UN LENGUAJE IMPLICITO BASADO EN RASGOS PARA LA PROGRAMACION DE ROBOTS

Jorge E. Morales V.

Grupo de Investigación DFAC
Departamento de Ingeniería de Sistemas y Computación
Universidad de los Andes
Apartado Aéreo 4976 - Bogotá, Colombia

Resumen: En este artículo se describe un lenguaje de nivel implícito para la programación de robots en tareas de ensamblaje de piezas metalmecánicas. Es un lenguaje que utiliza, además de la información geométrica de las piezas, una descripción de sus características de ensamblaje (features), siguiendo el enfoque propuesto por [MANT90] y por [DEL91]. El proceso de programación en este lenguaje se plantea como un nuevo paradigma para la programación de robots y se aleja un poco del enfoque clásico: se parte de un borrador de solución y se va refinando, poco a poco, utilizando las herramientas de soporte del ambiente, hasta obtener una solución en términos de operaciones del robot, como lo hacen LAMA [LOZ76] y AUTOPASS [LIEB77]. La diferencia con estos lenguajes es el uso de características de los elementos involucrados en el ensamblaje, para que el proceso de refinamiento de la solución cuente con información adicional que lo guíe y poder así, restringir el espacio de búsqueda de la solución. Todo este trabajo está enmarcado dentro del desarrollo del ambiente para programación de robots MSIM [THO89] [VILLA91] [VILLE91].

Palabras Clave: Robótica, Programación de Robots, Lenguajes de Programación, Procesos de Fabricación, Modelaje por Rasgos

0. Introducción

En la actualidad, la programación de robots industriales para tareas de ensamblaje se hace a través de lenguajes explícitos, con instrucciones que describen los movimientos que debe realizar el robot para resolver un problema (tipo VAL-II [SHI84] o LM [ITM84]). También es usual que se utilice programación gestual, (aprendizaje y repetición). En los programas desarrollados con estos dos métodos, el contexto de la tarea se encuentra en la mente del programador y no se refleja en la sintaxis del programa. Además, requieren una amplia experiencia por parte del programador y se debe mantener el robot fuera de línea durante la fase de pruebas. Por estas razones, se han venido diseñando ambientes para la programación de robots donde de diferentes formas se ha tratado de apoyar la labor de programación [LOZ76] [LIEB77] [BAS89].

La arquitectura de estos ambientes va desde estructuras sencillas que manejan únicamente lenguajes de nivel explícito, hasta ambientes con herramientas de apoyo a la programación implícita, en los cuales se describe la tarea que se quiere resolver y no la forma de hacerlo. En los primeros, la programación se hace en términos de operaciones del robot; se cuenta con editores y compiladores que traducen las instrucciones de movimiento en movimientos del robot. El siguiente fragmento de código escrito en VAL-II hace que el robot se acerque a una pieza para agarrarla y muestra el tipo de instrucciones que manejan estos lenguajes:

```
move   #ppoint(30,25,25,30,45,45)
appro  pos[23],-50
departs -50
```

Los ambientes más sofisticados, por su parte, incluyen lenguajes que manejan instrucciones de más alto nivel y deben contar con un conjunto más extenso de herramientas para poder traducir dichas instrucciones en operaciones de movimiento del robot. En estos lenguajes solo se deben describir los pasos para ejecutar la tarea de ensamblaje [LOZ76][LIEB77], siendo así más fácil de entender la semántica del programa y más sencillo para el programador expresarse en dichos términos. El siguiente es un fragmento de programa en AUTOPASS para tomar una pieza e insertarla en otra:

grasp piston-pin
Insert piston-pin into piston-hole
ungrasp piston-pin

Los intentos que se han hecho para diseñar este tipo de lenguajes están basados en la descripción geométrica de los elementos activos en un proceso de ensamblaje, más un conjunto de restricciones a cumplir que expresan su estado final [LOZ76] [LIEB77] [BAS89]. Estos lenguajes no cuentan con ninguna información adicional que facilite el proceso de ensamblaje por parte del robot, lo que conlleva a algoritmos altamente combinatorios con tiempos de ejecución muy altos. Por esta razón, entre otras, su difusión a nivel comercial se ha visto muy limitada.

