Modificación de librería de control para mejorar interacción de robot con el entorno

Control Library Modification To Improve Robot Interaction With Environment

José Antonio Ruiz-Jara[1]

Bajo la tutela de: Martín Naya Varela[2], Francisco Bellas Bouz[3] y Esteban Arias Méndez[4],

Ruiz-Jara, J. Modificación de librería de control para mejorar interacción de robot con el entorno. *Tecnología en Marcha*. Vol. 33, especial Movilidad estudiantil. Pág 215-226.

https://doi.org/10.18845/tm.v33i7.5495

1 Estudiante de Ingeniería en Computación. Instituto Tecnológico de Costa Rica, Cartago, Costa Rica. Correo electrónico: jruizj6@gmail.com.
2 Investigador en el Grupo Integrado de Ingeniería de la Universidade da Coruña. España. Correo electrónico: martin.naya@udc.es
3 Profesor titular en la Universidade de Coruña. España. Correo electrónico: francisco.bellas@udc.es
4 Profesor titular en el Instituto Tecnológico de Costa Rica. Costa Rica. Correo electrónico: esteban.arias@itcr.ac.cr

Palabras Clave

Interacción Humano-Robot; Visión Artificial; Sensorización; Robot Humanoide; Robot Poppy, Pypot.

Resumen

Uno de los objetivos del proyecto Poppy es hacer la plataforma accesible para principiantes y expertos por igual, permitiendo que los robots puedan ser utilizados en múltiples ámbitos, entre ellos investigación y enseñanza.

Durante el trabajo con la librería de control Pypot, encargada de controlar los robots Poppy, se han detectado varias limitaciones existentes que se pretenden subsanar. Una de las más importantes, a nivel del trabajo de investigación planteado, es la poca cantidad de sensores a los que se puede acceder mediante ésta.

Gracias a que tanto el software como el hardware son públicos y abiertos para que los usuarios puedan ajustarlos a sus necesidades, se pretende realizar una serie de modificaciones a la librería de control, para poder utilizar el robot humanoide Poppy en el proyecto de investigación. Uno de los principales requerimientos es que la librería sea capaz de utilizar y comunicarse con los nuevos sensores y actuadores añadidos al robot.

Finalmente, en un periodo de cuatro meses de trabajo, se han podido finalizar exitosamente mejoras al módulo procesado de imagen, comunicación con placas IMU y Arduino, implementación de reconocimiento facial, sensores de proximidad en simulación, uso de motores personalizados en simulación, ejecución simultanea de instrucciones en múltiples robots, despliegue de animaciones mediante la pantalla LCD, y un módulo que permite combinar más de una funcionalidad para poder crear comportamientos.

Keywords

Human-Robot Interaction; Artificial Vision; Sensorization; Humanoid Robot; Poppy Robot; Pypoy.

Abstract

One of the objectives of the Poppy Project is to offer a platform accessible to beginners and experts, allowing robots to be used in multiple fields, including research and teaching.

During the work with Pypot, the control library in charge of controlling the Poppy robots, several limitations have been detected that are needed to be fixed. One of the most important problem related to the research work that is going to be done, is the small number of sensors that it can access.

Thanks to the fact that both software and hardware are open source and the users can adjust them to their needs, a series of modifications are intended to be done on the control library, to use the humanoid robot Poppy in the research project. One of the main requirements is the capacity to use via Pypot the new sensors and actuators added to the robot.

Finally, in a period of four months of work, the library got improvements in the image processing module, communication with IMU and Arduino boards, implementation of facial recognition, use of proximity sensors in simulation, use of custom motors in simulation, simultaneous execution of instructions in multiple robots, display of animations through the LCD screen, and a behavior controller that allows combining more than one functionality to create behaviors.

