PLATAFORMA DE SOFTWARE PARA MODELADO Y SIMULACIÓN DE ROBOTS MANIPULADORES

Andrés Jaramillo Botero† , Antonio Matta Gómez,
Juan Fernando Correa Caicedo, Wilber Pérez Castro

Grupo de Automática y Robótica
Pontificia Universidad Javeriana - Cali.
Calle 18 # 118-250, Cali, Colombia
E-mail: gar@puj.edu.co

Modelar y simular tridimensionalmente sistemas roboticos manipuladores es importante debido al alto costo de inversión requerido en la implantacion de los mismos en una aplicación real. El presente articulo describe las principales partes que conforman el sistema modular de software incorporados en ROBOMOSP [1], una plataforma que se constituye en un entorno ideal para: el diseño de robots manipuladores, el entrenamiento fuera de línea de operadores y la operación en línea de robots en entornos reales de variada aplicación. ROBOMOSP aporta elementos novedosos respecto de otras herramientas comercialmente disponibles, entre los cuales están la solución a problemas dinámicos (inverso y directo) con cálculo automático de parámetros de masa a partir de objetos gráficos, solución completa al problema cinemático (inverso y directo), programación fuera de línea a partir de un lenguaje estándar (IRL), programación en línea mediante un interpretador de comandos gráfico, interfaz API para experimentación de nuevos algoritmos por parte del usuario, y el soporte de comunicación por sockets. ROBOMOSP es un ambiente multiplataforma (Linux®, MacOSX®, MS Windows®) construido mediante el uso de herramientas de software de dominio público. El artículo termina con una breve discusión sobre el trabajo actual y futuro alrededor de ROBOMOSP.
†Autor de contacto.

3D modelling and simulation of robotic manipulators is becoming increasingly important due to the high cost of implementing physical solutions in real world applications. This article describes the design and construction principles, at both the software and theoretical levels, of ROBOMOSP (Robotics Modelling and Simulation Plataform) a complete modelling and simulation software tool intended to serve as a robot design environment, as a training tool and as an on-line interface for the operation of robotics environments in multiple applications. ROBOMOSP introduces novel elements to support the specified utility criterion, when compared to other existing, commercial and non-commercial, software packages. It allows for automated calculation of mass properties of individual and colective parts from graphical objects, full dynamic analysis (inverse and forward solutions), full kinematics analysis (inverse and forward solutions), off-line programming via the DIN standard IRL (International Robot Language), socket communications for remote operation, and a high level API that allows users to incorporate their own algorithmic solutions. ROBOMOSP is entirely multiplatform (Linux ®, MacOSX ®, MS Windows ®) developed using public domain software tools. The article concludes with a brief discussion about current and future work on ROBOMOSP.

Key Words: robotics, modelling, simulation, kinematics, dynamics, trajectories, offline-programming

Recibido: 16-septiembre-2004 Revisado: 06-octubre-2004 Aceptado: 05-noviembre-2004

0. INTRODUCCION

La capacidad para representar el comportamiento de sistemas complejos mediante modelos precisos cobra importancia con el advenimiento de sistemas computacionales de alto rendimiento. De acuerdo con el nivel de complejidad de un sistema particular se emplean modelos analíticos que permiten soluciones cerradas, numéricas o de observación, que se derivan de una identificación de las características críticas del sistema. Los manipuladores robóticos son sistemas altamente complejos tanto en su diseño como en su operación; en consecuencia, el desarrollo de plataformas computacionales que permitan

modelar su diseño y simular su comportamiento, se constituyen como fundamentales tanto para el equipo diseñador como para los usuarios.

Teniendo en cuenta los altos costos de inversion de los sistemas robóticos manipuladores, se hace imprescindible predecir su utilidad en una aplicación particular. Esto ha fomentado la creación de herramientas de software gráfico (RoboWorks© [2], RobotAssist [3], Easy-ROB3D™ [4]) por múltiples proveedores e incluso por los mismos fabricantes de robots. Sin embargo, las herramientas disponibles en el mercado son de tipo propietario, particularizadas (algunas de ellas) por manipulador, o limitadas en sus características (Vg. no soportan simulación dinámica, no soportan cálculo automático de parámetros físicos, ni incorporan lenguajes estándares de programación, entre otras).

Este artículo describe el diseño de un sistema de modelado, simulación y programación de robots manipuladores denominado ROBOMOSP (ROBOtics MOdeling and Simulation Platform) que satisface las limitaciones encontradas en las herramientas referidas en el párrafo anterior. ROBOMOSP es un sistema CAD 3D cross-compatible entre los sistemas operativos Linux®, MacOSX® y MS Windows®, desarrollado con herramientas Open Source [5]. La herramienta permite modelar componentes físicos y su integración en el diseño de robots manipuladores, composición de entornos de trabajo, definición de trayectorias espaciales, simulación y control cinemático y dinámico, programación de tareas complejas mediante lenguaje de alto nivel, ejecución de algoritmos de usuario a través de una interfaz API, y operación remota por medio de sockets.

La primera sección describe los aspectos generales de la metodología empleada para el desarrollo de ROBOMOSP. La segunda sección describe información relacionada con su arquitectura, interfaz de usuario e implementación. En la sección 3 se describen los módulos desarrollados para el modelado de los elementos que componen un sistema manipulador robótico y su entorno. También se detalla el módulo de construcción de sólidos rígidos desde una perspectiva gráfica y se explica su modelado como objetos que poseen propiedades de masa. A continuación se describe el soporte para el modelado cinemático y dinámico de robots manipuladores en la herramienta. Adicionalmente, se discute el proceso de construcción de trayectorias y se hace referencia al «mundo» en donde se integran todos los elementos anteriores. En la sección 4 se describe el soporte para programación fuera de línea de robots utilizando el lenguaje estándar IRL [6]. La sección 5 trata sobre la solución implementada a los problemas de cinemática y dinámica, inversos y directos. El artículo termina resaltando los logros obtenidos y haciendo una breve descripción de las actualizaciones propuestas para ROBOMOSP.

1. ASPECTOS GENERALES

ROBOMOSP está organizado por subsistemas jerárquicos: interfaz gráfica de usuario, subsistema de modelado 3D, subsistema de simulación gráfica 3D, subsistema robótico, subsistema de programación y API (comandos IOCI: Input Output Command Interface) que sirve de interfaz para intercambiar información con aplicaciones externas.

Dado que el proceso de construcción de ROBOMOSP ha sido de tipo evolutivo [7], se ha utilizado el modelo en espiral basado en componentes [8] como metodología de desarrollo. Esta metodología permite el diseño e implementación de sistemas cuyas funcionalidades son mejoradas o adicionadas al avanzar en ciclos, previamente definidos, de la espiral central de desarrollo. Conjuntamente se deben seguir los lineamientos de conexión y organización entre los módulos del sistema que se especifican en la arquitectura del software.[9]

2. DISEÑO E IMPLEMENTACION DE LA ARQUITECTURA DEL SOFTWARE

2.1. Diseño

Cada subsistema de ROBOMOSP está conformado por bloques funcionales que representan alguna de las operaciones y utilidades de alto nivel soportadas. A su vez, un bloque funcional esta compuesto por entidades básicas de encapsulamiento de servicios denominadas componentes [10].

La figura 1 muestra la forma en que estos bloques funcionales están conectados entre sí y cómo es el flujo de información entre los mismos.

