Universidad Politécnica de Madrid

Reconocimiento de iconos

Robótica y Percepción Computacional

Luciano García Giordano - 150245
Gonzalo Flórez Arias - 150048
Salvador González Gerpe - 150044

Índice

Introducción

En esta memoria se recoge el trabajo realizado para el módulo de Reconocimiento de Iconos de la asignatura Robótica y Percepción Computacional. En dicho módulo se ha logrado, partiendo del previo módulo de segmentación, lograr reconocer iconos entre 4 símbolos distintos, tanto en imágenes estáticas como en un vídeo dinámico grabado con el robot de la asignatura. Para ello se han obtenido una serie de imágenes ejemplo de cada icono y se han calculado los momentos invariantes de Hu de dichas imágenes, formando así un dataset de entrenamiento para luego clasificar las imágenes estáticas y el vídeo.

Dicho video incluye la segmentación de la imagen en cada frame, a su derecha la extracción del símbolo y, por detrás a la izquerda, en la termina, se puede observar el nombre de cada uno de los símbolos o que no hay símbolo en la imagen o que éste toca el borde de la imagen. Nótese que, por tratarse de un contexto real, el video tiene poca estabilidad por la vibración y estabilidad del propio robot.

Dataset de entrenamiento

Para poder generar un buen dataset de entrenamiento se han seguido las pautas indicadas por el profesor de la asignatura. Para empezar, se imprimió el siguiente fichero:

Figura 1: archivo con los recortables de los iconos

Que incluye los 4 elementos que debemos ser capaces de clasificar: la cabina telefónica, la cruz, las escaleras y el servicio de caballeros.

Con esos mismos 4 recortes se fue un día al laboratorio, y empleando la cámara del robot de la asignatura, se tomaron fotos de cada uno de los símbolos. Se procuró variar la posición de la cámara y la orientación de los símbolos, para poder generar un dataset que no sólo funcione bien a la hora de entrenar, sino que clasifique bien los iconos al enfrentarse al vídeo real.