Introducción

Durante muchos años, los robots humanoides han sido principalmente utilizados para estudiar y simular comportamientos del ser humano [1] [2], a tal punto que existen robots que han sido llevados fuera de los límites del planeta [3]. Siguiendo la línea de investigación de los robots humanoides, dentro del Grupo Integrado de Ingeniería de la Universidade da Coruña, el investigador Martín Naya estudia la influencia de la morfología del robot en el aprendizaje y adaptación a su entorno, a una tarea específica, o en la interacción del robot con el ambiente en el que se encuentra.

El robot seleccionado para realizar la investigación es el robot humanoide Poppy, creado por el equipo Flowers de los laboratorios INRIA en Francia. Las principales características de este robot son el código abierto de su software y el acceso a los usuarios a los esquemas del hardware para que puedan crear su propio robot. El robot Poppy está inspirado en aspectos básicos de la morfología humana, enfocándose principalmente en una estructura que permita simular los movimientos humanos de una forma más natural y fluida [4].

Pypot [5] es la librería de código abierto desarrollada con la finalidad de poder controlar los robots de la familia Poppy y cualquier otro robot que haga uso de motores dynamixel. El robot Poppy dispone además de un modelo simulado al cual la librería se comunica mediante el API remoto del simulador V-REP.

No obstante, a pesar de que la librería Pypot permite trabajar con una versión simulada del robot y con el robot real, durante el trabajo con esta librería de control, se han detectado varias limitaciones existentes que se pretenden subsanar. Una de las más importantes, a nivel del trabajo de investigación que se planea llevar a cabo, es la poca cantidad de sensores a los que se puede acceder mediante ésta, lo cual es entendible debido a que es una librería de uso general. Gracias a que el proyecto Poppy es de código abierto, es posible agregar a la librería aquellos sensores y actuadores que sean necesarios para que el robot pueda interactuar de una mejor manera con el usuario y su entorno.

Tomando como base el problema descrito anteriormente, la solución planteada para solventarlo consiste en la ampliación, flexibilización y mejora de la librería de control Pypot, principalmente en los módulos de sensorización, actuadores y comunicación con el API Remoto del simulador V-REP, con la finalidad de dotarla de nuevas funcionalidades para su aplicación concreta con el robot humanoide Poppy, tanto en el entorno de simulación como en el robot real.

Tecnologías utilizadas

Debido a que el trabajo realizado se enfocaba principalmente en el desarrollo de nuevas funcionalidades para el robot, las tecnologías utilizadas fueron principalmente herramientas de software. La principal tecnología utilizada es el lenguaje de programación Python, el lenguaje interpretado y de alto nivel [6] en el cual está desarrollada la librería Pypot, la cual es la librería encargada de controlar todos los aspectos del robot, desde los motores hasta los sensores y actuadores adicionales. Cabe destacar que se ha utilizado la variante de OpenCV para Python, en las tareas que involucran operaciones con matrices, específicamente en el manejo de imágenes. Adicionalmente se ha utilizado el lenguaje de programación compatible con las placas Arduino, para el desarrollo de un protocolo de comunicación que será descrito más adelante.

Otra herramienta de software utilizada es el simulador V-REP. Este framework de simulación permite la creación de escenas en las cuales contengan modelos de robot y sus respectivos sensores y actuadores, con los cuales se puede interactuar mediante API para clientes remotos desarrollados en algunos lenguajes de programación entre los cuales Python se encuentra incluido [7].

Además de las herramientas de software, se ha utilizado un robot humanoide Poppy equipado con sensores capacitivos, micrófonos, altavoz, pantalla LCD táctil y una placa Arduino. A lo largo del proceso de desarrollo de utilizó una Odroid UX4 y una Raspberry Pi 3 como placa base del robot, en la cual se ejecuta la librería Pypot para controlar al robot.

