UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dept. Ingeniería de Sistemas y Automática (DISA)

Práctica 3

Entorno de trabajo: Coppeliasim EDU y Matlab

Mecatrónica
Grado en Ingeniería Informática

Alejandro Vignoni

Departamento de Ingeniería de Sistemas y Automática (DISA)

Objetivos

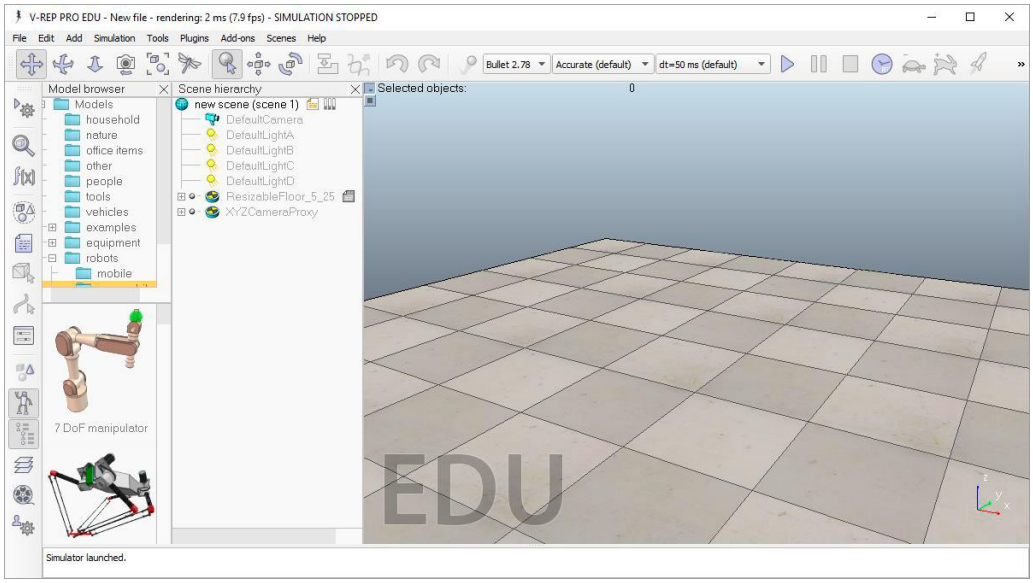
- ✓ Presentar la utilización del software Coppeliasim EDU para la simulación de robot.
- ✓ Simular el comportamiento del mini-robot LEGO Mindstorm EV3 en dicho software.
- ✓ Utilizar la herramienta de simulación Coppeliasim EDU junto con Matlab para estudiar y calibrar los sensores presentes en robots móviles

Instalación de Coppeliasim

Para poder simular nuestro robot, se utiliza el entorno de simulación Coppeliasim EDU. En primer lugar, hemos de instalar Coppeliasim EDU en nuestro equipo. Para ello basta con instalar los paquetes que aparecen en su web:

https://www.coppeliarobotics.com/downloads

Con ello, nos descargaremos el archivo de instalación al ordenador para posteriormente extraerlo (Linux) o instalarlo (Windows). Una vez concluida la instalación ya podemos empezar a utilizar el entorno, como el que aparece a continuación.

