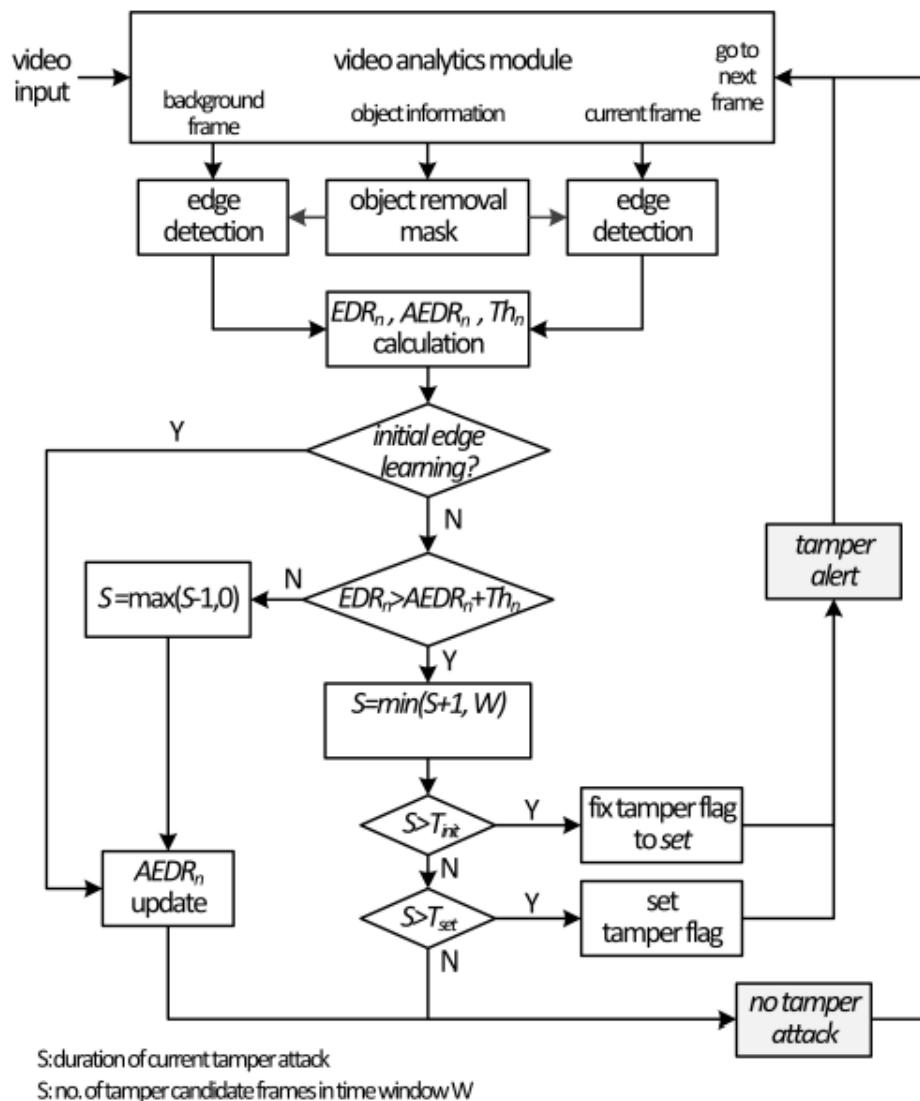


Phát hiện phá hoại camera

(Implement từ bài báo: <https://sci-hub.se/10.3390/s150510315>)

Thuật toán



Flowchart đã được đơn giản hóa hơn so với paper gốc

Ban đầu ta sẽ điều chỉnh video về kích thước 320x240 để đảm bảo về mặt tốc độ xử lý

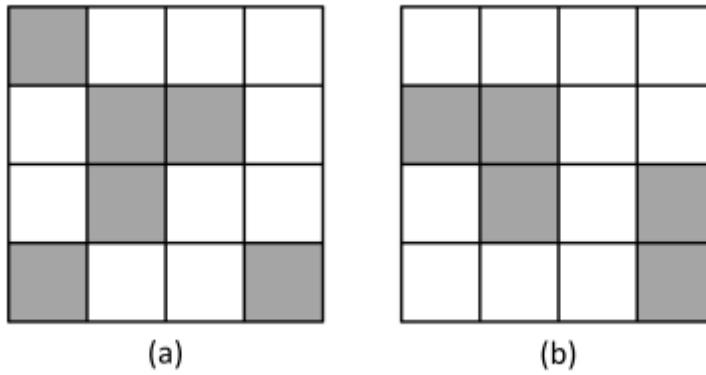
Lưu ý, ta sẽ dùng thuật toán Sobel để phát hiện cạnh thay vì Canny (bởi Canny cho ra cạnh rất mỏng, nên thuật toán rất dễ hoạt động sai)

