GDL** reales (**θ1** a **θ6**) de las articulaciones mecánicas de rotación de la pierna y cinco **GDL** reales (**θ13** a **θ17**) correspondientes a las articulaciones mecánicas de rotación del tronco y brazo. El problema se descompone en dos partes principales:

- La parte más compleja consiste en resolver el problema cinemático inverso para obtener los **GDL** virtuales y los reales de la pierna (**θvd1** a **θvd6** y **θ1** a **θ6**) que hacen que el **CM** cumpla con la configuración deseada del centro de masas $\mathbf{g_{sh}(\theta)}$. Estos **GDL** son los únicos que afectan directamente a la locomoción bípeda del humanoide.
- La segunda parte consiste en obtener los **GDL** del brazo (**θ14**, **θ15** y a **θ16**) que contribuyen a mantener el **CM** lo más cercano posible a su posición geométrica natural, esto es, la dada cuando el robot se encuentra en posición firme de equilibrio estático. Los **GDL** del brazo se utilizan para mantener una posición más equilibrada del humanoide en todo momento, facilitando así el control, pero no generan directamente locomoción. No utilizaremos las articulaciones **θ13** y **θ17** puesto que no afectan al movimiento estudiado.

MD Paso UNO - Formulación del Problema Cinemático Directo:

- Los ejes de aplicación de los **GDL** vienen expresados según "(3-10)".

$$\upsilon_{vd1}=\begin{bmatrix}1\\0\\0\end{bmatrix};\ \upsilon_{vd2}=\begin{bmatrix}0\\1\\0\end{bmatrix};\ \upsilon_{vd3}=\begin{bmatrix}0\\0\\1\end{bmatrix};\ \omega_{vd4}=\begin{bmatrix}0\\1\\0\end{bmatrix};\ \omega_{vd5}=\begin{bmatrix}0\\0\\1\end{bmatrix};\ \omega_{vd6}=\begin{bmatrix}1\\0\\0\end{bmatrix} \quad \textbf{(3-10)}$$

$$\omega_1=\begin{bmatrix}0\\0\\1\end{bmatrix};\ \omega_2=\begin{bmatrix}1\\0\\0\end{bmatrix};\ \omega_3=\begin{bmatrix}1\\0\\0\end{bmatrix};\ \omega_4=\begin{bmatrix}1\\0\\0\end{bmatrix};\ \omega_5=\begin{bmatrix}0\\0\\1\end{bmatrix};\ \omega_6=\begin{bmatrix}0\\1\\0\end{bmatrix}$$

- Los valores de los *twists* ξ para cada una de los **GDL** quedan formulados de una forma sencilla por "(3-11)" {RobotMan-f(18):**prismatictwist**} para las de traslación y {RobotMan-f(19):**revolute*twist***} para las de revolución.

$$\xi_{vd1}=\begin{bmatrix}\upsilon_{vd1}\\0\end{bmatrix};\ \xi_{vd2}=\begin{bmatrix}\upsilon_{vd2}\\0\end{bmatrix};\ \xi_{vd3}=\begin{bmatrix}\upsilon_{vd3}\\0\end{bmatrix}$$

$$\xi_{vd4}=\begin{bmatrix}-\omega_{vd4}\times S\\ \omega_{vd4}\end{bmatrix};\ \xi_{vd5}=\begin{bmatrix}-\omega_{vd5}\times S\\ \omega_{vd5}\end{bmatrix};\ \xi_{vd6}=\begin{bmatrix}-\omega_{vd6}\times S\\ \omega_{vd6}\end{bmatrix} \quad \textbf{(3-11)}$$

$$\xi_1=\begin{bmatrix}-\omega_1\times kd\\ \omega_1\end{bmatrix};\ \xi_2=\begin{bmatrix}-\omega_2\times kd\\ \omega_2\end{bmatrix};\ \xi_3=\begin{bmatrix}-\omega_3\times rd\\ \omega_3\end{bmatrix};\ \xi_4=\begin{bmatrix}-\omega_4\times pd\\ \omega_4\end{bmatrix};\ \xi_5=\begin{bmatrix}-\omega_5\times pd\\ \omega_5\end{bmatrix};\ \xi_6=\begin{bmatrix}-\omega_6\times pd\\ \omega_6\end{bmatrix}$$

- Siendo la fórmula para la transformación entre **H** y **S** en la configuración de referencia del manipulador $\mathbf{g_{sh}(0)}$ muy sencilla de definir "(3-12)".

$$g_{sh}(0)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & H-S\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad \textbf{(3-12)}$$

- El problema cinemático directo se soluciona con la simple aplicación del **POE** "(3-13)" {RobotMan-f(3):**forwardkinematics**} que nos da el valor de $\mathbf{g_{sh}(\theta)}$. Nótese que el tratamiento matemático de la posición del pie (i.e., traslaciones **θvd1** a **θvd3**) es el mismo que el de las articulaciones de rotación.

$$g_{sh}(\theta)=e^{\xi_{vd1}\hat{}\,\theta_{vd1}}\cdots e^{\xi_{vd6}\hat{}\,\theta_{vd6}}\cdot e^{\xi_1\hat{}\,\theta_1}\cdots e^{\xi_6\hat{}\,\theta_6}\cdot g_{sh}(0) \qquad \textbf{(3-13)}$$

MD Paso DOS - Obtención de las variables θvd1 a θvd6: En la ejecución del **UPA** (ver 3.5) obtuvimos el camino factible para el pie derecho del humanoide "(5-3)" $\mathbf{Q_{rD}}$, que está constituido por configuraciones $\mathbf{q_{Di}}$, cada una de las cuales se corresponde con una configuración $\mathbf{q_{CMi}}$ (i.e., matriz homogénea $\mathbf{g_{sh}(\theta)}$) del camino factible del centro de masas $\mathbf{Q_{rCM}}$ (ver {RobotMan-f(2):**BodyGenCoor2homogeneous**} para la conversión de $\mathbf{q_{CM}}$ a $\mathbf{g_{sh}(\theta)}$ y {RobotMan-f(4):**homogeneous2BodyGenCoor**} para la conversión de $\mathbf{g_{sh}(\theta)}$ a $\mathbf{q_{CM}}$). Dicho de otro modo, la planificación de la trayectoria de los pies del humanoide realizada por el algoritmo **UPA** (ver 3.3, 3.5 y en detalle 3.3.2), nos da para cada configuración $\mathbf{g_{sh}(\theta)}$ los valores correspondientes a la posición y orientación del pie (i.e., **θvd1** a **θvd6**), como componentes de la configuración $\mathbf{q_{Di}}$ "(3-14)".

$$g_{sh}(\theta)\equiv q_{CMi}\in Q_{rCM}\Rightarrow q_{Di}=\{\theta_{vd1},\theta_{vd2},\theta_{vd3},\theta_{vd4},\theta_{vd5},\theta_{vd6},\}\in Q_{rD} \qquad \textbf{(3-14)}$$

MD Paso TRES - Obtención de θ_3: Para ello, pasamos $\mathbf{g_{sh}(0)}$ y las exponenciales de los **GDL** virtuales (**θvd1** a **θvd6**) al otro lado de la ecuación "(3-13)" y aplicamos ambos lados de esta ecuación al punto **pd**, para posteriormente hallar el módulo de la diferencia de esos términos con el punto **kd**, con lo que obtenemos la ecuación "(3-15)", que presenta las siguientes características:

- Las exponenciales de las variables $\mathbf{\theta_6}$, $\mathbf{\theta_5}$ y $\mathbf{\theta_4}$ representan geométricamente tres movimientos rotativos (tipo *screw*) aplicados sucesivamente a un punto **pd**, que se encuentra situado en los ejes de los tres *twists* ξ_6, ξ_5 y ξ_4 correspondientes. De modo que por aplicación de la propiedad "(2-14)" (no afectación del giro sobre su propio eje), para cualesquiera valores de las variables el punto **pd** no quedará afectado. La consecuencia matemática es que las variables $\mathbf{\theta_6}$, $\mathbf{\theta_5}$ y $\mathbf{\theta_4}$, así como sus exponenciales, se pueden eliminar, puesto que no afectan.

- Las exponenciales de las variables $\mathbf{\theta_1}$ y $\mathbf{\theta_2}$ representan geométricamente dos movimientos rotativos (tipo *screw*) aplicados sucesivamente a un punto dado por el producto de la exponencial de $\mathbf{\theta_3}$ por el punto **pd**. Posteriormente se

obtiene la norma de la diferencia entre el resultado de esas operaciones y un punto **kd** que se encuentra situado en los ejes de los dos *twists* ξ_1 y ξ_2. De modo que por aplicación de la de la propiedad "(2-15)" (conservación de la norma), para cualesquiera valores de las variables, la distancia entre el producto de la exponencial de θ_3 por el punto **pd** y el punto **kd** no quedará afectada. La consecuencia matemática es que las variables θ_1 y θ_2, así como sus exponenciales, se pueden eliminar, puesto que no afectan.

Por todo ello, la ecuación "(3-15)" tiene el término izquierdo conocido (valor **δ**) y un término derecho que sólo se ve afectado por el tercer **GDL**, por lo que queda transformada en la ecuación "(3-16)" que no es sino la formulación del **problema canónico Paden-Kahan-TRES**, por lo que por aplicación geométrica de éste, obtenemos directamente los dos valores posibles para θ_3 con la función "(2-35)".

$$\left\| e^{-\xi_{vd6}\hat{}\,\theta_{vd6}} \cdots e^{-\xi_{vd1}\hat{}\,\theta_{vd1}} \cdot g_{sh}(\theta) \cdot g_{sh}(0)^{-1} \cdot pd - kd \right\| = \left\| e^{\xi_1\hat{}\,\theta_1} \cdots e^{\xi_6\hat{}\,\theta_6} \cdot pd - kd \right\| \quad \textbf{(3-15)}$$

$$\delta = \left\| e^{\xi_3\hat{}\,\theta_3} \cdot pd - kd \right\| \xrightarrow{Paden-Kahan-TRES} \theta_3 \quad Doble \quad \textbf{(3-16)}$$

MD Paso CUATRO - Obtención de las variables θ_1 y θ_2: Pasamos $\mathbf{g_{sh}(0)}$ y las exponenciales de los **GDL** virtuales (**θvd1** a **θvd6**) al otro lado de la ecuación "(3-13)" y aplicamos ambos lados de esta ecuación al punto **pd**, para obtener "(3-17)" que tiene el término izquierdo conocido (igual a un punto cualquiera que llamaremos k'), y un término derecho que no se ve afectado por las variables θ_6, θ_5 y θ_4, debido a la propiedad "(2-14)", al estar el punto **pd** sobre los ejes de los tres *twists* ξ_6, ξ_5 y ξ_4. El producto de la exponencial de θ_3 por el punto **pd** será también un valor conocido (igual a un punto que llamaremos p'), al haber obtenido θ_3 en el tercer paso. De esta forma, la ecuación "(3-17)" se transforma en la ecuación "(3-18)", que no es sino la formulación del **problema canónico Paden-Kahan-DOS**, por lo que por aplicación geométrica de éste "(2-31)" obtenemos los dos valores posibles para la pareja de variables θ_2 y θ_1.

$$e^{-\xi_{vd6}\hat{}\,\theta_{vd6}} \cdots e^{-\xi_{vd1}\hat{}\,\theta_{vd1}} \cdot g_{sh}(\theta) \cdot g_{sh}(0)^{-1} \cdot pd = e^{\xi_1\hat{}\,\theta_1} \cdots e^{\xi_6\hat{}\,\theta_6} \cdot pd \quad \textbf{(3-17)}$$

$$k' = e^{\xi_1\hat{}\,\theta_1} \cdot e^{\xi_2\hat{}\,\theta_2} \cdot p' \xrightarrow{PadenKahan-DOS} \theta_1, \theta_2 \quad Doble \quad \textbf{(3-18)}$$

MD Paso QUINTO – Obtención de las variables θ_4 y θ_5: Para ello, pasamos $\mathbf{g_{sh}(0)}$, y las exponenciales ya conocidas (las de los **GDL** virtuales **θvd1** a **θvd6** y los tres primeros **GDL** reales θ_1, θ_2 y θ_3) al otro lado de la ecuación "(3-13)" y aplicamos ambos lados de esta ecuación al punto **md** para obtener la ecuación "(3-19)", que tiene el término izquierdo conocido (igual a un punto cualquiera que llamamos k'') y un término derecho que no se ve afectado por la variable θ_6, debido a la propiedad "(2-14)" al estar el punto **md** sobre el eje del *twist* ξ_6. De esta forma, la ecuación "(3-19)" queda transformada en la ecuación "(3-20)", que es de nuevo la formulación del **problema canónico Paden-Kahan-DOS**, por lo que por aplicación geométrica "(2-31)" obtenemos los dos valores posibles para la pareja de variables θ_4 y θ_5.

$$e^{-\xi_3\hat{}\theta_3}\cdots e^{-\xi_1\hat{}\theta_1}\cdot e^{-\xi_{vd6}\hat{}\theta_{vd6}}\cdots e^{-\xi_{vd1}\hat{}\theta_{vd1}}\cdot g_{sh}(\theta)\cdot g_{sh}(0)^{-1}\cdot md = e^{\xi_4\hat{}\theta_4}\cdot e^{\xi_5\hat{}\theta_5}\cdot e^{\xi_6\hat{}\theta_6}\cdot md \quad \textbf{(3-19)}$$

$$k'' = e^{\xi_4\hat{}\theta_4}\cdot e^{\xi_5\hat{}\theta_5}\cdot md \xrightarrow{PadenKahan\ -DOS} \theta_4,\theta_5 \quad Doble \quad \textbf{(3-20)}$$

MD Paso SEXTO – Obtención de la variable θ_6: Para ello, pasamos $\mathbf{g_{sh}(0)}$ y todas las exponenciales ya conocidas (**θvd1**, **θvd2**, **θvd3**, **θvd4**, **θvd5**, **θvd6**, θ_1, θ_2, θ_3, θ_4 y θ_5) al otro lado de la ecuación "(3-13)" y aplicamos ambos lados de esta ecuación al punto **S**, para obtener la ecuación "(3-21)", que tiene el término izquierdo conocido (igual a un punto cualquiera que llamamos k'''.) De esta forma, la ecuación queda transformada en la ecuación "(3-22)", que es la formulación del **problema canónico Paden-Kahan-UNO**, por lo que por aplicación geométrica directa de éste "(2-29)" obtenemos el único valor posible para la variable θ_6.

$$e^{-\xi_5\hat{}\theta_5}\cdots e^{-\xi_1\hat{}\theta_1}\cdot e^{-\xi_{vd6}\hat{}\theta_{vd6}}\cdots e^{-\xi_{vd1}\hat{}\theta_{vd1}}\cdot g_{sh}(\theta)\cdot g_{sh}(0)^{-1}\cdot S = e^{\xi_6\hat{}\theta_6}\cdot S \quad \textbf{(3-21)}$$

$$k''' = e^{\xi_6\hat{}\theta_6}\cdot S \xrightarrow{PadenKahan\ -UNO} \theta_6 \quad Simple \quad \textbf{(3-22)}$$

MD Paso SÉPTIMO – Obtención de las variables θ_{14} a θ_{16}: Estos son los **GDL** mecánicos del brazo derecho, que deben contribuir a mantener una posición más equilibrada del humanoide en todo momento, facilitando así el control. Podrían dejarse estos valores incluso a cero sin impedir una locomoción bípeda del robot, pero una buena solución supone el control dinámico de las inercias de los miembros, lo que queda fuera del alcance de este apartado de soluciones cinemáticas (ver 3.4.4 para la dinámica). Sin embargo, presentamos una solución práctica muy sencilla que produce unos resultados realmente buenos, mejorando el equilibrio y la locomoción bípeda. La idea es utilizar el concepto de coordinación sagital (ver modelo **DCS** 3.4) entre los miembros del humanoide, esto es tan sencillo como hacer que los **GDL** del brazo derecho sean iguales a algunos **GDL** de la pierna izquierda (que obtendremos seguidamente para el manipulador izquierdo), de la forma: clavícula derecha igual a cadera izquierda en el eje X, hombro derecho igual a cadera izquierda en el eje Z y codo derecho igual a rodilla izquierda. Así nos quedan los valores de "(3-23)".

$$\theta_{14} = \theta_9 \quad ; \quad \theta_{15} = \theta_8 \quad ; \quad \theta_{16} = \theta_{10} \quad \textbf{(3-23)}$$

De esta forma queda resuelto de forma geométrica, cerrada y completa el problema cinemático inverso de la pierna derecha para el posicionamiento del **CM**, o lo que es lo mismo del tronco del humanoide. Lo que es más, dados los valores requeridos para la posición y orientación del pie (**θvd1** a **θvd6**), no sólo se ha encontrado una solución para el problema (i.e., un conjunto de valores θ_1, θ_2, θ_3, θ_4 , θ_5 y θ_6), sino que de existir se han resuelto en una sola formulación las ocho posibles soluciones de este problema (i.e., los ocho conjuntos de valores para θ_1, θ_2, θ_3, θ_4 , θ_5 y θ_6) que son el límite teórico para una pierna como la del humanoide **RH0** con seis **GDL** mecánicos. Obsérvese para ello en "(3-24)" las combinaciones posibles de las soluciones obtenidas.

$$Soluciones\_MD = \theta_3 Doble \times \theta_2\theta_1 Doble \times \theta_5\theta_4 Doble \times \theta_6 Simple = 8 \quad \textbf{(3-24)}$$

Ahora resolveremos el **MANIPULADOR IZQUIERDO MZ** del **RH0**. El procedimiento es exactamente el mismo que el que acabamos de desarrollar para el manipulador derecho, por lo que lo veremos con algo menos de detalle. La única información necesaria es el esquema cinemático del robot (ver Figura 3-15), que está formado por: seis **GDL** virtuales (**θvz1** a **θvz3** de traslación y **θvz4** a **θvz6** de rotación), correspondientes a la posición y orientación del pie izquierdo, seis **GDL** reales (**θ12** al **θ7**) de las articulaciones mecánicas de rotación de la pierna y cinco **GDL** reales (**θ13** y **θ18** a **θ21**) correspondientes a las articulaciones mecánicas de rotación del tronco y brazo. El problema se descompone en dos partes principales:

- La más importante consiste en resolver el problema cinemático inverso para obtener los **GDL** virtuales y los reales de la pierna (**θvz1** a **θvz6** y **θ12** a **θ7**), que hacen que el **CM** cumpla con la configuración deseada del centro de masas $\mathbf{g_{sh}(\theta)}$. Estos **GDL** son los únicos que afectan directamente a la locomoción bípeda del humanoide.
- La segunda parte consiste en obtener los **GDL** del brazo (**θ18**, **θ19** y **θ20**), que contribuyen a mantener el **CM** lo más cercano posible a su posición geométrica natural, esto es, la dada cuando el robot se encuentra en posición firme de equilibrio estático. Los **GDL** del brazo se utilizan para mantener una posición más equilibrada del humanoide en todo momento, facilitando así el control, pero no generan directamente la locomoción bípeda. No usamos las articulaciones **θ13** y **θ21** porque no afectan a la locomoción estudiada.

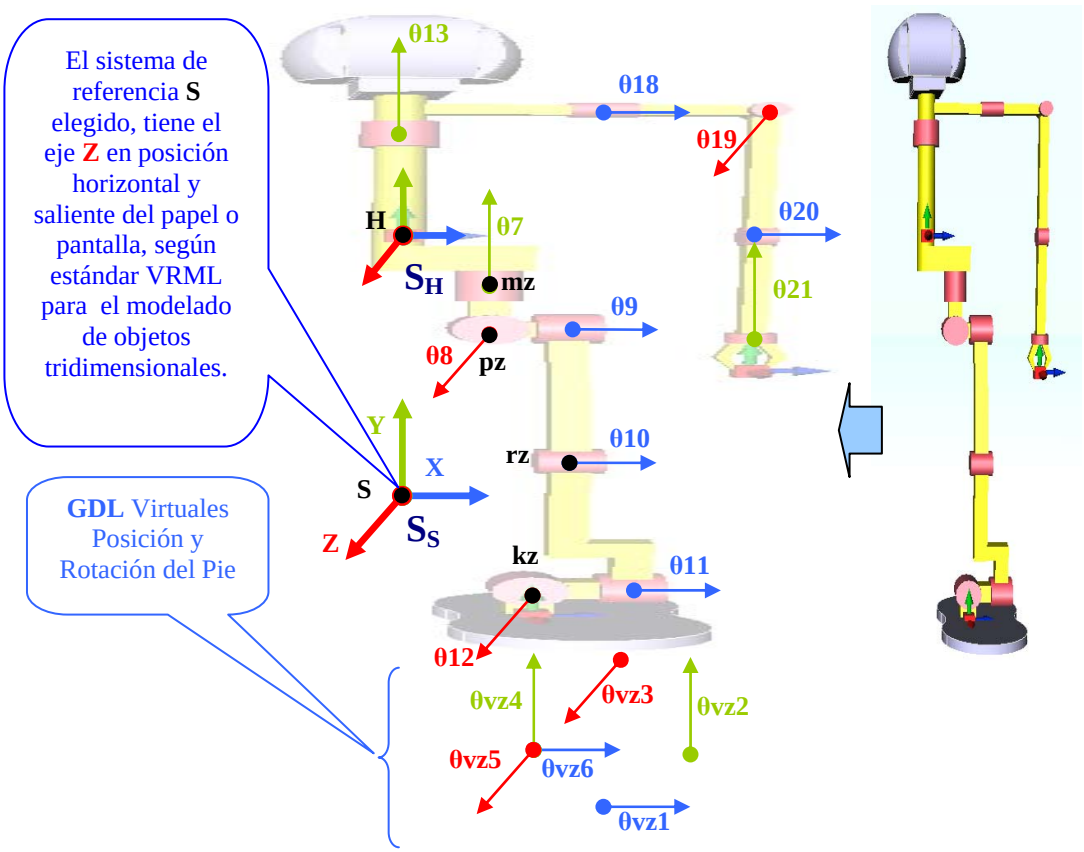


Figura 3-15: Esquema cinemático del manipulador izquierdo MZ según modelo del RH0.

MZ Paso UNO - Formulación del Problema Cinemático Directo:

- Los ejes de aplicación de los **GDL** vienen expresados según "(3-25)".

$$\left.\begin{array}{l} \upsilon_{vz1}=\begin{bmatrix}1\\0\\0\end{bmatrix};\ \upsilon_{vz2}=\begin{bmatrix}0\\1\\0\end{bmatrix};\ \upsilon_{vz3}=\begin{bmatrix}0\\0\\1\end{bmatrix};\ \omega_{vz4}=\begin{bmatrix}0\\1\\0\end{bmatrix};\ \omega_{vz5}=\begin{bmatrix}0\\0\\1\end{bmatrix};\ \omega_{vz6}=\begin{bmatrix}1\\0\\0\end{bmatrix} \\ \omega_{12}=\begin{bmatrix}0\\0\\1\end{bmatrix};\ \omega_{11}=\begin{bmatrix}1\\0\\0\end{bmatrix};\ \omega_{10}=\begin{bmatrix}1\\0\\0\end{bmatrix};\ \omega_{9}=\begin{bmatrix}1\\0\\0\end{bmatrix};\ \omega_{8}=\begin{bmatrix}0\\0\\1\end{bmatrix};\ \omega_{7}=\begin{bmatrix}0\\1\\0\end{bmatrix} \end{array}\right\} \quad (3\text{-}25)$$

- Los *twists* ξ quedan formulados por "(3-26)". {RobotMan-f(18):**prismatictwist**} para la de translación y {RobotMan-f(19):**revolute*twist***} para las de revolución.

$$\left.\begin{array}{c} \xi_{vz1}=\begin{bmatrix}\upsilon_{vz1}\\0\end{bmatrix};\ \xi_{vz2}=\begin{bmatrix}\upsilon_{vz2}\\0\end{bmatrix};\ \xi_{vz3}=\begin{bmatrix}\upsilon_{vz3}\\0\end{bmatrix} \\ \xi_{vz4}=\begin{bmatrix}-\omega_{vz4}\times S\\ \omega_{vz4}\end{bmatrix};\ \xi_{vz5}=\begin{bmatrix}-\omega_{vz5}\times S\\ \omega_{vz5}\end{bmatrix};\ \xi_{vd6}=\begin{bmatrix}-\omega_{vz6}\times S\\ \omega_{vz6}\end{bmatrix} \\ \xi_{12}=\begin{bmatrix}-\omega_{12}\times kz\\ \omega_{12}\end{bmatrix};\ \xi_{11}=\begin{bmatrix}-\omega_{11}\times kz\\ \omega_{11}\end{bmatrix};\ \xi_{10}=\begin{bmatrix}-\omega_{10}\times rz\\ \omega_{10}\end{bmatrix} \\ \xi_{9}=\begin{bmatrix}-\omega_{9}\times pz\\ \omega_{9}\end{bmatrix};\ \xi_{8}=\begin{bmatrix}-\omega_{8}\times pz\\ \omega_{8}\end{bmatrix};\ \xi_{7}=\begin{bmatrix}-\omega_{7}\times pz\\ \omega_{7}\end{bmatrix} \end{array}\right\} \quad (3\text{-}26)$$

- La transformación de **H** a **S** en la configuración de referencia $\mathbf{g_{sh}(0)}$, es "(3-27)".

$$g_{sh}(0)=\begin{bmatrix}1&0&0&0\\0&1&0&H-S\\0&0&1&0\\0&0&0&1\end{bmatrix} \quad (3\text{-}27)$$

- El problema cinemático directo del **MZ** se soluciona con la aplicación del **POE** "(3-28)" {RobotMan-f(3):**forwardkinematics**} para el valor de $\mathbf{g_{sh}(\theta)}$.

$$g_{sh}(\theta)=e^{\xi_{vz1}\hat{}\,\theta_{vz1}}\cdots e^{\xi_{vz6}\hat{}\,\theta_{vz6}}\cdot e^{\xi_{12}\hat{}\,\theta_{12}}\cdots e^{\xi_{7}\hat{}\,\theta_{7}}\cdot g_{sh}(0) \quad (3\text{-}28)$$

MZ Paso DOS - Obtención de las variables θvz1 a θvzd6: En la ejecución del **UPA** (ver 3.5) se obtiene el camino factible para el pie izquierdo "(5-2)" $\mathbf{Q_{rZ}}$, formado por configuraciones $\mathbf{q_{Zi}}$, cada una de las cuales se corresponde con otra $\mathbf{q_{CMi}}$ (i.e., **gsh(θ)**) del camino factible del **CM** $\mathbf{Q_{rCM}}$. El algoritmo **UPA** (ver 3.3, 3.5 y en detalle 3.3.2), nos da para cada configuración $\mathbf{g_{sh}(\theta)}$ los valores correspondientes a la posición y orientación del pie (i.e., **θvz1** a **θvz6**), como componentes de la configuración $\mathbf{q_{Zi}}$ "(3-29)".

$$g_{sh}(\theta)\equiv q_{CMi}\in Q_{rCM}\Rightarrow q_{Zi}=\{\theta_{vz1},\theta_{vz2},\theta_{vz3},\theta_{vz4},\theta_{vz5},\theta_{vz6},\}\in Q_{rZ} \quad (3\text{-}29)$$

MZ Paso TRES - Obtención de θ_{10}: Pasamos $\mathbf{g_{sh}(0)}$ y las exponenciales de **θvz1** a **θvz6** al otro lado de la ecuación "(3-28)" y aplicamos ambos lados al punto **pz**, para posteriormente hallar el módulo de la diferencia de esos términos con el punto **kz**, con lo que obtendremos la ecuación "(3-30)", que presenta las siguientes características:

- Las exponenciales de θ_9, θ_8 y θ_7 son tres rotaciones aplicadas sucesivamente a un punto **pz** que se encuentra situado en los ejes de los tres *twists* ξ_9, ξ_8 y ξ_7, de modo que por la propiedad "(2-14)" el punto **pz** no quedará afectado. En consecuencia, las variables θ_9, θ_8 y θ_7 se pueden eliminar, puesto que no afectan.

- Las exponenciales de las variables θ_{12} y θ_{11} son dos rotaciones aplicadas sucesivamente a un punto dado por el producto de la exponencial de θ_{10} por el punto **pz.** Posteriormente se obtiene la norma de la diferencia entre ese resultado y un punto **kz** que se encuentra en los ejes de los *twists* ξ_{12} y ξ_{11}, de modo que por aplicación de la de la propiedad "(2-15)" la distancia entre el producto de la exponencial de θ_{10} por el punto **pz** y el punto **kz** no quedará afectada. En consecuencia las variables θ_{12} y θ_{11} se pueden eliminar.

Por todo ello, la ecuación "(3-30)" tiene el término izquierdo conocido (valor **δ**) y un término derecho que sólo se ve afectado por θ_{10}, por lo que nos queda "(3-31)", que es la formulación del **problema canónico Paden-Kahan-TRES**, por lo que obtenemos directamente los dos valores posibles para θ_{10} con la función "(2-35)".

$$\left\| e^{-\xi_{vz6}\hat{}\,\theta_{vz6}} \cdots e^{-\xi_{vz1}\hat{}\,\theta_{vz1}} \cdot g_{sh}(\theta) \cdot g_{sh}(0)^{-1} \cdot pz - kz \right\| = \left\| e^{\xi_{12}\hat{}\,\theta_{12}} \cdots e^{\xi_7\hat{}\,\theta_7} \cdot pz - kz \right\| \quad \textbf{(3-30)}$$

$$\delta = \left\| e^{\xi_{10}\hat{}\,\theta_{10}} \cdot pz - kz \right\| \xrightarrow{Paden-Kahan-TRES} \theta_{10} \quad Doble \quad \textbf{(3-31)}$$

MZ Paso CUATRO - Obtención de las variables θ_{12} y θ_{11}: Pasamos $\mathbf{g_{sh}(0)}$ y las exponenciales de **θvz1** a **θvz6** al otro lado de la ecuación "(3-28)" y aplicamos ambos lados al punto **pz** para obtener "(3-32)", que tiene el término izquierdo conocido (igual a un punto k'), y un término derecho que no se ve afectado por las variables θ_9, θ_8 y θ_7, debido a la propiedad "(2-14)", al estar el punto **pz** sobre los ejes de los *twists* ξ_9, ξ_8 y ξ_7. El producto de la exponencial de θ_{10} por el punto **pz** es conocido (igual a un punto p'). De esta forma, la ecuación "(3-32)" se transforma en "(3-33)" que es la formulación del **problema canónico Paden-Kahan-DOS**, por lo que por aplicación de "(2-31)" obtenemos los dos valores posibles para la pareja de variables θ_{12} y θ_{11}.

$$e^{-\xi_{vz6}\hat{}\,\theta_{vz6}} \cdots e^{-\xi_{vz1}\hat{}\,\theta_{vz1}} \cdot g_{sh}(\theta) \cdot g_{sh}(0)^{-1} \cdot pz = e^{\xi_{12}\hat{}\,\theta_{12}} \cdots e^{\xi_7\hat{}\,\theta_7} \cdot pz \quad \textbf{(3-32)}$$

$$k' = e^{\xi_{12}\hat{}\,\theta_{12}} \cdot e^{\xi_{11}\hat{}\,\theta_{11}} \cdot p' \xrightarrow{PadenKahan-DOS} \theta_{12}, \theta_{11} \quad Doble \quad \textbf{(3-33)}$$

MZ Paso QUINTO – Obtención de las variables θ_9 y θ_8: Para ello, pasamos $\mathbf{g_{sh}(0)}$ y las exponenciales ya conocidas (**θvz1** a **θvz6,** θ_{12}, θ_{11} y θ_{10}) al otro lado de la ecuación "(3-28)" y aplicamos ambos lados al punto **mz**, para obtener la ecuación "(3-34)", que tiene el término izquierdo conocido (igual k'') y un término derecho que no se ve afectado por la variable θ_7, debido a la propiedad "(2-14)", al estar el punto **mz** sobre

el eje del *twist* ξ_7. La ecuación "(3-34)" queda transformada en "(3-35)" que es la formulación del **problema canónico Paden-Kahan-DOS**, por lo que por aplicación geométrica "(2-31)" obtenemos los valores para las variables θ_9 y θ_8.

$$e^{-\xi_{10}\hat{}\theta_{10}}\cdots e^{-\xi_{12}\hat{}\theta_{12}}\cdot e^{-\xi_{vz6}\hat{}\theta_{vz6}}\cdots e^{-\xi_{vz1}\hat{}\theta_{vz1}}\cdot g_{sh}(\theta)\cdot g_{sh}(0)^{-1}\cdot mz = e^{\xi_9\hat{}\theta_9}\cdot e^{\xi_8\hat{}\theta_8}\cdot e^{\xi_7\hat{}\theta_7}\cdot mz \quad \textbf{(3-34)}$$

$$k'' = e^{\xi_9\hat{}\theta_9}\cdot e^{\xi_8\hat{}\theta_8}\cdot mz \xrightarrow{PadenKahan-DOS} \theta_9,\theta_8 \quad Doble \quad \textbf{(3-35)}$$

MZ Paso SEXTO – Obtención de la variable θ_7: Para ello, pasamos $\mathbf{g_{sh}(0)}$ y todas las exponenciales ya conocidas (**θvz1** a **θvz6** y θ_{12} a θ_8) al otro lado de la ecuación "(3-28)" y aplicamos ambos lados de esta ecuación al punto **S**, para obtener "(3-36)" que tiene el término izquierdo conocido (igual a un punto k'''), de forma que nos queda la ecuación "(3-37)", que es el **problema canónico Paden-Kahan-UNO**, por lo que por aplicación de "(2-29)" obtenemos el único valor posible para la variable θ_7.

$$e^{-\xi_8\hat{}\theta_8}\cdots e^{-\xi_{12}\hat{}\theta_{12}}\cdot e^{-\xi_{vz6}\hat{}\theta_{vz6}}\cdots e^{-\xi_{vz1}\hat{}\theta_{vz1}}\cdot g_{sh}(\theta)\cdot g_{sh}(0)^{-1}\cdot S = e^{\xi_7\hat{}\theta_7}\cdot S \quad \textbf{(3-36)}$$

$$k''' = e^{\xi_7\hat{}\theta_7}\cdot S \xrightarrow{PadenKahan-UNO} \theta_7 \quad Simple \quad \textbf{(3-37)}$$

MZ Paso SÉPTIMO – Obtención de las variables θ_{18} a θ_{20}: Estos son los **GDL** del brazo derecho que deben contribuir a la posición equilibrada del humanoide, facilitando así el control. Podrían anularse estos valores sin impedir una locomoción bípeda, pero una buena solución supone control dinámico de inercias, lo que queda fuera del alcance de la solución cinemática aquí presentada (ver 3.4.4 para dinámica). Presentamos una solución práctica que produce resultados muy buenos, cuya idea es utilizar el concepto de coordinación sagital entre los miembros del humanoide (ver **DCS** 3.4). Esto es tan sencillo como hacer que algunos **GDL** del brazo izquierdo sean iguales a otros **GDL** de la pierna derecha (que ya obtuvimos anteriormente del manipulador derecho), de la siguiente forma: clavícula izquierda igual a cadera derecha en el eje X, hombro izquierdo igual a cadera derecha en el eje Z y codo izquierdo igual a rodilla derecha. De ese modo nos quedan los valores dadas en "(3-38)" para las variables θ_{18}, θ_{19} y θ_{20}.

$$\theta_{18} = \theta_4 \quad ; \quad \theta_{19} = \theta_5 \quad ; \quad \theta_{20} = \theta_3 \quad \textbf{(3-38)}$$

De esta forma queda resuelto de forma geométrica, cerrada y completa el problema cinemático inverso de la pierna izquierda para el posicionamiento del **CM**, o lo que es lo mismo del tronco del humanoide. Lo que es más, dados los valores requeridos para la posición y orientación del pie (**θvz1** a **θvz6**), no sólo se ha encontrado una solución para el problema (i.e., un conjunto de valores θ_{12}, θ_{11}, θ_{10}, θ_9 , θ_8 y θ_7), sino que de existir se han resuelto en una sola formulación las ocho posibles soluciones de este problema (i.e., los ocho conjuntos de valores para θ_{12}, θ_{11}, θ_{10}, θ_9 , θ_8 y θ_7), que son el límite teórico para una pierna como la del humanoide **RH0** con seis **GDL** mecánicos. Obsérvese para ello en "(3-39)" las combinaciones posibles según las soluciones obtenidas.

$$Soluciones\_MZ = \theta_{10}Doble \times \theta_{12}\theta_{11}Doble \times \theta_9\theta_8Doble \times \theta_7Simple = 8 \quad \textbf{(3-39)}$$

Hemos demostrado en este apartado que podemos hallar de forma cerrada, completa y geométrica la solución al problema cinemático inverso del humanoide **RH0**.

3.4.3 Modelo Dinámico DCS del RH0.

El desarrollo completo de un control dinámico para el humanoide **RH0** queda fuera del alcance de esta tesis. Sin embargo, creemos interesante plantear el problema desde el punto de vista de los estudios sobre la dinámica y control de manipuladores con matemática de Lie que realizamos en 2.4 y 2.5, extendiéndolos para humanoides.

En primer lugar aplicaremos el concepto **DCS** para modelar dinámicamente el robot **RH0**. El nuevo modelo **DCS** presentado en 3.2 considera la división mecánica sagital de un humanoide (de forma teórica), para poder analizar más sencillamente los manipuladores (izquierdo y derecho) acoplados por las partes comunes (i.e., pelvis y el tronco), que constituyen el conjunto mecánico del humanoide.

De este modo, el modelo dinámico **DCS** del **RH0** queda constituido por dos robots manipuladores según la Figura 3-16. Cada uno cuenta con seis **actuadores virtuales** que vienen dados por las fuerzas y pares generalizados que la superficie de apoyo ejerce sobre los pies (i.e., **Γvz1** a **Γvz6** para el izquierdo y **Γvd1** a **Γvd6** para el derecho), más **11** actuadores reales que ejercen pares **Γi** sobre las articulaciones mecánicas.

El problema dinámico inverso del **RH0** queda definido según las fórmulas siguientes correspondientes al manipulador izquierdo "(3-40)" y derecho "(3-41)".

$$M_Z(\theta_Z)\ddot{\theta}_Z + C_Z(\theta_Z,\dot{\theta}_Z)\dot{\theta}_Z + N_Z(\theta_Z,\dot{\theta}_Z) = \Gamma_Z \qquad \textbf{(3-40)}$$

$$M_D(\theta_D)\ddot{\theta}_D + C_D(\theta_D,\dot{\theta}_D)\dot{\theta}_D + N_D(\theta_D,\dot{\theta}_D) = \Gamma_D \qquad \textbf{(3-41)}$$

El **CM** es un punto muy importante común a ambos manipuladores, puesto que allí se localiza el sistema de referencia del tronco $\mathbf{S_H}$ ya que nos sirve para desarrollar el movimiento de locomoción bípeda, al hacer que las configuraciones del **CM** $\mathbf{q_{CM}}$ (i.e., $\mathbf{g_{sh}(\theta)}$), recorran el camino factible $\mathbf{Q_{rCM}}$ obtenido por el algoritmo **UPA**. Resolviendo los problemas cinemáticos inversos de los dos manipuladores para cada punto del $\mathbf{Q_{rCM}}$ (como hemos visto en 3.4.2) obtenemos las trayectorias para todas las articulaciones **θ** (i.e., **θvz1** a **θvz6**, **θvd1** a **θvd6** y **θ1** a **θ21**), si además conocemos las velocidades de las articulaciones y las fuerzas generalizadas que ejerce el suelo sobre los pies, podríamos aplicar las ecuaciones "(3-40)" y "(3-41)" para obtener los valores de los pares **Γi** que necesitamos ejercer sobre las articulaciones para obtener la locomoción bípeda.

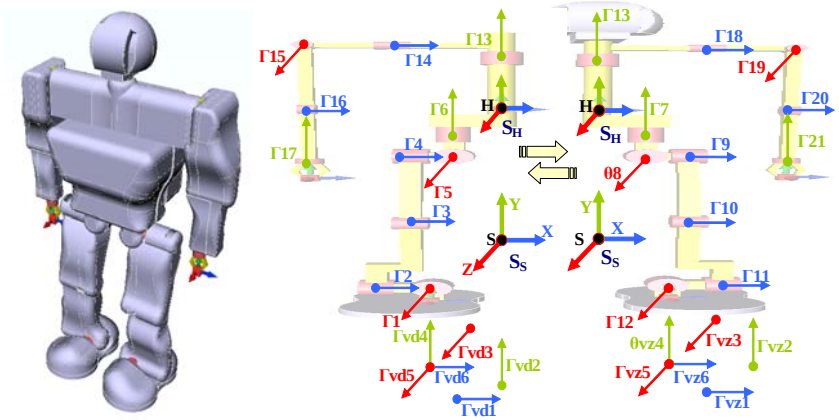


Figura 3-16: Humanoide RH0 y su Modelo dinámico DCS.

3.4.4 Problema Dinámico Inverso y control del RH0.

Consiste en encontrar los pares **Γi** de los actuadores del humanoide que consiguen generar el movimiento de locomoción. Si el modelo dinámico del **RH0** presentado en 3.4.3 fuese perfecto podríamos encontrar **Γi** mediante las ecuaciones de Lagrange "(3-40)" y "(3-41)" para los manipuladores izquierdo y derecho respectivamente. Los valores de medidas e inercias reales del **RH0** pueden encontrarse en el apéndice B.1.

Utilizaremos la matemática de grupos de Lie para definir las ecuaciones de Lagrange (ver 2.4.1 y 2.4.2) en términos de las posiciones **θ** y velocidades de las articulaciones. En la ecuación "(3-40)" para el manipulador izquierdo el primer término incluye $\mathbf{M_Z}$ que es matriz de inercia definida según "(2-39)", el segundo término incluye la matriz $\mathbf{C_Z}$ definida según "(2-42)"que tiene en cuenta los términos centrífugos y de Coriolis y el tercer término incluye la matriz de potencial $\mathbf{N_Z}$ que tiene en cuenta las fuerzas externas de gravedad y fricción. En la ecuación "(3-41)" para el manipulador derecho, los conceptos $\mathbf{M_D}$, $\mathbf{C_D}$ y $\mathbf{N_D}$ se definen de igual forma.

Los atributos dinámicos del **RH0** (i.e., matrices $\mathbf{M_Z}$, $\mathbf{C_Z}$, $\mathbf{M_D}$ y $\mathbf{C_D}$) pueden determinarse con sólo conocer la geometría de los *twists* $\boldsymbol{\xi_i}$ de las articulaciones y los tensores de inercia de los eslabones $\boldsymbol{\Psi_i}$ (existen funciones en la librería **RobotMan** del Apéndice-B para calcularlas.) Nos quedarían por desarrollar $\mathbf{N_Z}$ y $\mathbf{N_D}$, aunque se suelen despreciar las fricciones y nos quedan unas matrices gravitacionales bastante sencillas.

En la práctica, sin embargo, debido a la complejidad del problema (e.g., errores y perturbaciones) es necesario aplicar un sistema de control que corrija las fuerzas aplicadas en respuesta a las desviaciones de la trayectoria deseada. Vamos a exponer un ejemplo de control dinámico en el espacio de las articulaciones por par computado, siguiendo las teorías expuestas (también se podría hacer un desarrollo similar al efectuado en 2.5.2 para el espacio de trabajo). Siendo para el manipulador izquierdo: $\boldsymbol{\theta_{dZ}}$ la trayectoria deseada para las coordenadas generalizadas, $\mathbf{K_{vZ}}$ la matriz de ganancia constante de velocidad, $\mathbf{K_{pZ}}$ la matriz de ganancia constante de posición y $\mathbf{e_Z}$ el error de posición de las articulaciones (i.e., $\mathbf{e_Z=\theta-\theta_{dZ}}$), la ley de control quedaría expresada según la ecuación "(3-42)". Para el manipulador derecho se definen los conceptos simétricos de $\boldsymbol{\theta_{dD}}$, $\mathbf{K_{vD}}$, $\mathbf{K_{pD}}$ y $\mathbf{e_D}$, obteniendo la ley de control dada por la ecuación "(3-43)". Por supuesto hay que destacar que tenemos una restricción muy importante entre las dos leyes de control, que es la dada por el hecho de que el tronco del **RH0** es común a ambos manipuladores, por lo que su posición, velocidad y aceleración debe ser la misma.

$$M_Z(\theta)\left(\ddot{\theta}_{dZ} - K_{vZ}\dot{e}_Z - K_{pZ}e_Z\right) + C_Z\left(\theta_Z, \dot{\theta}_Z\right)\dot{\theta}_Z + N_Z\left(\theta_Z, \dot{\theta}_Z\right) = \Gamma_Z \quad \textbf{(3-42)}$$

$$M_D(\theta)\left(\ddot{\theta}_{dD} - K_{vD}\dot{e}_D - K_{pD}e_D\right) + C_D\left(\theta_D, \dot{\theta}_D\right)\dot{\theta}_D + N_D\left(\theta_D, \dot{\theta}_D\right) = \Gamma_D \quad \textbf{(3-43)}$$

En conclusión, a pesar de que un desarrollo completo queda fuera del alcance de la tesis, hemos presentado una formulación geométrica para el problema dinámico inverso del humanoide **RH0**. La solución es geométrica ya que las ecuaciones del movimiento se expresan como productos matriciales (i.e., conjuntos de **POE**), con la gran ventaja computacional que ello supone.

3.5 APLICACIÓN del Nuevo Algoritmo UPA al Humanoide RH0.

Nos enfrentamos en este apartado al problema de resolver la planificación de movimientos para la locomoción bípeda del humanoide **RH0**. Para ello, aplicaremos el nuevo desarrollo teórico, esto es, el nuevo algoritmo **UPA** presentado en 3.3 de forma general para cualquier humanoide, pero ahora teniendo en cuenta las especificidades del robot **RH0**.

Para cualquier aplicación compleja, el **RH0** deberá desarrollar una locomoción bípeda compuesta por varios pasos junto con probables cambios de dirección, que tiene que ser dirigida por algún método de planificación de trayectorias, en esta tesis, la información correspondiente al camino de navegación global se obtiene con el modelo **TCG** 4.4, que utiliza el nuevo algoritmo **M3R** 4.3. (ver las aplicaciones de éstos para el **RH0** en 5.4.2 y 5.4.3 respectivamente). Obtener una solución cerrada para un movimiento tan complejo resulta muy poco práctico, puesto que supone un movimiento que necesita mucho tiempo para desarrollarse dentro de un entorno perfectamente conocido, cuando en realidad los mapas basados en modelado del entorno son raramente perfectos y la acción puede ser dinámica, con cambios en el número y estructura de los obstáculos. Por lo tanto, resulta más útil desarrollar un movimiento elemental de locomoción bípeda (i.e. un paso), que pueda considerar información sensorial para poder planificar actos reflejos.

Como ya hemos estudiado en 3.1, la mecánica de la locomoción bípeda tiene muchas veces una estructura periódica y que nos permite restringir el problema a la resolución del problema mecánico correspondiente a un único ciclo del paso, esto es, a la resolución de una sola etapa de soporte seguida de una sola etapa de transición. De esta forma, la solución para una locomoción bípeda compleja puede desarrollarse como un conjunto encadenado de soluciones para un único paso, siempre que la solución para éste sea lo suficientemente general para contemplar todos los posibles casos, como son diferentes longitudes, anchuras y altitudes de paso, o cambios en la dirección del movimiento.

Aplicando el nuevo algoritmo "Un Paso Adelante" (**UPA**) para el movimiento de locomoción bípeda del **RH0**, desarrollaremos una solución de propósito general para el problema mecánico consistente en mover el **RH0** un único paso adelante hacia un objetivo dado. El **UPA** recibe como entradas el objetivo local hacia el que se debe caminar (obtenido de la navegación global), el modelo del entorno local (que puede ser mejorado mediante integración sensorial) y las características típicas de la locomoción bípeda (i.e., longitud, anchura y altura de paso, altura del **CM**). Los resultados obtenidos en la ejecución del algoritmo **UPA** son los datos correspondientes, en primer lugar a las trayectorias de los **GDL** virtuales $\boldsymbol{\theta v_{PM}}$ del pie móvil (**PM**) durante el ciclo del paso (i.e., trayectoria de la posición y rotación del **PM**) y en segundo lugar a las trayectorias de los **GDL** virtuales $\boldsymbol{\theta v_{CM}}$ del **CM** del humanoide durante el ciclo del paso (i.e., trayectoria de la posición y rotación del **CM**, que define el movimiento global del humanoide), que por supuesto respetará las restricciones dadas para el equilibrio estático y dinámico 3.1.1 de la locomoción bípeda, esto es, que el **ZMP** "(3-5)"se encuentra en todo momento dentro del área de soporte.

3.5.1 Formalización del algoritmo UPA para el RH0.

Como vimos en 3.3.1, el algoritmo **UPA** desarrolla en cinco fases el ciclo del paso teórico para la locomoción bípeda de un humanoide genérico (según 3.1.) Para obtener los resultados buscados en la aplicación al robot **RH0** (i.e., los caminos factibles para el **PM** $\mathbf{Q_{rPM}}$ y para el **CM** $\mathbf{Q_{rCM}}$), desarrollaremos exactamente las mismas cinco fases, pero teniendo en cuenta la estructura mecánica del **RH0** que podemos ver en B.1:

- **Fase-1 – Orientar:** Sin mover los pies, se orienta el cuerpo del humanoide hasta alinearlo lo más posible con la dirección del objetivo. Es parte de la etapa de soporte, cuando los dos pies están en contacto con el suelo. Se mantiene el **ZMP** dentro del $\mathbf{A_s}$ dada por el patrón de soporte formado por los dos pies, para garantizar la estabilidad.
- **Fase-2 Inclinar:** Sin mover los pies, se inclina el cuerpo del humanoide para que el **ZMP** se mueva hasta alcanzar una posición interna al patrón de soporte dado por el pie que quedará como apoyo. En el caso del **RH0**, en lugar de considerar un área de soporte circular para el pie, tenemos un $\mathbf{A_s}$ casi elíptico, lo que nos permite desarrollar una geometría del paso más amplia.
- **Fase-3 Elevar:** Se eleva el **PM** desde su contacto con el suelo, pasando desde su **PEP** hasta la posición dada por la altura de paso y de ahí hasta la **PEA** dada por la longitud de paso.
- **Fase-4 Apoyar:** Aterriza el **PM** tocando el suelo. El **ZMP** sigue dentro del patrón de soporte dado por el pie de apoyo, puesto que estamos aún en la parte final de la etapa de transferencia.
- **Fase-5 Balancear:** Sin mover los pies, se balancea el cuerpo del humanoide para que el **ZMP** se mueva hasta alcanzar la posición central del patrón de soporte dado por los dos pies. De esta forma, al final de la fase cinco, nos encontramos en condiciones de repetir otro paso con la ejecución periódica del algoritmo **UPA** desde el comienzo.

Sin embargo, en la aplicación práctica del algoritmo **UPA** para el robot humanoide **RH0** apreciamos que un movimiento complejo arbitrario no puede ser desarrollado como repetición de un único paso elemental, sino que en realidad son necesarios **CUATRO TIPOS DE PASO** básicos para poder lograr cualesquiera movimientos complejos del **RH0**, por lo que el algoritmo **UPA**, aún siendo el mismo en su concepción, presenta las siguientes evoluciones en su desarrollo geométrico aplicado al **RH0** que veremos implementados con más detalle en 3.5.2:

- **Paso de Salida:** El **RH0** se encuentra con los dos pies a la misma distancia de un objetivo dado, medida según el eje de dirección $\boldsymbol{\upsilon_d}$ del movimiento de locomoción del robot. Entonces el paso básico según **UPA** adelanta el pie móvil hasta una posición dada por la longitud de paso **Lp** en la dirección de $\boldsymbol{\upsilon_d}$.
- **Paso de Zancada:** El **RH0** se encuentra con el **PM** retrasado con respecto a la posición del **CM**, que a su vez se encuentra retrasado con respecto a la posición del **PF**, medidas que se toman según el eje de dirección $\boldsymbol{\upsilon_d}$ del movimiento de locomoción. El paso según **UPA** adelanta el **PM** en la dirección de $\boldsymbol{\upsilon_d}$ hasta una posición a distancia **Lp** por delante del **PF**.

- **Paso de Entrada:** El **RH0** se encuentra con el **PM** retrasado con respecto a la posición del **CM**; medida que se toman según el eje de dirección υ_d del movimiento de avance del robot. Entonces el paso según **UPA** adelanta el **PM** en la dirección de υ_d, hasta una posición tal que los dos pies se encuentran a la misma distancia del objetivo y en línea con el **ZMP**.
- **Paso de Giro:** El **RH0** se encuentra con los dos pies a la misma distancia de un objetivo dado, medida según el eje de dirección υ_d del movimiento de locomoción del humanoide. En este caso, el paso básico del **UPA** adelanta el pie móvil hasta una posición dada por el giro del robot sobre su propio eje, según el radio de la anchura de paso $\mathbf{W_p}$ y un ángulo que vendrá dado por las restricciones mecánicas de los **GDL** del **RH0**.

Para cada uno de los cuatro tipos básicos de paso, según las cinco fases del algoritmo **UPA**, **obtenemos de forma geométrica** los cinco puntos principales que forman cada uno de los caminos factibles $\mathbf{Q_{rPM}}$ y $\mathbf{Q_{rCM}}$, respetando las restricciones dadas para el equilibrio 3.1.1 de la locomoción bípeda, es decir, que el **ZMP** "(3-5)"se encuentre en todo momento dentro del $\mathbf{A_s}$. Definimos por tanto las expresiones que deben de presentar los resultados del **UPA** para cada uno de los cuatro tipos de paso:

- $\mathbf{Q_{rCM}}$ **- Camino Factible del CM**: Trayectoria compuesta por configuraciones alcanzables "(4-6)" pertenecientes todas ellas al espacio libre "(4-3)", que presentan continuidad "(4-5)". Donde $\mathbf{q_{CM}}$ es una configuración del **CM** del humanoide, dada por los **GDL** ficticios $\boldsymbol{\theta}\mathbf{v_{CM}}$ del **CM** (i.e., $\boldsymbol{\theta}\mathbf{v_{CM1}}...\boldsymbol{\theta}\mathbf{v_{CM6}}$). El $\mathbf{Q_{rCM}}$ viene definido según "(3-44)", por cinco puntos principales (i.e., $\mathbf{q_{CM1}}...\mathbf{q_{CM5}}$) que son soluciones geométricas a las cinco fases del algoritmo **UPA**.

$$Q_{rCM} = \{q_{CM1}...q_{CM2}...q_{CM3}...q_{CM4}...q_{CM5}\} \qquad \textbf{(3-44)}$$

- $\mathbf{Q_{rPM}}$ **- Camino Factible del PM**: Trayectoria de configuraciones alcanzables "(4-6)" pertenecientes al espacio libre "(4-3)" con continuidad "(4-5)". Definimos $\mathbf{q_{PM}}$ como una configuración del **PM** dada por los **GDL** ficticios $\boldsymbol{\theta}\mathbf{v_{PM}}$ del **PM** (i.e., $\boldsymbol{\theta}\mathbf{v_{PM1}}...\boldsymbol{\theta}\mathbf{v_{PM6}}$). El $\mathbf{Q_{rPM}}$ viene definido según "(3-45)" por cinco puntos principales (i.e., $\mathbf{q_{PM1}}...\mathbf{q_{PM5}}$) que son también soluciones geométricas de las cinco fases del algoritmo **UPA**.

$$Q_{rPM} = \{q_{PM1}...q_{PM2}...q_{PM3}...q_{PM4}...q_{PM5}\} \qquad \textbf{(3-45)}$$

Teniendo los valores principales de $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$, estos caminos factibles se completan por interpolación teniendo en cuenta la geometría mecánica del **RH0** (ver apéndice B). Es posible aplicar diversos criterios al elegir las curvas de interpolación, pero en esta tesis, como veremos en detalle en 3.5.2, elegimos curvas que minimizan las velocidades de las diferentes articulaciones del humanoide en el desarrollo del paso, que se aproximan bastante a los estudios de locomoción bípeda humana introducidos en 1.3.1.

Una vez obtenidos los resultados del algoritmo **UPA** (i.e, $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$), podremos generar el movimiento de locomoción bípeda correspondiente a un paso del robot humanoide **RH0** resolviendo el problema cinemático inverso mediante técnicas de Lie (ver 3.4.1 y 3.4.2). Ahora vamos a presentar los detalles de la implementación de esta formalización con los desarrollos geométricos eficaces del **UPA** para el **RH0**.

3.5.2 Esquemas Geométricos UPA de los pasos básicos del RH0.

Seguidamente describiremos en detalle las soluciones geométricas para los cuatro tipos básicos de paso (i.e. Salida, Zancada, Entrada y Giro), formalizados en 3.5.1 para el humanoide **RH0**. Cada uno de estos pasos consta de cinco fases según el algoritmo **UPA**, que se desarrollan de forma eficaz según lo detallado en 3.3.2, por lo que en este apartado no vamos a repasar pormenorizadamente cada una de ellas, sino que desarrollaremos diagramas de paso completos para cada uno de los tipos de paso.

Para entender los diagramas de paso de las descripciones geométricas eficaces se pueden repasar las definiciones detalladas en el glosario de la tesis (Apéndice C): Área de sopote del pie móvil $\mathbf{As_{PM}}$, área de soporte del pie fijo $\mathbf{As_{PF}}$, pasillo de dirección **Pd** que viene definido por la anchura de paso $\mathbf{W_p}$, eje longitudinal $\boldsymbol{\upsilon_l}$, eje de dirección $\boldsymbol{\upsilon_d}$, eje de avance del pie móvil $\boldsymbol{\upsilon_{PM}}$, la proyección de la configuración del centro de masas $\mathbf{q_{CM}}$ (i.e. **ZMP**) y la configuración del pie móvil $\mathbf{q_{PM}}$.

En los diagramas tenemos un objetivo local $\mathbf{q_{ol}}$ (representado por una semiesfera roja), que indica la dirección y sentido para el movimiento de locomoción del robot **RH0**. Este objetivo local para la planificación de movimientos suele venir dado como resultado de la navegación global del humanoide (e.g., con el modelo **TCG** o el algoritmo **M3R**), aunque puede definirse con cualquier otro modelo o algoritmo, como pueden ser modelos reactivos de respuesta refleja o señales de interacción con un operador. En los experimentos con el **RH0**, se puede hacer que un sistema de visión artificial montado en el humanoide detecte a una persona con un jersey de color rojo que se encuentra en el entorno local del robot **RH0**, calculando la distancia y orientación al mismo, de forma que así queda definido el $\mathbf{q_{ol}}$. En otro experimento, el objetivo local $\mathbf{q_{ol}}$ se calcula a partir del análisis del sonido emitido por un humano que se encuentra en el entorno del robot y llama de viva voz al **RH0**.

En primer lugar partimos con el robot **RH0** según condiciones iniciales dadas por la posición de equilibrio estático en una etapa de doble soporte (ver Figura 3-17).

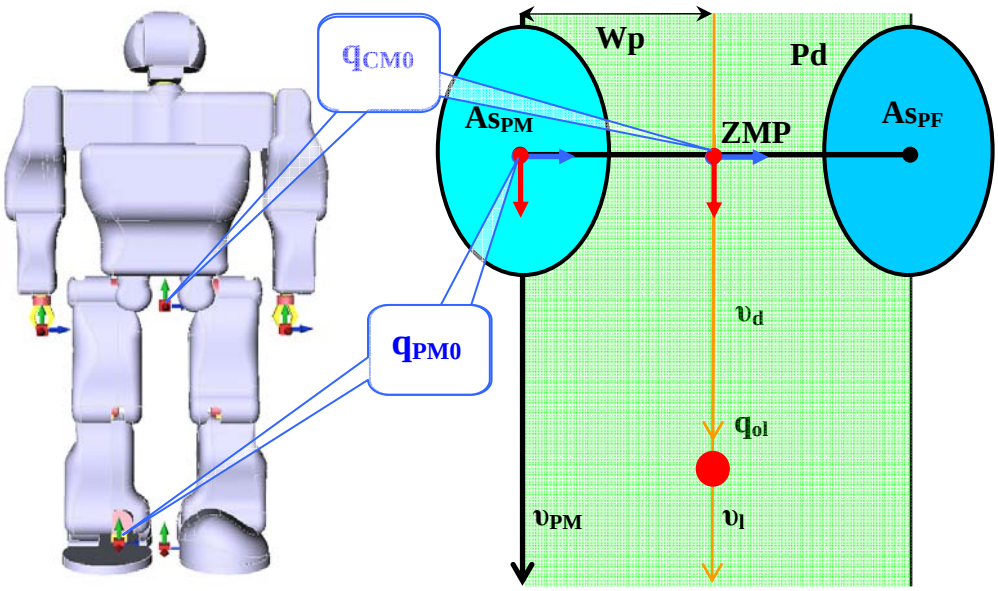


Figura 3-17: Condiciones Iniciales para el RH0 antes de la aplicación del algoritmo UPA.

3.5.2.1 *PASO de SALIDA del RH0 con el algoritmo UPA.*

El **RH0** se encuentra con los dos pies a la misma distancia del objetivo (medida según el eje de dirección υ_d). Entonces el paso básico según **UPA** adelanta el pie móvil (en este caso el derecho) hasta una posición dada por la longitud de paso **Lp** en la dirección del movimiento υ_d, como podemos ver en la Figura 3-18.

Los caminos factibles (i.e. $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$) se obtienen partiendo de cinco configuraciones básicas dadas por las cinco fases del algoritmo **UPA** y completando las trayectorias por interpolación. Para la evolución de la configuración del **PM**, la curva de posición tiene forma de coseno en el plano YZ sobre la dirección υ_{PM}, mientras que la curva de posición en la evolución del **CM** es algo más compleja describiendo una potencial en el plano XZ y una cosenoidal en el plano YZ (ver Figura 3-18).

Presentaremos a continuación las formulaciones para los caminos factibles de las configuraciones del centro de masas y del pié móvil (i.e. $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$) en el caso del **PASO de SALIDA** del **RH0**. Estos caminos son trayectorias que minimizan las velocidades de las diferentes articulaciones del humanoide en el desarrollo del paso, aproximando los resultados de algunos estudios sobre locomoción humana 1.3.1.

