Capítulo 3

Infraestructura

En este capítulo se explican los principales ingredientes software en los que nos hemos apoyado para desarrollar el trabajo. Tales como el entorno JdeRobot, el simulador Gazebo (con el cual podemos simular las acciones que realizaría un robot en un mundo determinado), la librería de OpenCV (empleada en todo lo relacionado con el tratamiento de imagen), PyQt (para el desarrollo de la interfaz gráfica) y Python como lenguaje de programación.

Como se argumentó en el capítulo 1 es uno de los ejes de JdeRobot-Academy.

[20]

Los mundos simulados con Gazebo son mundos 3D, que se cargan a partir de ficheros con extensión “.world”. Estos ficheros son fichero XML definidos en el lenguaje Simulation Description Format (SDF). Este lenguaje contiene una descripción completa de todos los elementos que tiene el mundo y los robots, incluyendo:

* Escena: Luz ambiente, propiedades del cielo, sombras, etc.
* Mundo: Representa el mundo como un conjunto de modelos, plugins y propiedades físicas.
* Modelo: Links, joint, objetos de colisión, sensores, etc.
* Físicas: Gravedad, motor físico, paso del tiempo, colisiones, inercias, etc.
* Plugins: Sobre un mundo, modelo o sensor.
* Luz: Los puntos y origen de la luz.

Las etiquetas empleadas en el fichero para representar estos elementos son: Scene, World, Model, Physics, Plugin, y Ligth.

Los modelos de robots que se emplean en la simulación son creados mediante algún programa de modelado 3D (Blender, Sketchup…). Estos robots simulados necesitan ser dotados de inteligencia para lo cual se emplean los plugins. Estos plugins pueden dotar al robot de inteligencia u ofrecer la información de sus sensores a aplicaciones externas y recibir de éstas comandos para los actuadores de los robots.

* 1. Entorno JdeRobot

JdeRobot es una plataforma de software libre para el desarrollo de aplicaciones con robots y visión artificial. Esta plataforma fue creada por el Grupo de Robótica de la Universidad Rey Juan Carlos en 2003 y está licenciada como GPLv3.

Está desarrollado en C y C++, aunque contiene componentes desarrollados en lenguajes como Python y Java. El entorno que ofrece está basado en componentes, los cuales se ejecutan como procesos. Dichos componentes interoperan entre sí a través del middleware de comunicaciones ICE. ICE permite la interoperación entre los componentes incluso estando desarrollados en diferentes lenguajes.

Es capaz de llevar a cabo diferentes tareas en tiempo real de forma sencilla. Cada componente driver está asociado a un dispositivo hardware del robot, e incluye funciones para poder emplearlo. Esto simplifica el acceso a los diferentes componentes hardware, ya que con una simple función se puede acceder a ellos.

Las aplicaciones constan de uno o varios componentes. Los que interactúan directamente con los sensores y actuadores del robot se llaman drivers, que son los encargados de controlar que los robots reciben órdenes a través de interfaces ICE o Ros messages. Otros llevan en su código las funciones perceptivas, procesamiento de señales o la lógica de control e inteligencia del robot. En la siguiente imagen se puede ver un ejemplo de esta comunicación con un AR Drone empleando interfaces ICE. La misma lógica de comportamiento se puede conectar al driver del drone real o al driver del drone simulado. Basta con cambiar la configuración.

