Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**

Máster Universitario en Inteligencia Artificial

Práctica 2

Desarrollo de un Navegador Mediante Mapas Topológicos Visuales

Autores:

Alejandro Benítez López

Elena Benito Frey

Mario González Carbayo

Isidro López Dominguez

Blanca Martínez Donoso

Ángel Pavón Pérez

Madrid, 27 de noviembre de 2020

Índice

Índice de figuras

Índice de tablas

1. Introducción

En esta práctica se desarrollará un navegador capaz de reconocer su entorno dentro de un escenario cerrado e interior con la ayuda del diseño de un mapa topológico visual. Para ello haremos uso del algoritmo de clasificación K-NN (K-Nearest Neighbors) con el que podremos entrenar nuestro navegador utilizando un dataset de imágenes tomadas de cada uno de los puntos a reconocer (landmarks). De esta forma, el clasificador podrá usarse con nuevos vídeos del entorno, no vistos anteriormente, para detectar en qué posición del mapa se encuentra la cámara. En total nuestro navegador debe ser capaz de reconocer, con la mayor precisión posible, nueve de estos landmarks que se encuentran distribuidos por el mapa topológico de forma que pueda cubrir toda la extensión del escenario. En nuestro caso, el escenario será un entorno virtual de una casa sobre el que podemos movernos con libertad, de manera que diseñaremos un recorrido en el que el navegador irá dando como salida en qué punto del mismo se encuentra. La idea de este proyecto, es que si se diese el caso de que un robot llevase una cámara, este clasificador pudiera ayudarle a localizarse en el mapa.

2. Mapa topológico

En primer lugar, para poder llevar a cabo todo el proyecto, debemos escoger un determinado escenario sobre el que haremos el reconocedor. Este proceso normalmente se lleva a cabo en escenarios reales, como podría ser una casa particular o la planta de algún edificio, sin embargo, debido a la situación actual del COVID-19, hemos decidido tomar un camino alternativo para facilitar el trabajo en grupo en el que hemos utilizado un escenario virtual a través de entornos simulados. Podíamos utilizar escenarios dentro de videojuegos o plataformas de realidad virtual. En nuestro caso hemos decidido utilizar, debido a su sencillez para trabajar sobre él, un simulador de viviendas (HomeByMe) en el que podemos diseñar con total libertad un escenario para crear el mapa topológico. En este simulador podemos añadir o quitar habitaciones, además de poder escoger entre distintos diseños con los que diferenciaremos notablemente los distintos landmarks para allanar el camino del clasificador. El simulador incluye la posibilidad de moverse y observar el escenario desde distintas perspectivas, lo que ayudará tanto al diseño del dataset de imágenes para entrenar el clasificador como para el diseño y grabación del recorrido que idealmente realizará nuestro robot.

Como vemos en la Figura 1, hemos diseñado un piso con una serie de habitaciones y un pasillo para poder tener varios ambientes sobre los que probar el reconocedor.

Figura 1: Casa creada con la herramienta HomeByMe

Tras crear la casa que puede verse en la Figura 1, se han decidido escoger 9 landmarks distintos de forma que pudiéramos cubrir todos las zonas de la casa con la mayor amplitud posible:

- **Landmark 0**: La cocina de la casa. 72 imágenes
- **Landmark 1**: El salón de la casa. 61 imágenes
- **Landmark 2**: La oficina de la casa. 92 imágenes
- **Landmark 3**: La parte de abajo del salón de juegos de la casa. 81 imágenes
- **Landmark 4**: La parte de arriba del salón de juegos de la casa. 60 imágenes
- **Landmark 5**: El baño de la casa. 86 imágenes
- **Landmark 6**: La parte derecha del pasillo principal de la casa. 77 imágenes
- **Landmark 7**: La parte izquierda del pasillo principal de la casa. 62 imágenes
- **Landmark 8**: La habitación de la casa. 64 imágenes

De esta forma, el mapa topológico creado puede verse en la Figura 2. A continuación, para cada landmark se toman una media de 75 fotografías en las que vamos variando ligeramente la perspectiva de cada imagen de manera que podamos recoger una gran parte de los posibles puntos de vista que podría tener el robot al realizar un recorrido a través de ese landmark, así como poder recoger de la manera más óptima posible todos los elementos del landmark para ayudar tanto al reconocedor como a la mejora de nuestro clasificador.