También existe una subordinación jerárquica entre subsistema y subsistema, debido a lo cual, la arquitectura del software del sistema ROBOMOSP es de estilo estratificado, es decir, está dividida en capas de acuerdo a la dependencia existente entre los componentes pertenecientes a cada uno de los bloques funcionales.

En la figura 2 se pueden apreciar las cuatro capas estratificadas del sistema ROBOMOSP. La capa central realiza las operaciones básicas (reservar memoria, iniciar estructuras y procesar el llamado a las funciones de los componentes que la conforman) y todos los componentes de las tres capas superiores pueden acceder a ella directamente. En la capa de procesamiento de datos se encuentran los componentes del subsistema robótico, los cuales realizan los cálculos matemáticos más complejos y sus resultados alimentan al subsistema

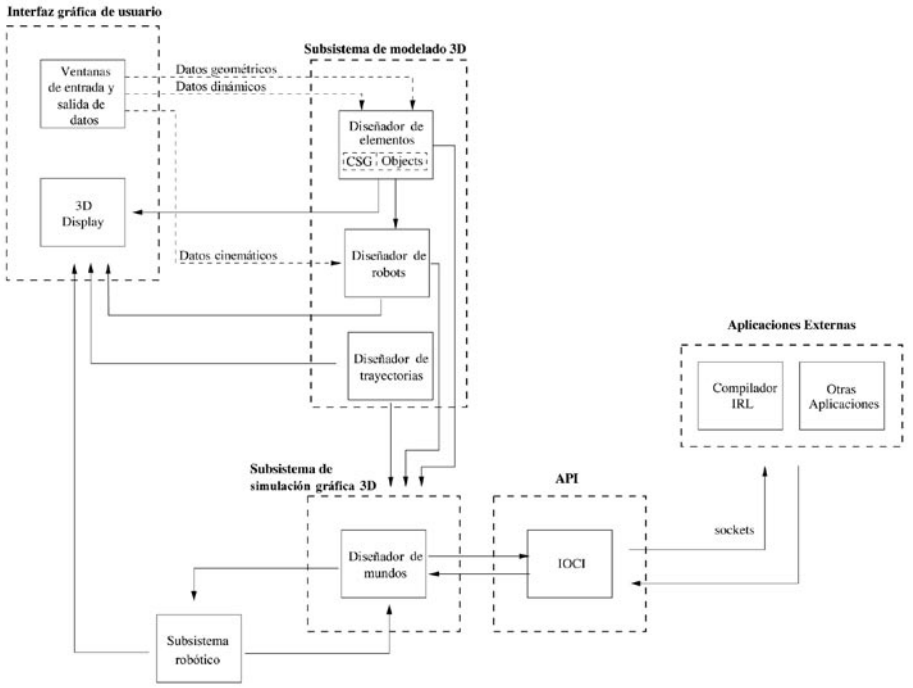

Figura 1. Bloques funcionales del sistema ROBOMOSP.

de simulación gráfica 3D. La capa de la aplicación proporciona los servicios de modelado 3D y la comunicación con el subsistema robótico. En la capa de la interfaz de usuario se encuentran los componentes que implementan las operaciones de interacción gráfica entre el usuario y los subsistemas internos de ROBOMOSP.

2.2. Interfaz Gráfica de Usuario

La interfaz gráfica de usuario (ver figura 3) fue realizada con el propósito de facilitar el modelado y la simulación interactiva de los diversos elementos pertenecientes a los bloques funcionales de ROBOMOSP. Está conformada por cinco ventanas principales:

1. Ventana Object Navigator:

Permite acceder a la información almacenada en los componentes de los subsistemas de modelado 3D y simulación gráfica 3D (materials, CSGs, objects, robots, trajectories, worlds), de tal forma que el usuario puede realizar operaciones comunes sobre dicha información: abrir, salvar, salvar como, borrar, recargar, editar y consultar memoria utilizada. Los componentes desplegados

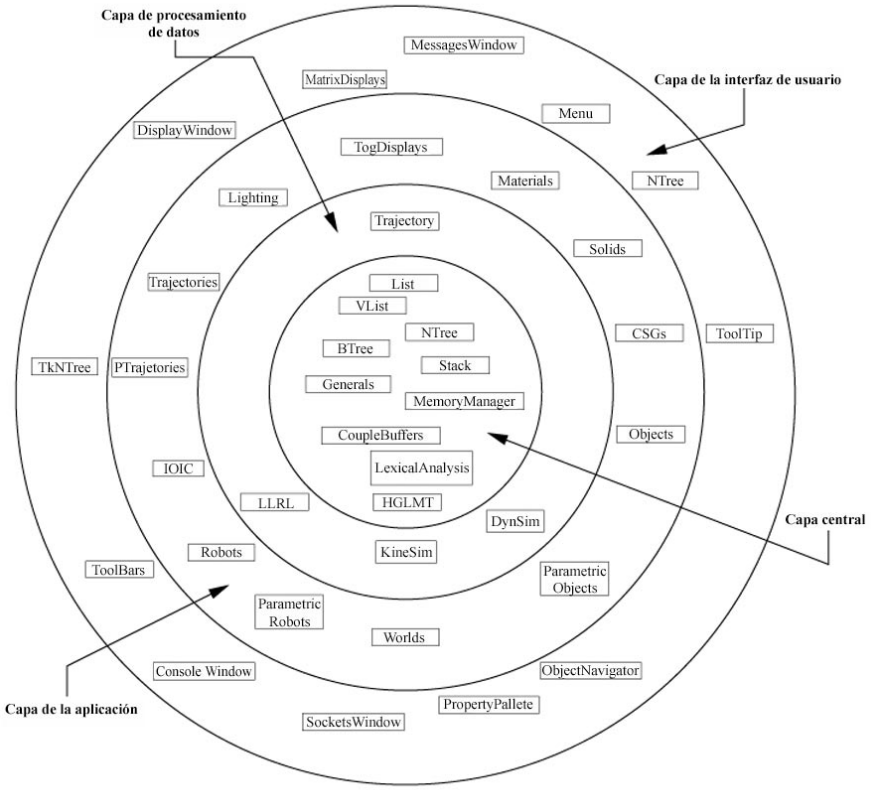

Figura 2. Arquitectura del software del sistema ROBOMOSP.

en esta ventana son inmediatemente graficados tridimensionalmente en la ventana Display Window.

2. Ventana Properties Palette:

En esta ventana se muestran las propiedades de los elementos desplegados en la ventana Object Navigator que el usuario aquí puede modificar.

3. Ventana Display Window:

En ella se representan, mediante sólidos 3D, cualquier componente activo en la ventana Object Navigator.

4. Ventana Messages Window:

Ventana que despliega los mensajes, tanto de error como de éxito, que pueda generar el sistema.

