ENTREGABLE
PROYECTOS—
2023

DESARROLLO DE UN SISTEMA DE GENERACIÓN DE TRAYECTORIAS COMPLEJAS EN ROBOTS BASADO EN APRENDIZAJE POR DEMOSTRACIÓN "ROBOTRACK"

Entregable: E5.1- Desarrollo e integración de sistemas

Número de proyecto: 22200073
Expediente: IMDEEA/2022/9
Duración: Del 01/07/2022 al 30/09/2023
Coordinado en AIDIMME por: JOSÉ LUÍS SÁNCHEZ

AIDIMME
INSTITUTO TECNOLÓGICO

ÍNDICE

A. OBJETIVO DEL ENTREGABLE

Este entregable recoge las actividades realizadas durante la ejecución del **PT5 - Desarrollo del sistema software de conversión y transferencia de trayectorias y acciones**

El objetivo de este paquete de trabajo era desarrollar el sistema completo que permita, a partir de los datos capturados, analizar los mismos, convertirlos en un programa de robot que imite las acciones capturadas, probar el programa en un simulador virtual y finalmente transferirlo al controlador del robot.
Todos estos sistemas se debían integrar en el interfaz HMI, de forma que se garantizase su usabilidad por parte de usuarios industriales.

El entregable 5.1 recoge los análisis, diseños y desarrollos llevados a cabo para dar forma final y completa la aplicación Robotrack. Continuando con los desarrollados generados en el paquete de trabajo PT4, se ha desarrollado el sistema de suavizado y granulado de los datos capturados en bruto, la generación de la simulación y el script final bajo la sintaxis de programación de un modelo concreto de robot.

B. ACTIVIDADES REALIZADAS

1. SISTEMA DE ANÁLISIS DE TRAYECTORIAS Y ACCIONES

El sistema de análisis de trayectorias y acciones está compuesto por los pasos de suavizado y granulado de los datos capturados del movimiento de la mano derecha durante el proceso de demostración humana. En esta sección se desarrollará el funcionamiento de ambos.

1.1. Suavizado de datos

El objetivo de esta etapa es eliminar las posibles imperfecciones que por distintas razones puedan aparecer en la trayectoria ejecutada, como la imprecisión de la cámara, el propio movimiento del operario o cualquier otro posible defecto producido en el proceso de toma de datos, así como distribuir uniformemente los puntos tomados para facilitar la tarea al robot.

Para ello, se ha programado un algoritmo basado en medias móviles, que permite en base al número de periodos que el usuario de la interfaz desee, obtener una trayectoria suavizada que elimine los errores propios del proceso de captura de datos.

El usuario puede elegir la fuerza con la que se va a aplicar el suavizado mediante un formulario. El frontend entonces envía una petición HTTP al backend en la que incluirá el dato de fuerza y el token de datos. El backend crea un archivo de datos suavizado aplicando una media móvil a los datos originales, extrayendo la cantidad de puntos sobre los que promediar a partir del valor de fuerza proporcionado por el frontend. Por último, el backend devuelve los datos suavizados al frontend que los muestra al usuario. Éste tendrá la posibilidad de suavizar con otro valor de fuerza o bien pasar a la etapa siguiente.

En la siguiente imagen se muestra un ejemplo de trayectoria en bruto registrada por el sistema de captura de datos, así como el resultado de la fase de suavizado aplicando el algoritmo de medias móciles para cinco y diez periodos respectivamente.

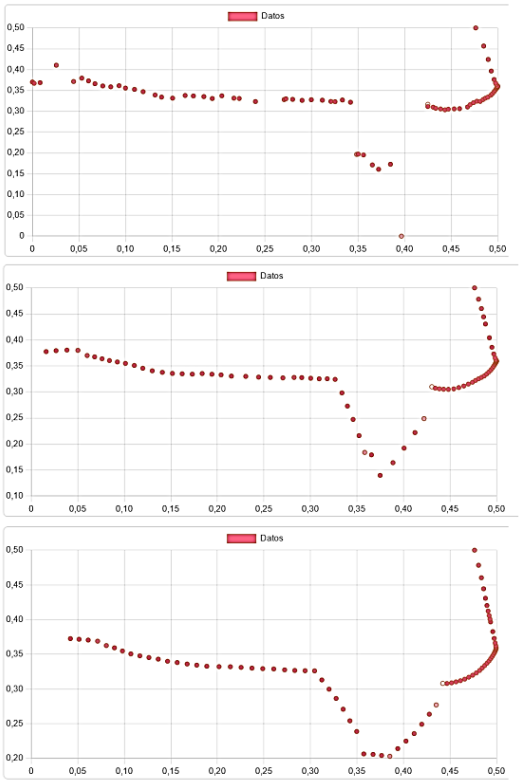

Figura 1 - Curva sin suavizar, suavizada con 5 puntos y con 10
Fuente: elaboración propia

1.2. Granulado de datos

El proceso de granulado reduce el número de puntos presentes en la muestra dejando los más representativos para correcta ejecución de la trayectoria. El objetivo de este proceso es reducir los parones que puede sufrir el robot al llegar a cada punto incluido en la muestra, así como simplificar el programa de robot.

