30 de marzo de 2020

Memoria Segmentación - Robótica y Percepción Computacional

Javier Antonio Román López (X150128)
Marcos Tirado Soto (w140176)

Índice

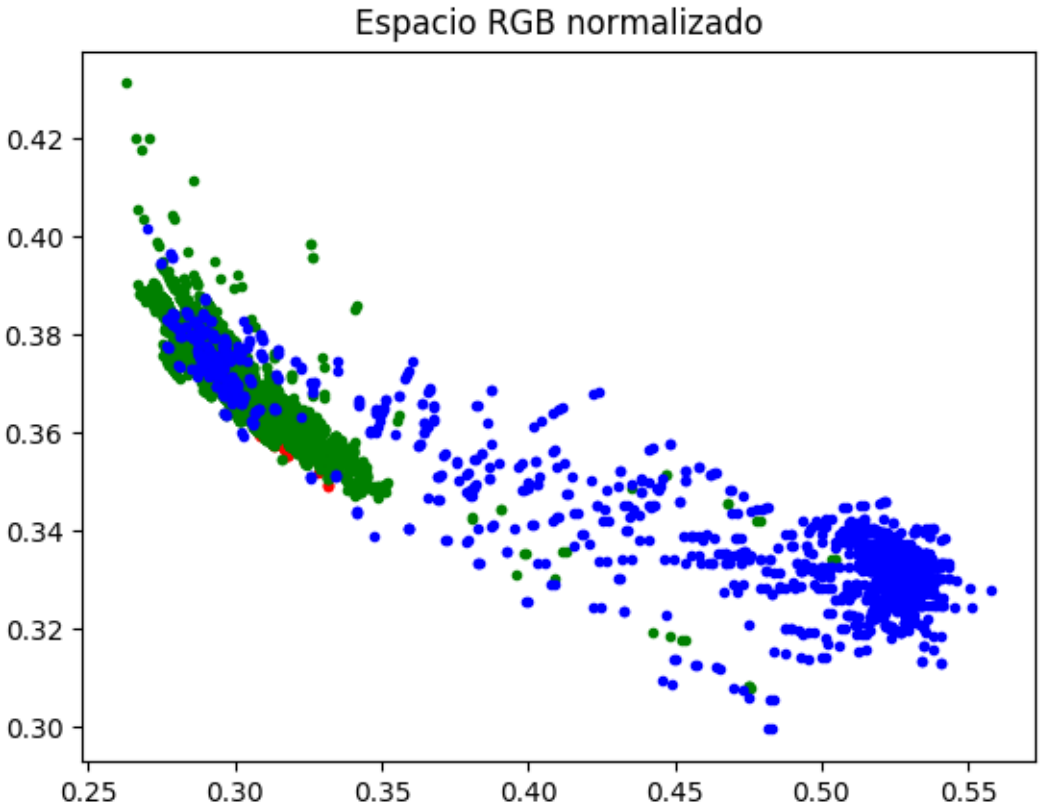
1. Solución

El modelo utilizado ha sido el k-neighbors de la librería de sklearn, específicamente el NearestCentroid con resultado bastante bueno. Para el entrenamiento se han utilizado dos imágenes, una de los vídeos de muestra y la misma sobre la que ha pintado con tres colores los distintos objetos ha identificar. Posteriormente se aplica dicho modelo a 1 de cada 25 frames del mismo vídeo.

2. Explicación del código

El código sigue el esqueleto facilitado. Primero se crea el clasificador y se cargan las imagenes con las que se entrenara, pasándolas a formato RGB. Se crean tres vectores de numpy con los datos correspondientes a cada uno de los objetos a identificar a partir de los píxeles pintados y otros tres vectores del mismo tamaño que utilizaremos como etiquetas. Se realiza el entrenamiento del clasificador con la función fit y de entrada la concatenación de los vectores de los datos por un lado y de las etiquetas por otro. Se prepara las variables necesarias para la captura del vídeo y se procede a la transformación del mismo si la variable ret de la captura de la imagen no es nula. Solo cogemos 1 frame de cada 25 para ello usamos la variable count y comprobamos que sea modulo 25. En caso de ser un frame de los seleccionados este se redimensiona para poder pasarlo al clasificador con la función predict dando como resultado una imagen nueva con los pixeles pintados de los colores correspondientes. Finalmente se escribe cada una de las imágenes y se genera el vídeo deseado.

3. Imágenes



La distribución en dos dimensiones se me muy superpuesta sin embargo en tres dimensiones se puede ver como las nubes de puntos difieren mucho, por lo que la diferenciación de los objetos con el algoritmo utilizado tiende a ser bastante buena.

4. Tiempos de ejecución

- Tiempo entrenamiento:
  0.2375790596008 segundos
- Tiempo total (solo 1 de cada 25 frames):
  9.33829237 segundos

5. Conclusión

El algoritmo utilizado da un buen resultado y en un tiempo muy aceptable. Queda pendiente como mejora la implementación del calculo de distancia a objetos.