

In [5]:

```
import cv2
import numpy as np
import pandas as pd
import time
```

In [6]:

```
data = pd.read_csv('CAM7_20220203_092129_output_Tail_unique.csv')
data.head()
```

Out[6]:

	VERSION	CAMERA_NUMBER	VIDEO_NAME	FRAMENUMBER	CLASS	SCORE_CLASS	X1_ORGAN	Y1_ORGA
0	2023-06-05	CAM7	CAM7_20220203_092129.avi	0	Tail	0.97	68.0	1120
1	2023-06-05	CAM7	CAM7_20220203_092129.avi	1	Tail	0.98	66.0	1120
2	2023-06-05	CAM7	CAM7_20220203_092129.avi	2	Tail	0.95	56.0	1120
3	2023-06-05	CAM7	CAM7_20220203_092129.avi	3	Tail	0.94	41.0	1119
4	2023-06-05	CAM7	CAM7_20220203_092129.avi	4	Tail	0.98	32.0	1120

In [7]:

```
x1 = data.iloc[:,6]
y1 = data.iloc[:,7]
x2 = data.iloc[:,8]
y2 = data.iloc[:,9]
```

In [8]:

```
# Open the video file
input_video_path = 'rgb_CAM7_20220203_092129_raw_input_Tail.avi'
# output_video_path = 'output_video.mp4'
cap = cv2.VideoCapture(input_video_path)

# Initialize time variables for text refresh
last_text_update_time = time.time()
text_refresh_interval = 0.5 # seconds

if not cap.isOpened():
    print("Error: Could not open video file")
    exit()

# Create separate windows for displaying frames
cv2.namedWindow('Original Frame', cv2.WINDOW_NORMAL)
cv2.resizeWindow('Original Frame', 540, 304)
cv2.namedWindow('Gray', cv2.WINDOW_NORMAL)
cv2.resizeWindow('Gray', 540, 304)
cv2.namedWindow('Frame Contour', cv2.WINDOW_NORMAL)
cv2.resizeWindow('Frame Contour', 540, 304)
cv2.namedWindow('Threshold', cv2.WINDOW_NORMAL)
cv2.resizeWindow('Threshold', 540, 304)

# Loop through the frames and rotate each frame
prev_frame = None
i=0
frame_count = 0

while True:
```

```

ret, frame = cap.read()
if not ret:
    break
(h,w,_)=frame.shape

# Convert the frame to grayscale
frame_gray1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

if prev_frame is not None:
    # Calculate frame difference
    frame_diff = cv2.absdiff(prev_frame, frame_gray)
    # frame_diff = cv2.cvtColor(frame_diff, cv2.COLOR_GRAY2BGR)
    frame_diff = cv2.addWeighted(frame_diff, 3, frame_gray1, 0.5, 0)
else:
    frame_diff = frame.copy() # Initialize frame_diff with the first frame

prev_frame = frame_gray # Update the previous frame
frame_gray2 = cv2.bitwise_not(frame_gray1)

c = 255/(np.log(1 + np.max(frame_gray2)))
log_transformed = c * np.log(1 + frame_gray2)
# Chỉ định kiểu dữ liệu cho biến log_transformed.
log_transformed = np.array(log_transformed, dtype=np.uint8)
# Áp dụng sự sửa đổi gamma cho ảnh.
frame_gray2 = np.array(255 * (frame_gray2 / 255) ** 0.8, dtype=np.uint8)

# Applying Gaussian Blur
frame_gray2 = cv2.GaussianBlur(frame_gray2, (5,5),10)

# Image thresholding
_, frame_gray2 = cv2.threshold(frame_gray2, 45, 255, cv2.THRESH_BINARY_INV)

# Erosion followed by dilation để lọc gợn ảnh
kernel = np.ones((3,1),np.uint8) # Lọc theo chiều ngang
opening1 = cv2.morphologyEx(frame_gray2, cv2.MORPH_OPEN, kernel)
kernel = np.ones((1,3),np.uint8) # Lọc theo chiều dọc
opening2 = cv2.morphologyEx(frame_gray2, cv2.MORPH_OPEN, kernel)
# Cộng 2 ảnh vừa lọc
frame_gray2 = cv2.addWeighted(opening1, 0.5, opening2, 0.5, 0.0)

# Rotating the frame 90 degrees
center = (w//2,h//2)

M = cv2.getRotationMatrix2D(center, -90, 1.0)
frame = cv2.warpAffine(frame, M, (w,h))
# frame = cv2.resize(frame, (1080, 608))
frame_diff = cv2.warpAffine(frame_diff, M, (w,h))
# frame_diff = cv2.resize(frame_diff, (1080, 608))
frame_gray1 = cv2.warpAffine(frame_gray1, M, (w,h))
# frame_gray1 = cv2.resize(frame_gray1, (1080, 608))
frame_gray2 = cv2.warpAffine(frame_gray2, M, (w,h))
# frame_gray2 = cv2.resize(frame_gray2, (1080, 608))
rectang = cv2.rectangle(frame_gray2, (int(381+x1[i]*9/8.5), int(y1[i]*9/15)), (381+int(x2[i]*9/8.2), int(y2[i]*9/13)), (0,255,0), 1)
rectang1 = cv2.rectangle(frame, (int(381+x1[i]*9/8.5), int(y1[i]*9/15)), (381+int(x2[i]*9/8.2), int(y2[i]*9/13)), (0,255,0), 1)

# Find contours of white regions inside the rectangle
gray_roi = frame_gray2[int(y1[i] * 9 / 15):int(y2[i] * 9 / 13),
                        int(381 + x1[i] * 9 / 8.5):381 + int(x2[i] * 9 / 8.2)]
contours, _ = cv2.findContours(gray_roi, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Draw and calculate contour length
for contour in contours:
    # Calculate the contour's position in the original frame
    contour_offset_x = int(381 + x1[i]*9/8.5)
    contour_offset_y = int(y1[i]* 9/15)
    for point in contour:
        point[0][0] += contour_offset_x
        point[0][1] += contour_offset_y # Shift contour to original frame
    # Draw the contours on the original frame in green color