El proceso es muy similar al caso anterior, el frontend manda una petición al backend, este recibe el token y la fuerza de granulado y se ocupará de crear un nuevo archivo. En este caso, el sistema buscará eliminar aquellos puntos que no tengan la distancia suficiente del punto anterior, obteniendo esta distancia de la fuerza de granulado. En la imagen se puede ver el resultado de aplicar un granulado de fuerza 3 y 6 al conjunto de datos introducido anteriormente.

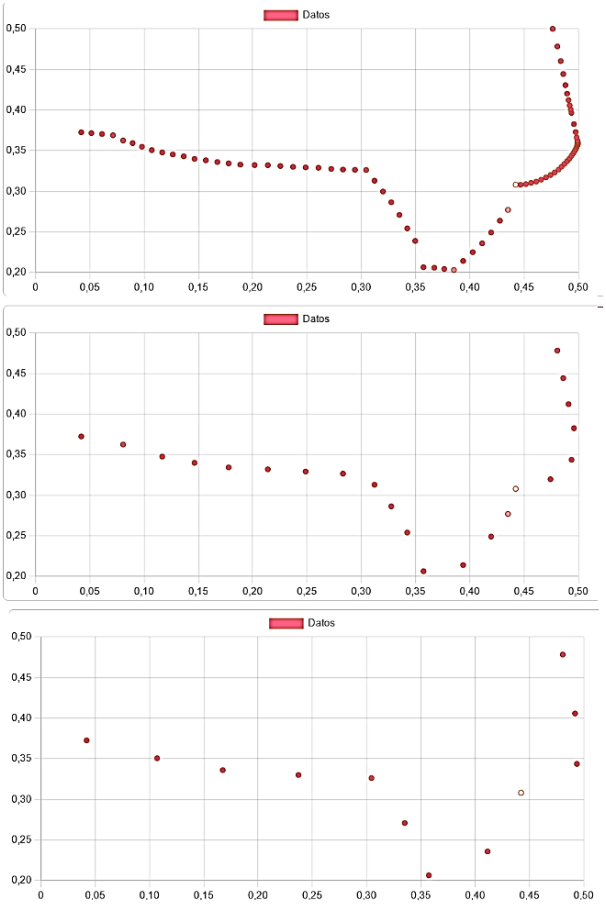

Figura 2 - Curva sin granular, y granulada con fuerza 3 y 6
Fuente: elaboración propia

2. SISTEMA DE GENERACIÓN DE PROGRAMA DE ROBOT

Para el buen funcionamiento de Robotrack se requiere la definición de algunos parámetros cuyo objetivo es ajustar el funcionamiento de la aplicación a distintas situaciones y casuísticas, que permitirán la generación final del programa de robot. En esta sección se revisa el funcionamiento de estas variables y el efecto que produce su modificación.

Offset. Las variables de offset, permiten ajustar la distancia desde la base del robot a la que va a operar el robot. El robot toma como punto (0,0,0) su propia base, por lo que, si no hay un ajuste de distancia, este fallará al chocarse consigo mismo. La variable "*offsetx*" permite señalar el desplazamiento en el eje x necesario para marcar el origen de coordenadas real y por su parte la variable "*offsety*" realiza lo propio en el eje y.

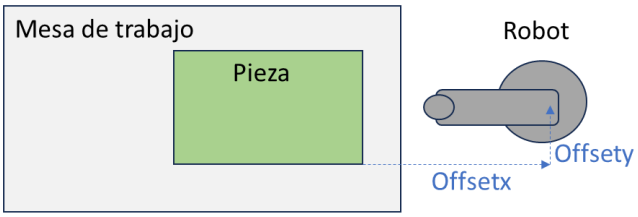

Figura 3 – Representación de Offsetx y Offsety
Fuente: elaboración propia

Escala. Las variables de escala permiten ajustar las diferencias de perspectivas existentes entre la toma de datos mediante el dispositivo de captura. De esta forma, la variable "*escalax*" realizará un escalado lineal con el valor indicado de todos los puntos en el eje x, por su parte la variable "*escalay*" se ocupa del otro eje.

