Departamento de Ingeniería de Sistemas y Automática

Universidad Politécnica de Valencia

Modelado avanzado en *VirtualRobot*

Martin Mellado Arteche

Escuela Técnica
Superior de Ingeniería
Informática

Abril, 2012

Contenido

Modelado avanzado en *VirtualRobot*

1. Objetivo

El objetivo de este documento es presentar las posibilidades de modelado de componentes en *VirtualRobot* a partir del conocimiento de la estructura de los ficheros ASCII en que se guardan los datos o con el uso de aplicaciones específicas de modelado. Se supone que se conoce el uso del sistema de simulación *VirtualRobot Simulator (VRS)* y del sistema de modelado de objetos, piezas y entornos de robots denominado *VirtualRobot Modeller* (*VRM*), así como de su uso dentro de *VRS*.

2. Modificación de objetos y piezas por sus ficheros ASCII

Un objeto modelado para su uso en *VRS* se guarda en un formato OBF[1] que básicamente tiene la definición de una o más primitivas con la siguiente estructura:

```
; OBJECT DESCRIPTION FILE (OBF)
; This file describes an Object as a set of Primitives

  [GENERAL]                          ; General Information
  ..

  [PRIMITIVE1]                       ; First Primitive of Object
      ..

  [PRIMITIVE2]                       ; Second Primitive of Object
      ..
    ..                               ; As many Primitives as required to define Object (at least one)
```

Para una pieza, el formato es de tipo PAF[1] con una estructura muy similar que añade uno o más sistemas de operación como referencia para movimientos del robot, siendo la estructura de la siguiente forma:

```
; PART DESCRIPTION FILE (PAF)
; This file describes a Part as an Object with a Operating Frame defined for Robot Operation

  [GENERAL]                          ; General Information
      ..

  [OPERATION1]                       ; Location of Operation Frame 1 related to Part Frame
      ..

  [OPERATION2]                       ; Location of Operation Frame 2 related to Part Frame
      ..
    ..                               ; As many Operation Frames as required to define Part (at least one)

  [PRIMITIVE1]                       ; First Primitive of Part
  ..

  [PRIMITIVE2]                       ; Second Primitive of Part
  ..
    ..                               ; As many Primitives as required to define Part (at least one)
```

[1] Los formatos están descritos con detalle en el documento *VRFFD* (*VirtualRobot File Format Definition*) y resumidos en el apéndice B del tutorial de modelado en VirtualRobot

Mientras se mantenga esta estructura y se tenga cuidado con la numeración de las primitivas y los sistemas de operación, así como con todos los campos de definición[2], especialmente con las transformaciones (por ejemplo, no deben incluirse espacios en blanco), es fácil modificar los objetos y piezas, eliminando, añadiendo o modificando sus componentes.

Por ejemplo, si se quiere cambiar el color de algunas primitivas en un objeto ya salvado, se puede entrar en el fichero y cambiar los campos RGB. También es sencillo cambiar alguna dimensión geométrica de una primitiva o cambiar su localización alterando las transformaciones (en este caso, con especial cuidado). Cuando se añadan o eliminen primitivas, por ejemplo para repetir primitivas y localizarlas en sitios diferentes, es importante revisar la numeración de los elementos para que se mantenga de forma coherente.

Se recomienda hacer los cambios de uno en uno e ir probando los ficheros cargándolos en el simulador para ver los efectos, evitando de esta forma que un error impida cargar el fichero y no se sepa dónde se ha producido el error.