Para tomar estas imágenes hemos diseñado, a través de la librería OpenCV de Python, un pequeño código que nos permite tomar capturas a nuestro gusto sobre un vídeo de cada landmark en el que movemos la cámara por el entorno para poder captar todas las perspectivas posibles.

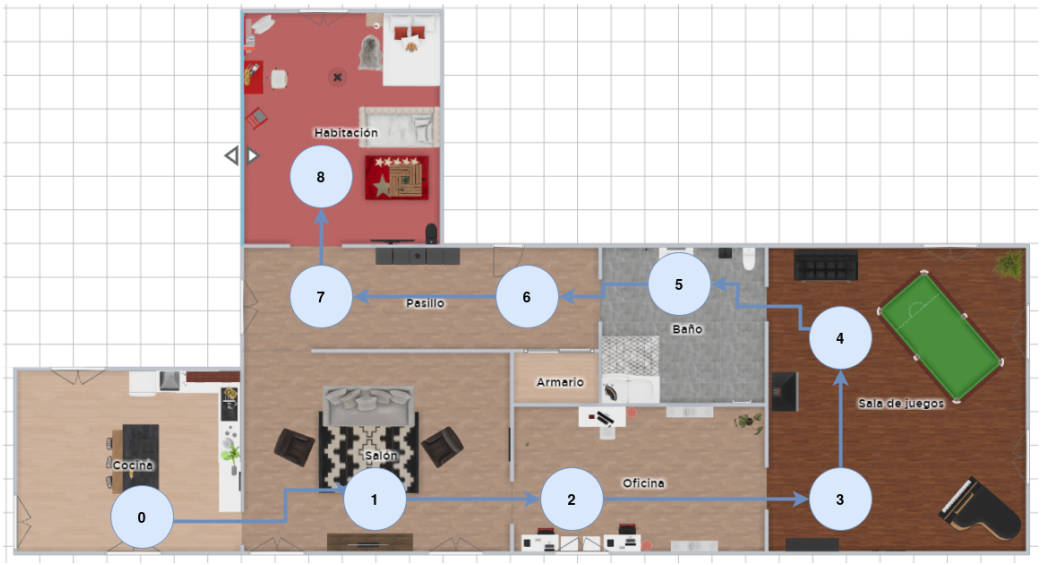

Figura 2: Mapa topológico

3. Implementaciones

Una vez está diseñado nuestro mapa topológico, en el que tenemos dispuestos todos los landmarks del escenario, así como sus respectivos sets de imágenes, podemos comenzar con la implementación del algoritmo de clasificación de imágenes. Para realizar el clasificador, debido a las características del algoritmo K-NN, será necesario obtener los histogramas de cada imagen. En nuestro caso realizaremos dos clasificadores distintos en función del tipo de histograma que recojamos de la imagen: en primer lugar realizaremos un clasificador con el histograma en escala de grises de cada imagen; en segundo lugar haremos otro clasificador en el que los histogramas correspondan a la distribución RGB de cada imagen. Por último, construiremos un reconocedor a través de los histogramas RGB de las imágenes pero utilizaremos una red neuronal para comparar los resultados de ambos métodos.

3.1. K-NN

K-Nearest Neighbors es un algoritmo basado en instancia de tipo supervisado, el cual puede usarse tanto para clasificación de nuevas imágenes como para predicción. A través de una simple distancia euclídea el algoritmo es capaz de encontrar los K elementos más cercanos a un nuevo elemento que queramos clasificar. Una vez obtiene los K elementos vecinos, el algoritmo reconoce cuántos de esos vecinos pertenecen a las distintas clases en las que podemos clasificar nuestro nuevo elemento, y de acuerdo a una distribución de probabilidad dada, asigna al nuevo elemento una clase, para finalmente almacenarlo y aumentar su dataset de elementos y por tanto su precisión. En nuestro caso, utilizaremos un algoritmo 1-NN donde automáticamente etiquetaremos el landmark al que pertenece cada imagen viendo cuál es la imagen vecina más próxima. Para poder medir la distancia entre imágenes, haremos uso del histograma en escala de grises y el histograma en escala RGB que pueden ser tratados como vectores a los que podemos calcular su distancia euclídea con otro vector de su mismo tipo. De esta manera, dada una imagen, ésta pertenecerá al landmark de aquella imagen cuyos vectores tengan la mínima distancia, habiéndose comparado las distancias entre todo el dataset disponible.