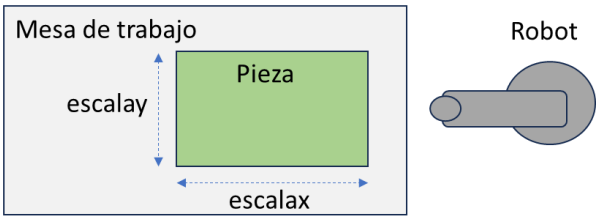

Figura 4 – Representación de escalax y escalay
Fuente: elaboración propia

Valor Z. El valor Z indicará al robot la altura a la que debe trabajar.

Factor de velocidad. Robotrack ajustará la velocidad de ejecución del robot dinámicamente dependiendo de la velocidad a la que haya realizado el operario que grabó los movimientos. Sin embargo, es posible ajustar el factor base desde el que se calcula esta velocidad de movimiento en cada punto.

Blend Radius. Se trata de un parámetro que marca un radio de curvatura a la hora de aproximarse a cada punto de la trayectoria desde el punto anterior de la misma. Este radio de curvatura hace que los movimientos del robot sean más suaves y fluidos. Si este parámetro está a cero, el robot se para en cada punto de la trayectoria.

Todos estos parámetros de configuración son tenidos en cuenta a la hora de generar el script final en el lenguaje de programación propio del robot que se vaya a utilizar, y están definidos en el backend de la aplicación.

Para que el sistema sea capaz de generar tanto una simulación como el programa final del robot, es necesario valerse de RoboDK (https://robodk.com/es/), un software de simulación y programación de robots industriales desarrollado por RoboDK Inc.

Figura 5 – Logo software RoboDK
Fuente: elaboración propia

El software dispone de una conexión con Python mediante una API externa. Para conectarse con esta API de manera sencilla, se utiliza la librería RoboDK de Python.

La librería RoboDK de Python está formada por un conjunto de herramientas y funciones diseñado para facilitar la interacción entre el software RoboDK y el lenguaje de programación Python. Esta librería actúa como un puente, permitiendo a los desarrolladores controlar y automatizar tareas en RoboDK utilizando scripts de Python. Entre sus principales funciones, se encuentra la capacidad de cargar modelos de robots y herramientas, definir y modificar trayectorias, simular movimientos del robot y generar programas para distintos controladores de robots. Asimismo, permite la integración con otros softwares y hardware, gracias a su capacidad para recibir y enviar información a través de la API.

Cuando se inicia el backend, Python inicia a su vez una instancia de RoboDK con la que comunicarse y cargará un robot en escena. En el caso del presente proyecto se han creado estaciones de trabajo con robots UR5 y UR16, ya que son los robots de AIDIMME planteados para ser utilizados durante el proyecto.

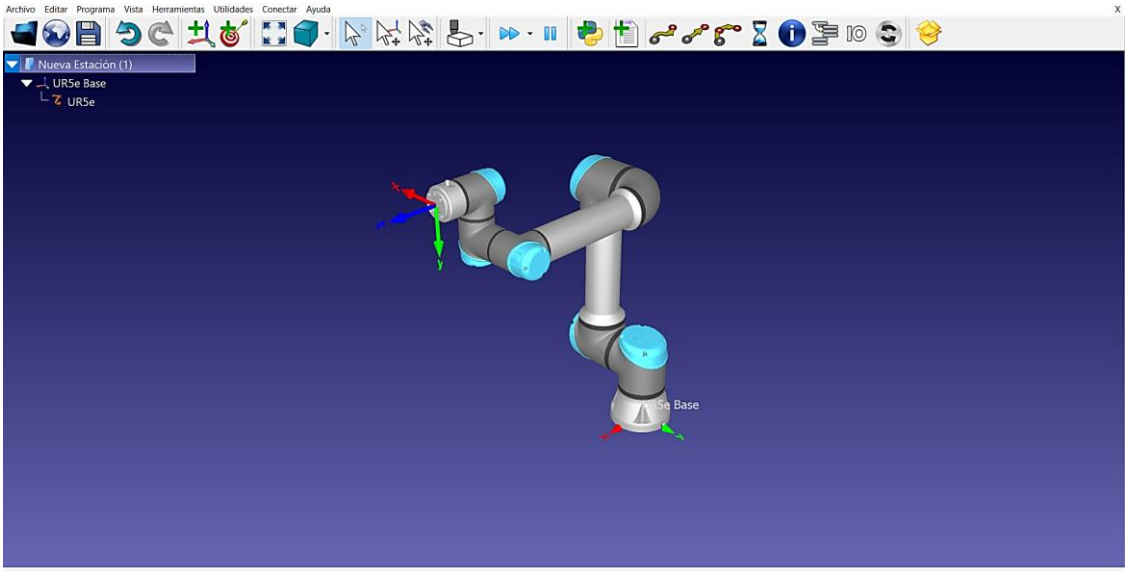

Figura 6 – Ejemplo de estación de trabajo creada para un robot UR5
Fuente: elaboración propia

El endpoint /generar_programa recibe una petición POST en la que se incluyen en el cuerpo de la petición un archivo JSON indicando el token identificador de la grabación de datos y el tipo de movimiento que se desea utilizar, siendo las posibilidades:

- Movimiento en L. Durante este tipo de movimiento, el robot se asegura de que la trayectoria del efector final sea una línea recta, sin importar cómo se tengan que mover las articulaciones del robot para lograrlo.

