

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN VIỄN THÔNG

-----o0o-----



LUẬN VĂN TỐT NGHIỆP

PHÁT HIỆN VÀ GIÁM SÁT ĐỐI TƯỢNG
THÔNG QUA THIẾT BỊ BAY KHÔNG NGƯỜI LÁI

GVHD: GS. TS. Lê Tiến Thường

SVTH:

Nguyễn Hoàng Ân 1410167

Phan Xuân Bách 1410178

TP. HỒ CHÍ MINH, THÁNG 9 NĂM 2020

LỜI CẢM ƠN

Giai đoạn làm đề tài tốt nghiệp là giai đoạn cuối cùng của quãng thời gian đại học. Có lẽ với nhiều người quãng thời gian này trôi qua khá dễ dàng, nhưng đối với một số, đó sẽ là một đoạn thời gian khó khăn trong cả suy nghĩ và tâm lý. Mất 6 năm trong ngưỡng cửa đại học, gặp rủi ro so với bình thường, chúng em đã cố gắng rất nhiều để đi tới được bước cuối cùng này trước khi vụt mất cơ hội cuối cùng bước ra khỏi ngưỡng cửa đại học với tấm bằng trong tay. Nhiều lúc tưởng như đã buông bỏ tất cả, nhưng sự kỳ vọng của mọi người xung quanh, người thân, gia đình lại vực dậy chúng em.

Xin gửi lời cảm ơn chân thành đến thầy hướng dẫn của chúng em, GS. TS. Lê Tiến Thường, thầy đã thật sự rất kiên nhẫn với chúng em trong suốt nhiều học kỳ mà không than vãn bất cứ điều gì, thầy cũng dạy cho chúng em nhiều điều để nghiên cứu, học tập, cũng như đôn đốc, giúp đỡ chúng em trong suốt quá trình làm đồ án, thực tập và cả luận văn như hiện tại.

Cũng xin gửi lời cảm ơn sâu sắc đến gia đình, người thân, bạn bè, thầy cô và tất cả mọi người quan tâm, giúp đỡ, đặt kỳ vọng cũng như an ủi, động viên đến chúng em để chúng em có thể duy trì tâm thái của bản thân để đối mặt với mọi khó khăn, trở ngại.

Một lần nữa, chúng em muốn gửi lời cảm ơn trân trọng nhất đến tất cả mọi người.

Tp. Hồ Chí Minh, ngày 20 tháng 12 năm 2019.

Sinh viên

MỤC LỤC

TÓM TẮT LUẬN VĂN	1
CHƯƠNG 1: GIỚI THIỆU	2
1.1 Tổng quan.....	2
1.2 Tình hình nghiên cứu trong và ngoài nước.....	5
1.3 Nhiệm vụ luận văn.....	7
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	8
2.1 Xử lý ảnh, xử lý video thời gian thực.....	8
2.2 Phát hiện đối tượng.	8
2.2.1 Định nghĩa phát hiện đối tượng	8
2.2.2 Thuật toán nhận dạng đối tượng	9
2.2.3 So sánh các mô hình nhận dạng đối tượng phổ biến	13
2.2.3.1 Regions with CNN features (R-CNN).....	13
2.2.3.2 YOLO (You only look once).....	15
2.2.4 Nhận diện khuôn mặt.....	22
2.2.4.1 SVM (Support Vector Machines) [15]	22
2.2.4.2 Face Recognition với Facenet [16].....	24
2.2.4.3 SSD dựa trên Resnet. [17]	28
2.3 Bài toán theo vết đối tượng trong xử lý thời gian thực.....	31
2.3.1 Định nghĩa bài toán theo vết đối tượng	31
2.3.2 Chính xác hóa đối tượng tương ứng (Object matching).....	32
2.3.3 Xử lý nhập nhằng (Occlusion)	32
2.3.4 Dự đoán chuyển động.....	33

2.4 Kỹ thuật video streaming [39]	33
2.4.1 Giao thức RTSP	36
2.4.2 Giao thức RTP	38
2.5 Thiết bị bay không người lái	41
2.5.1 Tổng quan về thiết bị bay không người lái	41
2.5.2 Cấu trúc một UAV cơ bản	43
2.5.3 Lý thuyết về hành vi bay của thiết bị bay UAV	47
CHƯƠNG 3: THIẾT KẾ VÀ THỰC HIỆN MÔ HÌNH	49
3.1 Mô hình với drone tự lắp ráp, cân chỉnh hệ thống cân bằng và ổn định bay	49
3.1.1 Tổng quan mô hình và ý tưởng thực hiện	49
3.1.2 Thử nghiệm và tiến hành cân chỉnh hệ thống	52
3.1.3 Can thiệp quá trình truyền tín hiệu giữa RX701 và MC	58
3.1.3.1 Tạo mạng Wifi cục bộ để truyền tín hiệu với Raspberry Pi	60
3.1.3.2 Mô phỏng tín hiệu PWM điều khiển bay thông qua Arduino Uno	64
3.1.4 Truyền video thời gian thực từ Raspberry về máy tính	71
3.1.4.1 Thư viện truyền FFMPEG	71
3.1.4.2 Thư viện truyền GStreamer	72
3.1.5 Lưu đồ giải thuật	75
3.2 Mô hình với drone hoàn thiện bởi nhà sản xuất DM107s	78
3.2.1 Drone DM107s	78
3.2.2 Ý tưởng cho mô hình	79
3.2.3 Can thiệp kết nối giữa drone và điện thoại	80
3.2.4 Phân tích gói dữ liệu gửi từ điện thoại về drone	83
CHƯƠNG 4: KẾT QUẢ THỰC HIỆN VÀ HƯỚNG PHÁT TRIỂN	90
4.1 Kết quả thực hiện	90
4.1.1 Giao diện giám sát và các phím chức năng	90
4.1.2 Mô hình với drone tự lắp ráp	91

4.1.3	Mô hình với drone hoàn thiện bởi nhà sản xuất	94
4.1.4	So sánh các mô hình nhận dạng	95
4.1.5	So sánh 2 mô hình đã thực hiện.....	96
4.2	Nhận xét ưu khuyết điểm.....	98
4.3	Hướng phát triển của đề tài.....	99
	TÀI LIỆU THAM KHẢO.....	100
	PHỤ LỤC CODE SỬ DỤNG TRONG BÀI.....	105
	Code điều khiển bay (Python)	105
	Code chạy trên Raspberry Pi (Python)	119
	Code trên Arduino Uno (C).....	121

DANH SÁCH HÌNH

Hình 1.1 Lịch sử các cuộc cách mạng công nghiệp.	2
Hình 1.2 Tìm kiếm đối tượng bằng hình ảnh trên Google Images.....	3
Hình 1.3 Dự báo về thị trường drone.	5
Hình 2.1 Phát hiện đối tượng.....	9
Hình 2.2 Phương pháp cửa sổ trượt (Sliding Window).....	10
Hình 2.3 Phương pháp đề xuất vùng (Region Proposal) [5].	10
Hình 2.4 Nhiều hộp giới hạn cùng bao lấy một đối tượng [5].	11
Hình 2.5 Công thức tính chỉ số IoU [6].....	12
Hình 2.6 Kết quả của chỉ số IoU đối với các hộp giới hạn khác nhau [6].	12
Hình 2.7 Cấu trúc mạng CNN hai lớp tích chập [7].....	13
Hình 2.8 Cấu trúc mạng R-CNN [13].	14
Hình 2.9 Ví dụ về cách tính tọa độ, kích thước hộp giới hạn cho lưới 3x3 [34].....	16
Hình 2.10 Cấu trúc mạng YOLO 24 lớp tích chập [14].....	17
Hình 2.11 Ví dụ cho dự đoán hộp giới hạn 25 thông số trong một hộp [12].....	18
Hình 2.12 Hộp giới hạn (nét liền), hộp anchor (nét đứt) và dự đoán vị trí [9].....	19
Hình 2.13 Cấu trúc mạng YOLOv3 [35].....	20
Hình 2.14 Lưới cho từng tỉ lệ hình [36].	20
Hình 2.15 Nội dung bên trong một ô của bản đồ đặc trưng [35].	20
Hình 2.16 Ví dụ về bài toán SVM.....	23
Hình 2.17 Siêu phẳng và các Support Vector	23
Hình 2.18 Tác dụng của hàm lỗi Triplet Loss [18] tối thiểu hóa khoảng cách đến phần dương, tối đa hóa khoảng cách đến phần âm	25
Hình 2.19 Cách Facenet (phần deep learning) hoạt động [38].....	25
Hình 2.20 Bước tiền xử lý của Facenet.....	26
Hình 2.21 Hoạt động của Facenet.	26
Hình 2.22 Mô hình SSD.	29
Hình 2.23 Độ phân giải nhỏ hơn của bản đồ đặc trưng giúp phát hiện vật lớn hơn.	30

Hình 2.24 Sơ đồ thực hiện bài toán theo vết đối tượng trong video	31
Hình 2.25 Sự chính xác hóa đối tượng	32
Hình 2.26 Kỹ thuật Streaming Video	34
Hình 2.27 Header của RTP Packet.	40
Hình 2.28 Thiết bị bay không người lái với rất nhiều hình dạng và cấu trúc.	41
Hình 2.29 Cấu tạo Drone cơ bản [41]	43
Hình 2.30 Trục dịch chuyển của IMU (3 trục phẳng và 3 trục xoay) [43]	45
Hình 2.31 Hoạt động đối xứng nhau của động cơ giúp thiết bị có khả năng bay	47
Hình 2.32 Nguyên lý di chuyển của mô hình Quacopter [45]	48
Hình 3.1 Sơ đồ hệ thống dự tính thực hiện	49
Hình 3.2 Cách kết nối các linh kiện theo hướng dẫn của nhà sản xuất.....	51
Hình 3.3 Mô hình UAV dùng cho đề tài.	51
Hình 3.4 Tay cầm điều khiển DEVO 7	53
Hình 3.5 GPS Module của Naza.....	54
Hình 3.6 Cài đặt GPS cho thiết bị bay [48].	55
Hình 3.7 Tinh chỉnh hướng của GPS Module [48]	55
Hình 3.8 Các chế độ bay của thiết bị [48]	56
Hình 3.9 Đèn tín hiệu khi bay ở các chế độ [48].....	57
Hình 3.10 Sơ đồ cho việc truyền tín hiệu	58
Hình 3.11 Tín hiệu PWM điều khiển động cơ tương tự tín hiệu nhận được từ tay cầm điều khiển	59
Hình 3.12 USB wifi của Samsung Smart TV	61
Hình 3.13 Router Tenda AC10.....	62
Hình 3.14 Chi tiết vị trí chân trên Arduino Uno R3 (Đen: GND; Vàng: chân theo PORTD, PIND, PORTB, PINB; Hồng: chân theo chương trình IDE; Xám: chân theo interrupt ngoài) [50].....	65
Hình 3.15 Nối Arduino với RX701 và mạch điều khiển bay [51]	69
Hình 3.16 Lưu đồ giải thuật cho việc chuyển tiếp dữ liệu	75
Hình 3.17 Lưu đồ giải thuật nhận dạng.....	76
Hình 3.18 Lưu đồ giải thuật điều khiển bay	77
Hình 3.19 Drone DM107s	78

Hình 3.20 Ý tưởng cho mô hình 2.....	79
Hình 3.21 Sơ đồ xử lý của mô hình 2.....	89
Hình 4.1 Giao diện người dùng	90
Hình 4.2 Kết quả nhận dạng khuôn mặt (Mô hình 1).....	92
Hình 4.3 Kết quả bay theo sát đối tượng (Mô hình 1)	93
Hình 4.4 Kết quả nhận dạng khuôn mặt (Mô hình 2).....	94
Hình 4.5 Tracking người dựa trên diện tích hộp giới hạn (Mô hình 2).....	95

DANH SÁCH BẢNG

Bảng 1. Bảng so sánh tốc độ các mô hình nhận dạng đối tượng qua tập test PASCAL VOC 2007	21
Bảng 2. Thông số hệ thống.....	50
Bảng 3. Gói UDP và công dụng.	84
Bảng 5. Tốc độ xử lý các mô hình nhận dạng thực tế	96
Bảng 6. So sánh 2 mô hình đã thực hiện	97

TÓM TẮT LUẬN VĂN

Trong những năm gần đây, xử lý ảnh số đã đạt được nhiều thành tựu và tiến bộ vượt bậc. Trong đó, nhận dạng và phân loại ảnh là một trong những hướng nghiên cứu được theo đuổi một cách tích cực nhất. Nhờ hệ thống xử lý ảnh với lượng lớn dữ liệu đào tạo cho trí tuệ nhân tạo, con người đã giảm được một khối lượng lớn công việc cũng như gia tăng sự chính xác trong việc đưa ra những quyết định liên quan đến xử lý hình ảnh trong nhiều lĩnh vực như quân sự và quốc phòng, sinh hóa và giải phẫu, hệ thống giao thông,... Trong các lĩnh vực đó, việc giám sát an ninh là vô cùng quan trọng cho đời sống hằng ngày của người dân cũng như rất tất yếu cho quân sự và quốc phòng. Nắm bắt tình hình đó, nhóm chúng em đã cùng nhau tìm hiểu, xây dựng và phát triển đề tài “Phát hiện và giám sát đối tượng thông qua thiết bị bay không người lái”. Luận văn tập trung vào việc phát triển một thiết bị bay thông minh có thể tìm kiếm, nhận dạng và bám sát theo đối tượng đó, thiết bị này cũng có thể thu thập, lưu trữ, truyền trực tiếp hình ảnh thu được về máy tính để người quan sát theo dõi, điều khiển, đồng thời nó còn có thể học tập và ghi nhớ một đối tượng nhất thời mà người điều khiển chỉ định để bất cứ khi nào gặp lại cũng có thể sẵn sàng theo chân đối tượng này.

CHƯƠNG 1: GIỚI THIỆU

1.1 Tổng quan

1.2 Tình hình nghiên cứu trong và ngoài nước

1.3 Nhiệm vụ luận văn

1.1 Tổng quan

Thế giới đã bước vào thời kì bùng nổ thông tin số với cuộc cách mạng công nghiệp 4.0 đang diễn ra rầm rộ tại các nước phát triển như Mỹ, châu Âu, một phần châu Á. Cách mạng công nghiệp đầu tiên sử dụng năng lượng nước và hơi nước để cơ giới hóa sản xuất. Cuộc cách mạng lần 2 diễn ra nhờ ứng dụng điện năng để sản xuất hàng loạt. Cuộc cách mạng lần 3 sử dụng điện tử và công nghệ thông tin để tự động hóa sản xuất [1]. Bây giờ, cuộc Cách mạng Công nghiệp Thứ tư đang nảy nở từ cuộc cách mạng lần ba, nó kết hợp các công nghệ lại với nhau, làm mờ ranh giới giữa vật lý, kỹ thuật số và sinh học [2].



Hình 1.1 Lịch sử các cuộc cách mạng công nghiệp.

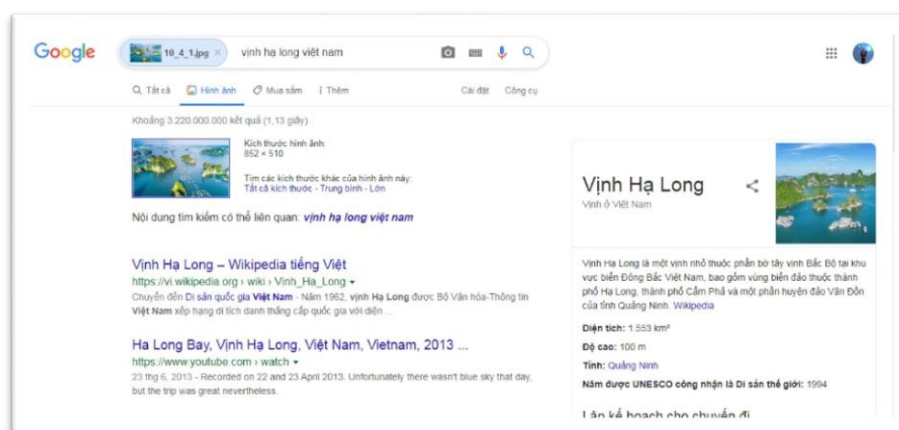
Những yếu tố cốt lõi của Kỹ thuật số trong cách mạng công nghiệp 4.0 sẽ là: Trí tuệ nhân tạo (AI), Vạn vật kết nối - Internet of Things (IoT) và dữ liệu lớn (Big Data).

Dưới tác động của cuộc cách mạng công nghiệp 4.0, các thiết bị ngày càng trở nên thông minh hơn, cuộc sống con người ngày càng dễ dàng và được nâng cao về chất lượng với các thiết bị thông minh hỗ trợ phần lớn công việc hằng ngày. Các

thiết bị này phần lớn đều được cung cấp chương trình trí tuệ nhân tạo (AI), chúng biết “học” để trở nên gần gũi với người dùng hơn, thao tác chính xác và đạt hiệu suất cao nhất.

Lĩnh vực xử lý hình ảnh cũng ngày càng được chú trọng và quan tâm nhiều hơn do sự ứng dụng cao của nó và thực tế là, dường như mọi lĩnh vực trong đời sống đều được tích hợp việc xử lý ảnh số.

Ví dụ như Google, họ đã phát triển công nghệ tìm kiếm bằng hình ảnh Google Images (tính năng này được hoàn thành vào tháng 12 năm 2001), ở ứng dụng này, để tìm kiếm một thứ gì đó trên công cụ tìm kiếm, thay vì phải nhập văn bản bằng cách gõ ký tự, giờ đây chỉ cần chụp ảnh từ điện thoại di động, thực hiện tìm kiếm theo hình ảnh, nội dung trong bức ảnh – các ngọn núi, sông hồ, cây cối, thậm chí cả động vật, máy móc, thiết bị,... cũng sẽ được tìm kiếm các nội dung tương tự từ kho hình ảnh khổng lồ ở máy chủ google cũng như hình ảnh từ các trang web đăng tải. Khi tìm kiếm với công cụ này, một hình ảnh thu nhỏ có nội dung khớp nhất với hình ảnh được tìm kiếm sẽ hiện bên dưới kèm theo đường dẫn đến trang web chứa hình ảnh được tìm thấy. Công cụ này giúp ích rất nhiều cho việc tìm kiếm những địa điểm, đồ vật mà mọi người không biết tên gọi của chúng.



Hình 1.2 Tìm kiếm đối tượng bằng hình ảnh trên Google Images.

