Investigación

La sección de Difusión de Investigación en Ingeniería, como su nombre lo indica, pretende divulgar el trabajo de investigación y desarrollo que se haga en esta Facultad y otras Facultades de Ingeniería del pais.

Esperamos que esta sección pueda servir para aumentar los mecanismos de comunicación de la comunidad cientifico-tecnológica en el país. Consecuentes con lo anterior invitamos a investigadores de otras universidades para que usen este espacio para divulgar resultados que sean de interés para un sector amplio de la ingeniería.

Jorge A. Villalobos S.

MSIM: Un Ambiente de Programación de Robots con Capacidad de Planificación de Tareas*

Resumen

Un robot industrial se puede programar de tres formas, cada una correspondiente a un nivel de abstracción distinto: de manera gestual, de manera textual explícita o de manera implícita. Un ambiente de programación de robots fuera de línea debe incluir un conjunto de herramientas que permita trabajar en cualquiera de los tres niveles, garantizando la compatibilidad entre ellos y entre los productos que se obtienen. MSIM, por su arquitectura y diseño, constituye una propuesta de solución al problema de programación de robots siguiendo este enfoque. Es un sistema modular, general, portable y con una estructura fácilmente extendible, que interactúa amigablemente con el usuario. Además de una plataforma de programación, MSIM se ha utilizado para estudiar los problemas informáticos de implementar de manera eficiente algoritmos, como los de planificación de tareas, muy pesados computacionalmente. Una presentación formal del trabajo se puede encontrar en (20) y en (13).

*Este trabajo ha sido parcialmente desarrollado por el Grupo de Robótica del Instituto de Cibernética (Universidad Politécnica de Cataluña) y parcialmente financiado por Colciencias en el marco del proyecto PIDAI (Universidad de los Andes).

Introducción

Según la difinición adoptada por la RIA (Robot Industry Association),un robot industrial es "un manipulador reprogramable multifuncional diseñado para mover materiales, piezas, herramientas, o artefactos especiales mediante movimientos variables programados para la ejecución de tareas, potencialmente muy diversas. "En la industria se utiliza sobre todo para la manipulación de materiales, carga de máquinas, pintura y sellado, soldadura, mecanizado, ensamblado e inspección (16).

JORGE VILLALOBOS: Ingeniero de Sistemas, Uniandes. DEA, Instituto Nacional Politécnico de Grenoble, Francia. Profesor del Dpto. de Ingeniería de Sistemas, Uniandes. Areas de especialización: Programación Orientada a Objetos, Programación de Robots.

La propiedad más importante de los robots dentro de la automatización industrial es su capacidad de ser reprogramados. Esto los hace flexibles y poderosos, porque se adapatan a las necesidades de producción sin que sea indispensable modificar físicamente las máquinas que elaboran el producto, como sí

ocurriría con la automatización rígida. Esta aparente ventaja se puede convertir en una desventaja si la labor de programación se vuelve demasiado dispendiosa o si los costos de reprogramar el robot resultan muy altos. Si es necesario detener una línea completa de producción durante varias semanas para hacer y probar un programa de robot, su utilización pierde rentabilidad y deja de ser una solución atractiva.

Los ambientes de programación de robots son un intento por resolver este problema (4) (17). Para esto reenfocan toda la ingeniería tradicional de software pensando en los problemas específicos de la programación de robots y creando herramientas de apoyo. MSIM fué concebido inicialmente en el Instituto de Cibernética de Barcelona (8) como una herramienta de programación fuera de línea. Actualmente ha evolucionado hasta convertirse en un ambiente de programación de robots, con un nuevo paradigma que permite al usuario manejar borradores de solución y moverse sobre diferentes niveles de abstracción en el momento de resolver cada una de las subtareas planteadas en la solución.

El ambiente está apoyado por un conjunto de planificadores de tareas que facilitan la labor de programación de acuerdo al esquema planteado anteriormente. Uno de los objetivos básicos del diseño fué conseguir una estructura general, para que el ambiente pudiera trabajar con un número considerable de robots industriales sin grandes cambios en la programación.