5. Ventana Console Window:

Ventana sobre la cual se ejecutan el conjunto de comandos de entrada y salida (Input Output Command Interface, IOCI) que conforman una interfaz de programación (API) a través de la cual aplicaciones externas pueden acceder

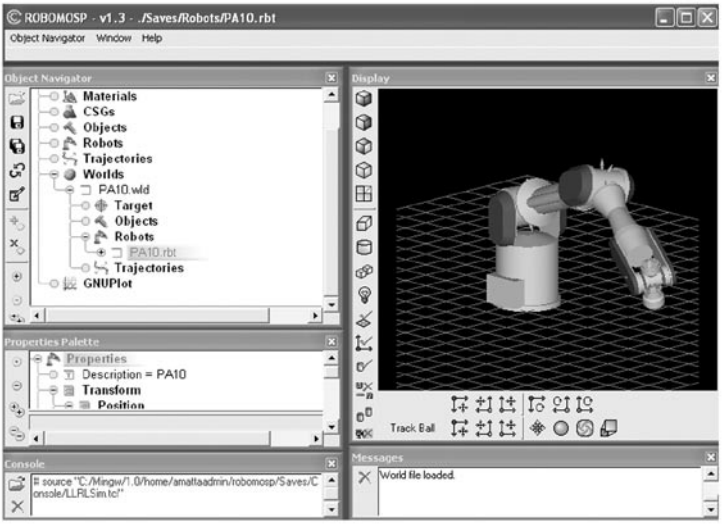

Figura 3. Interfaz gráfica de usuario del sistema ROBOMOSP.

a los subsistemas internos de ROBOMOSP, ya sea localmente o remotamente, mediante la utilización de sockets.

2.3. Implementación

Para la implementación se utilizaron herramientas y librerías de desarrollo de código abierto que satisfacen las condiciones de la licencia GNU Public License [11]. Por cuestiones de desempeño y portabilidad la mayor parte del sistema está codificado en ANSI C y la totalidad de la interfaz gráfica de usuario está implementada en TCL/TK [12]. La ventana de visualización de ROBOMOSP está desarrollada en Togl (un widget de TCL/TK); en ella se representan tridimensionalmente todos los sólidos del sistema, haciendo uso de la librería gráfica OpenGL [13].

Para los cálculos matemáticos matriciales se utiliza la librería Meschach [14] y para la generación automática de una mallas de polígonos a partir de un árbol CSG (Constructive Solid Geometry [15]) se hace uso de la librería NetGen [16].

3. MODELADO

Las tareas de modelado tridimensional soportadas por ROBOMOSP son realizadas por el subsistema de modelado 3D. Hasta el momento las

funcionalidades soportadas son: diseño de sólidos 3D con sus respectivos parámetros de posición, orientación y de superficie; creación y manipulación de materiales; definición de colores, luz ambiente y luz local; modos de despliegue de sólidos (wireframe, filled, flat shaded, smooth), soporte de vistas isométricas.

La construcción de sólidos se realiza de manera ascendente de acuerdo al nivel de complejidad de lo que se esté construyendo. En este orden de ideas se parte del diseño de sólidos CSGs (parámetros geométricos de un objeto), se continua con la integración en objetos compuestos CSG (Objects), para luego conformar el ensamble de robots manipuladores (Robots). Concurrentemente se pueden diseñar trayectorias espaciales (Trajectories) y objetos adicionales que conjuntamente con los robots manipuladores conforman un mundo de simulación (Worlds). (Ver figura 4).

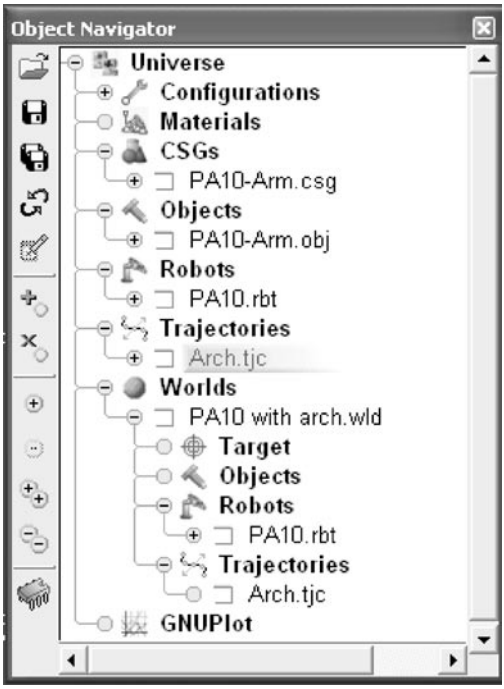

Figura 4. Orden de modelado de elementos.

3.1. Modelado geométrico de sólidos rígidos 3D

ROBOMOSP utiliza un modelo híbrido que combina la técnica CSG y la representación en malla (Mesh) de sólidos tridimensionales para definir geométricamente los sólidos rígidos 3D.

La técnica CSG permite modelar objetos complejos utilizando operaciones booleanas (Unión, Intersección, Diferencia), que se aplican a primitivas básicas pertenecientes a un árbol n-ario[15]. La utilización de esta técnica facilita el modelado interactivo de sólidos complejos, se ajusta a las ventajas que la interfaz gráfica de usuario ofrece y el número de objetos que se pueden representar es prácticamente ilimitado [17].

Una malla define la superficie de un sólido mediante un conjunto de polígonos triangulares (representación B-rep) y es utilizada para graficar el sólido obtenido a partir de un árbol CSG. (Ver figura 5). Esta malla se obtiene llamando a las funciones de generación de mallas de la librería NetGen [16].

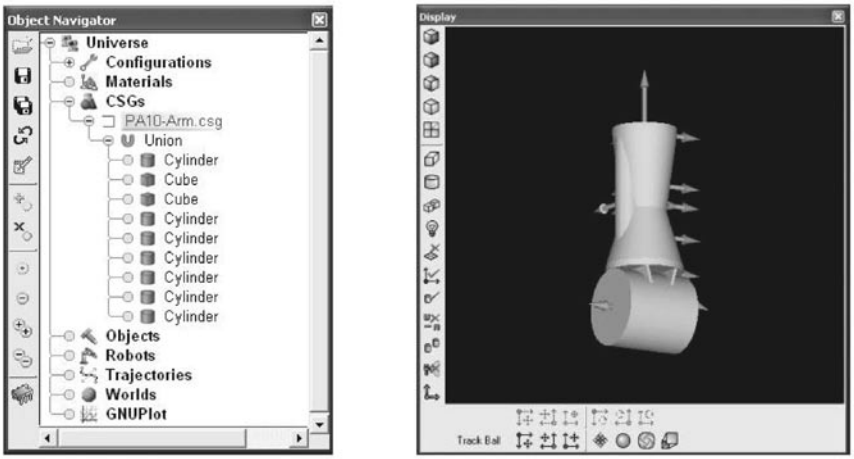

Figura 5. a) Árbol CSG del elemento antebrazo del robot Mitsubishi PA-10 [18]. b) Sólido CSG generado.

Para el posicionamiento en el espacio de estos sólidos CSG, ROBOMOSP utiliza una estructura denominada HGLMT (Homogeneous Graphics Library Matrix Transformation) que permite almacenar los valores de las matrices de posición y orientación para cada sólido rígido con referencia local o global (sistema de coor- denadas inercial).

3.2. Modelado de Objetos con propiedades de masa

Una vez construida la representación gráfica de los elementos que constituyen un cuerpo sólido rígido, el paso siguiente es caracterizar el sólido con sus parámetros de masa (necesarios para solucionar los problemas dinámicos, inverso y directo). El componente object del diseñador de elementos tiene como función principal permitir que el usuario ensamble las partes que constituyen

un cuerpo sencillo o compuesto y calcular sus propiedades de masa respecto de su centro de gravedad.