Sự đột phá từ việc tìm kiếm bằng hình ảnh mang lại bước tiến bộ rõ rệt cho ứng dụng xử lý ảnh việc tìm kiếm và hứa hẹn sẽ còn phát triển mạnh mẽ hơn cho các ứng dụng tìm kiếm bằng hình ảnh 3D hay việc tìm kiếm bằng video thời gian thực trong những năm tới [3].

Một nghiên cứu khác của Jain, Thakur và Suresh [4] ở đại học Nam California (University of Southern California – USC) đã thành công phát triển mô hình hỗ trợ cho người khiếm thị để họ có thể di chuyển an toàn dù không nhìn thấy gì. Với việc ứng dụng xử lý ảnh, mô hình gồm bộ vest và kính đen này được trang bị cho người khiếm thị, khi họ di chuyển, hệ thống sẽ tạo ra mô hình ba chiều của môi trường xung quanh bằng các camera gắn liền với kính đen. Các camera này quan sát các chướng ngại vật xung quanh, đưa tín hiệu về xử lý và ra lệnh cho các động cơ rung gắn trong áo vest hoạt động. Giả dụ như người khiếm thị nên di chuyển qua trái để tránh chướng ngại vật, khi đó, động cơ bên trái sẽ rung để ra hiệu cho người này di chuyển về bên phải an toàn. Việc phát triển mô hình này có thể giúp đỡ cho rất nhiều người khiếm thị, vì thế nó mang một giá trị nhân văn sâu sắc. Nó mang lại niềm tin và hy vọng cho người khuyết tật, rằng xã hội luôn quan tâm đến họ.

Trong lĩnh vực an ninh quốc phòng, xử lý ảnh – nhận dạng khuôn mặt, nhận dạng đối tượng cũng được quan tâm đến trong thời gian gần đây. Bill Casey – người quản lý chương trình của Trung tâm Sinh học FBI, cho biết rằng “Xử lý ảnh cũng là một công nghệ quan trọng trong các cuộc điều tra” [27].

“Khi săn lùng một tên tội phạm, chúng tôi xem qua cơ sở dữ liệu của chúng tôi với hơn chín triệu bức ảnh để tìm kiếm các đầu mối có thể điều tra” – ông Caysey chia sẻ.

Xử lý hình ảnh cũng làm thay đổi chiến tranh trong thập kỉ qua với sự xuất hiện của các thiết bị bay không người lái được điều khiển từ xa và truyền hình ảnh về căn cứ để xử lý từ xa. Tiến sĩ Richard Baraniuk, giáo sư kỹ thuật tại Đại học Rice - một tổ chức nghiên cứu tư nhân – nói rằng, trong vòng 15 năm tới hoặc ít hơn, hãy xem xét tạo ra một phép màu.

Qua đó ta cũng có thể thấy được rằng xử lý ảnh sẽ là một lĩnh vực phát triển cực kì mạnh trong tương lai gần.

Lấy một ví dụ cụ thể hơn cho camera giám sát dân dụng, nếu hiện tại hệ thống chỉ có thể thông báo rằng có đối tượng đang tiếp cận ngôi nhà, thì trong tương lai

vài năm tới, hệ thống có thể chỉ ra rõ ràng đối tượng đó có phải người trong gia đình hay không để cảnh báo tới chủ nhân ngôi nhà.

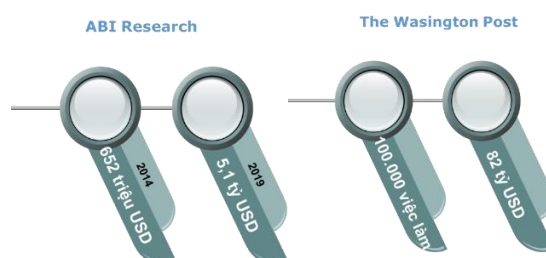
Dễ dàng có thể khẳng định rằng, qua thời gian, các thiết bị sẽ ngày càng thông minh hơn với việc tích hợp xử lý ảnh và trí tuệ nhân tạo AI.

1.2 Tình hình nghiên cứu trong và ngoài nước

Cùng với cuộc cách mạng công nghệ bùng nổ, drone dân dụng đang được đón chào khắp nơi. Trước đây, ngoài ứng dụng chuyên biệt trong quân sự, drone dân dụng chủ yếu được biết nhiều là dạng gắn camera (thường gọi flycam) cho mục đích quay phim, chụp ảnh từ trên không. Nhưng ngày nay drone đã phát triển vượt bậc, thâm nhập sâu rộng vào cuộc sống, từ thú chơi tiêu khiển hấp dẫn những người đam mê chinh phục bầu trời, cho đến bay phục vụ công việc của các tổ chức, doanh nghiệp, người dùng cá nhân và dĩ nhiên không thể thiếu là thực hiện những nhiệm vụ đặc biệt trong quân sự.

Ban đầu, chúng được dùng chủ yếu cho mục đích quân sự, như bay trinh sát với khả năng chụp không ảnh, truyền hình ảnh về căn cứ chỉ huy, hay tìm diệt những mục tiêu khó tiếp cận bằng vũ khí mang theo. Dần dần UAV ngày càng được sử dụng sâu rộng cho mục đích dân sự, từ việc đáp ứng thú chơi tiêu khiển của người dùng, cho đến giao hàng hóa, rải phân, tưới cây chăm sóc mùa màng, theo dõi đàn gia súc, giám sát rừng, vườn thú hoang dã, quay phim, chụp ảnh từ trên cao, cứu hộ cứu nạn những nơi hiểm trở...

Theo dự báo của ABI Research, thị trường drone thương mại cỡ nhỏ có qui mô khoảng 652 triệu USD trong năm 2014 sẽ tăng lên tới hơn 5,1 tỷ USD vào năm 2019, ước tính gấp đôi thị trường quốc phòng [26].



Hình 1.3 Dự báo về thị trường drone.

Drone có nhiều kích cỡ, từ nhỏ gọn có thể cầm trên tay, bay trong tầm quan sát được, cho đến những loại đồ sộ như máy bay thật giá trị hàng triệu USD với tầm bay xa hàng trăm kilomet. Drone dần dụng trở nên phổ biến nhờ những tiến bộ vượt bậc trong công nghệ và giá thành giảm nhanh, dễ sử dụng. Hiện tại, bạn chỉ cần bỏ ra vài trăm ngàn cho đến vài triệu đồng là đã có thể sở hữu một chiếc drone loại nhỏ với tay cầm điều khiển, thỏa mãn thú vui chinh phục bầu trời của mình. Drone dùng trong hoạt động kinh doanh, sản xuất cao cấp hơn, giá bán dĩ nhiên cũng cao hơn nhiều, nhưng tỏ ra hứa hẹn đem lại hiệu quả cao trong sử dụng.

Để đảm bảo drone bay an toàn, không gây ảnh hưởng an ninh, tránh bị lợi dụng dẫn đến vi phạm quyền riêng tư, nhiều quốc gia có những qui định hạn chế với hoạt động bay của Drone. Chẳng hạn, Cơ quan quản lý Hàng không Mỹ (FAA) quy định Drone dân dụng chỉ được bay ở độ cao dưới 122 mét với vận tốc bay tối đa 161 km/giờ, trong tầm mắt thấy được, và phải cách xa sân bay tối thiểu 8 km để tránh gây nguy hiểm cho máy bay có người lái [28].

Còn tại Việt Nam, hoạt động drone tuân theo Nghị định 36/2008/NĐ-CP về việc quản lý máy bay không người lái và các phương tiện bay siêu nhẹ, tiếp theo là Nghị định 79/2011/NĐ-CP do Chính phủ ban hành năm 2011 sửa đổi, bổ sung một số điều của Nghị định năm 2008.

Tuy nhiên, drone vẫn được dùng vô tư nhiều nơi cho nhiều mục đích khác nhau. Các chủ sở hữu có thể ‘quên’ hoặc thậm chí không biết là cần phải xin phép trong nhiều trường hợp.

Theo một ước tính mà tờ The Washington Post đăng tải, ngành công nghiệp Drone sẽ tạo ra 100.000 việc làm, đem lại lợi ích kinh tế 82 tỷ USD trong một thập kỷ tới [29].

Tuy nhiên, xét trên khía cạnh quyền riêng tư cá nhân, drone dễ bị xem là con ác mộng. Chúng có thể bị chủ nhân lạm dụng để thực hiện quan sát những khu vực nhạy cảm, làm dấy lên mối lo ngại xâm phạm quyền riêng tư khắp nơi. Drone có thể được trang bị vũ khí, thực hiện nhiệm vụ tấn công từ xa, trong khi người điều khiển ngồi ung dung nhấm nháp cà phê ở đâu đó trong một căn phòng mát lạnh.

Vì thế, giám sát đối tượng bằng thiết bị bay không người lái đã số được phát triển trong lĩnh vực quân sự, còn trong dân dụng thì hạn chế được nhắc đến.

1.3 Nhiệm vụ luận văn

Nội dung 1: Lý thuyết về xử lý ảnh số, xử lý video trong thời gian thực.

Ở nội dung này, chúng em tập trung nắm vững lý thuyết về xử lý ảnh và video, đặc biệt là xử lý trong thời gian thực và dùng camera động.

Nội dung 2: Thuật toán phát hiện đối tượng.

Tìm hiểu, so sánh các thuật toán, các mô hình phát hiện đối tượng đang phổ biến hiện nay. Nắm rõ sự khác nhau giữa chúng, ưu và khuyết điểm, từ đó chọn mô hình phù hợp cho đề tài.

Nội dung 3: Bài toán theo vết đối tượng.

Cách xử lý việc nhận dạng đối tượng, theo vết đối tượng trong video thời gian thực. Các vấn đề cần lưu ý ở bài toán này cũng như hướng khắc phục.

Nội dung 4: Thiết bị bay không người lái.

Nắm cơ bản về tình hình phát triển và ứng dụng của các loại thiết bị bay không người lái.

Nội dung 5: Cách lập trình bay, lý thuyết điều khiển hệ thống cân bằng hành vi bay.

Nắm được cấu trúc của một thiết bị bay cơ bản, nguyên tắc hệ thống cân bằng, các hành vi bay để ứng dụng viết chương trình điều khiển tự động.

Nội dung 6: Theo chân đối tượng – Sự kết hợp giữa nhận dạng, theo vết đối tượng qua video và giám sát đối tượng tự động bằng thiết bị bay không người lái.

Hoàn thiện mô hình thực tế, phát triển thêm các chức năng nếu có thể.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Xử lý ảnh, xử lý video thời gian thực

2.2 Phát hiện đối tượng

2.3 Bài toán theo vết đối tượng trong xử lý thời gian thực

2.4 Kỹ thuật video streaming

2.5 Thiết bị bay không người lái

2.1 Xử lý ảnh, xử lý video thời gian thực

Xử lý ảnh là một phân ngành trong xử lý số tín hiệu với tín hiệu xử lý là ảnh. Đây là một phân ngành khoa học mới rất phát triển trong những năm gần đây. Xử lý ảnh gồm 4 lĩnh vực chính: xử lý nâng cao chất lượng ảnh, nhận dạng ảnh, nén ảnh và truy vấn ảnh. Sự phát triển của xử lý ảnh đem lại rất nhiều lợi ích cho cuộc sống của con người.

Ngày nay xử lý ảnh đã được áp dụng rất rộng rãi trong đời sống như: photoshop, nén ảnh, nén video, nhận dạng biển số xe, nhận dạng khuôn mặt, nhận dạng chữ viết, xử lý ảnh thiên văn, ảnh y tế,...

Video là một tập tin chứa hình ảnh và âm thanh được đồng bộ với nhau, được tạo ra bởi 1 chuẩn nén nào đó, như MPEG, XviD, H264,... Các định dạng phổ biến nhất là MP4, AVI, WMV,... và mới nhất hiện nay là WEBM. Để xử lý hình ảnh từ video thì phải giải mã ra thành những frame hình rồi mới xử lý. Như vậy, xử lý video là việc xử lý một chuỗi hình ảnh liên tiếp nhau qua thời gian.

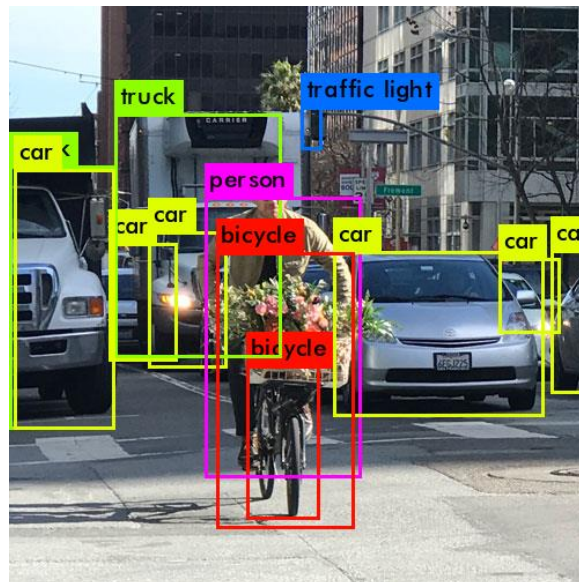
2.2 Phát hiện đối tượng.

2.2.1 Định nghĩa phát hiện đối tượng

Phát hiện đối tượng (Object Detection) là sự kết hợp giữa hai dạng bài toán: Nhận dạng đối tượng (Object Recognition) và Khoanh vùng đối tượng (Object Localization).

Nhận dạng đối tượng là bài toán nhận dạng vật bằng cách đưa một hình ảnh vào mạng Neuron và nhận lại kết quả là một nhãn cùng xác suất chính xác của nhãn đối tượng đó. Ví dụ như, khi ta đưa một hình ảnh con mèo làm đầu vào, kết quả sẽ đưa ra nhãn là “con mèo” và xác suất chính xác của nó là 0.9. Đây sẽ là trọng tâm của thuật toán phát hiện đối tượng vì nó sẽ cho ta biết được đối tượng có hay không có trong hình ảnh đầu vào.

Khoanh vùng đối tượng là bài toán xác định vị trí và vẽ hộp giới hạn bao quanh đối tượng [31]. Thuật toán sẽ vẽ hộp giới hạn và xuất ra kết quả là tọa độ của đối tượng cùng với chiều cao, chiều rộng của hộp giới hạn.



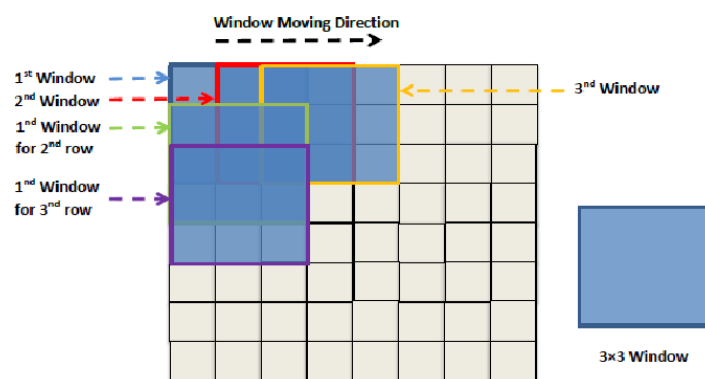
Hình 2.1 Phát hiện đối tượng.

Như vậy, một mô hình phát hiện đối tượng hoàn chỉnh không những nhận dạng đối tượng, xem thử đối tượng có trong hình ảnh hay không, mà còn chỉ ra vị trí chính xác cùng kích thước của đối tượng trong hình ảnh đầu vào.

2.2.2 Thuật toán nhận dạng đối tượng

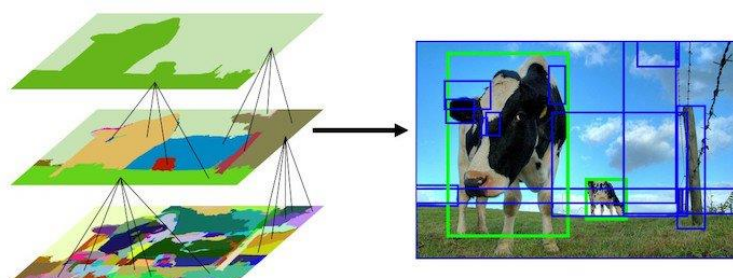
Để phát hiện đối tượng, trước hết ta phải chọn ra phần của ảnh có vật thể và cho thuật toán nhận dạng chạy trên phần đó. Một số cách để chọn ra phần ảnh được tính là có vật thể: Cửa sổ trượt (Sliding Window), đề xuất vùng (Region Proposal).

Phương pháp cửa sổ trượt (Sliding Window) thực hiện bằng việc dịch chuyển một cái hộp hình chữ nhật qua từng bước trong tấm hình. Mỗi khi dịch hộp đi, mô hình sẽ áp dụng thuật toán nhận dạng vào để xác định xem có nhận dạng được vật không. Tuy nhiên phương pháp này rất tốn thời gian và công sức, vì phương pháp này quét từng ô một trong ảnh để tìm vật thể. Không chỉ vậy, do kích thước của hộp không đổi, tỉ lệ khung không đổi theo, nên nếu thay đổi góc chụp ảnh của vật thể cần nhận dạng thì các đặc trưng như hình dạng và tỉ lệ khung thay đổi, dễ khiến việc nhận dạng thất bại [5].



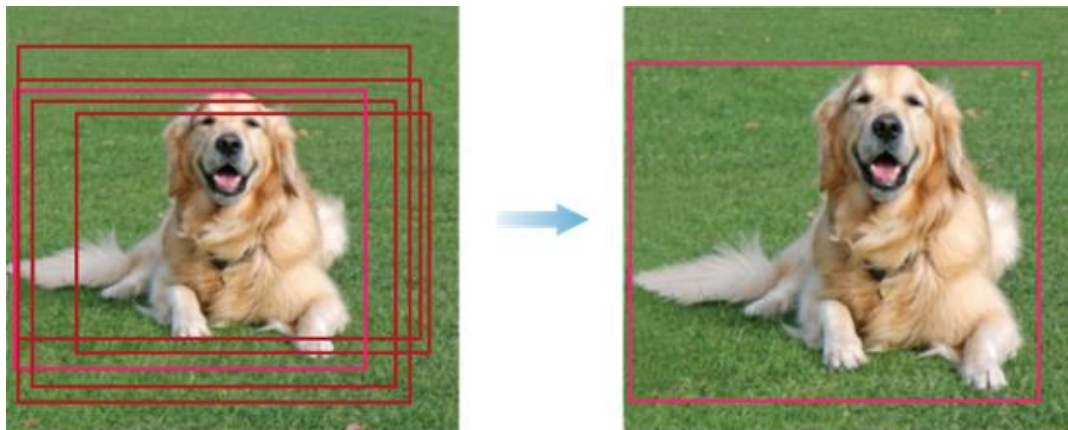
Hình 2.2 Phương pháp cửa sổ trượt (Sliding Window).

Phương pháp đề xuất vùng (Region Proposal) được sử dụng để giải quyết hai nhược điểm của phương pháp cửa sổ trượt. Đầu vào của mô hình sẽ nhận một ảnh chụp và đầu ra sẽ đưa ra các hộp giới hạn trên ảnh mà nó nghi là có vật trong đó. Các hộp giới sẽ đi qua thuật toán nhận dạng như phương pháp trước. Vùng nào có xác suất cao là vật thể nằm trong đó sẽ là vị trí của vật. Phương pháp này phân vùng ảnh thành các vùng lân cận nhau dựa trên sự tương đồng về màu sắc, kết cấu, không như phương pháp cửa sổ trượt là xét từng điểm ảnh. Nhờ vậy nó làm giảm số hộp giới hạn xuống để tăng tốc nhận dạng [5].



Hình 2.3 Phương pháp đề xuất vùng (Region Proposal) [5].

Sau khi có các hộp giới hạn chứa đối tượng, một vấn đề được đặt ra chính là có nhiều hộp giới hạn với các kích thước khác nhau cùng bao lấy một đối tượng trên hình ảnh như hình 2.4 bên trái mô tả. Thực tế là có rất nhiều hộp giới hạn như thế, tuy nhiên, trong hình ảnh chỉ vẽ một ví dụ cơ bản để dễ hình dung hơn về điều này mà không làm rối mắt khi quan sát.



Hình 2.4 Nhiều hộp giới hạn cùng bao lấy một đối tượng [5].

Một thuật toán được xây dựng để giải quyết vấn đề này có tên là Non-max suppression. Thuật toán này có thể trình bày qua 4 bước như sau [30]:


Bước 1: Ta lấy hộp giới hạn có độ tin cậy của class đang xét cao nhất làm chuẩn và cho rằng đây chính là hộp giới hạn của một đối tượng thuộc lớp đó có trong ảnh. Ta tạm gọi là `box_max`.

Bước 2: Xét hộp giới hạn có độ tin cậy cao thứ nhì (khác 0). Nếu chỉ số IoU (Intersection over Union) của hộp đang xét và `box_max` lớn hơn 0.5, ta sẽ xem như hai hộp giới hạn này đang bao phủ cho cùng một đối tượng nên chỉ giữ lại hộp có độ tin cậy cao hơn (`box_max`). Sau đó, độ tin cậy của hộp đang xét được gán về 0. Ngược lại, nếu chỉ số IoU của hai box này nhỏ hơn 0.5, ta sẽ giữ lại cả hai hộp giới hạn.

Bước 3: Lặp lại bước 2 với các hộp giới hạn có độ tin cậy thấp hơn (khác 0). Mục đích là để loại hết tất cả các hộp giới hạn đang bao phủ cùng một đối tượng với `box_max`.

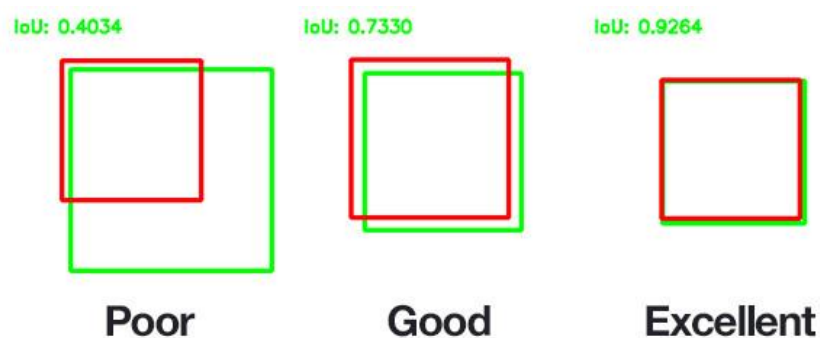
Bước 4: Sau khi loại hết các hộp cùng bao phủ một đối tượng với box_max, ta tiến hành xét duyệt hộp có độ tin cậy khác 0 và không cùng bao phủ một đối tượng với các box_max trước đó, gán nó là box_max. Sau đó tiếp tục lặp lại bước 1 cho đến khi đảm bảo rằng các hộp còn lại chỉ bao phủ một đối tượng riêng biệt thuộc lớp đang xét.

Chỉ số IoU (Intersection over Union)

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Hình 2.5 Công thức tính chỉ số IoU [6].

Chỉ số IoU chỉ đơn giản là một chỉ số đánh giá độ chính xác của việc phát hiện đối tượng. Nhìn vào công thức, ta có thể dễ dàng hiểu được rằng IoU là một tỉ lệ, trong đó tử số là diện tích phần chồng lên nhau của hai hộp giới hạn đang xét, và mẫu số sẽ là tổng diện tích phần hợp của hai hộp giới hạn này [6].



Hình 2.6 Kết quả của chỉ số IoU đối với các hộp giới hạn khác nhau [6].

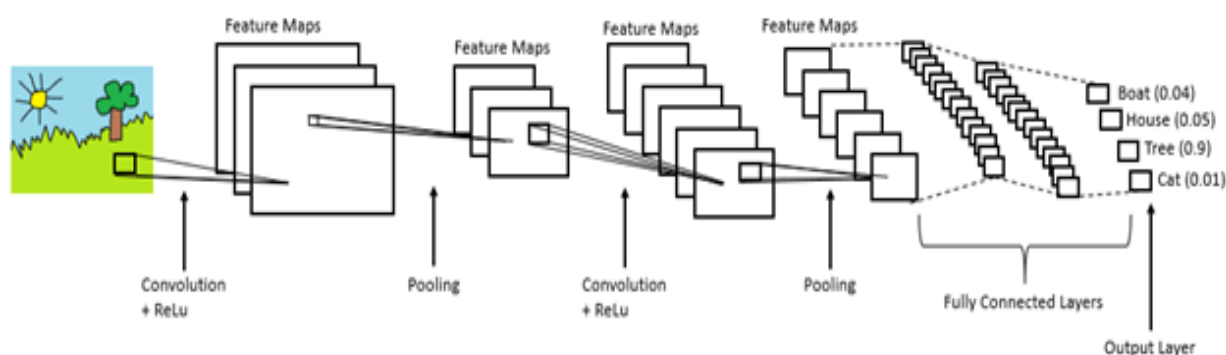
Trong thực tế, các hộp giới hạn được dự đoán đa số đều ít trùng khớp với các mẫu đối tượng được huấn luyện. Vì thế nên cần có một đại lượng để đánh giá sự trùng khớp giữa hộp giới hạn dự đoán và hộp giới hạn thực tế. Và đó là

IoU. Ở đây không xét về tọa độ tâm của các hộp mà chỉ cần đảm bảo rằng giữa dự đoán và mẫu chuẩn càng giống càng tốt.

2.2.3 So sánh các mô hình nhận dạng đối tượng phổ biến

2.2.3.1 Regions with CNN features (R-CNN)

CNN (Convolutional Neural Network – Mạng thần kinh tích chập)



Hình 2.7 Cấu trúc mạng CNN hai lớp tích chập [7].

Mạng neuron tích chập (CNN hay ConvNets và còn được gọi CNNs) là một thể loại chính trong việc nhận diện ảnh, phân loại ảnh, phân loại đối tượng, nhận diện gương mặt và một số lĩnh vực khác [7].

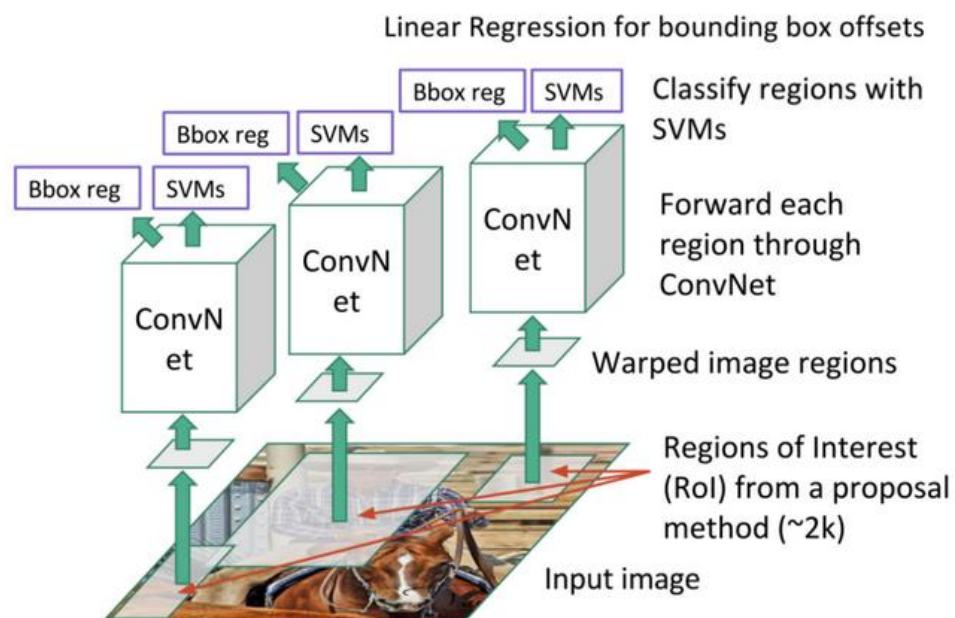
Về mặt kỹ thuật, mô hình học sâu CNN sẽ thực hiện huấn luyện và kiểm, mô hình ảnh đầu vào sẽ chuyển nó qua một loạt các lớp tích chập với các bộ lọc Kernels, sau đó đến Pooling, rồi tiếp theo là các lớp được kết nối hoàn chỉnh (FC - fully connected layers) và sau đó áp dụng chức năng softmax để phân loại một đối tượng có giá trị xác suất từ 0 đến 1. Hình 2.7 mô tả cấu trúc mạng cơ bản của CNN với hai lớp tích chập.

- Lớp Convolution (lớp tích chập) là lớp đầu tiên trích xuất các tính năng từ một hình ảnh, nó duy trì ràng buộc giữa các pixel bằng cách học các tính năng của hình ảnh với nguyên tắc học sử dụng các ô vuông nhỏ của dữ liệu đầu vào. Kết hợp ma trận hình ảnh với các ma trận bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ, và làm sắc nét bằng cách áp dụng các bộ lọc với nhau.

- Lớp Pooling có tác dụng giảm tham số khi hình ảnh quá lớn tuy nhiên vẫn sẽ giữ lại được thông tin quan trọng của hình ảnh.
- Lớp Fully Connected công việc của nó là làm phẳng ma trận thành vector và đưa vào một lớp được kết nối như một mạng lưới thần kinh.
- Cuối cùng sử dụng softmax hoặc sigmoid để phân loại đầu ra là đối tượng nào.

Một số kiến trúc CNN phổ biến như AlexNet, VGGNet, GoogLeNet và ResNet.

Mô hình R-CNN



Hình 2.8 Cấu trúc mạng R-CNN [13].

Hình 2.8 mô tả cấu trúc của một mạng R-CNN, trong đó, vùng RoI được tạo ra từ thuật toán Selective Search. Các vùng này sẽ được chiết suất đặc trưng, tạo thành các vector đặc trưng đại diện cho hình ảnh nhưng có kích thước nhỏ hơn nhiều bằng cách sử dụng mạng thần kinh tích chập (CNN). Tiếp theo, các vector này được đưa vào nhiều SVM cho mỗi đối tượng riêng biệt và kết luận xem đó là đối tượng nào trong tập huấn luyện đã được huấn luyện sẵn.

Như vậy, R-CNN được cấu tạo từ 3 module [31]:

+ Region proposal: phần gợi ý cho mô hình những vùng có khả năng có vật. RCNN sử dụng thuật **Selective search** để tạo ra những đề xuất vùng có vật có thể nhận dạng [32]. Selective search là thuật toán dựa vào gộp nhóm có thứ bậc (hierarchical grouping) dựa vào đặc điểm chung như màu sắc, kết cấu, hình dáng, sau đó thêm vào các hộp giới hạn dựa theo các phân vùng vừa tạo ra [5].

+ Chiết xuất đặc trưng: một mạng CNN lớn được sử dụng để tách ra các đặc trưng trong ảnh thành một vector. Cứ mỗi vùng được tạo ra từ bước 1 sẽ được thay đổi kích thước dữ liệu ảnh để tương thích với CNN được chọn, rồi đi qua mạng một lần để tạo vector [32].

+ Phân loại: vector này được đưa vào nhiều SVM, mỗi SVM là một lớp để phân loại vật thể. Cũng vector đó được đưa vào thuật toán Linear Regressor (hồi quy tuyến tính) để cố định lại tọa độ hộp giới hạn, giảm lỗi xác định vị trí [33].

Tuy nhiên nhược điểm lớn của mạng R-CNN là dự đoán chậm, thường mất 50 giây cho một hình. Việc huấn luyện phức tạp và tốn thời gian do phải huấn luyện cho từng module của mạng.

2.2.3.2 YOLO (You only look once)

YOLOv1

YOLO được Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi [14] phát triển vào năm 2015. Là một mô hình CNN để detect vật thể, YOLO có ưu điểm nổi trội là nhanh hơn nhiều so với những mô hình khác cùng chức năng, thậm chí chạy được trên các thiết bị như Raspberry Pi [10].

Có 4 lợi thế của YOLO [11]:

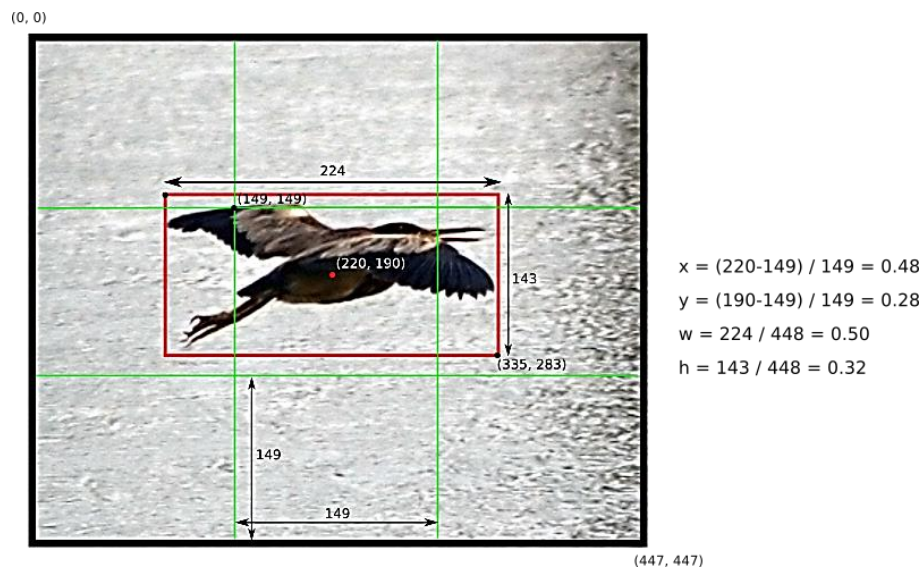
+ Tốc độ nhanh. Do hệ thống phát hiện và tìm tọa độ vật đã kết hợp thành một mạng CNN, ta có thể cho hình vào và thậm chí là đoạn phim để nhận dạng. So với Fast R-CNN, mỗi lần chạy chương trình sẽ tạo ra 2000 dự đoán vị trí, YOLO nhanh gấp hàng trăm lần.

+ Có thể “nhìn” cả bức ảnh so với từng phần như R-CNN, từ đó những dự đoán của nó được cung cấp nội dung toàn cục của bức ảnh. So với Fast R-CNN, YOLO chỉ mất dưới một nửa số lỗi nhận dạng background.

+ Gọn nhẹ. YOLO chỉ sử dụng một mạng đi thẳng từ các lớp tích chập sang lớp Fully connected. Hình ảnh chỉ đi qua một lần rồi nó sẽ trả về ngõ ra. Vì vậy ta mới gọi là thuật toán “chỉ cần nhìn một lần”.

+ Mã nguồn mở. Ta có thể thay đổi và cải tiến tùy theo ý muốn.

Bức hình khi vào đầu vào sẽ được chia theo lưới SxS. Nếu điểm giữa của vật rơi vào ô nào, ô đó sẽ chịu trách nhiệm nhận dạng vật đó. Mỗi ô này sẽ dự đoán tối đa B hộp giới hạn và độ tin cậy vào dự đoán trên. Độ tin cậy này không tiết lộ vật này là vật gì, chỉ thể hiện hệ thống tin báo hiệu phần trăm đây là vật có thể nhận dạng. Các hộp giới hạn mã hóa 5 chỉ số: độ tin cậy và thông số kích thước hộp giới hạn và tọa độ tâm hộp (x, y, w, h). Hệ tọa độ trong ô của lưới trên được chuẩn hóa giá trị từ 0 tới 1. Cho nên các giá trị x, y, w, h đều từ 0 đến 1. Trong đó, x, y thể hiện tọa độ điểm giữa của nguyên cả vật so với ô chứa điểm đó; w, h thể hiện chiều dài, rộng của hộp giới hạn so với kích thước ảnh/video đầu vào.

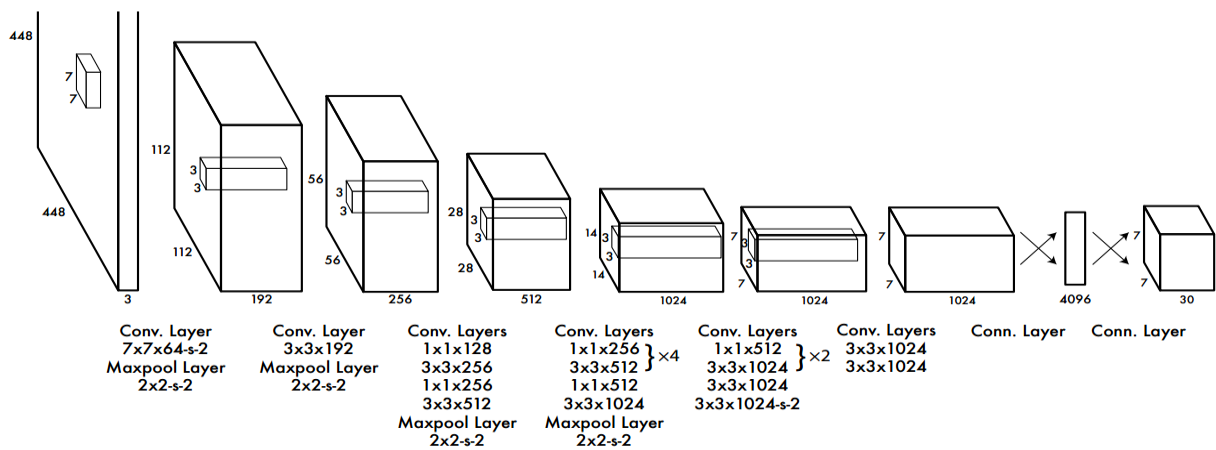


Hình 2.9 Ví dụ về cách tính tọa độ, kích thước hộp giới hạn cho lưới 3x3 [34].

Mỗi ô đồng thời cũng dự đoán xác suất của một số lớp cho trước. Số lượng lớp đặt là C. Xác suất này phụ thuộc vào việc tồn tại chỉ một vật ở trong một

ô, vì một ô chỉ đoán một vật kể cả có nhiều hơn một hộp giới hạn trong ô đó. Với $S \times S$ ô và mỗi ô được đoán B hộp, ta tạo một tensor có kích thước $S \times S \times (B \times 5 + C)$.

Vì có $S \times S$ ô, mỗi ô chỉ được dự đoán B hộp giới hạn, ta sẽ có $S \times S \times B$ hộp giới hạn. Ta cần có một hệ số ngưỡng để loại đi những hộp có độ tin cậy thấp hơn hệ số đó. Sau đó các hộp còn lại đi qua thuật toán Non-max suppression để loại bỏ các hộp giới hạn cùng bao lấy một đối tượng.



Hình 2.10 Cấu trúc mạng YOLO 24 lớp tích chập [14].

Thiết kế mạng YOLOv1 có 24 lớp tích chập. Một số lớp dùng kernel 1×1 để giảm độ sâu của bản đồ đặc trưng. Ở lớp tích chập cuối mạng cho ra ma trận khối $7 \times 7 \times 1024$. Sau khi đi qua 2 lớp Fully connected cho ra khối $7 \times 7 \times 30$ (30 bao gồm 20 lớp, 5 chỉ số cho hộp giới hạn 1, 5 cho hộp giới hạn 2 [12]).

YOLOv2

Khi so sánh YOLOv1 với Faster RCNN:

+ Tuy nhanh hơn rất nhiều nhưng YOLO vẫn có độ chính xác thấp hơn so với nó.

+ Mỗi ô trên hình chỉ đoán được 1 vật, dẫn đến nhận dạng thiếu

Vì vậy YOLOv2 được tạo ra cải tiến lên từ YOLOv1.

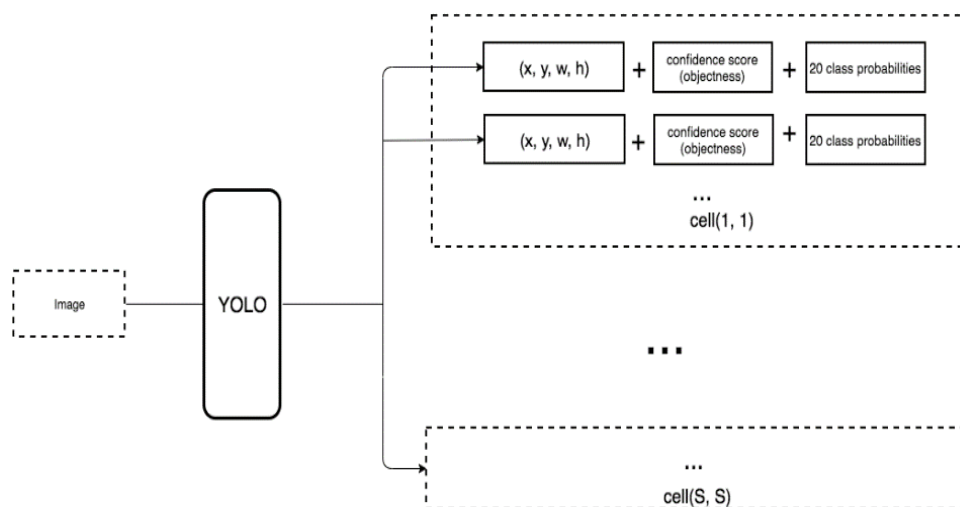
+ Sử dụng ý tưởng hộp giới hạn từ Faster RCNN và SSD, YOLOv2 dự đoán độ sai lệch của hộp giới hạn so với các hộp anchor. Từ những sai lệch và hộp anchor sẽ tính ra hộp giới hạn. Tất cả được thực hiện trong mạng CNN giống như SSD.

+ Thêm vào chuẩn hóa hàng loạt để tăng tốc độ và độ chính xác lên.

+ Loại bỏ mạng Fully connected nằm ở cuối mạng YOLO version 1, hình thành mạng Fully Convolutional Neural Network (mạng neuron toàn tích chập). YOLOv2 sử dụng mạng Darknet-19 với việc bỏ lớp cuối bằng lớp tích chập 1×1 với số đầu ra tương ứng với $B \times (5 + C)$.

+ Bây giờ mỗi hộp giới hạn sẽ có phân bố xác suất riêng, không còn giới hạn một lớp trên một ô nữa. Điều này cho phép YOLOv2 xác định đồ vật trong mỗi ô chính xác hơn. Bây giờ đầu ra là một tensor có kích thước $S \times S \times B \times (4+1+C)$.

- $S \times S$: số ô lưới được chia. Trong YOLOv2 là $S=13$.
- B : Số hộp. Trong YOLOv2 là $B = 5$
- 4: độ sai lệch so với hộp anchor
- 1: Độ tin cậy (Confidence Score hay Objectness)
- C : số class \Rightarrow phân bố xác suất



Hình 2.11 Ví dụ cho dự đoán hộp giới hạn 25 thông số trong một hộp [12].

Với mỗi ô của lưới, YOLOv2 dự đoán cho ra 5 thông số: t_x , t_y , t_w , t_h , t_o từ 5 tham số tính ra các giá trị của hộp giới hạn theo công thức:

$$b_x = \sigma(t_x) + c_x$$

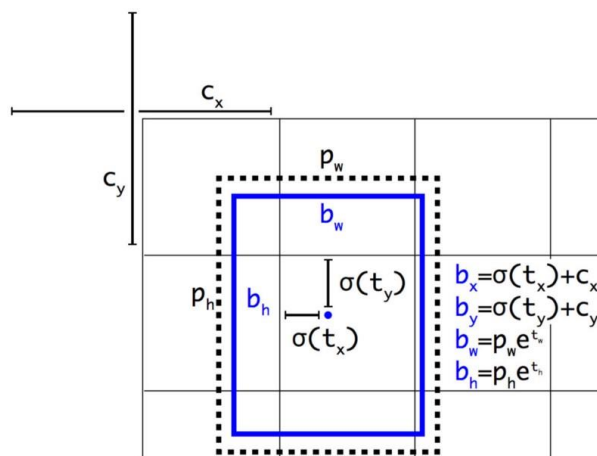
$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w \cdot e^{t_w}$$

$$b_h = p_h \cdot e^{t_h}$$

$$\text{Pr(object)} * \text{IoU(b, object)} = \sigma(t_o)$$

- b_x , b_y , b_w , b_h : tọa độ của hộp giới hạn
- t_x , t_y , t_w , t_h , t_o : tọa độ dự đoán của YOLO
- c_x , c_y : tọa độ góc trái gốc của hộp anchor
- p_w , p_h : chiều cao, chiều rộng gốc của hộp anchor

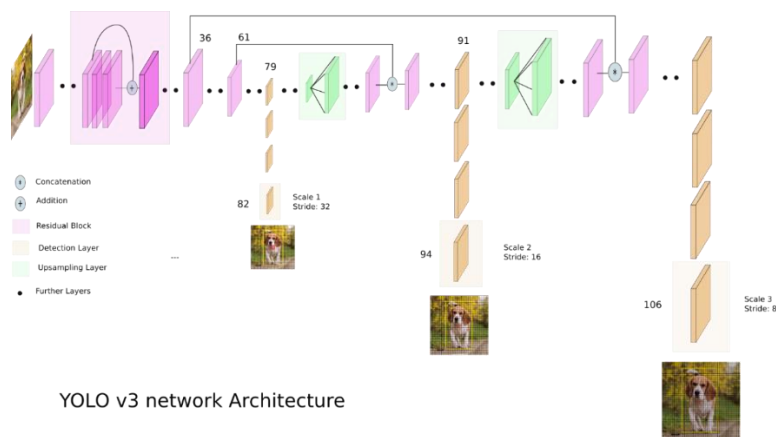


Hình 2.12 Hộp giới hạn (nét liền), hộp anchor (nét đứt) và dự đoán vị trí [9].

YOLOv3

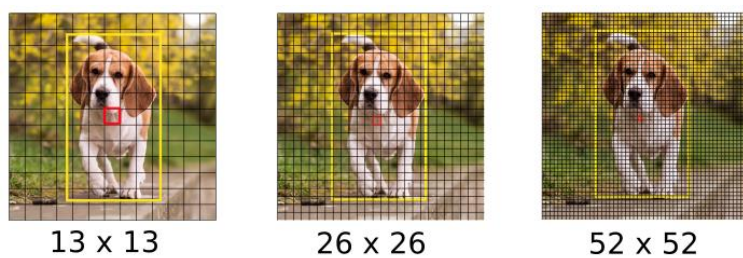
Vào năm 2018, YOLO lại được cập nhật lên phiên bản 3 nhằm nâng cao độ chính xác lên, đánh bại mạng SSD.

Thay vì sử dụng mạng như YOLOv2, YOLOv3 được tạo ra từ biến thể Darknet-53 sử dụng cho tìm bản đồ đặc trưng, 53 lớp kế theo thực hiện phát hiện vật thể thật sự.

