Práctica 1: Visión Artificial

Manuel Torres Acosta

Índice

Objetivo de la práctica

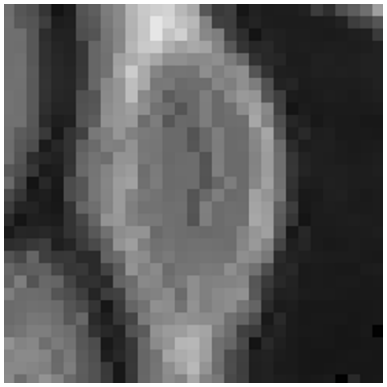


Figura 1: Cond. lagrimal

Figura 2: Coma

La práctica actual consiste en detectar los contornos en dos imágenes diferentes. Una imagen corresponde con el conducto lagrimal visualizado mediante una tomografía axial computerizada y la otra es una imagen sintética para probar la detección de contornos.

La detección de contornos es una de las tareas básicas que se realizan en la visión artificial. Consiste en delimitar las fronteras entre los diferentes objetos que aparecen en una imagen. Pese a que para los seres humanos y otros seres vivos la tarea resulta trivial, fue uno de los primeros rompecabezas a la hora de diseñar sistemas de visión artificial.

Hay que tener en cuenta además que los seres humanos percibimos la realidad de forma tridimensional ya que combinamos la información que proviene de nuestros dos ojos, lo cual nos puede dar pistas en muchas situaciones para distinguir unos objetos de otros.

En la visión artificial generalmente contamos únicamente con una imagen estrictamente bidimensional, de modo que inferir las delimitaciones entre objetos puede resultar más difícil.

Conducto lacrimal

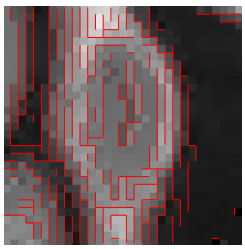
```
#Librerias cargadas: magick, image.ContourDetector

#Leemos la imagen
canal <- image_read("canal.png")

#Cargamos los datos de la imagen como una matriz
#que contiene los niveles de gris de cada pixel
PixMat <- image_data(canal,
                     channels = "gray")
PixMat <- as.integer(PixMat)
PixMat <- drop(PixMat)

#Aplicamos la deteccion de contornos
Contour <- image_contour_detector(PixMat)

#Mostramos la imagen con los contornos
plot(canal)
plot(Contour,
     add = TRUE,
     col = "red")
```

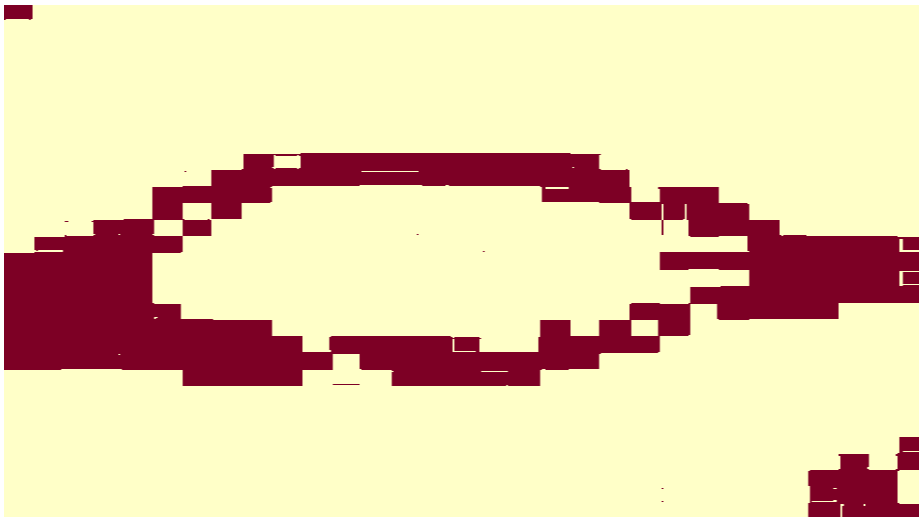


Vemos que no funciona demasiado bien si aplicamos el algoritmo de detección de contornos (Grompone von Gioi & Randall, 2016) sobre la imagen original ya que detecta varios contornos en los gradientes. Aplicamos primero una umbralización para eliminar dichos gradientes.

Definimos un punto critico para el valor de gris de cada píxel. A todos los valores por encima del punto crítico se les asignara un único valor, y a los que quedan por debajo se les asigna otro.