Para realizar el cálculo de las propiedades de masa de cada elemento gráfico se recurre a la malla generada por el módulo de CSG. Esta representación de vértices y caras aproxima al sólido con un poliedro, que debe ser de densidad uniforme. El cálculo de sus parámetros dinámicos (masa, centro de masa y tensor de inercia) se logra utilizando el método propuesto por Mirtich [20]. Los valores del tensor de inercia están expresados respecto del centro de masa que tiene como referencia el sistema de coordenadas del sólido CSG.

Cada elemento gráfico CSG es un nodo de un árbol n-ario que representa y relaciona el posicionamiento de los sólidos que conforman el objeto. Cada nodo tiene sus parámetros dinámicos referenciados al sistema de coordenadas local. El objeto completo está representado por este árbol y además por los parámetros dinámicos de sus partes.

La masa del objeto compuesto corresponde a,

$$M = \sum_{i=1}^{n} m_i, \tag{1}$$

donde m_i equivale a las masas de cada sólido i CSG.

El tensor de inercia del objeto compuesto J_{total}^{cm}, se deduce a partir de la aplicación del teorema de ejes paralelos sobre cada elemento CSG orientado respecto del sistema de coordenadas del objeto compuesto,

$$J_{obj_i} = R_{O_i}^{Obj_i} J_{O_i} \left(R_{O_i}^{Obj_i}\right)^T - m_i \tilde{p}_i \tilde{p}_i \tag{2}$$

sumando los tensores parciales, J_{obji}, derivados de la ecuación 2,

$$J_{total}^{Obj_i} \sum_{i=1}^{n} obj_i \tag{3}$$

calculando el centro de masa del objeto compuesto,

$$\widehat{CM}_{total}^{Obj} = \frac{\sum_{i=1}^{n} \widehat{CM}_i^{Obj} m_i}{\sum_{i=1}^{n} m_i} \tag{4}$$

y luego aplicando el teorema de ejes paralelos para el sistema completo, total total

$$J_{total}^{cm} = J_{total}^{Obj} + M\widetilde{ss} \tag{5}$$

donde, $\boldsymbol{R}_{Oi}^{Obj}$ es la matriz de orientación que relaciona el sistema local del sólido CSG (***Oi***) respecto del sistema del objeto (***Obj***), $\boldsymbol{J}_{Oi}$ corresponde al tensor de inercia del elemento CSG respecto de su propio sistema de coordenadas, $\widetilde{\boldsymbol{p}}_i$ equivale a la distancia desde el origen del sistema ***Obj*** al origen del sistema $\boldsymbol{O}_i$, $\widehat{\boldsymbol{CM}}_{i}^{Obj}$ expresa el centro de masa de cada elemento CSG respecto del sistema de coordenadas del objeto compuesto y $\widetilde{\boldsymbol{s}}$ corresponde al operador producto cruz en forma matricial que expresa la distancia entre el origen del sistema de referencia del objeto hasta el centro de masa total.

Es de anotar que los sólidos CSG que constituyen un objeto no pueden solaparse al estar en el ensamble del objeto compuesto.

La ventaja que tiene calcular los parámetros dinámicos de un objeto aproximándolo por un poliedro radica en la rapidez del cálculo. Esta aproximación sacrifica la precisión en un máximo calculado del 7% para el objeto esfera. Por otro lado, calcular las propiedades de masa de los elementos constitutivos de un objeto y relacionarlos en un árbol °exibiliza el proceso de diseño, puesto que los cambios realizados a un solo elemento no afectan a los demás. Esto permite observar interactivamente cómo cualquier alteración de las propiedades de masa de un solo elemento modifica las propiedades del ensamble.

3.3. Modelado de Robots

Este módulo maneja objetos modelados para definir la base del robot y una estructura jerárquica para modelar las uniones y los objetos de cada una de las articulaciones que conforman un brazo mecánico articulado.

Los objetos pertenecientes a la base del robot se posicionan con respecto al sistema de coordenadas mundo (referencia inercial). Cada articulación se define mediante una matriz D-H [21] (convención Denavit - Hartenberg), un tipo (rotacional o traslacional), el rango de movimiento correspondiente (valor mínimo y máximo del parámetro μ, para articulaciones rotacionales; o d, para articulaciones prismáticas), y si la parte asociada a dicha articulación va a estar o no visible, según lo desee el diseñador. Cada una de las articulaciones tiene un objeto asociado, que se referencia respecto al sistema de coordenadas de la articulación de la cual hace parte (convención D-H, parte anterior del objeto). De esta manera se define completamente el modelo cinemático, para expresar la geometría del movimiento en tiempo, de un robot manipulador (ver figura 6).

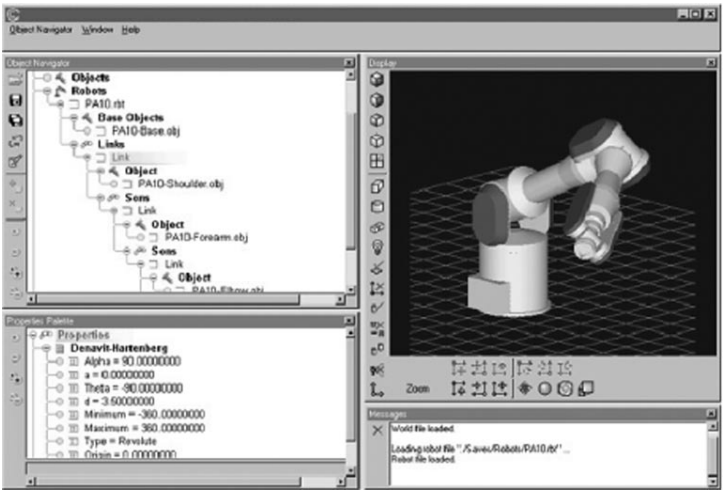

Figura 6. Parámetros cinemáticos del robot industrial Mitsubishi PA-10 [18].

Con la información de los parámetros dinámicos de cada uno de los objetos que forman los elementos del manipulador y el modelo cinemático, se calcula la matriz inercial de masa de cuerpo articulado [22] que relaciona las fuerzas y las aceleraciones que intervienen en el movimiento del sistema (problema dinámico).

3.4. Modelado de Trayectorias

El módulo de construcción de trayectorias está diseñado para permitir al usuario modelar los caminos y las condiciones cinemáticas con las que debe cumplir un robot al ejecutar una tarea deseada. Las variables de tiempo, velocidad y el tipo de movimiento con el cual se recorre la trayectoria son parametrizables. El módulo provee facilidades para especificar trayectorias en coordenadas cartesianas y articulares. Los tipos de trayectorias que se pueden construir son:

- Linea Recta.
- Arco de Circunferencia.
- Circunferencia.
- Curvas Paramétricas con splines.
- Curvas de Bézier.

En coordenadas articulares se permite especificar la posición inicial y final del robot, incorporando además ligaduras de camino. La generación de la trayectoria se hace a partir de polinomios interpolantes como los descritos en [23] o [24] que cumplan con las condiciones iniciales impuestas por el usuario. Sin embargo, como el proceso de modelado de las trayectorias articulares es

dependiente de la configuración cinemática del robot manipulador no es posible visualizarlas hasta su simulación.

