Project Computer Vision

Face mask detetion and tracking

Giảng viên hướng dẫn: **TS.**

Thành viên nhóm: Vương Đăng Huy

Bùi Song Kiệt

Nguyễn Thị Bích Thảo

Mục lục

1 Đặt vấn đề

Hiện nay, dịch Covid-19 đang xảy ra trên toàn thế giới, ảnh hưởng tới mọi mặt của đời sống xã hội. Covid-19 nguy hiểm và trở thành đại dịch do tính chất lây lan nhanh qua đường hô hấp của nó. Trong tình huống chưa có vắc-xin đặc hiệu thì việc phòng bệnh chủ yếu dựa vào các biện pháp như rửa tay sát khuẩn, dãn cách xã hội và quan trọng nhất là đeo khẩu trang. Do đó, các hệ thống phân loại người đeo và không đeo khẩu trang ngày càng có nhiều ứng dụng, tiêu biểu như các robot và máy bay không người lái tuần tra phát hiện và nhắc nhở người không đeo khẩu trang; hệ thống face mask detection tích hợp trong các camera giám sát tại các trung tâm thương mại, siêu thị, rạp chiếu phim…

Trong bối cảnh đó, nhóm quyết định xây dựng một hệ thống phát hiện người không đeo khẩu trang và đeo khẩu trang sai cách và theo dấu họ, nhằm tích hợp với các camera giám sát văn phòng, siêu thị, nhà máy… giúp quản lý tốt hơn việc phòng chống dịch của người dân.

2 Bài toán

Bài toán có thể được giải quyết qua hai module chính: object detection và object tracking. Module object detection nhằm phát hiện và phân loại khuôn mặt người đeo khẩu trang, không đeo khẩu trang và đeo khẩu trang sai cách. Ở đây nhóm sử dụng model tiny-yolo-v4 do model này cho performance khá tốt với bài toán object detection đơn giản và thời gian infer nhanh, thích hợp cho chạy realtime. Cứ 10 frames hệ thống sẽ thực hiện detect một lần, các khoảng giữa 10 frame đó, module object tracking sẽ bám vết theo các đối tượng.

3 Các bộ dữ liệu về phát hiện người đeo khẩu trang

Với số lượng ứng dụng face mask detection tăng nhanh như hiện nay, có nhiều bộ dataset cũng ra đời để đáp ứng nhu cầu huấn luyện mô hình. Các bộ dữ liệu phổ biến nhất hiện nay gồm có:

- Face mask detection:
    - Tổng số ảnh: 853

    - Số lớp: 3 lớp: đeo khẩu trang, không đeo khẩu trang và đeo khẩu trang sai cách.
    - Ảnh full body, chủ yếu là người châu Á.
- Face mask:
    - Tổng số ảnh: 12K
    - Số lớp: 2 lớp: đeo khẩu trang và không đeo khẩu trang.
    - Ảnh khuôn mặt, chủ yếu là người châu Âu.
- Masked face:
    - Tổng số ảnh: 8K
    - Số lớp: 2 lớp: đeo khẩu trang và không đeo khẩu trang.
    - Ảnh full body, đa số là ảnh sinh, có cả người châu Âu và châu Á.

Với mong muốn xây dựng hệ thống face mask detection dùng cho camera giám sát, có thể phát hiện cả người không đeo khẩu trang và người đeo khẩu trang sai cách, nhóm chọn bộ dataset face mask detection để làm dữ liệu training cho model object detection.