El objetivo del artículo es presentar de manera global el ambiente MSIM y se encuentra estructurado de la siguiente manera: primero se presenta la problemática de la programación de robots. Luego se muestra la funcionalidad y la arquitectura del ambiente propuesto como una respuesta a los problemas mostrados en la primera parte. Por último, se profundiza un poco en la manera de utilizar planificadores de tareas y lenguajes de alto nivel de abstracción como herramientas de apoyo a la labor de programación de robots. Para una presentación más formal y detallada del trabajo se recomienda consultar (20) y (13), donde se incluye una descripción de los principales algoritmos implementados en el sistema.

Programación De Robots

Desde la perspectiva del robot, un programa es una secuencia de posiciones y acciones de bajo nivel (abrir y cerrar la pinza, por ejemplo) que se le especifica de alguna manera, para que el robot resuelva un problema. El robot, utilizando diferentes algoritmos de control, envía a los motores de las articulaciones las señales adecuadas para alcanzar las posiciones pedidas y alterar de ese modo la escena o realizar algún otro tipo de acción como soldar o pintar. Desde la perspectiva del programador, esta visión puede cambiar un poco. Existen diferentes niveles de abstracción en los cuales él puede expresar la tarea que debe ejecutar el robot: desde la especificación física de cada posición, hasta la formulación de la tarea a través de dos situaciones una inicial y otra final, sin indicar de manera explícita la forma de pasar de la una a la otra. Estas dos formas nos definen los límites del espectro de posibilidades que se utilizan para programar robots: programación gestual y programación implícita. Además de estas, existe una manera intermedia conocida como programación textual explícita, equivalente de cierta forma a la programación imperativa en informática.

En la programación gestual el programador guía físicamente al robot para enseñarle la manera de ejecutar la tarea; el robot memoriza las posiciones y movimientos y luego los repite. Es la manera más fácil y económica de programar un robot, pero tiene muchas limitaciones.

En la programación textual explícita, el programador utiliza un lenguaje como VAL-II (15) o LM (7)con instrucciones de movimiento y, valiéndose de un compilador, genera las posiciones físicas que envía al robot. Es más general y poderosa que la programación gestual, pero todavía resulta insuficiente, como se muestra más adelante. Existen muchos lenguajes comerciales de programación explícita de robots y su utilización se encuentra bastante difundida. El siguiente es un fragmento de programa escrito en VAL-II, que toma una pieza y la lleva a otra posición, y puede dar una idea del tipo de instrucciones de estos lenguajes:

```
MOVE #PPOINT(0,90,90,60,45,30)
APPRO #POS1,-100
OPENI 50
DEPARTS -100
CLOSEI 20
MOVE #POS2
```

La programación implícita, por su parte, es el equivalente a la programación automática en informática y todavía se encuentra en fase de investigación. El programador especifica, con algún formalismo, la tarea que desea resolver (con dos escenas, por ejemplo), y el sistema, con algunas pautas generales y la descripción cinemática y geométrica del robot, genera las instrucciones de movimiento que resuelven la tarea. Esta última forma de programar es muy poderosa, porque es independiente del robot, fácil de mantener, rápida, etc., pero tiene varios problemas que no se han logrado resolver todavía de manera satisfactoria, sobre todo en lo concerniente a eficiencia. Existen versiones comerciales de implementaciones parciales de estos lenguajes (AUTOPASS (8),

LAMA (9)) pero, debido a las grandes restricciones que tienen, son muy poco utilizados comercialmente. El siguiente es un fragmento de código utilizado por AUTOPASS para ensamblar dos objetos y colocaralos sobre otro: especifica tres tareas, pero no se refiere en ningún momento a las posiciones que debe alcanzar el robot para resolverlas:

```
GRASP OBJETO1
INSERT OBJETO1 INTO OBJETO2
PLACE OBJETO2 ON OBJETO3
```