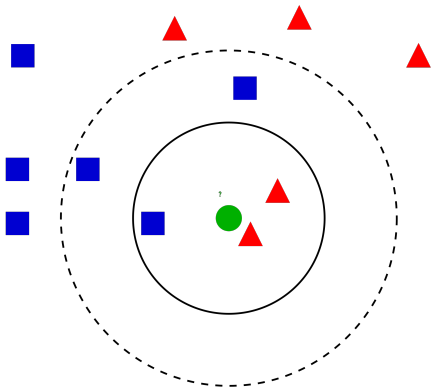

Figura 3: Algoritmo de clasificación

3.1.1. Escala de grises

En este apartado se va a explicar como ha funcionado el algoritmo 1-NN al utilizar las imágenes con escalas de grises y tomando como distancia entre estas imágenes la distancia euclídea de sus histogramas correspondientes.

En la tabla 1 podemos observar los resultados obtenidos en modo estático. Este modo sirve como entrenamiento de la red, en el que para realizarlo, aislamos una imagen de todo

el conjunto, y mediante el algoritmo, comprobamos cual es el landmark al que pertenece verificando posteriormente si el resultado es o no correcto. Posteriormente, realizamos el mismo proceso para cada imagen del dataset (leave-one-out).

Landmarks	**Porcentaje de acierto**
0	100 %
1	100 %
2	98,92 %
3	98,78 %
4	98,36 %
5	98,85 %
6	98,71 %
7	96,82 %
8	100 %
Total	98,94 %

Tabla 1: Resultados estáticos en escala de grises

Como podemos apreciar, hemos obtenido un porcentaje de acierto medio entre todos los landmarks de 98,94 %. Para nuestro clasificador, hemos considerado conveniente utilizar un umbral de distancia para el K-NN mínimo, es decir, tomamos como valor de la imagen de estudio aquel landmark al que pertenezca la imagen con el histograma más cercano, así evitamos la posibilidad de que no clasifique la imagen.

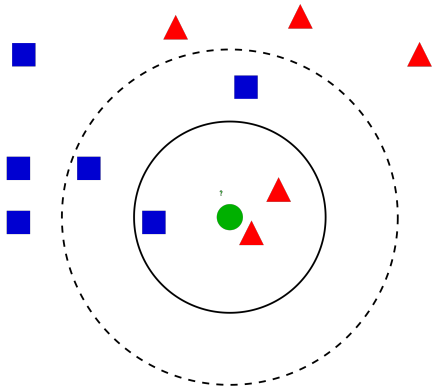

Figura 4: Algoritmo de clasificación

En la tabla 2, mostraremos los resultados obtenidos en imágenes con escalas de grises, pero en esta ocasión en modo dinámico.

Landmarks	Porcentaje de acierto
0	77,20 %
1	90,05 %
2	24,43 %
3	98,80 %
4	80,84 %
5	96,83 %
6	93,10 %
7	38,72 %
8	96,85 %
Total	77,42 %

Tabla 2: Resultados dinámicos en escala de grises

Podemos ver cómo el porcentaje de acierto ha empeorado considerablemente en comparación a los resultados obtenidos en la prueba estática. Esto se debe principalmente, a que el clasificador estático toma para cada landmark imágenes que se toman desde un punto fijo con pequeñas variaciones en su perspectiva facilitando el trabajo del clasificador, puesto que los histogramas de las imágenes de cada landmark presentan una diferencia muy pequeña. En cambio al hacer la prueba dinámica, estamos obteniendo imágenes de cada frame del vídeo del recorrido en posiciones distintas a las imágenes tomadas para el dataset, lo que hace que las distancias euclídeas entre las imágenes de la prueba dinámica y del dataset puedan presentar distancias euclídeas notables como para ser confundidas con menos dificultad con landmarks incorrectos.