4 Khám phá dữ liệu

Bộ dữ liệu được sử dụng là bộ *Face Mask Detection* của *kaggle*, có thể được mô tả như sau:

- Tổng số ảnh: 853
- Số lớp: 3 lớp: đeo khẩu trang, không đeo khẩu trang và đeo khẩu trang sai cách (with_mask, without_mask, mask_weared_incorrect)
- Tổng số bounding boxes: 4072
    - Lớp without_mask: 717 bounding boxes (18%)
    - Lớp with_mask: 3232 bounding boxes (79%)
    - Lớp mask_weared_incorrect: 123 bounding boxes (3%)
- Các ảnh trong bộ dữ liệu có kích thước tương đối đồng đều, tập trung chủ yếu trong khoảng kích thước 400x300.
- Số lượng bounding boxes trong một ảnh:
    - Mean: 4.77
    - Median: 2
    - Max: 115

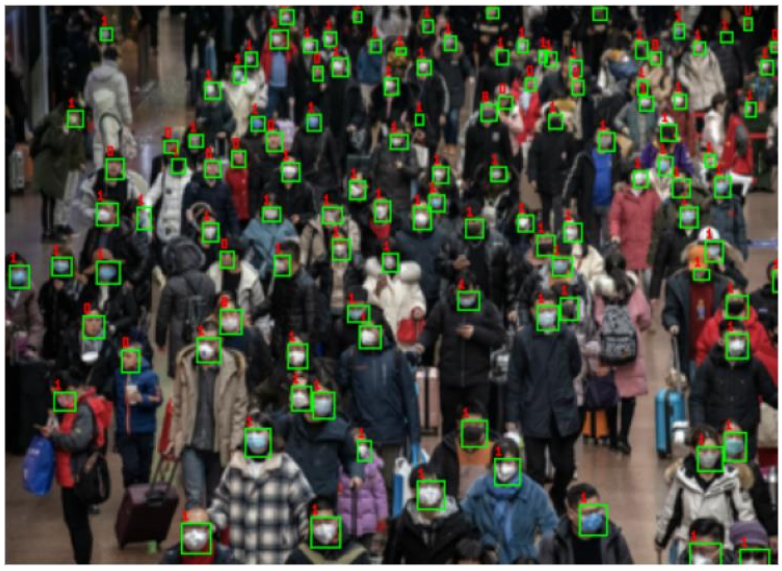
- Min: 1
- Standard deviation: 7

Nhận xét:

- Số lượng data nhỏ;
- Kích thước ảnh đồng đều là một lợi thế;
- Dữ liệu khuôn mặt chủ yếu của người châu Á, ảnh toàn cơ thể;
- Số lượng bounding boxes biến thiên lớn (độ lệch chuẩn = 7), ảnh nhiều bounding boxes nhất có số nhãn lên đến 115 (Hình), chất lượng hình ảnh trong bounding boxes rất kém;
- Xảy ra hiện tượng imbalance giữa các lớp: lớp with_mask chiếm đa số (79%) bộ dữ liệu, sau đó là lớp without_mask (chiếm 18%), lớp mask_weared_incorrect chiếm rất ít bounding boxes (3%).

Phương hướng giải quyết:

- Có thể sử dụng kết hợp với các bộ dữ liệu khác;
- Thu thập thêm dữ liệu;
- Thực hiện các biện pháp imbalance processing.



Hình 4.1 Ảnh nhiều bounding boxes nhất (115 bounding boxes) trong tập dữ liệu.

5 Phân công công việc

Từ hướng tiếp cận như trong phần 3, nhóm đưa ra danh sách các việc cần phải làm như trong bảng dưới.

Bảng 1 Bảng phân chia công việc các thành viên trong nhóm

STT	Công việc	Người đảm nhiệm	Start date	Due Date	Status
1	Tải dataset, tạo git repo	KietBS	02/02	02/02	Done
2	Thu thập, gán nhãn dữ liệu	All	02/02	07/02	Done
3	Inspect, visualize dữ liệu	ThaoNTB	07/02	07/02	Done
4	Tìm hiểu, triển khai mô hình YOLO	ThaoNTB	07/02	18/02	Done
5	Tìm hiểu, Triển khai mô hình theo dấu vật thể	HuyVD	07/02	18/02	Done
6	Xây dựng demo	KietBS	18/02	20/02	Done
7	Hoàn thiện slide, báo cáo	All	20/02	25/02	Done