Independiente de la forma que se haya escogido para programar el robot existe un problema que no se ha considerado y que genera grandes complicaciones: la incertidumbre acerca del estado del mundo. En los ambientes industriales, que son bastante controlados y poco variables, esta incertidumbre disminuye, pero aún así, es uno de los factores más difíciles de manejar en el proceso de programación.(Qué pasa si la pieza que debe agarrar el robot se encuentra desplazada unos milímetros de donde debería estar?. El robot podría chocar contra ella y producir resultados inesperados y peligrosos para toda la línea de producción. Es indispensable, entones, que el robot esté recibiendo constantemente información sobre el estado del mundo y la manera como va evolucionando la tarea. Para esto se han integrado a los robots sensores de visión, de fuerza, de proximidad, de tacto, etc. que lo ayudan a identificar situaciones potencialmente peligrosas y poder, así, adoptar a tiempo las acciones correctivas necesarias. También puede utilizar la información sensorial para corregir en ejecución algunas posiciones físicas del programa que no corresponden con la realidad.

El problema es,¿cómo integrar esta nueva información en el momento de hacer el programa? En programación gestual no hay manera de hacerlo y, por eso, se considera tan limitada. En la programación textual explícita se debe escribir un programa aparte para el manejo de los sensores que se ejecute en paralelo con el programa del robot y que de forma sincronizada informe al otro lo que va percibiendo del mundo. Esto no solo no es sencillo, sino que en muchos casos es más complicado que la misma programación del robot, exigiendo al programador bastante conocimiento en estrategias de movimiento basadas en información sensorial y reduciendo por tanto el personal capacitado para llevar a cabo esta labor. En programación implícita existen varios enfoques distintos sobre la manera de utilizar la información sensorial, pero todos incluyen la generación automática del programa de sensores y su sincronización con el programa del robot. Algunos sistemas, como el que se presenta, van más allá y son capaces de replanificar partes de la tarea según las condiciones especiales que se vayan detectando mediante los sensores.

La programación de un robot se puede hacer en línea, utilizando directamente el robot, o fuera de línea, utilizando herramientas CAD, simuladores gráficos y validadores de programas. Por costo, comodidad y seguridad es más conveniente esta última opción. MSIM está diseñado para la programación fuera de línea de robots, utilizando los tres niveles de abstracción y permitiendo que se combinen los resultados obtenidos desde cualquiera de ellos.

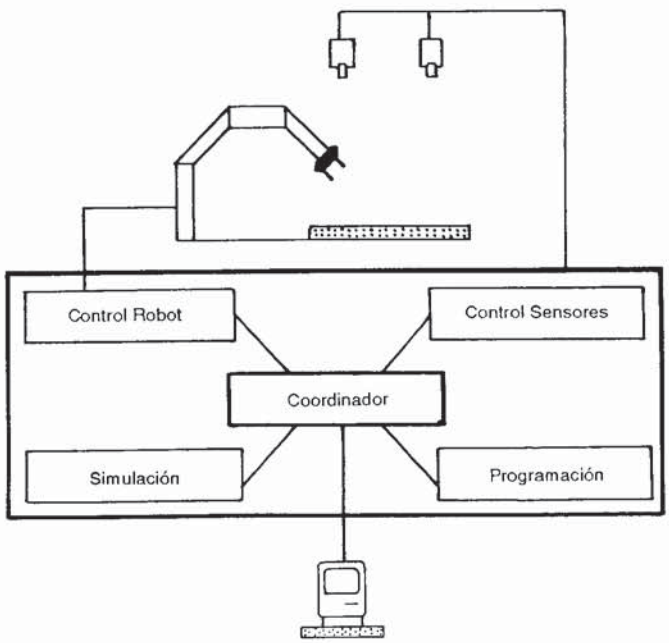


Figura 1. Estructura funcional de MSIM

Funcionalidad Del Ambiente

Desde un punto de vista puramente funcional, el ambiente está compuesto por cuatro módulos que se comunican y sincronizan a través de un coordinador, como se muestra en la figura 1.