Sensores y actuadores agregados

Uno de los principales puntos de la librería a mejorar, era la capacidad de poder controlar y utilizar sensores y actuadores externos, distintos a los ya integrados originalmente en la librería. Con la finalidad de dotar al robot con las capacidades básicas necesarias en las tareas de investigación, se tomó la decisión de agregar la capacidad de interactuar con los componentes que serán descritos a continuación.

a. Sensor de vision

Los sensores de visión [8] son sensores simulados, propios del simulador V-REP, que permiten la visualización de aquellos objetos renderizables en la escena, que se encuentren dentro de su campo de visión. El papel de los sensores de visión en la librería es proveer a un robot simulado las mismas funcionalidades que se pueden llevar a cabo con las cámaras de un robot físico, las cuales son utilizadas mediante OpenCV.

Si bien las cámaras físicas y los sensores de visión tienen métodos distintos para obtener una imagen, a nivel de código se aprovechan las bondades de la abstracción para que el usuario final sea capaz de utilizar ambos elementos indistintamente (ver figura 1). Esta implementación permite que sea más sencillo agregar un nuevo tipo de sensor de visión o cámara que pueda utilizar las funcionalidades existentes.

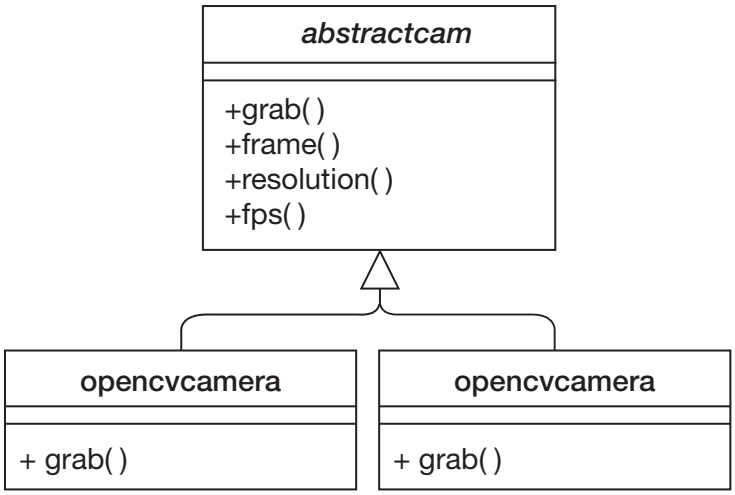

Figura 1. Implementación de clase abstracta para cámaras.

Las cámaras en general ya sean físicas (mediante OpenCV) o simuladas (sensor de visión), pueden ser declaradas en el archivo de configuración, el cual se detallará más adelante, indicando el tipo de cámara (física o simulada), cantidad de imágenes por segundo, resolución y el índice de la cámara (en caso de que exista más de una cámara física). Una vez declaradas en dicho archivo de configuración, podrán ser accesadas directamente desde la clase robot que se haya inicializado.

b. Sensor de proximidad

De igual manera que los sensores de visión, los sensores de proximidad [9] son sensores simulados que pueden detectar objetos en una escena, siendo utilizados como sensores ultrasónicos o infrarrojos entre otros. Este tipo de sensores solo está disponible para ser utilizado dentro de un robot simulado. Los sensores de proximidad permiten identificar si existen objetos dentro del rango de detección, la ubicación del objeto con respecto al centro del sensor, el *handler* del objeto detectado y el tamaño de la superficie detectada. Estos sensores deben ser declarados en el archivo de configuración para su posterior utilización mediante el objeto robot inicializado.

c. Motores simulados

Aprovechando que la funcionalidad existente para el control de motores es capaz de utilizar los motores físicos predeterminados del robot y los motores virtuales predefinidos en el esquema del robot simulado, se ha realizado una pequeña modificación que permitiera utilizar motores personalizados agregados a una escena en el simulador. Gracias a esto, es posible controlar objetos *joints* como si fuese un motor más dentro del esquema del robot y que no necesariamente tengan el mismo patrón de movimiento que los predefinidos. Ver ejemplo en figura 2.

```
"l_shoulder_y": {
  "offset": 90.0,
  "type": "MX-28",
  "id": 41,
  "angle_limit": [
    -180.0,
    150.0
  ],
  "orientation": "direct"
},
"l_arm_z_extension": {
  "offset": 0.0,
  "type": "simulated",
  "id": 98,
  "angle_limit": [
    0.0,
    1.0
  ],
  "orientation": "direct"
},
```

Figura 2. Ejemplo de definición de motor simulado en archivo de configuración.

