(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 685 250**

(51) Int. CI.:

G06F 8/52 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **11.05.2006 E 06113817 (8)**

(97) Fecha y número de publicación de la concesión europea: **04.07.2018 EP 1855194**

(54) Título: **Sincronización de un programa gráfico y un programa de robot**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
08.10.2018

(73) Titular/es:

ABB SCHWEIZ AG (100.0%)
Brown Boveri Strasse 6
5400 Baden, CH

(72) Inventor/es:

LÖNNEMARK, GUNILLA;
MURPHY, STEVE;
NOHRE, RAGNAR;
SKOGLUND, NIKLAS y
WADENHOF, DANIEL

(74) Agente/Representante:

ISERN JARA, Jorge

DESCRIPCIÓN

Sincronización de un programa gráfico y un programa de robot

Campo de la invención

La presente invención se refiere a un dispositivo y un método para sincronización de un programa de robot y un programa gráfico.

La invención es útil en relación con programación fuera de línea y simulación de un robot industrial y en particular cuando la simulación contiene una descripción gráfica de cómo el robot realizará su tarea y el controlador de robot contiene un lenguaje textual para describir cómo el robot realizará su tarea.

Descripción de la técnica relacionada

Los robots industriales son dispositivos altamente flexibles usados para una amplia diversidad de operaciones en muchas aplicaciones industriales diferentes. Los robots industriales se programan de manera conveniente mediante un lenguaje de programación de robot que es muy similar a los lenguajes de programación informáticos convencionales. El lenguaje de programación de robot incluye una secuencia de instrucciones donde cada instrucción le indica al controlador de robot lo que ha de hacer y cómo hacerlo. Los lenguajes de programación de robot típicos incluyen instrucciones para el movimiento de robot, manejar señales de entrada y salida, comentarios de programa y manejar el flujo de programa: bucles y condiciones. Cada fabricante de robot industrial tiene su propio lenguaje de programación de robot.

Los controladores de robot convencionales proporcionan un dispositivo de edición fijado al controlador de robot para dar instrucciones al robot y para grabar y editar la secuencia de instrucciones del programa de robot. Las secuencias de instrucciones se graban de manera conveniente en un fichero de ordenador para almacenamiento fuera de línea y para edición fuera de línea. En la forma fuera de línea el programa de robot incluye uno o más ficheros que contienen instrucciones con argumentos a cada instrucción que detallan la operación que se ha de hacer en cada etapa. Además de las instrucciones, puede haber datos asociados con cada instrucción, por ejemplo, el punto hasta el que se ha de mover el robot puede almacenarse en algunos casos en la instrucción, o, en otros casos, almacenarse en una sección de datos del programa y hacerse referencia por instrucciones específicas.

La programación de robots es un proceso que lleva tiempo y los métodos convencionales de uso del robot durante el proceso de programación y enseñanza vinculan el equipo de producción y retarda el inicio de la producción. Para ahorrar tiempo y velocidad de inicio de la producción, es altamente deseable programar un robot fuera de línea. Convencionalmente, esto se hace a través de una simulación gráfica fuera de línea. La simulación contiene una representación en 3D gráfica del robot y su entorno así como un medio gráfico de enseñanza y grabación de las operaciones y movimientos del robot. La simulación gráfica proporciona un método mucho más natural y fácil para programar y visualizar un robot industrial, sin vincular el equipo real. Además, el entorno gráfico permite una independencia del lenguaje de programación del robot usado por el fabricante de robots. Sin embargo, las simulaciones en 3D gráficas no tienen la capacidad de representar todas las características que están disponibles en el lenguaje de programación de robot real. Por ejemplo, las simulaciones en 3D no muestran el flujo de programa, bucles y condiciones, ni comentarios en el programa. La mayoría del manejo de entrada-salida tampoco se muestra en la simulación en 3D.

La salida del trabajo con la simulación gráfica es un programa gráfico que incluye una representación gráfica de lo que el robot debería hacer durante la operación. Por ejemplo, el programa gráfico incluye puntos en 3D y secuencias, puede incluir también la descripción de cómo el robot ha de tomar las esquinas, cómo de rápido la herramienta del robot se ha de mover con relación a una parte, así como información de proceso para procesos tales como soldadura por arco, soldadura por puntos, etc. Cuando la simulación y la programación fuera de línea están completadas, el programa gráfico debe convertirse o traducirse a un programa de robot real. Convencionalmente, la traducción desde el programa gráfico al programa de robot se consigue por un traductor unidireccional, que convierte puntos en 3D y secuencias y otra información a lo largo de un objeto en 3D en una secuencia de instrucciones de movimiento para un lenguaje de programación de robot específico. El programa de robot se transfiere a continuación al controlador de robot real.

Este programa de robot casi nunca está completamente correcto o completo. El flujo de programa debe añadirse para asegurar bucles correctos y manejo de condiciones. Se ha de añadir también las señales y manejo de entrada-salida. Los errores en calibración entre el mundo real y la simulación en 3D significan que muchas instrucciones y posiciones deben modificarse para que el robot funcione sin colisiones. Además, las últimas modificaciones al equipo significa que deben añadirse nuevas instrucciones. En la presión para mejorar el rendimiento del robot, las instrucciones se modifican para aumentar la velocidad del robot y reducir la precisión de los movimientos del robot en esquinas y en movimientos grandes. Finalmente, se comenta el programa en el controlador, permitiendo al programador futuro la capacidad de entender la lógica del programa y el razonamiento detrás de las modificaciones. Se añaden comentarios a través de todo el programa para mejorar la legibilidad y capacidad de mantenimiento del

programa de robot.

Todas las modificaciones anteriores, significan que el programa de robot real a menudo es muy diferente de la descripción en la simulación fuera de línea en 3D. Después de que se ha hecho este trabajo, convencionalmente, la simulación gráfica en 3D ya no puede usarse más. Si un programador usara el programa gráfico, generado durante la simulación en 3D, para hacer una modificación, entonces todos los cambios anteriores tendrían que rehacerse. Esto significa que la simulación en 3D ya no es una herramienta de programación válida y los beneficios de la programación fuera de línea se pierden a menos que la simulación en 3D pueda actualizarse con los datos de programa reales del programa de robot real.

