Trabajo 3: Reconocimiento 2D

Visión por Computador

Alejandro Paricio García, 761783
Fernando Peña Bes, 756012

Universidad de Zaragoza
12 de abril de 2021

Índice

1. Introducción

En esta práctica se pretende desarrollar un programa capaz de reconocer objetos planos a partir de muestras previamente aprendidas, utilizando las funciones proporcionadas por OpenCV. Para ello, se ha recurrido a la segmentación de las imágenes de objetos, para posteriormente extraer sus contornos y los descriptores necesarios. Finalmente, se ha implementado el modelo de reconocimiento, que consiste en el cálculo de la media y la varianza de los descriptores, permitiéndose así la posterior clasificación de los nuevos objetos en una, varias o ninguna de las clases.

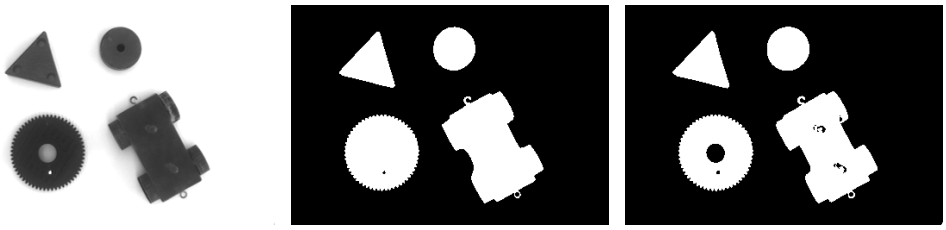
2. Segmentación de las imágenes

En esta primera fase de la práctica se busca emplear un algoritmo clásico de segmentación para transformar una imagen en una máscara binaria, para poder efectuar el reconocimiento de objetos. Los métodos a comparar son el *Adaptative Thresholding* y la *OTSU Binarization*.

El método de OTSU es un tipo de umbralización global en el que no es necesario determinar manualmente el valor que decide qué píxeles serán blancos y cuáles negros. El propio algoritmo determina ese valor automáticamente, intentando minimizar la varianza intra-clase de la intensidad de los píxeles (blancos o negros, dos clases). Por otro lado, el método adaptativo no utiliza un único valor de umbral para toda la imagen, sino que para cada píxel elige el mejor umbral de forma local en una región a su alrededor [1].

Cada uno de ellos tiene sus ventajas e inconvenientes. En el método OTSU no es necesario elegir manualmente valor alguno, mientras que en el *Adaptative Thresholding* hay que ajustar diversos parámetros, como el tamaño de la celda alrededor de cada píxel a considerar. Por otro lado, los métodos globales son muy susceptibles a cambios de iluminación, ya que si el umbral idóneo varía en función de la región de la imagen, pero se elige un umbral único, el resultado se puede ver muy perjudicado, mientras que con el otro método bien ajustado esto no sucede. Por tanto, cada uno de ellos tiene sus ventajas e inconvenientes, siendo necesario atender a qué tipo de imágenes se le aplica la umbralización a la hora de elegir.

Las imágenes resultantes de aplicar ambos métodos mediante las funciones de OpenCV *threshold()* y *adaptiveThreshold()*.



(a) Imagen original (b) Método OTSU (c) Adaptative thresholding

Figura 1: Comparación de los distintos métodos.

Como puede observarse, en la imagen original no hay grandes cambios de iluminación, a excepción de en el dentro de la rueda. Como consecuencia directa, el método adaptativo es el único que diferencia esta zona, aunque también identifica erróneamente las sombras de las chimeneas del vagón. El método OTSU no identifica ese centro, pero define muy

bien cada una de las 4 formas, sin áreas mal definidas como ocurría en el vagón con el anterior. Al no haber cambios de iluminación fuertes, se ha decidido continuar con el método OTSU, ya que la ventaja aportada por el adaptativo no va a resultar demasiado relevante.

3. Cálculo de componentes conexas

El siguiente paso en el reconocimiento de objetos es la obtención de las componentes conexas. Para ello, se ha utilizado la función de OpenCV *connectedComponents()*, que permite llevarlo a cabo. En las imágenes con las que se va a trabajar no hay figuras elongadas, por lo que se ha decidido utilizar 4-conectividad. El resultado obtenido es:

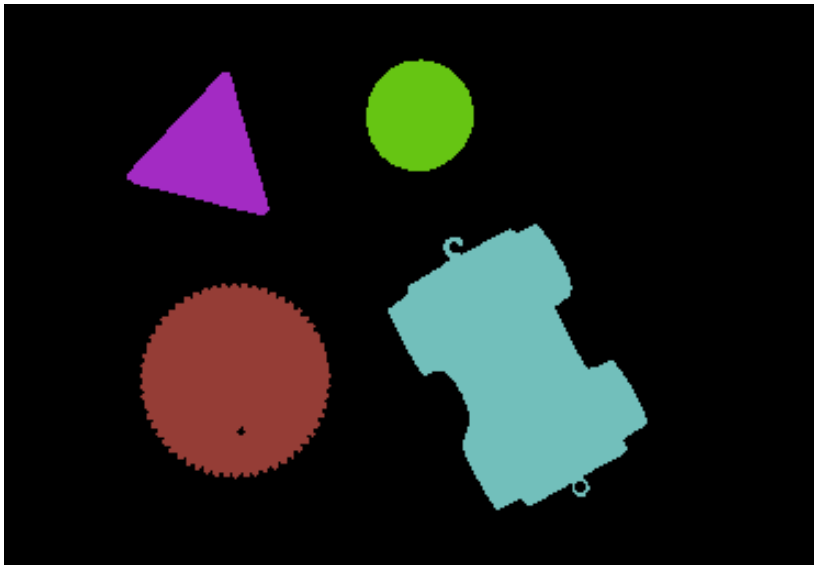


Figura 2: Componentes conexas

Se ha decidido eliminar un pequeño borde negro de 1 píxel de altura en la parte superior de la imagen *reco1.pgm*, que no formaba parte del fondo blanco y daba lugar a una componente no deseada.

4. Cálculo de descriptores

Acto seguido han de obtenerse los descriptores de cada uno de los objetos para el posterior aprendizaje supervisado. Para evitar la implementación desde cero de las funciones necesarias, se ha continuado utilizando que proporciona OpenCV. Concretamente, han de obtenerse los tres primeros momentos de Hu, el perímetro y el área de cada uno de ellos, para lo que se emplean las funciones *HuMoments()*, *arcLength()* y *contourArea()*. No obstante, primero fue necesario extraer los contornos de las imágenes utilizando *findContours()*, ya que estos métodos trabajan sobre contornos.

La función *HuMoments()* calcula los momentos de Hu utilizando los momentos normalizados, por lo que los valores son invariantes a translación, rotación y escala [2].

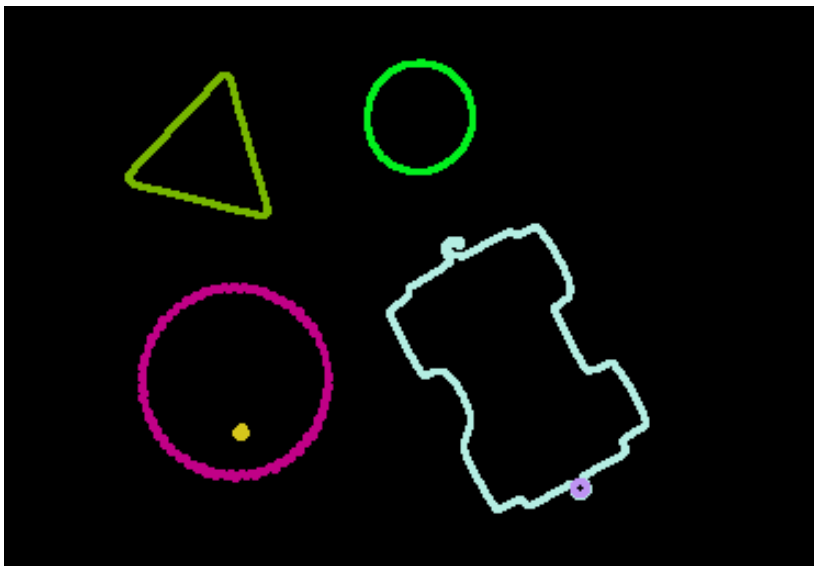


Figura 3: Contornos obtenidos

5. Reconocimiento de objetos mediante aprendizaje supervisado

En este apartado se pide implementar un programa capaz de reconocer los objetos de una imagen. Para ello, se debe aprender la media y la varianza de cada descriptor para cada tipo de objeto (vagón, rectángulo, triángulo, círculo o rueda), y utilizar el método de distancias mínimas de Mahalanobis para clasificar los nuevos objetos.

Como datos de entrenamiento se dispone de 5 muestras de cada tipo de objeto. Cada imagen de entrenamiento contiene un único objeto, por lo que hay 25 imágenes en total.

Se ha creado un programa llamado `aprender` que va toma como entrada una imagen de entrenamiento y el tipo de objeto que aparece en ella. Calcula los descriptores del objeto de tal y como se ha explicado en el Apartado 4 y recalcula la media y la varianza del de los descriptores del objeto utilizando la información de las muestras que ya se han procesado. El programa va almacenando en el fichero `muestras.txt` todas las muestras que ha aprendido hasta en momento, y en `objetos.txt` las estadísticas de los descriptores de cada tipo de objeto.

De forma matemática, si llamamos $\mathbf{X}$ a la matriz que contiene los m descriptores para cada muestra i, e $\mathbf{Y}$ al vector con la clase de cada muestra:

$$\mathbf{X} = \begin{bmatrix} — & \mathbf{x}^{(i)} & — \\ — & \mathbf{x}^{(i+1)} & — \\ & \vdots & \\ — & \mathbf{x}^{(N)} & — \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

$$\mathbf{x}^{(i)} = [\text{perímetro}^{(i)}, \text{área}^{(i)}, \phi_1^{(i)}, \phi_2^{(i)}, \phi_3^{(i)}]$$

,

para cada atributo k de cada clase j, el programa aprende su media μ_{jk} y su varianza σ^2_{jk} de la siguiente manera:

$$N_j = \sum_i [y^{(i)} = j]$$
$$\mu_{jk} = \frac{1}{N_j} \sum_{y^{(i)}=j} x_k^{(i)}$$
$$\sigma^2_{jk} = \frac{1}{N_j - 1} \sum_{y^{(i)}=j} (x_k^{(i)} - \mu_{jk})^2$$

6. Reconocimiento

Para reconocer los objetos de una imagen se ha creado el programa `reconocer`. Las imágenes con las que se trabaja en este apartado pueden contener varios objetos, por lo que primero hace separarlos obteniendo las componentes conexas, de forma que se pueda clasificar cada objeto de manera separada. Esto se realiza como en el Apartado 3.

Para reconocer la clase de un nuevo objeto, primero se obtienen sus descriptores de la misma forma que en el entrenamiento. A continuación, se calcula la distancia de Mahalanobis de estos descriptores a cada clase j:

$$D^2(\mathbf{x}, \omega_j) = \sum_{k=1}^{m} \frac{(x_k - \mu_{jk})^2}{\sigma^2_{jk}}$$

El objeto es más probable que pertenezca a la clase con la que tiene menor distancia de Mahalanobis. Sin embargo, es posible que el objeto no pertenezca a una clase conocida y sería incorrecto clasificarlo como perteneciente a la clase más cercana. Se conoce que la distancia de Mahalanobis sigue una distribución chi cuadrado con tantos grados de libertad como número de descriptores (m), así que podemos plantear un test de hipótesis sobre la pertenencia de la muestra a cada clase, $H_0 : \{\mathbf{x}$ viene de $\omega_j\}$. La probabilidad de que esta hipótesis sea cierta, a partir de la distancia de Mahalanobis es $Pr\left\{D^2(\mathbf{x}, \omega_j) < \chi^2_\alpha(m)\right\} = 1 - \alpha$, donde α es el nivel de confianza.

En la práctica se ha elegido como nivel de confianza $\alpha = 0{,}05$. Teniendo en cuenta que tenemos 5 descriptores y consultando los valores de la distribución χ^2, se obtiene que $\chi^2_{0{,}05}(5) = 11{,}07$. Por tanto si $D^2(\mathbf{x}, \omega_j) < 11{,}07$, la muestra $\mathbf{x}$ pertenecerá a la clase j con probabilidad 0,95.

En la Figura 4 se muestran las componentes conexas de la imagen `reco1.txt`, seguidas por el reconocimiento que realiza el programa.

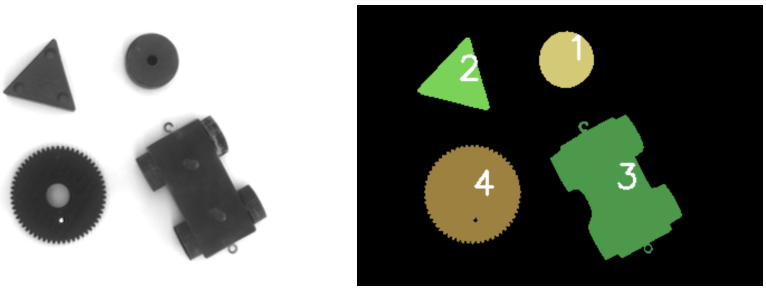


Figura 4: Componentes conexas de `reco1.pgm`

A continuación se muestra la salida del clasificador:

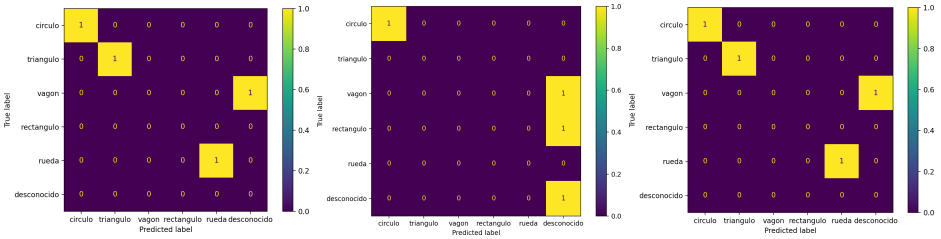
```
Objeto 1
Descriptores: 162.024 1869 0.159312 3.06047e-05 1.51279e-06
Distancias de Mahalanobis:
  * circulo: 1.66307
  * triangulo: 2683.21
  * rectangulo: 3688.73
  * rueda: 5191.33
  * vagon: 22872.2
Predicción 1: circulo

Objeto 2
Descriptores: 209.723 2096.5 0.190799 1.50345e-05 0.00425838
Distancias de Mahalanobis:
  * triangulo: 1.48022
  * circulo: 3.83451e+06
  * rectangulo: 1.61087e+07
  * vagon: 2.20185e+07
  * rueda: 9.18957e+08
Predicción 2: triangulo

Objeto 3
Descriptores: 477.085 8111.5 0.195812 0.00754137 1.10776e-06
Distancias de Mahalanobis:
  * vagon: 13.1692
  * circulo: 151181
  * rectangulo: 192643
  * triangulo: 281023
  * rueda: 3.28544e+06
Predicción 3: Objeto desconocido

Objeto 4
Descriptores: 380.818 5702 0.159344 4.70195e-06 1.83017e-07
Distancias de Mahalanobis:
  * rueda: 7.98115
  * vagon: 3290.86
  * triangulo: 32973.2
  * circulo: 41553.1
  * rectangulo: 61406.7
Predicción 4: rueda
```

Se puede ver que la distancia de los descriptores de cada objeto a la clase correcta es claramente menor que a las demás.

7. Regularización

El sistema se ha probado con las imágenes `reco1.pgm`, `reco2.pgm` y `reco3.pgm`, y los resultados se han recogido en las matrices de confusión que se pueden ver en la Figura 5.

Ningún objeto se ha clasificado con una clase incorrecta, pero en 5 ocasiones no se ha encontrado la clase del objeto. Los descriptores utilizados en el sistema son invariantes a translación y rotación ya que, aunque los momentos de Hu sean invariantes a translación



(a) `reco1.pgm` (b) `reco2.pgm` (c) `reco3.pgm`

Figura 5: Matrices de confusión utilizando varianza clásica

rotación y escala, el perímetro y área sólo son invariantes a translación y rotación. Sin embargo esto no es un problema, ya que en todas las imágenes tienen la misma escala.

En el clasificador aparecen problemas derivados de diferenciar los objetos por las componentes conexas que los representan. Si los objetos están superpuestos, como en las Figuras 6 (triángulo y rueda) y 7 (rectángulo), o tienen partes fuera de la imagen como el vagón de la Figura 7 el sistema no puede calcular sus descriptores correctamente y falla la clasificación.

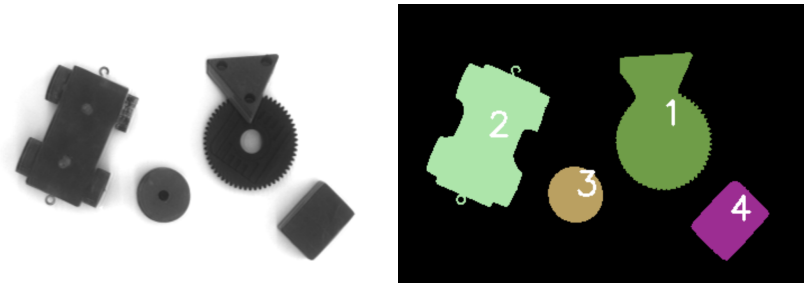


Figura 6: Componentes conexas de `reco2.pgm`

Figura 7: Componentes conexas de `reco3.pgm`

Sin embargo, hay casos en los que aunque se haya conseguido extraer correctamente la componente conexa que representa a un objeto, la distancia de Mahalanobis con la clase correcta no es lo suficiente pequeña para pasar el test de hipótesis. Esto sucede sobre todo con formas complicadas como en el vagón de `reco1.pgm`.

Se debe a que el número muestras de entrenamiento es demasiado pequeño, 5 para cada clase, lo que puede provocar que se subestime la varianza cuando se calcula de la

manera clásica. Como la varianza es optimista, se descartan rápidamente muestras que se desvían un poco de la media de la distribución.

Si se tuvieran más muestras se podría seguir entrenando el modelo para ajustar la varianza a un valor más acorde con la realidad, pero en este caso no es posible. Una solución es utilizar un estimador a priori de la varianza, que dependa de observaciones a simple vista de los datos de entrenamiento. El objetivo es incrementar el valor de la varianza, para que las distancia de Mahalanobis en la clasificación se reduzcan. El estimador a priori, σ_{0N}^2, va a ser el cuadrado de un cierto porcentaje de la media del descriptor correspondiente. Utilizando este estimador, la varianza se calcula de la siguiente manera

$$\sigma_N^2 = \frac{\sigma_{0N}^2 + \sum_{i=1}^{N}(x_i - \mu)^2}{N} = \frac{\sigma_{0N}^2}{N} + \frac{N-1}{N}\sigma^2$$

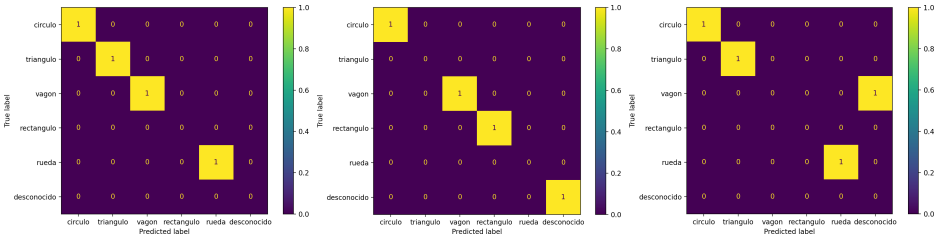
Al aumentar el porcentaje de la media, la varianza también aumenta y las distancias de Mahalanobis se reducen. Hay que elegir un porcentaje suficiente grande para que los objetos problemáticos pasen el test de hipótesis de la clase correcta, pero no lo demasiado para que se produzcan dudas sobre la clase porque el mismo objeto pasa varios tests de hipótesis. Para esta práctica se ha comprobado experimentalmente que 5 % es un valor adecuado.

Utilizando este estimador se consigue reconocer correctamente el vagón de `reco1.pgm` como se puede ver en la siguiente salida del clasificador:

```
Objeto 1
Descriptores: 162.024 1869 0.159312 3.06047e-05 1.51279e-06
Distancias de Mahalanobis:
  * circulo: 0.697235
  * rectangulo: 298.684
  * triangulo: 343.474
  * rueda: 1003.86
  * vagon: 1765.52
Predicción 1: circulo

Objeto 2
Descriptores: 209.723 2096.5 0.190799 1.50345e-05 0.00425838
Distancias de Mahalanobis:
  * triangulo: 1.23234
  * circulo: 4.75636e+06
  * rectangulo: 2.01128e+07
  * vagon: 2.73752e+07
  * rueda: 1.14584e+09
Predicción 2: triangulo

Objeto 3
Descriptores: 477.085 8111.5 0.195812 0.00754137 1.10776e-06
Distancias de Mahalanobis:
  * vagon: 3.80042
  * rectangulo: 10758
  * circulo: 47686.2
  * triangulo: 260434
  * rueda: 1.61387e+06
Predicción 3: vagon
```

```
Objeto 4
Descriptores: 380.818 5702 0.159344 4.70195e-06 1.83017e-07
Distancias de Mahalanobis:
  * rueda: 4.18894
  * vagon: 330.627
  * rectangulo: 3561.46
  * triangulo: 4245.71
  * circulo: 8883.39
Predicción 4: rueda
```

En la Figura 8 se encuentran las matrices de confusión al utilizar este estimador de la varianza. Todos los objetos se han reconocido correctamente, menos los que aparecen superpuestos o tienen partes fuera de la imagen.



(a) `reco1.pgm` (b) `reco2.pgm` (c) `reco3.pgm`

Figura 8: Matrices de confusión utilizando estimación de la varianza

Referencias

[1] Thresholding opencv. https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html. Consultado el 28/03/2021.

[2] Opencv hu moments. https://docs.opencv.org/master/d3/dc0/group__imgproc__shape.html#gab001db45c1f1af6cbdbe64df04c4e944. Consultado el 28/03/2021.