Trabajo 3:

Reconocimiento 2D

Visión por computador

Ingeniería Informática Unizar

Curso 2020/2021

Miguel Gómez 741302

Alejandro Omist 737791

Introducción

En este tercer trabajo de la asignatura de Visión por Computador se han desarrollado dos aplicaciones con OpenCV en c++, una se encarga de, a partir de un conjunto de imágenes de objetos, entrenar un clasificador para el reconocimiento de cada objeto (*aprender.cpp*) , y la otra, a partir del clasificador anterior determinar dichos objetos de una imagen (*reconocer.cpp*).

También se ha implementado un fichero de nombre *functions.h* que contiene las funciones necesarias para ambos programas.

Segmentación

El primer paso para el aprendizaje y reconocimiento de los objetos es transformar las imágenes a binario para diferenciar dicho objeto del fondo de la imagen. Para ello, una vez cargada una imagen en escala de grises, se compara cada píxel con un umbral, y se transforma a 0 si no lo supera y lo deja a 1 en caso contrario, de forma que quedaría una imagen binaria. Para realizar esta umbralización se emplearon dos métodos:

- **Método adaptativo:**

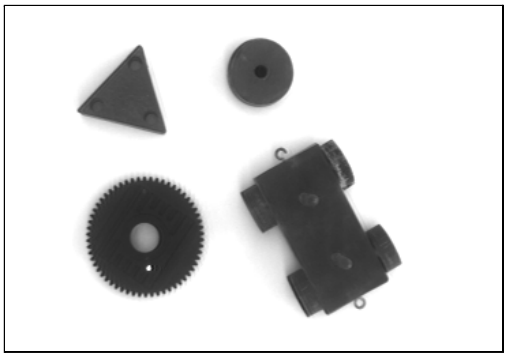
  Este método calcula un umbral personalizado para cada píxel basándose en las regiones cercanas a dicho píxel en lugar de utilizar un umbral global, ya que un mismo umbral para toda la imagen puede suponer un problema en imágenes que posean distinta iluminación. Para el caso particular de este trabajo, las imágenes con las que se han trabajado contienen la misma iluminación en todas sus zonas, por lo que no sería necesario este método.

- **Método Otsu:**

  Este método devuelve un umbral que separa los píxeles en dos clases, primer plano y fondo. El umbral se determina minimizando la variación de intensidad dentro de la clase o, de manera equivalente, maximizando la variación entre clases. Para su implementación se ha empleado la función *threshold* con el parámetro *THRESH_OTSU* proporcionada por OpenCV.

Finalmente, pese a que los métodos adaptativos funcionan razonablemente bien, se ha optado por emplear el método Otsu ya que al no requerir de parámetros es más versátil y funciona mejor con el conjunto de imágenes en general.

A continuación se muestra para una imagen, tanto su versión original como la umbralizada por el método Otsu:



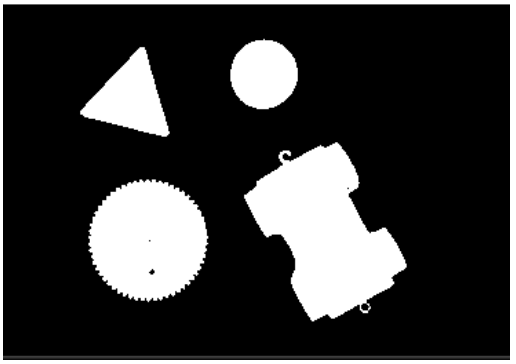


Imagen 1 y 2: imagen reco1.pgm y su umbral mediante el método Otsu.

Como se puede observar en la segunda imagen el método Otsu ignora el hueco en el interior del círculo y en la rueda entre otros detalles.

Obtención de blobs

Una vez se ha obtenido la imagen umbralizada, el siguiente paso es obtener los blobs correspondientes al objeto que aparece en la imagen. Para ello se ha empleado la función *findContours* con los parámetros *RETR_EXTERNAL* y *CHAIN_APPROX_SIMPLE* de OpenCV, ya que generaban mejores resultados. Esta función devuelve los contornos encontrados en la imagen.

De forma adicional se ha añadido un corte en el número de contornos obtenidos, siendo el máximo de 10, para evitar detecciones erróneas.