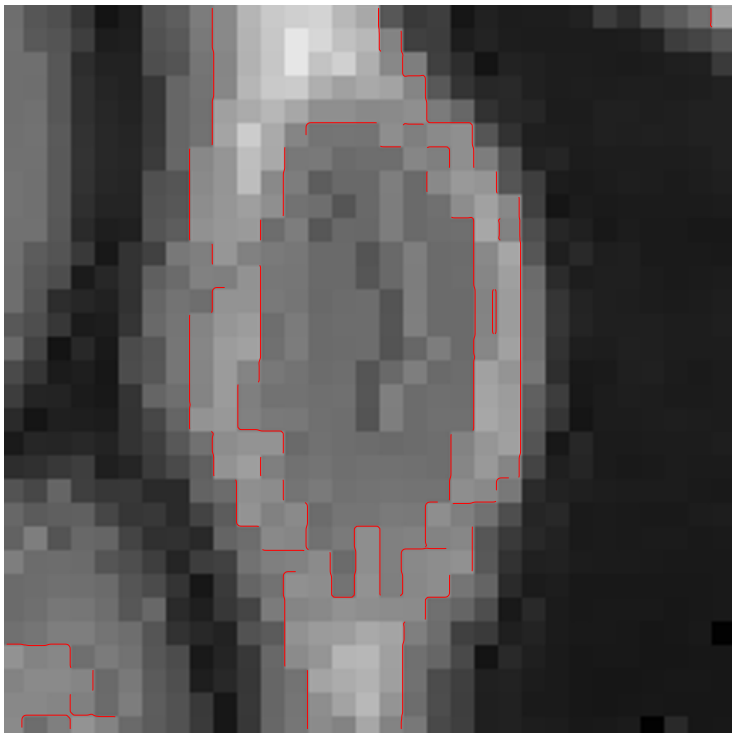
```
#Umbralizacion
PixMat <- ifelse(PixMat > 130, 255, 0)

#Mostramos la imagen umbralizada
image(PixMat,
      useRaster = TRUE,
      axes = FALSE)
```



En la imagen umbralizada se aprecia mucho mejor el conducto como tal (la imagen aparece rotada). Quitando algunos artefactos en las esquinas ya tendríamos el conducto aislado. A continuación, aplicamos el algoritmo de detección de contornos sobre la imagen umbralizada para ver si mejora el resultado.

```
Contour <- image_contour_detector(PixMat)
plot(canal)
plot(Contour, add = TRUE, col = "red")
```



La umbralización anula gran parte de los gradientes y permite ver el conducto de forma más detallada, aunque detecta también un segundo contorno en la esquina inferior izquierda. Veamos cómo se comporta el algoritmo en la segunda imagen.

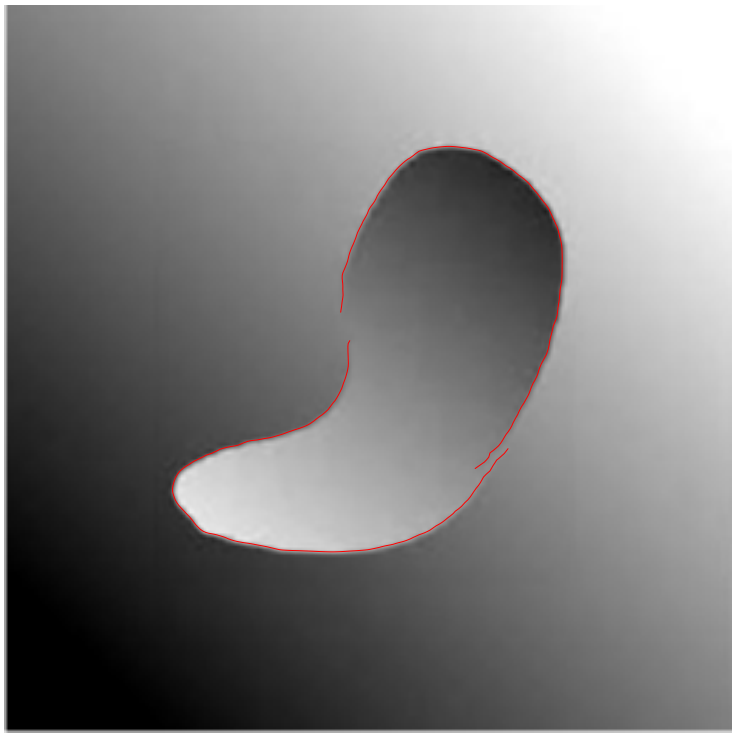
Coma

```
coma <- image_read("coma.png")

PixMat <- image_data(coma, channels = "gray")
PixMat <- as.integer(PixMat)
PixMat <- drop(PixMat)

Contour <- image_contour_detector(PixMat)

plot(coma)
plot(Contour, add = TRUE, col = "red")
```



En éste caso, el algoritmo detecta el contorno sin problema.

