**Solución propuesta (entrenamiento + alg segmentación)**

A la hora de implementar la parte de segmentación, hemos llegado a tomar una serie de decisiones concretas con el objetivo de lograr los mejores resultados posibles. En este sentido, parecen claras las dos vías por las que se puede optimizar un algoritmo de segmentación: mejorar la calidad de la segmentación y mejorar los tiempos de ejecución.

Nosotros tuvimos, ya desde el primer momento, claro que lo importante sería lograr una buena calidad en la segmentación, y que a partir de ahí el objetivo sería mejorar los tiempos que tarda el algoritmo en proporcionar la segmentación de cierta imagen (ya que en el robot real esto resulta ser importante).

Nuestra primera medida, frente al “diseño” original del algoritmo, fue cambiar el espacio de colores de RGB a HSV. A esto unimos el que finalmente nos quedamos solo con H y S, dejando fuera la V (para tratar de evitar problemas con los cambios de iluminación). Esto resultó ser definitivamente importante para lograr un buen clasificador.

Para etiquetar las imágenes se usó el programa proporcionado por el profesor, que permite colorear un vídeo abierto con OpenCV, por lo que tomamos algunas imágenes y las coloreamos, tratando de obtener datos variados para la línea, el suelo y las marcas. Con esto construimos nuestro dataset de entrenamiento, con unos 30000 píxeles (lo consideramos una cantidad muy elevada, pero hemos obtenido buenos resultados con ello).

Por otro lado, aunque la idea inicial era usar un clasificador euclídeo, de manera similar al desarrollado en la asignatura de Reconocimiento de Formas, pensamos que sería ventajoso buscar algo un poco más complejo pero que segmente mejor. Probamos, usando bibliotecas como sklearn, distintas opciones: primero un clasificador gaussiano, luego una red neuronal de 3 capas, luego una de dos capas (haciendo diversas pruebas con distintos números de nodos en cada capa). Nos quedamos finalmente con una red neuronal de 2 capas con 3 y 2 nodos en cada una. Con ello obtenemos unos resultados con altos porcentajes de acierto (99.5% de acierto usando un 90% de datos de training y un 10% de datos de test), pero asegurando a la vez una velocidad de ejecución digna para cuando la segmentación se realice dentro del ciclo del robot real.

Así, al ejecutar el programa se ejecuta el entrenamiento pertinente, y posteriormente se va leyendo el vídeo que se deseé segmentar y calculando dicha segmentación, coloreando y mostrando el resultado a la vez.

Pasan a comentarse ahora las mejoras llevadas a cabo desde la primera entrega de segmentación:

1. Gracias a un uso más adecuado de la biblioteca de sklearn, se logró que la ejecución del algoritmo de segmentación fuera paralelizable. De esa manera, no perdemos demasiada potencia de cálculo por utilizar un algoritmo más complejo, como son las redes de neuronas.
2. Se pasó a obtener los frames del vídeo de 4 en 4. De esa forma, reducimos el coste computacional a 1/16 del inicial.
3. Se redujeron los nodos en cada capa de la red neuronal, logrando así una velocidad de cómputo mayor sin perder mucha calidad en la segmentación.
4. Se implementó la parte opcional del seguimiento de la pelota (explicado a continuación).
5. Como medida para aumentar aún más el acierto en la segmentación, se decidió aumentar la saturación de la cámara a la hora de tomar los vídeos.
6. A lo largo de 2 días, tabajamos en el laboratorio de robótica. Allí pudimos grabar y segmentar vídeos tomados desde la propia cámara del robot, con el objetivo de tener ya un vídeo del entorno real de la práctica, además de empezar a probar situaciones ya reales de seguimiento de líneas en el suelo y de seguimiento de una pelota de ténis.

**Resultado de segmentación. Prueba en el robot real:** Tras probar el algoritmo que tenemos en el robot de la sala de robótica, podemos certificarnos de que funciona bien en un entorno real controlado. Además, podemos asegurarnos de que los tiempos de ejecución son, como mínimo, adecuados. En nuestro caso, la fase de segmentación se hace de tal forma que conseguimos concluirla en 0.01 segundos, lo que nos permite ejecutar muchos frames en cada segundo (serían 77 si el step que implementamos fuera el total del programa).

**Parte opcional: seguimiento de pelota:** También realizamos la parte opcional de la práctica, referida al seguimiento de un objeto esférico por parte del robot, que va controlando su distancia a dicho objeto según los resultados del tamaño de la pelota relativo a la cámara del robot. Logramos adaptar el algoritmo de segmentación para poder obtener la pelota de un vídeo, y calculamos la manera de obtener la distancia relativa de la cámara a la pelota, en base a la explicación dada en clase. También hemos realizado una primera integración de esto con la parte de control, de tal manera que el robot se acerque a la pelota si la encuentra a lo lejos y se aleje si la tiene muy cerca. Probamos el algoritmo en forma de “Brain” en el robot del laboratorio de robótica, logrando unos resultados interesantes. Incluimos en la carpeta de entrega un video de demostración del seguimiento junto con la segmentación hecha por el mismo programa.

**Breve explicación del código Python**

Para crear el dataset de entrenamiento y test,

Al comenzar a ejecutar el programa, se llama al clasificador, y se le pide que entrene a partir de los datos del dataset, guardado en un fichero (con datasetGenerator se genera este fichero, a partir de las imágenes coloreadas con etiquetaImagenes).

Se inicia la captura de imágenes, en base a un vídeo concreto (aunque puede realizarse sobre un vídeo en directo desde una cámara). Cada imagen recibida se reduce en un factor de 4, vertical y horizontalmente (no hemos considerado el trabajar con imágenes algo pequeñas un problema para el futuro, pues el tamaño sigue siendo considerable para poder ver correctamente la línea y cada marca), y en un bucle, se van leyendo las imágenes. Se pasa cada imagen al formato HSV, y se obtiene el H y el S. Se clasifica con predict cada pixel de la imagen con el clasificador, y se muestra por pantalla la imagen segmentada. Entre medias se realizan pruebas del tiempo tardado, usadas para el apartado correspondiente de esta memoria.

**Imágenes representativas**

Se incluyen imágenes representativas de los resultados obtenidos al segmentar:

**Tiempos de ejecución**

El tiempo de ejecución de un ciclo fue, de media, de 0,013 segundos. La velocidad de la segmentación fue de 77 frames por segundo.

**Conclusiones y mejoras pendientes**

Pensamos, en conclusión, que nuestro algoritmo de segmentación obtiene buenos resultados con las pruebas llevadas a cabo. Es cierto que podría ser más rápido (especialmente en nuestra primera versión, de la primera entrega de esta parte), pero hemos tratado de encontrar un equilibrio entre resultados y tiempos de ejecución que nos permita emplear este algoritmo una vez usemos el robot real.