Para dibujar los blobs se ha empleado la función *drawContours* de OpenCV. A continuación se muestran algunos ejemplos:

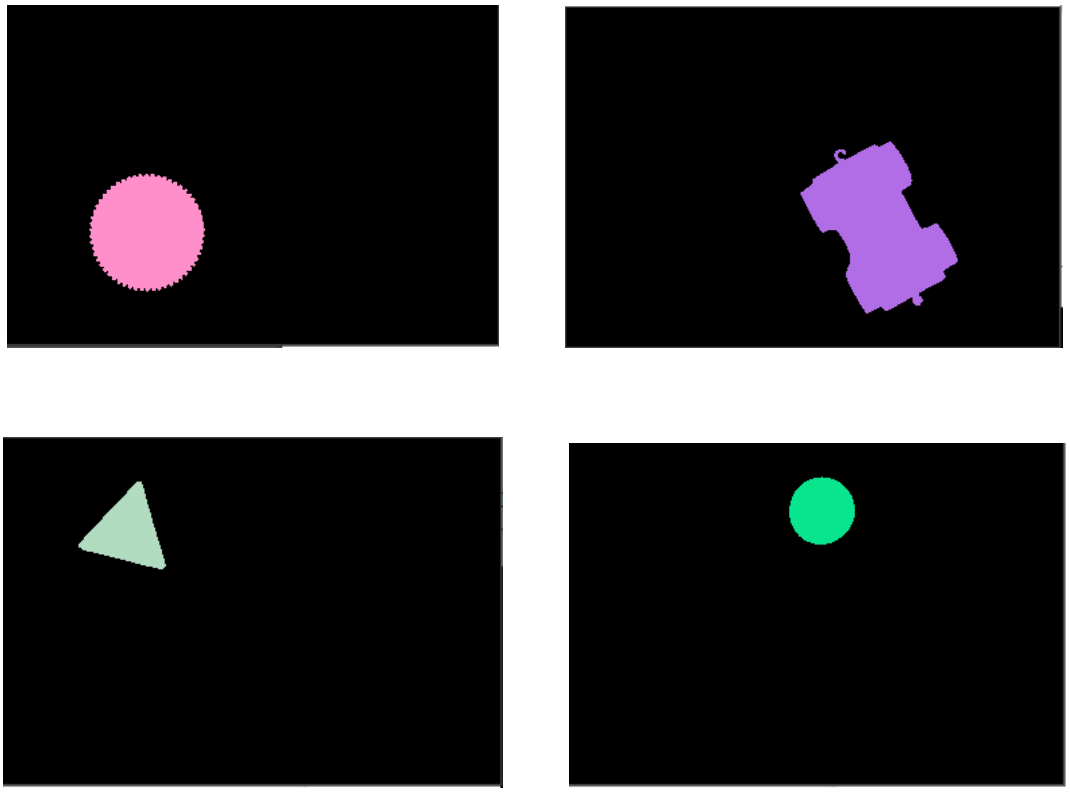


Imagen 3, 4, 5 y 6: blobs obtenidos de la imagen recon1.pgm.

Cálculo de descriptores

Una vez obtenidos los contornos se procede a calcular los descriptores de cada objeto. Estos descriptores son los valores que permiten caracterizar a un objeto, y deben ser invariantes a rotaciones y traslaciones.

Los descriptores utilizados para cada objeto son su área en píxeles, su perímetro y sus 3 primeros momentos invariantes. La forma de obtenerlos es mediante la función *moments* de OpenCV, la cual, calcula los momentos a partir de cada uno de los contornos del objeto.

El primer momento devuelto por la función es el área, calculada como el momento m00. El perímetro del contorno se puede calcular haciendo uso de la función *arcLength* de OpenCV, la cual devuelve el perímetro del contorno que se le pase como parámetro, asumiendolo como una curva cerrada. Por último, los momentos invariantes se obtienen con la función *HuMoments*, que a partir de los momentos obtenidos con *moments* devolverá los 7 momentos invariantes del objeto, de los que solo se utilizan los 3 primeros.

Reconocimiento mediante aprendizaje supervisado

Una vez que se han calculado los descriptores de las 5 imágenes de entrenamiento de cada uno de los objetos, se procede a calcular la media y la varianza de ellos.

Para la realización de este apartado se ha creado la clase Shape, que almacena y calcula la media y varianza de cada descriptor. También dispone de funciones para cargarlas y guardarlas en un fichero de formato de archivo YML. Se ha empleado este formato puesto que el código contenido en un archivo YML es fácilmente legible y útil para serializar datos.

Reconocimiento

Utilizando los descriptores calculados anteriormente durante el entrenamiento se calcula la distancia de *mahalanobis* para cada una de las clases (objetos). Para decidir a qué clase pertenece cada uno de los contornos reconocidos en la imagen se realiza un test de hipótesis, si más de una clase cumple dicho test se escoge por defecto la clase de menor distancia.