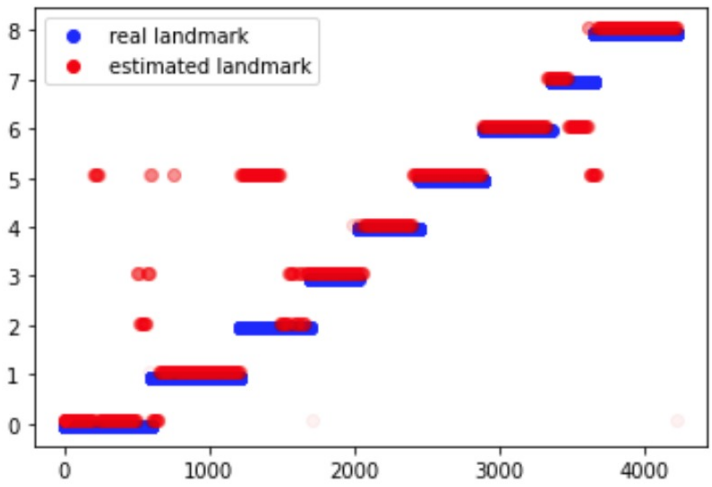

Figura 5: Comparativa de positivos

Como vemos en la figura 5, para cada frame la clasificación es bastante correcta en las zonas centrales del recorrido de cada landmark. Esto sucede debido a que en las zonas de cambio de landmark, no se definen con total precisión los límites de cada uno. En nuestro caso, hemos etiquetado cada frame con su landmark correcto según hemos considerado

de forma visual, pero el reconocedor puede presentar imprecisiones en estos puntos, sin embargo, como estos falsos positivos dependen de nuestro etiquetado de cada frame, no tiene mucho sentido tenerlos en cuenta.

Sumado a eso, tenemos que tener en cuenta que al trabajar en un entorno simulado los matices de sombras son muy pequeños, lo que acentúa el error en la escala de grises, y por tanto, como podemos ver en los histogramas medios de los landmark que más se confunden entre sí aparecen picos muy parecidos. Se muestran a continuación los histogramas de los landmarks que más se confunden entre sí, más en concreto, el de la izquierda se confunde con el de su respectiva derecha (2 con 5 y 7 con 6):