- Movimiento en J. Durante un movimiento en J, la trayectoria que sigue el efector final no es necesariamente una línea recta. Lo que importa aquí es el movimiento de las articulaciones individuales.

- Primer movimiento en J y resto en L

Python utilizará el token para obtener el último archivo de datos disponibles de granulado con esos datos y para cada punto utilizará una instrucción *program.movej* o *program.movel* para incluir en el programa, tal como se muestra en la ilustración.

```
if tipoMovimiento == "L" or tipoMovimiento == "M":
    program.MoveL(TxyzRxyz_2_Pose(
        [int(movement["x"]*1000)+offx, int(movement["y"]*1000)+offy, 300, 0, pi, -pi/2]))
else:
    program.MoveJ(TxyzRxyz_2_Pose(
        [int(movement["x"]*1000)+offx, int(movement["y"]*1000)+offy, 300, 0, pi, -pi/2]))
```

Figura 7 - Código para la creación de instrucciones de robot
Fuente: elaboración propia

Por último, Python utiliza *MakeProgram* para generar el archivo .script y lo enviará al frontend utilizando la función *send_file*.

3. SIMULADOR DE TRAYECTORIAS

Para realizar la simulación de la ejecución, se va a utilizar una vez más, el software RoboDK y la librería robodk de Python. La API de RoboDK no incluye ninguna funcionalidad propia para generar vídeos de una simulación, es por esto por lo que ha sido necesario programar esta parte utilizando Python.

Para obtener el efecto deseado se ha incluido un nuevo eje de coordenadas en el proyecto, y en este se ha incluido una cámara. Este elemento de RoboDK permite obtener una imagen 2D del entorno 3D desde el punto de coordenadas en el que se ha fijado la cámara, tal como se muestra en la siguiente imagen.

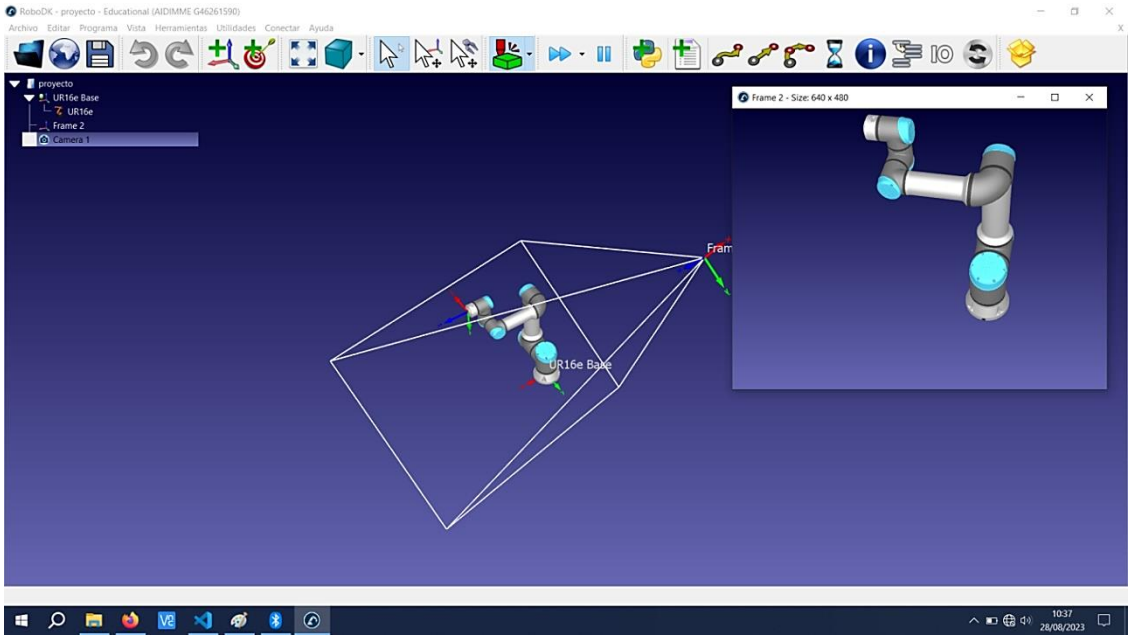

Figura 8 - Nuevo eje de coordenadas, cámara y visualización
Fuente: elaboración propia

El elemento de cámara incluye la capacidad de generar una instantánea en un momento dado y guardarla como archivo png.