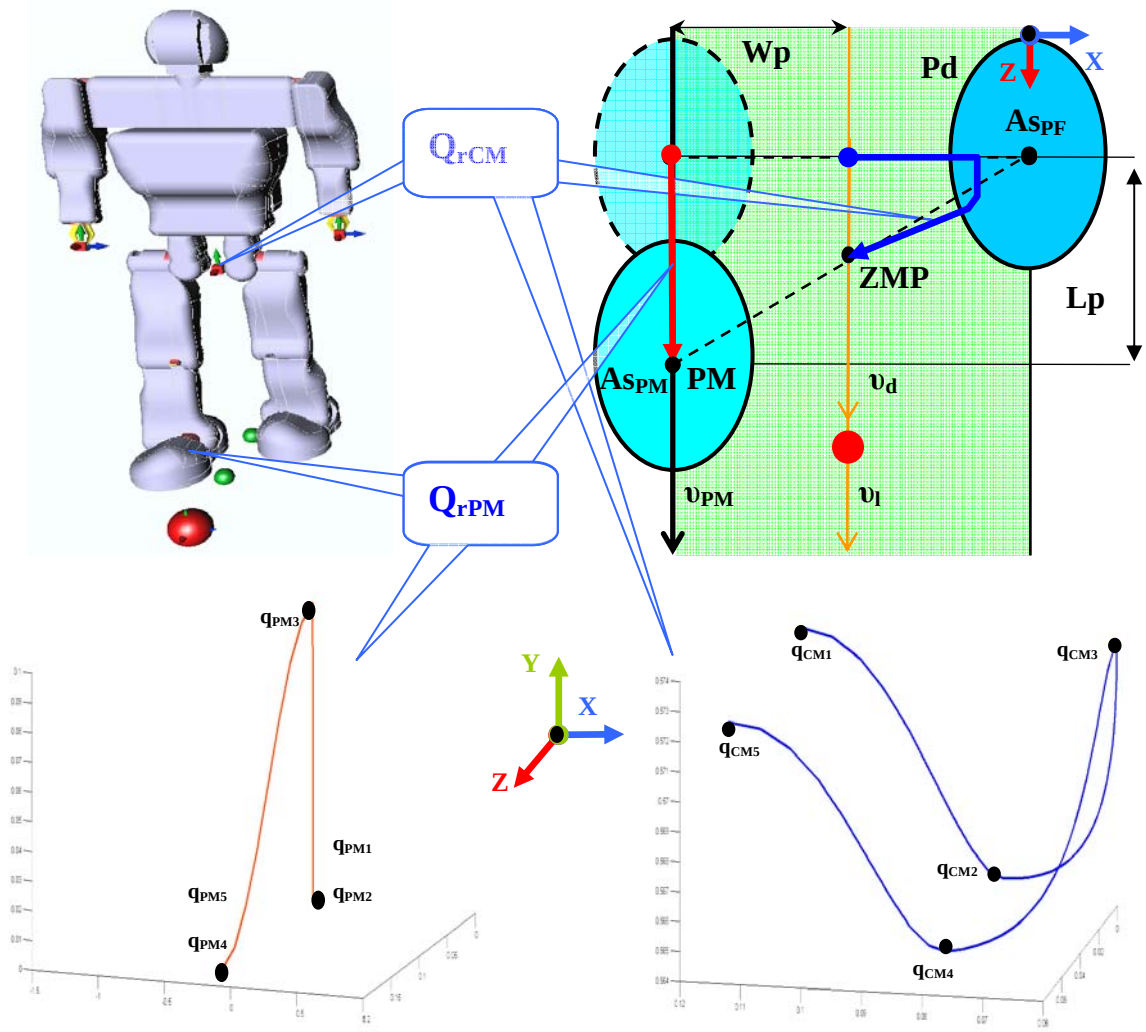


Figura 3-18: Diagrama de PASO de SALIDA del RH0 con el algoritmo UPA.

Formularemos $\mathbf{Q_{rCM}}$, donde $\mathbf{q_{CM}}$ es una configuración del **CM** dada por sus **GDL** ficticios $\boldsymbol{\theta}\mathbf{v_{CM}}$ (i.e., $\boldsymbol{\theta}\mathbf{v_{CM1}}...\boldsymbol{\theta}\mathbf{v_{CM6}}$) según las ecuaciones "(3-46)". El coeficiente potencial $\mathbf{Po_{CM}}$ puede tener diversos valores (e.g. $\mathbf{Po_{CM}}$=4 para este capítulo) en función de la estabilidad del patrón del paso y dependiendo del aprovechamiento que se quiera hacer de la longitud del $\mathbf{As_{PF}}$ para alcanzar mayor $\mathbf{L_p}$. El mínimo radio del $\mathbf{As_{PF}}$ viene dado por $\mathbf{RA_{AS}}$. La altura del centro de masas $\mathbf{H_{CM}}$ debe ser tal que permita tanto el giro del robot como la extensión de las piernas para ejecutar el paso. El coeficiente $\mathbf{K_{CM}}$, (e.g. $\mathbf{K_{CM}}$=0.004 para este capítulo) es absolutamente necesario para reducir $\mathbf{H_{CM}}$ en el desarrollo del paso, consiguiendo una extensión más eficaz de la pierna en la evolución del movimiento según las cinco fases del algoritmo **UPA**.

$$
\begin{aligned}
&Q_{rCM} = \{q_{CM\,1} \cdots q_{CM\,2} \cdots q_{CM\,3} \cdots q_{CM\,4} \cdots q_{CM\,5}\} \\
&q_{CMi} = \{\theta v_{CM\,1}, \theta v_{CM\,2}, \theta v_{CM\,3}, \theta v_{CM\,4}, \theta v_{CM\,5}, \theta v_{CM\,6}\} \\
&\theta v_{CM\,1} = \left\{ RA_{AS} - \frac{W_p + RA_{AS}}{L_p^{\,Po_{CM}}} \cdot \left(-\frac{L_p}{2} \cdots \frac{L_p}{2} \right)^{Po_{CM}} \;\forall\; q_{CMi} \neg (q_{CM\,1} < q_{CMi} < q_{CM\,5}) \right\} \\
&\theta v_{CM\,2} = \left\{ H_{CM} - K_{CM} \cos {}_{0}^{\pi} \left(-\frac{L_p}{2} \cdots q_{CM\,2} \right) \forall\, q_{CMi} \neg (q_{CM\,1} \leq q_{CMi} \leq q_{CM\,2}) \right\} \\
&\theta v_{CM\,2} = \left\{ H_{CM} - K_{CM} \cos {}_{-\pi}^{\pi} (q_{CM\,2} \cdots q_{CM\,4}) \forall\, q_{CMi} \neg (q_{CM\,2} \leq q_{CMi} \leq q_{CM\,4}) \right\} \\
&\theta v_{CM\,2} = \left\{ H_{CM} - K_{CM} \cos {}_{-\pi}^{0} \left(q_{CM\,4} \cdots \frac{L_p}{2} \right) \forall\, q_{CMi} \neg (q_{CM\,4} \leq q_{CMi} \leq q_{CM\,5}) \right\} \\
&\theta v_{CM\,3} = \left\{ 0 \;\forall\; q_{CMi} \neg (q_{CM\,1} \leq q_{CMi} \leq q_{CM\,3}) \wedge 0 \cdots \frac{L_p}{2} \forall\; q_{CMi} \neg (q_{CM\,3} < q_{CMi} < q_{CM\,5}) \right\} \\
&\theta v_{CM\,4} = \theta v_{CM\,5} = \theta v_{CM\,6} = 0
\end{aligned}
\qquad \textbf{(3-46)}
$$

Para $\mathbf{Q_{rPM}}$ definimos $\mathbf{q_{PM}}$ como una configuración del **PM** dada por los **GDL** ficticios $\boldsymbol{\theta}\mathbf{v_{PM}}$ del mismo (i.e., $\boldsymbol{\theta}\mathbf{v_{PM1}}...\boldsymbol{\theta}\mathbf{v_{PM6}}$) según las ecuaciones "(3-47)". La formulación más dificultosa de ver es la de $\boldsymbol{\theta}\mathbf{v_{PM6}}$, que viene dada por restricciones mecánicas en la articulación del tobillo del **RH0**, que obligan a girarlo conforme se eleva la rodilla.

$$
\begin{aligned}
&Q_{rPM} = \{q_{PM\,1} \cdots q_{PM\,2} \cdots q_{PM\,3} \cdots q_{PM\,4} \cdots q_{PM\,5}\} \\
&q_{PMi} = \{\theta v_{PM\,1}, \theta v_{PM\,2}, \theta v_{PM\,3}, \theta v_{PM\,4}, \theta v_{PM\,5}, \theta v_{PM\,6}\} \\
&\theta v_{PM\,1} = W_p \\
&\theta v_{PM\,2} = \{0 \;\forall\; q_{PMi} \neg (q_{PM\,1} \leq q_{PMi} \leq q_{PM\,2} \vee q_{PM\,4} \leq q_{PMi} \leq q_{PM\,5})\} \\
&\theta v_{PM\,2} = \left\{ \frac{H_p}{2} \cdot \left(1 + \cos \left(\frac{\pi}{L_p} (-L_p \cdots L_p) \right) \right) \forall\, q_{PMi} \neg (q_{PM\,2} \leq q_{PMi} \leq q_{PM\,4}) \right\} \\
&\theta v_{PM\,3} = \{0 \;\forall\; q_{PMi} \neg (q_{PM\,1} \leq q_{PMi} \leq q_{PM\,3}) \wedge L_p \;\forall\; q_{PMi} \neg (q_{PM\,4} \leq q_{PMi} \leq q_{PM\,5})\} \\
&\theta v_{PM\,3} = \{0 \cdots L_p \forall\; q_{PMi} \neg (q_{PM\,3} < q_{PMi} < q_{PM\,4})\} \\
&\theta v_{PM\,4} = \theta v_{PM\,5} = 0 \\
&\theta v_{PM\,6} = \frac{20 \cdot \pi}{180 \cdot H_p} \theta v_{PM\,2}
\end{aligned}
\qquad \textbf{(3-47)}
$$

En todo caso, como estas trayectorias de posición de las configuraciones del Paso de Salida no son sino proyecciones de las correspondientes al Paso de Zancada (i.e. el paso resulta ser la mitad de la zancada), nos remitimos al siguiente punto 3.5.2.2 donde se explicarán con mucho más detalle los desarrollos de las correspondientes ecuaciones.

3.5.2.2 *PASO de ZANCADA del RH0 con el algoritmo UPA.*

El **RH0** se encuentra con el **PM** retrasado con respecto a la posición del **CM** que a su vez se encuentra retrasado con respecto a la posición del **PF** (medidas que se toman según el eje de dirección υ_d del movimiento de locomoción). El paso según **UPA** adelanta el **PM**, en este caso el pie izquierdo, en la dirección de υ_d hasta una posición a distancia **Lp** por delante del **PF**, como se puede ver en la Figura 3-19.

Los caminos factibles para el centro de masas y para el pie móvil (i.e. $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$) se obtienen partiendo de las cinco configuraciones básicas dadas por las cinco fases del algoritmo **UPA** y completando las trayectorias por interpolación. Tras tener en cuenta la geometría mecánica del **RH0** (ver apéndice B) hemos elegido curvas de interpolación que minimizan las velocidades de las diferentes articulaciones del humanoide. Para la evolución de la configuración del **PM** la curva tiene forma de coseno en el plano YZ sobre la dirección υ_{PM}, mientras que la curva de evolución del **CM** es más compleja, describiendo una potencial en el plano XZ y una cosenoidal de dos ciclos en el plano YZ (ver la Figura 3-19). Seguidamente estudiaremos en más detalle las formas de estas curvas de interpolación y las razones para su elección en las aplicaciones de esta tesis para el humanoide **RH0**.

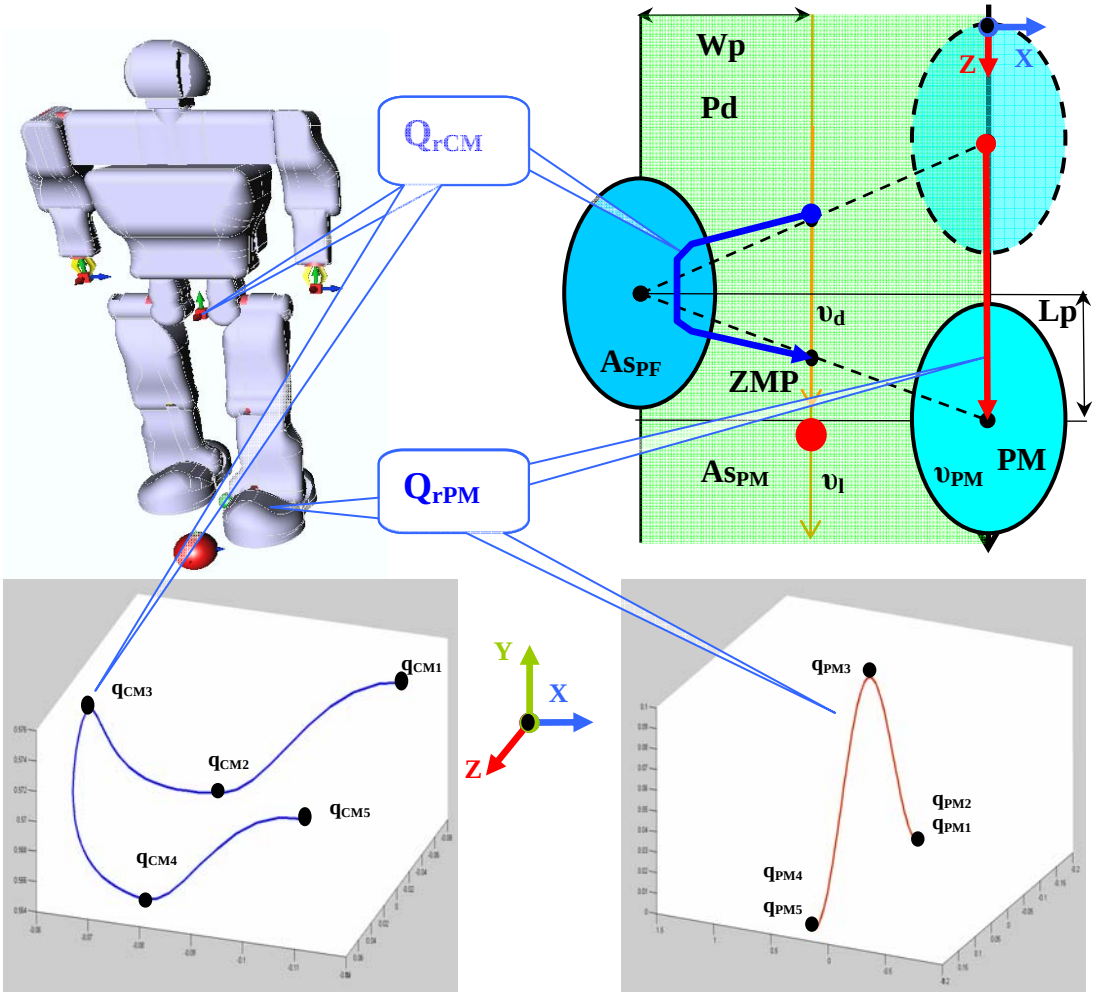


Figura 3-19: Diagrama de PASO de ZANCADA del RH0 con el algoritmo UPA.

El camino factible para el centro de masas (i.e. $\mathbf{Q_{rCM}}$) contiene cinco configuraciones claves dadas por la ejecución del algoritmo **UPA** como hemos visto en 3.5.1, estas son: Orientar $\mathbf{q_{CM1}}$, Inclinar $\mathbf{q_{CM2}}$, Elevar $\mathbf{q_{CM3}}$, Apoyar $\mathbf{q_{CM4}}$ y Balancear $\mathbf{q_{CM5}}$. Una vez obtenidas estas configuraciones del **CM** del humanoide, definimos una trayectoria completa $\mathbf{Q_{rCM}}$ interpolando según la curva de la Figura 3-20.

Como primera característica de la curva de interpolación de la trayectoria de posiciones del $\mathbf{Q_{rCM}}$ podemos apreciar que en el plano XZ tiene una forma potencial (de grado cuarto en este capítulo) para aprovechar el eje longitudinal (más largo) de la base del pie de apoyo (i.e **PF**) del **RH0**, permitiendo así que la zancada sea más larga (i.e. evolución desde $\mathbf{q_{CM2}}$ a $\mathbf{q_{CM4}}$), que en el caso genérico del algoritmo **UPA** (ver 3.3) que consideraba un base circular de los pies para el humanoide. En el plano YZ la curva de interpolación se modula como cosenoidal de doble frecuencia entre $\mathbf{q_{CM1}}$-$\mathbf{q_{CM2}}$ y $\mathbf{q_{CM4}}$-$\mathbf{q_{CM5}}$, en comparación con la frecuencia entre $\mathbf{q_{CM2}}$-$\mathbf{q_{CM4}}$, para permitir por una parte un tiempo más largo en la fase de elevación del pie en vuelo (lo que redunda en menores velocidades para la articulación de la rodilla, que es el valor más crítico), mientras que por otro lado, cuando las piernas se encuentran en su estiramiento máximo (i.e. Inclinar $\mathbf{q_{CM2}}$ y Apoyar $\mathbf{q_{CM4}}$), al encontrarse más bajo el **CM**, podemos alcanzar zancadas más largas. Tenemos que señalar que las variaciones absolutas en la altura del **CM** no son muy grandes, como puede verse en los gráficos de la Figura 3-20, En las simulaciones (ver Capítulo-5) y experimentos (ver Capítulo-6) de esta tesis, el máximo diferencial en la forma de la curva $\mathbf{Q_{rCM}}$ viene dado como una función de los valores de anchura $\mathbf{W_P}$, altura $\mathbf{H_P}$ y longitud de paso $\mathbf{L_P}$ del humanoide **RH0.**

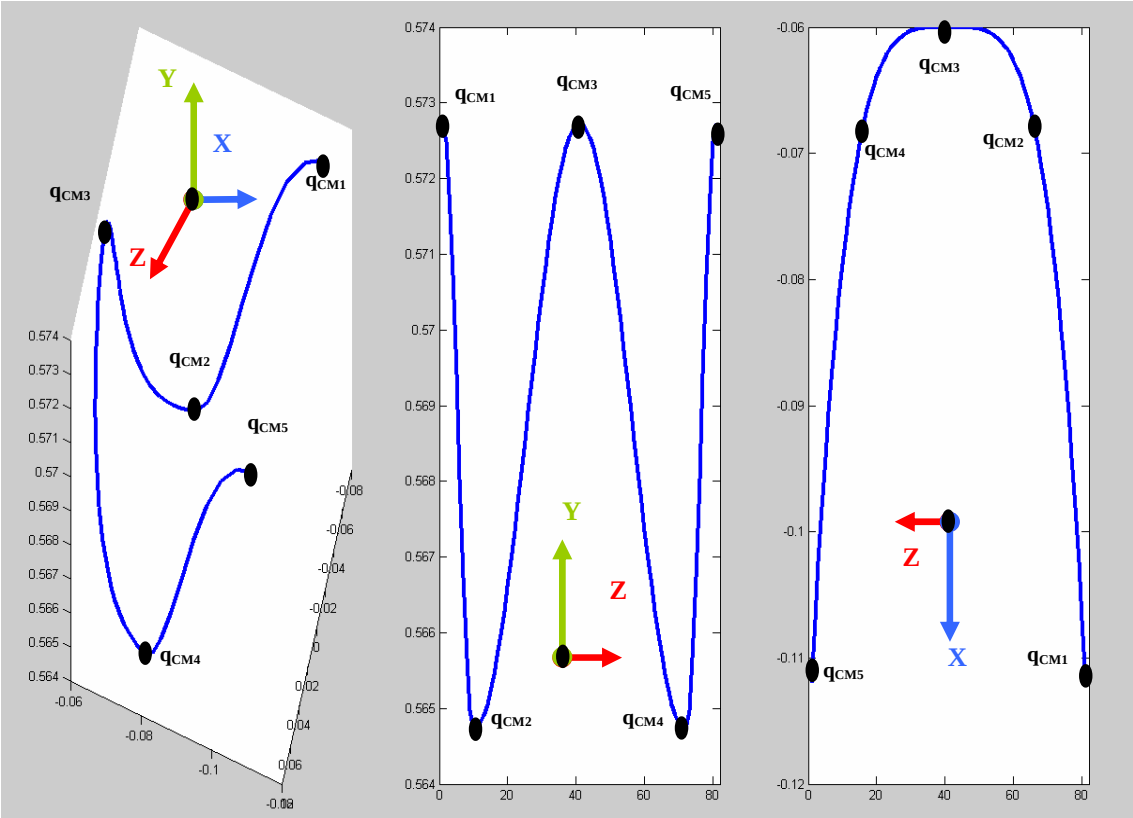


Figura 3-20: Camino Factible para el Centro de Masas Q_{rCM} en el Paso de ZANCADA del RH0.

Una vez explicada la filosofía de generación del camino factible para el centro de masas (i.e. $\mathbf{Q_{rCM}}$) en el caso del Paso de Zancada, podemos extender el mismo concepto para todos los tipos de paso básicos del algoritmo **UPA** para el **RH0**. Es fácil reconocer que el $\mathbf{Q_{rCM}}$ para los restantes tipos de paso (i.e. Salida, Entrada y Giro) puede obtenerse como deformación de la curva que presentamos aquí, esto es, como proyección de la misma en los planos carentes de movimiento en cada uno de ellos, ya que esos tipos de paso son subconjuntos del Paso de Zancada.

Para la construcción matemática de $\mathbf{Q_{rCM}}$ formulamos sus configuraciones $\mathbf{q_{CM}}$ dadas por sus **GDL** ficticios (i.e., $\boldsymbol{\theta v_{CM1}}...\boldsymbol{\theta v_{CM6}}$) según las ecuaciones "(3-48)". El coeficiente potencial $\mathbf{Po_{CM}}$ puede tener diversos valores (e.g. $\mathbf{Po_{CM}}$=4 para este capítulo) en función de la estabilidad del patrón de paso y dependiendo del aprovechamiento que se quiera hacer de la longitud del $\mathbf{As_{PF}}$, para alcanzar mayor $\mathbf{L_p}$. El mínimo radio del $\mathbf{As_{PF}}$ viene dado por $\mathbf{RA_{AS}}$. La altura del centro de masas $\mathbf{H_{CM}}$ debe ser tal que permita tanto el giro del robot como la extensión de las piernas para ejecutar el paso. El coeficiente $\mathbf{K_{CM}}$ (e.g. $\mathbf{K_{CM}}$=0.004 para este capítulo) es absolutamente necesario para reducir $\mathbf{H_{CM}}$ en el desarrollo del paso, consiguiendo una extensión más eficaz de la pierna en la evolución del movimiento según las cinco fases del algoritmo **UPA**.

$$
\begin{aligned}
&Q_{rCM} = \{q_{CM1}...q_{CM2}...q_{CM3}...q_{CM4}...q_{CM5}\}\\
&q_{CMi} = \{\theta v_{CM1}, \theta v_{CM2}, \theta v_{CM3}, \theta v_{CM4}, \theta v_{CM5}, \theta v_{CM6}\}\\
&\theta v_{CM1} = \left\{ RA_{AS} - \frac{W_p + RA_{AS}}{L_p{}^{Po_{CM}}} \cdot \left(-\frac{L_p}{2} \cdots \frac{L_p}{2} \right)^{Po_{CM}} \ \forall\ q_{CMi} \neg \left(q_{CM1} < q_{CMi} < q_{CM5} \right) \right\}\\
&\theta v_{CM2} = \left\{ H_{CM} - K_{CM} \cos_0^{\pi} \left(-\frac{L_p}{2} \cdots q_{CM2} \right) \forall q_{CMi} \neg \left(q_{CM1} \le q_{CMi} \le q_{CM2} \right) \right\}\\
&\theta v_{CM2} = \left\{ H_{CM} - K_{CM} \cos_{-\pi}^{\pi} \left(q_{CM2} \cdots q_{CM4} \right) \forall q_{CMi} \neg \left(q_{CM2} \le q_{CMi} \le q_{CM4} \right) \right\} \qquad \textbf{(3-48)}\\
&\theta v_{CM2} = \left\{ H_{CM} - K_{CM} \cos_{-\pi}^{0} \left(q_{CM4} \cdots \frac{L_p}{2} \right) \forall q_{CMi} \neg \left(q_{CM4} \le q_{CMi} \le q_{CM5} \right) \right\}\\
&\theta v_{CM3} = \left\{ -\frac{L_p}{2} \cdots \frac{L_p}{2} \forall\ q_{CMi} \neg \left(q_{CM1} < q_{CMi} < q_{CM5} \right) \right\}\\
&\theta v_{CM4} = \theta v_{CM5} = \theta v_{CM6} = 0
\end{aligned}
$$

El camino factible para el pie móvil (i.e. $\mathbf{Q_{rPM}}$) contiene cinco configuraciones claves dadas por la ejecución del algoritmo **UPA** como hemos visto en 3.5.1, estas son: Orientar $\mathbf{q_{PM1}}$, Inclinar $\mathbf{q_{PM2}}$, Elevar $\mathbf{q_{PM3}}$, Apoyar $\mathbf{q_{PM4}}$ y Balancear $\mathbf{q_{PM5}}$. Una vez obtenidas las configuraciones del **PM** del humanoide definimos una trayectoria de posición completa $\mathbf{Q_{rPM}}$ interpolando según la curva de la Figura 3-21. La curva de interpolación es una cosenoidal espacial pero no temporal, esto es, el **PM** recorre una curva con forma de coseno en el espacio, pero en el tiempo no es así, puesto que durante las fases de Orientación e Inclinación, así como durante las de Apoyo y Balanceo, el **PM** se encuentra sobre la superficie de soporte en la misma posición. La forma de coseno permite suavizar los movimientos de locomoción de modo que las velocidades de las articulaciones quedan reducidas a un cuasi-mínimo. Esta elección de la forma de la trayectoria del **PM** es un compromiso entre la mecánica del **RH0** y los resultados de trabajos de investigación (ver 1.3.1) en mecánica de locomoción bípeda.

La forma de la curva de posiciones para $\mathbf{Q_{rPM}}$ viene dada como una función de los valores de $\mathbf{H_P}$ (es diez centímetros en el ejemplo de la Figura 3-21) y de $\mathbf{L_P}$ del humanoide **RH0.** Una vez explicada la filosofía de generación del camino factible para el pie móvil (i.e. $\mathbf{Q_{rPM}}$) en el caso del Paso de Zancada, podemos extender el mismo concepto para todos los tipos de paso básicos del algoritmo **UPA** para el **RH0**. Es fácil reconocer que el $\mathbf{Q_{rPM}}$ para los restantes tipos de paso (i.e. Salida, Entrada y Giro) puede obtenerse como deformación de la curva que presentamos aquí, esto es, como proyección de la misma en los planos carentes de movimiento en cada uno de ellos, ya que esos tipos de paso son subconjuntos del Paso de Zancada.

Para definir matemáticamente $\mathbf{Q_{rPM}}$ debemos obtener sus configuraciones $\mathbf{q_{PM}}$ dadas por sus **GDL** ficticios (i.e., $\boldsymbol{\theta v_{PM1}}...\boldsymbol{\theta v_{PM6}}$) según las ecuaciones "(3-49)". La formulación más dificultosa de ver es la de $\boldsymbol{\theta v_{PM6}}$, que viene dada por las restricciones mecánicas en la articulación del tobillo del **RH0** que obliga a girarlo conforme se eleva la rodilla.

$$
\begin{aligned}
Q_{rPM} &= \{q_{PM\,1}...q_{PM\,2}...q_{PM\,3}...q_{PM\,4}...q_{PM\,5}\} \\
q_{PMi} &= \{\theta v_{PM\,1}, \theta v_{PM\,2}, \theta v_{PM\,3}, \theta v_{PM\,4}, \theta v_{PM\,5}, \theta v_{PM\,6}\} \\
\theta v_{PM\,1} &= W_p \\
\theta v_{PM\,2} &= \{0 \,\forall\, q_{PMi} \,\neg(q_{PM\,1} \le q_{PMi} \le q_{PM\,2} \vee q_{PM\,4} \le q_{PMi} \le q_{PM\,5})\} \\
\theta v_{PM\,2} &= \left\{\frac{H_p}{2}\cdot\left(1+\cos\left(\frac{\pi}{L_p}\left(-L_p \cdots L_p\right)\right)\right) \forall q_{PMi} \,\neg(q_{PM\,2} \le q_{PMi} \le q_{PM\,4})\right\} \qquad \textbf{(3-49)} \\
\theta v_{PM\,3} &= \{-L_p \forall\, q_{PMi} \,\neg(q_{PM\,1} \le q_{PMi} \le q_{PM\,2}) \wedge L_p \,\forall\, q_{PMi} \,\neg(q_{PM\,4} \le q_{PMi} \le q_{PM\,5})\} \\
\theta v_{PM\,3} &= \{-L_p \cdots L_p \forall\, q_{PMi} \,\neg(q_{PM\,2} < q_{PMi} < q_{PM\,4})\} \\
\theta v_{PM\,4} &= \theta v_{PM\,5} = 0 \\
\theta v_{PM\,6} &= \frac{20\cdot\pi}{180\cdot H_p}\theta v_{PM\,2}
\end{aligned}
$$

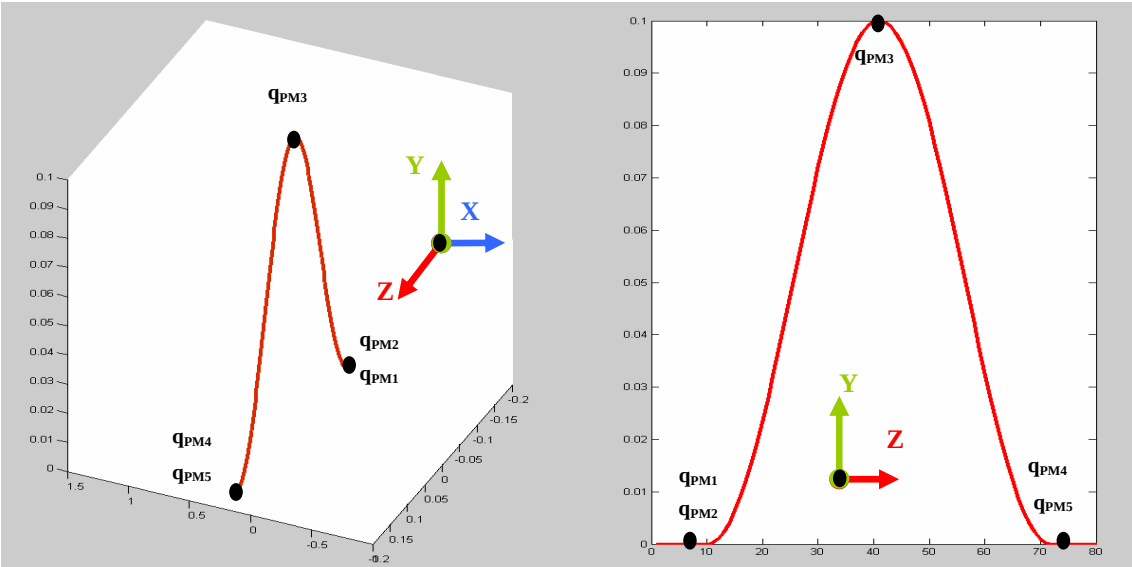


Figura 3-21: Camino Factible para el Pie Móvil Q_{rPM} en el Paso de ZANCADA del RH0.

3.5.2.3 *PASO de ENTRADA del RH0 con el algoritmo UPA.*

El **RH0** se encuentra con el **PM** retrasado con respecto a la posición del **CM** y el paso según **UPA** adelanta el **PM** en la dirección de υ_d hasta una posición tal que los dos pies se encuentran a la misma distancia del objetivo (ver Figura 3-22). Los caminos factibles (i.e. $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$) se obtienen como se explicó en detalle en 3.5.2.2, pero proyectando las trayectorias XY en el eje que une ambos pies para la segunda mitad del paso.

Los caminos factibles (i.e. $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$) parten de las cinco configuraciones básicas dadas por las cinco fases del algoritmo **UPA** y completan las trayectorias por interpolación. Para la evolución de la configuración del **PM** la curva de posición tiene forma de coseno en el plano YZ sobre la dirección υ_{PM}, mientras que la curva de posición en la evolución del **CM** es algo más compleja, describiendo una potencial en el plano XZ y una cosenoidal en el plano YZ (ver Figura 3-22), pero sólo para la primera mitad del paso.

Presentaremos a continuación los caminos factibles de las configuraciones del centro de masas y del pié móvil (i.e. $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$) en el caso del **PASO de ENTRADA** del **RH0**. Estos caminos son trayectorias que minimizan las velocidades de las diferentes articulaciones del humanoide en el desarrollo del paso, aproximando los resultados de algunos estudios sobre locomoción humana 1.3.1. Formularemos $\mathbf{Q_{rCM}}$, donde $\mathbf{q_{CM}}$ es una configuración del **CM** dada por sus **GDL** ficticios $\boldsymbol{\theta v_{CM}}$ (i.e., $\boldsymbol{\theta v_{CM1}}$...$\boldsymbol{\theta v_{CM6}}$) según las ecuaciones "(3-50)". El coeficiente potencial $\mathbf{Po_{CM}}$ puede tener diversos valores (e.g. $\mathbf{Po_{CM}}$=4 para este capítulo) en función de la estabilidad del patrón del paso y dependiendo del aprovechamiento que se quiera hacer de la longitud del $\mathbf{As_{PF}}$ para alcanzar mayor $\mathbf{L_p}$. El mínimo radio del $\mathbf{As_{PF}}$ viene dado por $\mathbf{RA_{AS}}$. La altura del centro de masas $\mathbf{H_{CM}}$ debe ser tal que permita la extensión de las piernas para ejecutar el paso. El coeficiente $\mathbf{K_{CM}}$ (e.g. $\mathbf{K_{CM}}$=0.004 para este capítulo) es absolutamente necesario para reducir $\mathbf{H_{CM}}$ en el desarrollo del paso consiguiendo una extensión más eficaz de la pierna en la evolución del movimiento según las cinco fases del algoritmo **UPA**.

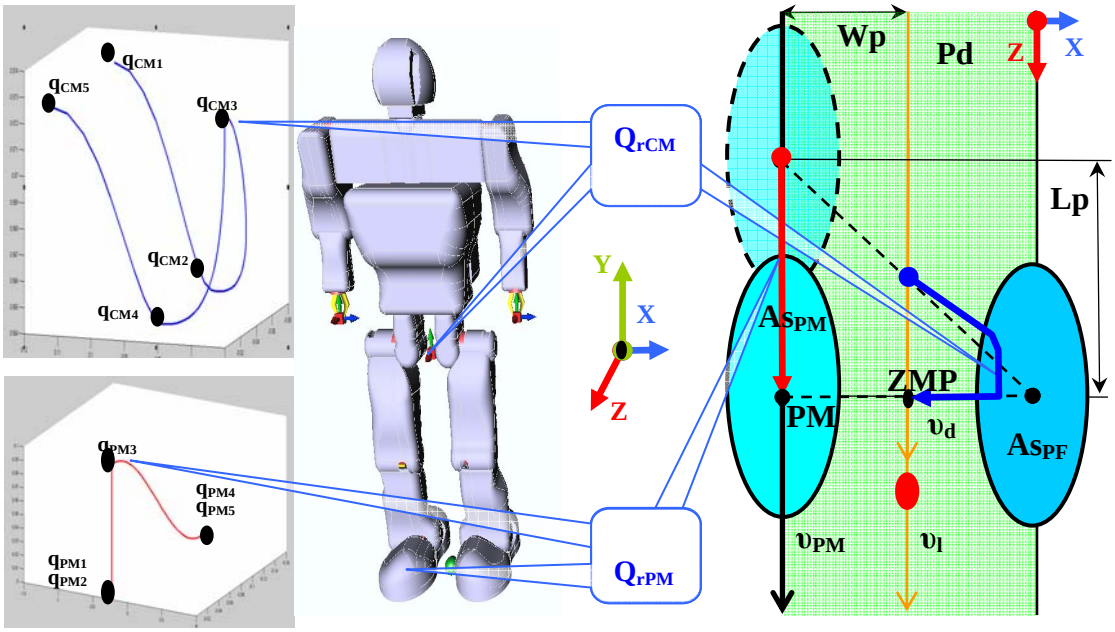


Figura 3-22: Diagrama de PASO de ENTRADA del RH0 con el algoritmo UPA.

$$
\begin{aligned}
&Q_{rCM} = \{q_{CM1}..q_{CM2}..q_{CM3}..q_{CM4}..q_{CM5}\}\\
&q_{CMi} = \{\theta v_{CM1}, \theta v_{CM2}, \theta v_{CM3}, \theta v_{CM4}, \theta v_{CM5}, \theta v_{CM6}\}\\
&\theta v_{CM1} = \left\{ RA_{AS} - \frac{W_p + RA_{AS}}{L_p^{Po_{CM}}} \cdot \left(-\frac{L_p}{2} \cdots \frac{L_p}{2} \right)^{Po_{CM}} \forall\, q_{CMi} \neg\left(q_{CM1} < q_{CMi} < q_{CM5}\right) \right\}\\
&\theta v_{CM2} = \left\{ H_{CM} - K_{CM} \cos_{0}^{\pi}\left(-\frac{L_p}{2} \cdots q_{CM2} \right) \forall q_{CMi} \neg\left(q_{CM1} \le q_{CMi} \le q_{CM2}\right) \right\}\\
&\theta v_{CM2} = \left\{ H_{CM} - K_{CM} \cos_{-\pi}^{\pi}\left(q_{CM2} \cdots q_{CM4}\right) \forall q_{CMi} \neg\left(q_{CM2} \le q_{CMi} \le q_{CM4}\right) \right\}\\
&\theta v_{CM2} = \left\{ H_{CM} - K_{CM} \cos_{-\pi}^{0}\left(q_{CM4} \cdots \frac{L_p}{2} \right) \forall q_{CMi} \neg\left(q_{CM4} \le q_{CMi} \le q_{CM5}\right) \right\}\\
&\theta v_{CM3} = \left\{ 0 \forall\, q_{CMi} \neg\left(q_{CM3} \le q_{CMi} \le q_{CM5}\right) \wedge -\frac{L_p}{2} \cdots 0 \forall\, q_{CMi} \neg\left(q_{CM1} < q_{CMi} < q_{CM3}\right) \right\}\\
&\theta v_{CM4} = \theta v_{CM5} = \theta v_{CM6} = 0
\end{aligned}
\qquad \textbf{(3-50)}
$$

Para $\mathbf{Q_{rPM}}$ definimos $\mathbf{q_{PM}}$ como una configuración del **PM** dada por los **GDL** ficticios $\boldsymbol{\theta v_{PM}}$ del mismo (i.e., $\boldsymbol{\theta v_{PM1}}$...$\boldsymbol{\theta v_{PM6}}$) según las ecuaciones "(3-51)". La formulación más dificultosa de ver es la de $\boldsymbol{\theta v_{PM6}}$ que viene dada por restricciones mecánicas en la articulación del tobillo del **RH0** que obliga a girarlo conforme se eleva la rodilla.

$$
\begin{aligned}
&Q_{rPM} = \{q_{PM1}..q_{PM2}..q_{PM3}..q_{PM4}..q_{PM5}\}\\
&q_{PMi} = \{\theta v_{PM1}, \theta v_{PM2}, \theta v_{PM3}, \theta v_{PM4}, \theta v_{PM5}, \theta v_{PM6}\}\\
&\theta v_{PM1} = W_p\\
&\theta v_{PM2} = \{0 \forall\, q_{PMi} \neg\left(q_{PM1} \le q_{PMi} \le q_{PM2} \vee q_{PM4} \le q_{PMi} \le q_{PM5}\right)\}\\
&\theta v_{PM2} = \left\{ \frac{H_p}{2} \cdot \left(1 + \cos\left(\frac{\pi}{L_p}\left(-L_p \cdots L_p\right) \right) \right) \forall q_{PMi} \neg\left(q_{PM2} \le q_{PMi} \le q_{PM4}\right) \right\}\\
&\theta v_{PM3} = \{0 \forall\, q_{PMi} \neg\left(q_{PM3} \le q_{PMi} \le q_{PM5}\right) \wedge -L_p \forall\, q_{PMi} \neg\left(q_{PM1} \le q_{PMi} \le q_{PM2}\right)\}\\
&\theta v_{PM3} = \{-L_p \cdots 0 \forall\, q_{PMi} \neg\left(q_{PM2} < q_{PMi} < q_{PM3}\right)\}\\
&\theta v_{PM4} = \theta v_{PM5} = 0\\
&\theta v_{PM6} = \frac{20 \cdot \pi}{180 \cdot H_p} \theta v_{PM2}
\end{aligned}
\qquad \textbf{(3-51)}
$$

En todo caso, como estas trayectorias de posición de las configuraciones del Paso de Entrada no son sino proyecciones de las correspondientes al Paso de Zancada (i.e. el paso resulta ser la mitad de la zancada), nos remitimos al punto 3.5.2.2 donde se explican mucho más en detalle los desarrollos de las correspondientes ecuaciones.

3.5.2.4 *PASO de GIRO del RH0 con el algoritmo UPA.*

El **RH0** se encuentra con los dos pies a la misma altura, esto es, separados por una distancia igual al doble de la anchura de paso $\mathbf{W_p}$ y el **UPA** adelanta el pie móvil hasta una posición dada por el giro del robot sobre su propio eje, según el radio de la anchura de paso $\mathbf{W_p}$ y un ángulo dado por las restricciones mecánicas de los **GDL** del **RH0** (ver Figura 3-23), pero que trata en lo posible acercarse hasta la dirección indicada por el objetivo $\boldsymbol{\upsilon_d}$.

Los caminos (i.e. $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$) se obtienen como se explicó en 3.5.2.2, pero proyectando las trayectorias según los ejes afectados por el giro del humanoide. De este modo, la trayectoria $\mathbf{Q_{rCM}}$ queda proyectada totalmente sobre el plano XY, considerando los ejes del pie fijo, por lo que la posición inicial y final del **CM** es la misma. La trayectoria del pie móvil $\mathbf{Q_{rPM}}$ resulta semejante a la del paso de salida 3.5.2.1 pero girada en función de la dirección del objetivo $\boldsymbol{\upsilon_d}$ o en su caso limitada por el giro máximo $\mathbf{G_{CM}}$ definido por las restricciones mecánicas del robot. Al repetir este movimiento sucesivamente en parejas (i.e. siempre se realizarán dos pasos de giro consecutivos pie derecho más pie izquierdo o viceversa) conseguimos un giro perfecto del humanoide **RH0** sobre su eje, como se podrá ver con detalle en 5.3.2, de forma que el humanoide queda preparado para ejecutar un ciclo de locomoción que se iniciará con un paso de salida 3.5.2.1.

Formularemos $\mathbf{Q_{rCM}}$ según las ecuaciones "(3-52)". $\mathbf{Po_{CM}}$ puede tener diversos valores (e.g. $\mathbf{Po_{CM}}$=4 para este capítulo). $\mathbf{H_{CM}}$ debe ser tal que permita el giro del robot. $\mathbf{K_{CM}}$ (e.g. $\mathbf{K_{CM}}$=0.004 para este capítulo) permite reducir $\mathbf{H_{CM}}$ en el desarrollo del paso. El mínimo radio del $\mathbf{As_{PF}}$ viene dado por $\mathbf{RA_{AS}}$. $\boldsymbol{\theta v_{PM5}}$ finaliza el movimiento girado en función de la dirección del objetivo $\boldsymbol{\upsilon_d}$ o limitado por el giro máximo $\mathbf{G_{CM}}$ (e.g. $\mathbf{G_{CM}}$=15º para este capítulo), para de esta forma preparar el cuerpo del humanoide a una repetición simétrica del este algoritmo de paso con el pie contrario, que permita la necesaria ejecución doble del Paso de Giro.

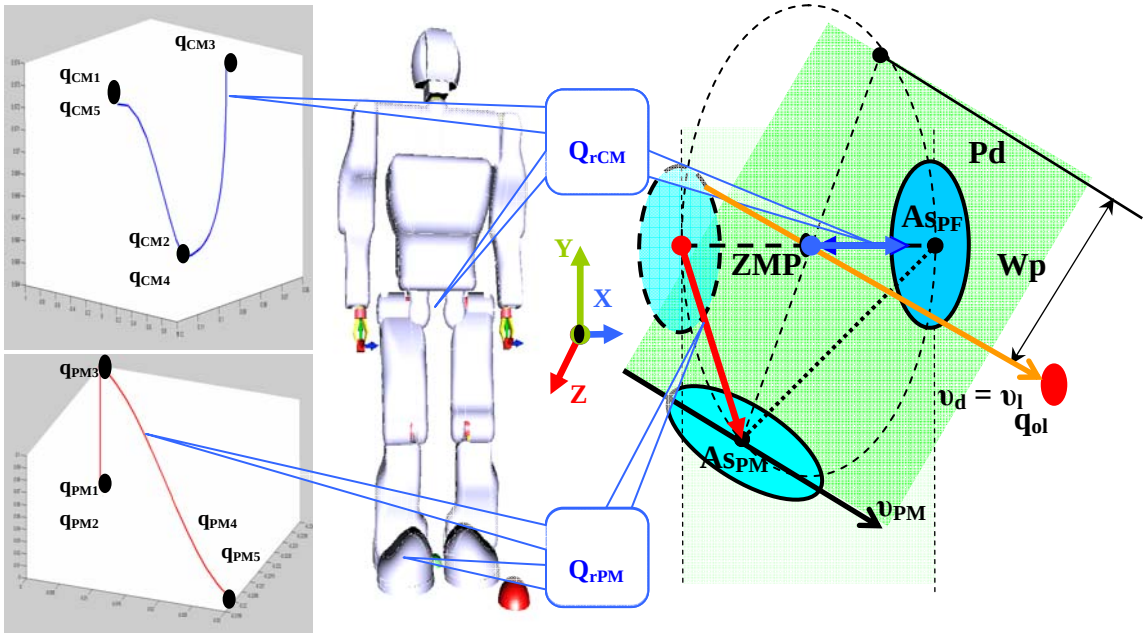


Figura 3-23: Diagrama de PASO de GIRO del RH0 con el algoritmo UPA.

$$
\begin{aligned}
&Q_{rCM} = \{q_{CM1} \cdots q_{CM2} \cdots q_{CM3} \cdots q_{CM4} \cdots q_{CM5}\} \\
&q_{CMi} = \{\theta v_{CM1}, \theta v_{CM2}, \theta v_{CM3}, \theta v_{CM4}, \theta v_{CM5}, \theta v_{CM6}\} \\
&\theta v_{CM1} = \left\{ RA_{AS} - \frac{W_p + RA_{AS}}{L_p^{Po_{CM}}} \cdot \left(-\frac{L_p}{2} \cdots \frac{L_p}{2} \right)^{Po_{CM}} \ \forall\ q_{CMi} \neg\left(q_{CM1} \le q_{CMi} \le q_{CM5}\right) \right\} \\
&\theta v_{CM2} = \left\{ H_{CM} - K_{CM} \cos_0^{\pi}\left(-\frac{L_p}{2} \cdots q_{CM2} \right) \forall q_{CMi} \neg\left(q_{CM1} \le q_{CMi} \le q_{CM2}\right) \right\} \\
&\theta v_{CM2} = \left\{ H_{CM} - K_{CM} \cos_{-\pi}^{\pi}\left(q_{CM2} \cdots q_{CM4}\right) \forall q_{CMi} \neg\left(q_{CM2} \le q_{CMi} \le q_{CM4}\right) \right\} \\
&\theta v_{CM2} = \left\{ H_{CM} - K_{CM} \cos_{-\pi}^{0}\left(q_{CM4} \cdots \frac{L_p}{2} \right) \forall q_{CMi} \neg\left(q_{CM4} \le q_{CMi} \le q_{CM5}\right) \right\} \\
&\theta v_{CM3} = \theta v_{CM4} = \theta v_{CM6} = 0 \\
&\theta v_{CM5} = \{0 \ \forall q_{CMi} \neg\left(q_{CM1} \le q_{CMi} \le q_{CM3}\right)\} \\
&\theta v_{CM5} = \{0 \cdots \min\left(G_{CM}, \upsilon_d\right) \forall q_{CMi} \neg\left(q_{CM3} \le q_{CMi} \le q_{CM4}\right)\} \\
&\theta v_{CM5} = \{\min\left(G_{CM}, \upsilon_d\right) \forall q_{CMi} \neg\left(q_{CM4} \le q_{CMi} \le q_{CM5}\right)\}
\end{aligned}
\qquad \textbf{(3-52)}
$$

Definimos $\mathbf{Q_{rPM}}$ según las ecuaciones "(3-53)". $\boldsymbol{\theta v_{PM6}}$ viene dada por restricciones mecánicas en la articulación del tobillo. La trayectoria del pie móvil $\mathbf{Q_{rPM}}$ resulta semejante a la del paso de salida 3.5.2.1 pero girada en función de la dirección del objetivo $\boldsymbol{\upsilon_d}$ o limitada por el giro máximo $\mathbf{G_{CM}}$ (e.g. $\mathbf{G_{CM}}$=15º para este capítulo).

$$
\begin{aligned}
&Q_{rPM} = \{q_{PM1} \cdots q_{PM2} \cdots q_{PM3} \cdots q_{PM4} \cdots q_{PM5}\} \\
&q_{PMi} = \{\theta v_{PM1}, \theta v_{PM2}, \theta v_{PM3}, \theta v_{PM4}, \theta v_{PM5}, \theta v_{PM6}\} \\
&\theta v_{PM1} = \{W_p \ \forall q_{PMi} \neg\left(q_{PM1} \le q_{PMi} \le q_{PM2}\right)\} \\
&\theta v_{PM1} = \{W_p \cdots W_p \cos\left(\min\left(G_{CM}, \upsilon_d\right)\right) \forall q_{PMi} \neg\left(q_{PM2} \le q_{PMi} \le q_{PM4}\right)\} \\
&\theta v_{PM1} = \{W_p \cos\left(\min\left(G_{CM}, \upsilon_d\right)\right) \forall q_{PMi} \neg\left(q_{PM4} \le q_{PMi} \le q_{PM5}\right)\} \\
&\theta v_{PM2} = \{0 \forall q_{PMi} \neg\left(q_{PM1} \le q_{PMi} \le q_{PM2} \vee q_{PM4} \le q_{PMi} \le q_{PM5}\right)\} \\
&\theta v_{PM2} = \left\{ \frac{H_p}{2} \cdot \left(1 + \cos\left(\frac{\pi}{L_p}\left(-L_p \cdots L_p\right) \right) \right) \forall q_{PMi} \neg\left(q_{PM2} \le q_{PMi} \le q_{PM4}\right) \right\} \\
&\theta v_{PM3} = \{0 \forall q_{PMi} \neg\left(q_{PM1} \le q_{PMi} \le q_{PM3}\right) \wedge W_p \sin\left(\min\left(G_{CM}, \upsilon_d\right)\right) \forall q_{PMi} \neg\left(q_{PM4} \le q_{PMi} \le q_{PM5}\right)\} \\
&\theta v_{PM3} = \{0 \cdots W_p \sin\left(\min\left(G_{CM}, \upsilon_d\right)\right) \forall q_{PMi} \neg\left(q_{PM3} < q_{PMi} < q_{PM4}\right)\} \\
&\theta v_{PM4} = 0 \\
&\theta v_{CM5} = \{0 \forall q_{CMi} \neg\left(q_{CM1} \le q_{CMi} \le q_{CM3}\right)\} \\
&\theta v_{CM5} = \{0 \cdots \min\left(G_{CM}, \upsilon_d\right) \forall q_{CMi} \neg\left(q_{CM3} \le q_{CMi} \le q_{CM4}\right)\} \\
&\theta v_{CM5} = \{\min\left(G_{CM}, \upsilon_d\right) \forall q_{CMi} \neg\left(q_{CM4} \le q_{CMi} \le q_{CM5}\right)\} \\
&\theta v_{PM6} = \frac{20 \cdot \pi}{180 \cdot H_p} \theta v_{PM2}
\end{aligned}
\qquad \textbf{(3-53)}
$$

Nos remitimos al punto 3.5.2.2 donde se explican en detalle estos desarrollos.

4 Planificación de Movimientos y Navegación para Robots Humanoides.

Los sistemas de control utilizados en la ingeniería de robots todavía tienen muy limitadas las capacidades para la planificación de movimientos y la navegación. La mayoría de trabajos publicados contiene un espacio con obstáculos convexos o de geometría específica, o presenta limitaciones para un número elevado de GDL, o tiene algoritmos que no garantizan el orden de convergencia, o son poco eficaces computacionalmente. En este capítulo tras formalizar la planificación y revisar los métodos más utilizados, nos centraremos en la presentación y desarrollo detallado de un nuevo algoritmo al que llamaremos ***Método Modificado de Marcha Rápida*** *(**MR3**), cuyo orden de convergencia puede llegar a ser* $\mathbf{O(N^n)}$*, y que aplicaremos a entornos con obstáculos de cualquier geometría para el cálculo de planificación de trayectorias globales libres de colisiones para robots humanoides, mediante el nuevo Modelo de Navegación que denominaremos* ***Trayectoria Corporal Global*** *(**TCG**).*

4.1 Formalización de la Planificación.

En primer lugar, aunque existen varios conceptos de planificación que están muy relacionados, por mayor claridad de exposición vamos a distinguir los dos siguientes tipos:

- **Planificación de Movimientos**: se realiza normalmente en el espacio de las articulaciones del robot (i.e., el Espacio de Configuraciones) y se ocupa del diseño de movimientos factibles (que no dejan de ser sino trayectorias) para cada uno de los grados de libertad del robot, de forma que se consigan los objetivos deseados, que principalmente en el caso que nos ocupa son lo relacionados con la locomoción bípeda del humanoide.
- **Planificación de Trayectorias**: se realiza normalmente en el espacio de trabajo del robot (i.e., el Espacio Físico) y es la responsable de la búsqueda de trayectorias factibles (e.g., libres de colisiones) dentro del ambiente de trabajo para las extremidades del humanoide y especialmente para el conjunto del sistema (e.g., para el tronco del robot.), esto último es la resolución del problema de navegación. Esta planificación de trayectorias (navegación) es una tarea controlada a nivel superior (normalmente) puesto que define objetivos que deben ser resueltos antes de iniciar el movimiento del robot.

El entorno de trabajo en el cual un humanoide realizará su tarea puede considerarse como un espacio que es función de los grados de libertad del robot, las restricciones y los obstáculos, esto es, lo que llamamos el **C-Space**. Existirá un subconjunto alcanzable (i.e., el espacio libre para un posible movimiento) y otro inalcanzable, al estar ocupado por los obstáculos del entorno o debido a restricciones cinemáticas. Con estas premisas previas, podemos presentar una formalización matemática del problema de planificación.

Definimos $\mathbf{q_h}$ como una configuración del humanoide, esto es, el vector que proporciona información sobre el estado actual del robot y que viene dado por las componentes (i.e., $\boldsymbol{\theta}$) correspondientes a cada **GDL** del mismo según "(4-1)".

$$q_h = f(\theta) \qquad \textbf{(4-1)}$$

El espacio total de configuraciones (i.e., **C-Space**) lo vamos a denotar por $\mathbf{Q_h}$. El subconjunto de $\mathbf{Q_h}$ ocupado por el robot cuando se encuentra en una configuración específica se denota por $\mathbf{Q_i(q_h)}$. En el espacio de trabajo lo normal es encontrar un conjunto de obstáculos definidos como objetos que se distribuyen por el espacio de configuraciones. El conjunto de configuraciones ocupadas por obstáculos se define por $\mathbf{O_i(q_o)}$ (i.e., espacio ocupado), de forma que el espacio libre para el humanoide (i.e., libre de obstáculos) $\mathbf{Q_{hl}}$, viene dado por el subconjunto de configuraciones "(4-2)".

$$Q_{hl} = \left\{ q_h \in Q_h \cap \left(\bigcup_{i=1}^{n} O_i(q_o) \right) = \varnothing \right\} \qquad \textbf{(4-2)}$$

Vamos a introducir ahora algunos conceptos de importancia para la planificación de movimientos:

- **Camino Factible - Q_r**: es una trayectoria compuesta por configuraciones pertenecientes todas ellas al espacio libre "(4-3)". El camino factible que queda parametrizado por una función tal como se muestra en la formula "(4-4)". Además, una propiedad importantísima es que la función de construcción del camino factible debe presentar continuidad, concepto que podemos reflejar con la expresión "(4-5)".

$$Q_r = \{q_s \dots q_j \dots q_f \ / \ q_j \in Q_{hl}\} \quad (4\text{-}3)$$

$$\gamma : [0,1] \rightarrow Q_{hl} \wedge \gamma(0) = q_s \wedge \gamma(1) = q_f \quad (4\text{-}4)$$

$$\lim_{q_j \rightarrow q_0} = \left\| \gamma(q_j), \gamma(q_0) \right\| = 0 \quad (4\text{-}5)$$

- **Configuración Alcanzable – q_n**: una configuración cualquiera se dice que es alcanzable, si existe al menos un camino factible que lleve al robot desde su configuración actual $\mathbf{q_s}$ hasta ella "(4-6)".

$$q_n \ / \ \exists Q_r = \{q_s \dots q_j \dots q_n \ / \ q_j \in Q_{hl}\} \quad (4\text{-}6)$$

- **Espacio Físico Observable**: Es el conjunto de todas las configuraciones alcanzables.

- **Propiedad de Algoritmo Completo**: En la planificación de movimientos y trayectorias esta propiedad se cumple si para un determinado problema, esta garantizado que el algoritmo encuentra una solución, en el caso de que esta exista. Si un algoritmo es completo nos devolverá como resultado un camino factible, si este existe, cualquiera que sea el entorno dado para el problema.

El problema de planificación queda definido como la búsqueda de una sucesión de configuraciones alcanzables, desde la inicial hasta la objetivo, estando todas ellas incluidas en el espacio libre de forma que construyen un camino factible.

Desde esta introducción inicial, que podemos considerar común, la formalización matemática de la planificación varía en función de los métodos implementados (4.2.1.) De modo que, por ejemplo, para métodos de descomposición en celdas necesitaremos introducir la matemática de grafos, para los métodos probabilísticos, algoritmos estadísticos específicos o para las funciones de potencial, algoritmos de integración de la ecuación diferencial. Dada la infraestructura matemática utilizada en esta tesis para la mecánica de humanoides (1.2), tenemos que hacer especial mención a la planificación en el espacio **SE(3)** con matrices de grupos de Lie, mediante la optimización de funciones de coste (ver [124]). Aunque en esta tesis nos centraremos en la formalización de la planificación de trayectorias, mediante técnicas geométricas que analizan el movimiento de interfaces (ver 4.3), que se fundamentan en la solución de una clase especial de ecuaciones de Hamilton-Jacobi y su relación con las leyes de conservación hiperbólica.

4.2 Planificación de Trayectorias y Navegación.

La navegación es la metodología que permite guiar el curso del humanoide de una forma segura dentro de su entorno de trabajo. La estructura de control de navegación básica se conforma por la relación entre cada una de las siguientes tareas que de una forma u otra se presentan en todos los humanoides:

- **Modelado del entorno**: con mapas que se obtienen a priori o se construye en tiempo real mediante uso de sensores.
- **Planificación de trayectorias**: para el conjunto del sistema móvil (viene a ser la planificación de trayectorias libre de colisiones para el tronco del humanoide.) Necesita un mapa del entorno, la descripción de la tarea y alguna estrategia de cálculo para obtener las secuencias de metas y objetivos que forman un camino continuo desde la posición actual del robot hasta su meta. Se deben distinguir los siguientes tipos:
    - **Planificación Global**: en la que se aproxima al camino final del robot según las especificaciones del problema. Se construye normalmente sobre mapas obtenidos a priori. Necesita conocer con el menor error posible la posición del robot, por lo que se suele utilizar odometría y/o referencias externas.
    - **Planificación Local**: resuelve las obstrucciones sobre la ruta global en el entorno local del humanoide. Se construye sobre la base de la fusión de información sensorial (normalmente en tiempo real), permitiendo reaccionar dinámicamente ante el entorno para no colisionar con obstáculos.
- **Control y seguimiento del camino**: control de los actuadores del humanoide para el desplazamiento del mismo, recorriendo el camino planificado.

Parece claro que para realizar misiones reales se hace necesario la combinación de los dos tipos básicos de planificación para la navegación, global y local o reactiva, si queremos desarrollar arquitecturas con buenas posibilidades de éxito.

4.2.1 Métodos de Planificación de Trayectorias.

El problema de planificación de trayectorias se aborda como un problema geométrico para poder producir soluciones prácticas. Esto es obviamente una simplificación (interesada) ya que se ignoran las propiedades dinámicas del robot eliminando los aspectos temporales. Los métodos de planificación más utilizados en las investigaciones de los pasados años han sido los siguientes:

- **Mapas de caminos** (***RoadMaps***): la idea en este tipo de métodos de planificación de caminos consiste en capturar la conectividad del espacio libre en el que se encuentra el robot, en una serie o red de curvas unidimensionales. Normalmente se desarrollan en dos etapas: obtención del grafo de conectividad, a partir del modelado del entorno y extracción del camino desde la configuración inicial a la final dentro del grafo. Existen numerosas variantes en esta metodología:

- **Grafos de visibilidad**: los obstáculos se modelan mediante polígonos y para la generación del grafo se introduce el concepto de "visibilidad", según el cual se definen dos puntos del espacio de configuraciones como visibles si y solo si el segmento que los une es una arista de un obstáculo o está completamente contenido en el espacio libre. Considerando que los nodos del grafo de visibilidad serán la posición del robot, la posición objetivo y los vértices de los obstáculos del entorno el grafo resultará de la unión mediante segmentos de todos aquellos nodos que sean visibles. Para hallar el camino, mediante un algoritmo de búsqueda en grafos se elige la ruta que una la configuración inicial con la final minimizando alguna función de coste.
- **Curvas Splines**: este enfoque trata de enlazar el punto origen y destino del robot con un conjunto de curvas b–splines, de forma que se consiga cumplir con las condiciones de continuidad del camino.
- **Diagramas de Voronoi**: se definen como una proyección del espacio en una red de curvas unidimensionales yacientes en el espacio libre. Formalmente es definen como una retracción con preservación de la continuidad del espacio de configuraciones libres. El diagrama de Voronoi resulta ser el lugar geométrico de las configuraciones que se encuentran a igual distancia de los dos obstáculos más próximos del entorno. El algoritmo de planificación consiste en encontrar la secuencia de segmentos del diagrama de Voronoi que conecten las retracciones de la posición original del robot con el objetivo
- **Modelado del Espacio Libre**: se aplica con obstáculos poligonales mediante el modelado del espacio libre con los denominados "cilindros rectilíneos generalizados", intentando que el robot navegue lo más alejado posible de los obstáculos desde una configuración inicial a la final a través de los ejes interconectados de los cilindros.