Convencionalmente, los fabricantes de robots y de simulación en 3D intentan actualizar la simulación en 3D desde el programa de robot mediante otro traductor unidireccional, que toma un programa de robot y lo convierte en un programa gráfico que es una representación en 3D del movimiento del robot. De esta manera, se descarta el flujo de programa y las condiciones, la mayoría del manejo de entrada-salida y los comentarios.

La solución convencional para tratar de evitar esta limitación es separar el programa de robot en secciones, subrutinas y/o módulos, donde las instrucciones del robot relacionadas con el movimiento, pueden cargarse de vuelta en la simulación en 3D sin pérdida, se mantienen en una sección separada, rutina, o módulo desde las instrucciones que manejan flujo de programa, condiciones, manejo de entrada-salida y comentarios. Este enfoque tiene desventajas en que todo el programa no puede manejarse por el sistema de simulación y hay trabajo manual necesario para volver a ensamblar el programa completo de nuevo después de las modificaciones a una sección, rutina o módulo desde el sistema de simulación en 3D. Por lo tanto, la simulación en 3D convencional y las herramientas de programación fuera de línea proporcionan un traductor de 'descarga' y un traductor de 'carga'.

Sin embargo, existe un deseo de usar una simulación gráfica en 3D para generar un programa de robot, para modificar el programa en el robot real, y para cargar sin interrupciones y, sin pérdidas, el programa de vuelta en la simulación en 3D y modificarlo de nuevo sin dividir artificialmente el programa.

El documento US6.594.822 desvela un método para actualizar un programa informático que comprende compilar el programa en ficheros de objeto, descomponer dichos ficheros en partes y añadir correcciones a estas partes y construir posteriormente grafos de dependencia de programa reducidos (RPDG) a partir de las partes y las correcciones para cada fichero de objeto. Las diferencias halladas entre los RPDG se usan para desarrollar un parche para actualizar el programa informático.

El documento US6.883.161 desvela una herramienta universal para compilación de grafos tal como diagramas de flujo en un ordenador, en la que el diagrama de flujo capturado puede traducirse en otro lenguaje tal como C, Fortran y similares.

El documento US6.138.270 desvela un método para detectar diferencias entre dos programas gráficos, en el que ambos programas comprenden códigos gráficos.

Objetos y sumario de la invención

El objeto de la presente invención es hallar una solución atractiva, que remedia los problemas anteriormente mencionados.

De acuerdo con un aspecto de la invención, este objeto se consigue por un método para sincronización de un programa de robot y un programa gráfico como se define en la reivindicación 1.

Un método de este tipo incluye las siguientes etapas: recibir información sobre cuál programa ha de ser el líder para la sincronización y cuál programa ha de ser el seguidor, convertir el programa gráfico en una primera secuencia de testigos que representa datos para movimientos y acciones contenidos en el programa gráfico, convertir el programa de robot en una segunda secuencia de testigos que representa datos para movimientos y acciones contenidos en el programa de robot, comparar la primera y segunda secuencias de testigos y basándose en la mismas generar comandos de modificación en dependencia de cuál programa es el líder y cuál es el seguidor, y editar el programa del seguidor basándose en los comandos de modificación generados de modo que la secuencia de testigos del seguidor coincida con la secuencia del líder.

La presente invención proporciona una solución mediante la cual el contenido de movimiento en un programa gráfico en un entorno de simulación fuera de línea y un programa de lenguaje nativo en un controlador de robot pueden actualizarse y mantenerse consistentes entre sí. Por consiguiente, la presente invención proporciona un método que puede sincronizar el programa gráfico, generado durante la simulación, con el programa de robot en el controlador de robot sin pérdida de flujo de programa, manejo de entrada-salida o comentarios y otras instrucciones no de movimiento. Las modificaciones del contenido de movimiento de cualquiera del programa gráfico o del programa de robot pueden transferirse de vuelta desde la representación gráfica a la representación del lenguaje de robot y viceversa sin re-generación o pérdida del resto del programa o contexto gráfico. Esto es de importancia crítica en

todas las implementaciones de programación fuera de línea.

El contenido de movimiento del programa gráfico y del programa de robot de lenguaje nativo se representa como una serie de testigos. El concepto de testigos forma una representación neutral de la secuencia y datos en los programas de lenguaje gráfico y nativo, que hace posible comparar los programas.

De acuerdo con una realización de la invención, la etapa de edición incluye editar la secuencia de testigos del seguidor de modo que la secuencia de testigos coincide con la secuencia del líder y convertir la secuencia editada de los testigos del seguidor de vuelta a la forma de programa original, es decir de vuelta al programa gráfico o al programa de robot dependiendo de cuál de ellos se seleccione para ser el seguidor. Aunque es posible generar comandos de modificación para edición directa del programa original, es decir el programa gráfico o el de robots, es mucho más fácil generar comandos de modificación para una secuencia de testigos y editar la secuencia.

De acuerdo con una realización de la invención, dichos programas se convierten en secuencias de sentencias construidas de dichos testigos, y dichos comandos de modificación incluyen insertar sentencia, borrar sentencia y testigo de edición. La descomposición en sentencias y testigos proporciona la capacidad de funcionar con instrucciones completas, es decir las sentencias, y parámetros de las instrucciones, es decir los testigos. Por ejemplo, una instrucción de movimiento sencilla puede consistir en la instrucción sencilla "Lineal p1, 1000", que da la instrucción al robot para moverse linealmente al punto "p1" a una velocidad de 1000 mm/s. La conversión a serie convertiría la instrucción en una única sentencia con 3 testigos: "lineal", "p1" y "1000". Cuando se sincroniza una representación gráfica con la representación del programa textual, el uso de sentencias y testigos permite la mayor libertad posible para que el motor de sincronización halle el número mínimo de cambios necesarios para proporcionar una representación en sincronización entre sí.