Hành vi phá hoại camera được phát hiện bởi độ biến mất cạnh pixel (Edge disappearance rate, gọi tắt là EDR) ở khung hình hiện tại và cạnh pixel của khung hình background. Công thức như sau:

$$EDR_n = 1 - \frac{\sum_{p \in R^C} e_{n,p}^{BG} \cdot e_{n,p}^C}{\sum_{p \in R^C} e_{n,p}^{BG}}$$

với $e_{n,p}^C$ và $e_{n,p}^{BG}$ lần lượt thể hiện sự tồn tại của cạnh tại pixel p ở khung hình hiện tại n và khung hình nền (khung hình nền này đã được update trước khung hình n). Giá trị của chúng là 255 nếu nó là pixel cạnh, 0 nếu nó không phải là pixel cạnh. R^C biểu diễn vùng trong khung hình hiện tại khi **không** bao gồm các vật thể nền (vật thể foreground) (tức là ta sẽ loại bỏ foreground khỏi việc tính toán EDR). Ví dụ:

$$\sum_{p \in R^C} e_{n,p}^{BG} = 6 \quad \sum_{p \in R^C} e_{n,p}^{BG} \cdot e_{n,p}^C = 3 \Rightarrow EDR_n = 0.5$$



- (a) ảnh pixel cạnh của background
- (b) ảnh pixel cạnh của current frame

Để quá trình loại bỏ hiệu quả hơn ta dùng cv2.dilate để làm đầy vật thể nền

Khung hình nền được tạo ra bằng **cv2.createBackgroundSubtractorMOG2()** với setting là **detectShadows = False, setNMixtures(4), setVarInit(100), learningRate=0.002**

(learningRate càng lớn thì độ hấp thụ khung hình hiện tại vào khung hình nền càng nhanh)
Khung hình nền sẽ dừng lại trước khi camera bị tấn công (nhưng vẫn ngầm cập nhật)

Sau đó ta cần phải có quyết định xem camera đó có bị phá hoại hay không bằng cách sử dụng ngưỡng thích ứng và so sánh sự khác nhau giữa EDR và trung bình EDR ($AEDR$). Ngưỡng thích ứng được tạo ra mà không cần điều chỉnh tham số đối với các đặc điểm ảnh thay đổi theo thời gian do ánh sáng hoặc do môi trường, đặc biệt với chuỗi video vào ban đêm có số lượng cạnh pixel thấp hơn so với ban ngày. Ngưỡng thích ứng tại frame thứ n là:

$$Th_n = \begin{cases} 150/E_n, & \text{if } \frac{E_n}{W \cdot H} < 0.026 \\ 400/E_n, & \text{if } 0.026 \leq \frac{E_n}{W \cdot H} < 0.046 \\ 1500/E_n, & \text{otherwise} \end{cases}$$

với E_n , W , H lần lượt là số pixel cạnh của khung hình nền thứ n , chiều rộng và chiều cao của khung hình hiện tại. Tác giả đã đặt $W = 320$, $H = 240$. Các thông số của ngưỡng thích ứng ở trên là của tác giả đặt ra sau khi phân tích nhiều cảnh khác nhau.

Trong quá trình khởi tạo (tức là trong vòng T_{init} khung hình đầu tiên thì ta sẽ không giám sát gì hết) ta sẽ tính $AEDR$ của các EDR trong T_{init} khung hình đầu tiên. T_{init} được đặt là 600. Sau quá trình này, $AEDR$ được cập nhật nếu không có hành vi tấn công camera như sau:

$$AEDR_n = \frac{1}{n-l} \sum_{m=1}^l (1 - f_m^t) \cdot EDR_m$$

với n và l lần lượt là số khung hình đã chạy qua và số khung hình bị tấn công camera cho đến khung hình hiện tại. f_m^t là flag (tamper flag) (được nói bên dưới) cho hành vi tấn công camera của khung hình thứ m , với giá trị là 0 nếu không có hành vi tấn công và 1 là có hành vi tấn công. Nếu có hành vi tấn công nào xảy ra thì $AEDR$ sẽ giữ nguyên giá trị.

Để tránh trường hợp báo động giả do nhiều và một số hiện tượng tức thời xảy ra (như chim bay qua camera, bị lóa sáng vài khung hình do phương tiện phản chiếu, ...) thì ta định nghĩa số xác thực giả mạo (tamper validation count) (S) như sau:

$$S = \begin{cases} \min(S + 1, W), & \text{if } EDR_n \geq AEDR_n + TH_n \\ \max(S - 1, 0), & \text{otherwise} \end{cases}$$

với W biểu diễn cho kích thước cửa sổ xác thực giả mạo (tamper validation window) theo đơn vị frame. *Ta cũng có thể coi đây là khoảng frame cho việc khôi phục lại background sau khi camera bị tấn công.*