- **Descomposición en Celdas**: probablemente son los métodos de planificación más estudiados y consisten en descomponer el espacio libre en regiones simples no solapadas, denominadas celdas, cuya unión es el espacio libre permite construir un grafo no dirigido (grafo de conectividad), que representa la relación de adyacencia entre las celdas. El camino entre dos configuraciones puede hallarse con una secuencia de celdas que se denominará "canal". Una vez obtenido el grafo, se puede emplear cualquier algoritmo de búsqueda para hallar el camino.
- **Campos de Potencial**: la idea consiste en que el robot se comporte como una partícula que se mueve dentro del campo (espacio) de configuraciones, bajo la influencia de potenciales artificialesv (normalmente un potencial atractivo provocado por el destino y potenciales repulsivos provocados por los obstáculos). Una vez calculada la función potencial, el camino entre el robot y el objetivo se encuentra resolviendo el gradiente negativo de la función para el valor inicial dado por la posición del robot. Este método ha sido ampliamente utilizado, aunque la función de potencial puede presentar mínimos locales que impiden la obtención de la trayectoria, lo que es un grave inconveniente.

- **Caminos Probabilísticos**: seleccionan aleatoriamente ciertas configuraciones del espacio libre interconectando pares de estas configuraciones por caminos factibles simples. Se construye una estructura de datos de forma incremental de una forma probabilística, para posteriormente usar la estructura de datos creada durante la llamada fase de consulta, en la resolución de problemas individuales de planificación de caminos. La estructura de datos es un grafo no dirigido, donde los nodos son configuraciones libres y los arcos corresponden a caminos factibles. Cuando se quiere calcular el camino global entre dos configuraciones, se concatenan arcos entre nodos mediante un algoritmo de búsqueda sobre el grafo generado. Este tipo de algoritmos ha tenido gran éxito en los últimos años aplicados a problemas con muchos **GDL**, aunque la usual inclusión de técnicas heurísticas no garantiza el orden de convergencia, puesto que a veces situaciones muy complejas son resueltas en un tiempo razonable, mientras que otras posiciones relativamente fáciles de solucionar necesitan un tiempo de computación excesivo.

- **Inteligencia Artificial**: una gran variedad de técnicas han sido aplicadas (e.g., redes neuronales, algoritmos genéticos) para la solución de la planificación de trayectorias, sobre todo desde el auge de los enfoques para control de robots basados en comportamiento y técnicas reactivas de navegación.

Todos los métodos revisados anteriormente presentan ventajas e inconvenientes, aunque en general, los problemas más importantes que aparecen en los algoritmos de planificación de trayectorias son los siguientes:

- **Modelado del entorno y los obstáculos**: es muy difícil modelar el entorno con precisión, más aún si existen cambios de forma dinámica, pero muchos algoritmos exigen la definición exacta del entorno a priori. Además, normalmente se exige que los obstáculos tengan una forma geométrica determinada (e.g. poliedros, esferas), como premisa básica de algunas ideas. Otras veces de exige que los obstáculos al menos sean convexos para reducir las posibilidades de existencia de mínimos locales en las soluciones.

- **Orden de complejidad computacional**: algunas soluciones algorítmicas se implementan con órdenes de complejidad muy altos que las hacen impracticables en cuanto el espacio de configuraciones es complejo.

En esta tesis, desarrollaremos un método nuevo para la planificación de trayectorias, basado en las técnicas de geometría computacional conocidas como "Métodos de Marcha Rápida" (ver apartado 4.3), que son herramientas numéricas para analizar el movimiento de interfaces. Por semejanza con los anteriormente revisados, podemos decir que los métodos geométricos de marcha rápida serían una mezcla entre los métodos de descomposición en celdas y los métodos de campos de potencial, aunque como veremos en detalle resuelven muchos de los inconvenientes que éstos plantean. El nuevo algoritmo que desarrollaremos a continuación, al que llamaremos Método Modificado de Marcha Rápida (**M3R**), puede ser utilizado para la planificación de caminos en robótica dentro de entornos con obstáculos de cualquier geometría, y en esta tesis nosotros lo utilizaremos para la planificación global de la trayectoria del robot humanoide con el nuevo modelo **TCG**.

4.3 NUEVO ALGORITMO - Método Modificado de Marcha Rápida (M3R).

Los métodos de marcha rápida son técnicas numéricas para analizar el movimiento de interfaces. Se basan en un cambio fundamental acerca de cómo podemos apreciar un frente en evolución, cambiando la perspectiva Lagrangiana por una Euleriana. Las técnicas numéricas resultantes se pueden usar para seguir frentes complejos que pueden evolucionar y cambiar de topología, para un número arbitrario de dimensiones. Son algoritmos de interés para numerosas disciplinas: geometría compleja, fotolitografía, sísmica, mecánica de fluidos, combustión, visión por computadora, microelectrónica, diseño, ingeniería de control y otras.

En el desarrollo de estas técnicas matemáticas debemos destacar los trabajos del profesor J.A. Sethian [104] de la Universidad de California BERKELEY, y su equipo de colaboradores, cuya inspiración ha sido fundamental en el desarrollo de algunos de los trabajos de esta tesis.

Se desarrolla en esta tesis un nuevo algoritmo que hemos denominado "Método Modificado de Marcha Rápida" (**M3R**), que puede servir tanto para la planificación de movimientos como para la navegación en robótica. Para problemas holonómicos este método presenta la ventaja de encontrar el camino sin mínimos locales. Se demuestra que este método permite encontrar siempre un camino casi-óptimo, sea cual fuere la naturaleza de los obstáculos, esto es, funciona con obstáculos de geometría no convexa e incluso con formas irregulares.

El algoritmo puede implementarse de una manera muy eficaz con un orden de convergencia computacional máximo $\mathbf{O(G^n logG)}$, siendo **G** el valor de discretización o número de puntos (equivalente al número de celdas, en una descomposición de ese tipo) en cada dimensión **n**.

4.3.1 Formulación de un frente con condiciones de contorno.

La propagación de frentes está presente en una gran variedad de situaciones e incluye desde las olas del mar a la evolución de las llamas en una combustión. Situaciones menos obvias para la presencia de fronteras son la evolución de formas, caracteres escritos e imágenes. Hay aplicaciones que no se han afrontado inicialmente como movimiento de interfaces pero que pueden abordarse desde ese punto de vista, como curvas geodésicas en superficies o planificación de caminos óptimos.

La teoría y las técnicas numéricas conocidas como Métodos de Marcha Rápida derivan de un planteamiento para describir el movimiento de interfaces, basado en una resolución de las ecuaciones en derivadas parciales como un problema de condiciones de contorno. Tienen el mérito de unificar las ideas relativas a la evolución de frentes en un marco general, proporcionando algoritmos para seguir la propagación de interfaces en todo un abanico de aplicaciones en la física e ingeniería.

Iniciaremos la introducción matemática para la formulación de un frente en evolución, considerando un límite genérico, por ejemplo, una expansión puntual unidimensional, una curva en dos dimensiones (ver Figura 4-1) o una superficie en tres dimensiones, que separa una región de otra.

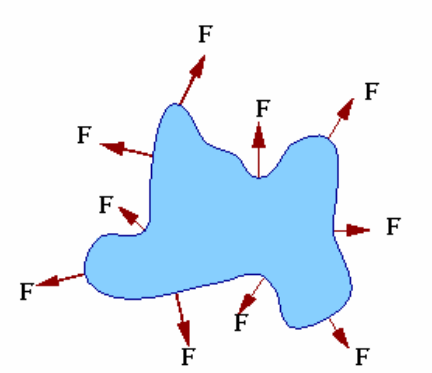


Figura 4-1: Frente propagándose con velocidad F.

Imagínese que esa curva o superficie se mueve en dirección normal a sí misma con una velocidad conocida **F**. El objetivo sería seguir el movimiento de la interfaz mientras esta evoluciona desde una perspectiva Euleriana, esto es, un marco en el que el sistema coordenado subyacente de referencia es inercial (i.e., permanece fijo.)

Tenemos una curva cerrada propagándose de forma normal a sí misma con una velocidad **F**. Además se asume que **F** es positiva (i.e., **F**>0), y por tanto el frente siempre se mueve hacia delante, esto es, el frente se encuentra en expansión. La función de velocidad **F** puede depender de muchos factores, de modo que podemos describirla como "(4-7)":

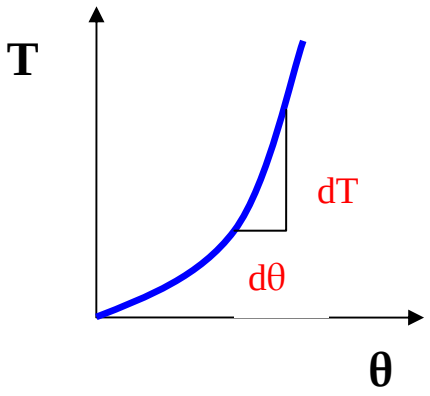
$$F = f\left(Lo, Gl, In\right) \qquad \textbf{(4-7)}$$

Donde:

- **Lo – Factores Locales**: Propiedades determinadas por la geometría local, como la curvatura o la dirección normal al frente.
- **Gl – Factores Globales**: Propiedades dadas por la posición y forma del frente.
- **In – Factores Independientes**: Propiedades debidas a factores que no tienen relación con la forma del frente, por ejemplo, la velocidad de un fluido subyacente que arrastra el conjunto de la interfaz.

Gran parte del desafío en los problemas modelados como frentes en evolución consiste en definir una función de velocidad adecuada, que represente fielmente el modelo físico. En nuestro caso de planificación de trayectorias, veremos más adelante que la definición puede ser muy sencilla para simplificar el algoritmo, sin pérdida de generalidad.

Una forma de caracterizar la posición de un frente en expansión es computar el tiempo de llegada **T**, en el que el frente alcanza cada punto del espacio matemático subyacente al interfaz. Es evidente que para una dimensión (ver Figura 4-2) podemos obtener la ecuación para la función de llegada **T** de una forma muy fácil, simplemente teniendo en cuenta el hecho de que la distancia es el producto de la velocidad por el tiempo "(4-8)". La derivada espacial de la función solución se convierte en el gradiente "(4-9)" y por lo tanto tendremos que la magnitud del gradiente de la función de alcance es inversamente proporcional a la velocidad "(4-10)". Para múltiples dimensiones, el mismo concepto tiene validez al ser el gradiente ortogonal a los conjuntos de nivel de funciones de alcance.



$$\theta = F \cdot T \qquad (4\text{-}8)$$

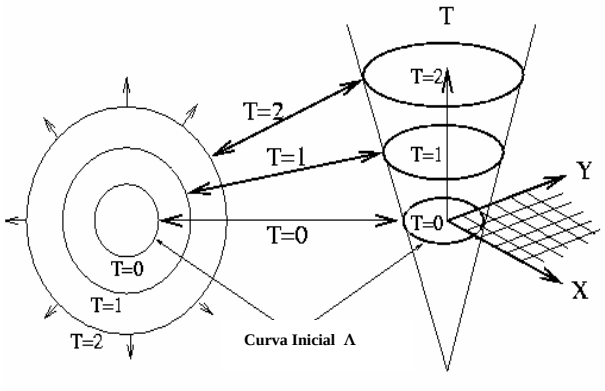
$$1 = F \frac{dT}{d\theta} \qquad (4\text{-}9)$$

$$\frac{1}{F} = \left| \nabla T \right| \qquad (4\text{-}10)$$

Figura 4-2: Formulación de la función de alcance T(θ), para un frente unidimensional.

De esta forma, podemos caracterizar el movimiento del frente como la solución de un problema de condiciones de contorno. Si la velocidad **F** depende sólo de la posición, entonces la ecuación "(4-10)" se reduce a lo que es conocida como **ecuación Eikonal** "(4-11)". Como ejemplo sencillo (ver Figura 4-3) definimos un frente circular **Λ** "(4-12)", para dos dimensiones, que avanza con velocidad unitaria (i.e., F=1.) Se puede ver la evolución del valor de la función de alcance **T(x, y)** conforme pasa el tiempo (i.e., **T=0, T=1, T=2...**) y el frente llega a puntos del plano en regiones más externas de la superficie. La condición de contorno es que el valor de la función de alcance es nulo en la curva inicial "(4-13)".

Este planteamiento desemboca en los llamados Métodos de Marcha Rápida, que son algoritmos numéricos consistentes, precisos y eficientes, para computar la ecuación Eikonal "(4-11)", basados en la satisfacción de la entropía en un frente que avanza y en técnicas de ordenación rápida. Estas perspectivas aportan muchas ventajas, aunque su demostración requiere un amplio desarrollo matemático, que se puede encontrar en los trabajos de Sethian [104] y de Yong [128]. En resumen estas ventajas son:



$$\left| \nabla T \right| F = 1 \qquad (4\text{-}11)$$

$$\Lambda(t) = \left\{ (x,y) \mid T(x,y) = t \right\} \qquad (4\text{-}12)$$

$$T\left(\Lambda(0)\right) = 0 \qquad (4\text{-}13)$$

Figura 4-3: Movimiento de un frente circular, como problema de condiciones de contorno.

- No cambian para mayor número de dimensiones, esto es, para hipersuperficies propagándose en tres o más dimensiones.
- Las propiedades geométricas intrínsecas del frente, tales como el vector normal o la curvatura, se pueden determinar con facilidad.
- Cambios topológicos en la evolución del frente se manejan de forma natural.
- Se pueden aproximar con precisión mediante técnicas numéricas a través de esquemas computacionales que explotan técnicas ya usadas para resolver las leyes de conservación hiperbólica. Se basan en las soluciones de viscosidad de las ecuaciones en derivadas parciales asociadas y garantizan que la solución es obtenida satisfaciendo las condiciones de entropía. Todo esto es posible, porque el role de la curvatura en la propagación de un frente, es matemáticamente análogo al role de la viscosidad en algunas leyes de conservación hiperbólica.

4.3.2 Aproximación de la ecuación Eikonal.

Como presentan Barth y Sethian [16], la **ecuación Eikonal** "(4-11)" es un caso específico de una clase más amplia de ecuaciones de Hamilton-Jacobi "(4-14)", donde $\mathbf{D_T}$ representa las derivadas parciales de la función de tiempo de alcance "**T**" para cada una de las variables del espacio.

$$HJ\ (D_T, \theta) = F\left|\nabla T\right| - 1 = 0 \qquad \textbf{(4-14)}$$

Una de las peculiaridades que aflora en la resolución de esta ecuación es que la solución no necesita ser diferenciable. Esta no diferenciabilidad está íntimamente conectada con la noción de soluciones débiles, necesarias para resolver el problema cuando el frente, al desarrollarse, llega a una esquina y se convierte en no diferenciable.

Extendiendo las ideas de aproximaciones en avance para el gradiente en múltiples dimensiones, podemos construir esquemas apropiados para resolver el Hamiltoniano numéricamente, y por ende la ecuación Eikonal. Por ejemplo, un esquema de primer orden para espacio convexo tiene la siguiente forma "(4-15)" para un número **n** de dimensiones **θ** del espacio.

$$\left[\sum_{i=1}^{n}\left(\max(\ D^{-\theta_i}T,0)^2 + \min(\ D^{+\theta_i}T,0)^2\right)\right]^{\frac{1}{2}} = \frac{1}{F} \qquad \textbf{(4-15)}$$

Donde las derivadas espaciales se definen según las leyes de conservación hiperbólica (que están íntimamente relacionadas con el Hamiltoniano), según las fórmulas "(4-16)" que se pueden encontrar en el libro de LeVeque [65]. Siendo **G** el intervalo de discretización espacial y **t** el tiempo.

$$D^{+\theta}T \equiv \frac{T(\theta + G, t) - T(\theta, t)}{G}$$
$$D^{-\theta}T \equiv \frac{T(\theta, t) - T(\theta - G, t)}{G} \qquad \textbf{(4-16)}$$

En la construcción de esquemas para resolver la ecuación Eikonal, los métodos estándar que se utilizan requieren iteración, por lo que nos encontramos con un problema fundamental como es el hecho de no tener garantizado el orden de convergencia. La ecuación "(4-15)" es cuadrática, por lo que para cada paso de integración hay que utilizar un método numérico lento, como es el binario o bisección. Esto hace que estas implementaciones sean de muy difícil aplicación a problemas de planificación de trayectorias en tiempo real.

Es aquí, donde para solventar los problemas mencionados de los métodos estándar de marcha rápida se introduce en esta tesis un nuevo algoritmo con el desarrollo del Método Modificado de Marcha Rápida (**M3R**). Una aportación teórica de esta tesis para este nuevo desarrollo aprovecha la particularidad de que la las soluciones para la ecuación "(4-14)" no necesariamente tienen que ser diferenciables, lo que aprovechamos para introducir un esquema lineal para la ecuación "(4-15)", de forma que la aproximación para la ecuación Eikonal queda formulada por "(4-17)" para un número **n** de dimensiones (i.e., **θ**) del espacio.

$$\max \bigcup_{i=1}^{n} \left(D^{-\theta_i}T, D^{+\theta_i}T\right) = \frac{1}{F} \qquad \textbf{(4-17)}$$

Donde se conservan las derivadas espaciales que se definían por las leyes de conservación hiperbólica "(4-16)", aunque se reformulan de forma más compacta como "(4-18)" . Siendo $\mathbf{G_i}$ el intervalo de discretización espacial para cada dimensión y **t** el tiempo.

$$D^{\pm\theta_i}T \equiv \frac{T(\theta_i,t) - T(\theta_i \pm G_i,t)}{G_i} \qquad \textbf{(4-18)}$$

Esta nueva formulación se basa en los mismos fundamentos matemáticos, postulados y demostraciones que podemos encontrar en los trabajos de LeVeque [65], Barth [16] y especialmente en los de Sethian [104] sobre geometría computacional. El nuevo algoritmo **M3R** tiene la misma base teórica que los Métodos de Marcha Rápida desarrollados por Sethian [104] para analizar geométricamente el avance de un frente, sólo que el **M3R** modifica la forma del interfaz en evolución (i.e., la función de alcance **T**), que se presenta como un frente plano en expansión.

Como ejemplo, para un problema bidimensional, el nuevo frente tiene formas cuadrangulares en avance, como se puede ver en la Figura 4-13 para una mejor compresión práctica del funcionamiento algoritmo. Para el caso de espacio Euclídeo tridimensional, el frente es expansión adquiere formas piramidales. El algoritmo se puede usar para problemas con mayor número de dimensiones, aunque su visualización ya no resulta tan trivial.

El enfoque que presentamos para el **M3R** resulta especialmente útil para resolver los problemas de planificación de trayectorias. Las razones para ello, son las siguientes:

- El **M3R** resulta ser un algoritmo geométrico, resoluble de forma directa, debido a su formulación lineal. Evitamos por tanto los problemas de iteración, los algoritmos numéricos lentos (e.g., bisección) y la incertidumbre en la convergencia. El resultado es que podemos crear esquemas eficientes con una eficiencia computacional muy alta (ver 4.3.4).

- La función de alcance para el **M3R** es siempre creciente en el sentido de propagación de la información, desde un una curva inicial (que podemos reducir hasta hacerla coincidir con un punto objetivo), hacia todos los puntos del espacio de configuraciones. Al usar la aproximación lineal resulta una función no diferenciable, pero sí continua, que es lo que necesitamos para hallar el camino entre cualquier punto del espacio y el objetivo, siguiendo los valores decrecientes máximos (equivalente al máximo gradiente negativo si la función fuese diferenciable) de dicha función de alcance (ver 4.3.7).

- La geometría de la función de alcance **T** no presenta mínimos locales cualquiera que sean las condiciones del entorno. Esto es, la función de alcance seguirá siendo creciente independientemente de la naturaleza de los obstáculos que se encuentren en el espacio. Esto hace que el **M3R** permita planificar las trayectorias en entornos con obstáculos de todo tipo (i.e., convexos, cóncavos, interconectados e irregulares), lo que es una gran ventaja si lo comparamos con otros métodos de planificación (ver 4.2.1).

- El algoritmo **M3R** no necesita ser desarrollado en los puntos del espacio de configuraciones que están ocupados por obstáculos o que por alguna razón no son de interés. Esto lo hace adecuado tanto para planificaciones globales como locales (ver 4.2), con la ventaja adicional de que cuanto mayor sea el número y tamaño de los obstáculos o áreas no interesantes, mayor será la rapidez del algoritmo para planificar.

Como no todo podían ser ventajas, como consecuencia de la naturaleza de la función propuesta "(4-17)", el camino obtenido en la planificación de trayectorias no será óptimo ni perfectamente suavizado. No obstante, no se considera que éste sea un inconveniente grave para el problema de control de robots humanoides que abordamos; en primer lugar porque la información obtenida se puede suavizar con un filtro matemático (si se necesita hacerlo) y en segundo lugar porque el camino obtenido se puede utilizar directamente (de forma habitual), como una entrada de referencia válida al sistema de control que se encargará de ajustarlo conforme a las condiciones de cada robot.

En resumen, el nuevo algoritmo que hemos introducido, llamado Método Modificado de Marcha Rápida (**M3R**) resulta adecuado para resolver problemas de planificación de trayectorias en robótica, dado que siempre obtiene una solución determinista, continua y completa, cualesquiera sean el número, naturaleza y estructura de los obstáculos en el entorno. Por todo ello, vamos a utilizar el **M3R** en esta tesis para el trabajo con humanoides.

4.3.3 Esquema Geométrico Eficiente para el algoritmo M3R.

En este apartado vamos a desarrollar un esquema eficiente detallado del algoritmo **M3R** para el caso bidimensional, esto es, un frente expandiéndose en una superficie plana. La implementación bidimensional resulta más sencilla de explicar, puesto que las ayudas gráficas son más claras, pero se entenderá fácilmente que la idea puede ser extendida a espacios de configuraciones con un número de dimensiones mucho mayor, en principio sin más limite que el de la eficiencia computacional.

La idea que subyace detrás de este esquema es la propiedad de causalidad. El método construye sistemáticamente la solución de la función de alcance **T** usando sólo los valores aguas arriba del avance del frente. Esto es, propagamos la información en un solo sentido, desde valores inferiores de **T** hacia aquellos que son superiores. Es ahí donde encontramos la causalidad, puesto que los valores superiores donde el frente avanza son calculados (i.e., causados) con valores inferiores firmes y conocidos (i.e., su valor no se modifica). Barreremos el frente considerando los puntos en la zona existente alrededor del frente actual y marchando hacia delante desde allí, congelando los valores de los puntos ya conocidos. La clave del algoritmo se encuentra en conocer el punto del frente a seleccionar para actualizar el valor de la función de alcance sobre los puntos adyacentes. Lo veremos en detalle seguidamente.

Vamos a considerar una superficie donde el intervalo de discretización para las dos dimensiones del espacio es **G** (ver Figura 4-4). Las condiciones de contorno son conocidas en el origen (lo que se muestra como esfera central negra en la figura), esto es, conocemos el valor de la función de alcance **T** en el contorno de la curva inicial, solo que por construcción algorítmica la curva inicial la reducimos hasta confundirse con el punto origen. Es por esta representación inicial por lo que el problema se puede confundir con uno de valor inicial, cuando en realidad es un problema de condiciones de contorno. Ahora necesitamos definir una función de velocidad de avance del frente **F** "(4-7)", que puede estar afectada por diferentes condiciones (locales, globales o independientes), lo que puede ser muy útil en algunos casos de planificación de trayectorias (ver 4.3.6), aunque para otros podremos definir una velocidad tan sencilla como queramos (e.g., **F**=constante), puesto que en realidad al ser el algoritmo **M3R** un esquema geométrico resulta ser independiente del tiempo. A continuación, explicaremos en detalle y de forma gráfica el avance del algoritmo.

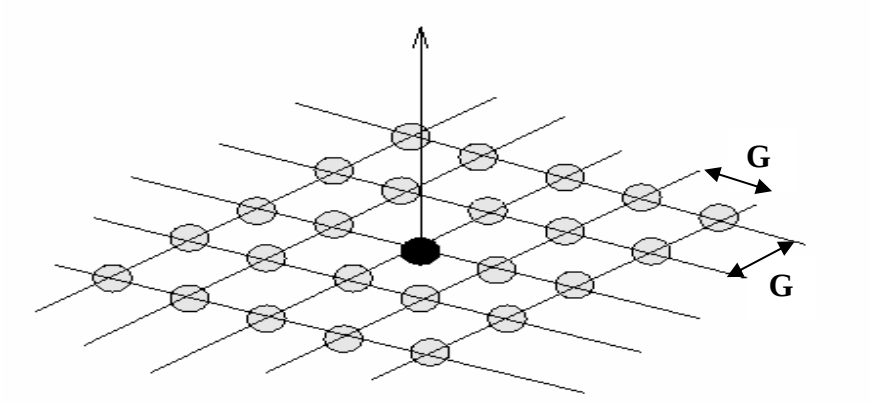


Figura 4-4: Inicio del Algoritmo M3R para un espacio Bidimensional.

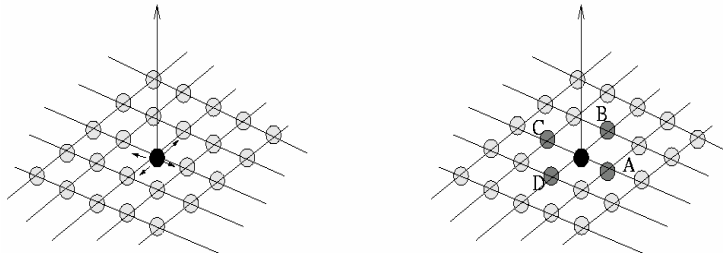


Figura 4-5: Actualización aguas abajo del M3R con computación de posibles valores.

En primer lugar iniciamos la marcha del frente (ver Figura 4-5), computando valores de la función de alcance **T** para los puntos vecinos del origen (resolviendo "(4-17)"). Obtendremos valores provisionales, lo que mostramos en con esferas de color gris.

Entonces (ver Figura 4-6), entre los puntos de valor provisional elegimos aquel que tenga el menor valor de **T** (i.e., menor tiempo de la función de alcance), en este caso el punto **A** (por ejemplo). Esto es debido a que por la propiedad de causalidad ningún punto se puede ver afectado por puntos que tengan valores mayores de **T**. El menor valor de **T** queda fijado como dato definitivamente aceptado por lo que congelamos el punto correspondiente, en este caso el **A** (cambiamos su color a negro). Computamos seguidamente los valores de la función **T** en los puntos vecinos del **A** (no se computan de nuevo los puntos ya aceptados), resolviendo de nuevo la ecuación Eikonal "(4-17)". Estos valores son provisionales, por lo que se marcan como puntos grises.

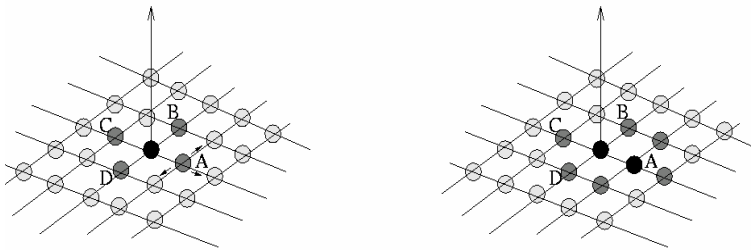


Figura 4-6: Valor aceptado A y actualización del M3R con computación de posibles valores.

Luego (ver Figura 4-7) el algoritmo prosigue de la misma manera, seleccionando de entre los puntos con valores provisionales (conjunto que es ahora mayor) el punto con menor valor de la función **T**, que digamos sea el punto **D** y calculando los valores provisionales de **T** (resolviendo "(4-17)") para los puntos vecinos. Observad que hay un punto ya calculado provisionalmente con **A** que se recalcula ahora para **D**.

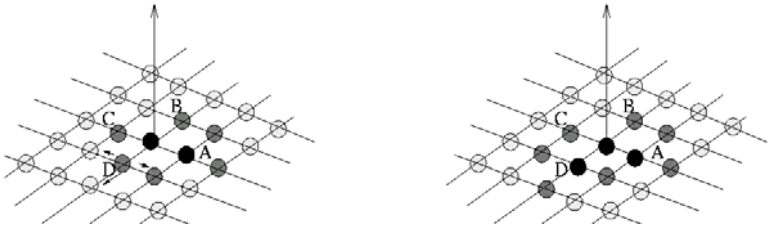


Figura 4-7: Valor aceptado D y actualización del M3R con computación de posibles valores.

El algoritmo repite el proceso descrito hasta que toda el área de interés ha sido barrida por el frente. Podemos resumir, como visión de conjunto, que el algoritmo **M3R** en su desarrollo va construyendo un frente que recorre todo el espacio de configuraciones, utilizando una estructura de datos formada por tres grupos bien diferenciados de valores: el primero es el conjunto de valores aceptados para la función de alcance **T**, el segundo es el conjunto de valores provisionales **T** que constituyen lo que llamaremos "**banda de prueba**" y el tercero es el conjunto formado por el resto de puntos del espacio de configuraciones que todavía no han sido alcanzados por el frente.

Comentaremos ahora el comportamiento del algoritmo **M3R** en presencia de obstáculos. En realidad resulta muy sencillo e intuitivo, puesto que es claro que un frente es expansión no puede alcanzar el interior de los obstáculos, lo que reflejamos con una modificación en el valor de la velocidad "(4-7)", que se anula (i.e., **F=0**) para todos los puntos de la discretización que caen dentro de obstáculos, áreas inaccesibles o regiones desconocidas. Para todos esos puntos, si resolviéramos "(4-17)" tendríamos que el valor de la función de tiempo de alcance es infinito (i.e., **T**=∞). En la práctica, cuanto mayor es el número y tamaño de los obstáculos conocidos, mayor es la velocidad del algoritmo, puesto que ni siquiera tenemos que resolver "(4-17)" para cada uno de ellos, sino que directamente incorporamos todos los puntos afectados con valor infinito de **T** al conjunto de valores aceptados.

Debemos de hacer notar, que aunque el nuevo algoritmo **M3R** pueda presentar en su esquema eficiente de desarrollo para planificación de trayectorias (con valor constante de la velocidad de avance del frente **F**), similitudes con otros algoritmos matemáticos de ordenación y búsqueda (e.g., Grouch-and-Ground), sus fundamentos físicos y significado geométrico lo hacen muy diferente. Recordamos que la función de velocidad **F** "(4-7)" puede depender de factores locales, globales o independientes que se integran en el camino solución de la planificación (e.g., ver el caso de superficies con diferentes adherencias de la Figura 4-11). Además, el algoritmo es válido para múltiples dimensiones y útil para el desarrollo de otras aplicaciones, como por ejemplo análisis computerizado de imágenes.

En conclusión, el esquema eficiente para el **M3R** es un algoritmo geométrico, que para cada punto de un espacio de configuraciones calcula su distancia hasta otro determinado punto, que se define como origen, eso sí, esa distancia se mide teniendo en cuenta las configuraciones no alcanzables.

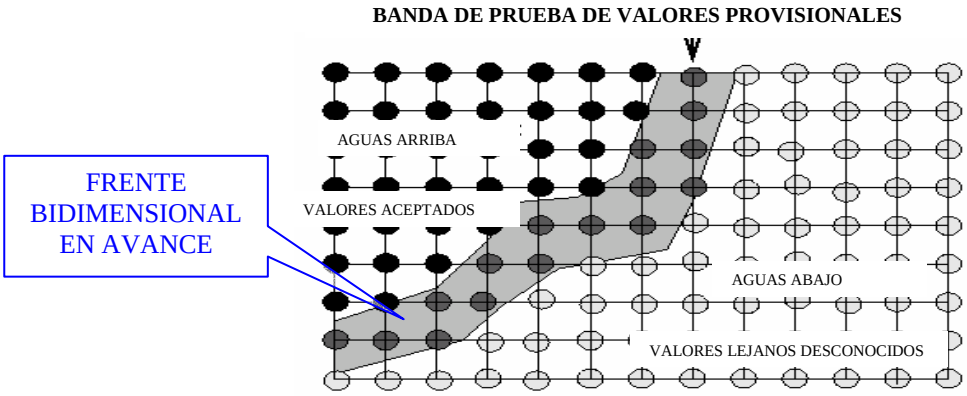


Figura 4-8: Visión del progreso del algoritmo M3R.

4.3.4 Flujograma del esquema Eficiente para el algoritmo M3R.

Todo el esquema geométrico eficiente para el desarrollo del algoritmo **M3R** que hemos descrito con detalle anteriormente, se implementa en una programa software que sigue el flujograma indicado en la Figura 4-9. Como el programa no consta de una única función, no figura como tal en la librería "RobotMan" (ver D), pero podemos encontrarlo en el soporte informático de esta tesis para cada uno de los ejemplos de simulación con realidad virtual y experimentos con el **RH0**.

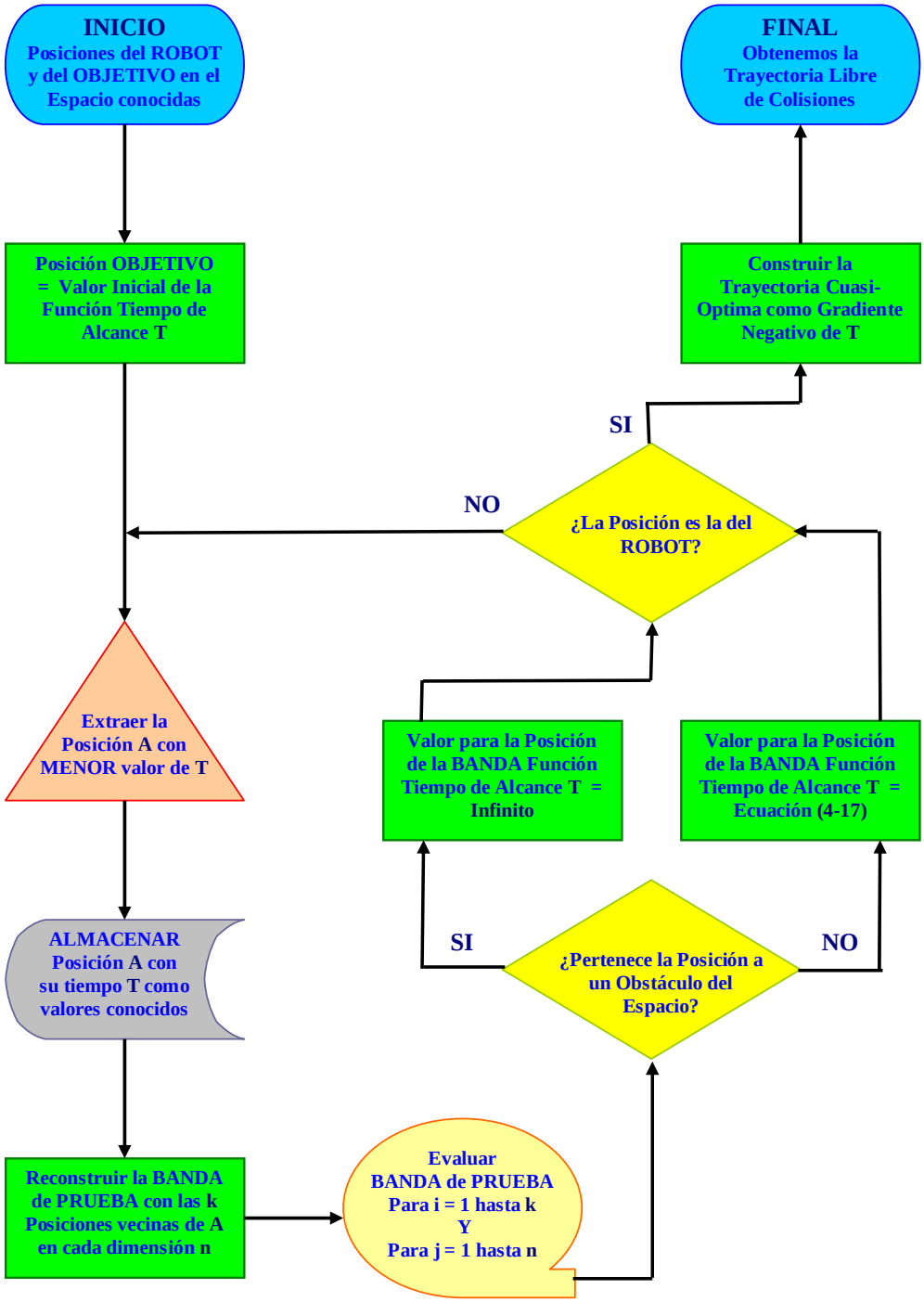


Figura 4-9: Flujograma para el desarrollo del Algoritmo M3R.

4.3.5 Eficiencia Computacional.

El **M3R** que acabamos de presentar (ver 4.3.3) es una técnica adaptativa óptima, que deja la labor computacional de resolver la formulación de condiciones de contorno para un frente en evolución en un orden de complejidad de $\mathbf{O(G^n logG)}$, siendo **G** el número de puntos en cada dimensión, que viene dado por el intervalo de discretización elegido y siendo **n** el número de dimensiones del espacio de configuraciones en el que se desarrolla el frente.

A primera vista, la eficiencia computacional del **M3R** no parece tan evidente, pero ventajas adicionales le proporcionan un ahorro computacional muy grande:

- No tiene restricciones de tiempo, por lo que la velocidad real del frente **F** es irrelevante para la eficacia del método y la podemos ajustar a nuestra conveniencia para facilitar la rapidez. Y dado que la velocidad de avance del frente **F** es irrelevante para el desarrollo del algoritmo, la resolución de la función de alcance **T** para cada punto resulta trivial.
- El número de elementos en la banda de prueba depende de la longitud del frente y en muchos casos, esta longitud es lo suficientemente pequeña como para considerar que a efectos prácticos la necesaria búsqueda del elemento menor dentro de la banda, es de orden **O(1)**.
- Para aplicaciones de planificación de trayectorias libres de colisiones tiene la ventaja de que no es necesario desarrollar el algoritmo en las celdas ocupadas por obstáculos, esto es, cuanto más numerosos y grandes sean los obstáculos, mayor será la velocidad del algoritmo.

Un punto clave del **M3R** descansa en encontrar un modo eficaz de localizar el punto de la banda de valores de prueba con el menor valor de la función de alcance **T**. Podemos usar una estructura de pila con punteros para almacenar los valores de **T** de la banda de prueba. Una pila de este tipo es un árbol binario completo en el que cada valor es menor o igual al de sus dos hijos. Esta estructura de padre en la posición **z** e hijos en las posiciones **2z** y **2z+1** se puede implementar en una cadena simple, en la que se puede localizar al padre de un hijo con un acceso de **O(1)**, de forma que la localización del punto con menor valor de **T** puede llevar como máximo un tiempo **O(logG)**.

Considerando todo lo anterior, de forma práctica cada punto de la red se visita una vez, de modo que en muchos casos la eficiencia computacional del Método Modificado de Marcha Rápida puede ser de orden $\mathbf{O(G^n)}$, siendo **G** el número de puntos en cada dimensión y **n** el número de dimensiones del espacio de configuraciones, lo que supone un rendimiento realmente muy bueno.

4.3.6 Aplicación del M3R para la Planificación de Trayectorias.

Un aspecto de la solución de viscosidad para ecuaciones de conservación hiperbólica (i.e., la curvatura de un frente en evolución), es que estas soluciones extraen de entre todas las posibles las que corresponden a la primera llegada de información desde una perturbación inicial. Teniendo esto en cuenta y dada una posición inicial y final en un determinado dominio, podemos utilizar el algoritmo **M3R** para encontrar el camino más corto desde el estado inicial al final para esa determinada métrica.

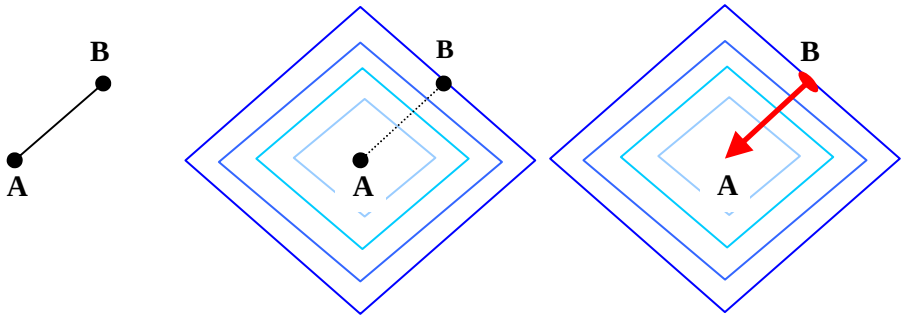


Figura 4-10: Expansión de un frente M3R y planificación del camino en el plano.

Continuando con el ejemplo bidimensional simple (ver Figura 4-10) sin obstáculos, tenemos un punto inicial en el plano **A** y un punto final como objetivo **B**. Desarrollamos el algoritmo **M3R** para la expansión del frente (según 4.3.3). Entonces la construcción explícita del camino más corto hasta el punto **A** vendrá de la solución dada por el **"gradiente negativo"** de la función de alcance, con valor inicial igual al punto **B**. En realidad, como el **M3R** genera una función no diferenciable pero continua, lo que llamamos "gradiente negativo" es en realidad la serie de valores decrecientes máximos de dicha función de alcance, lo que se consigue de forma trivial al no tener la función ningún mínimo local.

El ejemplo bidimensional siguiente (ver Figura 4-11) muestra como se puede aplicar la misma idea utilizando diferentes valores de velocidad **F** "(4-7)" en el espacio de configuraciones, en función de las condiciones locales. Permite calcular la trayectoria óptima entre una posición inicial y un destino final teniendo en cuenta que una superficie puede estar nevada (la azul oscura de la izquierda) en la que el movimiento se hace más difícil y otra seca (la azul clara de la derecha) por la que el tránsito es más fácil. Obsérvese que la trayectoria óptima en este caso no es la de menor longitud.

Se demuestra así una ventaja más de los Métodos de Marcha Rápida frente a otros enfoques, puesto que planifica trayectorias teniendo en cuenta condiciones locales y globales del espacio, además de las geométricas. A la izquierda de Figura 4-11 se muestra el resultado en función sólo de las condiciones locales y a la derecha se muestra la trayectoria óptima en función también de los obstáculos del entorno.

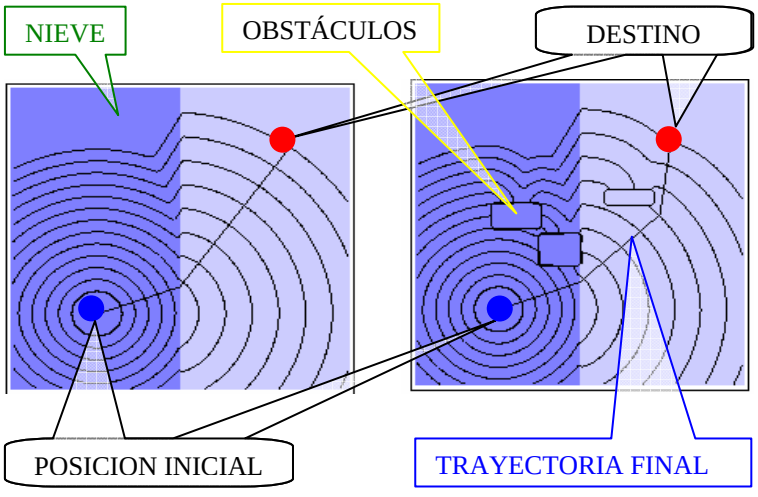


Figura 4-11: Planificación de trayectorias con obstáculos y condiciones locales.

4.3.7 Comparación del M3R con métodos de Campo Potencial.

Los métodos de campos de potencial, con sus numerosas variaciones, son unos de los más investigados para la planificación de trayectorias (ver 4.2.1). Son fáciles de tratar pero tienen los problemas relacionados con la existencia de mínimos locales por lo que no garantizan el alcance del objetivo.

Seguidamente se compara el algoritmo **M3R** con un típico campo de potencial (i.e., funciones cuadráticas de potencial atractivo hacia el objetivo y repulsivo desde los obstáculos). Para ello se plantea un problema de espacio bidimensional, con un robot móvil puntual y un obstáculo cóncavo formado por cinco cilindros interconectados, donde el objetivo consiste en encontrar una trayectoria entre la posición inicial del robot y un punto de destino.

Analizamos el resultado de la aplicación del algoritmo de campo de potencial al problema propuesto. En el gráfico de la izquierda de la Figura 4-12, aparece representado el valor de la función potencial para cada punto del campo. Se utiliza esta función potencial para la planificación de la trayectoria del robot siguiendo su gradiente negativo desde el punto inicial dado por la posición presente del robot. En el gráfico de la derecha de la Figura 4-12 podemos observar el resultado del algoritmo, que desafortunadamente no es satisfactorio, puesto que la trayectoria finaliza en un mínimo local. Todo ello a pesar de que el algoritmo de campo de potencial realmente aplicado cuenta con unas rutinas para solventar mínimos locales mediante el desarrollo de rutas aleatorias de salida (en el gráfico se pueden ver en la trayectoria final los intentos de salida del mínimo local). El método probablemente hubiese funcionado para obstáculos convexos, pero en este caso la geometría cóncava del obstáculo provoca que el mínimo local sea tan profundo que el algoritmo no puede encontrar una solución correcta.

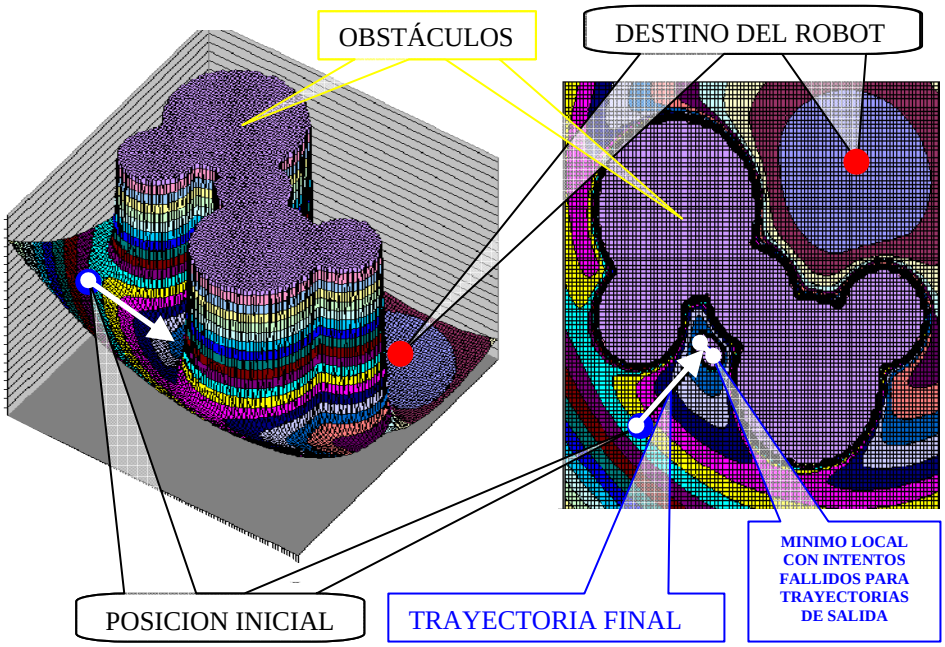


Figura 4-12: Algoritmo de Campo de potencial con problema de mínimo local.

Analizaremos ahora el resultado de aplicar el algoritmo **M3R** para la planificación de trayectorias, al problema de espacio bidimensional con un robot móvil puntual y un obstáculo cóncavo formado por cinco cilindros interconectados.

En el gráfico de la izquierda de la Figura 4-13 aparece representado el valor de la función de tiempo de alcance **T** para cada punto del campo. Se puede observar que el aspecto de la función **T** es el de un frente lineal en avance que presenta formas cuadrangulares o piramidales continuas y siempre crecientes.

Para la planificación de la trayectoria se construye una función o camino siguiendo los valores máximos negativos de la función de alcance **T**, comenzando desde la posición inicial del robot hasta llegar a la posición final de destino. En el gráfico de la derecha de la Figura 4-13 podemos observar el resultado el algoritmo **M3R**, que encuentra el camino entre el origen y el destino, de forma geométrica y directa generando una trayectoria cuasi-óptima. En el gráfico se puede ver que el camino bordea el obstáculo, evitando el paso por la zona que contenía el mínimo local en el algoritmo de potencial. En este caso no hay ningún problema de mínimos locales, como corresponde a la definición teórica de la función de alcance **T** para el algoritmo **M3R** que vimos en 4.3.2.

De esta forma, resolvemos el problema de navegación global en entornos reales bidimensionales (como en este ejemplo) o tridimensionales. El camino que resulta no es dinámicamente factible puesto que no es diferenciable, pero eso aquí no nos preocupa puesto que ese es un problema que resolverá la planificación local de movimientos y el control del robot. Lo que necesitamos para la navegación global es obtener un camino continuo entre el origen y el destino, y eso se consigue de forma geométrica, directa y cuasi-óptima con el **M3R** para cualquier tipo de obstáculos.

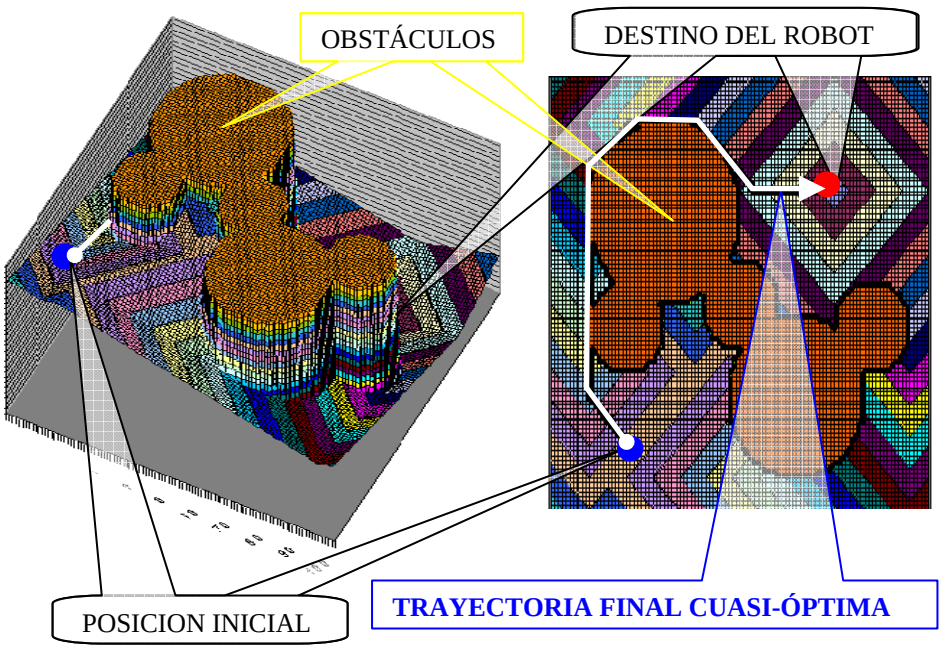


Figura 4-13: M3R para planificación de trayectorias con solución cuasi-óptima.

4.4 *NUEVO MODELO de Navegación – Trayectoria Corporal Global (TCG.)*

EL problema de navegación de un robot humanoide lo vamos a dividir en:

- **Planificación de Trayectorias o Navegación Global**: Este problema computa una trayectoria cinemática libre de colisiones entre la posición presente del humanoide y su posición final. El enfoque para la navegación global que proponemos en esta tesis es el que llamamos nuevo modelo de Trayectoria Corporal Global (**TCG**), que no es sino una aplicación directa del algoritmo **M3R** (ver 4.3) para calcular una trayectoria libre de colisiones del **ZMP** correspondiente al humanoide dentro del espacio Euclídeo tridimensional **SE(3)**, sujeto a las restricciones dadas por el espacio de trabajo.
- **Planificación de Movimientos o Navegación Local**: Este problema recibe como información de entrada el camino global y genera trayectorias para las articulaciones del robot, de forma que el humanoide siga el camino global de una forma mecánicamente estable. Este segundo problema es el que quedó resuelto por el nuevo algoritmo **UPA** que se desarrolla en el apartado 3.3.

Como ejemplo de aplicación del **TCG**, en la Figura 4-14 se pueden ver diez trayectorias cuasi-óptimas libres de colisiones que conectan diez diferentes posiciones iniciales para el humanoide con un objetivo final. Estos caminos son resultado de la ejecución del **TCG** en un espacio muy grande en comparación con el tamaño del robot humanoide, de forma que éste se puede considerar casi puntual. El espacio continen obstáculos de todo tipo (i.e., figuras verdes regulares, cóncavas y totalmente irregulares), lo que no presenta mayor dificultad para el **TCG**. Para entender mejor el funcionamiento del nuevo modelo **TCG**, nos remitimos a los detalles de su implementación en el simulador en 5.4 y con de los experimentos del robot humanoide **RH0** en 6.3.1.

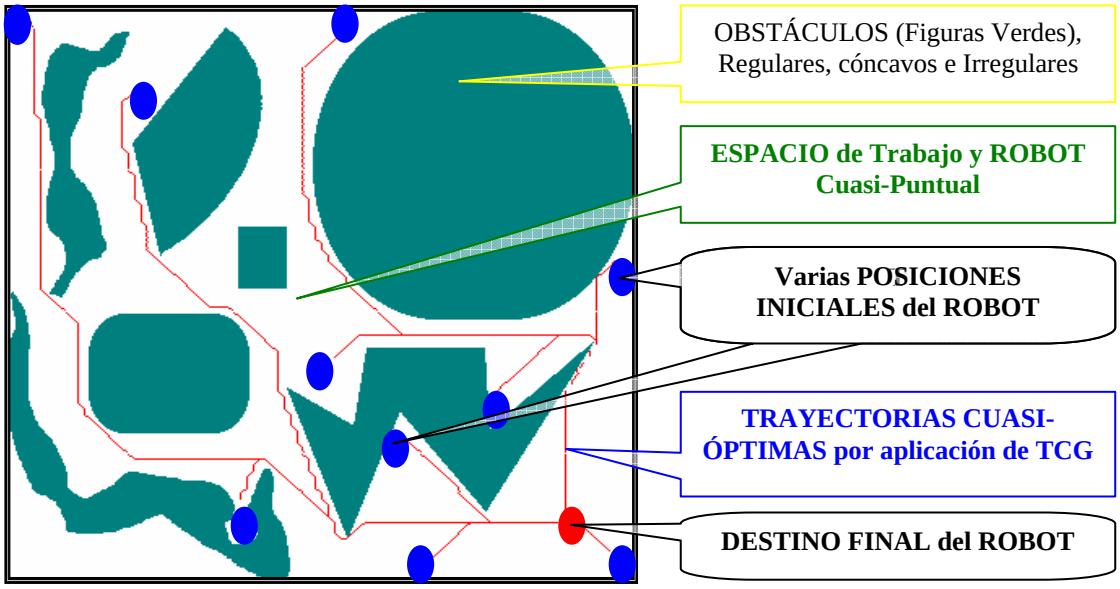


Figura 4-14: Modelo TCG que aplica el M3R para la navegación con obstáculos irregulares.

5 Simulador "RobManSim" para Robots Humanoides con Realidad Virtual.

Para poder desarrollar y probar los nuevos modelos teóricos y algoritmos que se introducen en esta tesis, es imprescindible contar con una plataforma potente de simulación que nos permita ganar productividad y reducir los riesgos y costes inherentes a la experimentación real con el ***RH0****. Para ello, se ha creado* ***el nuevo Simulador "RobManSim"****, que es una Plataforma de Realidad Virtual que reproduce el comportamiento mecánico del* ***RH0*** *en un entorno de trabajo que replica lo más exactamente posible al del robot real. Entre los estándares para la generación de modelos virtuales tridimensionales es escogieron* ***X3D*** *y* ***VRML,*** *sobre todo por su facilidad de integración con el entorno de programación de* ***MATLAB*** *que es la herramienta básica de programación de los sistemas de control que se han construido para el* ***RH0****. Sobre esta plataforma veremos en detalle simulaciones que nos permiten estudiar diferentes comportamientos del humanoide* ***RH0*** *bajo la aplicación de todos los nuevos algoritmos y modelos introducidos en esta tesis.*

5.1 Plataformas Robóticas de Simulación.

La existencia de plataformas computacionales de simulación es fundamental en la investigación robótica, especialmente si es con robots humanoides, puesto que nos permite desarrollar los controladores y la programación necesaria sin necesidad de poner en riesgo el complejo y caro sistema mecánico. En general, los objetivos que los simuladores permiten abordar son:

- Visualizar en tres dimensiones el entorno de trabajo y el modelo del robot en movimiento.
- Proporcionar un centro de pruebas para el desarrollo y evaluación de controles y software del robot.
- Servir como interfaz gráfico del usuario, que puede ser incluso interactivo en tiempo real con el robot.

Un requisito necesario para simulaciones realmente eficaces es que el comportamiento mecánico del robot virtual replique lo más exactamente posible al del robot real, por lo que el trabajo de preparación de una buena **Plataforma de Simulación de Realidad Virtual** resulta crucial. De ese modo, la programación desarrollada sobre el simulador podrá ser heredada por las aplicaciones reales.

Son varios los simuladores para robótica desarrollados en los últimos años (e.g., [27][53][62]), cada uno con su propio alcance, aunque con características comunes. Vamos a comentar detalles de los simuladores y herramientas que han servido de inspiración para el desarrollo del nuevo simulador "**RobManSim**" de esta tesis. Indudablemente el "Open Architecture Humanoid Robotics Platform" (**OpenHRP**), es un proyecto muy ambicioso y de gran alcance, que resulta de referencia obligada. Aunque no es propiamente un simulador de humanoides, la librería de software "**Robotics ToolBox for MATLAB**" también ha sido una referencia, como veremos.

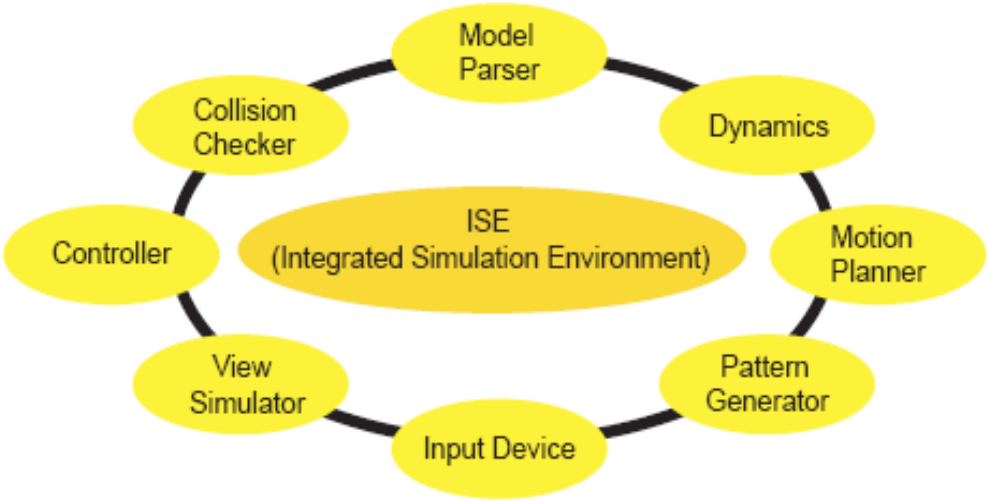


Figura 5-1: Configuración del OpenHRP.

Dentro de un importante programa de investigación impulsado por el Ministerio de Economía e Industria de Japón, se desarrolló la plataforma virtual para simulación de robots humanoides **OpenHRP**. La plataforma se implementa con una arquitectura de objetos distribuidos y su configuración es la indicada en la Figura 5-1. La descripción de los modelos del robot y del entorno se realiza con el formato **VRML** con la función "ModelParser". Cuenta con una función de comprobación de colisiones "CollisionChecker". También tiene una función para la implementación de diversas leyes de control del robot "Controller". Existe una función independiente para soportar la dinámica de los mecanismos robóticos "Dynamics". Por supuesto cuenta con una función de visualización de la simulación "ViewSimulator". Para la introducción de comandos manuales en la simulación, tiene una función de adaptación de dispositivos externos (e.g, joystick) "InputDevice". Finalmente, dos de los módulos fundamentales son el de planificación del movimiento "MotionPlanner", que genera trayectorias libres de colisiones y el de generación del patrón de paso "PatternGenerator" que genera movimientos estables para la locomoción del robot basándose en el control del **ZMP**. Englobando todas esas funcionalidades el **OpenHRP** cuenta con un potente entorno integrado de simulación que podemos ver en la Figura 5-2 y que constituye un interfaz potentísimo, tanto de desarrollo como de usuario. El software de soporte para la plataforma **OpenHRP** fue desarrollado por el departamento de investigación y desarrollo de **Honda**, y es suministrado en gran medida como una caja negra.

Al comentar la casuística del software cerrado frente al abierto es cuando tenemos que destacar el ejemplo de la librería "**Robotics Toolbox**" [27]. Esta librería creada en **MATLAB** proporciona muchas funciones útiles para el desarrollo de algoritmos en aplicaciones de robótica, aunque no está diseñada específicamente para robots humanoides o generación de simuladores de realidad virtual. Sin embargo, es una buena muestra de una librería muy útil con el código completamente abierto.