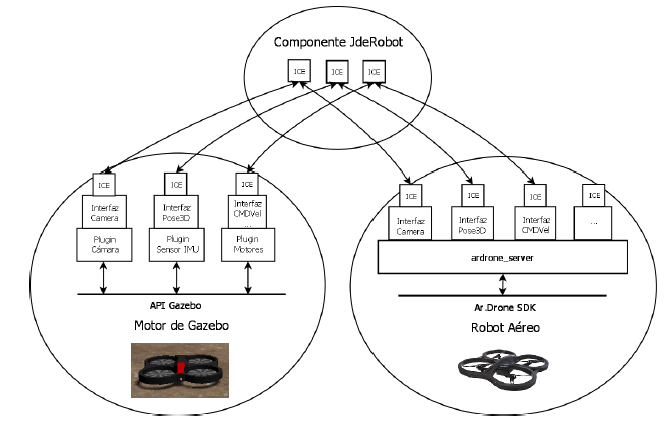


Figure 3.1: Ejemplo de componentes JdeRobot

En el ejemplo de la imagen se puede ver como los componentes de JdeRobot se comunican mediante ICE con los elementos del círculo izquierdo y con los del círculo derecho. En el círculo izquierdo se encuentran los dispositivos del robot simulado en Gazebo, que en este caso son una cámara, un sensor de posición y los motores. En el círculo derecho están los componentes de un robot real. Con esto podemos ver que con JdeRobot se puede teleoperar tanto con un robot simulado como uno real.

Esta plataforma soporta gran variedad de dispositivos como pueden ser el cuadricóptero AR Drone de Parrot, el robot Pioneer de MobileRobotics Inc., el robot Kobuki de Yujin Robot, el humanoide NAO de Aldebaran Robotics, cámaras firewire, USB e IP, los escáneres laser LMS de SICK y URG de Hokuyo, los simuladores Stage y Gazebo, sensores de profundidad como kinect y otros dispositivos X10 de dómotica. A parte de todo esto, tiene soporte para software externo como OpenCV, OpenGL, XForms, GTK, Player y GSL.

En el desarrollo de las prácticas se empleará la versión 5.5.2 de JdeRobot [1], ya que es la última versión estable.

Como se argumentó en el capítulo 1 es uno de los ejes de JdeRobot-Academy.

por compatibilidad con JdeRobot 5.5.2, que a su vez sigue en esa versión de Python para ser compatible con ROS Kinetic. El código en el que están escritos los componentes académicos y las soluciones es Python.

* 1. OpenCV

OpenCV es una librería de código abierto desarrollada por Intel y publicada bajo licencia de BSD. Esta librería implementa gran variedad de herramientas para la interpretación de la imagen. Sus siglas provienen de los términos anglosajones “Open Source Computer Vision Library”, es una librería destinada a aplicaciones de visión por computador en tiempo real.

Esta librería puede ser usada en MAC, Windows, Android y Linux, y existen versiones para C#, Python y Java, a pesar de que originalmente era una librería en C/C++. Además, hay interfaces en desarrollo para Ruby, Matlab y otros lenguajes.

OpenCV es una librería que principalmente implementa algoritmos para las técnicas de calibración, detección de rasgos, para el rastreo, análisis de la forma, análisis del movimiento, reconstrucción 3D, segmentación de objetos y reconocimiento. Los algoritmos se basan en estructuras de datos flexibles acopladas con estructuras IPL (Intel Image Processing Library), aprovechándose de la arquitectura de Intel en la optimización de más de la mitad de las funciones.

Fue diseñado para tener una alta eficiencia computacional. Está escrito en C y puede aprovechar las ventajas de los procesadores multinúcleo. La biblioteca de OpenCV contiene más de 500 funciones que abarcan muchas áreas de la visión artificial. También tiene una librería de aprendizaje automático (MLL, Machine Learning Library) destinada al reconocimiento y agrupación de patrones estadísticos.