Cuando el Backend recibe una petición al endpoint /simular, este se asegurará de que exista la cámara en el proyecto, y de no ser así, incluirá una nueva. Tras esto, Python ordena a RoboDK que realice movimientos a todos los puntos indicados por el conjunto de datos. Después de cada movimiento indicará al elemento de cámara que tome una instantánea que guardará en una carpeta destinada a tal efecto. Las instantáneas se numerarán en orden ascendente de forma que se mantenga la secuencia de ejecución en el orden correcto.

Una vez terminado el proceso de simulación, Python accederá a la carpeta de instantáneas y obtendrá todas aquellas que tienen el token solicitado y las incluirá en un nuevo archivo respetando el orden de generación. Este archivo será de tipo GIF, es decir, animado. Cuando el archivo está conformado, el backend lo enviará al frontend que tendrá la responsabilidad de mostrarlo al usuario.

4. COMUNICACIÓN Y TRANSFERENCIA DE PROGRAMA

Para la transferencia del programa generado a través de los desarrollos realizados, no se ha identificado en la API del software RoboDK una función que permita realizar la transferencia y ejecución de este al robot. Si se puede realizar desde un programa nativo generado desde el entorno de programación de RoboDK, pero no a través de las funciones de la API.

Esta situación no supone ningún problema, ya que el script generado se puede guardar en un pendrive, conectarlo al robot y cargarlo de forma manual sin ningún tipo de problema.

5. VALIDACIÓN DE DESARROLLOS

A continuación, se muestra el resultado de diferentes pruebas de validación llevadas a cabo para verificar el funcionamiento de los desarrollos descritos en los apartados anteriores, así como los ligados a la captura de datos durante la fase de demostración humana.

Estas pruebas suponen una fase previa e inicial a las que se desarrollan en el paquete de trabajo PT6, materializadas en un demostrador basado en un proceso industrial analizado previamente.

5.1. Sistema de captura de datos

Se accede a la aplicación del proyecto, donde la pantalla principal permite seleccionar el sistema de captura de datos a utilizar.

Figura 9 – Pantalla inicial de la aplicación Robotrack
Fuente: elaboración propia

Se selecciona la cámara que captura la posición (coordenadas x, y) de las articulaciones del usuario que realiza la demostración.

Figura 10 – Selección del sistema de captura de datos
Fuente: elaboración propia

Se activa la aplicación desarrollada para la cámara Kinect. El usuario realiza un movimiento en diagonal con la mano derecha, quedando los puntos registrados tal y como se muestra en la siguiente figura.

Figura 11 – Resultado de la grabación de los datos capturados por la cámara Kinect
Fuente: elaboración propia

5.2. Sistema de análisis de trayectorias y acciones

Al pulsar sobre el botón “Siguiente” se accede a la pantalla de suavizado.

Figura 12 – Selección de una fuerza de suavizado “4”
Fuente: elaboración propia

Se selecciona una fuerza de suavizado de “4”. Al pulsar sobre el botón “Suavizar” se aplica un modelo de medias móviles de cuatro periodos y muestra la nueva serie de datos en pantalla, tal y como se muestra en la siguiente figura.

Figura 13 – Resultado de haber aplicado el proceso de suavizado
Fuente: elaboración propia

Al pulsar sobre el botón “Siguiente” se accede a la pantalla de granulado.

Figura 14 – Selección de una fuerza de granulado "5"
Fuente: elaboración propia

Se selecciona una fuerza de granulado de "5". Al pulsar sobre el botón "Granular" se aplica un modelo donde los puntos de la serie se espacian y muestra la nueva serie de datos en pantalla, tal y como se muestra en la siguiente figura.

Figura 15 – Resultado de haber aplicado el proceso de granulado
Fuente: elaboración propia

5.3. Simulador

Se pulsa en el botón "Siguiente" y se accede a la pantalla de simulación. En pantalla aparece un video donde se ve como el robot realiza una trayectoria pasando por los diferentes puntos de la trayectoria final generada.