6 Quy trình thực hiện

6.1 Xử lý mất cân bằng dữ liệu

Một vài biện pháp xử lý imbalance giữa các lớp có thể kể đến như sau: thu thập thêm dữ liệu; sử dụng performance metric đặc trưng như kappa, ROC curve; lấy mẫu lại dữ liệu (resampling data), tạo synthetic sample (SMOTE)… Tùy từng đặc điểm của bộ dữ liệu, có thể chọn sử dụng một trong các cách trên hoặc sử dụng kết hợp để đạt hiệu quả cân bằng cao. Với bài toán hiện tại, nhóm sử dụng các phương pháp sau:

6.1.1 Thu thập dữ liệu

Do số lượng ảnh trong dataset nhỏ (853 ảnh), việc thu thập thêm dữ liệu trước khi thực hiện các bước augment data hay resample data là cần thiết để cải thiện độ chính xác của model và tránh overfitting. Một lý do nữa khiến cho phương pháp này hiệu quả là trong bối cảnh dịch bệnh Covid-19 như hiện nay, việc thu thập ảnh người đeo khẩu trang là rất dễ dàng và sẵn có. Với lớp không đeo khẩu trang cũng có thể lấy dữ liệu từ bộ face detection để sử dụng. Tuy nhiên, lớp đeo khẩu trang sai cách khó thu thập hơn, số lượng ảnh có sẵn (có thể download từ internet) là rất ít, do đó nhóm tiến hành tự thu thập và gán nhãn thông qua công cụ *labelImg*.

Kết quả sau bước thu thập dữ liệu như sau (bộ face mask detection + collected data):

- Tổng số ảnh: 2620
- Tổng số bounding boxes: 9106
    - Lớp without_mask: 2080
    - Lớp with_mask: 5672
    - Lớp mask_weared_incorrect: 854
- Tesing data:
    - Tổng số ảnh: 255
    - Tổng số bounding boxes: 1192
        - Lớp without_mask: 472
        - Lớp with_mask: 565
        - Lớp mask_weared_incorrect: 215

6.1.2 Data augmentation

Sau khi thu thập thêm dữ liệu, số lượng bounding boxes thuộc mỗi lớp tăng lên đáng kể, tuy nhiên tỷ lệ giữa các lớp vẫn rất lớn (lớp có khẩu trang gấp 6.6 lần lớp đeo sai cách), để khắc phục có thể dùng phương pháp resampling data hoặc data augmentation. Do resampling data chỉ đơn giản tạo các bản copy của ảnh thuộc lớp hiếm khiến cho dữ liệu tăng lên khiến cho data có nhiều mẫu trùng nhau, dễ gây overfitting. Nhóm lựa chọn phương pháp augmentation để tạo thêm ảnh mới là các biến đổi ngẫu nhiên từ ảnh gốc.

Các biến đổi gồm có:

- Thay đổi độ sáng;
- Thêm các bộ lọc làm mờ (gaussian blur, average blur, median blur)
- Làm sắc nét (sharpen)
- Hiệu ứng emboss
- Cộng/trừ/nhân/chia giá trị các pixel với một giá trị ngẫu nhiên thuộc một khoảng xác định
- thêm nhiễu (gaussian noise, salt and peper)
- Thay đổi độ tương phản tuyến tính (linear contrast)…

Cách augment data:

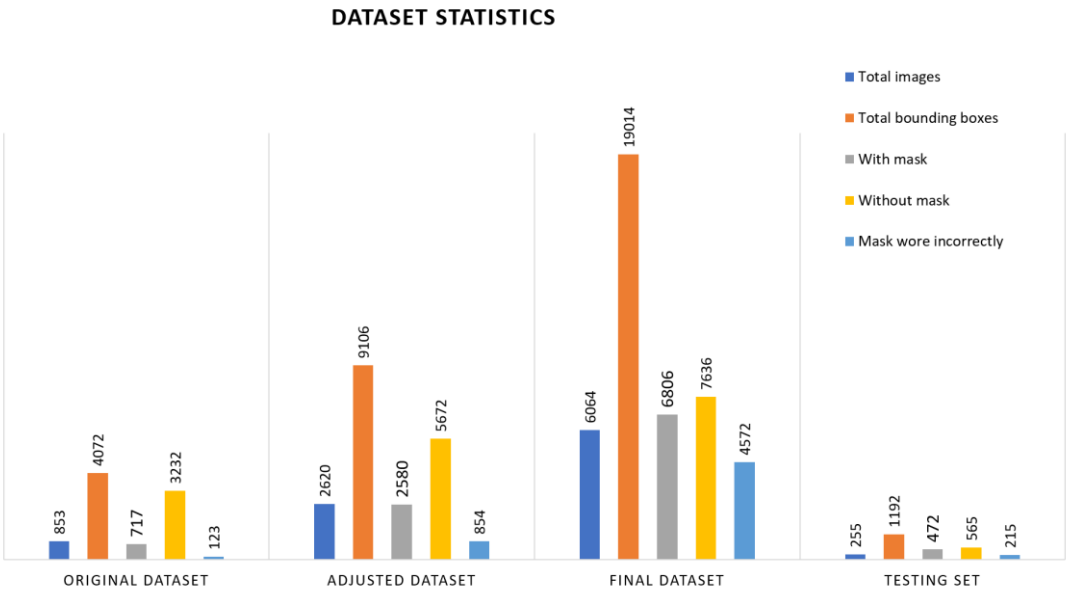
- Do số lượng bounding boxes thuộc lớp without_mask bằng một nửa lớp with_mask, nhóm chọn ra khoảng 200 ảnh có số lượng nhãn without_mask chiếm đa số, thực hiện sinh ba ảnh mới từ một ảnh gốc bằng các phương pháp ngẫu nhiên kể trên. Kết quả thu được số bounding boxes của hai lớp này xấp xỉ nhau.
- Số bounding boxes ở lớp mask_weared_incorrect nhỏ nhất, cần chọn các ảnh có lớp này chiếm đa số (không chứa quá nhiều hai nhãn còn lại), từ một ảnh gốc tạo thêm 8 ảnh mới. Kết quả số lượng bounding boxes lớp này bằng khoảng một nửa hai lớp còn lại.

Bộ dữ liệu huấn luyện hoàn chỉnh:

- Tổng số ảnh: 6064
- Tổng số bounding boxes: 19014

- Lớp without_mask: 6806
- Lớp with_mask: 7636
- Lớp mask_weared_incorrect: 4572

Hình 6.1 mô tả sự thay đổ về số lượng ảnh, số lượng bounding boxes và số lượng bounding boxes thuộc từng lớp từ dataset ban đầu đến sau bước thu thập data và data augmentation.



Hình 6.1 Sự thay đổ về số lượng ảnh, số lượng bounding boxes và số lượng bounding boxes thuộc từng lớp từ dataset ban đầu (original dataset) đến sau bước thu thập data (adjusted dataset) và data augmentation (final dataset).

6.1.3 Kết quả

Sau các bước tiền xử lý dữ liệu, bộ dữ liệu cuối cùng có thể được mô tả như sau:

- Tổng số ảnh: 6064 (gấp 7 lần dataset ban đầu)
- Tổng số bounding boxes: 19014
  - Lớp without_mask: 6806 (36%)
  - Lớp with_mask: 7636 (40%)
  - Lớp mask_weared_incorrect: 4572 (24%)

6.2 Object detection với tiny-yolo-v4

6.2.1 Xây dựng mô hình

Khác với R-CNN và Faster R-CNN, YOLO có cách tiếp cận bài toán nhận diện đồ vật khác hoàn toàn. Thay vì mô hình hai giai đoạn thường thấy với một bộ phân lớp thì mô hình YOLO đưa bài toán về dạng regression: mô hình YOLO sẽ trực tiếp dự đoán ra các khung hình có chứa số đồng thời tiến hành phân lớp cho các khung hình này một cách đồng thời. Ưu điểm rõ rệt của phương pháp này đó chính là tốc độ xử lý được tăng một cách đáng kể. Trong bài báo của YOLOv3, thì mô hình có thể đạt được tới trên 30 khung hình/giây (FPS) với mô hình đầy đủ, và trên 200 FPS với mô hình YOLO thu nhỏ. Mô hình thu nhỏ còn có thể chạy trên các máy tính xách tay không có GPU, cụ thể trên CPU Intel I7-7700HQ thì mô hình thu nhỏ có thể xử lý trung bình trên 3 khung hình một giây.