Cuối cùng, tamper flag cho khung hình thứ n được tính như sau:

$$f_n^t = \begin{cases} 1, & \text{if } S \geq T_{set} \\ 0, & \text{otherwise} \end{cases}$$

với T_{set} là khoảng khung hình đã được đặt sẵn cho tamper validation. Tồn tại trade-off giữa T_{set} và báo động giả. Giá trị này được đặt là 3 theo tác giả. Nếu số xác thực giả mạo lớn hơn 3 thì ta sẽ coi đó là hành động phá hoại (thường thì trường hợp lóa sáng hay chim bay qua ...) sẽ chỉ diễn ra chỉ trong vài frame cho nên số xác thực giả mạo được tạo ra để tránh gặp phải những trường hợp đó)

Nếu tấn công camera diễn ra lâu hơn chu kỳ tạo khung hình nền của MOG2, thì việc tấn công này sẽ không được cảnh báo do đã bị hấp thụ vào khung hình nền. Cho nên, nếu $S \geq T_{fix}$ thì ta sẽ fix cứng tamper flag (để báo cho người điều hành camera biết là camera đó không còn ở góc quay ban đầu, hoặc là bị che khuất, hoặc là bị làm mờ), và chỉ người điều hành mới có thể xóa nó. T_{fix} được đặt là 600

Code bài báo đã được implement, kèm theo dataset để đánh giá được mô tả dưới đây:

Dataset sử dụng

Dataset đầu tiên (để đánh giá): UHCTD dataset (<http://qil.uh.edu/main/datasets/>) là dataset video quay tại một địa điểm trong suốt 24 tiếng, có người đi lại, có giả lập một số trường hợp phá hoại camera: che khuất, làm mờ, quay góc; có một số ít trường hợp ngoại lệ thêm vào dataset: người đứng tại đó nói chuyện, camera bị nhiễu nhẹ, bị đổi màu, ... có file groundtruth để đánh giá

(mô tả chi tiết tại:

<https://sh-tsang.medium.com/review-uhctd-a-comprehensive-dataset-for-camera-tampering-detection-camera-tampering-detection-f2a132eb7aca>).

Một số hình ảnh ví dụ (tất cả các hành vi phá hoại ở dataset này diễn ra tức thì để có thể tiện đánh giá):



bình thường

bị che khuất

bị làm mờ

bị quay góc

Dataset thứ 2 (để chạy trường hợp thực tế), lấy từ camera có IP là .227, quay giao thông ngoài đường Lạc Long Quân ở cổng trước công ty. Với các trường hợp quay khác nhau như sáng, tối, mưa. Dataset này dùng để kiểm tra xem thuật toán có bị phát hiện nhầm khi gặp những trường hợp kia hay không. Có tổng cộng 24 video, những video không được liệt kê dưới đây thì không phải trường hợp đặc biệt. Cụ thể như sau:

Tối:

output17.avi: Trời mưa

Sáng:

output23.avi: Bị tắt nắng đột ngột

output22.avi: Người di chuyển gần camera + trời nắng (cực khó)

output18.avi: Mưa, các frame cuối camera bị rung lắc

output15.avi: Lúc nắng lúc không (ánh nắng thay đổi chậm)

output14.avi: Trường hợp xe di chuyển khỏi vị trí cố định

output10.avi: Trường hợp thay đổi màu sắc

output4.avi: Trường hợp nắng đột ngột

Một số hình ảnh minh họa:



Đánh giá và phân tích

Theo phương pháp đề xuất này, cần phải có ít nhất $T_{init} = 600$ khung hình để khởi tạo background ổn định.

Metric đánh giá:

- TP: alert for a tamper attack
- TN: no alert for no tamper attack
- FN: no alert for a tamper attack
- FP: alert for no tamper attack

$$\text{Detection rate (DR)} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{Precision rate (PR)} = \text{TP} / (\text{TP} + \text{FP})$$

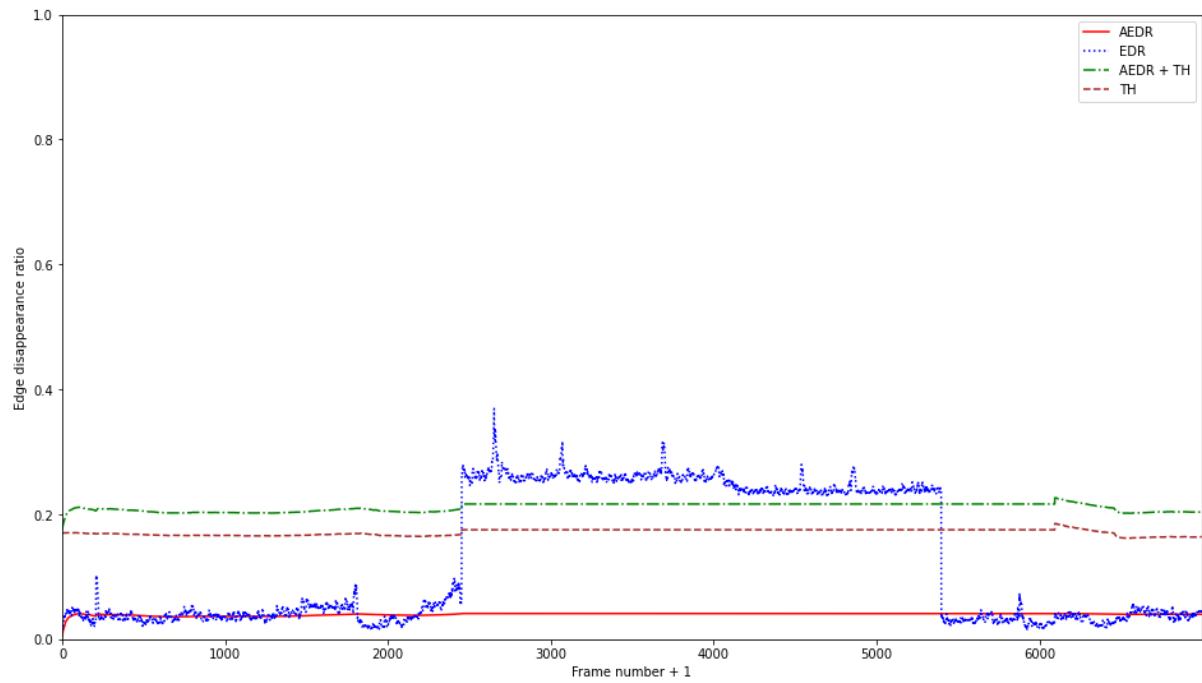
$$\text{Daily false alarm rate (DFAR)} = \text{FP} / T$$