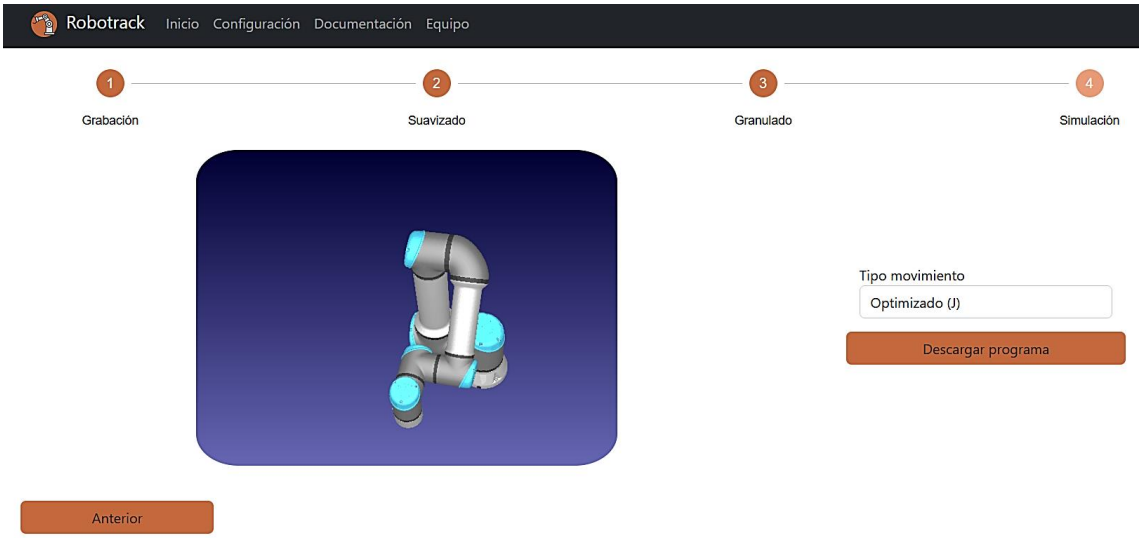

Figura 16 – Resultado de haber aplicado el proceso de simulación
Fuente: elaboración propia

5.4. Sistema de generación de programa de robot

Se pulsa sobre el botón “Descargar programa” y se muestra el avance la descarga del script para el robot. A continuación, se muestra un ejemplo de parte del código de programa generado en URscript para un robot UR16.

```
def ProgUR16e():
  # Global parameters:
  global speed_ms    = 0.250
  global speed_rads  = 0.750
  global accel_mss   = 1.200
  global accel_radss = 1.200
  global blend_radius_m = 0.001
  global ref_frame = p[0,0,0,0,0,0]

  #
  # TO REMOVE HEADER:
  # Go to "Program" -> "Post-Processor Editor"
  # Select "Universal Robots"
  # Set "INCLUDE_HEADER" to "False"

  # Main program:
  ref_frame = p[0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000]
  set_tcp(p[0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000])
        movej([1.566210,        -1.607659,        -2.271982,        -0.832745,        1.570796,
1.566210],accel_radss,speed_rads,0,0) # end trace
  speed_ms    = 0.054
        movej([1.658211,        -1.351754,        -2.486937,        -0.873696,        1.570796,
1.658211],accel_radss,speed_rads,0,0) # end trace
  speed_ms    = 0.137
        movej([1.409692,        -1.252574,        -2.541339,        -0.918478,        1.570796,
1.409692],accel_radss,speed_rads,0,0) # end trace
  speed_ms    = 0.005
        movej([1.472028,        -1.497231,        -2.378779,        -0.836377,        1.570796,
1.472028],accel_radss,speed_rads,0,0) # end trace
  speed_ms    = 0.002
        movej([1.387116,        -1.252173,        -2.541531,        -0.918687,        1.570796,
1.387116],accel_radss,speed_rads,0,0) # end trace
```

```
  speed_ms    = 0.005
        movej([0.954039,        -1.379074,        -2.469309,        -0.863999,        1.570796,
0.954039],accel_radss,speed_rads,0,0) # end trace
  speed_ms    = 0.011
        movej([0.709052,        -1.605338,        -2.274461,        -0.832597,        1.570796,
0.709052],accel_radss,speed_rads,0,0) # end trace
  speed_ms    = 0.009
        movej([0.981140,        -1.495543,        -2.380245,        -0.836603,        1.570796,
0.981140],accel_radss,speed_rads,0,0) # end trace
  speed_ms    = 0.446
        movej([1.344680,        -1.288988,        -2.523048,        -0.900356,        1.570796,
1.344680],accel_radss,speed_rads,0,0) # end trace
# End of main program
end
ProgUR16e()
```

Figura 17 – Programa de robot generado por medio de la aplicación Robotrack
Fuente: elaboración propia

5.5. Sistema de comunicación y transferencia de programa

En la siguiente imagen se puede ver como se ha cargado un ejemplo de programa en URscript ("robotrack_prueba1.script") generado por medio de la interfaz HMI desarrollada en Robotrack. La prueba se ha realizado en el programa Polyscope, entorno de programación de los robots Universal Robots.