Ampliación

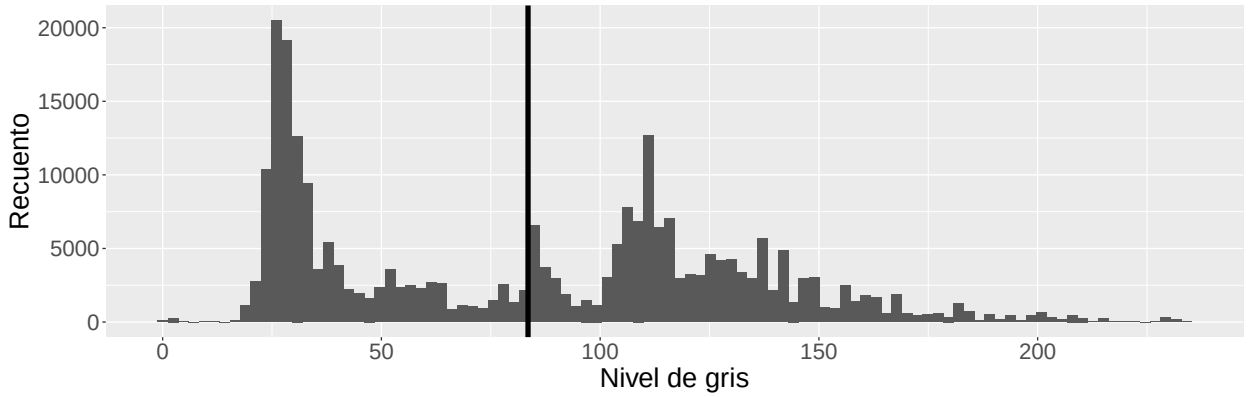
Histograma de la imagen

También podemos realizar otras operaciones con imágenes en R. Por ejemplo, podemos crear el histograma de la imagen del conducto lacrimal.

```
library(ggplot2)
HistData <- image_data(canal,
                       channels = "gray")
HistData <- as.integer(HistData)

#Convertimos la matriz a un vector y despues a dataframe
#para poder representarla con ggplot
HistData <- as.vector(HistData)
HistData <- data.frame(HistData)
colnames(HistData) <- "NivelGris"

#Mostramos el grafico
ggplot(HistData, aes(NivelGris)) +
  geom_histogram(bins = 100) +
  geom_vline(xintercept = mean(HistData$NivelGris), size = 2) +
  xlab("Nivel de gris") +
  ylab("Recuento") +
  theme(text = element_text(size = 22))
```



La barra negra vertical representa la media. Vemos que la distribución del gris en la imagen es bimodal, con un pico en torno al valor 25 y otro cercano al 112.

Filtros paso alto y paso bajo

También podemos utilizar el paquete EBImage para aplicar filtros de paso bajo/alto sobre la imagen.

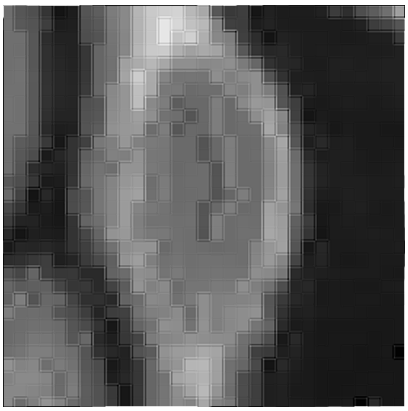
```
canal2 <- readImage("canal.png")

#Creamos el filtro paso bajo
lowPass <- makeBrush(81, shape = "disc",
                     step = FALSE) ^ 2
lowPass <- lowPass / sum(lowPass)

#Aplicamos el filtro sobre la imagen
ImageLow <- filter2(canal2, lowPass)
display(ImageLow)
```

```
#Creamos el filtro paso alto
highPass = matrix(1, nc = 3, nr = 3)
highPass[2,2] = -7

#Aplicamos el filtro sobre la imagen
ImageHigh <- filter2(canal2, highPass)
display(ImageHigh)
```



Umbralización automática

En el ejercicio de la práctica, especificamos un valor para el umbral de forma manual. La librería EBImage nos permite hacerlo de forma automática gracias al método de Otsu (1979). Dicho método asume que el histograma de la imagen sigue una distribución bimodal, como es nuestro caso. Para la imagen del conducto lacrimal obtendríamos el siguiente resultado.

```
#Creamos una nueva imagen que solo contiene el canal gris
canalGrey <- canal2
colorMode(canalGrey) <- Grayscale
canalGrey <- getFrame(canalGrey, 1)

#Calculamos el umbral
threshold <- otsu(canalGrey)

#Mostramos la imagen umbralizada
canalThreshold <- canalGrey > threshold
display(canalThreshold)
```

Vemos que en este caso la imagen umbralizada captaría bien el contorno del canal, pero no el canal como tal. También podemos especificar diferentes valores para el umbral en cada punto de la imagen en función de los valores de los alrededores.

```
#Especificamos las dimensiones de la region a analizar
Dim <- 80
canalThreshold <- thresh(canalGrey,
                         w = Dim,
                         h = Dim,
                         offset = 0.05)
display(canalThreshold)
```

Ahora recuperamos el interior del canal. Aunque la imagen contiene más artefactos que cuando establecimos el umbral de forma manual, éste método está automatizado y funcionará en imágenes semejantes pero permitiendo cierta variabilidad.

Referencias

Grompone von Gioi, R., & Randall, G. (2016). Unsupervised smooth contour detection.

Otsu, N. (1979). A threshold selection method from gray-level histograms.