(với T là số lượng frame của video đang xét)

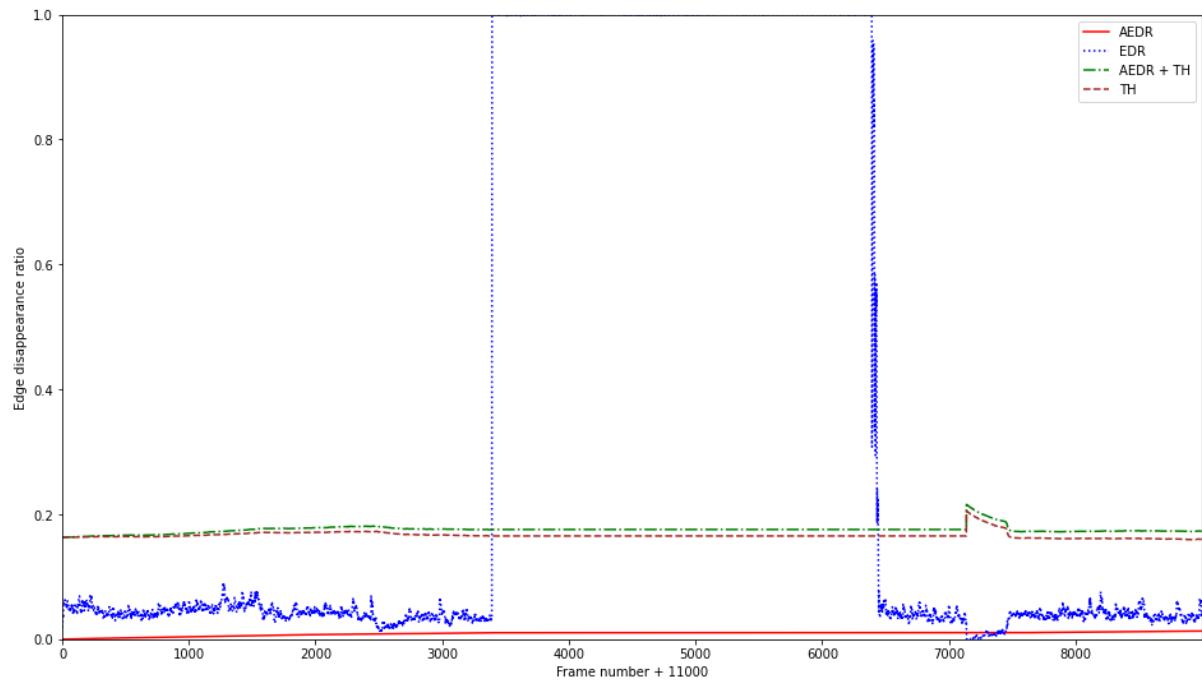
Kết quả đánh giá dataset đầu tiên:

Video	Attack type	Detection rate	Precision rate	Daily false alarm rate
Camera B/Training video/Day 2	covered (nằm trong frame 1 - 7000)	0.98	0.81	0.10
	defocus (nằm trong frame 11000-20000)	1	0.8	0.04
	moved (nằm trong frame 25600 - 31000)	1	0.81	0.02