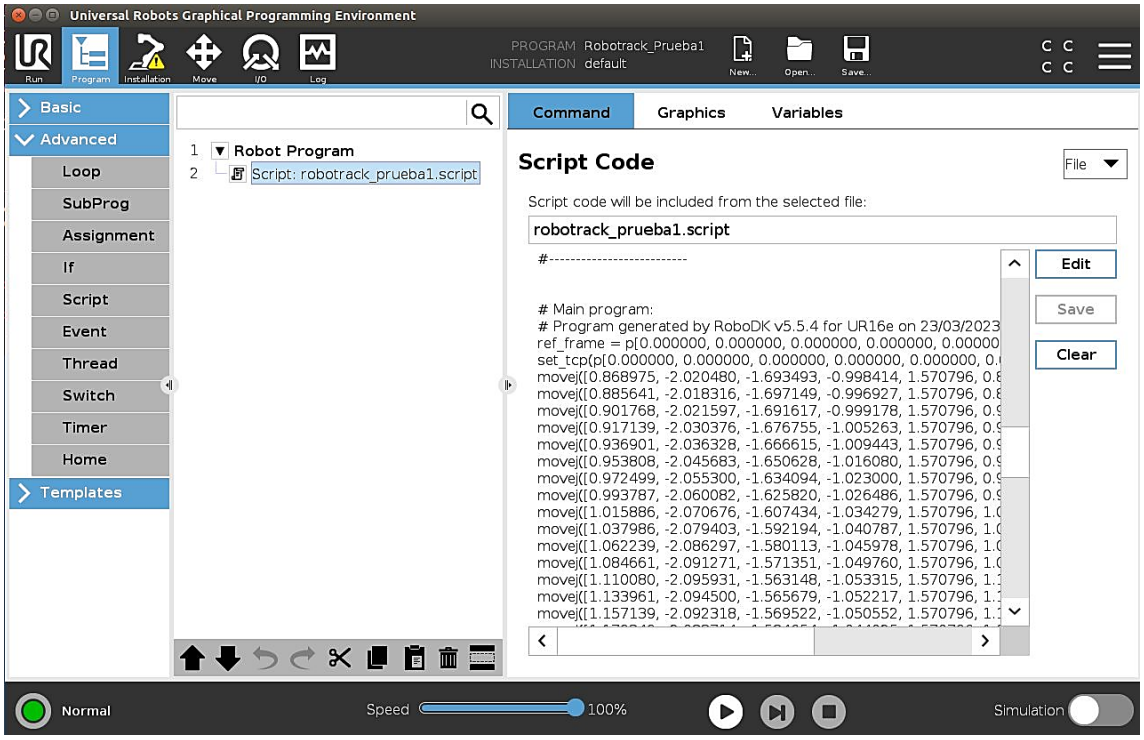

Figura 18 – Script generado por la HMI de Robotrack y cargado en Polyscope
Fuente: elaboración propia

En la siguiente imagen se puede ver como se simula desde el propio entrono Polyscope, la ejecución del programa generado desde la HMI de Robotrack.

AIDIMME INSTITUTO TECNOLÓGICO
GENERALITAT VALENCIANA
iVACE INSTITUTO VALENCIANO DE COMPETITIVIDAD EMPRESARIAL
Financiado por la Unión Europea