El coordinador, además de interactuar con el usuario, mantiene un modelo actualizado del robot y de su entorno, y administra el flujo de información y de control entre los diferentes módulos. El módulo de comunicación con el robot es el reponsable de enviar los comandos adecuados al controlador físico del robot para que ejecute un movimiento. Este módulo es dependiente del robot

que se utiliza y se debe escribir para cada robot que se quiera manejar desde el ambiente. El módulo de sensores, por su parte, recibe peticiones del coordinador para establecer algún hecho del entorno y retorna la información pedida. Por ahora funciona con sensores monovaluados (retornan un solo valor, como puede ser la proximidad a un objeto), pero se busca también integrar sensores como los de visión sobre los cuales es necesario hacer reconocimiento de formas antes de responder al requerimiento del coordinador. El problema de este último tipo de sensores es el tiempo que gasta un algoritmo en reconocer, a partir de una imagen, la posición y orientación de un objeto de la escena, o alguna otra característica de dicho cuerpo. Es importante recalcar que toda la sincronización entre el robot y los sensores pasa por el coordinador.

Figura 2.-Estructura funcional del módulo de programación.

El módulo de programación tiene la estructura interna mostrada en la figura 2, que corresponde a los tres niveles de programación presentados anteriormente.

El programador puede trabajar de manera gestual apoyándose en el simulador, de manera textual explícita utilizando el editor de texto del ambiente o de manera implícita generando un plan global de solución y resolviendo cada subtarea utilizando los planificadores disponibles. En (13) aparece una propuesta de lenguaje de planes basado en rasgos y un esquema de interacción y traducción hacia VAL-II, incluyendo la manera de generar el programa de sensores. Se presenta también la forma de replanificar subtareas, cuando la información sensorial indica la inaplicabilidad de la solución encontrada anteriormente por un planificador. Por ejemplo, si la pieza que desea agarrar se encuentra rotada algunos grados y por esta rotación la pinza no puede alcanzar el punto de agarre que se había seleccionado (porque colisiona contra otra pieza del entorno, por ejemplo), el coordinador pide a los planificadores otras soluciones aplicables a las nuevas condiciones y continúa con la ejecución. Mediante la utilización del lenguaje de planes propuesto, el programador puede manejar borradores de solución y trabajar de lo abstracto a lo concreto facilitando de esta manera la labor de programación.

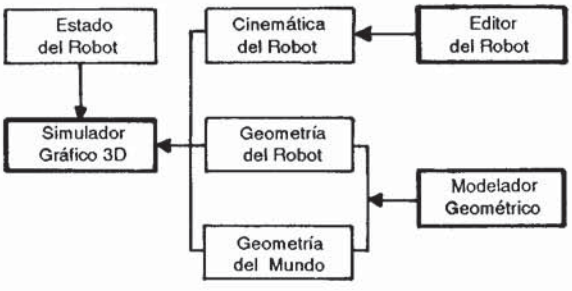


Figura 3.- Estructura funcional del módulo de simulación.

Por otra parte, existe el módulo de simulación y validación de programas que tiene la estructura funcional mostrada en la figura 3.

Este módulo permite al usuario visualizar de manera gráfica 3D (tres dimensiones) la ejecución de un programa de robot y verificar colisiones entre los elementos presentes en la escena. El simulador cuenta con el modelo cinemático del robot definido mediante los parámetros de Denavit-Hartenberg (16) y con los modelos geométricos del robot y de su entorno aproximados por poliedros de caras planas definidos con CGS (geometría constructiva de sólidos) y con una representación por la frontera (1). Toma como entrada código generado por el compilador de VAL-II y hace una simulación gráfica del movimiento del robot. Mantiene desplegada información del estado del robot, permite situar de manera interactiva al observador, tener visualización en

alambre o realista, colocar al observador en la pinza para visualización de aproximación y agarre, etc. Cuenta, además, con un sistema de ayuda en línea.