Ý tưởng cơ bản của YOLO là đưa hình ảnh về một kích thước vuông (418x418 hoặc 608x608, tuy nhiên mọi kích cỡ là bội của 32 đều có thể được sử dụng), sau đó chia hình ảnh ra thành một lưới các ô vuông nhỏ hơn. Đối với mỗi ô vuông thì mô hình sẽ dự đoán ra **B** khung hình có chứa vật thể mà trung tâm của hình nằm trong ô vuông đó. YOLO sẽ xử lý một hình ảnh trong duy nhất một lần chạy thay vì trích ra một số lượng lớn các khung hình để tiến hành phân lớp, điều này làm cho mô hình có được thông tin từ toàn thể bức hình trong quá trình phân tích. Mô hình vẫn sử dụng các mạng tích chập (convolutional) để trích xuất đặc trưng từ hình ảnh. 3 feature map đầu ra sẽ có nhiều kích cỡ khác nhau để dự đoán hình ảnh với các kích cỡ khác nhau (Với đầu vào 418x418 thì kích cỡ là 13x13, 26x26, 52x52).

Các anchor box của mô hình YOLO được xác định tương đối so với kích thước của ảnh chứ không phải là một số pixel nhất định, điều này là một chỉnh sửa trong các phiên bản sau của YOLO. Tức là kích cỡ của các anchor box này đã được chuẩn hóa (normalize) trong khoảng 0-1 dựa trên kích cỡ đầu vào. Các anchor box có được xác định bằng thuật toán k-means tương tự như các thuật toán CV khác, và chỉ được sử dụng tại lớp dự đoán. Các kích thước của anchor box trong quá trình khởi tạo được mặc định về cùng một giá trị, tuy nhiên sẽ được thay đổi, tăng thêm trong quá trình huấn luyện, điều này có nghĩa là mô hình sẽ tự hiệu chỉnh sao cho các anchor box

này có kích cỡ khớp nhất với đối tượng. Việc khởi tạo chỉ giúp quá trình này hội tụ nhanh hơn mà thôi. Càng lớp extra của của YOLO cũng không quá phức tạp, với lớp ROUTE là để kết hợp các layer hoặc chỉ dẫn các layer đã có trước đó đến vị trí mà ROUTE được đặt trong mạng. Ngoài ra YOLO còn sử dụng các lớp BatchNorm ngay trong các lớp tích chập, cũng như các lớp MaxPooling. Lớp dự đoán của YOLO được tác giả đặt tên YOLO, đấy sẽ là lớp sử dụng các Intersect over Union (IoU), các anchor box và Non-Maxima Suppression.

Đầu ra của của mạng YOLO sẽ là một vector chứa tọa độ của điểm trung tâm của đối tượng, kích cỡ của hộp chứa, và loại đối tượng dưới dạng one-hot encoding.

Nhóm đã chọn sử dụng mô hình tiny-YOLOv4 vì quá trình huấn luyện YOLOv4 đầy đủ là rất tốn kém thời gian (xấp xỉ 1 tuần trên Nvidia Titan X), ngay cả khi huấn luyện trên Google Colab. Đồng thời nhóm cũng sử dụng đầu vào có kích cỡ nhỏ hơn so với tiêu chuẩn (418x418) của Darknet để tăng tốc quá trình huấn luyện, cụ thể là (256x256). Số bước huấn luyện cũng được giảm đi do số đối tượng cần dự đoán nhỏ (10 đối tượng). Mạng này có 38 layers thay vì 106 như mạng đầy đủ, và có 2 lớp dự đoán. Nhóm dự đoán rằng độ chính xác của mô hình có thể sẽ giảm vì những sự hiệu chỉnh này, tuy nhiên thời gian huấn luyện được cải thiện một cách rất tốt, xuống còn khoảng 5 tiếng, phù hợp để huấn luyện sử dụng Google Colab.