3. Modificación de entornos por sus ficheros ASCII

Un entorno modelado para su uso en *VRS* se guarda en un formato ENF[3] que básicamente tiene la definición de un conjunto de objetos y/o piezas (al menos debe haber un componente) con la siguiente estructura:

```
; ENVIRONMENT GEOMETRIC DESCRIPTION FILE (ENF)
; This file describes an Environment as a set of Objects and Parts

    [GENERAL]                       ; General Information
      ..

    [OBJECT1]                       ; First Object
      ..

    [OBJECT2]                       ; Second Object
      ..
      ..                            ; As many Objects as required to define Environment (even none)

    [PART1]                         ; First Part
      ..

    [PART2]                         ; Second Part
      ..
      ..                            ; As many Parts as required to define Environment (even none)
      ..                            ; At least one Object or Part
```

Mientras se mantenga esta estructura y se tenga cuidado con la numeración de los objetos y las piezas, así como con todos sus primitivas y sistemas de operación y sus campos de definición, especialmente con las transformaciones (por ejemplo, no deben incluirse espacios en blanco), es fácil modificar los entornos, eliminando, añadiendo o modificando objetos y/o piezas.

Por ejemplo, si se quiere cambiar el color de algunas primitivas en un entorno ya salvado, se puede entrar en el fichero y cambiar los campos RGB. También es sencillo cambiar alguna dimensión geométrica de una primitiva o cambiar su localización alterando las transformaciones (en este caso, con especial cuidado). También se pueden modificar dónde se localizan los objetos y las piezas con sus transformaciones globales.

[2] Las primitivas geométricas y sus parámetros de definición están detallados en el documento *VRPD* (*VirtualRobot Primitive Definition*) y resumidos en el apéndice A del tutorial de modelado en *VirtualRobot*.
[3] El formato está descrito con detalle en el documento *VRFFD* (*VirtualRobot File Format Definition*) y resumido en el apéndice B del tutorial de modelado en *VirtualRobot*

Se pueden añadir, eliminar y modificar primitivas directamente, pero se recomienda hacer esto sobre objetos o primitivas para poder probar sus resultados y después montar los entornos.

Cuando se añadan o eliminen objetos y/o piezas, es importante revisar la numeración de todos elementos para que se mantenga de forma coherente. Esto es especialmente importante cuando las modificaciones se realizan mediante el uso de "*Copy&Paste*" de ficheros.

Se recomienda hacer los cambios de uno en uno e ir probando los ficheros cargándolos en el simulador para ver los efectos, evitando de esta forma que un error impida cargar el fichero y no se sepa dónde se ha producido el error al haberse realizado modificaciones en varias partes del fichero.

4. Modelado de cintas transportadoras lineales como robots

Las cintas transportadoras lineales son elementos típicos en células de fabricación que permiten desplazar cuerpos que se colocan encima de ellos de forma lineal. Pueden basarse en cintas o en rodillos (Figura 1).

Figura 1. Cintas transportadoras

En *VirtualRobot* las cintas transportadoras se modelan como robots de una articulación lineal (Figura 2). Los robots se definen con dos ficheros[4], uno para la cinemática de formato RKF y otro para la geometría de formato RGF. Al ser robots, se pueden controlar con el *VRSTeachPendant* y se pueden programar con las funciones disponibles de *VREAL* por ejemplo para mover su articulación o para cualquier acción que necesitemos sobre el robot.

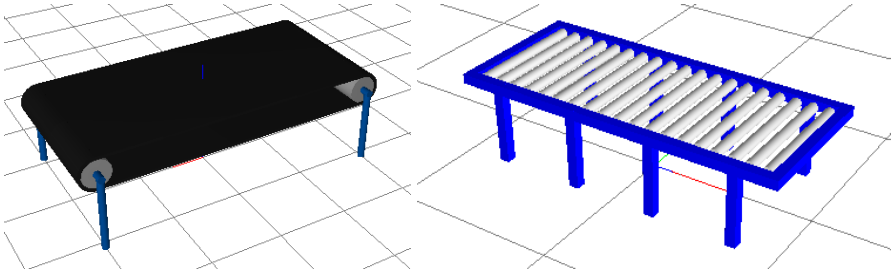

Figura 2. Ejemplos de modelos de cintas transportadoras en *VirtualRobot*

Tarea: Prueba a cargar las cintas transportadoras "Circular Track 4mx2mx130cm.rkf", "Track.rkf" y "x-Track1000.rkf" disponibles en el directorio de *VirtualRobot* "Models\Auxiliar Devices\Tracks". Comprueba con la aplicación *VRSTeachPendant* que sólo tienen un eje prismático y modifica el valor de su variable de control, reconociendo su efecto.