De acuerdo con una realización de la invención, se calcula la diferencia entre la primera y segunda secuencia de testigos por medio de un método de Subsecuencia Común Más Grande. Esta realización es particularmente ventajosa puesto que la última subsecuencia común más grande entre las dos representaciones proporciona una indicación para el conjunto más común de sentencias y testigos. El subconjunto común más grande es por lo tanto el conjunto de sentencias y testigos que no tienen que cambiarse. Esto significa que el algoritmo automáticamente halla el mínimo número de cambios necesarios para proporcionar una representación en sincronización con la otra. Hallar el mínimo número de cambios es ventajoso en que todos los comentarios circundantes, lógica, flujo de programa se mantienen como están sin riesgo de modificaciones indeseadas.

Como se ha mencionado anteriormente, los lenguajes de programación de robot pueden contener también una sección de datos que se usa por las instrucciones. En el ejemplo anterior, "p1" es una referencia a una sección de datos donde se mantienen las coordenadas reales (conjuntas o lineales) del objetivo. En una realización de la invención, las secciones de datos de un programa se sincronizan también con la representación gráfica. Cuando las secciones de datos se han de sincronizar, esto se hacen en primer lugar, y a continuación en segundo lugar se sincronizan las instrucciones. Esto se consigue mediante el mismo método de generación de sentencias y testigos y hallando la Subsecuencia Común Más Grande para dirigir las modificaciones a cualquiera de la representación gráfica o textual.

De acuerdo con un aspecto adicional de la invención, el objeto se consigue por un producto de programa informático directamente cargable en la memoria interna de un ordenador o un procesador, que comprende porciones de código de software para realizar las etapas del método de acuerdo con el conjunto de reivindicaciones de método adjuntas, cuando el programa se ejecuta en un ordenador. El programa informático se proporciona en cualquiera de un medio legible por ordenador o a través de una red.

Es fácilmente darse cuenta de que el método de acuerdo con la invención, como se define en el conjunto de reivindicaciones de método adjuntas, es adecuado para la ejecución por un programa informático que tiene instrucciones que corresponden a las etapas en el método inventivo cuando se ejecutan en una unidad de procesador.

De acuerdo con otro aspecto de la invención, el objeto se consigue por un medio legible por ordenador que tiene un programa grabado en el mismo, cuando el programa hace que un ordenador realice las etapas del método de acuerdo con el conjunto de reivindicaciones de método adjuntas, y el programa se ejecuta en el ordenador.

De acuerdo con otro aspecto de la invención, este objeto se consigue por un dispositivo para sincronización de un programa de robot y un programa gráfico como se define en la reivindicación 7.

Un dispositivo de este tipo comprende: un primer convertidor a serie adaptado para convertir un programa gráfico en una primera secuencia de testigos que representa datos para movimientos y acciones contenidos en el programa gráfico, un segundo convertidor a serie adaptado para convertir el programa de robot en una segunda secuencia de testigos que representa datos para movimientos y acciones contenidos en el programa de robot, y un motor de sincronización adaptado para comparar la primera y segunda secuencias de testigos y basándose en las mismas genera comandos de modificación, en el que dichos primeros convertidores a serie están adaptados para recibir

dichos comandos de modificación y editar el programa gráfico basándose en los comandos de modificación de modo que la secuencia de testigos del programa gráfico coincide con la secuencia de testigos del programa de robot y dichos segundo convertidores a serie están adaptados para recibir dichos comandos de modificación y editar el programa de robot basándose en los comandos de modificación de modo que la secuencia de testigos del programa de robot coincide con la secuencia de testigos del programa gráfico.

La presente invención emplea un motor de sincronización que funciona junto con dos convertidores a serie. Un convertidor a serie funciona hacia el programa gráfico, mientras que el otro convertidor a serie funciona hacia el programa de robot. Los convertidores a serie son responsables de convertir el programa gráfico y el programa de robot en una secuencia de testigos. De acuerdo con una realización de la invención, el programa gráfico y el programa de robot se convierten en una secuencia de sentencias. Cada sentencia está construida de testigos donde cada testigo representa una pieza indivisible de información. El motor de sincronización calcula las diferencias entre las dos secuencias de sentencias, por ejemplo, mediante un método de Subsecuencia Común Más Grande. Las diferencias se convierten en instrucciones de edición, denominadas comandos de modificación, y se envían de vuelta a uno o al otro convertidor a serie para modificar cualquiera del programa gráfico o del programa de robot.

De acuerdo con una realización de la invención, el dispositivo está adaptado para recibir información sobre cuál programa ha de ser el líder para la sincronización y cuál programa ha de ser el seguidor, y el motor de sincronización está adaptado para generar y enviar dichos comandos de modificación a cualquiera del primero o el segundo convertidor a serie en dependencia de cuál programa es el líder y cuál es el seguidor. El seguidor está sincronizado de acuerdo con el líder. Esta realización hace posible que un usuario seleccione cuál de los programas, es decir, el programa gráfico o el de robots, ha de sincronizarse de acuerdo con el otro programa.

De acuerdo con otro aspecto de la invención, el motor de sincronización usa cualquier otra forma de algoritmo de diferenciación que produce un conjunto de cambios que proporciona una secuencia de sentencias en acuerdo entre sí.

De acuerdo con otro aspecto de la invención, la representación gráfica no está basada en 3D, sino que está basada en iconos, donde cada icono representa una instrucción o acción para el robot. Los iconos se ensamblan en secuencias para representar la tarea para el robot.

Breve descripción de los dibujos

La invención se explicará ahora más estrechamente por la descripción de diferentes realizaciones de la invención y con referencia a las figuras adjuntas.

La Figura 1 muestra un diagrama de bloques a través de un dispositivo de sincronización de acuerdo con la invención.