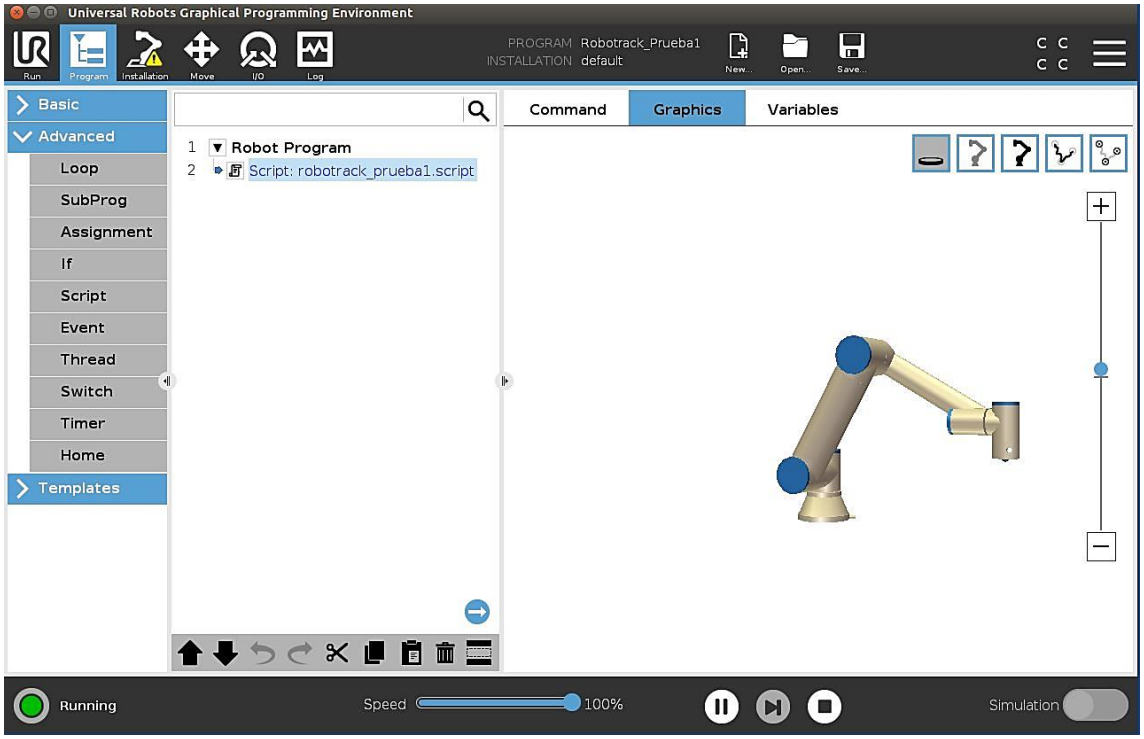

Figura 19 – Ejecución en Polyscope del script generado en la HMI de Robotrack
Fuente: elaboración propia

A continuación, se muestra también una fotografía del "pendant" (consola de programación) de un robot UR16 sobre el que se probó la transferencia y ejecución de un programa generado a través de la HMI Robotrack.

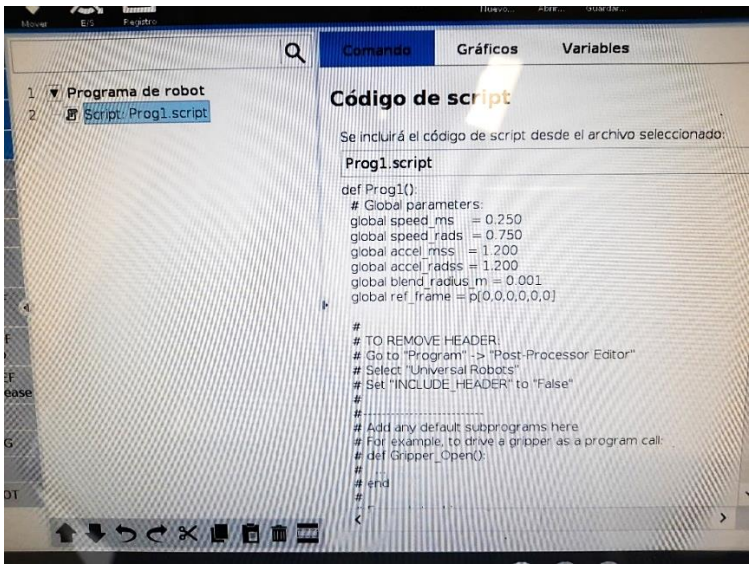

Figura 20 – Programa cargado en el pendant de un robot UR16
Fuente: elaboración propia

C. RESUMEN Y CONCLUSIONES

Se ha desarrollado una aplicación que, a partir de los datos en bruto generados por la cámara de visión, permite al usuario realizar un suavizado y granulado de los mismos.

El proceso de suavizado se genera a partir de un modelo de medias móviles donde el usuario del sistema puede definir cuantos periodos quiere. Durante le proceso de granulado el usuario puede seleccionar cada cuantos centímetros quiere marcar un punto de la trayectoria. Hay que tener en cuenta que un robot no puede ejecutar trayectorias donde los puntos estén separado muy poca distancia.

Además de estos dos procesos, el usuario puede generar un video donde se simula la ejecución de la trayectoria creada, y se genera un programa con instrucciones en la sintaxis de un modelo específico de robot.

Esta etapa de generación del programa final se apoya en determinadas funciones de la API del software RoboDK.

Con la colaboración de:

AIDIMME
INSTITUTO TECNOLÓGICO

Domicilio fiscal —
C/ Benjamín Franklin 13. (Parque Tecnológico)
46980 Paterna. Valencia (España)
Tlf. 961 366 070 | Fax 961 366 185

Domicilio social —
Leonardo Da Vinci, 38 (Parque Tecnológico)
46980 Paterna. Valencia (España)
Tlf. 961 318 559 - Fax 960 915 446

aidimme@aidimme.es
www.aidimme.es