Para modelar una nueva cinta transportadora se aconseja empezar con los ficheros de alguna cinta que sea similar entre las disponibles. Copiar los ficheros RKF y RGF a otros con el nombre deseado, debiendo ser el nombre exactamente igual en ambos ficheros. Por ejemplo, copiar "Track.rkf" y "Track. rgf" a "MiTrack. rkf" y "MiTrack.rgf".

[4] Estos formatos están descritos en detalle en el documento *VRFFD* (*VirtualRobot File Format Definition*)

Las operaciones típicas de modelado que se pueden hacer con una cinta transportadora (usando los nuevos ficheros) son las siguientes:

- Modificar el rango de desplazamiento, es decir, el rango de la articulación prismática de movimiento. Se realiza en el fichero RKF y en concreto en la sección:

  ```
  [JOINT_RANGES]
  Joint1              = -1000,1000
  ```

  Este ejemplo muestra una articulación que se desplaza por un recorrido total de 2m (2000mm). Cambiando estos valores se puede cambiar el rango de movimiento. Es posible fijar rangos muy grandes para poder estar avanzando continuamente una cinta, o en su programación, deshabilitar el control de rangos para que el movimiento nunca esté limitado.

- Modificar la forma geométrica de la cinta. Se recomienda modelar un objeto con la forma desea y utilizarlo para generar el modelo de la cinta como el componente LINK0 del fichero RGF. Para ello, se abre el fichero OBF modelado, se hace una operación de remplazar "[PRIMITIVE" por "[LINK0_PRIMITIVE" y se copian todas las primitivas de este fichero sustituyendo las primitivas del LINK0 del fichero RGF.

Tarea: Modela una nueva cinta transportadora partiendo de algún modelo existente.

A la hora de programar una aplicación[5] con la cinta transportadora se cargará el fichero RKF como un robot más (pudiendo localizarlo donde se desee) y se recomienda programar el movimiento de la cinta con el movimiento por articulaciones modificando sólo la primera articulación (por ejemplo usando la función `alMoveOneRobotJoint`). Con esta función se desplazará el sistema de referencia final del robot.

Se pueden deshabilitar los límites del rango del robot usando `NO_CHECK_RANGE` en el parámetro `checkRange` de la función `alSetRobotCheckRanges`.

Lo habitual es que se desee coger una o más piezas (supuestamente colocadas encima de la cinta) y desplazarlas al mover la cinta. Si las cintas transportadoras se han modelado según se ha comentado anteriormente, dispondrán de 10 *toolFrames* definidos (TCP0…TCP9) y se podrán agarrar hasta 10 piezas (obviamente no se pueden agarrar objetos ya que no disponen de sistemas de operación).

Para agarrar una pieza con la cinta debe activarse un *toolFrame* que esté libre con `alSetActiveToolFrame`, y coger la pieza con `alPickPart` asegurándose de que como parámetro `checkopFrame` se usa `NO_CHECK_COINCIDENCE`.

Ahora cuando se mueva la cinta, todas las piezas "agarradas" se moverán con la cinta. Es función del que modele y programe la aplicación conseguir que las piezas agarradas estén visualmente sobre la cinta transportadora y no *volando* o en su interior para que la aplicación tenga sentido.

Recordando que una pieza no puede agarrarse por más de un robot, la aplicación deberá encargarse de que la cinta libere la pieza si otro robot debe cogerla. Igualmente, para que la cinta agarre una pieza ésta debe estar libre. Un caso de aplicación industrial típico un robot cogerá una pieza y la liberará encima de la cinta, la cinta la agarrará y moverá con un recorrido lineal y liberará cuando se pare para que otro robot la pueda coger y quitar de la cinta. Si se desea que una pieza caiga al suelo al salirse fuera de la cinta, la aplicación deberá reconocer que esta situación se produce y entonces producir un movimiento descendente de la pieza, por ejemplo usando la función `alSetPartLocation`. Como alternativa se puede crear un robot sin elementos visible que agarre la pieza y la desplace hacia abajo.