```

```

    frame_contour = cv2.cvtColor(frame_gray2, cv2.COLOR_GRAY2BGR)
    cv2.drawContours(frame_contour, [contour], -1, (0, 255, 0), 0)
    cv2.drawContours(frame, [contour], -1, (0, 255, 0), -1) # -1 thickness means fill in the contour

    contour_length = cv2.arcLength(contour, closed=True)
    # Check if it's time to update the text
    current_time = time.time()
    if current_time - last_text_update_time >= text_refresh_interval:
        text = f"Length: {round(contour_length, 2)}"
        last_text_update_time = current_time

    # Display the contour length on the frame
    cv2.putText(frame, text, (600, 600), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 255, 0), 4
)

    i += 1
    frame_count += 1

    # frame = cv2.rectangle(frame, (), (), (0,255,0),1)
    cv2.imshow('Original Frame', frame)
    cv2.imshow("Frame Contour", frame_contour)
    cv2.imshow('Gray', cv2.bitwise_not(frame_gray1))
    cv2.imshow('Threshold', frame_gray2)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release video capture and writer objects
cap.release()

# Close all OpenCV windows (if any)
cv2.destroyAllWindows()

# print("Video has been rotated and saved to", output_video_path)

```

```

C:\Users\admin\AppData\Local\Temp\ipykernel_7832\522824876.py:51: RuntimeWarning: divide
by zero encountered in log
    log_transformed = c * np.log(1 + frame_gray2)
C:\Users\admin\AppData\Local\Temp\ipykernel_7832\522824876.py:53: RuntimeWarning: invalid
value encountered in cast
    log_transformed = np.array(log_transformed, dtype=np.uint8)

```