En el reconocimiento de la primera imagen no hay problemas puesto que los objetos no colisionan entre sí, pero en la segunda y la tercera si.

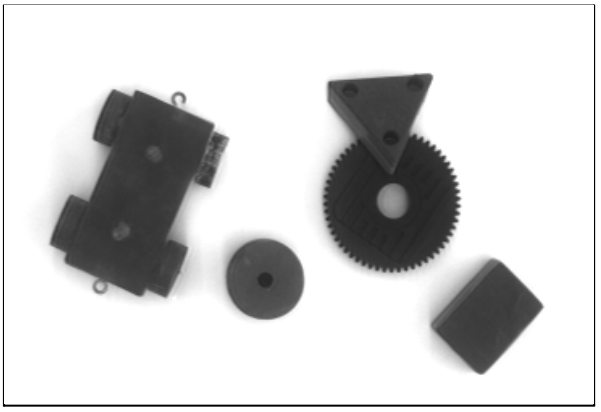


Imagen 7: imagen reco2.pgm

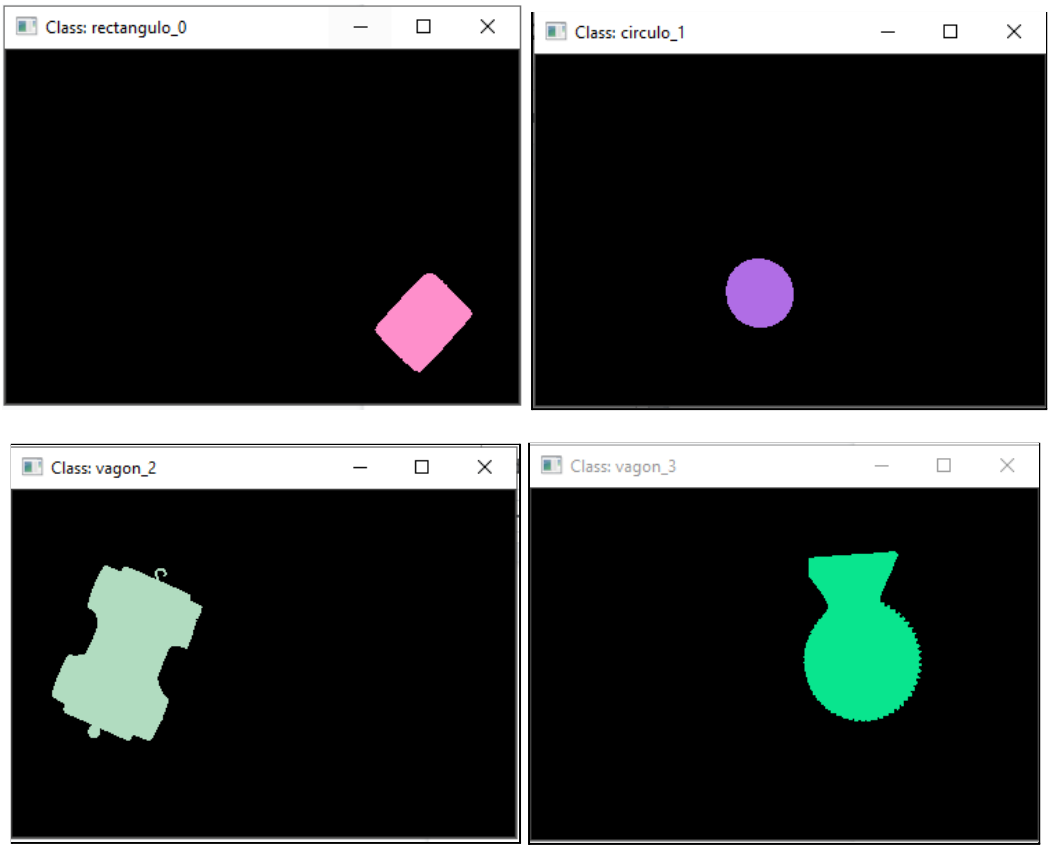


Imagen 8, 9, 10, 11: figuras reconocidas de la imagen reco2.pgm

Tal y como se puede observar en la imagen inferior derecha se produce un falso positivo, la rueda superpuesta por el triángulo la ha catalogado como un vagón.

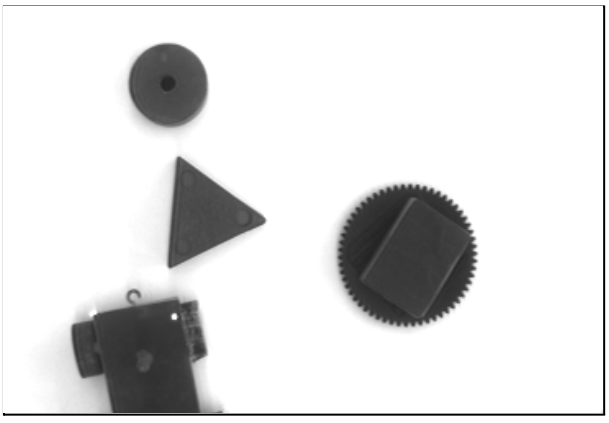


Imagen 12: imagen reco3.pgm

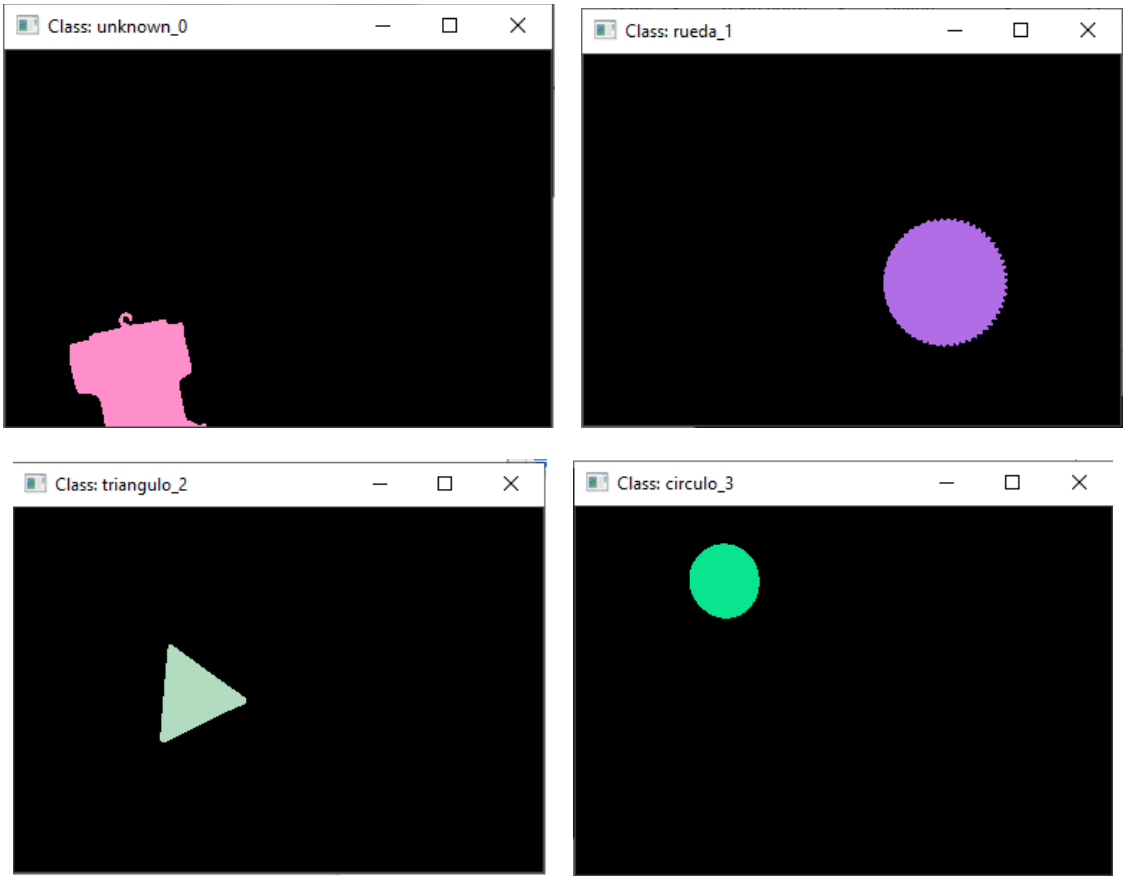


Imagen 13, 14, 15, 16: figuras reconocidas de la imagen reco3.pgm

En este caso, la figura vagón no la reconoce puesto que no se ve completo el objeto en la imagen original. Y tampoco reconoce el rectángulo superpuesto a la rueda.

- **Regularización**

Puesto que el número de muestras para el entrenamiento del clasificador es demasiado pequeño se producen falsos positivos, como ya se ha mostrado anteriormente. Para corregir esto, se puede regularizar la varianza mediante una estimación a priori.

$$\sigma^2 = (alfa * mean)^2$$

Se ha empleado un valor de alfa de 0.1.