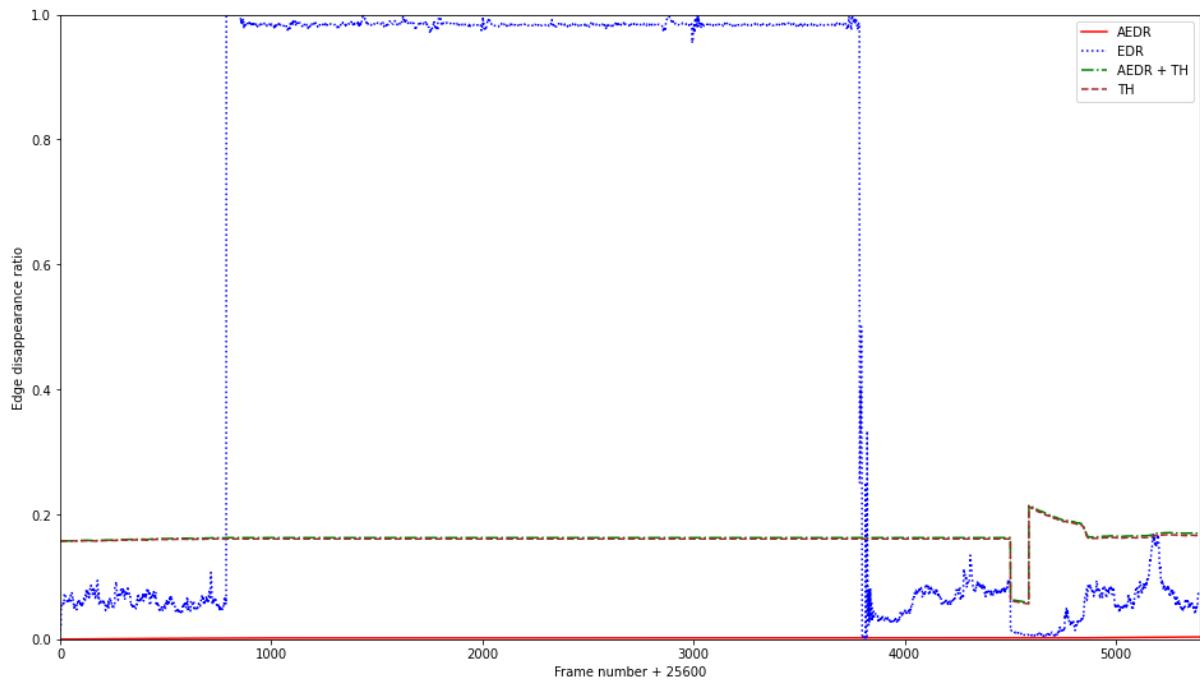
Biểu đồ của trường hợp **che khuất** (cột Oy là độ biến mất cạnh, cột Ox là số frame đã trôi qua, nếu **đường màu xanh dương** mà vượt qua **đường màu xanh lá** thì đó là trường hợp phá hoại camera):



Biểu đồ trường hợp bị **làm mờ**:

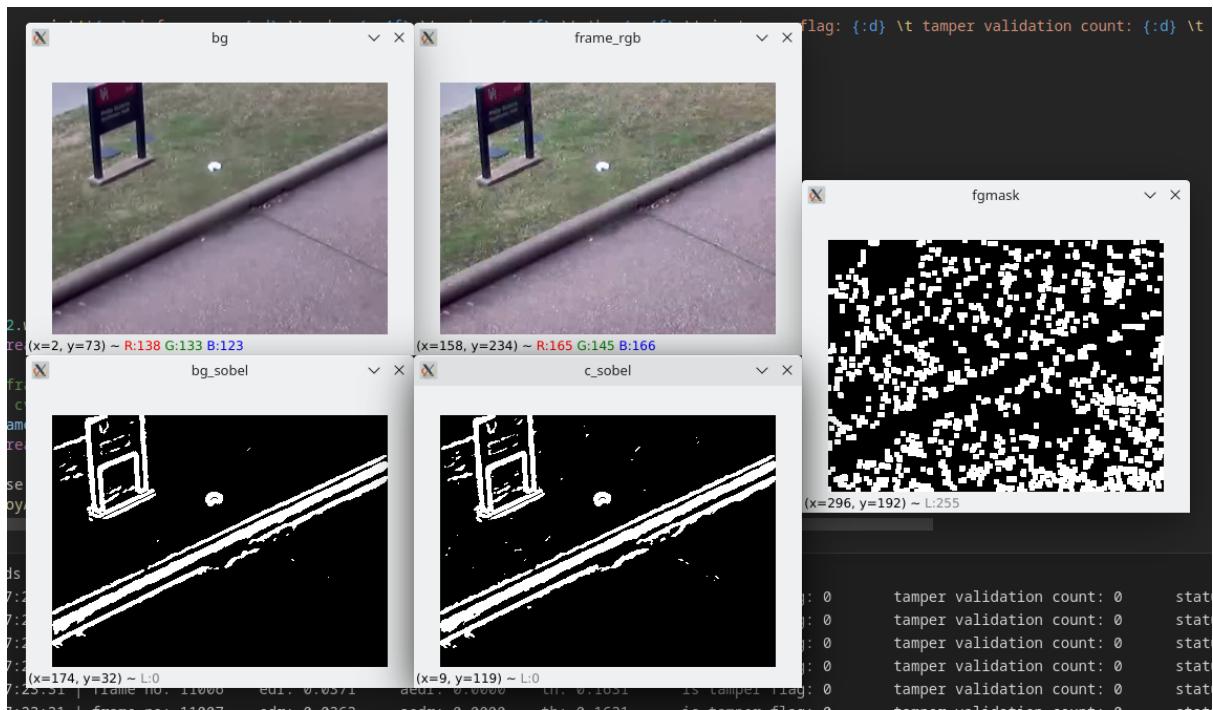


Biểu đồ trường hợp bị **quay góc**:



=> Qua 3 biểu đồ ở trên, ta có thể thấy được thuật toán có thể phát hiện tốt ra 3 trường hợp phá hoại.