Tarea: Programa la cinta transportadora modelada en la tarea anterior para que agarre y mueva alguna pieza disponible en el entorno. Probar con la pieza en cualquier lugar y con la pieza colocada encima de la cinta.

[5] La documentación de las funciones está en el documento *VREAL* (*VirtualRobot External Access Libreary*)

5. Modelado de mesas giratorias y cintas transportadoras circulares

Las mesas giratorias y las cintas transportadoras circulares también son elementos típicos en células de fabricación que permiten rotar (a la vez que desplazar) cuerpos que se colocan encima de ellos de forma circular. Algunos ejemplos se pueden ver en la Figura 3.

Figura 3. Mesa giratoria y cintas transportadoras circular

En *VirtualRobot* se pueden modelar ambos elementos como con las cintas transportadoras, pero la única articulación de que disponen será de revolución. Las posibilidades que ofrecen estos elementos son como las de los anteriores, tanto en modelado como en programación, tratándose de la misma forma. En la Figura 4 se pueden ver dos ejemplos de modelos.

Las cintas transportadoras circulares se suelen empalmar con cintas transportadoras lineales. En estos casos hay que controlar en la aplicación que las piezas se liberen de una cinta y después se agarren con otra para producir movimientos continuos.

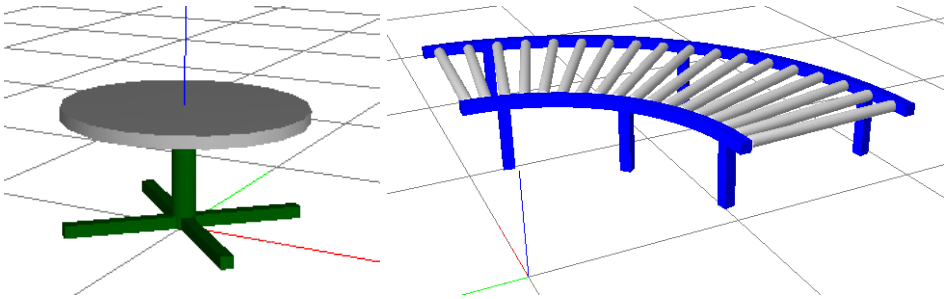

Figura 4. Ejemplos de modelos de mesa giratoria y cinta transportadora circular en *VirtualRobot*

Tarea: Modela una nueva mesa giratoria o una cinta transportadora circular partiendo de algún modelo existente. Prográmala para que agarre y mueva alguna pieza disponible en el entorno.

6. Acoplamiento de una herramienta a un robot industrial

Los robots manipuladores industriales necesitan tener acopladas herramientas para poder realizar acciones en su entorno. Como herramientas típicas tenemos las pinzas y ventosas que sirven para manipular piezas, o las herramientas de soldadura, pintura, mecanizado, etcétera que operan sobre piezas.

VirtualRobot dispone de algunas herramientas en "Models\Tool Systems" para simular herramientas de soldadura al arco y por puntos, de pintura por spray, de mecanizado, ventosas y pinzas neumáticas y (supuestamente) servocontroladas. Para acoplar una de estas herramientas a un robot que no tenga herramienta se dispone de la aplicación "VRM Tool Attach" en el menú "File>>VRM Tools".

Para acoplar una herramienta a un robot (sin herramienta) se ejecuta "VRM Tool Attach" seleccionando un robot y una herramienta y acoplándola con el botón "Attch Tool". En la ventana de localización de la herramienta, en principio sólo hay que modificarla en casos

especiales, si el modelado de la herramienta no es coherente con el toolFrame del robot o si tras haberlo probado detectamos que no se ha acoplado correctamente.

Tarea: Prueba acoplar la herramienta "ArcWeld1" de "Models\Tool Systems\Arc Weld Guns" al robot "kuka kr 125 L90" disponible en "Models\Robots\Kuka\heavy payload" y observa el resultado y en concreto la orientación de la herramienta. Repite el proceso salvando con otro nombre y cambia el valor Gamma a 0º, comparando el resultado.