Las Figuras 2-11 muestran un ejemplo detallado de cómo funciona la invención.

La Figura 12 muestra un diagrama de flujo a través de un método de acuerdo con una realización de la invención.

Descripción detallada de realizaciones preferidas de la invención

La Figura 1 muestra un diagrama de bloques a través de un dispositivo de sincronización de acuerdo con la invención. El dispositivo contiene tres objetos; un primer convertidor a serie 1, que convierte un programa gráfico en una serie de sentencias y recibe comandos de modificación, un segundo convertidor a serie 2, que convierte un programa de robot en una serie de sentencias y recibe comandos de modificación, y un motor de sincronización 3, que halla las diferencias entre las dos series de sentencias y envía comandos de modificación a los convertidores a serie. El motor de sincronización halla los cambios necesarios para hacer que la secuencia del seguidor coincida con la secuencia del líder. El programa gráfico se almacena en un área de memoria en un ordenador usado para programación fuera de línea del robot. El programa de robot se almacena en un área de memoria en el controlador de robot 5. Todos los tres objetos se implementan por software.

El programa de robot se almacena en un área de memoria en el controlador de robot o en un controlador virtual en el ordenador fuera de línea. Además, la invención incluye la situación donde el programa de robot se almacena en un fichero y se proporciona un analizador que puede leer las secciones de datos e instrucciones en el programa de robot y proporciona respuestas necesarias al convertidor a serie para generar las secuencias de sentencias y testigos y para cambiar el programa en respuesta a la sincronización.

En una realización de la invención, el programa gráfico se implementa en el sistema de simulación en 3D, el programa de robot se implementa en un controlador virtual que simula de manera precisa la capacidad del controlador real para almacenar y modificar programas. Los motores de sincronización se implementan en el sistema de simulación en 3D y comunican a la representación gráfica y al controlador virtual.

Un programa gráfico comprende rutas de objetivos y acciones proporcionadas en secuencia y mostradas en una pantalla gráfica. Cada objetivo contiene atributos para el movimiento, lineal, conjunto, velocidad, zona, etc. Las acciones contienen eventos tales como eventos de E/S y/o de programa que tendrán lugar a lo largo de la ruta. La ruta, objetivos, atributos y acciones son necesarios para programación gráfica y el análisis de un programa de robot gráfico.

El primer convertidor a serie 1 se le proporciona una ruta y convierte la secuencia gráfica de objetivos y acciones a una secuencia de sentencias compuestas de testigos. Cada sentencia representa cualquiera de un movimiento completo o una acción individual. Un movimiento completo contiene todos los testigos para posición, herramienta y estructura de objeto que describen la geometría del robot en la posición objetivo. Cada testigo representa los datos para el movimiento, es decir datos para la estructura de herramienta, datos para la estructura de objeto, datos para la velocidad, etc. El primer convertidor a serie 1 considerará únicamente estos movimientos y acciones, que tienen una representación de programa equivalente para conversión a serie). El convertidor a serie 1 no tiene que considerar todas las instrucciones posibles disponibles para el robot específico. Únicamente necesitan considerarse las instrucciones que son de interés o de capacidad para sincronizar. Por ejemplo, en el caso donde no hay representación gráfica, por ejemplo un bucle FOR-NEXT, entonces el convertidor a serie no necesita convertir el bucle FOR-NEXT y la sincronización continúa de todas formas.

Durante la sincronización, cuando el primer convertidor a serie 1 es el seguidor, debe aceptar comandos de modificación desde el motor de sincronización e insertar, borrar sentencias y editar testigos; convirtiéndolos de vuelta a la representación gráfica en su posición correcta a lo largo de la ruta.

Un programa de robot de lenguaje nativo comprende elementos de flujo de programa, tales como subrutinas, funciones, if-then-else, while, instrucciones de E/S, instrucciones de proceso, instrucciones de movimiento así como elementos de datos para mantener el objetivo específico, datos de velocidad, herramienta, etc.

El segundo convertidor a serie 2 se le proporciona cualquiera de un programa, subrutina o una serie de movimientos, dependiendo del lenguaje de robot. El segundo convertidor a serie 2 a continuación construye una secuencia de sentencias, conteniendo cada sentencia únicamente los movimientos completos y acciones pero con la misma secuencia como se hallara en el programa de robot. Cada sentencia contiene testigos que representan los datos para el movimiento y acciones.

Durante la sincronización, cuando el segundo convertidor a serie 2 es el seguidor, debe recibir comandos de modificación desde el motor de sincronización e insertar, borrar sentencias y editar testigos, convirtiéndolos de vuelta en la representación de programa de robot.

La función del motor de sincronización de objeto es comparar una secuencia de sentencias y testigos y enviar comandos de modificación a cualquiera del primer convertidor a serie 1 o del segundo convertidor a serie 2, dependiendo de cuál sea el líder y cuál sea el seguidor. La comparación puede basarse en muchos métodos conocidos para comparar datos, por ejemplo el método de Subsecuencia Común Más Grande que minimiza las modificaciones para el seguidor. Los comandos de modificación se envían a cualquiera del convertidor a serie en la forma: insertar sentencia, borrar sentencia, editar testigo.

A continuación, se explicará un ejemplo de la invención en detalle con referencia a las Figuras 2-11. El ejemplo comienza con un programa gráfico, como se muestra en la Figura 2, y un programa textual, como se muestra en la Figura 3. El programa gráfico en la Figura 2 muestra un robot simulado 8 y una ruta 9 en espacio, o igualmente aplicable, una ruta fijada a un objeto geométrico. En este ejemplo, la ruta tiene tres puntos, p1, p2, y p3 con un número de atributos que describen cómo debería moverse la simulación a los puntos (linealmente), la velocidad a usarse, (800 mm/s), el tamaño de la zona de precisión de punto (100 mm) y la herramienta (herramienta 10) a usarse, y opcionalmente el objeto de trabajo (wobj0) en el que se representa el punto. El programa textual mostrado en la Figura 3 contiene la representación del programa de robot dada del programa gráfico. Las dos representaciones puede decirse que están ‘en sincronización’ entre sí.