En la construcción de trayectorias cartesianas existen varios métodos para la generación de caminos. En ROBOMOSP se incorporan dos métodos: funciones analíticas e interpolación polinomial [25]. Las funciones analíticas restringen geométricamente el movimiento del robot. Con este tipo de funciones se pueden generar rectas, arcos y circunferencias. Adicionalmente, ROBOMOSP permite parametrizar las variables cinemáticas de movimiento (Vg. velocidad constante, aceleración constante, curvas de velocidad trapezoidales, entre otras).

La interpolación polinomial se ajustan a la especificación de tareas por puntos de consigna en el camino, sin exigir una trayectoria específica entre éstos, siempre y cuando se cumpla con las restricciones posicionales en los tiempos deseados y con las condiciones de velocidad (si las hay) impuestas por el usuario. Los métodos de interpolación son las curvas paramétricas cuya construcción se basa en encontrar un polinomio interpolante que une los puntos definidos por el usuario de una manera suave y continua. El método de interpolación empleado para este tipo de curvas es el de splines cúbicos [26] y de grado cuatro, que permiten especificar la velocidad de paso en cada uno de los puntos, mientras que los de grado tres pueden ser sujetos o naturales. También se implementaron curvas de Bézier modificadas para permitir la incorporación de parámetros de movimiento tales como tiempo, velocidades iniciales y finales.

$$P(t) = \frac{1}{(b-a)^n} \sum_{i=0}^{n} P_i B_{i,n}(t) \tag{6}$$

donde, $B_{i,n} = conv(n, i)(t-a)^i (b-t)^{n-i}$ para $a \le t \le b$

$$P'(a) = V_i = \frac{1}{b-a} (2P1 - 3P0) \tag{7}$$

$$P'(b) = V_f = \frac{3}{b-a} (P3 - P2) \tag{8}$$

ROBOMOSP soporta la definición de tareas complejas que requieren de la construcción de trayectorias compuestas por diferentes funciones primitivas en cualquier combinación de funciones analíticas e interpoladas. Esto se logra con la introducción del concepto de tramo o segmento dentro de la forma de

construcción y definición de una trayectoria. Cada segmento de una trayectoria compuesta puede construirse de manera independiente, permitiendo la modularidad y la reutilización de trayectorias almacenadas.

Al componer una trayectoria compuesta, los segmentos consecutivos pueden estar separados espacialmente o coincidir en su posición, pero los valores de orientación o velocidades de translación o rotación pueden presentar discontinuidades espaciales. Para evitar estas anomalías, la herramienta calcula automáticamente una transición suave entre segmentos consecutivos. El usuario tiene la posibilidad de variar los parámetros de dicha unión con el fin de ajustarlos a las condiciones físicas realizables por el manipulador a utilizar.

Los resultados obtenidos por el módulo se pueden observar en representación tridimensional en la ventana 3D Display Window de ROBOMOSP, o a través del graficador GNUplot [27] embebido en la plataforma que permite evaluar la evolución de cada una de las variables euclidianas respecto de tiempo, (ver figura 7).

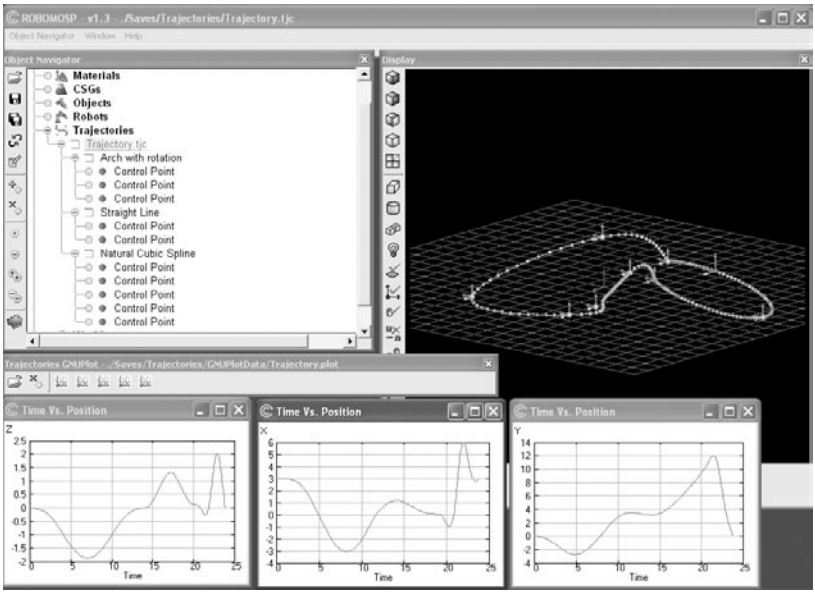

Figura 7. Visualización de una trayectoria en construcción y su comportamiento en GNUplot.

3.5. Modelado de Mundos

ROBOMOSP permite el diseño y modelado de un mundo de simulación (virtual environment) denominado Worlds, el cual agrupa los objetos, robots o trayectorias definidos por el usuario para realizar simulaciones de tipo

cinemático o dinámico. Cada elemento definido en un mundo es transformado internamente por ROBOMOSP en su versión paramétrica con el fin de establecer las relaciones espaciales y agilizar los cálculos requeridos al momento de una simulación. ROBOMOSP maneja internamente una lista indexada de los elementos de un mundo para acceder directa y rápidamente a cualquier componente o parámetro del mismo.

De la versión paramétrica de estos elementos se resalta la estructura \robot paramétrico» que tiene asociado un Teach Pendant (o terminal de enseñanza). Esta estructura se crea a partir de la indexación realizada sobre cada una de las articulaciones del robot, permitiendo al usuario modificar manualmente los parámetros de movimiento articular desde la interfaz gráfica de usuario.

En la figura 8 se pueden apreciar los elementos que conforman un mundo y la forma en que están relacionados.

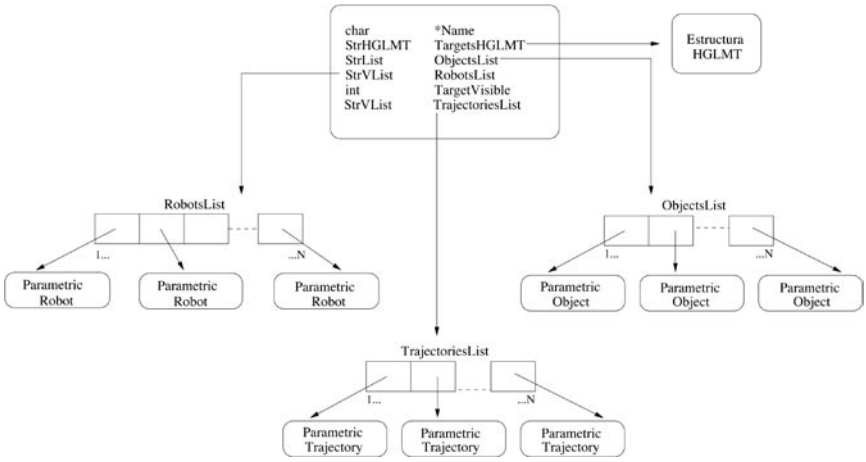

Figura 8. Componentes de un Mundo.

Es posible realizar simulaciones tridimensionales de tipo cinemático o dinámicoen cualquier mundo activo a través del subsistema robótico (ver sección 5).