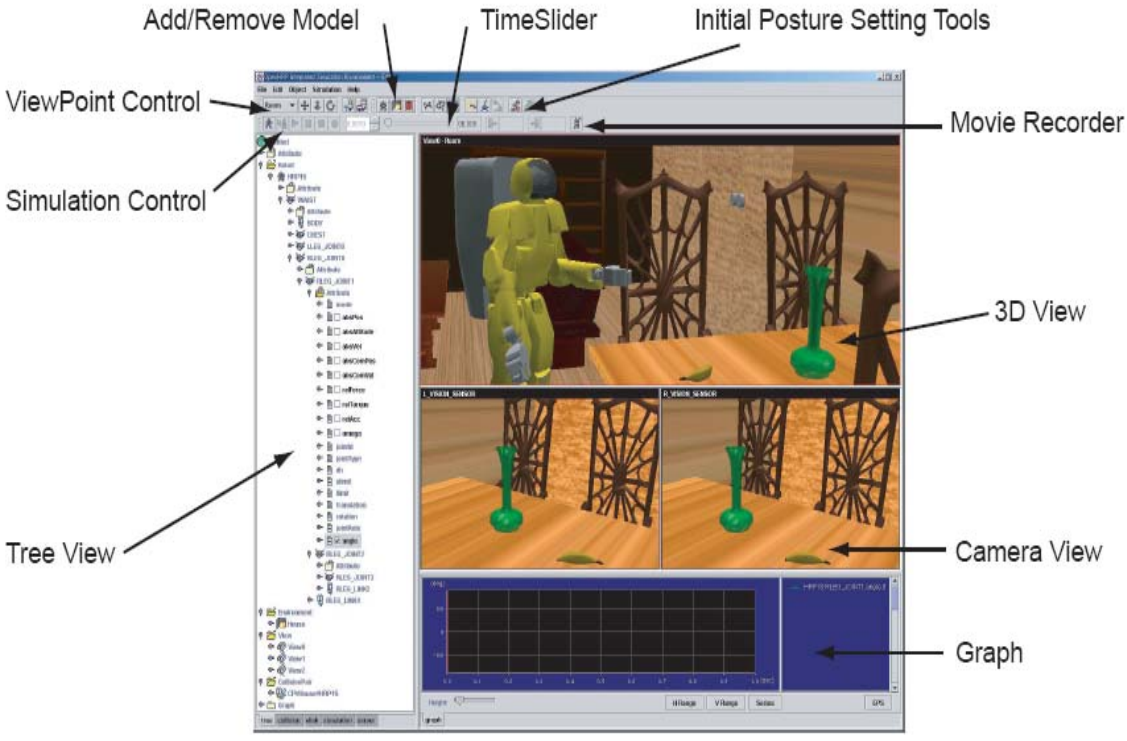


Figura 5-2: Entorno Integrado de Simulación del OpenHRP.

5.2 El Simulador RobManSim - Plataforma de Realidad Virtual.

El Simulador que se ha construido en esta tesis es la Plataforma de Realidad Virtual que hemos llamado **RobManSim** (atendiendo a la definición en inglés de **Rob**ot **Man**ipulation **Sim**ulator). El simulador se ha creado para poder generar y probar los nuevos algoritmos. La verificación de los controladores del humanoide por medio del simulador es una tarea crucial, puesto que para este tipo de máquinas un error en el control real que suponga pérdida de equilibrio puede resultar en daños graves para el mecanismo. Necesitamos que el humanoide virtual presente el mismo comportamiento que su pareja real utilizando los mismos algoritmos de control.

El simulador **RobManSim** se implementa con la arquitectura y configuración mostrada en la Figura 5-3:

- La función de "Modelado" utiliza para la generación de modelos de realidad virtual tridimensional el **ISO X3D** (Extensible 3D Graphics) [37] y usa **VRML** (Virtual Reality Modeling Language) [4] para los modelos de robots humanoides, como podemos analizar con detalle en 5.2.1.

- El "Control" es la función que se implementa con el entorno de programación de **MATLAB** y **SIMULINK** para el sistema de control del mecanismo robótico. La función de "DispositivosEntrada" permite la introducción de comandos manuales desde dispositivos externos (e.g., en esta tesis se utiliza un ratón Magellan de 6 **GDL**). La función "Planificación" es la dedicada a la obtener trayectorias de navegación libres de colisiones que se implementa con el nuevo modelo **TCG** desarrollado en esta tesis. La función "PatrónPaso" para el robot humanoide se genera con el nuevo algoritmo **UPA** presentado en la tesis. Englobando todas estas funciones el simulador **RobManSim** cuenta con un Entorno de Simulación Integrado que veremos en detalle en 5.2.2.

- La arquitectura cuenta con un Interfaz Gráfico de Usuario, que permite realizar prebas de implementación de los algoritmos desarrollados. Las características se discutirán en 5.2.3.

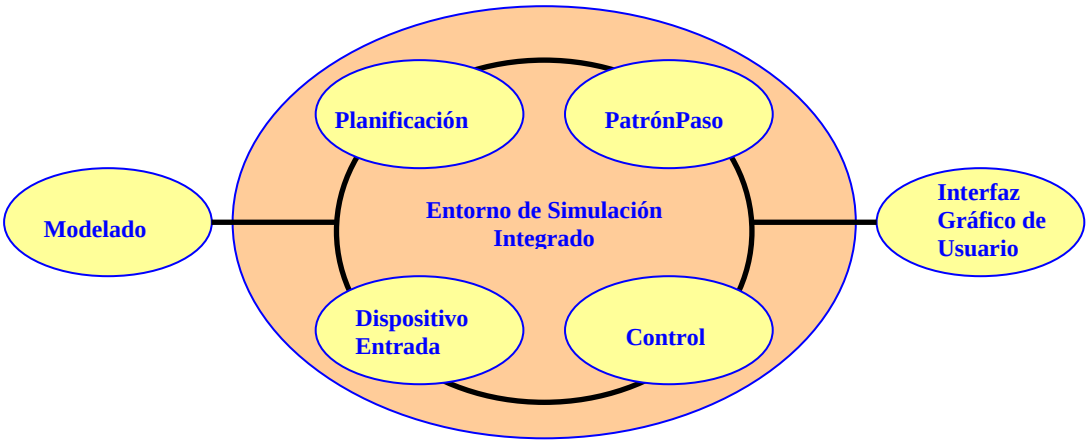


Figura 5-3: Configuración de la Arquitectura RobManSim.

5.2.1 Creación de los Modelos de Realidad Virtual.

Existen en la actualidad diversos estándares para la definición de modelos y objetos tridimensionales en entornos de realidad virtual. El estándar **ISO X3D** que hemos utilizado en esta tesis para la creación de los mundos virtuales soporta varios tipos de ficheros, formatos y lenguajes de programación, proporcionando una magnífica interoperabilidad para representar datos en tres dimensiones, así como flexibilidad para manipular escenas de forma interactiva. El **X3D** es un estándar más refinado y avanzado que su predecesor el **VRML**, que fue el primero en ser utilizado en el desarrollo del simulador de esta tesis y que se ha mantenido como estándar para la descripción de los modelos de robots humanoides, debido a que se integra de forma directa con los módulos de Realidad Virtual de **MATLAB**, que es la herramienta de desarrollo básica que hemos utilizado para probar y depurar los nuevos algoritmos de control que implementan los trabajos de esta tesis.

Dados los estándares elegidos, los modelos para los mundos tridimensionales se desarrollan como ficheros *.x3d* cuyo análisis se puede encontrar con más detalle en Geroimenko y Chen [37], mientras que los modelos de los robots, especialmente el modelo del humanoide **RH0**, quedan encapsulados en ficheros de tipo *.wrl* para cuyos detalles nos referiremos a Ames et al [4]. Al estar estos estándares y lenguajes de programación tan relacionados, no hay problema para exportar los mencionados tipos de fichero de un formato al otro. Hemos dado las referencias para una formación en estos lenguajes de programación y como ilustración del aspecto que presentan estos ficheros fuente, podemos ver en la Figura 5-4 un pequeño pedazo de código para la representación del brazo izquierdo del robot humanoide **RH0**.

```
DEF l_upperarm_ver Transform {
                translation         0 -0.12 0
                children DEF Shape Shape {
                        appearance      Appearance {
                                material  Material {
                                          ambientIntensity    0
                                          diffuseColor        0.8 0.8 0
                                          emissiveColor       0.15 0.15 0
                                          specularColor       0.5 0.5 0.5
                                }
                        }
                        geometry Box {
                                size      0.02 0.25 0.02
                        }
                }
        }
                DEF l_upperarm_extern Transform {
                translation         -0.01 -0.01 0.075
                rotation    -1 0 0  1.5708
                scale       0.001 0.001 0.001
```

Figura 5-4: Tipo de fichero para un Objeto VRML.

El tipo de fichero que representa una modelo de realidad virtual, tal y como acabamos de verlo en la Figura 5-4, puede ser generado con cualquier editor de texto, pero resulta evidente que la productividad en ese caso sería bajísima. Para crear mundos virtuales complejos como los que necesitamos para el simulador de esta tesis, que contienen robots humanoides dentro de un entorno de trabajo habitual para las personas, necesitamos utilizar entornos de desarrollo modernos que incluyan programación orientada a objetos. Existen numerosas herramientas que nos pueden ayudar en este propósito, desde sistemas propietarios del más variado nivel hasta editores gratuitos que podemos encontrar en Internet. La elección del **Editor 3D** dependerá obviamente de los requisitos de nuestro trabajo, ya que resulta muy difícil encontrar paquetes gratuitos que puedan resolver gráficos de decenas de miles de polígonos.

Para los trabajos de esta tesis, resulta fundamental que las geometrías de los objetos del mundo virtual se correspondan con el real, así como resulta de extrema importancia que la estructura del robot humanoide quede exactamente representada en cuanto a sus magnitudes mecánicas. Sin embargo, la calidad de imagen de los gráficos no exige unos requisitos elevados. Como consecuencia, la potencia del **Editor 3D** que necesitamos es mediana y podemos utilizar algunas de las alternativas gratuitas de código abierto que hay en la red. En nuestro caso, y dado que trabajaremos con **MATLAB-SIMULINK** para el desarrollo de los sistemas de control, elegimos el **Editor 3D V-Realm Builder** de realidad virtual que se indica como preferente para la integración con **MATLAB** y que podemos ver en la Figura 5-5. Este editor, como la mayor parte de este tipo de programas, cuenta con barras de herramientas específicas para **3D** y librerías de objetos tridimensionales básicos. Podemos apreciar en la Figura 5-5 el árbol de objetos que constituye el mundo virtual de un entorno de trabajo con el robot humanoide **RH0**, muchos de los cuales son típicos de los ficheros de realidad virtual (i.e. los puntos de vista, el fondo, la información, la navegación), amén de los obstáculos, la casa y el robot **RH0**.

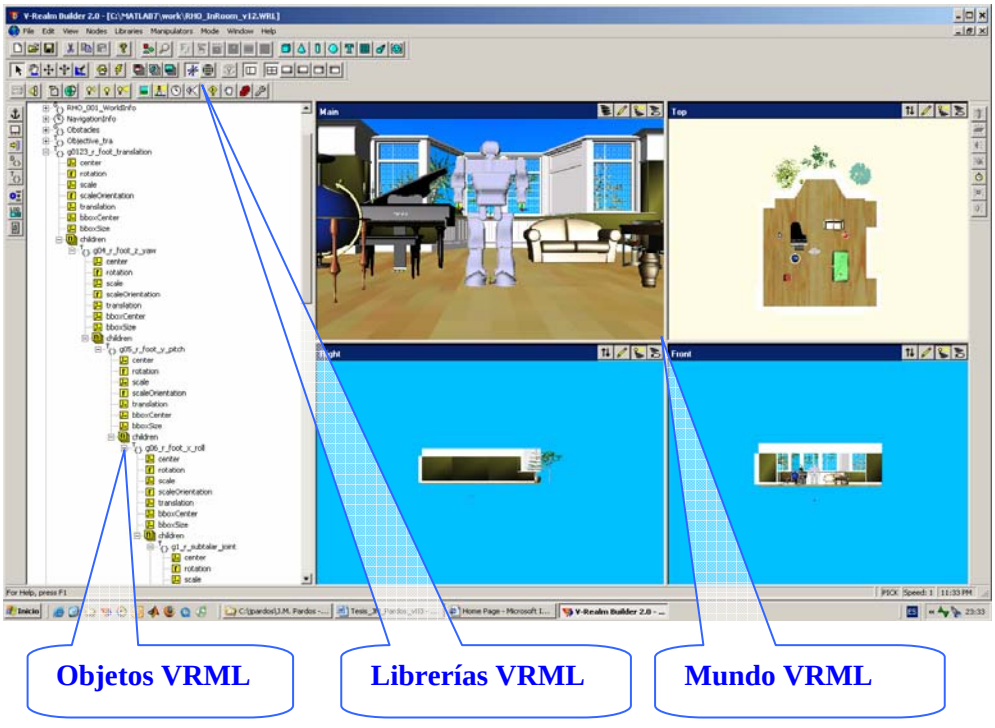


Figura 5-5: Editor de Modelos de Realidad Virtual.

5.2.2 Entorno de Simulación Integrado.

MATLAB es un lenguaje de programación para computación técnica que integra visualización y programación en un entorno de fácil uso, donde los problemas y las soluciones se expresan en una notación matemática familiar que además se presenta con funciones transparentes. Además permite realizar análisis de datos, explorar funciones de ingeniería y visualizar gráficos científicos, incluso con la capacidad de integración de ficheros en formatos de realidad virtual. **SIMULINK** es un paquete de software para el modelado, simulación y análisis de sistemas dinámicos lineales y no lineales, con muestreo continuo, discreto o híbrido de funciones.

Es por todo ello por lo que hemos elegido **MATLAB-SIMULINK** como herramientas de trabajo para la creación de modelos de simulación y desarrollo de prototipos. Como consecuencia, utilizaremos estos paquetes de software para la construcción del Entorno de Simulación Integrado necesario para el desarrollo de los nuevos algoritmos propuestos en esta tesis (ver la Figura 5-6), así como la librería de software **ROBOTMAN** que se presenta en el apéndice-D.

El Entorno de Simulación Integrado cuenta con cuatro áreas fundamentales de trabajo que podemos ver en la Figura 5-6: **Recursos de Trabajo**, **Entorno de Programación**, **Entorno de Sistemas de Control** y **Modelos de Realidad Virtual**.

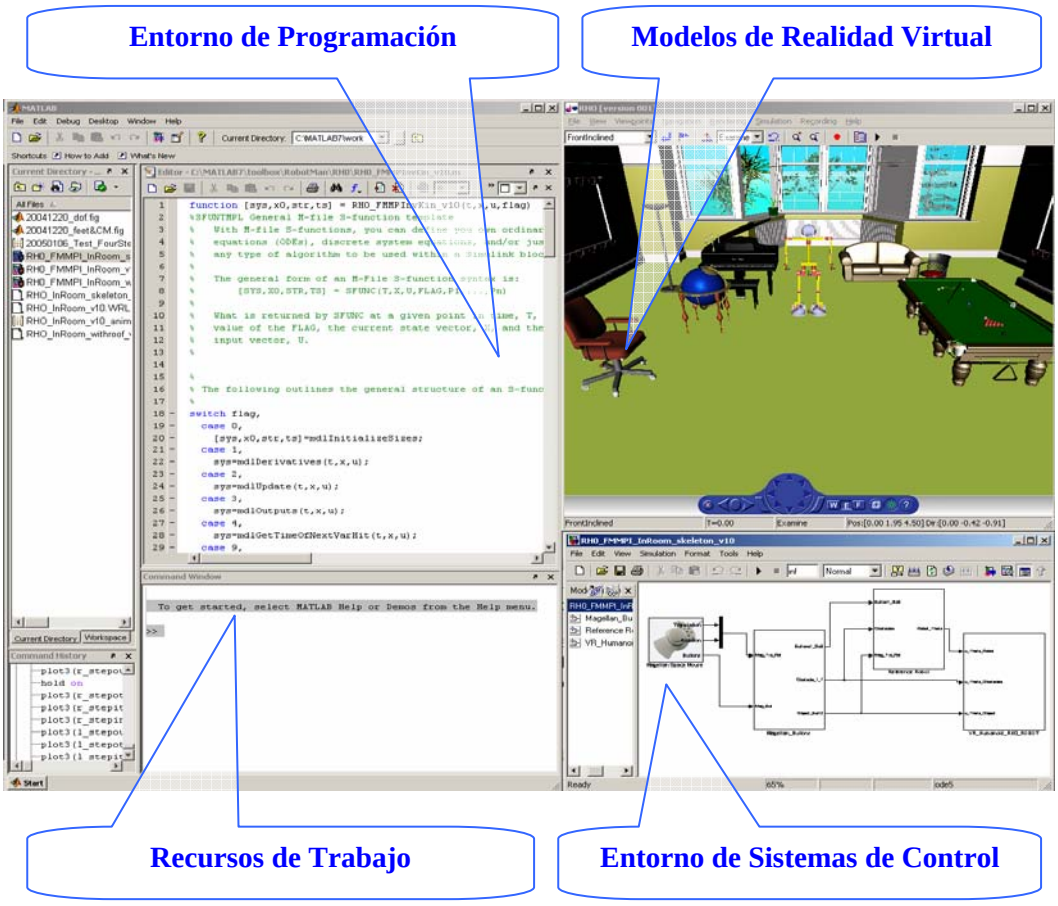


Figura 5-6: Entorno de Simulación Integrado.

Los **Recursos de Trabajo** son los proporcionados por **MATLAB**: directorios de trabajo, paquetes de herramientas, funciones básicas, recursos gráficos y la capacidad de integración para construir el Entorno de Simulación Integrado.

El **Entorno de Programación** que podemos ver en la Figura 5-7 nos aporta todas las herramientas del lenguaje **MATLAB**: lenguaje productivo, funciones transparentes, librerías especializadas y herramientas de depuración. En este entorno, desarrollamos los nuevos algoritmos de esta tesis y refinamos los programas de control. Así mismo nos sirve también para desarrollar la librería de software **ROBOTMAN** que se presenta en el apéndice-D, que constituye la estructura básica para la construcción de los algoritmos y controles.

El **Entorno de Sistema de Control** emplea las herramientas de **SIMULINK** para crear los diferentes bloques que constituyen los controles para la dinámica del sistema mecánico constituido por el robot humanoide y su entorno. En la Figura 5-8 podemos ver un diagrama de bloques de un sistema típico para la ejecución del modelo de trabajo del humanoide en el Entorno de Simulación Integrado. El Entorno de Sistema de Control tiene cinco bloques bien diferenciados:

- Una función "DispositivosEntrada" para un Ratón Magellan que proporciona referencias en seis dimensiones (i.e. tres posiciones y tres rotaciones).
- Un bloque de Activación de Referencias, tanto para el objetivo hacia el que el robot debe dirigirse como para el movimiento de los obstáculos del entorno.
- Un Sistema de Control que contiene las llamadas a los nuevos algoritmos de la tesis (contiene las funciones de "Planifiación", "PatrónPaso" y "Control").
- Un enlace del control con el Modelo de Realidad Virtual que contiene al robot humanoide y su entorno, con el Interfaz Gráfico de Usuario.
- Un paquete de variables de entrada modificables antes y durante la simulación: Lp, Hp, Wp, Gcm, Hcm, Kcm, RAas, Pocm, Ca.

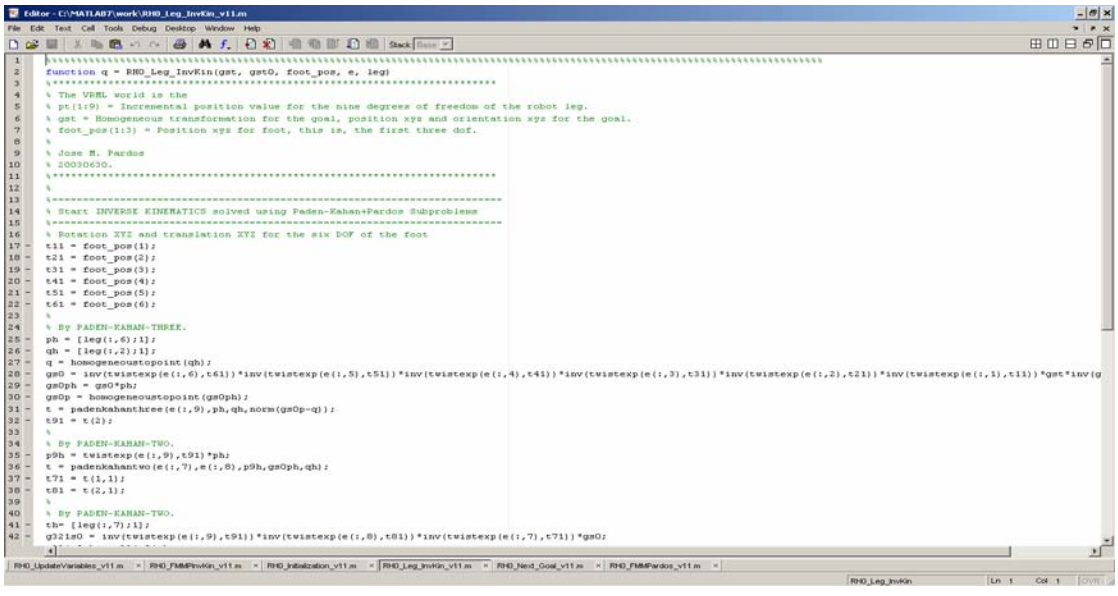


Figura 5-7: Entorno de Programación.

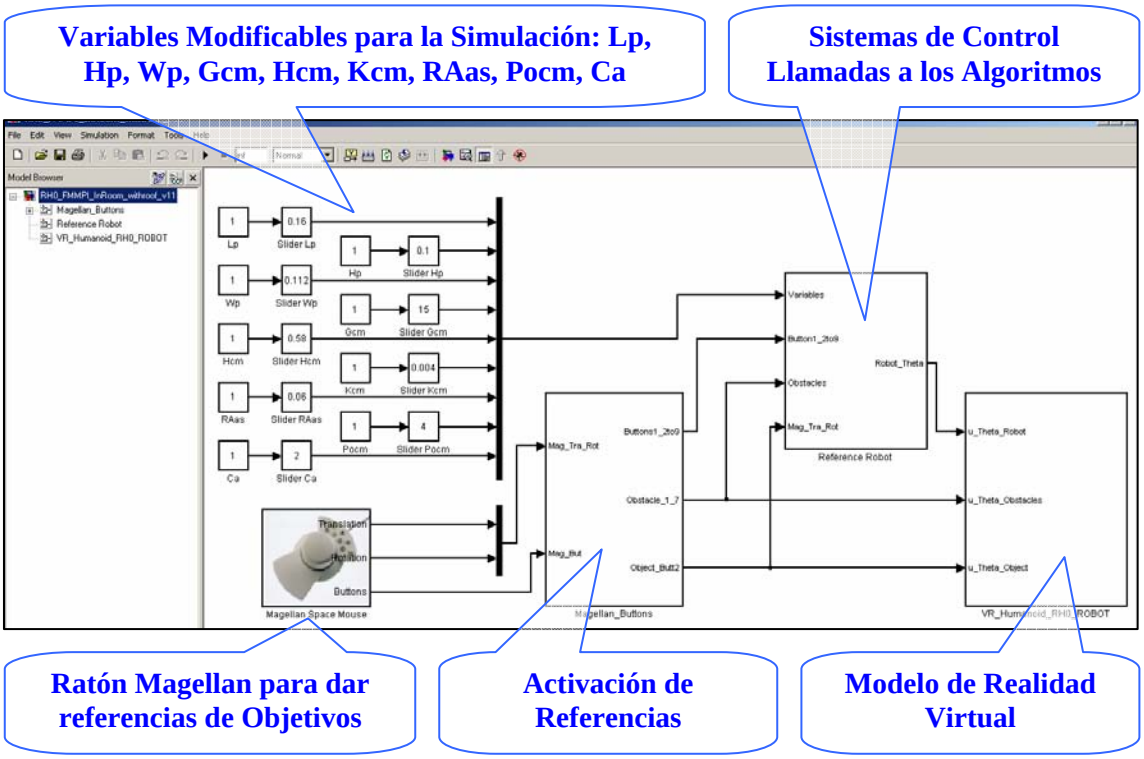


Figura 5-8: Entorno de Sistemas de Control.

Una entrada para un Ratón Magellan que proporciona referencias de objetivos en seis dimensiones (i.e. tres posiciones y tres rotaciones), envía señales que pasan al bloque de Activación de Referencias. En ese bloque, como vemos en la Figura 5-9, la información recibida sirve para mover tanto el objetivo hacia el que el robot debe dirigirse como los obstáculos del entorno, tanto en su posición como en su rotación, en función de la activación de los botones axuliares del ratón.

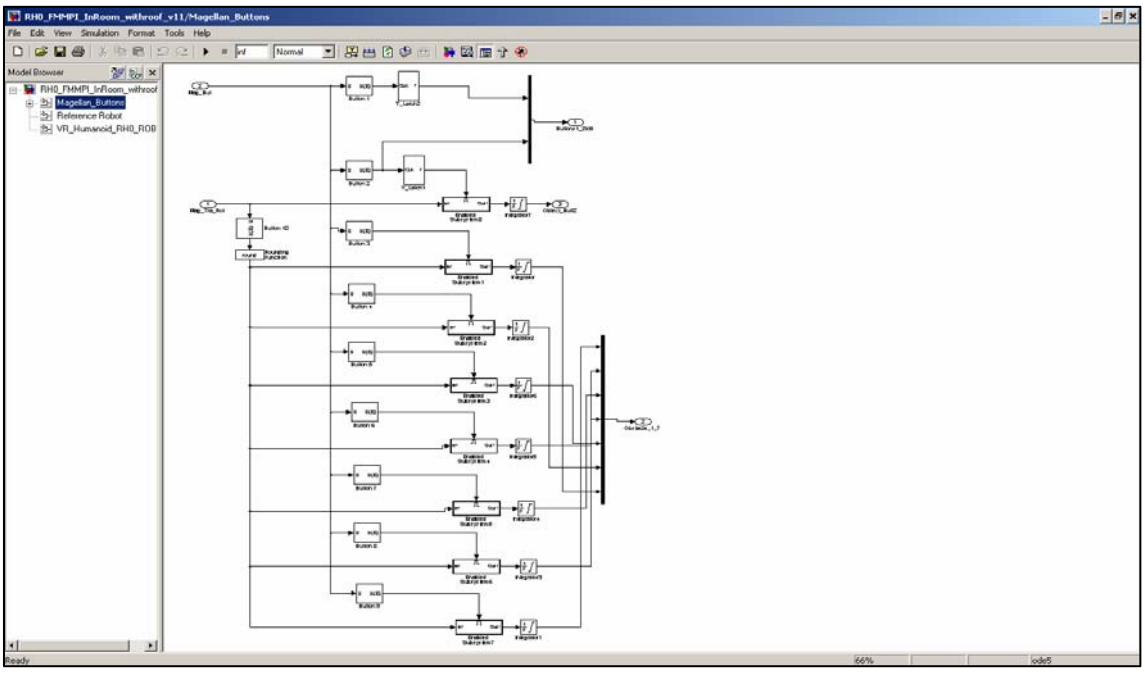


Figura 5-9: Sistemas de Control – Activación de Referencias.

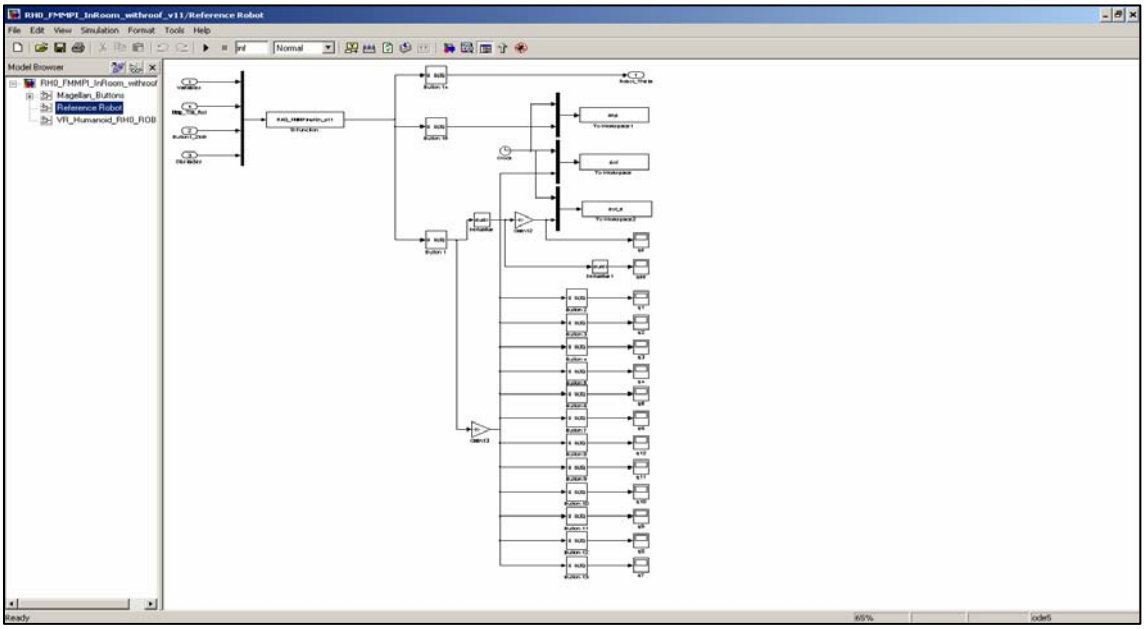


Figura 5-10: Sistemas de Control – Llamadas a los Algoritmos.

El Sistema de Control contiene las llamadas a los nuevos algoritmos de la tesis, tanto para el control de Locomoción como para la Navegación. Además, el bloque que podemos ver en detalle en la Figura 5-10, envía al entorno de trabajo los valores de las variables de salida que nos interesan: posiciones y velocidades de las articulaciones del robot, así como posiciones del **CM** y los pies del humanoide.

Un enlace del sistema de control con el Modelo de Realidad Virtual que contiene al robot humanoide y su entorno, que es lo que nos proporciona el bloque detallado en la Figura 5-11, nos permite ejecutar las simulaciones complejas y grabar los resultados tanto numéricos como en formato de imagen virtual.

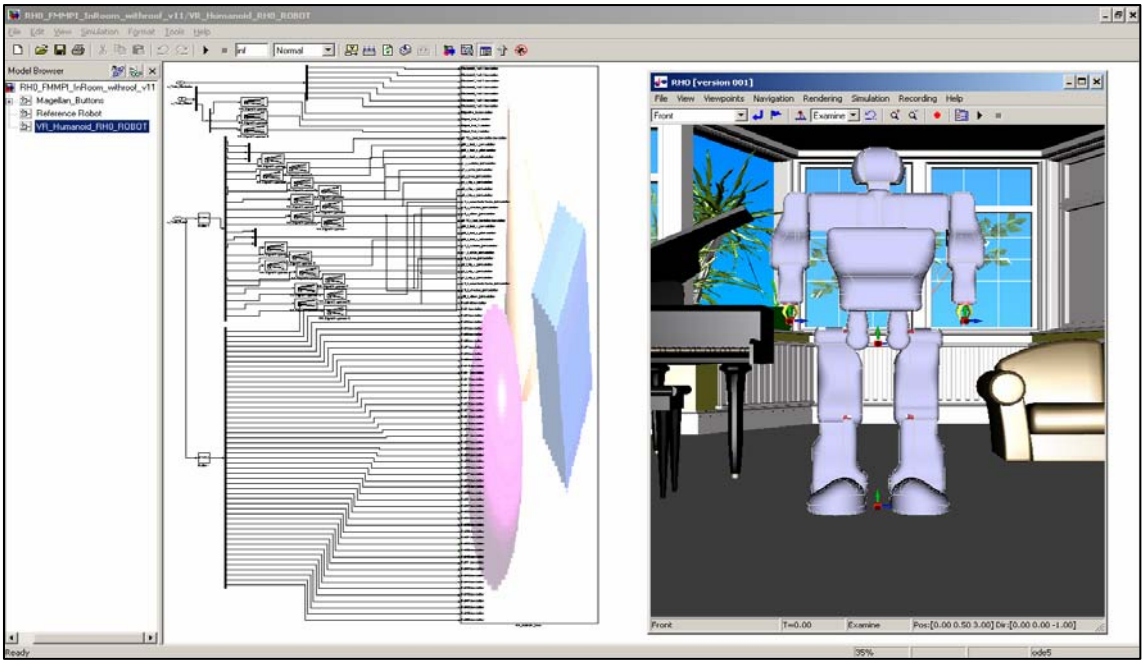


Figura 5-11: Sistemas de Control y Modelos de Realidad Virtual.

5.2.3 Interfaz Gráfico de Usuario para el Simulador.

Uno de las tareas básicas del Entorno de Simulación Integrado es el enlace con un **Interfaz Gráfico de Usuario** cuyo objetivo es proporcionar una visión integrada de la situación mecánica del robot y sus actividades dentro de su entorno de trabajo, sin la necesidad de recurrir a detalles de bajo nivel, esto es, con la posibilidad de experimentar diferentes simulaciones simplemente manipulando las variables de alto nivel que afectan al humanoide y su entorno.

El Interfaz Gráfico de Usuario, como podemos ver en la Figura 5-12, se compone de unas ventanas gráficas muy intuitivas, fáciles de navegar y orientadas a un sencillo control de la simulación por parte del usuario. La ventana principal contiene al mundo de Realidad Virtual junto con el modelo mecánico del humanoide e incluye funcionalidades para establecer los parámetros de simulación, arrancar y parar la ejecución de las simulaciones o grabar los resultados de las mismas. Otras ventanas son fundamentalmente controles de las variables principales que el usuario puede modificar para alterar los parámetros relevantes de la simulación y que son: Longitud de paso **Lp**, Altura de paso **Hp**, Anchura de paso **Wp**, Giro máximo del centro de masas **Gcm**, Altura del centro de masas **Hcm**, Coeficiente cosenoidal para las trayectorias **Kcm**, Radio del área de soporte **RAas**, Potencial para generación de trayectorias **Pocm** y Cadencia **Ca**.

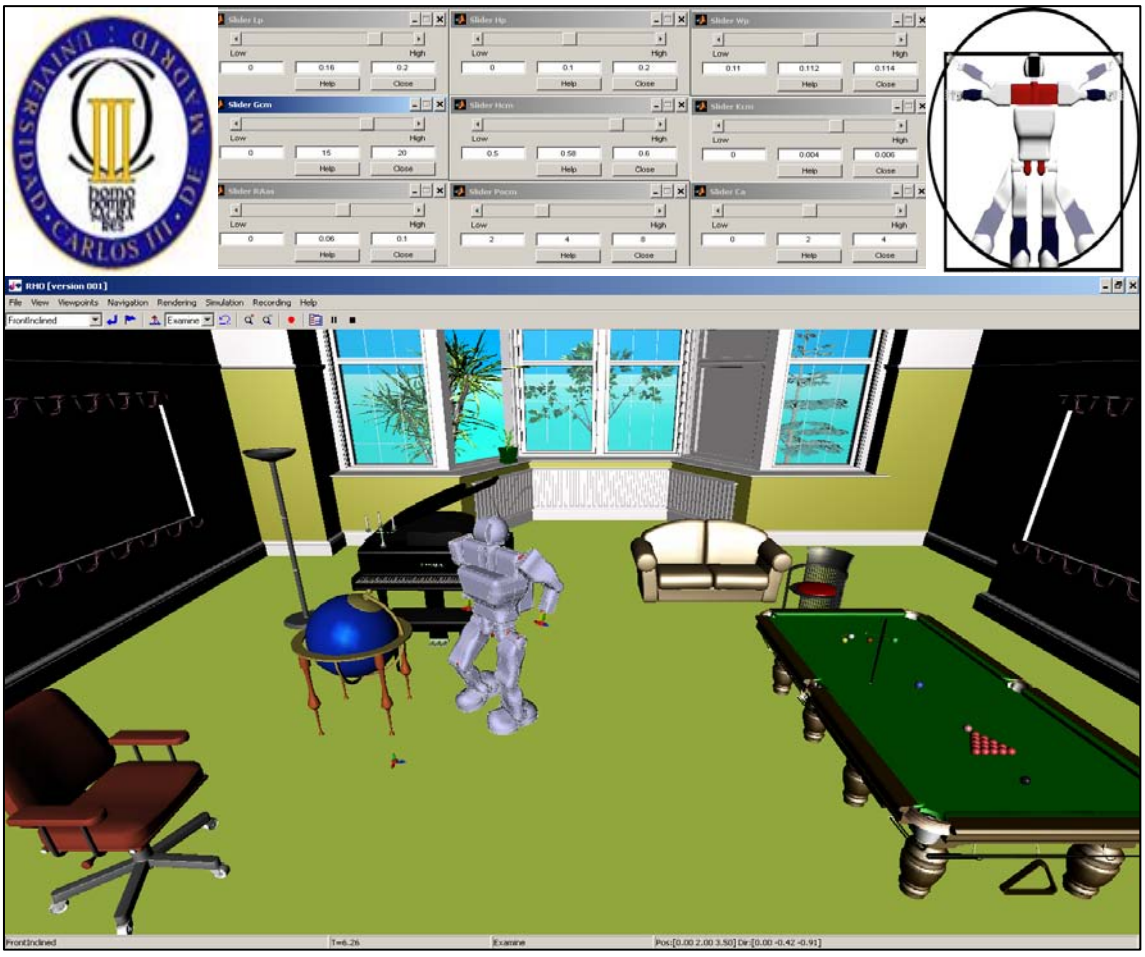


Figura 5-12: Interfaz Gráfico de Usuario para el Simulador.

5.3 Locomoción Bípeda del Humanoide RH0.

Par resolver la planificación de movimientos del humanoide **RH0** que realiza una locomoción bípeda, aplicaremos el nuevo algoritmo **UPA** (ver 3.3). Como ya vimos en detalle en 3.5, el **UPA** recibe como entradas el objetivo local hacia el que se debe caminar, el modelo del entorno local (que en este caso es una habitación con una serie de objetos: piano, sofá, mesa de billar, silla, sillón, lámpara y bola planetaria) y las características típicas para la locomoción bípeda deseada del robot (i.e., **Lp**, **Hp**, **Wp**, **Gcm**, **Hcm**, **Kcm**, **RAas**, **Pocm** y **Ca**.). Los resultados obtenidos en la ejecución del algoritmo **UPA** son los datos correspondientes, en primer lugar a las trayectorias de los **GDL** virtuales $\boldsymbol{\theta v_{PM}}$ del pie móvil (**PM**) durante el ciclo del paso (i.e., trayectoria de la posición y rotación del **PM**), y en segundo lugar a las trayectorias de los **GDL** virtuales $\boldsymbol{\theta v_{CM}}$ del **CM** del humanoide durante el ciclo del paso (i.e., trayectoria de la posición y rotación del **CM**, que define el movimiento global del humanoide). Los resultados respetan las restricciones dadas para el equilibrio estático y/o dinámico 3.1.1 de la locomoción bípeda, esto es, que $\mathbf{P_{CM}}$ "(3-1)"o el **ZMP** "(3-5)"se encuentren en todo momento dentro del área de soporte.

El algoritmo **UPA** (ver 3.3.1) desarrolla en cinco fases cada ciclo del paso teórico para la locomoción bípeda de un humanoide genérico (según 3.1.): Orientar, Inclinar, Elevar, Apoyar y Balancear. No obstante, en la aplicación práctica del algoritmo **UPA** para el robot humanoide **RH0** apreciamos que un movimiento complejo que no puede ser desarrollado como repetición de un único paso elemental, sino que en realidad son necesarios cuatro tipos básicos para poder lograrlo (ver 3.5.2): Paso de Salida, Paso de Zancada, Paso de Entrada y Paso de Giro. Para obtener los resultados buscados en la aplicación al robot **RH0** (i.e., los caminos factibles para el **PM** $\mathbf{Q_{rPM}}$ y para el **CM** $\mathbf{Q_{rCM}}$), desarrollaremos exactamente las mismas cinco fases para cada uno de los tipos de paso, que serán seleccionados teniendo en cuenta el entorno de trabajo del robot y la configuración del humanoide **RH0** en cada etapa de la locomoción bípeda. En los siguientes puntos de este capítulo se van a presentar simulaciones que resuelven problemas de locomoción que incluyen todas las posibilidades de paso para el **RH0**.

Aunque nuestro objetivo es resolver locomociones bípedas complejas, tenemos que recordar que el algoritmo **UPA** resuelve exclusivamente un solo paso del humanoide **RH0**. Una primera razón para ello es que en la experimentación real con el robot, tras cada ejecución de un paso es conveniente integrar información sensorial del entorno antes de dar el siguiente (i.e., antes de ejecutar el algoritmo **UPA** de nuevo), pudiendo evitar obstáculos de una forma más eficiente. Una segunda razón es el objetivo de elegancia y simplicidad que se ha tratado de imprimir en todos los desarrollos de esta tesis en la búsqueda de soluciones eficaces, de modo que en este caso en lugar de resolver una locomoción bípeda compleja de forma completa, el algoritmo **UPA** permite resolverla de modo mucho más eficaz como encadenamiento de soluciones genéricas al problema de un solo paso del humanoide **RH0**.

Tras obtener los resultados de las ejecuciones del algoritmo **UPA** (i.e., $\mathbf{Q_{rCM}}$ y $\mathbf{Q_{rPM}}$), generamos el movimiento de locomoción bípeda del robot humanoide **RH0** resolviendo el problema cinemático inverso mediante técnicas de Lie, como se han implementado matemáticamente (ver 3.4.1 y 3.4.2) para el humanoide **RH0.**

5.3.1 El robot RH0 caminando cuatro pasos en línea recta.

Para esta simulación partimos de un entorno de Realidad Virtual que incluye al humanoide **RH0** inmerso en un entorno de trabajo constituido por una habitación con varios objetos, como se ve en la Figura 5-13. El objetivo de esta locomoción bípeda para el **RH0** es dar cuatro pasos en línea recta partiendo de una posición erguida y estable de reposo. Para ejecutar el movimiento propuesto el robot necesita utilizar tres de los tipos de paso básicos del algoritmo **UPA** (i.e. paso de salida, paso de zancada y paso de entrada), puesto que el movimiento completo quedaría como sigue: robot en posición de reposo, bajar nivel del **CM**, paso derecho de salida, zancada izquierda, zancada derecha, paso izquierdo de entrada y subida del **CM**, con lo que el robot queda de nuevo en posición erguida. Las características de la locomoción bípeda del **RH0** se han elegido de entre las posibles según su mecánica (i.e. respetando los límites cinemáticos y dinámicos del robot - ver apéndice B), pero de forma que el paso se ejecute de una forma natural en menos de 3sg. Para esta simulación las características son:

- **Longitud de Paso L_p** = 0,160m, esto es **Longitud de Zancada L_Z** = 0,320m.
- **Altura de Paso H_p** = 0,100m.
- **Anchura de Paso W_p** = 0,112m.
- **Altura del centro de masas Hcm** = 0,58m.
- **Coeficiente Cosenoidal para las trayectorias Kcm** = 0,004m.
- **Radio del área de soporte RA_{AS}** = 0,06m.
- **Potencial para generación de trayectorias Pocm** = 4.
- **Cadencia Ca** = 0,3429pasos/sg.
- **Discretización de los datos (necesaria por Hardware y Drivers) S_D** = 0,036sg.

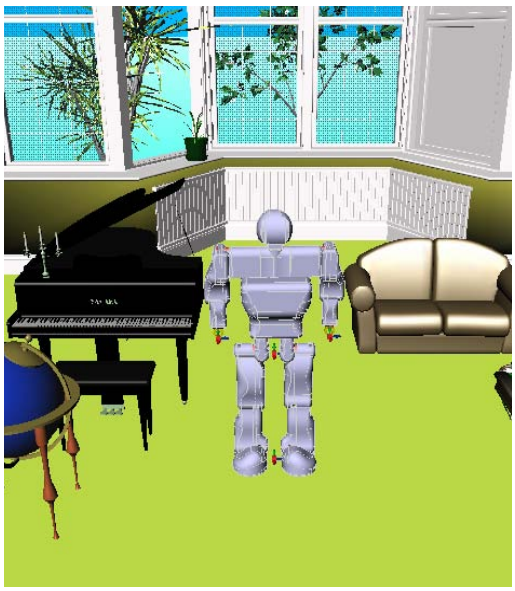


Figura 5-13: El RH0 dentro de un entorno de trabajo de Realidad Virtual.

Dados los valores de **Ca** y $\mathbf{S_D}$ **cada paso del humanoide RH0 empleará en desarrollarse 2,916sg** y cada ejecución del algoritmo **UPA** y que resuelve ese paso generará una **tabla con 81 valores para cada GDL** del robot. Mostramos en 3.3.2 que en las cinco fases del **UPA** se obtienen los caminos factibles para un paso. Aquí simulamos cuatro ejecuciones consecutivas del **UPA** que se convierten en cuatro pasos del **RH0**. Esta locomoción viene precedida por un tiempo de paso dedicado a bajar el **CM** desde su posición con el robot completamente erguido y va seguida por otro tiempo de paso para subirlo. Así, los caminos contienen 30 puntos principales:

- $\mathbf{Q_{rCM}}$ **- Camino Factible del CM**: Trayectoria compuesta por configuraciones $\mathbf{q_{CMi}}$ del centro de masas, según "(5-1)".

$$Q_{rCM} = \{q_{CM1} \ldots q_{CM2} \ldots q_{CM3} \ldots q_{CM4} \ldots q_{CM5} \ldots q_{CM26} \ldots q_{CM27} \ldots q_{CM28} \ldots q_{CM29} \ldots q_{CM30}\} \quad \textbf{(5-1)}$$

- $\mathbf{Q_{rZ}}$ **- Camino Factible del pie izquierdo**: Trayectoria compuesta por configuraciones $\mathbf{q_{Zi}}$ del pie izquierdo según "(5-2)". Viene dada por el camino $\mathbf{Q_{rPM}}$ resultado del **UPA**, cuando es el pie izquierdo el móvil.

$$Q_{rZ} = \{q_{Z1} \ldots q_{Z2} \ldots q_{Z3} \ldots q_{Z4} \ldots q_{Z5} \ldots q_{Z26} \ldots q_{Z27} \ldots q_{Z28} \ldots q_{Z29} \ldots q_{Z30}\} \quad \textbf{(5-2)}$$

- $\mathbf{Q_{rD}}$ **- Camino Factible del pie derecho**: Trayectoria compuesta por configuraciones $\mathbf{q_{Di}}$ del pie derecho, según "(5-3)". Viene dada por el camino $\mathbf{Q_{rPM}}$ resultado del **UPA**, cuando el pie móvil es el derecho.

$$Q_{rD} = \{q_{D1} \ldots q_{D2} \ldots q_{D3} \ldots q_{D4} \ldots q_{D5} \ldots q_{D26} \ldots q_{D27} \ldots q_{D28} \ldots q_{D29} \ldots q_{D30}\} \quad \textbf{(5-3)}$$

Las trayectorias completas para los caminos factibles del cuerpo y pies del **RH0** se obtienen por interpolación (según 3.5.2), quedando según vemos en la Figura 5-14.

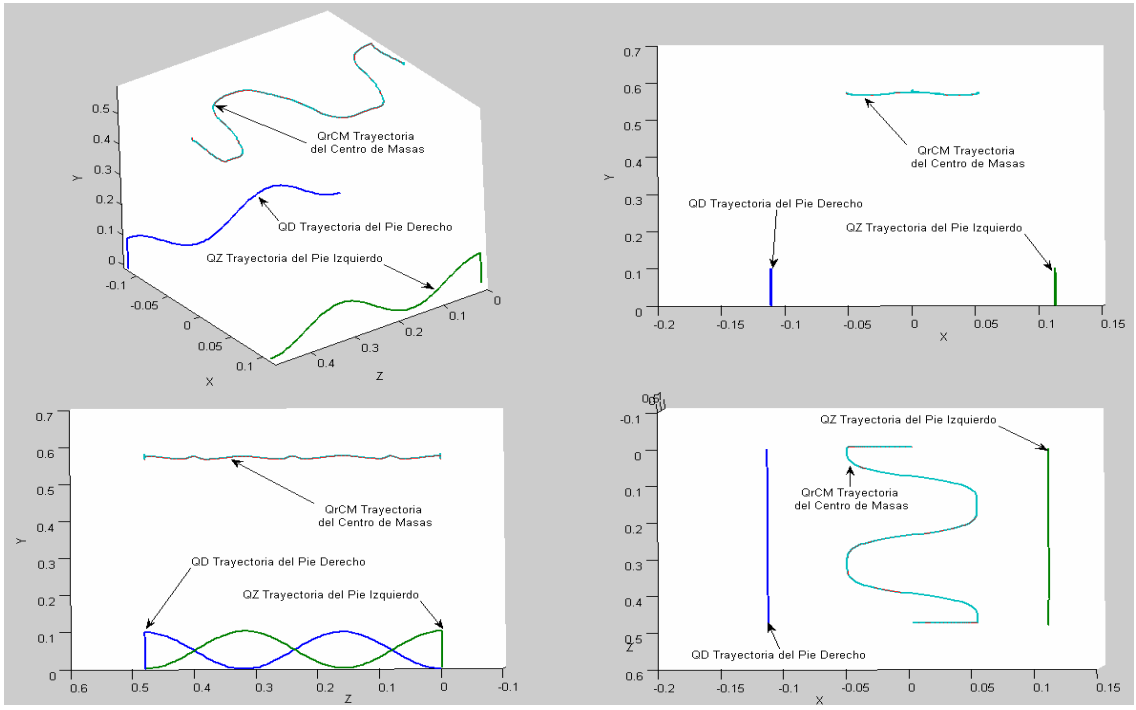


Figura 5-14: Trayectorias de pies y CM del RH0 caminando en línea recta.

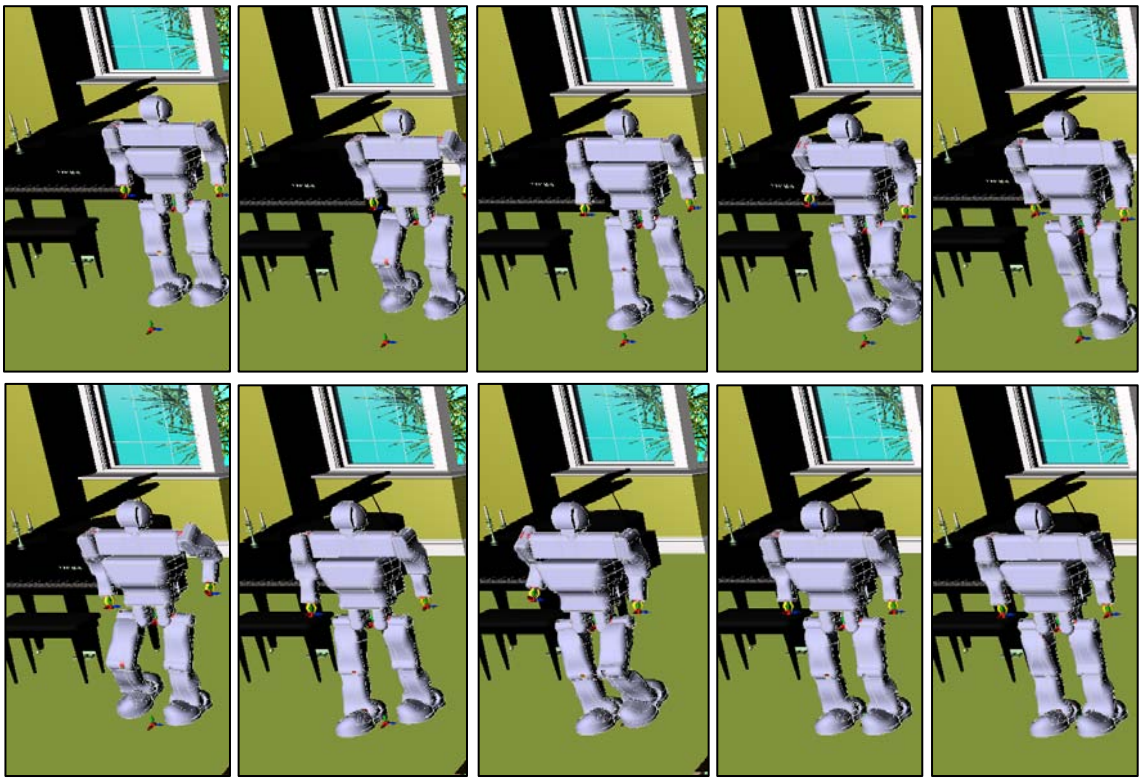


Figura 5-15: Simulación del RH0: Bajar CM, Paso, Zancada, Zancada, Paso, Subir CM.

Todo el movimiento de este ejemplo se desarrolla en menos de 18sg en las seis fases principales que podemos ver en la secuencia de simulación mostrada en la Figura 5-15. Los valores de los **GDL** del humanoide **RH0**, que son los valores de las posiciones de referencia para las articulaciones del robot, son los datos más importantes para el control de la mecánica del **RH0** y son obtenidos como salida del algoritmo **UPA**, así como la información relativa a las posiciones de los pies y del **CM**. Como ejemplo, en la Figura 5-16 podemos ver un extracto de tablas conteniendo los mencionados datos, discretizados según $\mathbf{S_D}$. Para poder analizar los datos numéricos en detalle, remitimos al lector al soporte informático que acompaña esta tesis, que incluye todo el software con las simulaciones y resultados.

t	q1	q2	q3	q4	q5	q6	q12	q11	q10	q9	q8	q7
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,036	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,072	0,000	0,975	-1,951	0,975	0,000	0,000	0,000	0,975	-1,951	0,975	0,000	0,000
0,108	0,000	1,379	-2,759	1,379	0,000	0,000	0,000	1,379	-2,759	1,379	0,000	0,000
0,144	0,000	1,689	-3,379	1,689	0,000	0,000	0,000	1,689	-3,379	1,689	0,000	0,000
0,180	0,000	1,951	-3,902	1,951	0,000	0,000	0,000	1,951	-3,902	1,951	0,000	0,000
0,216	0,000	2,181	-4,362	2,181	0,000	0,000	0,000	2,181	-4,362	2,181	0,000	0,000
0,252	0,000	2,389	-4,779	2,389	0,000	0,000	0,000	2,389	-4,779	2,389	0,000	0,000
0,288	0,000	2,581	-5,162	2,581	0,000	0,000	0,000	2,581	-5,162	2,581	0,000	0,000
0,324	0,000	2,759	-5,518	2,759	0,000	0,000	0,000	2,759	-5,518	2,759	0,000	0,000
0,360	0,000	2,927	-5,853	2,927	0,000	0,000	0,000	2,927	-5,853	2,927	0,000	0,000
0,396	0,000	3,085	-6,170	3,085	0,000	0,000	0,000	3,085	-6,170	3,085	0,000	0,000
0,432	0,000	3,236	-6,471	3,236	0,000	0,000	0,000	3,236	-6,471	3,236	0,000	0,000
0,468	0,000	3,379	-6,759	3,379	0,000	0,000	0,000	3,379	-6,759	3,379	0,000	0,000

t	X_right_foot	Y_right_foot	Z_right_foot	X_left_foot	Y_left_foot	Z_left_foot	X_left_ZMP	Y_left_ZMP	Z_left_ZMP
0,000	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,036	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,072	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,108	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,144	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,180	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,216	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,252	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000

Figura 5-16: Valores para las articulaciones del RH0 y Movimientos de pies y CM.

En la Figura 5-17 se pueden ver las trayectorias de las posiciones (en grados) de las articulaciones de las piernas del **RH0**. Podemos comprobar que todas (i.e. q1 a q12) se encuentran dentro de sus límites mecánicos (ver apéndice B) sin ningún problema.

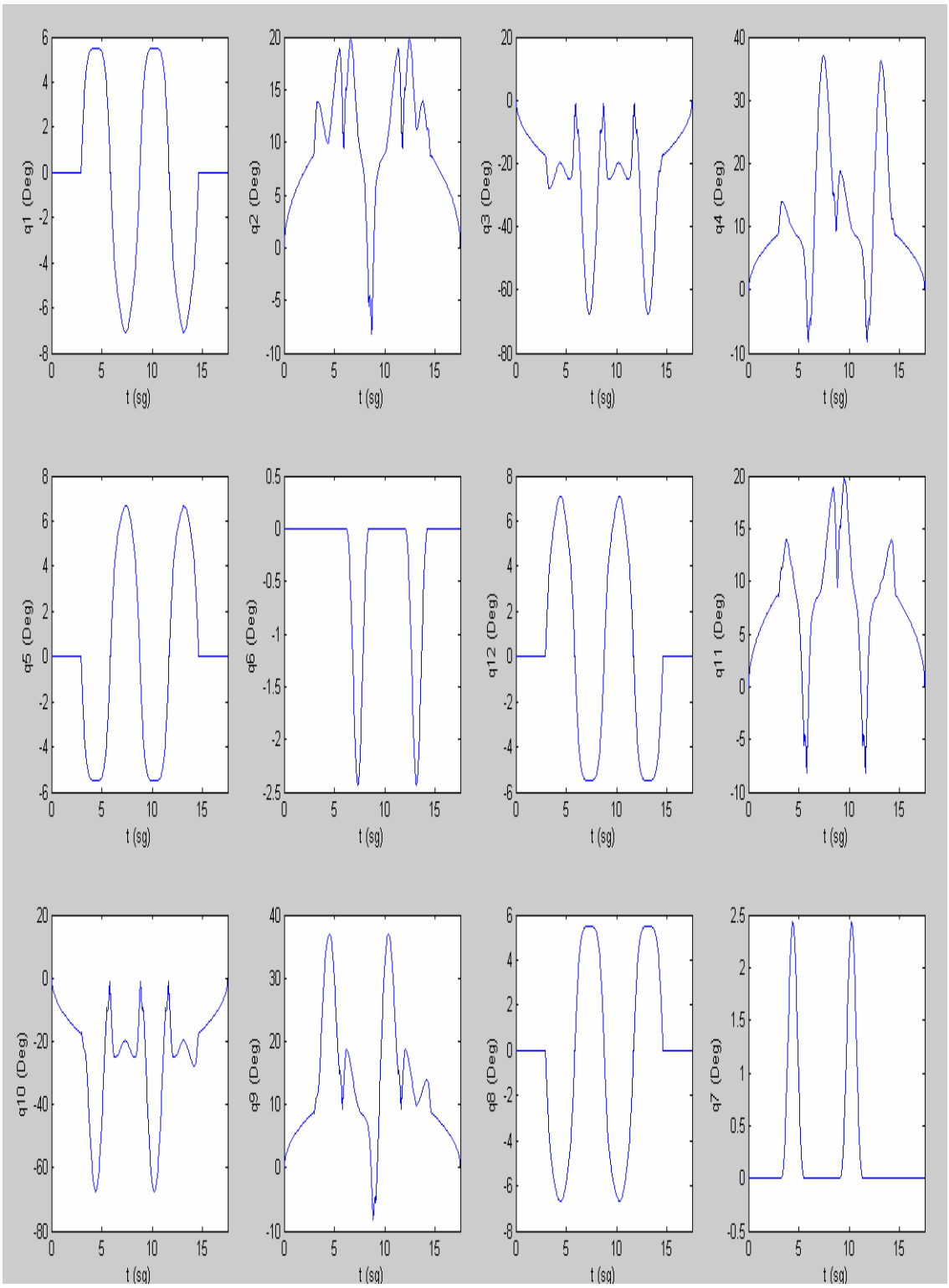


Figura 5-17: RHO cuatro pasos en línea recta - POSICIÓN de las articulaciones.

En la Figura 5-18 se pueden revisar las velocidades de las articulaciones de las piernas del robot **RH0**. Se comprueba que todas ellas (i.e. q1_d a q12_d) se encuentran dentro del rango aceptable (ver apéndice B) ya que aún las más cargadas (las rodillas), no llegan al límite de 25rpm de los motores en ningún momento.

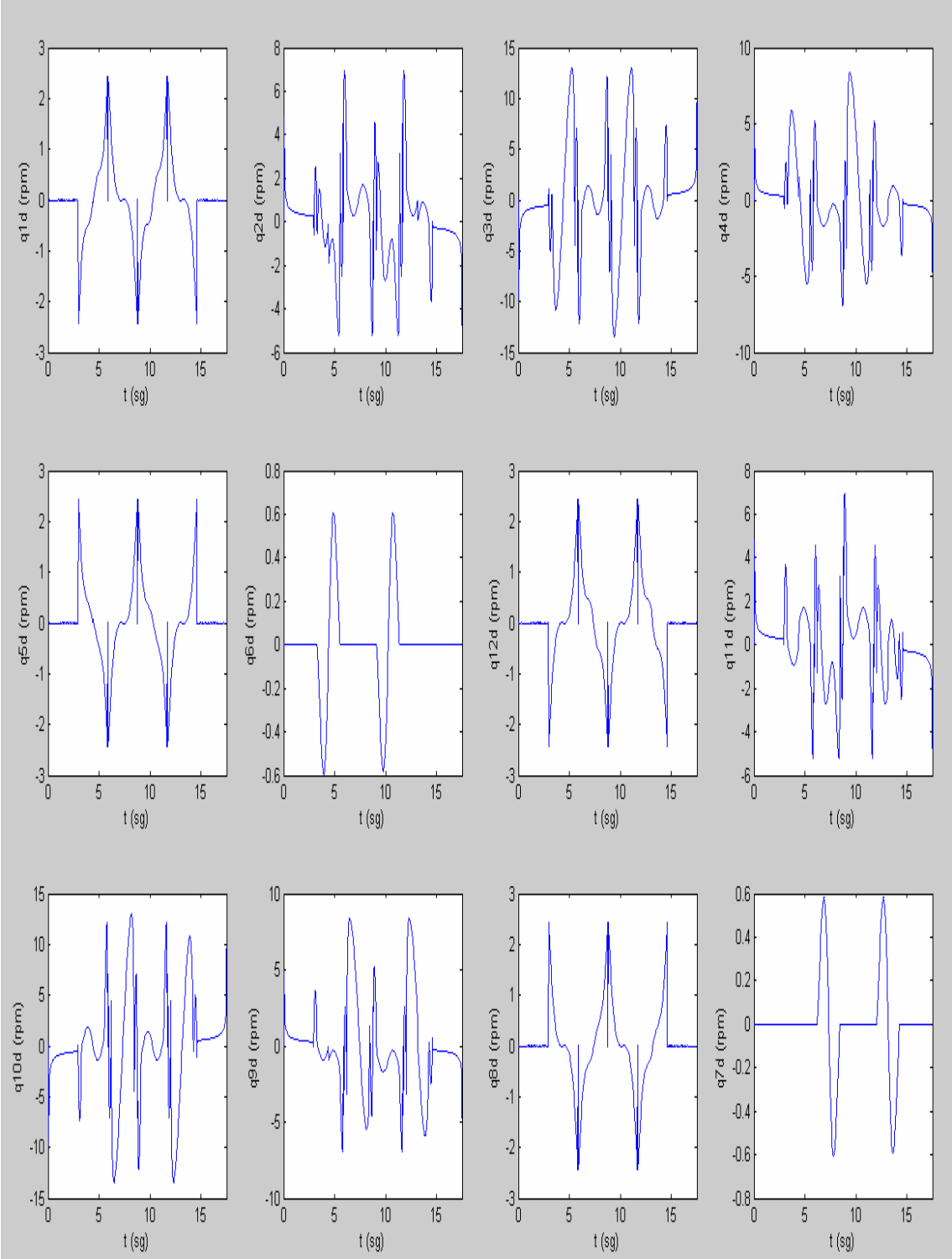


Figura 5-18: RHO cuatro pasos en línea recta - VELOCIDAD de las articulaciones.

5.3.2 El robot RH0 girando 90º sobre su propio eje.

Como se ve en la Figura 5-19, para esta simulación trabajamos con el mismo entorno de Realidad Virtual (ver apéndice B), que incluye al humanoide **RH0** inmerso en un entorno de trabajo constituido por una habitación con varios objetos. El objetivo para el **RH0** es dar pasos en círculo, girando sobre su propio eje, hasta quedar en una posición girada 90º con respecto a la inicial. Para ejecutar el movimiento propuesto el robot necesita utilizar sólo el paso básico de giro del algoritmo **UPA**. El movimiento se realizará ejecutando seis pasos de giro dobles (i.e. pares pie derecho y pie izquierdo), puesto que cada paso doble de giro produce como resultado una rotación respecto a su propio eje de 15º.

Las características de la locomoción del humanoide **RH0** se eligen de entre las posibles (i.e. respetando los límites cinemáticos y dinámicos del robot - ver apéndice B), pero de forma que cada paso se ejecute de una forma natural en menos de 3sg. Para esta simulación las características son:

- **Altura de Paso H_p** = 0,100m.
- **Giro máximo del centro de masas Gcm** = 15º.
- **Altura del centro de masas Hcm** = 0,58m.
- **Coeficiente Cosenoidal para las trayectorias Kcm** = 0,004m.
- **Radio del área de soporte RA_{AS}** = 0,06m.
- **Potencial para generación de trayectorias Pocm** = 4.
- **Cadencia Ca** = 0,3429pasos/sg.
- **Discretización de los datos (necesaria por Hardware y Drivers) S_D** = 0,036sg.

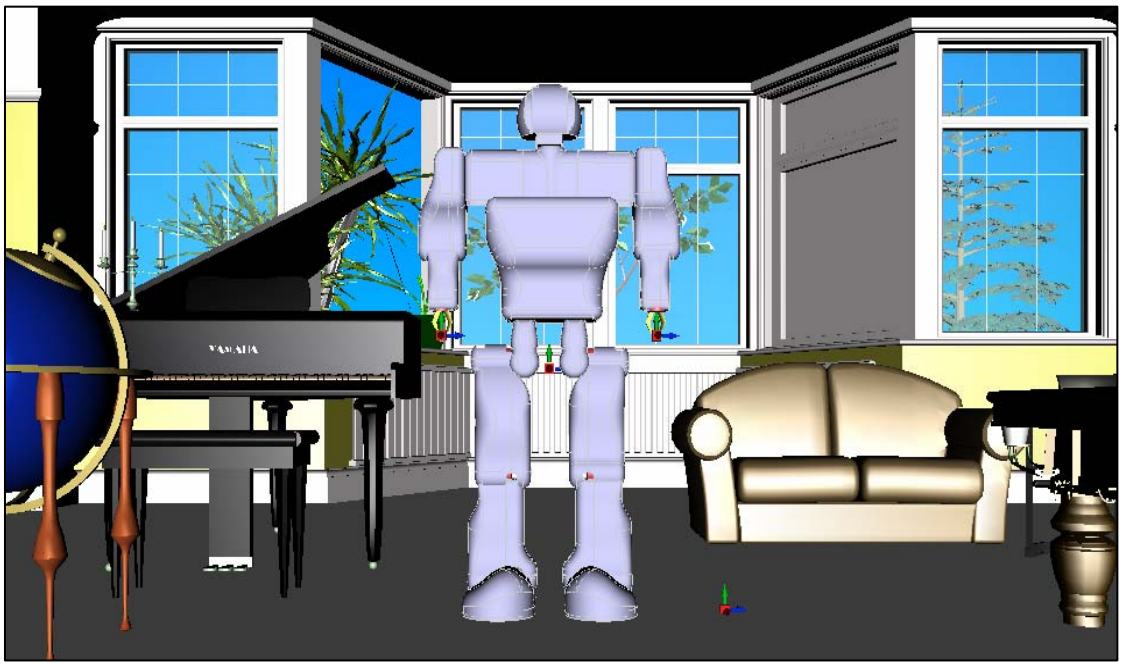


Figura 5-19: El RH0 dentro de un entorno local de Realidad Virtual.

Dados los valores de **Ca** y $\mathbf{S_D}$, **cada paso del humanoide RH0 empleará en desarrollarse 2,916sg**, y cada ejecución generará una **tabla con 81 valores para cada GDL** del robot. En 3.3.2 se explicó como obtener las cinco fases del **UPA** para un paso. Aquí simulamos doce ejecuciones consecutivas del **UPA** que se convierten en seis pasos dobles de giro para el **RH0**. De este modo, los caminos contienen 60 puntos principales:

- $\mathbf{Q_{rCM}}$ **- Camino Factible del CM**: Trayectoria compuesta por configuraciones $\mathbf{q_{CMi}}$ del centro de masas, según "(5-4)".

$$Q_{rCM} = \{q_{CM1} \ldots q_{CM2} \ldots q_{CM3} \ldots q_{CM4} \ldots q_{CM5} \ldots q_{CM56} \ldots q_{CM57} \ldots q_{CM58} \ldots q_{CM59} \ldots q_{CM60}\} \quad \textbf{(5-4)}$$

- $\mathbf{Q_{rZ}}$ **- Camino Factible del pie izquierdo**: Trayectoria compuesta por configuraciones $\mathbf{q_{Zi}}$ del pie izquierdo, según "(5-5)". Viene dada por el camino $\mathbf{Q_{rPM}}$ resultado del **UPA**, cuando es el pie izquierdo el móvil.

$$Q_{rZ} = \{q_{Z1} \ldots q_{Z2} \ldots q_{Z3} \ldots q_{Z4} \ldots q_{Z5} \ldots q_{Z56} \ldots q_{Z57} \ldots q_{Z58} \ldots q_{Z59} \ldots q_{Z60}\} \quad \textbf{(5-5)}$$

- $\mathbf{Q_{rD}}$ **- Camino Factible del pie derecho**: Trayectoria compuesta por configuraciones $\mathbf{q_{Di}}$ del pie derecho, según "(5-3)". Viene dada por el camino $\mathbf{Q_{rPM}}$ resultado del **UPA**, cuando el pie móvil es el derecho.

$$Q_{rD} = \{q_{D1} \ldots q_{D2} \ldots q_{D3} \ldots q_{D4} \ldots q_{D5} \ldots q_{D56} \ldots q_{D57} \ldots q_{D58} \ldots q_{D59} \ldots q_{D60}\} \quad \textbf{(5-6)}$$

Las trayectorias completas de los caminos factibles (i.e., $\mathbf{Q_{rCM}}$, $\mathbf{Q_{rZ}}$ y $\mathbf{Q_{rD}}$) se obtienen por interpolación según se detalló en 3.5.2, quedando según vemos en la Figura 5-20.

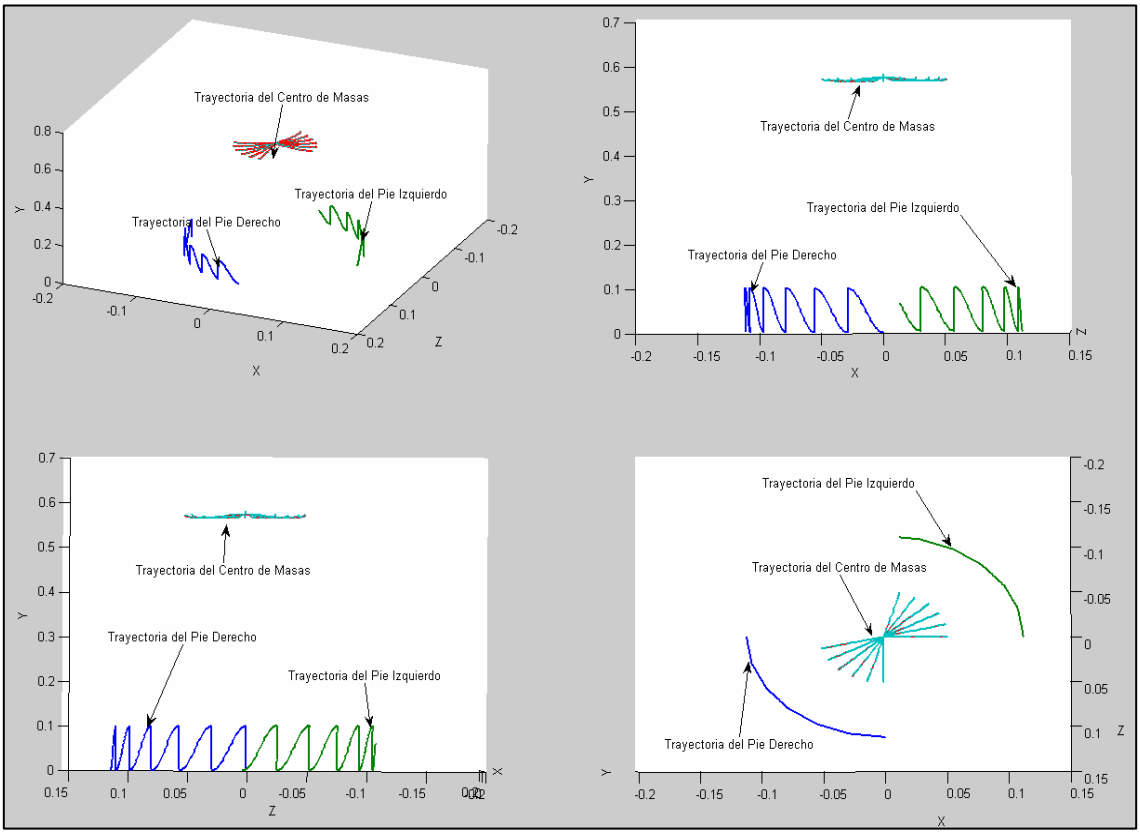


Figura 5-20: Trayectorias para los pies y el CM del RH0 girando sobre su propio eje.

Figura 5-21: Secuencia de Simulación del RH0: Seis dobles pasos de Giro consecutivos.

Por lo tanto, el movimiento de este ejemplo se desarrolla en menos de 36sg en doce fases, de cuya secuencia de simulación es mostrado un extracto en la Figura 5-21.

Los valores de los **GDL** del humanoide **RH0**, que son los valores de las posiciones de referencia para las articulaciones del robot, son los datos más importantes para el control de la mecánica del **RH0**, y son obtenidos como salida del algoritmo **UPA**, así como la información relativa a las posiciones de los pies y del **CM**. Como ejemplo, en la Figura 5-22 podemos ver un extracto de tablas conteniendo todos estos datos. Para poder analizar todos los datos con detalle, hay que remitirse al soporte informático que acompaña esta tesis, que incluye todo el software con las simulaciones y resultados.

t	q1	q2	q3	q4	q5	q6	q12	q11	q10	q9	q8	q7
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,036	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,072	0,000	0,975	-1,951	0,975	0,000	0,000	0,000	0,975	-1,951	0,975	0,000	0,000
0,108	0,000	1,379	-2,759	1,379	0,000	0,000	0,000	1,379	-2,759	1,379	0,000	0,000
0,144	0,000	1,689	-3,379	1,689	0,000	0,000	0,000	1,689	-3,379	1,689	0,000	0,000
0,180	0,000	1,951	-3,902	1,951	0,000	0,000	0,000	1,951	-3,902	1,951	0,000	0,000
0,216	0,000	2,181	-4,362	2,181	0,000	0,000	0,000	2,181	-4,362	2,181	0,000	0,000
0,252	0,000	2,389	-4,779	2,389	0,000	0,000	0,000	2,389	-4,779	2,389	0,000	0,000
0,288	0,000	2,581	-5,162	2,581	0,000	0,000	0,000	2,581	-5,162	2,581	0,000	0,000
0,324	0,000	2,759	-5,518	2,759	0,000	0,000	0,000	2,759	-5,518	2,759	0,000	0,000
0,360	0,000	2,927	-5,853	2,927	0,000	0,000	0,000	2,927	-5,853	2,927	0,000	0,000
0,396	0,000	3,085	-6,170	3,085	0,000	0,000	0,000	3,085	-6,170	3,085	0,000	0,000
0,432	0,000	3,236	-6,471	3,236	0,000	0,000	0,000	3,236	-6,471	3,236	0,000	0,000
0,468	0,000	3,379	-6,759	3,379	0,000	0,000	0,000	3,379	-6,759	3,379	0,000	0,000

t	X_right_foot	Y_right_foot	Z_right_foot	X_left_foot	Y_left_foot	Z_left_foot	X_left_ZMP	Y_left_ZMP	Z_left_ZMP
0,000	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,036	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,072	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,108	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,144	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,180	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,216	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000
0,252	-0,112	0,000	0,000	0,112	0,000	0,000	0,000	0,580	0,000

Figura 5-22: Valores para las articulaciones del RH0 y Movimientos de pies y CM.

En la Figura 5-23 se pueden ver las trayectorias de las posiciones de las articulaciones de las piernas del **RH0** (en grados). Podemos comprobar que todas (i.e. q1 a q12) se encuentran dentro de sus límites mecánicos (ver apéndice B).

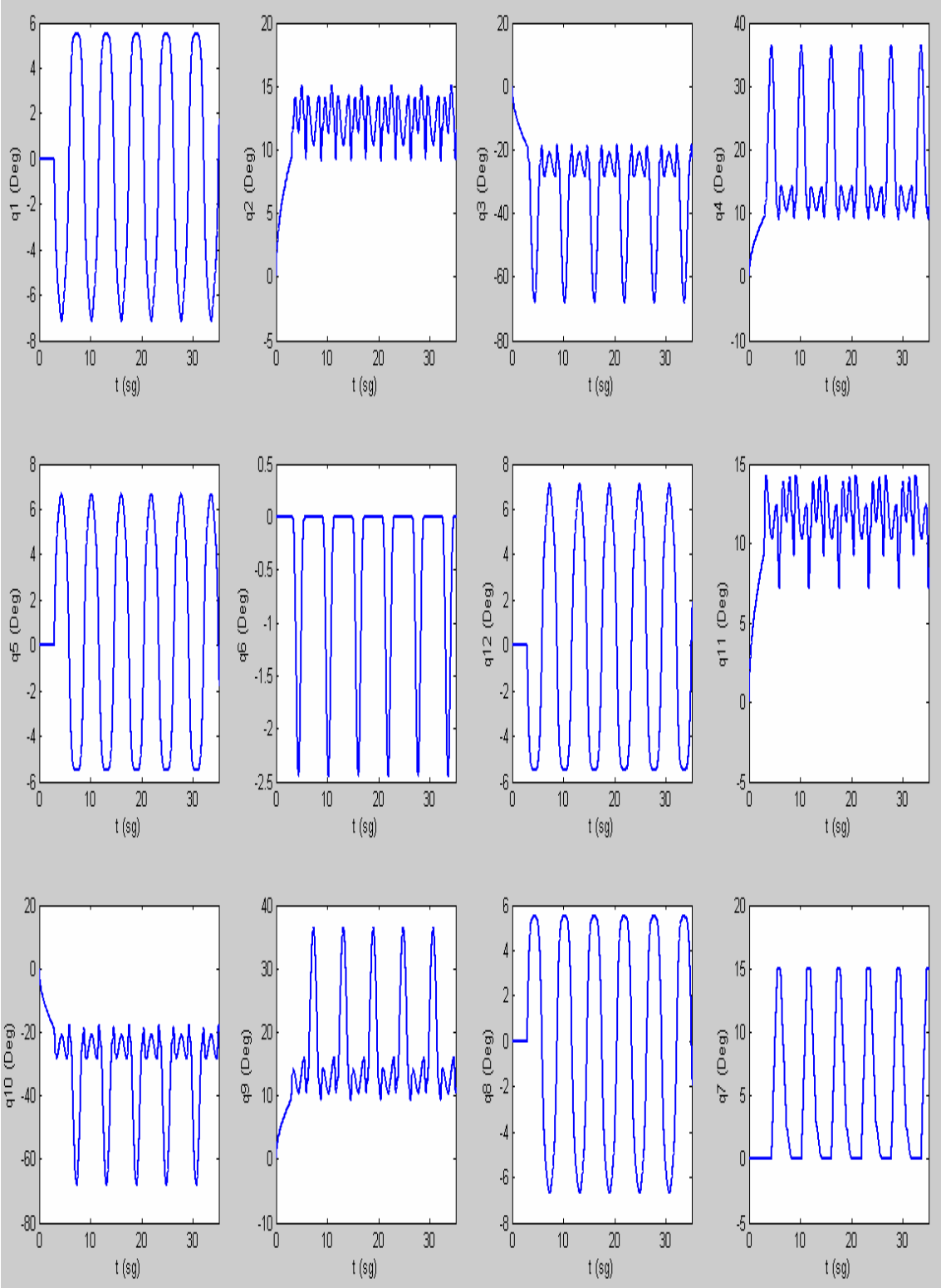


Figura 5-23: RHO girando sobre su propio eje - POSICIÓN de las articulaciones.

En la Figura 5-24 se pueden revisar las velocidades de las articulaciones de las piernas del robot **RH0**. Todas ellas (i.e. q1_d a q12_d) se encuentran dentro de rango admisible (ver apéndice B), ya que aún las más cargadas, que son las rodillas, no llegan al límite de 25rpm de los motores en ningún momento.

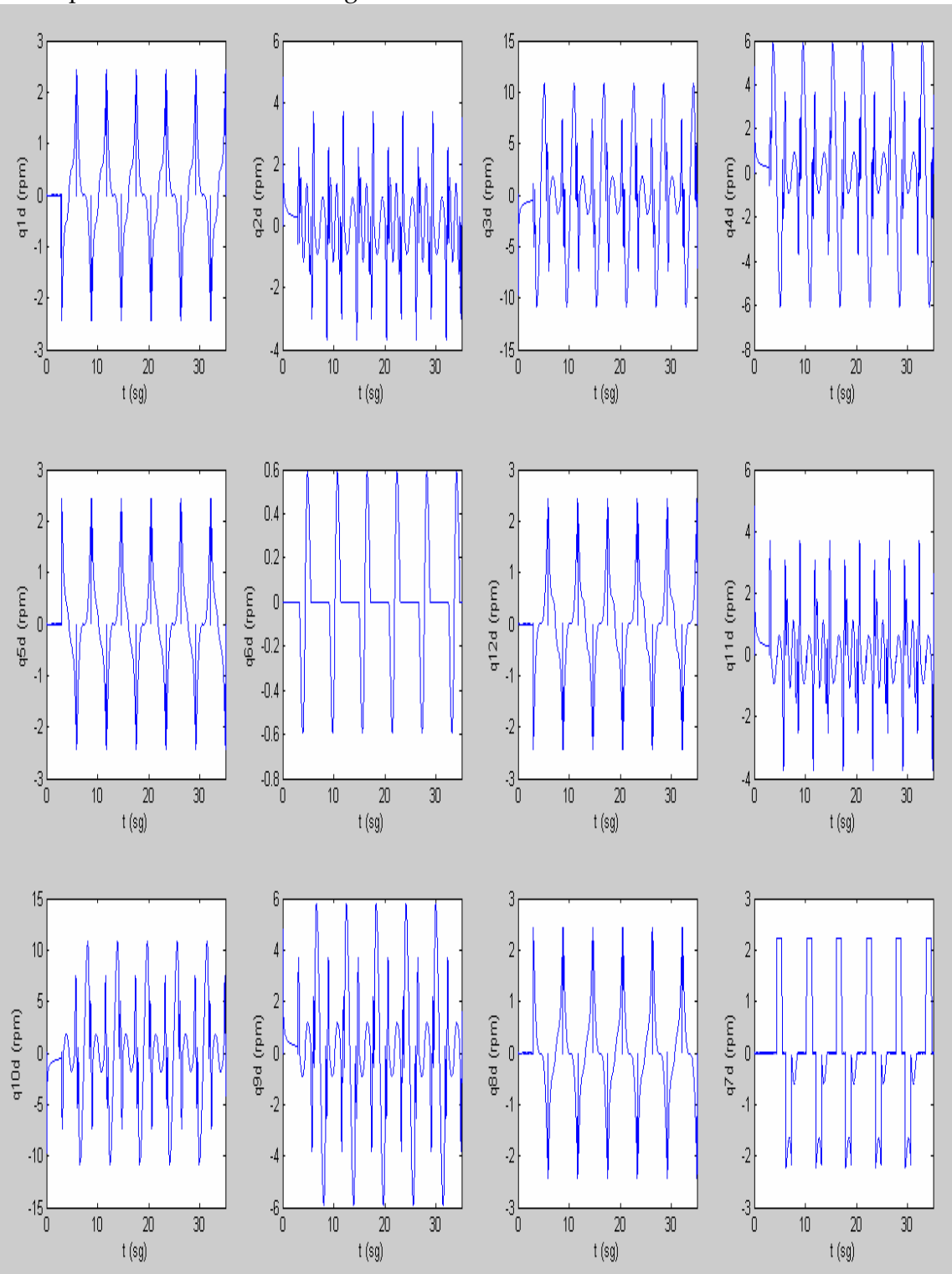


Figura 5-24: RHO girando sobre su propio eje - VELOCIDAD de las articulaciones.

5.4 Navegación y Planificación de Trayectorias del Humanoide RH0.

La navegación y planificación de trayectorias para el **RH0** son complejas debido a sus 21 **GDL**, sus restricciones mecánicas y las restricciones del modelo del entorno local (que en este caso es una habitación con una serie de objetos: piano, sofá, mesa de billar, silla, sillón, lámpara y bola planetaria). Todo este entorno de trabajo inicial lo tenemos representado en nuestra plataforma virtual tal y como se ve en la Figura 5-25 en su vista de planta. No es necesario que el entorno de trabajo se conozca con absoluta precisión, puesto que la potencia del algoritmo **M3R** permite trabajar con objetos de todo tipo (incluso imperfectamente definidos) pero obviamente necesitamos algún tipo de mapa sobre el que lanzar el algoritmo de Navegación global.

Para resolver el problema y obtener **Las Trayectorias de Navegación Global y Local del Humanoide RH0** aplicaremos el nuevo algoritmo **TCG** (ver 4.4), obteniendo la trayectoria cinemática libre de colisiones del **ZMP** correspondiente al humanoide, por aplicación directa del algoritmo **M3R** (ver 4.3), dentro del espacio **SE(3)**, sujeto a las restricciones dadas por el mapa disponible del espacio de trabajo. El resultado del algoritmo **M3R** es la trayectoria $\mathbf{Q}_r$ de referencia del algoritmo **UPA** (ver 3.3, 3.5 y 5.3), que resuelve el movimiento de locomoción bípeda del humanoide **RH0**.

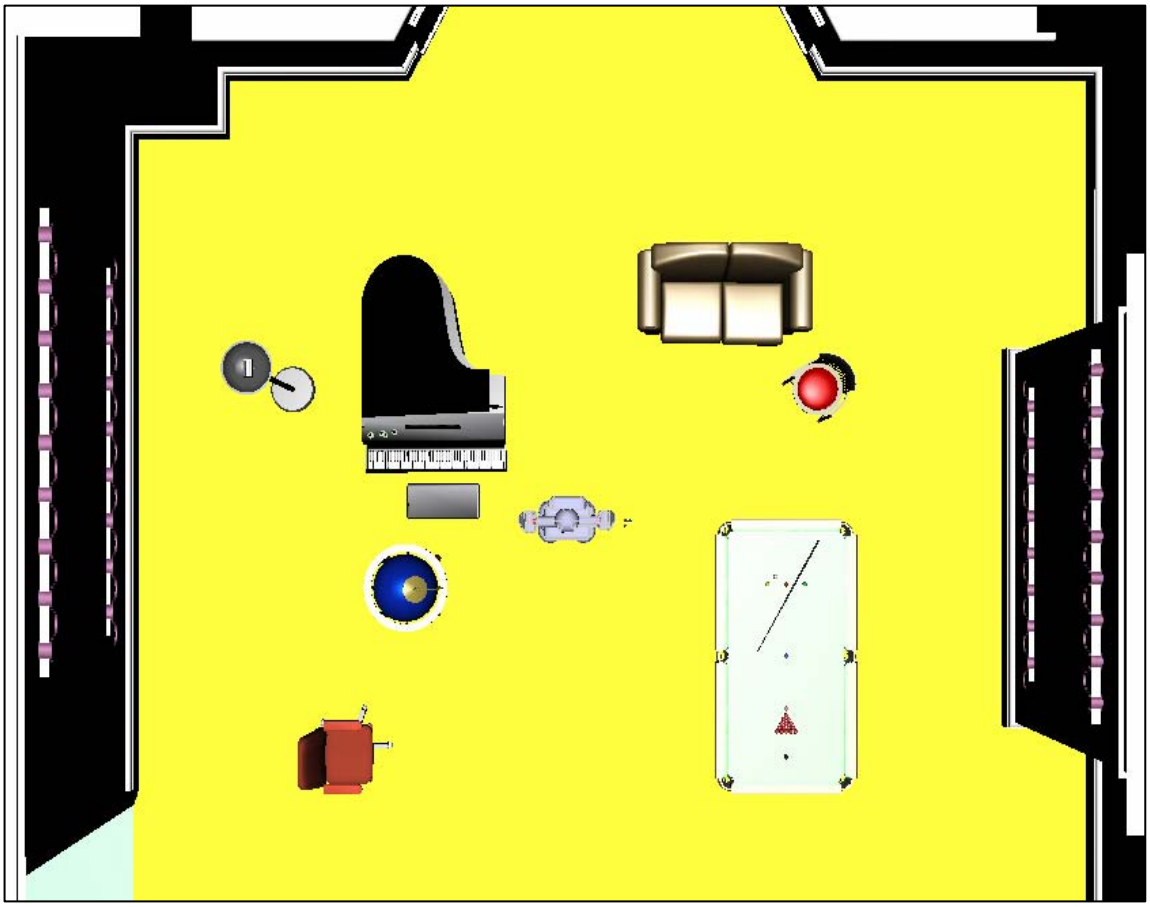


Figura 5-25: Vista en planta del estado inicial del entorno del RH0.

5.4.1 TGG con el Nuevo algoritmo M3R en un entorno sencillo.

La navegación y control del movimiento para el **RH0** son complejos debido a sus 21 **GDL** y las restricciones tanto mecánicas como de equilibrio. Al aplicar al humanoide **RH0** el nuevo **TCG** (ver 4.4), obtendremos:

- **La Trayectoria de Navegación Global del RH0**: Trayectoria cinemática libre de colisiones del **ZMP** correspondiente al humanoide, obtenida por aplicación directa del algoritmo **M3R** (ver 4.3), dentro del espacio **SE(3)**, sujeto a las restricciones dadas por el mapa disponible del espacio de trabajo.
- **La Planificación de Movimientos del RH0**: Teniendo como entrada la trayectoria global libre de colisiones, genera trayectorias para las articulaciones del robot por aplicación del nuevo algoritmo **UPA** (ver 3.3 y 3.5).

El nuevo algoritmo **M3R** (ver 4.3) calcula una trayectoria libre de colisiones para el **CM** del humanoide, sujeto a las restricciones dadas por el mapa del entorno disponible y los obstáculos en **SE(3)**. Ahora lo aplicaremos a la navegación global del **RH0**.

Como consecuencia de las propiedades teóricas del algoritmo **M3R**, la planificación global puede realizarse para entornos con cualquier tipo de obstáculos (incluso cóncavos o irregulares), con una gran eficacia computacional. Como ejemplo podemos ver en la Figura 5-26 el camino global $\mathbf{Q_r}$ cuasi-óptimo, obtenido para el humanoide **RH0** en un entorno de trabajo sencillo con varios obstáculos geométricos.

El resultado obtenido con el **M3R** no es otro que la salida del **TCG**, esto es, el camino factible de navegación global del humanoide $\mathbf{Q_r}$, que es la información de referencia del algoritmo **UPA** (ver 3.3 y 3.5), para poder resolver la planificación local y movimiento de locomoción bípeda del humanoide.

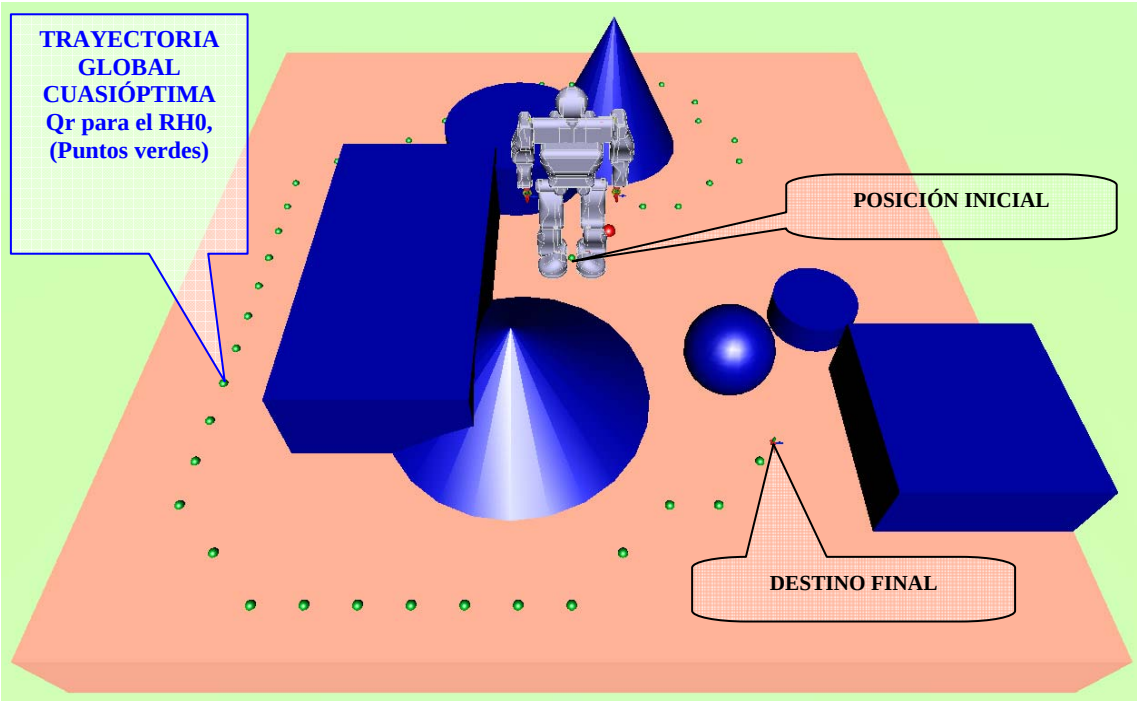


Figura 5-26: Algoritmo M3R aplicado a un entorno del RH0.

5.4.2 Navegación Global del RH0 en un entorno complejo.

Para apreciar como trabaja el algoritmo **TCG** comenzamos con el entorno de trabajo visto en la Figura 5-25, y aplicamos movimiento a los objetos que están en la habitación, de forma que quedan como se puede apreciar en la Figura 5-27, rodeando casi por completo al humanoide **RH0**. Hay que señalar que la configuración de obstáculos formada, provoca que el problema de navegación sea extremadamente difícil, puesto que para cualquier objetivo que se encuentre por fuera de la barrera de obstáculos, muchos algoritmos de planificación fracasarían por la existencia de un mínimo local fortísimo que empuja al humanoide a quedarse dentro del encierro. Vamos a ver en esta simulación, con que eficacia el **TCG** resuelve el problema.

El núcleo del **TCG** es el nuevo algoritmo **M3R** (ver 4.3) que calcula una trayectoria libre de colisiones para el **CM** del humanoide, sujeto a las restricciones dadas por el mapa del entorno disponible en **SE(3)**, que obviamente para una planificación global debe ser conocido, pero no necesariamente perfecto, puesto que como consecuencia de las propiedades del algoritmo **M3R**, la planificación global puede realizarse para entornos con cualquier tipo de obstáculos (incluso cóncavos, irregulares, interconectados o imprecisos), con una gran eficacia computacional. Como ejemplo podemos ver en la Figura 5-27 el camino global $\mathbf{Q_r}$ cuasi-óptimo, obtenido para el humanoide **RH0** en el entorno de trabajo que preparamos anteriormente.

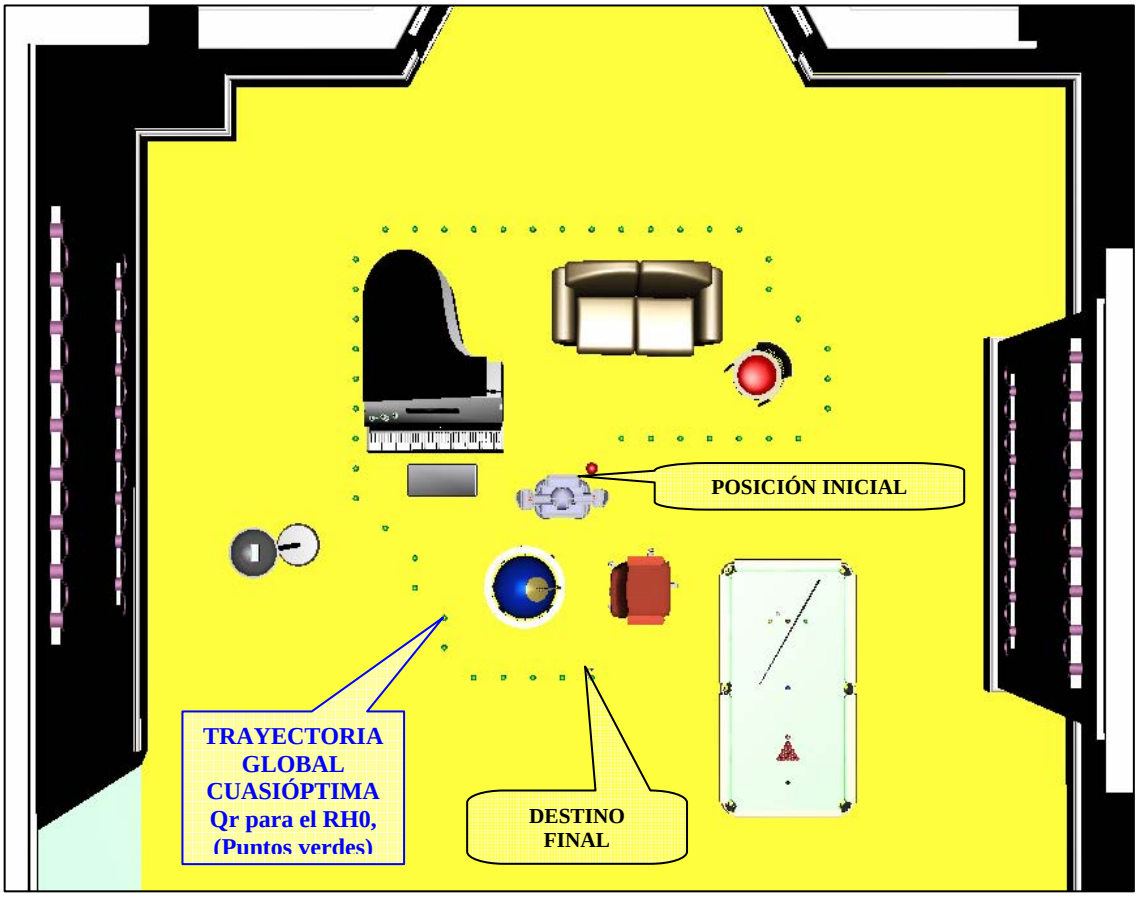


Figura 5-27: Vista en planta con obstáculos para el RH0 y aplicación del M3R.

Algoritmo	M3R	FMM	Caminos Probabilisticos	Redes Neuronales	Algoritmos Geneticos	Campos de Potencial	Grafos de Visibilidad	Diagramas de Voronoi
Solucion de la Trayectoria de NAVEGACION GLOBAL	t < 0.04sg	t < 0.4sg	t < 0.8sg	t < 4sg	t < 4sg	No hay Solucion por Minimos Locales	No hay Solucion por Tipo de Obstaculos	No hay Solucion por Tipo de Obstaculos

Los Grafos de Visibilidad necesitan que los obstáculos sean poligonales y no irregulares como en este ejemplo

Los Diagramas de Voronoi tradicionales no pueden capturar la conectividad del espacio de configuraciones si los obstáculos son irregulares

Figura 5-28: Tiempos de ejecución del M3R para Navegación.

En cuanto a la eficacia del nuevo algoritmo **M3R** (ver 4.3), comparamos los tiempos de ejecución con otros algoritmos de planificación de trayectorias (ver **Error! Reference source not found.**). Destacamos que el **M3R** es unas diez veces más rápido que el algoritmo **FMM**, veinte veces más rápido que uno de búsqueda probabilística, e incomparablemente más rápido que algoritmos basados en inteligencia artificial (y esto cuando estos últimos resuelven el problema, puesto que no tienen garantizada la convergencia). El algoritmo **M3R** es más rápido cuanto mayor es el número y tamaño de los obstáculos en el entorno de trabajo, no tiene problemas de mínimos locales y sirve para cualquier tipo de obstáculos, encontrando SIEMPRE una solución geométrica de $\mathbf{Q_r}$ como trayectoria global de navegación para el humanoide **RH0**.

5.4.3 Navegación Local del RH0 en un entorno complejo.

Debido a la imperfección de los mapas globales del entorno de trabajo y también porque éste es dinámico, la operación real con el humanoide **RH0** requiere un análisis por integración sensorial del entorno que rodea al robot, antes de ejecutar cada paso de locomoción bípeda mediante el algoritmo **UPA**. El algoritmo utilizado para esta planificación local es el mismo **M3R** (ver 4.3), que resulta aún más eficaz, dado que el espacio de trabajo que tiene que analizar es más reducido, de forma que el tiempo de ejecución para el ejemplo de la Figura 5-29 es de tan solo 4ms.

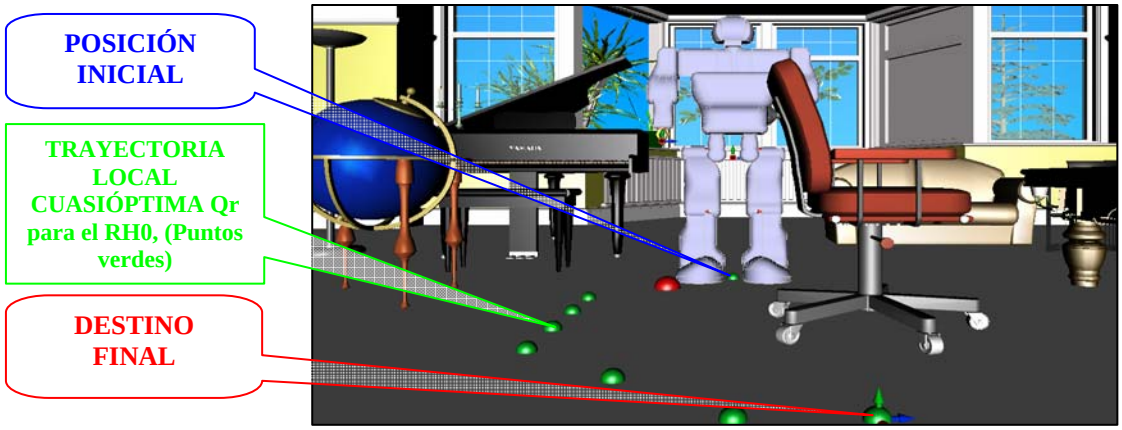


Figura 5-29: Algoritmo M3R aplicado para la Navegación Local del RH0.

5.5 Tarea Completa – Locomoción más Navegación complejas del RH0.

En este apartado vamos a simular como el humanoide **RH0** realiza una tarea compleja de forma completa. El robot se encuentra inmerso en el entorno de trabajo complejo que presentamos en la Figura 5-30, que contiene unos ciertos obstáculos (i.e. muebles). La tarea consiste en definir un objetivo dentro del entorno de trabajo, de forma que el robot camine desde su posición inicial hasta ese objetivo definido, por supuesto evitando colisionar con los obstáculos, respetando las restricciones mecánicas del **RH0** y manteniendo el equilibrio de locomoción bípeda. Para conseguirlo, la simulación tiene que desarrollar todos los algoritmos propuestos en esta tesis, puesto que debe resolver los siguientes problemas:

- **Navegación Global y Local**: Mediante el nuevo algoritmo **TCG** (ver 5.4.2), obtenemos una trayectoria libre de colisiones factible $\mathbf{Q_r}$ para el cuerpo del humanoide, basado en el nuevo algoritmo **M3R**. La navegación se puede reforzar con integración sensorial local que permite una replanificación de trayectorias, también con el nuevo algoritmo **M3R**, como hemos visto en 5.4.3.
- **Locomoción Bípeda**: Con la utilización del nuevo algoritmo **UPA** (ver 3.5), generamos los caminos de las configuraciones para el cuerpo (i.e., $\mathbf{Q_{rCM}}$) y los pies (i.e.,$\mathbf{Q_{rZ}}$ y $\mathbf{Q_{rD}}$) del humanoide, que construyen una locomoción bípeda. Es obvio que para esta simulación el humanoide **RH0** debe combinar la locomoción bípeda en línea recta con la locomoción bípeda de giro.
- **Control cinemático**: Resolvemos de una forma geométrica el problema cinemático inverso completo del nuevo modelo cinemático **DCS** del **RH0** (ver 3.4) para cada punto de $\mathbf{Q_{rCM}}$.

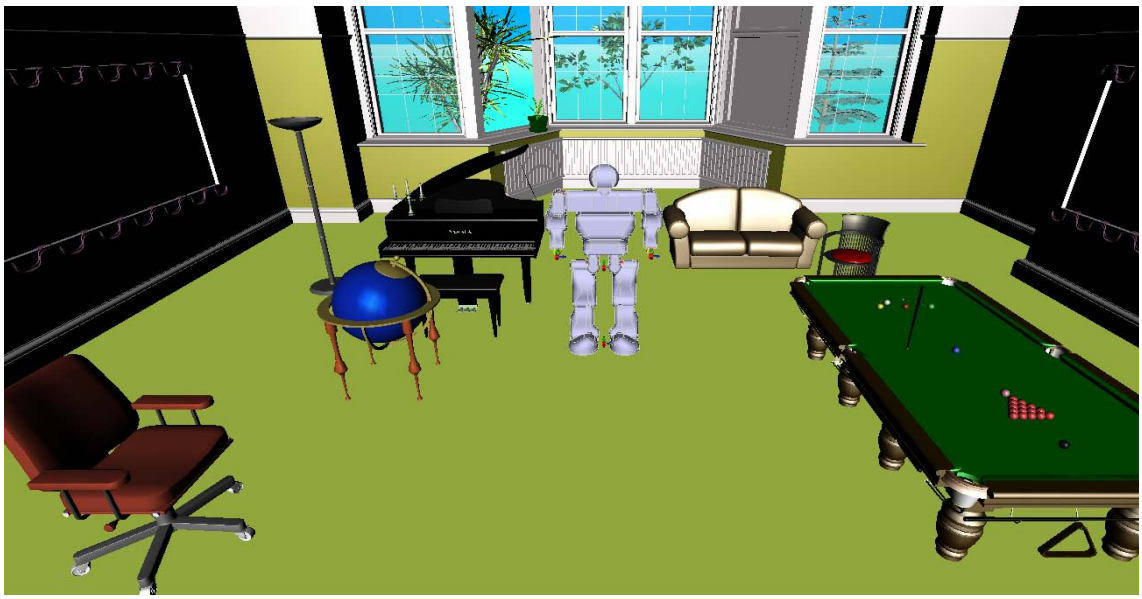


Figura 5-30: Entorno de Trabajo Complejo para el humanoide RH0.

En la Figura 5-31 podemos ver como tras mover los objetos, el nuevo algoritmo de navegación **TCG** calcula la trayectoria global cuasi-óptima. **El tiempo de cálculo es menor a los 40ms** para esta configuración del entorno de trabajo.

Las características de la locomoción del humanoide **RH0** se eligen de entre las posibles (i.e. respetando los límites mecánicos del robot - ver apéndice B), y son:

- **Longitud de Paso L_p** = 0,160m
- **Altura de Paso H_p** = 0,100m.
- **Anchura de Paso W_p** = 0,112m.
- **Giro máximo del centro de masas Gcm** = 15º.
- **Altura del centro de masas Hcm** = 0,58m.
- **Coeficiente Cosenoidal para las trayectorias Kcm** = 0,004m.
- **Radio del área de soporte RA_{AS}** = 0,06m.
- **Potencial para generación de trayectorias Pocm** = 4.
- **Cadencia Ca** = 0,3429pasos/sg.
- **Discretización de los datos (necesaria por Hardware y Drivers) S_D** = 0,036sg.

Dados los valores de **Ca** y **S_D**, **cada paso del humanoide RH0 empleará en desarrollarse 2,916sg**, y cada ejecución del algoritmo **UPA** que resuelve ese paso generará una **tabla con 81 valores para cada GDL** del robot. En la Figura 5-32 podemos apreciar un extracto de la secuencia de simulación compleja de locomoción bípeda.

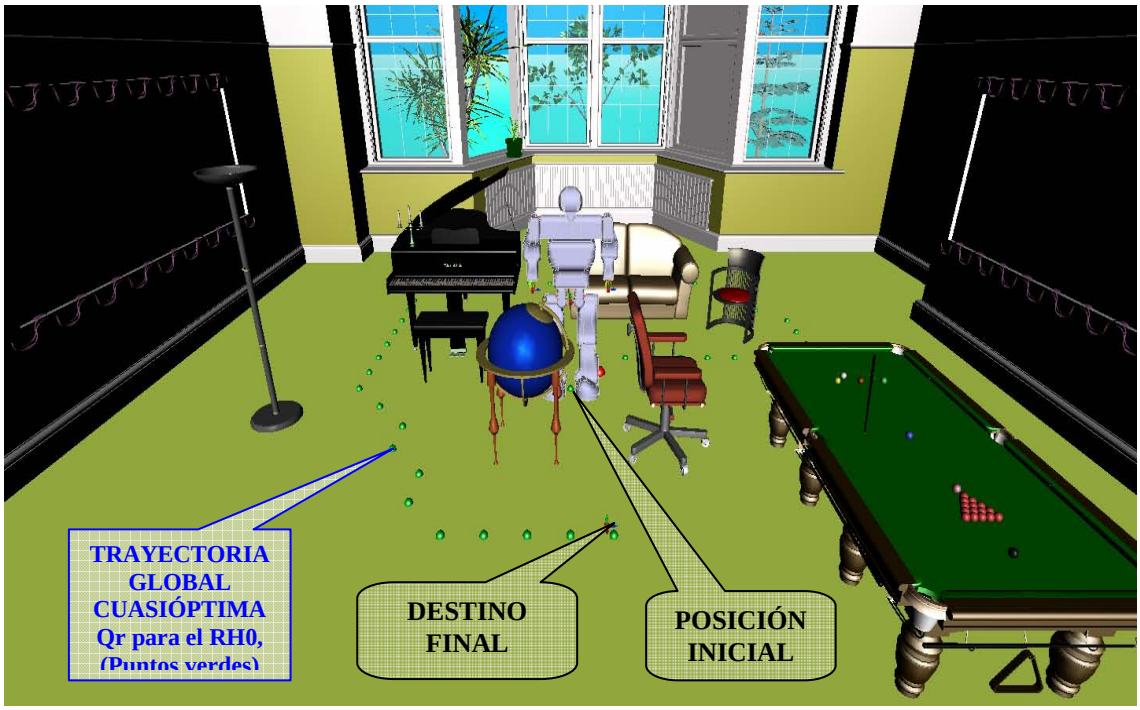


Figura 5-31: Trayectoria de Navegación Global en un entorno complejo.

Figura 5-32: Simulación de Locomoción Compleja del RH0.

Los valores de los **GDL** del humanoide **RH0** son los datos más importantes para el control y son obtenidos como salida del algoritmo **UPA**. La información relativa a las posiciones de los pies y del **CM** es obtenida como resultado de la simulación. Para poder analizar todos estos datos con detalle, hay que remitirse al soporte informático que acompaña esta tesis, que incluye todo el software de las simulaciones y resultados, que demuestran que no se sobrepasan en ningún momento ni las restricciones mecánicas del robot ni de los motores de las articulaciones. Podemos ver una imagen de las trayectorias en la Figura 5-33, con detalles de la forma que toman los caminos para el cuerpo $\mathbf{Q_{rCM}}$, el pie izquierdo $\mathbf{Q_{rZ}}$ y el pie derecho $\mathbf{Q_{rD}}$ del humanoide **RH0**.

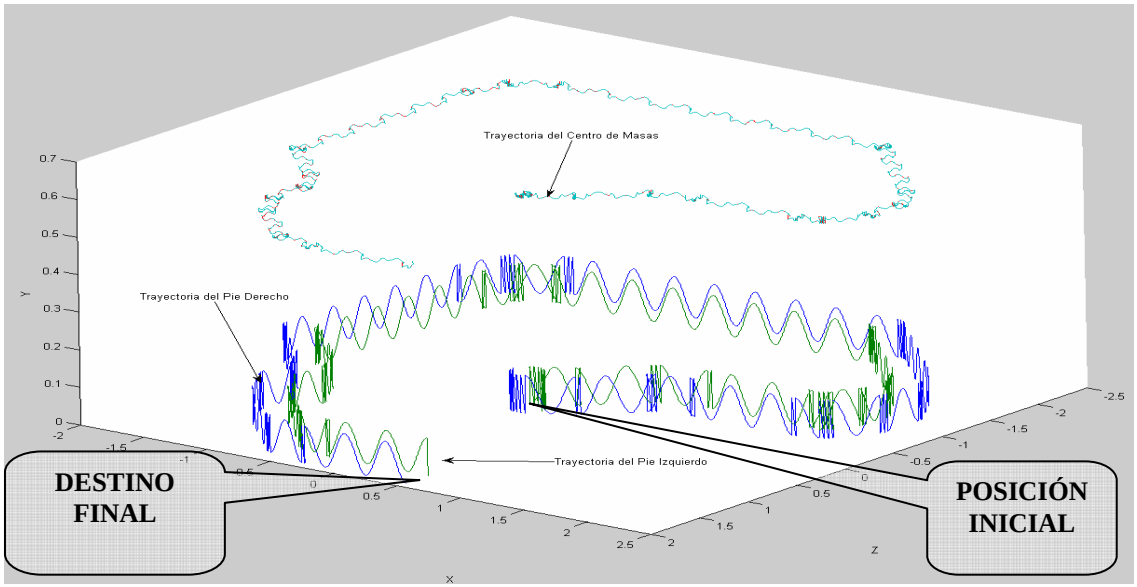


Figura 5-33: Trayectorias para CM y pies del RH0 en una Locomoción Bípeda compleja.

5.6 Análisis de los resultados obtenidos con el simulador RobManSim.

Como un objetivo no fundamental, pero si necesario e importante de esta tesis, se ha construido la Plataforma de Realidad Virtual **RobManSim**, que es un simulador que reproduce el comportamiento mecánico del robot humanoide **RH0** en su entorno real de trabajo. El simulador permite desarrollar los nuevos algoritmos de esta tesis de forma segura. El simulador **RobManSim** presenta como resultados más destacables:

- La función de "Modelado" (ver 5.2.1) utiliza para la generación de modelos de realidad virtual **ISO X3D** [37] y **VRML**, que son estándares abiertos bastante avanzados que se encuentran en evolución. Una indicación de la calidad de esta función puede ser el hecho de que **OpenHRP** también utiliza **VRML**.
- El simulador cuenta con un **Entorno de Simulación Integrado** (ver 5.2.2) que tiene la ventaja de estar realizado íntegramente con código abierto y gratuito, frente a otras alternativas (e.g., **OpenHRP**) con código propietario y partes de la arquitectura encerradas en cajas negras. Las funciones que engloba son:
    - "Control", implementado con **MATLAB** y **SIMULINK**. Frente a esto, **OpenHRP** utiliza **CORBA**, que es independiente de los sistemas operativos y lenguajes de programación, lo que sin duda es una ventaja. Además **OpenHRP** integra en forma de objetos: el control, el modelado dinámico de los robots y la función de detección de colisiones.
    - "DispositivosEntrada" para los dispositivos externos (e.g., en esta tesis se utiliza un ratón Magellan de 6 **GDL**).
    - "Planificación" para obtener trayectorias de navegación libres de colisiones, implementada con el nuevo modelo **TCG**.
    - "PatrónPaso" para diseñar el paso de locomoción bípeda en equilibrio del robot humanoide, que se genera con el nuevo algoritmo **UPA**.
- El simulador cuenta con un eficiente **Interfaz Gráfico de Usuario**, montado con **MATLAB** y **SIMULINK** (ver 5.2.3). El interfaz de **OpenHRP** está mucho más desarrollado en cuanto a funcionalidades y es de plataforma independiente.
- Es posible simular comportamientos de humanoides de forma precisa, como hemos visto para varios ejemplos: caminando cuatro pasos en línea recta (ver 5.3.1), girando 90º sobre su propio eje (ver 5.3.2), calculando una trayectoria global (ver 5.4.2), calculando una trayectoria local (ver 5.4.3), o **realizando un movimiento complejo de locomoción y navegación bípedas** (ver 5.5).
- Se ha probado la consistencia entre las simulaciones obtenidas para el robot y varios experimentos reales con el humanoide **RH0**: como podremos comprobar para los casos del robot dando cuatro pasos en línea recta (ver 6.2.1), el robot girando sobre su propio eje (ver 6.2.2), o el robot siguiendo una trayectoria de navegación libre de colisiones (ver 6.3.2).

6 EXPERIMENTACIÓN con el Robot Humanoide RH0.

En este capítulo, verificaremos los nuevos modelos teóricos y algoritmos que se introducen en esta tesis, mediante la experimentación con el robot humanoide ***RH0*** *construido en la* ***Universidad Carlos III de Madrid****. Los experimentos nos permiten estudiar los comportamientos reales del humanoide* ***RH0****, de forma que podemos corroborar que las nuevas ideas realmente funcionan. Los algoritmos experimentados son los más importantes de la tesis: Nuevo algoritmo de Locomoción Bípeda* ***UPA*** *(Un Paso Adelante) que utiliza el nuevo modelo mecánico* ***DCS*** *(División Cinemática Sagital), y el nuevo algoritmo de planificación de trayectorias* ***M3R*** *(Método Modificado de Marcha Rápida), en el que se basa el nuevo modelo de navegación* ***TCG*** *(Trayectoria Corporal Global). Debido al reciente montaje de la plataforma de experimentación, el sistema robótico carece todavía de sistemas de control avanzados para el seguimiento de las trayectorias de locomoción y navegación generadas para el robot* ***RH0*** *por los nuevos algoritmos, aunque no obstante, se demuestra en este capítulo con varios experimentos reales, la bondad y aplicabilidad para robots humanoides de las nuevas ideas y desarrollos introducidos en esta tesis.*

6.1 El Sistema Robótico - Plataforma de Experimentación.

El robot humanoide **RH0** construido en el **Universidad Carlos III de Madrid** nos sirve como plataforma de experimentación para la implementación real de los nuevos algoritmos presentados en esta tesis. El sistema robótico que constituye esta plataforma experimental está formado por un robot humanoide, cuyas especificaciones mecánicas podemos encontrar en el Apéndice-B, junto con una arquitectura de control cuya estructura general podemos ver en la Figura 6-1.

Está fuera del alcance de esta tesis la descripción detallada tanto del diseño mecánico como de la arquitectura de control del robot humanoide **RH0**. No obstante, para un entendimiento más completo de la aplicación de los algoritmos presentados, resulta útil conocer algunos datos acerca de la estructura mecánica del humanoide (ver Apéndice-B). El robot se ha diseñado como un sistema compacto de poco peso, con la típica estructura de "cantilever" para permitir un alcance amplio de las piernas en el proceso de locomoción bípeda, que produzca un par flector reducido en la cadera.

La arquitectura de control se diseñó con un objetivo muy claro de reducción de pesos, para lo que se racionalizaron los sistemas de control utilizando dispositivos compactos. En la Figura 6-1 se muestra esta arquitectura que utiliza una red de comunicación con **bus CAN** dual, un sistema de computadoras a bordo del humanoide y un interfaz hombre-máquina. El bus se usa para el control y supervisión de las articulaciones del robot mediante controladores embebidos, así como para la recepción de la información procedente de sensores. El sistema de hardware a bordo del robot está formado por dos computadoras basadas en **PC-104**. El interfaz hombre-máquina está basado en una **PDA** inalámbrica conectada con las computadoras de a bordo.

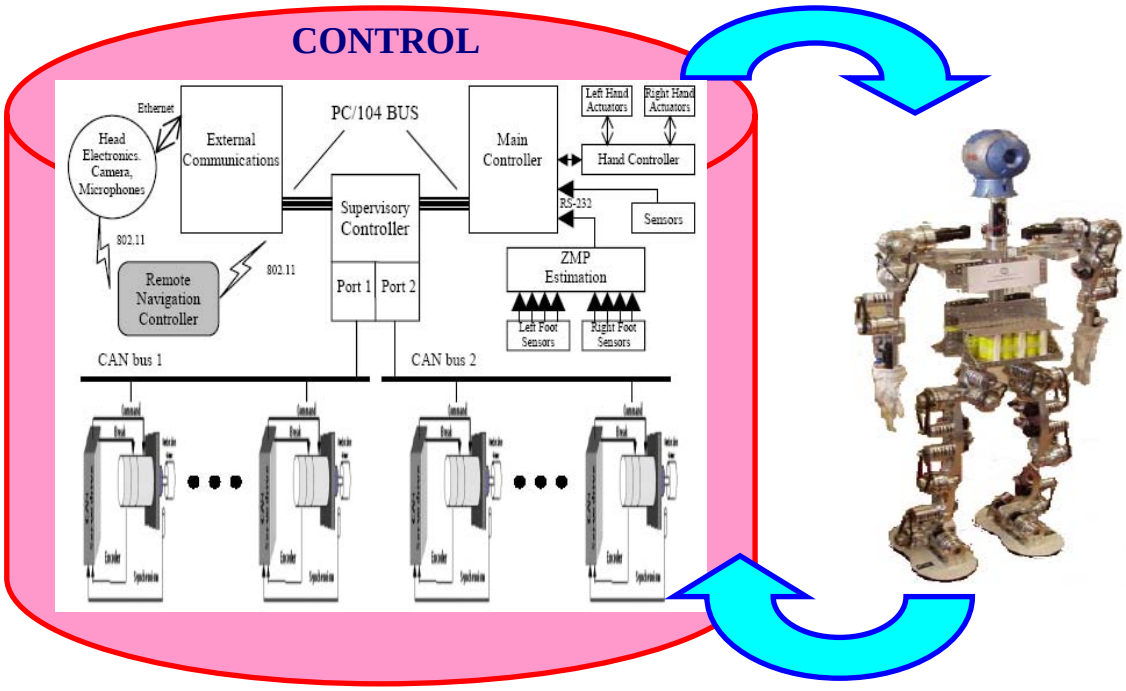


Figura 6-1: Plataforma de Experimentación – Robot RH0 y Arquitectura de Control.

6.2 Locomoción Bípeda Real del Humanoide RH0.

Tras haber desarrollado los nuevos modelos teóricos y algoritmos para la locomoción bípeda de un humanoide con el simulador **RobManSim**, tal y como vimos en 5.3, pasamos ahora a su aplicación y experimentación real sobre el robot **RH0**.

El montaje y ajuste mecánico del robot **RH0** es un trabajo complejo del que no daremos más detalles al salir fuera del alcance de esta tesis, aunque obviamente los datos fundamentales de la estructura electromecánica del robot (ver Apéndice-B), así como los de las restricciones de las articulaciones, son absolutamente necesarios para la implementación real de los nuevos desarrollos de la tesis para este humanoide. Tras el montaje, son necesarios numerosos ensayos para la preparación y ajuste de todas y cada una de las articulaciones del robot **RH0**, así como posteriormente de los miembros del humanoide como subconjuntos independientes, antes de poder abordar una locomoción bípeda real. Todos estos ensayos previos se realizaron por motivos de seguridad con el robot **RH0** amarrado a una sistema de soporte fijo, como se puede apreciar en la fotografía izquierda de la Figura 6-2. Igualmente por motivos de seguridad, para los primeros ensayos de locomoción bípeda completa, se ha mantenido un soporte de seguridad tal y como se puede ver en la fotografía derecha de la Figura 6-2, aunque tras el progreso de las pruebas, se han realizado numerosos experimentos con el robot humanoide totalmente libre, como se detallará en los siguientes apartados 6.2.1 y 6.2.2 para una locomoción bípeda en línea recta y de giro, respectivamente.

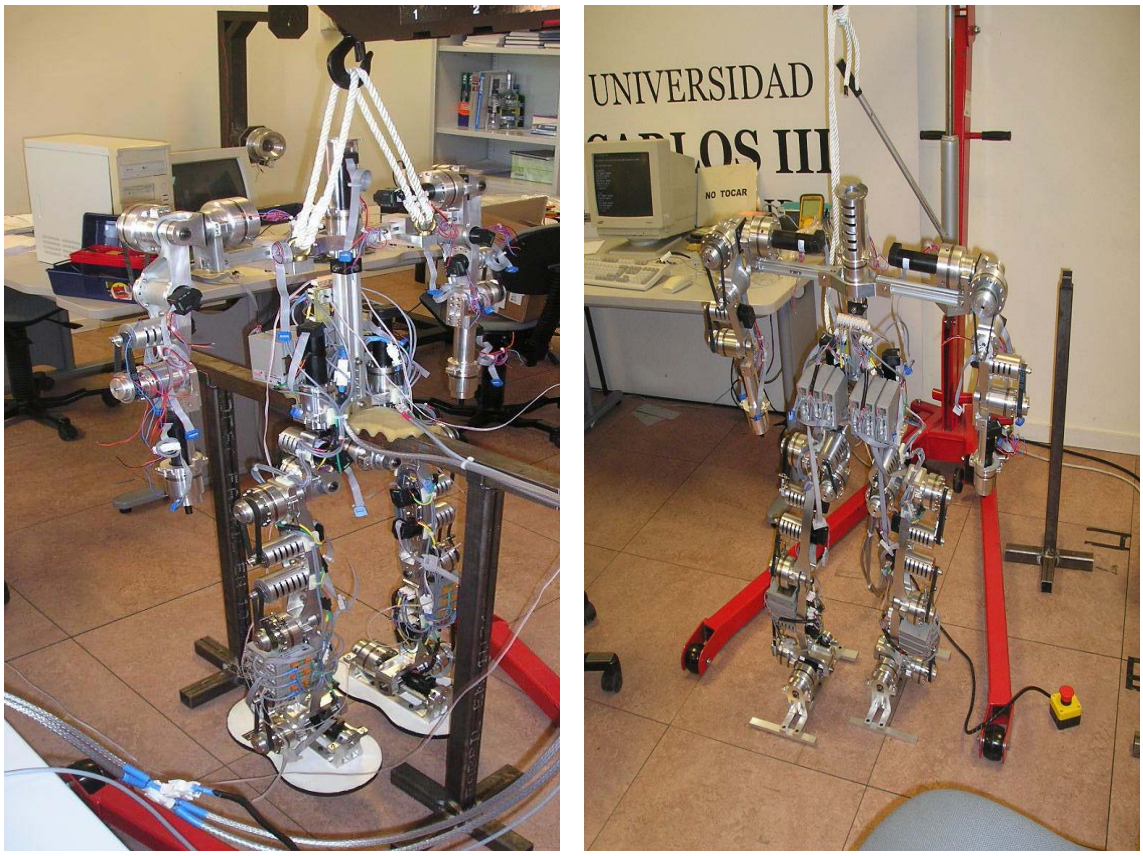


Figura 6-2: El RH0 en su soporte de trabajo.

Par resolver la locomoción bípeda del humanoide **RH0** aplicaremos el nuevo algoritmo **UPA** (ver 3.3). Como ya vimos en detalle en 3.5, el **UPA** recibe como entradas el objetivo local hacia el que se debe dirigir el robot y las características típicas para la locomoción bípeda deseada del robot (i.e. Longitud de paso **Lp**, Altura de paso **Hp**, Anchura de paso **Wp**, Giro máximo del centro de masas **Gcm**, Altura del centro de masas **Hcm**, Coeficiente cosenoidal para las trayectorias **Kcm**, Radio del área de soporte **RAas**, Potencial para generación de trayectorias **Pocm** y Cadencia **Ca**), para desarrollar en cinco fases cada ciclo del paso (según 3.1.). Los resultados obtenidos del **UPA** son los datos correspondientes, en primer lugar a las trayectorias de los **GDL** virtuales $\boldsymbol{\theta v_{PM}}$ del pie móvil (**PM**) durante el ciclo del paso (i.e., trayectoria de la posición y rotación del **PM**), y en segundo lugar a las trayectorias de los **GDL** virtuales $\boldsymbol{\theta v_{CM}}$ del **CM** del humanoide durante el ciclo del paso (i.e., trayectoria de la posición y rotación del **CM**, que define el movimiento global del humanoide), que por supuesto respetará las restricciones dadas para el equilibrio (ver 3.1.1) de la locomoción bípeda, esto es, que el **ZMP** "(3-5)"se encuentren en todo momento dentro del área de soporte. Tras la ejecución del **UPA**, generamos el movimiento de locomoción bípeda, resolviendo el problema cinemático inverso mediante técnicas matemáticas de Lie, como se han implementado anteriormente (ver 3.4.1 y 3.4.2) para el humanoide **RH0.**

El algoritmo **UPA** resuelve un único paso del humanoide **RH0**. Una primera razón para ello es que en la experimentación real con el robot, tras cada ejecución de un paso, es conveniente integrar información sensorial del entorno antes de dar el siguiente. Una segunda razón es la simplicidad del desarrollo, de modo que en este caso, en lugar de resolver una locomoción bípeda compleja de forma completa, el algoritmo **UPA** permite resolverla de modo mucho más eficaz como encadenamiento de soluciones genéricas al problema de un solo paso del humanoide **RH0**.

Debido al reciente montaje del humanoide **RH0**, la plataforma experimental carece todavía de sistemas de control avanzados, esto es, el control se realiza en bucle abierto, lo que obviamente limita en gran medida la obtención de movimientos precisos y rápidos del complejo mecanismo. No obstante, se demuestra con varios experimentos reales la bondad de las nuevas ideas introducidas, como se detallará en los apartados 6.2.1 y 6.2.2 para una locomoción bípeda del humanoide **RH0** de cuatro pasos en línea recta y de giro sobre su propio eje. Estos dos experimentos se realizan en línea con las simulaciones de 5.3.1 y 5.3.2 para una mayor consistencia del discurso de esta tesis y mejor comparación de resultados, que pueda resultar útil didácticamente. Se presentan estos trabajos con datos numéricos, gráficos y videos, donde podemos apreciar que a pesar de todos los problemas que incorpora la realidad a estos experimentos (i.e., las señales de referencia para las articulaciones se aplican en bucle abierto, aparecen interferencias de control, existen flexibilidades mecánicas de los miembros y pandeos de las estructuras que no se tuvieron en cuenta en las simulaciones), el resultado es realmente positivo y el robot humanoide **RH0** ejecuta los movimientos previstos, aún con menor velocidad y elegancia que en las simulaciones (debido a los problemas anteriormente mencionados), demostrando la robustez y aplicabilidad para robots humanoides reales de las nuevas ideas y algoritmos desarrollados en esta tesis.

Para una mayor exposición de los resultados de esta tesis, se incluyen en la documentación electrónica de la misma, videos relativos a otros experimentos reales de locomoción bípeda del robot humanoide **RH0**, como son: caminar con pasos laterales, equilibrio sobre un pie, caminar por un pasillo o caminar al aire libre.

6.2.1 El Humanoide RH0 caminando cuatro pasos en línea recta.

En este apartado vamos a realizar un experimento de locomoción bípeda con el robot **RH0** por el que el humanoide ejecuta cuatro pasos en línea recta, lo que nos permite comprobar el funcionamiento y validez del algoritmo **UPA**. El movimiento propuesto necesita que el robot ejecute los pasos estándar de salida, zancada y entrada (ver 3.5.2.1 3.5.2.2 y 3.5.2.3), de forma que el movimiento completo es como sigue: robot en posición de reposo, paso derecho de salida, zancada izquierda, zancada derecha y paso izquierdo de entrada. Con lo que el robot queda de nuevo en posición erguida.

Las características de la locomoción bípeda del humanoide **RH0** se han elegido de entre las posibles según su mecánica (i.e. respetando los límites cinemáticos y dinámicos del robot - ver apéndice B). Además, debido a las limitaciones del sistema de control y de la plataforma experimental, siguiendo un criterio de prudencia y seguridad, las características se han elegido de una forma más conservadora que la utilizada en la simulación presentada en 5.3.1 (para cuatro pasos en línea recta igualmente), esto es, aquí para el experimento real reducimos el radio del área de soporte **RAas**, para garantizar un equilibrio más estable en la fase de locomoción con apoyo simple, reducimos así mismo la cadencia **Ca** para realizar los pasos con menor inercia en los mecanismos y doblamos la discretización de los datos $\mathbf{S_D}$, puesto que el driver real de las articulaciones no es tan exigente como preveíamos en las simulaciones. Para este experimento las características de la locomoción quedan pues como siguen:

- **Longitud de Paso L_p** = 0,160m
- **Altura de Paso H_p** = 0,100m.
- **Anchura de Paso W_p** = 0,112m.
- **Giro máximo del centro de masas Gcm** = 15^o.
- **Altura del centro de masas Hcm** = 0,58m.
- **Coeficiente Cosenoidal para las trayectorias Kcm** = 0,004m.
- **Radio del área de soporte RAas** = 0,03m.
- **Potencial para generación de trayectorias Pocm** = 4.
- **Cadencia Ca** = 0,043pasos/sg.
- **Discretización de los datos (necesaria por Hardware y Drivers) S_D** = 0,072sg.

Evidentemente, dados los valores de **Ca** y $\mathbf{S_D}$, **cada paso del humanoide RH0 empleará en desarrollarse 23sg**, y cada ejecución del algoritmo **UPA** que resuelve ese paso generará una **tabla con 323 valores para cada GDL** del robot.

Ya mostramos en 3.3.2 como con las cinco fases del **UPA** se obtienen los caminos factibles para un paso. Aquí ejecutamos cuatro veces consecutivas el **UPA** para que se conviertan en cuatro pasos del **RH0**. Esta locomoción viene precedida por un tiempo de paso dedicado a bajar el **CM** desde su posición con el robot completamente erguido, aunque este movimiento no los incluimos en el experimento y partimos con el robot ya en posición de iniciar el primer paso derecho de salida. De este modo los caminos factibles calculados por el algoritmo **UPA** quedan de la siguiente forma:

- **Q_{rCM} - Camino Factible del CM**: Trayectoria compuesta por configuraciones **q_{CMi}** del centro de masas, según "(6-1)".

$$Q_{rCM} = \{q_{CM1} \ldots q_{CM2} \ldots q_{CM3} \ldots q_{CM4} \ldots q_{CM5} \ldots q_{CM16} \ldots q_{CM17} \ldots q_{CM18} \ldots q_{CM19} \ldots q_{CM20}\} \quad \textbf{(6-1)}$$

- **Q_{rZ} - Camino Factible del pie izquierdo**: Trayectoria compuesta por configuraciones **q_{Zi}** del pie izquierdo, según "(6-2)". Viene dada por el camino **Q_{rPM}** resultado del **UPA**, cuando es el pie izquierdo el móvil.

$$Q_{rZ} = \{q_{Z1} \ldots q_{Z2} \ldots q_{Z3} \ldots q_{Z4} \ldots q_{Z5} \ldots q_{Z16} \ldots q_{Z17} \ldots q_{Z18} \ldots q_{Z19} \ldots q_{Z20}\} \quad \textbf{(6-2)}$$

- **Q_{rD} - Camino Factible del pie derecho**: Trayectoria compuesta por configuraciones **q_{Di}** del pie derecho, según "(6-3)". Viene dada por el camino **Q_{rPM}** resultado del **UPA**, cuando el pie móvil es el derecho.

$$Q_{rD} = \{q_{D1} \ldots q_{D2} \ldots q_{D3} \ldots q_{D4} \ldots q_{D5} \ldots q_{D16} \ldots q_{D17} \ldots q_{D18} \ldots q_{D19} \ldots q_{D20}\} \quad \textbf{(6-3)}$$

Una vez obtenidos estos caminos factibles, las trayectorias completas en función de la discretización necesaria por el hardware y software de control **S_D**, se obtienen por interpolación según se detalló en 3.5.2, quedando según podemos ver en la Figura 6-3. Tras obtener los resultados de las ejecuciones del algoritmo **UPA** (i.e., **Q_{rCM}**, **Q_{rZ}** y **Q_{rD}**), generamos los valores de referencia para las posiciones de las articulaciones del robot (i.e., **q1** a **q12**) resolviendo el problema cinemático inverso mediante técnicas de Lie, como se ha implementado (ver 3.4.1 y 3.4.2) para el **RH0**. Estos valores de referencia (i.e., **q1** a **q12**) son los que se aplican a los drivers de cada motor de las articulaciones para generar el movimiento de locomoción bípeda del humanoide **RH0.**

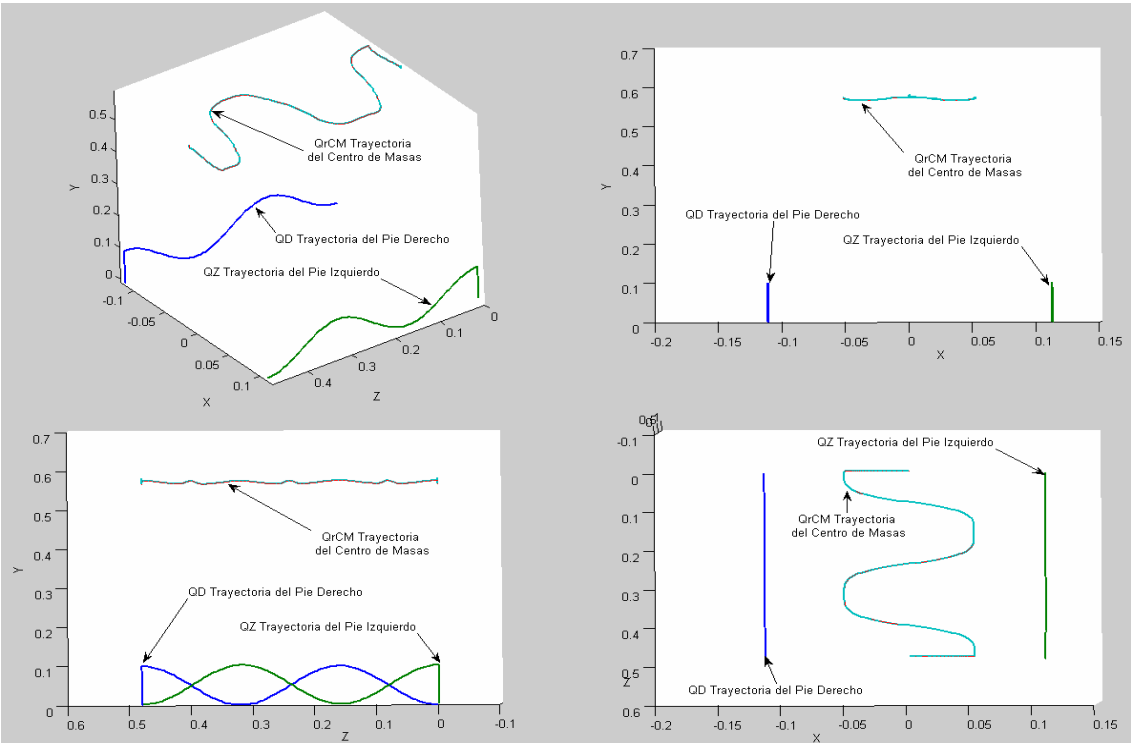


Figura 6-3: Trayectorias de pies y CM del RH0 caminando en línea recta.

Mostramos en la Figura 6-4 algunas imágenes del movimiento de locomoción bípeda real desarrollado en este experimento, en comparación con las imágenes de una simulación con el mismo movimiento del robot humanoide **RH0**.

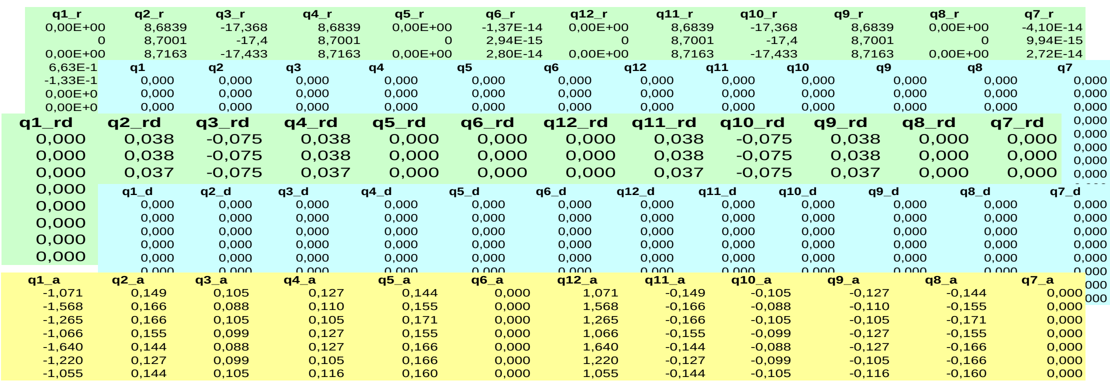
Figura 6-4: Experimento – El RH0 caminando cuatro pasos.

En cuanto a los datos numéricos de este experimento de locomoción bípeda, podremos encontrar en el soporte informático de esta tesis tablas similares a las mostradas en la Figura 6-5 que contienen para todo el movimiento de cuatro pasos en línea recta los siguientes: señales de referencia (i.e., **q1_r** a **q12_r**) y valores reales (i.e., **q1** a **q12**) de las posiciones de los **GDL** del humanoide **RH0** (i.e., posiciones de las articulaciones), valores estimados mediante simulación (i.e., **q1_rd** a **q12_rd**) y reales (i.e., **q1_d** a **q12_d**) de las velocidades de las articulaciones, así como los consumos de los motores correspondientes (i.e., **q1_a** a **q12_a**). Además una representación gráfica de todos estos valores, que puede resultar ilustrativa, se presentará con detalle en las páginas siguientes, aunque para una mejor interpretación hay que tener en cuenta:

- El control de las articulaciones del robot humanoide **RH0** se hace en **BUCLE ABIERTO**, esto es, una vez obtenidas mediante el nuevo algoritmo **UPA** las señales de referencia para las posiciones de los **GDL** (i.e., **q1_r** a **q12_r**), estos valores se envían directamente a los drivers que actúan sobre los motores de las articulaciones, esperando un posicionamiento correcto o al menos aceptable para el movimiento. Esta situación práctica de actuación sin ningún tipo control por realimentación o de regulación automática avanzada, por supuesto provoca que las señales de referencia no sean seguidas con precisión por el complejo sistema mecánico, debido a todo tipo de perturbaciones (ge, flexibilidades, pandeos, vibraciones…), lo que se refleja en las diferencias entre valores de referencia y reales de las variables.

- Para reducir el estrés de la plataforma de experimentación, se ha decidido realizar estos primeros experimentos con el humanoide **RH0** utilizando sólo diez de los veintiún motores del robot (i.e., **q1** a **q5** y **q12** a **q8**), con lo que reducimos especialmente los consumos de energía. Como contrapartida, nos encontramos con la desventaja de que el movimiento de locomoción bípeda real resulta ser menos natural y elegante que el obtenido en las simulaciones, debido a la no utilización de los brazos para el equilibrio y la reducción del movimiento libre de la cadera. Esta situación también se refleja en las diferencias entre valores de referencia y reales de las variables, puesto que el bloqueo de algunas articulaciones afecta al movimiento real de las otras.

Aún a pesar de todas las dificultades presentes comentadas anteriormente y que se irán resolviendo con desarrollos futuros para el robot humanoide **RH0**, el movimiento de locomoción bípeda se ejecuta con razonable eficacia, lo que es una prueba de la robustez de los nuevos algoritmos desarrollados en esta tesis.



q1_r	q2_r	q3_r	q4_r	q5_r	q6_r	q12_r	q11_r	q10_r	q9_r	q8_r	q7_r
0,00E+00	8,6839	-17,368	8,6839	0,00E+00	-1,37E-14	0,00E+00	8,6839	-17,368	8,6839	0,00E+00	-4,10E-14
0	8,7001	-17,4	8,7001	0	2,94E-15	0	8,7001	-17,4	8,7001	0	9,94E-15
0,00E+00	8,7163	-17,433	8,7163	0,00E+00	2,80E-14	0,00E+00	8,7163	-17,433	8,7163	0,00E+00	2,72E-14
6,63E-1											
-1,33E-1											
0,00E+0											
0,00E+0											

q1	q2	q3	q4	q5	q6	q12	q11	q10	q9	q8	q7
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000

q1_rd	q2_rd	q3_rd	q4_rd	q5_rd	q6_rd	q12_rd	q11_rd	q10_rd	q9_rd	q8_rd	q7_rd
0,000	0,038	-0,075	0,038	0,000	0,000	0,000	0,038	-0,075	0,038	0,000	0,000
0,000	0,038	-0,075	0,038	0,000	0,000	0,000	0,038	-0,075	0,038	0,000	0,000
0,000	0,037	-0,075	0,037	0,000	0,000	0,000	0,037	-0,075	0,037	0,000	0,000
0,000											
0,000											
0,000											
0,000											
0,000											

q1_d	q2_d	q3_d	q4_d	q5_d	q6_d	q12_d	q11_d	q10_d	q9_d	q8_d	q7_d
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000

q1_a	q2_a	q3_a	q4_a	q5_a	q6_a	q12_a	q11_a	q10_a	q9_a	q8_a	q7_a
-1,071	0,149	0,105	0,127	0,144	0,000	1,071	-0,149	-0,105	-0,127	-0,144	0,000
-1,568	0,166	0,088	0,110	0,155	0,000	1,568	-0,166	-0,088	-0,110	-0,155	0,000
-1,265	0,166	0,105	0,105	0,171	0,000	1,265	-0,166	-0,105	-0,105	-0,171	0,000
-1,066	0,155	0,099	0,127	0,155	0,000	1,066	-0,155	-0,099	-0,127	-0,155	0,000
-1,640	0,144	0,088	0,127	0,166	0,000	1,640	-0,144	-0,088	-0,127	-0,166	0,000
-1,220	0,127	0,099	0,105	0,166	0,000	1,220	-0,127	-0,099	-0,105	-0,166	0,000
-1,055	0,144	0,105	0,116	0,160	0,000	1,055	-0,144	-0,105	-0,116	-0,160	0,000

Figura 6-5: Valores de referencias, posiciones, velocidades y amperajes de los GDL del RH0.

En la Figura 6-6 se pueden ver las trayectorias de las referencias (azul) y posiciones reales (rojo) de las articulaciones de las piernas del **RH0**. Podemos comprobar que todas (i.e. **q1 a q12**) se encuentran dentro de sus límites mecánicos (ver apéndice B).

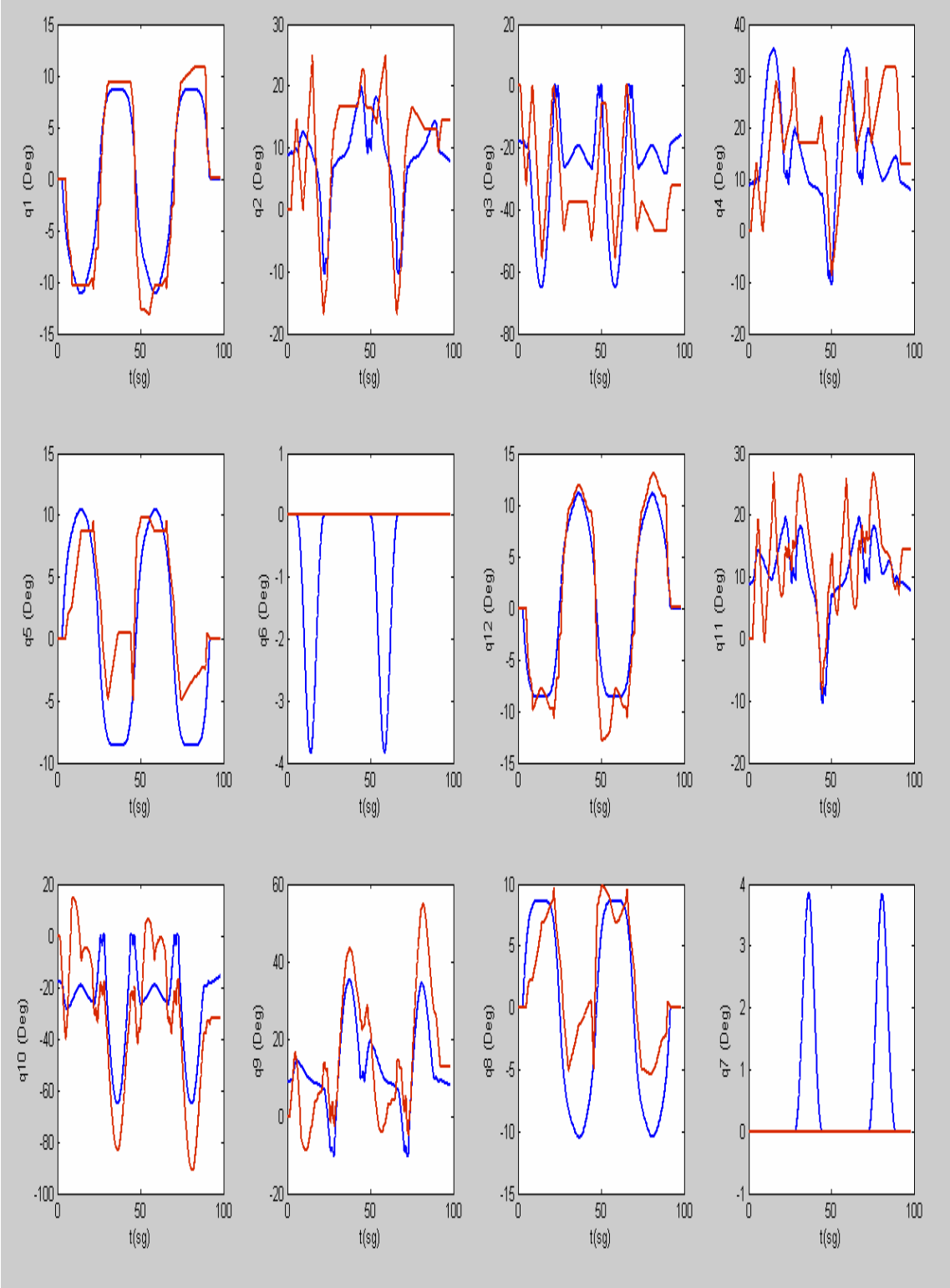


Figura 6-6: Cuatro pasos en línea recta – REFERENCIA y POSICIÓN de los GDL del RH0.

En la Figura 6-7 se pueden ver las velocidades simuladas (azul) y reales (rojo) de las articulaciones (en r.p.m), de las piernas del **RH0**. Podemos comprobar que todas (i.e., **q1d** a **q12d**) se encuentran dentro de sus límites mecánicos (ver apéndice B).

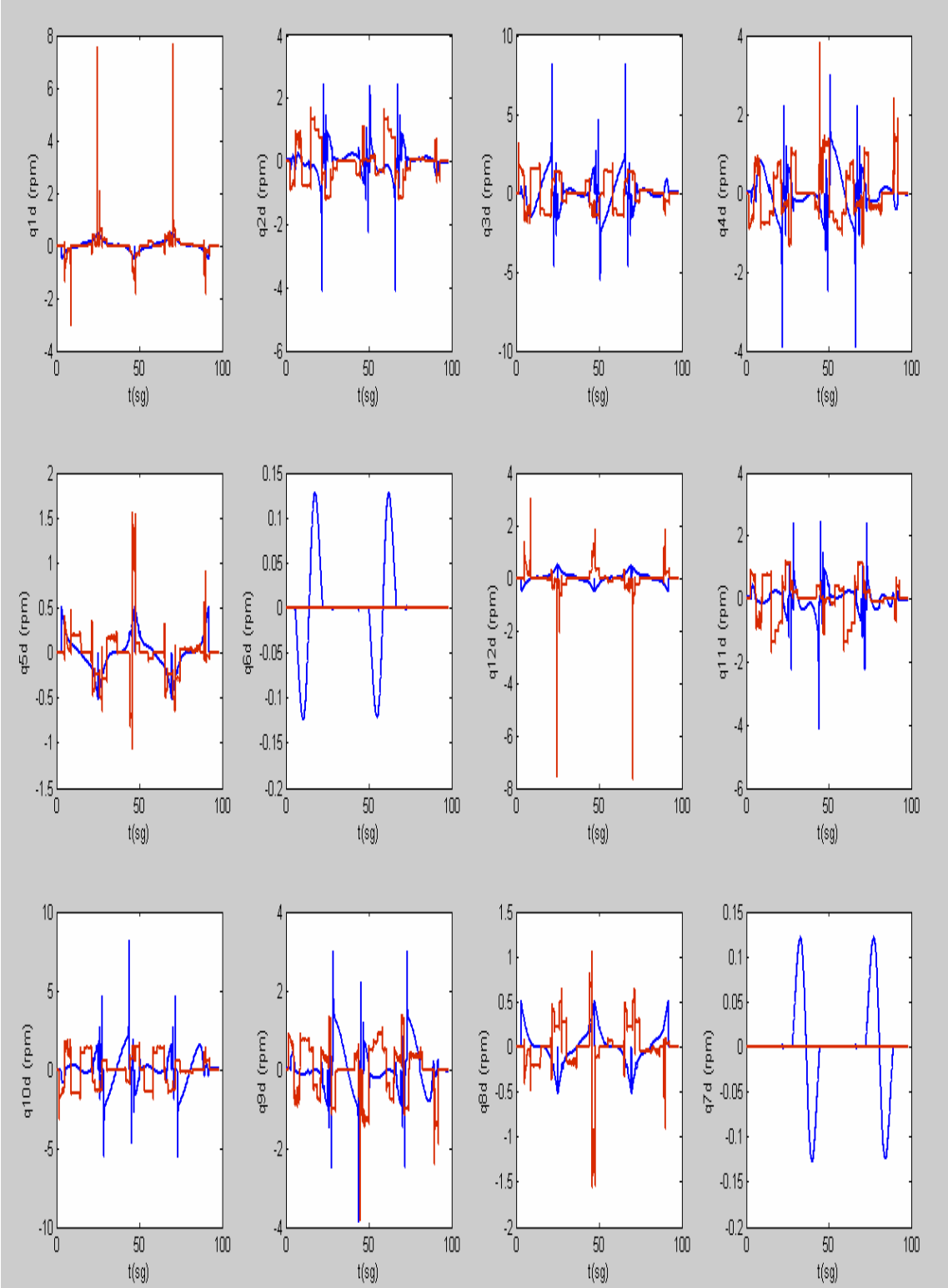


Figura 6-7: Cuatro pasos en línea recta – Simulación y VELOCIDAD de los GDL del RH0.

En la Figura 6-8 se pueden ver los consumos reales de los motores (i.e. **q1a** a **q12a**) de las articulaciones de las piernas del **RH0**.

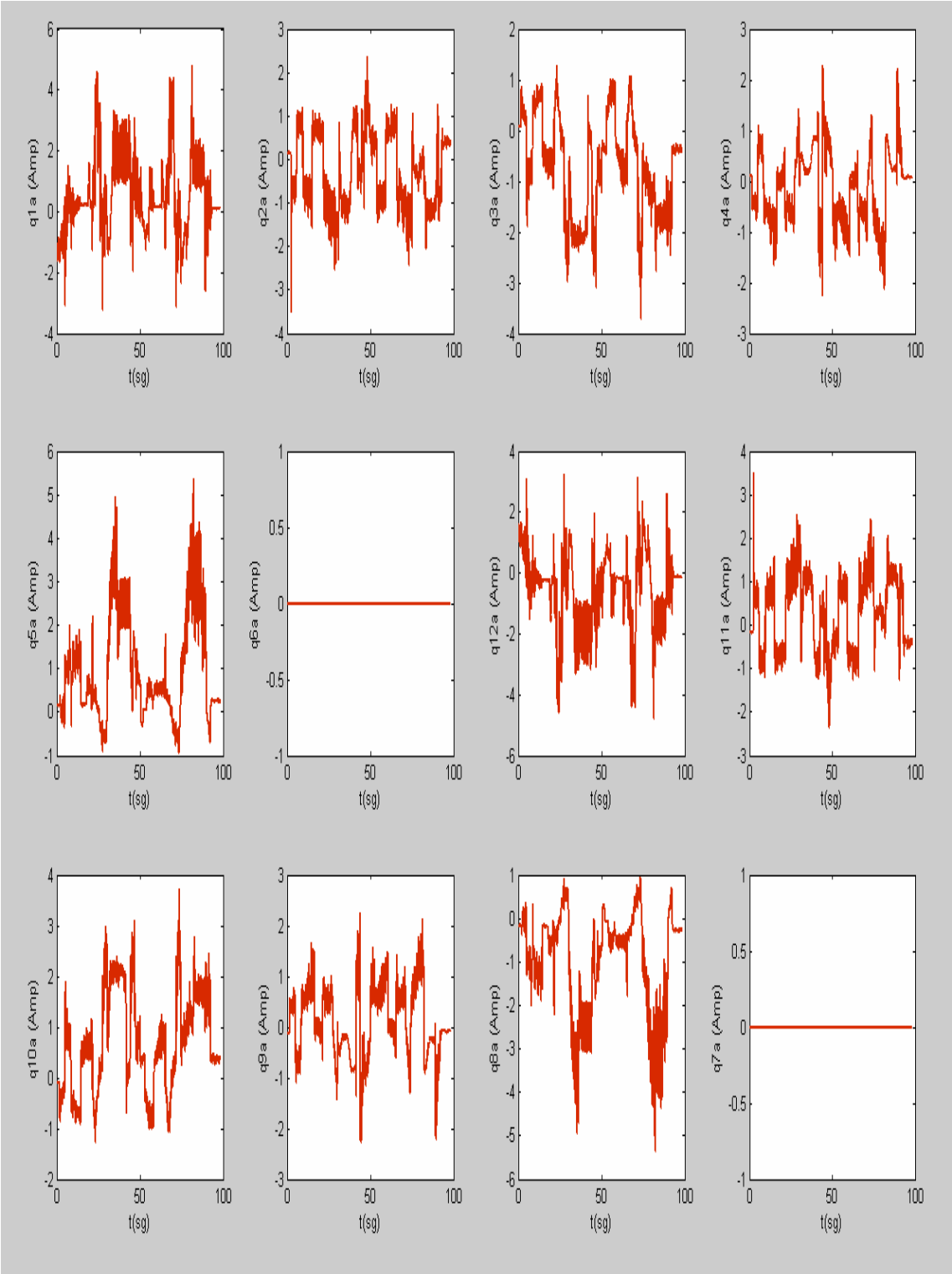


Figura 6-8: Cuatro pasos en línea recta - CONSUMOS de los motores de los GDL del RH0.

6.2.2 El robot RH0 girando 90º sobre su propio eje.

El objetivo de este experimento es generar una locomoción bípeda que haga girar al robot **RH0** (ver Figura 6-9) dando pasos en círculo alrededor de su eje, hasta quedar en una posición girada 90º con respecto a la posición inicial. Para ejecutar el movimiento propuesto, el robot necesita utilizar sólo el paso básico de giro del algoritmo **UPA** que podemos estudiar en 3.5.2.4. El movimiento se realizará ejecutando ocho pasos de giro dobles (i.e. pares pie derecho y pie izquierdo), puesto que cada paso doble de giro produce como resultado una rotación respecto a su propio eje de 10º, aunque podría ser mayor tal y como se mostró en la simulación 5.3.2.

Las características de la locomoción bípeda del humanoide **RH0** se han elegido de entre las posibles según su mecánica (i.e. respetando los límites cinemáticos y dinámicos del robot - ver apéndice B). Además debido a las limitaciones del sistema de control y de la plataforma experimental que se han comentado con detalle en 6.2.1, siguiendo un criterio de prudencia y seguridad, las características se han elegido de una forma más conservadora, esto es, aquí para el experimento real reducimos el radio del área de soporte **RAas**, para garantizar un equilibrio más estable en la fase de locomoción con apoyo simple, reducimos así mismo la cadencia **Ca** para realizar los pasos con menor inercia en los mecanismos y doblamos la discretización de los datos $\mathbf{S_D}$, puesto que el driver real de las articulaciones no es tan exigente como lo que preveíamos en las simulaciones. Para este experimento las características de la locomoción son:

- **Longitud de Paso L_p** = 0,160m
- **Altura de Paso H_p** = 0,100m.
- **Anchura de Paso W_p** = 0,112m.
- **Giro máximo del centro de masas Gcm** = 10º.
- **Altura del centro de masas Hcm** = 0,58m.
- **Coeficiente Cosenoidal para las trayectorias Kcm** = 0,004m.
- **Radio del área de soporte RAas** = 0,03m.
- **Potencial para generación de trayectorias Pocm** = 4.
- **Cadencia Ca** = 0,043pasos/sg.
- **Discretización de los datos (necesaria por Hardware y Drivers) S_D** = 0,072sg.

Figura 6-9: El RH0 girando sobre su propio eje.

Evidentemente, dados los valores de **Ca** y S_D, **cada paso del humanoide RH0 empleará en desarrollarse 23sg**, y cada ejecución del algoritmo **UPA** que resuelve ese paso generará una **tabla con 323 valores para cada GDL** del robot.

En cuanto a los datos numéricos de este experimento de locomoción bípeda, podremos encontrar en el soporte informático de esta tesis tablas similares a las mostradas en la Figura 6-10 que contienen para todo el movimiento de giro las señales de referencia (i.e., **q1_r** a **q12_r**) y valores reales (i.e., **q1** a **q12**) de las posiciones de los **GDL** del humanoide **RH0** (i.e., posiciones de las articulaciones). Una representación gráfica de todos estos valores, se presenta en la página siguiente, aunque para una mejor interpretación hay que tener en cuenta:

- El control de las articulaciones del robot humanoide **RH0** se hace en **BUCLE ABIERTO**, esto es, una vez obtenidas mediante el nuevo algoritmo **UPA** las señales de referencia para las posiciones de los **GDL** (i.e., **q1_r** a **q12_r**), estos valores se envían directamente a los drivers que actúan sobre los motores de las articulaciones, esperando un posicionamiento correcto o al menos aceptable para el movimiento. Esta situación práctica de actuación sin ningún tipo control por realimentación o de regulación automática avanzada, por supuesto provoca que las señales de referencia no sean seguidas con precisión por el complejo sistema mecánico, debido a todo tipo de perturbaciones (e.g., flexibilidades, pandeos, vibraciones…), lo que se refleja en las diferencias entre valores de referencia y reales de las variables.

- Para reducir el estrés de la plataforma de experimentación, se ha decidido realizar estos primeros experimentos con el humanoide **RH0** utilizando sólo diez de los veintiún motores del robot (i.e., **q1** a **q5** y **q12** a **q8**), con lo que reducimos especialmente los consumos de energía. Como contrapartida, nos encontramos con la desventaja de que el movimiento de locomoción bípeda real resulta ser menos natural y elegante en comparación con el obtenido en las simulaciones, debido a la no utilización de los brazos para el equilibrio y la reducción del movimiento libre de la cadera. Esta situación también se refleja en las diferencias entre valores de referencia y reales de las variables, puesto que el bloqueo de algunas articulaciones afecta al movimiento real de las otras.

Aún a pesar de todas las dificultades el movimiento de locomoción bípeda se ejecuta con razonable eficacia, lo que consideramos una prueba de la robustez de los nuevos algoritmos de esta tesis.

q1_r	q2_r	q3_r	q4_r	q5_r	q6_r	q12_r	q11_r	q10_r	q9_r	q8_r	q7_r
0,00E+00	8,6839	-17,368	8,6839	0,00E+00	-1,37E-14	0,00E+00	8,6839	-17,368	8,6839	0,00E+00	-4,10E-14
0	8,7001	-17,4	8,7001	0	2,94E-15	0	8,7001	-17,4	8,7001	0	9,94E-15
0,00E+00	8,7163	-17,433	8,7163	0,00E+00	2,80E-14	0,00E+00	8,7163	-17,433	8,7163	0,00E+00	2,72E-14

6,63E-1	q1	q2	q3	q4	q5	q6	q12	q11	q10	q9	q8	q7
-1,33E-1	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,00E+0	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,00E+0	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000

Figura 6-10: Giro del RH0 – Valores Referencia y Posiciones de los DOF.

En la Figura 6-11 se pueden ver las trayectorias de las referencias (azul) y de las posiciones (rojo) de las articulaciones de las piernas del **RH0** (en grados). Podemos comprobar que todas (i.e. **q1** a **q12**) se encuentran dentro de sus límites mecánicos (ver apéndice B).

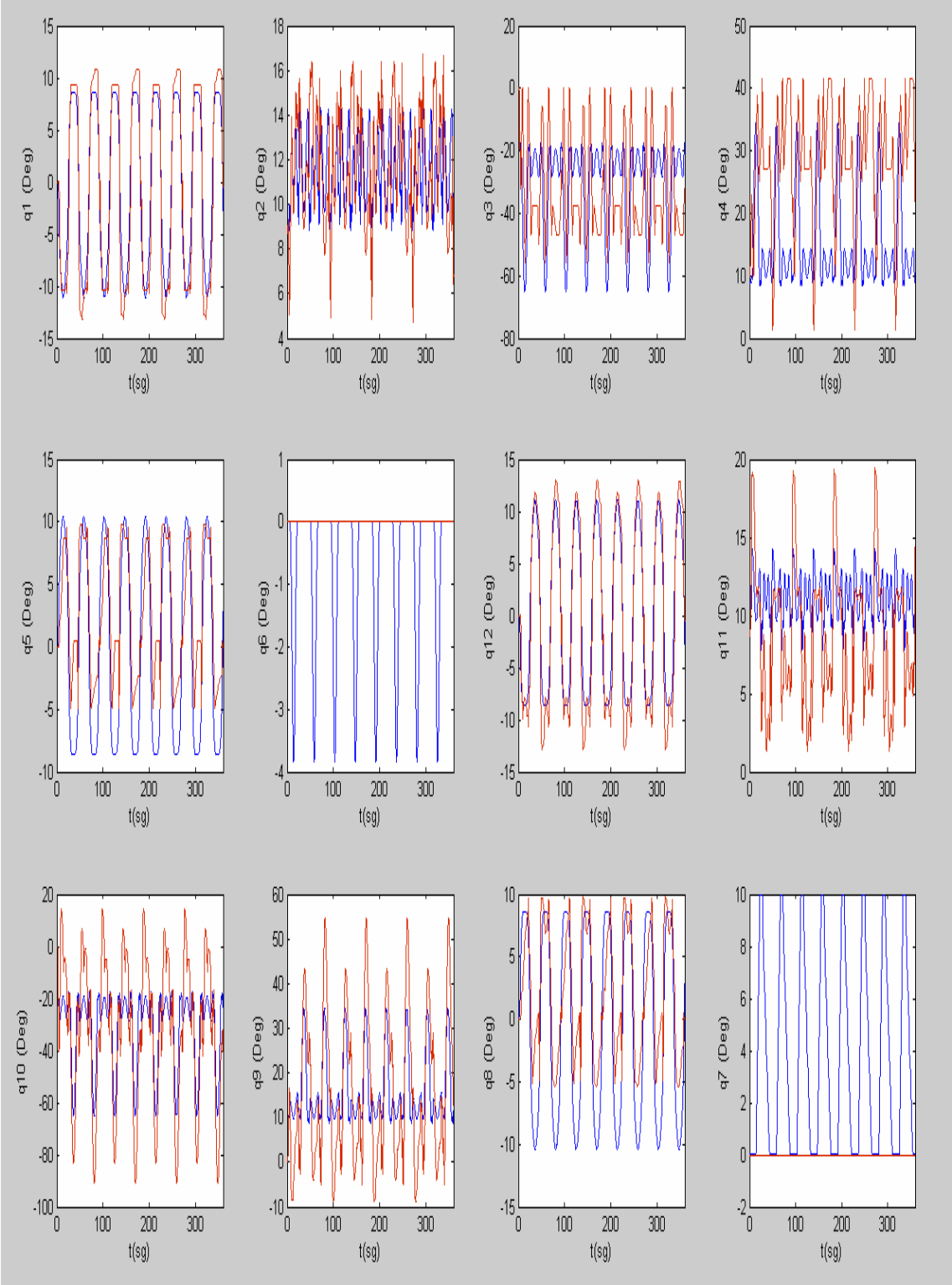


Figura 6-11: RHO girando sobre su eje – REFERENCIA Y POSICIÓN de los DOF.

6.3 Navegación y Planificación experimental del Humanoide RH0.

Como ya vimos en detalle en 5.4, la navegación y planificación de trayectorias para el **RH0** son complejas. Para resolver **Las Trayectorias de Navegación Global y Local del Humanoide RH0** aplicaremos el nuevo algoritmo **TCG** (ver 4.4), obteniendo la trayectoria cinemática libre de colisiones del **ZMP** correspondiente al humanoide, por aplicación directa del algoritmo **M3R** (ver 4.3), dentro del espacio **SE(3)**, sujeto a las restricciones dadas por el mapa disponible del espacio de trabajo. El resultado del algoritmo **M3R** es la trayectoria $\mathbf{Q_r}$ de referencia del algoritmo **UPA** (ver 3.3, 3.5 y 5.3), que resuelve el movimiento de locomoción bípeda del humanoide **RH0**.

6.3.1 Navegación Global para el RH0.

Debido al estado actual de la construcción mecánica del **RH0**, por motivos de seguridad resulta imposible realizar una locomoción bípeda compleja, pero podemos apreciar como trabaja el algoritmo **TCG** en el entorno de trabajo. El núcleo del **TCG** es el nuevo algoritmo **M3R** (ver 4.3) que calcula una trayectoria libre de colisiones para el **CM** del humanoide, sujeto a las restricciones dadas por el mapa del entorno disponible en **SE(3)**, que obviamente para una planificación global debe ser conocido, pero no necesariamente perfecto, puesto que como consecuencia de las propiedades del algoritmo **M3R**, la planificación global puede realizarse para entornos con cualquier tipo de obstáculos (incluso cóncavos, irregulares, interconectados o imprecisos), con una gran eficacia computacional. Como resultado del experimento, podemos ver en la Figura 6-12 el camino global $\mathbf{Q_r}$ cuasi-óptimo para la Navegación Global del humanoide **RH0** dentro del entorno de trabajo, aún cuando por las limitaciones de construcción este resultado no se va a aplicar como referencia para una locomoción compleja en este momento, se aprecia la validez del algoritmo **TCG**.

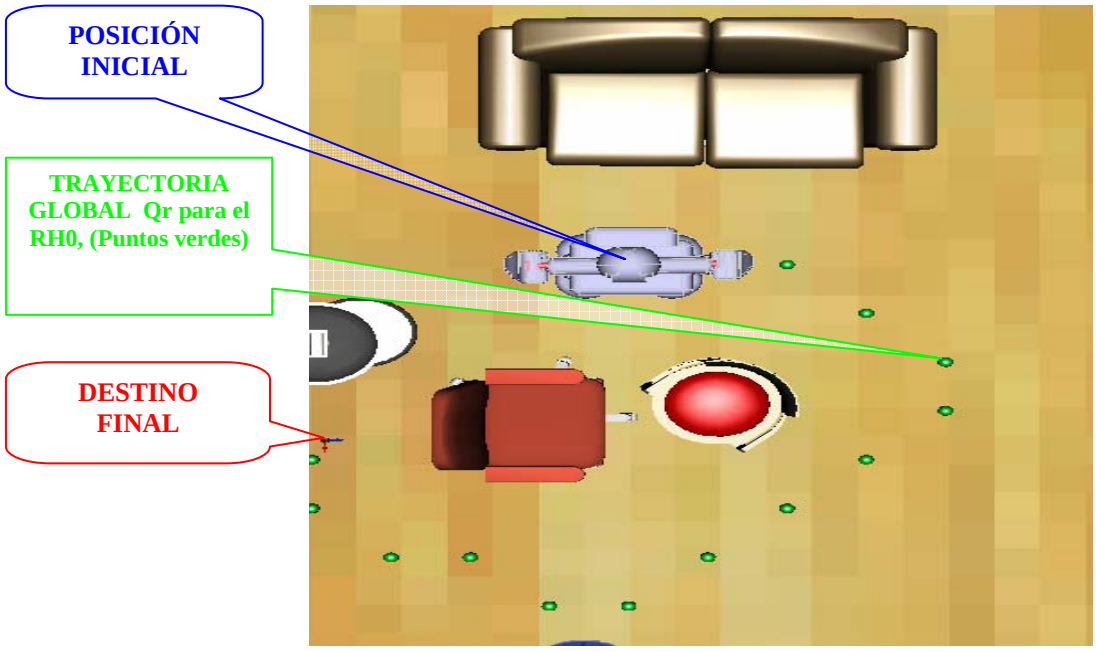


Figura 6-12: Navegación Global con obstáculos para el RH0.

En cuanto a la eficacia del nuevo algoritmo **M3R** (ver 4.3), comprobamos en la experimentación las características teóricas del mismo y comparamos los tiempos de ejecución con otros algoritmos de planificación de trayectorias. Destacamos que el **M3R** tiene un tiempo de ejecución medio (dependiendo de la configuración del entorno) menor de ***50ms***, siendo unas diez veces más rápido que el algoritmo **FMM**, veinte veces más rápido que uno de búsqueda probabilística, e incomparablemente más rápido que algoritmos de inteligencia artificial. Además, el algoritmo **M3R** es más rápido cuanto mayor es el número y tamaño de los obstáculos en el entorno de trabajo, no tiene problemas de mínimos locales (e.g. la configuración de este experimento impide que otros algoritmos puedan siquiera resolver el problema de navegación global), y sirve para cualquier tipo de obstáculos, encontrando SIEMPRE una solución geométrica de $\mathbf{Q_r}$ como trayectoria de Navegación para el humanoide **RH0**.

6.3.2 Planificación Local de Trayectorias para el RH0.

La operación real con el humanoide **RH0** requiere un análisis por integración sensorial antes de ejecutar cada paso de locomoción bípeda mediante el algoritmo **UPA**. El algoritmo utilizado para esta planificación local es el mismo **M3R** (ver 4.3), que resulta aún más eficaz, dado que el espacio de trabajo que tiene que analizar es más reducido, de forma que el tiempo de ejecución para el experimento de la Figura 6-13 es de tan solo ***3,5ms***. Los puntos que componen la trayectoria cuasi óptima **Qr** de la Planificación para un movimiento Local del **RHO**, que resultan de la aplicación del algoritmo **M3R** para el entorno real mostrado, se pueden ver en la Figura 6-13.

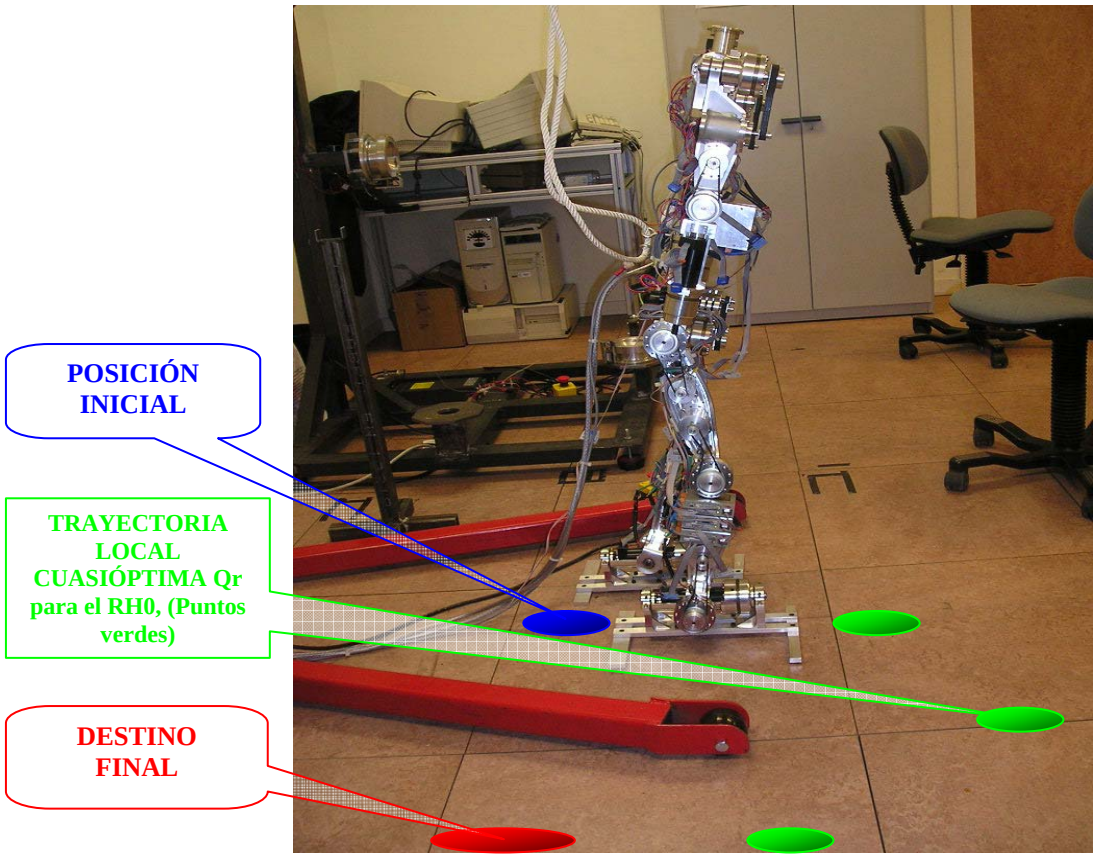


Figura 6-13: Navegación Local del Humanoide RH0.

7 Conclusiones.

La investigación con humanoides continúa siendo un gran desafío técnico, debido a que son sistemas muy complejos con muchos ***GDL*** *y restricciones.* ***Esta tesis se plantea con el objetivo de buscar una Solución Completa, más eficaz que las existentes, para resolver en tiempo real los problemas de Locomoción y Navegación Bípedas de robots Humanoides****. Creemos que los mejores diseños y aplicaciones son concebidos con elegancia de pensamiento y que la abstracción nos ahorra tiempo a largo plazo, lo cual que esta tesis presenta una formulación matemática algo más abstracta de la habitual, basada en* ***Geometría Diferencial****, para la resolución de los problemas. Se ha comprobado la eficacia de los nuevos modelos y algoritmos desarrollados en esta tesis con el* ***humanoide RH0*** *de la* ***Universidad Carlos III de Madrid****.*

7.1 Sumario.

Esta tesis pionera en España, presenta una ***Solución Completa*** para resolver ***los problemas de Locomoción y Navegación Bípedas de Robots Humanoides***, mediante la introducción de nuevos modelos y algoritmos geométricos de propósito general, no presentados anteriormente en la literatura. Las nuevas soluciones introducidas son flexibles, eficaces y susceptibles de ser aplicadas en tiempo real. Además, la tesis aplica esos nuevos desarrollos en un **nuevo Simulador de Realidad Virtual (RobManSim)** y en experimentos reales con el humanoide **RH0** de 21 grados de libertad, construido en la **Universidad Carlos III de Madrid**.

Para la mecánica y control de la Locomoción Bípeda del humanoide, introducimos formulaciones de ***Geometría Diferencial***, basadas en herramientas matemáticas de la teoría de **Grupos y Álgebras de Lie** a través del **SE(3)**. Para la Navegación Bípeda del robot, presentamos soluciones de **Geometría Computacional**, cimentadas en técnicas proporcionadas por la teoría de las **Leyes de Conservación Hiperbólica** aplicadas al análisis de interfaces en evolución.

Los trabajos de esta tesis son atractivos porque proporcionan un marco matemático unificado para la robótica. Se presentan soluciones geométricas cerradas sin problemas de convergencia, muy útiles para trabajos en tiempo real. La independencia de esta formulación con respecto a la elección de los sistemas de referencia es otra ventaja. La diferenciación de las ecuaciones con respecto a parámetros mecánicos resulta muy sencilla, puesto que la exponencial de matrices es la primitiva básica en la que se asienta la formulación geométrica diferencial. Clasificamos las contribuciones de esta tesis en tres grupos que veremos seguidamente: nuevos modelos teóricos, nuevos desarrollos algorítmicos y nuevos trabajos técnicos.

NUEVOS MODELOS TEÓRICOS de esta Tesis - Las más relevantes que se han abordado y desarrollado son:

- **"División Cinemática Sagital" (DCS):** Este nuevo modelo mecánico de un humanoide es presentado en 3.2 y permite obtener soluciones cerradas del problema cinemático inverso, soslayando los problemas de bucles cinemáticos y falta de conexión del robot con la referencia durante la locomoción bípeda, que presentan otros enfoques. En 3.4 este concepto se aplica al humanoide **RH0.**

- **"Trayectoria Corporal Global" (TCG):** En 4.4 introducimos este nuevo modelo de Navegación geométrica para humanoides, que se basa en la utilización del nuevo algoritmo **M3R**. En 5.4 y 6.3 esta idea se prueba con el **RH0.**

- "**Pardos Uno – Translación a una cierta distancia**": En 2.3.4 se presenta este nuevo problema canónico, que es una nueva contribución teórica para resolver con matemática de Lie problemas mecánicos de robots con articulaciones prismáticas de translación. Se demuestra en A.1 para un robot Stanford.

- "**Aproximación Lineal de la Ecuación Eikonal":** En 4.3.2 se introduce esta nueva aproximación de la Ecuación Eikonal, que es una formulación lineal (4-17)(4-18). Nos servirá para obtener soluciones directas, abandonando las tradicionales aproximaciones cuadráticas por iteración, que suelen resultar con problemas de convergencia. Esta contribución es la base del algoritmo **M3R**.

- "**Trayectorias Naturales para el Centro de Masas y los pies del Humanoide en el paso de Locomoción Bípeda":** En 3.5.2 se presentan estas nuevas ecuaciones eficaces para las trayectorias de los pasos básicos del robot (e.g. Salida, Zancada, Entrada, Giro) aplicados al **RH0**, con las cuales se pueden construir locomociones complejas. Se basan en el análisis de la Biomecánica humana.

NUEVOS DESARROLLOS ALGORÍTMICOS de esta Tesis - Los más importantes que se han diseñado, desarrollado y probado son:

- **"Un Paso Adelante" (UPA):** En 3.3 se presenta y desarrolla este nuevo algoritmo para la Locomoción Bípeda de un humanoide genérico. Básicamente, dado un objetivo para el movimiento, se resuelve de forma geométrica directa la cinemática inversa del humanoide necesaria para que éste avance un paso hacia ese objetivo, respetando las restricciones mecánicas, del entorno y de equilibrio. Un movimiento más complejo, resulta tan sencillo como concatenar objetivos desde un planificador global, para que el humanoide vaya generando un movimiento continuo correcto, como encadenamiento de ejecuciones de este algoritmo. En 5.3 y 5.5 se muestran simulaciones usando este algoritmo. En 6.1 el algoritmo **UPA** se aplica a experimentos reales con el humanoide **RH0**.

- **"Método Modificado de Marcha Rápida" (M3R)**: En 4.3 se desarrolla este nuevo algoritmo para la Navegación, que resuelve de forma geométrica la planificación de trayectorias libres de colisiones en un espacio tridimensional, sea cual fuere la estructura del entorno de trabajo y el tipo de obstáculos, esto es, funciona igual con objetos convexos, cóncavos, interconectados e irregulares. El nuevo algoritmo se basa en las leyes de conservación hiperbólica y presenta ventajas en comparación con otros métodos, puesto que tiene la convergencia de orden uno garantizada. En 5.4 y 5.5 se muestran simulaciones y en 6.3 el algoritmo **M3R** se aplica a experimentos con el **RH0**.

NUEVOS TRABAJOS TÉCNICOS en esta Tesis - Tenemos que destacar:

- **"Simulador para Robots Humanoides con Realidad Virtual – RobManSim"** (ver capítulo 5): Creado para un entorno de simulación **VRML**, nos permite desarrollar y probar las soluciones algorítmicas con productividad y sin riesgo, antes de pasar a los experimentos con el **RH0**. Se ha construido un Entorno de Simulación Integrado con un Interfaz Gráfico de Usuario.

- **"Librería de Software RobotMan"** (ver apéndice D): Construida como *Toolbox* de **MATLAB** con todo el código fuente disponible. Sirve para el desarrollo de aplicaciones basadas en la teoría de **Grupos de Lie**. En las ecuaciones que aparecen en la redacción de esta tesis hay referencias a funciones de esta librería, de modo que este apéndice es una ayuda muy importante para entender en detalle las nuevas ideas y algoritmos presentados.

7.1.1 Palabras Clave.

Locomoción Bípeda, Navegación, Robot Humanoide, Grupos de Lie, Producto de Exponenciales **POE**, Problemas de **Paden-Kahan**, Cinemática Inversa, Dinámica Inversa, División Cinemática Sagital **DCS**, Un Paso Adelante **UPA**, Método Modificado de Marcha Rápida **M3R**, Trayectoria Corporal Global **TCG**, Ecuación Eikonal.

7.2 Trabajo Futuro.

Esperamos que la elegancia de la teoría subyacente a estos trabajos anime a quien lea esta tesis para avanzar en la misma línea de investigación, para resolver el modelado dinámico de robots humanoides, incluyendo dinámicas complejas del entorno, problemas de manipulación de objetos y problemas de elasticidad en las superficies. Ahora que se están finalizando los proyectos de construcción electromecánica del humanoide **RH0** de la **Universidad Carlos III de Madrid** como gran plataforma real de ensayos y experimentación, es posiblemente el mejor momento para abordar estos y otros trabajos de futuro, que se han salido del alcance de esta tesis precisamente por no haber podido disponer de un robot como el **RH0** con mayor antelación.

En cuanto a los **PROBLEMAS TEÓRICOS** por investigar, tenemos:

- **"Problema Dinámico Inverso de Humanoides":** La continuación natural del desarrollo para la cinemática inversa, sería completar los esquemas de resolución geométricos para la dinámica de humanoides que ya se han introducido. Tras obtener de forma geométrica, tanto la matriz de inercias **M** como la de Coriolis **C** en 2.4.2, nos faltaría hacer lo mismo para la matriz **V** de fuerzas gravitatorias y de fricción.
- **"Control Dinámico de Humanoides":** Hay todavía mucha investigación por delante para completar la formulación de par computado en términos de **Álgebra de Lie** que presentamos en 2.5, incluyendo el tratamiento de restricciones y perturbaciones. Lo que es más, todo la teoría de regulación automática nos puede traer nuevas soluciones adaptables a la matemática subyacente, especialmente para añadir robustez al sistema de control.

En lo referente a otros **PROBLEMAS ALGORÍTMICOS** por desarrollar:

- **"Locomoción Bípeda Dinámica":** Necesitaremos algoritmos más eficientes para resolver el problema dinámico de la locomoción bípeda del humanoide, ya que las formulaciones geométricas de la dinámica que hemos introducido, procesan un volumen de información mucho mayor que las soluciones cinemáticas y no son adecuados para tiempo real.
- **"Navegación Bípeda y Planificación de Movimientos":** Extender las soluciones del algoritmo **M3R** para resolver la planificación de movimientos del humanoide en el espacio de trabajo y en el espacio de las articulaciones, esto es, hacer evolucionar el algoritmo para mayor número de dimensiones. Además, integrar información sensorial en la planificación local.

En cuanto a los **PROBLEMAS TÉCNICOS** por mejorar:

- **"Simulador de Realidad Virtual Extendido":** Creación de un entorno genérico de simulación por objetos, que nos permita ensayar una variedad de robots.
- **"Librería de Software RobotMan Ampliada":** Incluir nuevos algoritmos y migrarla a otros lenguajes de programación que puedan ser de interés general como **Lenguaje-C**. Desarrollar la librería en lenguajes de formato simbólico como **Maple** y **Mathematica**.

A - Apéndice: Resolución de Mecánicas usando Álgebras de Lie.