Arquitectura Del Ambiente

Desde un punto de vista informático, MSIM fué construído con una arquitectura de capas sobre un modelo orientado por objetos (12). Esto facilita incluir en él nuevas opciones y facilidades, sin necesidad de modificar lo ya existente .También permite manejar múltiples vistas del mismo modelo, según las necesidades, aumentando así las posibilidades de expresar problemas de diferentes tipos, en diferentes términos.

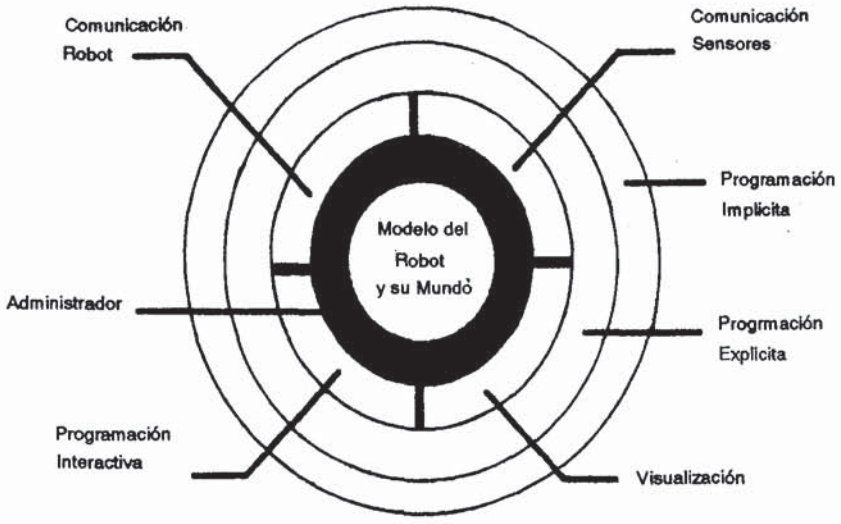


Figura 4.-Arquitectura informática de MSIM

El núcleo del sistema es un modelo operacional del robot y de su mundo, que simula su comportamiento. Eso quiere decir que responde exactamente igual a como respondería en la realidad el robot a estímulos externos (instrucciones del programador, choque con una pieza, etc.). Este modelo maneja un primer nivel de razonamiento geométrico (14) que permite al robot agarrar piezas, transportar ensambles decidir si hay una colisión, etc., y constituye el soporte de todo el ambiente.

Cubriendo este núcleo aparece una primera capa, que corresponde al dministrador del modelo, encargado de la interacción del núcleo del sistema con el robot físico, los sensores y el usuario, y cuya misión es garantizar en todo momento, la coherencia entre la situación actual del robot, lo que perciben los sensores del entorno del robot y lo que ve y quiere hacer el usuario.

La comunicación con el usuario se encuentra en la segunda capa del sistema, en relación directa con el administrador del núcleo; se puede dividir en dos partes: visualización y programación gestual o interactiva. La visualización mantiene informado al programador sobre el estado del modelo utilizando una representación gráfica 3D. De esta forma el usuario obtiene una visión dinámica, completa y actualizada de la escena, incluyendo la información obtenida por los sensores. La manipulación gestual permite al usuario operar directamente sobre el robot, enviarle órdenes de bajo nivel, mover sus articulaciones, manipular las piezas, etc. Esta capa toma los eventos generados por el usuario (mover un dial, oprimir un botón, etc.), los interpreta y comunica la orden al administrador.

Sobre la capa de manipulación interactiva se tiene la parte de programación explícita; está compuesta por un editor de programas y un compilador de VAL-II que traduce a código de más bajo nivel las instrucciones de movimiento dadas por el programador. Este código es enviado a la capa inferior como órdenes del usuario. Este compilador fué escrito utilizando YACC y LEX, herramientas UNIX para la generación de analizadores sintácticos y léxicos.