En este ejemplo, mostraremos el caso donde el programa gráfico es el maestro y el programa textual es la copia. Por lo tanto, en el ejemplo, un usuario de la simulación y herramienta de programación fuera de línea puede hacer una modificación al programa gráfico. Un caso típico es la adición de un nuevo punto y una modificación del objeto de trabajo usado para p1. El programa gráfico modificado se muestra en la Figura 4. En este punto en el ejemplo, el objetivo es sincronizar el programa gráfico modificado con el programa textual original para producir un nuevo programa textual que contiene los nuevos elementos y modificaciones.

La generación de sentencias y testigos se describirá ahora con referencia a las Figuras 5-8. Incluso aunque la sincronización funciona igualmente bien con datos como con las instrucciones en el programa, se ha hallado que es mejor sincronizar los datos en primer lugar. El convertidor a serie 1 para la representación gráfica produciría los datos convertidos a serie como se muestra en la Figura 5. La Figura 5 muestra datos convertidos a serie después de la modificación gráfica. El convertidor a serie 1 produciría también las instrucciones convertidas a serie como se muestra en la Figura 6. La Figura 6 muestra instrucciones convertidas a serie después de la modificación gráfica.

Por consiguiente, el convertidor a serie 2, fijado al programa textual produciría también sus propios datos de conversión a serie e instrucciones del programa textual. Estos se muestran en la Figura 7 y 8. La Figura 7 muestra datos convertidos a serie desde el programa textual. La Figura 8 muestra instrucciones convertidas a serie desde el programa textual.

La siguiente etapa es usar el motor de sincronización 3 para comparar los datos convertidos a serie e instrucciones convertidas a serie y generar los comandos de modificación que harían a la versión textual (la copia) igual que la versión gráfica (el maestro). Obsérvese que cualquier versión gráfica o textual puede ser la versión maestra para los fines de sincronización. El motor de sincronización toma los datos convertidos a serie anteriores y los flujos de instrucciones y produce comandos de edición que hacen la copia idéntica al maestro.

Las Figuras 9 y 10 muestran los comandos de edición generalizados que se generarían desde un motor de sincronización usando el algoritmo de Subconjunto Común Más Grande. La Figura 5 muestra comandos de edición para los datos y la Figura 6 muestra comandos de edición para las instrucciones. Estos comandos de edición se envían desde el motor de sincronización al convertidor a serie 2. El convertidor a serie 2 aplica los comandos de edición al programa textual. El programa textual modificado resultante se muestra en la Figura 11 con las modificaciones resultantes mostradas en cursiva.

Es importante observar que los comandos de edición son un conjunto mínimo de comandos de edición necesarios para proporcionar el programa textual en sincronización con el programa gráfico. Este enfoque deja el texto restante sin tocar y permite que se conserven espacio en blanco y comentarios. Además, el enfoque permite que el programa gráfico o el programa textual sea el maestro sin ninguna limitación añadida.

La Figura 12 es una ilustración de diagrama de flujo del método y el producto de programa informático de acuerdo con una realización de la presente invención. Se entenderá que cada bloque del diagrama de flujo puede implementarse por instrucciones de programa informático.

Un programa gráfico y el programa de lenguaje nativo se mantienen en sincronización por la siguiente secuencia:

Bloque 11: se recibe información sobre cuál programa ha de ser el líder para la sincronización y cuál programa ha de ser el seguidor. El usuario selecciona qué programa (gráfico, o robot) ha de ser el líder para la sincronización y cuál ha de ser el seguidor. El programa que se selecciona como el seguidor ha de modificarse de acuerdo con el programa que se selecciona como el líder, es decir se cambia el programa seguidor y el programa líder permanece sin cambiar.

Bloque 12: el programa gráfico se convierte en una primera secuencia de testigos que representa datos para movimientos y acciones contenidos en el programa gráfico. El convertidor a serie 1 convierte los programas gráficos a secuencias de sentencias y testigos.

Bloque 14: el programa de robot se convierte en una segunda secuencia de testigos que representa datos para movimientos y acciones contenidos en el programa de robot. El convertidor a serie convierte los programas de robots a secuencias de sentencias y testigos, como se muestra en la Figura 2. En qué orden se ejecuta la conversión del programa gráfico y el de robot es irrelevante, es decir el programa de robot puede convertirse antes que el programa gráfico o la conversión podría realizarse en paralelo.

Los datos representados por los testigos se intercambian desde el convertidor a serie 1 al convertidor a serie 2 dependiendo de cuál sea el líder.

Bloque 16: se compara la primera y segunda secuencias de testigos. Las secuencias de sentencias y testigos se envían al motor de sincronización. El motor de sincronización compara las secuencias.

Bloque 18: se generan comandos de modificación en dependencia de cuál programa es el líder y cuál es el seguidor basándose en la comparación entre la primera y segunda secuencias. El motor de sincronización envía comandos de modificación a cualquiera del convertidor a serie 1 o al convertidor a serie 2 dependiendo de cuál sea el seguidor.

Bloque 20: se edita la secuencia de testigos del seguidor basándose en los comandos de modificación generados de modo que la secuencia de testigos del seguidor coincide con la secuencia del líder. El siguiente convertidor a serie recibe los comandos de modificación y edita su programa en consecuencia de manera que sus secuencias de sentencias y testigos coinciden con el líder de manera exacta.

Bloque 22: la secuencia editada de testigos del seguidor se convierte de vuelta a la forma de programa original, es decir de vuelta al programa gráfico o el de robot.

El software para llevar a cabo el método de acuerdo con la invención, por ejemplo, se almacena y ejecuta en el controlador de robot o en el ordenador usado para la programación y simulación fuera de línea. Sin embargo, es posible también almacenar y ejecutar el software en cualquier otro ordenador externo.

