Imagen que contiene Diagrama

Descripción generada automáticamente Logotipo

Descripción generada automáticamente con confianza media

**DIARIO DE LABORATORIO**

**LABORATORIO DE INTEGRACIÓN SENSORIAL E INTERACCIÓN PERSONA-ROBOT**

MÁSTER EN INGENIERÍA INDUSTRIAL

**Malena Potesta González**

***Curso 2023-2024***

**Índice**

# Día 30/11/2023; Definición de etapas y objetivos

Definición de las etapas del trabajo. El diagrama de flujo del trabajo es el siguiente:

Diagrama

Descripción generada automáticamente

Figura 1.- Fases del trabajo

Para conectar RoboDK con Matlab tiene que estar abierta la simulación en RoboDK antes de ejecutar el código. Los códigos de ejemplo están en la web de RoboDK: <https://robodk.com/Matlab-API>

# Día 03/12/2023: Estudio del entorno de trabajo

Estudio de la conexión Matlab-RoboDK: En la *Comand Window* de Matlab poner doc Robolink y te lleva a una ayuda sobre la API. Para que funcione la programación desde Matlab tienen que estar en la misma carpeta el archivo .rdk y el .m. No olvidarse de establecerlos como *Current Folder* antes de ejecutar el código de Matlab.

En este vídeo se explica el funcionamiento de la API de la Matlab y cómo funcionan los movimientos en RoboDK. Es útil para saber programar los movimientos del robot: <https://www.youtube.com/live/gsmKY3Ine4w?si=XLI5ucRP2_L-fyIv> . Es necesario que en todos los scripts se incluyan las siguientes líneas, que son las que permiten establecer la conexión RoboDK-Matlab:

%%

% Generate a Robolink object RDK. This object interface with RoboDK

RDK = Robolink;

% Get the library path

path=RDK.getParam('PATH\_LIBRARY');

Consultar información sobre las matrices homogéneas y las rotaciones con RoboDK.

# Día 04/12/2023: Trabajos previos con RealSense

La aplicación de RealSense se descarga del siguiente link: <https://github.com/IntelRealSense/librealsense/releases/tag/v2.54.2> , el archivo está en Assets, es el Intel.Real.Sense.SDK-WIN10-2.53.1.4623.exe. Esta versión es la anterior a la 2.54.2, que aparece en el primer link. Es mejor descargar la versión anterior porque permite trabajar con Matlab (en la nueva no aparece esta opción). El link de descarga de la versión 2.53.1 es: <https://github.com/IntelRealSense/librealsense/releases/tag/v2.53.1>

En este link se encuentra una pequeña guía para trabajar con la cámara y Matlab: <https://github.com/adi226/Intel-RealSense-Guide/blob/master/Intel%20Realsense%20Guide.pdf>

Al trabajar con la cámara RealSense da un error al intentar capturar colores. La profundidad la captura correctamente. Probablemente, sea un problema de configuración de la misma. Aunque, tal vez sea más sencillo trabajar con una cámara RGB.

Computadora de escritorio sobre superficie de madera

Descripción generada automáticamente con confianza media

Figura 2.- Cámara RealSense

# 05/12/2023: Evaluación de cámaras y captura de movimiento con Mocap

La segunda parte del trabajo tiene que hacerse con una cámara RGB, que permite capturar colores y formas. El tratamiento de la imagen debe ser en blanco y negro. En el Campus Virtual está el fichero de Matlab con los pasos a seguir. El funcionamiento de la cámara es muy sencillo, basta con conectarla al ordenador a través del cable USB y tener en Matlab descargada la *toolbox* para trabajar con cámaras de vídeo.

Una cámara fotográfica

Descripción generada automáticamente con confianza media

Figura 3.- Cámara RGB

Captura de Mocap: Para una primera prueba del tratamiento de datos con Matlab para pasárselos al robot, se capturan una serie de movimientos con las cámaras creando un único sólido rígido. El objetivo es convertir este movimiento en puntos con coordenadas x, y, z que se le puedan pasar al robot como *targets*. El sistema de referencia de las cámaras y el robot no es el mismo, es por ello que se debe crear un nuevo sistema de referencia en RoboDK que coincida con el de las cámaras. Es necesario medir las distancias del *ground plane* de Mocap respecto de la base del robot.

Gráfico

Descripción generada automáticamente

Figura 4.- Ejes en Mocap. En rojo X, en verde Z y en azul Y

Diagrama

Descripción generada automáticamente

Figura 5.- Ejes de la base del UR3e. En rojo X, en verde Y y en azul Z

# Día 11/12/2023; Tratamiento de imágenes para machine learning

Dentro de la segunda parte del trabajo se busca que el robot sea capaz de distinguir piezas con grietas y sin ellas. Para evaluar el funcionamiento de la cámara RGB se utiliza el archivo de GitHub *script\_camara*. Con este archivo se realizan diferentes comprobaciones para garantizar la correcta captura de las imágenes.

Después de varias pruebas, se concluye que la cámara es muy sensible a la luz, de modo que para que pueda apreciar bien las formas de los legos se debe poner un fondo negro detrás del objeto a capturar. A su vez, la definición de la cámara no es alta y en la imagen que debe guardarse para, posteriormente, clasificarla, no se ve bien la grieta, lo que puede dificultar el tratamiento de las imágenes. Probablemente, sea necesario utilizar una cámara con mayor definición.

# Día 12/12/2023: Conexión RoboDK-Matlab y Matlab-UR3e

Una vez comprobado que la conexión RoboDK Matlab funciona queda establecer la conexión con el robot real. En primer lugar, se debe establecer la conexión en remoto del UR3e y RoboDK. Se debe verificar que están conectados a la misma red y establecer en la consola del UR3 la opción de control remoto. En RoboDK, en la pestaña Conectar en la IP se debe establecer la misma IP a la que está conectado el UR3e y darle a conectar. En Matlab, se debe escribir el comando robot.Connect(), si el resultado es 1 la conexión se ha hecho correctamente, si es 0 es que no se ha establecido la conexión. A continuación, ejecutar el comando robot.RunProgram(). Se puede comprobar que el robot se mueve tanto en la simulación como en la realidad (en esta página se explica como establecer la conexión y da información sobre estos comandos: <https://robodk.com/forum/Thread-Using-Matlab-to-control-real-robot> ).

En esta sesión, se ejecutó sobre el robot el scrip titulado myUR3 y se comprobó su correcto funcionamiento. Finalmente, se grabó con Mocap un movimiento para, posteriormente, tratarlo desde Matlab y poder crear un programa para que el robot siga esos movimientos. El *ground plane* en Mocap se estableció en la base del robot, si bien, esta no es la mejor posición para colocar el eje de referencia en RoboDK debido a que el robot tiene problemas para alcanzar las posiciones:

Imagen que contiene interior, tabla, cocina, mostrador

Descripción generada automáticamente

Figura 6.- *Ground plane* de Mocap en la captura del día 12/12/2023

# Día 13/12/2023: Preparación del script en Matlab

En esta sesión se consiguió crear un *srcipt* de Matlab que genera un programa en RoboDK con todos los puntos objetivo a los que debe moverse el robot. Para ello*,* se toma como ejemplo el *script* de la web <https://robodk.com/Matlab-API> , del apartado How to generate a robot program. Además, en la web hay un ejemplo de colocación de la herramienta que puede ser útil a futuro.

Al *script* se le añade una sección en la que los datos obtenidos como *quaternions* se transforman en matrices homogéneas.

%%

% Creación del programa de movimiento del robot

prog=RDK.AddProgram('My Prog');

% Create a joint target home

target = RDK.AddTarget('Home',B0,robot);

target.setAsJointTarget();

target.setJoints(Jrest);

% Add joint movement into the program

prog.MoveJ(target);

% define the pose (H transforms) Cambiar las unidades

W=W;

X=X;

Y=Y;

Z=Z;

X1=X1\*1000;

Y1=Y1\*1000;

Z1=Z1\*1000;

q=quaternion(W,X,Y,Z);

longq=length(q);

%%

for i=1:100:longq

targetname = sprintf('Target%i',i);

target = RDK.AddTarget(targetname, B0, robot);

ad=[zeros(4,3), [X1(i);Y1(i);Z1(i);0]];

qh=quat2tform(q(i))+ ad;

disp(qh);

target.setPose(qh);

prog.MoveJ(target);

end

%%

robot.Connect();

% Run the program we just created

prog.RunProgram();

Mocap da problemas a la hora de capturar el movimiento como sólido rígido, pierde muchos datos y la trayectoria representada es muy pequeña, de modo que el robot no puede alcanzar esos targets.

# Día 14/12/2023: Solución de problemas con Mocap y creación del programa de movimiento en Matlab y RoboDK

La sesión comienza con la correcta colocación de las cámaras entorno al espacio de movimiento del robot. Además, se adecua el laboratorio para evitar reflejos indeseados y se realizan varias pruebas con las cámaras hasta que se consigue que dejen de captar como marcadores destellos del ambiente.

A continuación, se decide que para garantizar que los movimientos grabados por las cámaras son realizables por el UR3e lo mejor es colocar marcadores sobre el extremo del robot y grabar su propio movimiento. Para ello, desde el panel de control del UR3e, con el comando *Mover* se le manda a la posición indicada en la Figura 7 y se captura el movimiento del robot desde la posición de inicio al nuevo objetivo y desde el nuevo objetivo a inicio. El *ground plane* se coloca sobre la mesa, próximo al robot.

Pantalla de juego de computadora

Descripción generada automáticamente con confianza media

Figura .- Posición objetivo del UR3e

Una sala de estar

Descripción generada automáticamente con confianza media

Figura .- Colocación de los marcadores de Mocap en el UR3e

Lo siguiente es importar los datos a Matlab y eliminar aquellos que no son válidos. Además, se adaptó el *script* para que de los más de 1800 datos solo se consideraran como objetivo 18. Debe tenerse en cuenta que en Mocap las distancias se miden en m y en RoboDK en mm, de modo que los datos calculados por Mocap deben pasarse a mm. Otro punto a tener en cuenta es que los ejes deben colocarse adecuadamente en RoboDK porque pequeñas variaciones en su posición hacen que el robot no pueda alcanzar los objetivos.

for i=1:100:longq

targetname = sprintf('Target%i',i);

target = RDK.AddTarget(targetname, B0, robot);

ad=[zeros(4,3), [X1(i);Y1(i);Z1(i);0]];

qh=quat2tform(q(i))+ ad;

disp(qh);

target.setPose(qh);

prog.MoveJ(target);

end

El resultado final en RoboDK es:

Diagrama, Esquemático

Descripción generada automáticamente

Figura .- Espacio de trabajo en RoboDK