Sau khi xây dựng được bộ nhãn từ định dạng VOC/Pascal sang Yolo, nhóm tiến hành xây dựng và huấn luyện model Yolo trên Google Colab. Quá trình huấn luyện tương đối nhanh và mô hình đạt được kết quả khả quan trên tập huấn luyện. Cấu trúc của mạng được sử dụng được miêu tả dưới đây:

Bảng 6.1 Sơ đồ cấu trúc tiny-YOLOv4

#	**Layers**
0-3	3 x Conv + Route
4-6	2 x Conv + Route
7-9	Conv + Route + MaxPooling
10-11	Conv + Route
12-14	2 x Conv + Route

15-17	Conv + Route + MaxPooling
18-19	Conv + Route
20-22	2 x Conv + Route
23-25	Conv + Route + MaxPooling
26-29	4 x Conv
30 – 31	Detection (YOLO) + Route
32 – 34	Conv + Upsample + Route
35 – 36	2 x Conv
37	Detection (YOLO)

6.2.2 Kết quả huấn luyện mô hình

Sau khi được huấn luyện, dưới đây là kết quả của mô hình YOLO khi chạy kiểm tra trên 9000 ảnh kiểm tra. Trên Hình 6.2 là một số kết quả của mô hình YOLO

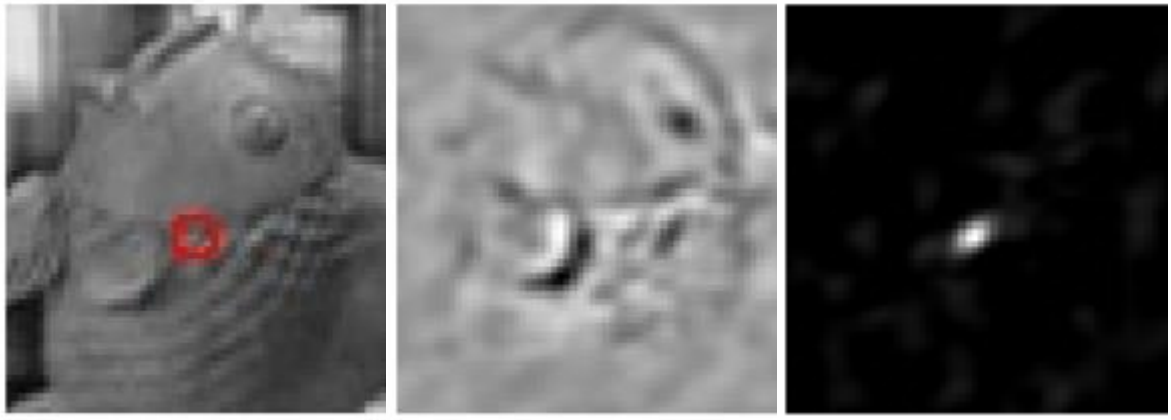
Hình 6.2 Kết quả của mô hình YOLO

- Thời gian xử lý trung bình: 3ms/ảnh
- Độ chính xác trung bình: mAP = 90%

6.3 Object tracking

Trong đề tài này, nhóm sử dụng bộ theo dấu vật thể sử dụng các bộ lọc tương qua n phát triển từ phương pháp MOSSE (Minimum Output Sum of Square Error) ([1], [2]).

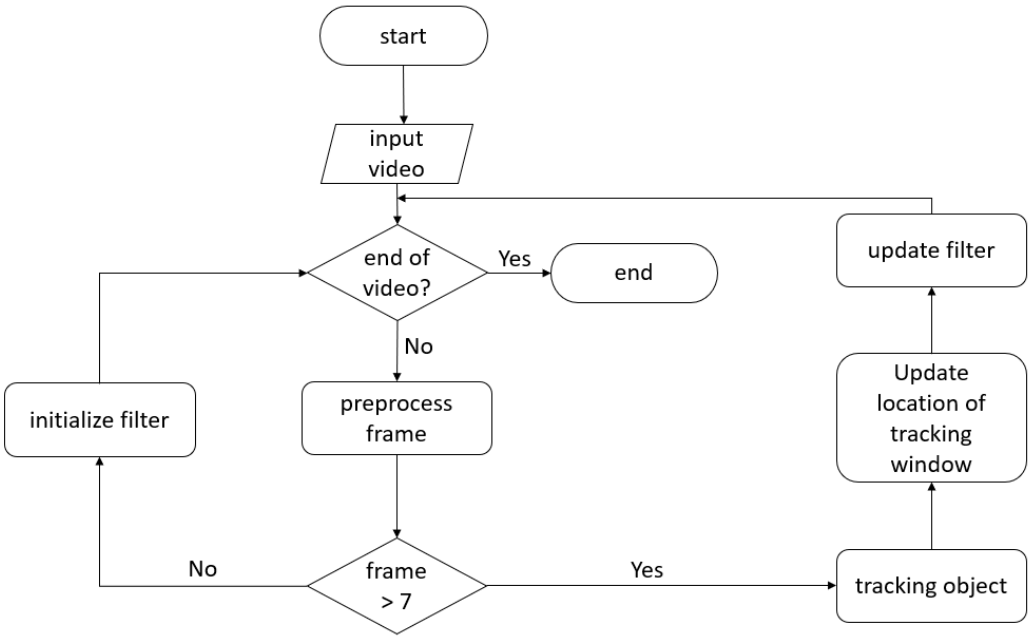
Ý tưởng chủ đạo của phương pháp này như sau: Ở mỗi khung hình, chúng ta trích xuất ra một vùng ảnh có kích thước cố định được gọi là cửa sổ theo dấu (Tracking window). Cửa sổ này sẽ được đem đi tính tương quan với một bộ lọc **h**. Bộ lọc này có một điểm đặc biệt là nếu như có vật thể mà chúng ta muốn theo dấu ở trong cửa sổ, thì vị trí của vật thể sẽ có kết quả giá trị tương quan là cao nhất. Trên Hình 7.1 là ví dụ về bộ lọc tương quan.



a b c

Hình 6.3 a) Cửa sổ theo dấu b) Bộ lọc tương quan c) Ảnh giá trị tương quan

Vậy vấn đề là làm thế nào để có thể tính được giá trị của bộ lọc này? Trên Hình 6.4 mô tả lưu đồ thuật toán để tính giá trị bộ lọc tương quan.



Hình 6.4 Lưu đồ thuật toán theo dấu vật thể

Giả sử chúng ta có một tập gồm N dữ liệu huấn luyện f_i (ví dụ như Hình 6.3a) và một tập các dữ liệu mục tiêu g_i (ví dụ như Hình 6.3c) với $i = \overline{1,N}$. Khi đó bộ lọc tương quan $\boldsymbol{h}$ cần tìm phải thỏa mãn:

$$h = \arg\min_{h} \sum_{i}^{N} |f_i \star h - g_i|^2 \ (1)$$

Trong đó $\star$ là toán tử tương quan. Xử lý trực tiếp với toán tử tương quan là tương đối phức tạp, vì vậy để đơn giản cho việc tính toán, chúng ta chuyển qua miền tần số bằng phép biến đổi DFT (Discrete Fourier Transform), khi đó phương trình trên trở thành:

$$H^* = \arg\min_{H^\star} \sum_{i}^{N} |F_i \cdot H^* - G_i|^2 \ (2)$$

Trong đó F_i, G_i, H lần lượt là biến đổi Fourier của f_i, g_i, h, $*$ là toán tử lấy liên hợp phức và $\cdot$ là toán tử nhân tương ứng từng phần tử. Bằng một số phép biến đổi, nghiệm của bài toán trên có thể được tính theo công thức:

$$H^* = \frac{\sum_i G_i \cdot F_i^*}{\sum_i F_i \cdot F_i^*} = \frac{A}{B} \ (3)$$

Với:

$$A = \sum_i G_i \cdot F_i^* \quad (4)$$

$$B = \sum_i F_i \cdot F_i^* \quad (5)$$

Các mẫu f_i có thể được lấy từ một vài khung hình đầu tiên, trong lưu đồ trên Hình 6.4 lựa chọn N = 7, còn g_i là do chúng ta tự định nghĩa. Số lượng khung hình cần sử dụng N có thể bằng 1 tuy nhiên kết quả sẽ kém ổn định hơn.

