



Universidad Rey Juan Carlos

Visión Artificial

Clasificación de tipos de pastillas usando Visión Artificial

Introducción	3
Idea principal para el Clasificador	3
Clases	4
Estudio teórico y Descripción del Código	7
Color	7
Umbralización	7
Filtro de Gauss	7
Clases y Reconocimiento de Patrones	7
Notas finales y Bibliografía	9
Bibliografía utilizada:	9

Introducción

En la práctica planteamos el desarrollo de un sistema automático de clasificación de diferentes tipos de pastillas de diversas formas, texturas, tamaños y colores. Para ello utilizaremos técnicas de clasificación supervisadas de objetos dado que las pastillas pueden presentar zonas de brillos y sombra, así como variar su posición dependiendo de la imagen analizada.

Para el correcto desarrollo de la práctica y el uso de las librerías ofrecidas por OpenCV se proporciona una base de datos de pastillas con la que podemos entrenar el clasificador y otra base que será la utilizada a modo de test.

El video de ejecución de la práctica puede verse en el siguiente enlace:

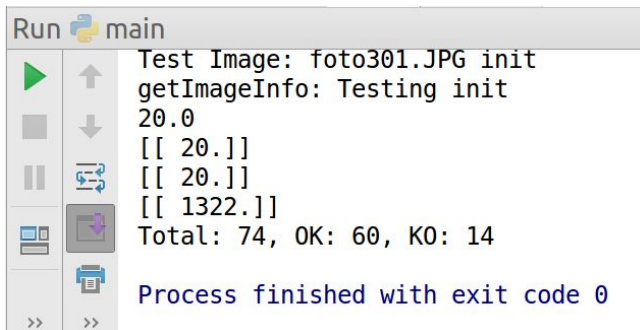
- <https://youtu.be/2a7rqUx-JtQ>

Idea principal para el Clasificador

En la práctica utilizaremos los clasificadores que trae por defecto OpenCV, con lo que tendremos que "entrenar" al programa para que clasifique los objetos que queremos. La idea principal es recorrer una serie de imágenes ya clasificadas, denominadas imágenes de entrenamiento (training) y almacenar las características que consideremos oportunas. Lo más relevante que podemos apreciar en el conjunto de entrenamiento son el color y la forma de la misma.

Para el color he definido varios rangos de colores con los mínimos solapamientos posibles, almacenando por cada clase sus máximos y mínimos en formato HSV. Para la forma vamos a hacer uso de técnicas más avanzadas para conseguir determinar el borde y, a partir de ello, conseguir variables tan determinantes como pueden ser el rectángulo contenedor o los centroides de la imagen.

Una vez determinadas las variables y las etiquetas de las clases a utilizar (cabe destacar que daremos la posibilidad al usuario de poder almacenar o recuperar desde un fichero estos datos), procederemos a iterar sobre las imágenes de test, aplicando el mismo proceso para el borde. Para el color he decidido recorrer todos los posibles colores que tenemos almacenados para las pastillas (máximos y mínimos) y crear una máscara por cada imagen, quedandome con la máscara que tenga más píxeles de todas las configuradas.





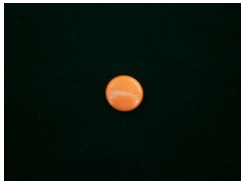




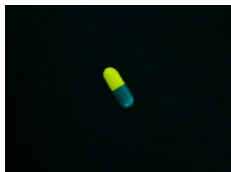












```
Run main
Test Image: foto301.JPG init
getImageInfo: Testing init
20.0
[[ 20.]]
[[ 20.]]
[[ 1322.]]
Total: 74, OK: 60, KO: 14
Process finished with exit code 0
```

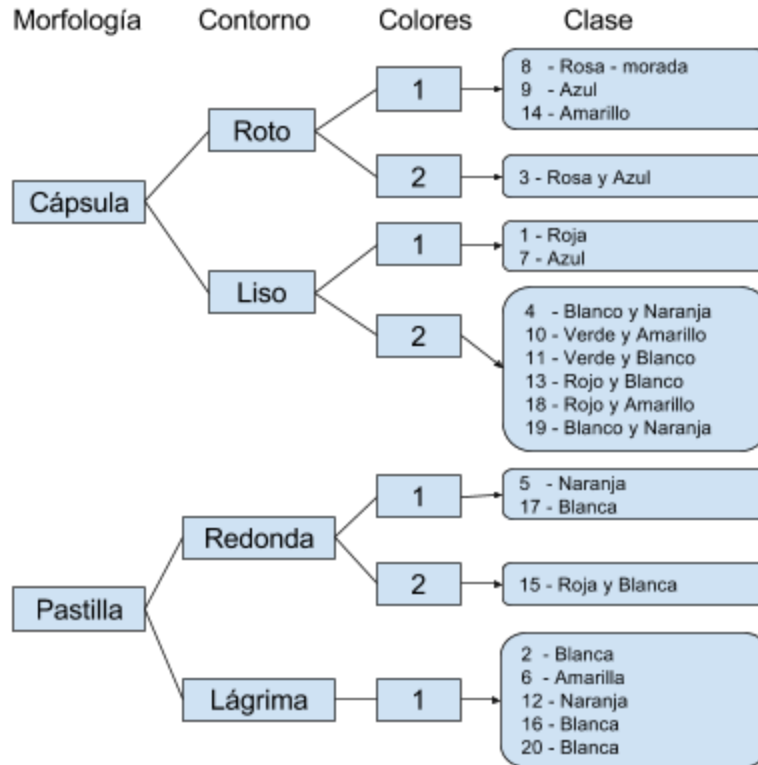
Por último, por cada acierto (se sabe que hemos acertado porque las imágenes están divididas en carpetas por clase) añadiremos un punto a un contador que mostraremos al final del programa, con los aciertos y los errores.

Clases

Como parte del entrenamiento para el clasificador, se han proporcionado 20 clases de pastillas con distintas características en cuanto a forma, color y textura, tal que:

Clase 1 <ul style="list-style-type: none"> • Roja • Cápsula 		Clase 2 <ul style="list-style-type: none"> • Blanca • Ovalada • Marca central 	
Clase 3 <ul style="list-style-type: none"> • Rosa y azul • Cápsula • Marca central • A veces con texto 		Clase 4 <ul style="list-style-type: none"> • Blanca y Naranja • Cápsula 	
Clase 5 <ul style="list-style-type: none"> • Naranja • Redonda 		Clase 6 <ul style="list-style-type: none"> • Amarilla • Forma de lágrima • A veces con texto 	
Clase 7 <ul style="list-style-type: none"> • Azul • Cápsula • A veces con texto 		Clase 8 <ul style="list-style-type: none"> • Rosa - morada • Cápsula • A veces con texto • Marca central 	
Clase 9 <ul style="list-style-type: none"> • Azul • Cápsula • Marca central 		Clase 10 <ul style="list-style-type: none"> • Verde y amarilla • Cápsula 	

Clase 11 <ul style="list-style-type: none"> • Blanca y verde • Cápsula 		Clase 12 <ul style="list-style-type: none"> • Naranja • Forma de lágrima 	
Clase 13 <ul style="list-style-type: none"> • Blanca y roja • Cápsula • A veces con texto 		Clase 14 <ul style="list-style-type: none"> • Amarilla • Cápsula • Marca central 	
Clase 15 <ul style="list-style-type: none"> • Naranja y Blanco • Redonda 		Clase 16 <ul style="list-style-type: none"> • Blanca • Ovalada • Con marca 	
Clase 17 <ul style="list-style-type: none"> • Blanca • Redonda 		Clase 18 <ul style="list-style-type: none"> • Roja y amarilla • Cápsula • Con textura 	
Clase 19 <ul style="list-style-type: none"> • Blanca y naranja • Cápsula 		Clase 20 <ul style="list-style-type: none"> • Blanca • Lágrima 	



Atendiendo a la morfología podemos distinguir claramente entre cápsulas y pastillas. Las cápsulas serán aquellas más alargadas mientras que las pastillas están más cercanas a la morfología de una circunferencia, atendiendo a su circularidad.

El contorno nos determinará si la cápsula está partida por la mitad o si el borde es liso; en el caso de las pastillas quedará determinada la forma de lágrima o de círculo.

Por último, atendiendo al número de colores y al rango de los mismos, podremos discernir entre los 20 diferentes tipos de clases que se nos presentan como ejemplo.

Como puede observarse, si bien hay clases que se pueden diferenciar perfectamente, otras requieren un análisis más exhaustivo. Por ejemplo, los centroides de los colores de las dos mitades de las pastillas que, en el espacio de color RGB serían 6 valores (si las pastillas tienen un solo color los valores de los centroides en cada uno de los tres planos coinciden).

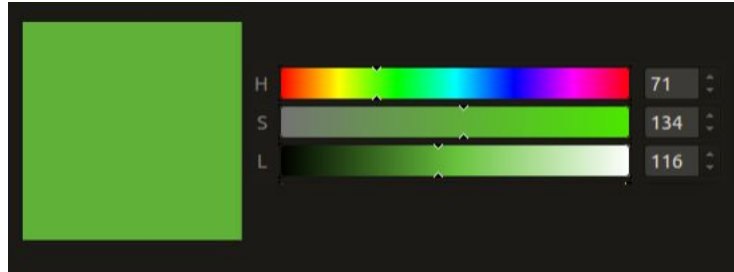
También me he encontrado con clases muy similares, que requerirían un tratamiento mucho más complejo que queda fuera del objetivo de la práctica (Clases 2, 16 y 20 o las Clases 4 y 19).

Estudio teórico y Descripción del Código

Para comprender perfectamente esta práctica, hemos de definir ciertos conceptos teóricos.

Color

Conceptualmente openCV busca los píxeles de la imagen que estén entre un valor mínimo y máximo, creando 2 *Arrays* con los valores H, S y V, *Hue*, *Saturation* and *Value*. Para la práctica he utilizado GIMP para definir los rangos máximos y mínimos.

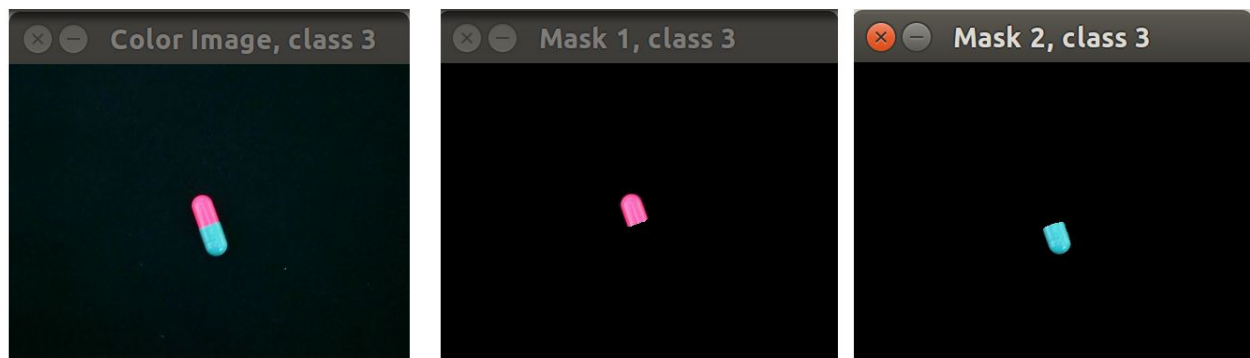


Umbralización

Se trata de una técnica de preproceso por el que a cada pixel de la imagen se le otorga 0 ó 1 (negro o blanco) en la imagen de salida de modo que se crea una imagen binaria a partir de una en escala de grises. Normalmente se utiliza el histograma para definir el umbral.

Filtro de Gauss

Técnica de filtrado que utilizamos tras el preproceso básico por la que eliminamos la información no deseada dejando pasar sólo aquello que interesa para la segmentación; el filtro de Gauss es como el del cálculo de la media, pero dando mayor importancia a los píxeles centrales de la vecindad, quedando una imagen más o menos borrosa dependiendo del diámetro de la campana, suavizando y eliminando ruido con ello.



Clases y Reconocimiento de Patrones

La segmentación consiste en la partición de una imagen en regiones homogéneas con respecto a las diferentes características de la misma; he tratado de hacer una segmentación mediante umbralización y detección de bordes para poder catalogar las imágenes dentro de las 20 clases predefinidas; para ello he escogido diferentes rangos de colores (HSV) en la fase de entrenamiento para aplicar un "*Adaptative Threshold*", al que pasando como parametro una imagen gris, nos define el borde con el que estamos tratando.



En el posterior testeo de las imágenes, podemos estudiar el rango de color y la forma de la misma para poder cotejarlos contra los ejemplos que hemos obtenido en la fase de entrenamiento.

Las características (elegidas debido a su poder discriminante) serán tratadas matemáticamente. Por ejemplo, la forma del objeto nos permitirá determinar mediante su firma si el centroide (o centro de masas) o según la circundaridad corresponde a una cápsula o a una pastilla. Los colores también son determinantes. Se han elegido un total de 18 características que son utilizadas por el algoritmo para encontrar el “vecino más cercano”

```

275 def getColorInfo(hsv, image, th3,j,i,w,h):
276     # loop over the boundaries
277     maxColorsInMask = 0
278     color = next(iter(colors.values()))
279     #rect = cv2.rectangle(image, (x,y),(x+w,y+h), (0,0,255), 3)
280     rect = image[i:i + h, j:j + w]
281     median = rect.mean()
282     for key in colors:
283         lower_upper = colors[key]
284         # create NumPy arrays from the boundaries
285         lower = np.array(lower_upper[0], dtype="uint8")
286         upper = np.array(lower_upper[1], dtype="uint8")
287
288         # find the colors within the specified boundaries and apply
289         # the mask
290         mask = cv2.inRange(hsv, lower, upper)
291         output = cv2.bitwise_and(hsv, hsv, mask=mask)
292
293         pixelInMask = cv2.countNonZero(mask)
294         if (pixelInMask > maxColorsInMask):
295             maxColorsInMask = pixelInMask
296             color = key
297
298     return colors[color]

```

El código de la práctica está dividido en diferentes funciones que hacen más legible la lectura y comprensión del mismo. Por ejemplo, la función `getColorInfo` recibe como parámetros la imagen y el área donde se encuentra nuestro objeto a reconocer e itera sobre el mapa de rango de colores que hemos usado en la fase de entrenamiento para ir creando diferentes máscaras para la imagen, devolviendo los colores que corresponden a la máscara que más

pixeles tenga. Este resultado es utilizado para llamar al método `findNearest` y con ello obtener la clase a la que supuestamente pertenece nuestro objeto.

Notas finales y Bibliografía

En un principio he intentado realizar el entrenamiento siguiendo el metodo *HAAR Cascade*, pero la complejidad aumentaba, por lo que he decidido hacerlo haciendo uso de los conceptos aprendidos en la parte teórica de la asignatura. En la asignatura no se ha explicado nada de la transformada wavelet de Haar, por lo que he considerado que queda alejado del objetivo de la misma.

La práctica es muy compleja. Desde la instalación de todas las herramientas hasta la aplicación de los conceptos teóricos con las librerías elegidas; no obstante, es satisfactorio ver como los resultados coinciden con lo esperado y como hemos de aplicar uno a uno los procesos y que sentido tienen en cada fase del tratamiento de imágenes digitales.

Durante la realización de la práctica me he encontrado con falsos positivos, por lo que he tenido que ir cambiando los rangos de los colores y de otras directrices a medida que avanzaba con la misma. Si pudiésemos controlar la iluminación en la fase de captura se facilitaría el proceso y podríamos reducir brillos e iluminaciones no uniformes.

Bibliografía utilizada:

- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html
- <http://robologs.net/2016/05/18/detectar-multiples-colores-con-opencv-y-python/>
- <http://docs.opencv.org/2.4/index.html>
- <http://web-sisop.disca.upv.es/imd/cursosAnteriors/2k32k4/copiaTreballs/serdelal/trabajo1MD.xml>
- <https://realpython.com/blog/python/face-recognition-with-python/>
- http://docs.opencv.org/3.1.0/d6/d00/tutorial_py_root.html#gsc.tab=0
- <https://www.youtube.com/watch?v=KFC6jxBkTBQ>
- http://docs.opencv.org/2.4/modules/ml/doc/k_nearest_neighbors.html
- http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_ml/py_knn/py_knn_understanding/py_knn_understanding.html
- <https://robologs.net/2014/07/02/deteccion-de-colores-con-opencv-y-python/>
- http://docs.opencv.org/trunk/d7/d4d/tutorial_py_thresholding.html
- http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_ml/py_knn/py_knn_opencv/py_knn_opencv.html#knn-opencv