Robots Móviles - Entrega 4

# Memoria del Proyecto: Robot Guardian con CNN

Pablo José Cremades Garrido
pablo.cremades@mocitos.es

Escuela Politécnica Superior
Universidad de Alicante

14 de enero de 2024

---

**Resumen**

En el ámbito de la robótica móvil, el mapeado de entornos es una tarea crítica. Este proyecto aborda la utilización de los paquetes de ROS para crear una simulación del Turtlebot capaz de orientarse y navegar a la vez que mapea un entorno desconocido.

---

# Índice

# 1. Introducción

En una práctica anterior se intentó realizar un robot que mapease automáticamente un entorno. Para ello se hizo uso de paquetes como move_base, slam y lite_explore. En este proyecto se termina de perfilar el mapping autónomo y además se le añaden al proyecto herramientas para poder integrar reconocimiento facial tanto al robot como al usuario.

El usuario mencionado en este caso es la persona observando el video feed del robot que para esta entrega se ha habilitado en los archivos launch automáticamente.

# 2. Metodología

La práctica se ha realizado siguiendo estos pasos:

1. Instalación de los paquetes necesarios en ROS para trabajar con TurtleBot3 y el simulador Gazebo.
2. Configuración del entorno de simulación en Gazebo para representar dos escenarios distintos: un edificio y un espacio abierto con obstáculos.
3. Ejecución del algoritmo de SLAM para mapear ambos entornos mientras se realiza la localización simultánea del robot.
4. Utilización del comando `map_saver` para guardar los mapas generados en el formato apropiado.
5. Descarga y utilización del paquete público en ROS de explore_lite
6. Creación y perfilación de una red CNN para reconocimiento facial
7. Diseño de los diferentes métodos para poder guardar las caras de los diferentes usuarios y su implementación.
8. Ensamblaje del proyecto y documentación.

# 3. Set-up mapping

Para este proyecto se unificaron todos los archivos launch para facilitar y aligerar la experiencia del usuario. Por ejemplo para poder realizar el mapeo automático del entorno tan solo hace falta lanzar: auto_mapping_6x6.launch ó auto_mapping_office.launch (volver a lanzar en otra consola roslaunch explore_lite explore.launch"si se quedase parado pero con ruta calculada y todavía hubiese parte del mapa sin explorar. Este error tiene que ver con las comunicaciones internas entre paquetes. Ocurre poco y arreglarlo no me ha sido posible).
Finalmente con el comando: rosrun map_server map_saver -f mi_mapa se guardará el mapa en el proyecto y quedará para su uso en las siguientes partes.

# 4. Set-up CNN

Para utilizar el reconocimiento facial primero el nuevo usuario debe tener una webcam en su portatil y lanzar el script: video_face_record.py. Este script abrirá la cámara, pedirá al sujeto que se centre y empezará a capturar imágenes del mismo así como localizar su cara, recortarla del frame y prepararla para su uso en el reentrenamiento de la red. Por último mover las imágenes de la carpeta cropped_faces

a la carpeta pictures. Cada nuevo usuario deberá crear su propio nuevo directorio dentro de "picturesz reentrenar la red.

Por último verificar que la red funciona correctamente mediante el script: test_CNN.py que abrirá otra ventana y en vivo identificara las caras con un porcentaje de seguridad.

Como el trabajo se hizo por simulación esta red solo pudo ser testada mediante webcam. La carpeta uncut sirve para añadir imágenes para aprendizaje manualmente. Al guardarlas ahí y modificando los directorios en preproccesing_uncuts.py.

# 5. Set-up patrolling waypoints

En este punto se explica cómo configurar una ruta de patrulla. En el launch: roslaunch controller patrol_6x6.launch", tenemos un ejemplo en el que se puede observar cómo unir AMCL cargando el mapa generado en puntos anteriores con el script controller.py para configurar los puntos elegidos.
Mediante RVIZ se coloca el robot con 2D NAV GOALS en posiciones que queremos convertir en ruta de patrulla. Se guardan las coordenadas y se crea un punto en el script controller.py siguiendo la siguiente estructura: "send_goal_to_move_base(XPOS, YPOS, 1.0)".