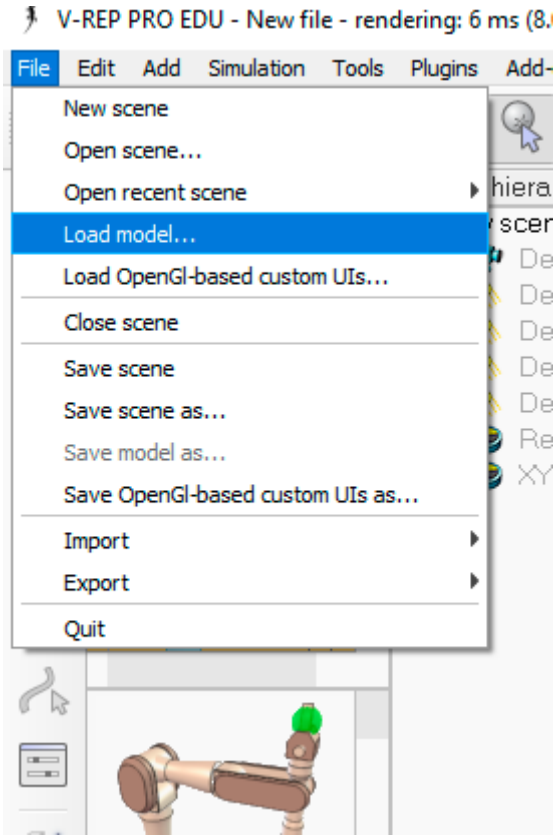


Modelo del LEGO Mindstorm EV3 en Coppeliasim

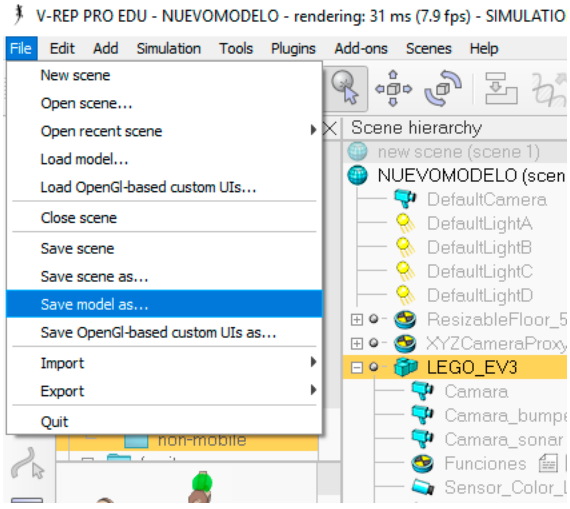
En el simulador, para cargar el modelo solo tendremos que ir, en la barra de menús, a "File → Load model…", y buscar, en el directorio donde se han extraído los ficheros descargados de poliformaT, en la carpeta *Modelo*, los dos modelos disponibles de EV3 (Alberto Martín Domínguez, Universidad de Málaga, https://github.com/albmardom/EV-R3P):

- "Modelo_Lego_EV3.ttm". Este modelo es el más detallado, pero, por el contrario, es el más pesado para renderizar; no recomendable para equipos de bajo rendimiento.
- "Modelo_Lego_EV3_Lightwight.ttm". Este modelo es menos detallado que el anterior, pero en cambio es más eficiente en la renderización;

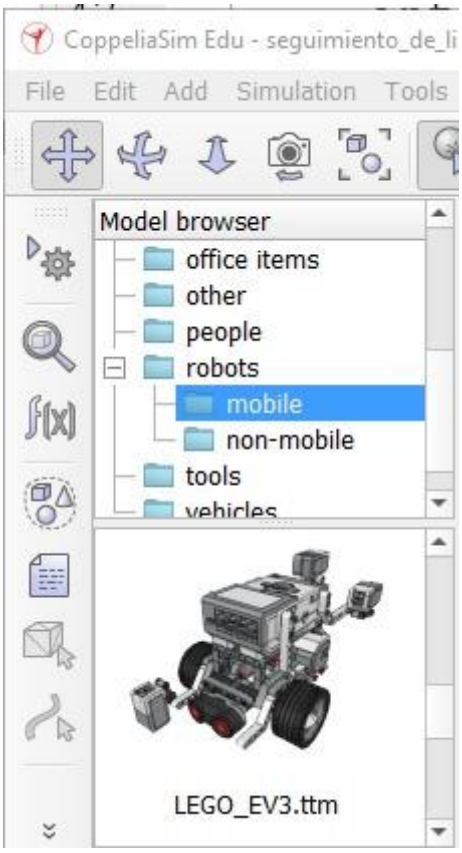
recomendado para ordenadores que no puedan utilizar fluidamente el modelo anterior.



Si se quiere que el modelo cargado aparezca en el Explorador de Modelos del simulador, basta con guardar el modelo en "File → Save model as…" (teniendo el objeto "LEGO_EV3" seleccionado en la escena), en el siguiente directorio: <DIRECTORIO_INSTALACIÓN >/Models/robots/mobile.

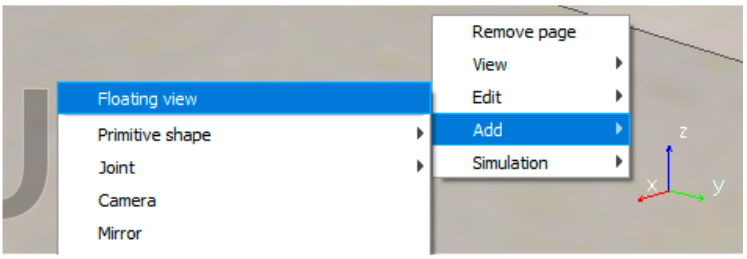


Hecho esto, los modelos deberían aparecer en el Explorador de Modelos, situado a la derecha de la pantalla.

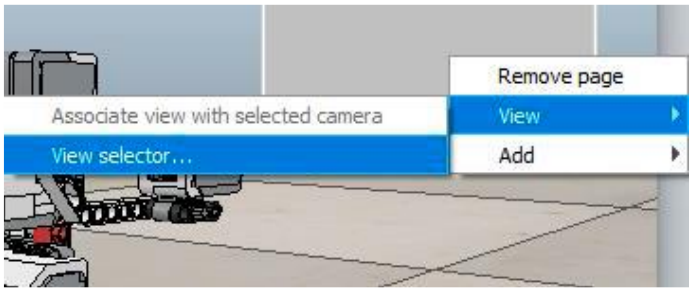


Vistas

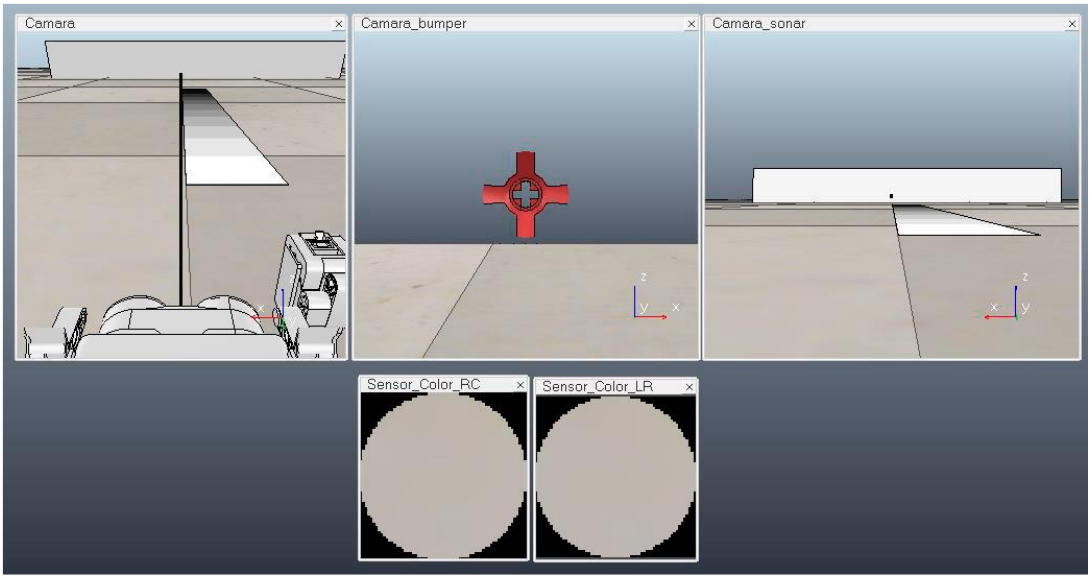
A ambos modelos se les ha dotado de cámaras para ayudar al desarrollo de programas. Podemos obtener imágenes de ellas si añadimos vistas flotantes; estas se añaden haciendo clic derecho en la escena y luego "add → Floating view".



Ahora, haciendo clic derecho dentro de la vista flotante, seleccionamos "view → View selector…" y ahí seleccionaremos la cámara que queramos ver.



Este procedimiento también sirve para ver las imágenes del sensor de visión, como, por ejemplo, los modos del sensor de color del EV3; en la figura siguiente se muestran todas las cámaras del robot como vistas flotantes.



Escenas

Además, se proporcionan escenas, que pueden servir de apoyo para el alumno a la hora de comprobar el funcionamiento de robot:

- ✓ "escenaTest.ttt". Esta escena está indicada para probar toda la toolbox mediante el test proporcionado (tests.m).

Instalación Toolbox en Matlab.

Una vez instalado Coppeliasim podemos iniciar Matlab para proceder a instalar el Toolbox de comunicación entre Matlab y el simulador. Para ello, en Matlab, ir a la Carpeta donde se han extraído los ficheros descargados de poliformaT, en la subcarpeta Toolbox, hacer doble click el archivo "LEGO EV3 V-REP toolbox x64 (NXC version).mltbx" (o el x86 en caso de tener instalado Matlab 32bits). A continuación, Matlab instalará la toolbox (Alberto Martín Domínguez, Universidad de Málaga, https://github.com/albmardom/EV-R3P).

Para comprobar si se ha instalado correctamente la toolbox, escribir en la consola de Matlab un comando de ayuda con algunas de las funciones como argumento, como, por ejemplo: "help OnFwd".

```
>> help OnFwd
 OnFwd proporciona una velocidad al motor indicado
    OnFwd(motor, velocity) activa el motor "motor" a una velocidad
    "velocity" indicada entre -100 y 100 hacia adelante (siendo el porcentaje de velocidad a
    partir de la velocidad máxima). Las constantes de motores que se pueden
    usar son las siguientes:

    * OUT_A: puerto A de actuadores (motor A)
    * OUT_C: puerto C de actuadores (motor C)
    * OUT_AC: puertos A y C de actuadores (ambos motores)

    See also OnRev, Off, MotorRotationCount, ResetRotationCount

fx >>
```

Listado de funciones disponibles en el toolbox

En esta sección se listarán, por categorías, todas las funciones que la toolbox posee para el control del robot; estas son:

- ✓ Funciones de actuadores:
  - Off
  - OnFwd
  - OnRev
- ✓ Funciones de inicio de sensores:
  - SetSensorHtGyro
  - SetSensorLight
  - SetSensorTouch
  - SetSensorUltrasonic
- ✓ Funciones de sensores (entre paréntesis la función de inicio de sensor que se tiene que utilizar previamente para definir el puerto de entrada que se va a utilizar para este):
  - MotorRotationCount
  - SENSOR_1 (SetSensorLight o SetSensorTouch)
  - SENSOR_2 (SetSensorLight o SetSensorTouch)
  - SENSOR_3 (SetSensorLight o SetSensorTouch)
  - SENSOR_4 (SetSensorLight o SetSensorTouch)
  - Sensor (SetSensorLight o SetSensorTouch)
  - SensorHtGyro (SetSensorHtGyro)
  - SensorUS (SetSensorUltrasonic)
  - ResetRotationCount
- ✓ Funciones de la interfaz gráfica:
  - ButtonPressed
  - ClearScreen
  - NumOut
  - TextOut
- ✓ Funciones de manejo de ficheros:
  - CloseFile
  - CreateFile
  - DeleteFile
  - OpenFileRead
  - ReadLnString
  - WriteLnString
- ✓ Otras funciones:
  - CurrentTick
  - FreeMremory
  - Stop
  - Wait
- ✓ Funciones que no pertenecen al repertorio de funciones de NXC (entre paréntesis una breve descripción de esta):
  - ResetAngle (reseteo del modo de conteo de ángulos en el sensor giroscópico)
  - SensorAngle (devuelve el ángulo del modo conteo de ángulos en el sensor giroscópico)
  - SensorColor (devuelve el código de color leído por el sensor de color)

- StatusLight (apaga o enciende las luces de estado de la interfaz con el color especificado y si estas parpadean)

✓ Funciones privadas:
- privateSensor
- isFileNXC

Función EjecutarCodigoNXC

Para poder usar toda esta toolbox sin necesidad de preocuparse por la inicialización de las constantes y la conexión con el simulador, se ha creado una función que oculta este proceso y, además, permite el uso de logs para facilitar la labor de depuración. También se encarga de manejar las excepciones para que, en caso de error, la conexión con Coppeliasim no se quede abierta y el espacio de trabajo quede limpio. Para poder usar esta función tenemos las siguientes llamadas:

- ✓ "ejecutarCodigoNXC(String script)". Ejecuta el programa de NXC con el nombre "script". También, se permite la escritura como comando de Matlab: "ejecutarCodigoEV3 script".
- ✓ "ejecutarCodigoNXC(String script, Bool log)". Ejecuta el programa de NXC con el nombre "script" y, si "log" es true o 1, guarda en un fichero de texto los mensajes escritos en la consola de Matlab con el nombre "script".
- ✓ "ejecutarCodigoNXC(String script, Bool log, String fichero)". Igual que el anterior, solo que, el archivo de log se guarda con el nombre "fichero".

Para probar que todo el proceso de instalación ha sido satisfactorio se debe en primer lugar proceder a iniciar Coppeliasim, y abrir la escena "escenaTest.ttt". Luego desde Matlab, ir a la Carpeta donde se han extraído los ficheros descargados de poliformaT, en la subcarpeta Test, ejecutar el script "tests.m".

Trabajo a Realizar

El trabajo a realizar incluye varios pasos, desde la creación de funciones para linealizar datos de sensores hasta lograr implementar satisfactoriamente la toma de datos con el entorno Coppeliasim y Matlab para la posterior calibración de sensores del Lego EV3.

Linealización de dados de Calibración

Ejercicio 1. Funciones de linealización

a) Realiza una función de Matlab que proporcione los parámetros m y b de la recta para el método de linealización por puntos extremos. Dicha función deberá aceptar como parámetros de entrada los vectores X e Y que contendrán las entradas y las salidas del sensor.

b) Realiza una nueva función de Matlab que nos proporcione de nuevo los parámetros m y b de la recta para el método de linealización de mínimos cuadrados.

c) Obtén los parámetros de la recta de calibración por el método del punto final y mínimos cuadradas para el siguiente conjunto de puntos:

X=[0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0]

Y=[12.0 9.5 7.0 5.2 4.0 3.1 2.7 2.0 1.5 1.0 0.5 0.2 0.1]

Incluye en las funciones creadas en los ejercicios 1 y 2 la representación en la misma gráfica de la curva de calibración X-Y, así como la recta de calibración obtenida.

Obtención de datos para Calibración de la intensidad del sensor

Utilizando la `escenaTest_medida_distancia.ttt` anterior vamos a implementar un algoritmo en Matlab, para que el robot inspeccione la escena y obtenga medidas de los valores de intensidad medida correspondientes a los distintos valores de grises. Los datos reales correspondientes los tomaremos de forma cualitativa, dividiendo un intervalo (blanco-negro) entre la cantidad de cuadros de distintos tonos de grises.