Để tránh trường hợp B bằng 0 người ta cộng thêm một giá trị λ gọi là hệ số chính quy, khi đó giá trị của H^* là:

$$H^* = \frac{\sum_i G_i \cdot F_i^*}{\sum_i F_i \cdot F_i^* + \lambda} \quad (6)$$

Để tăng tính ổn định, A và B được cập nhật theo công thức:

$$A_i = \eta(G_i \cdot F_i^*) + (1 - \eta)A_{i-1} \quad (7)$$

$$B_i = \eta(F_i \cdot F_i^* + \lambda) + (1 - \eta)B_{i-1} \quad (8)$$

Với η là tốc độ học và có giá trị trong khoảng [0,1].

Như vậy, từ một tập các mẫu huấn luyện, ta tìm được một bộ hệ số cho bộ lọc tương quan $\boldsymbol{h}$, đây được coi là giá trị khởi tạo cho bộ lọc. Vậy khi một khung hình mới vào thì làm thế nào để chúng ta có thể phát hiện được vị trí mới của vật thể?

Giả sử ta biết được vị trí của cửa sổ theo dấu vật thể ở khung hình thứ $k-1$ tại tọa độ x_{k-1}, y_{k-1}, khi đó vị trí của vật thể ở trong khung hình thứ k là:

$$x_k, y_k = \arg\max_{x,y} DFT^{-1}\left(\frac{A_{k-1} \cdot Z}{B_{k-1} + \lambda}\right) \quad (9)$$

Trong đó Z là biến đổi Fourier của cửa sổ theo dấu lấy trên khung hình thứ k tại vị trí x_{k-1}, y_{k-1}. Tương tự, tọa độ của vật thể trong khung hình thứ $k+1$ sẽ được tính theo cửa sổ lấy trên khung hình thứ $k+1$ tại vị trí x_k, y_k. Hệ số của bộ lọc tương quan cũng sẽ được cập nhật lại như công thức số (7) và (8).

7 Tổng kết

Trong đề tài này, nhóm thu được một số kinh nghiệm như sau:

- Vận dụng được các kiến thức đã học về Deep Learning, Computer Vision để xây dựng được một mô hình áp dụng cho bài toán phát hiện, nhận diện vật thể và theo bám vật thể.
- Nắm được quy trình giải quyết một vấn đề cụ thể, từ chuẩn bị dữ liệu cho đến xử lý.
- Các kỹ năng mềm khác như làm việc nhóm, thuyết trình…

Trong quá trình thực hiện đề tài, vẫn còn một số khó khăn mà nhóm gặp phải đó là:

- Sự mất cân bằng giữa các lớp dữ liệu trong tập dữ liệu. Mặc dù nhóm đã cố gắng thu thập thêm cũng như áp dụng một số phương pháp tự động làm giàu dữ liệu nhưng cũng mới chỉ giải quyết được phần nào.
- Quá trình huấn luyện mô hình mất nhiều thời gian và đòi hỏi tài nguyên tính toán lớn.

8 Đề xuất cải thiện đề tài

Nhóm đưa ra một số đề xuất để cải tiến tiếp đề tài như sau:

- Tiếp tục tìm kiếm, làm giàu hơn dữ liệu để giảm thiểu sự mất cân bằng giữa các lớp dữ liệu.
- Hiện tại với bài toán theo dấu vật thể mới chỉ dừng ở mức theo dấu trong một camera, có thể nghiên cứu các phương pháp để mở rộng bài toán ra với nhiều camera (multiview)

9 Danh mục tài liệu tham khảo

[1] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, Yui Man Lui. *Visual Object Tracking using Adaptive Correlation Filters*

[2] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, Michael Felsberg. *Accurate Scale Estimation for Robust Visual Tracking*