La última capa del sistema, de programación implícita, se encuentra localizada sobre la de programación textual explícita. Está compuesta por un conjunto de planificadores que generan código VAL-II y se explican en detalle en la siguiente sección.

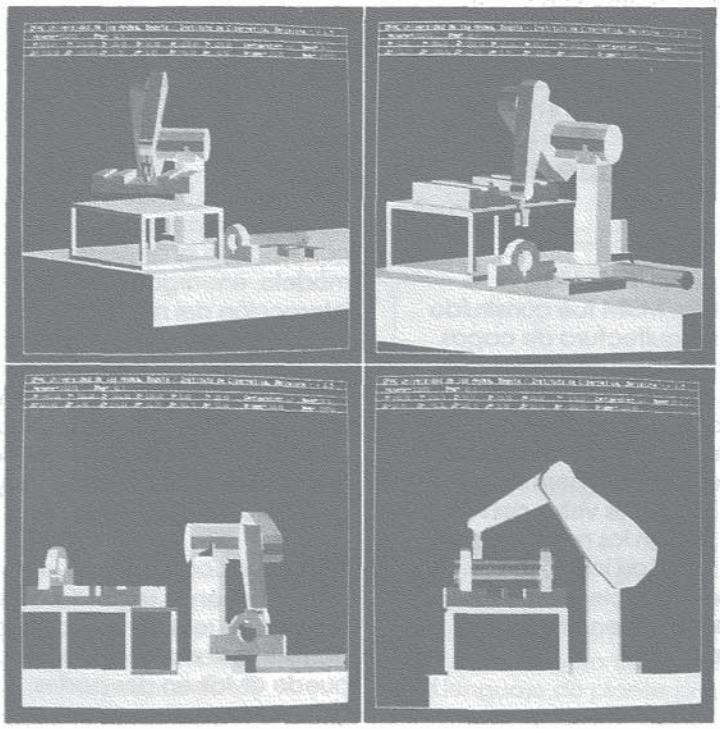
MSIM está escrito en C utilizando la librería gráfica PHIGS. Es completamente portable entre máquinas UNIX y actualmente se encuentra funcionando en estaciones gráficas Apollo, Sun e IBM- RT. Existe también una versión en C++2.0 y una versión sobre la librería gráfica CORE de Sun, utilizadas para estudiar los problemas de mantenimiento y portabilidad de ambientes gráficos orientados por objetos.

La Planificación Como Herramienta De Programación

Para escribir un programa en MSIM, el usuario sigue el siguiente proceso: crea en el editor de planes una primera aproximación de la solución -llamada un borrador o un bosquejo- en la cual, a grandes rasgos, resuelve el problema partiéndolo en subtareas más sencillas. Luego, comienza a resolver cada subtarea y a ajustar el plan global a medida que se vayan concretando las soluciones parciales, utilizando todas las herramientas del ambiente. En particular puede utilizar los planificadores que son capaces de sugerir de manera interactiva soluciones a tareas sencillas. Toda la coherencia entre el borrador y sus partes la garantiza el sistema, reescribiendo el borrador hasta llegar a la solución expresada en un lenguaje que se puede considerar de nivel implícito.

El ambiente cuenta con cuatro planificadores: dos para encontrar trayectorias libres de colisión (uno para movimientos grandes y otro para movimientos finos,) un planificador de agarre y un planificador de tareas simples.

El planificador de movimientos finos sin colisión está basado en la idea de potenciales. En este algoritmo se simula una fuerza de repulsión de los obstáculos y una fuerza de atracción hacia la posición final. El algoritmo está apoyado por heurísticas que impiden que el robot se quede en mínimos locales. La función de potencial fué una modificación de la propuesta en (6). Este planificador ha demostrado su eficiencia para hacer movimientos cortos pues genera trayectorias suaves y seguras, de manera rápida y a un costo computacional bajo.