Este trabajo hace parte de un ambiente para la programación de robots [VILLA91] y describe un lenguaje que maneja la información geométrica de las piezas más sus rasgos o características de ensamblaje [MAN90][DEL91], que facilitan el proceso de traducción del nivel en que se expresa la solución de la tarea al nivel en el que se desarrollan las operaciones. El lenguaje propuesto está soportado por un ambiente que integra al desarrollo de programas de robots simuladores gráficos, planificadores, programación gestual, lenguajes explícitos y manejo de planes de solución. El objetivo principal de este trabajo es estudiar la factibilidad de este tipo de aproximaciones, por esto, se limita su alcance a procesos de ensamblaje de piezas metalmecánicas, ya que existen trabajos previos en modelaje de piezas por rasgos que facilitan su formalización [DEL91].

La estructura de este artículo es la siguiente: se presenta un conjunto de definiciones útiles para la compresión del tema y luego se muestra la propuesta de lenguaje de nivel implícito orientado por rasgos, incluyendo su sintaxis y su semántica.

1. Algunas Definiciones

A continuación, se presenta un conjunto de definiciones para formalizar los conceptos manejados por el lenguaje.

1.1. El Robot

Un robot industrial R es una secuencia de N cuerpos o eslabones rígidos (grados de libertad) en forma de cadena cinemática abierta simple, a través de N pares rotacionales o prismáticos. Está caracterizado por una descripción geométrica de cada

eslabón y por una descripción cinemática (parámetros de Denavit-Hartenberg [SIL89]).

1.2. El Mundo

Un objeto O del mundo está definido por su geometría (con respecto a un sistema de referencia local), por una transformación que lo ubica en el sistema de referencia del mundo y por un conjunto de rasgos del objeto, de interés en la labor de ensamblaje.

El mundo M es un conjunto de objetos, un conjunto de relaciones entre ellos (relaciones de apoyo, relaciones de encadenamiento, etc), un conjunto de relaciones entre los objetos y el robot y un sistema de referencia de base.

1.3. Una Escena del Mundo

Una escena E es el estado del modelo del mundo M en un momento dado. Un bosquejo de escena es un estado en el cual se pueden encontrar indefinidas algunas características del mundo.

1.4. Una Tarea de Ensamblaje

Una tarea de ensamblaje T es una transformación del mundo M en la cual solo se modifica la localización de los objetos, no su geometría ni sus propiedades. Está definida por dos escenas del mundo {E1,E2} que describen una situación inicial y una situación final de M.

1.5. Una Subtarea de Ensamblaje

Una subtarea de ensamblaje ST es una transformación del mundo M donde solo se modifica la localización de un objeto o algunas relaciones entre ellos expresada de forma no ambigua. Está definida por dos escenas del mundo {E1,E2}. ST puede ser cualquier transformación del mundo obtenida por una de las siguientes acciones de ensamblaje:

- agarrar
- soltar
- insertar
- mover
- localizar
- girar
- unir

1.6. Un Plan de Ensamblaje

Un plan de solución P para una tarea T especificada por {E1, E2} es una secuencia de subtareas ST1 ... STn tales que la situación inicial de ST1 es E1, la situación final de STn es E2 y la situación final de STi es igual a la situación inicial de STi+1. El programa de ensamblaje PE correspondiente a este plan está definido por la secuencia de acciones de ensamblaje A1 ... An asociadas con cada una de las subtareas.

1.7. Un Borrador de Plan

Un borrador de solución B para una tarea T especificada por {E1, E2}, es una secuencia de tareas T1 ... Tn especificadas mediante bosquejos de escena. Esto quiere decir que hay decisiones que el programador no ha concretado en su plan de solución.

1.8. Un Programa de Robot

Un programa del robot PR es una secuencia de operaciones de movimiento sobre un robot R expresada en un lenguaje explícito, (VAL-II [SHI84] o LM [ITM84]), que efectúa la tarea T definida por {E1, E2} siguiendo un plan especificado en un programa de ensamblaje P. Es decir, {E1} PR {E2}.

1.9. Un Rasgo de Ensamblaje

Un rasgo de ensamblaje RE es una característica de un objeto del mundo, útil para su manipulación por medio de un robot. Por ejemplo, los puntos de agarre de un objeto para una pinza dada, son interesantes para encontrar soluciones a un problema de ensamblaje.

1.10. Programación Implícita

Dado un modelo del mundo M, un robot R y una tarea T que describe el ensamblaje que se desea desarrollar, se debe construír un plan P de solución y un programa de robot PR que haga las transformaciones definidas en P para resolver T.

2. Descripción del Lenguaje

En esta sección se presenta una propuesta de un lenguaje implícito para la programación de robots basado en rasgis para ensamblaje mecánico.

Este lenguaje parte de las propuestas de [LOZ76] y [LIEB77] para desarrollar un lenguaje implícito, donde el programador se pueda expresar en términos de las acciones propias de tareas de ensamblaje, enriqueciendo el mundo con el concepto de rasgo para así, poder restringir el espacio de búsqueda de la solución. Adicionalmente, se incorpora el concepto de borrador de solución a partir del cual se inicia una fase de refinamiento del programa hasta llegar a una solución final.

Para ofrecer estas herramientas de desarrollo, es necesario un compilador donde el concepto de programador, como herramienta de apoyo al desarrollo de la solución, está ligado a cada uno de los procesos de la compilación. Además, este compilador cuenta con un conjunto de herramientas que se pueden agrupar según [BAS89] en:

1) Herramientas para agarrar, soltar y transportar.
Se encargan de planificar los movimientos para agarrar, soltar o transportar piezas o mover el robot en un ambiente libre de colisiones y cumpliendo con los requerimientos que se deseen (camino más corto, más seguro, rectilíneo, etc). Dentro de estas herramientas están los planificadores y el controlador para programación gestual.

2) Herramientas para construír una solución de la tarea.
Se encargan de coordinar el desarrollo del plan de solución a partir de un borrador para que no surjan conflictos entre las subtareas, por ejemplo, el haber tomado un objeto por el punto donde se debe insertar. Es importante tener en cuenta que una subtarea es altamente dependiente de las que le siguen, por lo que se debe hacer una coordinación global durante el proceso de construcción de la solución. Finalmente, estas herramientas generan, con ayuda del programador y a partir del plan de ensamblaje, un programa de robot que cumple con la tarea propuesta.

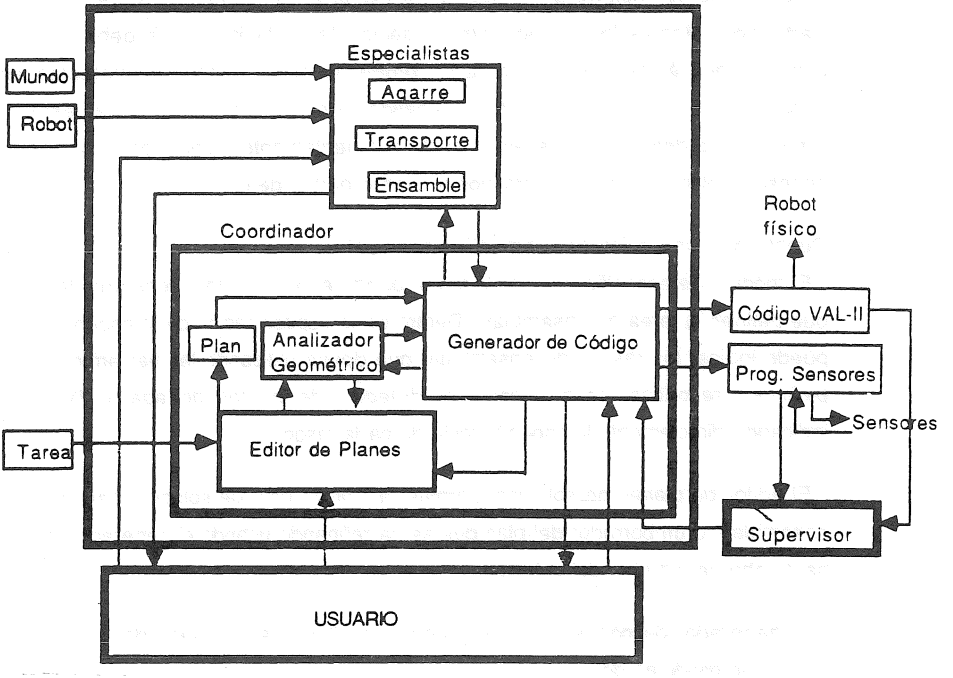
3) Herramientas para supervisar la ejecución del programa de robot.
Se encargan de coordinar el desarrollo de la ejecución, manejar información sensorial y permitir detección y corrección de errores en ejecución.