Los nuevos motores deben ser especificados en el archivo de configuración, con sus respectivos parámetros personalizados.

d. Unidad de medición inercial

Una Unidad de Medición Inercial, IMU según sus siglas en inglés, es un dispositivo electrónico que permite obtener mediciones de aceleración, orientación, magnetismo entre otros. Es utilizado principalmente en tareas de estabilización y navegación [10]. Se ha dotado a la librería con la capacidad de utilizar una *9DOF Razor IMU* mediante una clase que emplea comunicación USB Serial, para poder obtener datos de acelerómetro, giroscopio y magnetómetro desde la IMU. Desde el archivo de configuración se puede definir la frecuencia de comunicación y el puerto a utilizar.

e. Placa arduino

El uso de una placa Arduino viene a reducir la carga de trabajo que tendría la placa base del robot, ya sea una placa Odroid o una Raspberry Pi, al ser la encargada de las tareas que involucren el manejo de actuadores o sensores analógicos. Para la actual etapa de desarrollo se implementaron los respectivos métodos que permitan enviar señales altavoces y obtener datos desde sensores capacitivos, micrófonos. A su vez, se han definido las bases para enviar mensajes que muevan los motores y obtener la posición actual de estos. Los motores que sean agregados a una placa Arduino serán completamente independientes a los que se encuentran en el robot directamente, por lo cual tampoco deberán ser declarados en el archivo de configuración.

Una vez cubierta la comunicación interna entre la Arduino y sus respectivos dispositivos, se implementó un protocolo de comunicación, para poder enviar mensajes bidireccionales entre la placa base del robot hasta la placa Arduino. Dicho protocolo hace uso de la comunicación mediante USB Serial.

En este protocolo de comunicación, la placa Arduino es la encargada de enviar mensajes de estado, los cuales contengan la información actual de los micrófonos, sensores capacitivos y posición de los motores. La frecuencia en la que se envía el mensaje de estado es parametrizable y que puede ser modificada por el usuario. La placa base del robot, es la encargada de recibir los mensajes de estado y almacenar los valores de los datos recibidos para su posterior uso. Además de recibir los mensajes, la placa base puede enviar mensajes para configurar la frecuencia de mensajes de estado, reproducir un sonido y mover motores. Ambas placas envían un mensaje ACK luego de hacer recibido exitosamente un mensaje.

Cuadro 1. Formato de mensaje entre placa base y placa Arduino

Cabecera. (1 byte)	Tipo mensaje (1 byte)	Tamaño de datos (1 byte)	Checksum cabecera (1 byte)	Datos. (Variable)	Checksum datos. (1 byte)

En el cuadro 1 se puede observar la estructura definida para los mensajes en el protocolo de comunicación. La cabecera del mensaje indica que en ese punto inicia la secuencia de datos que ha sido enviada o recibida.

El tipo de mensaje hace referencia al tipo de mensaje que desea transmitir. Cada mensaje tendrá un código asignado que permitirá identificarlo. El checksum de la cabecera servirá para verificar que se ha recibido una cabecera de mensaje válida.

El tamaño de datos indica cuantos bytes deberán ser leídos en el sector de datos, antes de leer el checksum de estos. El tamaño de los datos variará en función al mensaje enviado.

```
def loop():
    while recibir_mensajes:
        byte_leido = obtener_byte()
        if byte_leido is not None:
            verificar_inicio_cabecera(byte_leido)
            buffer_cabecera.append(byte_leido)
            if buffer_cabecera_lleno():
                if (es_valido_checksum_cabecera()
                        and
                        es_valido_tipo_message()):
                    leer_y_validar_datos_mensaje()
                    procesar_mensaje()
                reiniciar_buffer()
```

Figura 3. Pseudocódigo de método de lectura de mensajes.