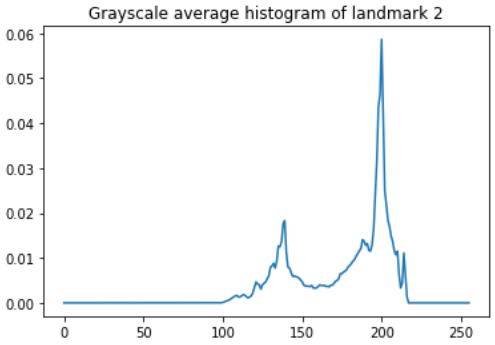

Figura 6: Histograma medio en escala de grises del landmark 2

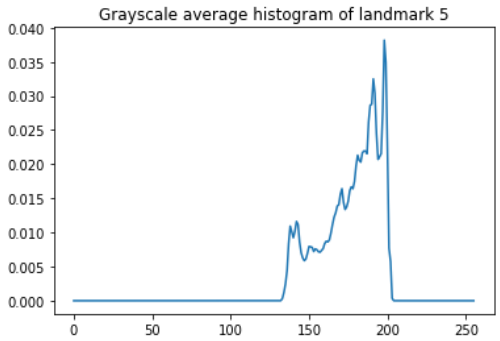

Figura 7: Histograma medio en escala de grises del landmark 5

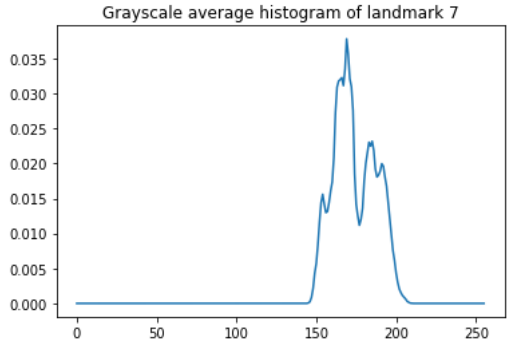

Figura 8: Histograma medio en escala de grises del landmark 7

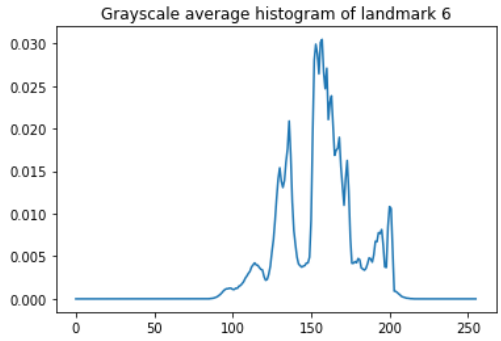

Figura 9: Histograma medio en escala de grises del landmark 6

Se ve cómo ambos histogramas son sustancialmente parecidos, pudiendo provocar falsos positivos en el reconocedor. En en caso del landmark 7, que se confunde con el 6, pese a que en la prueba estática no fallaba, vemos que en la prueba dinámica si presenta un alto porcentaje de error, debido a que estos dos landmarks corresponden al inicio y final de un pasillo, donde en la prueba dinámica, las imágenes de posiciones intermedias tienen histogramas de poca diferencia entre ambos. A pesar de esto, hemos conseguido obtener resultados razonablemente buenos, estando por encima de un porcentaje de acierto del 80 % en todos los landmarks excepto el landmark 2 y 7 debido a los problemas comentados anteriormente.

Para completar los resutados, adjuntaremos un enlance en el que podremos encontrar un vídeo de la prueba dinámica sobre el recorrido que hemos diseñado dentro del escenario virtual, en que podremos ver en la esquina superior izquierda dos parámetros: el primero será el landmark al que pertenece cada frame según la predicción del reconocedor, y por

otro lado, encontramos justo debajo el valor real al que pertenece cada frame según la distribución de landmarks dentro del mapa topológico.

Enlace al vídeo

3.1.2. RGB

En este apartado se va a explicar cómo ha funcionado el algoritmo 1-NN al utilizar las imágenes con color RGB y tomando como distancia entre estas imágenes la distancia euclídea de sus histogramas correspondientes. De la misma manera que el caso anterior, realizamos un entrenamientos estático para comprobar la validez de nuestro dataset, para posteriormente realizar la prueba dinámica sobre el recorrido seleccionado.

En la tabla 3, podremos observar los resultados obtenidos en modo estático.

Landmarks	**Porcentaje de acierto**
0	100 %
1	100 %
2	98,93 %
3	98,78 %
4	98,36 %
5	98,85 %
6	98,72 %
7	96,83 %
8	100 %
Total	98,94 %

Tabla 3: Resultados estáticos en RGB

En comparación con los resultados obtenidos en modo estático con escala de grises, podemos observar una clara mejoría, habiendo obtenido un porcentaje de acierto en RGB del 98,94 %.

Por último, en la tabla 4 mostramos los resultados obtenidos en modo dinámico. De nuevo, vemos que en el caso dinámico disminuye el porcentaje de acierto, consiguiendo un total de 80,74 %, valor superior al obtenido en modo dinámico con escala de grises (77,42 %). En cuanto a la figura 10, vemos cómo de nuevo tenemos el fenómeno de posibles falsos positivos en los cambios de landmark que podemos despreciar debido a que dependen de nuestro etiquetado. No obstante, se observa como para los landmarks 2 y 7 el acierto es algo mayor, de nuevo confundiéndose bastante con los landmarks 5 y 6 respectivamente.

