```
"""goals6simple.py

  Read the camera image in preparation for some image manipulation
  and object detection.

"""

# Import OpenCV
import cv2

# Set up video capture device (camera).  Note 0 is the camera number.
# If things don't work, you may need to use 1 or 2?
camera = cv2.VideoCapture(0, cv2.CAP_V4L2)
if not camera.isOpened():
    raise Exception("Could not open video device: Maybe change the cam number?")

# Change the frame size and rate.  Note only combinations of
# widthxheight and rate are allowed.  In particular, 1920x1080 only
# reads at 5 FPS.  To get 30FPS we downsize to 640x480.
camera.set(cv2.CAP_PROP_FRAME_WIDTH,  640)
camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
camera.set(cv2.CAP_PROP_FPS,           30)

# Change the camera settings.
exposure = 235
wb = 3273
focus = 0

#camera.set(cv2.CAP_PROP_AUTO_EXPOSURE, 3)       # Auto mode
camera.set(cv2.CAP_PROP_AUTO_EXPOSURE, 1)       # Manual mode
camera.set(cv2.CAP_PROP_EXPOSURE, exposure)     # 3 - 2047, default 250

#camera.set(cv2.CAP_PROP_AUTO_WB, 1.0)           # Enable auto white balance
camera.set(cv2.CAP_PROP_AUTO_WB, 0.0)           # Disable auto white balance
camera.set(cv2.CAP_PROP_WB_TEMPERATURE, wb)     # 2000 - 6500, default 4000

#camera.set(cv2.CAP_PROP_AUTOFOCUS, 1)           # Enable autofocus
camera.set(cv2.CAP_PROP_AUTOFOCUS, 0)           # Disable autofocus
camera.set(cv2.CAP_PROP_FOCUS, focus)           # 0 - 250, step 5, default 0

camera.set(cv2.CAP_PROP_BRIGHTNESS, 154)        # 0 - 255, default 128
camera.set(cv2.CAP_PROP_CONTRAST,   128)        # 0 - 255, default 128
camera.set(cv2.CAP_PROP_SATURATION, 210)        # 0 - 255, default 128


# Keep scanning, until 'q' hit IN IMAGE WINDOW.
count = 0
while True:
    # Grab an image from the camera.  Often called a frame (part of sequence).
    ret, frame = camera.read()
    count += 1

    # Grab and report the image shape.
    (H, W, D) = frame.shape
    #print(f"Frame #{count:3} is {W}x{H} pixels x{D} color channels.")

    # Convert the BGR image to RGB or HSV.
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)      # For other objects
    # hsv = cv2.cvtColor(frame, cv2.COLOR_RGB2HSV)      # For red objects

    # Print color of center pixel
    print('BGR at center pixel: ', frame[W//2, H//2])
    print('HSV at center pixel: ', hsv[W//2, H//2])


    # Cross hair on center pixel
    (xA1, yA1) = (W // 2, 0)
    (xA2, yA2) = (W // 2, H - 1)
    (xB1, yB1) = (0, H // 2)
    (xB2, yB2) = (W - 1, H // 2)
    cv2.line(frame, (xA1, yA1), (xA2,yA2), (0,0,255), 1)
    cv2.line(frame, (xB1, yB1), (xB2,yB2), (0,0,255), 1)


    binary = cv2.inRange(hsv, (75, 115, 50), (115, 230, 190))
    binary = cv2.erode(binary, None, iterations=3)
    binary = cv2.dilate(binary, None, iterations=1)

    # Add contours
    (contours, hierarchy) = cv2.findContours(binary, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

```
81      contours = sorted(contours, key=cv2.contourArea, reverse=True)
82     # cv2.drawContours(frame, contours, -1, (0,0,255), 2)
83
84      CUTOFF = 1000
85
86      for contour in contours:
87          if cv2.contourArea(contour) > CUTOFF:
88              cv2.drawContours(frame, [contour], 0, (0,255,0), 2)
89
90              # single contour centroid method
91              # M = cv2.moments(contour)
92              # area = M['m00']
93              # x_c = int(M['m10'] / M['m00'])
94              # y_c = int(M['m01'] / M['m00'])
95
96              # fit an ellipse to a single contour
97              # ellipse = cv2.fitEllipse(contour)
98              # ((xe, ye), (w, h), angle) = cv2.fitEllipse(contour)
99              # cv2.ellipse(frame, ellipse, (0,255,255), 2)
100             # x_c = xe + w/2
101             # y_c = ye + h/2
102             # print(f'({xe}, {ye})')
103             # cv2.circle(frame, (int(xe), int(ye)), 4, (0, 255, 255), -1)
104         else:
105             cv2.drawContours(frame, [contour], 0, (0,0,255), 2)
106
107     # Show the processed image with the given title.  Note this won't
108     # actually appear (draw on screen) until the waitKey(1) below.
109     cv2.imshow('Processed Image', frame)
110     cv2.imshow('Binary Image', binary)
111
112     # Check for a key press IN THE IMAGE WINDOW: waitKey(0) blocks
113     # indefinitely, waitkey(1) blocks for at most 1ms.  If 'q' break.
114     # This also flushes the windows and causes it to actually appear.
115     if (cv2.waitKey(1) & 0xFF) == ord('q'):
116         break
117
118 # Close everything up.
119 camera.release()
120 cv2.destroyAllWindows()
```