Esta tesis usa técnicas de geometría diferencial de Grupos y Álgebras de Lie, de forma que con la introducción del grupo ***SE(3)****, su álgebra* ***se(3)*** *y la fórmula del* ***POE****, podemos representar de forma adecuada la mecánica del sólido rígido. Para una introducción más detallada del uso práctico de estas técnicas matemáticas, se presentan en este apéndice varios ejemplos completos, como son las soluciones geométricas cerradas para la cinemática inversa de robots tipo* ***STANFORD*** *y* ***PUMA****, que hacen uso tanto de las herramientas clásicas de Lie, como de los desarrollos de esta tesis (e.g., problema canónico presentado en 2.3.4). La revisión previa de estos ejemplos permite seguir más claramente los desarrollos matemáticos presentados en el apartado 3.3 de la tesis, para la solución de la mecánica de robots humanoides.*

A.1 Problemas de Robots tipo STANFORD de 6 GDL.

Analizaremos la cinemática de un robot manipulador tipo Stanford que tiene cinco articulaciones de rotación (θ_1, θ_2, θ_4, θ_5, θ_6) y una de traslación θ_3. Se puede ver el robot en el gráfico derecho de la Figura A-1, junto con un esquema cinemático que se repite en la parte izquierda de la figura para mayor claridad.

Se hace notar que el problema NO se encuentra simplificado, en el sentido de que el sistema de referencia de la herramienta del robot **H**, no se encuentra situado en el punto de cruce de los ejes de los tres últimos grados de libertad **p**, como sucede en ejemplos estándar presentados con otro tipo de herramientas matemáticas. De esta forma no podemos aplicar desacoplo cinemático para resolver el problema cinemático inverso, que permanece completo, con doce ecuaciones y seis incógnitas a resolver.

Veremos que con el **POE** de álgebras de Lie es muy sencillo resolver tanto la cinemática directa como inversa, con tan sólo conocer su esquema cinemático, esto es, los ejes de actuación de los **GDL** y las posiciones iniciales de la herramienta **H** y el sistema de referencia inercial **S**. El problema cinemático inverso requiere utilizar los problemas canónicos (ver 2.3.3) y un poco de cálculo algebraico manejando la fórmula de la cinemática directa "(A-4)". El método puede parecer un poco confuso la primera vez que se utiliza, pero tras un par de ejemplos, se aprecia la sencillez, potencia y elegancia de este planteamiento matemático.

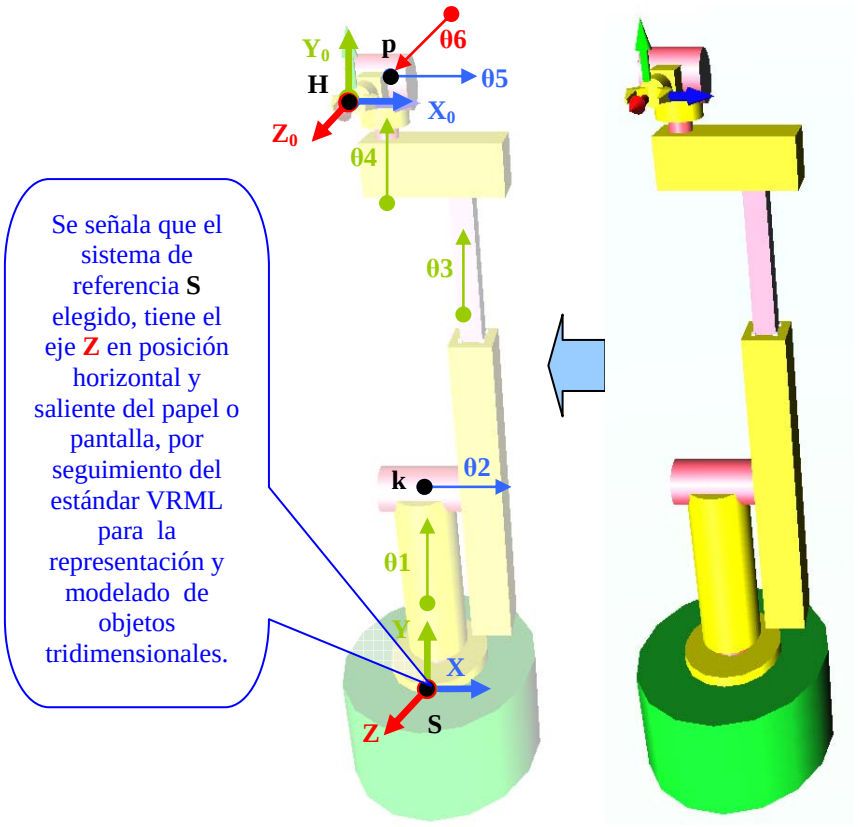


Figura A-1: Esquema Cinemático del Robot tipo STANFORD.

Los ejes de aplicación de los seis **GDL** vienen dados por "(A-1)".

$$\omega_1=\begin{bmatrix}0\\1\\0\end{bmatrix};\omega_2=\begin{bmatrix}1\\0\\0\end{bmatrix};\upsilon_3=\begin{bmatrix}0\\1\\0\end{bmatrix};\omega_4=\begin{bmatrix}0\\1\\0\end{bmatrix};\omega_5=\begin{bmatrix}1\\0\\0\end{bmatrix};\omega_6=\begin{bmatrix}0\\0\\1\end{bmatrix} \quad \textbf{(A-1)}$$

Entonces, los valores de los *twists* ξ para cada una de las articulaciones quedan formulados de una forma sencilla por "(A-2)". {RobotMan-f(18):**prismatictwist**} para la de translación y {RobotMan-f(19):**revolute*twist***} para las de revolución.

$$\left.\begin{aligned}\xi_1=\begin{bmatrix}\upsilon_1\\\omega_1\end{bmatrix}=\begin{bmatrix}-\omega_1\times k\\\omega_1\end{bmatrix};\xi_2=\begin{bmatrix}\upsilon_2\\\omega_2\end{bmatrix}=\begin{bmatrix}-\omega_2\times k\\\omega_2\end{bmatrix};\xi_3=\begin{bmatrix}\upsilon_3\\\omega_3\end{bmatrix}=\begin{bmatrix}\upsilon_3\\0\end{bmatrix}\\ \xi_4=\begin{bmatrix}\upsilon_4\\\omega_4\end{bmatrix}=\begin{bmatrix}-\omega_4\times p\\\omega_4\end{bmatrix};\xi_5=\begin{bmatrix}\upsilon_5\\\omega_5\end{bmatrix}=\begin{bmatrix}-\omega_5\times p\\\omega_5\end{bmatrix};\xi_6=\begin{bmatrix}\upsilon_6\\\omega_6\end{bmatrix}=\begin{bmatrix}-\omega_6\times p\\\omega_6\end{bmatrix}\end{aligned}\right\} \quad \textbf{(A-2)}$$

Siendo la fórmula para la transformación entre **H** y **S** en la configuración de referencia del manipulador $\mathbf{g_{sh}(0)}$, muy sencilla de definir "(A-3)".

$$g_{sh}(0)=\begin{bmatrix}1&0&0&0\\0&1&0&p-S\\0&0&1&H-p\\0&0&0&1\end{bmatrix} \quad \textbf{(A-3)}$$

El problema cinemático directo se soluciona con la simple aplicación del **POE** "(A-4)" {RobotMan-f(3):**forwardkinematics**}, que nos da el valor de $\mathbf{g_{sh}(\theta)}$. Nótese que el tratamiento matemático de la articulación de translación $\boldsymbol{\theta_3}$ es el mismo que el de las articulaciones de rotación.

$$g_{sh}(\theta)=e^{\xi_1\hat{}\,\theta_1}\cdot e^{\xi_2\hat{}\,\theta_2}\cdot e^{\xi_3\hat{}\,\theta_3}\cdot e^{\xi_4\hat{}\,\theta_4}\cdot e^{\xi_5\hat{}\,\theta_5}\cdot e^{\xi_6\hat{}\,\theta_6}\cdot g_{sh}(0) \quad \textbf{(A-4)}$$

Pare resolver el problema cinemático inverso de seis variables y doce ecuaciones, básicamente vamos a operar sobre la ecuación "(A-4)" en cuatro pasos.

Primer paso - Obtención de la variable $\boldsymbol{\theta_3}$: Para ello, pasamos $\mathbf{g_{sh}(0)}$ al otro lado de la ecuación "(A-4)" y aplicamos ambos lados de esta ecuación al punto **p**, para posteriormente hallar el módulo de la diferencia de esos términos con el punto **k**, con lo que obtendremos la ecuación "(A-5)". La razón para operar de este modo, puede no ser evidente en un primer momento, pero la explicamos con detalle a continuación.

$$\left\|g_{sh}(\theta)\cdot g_{sh}(0)^{-1}\cdot p-k\right\|=\left\|e^{\xi_1\hat{}\,\theta_1}\cdot e^{\xi_2\hat{}\,\theta_2}\cdot e^{\xi_3\hat{}\,\theta_3}\cdot e^{\xi_4\hat{}\,\theta_4}\cdot e^{\xi_5\hat{}\,\theta_5}\cdot e^{\xi_6\hat{}\,\theta_6}\cdot p-k\right\| \quad \textbf{(A-5)}$$

Hemos operado de este modo, para obtener una ecuación "(A-5)" que presenta las siguientes características especiales que vamos a explotar.

- Las exponenciales de las variables θ_6, θ_5 y θ_4, representan geométricamente tres movimientos rotativos (tipo *screw*) aplicados sucesivamente a un punto **p**, que se encuentra situado en los ejes de los tres *twists* ξ_6, ξ_5 y ξ_4 correspondientes. De modo que por aplicación de la propiedad "(2-14)" (no afectación del giro sobre su propio eje), para cualesquiera valores de las variables, el punto **p** no quedará afectado (i.e., no se moverá). La consecuencia matemática es que las variables θ_6, θ_5 y θ_4, y sus exponenciales, se pueden eliminar, puesto que no afectan.

- Las exponenciales de las variables θ_1 y θ_2, representan geométricamente dos movimientos rotativos (tipo *screw*) aplicados sucesivamente a un punto dado por el producto de la exponencial de θ_3 por el punto **p**. Posteriormente se obtiene la norma de la diferencia entre el resultado de esas operaciones y un punto **k** que se encuentra situado en los ejes de los dos *twists* ξ_1 y ξ_2 correspondientes. De modo que por aplicación de la de la propiedad "(2-15)" (conservación de la norma), para cualesquiera valores de las variables, la distancia entre el producto de la exponencial de θ_3 por el punto **p** y el punto **k** no quedará afectada. La consecuencia matemática es que las variables θ_1 y θ_2, y sus exponenciales, se pueden eliminar, puesto que no afectan.

Por todo ello, la ecuación "(A-5)" tiene el término izquierdo conocido (valor **δ**) y un término derecho que sólo se vea afectado por el tercer **GDL**, por lo que queda transformada en la ecuación "(A-6)", que no es sino la formulación del **problema canónico Pardos-Uno**, por lo que por aplicación geométrica de éste, obtenemos directamente los dos valores posibles para θ_3 con la función "(2-35)".

$$\delta = \left\| e^{\xi_3 \hat{} \theta_3} \cdot p - k \right\| \xrightarrow{Pardos-UNO} \theta_3 \quad Doble \qquad \textbf{(A-6)}$$

Segundo paso – Obtención de las variables θ_1, y θ_2: Para ello, pasamos $\mathbf{g_{sh}(0)}$ al otro lado de la ecuación "(A-4)" y aplicamos ambos lados de esta ecuación al punto **p**, para obtener la ecuación "(A-7)", que tiene el término izquierdo conocido (igual a un punto cualquiera que llamaremos k'), y un término derecho que no se ve afectado por las variables θ_6, θ_5 y θ_4, debido a la propiedad "(2-14)" (no afectación del giro sobre su propio eje), al estar el punto **p** sobre los ejes de los tres *twists* ξ_6, ξ_5 y ξ_4. El producto de la exponencial de θ_3 por el punto **p** será también un valor conocido (igual a un punto que llamaremos p'), al haber obtenido θ_3 en el primer paso. De esta forma, la ecuación "(A-7)" queda transformada en la ecuación "(A-8)", que no es sino la formulación del **problema canónico PadenKahan-Dos**, por lo que por aplicación geométrica de éste "(2-31)", obtenemos los dos valores posibles para la pareja de variables θ_2 y θ_1.

$$g_{sh}(\theta) \cdot g_{sh}(0)^{-1} \cdot p = e^{\xi_1 \hat{} \theta_1} \cdot e^{\xi_2 \hat{} \theta_2} \cdot e^{\xi_3 \hat{} \theta_3} \cdot e^{\xi_4 \hat{} \theta_4} \cdot e^{\xi_5 \hat{} \theta_5} \cdot e^{\xi_6 \hat{} \theta_6} \cdot p \qquad \textbf{(A-7)}$$

$$k' = e^{\xi_1 \hat{} \theta_1} \cdot e^{\xi_2 \hat{} \theta_2} \cdot p' \xrightarrow{PadenKahan-DOS} \theta_1, \theta_2 \quad Doble \qquad \textbf{(A-8)}$$

Tercer paso – Obtención de las variables θ_4, y θ_5: Para ello, pasamos $\mathbf{g_{sh}(0)}$ y las tres exponenciales ya conocidas (las de los tres primeros **GDL - θ_1, θ_2** y θ_3) al otro lado de la ecuación "(A-4)" y aplicamos ambos lados de esta ecuación al punto **H**, para obtener la ecuación "(A-9)", que tiene el término izquierdo conocido (igual a un punto cualquiera que llamamos k'') y un término derecho que no se ve afectado por la variable θ_6, debido a la propiedad "(2-14)" (no afectación del giro sobre su propio eje), al estar el punto **H** sobre el eje del *twist* ξ_6. De esta forma, la ecuación "(A-9)" queda transformada en la ecuación "(A-10)", que es de nuevo la formulación del **problema canónico PadenKahan-Dos**, por lo que por aplicación geométrica de éste "(2-31)", obtenemos los dos valores posibles para la pareja de variables θ_4 y θ_5.

$$e^{-\hat{\xi}_3\theta_3}\cdot e^{-\hat{\xi}_2\theta_2}\cdot e^{-\hat{\xi}_1\theta_1}\cdot g_{sh}(\theta)\cdot g_{sh}(0)^{-1}\cdot H = e^{\hat{\xi}_4\theta_4}\cdot e^{\hat{\xi}_5\theta_5}\cdot e^{\hat{\xi}_6\theta_6}\cdot H \qquad \textbf{(A-9)}$$

$$k'' = e^{\hat{\xi}_4\theta_4}\cdot e^{\hat{\xi}_5\theta_5}\cdot H \xrightarrow{PadenKahan\ -DOS} \theta_4,\theta_5 \quad Doble \qquad \textbf{(A-10)}$$

Cuarto paso – Obtención de la variable θ_6: Para ello, pasamos $\mathbf{g_{sh}(0)}$ y las cinco exponenciales ya conocidas (las de los cinco primeros **GDL - θ_1, θ_2, θ_3, θ_4** y θ_5) al otro lado de la ecuación "(A-4)" y aplicamos ambos lados de esta ecuación al punto **S**, para obtener la ecuación "(A-11)", que tiene el término izquierdo conocido (igual a un punto cualquiera que llamamos k'''.) De esta forma, la ecuación "(A-11)" queda transformada en la ecuación "(A-12)", que es la formulación del **problema canónico PadenKahan-Uno**, por lo que por aplicación geométrica directa de éste "(2-29)", obtenemos el único valor posible para la variable θ_6.

$$e^{-\hat{\xi}_5\theta_5}\cdot e^{-\hat{\xi}_4\theta_4}\cdot e^{-\hat{\xi}_3\theta_3}\cdot e^{-\hat{\xi}_2\theta_2}\cdot e^{-\hat{\xi}_1\theta_1}\cdot g_{sh}(\theta)\cdot g_{sh}(0)^{-1}\cdot S = e^{\hat{\xi}_6\theta_6}\cdot S \qquad \textbf{(A-11)}$$

$$k''' = e^{\hat{\xi}_6\theta_6}\cdot S \xrightarrow{PadenKahan\ -UNO} \theta_6 \quad Simple \qquad \textbf{(A-12)}$$

De esta forma queda resuelto de forma geométrica, determinista, cerrada y completa el problema cinemático inverso. Lo que es más, no sólo se ha encontrado una solución para el problema (i.e., un conjunto de valores θ_1, θ_2, θ_3, θ_4 , θ_5 y θ_6), sino que de existir, se han resuelto en una sola formulación las ocho posibles soluciones de este problema (i.e., los ocho conjuntos de valores para θ_1, θ_2, θ_3, θ_4 , θ_5 y θ_6), que son el límite teórico para un robot manipulador de tipo Stanford. Obsérvese para ello las combinaciones de las posibles soluciones en "(A-13)".

$$N\_Soluciones = \theta_3 Doble \times \theta_2\theta_1 Doble \times \theta_5\theta_4 Doble \times \theta_6 Simple = 8 \qquad \textbf{(A-13)}$$

Una vez que hemos resuelto en detalle la cinemática inversa para un robot manipulador (en este caso para el robot tipo Stanford), podemos aplicar las mismas herramientas geométricas (y otras similares), para resolver las ecuaciones matemáticas de casi todos los tipos de robots industriales que existen.

En el siguiente punto A.2, veremos otro ejemplo para consolidar las ideas presentadas, pero con menos detalle en las justificaciones matemáticas, ya que son las mismas que acabamos de explicar en este apartado.

A.2 Problemas de Robots tipo PUMA de 6 GDL.

Tras el desarrollo detallado del ejemplo anterior (A.1), analizaremos ahora de una forma mucho más escueta (pues ya se conoce la mecánica de trabajo), la cinemática de un robot manipulador tipo PUMA. En la Figura A-2 se puede ver el robot junto con un esquema cinemático que se repite en la parte izquierda de la figura. El problema tampoco se encuentra simplificado por desacoplo cinemático, por lo que el problema cinemático inverso tiene doce ecuaciones con seis incógnitas a resolver.

Los ejes de aplicación de los seis **GDL** vienen dados por "(A-14)".

$$\omega_1=\begin{bmatrix}0\\1\\0\end{bmatrix};\ \omega_2=\begin{bmatrix}1\\0\\0\end{bmatrix};\ \omega_3=\begin{bmatrix}1\\0\\0\end{bmatrix};\ \omega_4=\begin{bmatrix}0\\1\\0\end{bmatrix};\ \omega_5=\begin{bmatrix}1\\0\\0\end{bmatrix};\ \omega_6=\begin{bmatrix}0\\0\\1\end{bmatrix} \qquad \textbf{(A-14)}$$

Los valores de los *twists* ξ para las articulaciones son formulados por "(A-15)".

$$\left.\begin{aligned}\xi_1=\begin{bmatrix}\upsilon_1\\\omega_1\end{bmatrix}=\begin{bmatrix}-\omega_1\times k\\\omega_1\end{bmatrix};\ \xi_2=\begin{bmatrix}\upsilon_2\\\omega_2\end{bmatrix}=\begin{bmatrix}-\omega_2\times k\\\omega_2\end{bmatrix};\ \xi_3=\begin{bmatrix}\upsilon_3\\\omega_3\end{bmatrix}=\begin{bmatrix}-\omega_3\times r\\\omega_3\end{bmatrix}\\ \xi_4=\begin{bmatrix}\upsilon_4\\\omega_4\end{bmatrix}=\begin{bmatrix}-\omega_4\times p\\\omega_4\end{bmatrix};\ \xi_5=\begin{bmatrix}\upsilon_5\\\omega_5\end{bmatrix}=\begin{bmatrix}-\omega_5\times p\\\omega_5\end{bmatrix};\ \xi_6=\begin{bmatrix}\upsilon_6\\\omega_6\end{bmatrix}=\begin{bmatrix}-\omega_6\times p\\\omega_6\end{bmatrix}\end{aligned}\right\} \qquad \textbf{(A-15)}$$

Siendo la configuración de $\mathbf{g_{sh}(0)}$, muy sencilla de definir mediante "(A-16)".

$$g_{sh}(0)=\begin{bmatrix}1&0&0&0\\0&1&0&p-S\\0&0&1&H-p\\0&0&0&1\end{bmatrix} \qquad \textbf{(A-16)}$$

El problema cinemático directo $\mathbf{g_{sh}(\theta)}$ se soluciona con el **POE** "(A-17)" {RobotMan-f(3):**forwardkinematics**}, y el cinemático inverso en cuatro pasos aplicados sobre ella.

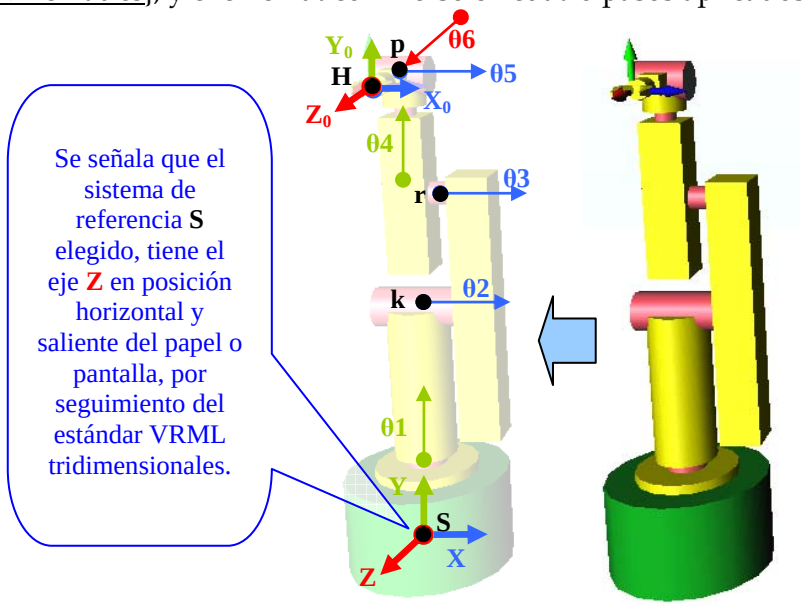


Figura A-2: Esquema Cinemático del Robot tipo PUMA.

$$g_{sh}(\theta) = e^{\xi_1 \hat{} \theta_1} \cdot e^{\xi_2 \hat{} \theta_2} \cdot e^{\xi_3 \hat{} \theta_3} \cdot e^{\xi_4 \hat{} \theta_4} \cdot e^{\xi_5 \hat{} \theta_5} \cdot e^{\xi_6 \hat{} \theta_6} \cdot g_{sh}(0) \quad \textbf{(A-17)}$$

Primer paso - Obtención de la variable θ_3: Pasamos $\mathbf{g_{st}(0)}$ al otro lado de "(A-17)" y aplicamos al punto **p**, para hallar el módulo de la diferencia de esos términos con el punto **k**, con lo que obtendremos "(A-18)". Las variables θ_6, θ_5 y θ_4, se pueden eliminar por la propiedad "(2-14)" y las variables θ_1, y θ_2 por la propiedad "(2-15)". Queda la ecuación "(A-19)", que es un **problema canónico Paden-Kahan-Tres**, por lo que obtenemos los dos valores posibles para θ_3 con la función "(2-33)".

$$\left\| g_{sh}(\theta) \cdot g_{sh}(0)^{-1} \cdot p - k \right\| = \left\| e^{\xi_1 \hat{} \theta_1} \cdot e^{\xi_2 \hat{} \theta_2} \cdot e^{\xi_3 \hat{} \theta_3} \cdots e^{\xi_6 \hat{} \theta_6} \cdot p - k \right\| \quad \textbf{(A-18)}$$

$$\delta = \left\| e^{\xi_3 \hat{} \theta_3} \cdot p - k \right\| \xrightarrow{PadenKahan\ -TRES} \theta_3 \quad Doble \quad \textbf{(A-19)}$$

Segundo paso – Obtención de las variables θ_1, y θ_2: Pasamos $\mathbf{g_{sh}(0)}$ al otro lado de la de "(A-17)", y aplicamos al punto **p**, para obtener "(A-20)", que no se ve afectada por θ_6, θ_5 ni θ_4, debido a "(2-14)", y con el producto de la $\mathbf{exp}\theta_3$ por **p** conocido (p'). Queda "(A-21)", que es un **problema canónico PadenKahan-Dos**, por lo que por "(2-31)", obtenemos los dos valores posibles para la pareja de variables θ_2 y θ_1.

$$g_{sh}(\theta) \cdot g_{sh}(0)^{-1} \cdot p = e^{\xi_1 \hat{} \theta_1} \cdot e^{\xi_2 \hat{} \theta_2} \cdot e^{\xi_3 \hat{} \theta_3} \cdot e^{\xi_4 \hat{} \theta_4} \cdot e^{\xi_5 \hat{} \theta_5} \cdot e^{\xi_6 \hat{} \theta_6} \cdot p \quad \textbf{(A-20)}$$

$$k' = e^{\xi_1 \hat{} \theta_1} \cdot e^{\xi_2 \hat{} \theta_2} \cdot p' \xrightarrow{PadenKahan-DOS} \theta_1, \theta_2 \quad Doble \quad \textbf{(A-21)}$$

Tercer paso – Obtención de las variables θ_4, y θ_5: Para ello, pasamos $\mathbf{g_{sh}(0)}$ y las tres exponenciales ya conocidas al otro lado de "(A-17)" y aplicamos al punto **H**, para obtener "(A-22)", que no se ve afectada por θ_6, debido a la propiedad "(2-14)". Queda la ecuación "(A-23)", que es un **problema canónico PadenKahan-Dos**, por lo que por aplicación de "(2-31)", obtenemos los dos valores para la pareja de variables θ_4 y θ_5.

$$e^{-\xi_3 \hat{} \theta_3} \cdot e^{-\xi_2 \hat{} \theta_2} \cdot e^{-\xi_1 \hat{} \theta_1} \cdot g_{sh}(\theta) \cdot g_{sh}(0)^{-1} \cdot H = e^{\xi_4 \hat{} \theta_4} \cdot e^{\xi_5 \hat{} \theta_5} \cdot e^{\xi_6 \hat{} \theta_6} \cdot H \quad \textbf{(A-22)}$$

$$k'' = e^{\xi_4 \hat{} \theta_4} \cdot e^{\xi_5 \hat{} \theta_5} \cdot H \xrightarrow{PadenKahan\ -DOS} \theta_4, \theta_5 \quad Doble \quad \textbf{(A-23)}$$

Cuarto paso – Obtención de la variable θ_6: Pasamos $\mathbf{g_{sh}(0)}$ y las cinco exponenciales ya conocidas al otro lado de (A-17)" y aplicamos al punto **S**, para obtener (A-24)", que reducida queda en "(A-25)", que es un **problema canónico PadenKahan-Uno**, por lo que por aplicación de éste "(2-29)", obtenemos el valor posible para la variable θ_6.

$$e^{-\xi_5 \hat{} \theta_5} \cdot e^{-\xi_4 \hat{} \theta_4} \cdot e^{-\xi_3 \hat{} \theta_3} \cdot e^{-\xi_2 \hat{} \theta_2} \cdot e^{-\xi_1 \hat{} \theta_1} \cdot g_{sh}(\theta) \cdot g_{sh}(0)^{-1} \cdot S = e^{\xi_6 \hat{} \theta_6} \cdot S \quad \textbf{(A-24)}$$

$$k''' = e^{\xi_6 \hat{} \theta_6} \cdot S \xrightarrow{PadenKahan\ -UNO} \theta_6 \quad Simple \quad \textbf{(A-25)}$$

Quedan resueltas de forma geométrica cerrada las ocho posibles soluciones del problema cinemático inverso. Las combinaciones posibles vienen dadas por "(A-26)".

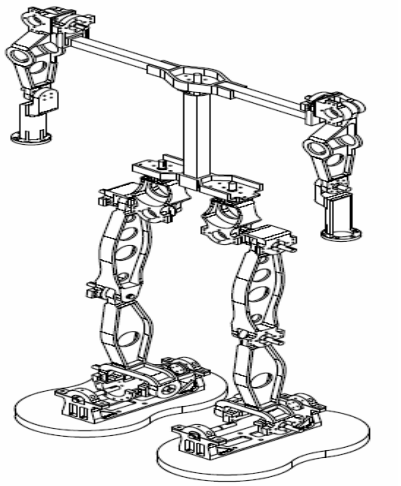
$$N\_Soluciones = \theta_3 Doble \times \theta_2\theta_1 Doble \times \theta_5\theta_4 Doble \times \theta_6 Simple = 8 \quad \textbf{(A-26)}$$

B- Apéndice: Descripción Mecánica del Humanoide RH0.

Aunque está fuera del alcance de esta tesis la descripción detallada tanto del diseño como de la mecánica del robot humanoide ***RH0****, para el entendimiento en profundidad de los algoritmos presentados en esta tesis, resulta necesario conocer algunos datos acerca de la estructura mecánica del* ***RH0****, por lo que incluimos en este apéndice las medidas y valores más relevantes.*

B.1 Medidas y Valores de la Estructura.

Podemos encontrar en la Figura B-1 los detalles acerca de las medidas de la estructura mecánica del robot humanoide **RH0**, así como sus valores de inercia más relevantes. Estos datos son necesarios para la implementación práctica de los algoritmos que se han presentado tanto en 3.3 como en 3.4 para las soluciones cinemáticas y dinámica.



	Masa (gramos)	Peso en el robot	Centro de Gravedad (mm)		
			Gx	Gy	Gz
Pie	1.229,88	2.459,77	-9,57	0,99	-38,41
Tobillo	1.394,91	2.789,82	-0,74	-29,73	-1,75
Tibia	1.023,45	2.046,90	-53,91	-5,94	118,73
Muslo	2.668,05	5.336,09	-48,88	-14,25	386,13
Cadera Inferior	895,82	1.791,64	-39,41	-0,01	551,77
Cadera Media	1.732,80	3.465,60	44,06	-32,37	590,51
Cadera Superior	6.456,79	6.456,79	111,55	-0,06	658,53
Columna	1.487,11	1.487,11	112,00	-0,41	983,91
Hombro	1.131,16	2.262,32	-74,04	-0,50	1.031,50
Punta de Hombro	846,43	1.692,85	-168,11	-33,99	1.038,32
Brazo	1.471,69	2.943,37	-194,66	-27,73	896,99
Antebrazo	861,36	1.722,72	-181,55	-21,59	690,81
		34.454,98			
Mitad Derecha del Robot	22.147,70		-37,74	-12,58	516,42
Robot Entero sin baterias	41.897,96		109,62	-13,70	509,71

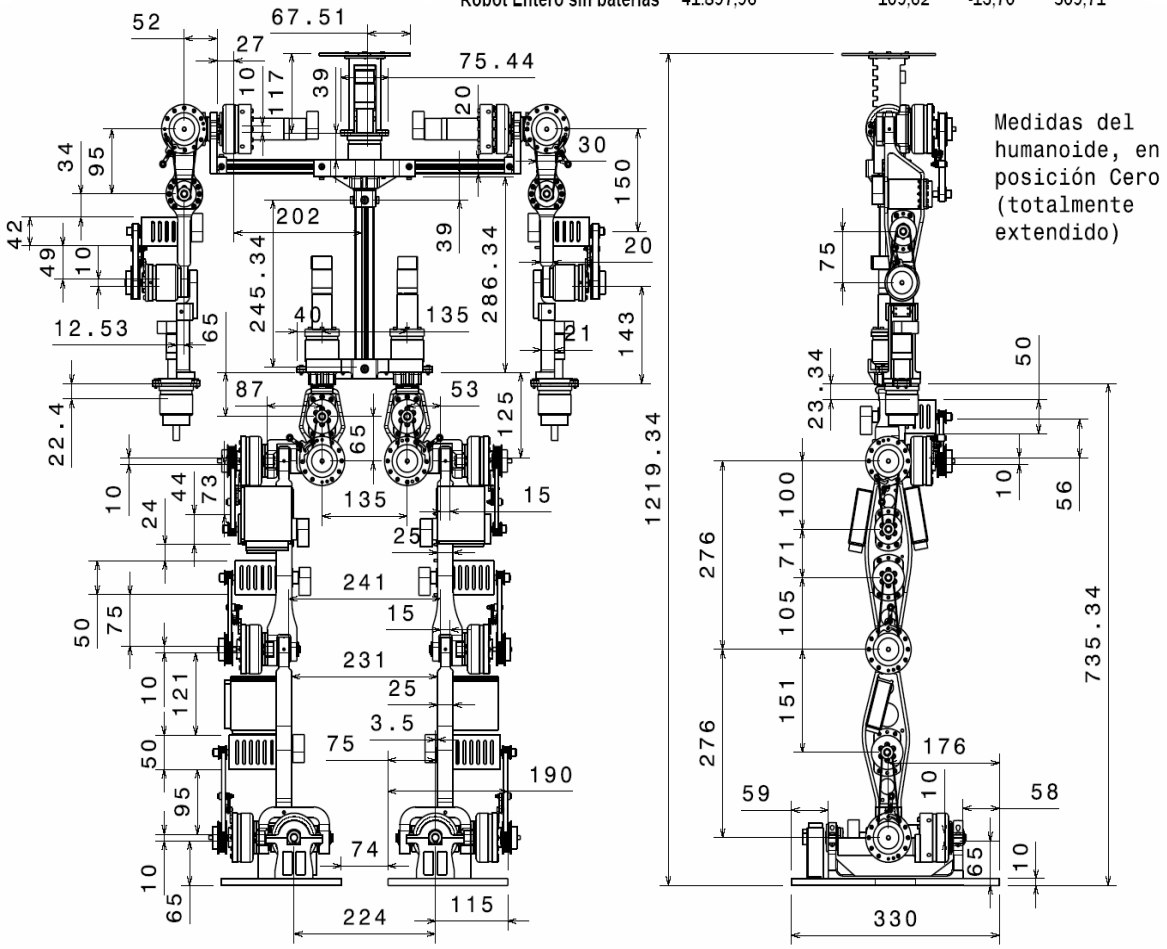


Figura B-1: Medidas de la estructura del Humanoide RH0.

C - Apéndice: Glosario.

Este apéndice contiene definiciones de términos que pueden ser de utilidad en la lectura de esta tesis, sobre todo para quien no se encuentre muy introducido en el lenguaje usado en la investigación de la Locomoción y Navegación de robots Humanoides.

Algoritmo Completo: Un algoritmo es completo para un problema, si está garantizado que para cualquier planteamiento del problema, se encontrará una solución en el caso de que ésta exista.

Altura de Paso: La distancia entre el punto más elevado alcanzado por el pie móvil a lo largo del desarrollo del paso y la base de soporte.

Anchura de Paso: La distancia entre el eje de dirección y el eje de avance del pie móvil.

Área de soporte: En locomoción, es el área interior al contorno formado por el patrón de soporte.

Área de soporte del Pie Fijo: La definida por el patrón de soporte dado por un círculo situado en el centro de la base del pie fijo, cuya superficie está totalmente contenida en la planta del pie fijo.

Área de soporte del Pie Móvil: La definida por el patrón de soporte dado por un círculo situado en el centro de la base del pie móvil, cuya superficie está totalmente contenida en la planta del pie móvil.

Cadencia: Para locomoción, es el número de pasos en un intervalo de tiempo estándar definido.

Configuración: Es la especificación de la posición de cada punto en un objeto relativa a un sistema de referencia fijo. Para un robot articulado, la configuración viene dada por todos los valores de los grados de libertad de las articulaciones.

C-Space: Es el espacio matemático de dimensión n de las configuraciones del robot, que viene dado por el número n de grados de libertad correspondientes a las articulaciones.

Diagrama del Paso: Ilustra las etapas por las que pasan las piernas del humanoide en función del tiempo.

Eje de avance del Pie Móvil: El definido por una línea paralela al eje de dirección, que se encuentra a una distancia igual a la anchura de paso, por el lado correspondiente al pie móvil.

Eje de Dirección: El definido por la línea que pasa por la proyección del centro de masas y el próximo objetivo local, antes de iniciarse el movimiento de locomoción bípeda.

Eje Longitudinal: El eje que contenido en el plano sagital, va desde la parte posterior a la anterior del humanoide.

Estabilidad Dinámica de la Postura: Se produce cuando existe una región cerrada para la trayectoria del movimiento generalizado del humanoide (i.e., posición

y orientación del robot), tal que tras una perturbación interna a esa región, el movimiento generalizado vuelve a la misma trayectoria eje que contenido en el plano sagital, va desde la parte posterior a la anterior del humanoide.

Estabilidad Dinámica de la Trayectoria: Se consigue cuando la velocidad de locomoción media del humanoide, vuelve a su dirección y magnitud original, tras una perturbación.

Factor de Función: Describe el porcentaje de tiempo, en que una pierna se encuentra en la etapa de soporte del ciclo del paso. En el caminar de los humanos, por ejemplo, el factor de función es menor del 20%.

Longitud de Paso: Es la distancia entre el mismo punto de cada pie durante la etapa de soporte, cuando el sistema se encuentra en movimiento.

Longitud de Zancada: Es la distancia entre dos apoyos consecutivos del mismo pie. Está compuesta por la longitud de un paso izquierdo más la de un paso derecho, de forma que es igual, normalmente, al doble de la longitud de paso. Es igual a la distancia que se traslada el tronco del humanoide en una zancada.

Mapa de proyección: Es una estructura de datos para almacenar la información de un entorno físico o imagen.

Margen de Estabilidad Dinámico: Para un determinado momento, es la distancia más corta entre la proyección del centro de presiones y el patrón de soporte. Es una estructura de datos para almacenar la información de un entorno físico o imagen.

Margen de Estabilidad Estática: Para un determinado momento, es la distancia más corta entre la proyección del centro de masas y el patrón de soporte.

Margen de Estabilidad Longitudinal del Paso: Es la mínima distancia entre la proyección del centro de masas y el patrón de soporte, medida en la dirección del eje longitudinal.

Número de Froude: Número adimensional que sirve para determinar cómo es el tipo de locomoción que se está produciendo, esto es, sirve por ejemplo para diferenciar una locomoción que camina de una que corre. Se pueden ver modos de utilización en Alexander [3].

Pasillo de dirección: Superficie que el humanoide puede alcanzar mediante locomoción bípeda, manteniendo la presente dirección de movimiento, bien sea con sentido de avance o de retroceso. Esto es aproximadamente igual al área interior a las líneas paralelas al eje longitudinal del humanoide, situadas a una distancia del mismo igual a la anchura de paso.

Paso: Es el avance de una pierna.

Patrón de Soporte: Es la envolvente convexa de la proyección de los elementos de soporte, sobre el plano a apoyo.

Plano Sagital: Denota el plano que divide al humanoide bilateralmente en dos mitades iguales, izquierda y derecha.

Posición Extrema Posterior: La posición de la pierna del robot que se produce en la transición de elevación.

Posición Extrema Anterior: La posición de la pierna del robot que se produce en la transición de colisión.

Proyección del Centro de Masas: Es el punto obtenido al proyectar verticalmente e centro de masas sobre la superficie de soporte, (frecuentemente el plano horizontal.)

Proyección del Centro de Presión: Es el punto obtenido al proyectar el centro de masas en la dirección de la resultante de fuerzas del sistema, sobre la superficie de soporte, (frecuentemente el plano horizontal.)

Punto de Momento Cero: Es el punto sobre la superficie de soporte del humanoide, donde se anula el momento resultante de las fuerzas externas, la gravedad y las fuerzas de inercia.

Región Inocupable: Es una parte del espacio físico con la que el robot nunca podría ínter seccionar, bien sea debido a la presencia de obstáculos o por la propia cinemática del robot.

Sistema de referencia del Suelo: Es el sistema coordenado de referencia inercial con respecto al suelo.

Sistema de referencia del tronco: Es el sistema coordenado de referencia del tronco del humanoide, normalmente, situado en el centro de masas del tronco.

***Twist*:** Matriz $\xi\hat{}$ que identifica el álgebra de Lie asociada con el grupo especial Euclídeo.

Velocidad de Locomoción: Es la longitud de la zancada, dividida por la duración de la misma.

Velocidad Relativa del Pie: Es la velocidad del pie con respecto al tronco del humanoide; que está limitada por el diseño electromecánico del robot.

Zancada: Es el paso consecutivo de las dos piernas.

D - Apéndice: Librería de Software ROBOTMAN.

Este apéndice contiene las funciones incluidas en la librería de software ***RobotMan*** *(Robot Manipulation) que se ha creado para los trabajos de esta tesis. La implementación ha sido realizada como una Toolbox de* ***MATLAB****, aunque ésta puede usarse como una guía para desarrollos en otros lenguajes de programación, dado que todo el código fuente está disponible, tanto el de* ***RobotMan*** *como el de las funciones de* ***MATLAB*** *utilizadas. Por consistencia e integración con otros trabajos, la librería se ha escrito en lengua inglesa (nombres de las funciones y comentarios). Fundamentalmente, RobotMan facilita el análisis de la cinemática y dinámica de robots mediante fórmulas de la teoría de Grupos de Lie y la geometría de Screws en* $\boldsymbol{R^3}$.

{RobotMan-f(1): bodyjacobian

```
% "bodyjacobian" computes the BODY MANIPULATOR JACOBIANn for a robot of any number of
links.
% Use in SE(3).
%
%      jst = bodyjacobian([x1...xn],[t1...tn],gst0)
%
% BODY MANIPULATOR JACOBIAN: At each configuration of theta, maps the joint velocity
vector, into the
% corresponding velocity of the end effector.
% The "ith" column of jst is the "ith" joint twist, written with respect to the tool
frame at to the current
% manipulator configuration.
%
% The pairs {xi,ti} define the joint twist "xi" and joint angle "ti" (or displacemnet)
for each joint of the manipulator.
% xi is a vector 6x1, ti is a value, so the [x1...xn] is a matrix 6xn and [t1...tn] is a
transpose vector 1xn.
% jst is a matrix 6xn.
%
% "gst0" = gst(0), is th location of the tool frame "T" at the reference configuration.
% "gst" and "gst0" are homogeneous matrix 4x4.
% gst(theta)=exp(E1^theta1)*...*exp(En^thetan)*gst(0)
%
%   b          |v1'' v2'' ... vn''|
% Jst(theta) = |                  |
%              |w1'' w2'' ... wn''|
%            |vi''|
% With: Ei''=|    |=Ad^-1                                        *Ei
%            |wi''|    (exp(Ei^thetai)*...*exp(En^thetan)*gst0)
%    s            b
% Jst(t)=Adg(t)*Jst(t) ; and Adg(t) is the Adjoint transformation of gst(theta).
%
% See also: spatialjacobian, forwardkinematics, twistexp, rigidtwist, rididadjoint.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function jst = bodyjacobian(x,t,g0)

    jst = [];
    gst = g0;
    for i = size(x,2):-1:1,
        gst = twistexp(x(:,i),t(i))*gst;
        xip = inv(rigidadjoint(gst))*x(:,i);
        jst = [xip jst];
    end
```

{RobotMan-f(2): BodyGenCoor2homogeneous

```
% "BodyGenCoor2homogeneous" transforms configurations, this is, general body coordinates
to homogeneous matrix representation.
% Use in SE(3).
%
%       gst=BodyGenCoor2homogeneous(goal)
%
%*********************************************************************

% goal = matrix(1:6) for the x,y,z translation and x,y,z rotation, Body General
Coordinates, which correspond with the input.

% gst = homogeneous transformation corresponding with the previous inputs.

% Jose M. Pardos

% 20030630.

%*********************************************************************

%

% "gst" defines the desired configuration for the tool, this is, the goal.

% object_pos(x,y,z) define the desired position for the origin of the Tool CS on respect
of the fixed frame S.

% object_rot define the Roll-Pitch-Yaw (respectively x-y-z) angles of rotation on the
fixed frame S, for the origin of the Tool CS.
%
% See also: homogeneous2BodyGenCoor.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function gst = BodyGenCoor2homogeneous(goal)
       gst_rot_x = [1 0 0; 0 cos(goal(4)) -sin(goal(4)); 0 sin(goal(4)) cos(goal(4))];
       gst_rot_y = [cos(goal(5)) 0 sin(goal(5)); 0 1 0; -sin(goal(5)) 0 cos(goal(5))];
       gst_rot_z = [cos(goal(6)) -sin(goal(6)) 0; sin(goal(6)) cos(goal(6)) 0; 0 0 1];
       gst = rptohomogeneous(gst_rot_x*gst_rot_y*gst_rot_z,[goal(1);goal(2);goal(3)]);
return
```

{RobotMan-f(3): forwardkinematics

```
% "forwardkinematics" computes the forward kinematics via the product of exponentials
formula.
% Use in SE(3).
%
%      gst = forwardkinematics([x1...xn],[t1...tn],gst0)
%
% PRODUCT OF EXPONENTIALS FORMULA FOR THE MANIPULATOR FORWARD KINEMATICS - RELATIVE
MOTION OF A RIGID BODY.
% combining the individual joint motions, the map gst:Q -> SE(3).
% This transformation is different from other rigid transformations. We interpret it not
as mapping points from one
% coordinate frame to another, but rather as mapping points from initial coordinates, to
theirs after the rigid motion is applied.
% The reference configuration is firstly moved by the twist xn, secondly by the twist
xn-1 and so on till the last twist applied x1.
%
% The pairs {xi,ti} define the joint twist "xi" and joint angle "ti" (or displacemnet)
for each joint of the manipulator.
% xi is a vector 6x1, ti is a value, so the [x1...xn] is a matrix 6xn and [t1...tn] is a
transpose vector 1xn.
% "gst0" = gst(0), is th location of the tool frame "T" at the reference configuration.
% "gst" and "gst0" are homogeneous matrix 4x4.
%
% gst(theta)=exp(E1^theta1)*...*exp(En^thetan)*gst(0)
% gst(t)=exp(x1^t1)*exp(x2^t2)*...*exp(xn^tn)*gst(0)
%
% With:
%        |v|
% E = xi=| |
%        |W|
%              |exp(W^Th)  (I-exp(W^Th))*(W x v)+W*W'*v*Theta|
% exp(E^Theta)=|                                              | = exp(x^t)
%              |0                          1                 |
% exp(W^Th)=I + W^ * sin(Th)+ W^ * W^ * (1-cos(Th))
% With W^=skew(W)
%
% See also: twistexp, rigidtwist.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function gst = forwardkinematics(x,t,gst0)

    gst = twistexp(x(:,1),t(1));
    for i = 2:size(x,2),
        gst = gst * twistexp(x(:,i),t(i));
    end
    gst = gst * gst0;
```

{RobotMan-f(4): homogeneous2BodyGenCoor

```
% "homogeneous2BodyGenCoor" transforms matrix representation to configurations, this is,
% general body coordinates to homogeneous.
%
% Use in SE(3).
%
%      gst=homogeneous2BodyGenCoor(goal)
%
%*************************************************************************
%  goal  =  matrix(1:6)  for  the  x,y,z  translation  and  x,y,z  rotation,  Body  General
Coordinates, which correspond with the input.
% gst = homogeneous transformation corresponding with the previous inputs.
%*************************************************************************

% "gst" defines the desired configuration for the tool, this is, the goal.
% object_pos(x,y,z) define the desired position for the origin of the Tool CS on respect
of the fixed frame S.
% object_rot define the Roll-Pitch-Yaw (respectively x-y-z) angles of rotation on the
fixed frame S, for the origin of the Tool CS.
%
% See also: BodyGenCoor2homogeneous.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function gst = homogeneous2BodyGenCoor(goal)
e1 = prismatictwist([0;0;0],[1;0;0]); % Actually the "q" does not matter for prismatic
twist.
e2 = prismatictwist([0;0;0],[0;1;0]); % Actually the "q" does not matter for prismatic
twist.
e3 = prismatictwist([0;0;0],[0;0;1]); % Actually the "q" does not matter for prismatic
twist.
e4 = revolutetwist([0;0;0],[1;0;0]);
e5 = revolutetwist([0;0;0],[0;1;0]);
e6 = revolutetwist([0;0;0],[0;0;1]);
e = [e1 e2 e3 e4 e5 e6];
gst0 = RPToHomogeneous(eye(3),[0;0;0]);
t11 = gst(1,4);
t21 = gst(2,4);
t31 = gst(3,4);
gs=inv(twistexp(e(:,3),t31))*inv(twistexp(e(:,2),t21))*inv(twistexp(e(:,1),t11))*gst*inv
(gst0);
%
% By PADEN-KAHAN-TWO.
th= [0; 0; 1; 1];
t = padenkahantwo(e(:,4),e(:,5),th,gs*th,[0; 0; 0; 1]);
t41 = t(1,1);
t51 = t(2,1);
%
% By PADEN-KAHAN-ONE.
sh = [1; 0; 0; ;1];
t61=padenkahanone(e(:,6),sh,(inv(twistexp(e(:,5),t51))*inv(twistexp(e(:,4),t41))*gs)*sh)
;
goal = [t11 t21 t31 t41 t51 t61];
```

{RobotMan-f(5): homogeneoustopoint

```
% "homogeneoustopoint" Convert a point "ph" 4x1 in homogeneous coordinates to "p" 3x1.
% Use in SE(3).
%
%      p = homogeneoustopoint(ph)
%
% Returns a point Euclidean coordinates, vector 3x1.
%    |p1|
% p =|p2|
%    |p3|
% Con
%
% See also: .
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function p = homogeneoustopoint(ph)

       p = [ph(1,1);ph(2,1);ph(3,1)];
```

{RobotMan-f(6): homogeneousto*twist*

```
% "homogeneoustotwist" Convert a matrix "E^" 4x4 into a twist "xi" 6x1.
% Use in SE(3).
%
%      xi = homogeneoustotwist(H)
%
% Returns a twixt "xi" from a homogeneous "h" matrix 4x4.
%    |v|         |W^ v|v
% xi=| |  <=  E^=|    |
%    |W|         |0  0|
% Con W^=skew(W)
%
% See also: vee.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function r = homogeneoustotwist(h)

       s = unskew([h(1,1) h(1,2) h(1,3);
       h(2,1) h(2,2) h(2,3);
       h(3,1) h(3,2) h(3,3)]);
       r = [h(1,4);h(2,4);h(3,4);s(1,1);s(2,1);s(3,1)];
```

{RobotMan-f(7): linkinertia

```
% "linkinertia" computes the LINK INERTIA MATRIX for a robot's link.
% Use in SE(3).
%
%      im = linkinertia([x1...xn],[t1...tn],gsi0,m,IT)
%
% It gives the LINK INERTIA MATRIX corresponding to the Lagangian's equations of the
dynamics of an Open chain manipulator.
% Beware that this function does compute ONLY the inertia matrix of a link, thus for
getting the total manipulator Inertia
% matrix, you need to add all links inertia matrices.
%
% The pairs {xi,ti} define the joint twist "xi" and angle "ti" for each joint of the
manipulator which affects to the link.
% xi is a vector 6x1, ti is a value, so the [x1...xn] is a matrix 6xn and [t1...tn] is a
transpose vector 1xn.
% IT is the Inertia Tensor for the link 3x3. m is a the mass of the link. the Inertia
matrix "im" is 6xn.
%
% "gsi0" = gsi(0), is th location of the link's center of mass at the reference
configuration.
% "gsi" and "gsi0" are homogeneous matrix 4x4.
% gsi(theta)=exp(E1^theta1)*...*exp(En^thetan)*gsi(0)
%               |mI  0|
% im = Jsib(th)'*|     |*Jsib(th)=Jstb(th)'*M*Jsib(th)= Link Inertia Matrix
%               |0  IT|
% with M Generalized Inertia Matrix, defined through the diagonal mass and the inertia
tensor.
% and Jsib as the manipulator jacobian on the body coordinate frame.
%
% See also: spatialjacobian, forwardkinematics, twistexp, bodyjacobian, rididadjoint.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function im = linkinertia(x,t,g0,m,it)

    gim = [m*eye(3), zeros(3); zeros(3), it];
    jsib = bodyjacobian(x,t,g0);
    im = jsib'*gim*jsib;
```

{RobotMan-f(8): linkinertiasum

```
% "linkinertiasum" computes the MANIPULATOR INERTIA MATRIX for the sum of two open chain
links.
% Use in SE(3).
%
%      im = linkinertiasum(im1,im2)
%
% It gives the MANIPULATOR INERTIA MATRIX (MIM) corresponding to the Lagangian's
equations of the dynamics of the robot
% formed by this two links. im1 and im2 are the LINK INERTIA MATRIX corresponding to
each link.
% Be awere that you need to repeat this funtion by as many links as the robot has for
getting the total MIM of a robot.
%
%                            |mI  0|
% IM = Sum(i=1,2)[Jsib(th)'*|     |*Jsib(th)] = im1+im2
%                            |0  IT|
% Be careful, because the inertia matrix of each link can have a different dimension.
Therefore, you must equal
% the size of the links im1 and im2 before to add them.
%
% See also: linkinertia, bodyjacobian.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function im = linkinertiasum(im1,im2)

    im = zeros(max(size(im1,1),size(im2,1)),max(size(im1,2),size(im2,2)));
    for i = 1:size(im1,1)
        for j = 1:size(im1,2)
        im(i,j) = im(i,j) + im1(i,j);
        end
    end
    for i = 1:size(im2,1)
        for j = 1:size(im2,2)
        im(i,j) = im(i,j) + im2(i,j);
        end
    end
```

{RobotMan-f(9): linkinertiatrans

```
% "linkinertiatrans" computes the LINK TRANSFORMED INERTIA MATRIX for a robot's link.
% Use in SE(3).
%
%      im = linkinertiatrans(gsi0,m,IT)
%
% It gives the LINK TRANSFORMED INERTIA MATRIX corresponding to the inertia of the link
into the base frame of the manipulator.
%
% IT is the Inertia Tensor for the link 3x3. m is a the mass of the link. the Inertia
matrix "im" is 6x6.
%
% "gsi0" = gsi(0), is th location of the link's center of mass at the reference
configuration.
% "gsi" and "gsi0" are homogeneous matrix 4x4.
% Adgsli0 = Ad(gsli0^-1)= Adjoint transformation of the inverse of the center of mass
ref config.
%               |mI  0|
% im = Adgsli0' *|     |* Adgsli0 = Adgsli0' * M * Adgsli0 = Link Transformed Inertia
Matrix.
%               |0  IT|
% with M Generalized Inertia Matrix, defined through the diagonal mass and the inertia
tensor.
%
% See also: linkinertia, rigidadjoint.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function im = linkinertiatrans(g0,m,it)

    gim = [m*eye(3), zeros(3); zeros(3), it];
    adi0 = rigidadjoint(inv(g0));
    im = adi0'*gim*adi0;
```

{RobotMan-f(10): manipulatoradjoint

```
% "manipulatoradjoint" Computes the ADJOINT TRANSFORMATION for a list of twists-
magnitudes.
% Use in SE(3).
%
%      A = manipulatoradjoint(x,t)
%
% ADJOINT TRANSFORMATION: This is a special notation which gives us a most convenient
form of the Adjoint
% of an open chain manipulator. We use this notation for an easy calculation of the
Manipulator Inertia Matrix and
% the Manipulator Coriolis Matrix.
% x is the list of twists 6xn.
% t is the list of magnitudes of the twists Theta 1xn.
%      I                                    if i=j
% Aij= Ad^-1[(exp(Ej+1,Tj+1)...(exp(Ei,Ti)] if i>j
%      0                                    if i<j
%
% With Aij being an element 6x6 of the Manipulator adjoint transformation A 6x6xixj
(four dimensions).
% With Ad the adjoint transformation "rigidadjoint". Maps twist vectors to twist
vectors. Compute Ad in R^6 from homogeneous 4x4.
%
% See also: manipulatorinertia, manipulatorcoriolis, rigidadjoint.
%
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function a = manipulatoradjoint(x,t)

    n = size(x,2);
    ax = zeros(4,4,n,n);
    ay = eye(6,6);
    a = zeros(6,6,n,n);
    if n > 1
        for i=2:n
            ax(:,:,i,i-1)=twistexp(x(:,i),t(i));
        end
    end
    if n > 2
        for i=n:-1:3
            for j=i-2:-1:1
            ax(:,:,i,j)=ax(:,:,j+1,j)*ax(:,:,i,j+1);
            end
        end
    end
    for i=1:n
        a(:,:,i,i)=ay;
    end
    if n > 1
        for i=2:n
            for j=1:i-1
            a(:,:,i,j)=inv(rigidadjoint(ax(:,:,i,j)));
            end
        end
    end
```

{RobotMan-f(11): manipulatorcoriolis

```
% "manipulatorcoriolis" computes the MANIPULATOR CORIOLIS MATRIX for an open chain
manipulator.
% Use in SE(3).
%      MC = manipulatorcoriolis(x, t, dt, g0, m, it)
% Gives the MANIPULATOR CORIOLIS MATRIX (C) corresponding to the Lagangian's equations:
% M(t)*ddt + C(t,dt)*dt + N(t,dt) = T
% of the dynamics of the robot formed by links on an open chain.
% "x" is the matrix of the twists which affect the links of the manipulator 6xn.
% "t" is the matrix of the magnitudes "theta" of the respective twists 1xn.
% "dt" is the matrix of the velocities "derive theta" of the respective twists 1xn.
% "g0" is the matrix of the homogeneous transformations which represent the links'
center of masses' locations,
% at the reference configuration 4x4xn (three dimension matrix).
% "m" is the matrix of the masses of each link of the manipulator 1xn.
% "it" is the matrix of the inertia tensors of the links 3x3xn (three dimension matrix).
% with a number of links n.
%
%     |C11...C1n|
% C = |         |, With Cij = 1/2 * Sum(1,n)[( dMij/dtk + dMik/dtj - dMkj/dti ) * dtk]
%     |Cn1...Cnn|
% where:
%  dMij/dtk  =  Sum(l=max(i,j),n)[[Ak-1i*Ei,Ek]'*Alk'*Ml*Alj*Ej  +  Ei'*Ali'*Ml*Alk*[Ak-
1j*Ej,Ek]]
% and so on and so forth.
% With Ml being the link inertia transformed matrix of the link l 6x6, incapsulated into
a three dimensions matrix 6x6xl.
% With Ei being the twist xi 6x1.
% With Aij being an element 6x6 of the Manipulator adjoint transformation A 6x6xixj
(four dimensions).
%
% See also: manipulatorinertia, manipulatoradjoint, linkinertiatrans, linkinertia.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
%
% $Revision: 1.1 $

function cm = manipulatorcoriolis(x, t, dt, g0, m, it)

        ma = manipulatoradjoint(x,t);
        n = size(x,2);
        mt = zeros(6,6,n);
        mc = zeros(n,n);
        for i=1:n
            mt(:,:,i) = linkinertiatrans(g0(:,:,i),m(i),it(:,:,i));
        end
        for i=1:n
            for j=1:n
                for k=1:n
                    a = 0;
                    b = 0;
                    c = 0;
                    for p=max([i j]):n
a=a+twistbracket(ma(:,:,k,i)*x(:,i),x(:,k))'*ma(:,:,p,k)'*mt(:,:,p)*ma(:,:,p,j)*x(:,j);
a=a+x(:,i)'*ma(:,:,p,i)'*mt(:,:,p)*ma(:,:,p,k)*twistbracket(ma(:,:,k,j)*x(:,j),x(:,k));
end
                    for p=max([i k]):n
b=b+twistbracket(ma(:,:,j,i)*x(:,i),x(:,j))'*ma(:,:,p,j)'*mt(:,:,p)*ma(:,:,p,k)*x(:,k);
b=b+x(:,i)'*ma(:,:,p,i)'*mt(:,:,p)*ma(:,:,p,j)*twistbracket(ma(:,:,j,k)*x(:,k),x(:,j));
end
                    for p=max([k j]):n
c=c+twistbracket(ma(:,:,i,k)*x(:,k),x(:,i))'*ma(:,:,p,i)'*mt(:,:,p)*ma(:,:,p,j)*x(:,j);
c=c+x(:,k)'*ma(:,:,p,k)'*mt(:,:,p)*ma(:,:,p,i)*twistbracket(ma(:,:,i,j)*x(:,j),x(:,i));
end
                    mc(i,j) = mc(i,j) + (a+b-c)*dt(k);
                end
            end
        end
        cm = mc * 0.5;
```

{RobotMan-f(12): manipulatorinertia

```
% "manipulatorinertia" computes the MANIPULATOR INERTIA MATRIX for an open chain
manipulator.
% Use in SE(3).
%
%      MI = manipulatorinertia(x, t, g0, m, it)
%
% Gives the MANIPULATOR INERTIA MATRIX (M) corresponding to the Lagangian's equations:
% M(t)*ddt + C(t,dt)*dt + N(t,dt) = T
% of the dynamics of the robot formed by links on an open chain.
%
% "x" is the matrix of the twists which affect the links of the manipulator 6xn.
% "t" is the matrix of the magnitudes "theta" of the respective twists 1xn.
% "g0" is the matrix of the homogeneous transformations which represent the links'
center of masses' locations,
% at the reference configuration 4x4xn (three dimension matrix).
% "m" is the matrix of the masses of each link of the manipulator 1xn.
% "it" is the matrix of the inertia tensors of the links 3x3xn (three dimension matrix).
% with a number of links n.
%      |M11...M1n|
% MI = |         |, With Mij = Sum(l=max(i,j),n)[Ei'*Ali'*Ml*Alj*Ej
%      |Mn1...Mnn|
%
% With Ml being the link inertia transformed matrix of the link l 6x6, incapsulated into
a three dimensions matrix 6x6xl.
% With Ei being the twist xi 6x1.
% With Aij being an element 6x6 of the Manipulator adjoint transformation A 6x6xixj
(four dimensions).
%
%   See   also:   manipulatoradjoint,   linkinertiatrans,   linkinertia,   bodyjacobian,
linkinertiasum.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function mi = manipulatorinertia(x, t, g0, m, it)

        a = manipulatoradjoint(x,t);
        n = size(x,2);
        mt = zeros(6,6,n);
        mi = zeros(n,n);
        for i=1:n
            mt(:,:,i) = linkinertiatrans(g0(:,:,i),m(i),it(:,:,i));
        end
        for i=1:n
            for j=1:n
                for k=max(i,j):n
                    mi(i,j) = mi(i,j)+x(:,i)'*a(:,:,k,i)'*mt(:,:,k)*a(:,:,k,j)*x(:,j);
                end
            end
        end
```