En todos los landmarks, menos el 2 y 7, se obtienen porcentajes de aciertos superiores al 82 %. Asimismo, en los landmarks 2 y 7, a pesar de haber conseguido resultados de nuevo no suficientemente buenos, mejoran ligeramente con respecto a los resultados comentados anteriormente con escala de grises en modo dinámico. Comparando sus histogramas (figuras 11, 12, 13 y 14), vemos cómo esta vez presentan mucha más diferencia pero de nuevo no la suficiente como para poder obtener un acierto completo. No obstante, si nos fijamos en la escala del eje Y podemos ver cómo el numero de píxel medio con cada color básico presentan picos muy similares, lo cual puede explicar los falsos positivos. De forma cualitativa este es un resultado que cabría esperar pues los landmarks 2, y 5 son estancias completamente blancas, y los landmarks 6 y 7 pertenecen al mismo pasillo en

Landmarks	Porcentaje de acierto
0	82,20 %
1	94,13 %
2	42,17 %
3	99,70 %
4	84,11 %
5	96,83 %
6	91,38 %
7	37,04 %
8	99,12 %
Total	80,74 %

Tabla 4: Resultados dinámicos en RGB

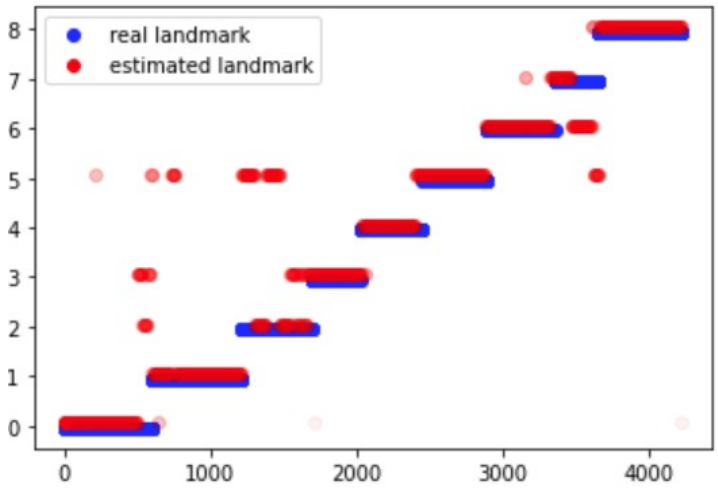

Figura 10: Comparativa de positivos

los que encontramos una gama cromática prácticamente igual. No obstante, como hemos comentado, se obtiene mayor porcentaje de acierto debido a que ahora en nuestra distancia euclídea participan más dimensiones, de manera que las pequeñas diferencias se hacen más notables.

Para completar nuestros resultados, de la misma manera que el caso anterior adjuntamos un enlace a la prueba dinámica donde se muestra en la parte superior el landmark que predice el reconocedor en cada instante del recorrido junto con el landmark correcto para cada frame de manera que podemos comparar los positivos y los falsos positivos.

