**DEFENSA DE “Nuevas Prácticas Docentes de Robótica en el Entorno JdeRobot-Academy”**

**DIAP (3)**

A lo largo de la historia el hombre ha tratado siempre de apoyarse en la ciencia para construir y emplear herramientas o máquinas que consiguen reducir su carga de trabajo, propósito en el cual encaja perfectamente la robótica. Es por eso que esta rama de la ingeniería ha sufrido un crecimiento exponencial en los últimos años, valiéndose de la informática, la electrónica y muchas otras disciplinas para desarrollar sistemas que permitan facilitar la vida, no sólo en el aspecto laboral, sino también en la vida cotidiana.

Así, los robots empiezan a sustituir al hombre en tareas básicas, repetitivas o de difícil realización en la industria, las tareas domésticas, la medicina, la logística o el ámbito militar entre otros.

**DIAP (5)**

La tarea de diseñar robots debe ser dividida en distintos módulos con el fin de poder llevar a cabo un proceso de desarrollo y puesta en escena menos costoso y más fiable, ya que no suele ser viable escribir la lógica de control de una máquina en ella directamente, sin un proceso previo de testeo. Por ello, los proyectos de robótica suelen involucrar, al menos, estos 3 componentes:

-Debe haber software, lógica que les dota de autonomía para desempeñar una tarea. Interesa que sea versátil e intercambiable, es decir, que no dependa del entorno y que sea compatible entre distintos sistemas.

-Las plataformas *middleware*, que facilitan el plan de diseño ocultando la complejidad y soportando el software distribuido, para poder llevar a cabo un proceso exhaustivo de pruebas y depuración. (ALGUNOS EJEMPLOS SON…)

-Por último, el empleo de simuladores es vital para ahorrar costes, ya que en caso de haber cometido errores en el proceso previo, pueden descubrirse en ellos antes de introducirlo en un robot. (ALGUNOS EJEMPLOS SON…)

**DIAP (7 y 8)**

(7) Con todo ello, se hace necesario instruir nuevos profesionales con capacidades específicas para comprender y manipular estos componentes. Surgen por ello recientemente entornos de aprendizaje en el campo de la robótica, entre los que está JdeRobot-Academy, que se apoya en el *middleware* JdeRobot, que realiza las tareas propias de dichas plataformas (IMAGEN: Se basa en un componente académico que abstrae al alumno de la infraestructura de las prácticas e incluye la lógica del robot a programar y una interfaz, que conecta con el simulador que contiene modelos y robots simulados que se comunican a través de *plugins*, todo ello accesible desde APIs sencillos. Soluciona todo lo necesario para que el programador sólo se preocupe de escribir el código). Para ello, se vale de Python como lenguaje de programación de la lógica y la infraestructura, del simulador Gazebo (que permite disponer de entornos y modelos 3D), y de un nodo académico específico que contiene todo lo necesario para abordar un problema concreto de la robótica.

(8) Este proyecto se apoya en JdeRobot para crear dos nuevas prácticas que abordan problemas clásicos de robótica, extendiendo así su oferta de aprendizaje. (Follow\_face y laser\_loc).

**DIAP (Práctica 1) (10,11,12,13)**

(10) Ésta primera práctica tiene como objetivo que una cámara real sea capaz de seguir caras de personas. La cámara, sólo proporcionará fotogramas captados (a través de una capturadora de vídeo) al nodo académico, y dispondrá de dos actuadores de movimiento que controlarán su movimiento horizontal y vertical (*Pan* y *Tilt* respectivamente). Con ello, el alumno debe programar un algoritmo de segmentación y la lógica de control reactivo gradual para el seguimiento.

El robot utilizado en esta práctica será la cámara Sony Evi modelo d100p, que requiere de dos *dirvers* que hemos preparado para la comunicación e intercambio de datos entre esta y el nodo:

* ROS usb\_cam: driver del middleware ROS que interactúa con cámaras USB estándar y publica sus imágenes bajo topics de ROS. El nodo dispone del fichero de configuración pertinente para este driver.
* Teleoperador evicam\_driver: driver desarrollado en C++ para controlar el cuello mecánico de la cámara.

(11) El componente académico de esta práctica resuelve varios módulos que la soportan y que sirven de gran ayuda para poder abordarla.

- GUI API: interfaz que ayuda a depurar el código, ofrece las siguientes herramientas:…(EXPLICAR IMAGEN)

-HAL API: Ofrece acceso a la cámara y a sus actuadores en forma de métodos simples (ocultando el middleware de comunicaciones con el hardware).