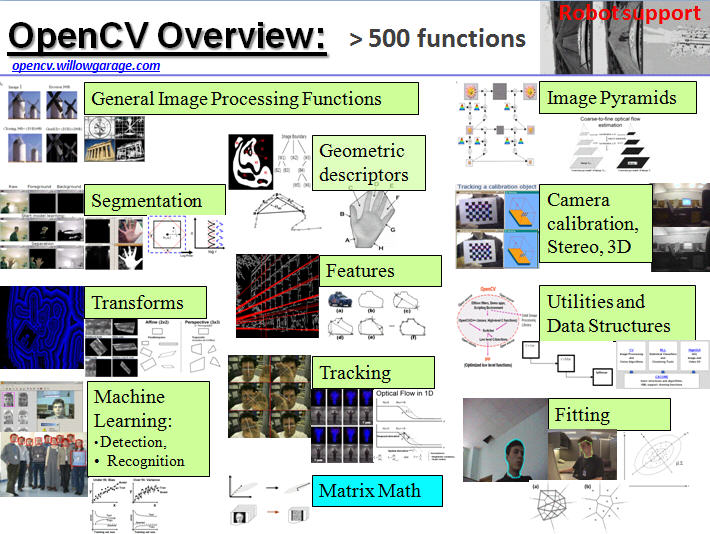


Figura 3.3: Funciones de OpenCV

OpenCV está compuesto por numerosas librerías con las cuales podemos manejar estructuras de datos, detectar bordes y esquinas, escalar o rotar imágenes, modificar el espacio de color de una imagen, realizar matching (comparación de formas para poder saber el parecido entre dos imágenes), detectar líneas y círculos, tratar objetos en 3D, crear ventanas y asociar eventos a dichas ventanas, etc.

Desde su aparición OpenCV ha sido usado en numerosas aplicaciones. Entre las cuales se encuentran la unión de imágenes de satélites y de mapas web, la reducción de ruido en imágenes médicas, los sistemas de detección de movimiento, la calibración de cámaras, el manejo de vehículos no tripulados, el reconocimiento de gestos, etc.

Hay una gran cantidad de empresas y centros de investigación que emplean estas técnicas como IBM, Microsoft, Intel, SONY, Siemens, Google, Stanford, MIT, CMU, Cambridge e INRIA.

En este trabajo se ha empleado la versión 3.2 de OpenCV [69] en Python. Esta librería se empleará para realizar todo lo relacionado con el tratamiento de imágenes. Con ello se extraerán datos que puedan emplearse a la hora de tomar decisiones para que los robots funcionen correctamente.

* 1. PyQt

PyQt es un conjunto de enlaces Python para el conjunto de herramientas Qt, las cuales se emplean para el desarrollo de la interfaz gráfica. Fue desarrollado por Riverbank Computing Ltd y es soportado por Windows, Linux, Mac OS/X, iOS y Android.

Qt es un [entorno](https://es.wikipedia.org/wiki/Framework" \o "Framework) [multiplataforma](https://es.wikipedia.org/wiki/Multiplataforma) [orientado a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos) desarrollado en C++ que permite desarrollar interfaces gráficas e incluye sockets, hilos, Unicode, bases de datos SQL, etc. PyQt combina todas las ventajas de Qt y Python, pues permite emplear todas las funcionalidades ofrecidas por Qt con un lenguaje de programación tan sencillo como Python.

En este proyecto se ha empleado la versión 5 de PyQt. PyQt5 es un conjunto de enlaces Python para Qt5, disponible en Python 2.x y 3.x. Tiene más de 620 clases y 6000 funciones y métodos. PyQt5 dispone de una licencia dual, es decir, los desarrolladores pueden elegir entre una licencia GPL (General Public Licence) o una licencia comercial.

La interfaz gráfica de los componentes académicos creados en las prácticas está escrita usando PyQt. Las clases de PyQt5 se dividen en ciertos módulos, tales como QtCore, QtGui, QtWidgets, QtXml, QtSql, etc. En las prácticas desarrolladas se ha hecho uso de los siguientes módulos:

* QtCore: contiene las funcionalidades principales que no tienen que ver con la GUI. Este módulo se emplea para trabajar con archivos, diferentes tipos de datos, hilos, procesos, url, etc.
* QtGui: contiene clases para el desarrollo de ventanas, gráficos 2D, imágenes y texto.
* QtWidgets: dispone de clases que proporcionan un conjunto de elementos de interfaz de usuario para crear interfaces de usuario clásicas de escritorio.