7. Modelado de una herramienta para acoplar a un robot industrial

Un sistema de herramienta se define en un fichero de formato TSF[6] que tiene el siguiente formato (considerando el caso más sencillo: sin adaptador y con sólo una herramienta):

```
; TOOL SYSTEM DESCRIPTION FILE (TSF)
; This file describes a Tool as a Tool

[GENERAL]                                   ; General Information
..

[CONFIGURATION]                             ; Tool System Configuration
..

[TCP1]                                      ; Tool Centre Point 1
..

[TCP2]                                      ; Tool Centre Point 2
..
..                                  ; As many TCPs as defined in No_TCPs (even none)

[TOOL1]                                     ; Tool section is Optional according to No_Tools field
..

[TOOL1_STATUS1]                             ; Effect for Tool1 on Status 1 (Optional section)
..
[TOOL1_STATUS1_PRIMITIVE1]                  ; Tool1 on Status 1 (First Tool Status is the default one)
..
[TOOL1_STATUS1_PRIMITIVE2]                  ; Defined as a Primitive in Object Geometric Description File
..                                  ; As many Primitives as required to define Tool Status (at least 1)

[TOOL1_STATUS2_PRIMITIVE1]                  ; Tool1 on Status 2
..                                  ; As many Tool Status as required to define Tool (at least 1)
```

No existe ninguna aplicación que permita generar ficheros con este formato, por lo que se debe realizar manipulando ficheros ASCII.

Las siguientes recomendaciones se deben tener en cuenta para modelar sistema de herramienta:

- Se modelarán objetos que representen cada uno de los TOOL_STATUS de la herramienta. Lo habitual es modelar un estado básico y después ir salvando copias con modificaciones sobre este estado. Por ejemplo, para una pinza, se puede modelar con los dedos abiertos y salvar como pinza0 y si se quieren cuatro estados más, ir cerrando progresivamente los dedos para salvar estados diferentes como objetos pinza1, …, pinza4. Estos cinco objetos deben usarse como base para crear cinco TOOL1_STATUS.
- Para evitar el uso de transformaciones en el paso de acoplar la herramienta a un robot (botón "Attach Tool" de la aplicación "VRM Tool Attach") es conveniente que se conozca primero cómo es el sistema de coordenadas final de herramienta *toolFrame0* del robot que se va a usar. Generalmente tendrá la Z apuntando hacia afuera de la brida final del robot.

[6] El formato TSF está descrito en el en el documento VRFFD (VirtualRobot File Format Definition)

En este caso, hay que modelar la herramienta con respecto el eje Z del mundo para que se acople correctamente al robot. Es decir, un cilindro modelado con el eje a lo largo del Z del mundo aparecerá con su eje a lo largo del eje Z del sistema de herramienta del robot.

- Con el uso de "*Copy&Paste*" se montará un fichero TSF (se puede partir de uno de los ya existentes para una herramienta similar a la deseada) con la estructura anterior. Una vez más, habrá que tener gran cuidado con la numeración de los clausulas TOOL1_STATUS y PRIMITIVE.

Tarea: Modela una herramienta tipo pinza con dos estados (abierto y cerrado) y acóplala a un robot de Kuka verificando su correcto funcionamiento.

8. Modelado de un robot articulado

El modelado de un robot es bastante complicado. La única ayuda es partir de un robot que tenga una estructura cinemática similar. Entonces se deben ajustar los parámetros cinemáticos del fichero RKF y la geometría del fichero RGF. Para esto, se pueden obtener primero ficheros OBF de los elementos del robot original, cambiar las primitivas por nuevas primitivas que modelen el brazo-robot y volver a montar un nuevo fichero RGF (y el RKF compartiendo nombre). Los OBF generados deben estar modelados de tal forma que si los cargamos todos juntos en *VritualRobot* se vea el robot completo en su configuración de sincronismo.

Realmente resulta muy complicado y sólo es viable este modelado para expertos en *VirtualRobot* que sepan cómo funciona internamente en el modelado de los robots.