REIVINDICACIONES

1. Un método para sincronizar un programa gráfico generado en un entorno de simulación fuera de línea y un programa de robot, en el que el programa gráfico incluye una representación en 3D gráfica de movimientos de robot a lo largo de una ruta (9) que incluye puntos en 3D (p1, p2, p3) y acciones proporcionadas en una secuencia, las acciones contienen eventos que tendrán lugar a lo largo de la ruta, y el programa de robot incluye una secuencia de instrucciones textuales para movimientos de robot y acciones, caracterizado por que el método comprende:

recibir información sobre cuál de los programas permanece sin cambiar durante la sincronización, a continuación denominado un programa líder para la sincronización, y cuál de los programas va a modificarse de acuerdo con el programa que se selecciona como el programa líder, a continuación denominado un programa seguidor,
convertir la secuencia de puntos en 3D y acciones del programa gráfico en una primera secuencia de testigos que incluye datos e instrucciones para movimientos y acciones contenidos en el programa gráfico,
convertir el contenido de movimiento del programa de robot en una segunda secuencia de testigos que incluye datos e instrucciones para movimientos y acciones contenidos en el programa de robot,
comparar la primera y segunda secuencias de testigos y basándose en las mismas
generar comandos de modificación en dependencia de cuál programa es el líder y cuál es el seguidor, y
editar el programa seguidor basándose en los comandos de modificación generados de modo que la secuencia de testigos del seguidor coincide con la secuencia del líder.

2. El método de acuerdo con la reivindicación 1, en el que la etapa de edición incluye editar la secuencia de testigos del seguidor de modo que la secuencia de testigos coincide con la secuencia del líder y convertir la secuencia editada de testigos del seguidor de vuelta a la forma de programa original.

3. El método de acuerdo con la reivindicación 1 o 3, en el que dichos programas se convierten en secuencias de sentencias construidas de dichos testigos, y dichos comandos de modificación incluyen insertar sentencia, borrar sentencia y editar testigo.

4. El método de acuerdo con cualquiera de las reivindicaciones anteriores, en el que el método comprende hallar la subsecuencia común más grande entre la primera y segunda secuencias y determinar la diferencia entre la primera y segunda secuencias de testigos basándose en las mismas.

5. Un producto de programa informático directamente cargable en la memoria interna de un ordenador, que comprende software para realizar las etapas de cualquiera de las reivindicaciones 1-4.

6. Un medio legible por ordenador, que tiene un programa grabado en el mismo, donde el programa hace que un ordenador realice las etapas de cualquiera de las reivindicaciones 1-4, cuando dicho programa se ejecuta en el ordenador.

7. Un dispositivo para sincronizar un programa gráfico generado en un entorno de simulación fuera de línea y un programa de robot, en el que el programa gráfico incluye una representación en 3D gráfica de movimientos de robot a lo largo de una ruta que incluye puntos en 3D (p1, p2, p3) y acciones proporcionadas en una secuencia, las acciones contienen eventos que tendrán lugar a lo largo de la ruta, y el programa de robot incluye una secuencia de instrucciones textuales para movimientos de robot y acciones, caracterizado por que el dispositivo comprende:

medios (1) adaptados para convertir la secuencia de puntos en 3D y acciones del programa gráfico en una primera secuencia de testigos que incluye datos e instrucciones para movimientos y acciones contenidos en el programa gráfico,
medios (2) adaptados para convertir el contenido de movimiento del programa de robot en una segunda secuencia de testigos que incluye datos e instrucciones para movimientos y acciones contenidos en el programa de robot, y
medios (3) adaptados para comparar la primera y segunda secuencias de testigos y basándose en las mismas generar comandos de modificación, en el que dichos medios (1) adaptados para convertir un programa gráfico están adaptados para recibir dichos comandos de modificación y editar el programa gráfico basándose en los comandos de modificación de modo que la secuencia de testigos del programa gráfico coincide con la secuencia de testigos del programa de robot y dichos medios (2) adaptados para convertir el programa de robot están adaptados para recibir dichos comandos de modificación y editar el programa de robot basándose en los comandos de modificación de modo que la secuencia de testigos del programa de robot coincide con la secuencia de testigos del programa gráfico.

8. El dispositivo de acuerdo con la reivindicación 7, en el que el dispositivo está adaptado para recibir información sobre cuál de los programas permanece sin cambiar durante la sincronización, a continuación denominado un programa líder para la sincronización, y cuál de los programas a modificarse de acuerdo con el programa que se selecciona como el programa líder, a continuación denominado un programa seguidor, y dichos medios (3) adaptados para comparar la primera y segunda secuencias de testigos están adaptados para generar y enviar dichos comandos de modificación a cualquiera de dichos medios (1) adaptados para convertir un programa gráfico o

dichos medios (2) adaptados para convertir el programa de robot en dependencia de cuál programa es el líder y cuál es el seguidor.

9. El dispositivo de acuerdo con la reivindicación 8, en el que dichos medios de convertidor (1, 2) están adaptados para editar la secuencia de testigos del seguidor de modo que la secuencia de testigos coincide con la secuencia del líder y convertir la secuencia editada de testigos del seguidor de vuelta a la forma de programa original.