Para los movimientos grandes, se cuenta con un planificador de trayectorias que trabaja sobre un espacio de configuraciones discretizado y aproximado(10) (11). La idea de fondo es cambiar el espacio de búsqueda de soluciones a un espacio generado por los tres primeros grados de libertad de robot (los de posicionamiento) aproximando los demás grados de libertad (los de orientación) a través de una esfera que los incluya. De esta manera el problema se reduce a encontrar la manera de mover un punto en un espacio de 3 dimensiones donde cada eje corresponde a los valores de una articulación. Este planificador requiere grandes recursos computacionales y toma algún tiempo en encontrar una solución debido, más que todo, a que los algoritmos para el cambio de espacio tienen altas complejidades. El planificador muestra gráficamente la solución encontrada y le presenta al programador toda la información asociada con la trayectoria, para que este pida la generación de código en VAL-II, solicite otra solución. o fije restricciones adicionales.

El planificador de agarre es capaz de sugerir al programador diferentes puntos para tomar una pieza, ya sea para ensamblarla o para transportarla (18)(2).Trabaja para pinzas con dedos paralelos y funciona buscando posiciones estables. Localiza todas las parejas de caras paralelas cuyas proyecciones de una sobre la otra se intersecten. Busca también parejas cara-arista que sean estables. Este planificador está implementado con un algoritmo de complejidad lineal y utiliza como estructura de representación interna una esfera de Gauss tal como se sugiere en (5). El planificador está completamenfe parametrizado, de manera que el usuario determina el nivel de precisión al cual debe trabajar, el tipo de soluciones que le interesa, el grado mínimo de estabilidad, etc.

Su funcionamiento es totalmente interactivo, en el sentido que después de establecer las condiciones exigidas por el usuario comienza a mostrarle gráficamente cada uno de los posibles puntos de agarre con su

respectiva dirección de aproximación libre de colisión. Cuando el usuario escoge una solución, el planificador genera el código VAL-II necesario para hacer ese agarre.

El planificador de tareas simples resuelve problemas de movimiento de piezas en la escena del robot (2). Para esto divide toda tarea en tres subtareas más sencillas: agarrar la pieza, transportarla y dejarla en el sitio pedido. Luego utiliza los otros planificadores para encontrar soluciones parciales y las combina hasta encontrar una solución viable a la tarea global. Tan pronto encuentra una solución se la presenta gráficamente al programador y, si éste lo desea, genera el código correspondiente en VAL-II. Su funcionamiento también es parametrizable, permitiendo al programador especificar el tipo de solución que busca o adicionándole restricciones. Puede, por ejemplo, pedir que el cuerpo sea agarrado únicamente por caras paralelas con un factor de estabilidad superior a algún valor, o que, para transportarlo, se busque la ruta más corta o la más segura, etc.

El Futuro De MSIM

En este momento se trabaja en varios frentes sobre MSIM. Como primera medida se están incorporando al ambiente dos nuevos robots diseñados en la Universidad y actualmente en proceso de fabricación (Andes-1 y Andes-2). Se está terminando la implementación del lenguaje implícito basado en rasgos descrito en (13) y mencionado en este artículo y se están diseñando nuevas facilidades en lo que respecta al manejo de borradores durante la fase de programación. Se trabaja en la implementación de algoritmos de reconocimiento de formas para incorporación de sensores no monovaluados. Se estudian nuevas estrategias de replanificación en línea (3), que sean más eficientes y más generales, para que funcionen en un mayor número de casos. Se está implantando otro nuevo planificador de trayectorias, que sea más preciso y que permita trabajar en ambientes altamente congestionados.

Conclusiones

En este artículo se presenta MSIM, un ambiente de programación de robots que permite trabajar al usuario en tres diferentes niveles de abstracción, cada uno correspondiente a una forma de programar un robot:programación gestual, programación textual explícita y programación implícita. Incluye además un nuevo paradigma para la programación de robots que permite al usuario manejar borradores de soluciones, apoyarse en planificadores para resolver algunas subtareas, y moverse sobre los tres niveles de programación para plantear la solución completa de un problema. Se muestra la estructura funcional e informática de MSIM como una propuesta general para este tipo de ambientes.