Figura 2: ejemplo de una de las imágenes tomadas

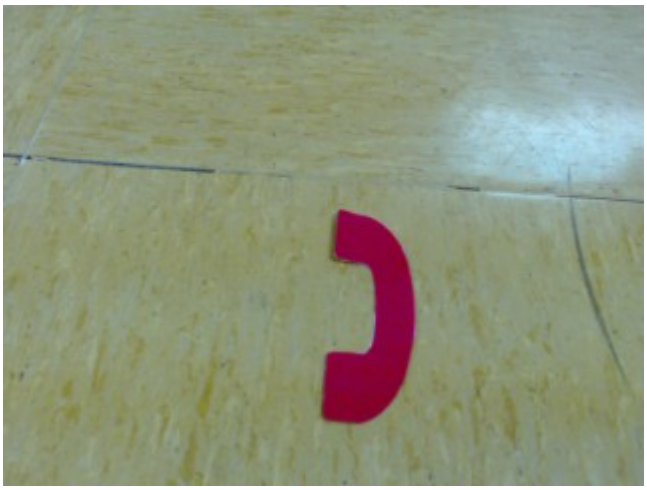

Figura 3: ejemplo de una de las imágenes tomadas

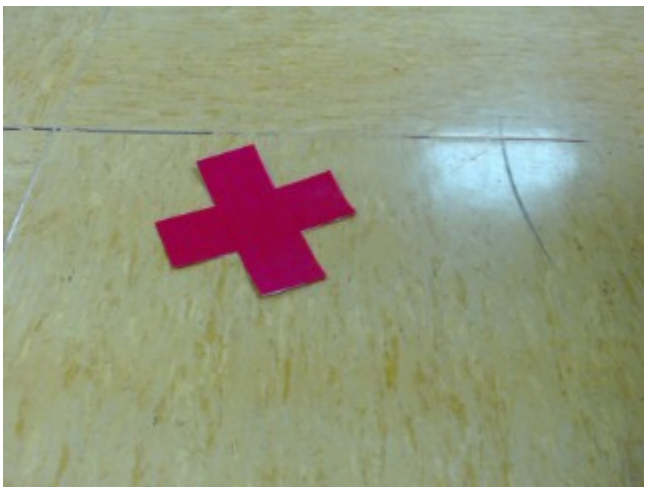

Figura 4: ejemplo de una de las imágenes tomadas

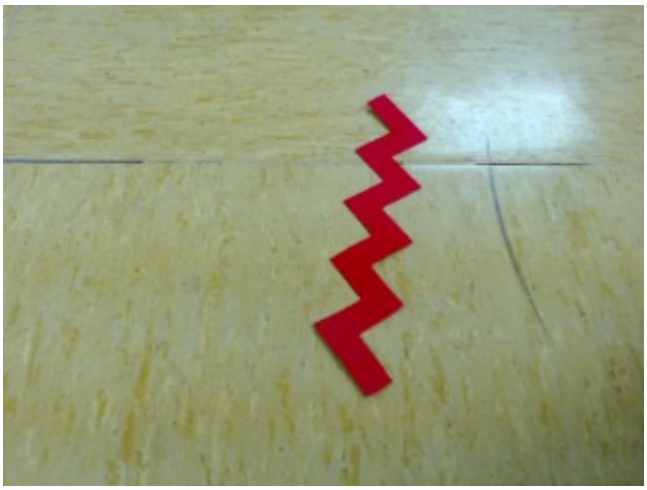

Figura 5: ejemplo de una de las imágenes tomadas

Acabamos haciendo más de 100 fotos por símbolo, pues no estábamos seguros de si algunas fotos no se segmentarían del todo bien, de tal manera que aun desechando algunas tuviésemos al menos 100 de cada símbolo.

Posteriormente, se segmentaron las imágenes tomadas, y se desechó todo aquello que no fuese un icono en la imagen. Se calcularon los momentos de Hu, según la sentencia siguiente:

```
moments = cv2.HuMoments(cv2.moments(im)).flatten()
```

De cada uno de los tipos de iconos se guardó un archivo .txt con los momentos de Hu de los elementos de dicho tipo, de tal manera que a la hora de clasificar se pudieran recoger los datos de entrenamiento de dichos ficheros. Es este, por tanto, nuestro dataset de entrenamiento.

Clasificadores

Se han empleado para el reconocimiento de iconos varios tipos de clasificadores. Por un lado, se han usado los indispensables para esta práctica: el algoritmo 1-NN, o clasificador del vecino más próximo, y el algoritmo basado en la distancia de Mahalanobis.

Por otro lado se han decidido probar otros clasificadores, para ver si se podían mejorar los resultados de los 2 obligatorios. Se ha probado con Multilayer Perceptron, Naive Bayes, Support Vector Machine, K-NN (K>1) y Decision Tree. Para la implementación de todos los clasificadores se ha empleado Sklearn, salvo en el caso de la distancia de Mahalanobis, en que se ha usado Scipy para obtener la distancia de Mahalanobis, obteniendo de manera manual las medias y las matrices de covarianza de cada clase.

Resultados

Prueba estática

Se usó para la prueba estática el método LeaveOneOut de sklearn, de tal manera que en cada iteración se entrena con todo el dataset menos un dato, y se trata de clasificar dicho dato, usando como clasificador cada uno de los ya mencionados, buscando en ellos los parámetros óptimos que mejores porcentajes de acierto generen. Si el clasificador acierta en la predicción, se apunta y se continúa el bucle. Al final se puede obtener el porcentaje de aciertos como el número de iteraciones acertadas entre el número de datos totales.

Se aplicó dicha prueba estática a todos los clasificadores anteriormente expuestos (salvo al Multilayer Perceptron, del que no logramos obtener resultados lógicos), lográndose los siguientes resultados:

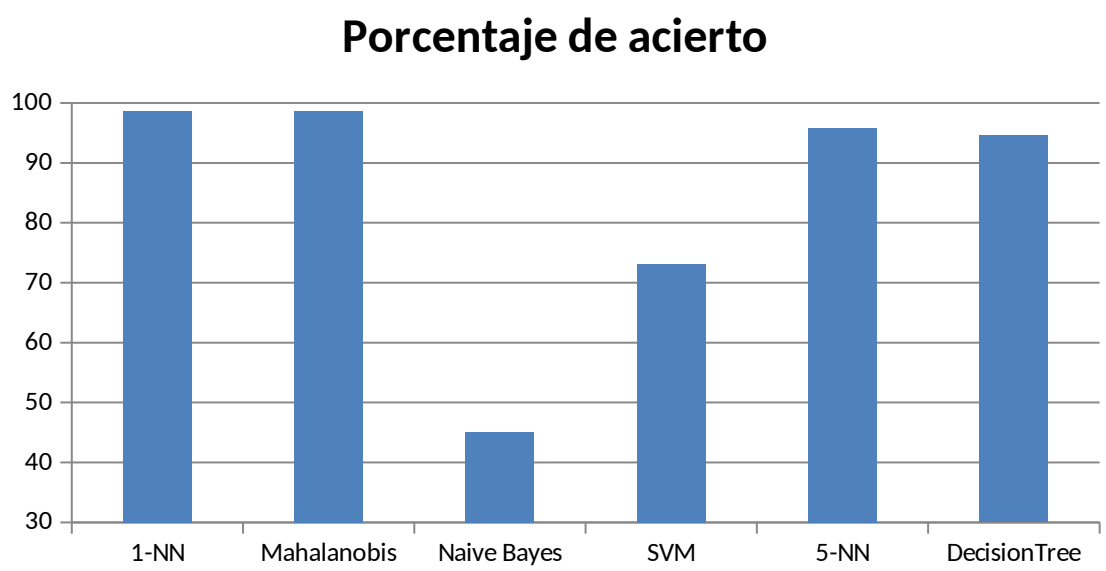

Figura 6: porcentajes de acierto de los distintos clasificadores

Como se puede observar, los algoritmos propuestos para la realización de esta práctica, 1-NN y Mahalanobis, han acabado generando unos resultados muy buenos, con unos aciertos en torno al 98.7%. Por otro lado, algunos clasificadores como Naive Bayes, que se queda en un 45%, o la máquina de soporte vectorial (SVM), con un 73,17%, se quedan indudablemente por debajo del resto. Además, nos ha sorprendido inicialmente el alto porcentaje de acierto del árbol de decisión [3], con un 95% de porcentaje de aciertos.`Todos los resultados aquí comentados se han obenido mediante el uso de validación cruzada de tipo Leave-One-Out.

Prueba dinámica

Como prueba final, se he grabado un vídeo con la cámara del robot del laboratorio, siguiendo un circuito en el que aparecen varias veces los distintos iconos a reconocer, y se ha realizado un programa que muestra cómo se va segmentando el vídeo, cómo se binarizan los iconos

para separarlos del resto de la imagen, y el icono que el reconocedor encuentra en cada momento. Ya que aunque no se encuentre ningún icono en pantalla (ni siquiera haya ningún valor a 1 en la imagen binarizada) se calculaban los momentos de Hu, lo que se ha hecho es que no se realice ningún intento de reconocimiento si no hay al menos 200 píxeles que representen a un icono en pantalla. A la hora de unificar este módulo con el resto del proyecto, por supuesto, deberemos elevar la lógica y considerar que sólo se reconozcan iconos si están enteros dentro de los límites de la pantalla, pues si sólo se puede ver medio icono los momentos de Hu no nos permiten discriminar correctamente.

Se muestran a continuación algunos ejemplos de la prueba dinámica. A la izquierda se muestra el vídeo real, en el centro la segmentación (se elimina un porcentaje de los píxeles superiores, para poder aumentar la velocidad sin perder mucha información importante), y a la derecha la binariazación de los símbolos. Detrás de esta última, la traza de ejecución que indica en cada momento qué símbolo se ve en pantalla, o “ningún símbolo” si no se ve ninguno.