A continuación se presenta en detalle la estructura del compilador y cada uno de los módulos que lo componen. Luego, se presenta la funcionalidad del compilador y

finalmente se presenta una propuesta de sintaxis para el lenguaje de especificación de planes.

2.1. Estructura del Compilador

En esta sección se muestra la estructura operacional del compilador. Parte de la propuesta hecha en [BAS89] de un compilador compuesto por tres módulos principales: Coordinador, Supervisor y Especialistas, donde éstos últimos son submódulos para planeación y monitoreo de operaciones.



Las características de cada uno de estos módulos son:

2.1.1. El Coordinador

Trabaja desde la descripción de la tarea de ensamblaje en términos de {E1, E2}. Elabora el plan a desarrollarse basado en una descripción del borrador de la solución hecha por el programador o generada automáticamente usando una herramienta de razonamiento geométrico [PER86][LAUG87][THEV88]. Finalmente, toma el programa de ensamblaje asociado al plan y lo convierte en un programa de robot con ayuda de los especialistas y el programador.

Este módulo se encarga del control de la evolución de la solución durante la fase de traducción. Maneja los especialistas y decide las soluciones que debe adoptar del conjunto que éstos proponen para un problema dado. Cuando se detectan errores durante la planificación, decide la localidad del error para llamar al especialista respectivo o determina si el error es lo suficientemente global como para que sea necesario plantear en otros términos el plan o partes de él.

Cuenta con los siguientes submódulos principales:
- El módulo de especificación de la tarea: donde el programador construye las escenas que definen la tarea de ensamblaje. Dentro de la especificación de la escena el usuario puede indicar los rasgos de ensamblaje que desee utilizando las herramientas que le permiten reconocer rasgos como el detector de puntos de agarre [VILLE91] o indicando directamente la especificación de cada rasgo.

- El editor de planes de solución: permite definir un plan de solución para la tarea de ensamblaje o un borrador del plan que se va refinando usando el generador de código, hasta obtener un plan del solución.

- El generador de código: es el encargado de llevar la evolución de la solución. Su trabajo se divide en dos:
Cuando trabaja sobre un borrador de solución. Se encarga de guiar al programador hacia una especificación completa de cada tarea intermedia hasta obtener un plan de solución para cada una de ellas que sea compatible con el plan global. A este nivel detecta incompatibilidades y guía la replanificación de las tareas necesarias para mantener coherencia.

Cuando trabaja sobre una subtarea debidamente especificada, se encarga de traducir dicha tarea en un fragmento de programa de robot usando los especialistas e interactuando con el programador. En caso de no poder expresar la subtarea en

términos de operaciones de movimiento del robot, interactúa con el programador para replantear la subtarea en otros términos o modifcar el plan de solución desde tareas anteriores.

2.1.2. Los Especialistas

Son un conjunto de herramientas que se encargan de planificar movimientos del robot para apoyar el desarrollo de traducción de una subtarea en términos de operaciones de movimiento sobre el robot. Estos especialistas funcionan de manera semiautomática, planificando una labor específica hasta obtener una solución o fallar; en caso de presentarse una falla, consultan al programador, para que éste especifique la subtarea en otros términos o guíe la solución indicando operaciones intermedias. De ser necesario informan al generador de código la necesidad de replanificar una subtarea anterior. Hay tres especialistas en operaciones complejas:

2.1.2.1. El Especialista de Agarre

Se encarga de generar el conjunto de operaciones que permiten agarrar un objeto dado. Puede trabajar en dos modos: un modo exploratorio basado en una descripción incompleta de una acción de agarre o utilizando un punto de agarre definido a través de un rasgo. En el primer modo hace lo siguiente:

Analiza las características del estado actual del mundo y la forma como el objeto va a ser manipulado para determinar si existen puntos de agarre asociados al objeto que se puedan usar para solucionar el problema. Si la compilación se está haciendo en forma completamente automática, trata de solucionar el problema propuesto con cada punto de agarre en el conjunto. Si el programador está interactuando con el especialista, presenta el conjunto de puntos para que éste escoja el punto con el cual desea desarrollar la subtarea. Finalmente pasa a la fase de planificación de la trayectoria para el acercamiento al punto. Si no es posible solucionar el problema usando el conjunto de puntos disponibles, usa el conjunto de restricciones del mundo y el detector de puntos de agarre [VILLE91] para determinar otros posibles puntos. Luego, los puntos viables los califica por algún criterio (estabilidad, etc) y la lista se incluye en el conjunto de puntos de agarre del objeto, para repetir el proceso de selección de un punto.

El otro modo de operación es partiendo de la definición del punto de agarre por parte del usuario. Con este mecanismo evalúa la viabilidad de agarrar el objeto por dicho punto y pasa a planificar la trayectoria de acercamiento.

En planificación toma información del mundo para construír el espacio de búsqueda y planear los movimientos para llegar hasta el punto de agarre usando un planificador de movimientos finos [VILLE91]. El resultado es una solución al problema de agarre para que el coordinador a través del generador de código o el programador, escoja si es la que le conviene para su tarea global. En el código de nivel explícito generado para desarrollar la operación incluye un esquema de retroalimentación basado en información sensorial obtenida durante la ejecución.

En caso de detectar un error durante esta planificación (no se puede agarrar por ese punto porque está obstruído, por ejemplo), retorna a la fase de selección de un punto en modo exploratorio.

Finalmente durante la ejecución, el especialista está listo para corregir errores locales haciendo una replanificación de la trayectoria o detectar otros puntos de agarre (puede entrar en modo de operación automática durante la replanificación).

2.1.2.2. El Especialista de transporte

Este especialista se encarga de generar trayectorias libres de colisión para llevar el manipulador hasta los puntos de agarre o de ensamble, con o sin objeto en la pinza. Esta labor la hace usando una representación discreta del espacio de configuraciones del manipulador [LOZ83][LOZ87][MOR91] para los movimientos gruesos y un planificador por potenciales [VILLE91] para movimientos finos. En caso de no encontrar solución a un problema de planificación, interactúa con el programador para que éste indique puntos intermedios que faciliten la planificación automática o que explícitamente indique la ruta que se debe seguir para cumplir la trayectoria.

En tiempo de ejecución está pendiente de replanificar para corregir errores generando otras posibles soluciones.

2.1.2.3. El Especialista de Ensamble

Se encarga de ensamblar las piezas en la posición final requerida. Puede operar en dos modos: Un modo exploratorio, donde busca las posibles formas de ensamblar los elementos, usando una herramienta que hace razonamiento geométrico para determinar una forma de ensamblaje entre dos elementos definidos por una o varias piezas [PER86][LAUG87][THEV88]. Pasa luego a la fase de planificación de movimientos usando un planificador de movimientos finos [VILLE91]. El otro modo de operación parte de las superficies de ensamble definidas por el programador y a partir de allí planifica la trayectoria a seguir para ensamblar los elementos.

Al planificar desarrolla una serie de alternativas para usar en caso de detección de errores. De no encontrar una alternativa viable, el programador puede sugerir soluciones o indica explícitamente la forma de hacer el ensamblaje. De no existir solución, informa al generador de código para el replanteamiento del plan. Además, utiliza la herramienta de razonamiento geométrico para determinar evoluciones en la escena, tales como nuevas relaciones de encadenamiento de objetos, apoyo, etc.

Este especialista incluye puntos de chequeo sensorial dentro del código generado para que en ejecución se tomen alternativas o se replanifique sobre errores de ser necesario.

2.1.3. El Supervisor

El supervisor es el encargado de llevar el control de ejecución del programa del robot en tiempo real. Recibe toda la información disponible en los sensores, la analiza y determina la mejor respuesta para un requerimiento dado. Se encarga de mantener actualizado el modelo del mundo para poder replanificar a partir de cualquier punto de la solución. En caso de detección de errores, se encarga de interrumpir la ejecución de la tarea y darle la información necesaria al controlador para que replanifique a partir del punto donde se detectó el error.

2.2. Funcionalidad del Compilador

El compilador globalmente se puede visualizar como dos grandes módulos: primero el especificador, encargado de la generación de un borrador del plan de solución y segundo el generador de código explícito, encargado de refinar e interpretar dicho plan hasta obtener una solución a la tarea especificada en términos de operaciones del robot.

2.2.1. Especificación del Plan de Solución

Se hace a través de los submódulos del módulo Controlador del compilador. Usa el lenguaje para especificar borradores de solución al problema de ensamblaje o para especificar un plan de solución completo. Para especificar un borrador, el programador puede dejar sin especificar elementos de las instrucciones, por ejemplo GRASP OBJ1, donde no se especifica el punto de agarre por el cual se debe cumplir la labor, de tal forma que no está indicando completamente la transformación del mundo. También puede omitir operaciones, bien para que el compilador las genere automáticamente o para especificarlas durante la traducción, por ejemplo INSERT OBJ1 INTO OBJ2, donde no se especifica la operación de agarrar el OBJ1.

Para obtener el borrador del plan general de solución, el programador puede hacer uso de la herramienta de razonamiento geométrico [PER86][LAUG87][THEV88] para que se genere un esqueleto del plan de solución y refinarlo durante la interpretación; de esta forma, es posible generar varios planes a partir de un borrador inicial, para que se puedan analizar las ventajas de unos y otros antes de escoger la solución que se desea utilizar.

Para obtener el programa final del robot el programador puede hacer evolucionar el borrador de tres formas diferentes:
- trabajar secuencialmente en la obtención de la solución. Es decir, para cada una de las tareas que definen el borrador, especificarla hasta poder obtenerla en términos del lenguaje explícito y solo al finalizarla pasar a la siguiente tarea.

- refinar el borrador iterativamente hasta obtener el plan de solución y solo después de completarlo empezar la traducción de cada subtarea en código explícito.

- combinando los dos anteriores, de tal forma que inicialmente compila las tareas bien definidas y con base en éstas refina la especificación de las demás hasta completar la solución.

2.2.2. Interpretación de un Plan de Solución

Para pasar a la fase de interpretación de un plan de solución no es necesario que el plan esté completamente especificado, sino que se puede partir de un borrador y refinarlo. El generador de código cuenta con los especialistas para traducir las subtareas en operaciones del lenguaje de nivel explícito porque, en general, cada acción de ensamblaje corresponde a un conjunto de operaciones sobre el robot [LOZ76]. Posee un conjunto de herramientas para la labor de traducción, adicionales a las que ofrecen los especialistas, como son:
- Un planificador de tareas sencillas para completar un borrador con las subtareas omitidas por el programador. Este planificador es bastante simple. Chequea los prerrequisitos de cada subtarea e incluye antes las acciones que se omitieron para que el intérprete las complete según el modo de operación que se esté usando.

- Herramientas para programación gestual: modo *teach*, movimiento de las articulaciones, movimientos en coordenadas del mundo, de la pinza, etc.

- Simulador interactivo de las operaciones sobre el robot.

Para iniciar la traducción de una subtarea, primero se completa la especificación en forma automática o según especificación del programador. Es decir, si el programador

Indica el formato completo de la subtarea, se pasa a la fase de generación de código; pero si hay omisiones en la especificación, se usan los especialistas, de tal forma que no se hace una planeación particular para una subtarea muy específica, sino que se exploran todas las posibles soluciones a la subtarea y se escoge alguna en forma automática o por decisión con el programador.

En caso de que falten subtareas prerrequisito de la tarea, por ejemplo una operación de agarre para desarrollar una operación de ensamble, se usa el planificador de tareas sencillas para especificarlas y se interpretan antes de la subtarea actual.

La labor de interpretación de una subtarea se puede hacer de tres modos diferentes dependiendo de lo que desee el programador:

1) **Modo Automático**: El programador especifica al intérprete una instrucción dada. Este cuenta con un formato de traducción de cada subtarea (instrucción) en un conjunto de operaciones del lenguaje explícito más las pre y postcondiciones de cada una de ellas [LIEB77]. Si el estado del mundo permite la ejecución de la acción entonces entra a tratar de traducir dicha acción en un conjunto de operaciones expresadas en el lenguaje explícito, haciendo uso del formato especificado para la subtarea. En caso de no encontrar una solución posible a la acción especificada, entonces interactúa con el programador para que este guíe la solución indicando pasos intermedios, replanteando la acción de otra forma o pasando a otro modo de operación del intérprete y tomar parte activa en la solución de la subtarea. Esta replanificación por parte del programador puede implicar modificar o replanificar subtareas anteriormente interpretadas.

2) **Modo Explícito:** El programador, en este modo, toma el control de la traducción y especifica directamente un conjunto de operaciones en el lenguaje explícito que cumplen con la acción deseada.

3) **Modo Gestual:** El programador toma control directo del modelo del robot en el ambiente y lo manipula modificando los valores articulares. En este modo cuenta con la posibilidad de enseñarle posiciones al manipulador y grabarlas, incluyéndolas en la solución.