En la figura 3 se puede observar un ejemplo del método de recepción de mensajes, el cual se encarga de verificar que ha recibido un mensaje válido verificando en primera instancia la cabecera del mensaje, y posteriormente recibiendo los datos del mensaje y procesándolo si los datos son válidos.

Funcionalidades implementadas

a. Acceso a funciones del API Remoto del Simulador V-REP

La librería Pypot hace uso de distintas funciones y métodos del API remoto del simulador V-REP para poder controlar la escena y objetos dentro de una simulación, sin embargo, muchas de esas funcionalidades no pueden ser accedidas directamente por el usuario, por lo que es necesario importar tanto el API de V-REP como Pypot en los scripts de Python.

Dicha limitación fue considerada como uno de los puntos a mejorar; acceder a todo desde un mismo punto, en este caso Pypot.

Entre las funcionalidades que fueron añadidas para un acceso directo desde Pypot se puede encontrar:

- Controles de escena: iniciar, pausar, detener, reiniciar.
- Cambiar simulación a modo asíncrono
- Cambiar simulación a modo síncrono. Este modo bloquea la escena hasta recibir la señal para continuar con el siguiente paso en la simulación.
- Enviar señal para ejecutar el siguiente paso en una simulación en modo síncrono.
- Obtener datos desde sensores de visión y sensores de proximidad

b. Inicializar sensores desde archivo de configuración

La librería Pypot cuenta con la funcionalidad de cargar la configuración de un robot mediante un archivo JSON, el cual indique los diferentes controladores, sensores, grupos de motores y motores que serán utilizados. A dicha funcionalidad se le han realizado los cambios para que sea compatible con robots simulados, pues originalmente la librería solo permitía especificar un archivo de configuración a la hora de inicializar un robot físico. Los nuevos sensores agregados son compatibles con esta funcionalidad, por lo que pueden ser declarados directamente en el archivo de configuración para su posterior uso en el robot (ver figura 4).

```
"sensors":{
  "head_camera": {
    "type": "VrepVisionSensor",
    "fps": 25
  },
  "frontal_proximity_sensor": {
    "type": "ProximitySensor"
  }
},
```

```
"sensors":{
  "imu_sensor": {
    "type": "IMU"
  },
  "top_camera": {
    "type": "OpenCVCamera",
    "index": 0,
    "fps": 20.0,
    "resolution": [
      320,
      240
    ]
  }
}
```

Figura 4. Ejemplo de sensores en archivo de configuración.

c. Filtado y detección de objetos mediante de objetos mediante la cámara

La segmentación de una imagen es la etapa que, en los sistemas de visión artificial, se realiza luego de la adquisición y preprocesamiento de la imagen [11]. Entre las modificaciones añadidas a la librería, existen algunas funcionalidades propias de un sistema de visión artificial, como por ejemplo un módulo de segmentación de imágenes, que incluye el filtrado de color en una imagen obtenida desde la cámara. El filtrado de color elimina aquellos segmentos que se encuentren fuera de un rango de colores parametrizable, con tal de eliminar información irrelevante para el usuario (ver figura 5).