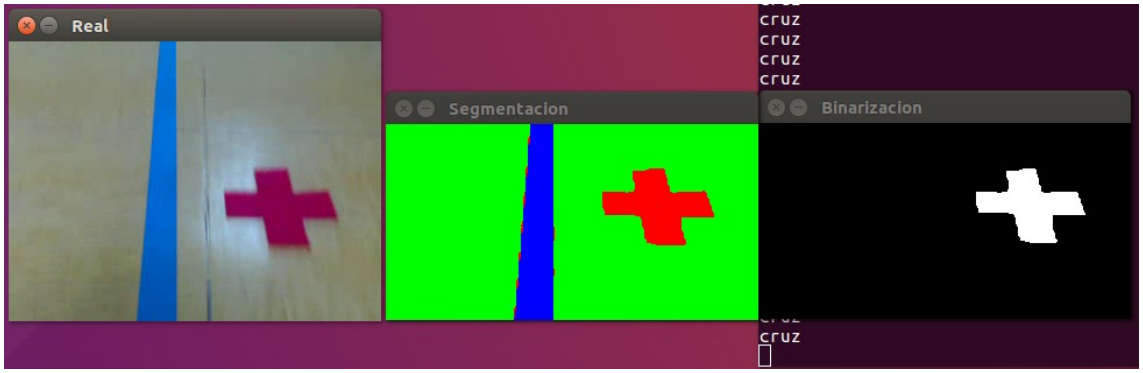

Figura 7: reconocimiento de una cruz

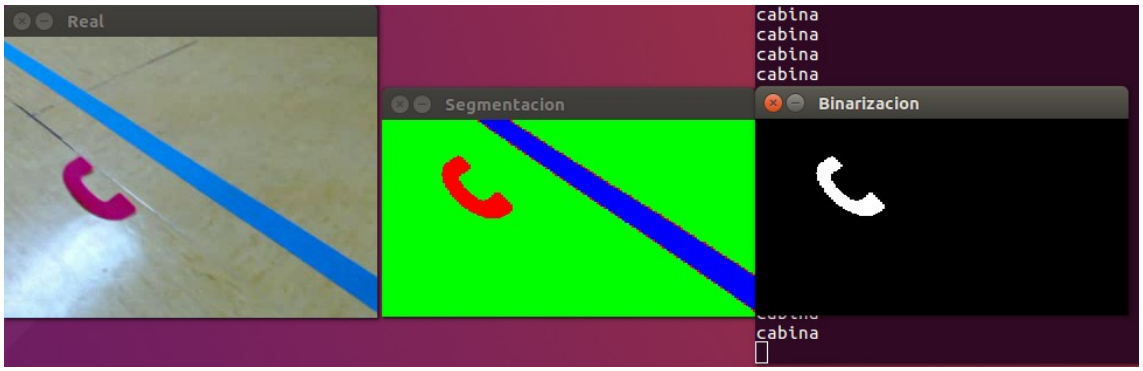

Figura 8: reconocimiento de una cabina telefónica

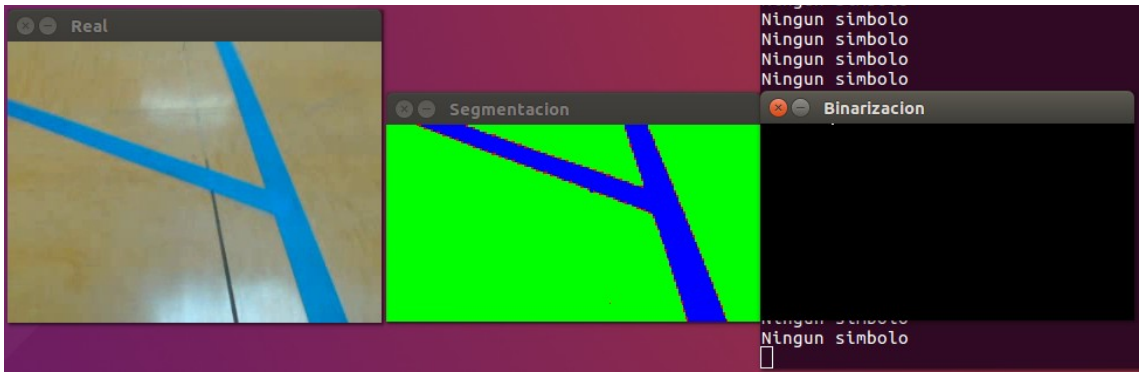

Figura 9: en este frame no se encuentra ningún símbolo

Además de esta memoria, se ha entregado un vídeo en el que también se pueden comprobar los resultados de esta prueba dinámica.

Conclusiones

En conclusión, hemos obtenido, mediante el procedimiento explicado en clase, un programa en Python capaz de realizar el análisis de una segmentación de formas de una imágen, reconociendo algunos patrones específicos, que en este proyecto corresponden a cuatro formas específicas. Para ello, hemos utilizado las funciones de la biblioteca de Python OpenCV, además de otras funciones implementadas en otros paquetes, como scikit-learn y SciPy.

Pensamos que una de las claves de los buenos resultados se deban a que, tanto a la hora de la segmentación del vídeo, como a la hora de tomar las imágenes de entrenamiento de los distintos iconos, se aplicó un algoritmo de segmentación en sí muy correcto, con lo que se logra distinguir muy claramente aquello que es un icono y aquello de que no. Esto, unido a que los momentos invariantes de Hu han acabado siendo buenos discriminantes de los distintos tipos de símbolos, nos ha permitido llegar a los resultados obtenidos.

Referencias

[1] Documentación scikit para validación cruzada: http://scikitlearn.org/stable/modules/cross_validation.html

[2] Documentación scipy para calculo de distancias (incluída euclidea y Mahalanobis): https://docs.scipy.org/doc/scipy-0.14.0/reference/spatial.distance.html

[3] Scikit-learn – Decision Tree Classifier Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier