Visión por computadora

-

Trabajo 2. Reconocimiento 2D

Grado en ingeniería informática
Asignatura: Visión por computadora
Autores: Navarro Torres, Agustín (587570)
Reyes Apaestegui, Sergio Adrian (642535)

Umbralización

Otsu vs adaptativo:

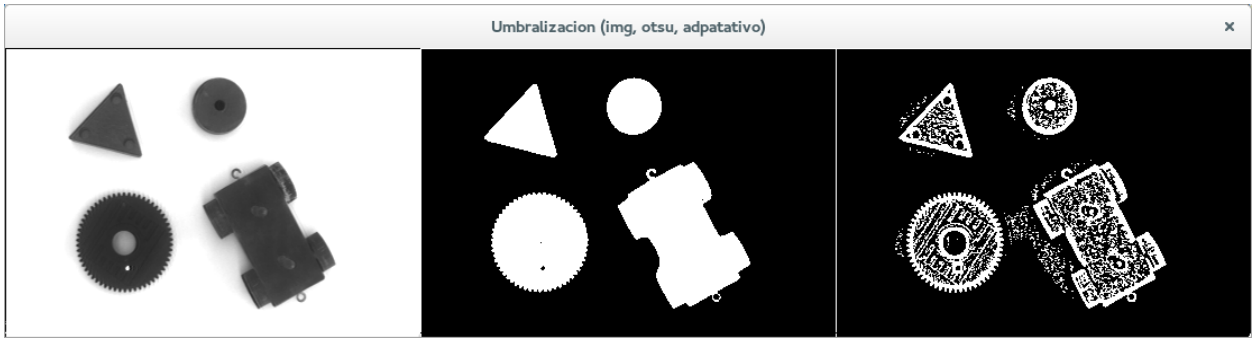


Figura: Imagen original, otsu binaria y adaptativa binaria respectivamente

Debido a que la imagen no sufre de distintos cambios de luz el método otsu genera un mejor resultado, ya que el adaptativo intenta aplicar el threshold por distintas zonas de la imagen, el cual es mejor cuando varía la intensidad de luz en la escena.

Blobs

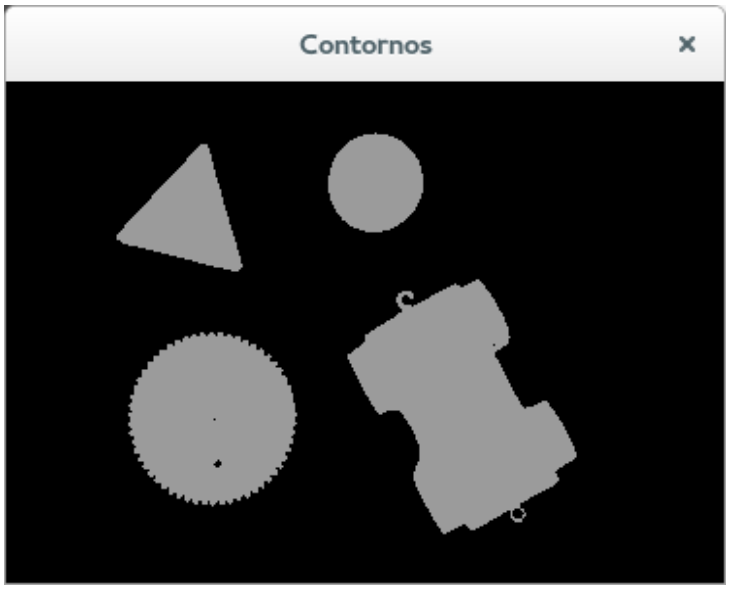


Figura: Contornos dibujados

Usando el método cv2.findContours a partir de la imagen binaria se obtiene los contornos y una matriz con la jerarquía de estos, esta matriz indica los contornos hijos, para los agujeros.

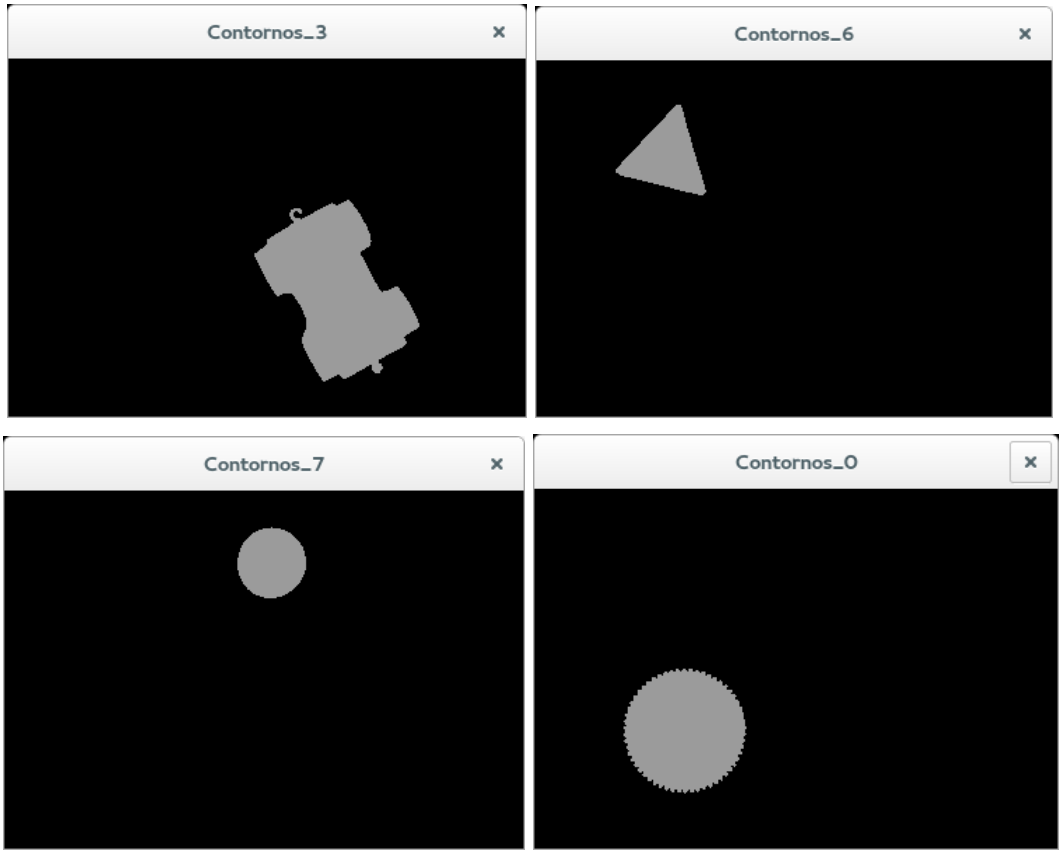


Figura: Blobs de la imagen anterior

Descriptores

Usando funciones de opencv se puede calcular los descriptores que se necesitan de la siguiente forma:

- momentos - cv2.moments()
- área - el momento 00, o usando cv2.contourArea()
- perímetro - cv2.arcLength()
- momentos invariantes - cv2.HuMoments()

```
Mom:
{'m00': 5694.0,
 'm01': 976837.0,
 'm02': 170204134.3333333,
 'm03': 30099523804.300003,
 'm10': 602851.1666666666,
 'm11': 103423166.66666666,
 'm12': 18020403430.716667,
 'm20': 66370922.166666664,
 'm21': 11386197321.35,
 'm30': 7565911944.650001,
 'mu02': 2622377.296276778,
 'mu03': 302463.99502563477,
 'mu11': 735.1434551030397,
```

```
'mu12': -179144.80046105385,
'mu20': 2544169.5935438946,
'mu21': -254884.6035132408,
'mu30': 176244.57112503052,
'nu02': 0.08088355862969955,
'nu03': 0.00012363175296354101,
'nu11': 2.26744713037627e-05,
'nu12': -7.32251973112593e-05,
'nu20': 0.07847135146245841,
'nu21': -0.00010418374039226837,
'nu30': 7.203973245360422e-05}
Area: 5694.0
Perimetro: 382.646749377
Hu moments: [[ 1.59354910e-01]
[ 5.82079994e-06]
[ 2.75353417e-07]
[ 3.79630520e-10]
[ 3.56458529e-18]
[ 9.06876548e-13]
[ 1.53586930e-18]]
```

Realización de Aprender

Ya que para el reconocimiento se usa la distancia de mahalanobis, se guardan todos los datos de entrenamiento relevantes: la media y la varianza de los descriptores en una matriz, usando un diccionario de python donde los descriptores son las columnas y las filas son las clases. Para el cálculo de la media y varianza se han utilizado funciones de numpy, np.mean() y np.var(). A la hora de guardar los datos obtenidos en un fichero se utiliza la librería *pickle* de python que permite la serialización de objetos, en nuestro caso un diccionario, en un fichero.

Reconocimiento

Como el diccionario guarda una matriz de varianzas y medias de cada clase la distancia de mahalanobis se calcula con una operación matricial:

```
mahal = ((observado - descriptores['means']) ** 2) / (descriptores['vars'])
```

Como mahal contiene la distancia de mahalanobis del dato observado con respecto a cada clase en un array, aplicando el test de chi-square se puede clasificar el objeto en base a los datos que se han aprendido a priori.

Regularización

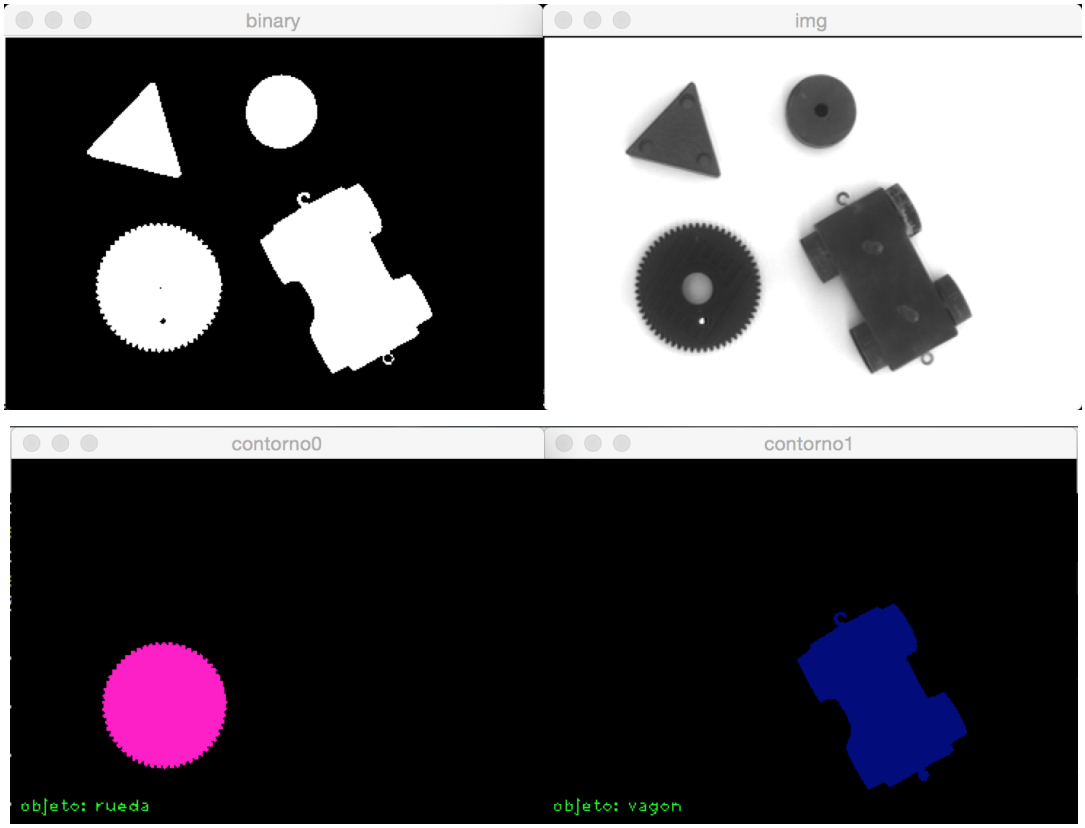
Para compensar el pequeño número de muestras, de las cuales disponemos, se ha realizado un nuevo reconocer que compensa este pequeño número de muestras, debido a que el inicial tiende a subestimar la varianza, y se ha realizado con distintas pruebas su funcionamiento.

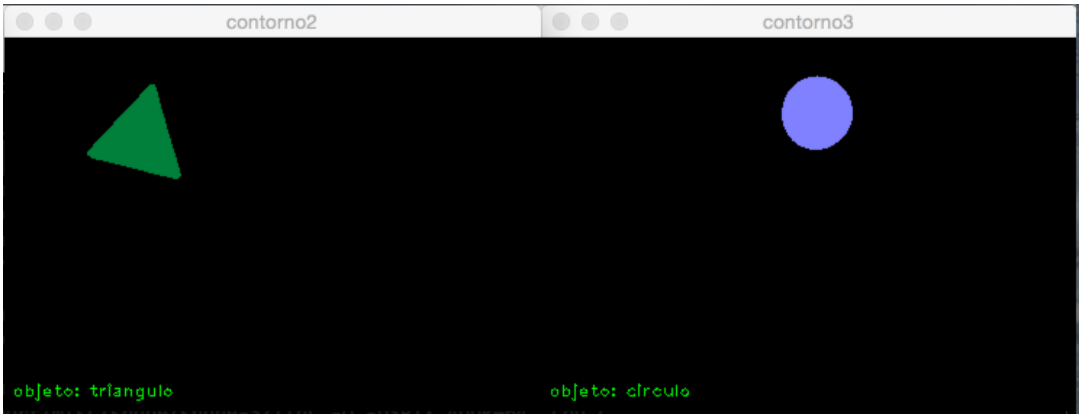
```
2.  ((observado - descriptores['means']) ** 2) /
    (descriptores['vars']*(np.array(descriptores['N'])-1)/descriptores['N']+(0.03*descriptores['means'])**2)
```

Mahal contiene la distancia de mahalonobis regularizada como se ha visto en clase.

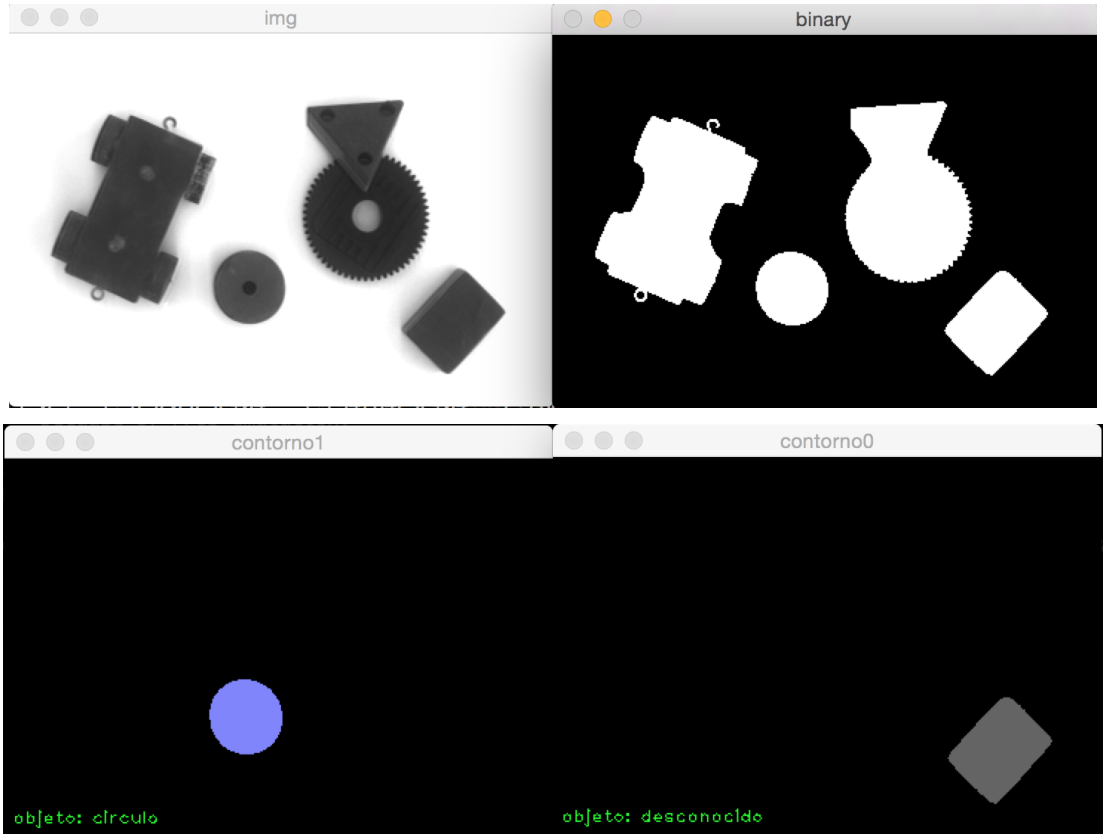
Pruebas

Reconocimiento de la imagen *reco1.pgm*:

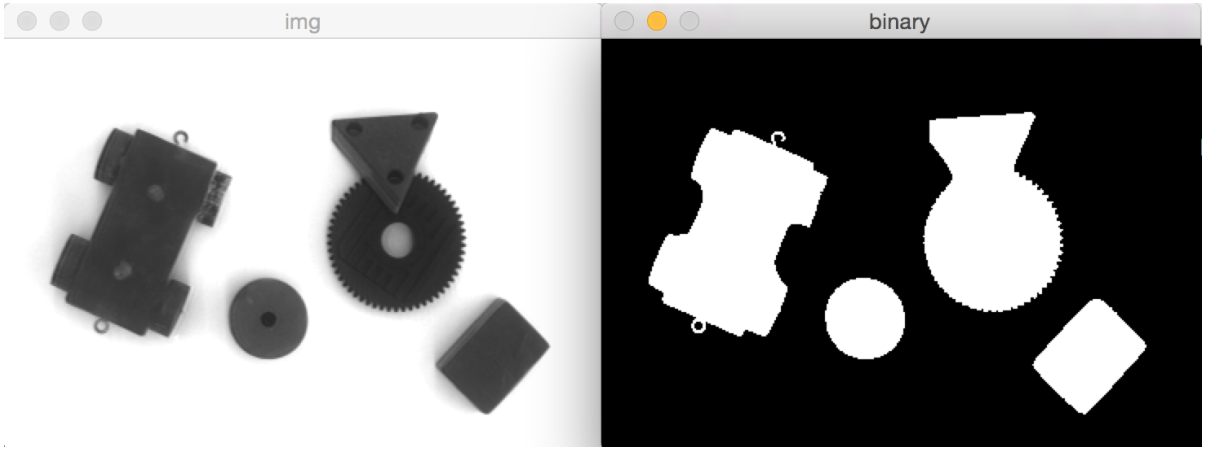


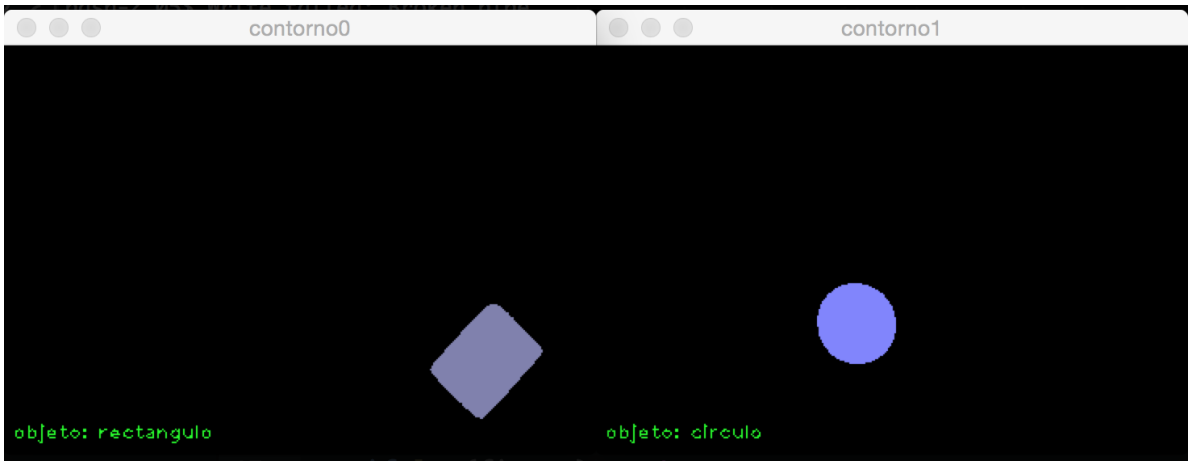


Reconocimiento de la imagen *reco2.pgm* sin regularización:

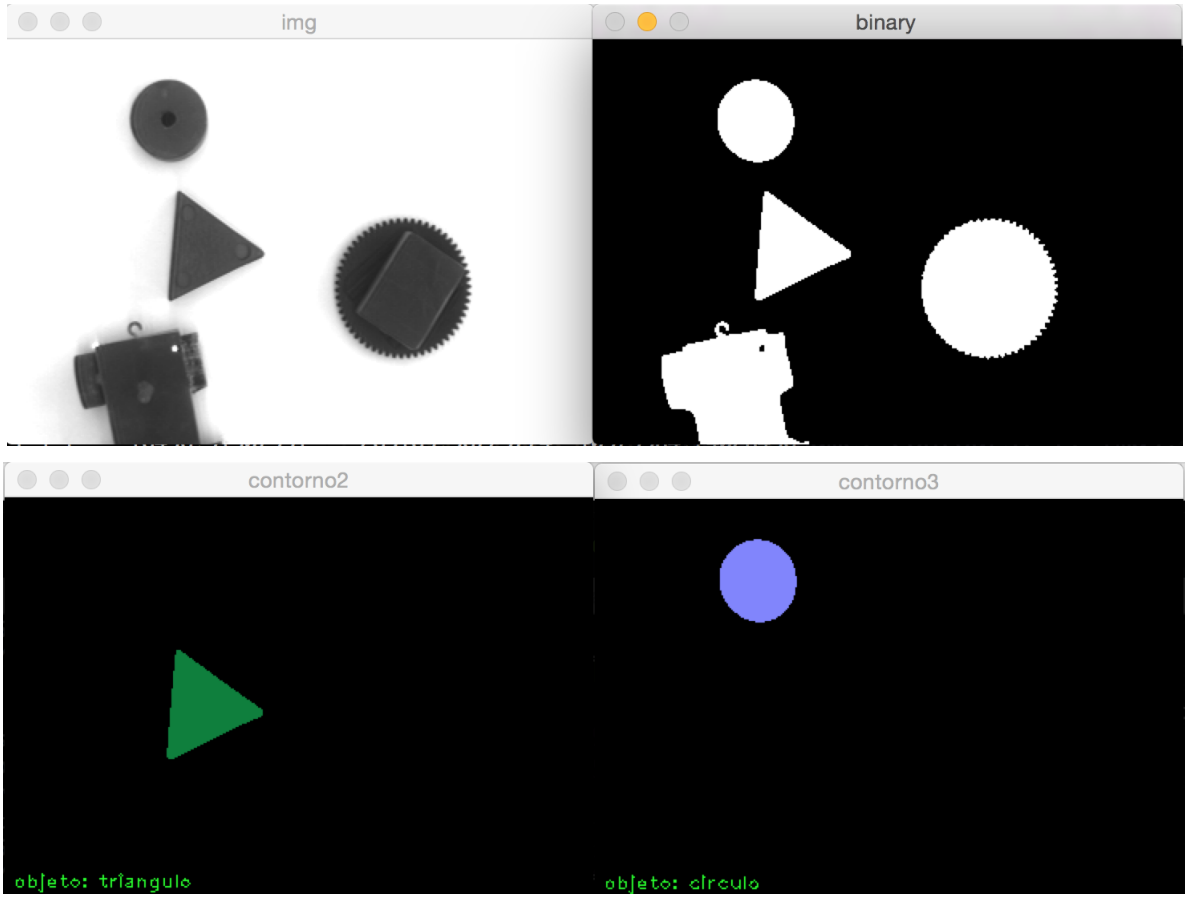


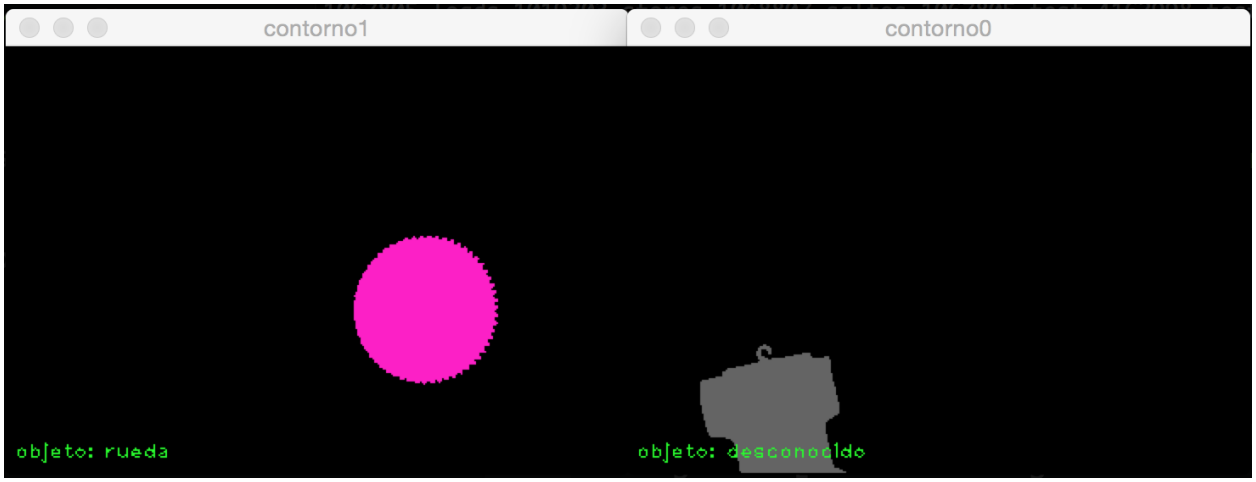
Reconocimiento de la imagen *reco2.pgm* con regularización:





Reconocimiento de la imagen *reco3.pgm*:




contorno1
contorno0
objeto: rueda
objeto: desconocido