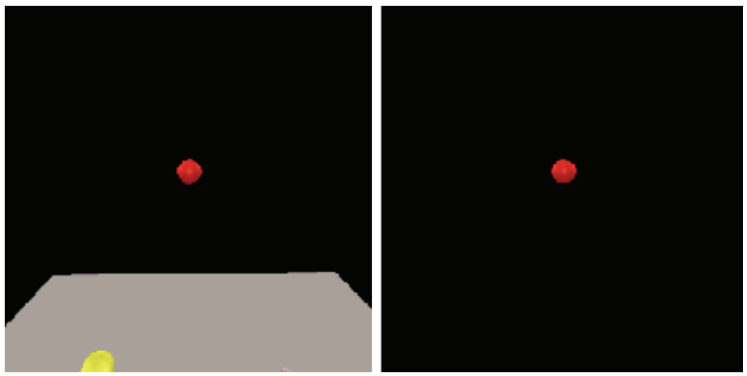

Figura 5. Resultado de filtro aplicado, dentro del rango de color rojo.

Otra de las funcionalidades agregadas es la detección de objetos y la obtención de perímetro, centro y área de dichos objetos. Si bien se puede utilizar en una imagen recién obtenida, se pueden obtener mejores resultados si se utiliza en imágenes que hayan pasado por un filtro de color (ver figura 6). En la implementación del algoritmo, se ha decidido ignorar los objetos que tengan área igual a cero, pues son considerados como ruido en el tipo de imágenes que se van a procesar.

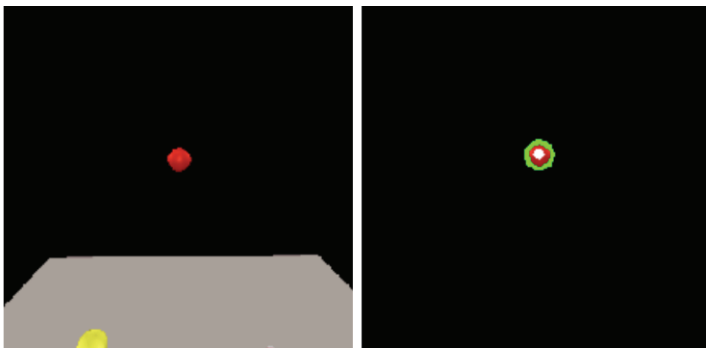

Figura 6. Comparación entre imagen original e imagen obtenida luego de utilizar la función de detectar objetos.

d. Reconocimiento facial

Se ha agregado un módulo de reconocimiento facial, el cual utiliza una librería externa que mediante el uso de *deep learning* permite reconocimiento facial [12].

El flujo de ejecución de la implementación desarrollada involucra encontrar rostros en la imagen obtenida desde la cámara (utilizando un clasificador *Haar Cascade* de OpenCV entrenado

para detectar rostros), para luego recortarlo y analizarlo con la librería externa para realizar el reconocimiento.

Debido a que las placas utilizadas no tienen un gran poder de procesamiento, se ha decidido realizar algunas optimizaciones a la hora de analizar la imagen:

- Se reduce el tamaño de la imagen, en base a un factor de reducción parametrizable (por defecto se reduce a ¼ del tamaño original).
- Se recorta la imagen para obtener una nueva imagen la cual solo contenga el rostro detectado, y finalmente se analiza para determinar si es un rostro conocido o no.
- Dado que el video que se obtiene desde la cámara en realidad son un conjunto de imágenes que reciben una determinada cantidad de veces por segundo, el proceso de detección y reconocimiento se realiza a 1 de cada 2 imágenes obtenidas.

Existen dos formas de entrenar la red encargada del reconocimiento facial: mediante rostros encontrados en la cámara o desde un directorio de imágenes indicado por el usuario. Como resultado del entrenamiento se genera un archivo, que podrá ser utilizado para inicializar la red de reconocimiento facial.

e. Visualización de animaciones

Mediante el desarrollo de la clase *ImageDisplay* se permite el manejo de archivos en formato AVI y GIF, y se dota a la librería con la capacidad de mostrar animaciones en la pantalla. El flujo de trabajo de esta clase consiste en crear una ventana a pantalla completa, utilizando la funcionalidad que provee OpenCV, la cual se actualiza cada determinado tiempo con las distintas secuencias de imágenes cargadas desde los diferentes formatos compatibles. Debido a la necesidad de un medio para mostrar las animaciones, esta funcionalidad es compatible únicamente con la versión física del robot siempre y cuando se cuente con una pantalla para poder observar su correcto funcionamiento.

Para inicializar un objeto de la clase *ImageDisplay*, es necesario indicar el nombre y la ruta de las animaciones que se desean mostrar y el nombre de la animación que se mostrará al iniciar la clase (ver figura 7).

```
animations = {'idle': '/home/poppy/videoPlayback/idle.gif',
              'face_detected': '/home/poppy/videoPlayback/found.gif'}
ImageDisplay(animations, 'idle')
```

Figura 7. Ejemplo de uso de la clase ImageDisplay.

Dichas animaciones son transformadas en un array de imágenes, el cual será utilizado para actualizar la ventana. La clase tiene la capacidad de cargar más de una animación a la vez, por lo que es posible cambiar la animación que se está mostrando por cualquiera de las otras que hayan sido especificadas a la hora de la inicialización del objeto.

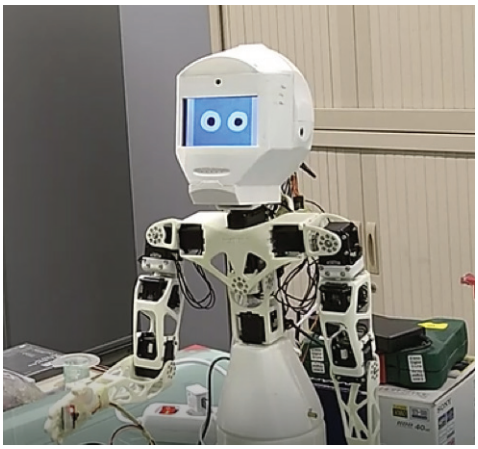

Figura 8. Robot mostrando animación en la pantalla LCD.

f. Imitador

Se ha desarrollado una clase que permite ejecutar una misma instrucción en más de un robot a la vez. La clase es capaz controlar robots locales (simulador y/o robot real en la máquina que ejecuta el programa, como se muestra en la figura 8) y robots remotos (utilizando el REST API del a clase HTTPRobotServer), a los cuales se puede acceder mediante su respectiva dirección IP y el puerto habilitado para el REST API. Esta funcionalidad puede ser utilizada para observar que tan distinto se comportan distintos robots a un mismo conjunto de acciones indicadas por el usuario.

Resultados obtenidos

Si bien la mayoría de las funcionalidades añadidas a la librería son independientes unas de otras, es importante que puedan ser utilizadas en conjunto para poder aprovechar de una mejor manera sus capacidades. Con el fin de demostrar el resultado del trabajo realizado, se ha creado una clase que utiliza dos o más funcionalidades, con el fin de crear comportamientos para el robot. Entre los comportamientos creados podemos encontrar:

- Seguimiento de objetos de un color determinado: El robot es capaz de seguir con su cabeza un objeto de un determinado color, haciendo uso de la cámara para filtrar el objeto y obtener su centro, el cual luego es utilizado para calcular su distancia con respecto al centro de la imagen obtenida por la cámara, para así tratar de centrar el objeto en la toma y lograr el efecto de seguimiento con la cabeza.
- Saludar a conocidos: Se inicializa la funcionalidad de detección de rostros y se muestra una animación en la pantalla que indica que se está buscando un rostro conocido. En el momento que se reconoce un rostro, cambia la animación de la pantalla, el robot estira el brazo para saludar y se emite un audio en el que se saluda a la persona reconocida. El audio, que se emite mediante el altavoz, es previamente cargado a la memoria que utiliza el componente de sonido conectado a la placa Arduino.

Conclusiones

Como se ha evidenciado en los párrafos anteriores, se han desarrollado diferentes funcionalidades que añaden nuevas capacidades a la librería y potencian las ya existentes, con el principal objetivo de mejorar la interacción del robot tanto con el usuario como con el ambiente.

