Clasificando el dataset

Proceso de las imagenes

Convertir la imagen a escala de grises
Filtrar la imagen para eliminar el ruido
Aplicar el detector de bordes Canny
Buscar los contornos dentro de los bordes detectados
Dibujar dichos contornos

Convertir la imagen en escala de grises con OpenCV

Trabajar con imágenes a color, hace el coste computacional crezca de manera exponencial, entonces una vez que has cargado la imagen, lo primero es convertir a escala de grises con OpenCV.

Para aplicar el detector de bordes Canny necesitamos que la imagen esté en escala de grises. En OpenCV hay un método que permite realizar la escala de grises.

```
1  import numpy as np
2  import cv2
3
4  # Cargamos la imagen
5  original = cv2.imread("imagenes/modelo1.jpg")
6  cv2.imshow("original", original)
7
8  # Convertimos a escala de grises
9  gris = cv2.cvtColor(original, cv2.COLOR_BGR2GRAY)
10
```

Filtrado del ruido en una imagen

El objetivo es suavizar la imagen es decir, eliminar los detalles, las imágenes digitales no son perfectas. El ruido inherente de los propios dispositivos o los efectos contraproducentes por iluminación alteran la realidad.

```
# Aplicar suavizado Gaussiano
gauss = cv2.GaussianBlur(gris, (5,5), 0)

cv2.imshow("suavizado", gauss)
```

Aplicación del umbral

Es el proceso por el cual una imagen en escala de grises es convertida a una nueva con solo dos niveles blanco y negro, de esta manera los objetos quedan separados del fondo.

```
# Si todavía no hemos obtenido el fondo, lo obtenemos
# Será el primer frame que obtengamos
if fondo is None:
    fondo = gris
    continue

# Calculo de la diferencia entre el fondo y el frame actual
resta = cv2.absdiff(fondo, gris)

# Aplicamos un umbral
umbral = cv2.threshold(resta, 25, 255, cv2.THRESH_BINARY)[1]
```

Detección de Contornos

Una vez que tenemos la imagen con píxeles blancos o negros, tenemos que detectar los bordes. Una vez que se tienen los bordes, hay que decidir cuales de ellos forman contornos y cuales no.

```
# Detectamos los bordes con Canny
canny = cv2.Canny(gauss, 50, 150)

cv2.imshow("canny", canny)

# Buscamos los contornos
(_, contornos,_) = cv2.findContours(canny.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)



cv2.drawContours(original,contornos,-1,(0,0,255), 2)
cv2.imshow("contornos", original)

cv2.waitKey(0)
```

Solo detectar los externos (cv2.RETR_EXTERNAL) y se hará una aproximación para eliminar los píxeles del contorno redundantes (cv2.CHAIN_APPROX_SIMPLE).