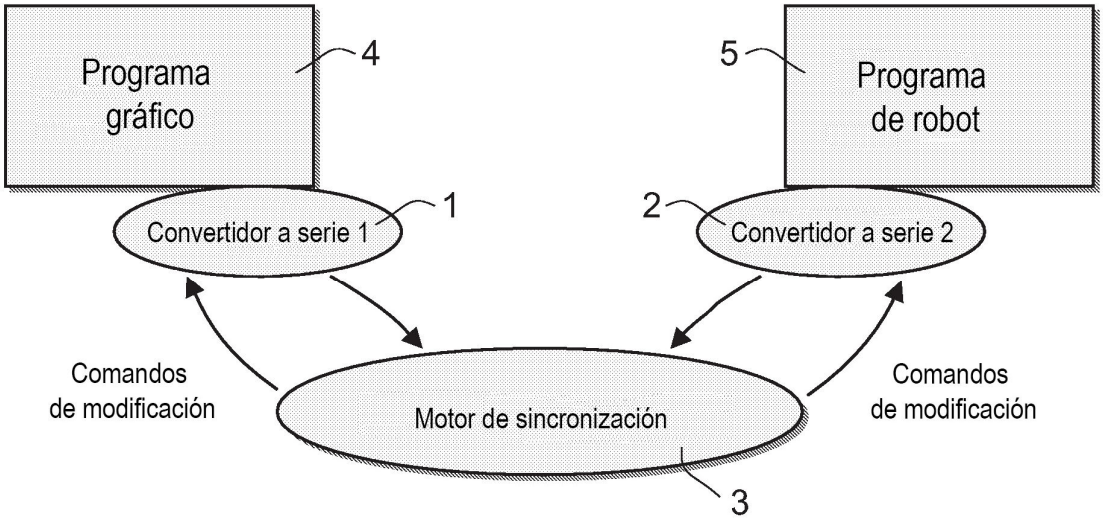
Programa gráfico
4
5
Programa de robot
1
Convertidor a serie 1
2
Convertidor a serie 2
Comandos de modificación
Motor de sincronización
Comandos de modificación
3

Fig. 1

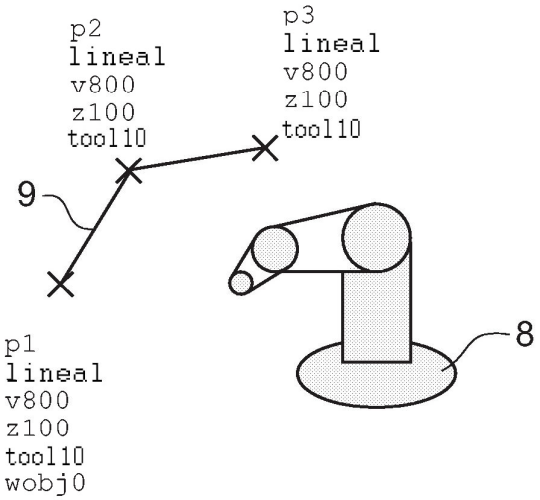
p2
lineal
v800
z100
tool10
p3
lineal
v800
z100
tool10
9
8
p1
lineal
v800
z100
tool10
wobj0

Fig. 2

Programa:

```
objetivo p1:=[[1881,-983,676],[0.330,0.571,0.651,0.376],[-1,1,-1,-1]];
objetivo p2:=[[1881,-649,879],[0.330,0.571,0.651,0.376],[-1,2,-2,-1]];
objetivo p3:=[[1868,-143,866],[0.571,0.330,0.376,0.651],[-1,1,-1,-1]];

PROC path()
  MoveL p1,v800,z100,tool0\WObj:=wobj0;
  AnotherInstr;
  MoveL p2,v800,z100,tool0;
  MoveL p3,v800,z100,tool0;
  AnotherInstr;
ENDPROC
```

Fig. 3

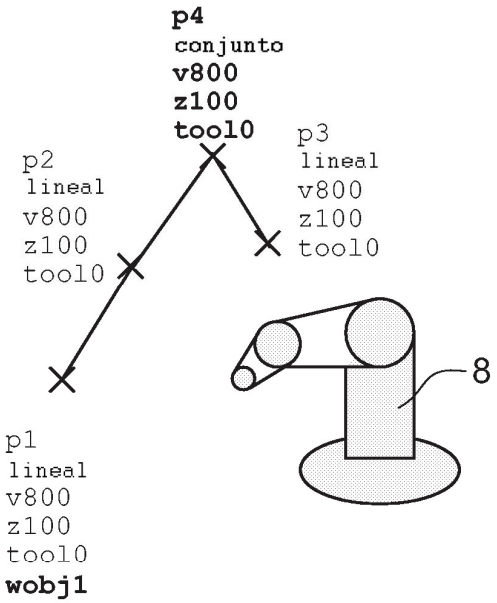

Fig. 4

```
{sentencia 1
   {testigo 11 {p1}}
   {testigo 12 {objetivo}}
   {testigo 13 {[[1881,-983,676],[0.330,0.571,0.651,0.376],[-1,1,-1,-1]]}}}
{sentencia 2
   {testigo 21 {p2}}
   {testigo 22 {objetivo}}
   {testigo 23 {[[1881,-649,879],[0.330,0.571,0.651,0.376],[-1,2,-2,-1]]}}}
{sentencia 3
   {testigo 31 {p4}}
   {testigo 32 {objetivo}}
   {testigo 33 {[[1881,-449,964],[0.330,0.571,0.651,0.376],[-1,1,-1,-1]]}}}
{sentencia 4
   {testigo 41 {p3}}
   {testigo 42 {objetivo}}
   {testigo 43 {[[1868,-143,866],[0.571,0.330,0.376,0.651],[-1,1,-1,-1]]}}}
```

Fig. 5

```
{sentencia 5
   {testigo 51 {MoveL}}
   {testigo 52 {ToPoint p1}}
   {testigo 53 {Velocidad v800}}
   {testigo 54 {Zona   z100}}
   {testigo 55 {Herramienta tool0}}
   {testigo 56 {WObj wobj1}}}
{sentencia 6
   {testigo 61 {MoveL}}
   {testigo 62 {ToPoint p2}}
   {testigo 63 {Velocidad v800}}
   {testigo 64 {Zona   z100}}
   {testigo 65 {Herramienta tool0}}}
{sentencia 7
   {testigo 71 {MoveJ}}
   {testigo 72 {ToPoint p4}}
   {testigo 73 {Velocidad v800}}
   {testigo 74 {Zona   z100}}
   {testigo 75 {Herramienta tool0}}}
{sentencia 8
   {testigo 81 {MoveL}}
   {testigo 82 {ToPoint p3}}
   {testigo 83 {Velocidad v800}}
   {testigo 84 {Zona   z100}}
   {testigo 85 {Herramienta tool0}}}
```