-Código auxiliar: código que no forma parte del foco del algoritmo pero que ayuda a programar la solución (API)

(12) Además de la infraestructura de la práctica, se ha propuesta una solución de referencia. Dado que el objetivo es seguir caras, la solución debe estar compuesta por un algoritmo de análisis y segmentación de imagen y otro de pilotaje:

- La información de imagen no es útil sin un algoritmo de extracción de los datos útiles. El algoritmo propuesto como solución comienza con la **extracción de características** (por ejemplo el hecho de que la región de los ojos es más oscura que la de la nariz), y se utilizan para entrenar un sistema de clasificación, que en este caso se tratará **de clasificadores en cascada**, con el fin de afinar la clasificación y eliminar falsos positivos. Para ello, nos valemos de la **biblioteca OpenCV**, útil tanto en la fase de entrenamiento y creación de clasificadores como en la posterior segmentación.

- Una vez extraída la posición de la cara utilizamos un control gradual reactivo (datos de sensores que se traducen en órdenes a los motores, que son saltos pequeños de posición) y vamos controlando en cada iteración si se ha alcanzado la posición deseada.

(13) Así, uniendo todos los pasos y añadiendo otros módulos de código para crear un algoritmo más robusto (por ejemplo de gestión de “no detección”) se alcanza la solución.

**DIAP (Práctica 2) (15,16,17,18,19)**

(15) La segunda práctica creada pretende abordar un problema de localización en un entorno y basada en la información que proporciona un sensor láser.

En esta ocasión, la infraestructura cuenta con el nodo académico correspondiente que habilita una interfaz y recoge el código a implementar, de nuevo ocultando la infraestructura, y de una parte de simulación con Gazebo, que estará compuesta por un robot simulado modelo Roomba de iRobot (con el sensor láser y el odométrico) y por un mundo simulado. El nodo solucionará en este caso la comunicación con las interfaces del robot simulado con *plugins* de ROS.

(16) En cuanto al GUI preparado para esta práctica, incluye varios elementos de depuración, visualización y control. En primer lugar, la **gráfica de representación de los datos láser**, que representará la lectura del sensor real y el “láser teórico”, una lectura que el robot obtendría teóricamente si ocupase una posición concreta. Por otro lado, se dispone del **mapa de referencia** del entorno, el cual utilizará el robot para realizar los cálculos pertinentes. En él se representará mucha información como las salidas de cada iteración del algoritmo, la posición real del robot, la trayectoria que ha seguido, o las distintas estimaciones de posición que se hacen a lo largo del tiempo. Por último, hay un **teleoperador** para mover el robot, además de los elementos básicos de inicio y parada de ejecución.

(17) La solución de referencia emplea un **filtro de partículas**, basado en evolucionar un conjunto muestral de posibles posiciones del robot en el entorno, y en el cálculo de probabilidad de ocupar cada posición. En primer lugar, se **inicializa** la primera generación de muestras o partículas de modo aleatorio y se calcula su realizando un **trazado de rayos** sobre el mapa, para aplicar sobre él un **modelo de observación** en función del parecido con la lectura real. Luego se incorpora el **modelo de movimiento** para tener en cuenta la información de odometría y obtener generaciones de menor calidad o descartar datos. A partir de ahí, se hace uso de un **filtro de partículas** para obtener las siguientes generaciones. Se comienza con el **algoritmo de la ruleta,** se divide la ruleta en sectores proporcionales a la probabilidad de cada partícula (siendo el total de la ruleta la probabilidad acumulada), se escoge un número aleatorio en ese rango (equivalente a girar la ruleta) y la partícula cuya porción contenga dicho número genera descendencia. Dicha descendencia se apoyará en técnicas elitistas, ruido térmico y remuestreo en función de la probabilidad para conseguir una evolución inteligente de las generaciones. Por último, es importante añadir **métodos de optimización** como técnicas de programación multihilo, precomputación.

(18) Con todo ello, un ejemplo de la ejecución sería el siguiente: en cada iteración se descartan aquellas partículas que no aportan información y se generan otras nuevas para la exploración de las zonas probables, llegando al final a la convergencia, cuando se puede realizar una estimación.

(19) Otro ejemplo con movimiento involucrado, donde se ve que en las primeras estimaciones se obtiene un ligero error que en seguida se estabiliza con la incorporación de nueva información, haciendo finalmente coincidir las trayectorias real y estimada.

**DIAP (21)**

(21) La última parte del proyecto se basó en realizar versiones de ambas prácticas a través de la herramienta web Jupyter, documentos con texto con recursos multimedia y celdas de código Python entre otros, manteniendo así la funcionalidad de la aplicación original.

Esto junto con modelos de simulación basados en *ROS Messages* amplía en gran medida el número de usuarios potenciales de las prácticas, siendo que ROS está disponible en las plataformas principales (Fedora, Debian, Windows, MacOS,…), y Jupyter en cualquier plataforma con disponibilidad web.

**DIAP (22)**

(22) Leer