Enlace al vídeo

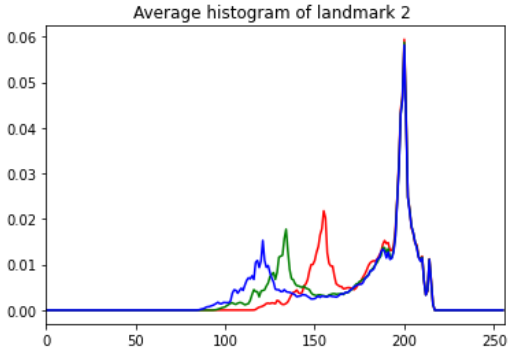

Figura 11: Histograma medio a color del landmark 2

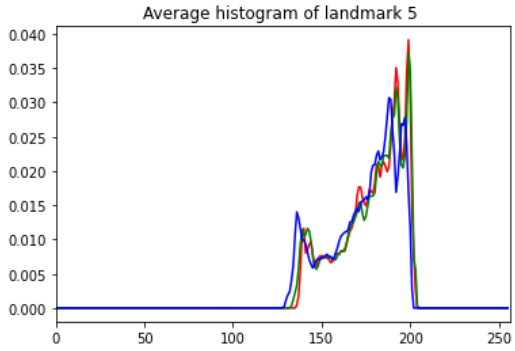

Figura 12: Histograma medio a color del landmark 5

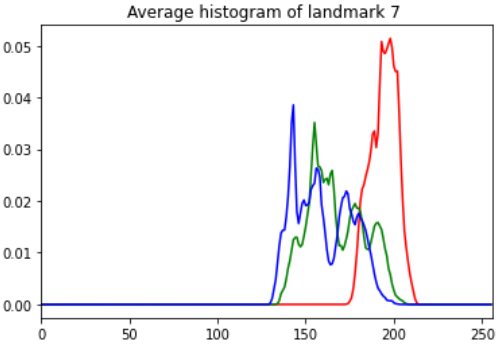

Figura 13: Histograma medio a color del landmark 7

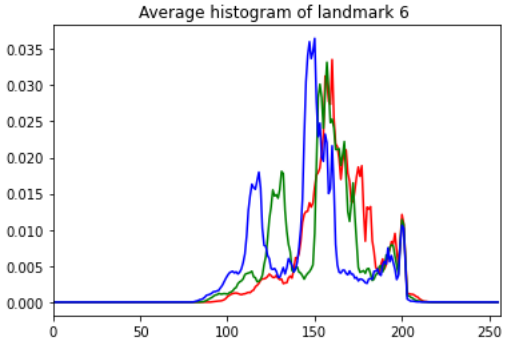

Figura 14: Histograma medio a color del landmark 6

3.2. Redes neuronales

Para dar más variedad de resultados y por puro afán científico también hemos realizado pruebas aplicando redes neuronales convolucionales a la tarea de clasificación de imágenes en los landmarks. Para ello hemos utilizado las mismas imágenes de cada landmark pero reducidas a 128×128 píxeles. Para el experimento se ha usado el esquema que se puede ver en la tabla 5.

Layer (type)	Output Shape	Parameters
conv2d (Conv2D)	(None, 118, 118, 96)	3,944
max_pooling2d (MaxPooling2D)	(None, 59, 59, 96)	0
conv2d_1 (Conv2D)	(None, 53, 53, 256)	1,204,480
max_pooling2d_1 (MaxPooling2D)	(None, 26, 26, 256)	0
conv2d_2 (Conv2D)	(None, 22, 22, 384)	2,457,984
max_pooling2d_2 (MaxPooling2D)	(None, 11, 11, 384)	0
conv2d_3 (Conv2D)	(None, 7, 7, 384)	3,686,784
flatten (Flatten)	(None, 18816)	0
dense (Dense)	(None, 1024)	19,268,608
dense_1 (Dense)	(None, 1024)	1,049,600
dense_2 (Dense)	(None, 9)	9,225

Total params: 27,711,625
Trainable params: 27,711,625
Non-trainable params: 0

Tabla 5: Esquema de la red neuronal usada

Cabe destacar que aunque parecía bastante prometedor, no se ha conseguido obtener un buen resultado en la prueba dinámica, lo que puede deberse a dos motivos:

- Las redes convolucionales buscan patrones simples en las imágenes de entrada y preparan un conjunto de estas características para enviarlos a un separador lineal. Dado que las imágenes obtenidas para el entrenamiento se realizan de manera estática, es posible que al añadir desplazamiento en la prueba dinámica, la perspectiva mueva estos patrones (líneas verticales pasan a diagonales, etc) provocando una extrema confusión y creando diferencias entre las imágenes de entrenamiento y las de test.

- Existen demasiados elementos indistinguibles entre imágenes. Por ejemplo, techos o marcos de puertas, los cuales son iguales en todas las habitaciones y puede generar confusión a la hora de diferenciar ciertas imágenes de distintos landmarks. Si lo combinamos con un sobreajuste de los datos de entrenamiento (entrenamos con 700 imágenes sobre las que se realizan cientos de iteraciones de entrenamiento), esto puede ser una causa de los malos resultados obtenidos.

Con todo ello, en la prueba estática podemos llegar al $99,99\,\%$ de acierto, pero esto puede deberse sin problema al sobreajuste, por lo que no es una métrica extremadamente fiable. Por otro lado, en la prueba dinámica la tasa de acierto es muy oscilante, alrededor del $50\,\%$, con un mejor resultado de $56\,\%$, lo cual no es un gran demostración de efectividad.

En cualquier caso, el código está en el github que hemos comentado anteriormente y podemos encontrar su versión en *.ipynb* o su versión en *.py* bajo la carpeta de *neural network python code*.