{RobotMan-f(13): padenkahanone

```
% "padenkahanone" Find the ROTATION ABOUT A SINGLE AXIS of a twist on R^3.
% Use in SE(3).
%
%      theta = padenkahanone(xi, p, q)
%
% Compute the angle "Theta" of the twist xi, to move the point "p" from its original
position to the point "q".
% The points "p" and "q" are on the same plane perpendicular to the axis of the twist
xi.
%         |v|
% E = xi =| | 6x1; and the points p and q are 3x1 coordinates.
%         |w|
% exp(E^Theta) * p = q
% Based on the work of Paden & Kahan subproblems for INVERSE KINEMATICS.
% The problem could have two solutions t1=th and t2=2pi-t1, but the function gives only
the the first theta t=t1.
%
%   See   also:   padenkahantwo,   padenkahanthree,   twistaxispoint,   twistaxisdirection,
twistintersection.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function t = padenkahanone(x,p,q)

    p = [p(1,1); p(2,1); p(3,1)];
    q = [q(1,1); q(2,1); q(3,1)];
    r = twistaxispoint(x);
    w = twistaxisdirection(x);
    u = (p-r)-w*w'*(p-r);
    v = (q-r)-w*w'*(q-r);
    t = real(atan2(w'*(cross(u,v)),u'*v));
```

{RobotMan-f(14): padenkahanthree

```
% "padenkahanthree" Find the ROTATION TO A GIVEN DISTANCE of a twist on R^3.
% Use in SE(3).
%
%      theta = padenkahanthree(xi, p, q, d)
%
% Compute the angle "Theta" of the twist xi, to move the point "p" from its original
position to the point "c" or "e",
% complying that the distance between "c" or "e" to "q" is "d".
% The points "p", "c" and "e" are on the same plane perpendicular to the axis of the
twist xi.
%         |v|
% E = xi =| | 6x1; the points p, q, c, e are 3x1 and d is R.
%         |w|
% exp(E^Theta) * p = c or e; and d=norm(c-q) and d=norm(e-q).
% Based on the work of Paden & Kahan subproblems for INVERSE KINEMATICS.
% The problem could have zero, one or two solutions. We do not consider the alternative
solutions t1=th and t2=2pi-th.
% To go from "p" to "c" we use t1 and to go from "p" to "e" we use t2. The function
gives two results theta = [t1 t2]
%
% See also: twistaxispoint, twistaxisdirection, padenkahanone, padenkahantwo.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function t = padenkahanthree(x,p,q,d)

    p = [p(1,1); p(2,1); p(3,1)];
    q = [q(1,1); q(2,1); q(3,1)];
    r = twistaxispoint(x);
    w = twistaxisdirection(x);
    u = (p-r)-w*w'*(p-r);
    v = (q-r)-w*w'*(q-r);
    t0 = atan2(w'*(cross(u,v)),u'*v);
    dp = sqrt(d^2-(w'*(p-q))^2);
    t1 = t0 - acos((norm(u)^2+norm(v)^2-dp^2)/(2*norm(u)*norm(v)));
    t2 = t0 + acos((norm(u)^2+norm(v)^2-dp^2)/(2*norm(u)*norm(v)));
    t = real([t1 t2]);
```

{RobotMan-f(15): padenkahantwo

```
% "padenkahantwo" Find the ROTATION ABOUT TWO SUBSEQUENT AXIS of two twist on R^3.
% Use in SE(3).
%
%      theta = padenkahantwo(x1, x2, p, q, r)
%
% Compute the angles "Th2" and "Th1" of two subsequently applies twists x2 and x1, to
move the point "p" to the point "q".
% Beware that the first twist applied is the x2 and subsequently x1.
% The points "p" is first moved to the point "c" or "d" according with Paden Kahan
subproblem one.
% and then from "c" o "d" is moved to "q" according again, with padenkahanone.
%
% exp(E1^Theta1) * exp(E2^Theta2) * p = q
%          |v|
% Ei = xi =| | is 6x1 ; and p, q, r are points 3x1.
%          |w|
% Based on the work of Paden & Kahan subproblems for INVERSE KINEMATICS.
%  "r"  is  the  intersection  of  both  twist  axis.  It  could  be  calculated  with
"twistintersection" or given explicitely.
% The problem could have 0, 1 (c=d) or 2 solutions as a combination of the possible
paths from p-c-q or p-d-q.
% Two possible paths between points t=th and t'=2pi-th, but we do not consider the
second as solutions.
% The function gives only the first results as pairs (e.g. theta2=t21 theta1=t11). Theta
= [t11 t12; t21 t22] 2x2.
%
% See also: padenkahanone, twistaxispoint, twistaxisdirection, twistintersection.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function t = padenkahantwo(x1,x2,p,q,r)

    p = [p(1,1); p(2,1); p(3,1)];
    q = [q(1,1); q(2,1); q(3,1)];
    r = [r(1,1); r(2,1); r(3,1)];
    w1 = twistaxisdirection(x1);
    w2 = twistaxisdirection(x2);
    u = p-r;
    v = q-r;
    a = ((w1'*w2)*w2'*u-w1'*v)/((w1'*w2)^2-1);
    b = ((w1'*w2)*w1'*v-w2'*u)/((w1'*w2)^2-1);
    m = sqrt((norm(u)^2-a^2-b^2-2*a*b*w1'*w2)/norm(cross(w1,w2))^2);
    c = r + a*w1+b*w2+m*cross(w1,w2);
    d = r + a*w1+b*w2-m*cross(w1,w2);
    t21 = padenkahanone(x2,p,c);
    t22 = padenkahanone(x2,p,d);
    t11 = padenkahanone(x1,c,q);
    t12 = padenkahanone(x1,d,q);
    t = real([t11 t12; t21 t22]);
```

{RobotMan-f(16): pardosone

```
% "pardosone" Find the TRANSLATION TO A GIVEN DISTANCE of a twist on R^3.
% Use in SE(3).
%
%      theta = pardosone(xi, p, q, d)
%
% Compute the magnitude (translation) "Theta" of the twist xi, to move the point "p"
from its original position
% to the point "c" or "e" complying that the distance between "c" or "e" to "q" is "d".
% The points "p", "c" and "e" are on the same line, parallel to the axis of the twist
xi.
%         |v|
% E = xi =| | 6x1; the points p, q, c, e can be 3x1 or 4x1 (homogeneous) and d is R.
%         |w|
% exp(E^Theta) * p = c or e; and d=norm(c-q) and d=norm(e-q).
%
% Based on the work of Paden & Kahan subproblems for INVERSE KINEMATICS.
%  This is an adaptation of the Paden-Kahan subproblem three (rotation to a given
distance) applied for revolute joints.
% I tried to adapt the same scheme for prismatic joints. The problem could have zero,
one or two solutions.
% To go from "p" to "c" we use t1 and to go from "p" to "e" we use t2. The function
gives two results theta = [t1 t2]
%
% See also: twistaxispoint, twistaxisdirection, padenkahanthree.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function t = pardosone(x,p,q,d)

    p = [p(1,1); p(2,1); p(3,1)];
    q = [q(1,1); q(2,1); q(3,1)];
    r = twistaxispoint(x);
    w = twistaxisdirection(x);
    st = sign(w'*(q-r)-w'*(p-r)); % determines the sign of the magnitude of the twist.
    pq = p-q;
    pqwn = norm(w'*pq);
    c = abs(sqrt(d^2-norm(pq)^2+pqwn^2));
    t1 = st*pqwn+c;
    t2 = st*pqwn-c;
    t = real([t1 t2]);
```

{RobotMan-f(17): pointtohomogeneous

```
% "pointtohomogeneous" Convert a point "p" 3x1 into homogeneous coordinates "ph" 4x1.
% Use in SE(3).
%
%      ph = pointtohomogeneous(p)
%
% Returns a point in homogeneous coordinate, vector 4x1.
%    |p|
% ph=| |
%    |1|
% Con
%
% See also: .
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function ph = pointtohomogeneous(p)

       ph = [p(1,1);p(2,1);p(3,1);1];
```

{RobotMan-f(18): prismatictwist

```
% "prismatictwist" Construct a twist corresponding to a prismatic joint in the direction
"w" going through the point "q".
% Use in SE(3).
%
%       xi = prismatictwist(q, w)
%
% PrismaticTwist[q,w] gives the 6-vector corresponding to point q on the axis and a
screw with axis w for a prismatic joint
%
% Return the twist "xi" 6x1 coordinates of a screw with pitch "h" through the point "q"
and in the direction "w".
% h = inf, because a pure translation is associated with the prismatic joint.
%     |v|  |    w    |
% xi =| | =|         |: if h = inf, this is the case for pure translation.
%     |w|  |    0    |
%
% See also: screwtotwist, revolutetwist, twistpitch, twistaxis, twistmagnitude.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function x = prismatictwist(q, w)

    x = screwtwist(inf,q,w);
```

{RobotMan-f(19): revolute*twist*

```
% "revolute*twist*" Construct a *twist* corresponding to a revolute joint in the direction
"w" going through the point "q".
% Use in SE(3).
%
%      xi = revolute*twist*(q, w)
%
% Revolute*Twist*[q,w] gives the 6-vector corresponding to point q on the axis, and a
*screw* with axis w for a revolute joint.
%
% Return the *twist* "xi" 6x1 coordinates of a *screw* with *pitch* "h" through the point "q"
and in the direction "w".
% h = 0, because a pure rotation is associated with the revolute joint.
%     |v|  | -w x q  |
% xi =| | =|         |: if h = 0, this is the case for pure rotation.
%     |w|  |    w    |
%
% See also: *screw*to*twist*, *twistpitch*, *twist*axis, *twist*magnitude.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function x = revolute*twist*(q, w)

    x = *screwtwist*(0,q,w);
```

{RobotMan-f(20): rigidadjoint

```
% "rigidadjoint" Find the adjoint matrix associated with g.
% Use in SE(3).
%
%      Adg = rigidadjoint(g)
%
% ADJOINT TRANSFORMATION: transforms twist from one coordinate frame to another.
%  s      b   s           s
% V =Adg*V ; Vac = Adgab*Vbc ; E'=Adg*E
% The adjoint transformation maps twist vectors to twist vectors.
% Compute the Adg in R^6 from the homogeneous matrix g 4x4.
%      |R p^R|      |r p|
% Adg =|     | <= g=|   |
%      |0   R|      |0 1|
% With p^=skew(p)
% With W x v == cross product W by v.
%
% See also: .
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function a = rigidadjoint(g)

    r = rigidorientation(g);
    p = rigidposition(g);
    a = [r skew(p)*r; zeros(3) r];
```

{RobotMan-f(21): rigidangle

```
% "rigidangle" Find the magnitude Theta which generates a rigid motion.
% Use in SE(3).
%
%      Th = rigidangle(g)
%
% Compute the magnitude Theta which generates the homogeneous matrix g 4x4.
%   |r p|
% g=|   |
%   |0 1|
%                |exp(W^Th)  (I-exp(W^Th))*(W x v)+W*W'*v*Theta|
% g= exp(E^Theta)=|                                            |
%                |0                          1                |
% exp(E^Theta)=gst(theta)*gst(0)^-1
% Use Rodrigues's formula:
% exp(W^Th)=I + W^ * sin(Th)+ W^ * W^ * (1-cos(Th))
% With W^=skew(W)
% With W x v == cross product W by v.
%
% Find the angle "magnitude"="Theta" of a rotation matrix. R is the Rotation matrix 3x3.
% Th= arccos[(Trace(R)-1)/2]
% But if there is NO rotation, that is arccos[(Trace(R)-1)/2] = 0 then.
% the "magnitude"="Theta" of the position vector. P is the position vector 3x1.
% Th = norm(p)
%
% See also: .
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function t = rigidangle(g)

    r = rigidorientation(g);
    p = rigidposition(g);
    t = rotationangle(r);
    if t == 0
        t = norm(p);
    end
```

{RobotMan-f(22): rigidorientation

```
% "rigidorientation" Extract the orientation "R" 3x3, portion from a homogeneous
transformation "g" 4x4.
% Use in SE(3).
%
%      r = rigidorientation(g)
%
% Returns a point in homogeneous coordinate, vector 4x1.
%    |r11 r12 r13|      |r p|
% r =|r21 r22 r23| <= g=|   |
%    |r31 r32 r33|      |0 1|
% Con
%
% See also: .
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function r = rigidorientation(g)

     r = [g(1,1) g(1,2) g(1,3);
               g(2,1) g(2,2) g(2,3);
               g(3,1) g(3,2) g(3,3)];
```

{RobotMan-f(23): rigidposition

```
%  "rigidposition"  Extract  the  position  "p"  3x1,  portion  from  a  homogeneous
transformation "g" 4x4.
% Use in SE(3).
%
%      r = rigidposition(g)
%
% Returns a point in homogeneous coordinate, vector 4x1.
%    |p13|      |r p|
% p =|p23| <= g=|   |
%    |p33|      |0 1|
% Con
%
% See also: .
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function p = rigidposition(g)

       p = [g(1,4); g(2,4); g(3,4)];
```

{RobotMan-f(24): rigid*twist*

```
% "rigidtwist" Find the twist which generates a rigid motion.
% Use in SE(3).
%
%      xi = rigidtwist(g)
%
% Compute the twist xi in R^6 which generates the homogeneous matrix g 4x4.
%         |v|      |r p|
% E = xi =| | <= g=|   |
%         |w|      |0 1|
%                |exp(W^Th)  (I-exp(W^Th))*(W x v)+W*W'*v*Theta|
% g= exp(E^Theta)=|                                             |
%                |0                          1                 |
% exp(E^Theta)=gst(theta)*gst(0)^-1
% Use Rodrigues's formula:
% exp(W^Th)=I + W^ * sin(Th)+ W^ * W^ * (1-cos(Th))
% With W^=skew(W)
% With W x v == cross product W by v.
% We get "v" from p as:
% v = [[(I-exp(W^Th))*W^+W*W'*Theta]^-1]*p
%
% See also: twistexp.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function x = rigidtwist(g)

    r = rigidorientation(g);
    p = rigidposition(g);
    t = rotationangle(r);
    w = rotationaxis(r,t);
    if t == 0
        t = norm(p);
        if t == 0
            v = [0;0;0];
        else
            v = p/t;
        end
    else
        v = inv((eye(3)-r)*skew(w)+w*w'*t)*p;
    end
    x = [v;w];
```

{RobotMan-f(25): rotationangle

```
% "rotationangle" Find the angle "Theta" of a rotation matrix.
% Use in SO(3).
%
%      Th = rotationangle(R)
%
% Find the angle "Theta" of a rotation matrix. R is the Rotation matrix 3x3.
%
% Th= arccos[(Trace(R)-1)/2]
%
%
% See also: .
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function t = rotationangle(r)

    t = ([r(1,1)+r(2,2)+r(3,3)]-1)/2;
    if t <-1
           t = -1;
    elseif t > 1
        t = 1;
    end
    t=acos(t);
```

{RobotMan-f(26): rotationaxis

```
% "rotationaxis" Find the axis of a rotation matrix
% Use in SO(3).
%
%      W = rotationaxis(R, THETA)
%
% Find the axis of a rotation matrix. R is the Rotation matrix 3x3.
%              |r32-r23|
% W= 1/(2*sinTh)|r13-r31|
%              |r21-r12|
% If sinTh=0 the result is the null space.
%
% See also: .
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function w = rotationaxis(r, t)

    st = sin(t);
    if st == 0
        w = [0;0;0];
    else
        w = 1/(2*st)*[r(3,2)-r(2,3);r(1,3)-r(3,1);r(2,1)-r(1,2)];
    end
```

{RobotMan-f(27): rptohomogeneous

```
% "rptohomogeneous" Convert a rotoation r + translation p to a homogeneous matrix.
% Use in SE(3).
%
%      r = rptohomogeneous(r,p)
%
% Returns a matrix 4x4 from a rotation matrix 3x3 and a translation vector p 3x1.
%    |r p|
% r =|0 1|
%
% See also: .
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function r = rptohomogeneous(r,p)

    r = [r(1,1) r(1,2) r(1,3) p(1);
         r(2,1) r(2,2) r(2,3) p(2);
         r(3,1) r(3,2) r(3,3) p(3);
         0      0      0      1];
```

{RobotMan-f(28): SphereLineIntersection

```
% “SphereLineIntersection” computes exactly what it means.
%
% index = 0 => one solution.
% index > 0 => TWO solutions.
% index < 0 => NO solution.
% solutions = Matrix 3x2 holding the solutions. If there are not then nan
% See also: twistpitch, twistmagnitude.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $
%
function [solution, index]= SphereLineIntersection(lineP1, lineP2, sphereCenter,
sphereRadius)
solution = NaN*zeros(3,1);
x1 = lineP1(1);
y1 = lineP1(2);
z1 = lineP1(3);
x2 = lineP2(1);
y2 = lineP2(2);
z2 = lineP2(3);
x3 = sphereCenter(1);
y3 = sphereCenter(2);
z3 = sphereCenter(3);
a = (x2-x1)^2+(y2-y1)^2+(z2-z1)^2;
b = 2*[(x2-x1)*(x1-x3)+(y2-y1)*(y1-y3)+(z2-z1)*(z1-z3)];
c = x3^2+y3^2+z3^2+x1^2+y1^2+z1^2-2*[x3*x1+y3*y1+z3*z1]-sphereRadius^2;
index = b^2-4*a*c;
if index >= 0,
      u2 = (-b+sqrt(index))/(2*a);
      u1 = (-b-sqrt(index))/(2*a);
      solutionx1 = x1+u1*(x2-x1);
      solutiony1 = y1+u1*(y2-y1);
      solutionz1 = z1+u1*(z2-z1);
      solutionx2 = x1+u2*(x2-x1);
      solutiony2 = y1+u2*(y2-y1);
      solutionz2 = z1+u2*(z2-z1);
      if           sqrt((z1-solutionz1)^2+(x1-solutionx1)^2)>sqrt((z1-solutionz2)^2+(x1-
solutionx2)^2),
        solution = [solutionx2 solutiony2 solutionz2];
      else
        solution = [solutionx1 solutiony1 solutionz1];
      end
end
return;
```

{RobotMan-f(29): *screwtwist*

```
% "screwtwist" Extract the twist associated to a given screw.
% Use in SE(3).
%
%      xi = screwtwist(h, q, w)
%
% Return the twist "xi" 6x1 coordinates of a screw with pitch "h" through the
% point "q" and in the direction "w". If h==inf, then a pure translational
% twist is generated and w gives the direction of translation.
%
% Remember that a twist is a INFINITESIMAL generator of the Euclidean group,
% so this funtion gives us only the twist associated with the screw. Thus, the screw
% motion corresponds to motion along this constant twist by the magnitude of the screw.
%
%     |v|  | -w x q + h*w |
% xi =| | =|              |: if h is not infinity.
%     |w|  |      w       |
%
%     |v|   |     w       |
% xi =| | = |             |: if h is infinity, there is only translation (h=d/Theta).
%     |w|   |     0       |
%
% See also: twistpitch, twistaxis, twistmagnitude.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function x = screwtwist(h, q, w)

    if h == inf
        x = [w; 0; 0; 0];
    else
        x = [-cross(w,q)+h*w; w];
    end
```

{RobotMan-f(30): *screwwrench*

```
% "screwwrench" Extract the wrench associated to a given screw.
% Use in SE(3).
%
%      xi = screwwrench(h, q, w)
%
% Return the wrench "Fi" 6x1 coordinates of a screw with pitch "h" through the
% point "q" and in the direction "w". If h==inf, then a pure torque screw
% is generated and w gives the direction of the application of the torque.
%
% Remember that a wrench is a INFINITESIMAL generator of the Euclidean group,
% so this funtion gives us only the wrench associated with the screw. Thus, the screw
% force corresponds to force along this constant wrench by the magnitude of the screw.
%
%     |f|  |       w      |
% Fi =| | =|              |: if h is not infinity.
%     |T|  | -w x q + h*w |
%
%     |f|   |     0       |
% Fi =| | = |             |: if h is infinity, there is only torque (h=T/f).
%     |T|   |     w       |
%
% See also: screwtotwist, wrenchpitch, wrenchaxis, wrenchmagnitude.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function x = screwwrench(h, q, w)

    if h == inf
        x = [0; 0; 0; w];
    else
        x = [w; -cross(w,q)+h*w];
    end
```

{RobotMan-f(31): skew

```
% "skew" Generate a skew symmetric matrix from an axis.
% Use in SO(3).
%
%      r = skew(w)
%
% Returns a skew symmetric matrix s 3x3 from the vector 3x1 w[a1;a2;a3;].
%    |0  -a3  a2|
% r =|a3   0 -a1|
%    |-a2 a1   0|
% See also: unskew.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function r = skew(w)

       r = [0 -w(3) w(2); w(3) 0 -w(1); -w(2) w(1) 0];
```

{RobotMan-f(32): skewexp

```
% "skewexp" Matrix exponential for a skew symmetric matrix.
% Use in SO(3).
%
%      R = skewexp(W, THETA)
%
% Returns a rotation of THETA about the vector W 3x1.
% Use Rodrigues's formula:
% R(W,Th)=exp(W^ * Th)=I + W^ * sin(Th)+ W^ * W^ * (1-cos(Th))
% Con W^=skew(W)
%
% See also: skew, axistoskew.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function r = skewexp(w, t)

    ws = skew(w);
       r = eye(3)+ws*sin(t)+ws*ws*(1-cos(t));
```

{RobotMan-f(33): spatialjacobian

```
% "spatialjacobian" computes the Spatial Jacobian for a robot of any number of links.
% Use in SE(3).
%
%      jst = spatialjacobian([x1...xn],[t1...tn])
%
% SPATIAL MANIPULATOR JACOBIAN: At each configuration of theta, maps the joint velocity
vector, into the
% corresponding velocity of the end effector.
%  The contribution of the "ith" joint velocity to the end effector velicity is
independent of the configuration
% of later joints in the chain. Thus, the "ith" column of jst is the "ith" joint twist,
transformed to the current
% manipulator configuration.
%
% The pairs {xi,ti} define the joint twist "xi" and joint angle "ti" (or displacemnet)
for each joint of the manipulator.
% xi is a vector 6x1, ti is a value, so the [x1...xn] is a matrix 6xn and [t1...tn] is a
transpose vector 1xn.
% jst is a matrix 6xn.
%
% "gst0" = gst(0), is th location of the tool frame "T" at the reference configuration.
% "gst" and "gst0" are homogeneous matrix 4x4.
% gst(theta)=exp(E1^theta1)*...*exp(En^thetan)*gst(0)
%
%   s          |v1 v2' ... vn'|
% Jst(theta) = |              |
%              |w1 w2' ... wn'|
%           |vi'|
% With: Ei'=|   |=Ad                                        *Ei
%           |wi'|   (exp(E1^theta1)*...*exp(Ei-1^thetai-1))
%    s            b
% Jst(t)=Adg(t)*Jst(t) ; and Adg(t) is the Adjoint transformation of gst(theta).
%
% See also: bodyjacobian, forwardkinematics, twistexp, rigidtwist.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function jst = spatialjacobian(x,t)

    jst = [x(:,1)];
    gst = eye(4);
    for i = 2:size(x,2),
        gst = gst * twistexp(x(:,i-1),t(i-1));
        xip = rigidadjoint(gst)*x(:,i);
        jst = [jst xip];
    end
```

{RobotMan-f(34): *twist*axis

```
% "twistaxis" Find the axis associated with a twist in R^3.
% Use in SE(3).
%
%      l = twistaxis(xi)
%
% Compute the axis "l" of the screw corrsponding to a twist "xi" 6x1.
% The axis "l" is represented as a pair [q,w], where "q" is a point on the axis
% and "w" is a UNIT vector describing the direction of the axis.
%     |v|          W x v
% xi =| | => l = --------- + Kw : K in R, if w is not ZERO.
%     |w|   .     ||w||^2
%     |v|
% xi =| | => l = 0 + Kv : K in R, if w = 0.
%     |w|   .
% Be careful, because this definition is an extension for defining a SCREW
% associated with a twist, but It does not mean that the twist is a screw
% with the translation component parallel to the rotation component.
%
% See also: twistpitch, twistmagnitude.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function l = twistaxis(x)

    v = [x(1,1); x(2,1); x(3,1)];
    w = [x(4,1); x(5,1); x(6,1)];
    if norm(w) == 0
        q = [0; 0; 0];
        w = v / norm(v);
    else
        q = cross(w,v)/(norm(w)^2);
        w = w / norm(w);
    end
    l = [q w];
```

{RobotMan-f(35): *twist*axisdirection

```
% "twistaxisdirection" Find the direction associated with a twist in R^3.
% Use in SE(3).
%
%      w = twistaxisdirection(xi)
%
% Compute the direction "w" 3x1 (coordinates) of the screw corrsponding to a twist "xi"
6x1.
% The "w" is a UNIT vector describing the direction of the axis.
%     |v|          w
% xi =| | => w = ----- :  if w is not ZERO.
%     |w|        ||w||
%     |v|          v
% xi =| | => w = ----- :  if w is ZERO, that is, pure translation.
%     |w|        ||v||
% Be careful, because this definition is an extension for defining a SCREW
% associated with a twist, but It does not mean that the twist is a screw
% with the translation component parallel to the rotation component.
%
% See also: twistaxis.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function w = twistaxisdirection(x)

    l = twistaxis(x);
    w = [l(1,2); l(2,2); l(3,2)];
```

{RobotMan-f(36): *twist*axispoint

```
% "twistaxispoint" Find a point on the axis of the twist in R^3.
% Use in SE(3).
%
%      q = twistaxispoint(xi)
%
% Compute a point "q" 3x1 (coordinates) in the axis of the twist "xi" 6x1.
%     |v|          W x v
% xi =| | => q = --------- , if w is not ZERO.
%     |w|   .     ||w||^2
%     |v|
% xi =| | => q = 0 , if w = 0, that is, pure translation.
%     |w|   .
% Be careful, because this definition is an extension for defining a SCREW
% associated with a twist, but It does not mean that the twist is a screw
% with the translation component parallel to the rotation component.
%
% See also: twistaxis.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function q = twistaxispoint(x)

    l = twistaxis(x);
    q = [l(1,1); l(2,1); l(3,1)];
```

{RobotMan-f(37): *twist*bracket

```
% "twistbracket" Computes the bracket operation to two twists.
% Use in SE(3).
%
%      x = twistbracket(x1,x2)
%
% Returns a twixt "x" from a the application of the bracket operation to the twists x1
and x2.
% This operation is a generalization of the cross product on R3 to vectors in R6.
%    |v|                          v
% xi=| |  = [E1,E2] = [E1^*E2^-E2^*E1]
%    |W|
% Con ^ wedge or twisttohomogeneous operation.
% Con v vee or homogeneoustotwist operation.
%
% See also: homogeneoustotwist, twisttohomogeneous.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function x = twistbracket(x1,x2)

    E1 = twisttohomogeneous(x1);
    E2 = twisttohomogeneous(x2);
    x = homogeneoustotwist(E1*E2-E2*E1);
```

{RobotMan-f(38): *twist*exp

```
% "twistexp" Convert  a twist "xi" 6x1 into a matrix "exp(E^Theta)" 4x4.
% Use in SE(3).
%
%      g = twistexp(xi,theta)
%
% Returns a homogeneous "g" matrix 4x4 from a twist "xi".
%        |v|
% E = xi=| |
%        |W|
%              |exp(W^Th)  (I-exp(W^Th))*(W x v)+W*W'*v*Theta|
% exp(E^Theta)=|                                              |: if w is not 0.
%              |0                           1                |
%              |I v*Theta|
% exp(E^Theta)=|         |: if w is Zero.
%              |0     1  |
% exp(E^Theta)=gst(theta)*gst(0)^-1
% Use Rodrigues's formula:
% exp(W^Th)=I + W^ * sin(Th)+ W^ * W^ * (1-cos(Th))
% With W^=skew(W)
% With W x v == cross product W by v.
%
% BE AWARE that a value Theta = 0 produces a result g=I.
%
% See also: rigidtwist.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function g = twistexp(x,t)

    v = [x(1,1);x(2,1);x(3,1)];
    w = [x(4,1);x(5,1);x(6,1)];
    if norm(w) == 0;
        r = eye(3);
        p = v*t;
    else
        r = skewexp(w,t);
        p = (eye(3)-r)*(cross(w,v))+w*w'*v*t;
    end
    g = rptohomogeneous(r,p);
```

{RobotMan-f(39): *twist*intersection

```
% "twistintersection" Find the intersection point of two twist axis on R^3.
% Use in SO(3).
%
%      q = twistintersection(x1,x2)
%
% Compute a point "q" in intersectin of two twist "xi" 6x1 axis.
%     |v1|      |v2|       |x|                                          |inf|
% x1 =|  |; x2 =|  |=> q = |y| , if there is intersection, otherwise q = |inf|.
%     |w1|      |w2|       |z|                                          |inf|
%
% Parametric description of the axis of the twist x1 which contains x1p1=(x1,y1,z1) and
x1p2=(x2,y2,z2):
% x = x1+t*(x2-x1) ; y = y1+t*(y2-y1) ; z = z1+t*(z2-z1)
% Parametric description of the axis of the twist x2 which contains x2p1=(x3,y3,z3) and
x2p2=(x4,y4,z4):
% x = x3+s*(x4-x3) ; y = y3+s*(y4-y3) ; z = z3+s*(z4-z3)
%
% To detrmine whether these two lines intersect, solve simultaneously the overdetermined
system Ax=B.
% (x2-x1)t-(x4-x3)s = x3-x1
% (y2-y1)t-(y4-y3)s = y3-y1
% (z2-z1)t-(z4-z3)s = z3-z1
%
% See also: twistaxis, twistaxispoint, twistaxisdirection.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function q = twistintersection(x1,x2)

    x1p1 = twistaxispoint(x1);
    x1p2 = x1p1+twistaxisdirection(x1);
    x2p1 = twistaxispoint(x2);
    x2p2 = x2p1+twistaxisdirection(x2);
    A  =  [x1p2(1)-x1p1(1)  x2p1(1)-x2p2(1);  x1p2(2)-x1p1(2)  x2p1(2)-x2p2(2);  x1p2(3)-
x1p1(3) x2p1(3)-x2p2(3)];
    B = [x2p1(1)-x1p1(1); x2p1(2)-x1p1(2); x2p1(3)-x1p1(3)];
    ts = A\B;
 % Be very careful because this formula "x = A\B" Finds a least squares solution for
Overdetermined systems.
 % And we are trying to find whether or not the two axis intersect and NOT the closest
solution. So...
    t = ts(1);
    s = ts(2);
    q1 = x1p1 + t*(x1p2-x1p1);
    q2 = x2p1 + s*(x2p2-x2p1);
    if q1 == q2
        q = q1;
    else
        q = [inf;inf;inf];
    end
```

{RobotMan-f(40): *twist*magnitude

```
% "twistmagnitude" Find the magnitude of a twist in R^3.
% Use in SE(3).
%
%      m = twistmagnitude(xi)
%
% Compute the magnitude "M" of a twist "xi" 6x1.
%     |v|
% xi =| | => M = ||w||, if w is not ZERO.
%     |w|
%     |v|
% xi =| | => M = ||v||, if w = 0.
%     |w|
% Be careful, because this definition is an extension for defining a SCREW
% associated with a twist, but It does not mean that the twist is a screw
% with the translation component parallel to the rotation component.
%
% See also: twistpitch, twistaxis.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function m = twistmagnitude(x)

    v = [x(1,1); x(2,1); x(3,1)];
    w = [x(4,1); x(5,1); x(6,1)];
    if norm(w) == 0
        m = norm(v);
    else
        m = norm(w);
    end
```

{RobotMan-f(41): *twistpitch*

```
% "twistpitch" Find the pitch associate with a twist in R^3.
% Use in SE(3).
%
%      h = twistpitch(xi)
%
% Compute the pitch "h" of a twist "xi" 6x1.
%     |v|          W'*v
% xi =| | => h = ---------
%     |w|   .     ||w||^2
% Be careful, because this definition is an extension for defining a SCREW
% associated with a twist, but It does not mean that the twist is a screw
% with the translation component parallel to the rotation component.
%
% See also: twistaxis, twistmagnitude.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function h = twistpitch(x)

    v = [x(1,1); x(2,1); x(3,1)];
    w = [x(4,1); x(5,1); x(6,1)];
    if norm(w) == 0
        h = inf;
    else
        h = w'*v/(norm(w)^2);
    end
```

{RobotMan-f(42): *twist*tohomogeneous

```
% "twisttohomogeneous" Convert  a twist "xi" 6x1 into a matrix "E^" 4x4.
% Use in SE(3).
%
%      H = twisttohomogeneous(xi)
%
% Returns a twixt "xi" from a homogeneous "h" matrix 4x4.
%    |v|         |W^ v|^
% xi=| |  =>  E^=|    |
%    |W|         |0  0|
% Con W^=skew(W)
%
% See also: wedge.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function r = twisttohomogeneous(x)

    s = skew([x(4,1);x(5,1);x(6,1)]);
    r = [s(1,1) s(1,2) s(1,3) x(1,1);
         s(2,1) s(2,2) s(2,3) x(2,1);
         s(3,1) s(3,2) s(3,3) x(3,1);
         0      0      0      0];
```

{RobotMan-f(43): unskew

```
% "unskew" Generate an axis from an skew symmetric matrix.
% Use in SO(3).
%
%      r = unskew(w)
%
% Returns a vector 3x1 r[a1;a2;a3;] from a skew symmetric matrix w 3x3 .
%    |0  -a3  a2|
% w =|a3   0 -a1|
%    |-a2 a1   0|
% See also: skew.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function r = unskew(w)

       r=[w(3,2);w(1,3);w(2,1)];
```

{RobotMan-f(44): vectortohomogeneous

{RobotMan-f(45): vee

```
% "vee" Convert a matrix "E^" 4x4 into a twist "xi" 6x1.
% Use in SE(3).
%
%      xi = vee(H)
%
% Returns a twixt "xi" from a homogeneous "h" matrix 4x4.
%    |v|         |W^ v|v
% xi=| |  <=  E^=|    |
%    |W|         |0  0|
% Con W^=skew(W)
%
% See also: homogeneoustotwist.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function r = vee(h)

       s = unskew([h(1,1) h(1,2) h(1,3);
              h(2,1) h(2,2) h(2,3);
              h(3,1) h(3,2) h(3,3)]);
       r = [h(1,4);h(2,4);h(3,4);s(1,1);s(2,1);s(3,1)];
```

{RobotMan-f(46): wedge

```
% "wedge" Convert  a twist "xi" 6x1 into a matrix "E^" 4x4.
% Use in SE(3).
%
%      H = wedge(xi)
%
% Returns a twixt "xi" from a homogeneous "h" matrix 4x4.
%    |v|         |W^ v|^
% xi=| |  =>  E^=|    |
%    |W|         |0  0|
% Con W^=skew(W)
%
% See also: twisttohomogeneous.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function r = wedge(x)

    s = skew([x(4,1);x(5,1);x(6,1)]);
    r = [s(1,1) s(1,2) s(1,3) x(1,1);
         s(2,1) s(2,2) s(2,3) x(2,1);
         s(3,1) s(3,2) s(3,3) x(3,1);
         0      0      0      0];
```

{RobotMan-f(47): *wrench*axis

```
% "wrenchaxis" Find the axis associated with a wrench in R^3.
% Use in SE(3).
%
%      l = wrenchaxis(xi)
%
% Compute the axis "l" of the screw corrsponding to a wrench "xi" 6x1.
% The axis "l" is represented as a pair [q,w], where "q" is a point on the axis
% and "w" is a UNIT vector describing the direction of the axis.
%     |f|          f x T
% xi =| | => l = --------- + Kf : K in R, if f is not ZERO.
%     |T|         ||f||^2
%     |f|
% xi =| | => l = 0 + KT : K in R, if f = 0.
%     |T|
% Be careful, because this definition is an extension for defining a SCREW
% associated with a wrench, but It does not mean that the wrench is a screw
% with the lineal force component parallel to the axis of the torque component.
%
% See also: wrenchmagnitude, wrenchpitch, wrenchaxisdirection, wrenchaxispoint.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function l = wrenchaxis(x)

    f = [x(1,1); x(2,1); x(3,1)];
    T = [x(4,1); x(5,1); x(6,1)];
    if norm(f) == 0
        q = [0; 0; 0];
        w = T / norm(T);
    else
        q = cross(f,T)/(norm(f)^2);
        w = f / norm(f);
    end
    l = [q w];
```

{RobotMan-f(48): *wrench*axisdirection

```
% "wrenchaxisdirection" Find the direction associated with a wrench in R^3.
% Use in SE(3).
%
%      w = wrenchaxisdirection(xi)
%
% Compute the direction "w" 3x1 (coordinates) of the screw corrsponding to a wrench "xi"
6x1.
% The "w" is a UNIT vector describing the direction of the axis.
%     |f|          f
% xi =| | => w = ----- :  if f is not ZERO.
%     |T|        ||f||
%     |f|          T
% xi =| | => w = ----- :  if f is ZERO, that is, pure torque.
%     |T|        ||T||
% Be careful, because this definition is an extension for defining a SCREW
% associated with a wrench, but It does not mean that the wrench is a screw
% with the lineal force component parallel to the axis of the torque component.
%
% See also: wrenchmagnitude, wrenchpitch, wrenchaxis, wrenchaxispoint.
%
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function w = wrenchaxisdirection(x)

    l = wrenchaxis(x);
    w = [l(1,2); l(2,2); l(3,2)];
```

{RobotMan-f(49): *wrench*axispoint

```
% "wrenchaxispoint" Find a point on the axis of the wrench in R^3.
% Use in SE(3).
%
%      q = wrenchaxispoint(xi)
%
% Compute a point "q" 3x1 (coordinates) in the axis of the wrench "xi" 6x1.
%     |f|          f x T
% xi =| | => q = --------- , if f is not ZERO.
%     |T|   .     ||f||^2
%     |f|
% xi =| | => q = 0 , if f = 0, that is, pure torque.
%     |T|   .
% Be careful, because this definition is an extension for defining a SCREW
% associated with a wrench, but It does not mean that the wrench is a screw
% with the lineal force component parallel to the axis of the torque component.
%
% See also: wrenchmagnitude, wrenchpitch, wrenchaxis, wrenchaxispoint.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function q = wrenchaxispoint(x)

    l = wrenchaxis(x);
    q = [l(1,1); l(2,1); l(3,1)];
```

{RobotMan-f(50): *wrench*magnitude

```
% "wrench"magnitude" Find the magnitude of a wrench in R^3.
% Use in SE(3).
%
%      m = wrenchmagnitude(xi)
%
% Compute the magnitude "M" of a wrench "xi" 6x1.
%     |f|
% xi =| | => M = ||f||, if f is not ZERO.
%     |T|
%     |f|
% xi =| | => M = ||T||, if f = 0.
%     |T|
% Be careful, because this definition is an extension for defining a SCREW
% associated with a wrench, but It does not mean that the wrench is a screw
% with the lineal force component parallel to the axis of the torque component.
%
% See also: wrenchaxis, wrenchpitch.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function m = wrenchmagnitude(x)

    f = [x(1,1); x(2,1); x(3,1)];
    T = [x(4,1); x(5,1); x(6,1)];
    if norm(f) == 0
        m = norm(T);
    else
        m = norm(f);
    end
```

{RobotMan-f(51): *wrenchpitch*

```
% "wrenchpitch" Find the pitch associate with a wrench in R^3.
% Use in SE(3).
%
%      h = wrenchpitch(xi)
%
% Compute the pitch "h" of a wrench "xi" 6x1.
%     |f|          f'*T
% xi =| | => h = ---------
%     |T|         ||f||^2
% Be careful, because this definition is an extension for defining a SCREW
% associated with a wrench, but It does not mean that the wrench is a screw
% with the lineal force component parallel to the axis of the torque component.
%
% See also: wrenchaxis, wrenchmagnitude.
%
% Copyright (C) 2002-2002, by Jose M. Pardos.
%
% CHANGES:
% Revision 1.1  2002/10/01 00:00:01
% General cleanup of code: help comments, see also, copyright
% references, clarification of functions.
%
% $Revision: 1.1 $

function h = wrenchpitch(x)

    f = [x(1,1); x(2,1); x(3,1)];
    T = [x(4,1); x(5,1); x(6,1)];
    if norm(f) == 0
        h = inf;
    else
        h = f'*T/(norm(f)^2);
    end
```

Bibliografía

Listado por orden alfabético de autores.

[1] R.A. Abraham, and J.E. Marsden. Foundations of Mechanics. Perseus Publishing, 1999.

[2] D. Agahi and K. Kreutz-Delgado. A star topology dynamic model for efficient simulation of multilimbed robotic systems. In *Proc. IEEE InternationalConf. on Robotics and Automation*, pp. 352-7, 1994.

[3] R. Alexander. The gaits of bipedal and quadrupedal animals. In *Int. J. of Robotics Research*, 3(2):49–59, 1984.

[4] A.L. Ames, D.R. Nadeau y J.L. Moreland. VRML SourceBook. John Wiley & Sons, Inc, 1997.

[5] M. Arbulú, F. Prieto, L.M. Cabas, P. Staroverov, D. Kaynov, C. Balaguer. ZMP Human Measure System. In *8tn International Conference on Climbing and Walking Robots (Clawar'2005)*. London. United Kingdom. Sep, 2005.

[6] R.C. Arkin. Behavior Based Robotics. The MIT Press 1998.

[7] W.W. Armstrong. Recursive Solution to the Equations of Motion of an N-link Manipulator. In *Fifth World Congress on Theory of Machines and Mechanisms*, pp. 1343-46, 1979.

[8] V.I. Arnold. Mathematical Methods of Classical Mechanics. New York: Springer-Verlag, 1989.

[9] U.M. Ascher, C. Hongsheng, L.R. Petzold and S. Reich. Stabilization of constrained mechanical systems with DAEs and invariant manifolds. Mechanics of Structures and Machines, Vol. 23, No. 2, pp. 135-57, 1995.

[10] D. Bae and E. Haug. A Recursive Formulation for Constrained Mechanical Systems Dynamics: Part I. Open-Loop Systems. In *Mechanical Strtuctures and Machines*, Vol. 15, No. 3, pp. 359-382, 1987.

[11] C.A. Balafoutis, and R.V. Patel. Dynamic Analysis of Robotic Manipulators: A Cartesian Tensor Approach. Boston: Kluwer, 1991.

[12] C. Balaguer, A. Barrientos, F.J. Rodriguez, R. Aracil, E.A. Puente. Reduction of Free-space-loss for Good and Rapid 3D Path Planning of 6GDL Robots. Journal of Intelligent and Robotic Systems 13:263-278, Kluwer Academic Publishers. 1995.

[13] R.S. Ball. The Theory of *Screws*. Cambridge University Press, Cambridge, 1900.

[14] A. Barrientos, L.F. Peñín, C. Balaguer, R. Aracil. Fundamentos de Robótica. McGraw-Hill, 1997.

[15] A. Barrientos, R. Sanz, F. Matía, E. Gambao. Control de Sistemas Contínuos. McGraw-Hill, 1996.

[16] T.J. Barth, and J.A. Sethian. Numerical Schemes for the Hamilton-Jacobi and Level Set Equations. J. Comp. Physics. 1998.

[17] W.M. Boothby. An Introduction to Differentiable Manifolds and Riemannian Geometry. Boston: Academic Press, 2002.

[18] R.W. Brockett, A. Stokes, and F. Park. A Geometrical Formulation of the Dynamical Equations Describing Kinematic Chains. In *Proc. IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 637-41, 1993.

[19] R.W. Brockett. Robotic manipulators and the product of exponentials formula. In *Proc. Int. Symp. Math. Theory of Networks and Systems*, Beer Sheba, Israel, pp. 120-129, 1983.

[20] R.A. Brooks. Solving the Find-Path Problem by Good Representation of Free Space. In *IEEE* 0018-9472/83/0300-0190, 1983.

[21] L.M. Cabas, S. Torre, M. Arbulú, C. Balaguer. Development of the light-weight human size humanoid robot Rh-0. In *7th International Conference on Climbing and Walking Robots (Clawar'2004)*. Madrid. Spain. Sep, 2004.

[22] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami. Planning Biped Navigation Strategies in Complex Environments. In *IEEE International Conf. on Humanoid Robotics*, 2003.

[23] C. Chevallereau, A. Formalsky and B. Perrin. Low Energy Cost Reference Trajectories for a Biped Robot. In *Proc. IEEE International Conf. on Robotics & Automation*, pp. 1398-1404, 1998.

[24] P. Chiacchio, S. Chiaverini, L. Sciavicco and B. Siciliano. Task space dynamic analysis of multiarm system configurations. In *Int. J. Robotics Research*. Vol. 10, No. 6, pp. 708-715, 1991.

[25] G.S. Chirikjian and A.B. Kyatkin. Engineering Applications of Noncommutative Harmonic Analysis. CRC Press LLC, 2001.

[26] C.K. Chow and D.H. Jacobson. Studies of Human Locomotion via Optimal Programming. Mathematical Biosciences, Vol. 10, pp. 239{306, 1971.

[27] P.I. Corke. Robotics TOOLBOX for MATLAB. 2001.

[28] A.J. Davison, Y. González and N. Kita. Real-Time 3D SLAM with Wide-Angle Vision. In *Proc. IFAC Symposium on Intelligent Autonomous Vehicles*, 2004.

[29] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the Ninth International Conference on Computer Vision (ICCV'03)*, pages 1403–1410, Nice, France, 2003.

[30] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms base don matrices. Journal of Applied Mechanics, pp. 215-221, 1955.

[31] R. Featherstone. Robot Dynamics Algorithms. Kluwer Academic Publishers, 1987.

[32] A. Fijany. New factorizations and fast parallel algorithms for forward dynamics solution of single closed-chain multibody systems. In *Proc. IASTED Int. Conf. Robotics and Manufacturing*. Honolulu, HI, pp. 22-27, 1996.

[33] J. Furusho and A. Sano. Sensor-based control of a nine-link biped. Int. J. of Robotics Research, 9(2):83–98, April 1990.

[34] Y. Fujimoto and A. Kawamura. Simulation of an autonomous biped walking robot including environmental force interaction. IEEE Robotics and Automation Magazine, pp. 33–42, June 1998.

[35] J. Garcia de Jalon and E. Bayo. Kinematic and Dynamic Simulation of Multibody Systems: The Real Time Challenge. New York: Springer-Verlag, 1994.

[36] F. Génot and B. Espiau. On the control of the mass center of legged robots under unilateral constraints. In *International Conference on Climbing and Walking Robots (Clawar'1998) First International Symposium*, pages 9–14, 1998.

[37] V. Geroimenko and C. Chen. Visualizing Information Using SVG and X3D. Springer, 2004.

[38] H. Goldstein. Classical Mechanics. Reading, Massachusetts: Addison-Wesley, 2003.

[39] B. Goodwine and J. Burdick. Trajectory generation for kinematic legged robots. In *Int. Conf. on Robotics and Automation*, pages 2689–2696, 1997.

[40] A. Goswami, B. Espiau and A. Keramane. Limit Cycles in a Passive Compass Gait Biped and Passivity-Mimicking Control Laws. Autonomous Robots, Vol. 4, pp. 273-286, 1997.

[41] A. Goswami. Postural Stability of Biped Robots and the Foot-Rotation Indicator (FRI) Point. The International Journal of Robotics Research, vol.18, no. 6, June 1999, pp. 523-533.

[42] M. Green and D.J.N Limebeer. Linear Robust Control. Prentice-Hall, Englewood Cli s, NJ, 1995.

[43] K.C. Gupta. Kinematic analysis of manipulators using the zero reference position description. Int. J. Robotics Research. Vol. 5, 1986.

[44] M. Hardt, K. Kreutz-Delgado and J. William Helton. Minimal Energy Control of a Biped Robot with Numerical Methods and a Recursive Symbolic Dynamic Model. In *Proc. 37th IEEE Conference on Decision and Control*, pp. 413-6, 1998.

[45] S. Hayati. Hybrid position force control of multi-arm cooperating robots. In *Proc. IEEE Int. Conf. Robotics and Automation*. San Francisco, CA: IEEE, pp. 82-89, 1986.

[46] K. Hirai, M. Hirose, Y. Haikawa and Takenaka. The Development of Honda Humanoid Robot. In *IEEE Conference on Robotics and Automation*. v2, p.p. 1321-1326, 1998.

[47] D. Hsu, L. Kavraki, J. Latombe, R. Motwani and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. Algorithmic Foundation of Robotics, A K Peters Ltd, p.p 141-153, 1998.

[48] Y.K. Hwang and N. Ahuja. Gross motion planning – a survey. ACM Comput. Surv., 24(3):219-291, 1992.

[49] A. Jain and G. Rodriguez. Diagonalized Lagrangian Robot Dynamics. IEEE Transactions on Robotics and Automation, Vol. 11, No. 4, pp. 571-584, 1995.

[50] A. Jain, G. Rodriguez and K. Kreutz-Delgado. Multi-arm grasp and manipulation of objects with internal degrees of freedom. In *Proc. 29th IEEE Conf. on Decision and Control*, pp. 3110-11, 1990.

[51] W. Kahan. Lectures on computational aspects of geometry. Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. Unpublished, 1983.

[52] S. Kajita and K. Tani. Experimental Study of Biped Dynamic Walking. IEEE Control Systems, pp. 13-19, 1996.

[53] F. Kanehiro, K. Fujiwara et al. Open Architecture Humanoid Robotics Platform. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*. Washington, DC May 2002.

[54] F. Kanehiro, T. Yoshimi, S. Kajita, M. Morisawa et al. Whole Body Locomotion Planning of Humanoid Robots based on a 3D Grid Map. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005 .

[55] K. Kaneko, S. Kajita, F. Kanehiro, K. Yokoi, K. Fujiwara, H. Hirukawa, T. Kawasaki, M. Hirata and T. Isozumi. Design of Advanced Leg Module for Humanoid Robotics Project of METI. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, Washington, DC, May 2002.

[56] T. Kato, A. Takanishi, H. Jishikawa and I. Kato. The realization of the quasi-dynamic walking by the biped walking machine. In *Fourth Symposium on theory and Practice of Robots and Manipulators* (A. Morecki, G. Bianchi, and K. Kedzior, eds.), (Warsaw), pp. 341–351, Polish Scientific Publishers, 1983.

[57] S. Kawaji, K. Ogasawara and M. Arao. Rhytm-based control of biped locomotion robot. In *Int. Workshop on Advanced Motion Control Proc.*, pages 93–97, 1998.

[58] O. Khatib. A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation. IEEE Transactions on Robotics and Automation, Vol. RA-3, No. 1, pp. 43-53, 1987.

[59] O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. The International Journal of Robotics Research, 1, MIT, 1986.

[60] A. Konno. Design And Development of the Biped Prototype Robian. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, Wasington,DC; May 2002.

[61] K. Kreutz-Delgado, A. Jain and G. Rodriguez. Recursive Formulation of Operational Space Control. The International Journal of Robotics Research, Vol. 11, No. 4, pp. 320-28, 1992.

[62] J.J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba and H. Inoue. Motion Planning for Humanoid Robots. In *Proc. 11th Int'l Symp. of Robotics Research (ISRR 2003)*.

[63] A. Kun and T. Miller III. Aptive dynamic balance of a biped using neural networks. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pp. 240–245, Apr 1996.

[64] J.C. Latombe.Robot Motion Planning. Kluwer Academic Publications,1991.

[65] R.J. LeVeque. Numerical methods for Conservation Laws. Birkhauser, Basel, 1992.

[66] F. Lewis, C. Abdallah and D. Dawson. Control of Robot Manipulators. Macmillan, 1993.

[67] K.W. Lilly and D.E. Orin. Efficient Dynamic Simulation of Multiple Chain Robotic Mechanism. Journal of Dynamic Systems, Measurement, and Control, Vol. 116, pp. 223-31, 1994.

[68] K.W. Lilly. Effcient Dynamic Simulation of Robotic Mechanisms. Boston: Kluwer, 1993.

[69] H. Liu and H. Iba. A Layered Control Architecture for Humanoid Robot. In *2nd International Conference on Autonomous Robots and Agents*, 2003.

[70] T. Lozano-Perez. Spatial planning: A configuration approach. IEEE Transactions on Computers, 32(2): 108-120,1983.

[71] J. Luh, M. Walker and R. Paul. On-line computational scheme for mechanical manipulators. ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 102, No. 2, pp. 69-76, 1980.

[72] D.W. Marhefka and D.E. Orin. Quadratic optimization of force distribution in walking machines. In *Int. Conf. on Robotics and Automation*, p.p 477–483, 1998.

[73] J.E. Marsden and T.S. Ratiu. Introduction to Mechanics and Symmetry. New York: Springer-Verlag, 1999.

[74] B.M. Martin and J.E. Bobrow. Minimum effort motions for open chain manipulators with task dependent end-eector constraints. In *Proc. IEEE Int. Conf. Robotics and Automation. Albuquerque*, NM: IEEE, pp. 2044-2049, 1997.

[75] Matlab 6.5. The MathWorks, Inc., 2004.

[76] J.M. McCarthy. An Introduction to Theoretical Kinematics. Cambridge: MIT Press, 1990.

[77] T. McGeer. Passive Dynamic Walking. International Journal of Robotics and Research, v9, No. 2, p.p. 62-82, 1990.

[78] L. Meirovitch. Methods of Analytical Dynamics. Dover Publications, 2004.

[79] V.F Muñoz. Planificación de trayectorias para robots móviles. Tesis doctoral, Universidad de Málaga, 1995.

[80] R.M. Murray, Z. Li and S. Sastry. A Mathematical Introduction to Robotics. CRC Press, 1994.

[81] H.Miura and I. Shimoyama. Dynamic walk of a biped. International Journal of Robotics Research, vol. 3, pp. 60–74, 1984.

[82] Y. Nakamura and M. Ghodoussi. Dynamics computation of closed-link robot mechanisms with nonredundant and redundant actuators. IEEE. Trans. Robotics and Automation. Vol. 5, No. 3, pp. 294-302, 1989.

[83] J. Nakanishi, J. Morimoto, G. Endoa, G. Cheng et al. Learning from demonstration and adaptation of biped locomotion. Robotics and Autonomous Systems, Elsevier, 2004.

[84] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko. Task Model of Lower Body Motion for a Biped Humanoid Robot to Imitate Human Dances. In IEEE/RSJ *International Conference on Intelligent Robots and Systems*, 2004.

[85] E. Nicholls. Bipedal Dynamic Walking in Robotics. University of Western Australia, 1998.

[86] T. Oka, M. Inaba and H. Inoue. Describing a modular motion system based on a real time process network model. In *Int. Conf. on Intelligent Robots and Systems*, pages 821–827, 1997.

[87] D.E. Orin, R.B. McGhee, M. Vukobratovic and G. Hartoch. Kinematic and kinetic analysis of open-chain linkages utilizing Newton-Euler methods. Mathematical Biosciences. Vol. 43, pp. 107-130, 1979.

[88] K. Osuka. Control theoretic approach to motion control of legged robot. In *Int. Workshop on Advanced Motion Control Proc.*, pages 88–92, 1998.

[89] B. Paden and S. Sastry. Optimal kinematic design of 6R manipulators. Int.J. Robotics Research. Vol. 7, No. 2, pp. 43-61, 1988.

[90] J.M. Pardos and C. Balaguer. Humanoid Robot Kinematics Modeling Using Lie Groups. In *7th International Conference on Climbing and Walking Robots (Clawar'2004)*. Madrid. Spain. Sep, 2004.

[91] J.M Pardos and C. Balaguer. RH0 Humanoid Robot Bipedal Locomotion and Navigation Using Lie Groups and Geometric Algorithms. In *Conference on Intelligent Robots and Systems (IROS'2005)*. Edmonton. Canada. Aug, 2005.

[92] Va F.C. Park, J.E. Bobrow and S.R. Ploen. A Lie Group Formulation of Robot Dynamics. The International Journal of Robotics Research, Vol. 14, No. 6,pp. 609-618, 1995.

[93] F. Pfeiffer, K. Loeffer, M. Gienger. The Concept of Jogging JOHNNIE. In *Proceedings of the 2002 IEEE international conference on robotics &Automation, Washington*, DC, May 2002.

[94] S. Ploen. Geometric Algorithms for the Dynamics and Control of Multibody Systems. Ph.D. Thesis, University of California, Irvine, 1997.

[95] D. Pogorelov. Dierential-algebraic equations in multibody system modeling. Numerical Algorithms, Vol. 19, No. 4, pp. 183{94, 1998.

[96] M. H. Raibert. Legged Robots That Balance. Cambridge, MA: MIT Press, 2000.

[97] C. Ridderström. Stability of statically balanced stances for legged robots with compliance. In *Int. Conf. on Robotics and Automation*, Washington DC, USA, 2002.

[98] G. Rodriguez. Kalman Filtering, Smoothing, and Recursive Robot Arm forward and Inverse Dynamics. IEEE Journal of Robotics and Automation, Vol.3, No. 6, pp. 624-639, 1987.

[99] G. Rodriguez, A. Jain and K. Kreutz-Delgado. Spatial Operator Algebra for Multibody System Dynamics. Journal of the Astronautical Sciences, Vol. 40, No. 1, pp. 27-50, 1992.

[100] J. Rose and J.G. Grimble. Human Walking. Williams & Wilkins, 1994.

[101] M. Rostami and G. Bessonnet. Impactless sagittal gait of a biped robot during the single support phase. In *Proc. IEEE International Conf. on Robotics & Automation*, pp. 1385-91, 1998.

[102] L. Roussel, C. Canudas de Wit and A. Goswami. Generation of Energy Optimal Complete Gait Cycles for Biped Robots. In *Proc. IEEE International Conf. on Robotics & Automation*, pp. 2036-41, 1998.

[103] J.M. Selig. Geometric Fundamentals of Robotics. Springer-Verlag, 2004.

[104] J.A. Sethian. Level Set Methods and Fast Marching Methods. Cambridge University Press 1999.

[105] C. Shih, Y. Zhu and W.A. Gruver. Optimization of the biped robot trajecotry. In *Int. Conf. on Systems, Man and Cybernetics*, pages 899–903, 1991.

[106] W.M. Silver. On the equivalence of Lagrangian and Newton-Euler dynamics for manipulators. Int. J. Robotics Research. Vol. 1, No. 2, pp. 118-128, 1982.

[107] M. Sipser. Introduction to the Theory of Computation. PWS Publishing Company, 1997.

[108] J.J.E. Slotine and W. Li. Applied Nonlinear Control. New Jersey: Prentice Hall, 1991.

[109] S.M. Song and K.J. Waldron. Machines that Walk. MIT Press, Cambridge, MA, 1989.

[110] M.W. Spong and M. Vidyasagar. Robot Dynamics and Control. John Wiley & Sons, 1989.

[111] K. Sorao, T. Murakami and K. Ohnishi. A unified approach to ZMP and gravity center control in biped dynamic. In *Int. Workshop on Advanced Motion Control* Proc., page 112, 1997.

[112] A. Stokes and R. Brockett. Dynamics of kinematic chains. Int. J. Robotics Research. Vol. 15, No 4, pp. 393-405, 1996.

[113] S. Stitt and Y.F. Zhen. Distal learning applied to biped robots. In *Int. Conf. on Robotics and Automation*, volume 1, pages 137–142, 1994.

[114] T. Sugihara and Y. Nakamura. A Fast Online Gait Planning with Boundary Condition Relaxation for Humanoid Robots. In *IEEE International Conference on Robotics and Automation (ICRA'05)*, April, 2005.

[115] A. Takanishi, H.-o. Lim, M. Tsuda and I. Kato. Realisation of dynamic biped walking stabilised by trunk motion on a sagitally uneven surface. In *Proceedings of the IEEE Int. Workshop on Intelligent Robots and Systems (IROS)*, 1990.

[116] D. J. Todd. Walking machines – an introduction to legged robots. Kogan Page Ltd, London, 1986.

[117] S.Torre, L.M. Cabas, M. Arbulú, C. Balaguer. Inverse Dynamics of Humanoid Robot by Balanced Mass Distribution Method. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2004)*, Sep, 2004.

[118] L.W. Tsai. Robot Analysis. Wiley-Interscience, 1999.

[119] S. Tzafestas, M. Raibert and C. Tzafestas. Robust sliding-mode control applied to a 5-link biped robot. J. of Intelligent and Robotic Systems, 15(1):67–133, 1996.

[120] D. Vallejo, C. Jones and N. Amato. An adaptive framework for single shot motion planning. In *Proceedings of IEEE-RSJ, International Conference on Intelligent Robot and Systems*, 2000.

[121] C. Versino, L.M. Gambardella. Learning Fine Motion in Robotics: Design and Experiments, Recent Advances in Artificial Neural Networks, Design and Application. CRC press, 2000.

[122] M. Vukobratovic and J. Stepanenko. On the stability of anthropomorphic systems. J. Math. Biosciences, 15:1–37, 1972.

[123] M.K. Vukobratovic, V.F. Filaretov and A.I. Kozun A unified approach to mathematical modeling of robotic manipulator dynamics. Robotica. Vol. 12, pp. 411-420, 1994.

[124] G. Walsh, R. Montgomery and S.S. Sastry. Optimal Path Planning on Matrix Lie Groups. Electronics Research Laboratory, University of California Berkeley, 1994.

[125] J.M.Wendlandt and S.S. Sastry. RecursiveWorkspace Control of Multibody Systems: A Planar Biped Example. In *Proc. 35th IEEE Conf. on Decision and Control*, pp. 3575-80, 1996.

[126] G. Wyeth, D. Kee and Tak Fai. Evolving a Locus Based Gait for a Humanoid Robot. In *IEEE Int. Conference on Intelligent Robots and Systems (IROS'2003)*.

[127] J.I. Yamaguchi, A. Takanishi, y I. Kato, "Development of a biped walking robot compensating for three-axis moment by trunk motion," In *Proceedings of the 1993 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, July 1993.

[128] K. Yoneda and S Hirose. Tumble Stabilit Criterion of Integrated Locomotion and Manipulation. In *Proceedings of IEEE International Conference on Intelligent Robot and Systems*, 1996, pp. 870-876.

[129] J. Yong and X.Y. Zhou. Stochastic Controls, Hamiltonian Systems and HJB Equations. Springer. 1999.

[130] C. Yin, A. Albers, J. Ottnad et al. Stability Maintenance of a Humanoid Robot under Disturbance with Fictitious Zero-Moment Point (FZMP). *IEEE Int. Conference on Intelligent Robots and Systems (IROS'2005)*.

[131] Y. Yu and K. Gupta. Sensor based motion planning for manipulator arms: An Eye in hand system. In *IEEE International Conference on Robotics and Automation*, video session, 2000.

[132] K. Zhou. Essentials of Robust Optimal Control. US Imports & PHIPEs, 1998.

[133] H. Zhou and H. Hu. Kinematic model aided inertial motion tracking of human upper limb. *In Proceedings of the 2005 IEEE International Conference on Information Acquisition*, June 27 - July 3, Hong Kong and Macau, China, 2005.