4. PROGRAMACIÓN FUERA DE LÍNEA

ROBOMOSP incorpora un módulo compilador [19] que permite modelar y ejecutar acciones compuestas para los robots creados en su mundo, descritas en un lenguaje de alto nivel orientado a tareas. Como lenguaje para realizar

esta programación se eligió el Industrial Robot Language (IRL, especificado en el estándar DIN 66312 [6]). El compilador se incluye como módulo integrado a ROBOMOSP permitiendo al usuario ejecutar programas escritos en IRL y simular en el ambiente mundo las tareas asociadas con el código de usuario.

Tal como se puede apreciar en el siguiente fragmento de código, IRL posee características propias de un lenguaje de alto nivel estructurado e incluye particularidades que facilitan la especificación de tareas para un robot manipulador.

```
PROGRAM p_a_p; VAR
        INPUT BOOL:   ready at 1;
                      again at 2;
        OUTPUT BOOL: magnet at 2;
BEGIN
        IF NOT ready THEN HALT; ENDIF;
        R_ACC: = 300.0;
        R_SPEED: = 100.0;
top:
        MOVE LIN prepoint;
        MOVE LIN taking_up;
        magnet: = TRUE;
        MOVE LIN prepoint;
        MOVE LIN interm_point;
        MOVE LIN depot;
        magnet: = FALSE;
        MOVE LIN interm_point;
        IF again = TRUE THEN GOTO top; ENDIF;
        MOVE LIN home;
        HALT;
ENDPROGRAM
```

Los movimientos en IRL se componen de 4 partes:

- Tipo de movimiento. Se soportan dos tipos: absoluto, utilizando el comando MOVE; y el incremental, utilizando el comando MOVE INC.
- Movimiento. Se permiten: lineal (LIN), Punto a Punto (PTP) y Circular (CIRCLE) Cada uno de los anteriores se combina con el tipo de movimiento (Vg. MOVE INC LIN) para producir el resultado deseado.
- Camino de movimiento. El camino de los movimientos lineal y PTP puede ser construido a partir de un punto objetivo o mediante una secuencia de

puntos, definida como camino. Para el caso de movimiento circular, el camino está compuesto por una pareja de puntos, donde el primero indica el punto de control del movimiento circular y el segundo indica el punto de llegada del movimiento (objetivo).

- Parámetros de movimiento. No es obligatorio describir de manera explícita los parámetros de movimiento considerando que éstos son tomados automáticamente a partir de su definición en un archivo de especificación del sistema.

La versión implementada del lenguaje soporta un amplio número de tipos de datos que incluyen, además de los clásicos (numéricos, cadenas, caracteres, listas, archivos), tipos de datos propios de la especificación de robots manipuladores (robtarget, addjoint) y tipos de datos que facilitan el cálculo de posiciones y orientaciones en el espacio 3D (posición, orientación, pose). Adicionalmente, la implementación del lenguaje provee las operaciones necesarias para manejar estos datos.

La especificación del lenguaje incluye instrucciones para definir los movimientos de un robot y sus propiedades, soporta movimientos punto a punto, lineales y circulares. A través de la definición y manejo de señales, permite al programador controlar la interacción del robot con otros dispositivos que se encuentran dentro del mundo.

En una primera fase de compilación (front-end) el compilador genera un lenguaje intermedio (código de tres direcciones, adaptado a las características de IRL, y de los posibles lenguajes destino), este conjunto de instrucciones pasa a una etapa de postprocesamiento (back-end) la cual genera código en diferentes lenguajes destino, uno de ellos siendo ANSI C, el cual se utiliza para hacer el control de flujo del programa de usuario en comandos definidos por la interfaz IOCI que interactúa directamente con el subsistema robótico y el mundo de simulación de ROBOMOSP (ver figura 9).

La etapa de postprocesamiento es un proceso genérico que tiene como entradas el código intermedio generado por el compilador y un archivo que especifica la traducción del código intermedio al lenguaje destino deseado. De esta forma el compilador permite generar código final para diferentes robots (para control directo de manipuladores) y para ROBOMOSP .

4.1. Programación icónica de tareas

Otra alternativa de modelado y simulación de tareas soportada en ROBOMOSP se basa en la programación en línea mediante un interpretador de comandos.

El control de flujo y los parámetros de movimiento para una tarea particular se establecen mediante una interfaz icónica y la selección de objetos definidos en un mundo de simulación, tal como se especifica en el estándar internacional ISO-15187 [28]. Los comandos definidos tienen relación directa con aquellos soportados por la interfaz IOCI.

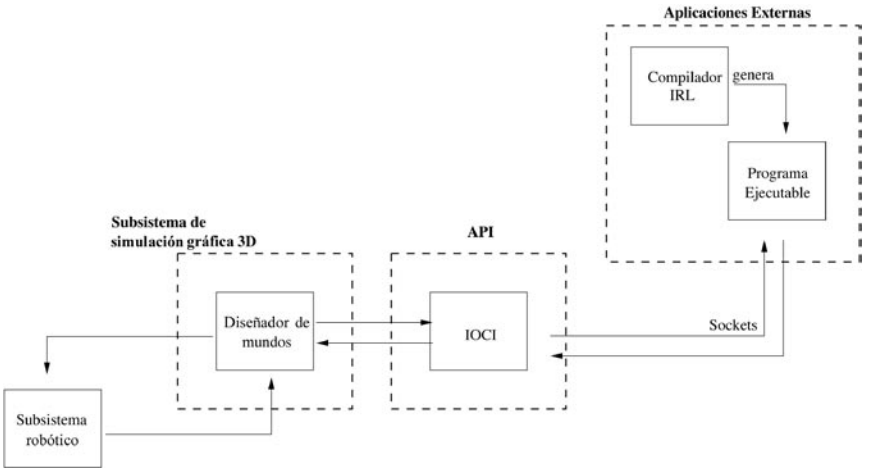

Figura 9. Integración del compilador IRL con ROBOMOSP.

5. SIMULACIÓN

5.1. Simulación cinemática

El subsistema robótico, cuyo componente principal se denomina LLRL (Low Level Robotics Library), contiene implementaciones genéricas para solucionar los problemas cinemático directo, ecuación (9), y cinemático inverso, ecuación (10).

La solución cinemática directa se emplea para simular movimientos espaciales a partir de un conjunto de valores de articulación. La implementación se basa en la multiplicación en secuencia de las matrices de transformación homogénea que definen las relaciones espaciales entre articulaciones ${}^{0}_{1..}A^{n-1}_{n}$, para encontrar la localización espacial del efector final respecto del sistema de referencia inercial, T_0^{n-1}.

$$A_1^0 \; A_2^1 \; \ldots \; A_0^{n-1} = T_0^{n-1} \tag{9}$$

La solución cinemática inversa se emplea para propósitos de control. En ella se encuentran los valores de articulación correspondientes a una posición y

orientación particular del efector final del robot. La solución genérica (para cualquier manipulador serial) incluida en ROBOMOSP está expresada en la ecuación (10).