Fig. 6

```
{sentencia 1
   {testigo 11 {p1}}
   {testigo 12 {objetivo}}
   {testigo 13 {[[1881,-983,676],[0.330,0.571,0.651,0.376],[-1,1,-1,-1]]}}}
{sentencia 2
   {testigo 21 {p2}}
   {testigo 22 {objetivo}}
   {testigo 23 {[[1881,-649,879],[0.330,0.571,0.651,0.376],[-1,2,-2,-1]]}}}
{sentencia 3
   {testigo 31 {p3}}
   {testigo 32 {objetivo}}
   {testigo 33 {[[1868,-143,866],[0.571,0.330,0.376,0.651],[-1,1,-1,-1]]}}}
```

Fig. 7

```
{sentencia 4
   {testigo 41 {MoveL}}
   {testigo 42 {ToPoint p1}}
   {testigo 43 {Velocidad v800}}
   {testigo 44 {Zona   z100}}
   {testigo 45 {Herramienta tool0}}
   {testigo 46 {WObj wobj0}}}
{sentencia 5
   {testigo 51 {MoveL}}
   {testigo 52 {ToPoint p2}}
   {testigo 53 {Velocidad v800}}
   {testigo 54 {Zona   z100}}
   {testigo 55 {Herramienta tool0}}}
{sentencia 6
   {testigo 61 {MoveL}}
   {testigo 62 {ToPoint p3}}
   {testigo 63 {Velocidad v800}}
   {testigo 64 {Zona   z100}}
   {testigo 65 {Herramienta tool0}}}
```

Fig. 8

```
Insertar sentencia de datos

Nueva sentencia =
{sentencia 0
   {testigo 0 {p4}}
   {testigo 0 {robtarget}}
   {testigo 0 {[[1881,-449,964],[0.330,0.571,0.651,0.376],[-1,1,-1,-1]]}}}
```

Fig. 9

```
Insertar sentencia de instrucción antes

Sentencia antigua =
{sentencia  6
    {testigo 61 {MoveL}}
    {testigo 62 {ToPoint p3}}
    {testigo 63 {Velocidad v800}}
    {testigo 64 {Zona   z100}}
    {testigo 65 {Herramienta tool0}}}

Nueva sentencia =
{sentencia  0
    {testigo 0 {MoveJ}}
    {testigo 0 {ToPoint p4}}
    {testigo 0 {Velocidad v800}}
    {testigo 0 {Zona   z100}}
    {testigo 0 {Herramienta tool0}}}


Sustituir testigo de instrucción

Testigo antiguo =
{testigo 46 {WObj wobj0}}

Testigo nuevo =
{testigo 0 {WObj wobj1}}
```

Fig. 10

```
objetivo p1:=[[1881,-983,676],[0.330,0.571,0.651,0.376],[-1,1,-1,-1]];
objetivo p2:=[[1881,-649,879],[0.330,0.571,0.651,0.376],[-1,2,-2,-1]];
objetivo p3:=[[1868,-143,866],[0.571,0.330,0.376,0.651],[-1,1,-1,-1]];
objetivo p4:=[[1881,-449,964],[0.330,0.571,0.651,0.376],[-1,1,-1,-1]];

PROC path()
  MoveL p1,v800,z100,tool0\WObj:=wobj1;
  AnotherInstr;
  MoveL p2,v800,z100,tool0;
  MoveJ p4,v800,z100,tool0;
  MoveL p3,v800,z100,tool0;
  AnotherInstr;
ENDPROC
```

Fig. 11

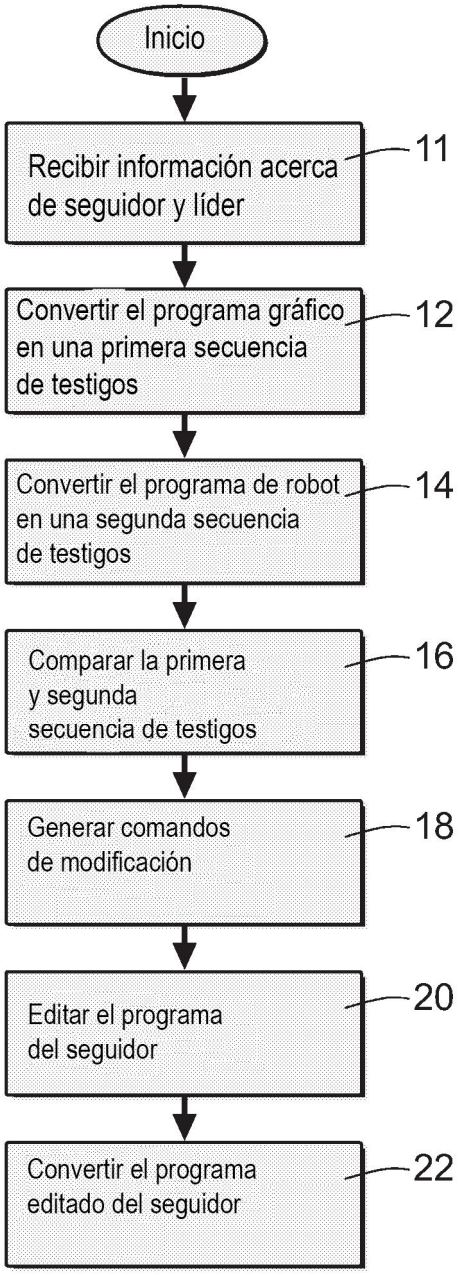
Inicio
Recibir información acerca de seguidor y líder
11
Convertir el programa gráfico en una primera secuencia de testigos
12
Convertir el programa de robot en una segunda secuencia de testigos
14
Comparar la primera y segunda secuencia de testigos
16
Generar comandos de modificación
18
Editar el programa del seguidor
20
Convertir el programa editado del seguidor
22

Fig. 12