Hình ảnh minh họa hoạt động của thuật toán



Chú thích:

Cửa sổ **bg**: background frame

Cửa sổ **bg_sobel**: cạnh của background frame

Cửa sổ **frame_rgb**: current frame

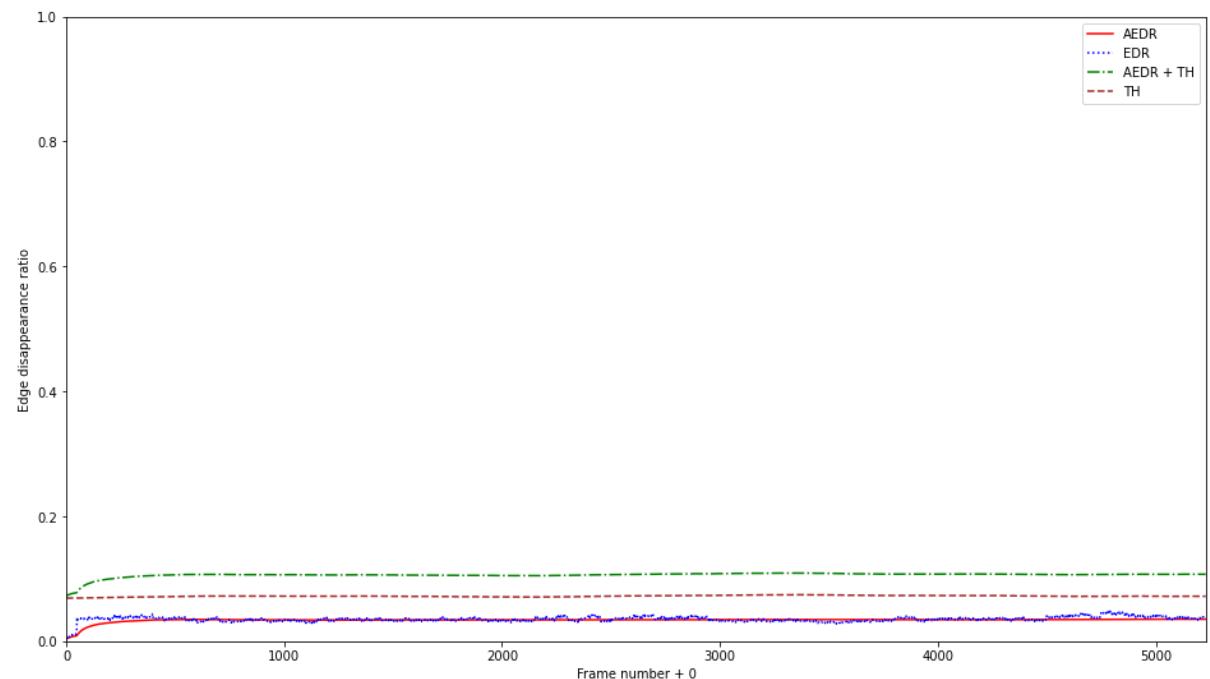
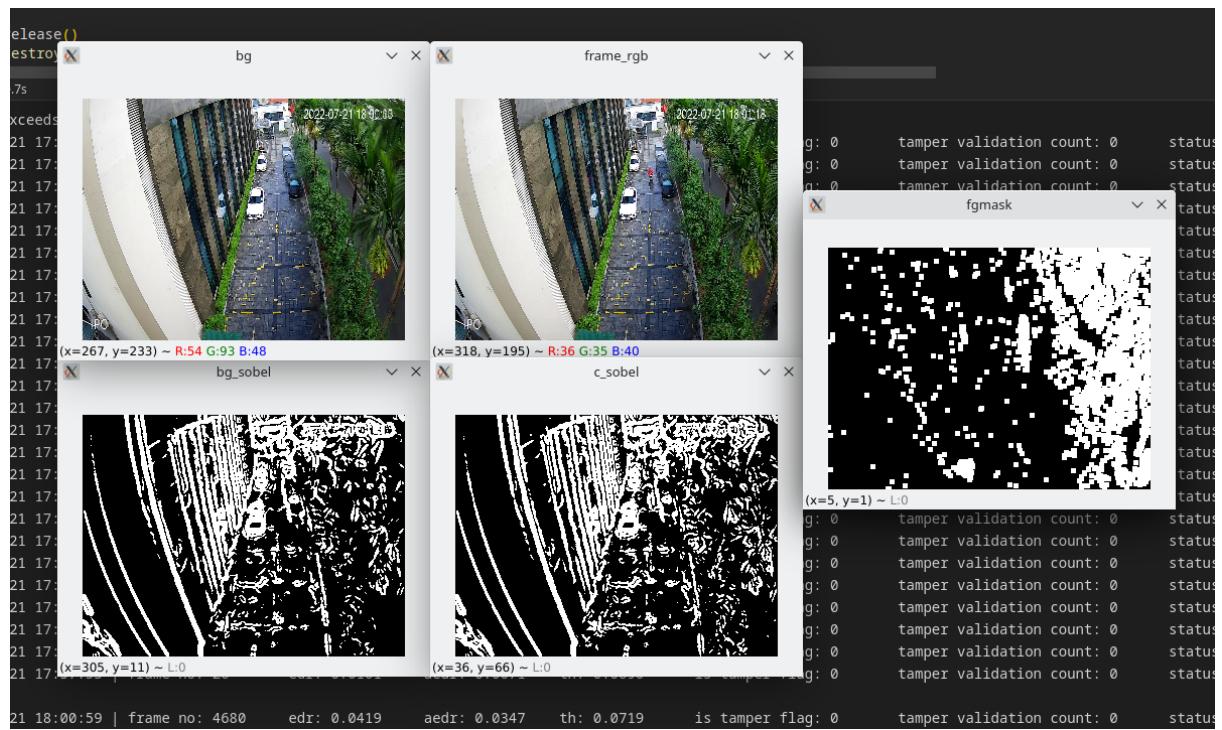
Cửa sổ **c_sobel**: cạnh của current frame

