**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 tarde el algoritmo (ya que en el robot real esto será algo 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 dataste 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 librerías como sklearn, distintas opciones: primero un clasificador gaussiano, luego una red neuronal de 3 capas, luego una de dos capas (con distinto número de nodos en cada una), quedándonos finalmente con una red neuronal de 2 capas con 3 y 2 nodos. 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 d 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.

**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. Un día fuimos al laboratorio de robótica, donde además de familiarizarnos con el robot probamos esta parte opcional, logrando unos resultados interesantes (siendo nuestro único problema el que la parte de control de estas pruebas aún no estaba correctamente parametrizado a la hora de elegir las velocidades de aceleración y giro.

**Breve explicación del código Python**

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**

**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.