Se han implementado exitosamente mejoras al módulo de sensores de visión o cámara, comunicación USB SERIAL con placas IMU y Arduino, implementación de reconocimiento facial, sensores de proximidad en simulación, uso de motores personalizados en simulación, ejecución simultanea de instrucciones en múltiples robots, despliegue de animaciones mediante la pantalla LCD, y un módulo que permite combinar mas de una funcionalidad para poder crear comportamientos.

Durante el trabajo realizado, se detectó que la fluidez del sistema en de la placa Odroid UX4 y la Raspberrry Pi 3 se veía reducida a la hora de ejecutar las tareas que involucran detección y reconocimiento facial.

Es importante destacar que los cambios realizados a la librería no afectan a los programas que hacen uso de versiones anteriores de la librería, pues fueron agregados de manera tal que no afectara el comportamiento de los módulos ya existentes de manera negativa.

Agradecimientos

Se agradece encarecidamente a los miembros del Grupo Integrado de Ingeniería (GII) de la Univesidade da Coruña (UDC) y a los miembros del Grupo PARMA del Instituto Tecnológico de Costa Rica, por la oportunidad, el trato y la ayuda brindada antes, durante y después de la realización del proyecto.

De igual manera, se agradece a los funcionarios que conforman el Programa de Pasantías para la Movilidad Estudiantil Internacional, Escuela de Computación, y Dirección De Cooperación Internacional del Instituto Tecnológico de Costa Rica.

Referencias

[1] Atkeson, C. G., Hale, J. G., Pollick, F. E., Riley, M., Kotosaka, S., Schaul, S., & Kawato, E. (2000). Using humanoid robots to study human behavior. IEEE Intelligent Systems and their applications, 15(4), 46-56.

[2] Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Kajita, S., Yokoi, K.,. & Isozumi, T. (2003, September). The first humanoid robot that has the same size as a human and that can lie down and get up. In Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on (Vol. 2, pp. 1633-1639). IEEE.

[3] Diftler, M. A., Mehling, J. S., Abdallah, M. E., Radford, N. A., Bridgwater, L. B., Sanders, A. M., … & Hargrave, B. K. (2011, May). Robonaut 2-the first humanoid robot in space. In Robotics and Automation (ICRA), 2011 IEEE International Conference on (pp. 2178-2183). IEEE

[4] Lapeyre, M., Rouanet, P., & Oudeyer, P. Y. (2013, March). Poppy: A new bio-inspired humanoid robot platform for biped locomotion and physical human-robot interaction. In Proceedings of the 6th International Symposium on Adaptive Motion in Animals and Machines (AMAM).

[5] poppy-project. (2017) Pypot: A Python lib for Dynamixel motors control. [En línea] Disponible en: https://github.com/poppy-project/pypot

[6] , M. F. (1999). Python: a programming language for software integration and development. *J Mol Graph Model, 17*(1), 57-61.

[7] Rohmer, E., Singh, S. P., & Freese, M. (2013, November). V-REP: A versatile and scalable robot simulation framework. In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on (pp. 1321-1326). IEEE.

[8] Coppelia Robotics. (s.f.) *Vision sensors* [En línea] Disponible en: http://www.coppeliarobotics.com/helpFiles/en/visionSensors.htm

[9] Coppelia Robotics. (s.f.) *Proximity sensors* [En línea] Disponible en: http://www.coppeliarobotics.com/helpFiles/en/proximitySensors.htm

[10] NavEx (2016) *What is an IMU?* [En Linea] Disponible en: https://www.spartonnavex.com/imu/

[11] Cadena Castro, L. M., & Heredia López, J. A. (2018). Sistema inteligente con visión artificial para el reconocimiento de piezas mecánicas en el Robot NAO.

[12] *Ageitgey*. (2018). *face_recognition*. [en línea] Disponible en: https://github.com/ageitgey/face_recognition