Cửa sổ **fgmask**: detect foreground để loại bỏ khỏi tính toán cạnh

- Lý do detection rate của trường hợp che khuất ở bảng trên chỉ là 0.98, trong khi đó 2 trường hợp còn lại là 1, là do khi che khuất thì thuật toán sẽ coi phần che khuất là vật thể foreground trong vòng một vài frame (tức là thuật toán này sẽ loại bỏ vật thể foreground khỏi quá trình tính toán sự khác biệt cạnh), sau đó mới phát hiện được cạnh ở frame hiện tại bị biến mất.
- Nhược điểm của thuật toán khi sử dụng dataset này: Nếu ta che khuất phần không có “cạnh” (giống phần khoanh đỏ ở hình dưới đây) thì thuật toán sẽ bị nhận diện nhầm là không bị che khuất (khi sử dụng trong trường hợp thực tế sẽ nhiều cạnh hơn cho nên ta sẽ không gặp trường hợp kiểu như này)

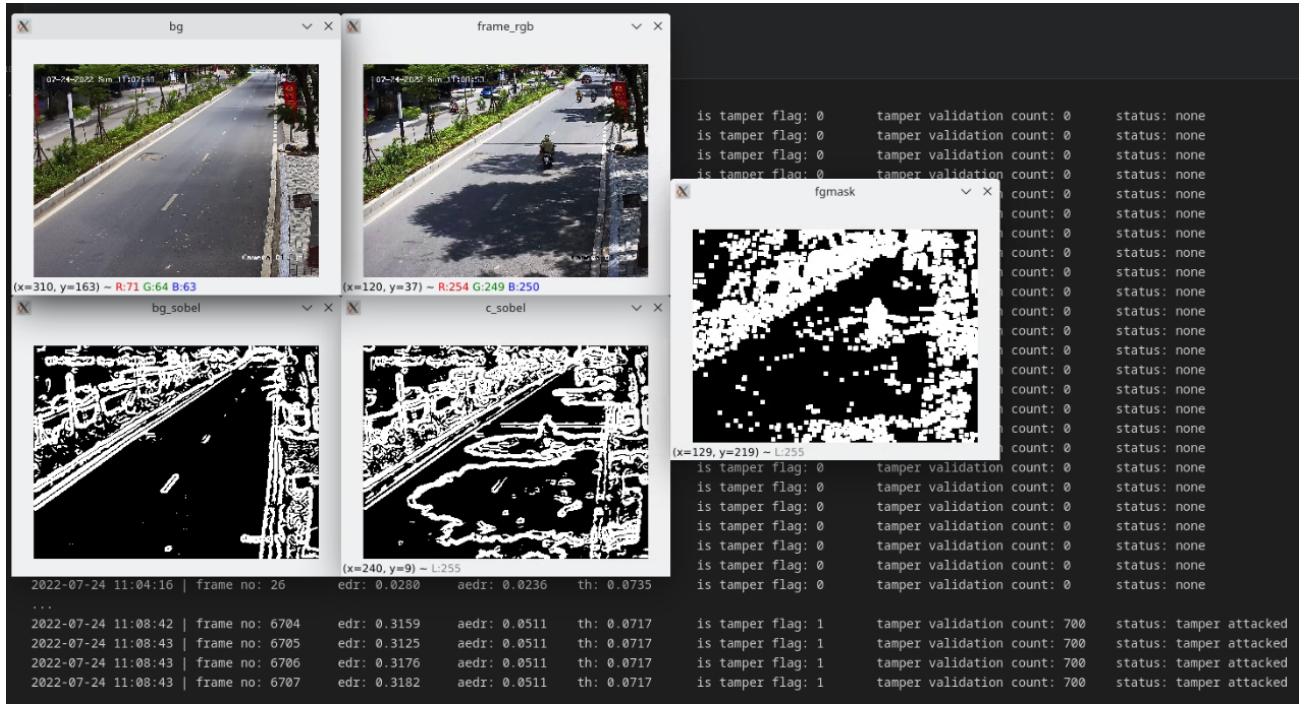


Phương pháp này hoạt động tốt nhất trong môi trường indoor, một số môi trường outdoor ở điều kiện **lý tưởng** (ánh sáng đều, thời tiết mưa) thì có thể hoạt động được (VD: bãi đỗ xe sân sau công ty, biển đồ đạc được thể hiện bên dưới)



=> không bị phá hoại

Một số ví dụ trường hợp ngoài đường **khó**:



Ví dụ như ta nhìn cửa sổ **bg** thì thấy trời đang không nắng, đột nhiên trời nắng như cửa sổ **frame_rgb** thì có thể nhận diện nhầm thành tấn công camera (do trời nắng làm cho cây có bóng xuống mặt đường, nên thuật toán sobel nhận đó là cạnh, dẫn đến sự khác biệt về cạnh giữa khung hình nền và khung hình hiện tại) => **trường hợp khó**

Trường hợp khó thứ 2 (output14.avi) là ô tô đang đỗ gần vỉa hè di chuyển như sau:



=> **cũng bị phát hiện nhầm**

Bổ sung: Thuật toán SIFT

Dùng để giải quyết trường hợp “nắng đột ngột” do cây tạo bóng ở video output4.avi. Ta sẽ đặt `sift = cv2.xfeatures2d.SIFT_create(20)`, tức là sẽ tạo ra 20 keypoints tốt nhất ở trên frame, sử dụng `ratio_test = 0.5` để matching các keypoint

Nếu áp dụng SIFT keypoint vào bài toán này thì sẽ xảy ra thêm các trường hợp như sau:

- Nếu ta áp dụng nó với trường hợp bị quay góc thì thuật toán SIFT hoạt động ổn định. Bởi ta sẽ so sánh sự khác biệt giữa vị trí của các keypoint giữa khung hình thứ n và khung hình thứ n-k (với k là số khung hình trước đó) bằng khoảng cách L2. Ta sẽ đặt ngưỡng cho các khoảng cách L2 của các keypoint được kết hợp tốt (good matches) là d_{thresh} . Nếu trên 40% các khoảng cách đó đều lớn hơn d_{thresh} thì chắc chắn đó là trường hợp quay góc. Ngoài ra trường hợp quay góc có thể không có keypoint được kết hợp tốt (tức là không có keypoint nào được matching)
- Áp dụng với trường hợp làm mờ thì số keypoint của thuật toán này tạo ra sẽ giảm (ít hơn so với số keypoint khởi tạo là 20) nếu ta làm mờ rất nhiều, nếu làm mờ bình thường thì có thể số keypoint vẫn là 20. Cho nên trường hợp này khó có thể chọn ngưỡng hoặc xử lý logic
- Nếu với trường hợp che khuất hoàn toàn thì tương đối dễ do toàn bộ keypoint của ảnh đã bị mất hết hoặc các keypoint không thể matching được với nhau. Nếu chỉ che khuất một phần thì trường hợp này rất khó để nhận biết do khi bị che khuất một phần thì số keypoint sẽ xuất hiện ở phần không bị che khuất (do ta để số keypoint khởi tạo là 20 cho nên phần không bị che khuất sẽ phải có các keypoint) như sau:



=> Do bài phát hiện cạnh thông nhất cả 3 trường hợp: quay góc, làm mờ, che khuất vào làm một là phát hiện sự thay đổi cạnh của cảnh, cho nên khi áp dụng thuật toán SIFT vào thì sẽ tạo ra thêm các trường hợp khó xử lý như đã nhận xét ở trên. Cho nên ta vẫn cần tìm thêm thuật toán thích hợp.

=> Nếu sử dụng thuật toán phát hiện phá hoại video được đề xuất ở trên vào trong môi trường indoor (camera đặt cao) thì tính ổn định của thuật toán sẽ cao hơn

Cải tiến trong tương lai

- Để có thể loại bỏ trường hợp phát hiện nhầm khi trời nắng, ta có thể sử dụng thuật toán xóa bóng vật thể đơn giản, nhẹ (để tránh trường hợp cây đổ bóng lên đường)
- Trường hợp ô tô đỗ gần đó mà đột ngột di chuyển đi, chưa xử lý được