Típicamente, para robots manipuladores de 12 o menos grados de libertad, la solución cerrada al problema expresado en la ecuación (9) es de tipo no determinístico (más ecuaciones que variables) por lo cual existen un número de soluciones proporcionales a la configuración cinemática del manipulador en cuestión.

$$\frac{dT}{dt}\Big|_{6x1} = J\frac{d\theta}{dt} \tag{10}$$

$$\frac{dT}{dt} = T_i - T_{i-1} = \begin{bmatrix} 0 & -\delta z & \delta y & dx \\ \delta z & 0 & -\delta x & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & & 1 \end{bmatrix} \tag{11}$$

$$\frac{dT}{dt}\Big|_{6x1} = [\, dx\ dy\ dz\ \delta x\ \delta y\ \delta z\,]^T \tag{12}$$

Donde $T \in \Re^{6x1}$ corresponde a la posición diferencial espacial del efector final (entre dos transformadas homogeneas $T_i - T_{i-1}$) expresada en forma vectorial, $\theta \in \Re^{nx1}$. El operador Jacobiano, $J \in \Re^{6xn}$, es utilizado para hallar la relación de movimiento diferencial entre el efector final, $\frac{dT}{dt}$, y las articulaciones que componen el robot manipulador, $\frac{d\theta}{dt}$. El método empleado es numérico y por lo tanto la garantía de convergencia dependerá de las condiciones iniciales de configuración y localización del manipulador.

ROBOMOSP permite, a través de una interfaz API, la definición de soluciones cinemáticas cerradas o numéricas por parte del usuario. Esta utilidad es importante considerando que las soluciones cerradas son más precisas y rápidas de calcular, favoreciendo la ejecución en tiempo real.

ROBOMOSP incluye soporte para transformaciones espaciales mediante matrices de transformación homogéneas, cuaterniones y ángulos de euler.

Las trayectorias articulares y espaciales, y demás parámetros cinemáticos pueden visualizarse a través de la interfaz gráfica o mediante gráficas de GNUPlot, (ver figura 10).

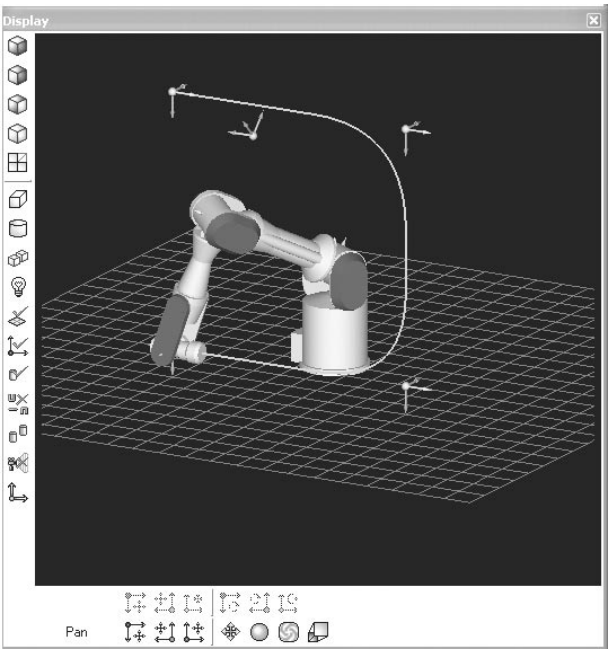

Figura 10. Simulación cinemática del recorrido de una trayectoria.

5.2. Simulación dinámica

La solución al problema dinámico de manipuladores se utiliza para determinar las fuerzas articulares que intervienen en un movimiento espacial dado (caso dinámico inverso) o para encontrar el movimiento espacial inducido por la aplicación de un conjunto de fuerzas articulares (caso dinámico directo). La solución implementada en ROBOMOSP se basa en la formulación en notación espacial de Newton-Euler [22].

5.2.1. Formulación de Newton-Euler para robots manipuladores seriales

Asumiendo una propagación de parámetros cinemáticos desde la base hacia el efector final (ver figura 11), las velocidades espaciales (los componentes traslacionales y rotacionales se apilan en vectores seis dimensionales) (V), para los cuerpos $i = 1 \ldots n$, se expresa como,

$$V_i = P^{T}_{i-1,i} V_{i-1} + HiQi \tag{13}$$

donde,

$$\widehat{P}_{i,i-1} = \begin{bmatrix} U & \tilde{P}_{i,i-1} \\ 0 & U \end{bmatrix}, \tag{14}$$

U corresponde a la matriz identidad $\in \Re^{3x3}$, $\tilde{P}_{i,i-1}$ a la representación en forma skew simétrica de un vector de distancia (equivalente al producto cruz), H_i denota la matriz de proyección de los ejes de movimiento y $\dot{Q}_i$ corresponde al vector de velocidades articulares. Diferenciando la ecuación anterior respecto de tiempo resulta en las aceleraciones espaciales ($\dot{V}$),

$$\dot{V}_i = \hat{P}^T_{i-1,i}\, \dot{V}_{i-1} + H_i \ddot{Q}_i + \dot{\hat{P}}^T_{i,i-1}\, V_{i-1} + \dot{H}_i \dot{Q}_i \qquad (15)$$

Una propagación inversa (o hacia abajo en la cadena) de los cuerpos $\boldsymbol{i} = \boldsymbol{n..1}$ de las fuerzas espaciales (F$\boldsymbol{i}$) de cada cuerpo define por completo las ecuaciones de movimiento,

$$F_i = \hat{P}_{i,i+1}\; F_{i+1} + I_i \dot{V}_i + \left[I_i + I_i \dot{\hat{S}}^T_{Oi,cm} \right] V_i \qquad (16)$$

donde, la matriz de inercia para el cuerpo $\boldsymbol{i}$, $\boldsymbol{I}_i \in \Re^{6x6}$, se obtiene a partir de la masa escalar y de los momentos de inercia con respecto a un punto de interés en el cuerpo. Se definen condiciones de frontera apropiadas según el tipo de robot en la base (fijo o flotante, para el caso de manipuladores espaciales) y para el cuerpo $\boldsymbol{n}$ en caso de existir fuerzas externas que afecten el efector final (condiciones de carga). Asumiendo para el cuerpo $\boldsymbol{i}$ una masa $\boldsymbol{m}_i$ y un tensor de inercia $\boldsymbol{J}_{i,cm}$ respecto de su centro de masa, $\boldsymbol{cm}$, el operador de inercia espacial se define por,

$$I_{i,cm} = \begin{bmatrix} J_{i,cm} & 0 \\ 0 & m_i U \end{bmatrix} \in \Re^{6x6} \qquad (17)$$

La inercia espacial, $\boldsymbol{I}_i$, medida sobre el punto de articulación, $\boldsymbol{O}_i$, está dada por la aplicación del teorema de ejes paralelos sobre la ecuación 17,

$$I_i = \begin{bmatrix} J_{Oi} & m_i \tilde{s}_{Oi,cm} \\ -m_i \tilde{s}_{Oi,cm} & m_i U \end{bmatrix} \qquad (18)$$

Finalmente, las fuerzas espaciales se proyectan sobre los ejes de movimiento del manipulador para obtener las fuerzas efectivas,

$$F_i = H_i^T P_i. \qquad (19)$$

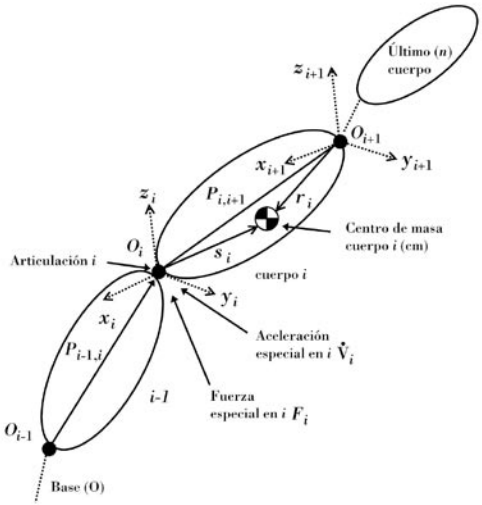

Figura 11. Sistema articulado serial de cuerpos rígidos.