Código de script lectura_sensor_ejemplo.m

```
%Declaracion global de variables
global intensity_v;
global tiempo_v;
%Inicialización de variables
intensity_v = 0;
tiempo_v = 0;
tiempo = 0;
% Informamos que el sensor de color está en el puerto 1
SetSensorLight(IN_1);
% Definimos la velocidad de avance nominal
v = 15;

% Mostramos por pantalla
TextOut(0,LCD_LINE1,'Seguidor de lineas');
TextOut(0,LCD_LINE2,'Presione el boton central para');
TextOut(0,LCD_LINE3,'comenzar');
% Esperamos a pulsar el botón central
while(~ButtonPressed(BTNCENTER))end
% Borramos la pantalla del robot
```

```
ClearScreen();

% Avanzamos con velocidad v en las dos ruedas (OUT_AC)
OnFwd(OUT_AC, v);
% para dar velocidad individualmente a cada rueda utilizar
% OnFwd(OUT_A, vl); rueda izquierda
% OnFwd(OUT_C, vr); rueda derecha


%Mientras no se presione el botón hacia abajo
while(~ButtonPressed(BTNEXIT))
    % Leemos el sensor
    intensity = Sensor(IN_1);
    % Incrementamos el tiempo (período de muestreo 50ms)
    tiempo = tiempo + 0.05;
    intensity_v = [intensity_v intensity];
    tiempo_v = [tiempo_v tiempo];
    cadena = ['intensidad = ' num2str(intensity)];
    TextOut(0,LCD_LINE1,cadena);
end
% paramos los motores
Off(OUT_AC);
% paramos la comunicación con Coppeliasim
Stop(1);
```

Para ejecutar este script, es necesario hacerlo de la siguiente manera (utilizando el script Ejecuta_lectura_Sensor_ejemplo.m):

```
%Declaracion global de variables
global intensity_v;
global tiempo_v;
%Inicialización de variables
intensity_v = 0;
tiempo_v = 0;

ejecutarCodigoNXC lectura_sensor_ejemplo

plot(tiempo_v,intensity_v)
```

Ejercicio 2. Calibración de los valores de intensidad del sensor.

La tarea a realizar consiste en ejecutar el script "Ejecuta_lectura_Sensor_ejemplo.m". Luego de la ejecución obtener los valores correspondientes a blanco, negro y grises, que serán necesarios en la calibración del sensor.

NOTA: Los datos reales correspondientes los tomaremos de forma cualitativa, dividiendo un intervalo (blanco-negro) entre la cantidad de cuadros de distintos tonos de grises.

Una vez obtenido el registro y la curva de calibración se pide

a) Analiza los parámetros característicos del sensor: Rango de medida, sensibilidad, banda muerta y resolución.
b) Obtén la linealización de la respuesta del sensor por los métodos de puntos extremos y por mínimos cuadrados.
c) Calcula los errores de linealidad, repetibilidad e histéresis cuando se utilizan cada una de las funciones lineales calculadas en el apartado anterior con respecto a los valores reales.

Obtención de datos para Calibración del sensor sonar

Utilizando la misma escena `escenaTest_medida_distancia.ttt` anterior vamos a implementar un algoritmo en Matlab, para que el robot inspeccione la escena y obtenga medidas de los valores de distancia medida.

Ejercicio 3. Calibración de los valores del sensor de distancia (Sonar).

La tarea a realizar consiste en ejecutar el script "Ejecuta_lectura_Sensor_ejemplo.m".

Utilizando los scripts Obtener_datos_sonar.mlx y testSonar_distancia.m obtener los datos de la curva de calibración del sonar.

Una vez obtenido el registro y la curva de calibración se pide

d) Analiza los parámetros característicos del sensor: Rango de medida, sensibilidad, banda muerta y resolución.
e) Obtén la linealización de la respuesta del sensor por los métodos de puntos extremos y por mínimos cuadrados.
f) Calcula los errores de linealidad, repetibilidad e histéresis cuando se utilizan cada una de las funciones lineales calculadas en el apartado anterior con respecto a los valores reales.

Lista con todos los videos útiles de Coppeliasim (Leopoldo Armesto, DISA, UPV)

https://www.youtube.com/playlist?list=PLjzuoBhdtaXOYfcZOPS98uDTf4aAoDSRR

Entregable (entrega hasta el 19 de Marzo)

Subir a poliformaT las funciones, scripts (ficheros de Matlab *.m) todo bien comentado. Escribir un pequeño reporte con los resultados de la calibración.