Durante el proceso de interpretación de una acción el programador puede hacer uso de las herramientas de que dispone en cualquiera de los tres modos como son especialistas, planificadores, modo teach, etc. y enriquecer las definiciones en los objetos (rasgos). Independientemente del modo bajo el cual el programador esté operando, siempre tiene una concepción global de los elementos del modelo como son

objetos, manipulador y rasgos e igualmente puede referirse a ellos a lo largo de la compilación desde cualquier nivel (plan, explícito o gestual).

2.3. Lenguaje de Especificación de la tarea

El lenguaje de Especificación de la tarea está dividido en dos: La parte declarativa que especifica las características de la escena y una parte operativa que describe el conjunto de acciones necesarias para desarrollar la tarea.

Globalmente, para definir el lenguaje de especificación de la tarea, se tomó en cuenta que la sintaxis del lenguaje puede ser sujeta a cambios para incluír el lenguaje dentro de un lenguaje informático ya existente, de tal forma que se puedan usar las facilidades que éste ofrece como son control de flujo, declaración de variables auxiliares, etc.

A continuación se presenta el conjunto de instrucciones básicas para la especificación del plan de solución:

INSERT Obj1.<nombre rasgo SE> **INTO** Obj2.<nombre rasgo SE> /

INSERT Obj1.<nombre rasgo SE> **IN** Obj2.<nombre rasgo SE> /

GRASP Obj.<nombre rasgo PA> /

FREE Obj /

PLACE Obj1.<nombre rasgo SS> **ON** Obj2.<nombre rasgo SS> /

FASTEN Obj1 **WITH** Obj2 /

UNFASTEN Obj1 **FROM** Obj2 /

NAME Ob1,...,Objn **ASSEMBLY** Objm /

TURN Obj **AROUND** <EJE> <numero de grados> /

MOVETO <PUNTO> /

MOVETO <TRANSFORMADA>

2.4. Tipos de Rasgos

Debido a que la cantidad de rasgos depende del alcance que se pretenda dar al sistema y de la variedad de objetos que se le permitirán ensamblar, definir el conjunto de

rasgos de un sistema es un problema para el cual no existe una metodología [MAN90]. Por lo tanto, la especificación de rasgos que se hace a continuación parte del conocimiento adquirido en el área de ensamblaje y diseño por rasgos [DEL91]. Se especifican tres tipos de rasgos que se seleccionaron tomando en cuenta la estructura definida para el compilador. Es decir, están involucrados en cualquier tipo de ensamblaje mecánico y se reconocen a partir del conjunto de acciones que el compilador debe realizar para interpretar una subtarea.

2.4.1. Rasgo: Punto de Agarre (PA)

2.4.1.1. Descripción:

Un punto de agarre se define como un punto asociado a un objeto, tal que por ese punto el objeto puede ser agarrado por un robot cumpliendo con los requerimientos de estabilidad necesarios. Es importante indicar que son dependientes no solo del objeto sino también del tipo de pinza con que cuente el robot para tomar el objeto.

2.4.1.2. Caracterización:

Un punto de agarre se especifica con:
- Una transformada: Especifica la transformación que define la posición en la cual debe quedar el sistema de referencia de la pinza sobre el objeto.

- Una distancia: Especifica un punto situado a dicha distancia sobre el eje Z del sistema de referencia de la pinza. A partir de este punto se debe desarrollar la planificación de movimientos para GRASP.

- Una apertura: Especifica la apertura de entrada para acercamiento.

2.4.2. Rasgo: Sección de Ensamblaje (SE)

2.4.2.1. Descripción:

Una sección de ensamblaje es un conjunto de elementos geométricos que pertenecen a un mismo objeto y que corresponden a una sección que entra en contacto (se ensambla) con otro objeto durante el proceso de ensamblaje.

2.4.2.2. Caracterización:

Una sección de ensamblaje se caracteriza globalmente por:

- Una transformada: Especifica la transformación que define la posición en la cual debe quedar el sistema de referencia de la otra sección de ensamblaje involucrada en el ensamble.

- Un conjunto de objetos: define el conjunto de objetos con los que se puede ensamblar dicho objeto por esa sección de ensamblaje.