5.2.2. Solución al problema dinámico directo

Para propósitos de simulación es necesario evaluar el comportamiento dinámico del sistema multicuerpo en su solución directa, es decir, determinar el movimiento inducido por un conjunto de fuerzas aplicadas sobre las articulaciones de un manipulador. La solución implementada en ROBOMOSP parte de las ecuaciones dinámicas planteadas en la sección anterior.

Para cada columna k_i $i = 1 \ldots n$ de la matriz de masa de cuerpo articulado M el vector de aceleraciones articulares, $\ddot{Q}i$ está dado por,

$$\ddot{Q}i = \begin{cases} 0 \; i \neq k \\ 1 \; i \neq k \end{cases} \tag{20}$$

Donde la columna ***ki*** de ***M*** se calcula por la aplicación del método de ***O*(*n*)** de Newton-Euler, descrito previamente, para encontrar las fuerzas equivalentes al conjunto de aceleraciones articulares requerido. A partir de las aceleraciones articulares se emplea un método de integración numérico (Runge-Kutta con tiempo de muestreo variable) para determinar las posiciones articulares en cada intervalo de tiempo de integración dinámica.

6. TRABAJO A FUTURO

La labor actual esta centrada en la adición de nuevas características a la herramienta, entre otras: la implementación del módulo de enseñanza gráfica en un mundo de simulación, la incorporación de un sistema de detección de

colisiones a partir de una representación gráfica de objetos en modo malla, el soporte de trayectorias libres de obstáculos presentes en un entorno de trabajo, la incorporación de objetos cinemáticos/dinámicos más complejos en en un mundo (Vg. maquinaria CNC), el soporte para filtros de importación/exportación de objetos gráficos, el desarrollo de librerias gráficas de robots comerciales y el soporte para herramientas intercambiables, animación en estereoscopía y el desarrollo de postprocesadores para lenguaje IRL en la generación de código ejecutable para controladores comerciales (PA-10 [18], ANSI C, KRL [29], MELFA [30], MRL [31], SRL [32], RAPID [33]).

7. OBSERVACIONES FINALES

Este artículo describe los módulos principales que componen la plataforma de modelado y simulación por software denominada ROBOMOSP desarrollada por el Grupo de Automática y Robótica de la Pontificia Universidad Javeriana de Cali.

La herramienta aporta componentes novedosos respecto de otras comercialmente disponibles, entre ellos: la solución a problemas dinámicos con cálculo automático de parámetros de masa a partir de objetos gráficos, la programación fuera de línea a partir de un lenguaje estándar (IRL), una interfaz API para experimentación de nuevos algoritmos por parte del usuario y el soporte de comunicación por sockets.

La herramienta multiplataforma (Linux, MacOSX, MS Windows) está disponible por requisición directa al grupo en gar@puj.edu.co.

8. REFERENCIAS

1. ROBOMOSP: Código de registro ante la Oficina de Protección de Derechos de Autor 2010081, Grupo de Automática y Robótica, 12/2002.
2. RoboWorks, Copyright 2000 Newtonium, http://www.newtonium.com/.
3. RoboAssist, Robot modeling and control package, New River Kinematics Inc,http://www.kinematics.com/robot/.
4. Easy-ROB3D TM , 3D Robot Simualation Tool, http://www.easy-rob.com/
5. Open Source TM , Open Source licencias aprobadas, http://www.opensource.org/
6. Deutches Institut for Normung, Industrial Robot Language (IRL), Alemania, 1996.
7. Antonio A. Matta, Wilber Perea, RobLab: Herramienta gráfica para el modelamiento de robots, Tesís de grado, Carrera de Ingeniería de Sistemas y Computación, Pontificia Universidad Javeriana - Cali, 2003.
8. Roger S. Pressman, Ingeniería del Software: un enfoque práctico, McGraw-Hill, España, 2002.
9. Barry Boehm, Using the WINWIN Spiral Model: A Case Study, Computer Magazine, 1998.
10. Jean-Guy Schneider and Oscar Nierstrasz, Software Architectures - Advances and Applications, Springer, United States, 1999.
11. GNU Public License, Descripción de la Licencia, http://www.gnu.org/copyleft/gpl.html.
12. John K. Ousterhout, Tcl and the Tk Toolkit, Addison-Wesley, Estados Unidos, 1993.
13. Jackie Neider and Tom Davis and Mason Woo, OpenGl Programming Guide The Oficial Guide to Learnig OpenGl, Addison-Wesley, Estados Unidos, 1993.
14. David E. Stewart and Zbigniew Leyk, Meschach Library version 1.2, School of Mathematical Sciences. Australian National University, Australia, 1994.
15. Jack Goldfeather and Henry Fuchs, Near Real-time CSG Rendering Using Tree Normalization and Geometric Pruning, IEEE Computer Graphics and Applications, 1989.
16. Joachim Sholber, NETGEN - An advancing front 2D/3D-mesh generator based on abstractrules, Comput. Visual. Sci, 1997.

17. Herwig Mayr, Virtual Automation Environments, Marcel Dekker Inc, New York, 2002.
18. PA-10 Portable General Purpose Intelligent Arm, Mitsubishi Heavy Industries, LTD.,5-1, Marunouchi 2-chome, Chiyoda-ku, Tokyo 100-8315, JAPAN.
19. Olver Ramos, Gloria Meneses, Maria Constanza Pabón, Compilador e Interfaz del lenguaje IRL (International Robot Language) para ROBOMOSP, Reporte Técnico GAR, Pontificia Universidad Javeriana, Cali, Colombia, 2004.
20. Brian Mirtich, Fast and Accurate Computation of Polyhedral Mass Properties, Journal of Graphics Tools, 1996.
21. J. Denavit and R.S. Hartenberg, A Kinematic Notion for Lower-pair Mechanisms Based on Matrices,Trans. of the ASME, J. of Applied Mechanics, 1954.
22. Andrés Jaramillo Botero, Teoría de diseño de brazos manipuladores robóticos para complejidad dinámica reducida, Revista Epiciclos, 2002.
23. King Sun Fu, Rafael C. González and C.S.George Lee, Robotics : control, sensing, vision, and intelligence, McGraw-Hill, Estados Unidos, 1987.
24. Antonio Barrientos, Fundamentos de robótica, McGraw-Hill, España, 1997.
25. Andrés Jaramillo-Botero, Juan Fernando Correa-Caicedo, Iván Javier Osorio-Parra, Capítulo: Planificación de trayectorias, documento de tesís de grado, Carrera de Ingeniería Electrónica, Pontificia Universidad Javeriana - Cali, 2004.
26. Burden, Richard L., Faires, J. Douglas, Análisis Numérico, International Thomson Editores, México, 1998.
27. Thomas Williams y Collin Kelley, GnuPlot: An interactive plotting program, Dartmouth University, 1998.
28. Manipulating industrial robots - Graphical user interaces for programming and operation of robots (GUI-R).
29. Kuka Roboter GmbH, Kuka Robotics Language.
30. MELFA, Mitsubishi Robots Programming Language.
31. MRL, Multiagent Robot Language.
32. SRL, Structured Robot Language.
33. RAPID, ABBA Robotics Language.