Para añadir información, también se ha probado a utilizar las imágenes de test para entrenar la red, por si se diera el caso de que algunos landmarks fuesen indiferenciables o hubiera problemas internos del algoritmo. En cualquier caso, se han extraído una serie de imágenes del dataset de test, se ha entrenado con ellas y se ha probado sobre el vídeo completo (generando redundancia entre el test y el train, lo cual es extremadamente poco fiable, pero haremos la excepción para esta comprobación). El resultado preliminar es un $97\,\%$ de acierto, con una única prueba, en tan solo 7 ciclos de entrenamiento sobre 1000 imágenes y sin comprobar un posible desplazamiento de los frames en la carga (opencv ha dado algún que otro problema al tratar de cargar miles de imágenes de un mismo vídeo). Para ser un poco más honestos, también hemos probado a entrenar con solamente 100 imágenes (salen a 11 por landmark aproximadamente), llegando a obtener un $87\,\%$ de acierto sin demasiadas pruebas, podría mejorar con un número masivo de pruebas reiniciando el modelo.

En conclusión podemos decir que el dataset de entrenamiento generado para las pruebas previas no es el más eficaz para una implementación de redes neuronales ya que el espectro de características que aprende la red no se corresponde con la realidad, y sería necesario utilizar una entrada de datos más similar al test dinámico para obtener unos resultados fiables.

Esto ha quedado patente al obtener un resultado tan alto al entrenar sobre el conjunto de test dinámico, dejando entrever que las dos teorías que se expresaron previamente para justificar el mal desempeño de la red estaban bien encaminadas, aunque hemos hecho un poco de "trampas". En cualquier caso, realizar estas pruebas y divagar sobre sus posibles problemas ha sido realmente instructivo.

No se descarta la eficacia de esta técnica, pero habría que enfocar el problema de una manera un poco distinta desde un comienzo.

4. Conclusiones

En general se ha podido apreciar que se obtienen buenos resultados en todos los modelos. En K-NN por ejemplo, se ha acertado con bastante precisión los landmarks aunque sobretodo ha habido problemas a la hora de clasificar las imágenes del landmark 2. Esto es así, debido a que los landmark 2 y 5 tienen unos colores parecidos (blancos), lo que hace que en escalas de grises se falle bastante, al igual que con RGB (aunque se falle un poco menos). De la misma manera este fenómeno ocurre con los landmarks 6 y 7.

Podemos destacar dos fuentes principales de error generales: la primera es el uso de un simulador virtual que no tiene matices de iluminación y sombras, esto hace que los histogramas de la imágenes sean más uniformes por lo que es más fácil obtener falsos positivos al considerar la distancia mínima. Por otra parte, los porcentajes de error se calculan teniendo en cuenta la clasificación que nosotros hacemos de cada punto del recorrido, esto siempre provoca un porcentaje de error pequeño y difícil de eliminar a la hora de compararlos con la clasificación que se realiza el reconocedor, pues puede interpretar los límites de cada landmark de manera ligeramente distinta a la nuestra.

En cuanto a los resultados estáticos, como cabría esperar se obtienen mejores resultados en comparación con los dinámicos. Esto se debe en un primer lugar, a que las imágenes para el testeo estático fueron seleccionadas manualmente por lo que están tomadas desde una buena posición mientras que la prueba dinámica, al simular el movimiento de un robot, se tienen que predecir muchas más imágenes en posiciones más complicadas. En segundo lugar, esta diferencia de error probablemente se deba también al número de imágenes a probar en cada prueba, ya que habrá un mayor número de imágenes que se tienen que tratar en un vídeo que en la prueba estática definida.

Por otra parte, se ha intentado ampliar el número de algoritmos a probar en este problema para ver otras posibles soluciones. Este ha sido el caso de las las redes neuronales convolucionales, en el cual no hemos conseguido obtener un buen modelo que resuelva bien el problema.

Finalmente, se proporciona un enlace a Github con todo el código implementado para la práctica, además de los vídeos, el dataset de los landmarks y algunas gráficas que no hemos incluido en esta memoria:

https://github.com/BenitezDev/Visual-Topological-Maps