# 6. Construcción de CNN y miscelánea

## 6.1. Selección de Herramientas

Para el desarrollo del proyecto se seleccionaron herramientas de vanguardia en el campo del aprendizaje automático y procesamiento de imágenes. Para la construcción y entrenamiento de la red neuronal convolucional (CNN) su utilizaron **TensorFlow** y **Keras**, por su flexibilidad y eficiencia. **OpenCV** se utilizó para el procesamiento y manipulación de imágenes.

## 6.2. Diseño Preliminar de la CNN

El diseño inicial de la CNN incluyó capas convolucionales para la extracción de características, seguidas de capas de agrupamiento para reducir la dimensionalidad. Capas densas se añadieron al final para la clasificación. La estructura se diseñó buscando un equilibrio entre precisión y eficiencia computacional.

## 6.3. Recopilación y Preprocesamiento de Datos

Se recopiló un conjunto de datos de imágenes faciales, las cuales fueron preprocesadas para normalizar su tamaño y formato. Las técnicas de preprocesamiento incluyeron normalización de píxeles, reconocimiento facial, recorte de la imagen y redimensionamiento.
Para los usuarios desconocidos se utiliza un paquete de imagenes de personas ficticias realizadas mediante AI. Su conjunto de datos se encuentra en referencias.

### 6.4. Problema de Sobreajuste

Se identificó sobreajuste en las primeras fases del proyecto. Para mitigarlo, se implementaron técnicas como **Dropout**, donde ciertas neuronas se desactivan aleatoriamente durante el entrenamiento, y **Data Augmentation**, para aumentar la diversidad del conjunto de datos.

### 6.5. Data Augmentation

Para poder trabajar con pocas imágenes de usuarios válidos se aplicaron transformaciones como rotación y cambios de brillo para simular diversas condiciones de iluminación y perspectivas. Esto ayudó a mejorar la robustez del modelo pero sigue pecando de sobreajuste y es todavía sensible a al cambio de color en iluminación ambiente (blanca, amarilla y azul) y al uso de accesorios como gafas y auriculares.

### 6.6. Early Stopping

Finalmente, para paliar aún más el sobreajuste y mejorar la generalización del modelo, se integró un *callback* de Early Stopping en el proceso de entrenamiento. Este método detiene el entrenamiento cuando la métrica de validación deja de mejorar, asegurando que el modelo no aprenda el ruido presente en los datos de entrenamiento.

## 7. Desarrollo y Resultados

Durante el desarrollo del proyecto se planteó realizar otras utilidades como comunicaciones con servidores y demás sin embargo la limitación a una simulación acabó desvirtuando esas posibilidades. Al final se quedó en un proyecto adaptable a muchos entornos distintos con herramientas que permiten su configuración de forma rápida y por un profesional cuya finalidad es la de patrullar y grabar una zona. Se queda el reconocimiento de usuarios o visitantes válidos algo apartado del conjunto pero se queda como prueba de concepto y a falta de cambiar el feed de la cámara por la del topic en un robot real. Por desgracia, este tipo de tecnologías de reconocimiento facial a día de hoy empiezan a ser ilegales en España para las empresas debido a la seguridad informática y protección de datos biométricos: https://www.avezalia.es/atencion-empresas-multas-por-uso-de-huella-dactilar-y-reconocimiento-facial-para-el-control-laboral
Se entiende que cierto personal especializado, como seguridad, sí que podría ceder su imagen pero no está especificado y tendremos que esperar a ver qué deciden los tribunales cuando se dé el caso.

Lo que sí se ha logrado es un mapeo automático y autónomo de entornos generales, el cambio de método de localización de SLAM a AMCL y una técnica para definir rutas de patrulla que se realizan con retransmisión en vivo indefinidamente.

## 8. Apéndice

Referencias a tutoriales, documentación y recursos utilizados para la práctica.
https://github.com/jeshoward/turtlebot_rrt/blob/master/docs/MAP_CONVERSION.md
chatgpt v4.0
https://archive.org/details/1mFakeFaces