2.4.3. Rasgo Sección de Soporte (SS)

2.4.3.1. Descripción:

Una sección de soporte se define como conjunto de elementos geométricos que componen una superficie de un objeto, sobre los cuales se puede apoyar el objeto en forma estable para colocarlo sobre una superficie base. La misión principal de estas secciones de soporte es apoyar las labores de ensamblaje donde sean necesarios pasos intermedios para tomarlo de otra forma.

2.4.3.2. Caracterización:

Una SS se caracteriza por:
- Una transformada: Especifica la transformción que define la posición en la cual debe quedar el sistema de referencia de la superficie que se vaya a usar como base.

4. Conclusiones

En este artículo se presenta un lenguaje de nivel implícito orientado por rasgos que pretende apoyar la programación de robots para tareas de ensamblaje. Este lenguaje parte de un nuevo paradigma de programación propuesto en [VILLA91] y que pretende integrar al lenguaje características del ambiente para facilitar con esto la programación de robots, ofreciendo la mayor cantidad de herramientas disponibles para dicha labor.

Es importante aclarar que el uso que se le de a una herramienta de estas características está sujeto a una metodología a partir del paradigma propuesto y cuya especificación debe ser el siguiente paso en la búsqueda de una nueva forma de programación de robots.

Agradecimientos

El trabajo que se presenta en este artículo es producto del esfuerzo conjunto de un grupo de trabajo en el área de robótica en la facultad de Ingeniería de la Universidad

de Los Andes, del cual se debe destacar al profesor Jorge Villalobos S. y al ingeniero Eduardo Villegas J.

Bibliografía

[BAS89] Basañez, L., Operation Specialists for Automatic Programming and Monitoring of Robotic Assembly Cells, Journal of Robotics and Computer Integrated Manufacturing, 1989.

[DEL91] De la Rosa, F., Una Propuesta de Diseño Asistido por Computador de Piezas Metalmecánicas Representados por Rasgos, Universidad de Los Andes, Tesis de Magister, Junio 1991.

[ITM84] ITMI (Industrial Technology and Machine Intelligence), LM Reference Manual, 1984.

[LAUG87] Laugier, C., Raisonnement Geometrique et Methodes de Decision en Robotique. Aplication a la Programmation Automatique des Robots, Tesis de doctorado, Institut National Polytechnique de Granoble, diciembre 1987.

[LIEB77] Lieberman, L., AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly, IBM Journal of Research and Development, Vol. 21, No. 4, Julio 1977.

[LOZ] Lozano-Perez, T., Lama: A language for Automatic Mechanical Assembly, MIT.

[LOZ76] Lozano-Perez, T., The design of a Mechanical Assembly System, MIT, 1976.

[LOZ83] Lozano-Perez, T., Spatial Planning: A Configuration Space Approach, IEEE Trans. on Computers, Vol. C-32, No. 2, febrero 1983.

[LOZ87] Lozano-Perez, T., A Simple Motion-Planning Algorithm for General Manipulators, IEEE Journal of Robotics and Automation, Vol. RA-3, No. 3, junio 1987.

[MANT90] Manthila, M., A Modeling System for Top-Down Design of Assembled Products, IBM Journal Res. Develop, Vol. 34, No. 5, septiembre 1990.

[MOR91] Morales, J., Planificador de Trayectorias basado en Espacios de Configuraciones Discretizados, Universidad de Los Andes, Informe Interno, 1991.

[PER86] Pertin, J., Modelisation du Raisonnement Geometrique pour la Programmation des Robots, tesis de Doctorado, Institut National Polytechnique de Grenoble, 1986.

[SHI84] Shimano, BI, VAL-II: A New Robot Control System for Automatic Manufacturing, Proc. of the Int. Conference on Robotics, marzo 1984.

[SIL89] Silva M., Introducción a la Robótica Industrial, Universidad de Zaragoza, Mayo 1989.

[THEV88] Theveneau, P., Utilisation du Raisonnement Geometrique pour la Planification en Robotique d'Assemblage: Le Systeme Pamela, Tesis de Doctorado, Institut National Polytechnique de Grenoble, 1988.

[THO89] Thomas, F., Paquete Gráfico para la Programación Fuera-de-línea y Simulación de un Robot PUMA 560, Instituto de Cibernética, Barcelona, 1989.

[VILLA91] Villalobos, J., MSIM: Un Ambiente de Programación de Robots con Capacidad de Planificación de Tareas, Universidad de Los Andes, 1991.

[VILLE91] 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.