Además de la construcción de MSIM, el trabajo ha servido para estudiar y plantear algunas soluciones a los problemas informáticos típicos en ambientes gráficos para la programación fuera de línea de máquinas industriales. También ha permitido estudiar la forma de aplicar la metodología de programación orientada por objetos a ambientes gráficos dinámicos, en donde las necesidades de eficiencia son un factor determinante en la estructura del software.

Agradecimientos

El trabajo que se presenta en este artículo es el producto del esfuerzo de muchas personas entre las cuales se deben destacar el profesor Federico Thomas del Instituto de Cibernética y los ingenieros Eduardo Villegas, Jorge Morales y Oscar Ramos de la Universidad de los Andes.

Bibliografía.

(1) *Akman, V., Geometry and Graphics Applied to Robotics, Theoretical Foundations of Computer Graphics and CAD, Springer-Verlag, 1.988.*

(2) *Alami, R., Simeon, T., Laumond, J., A Geometrical Approach to Planning Manipulation Tasks. The Case of Discrete Placements and Grasps, septiembre 1989.*

(3) *Basañez, L., Operation Specialists for Automatic Programming and Monitoring of Robotic Assembly Cells, Journal of Robotics and Computer-Integrated Manufacturing, 1989.*

(4) *Bellier, C., VCP: Un Module Interactif de Verification et Correction de Programme en Robotique d'Assemblage, Rapport de Recherche. LIFIA, Institut IMAG, febrero 1990.*

(5) *Gatrell, L., CAD-Based Grasp Synthesis Utilizing Polygons. Edges, and Vertexes, IEEE, 1989.*

(6) *Hwang, Y., Path Planning Using Potential Field Representation, IEEE, 1988*

(7) *ITMI(Industrial Technology and Machine Intelligence). LM Reference Manual, 1984*

(8) *LLieberman, L., AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly, IBM Journal of Research and Development, Vol. 21, No. 4, julio 1977*

(9) *Lozano-Pérez, T., LAMA: A Language for Automatic Mechanical Assembly, M.I.T. 1977*

(10) *Lozano-Pérez, T., Spatial Planning: A Configuration Space Approach. IEEE Trans. on Computers, Vol. C-32, No. 2, febrero 1983*

(11) *Lozano-Pérez, T., A Simple Motion-Planning Algorithm for General Robot Manipulators, IEEE Journal of Robotics and Automation, Vol. RA-3, No. 3, junio 1987*

(12) Meyer, B., Object Oriented Software Construction, Prentice-Hall. 1988

(13) Morales, J., Un Lenguaje Implícito Basado en Rasgos para la Programación de Robots, Universidad de los Andes, 1991

(14) Pertin, J., Modelisation du Raisonnement Geometrique pour la Programmation des Robots, Tesis de Doctorado, Institut National Polytechnique de Grenoble, 1986

(15) Shimano, B., VAL-II: A New Robot Control System for Automatic Manufacturing, Proc. of the Int. Conference on Robotics, marzo 1984

(16) Silva, M., Introduction a la Robotica Industrial, Universidad de Zaragoza, 1989

(17) Simeon, T., Generation Automatique de Trajectoires sans Collision et Planification de Taches de Manipulation en Robotique, Tesis de doctorado, Universite Paul Sabatier de Toulouse, enero 1989

(18) Theveneau, P., Utilisation du Raisonnement Geometrique pour la Planification en Robotique d'Assemblage: Le Sysreme Pamela, Tesis de Doctorado, Institut National Polytechnique de Grenoble, 1988

(19) Thomas, F., Paquete Gráfico para la Programación Fuera-de-línea y Simulación de un Robot PUMA 560, Instituto de Cibernética. 1989

(20) Villegas, E., Algoritmos Eficientes para Resolver los Problemas de Visualización, Colisión y Planificación en un Ambiente de Programación de Robots, Universidad de los Andes. 1991