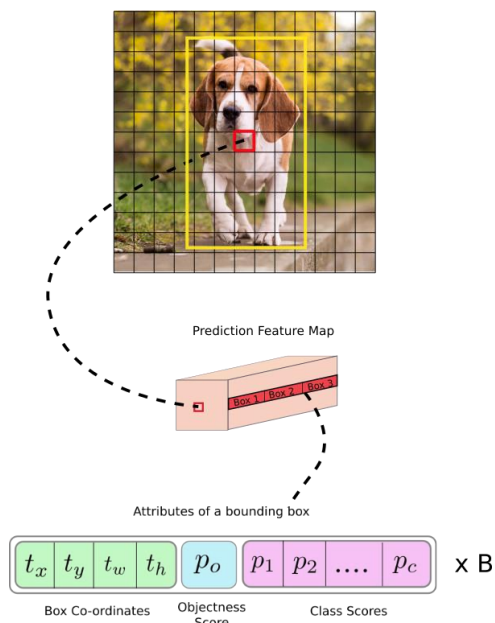


Hình 2.13 Cấu trúc mạng YOLOv3 [35].

Điểm nổi bật nhất của YOLOv3 là mạng thực hiện nhận dạng vật ở 3 tỉ lệ ảnh khác nhau. Các lớp thực hiện phát hiện vật tạo các bản đồ đặc trưng ở 3 kích cỡ khác nhau với kích cỡ bằng 32, 16, 8 cho ra bản đồ 13x13, 26x26, 52x52. Bản đồ đặc trưng càng có kích thước lớn ảnh càng rõ nét



Hình 2.14 Lưới cho từng tỉ lệ hình [36].



Hình 2.15 Nội dung bên trong một ô của bản đồ đặc trưng [35].

Nội dung bên trong mỗi ô của bản đồ đặc trưng tương tự như YOLOv2, gồm: tọa độ dự đoán với công thức tính hộp giới hạn như YOLOv2, xác suất có vật và xác suất vật đó thuộc lớp nào.

+ Bản đồ đặc trưng 13×13 được sử dụng cho lớp thứ 82 để dự đoán vật. Kết quả thu được là ma trận khối $13 \times 13 \times (B \times (5 + C))$.

+ Sau đó, bản đồ đặc trưng lớp 79 được trộn gấp 2 lần rồi nối lại với bản đồ đặc trưng lớp 61. Điều này giúp tăng thông tin “ý nghĩa” của bản đồ đặc trưng lên. Đến lớp thứ 94, thuật toán thu được ma trận khối $26 \times 26 \times (B \times (5 + C))$.

+ Thực hiện tương tự với bản đồ đặc trưng lớp 91 nối lại với bản đồ đặc trưng lớp 36 rồi qua lớp 106 thu lại ma trận khối $52 \times 52 \times (B \times (5 + C))$.

Tại mỗi tỉ lệ hình, YOLOv3 sử dụng 3 hộp anchor ($B = 3$). Với mỗi lần trộn lên, thuật toán có thể nhận ra những vật nhỏ hơn. Ta có tối đa số box nhận được:

$$3 \times (13 \times 13 + 26 \times 26 + 52 \times 52) = 10647 \text{ (hộp)}$$

Vậy số hộp tối đa có thể phát hiện là 10647, lớn hơn SSD 1.2 lần. Vì thế có thể lý giải cho việc phiên bản 3 chạy chậm hơn phiên bản 2.

Không như YOLOv2, mạng phiên bản mới đã thay đổi cách tính xác suất vật thuộc lớp nào. Thay vì sử dụng softmax để vật chỉ thuộc 1 lớp, ta sử dụng logistic regression để dự đoán, vì các dataset sau có nhiều lớp có thuộc tính gần nhau (giả sử Person và Woman) [37].

Bảng 1. Bảng so sánh tốc độ các mô hình nhận dạng đối tượng qua tập test PASCAL VOC 2007

MÔ HÌNH	MAP (MEAN AVERAGE PRECISION)	FPS	REAL TIME SPEED
FAST YOLO	52.7%	155	Yes
YOLO	63.4%	45	Yes

YOLO VGG-16	66.4%	21	No
FAST R-CNN	70.0%	0.5	No
FASTER R-CNN VGG-16	73.2%	7	No
FASTER R-CNN ZF	62.1%	18	No

Qua bảng so sánh tốc độ của các mô hình nhận dạng đối tượng qua tập kiểm tra PASCAL VOC 2007 ta thấy rõ sự vượt trội của các phiên bản mạng YOLO. So sánh về tốc độ xử lý, YOLO đạt tiêu chuẩn xử lý thời gian thực, đặc biệt phiên bản cải tiến Fast YOLO có tốc độ lên đến 155 FPS, cao vượt trội so với các mô hình khác, độ chính xác việc nhận dạng của mô hình này đạt tỉ lệ thật sự rất thấp 52,7%; đơn thuần YOLO xử lý ở FPS là 45 frame và độ chính xác trung bình là 63,4% ở ngưỡng chấp nhận được. Như vậy, so với việc tăng cao FPS thì ta nên ưu tiên sự chính xác mô hình nhiều hơn.

2.2.4 Nhận diện khuôn mặt

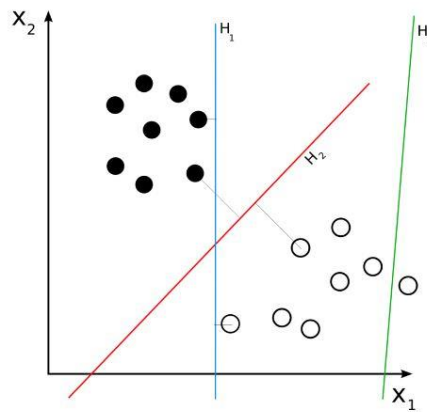
2.2.4.1 SVM (Support Vector Machines) [15]

SVM là một mô hình được ứng dụng trong nhiều ngành, ở trong xử lý ảnh, SVM cũng tác dụng to lớn nhiều trong các ứng dụng như trong ứng dụng nhận dạng ký tự viết tay, nhận dạng biển số xe, ...

SVM còn được biết đến như một mô hình máy học giám sát được dùng để phân tích, phân lớp dữ liệu, SVM là một khái niệm trong thống kê và khoa học máy tính cho một tập hợp các phương pháp học có giám sát liên quan đến nhau để phân loại và phân tích hồi quy.

SVM về cơ bản được xem một thuật toán phân loại nhị phân, SVM nhận dữ liệu vào và phân loại chúng vào hai lớp khác nhau. Với một bộ các ví dụ luyện tập thuộc hai thể loại cho trước, thuật toán luyện tập SVM xây dựng một mô hình SVM để phân loại các ví dụ khác vào hai thể loại đó.

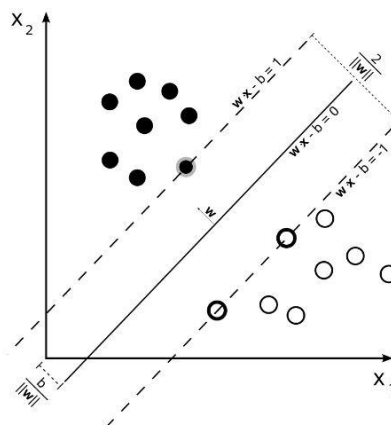
Ví dụ, ta có một không gian với nhiều điểm như hình sau:



Hình 2.16 Ví dụ về bài toán SVM

Yêu cầu của bài toán này là phải tìm một siêu phẳng có lề lớn nhất (lề ở đây là khoảng cách từ điểm cần xét đến siêu phẳng) phân tách các điểm dữ liệu ban đầu để huấn luyện và phân loại các điểm sau này. Các siêu phẳng đều có thể viết được dưới dạng tập hợp các điểm sao cho chúng thỏa mãn: $w \cdot x - b = 0$.

Ở ví dụ trên, giả sử ta có 3 tập hợp siêu phẳng là H_1 , H_2 , H_3 . Ta sẽ loại H_3 đầu tiên vì nó không thể phân loại các đối tượng. Đối với H_1 , ta cũng không chọn vì khoảng cách từ các điểm đến siêu phẳng chưa phải là lớn nhất. H_2 là siêu phẳng cần tìm vì nó thỏa mãn yêu cầu của siêu phẳng. Các điểm dữ liệu cho trước sẽ nằm trên các siêu phẳng song song (còn gọi là các Support Vector).



Hình 2.17 Siêu phẳng và các Support Vector

Như vậy, SVM xây dựng một siêu phẳng hoặc một tập hợp các siêu phẳng trong một không gian nhiều chiều hoặc vô hạn chiều, có thể được sử dụng cho phân loại, hồi quy, hoặc các nhiệm vụ khác. Để phân loại tốt nhất thì phải xác

định siêu phẳng (Optimal hyperplane) nằm ở càng xa các điểm dữ liệu của tất cả các lớp (Hàm lề) càng tốt, vì nói chung lề càng lớn thì sai số tổng quát hóa của thuật toán phân loại càng bé.

Muốn các điểm dữ liệu có thể được chia tách một cách tuyến tính, thì cần phải chọn hai siêu phẳng của lề sao cho không có điểm nào ở giữa chúng và khoảng cách giữa chúng là tối đa.

Trong nhiều trường hợp, không thể phân chia các lớp dữ liệu một cách tuyến tính trong một không gian ban đầu được dùng để mô tả một vấn đề. Vì vậy, nhiều khi cần phải ánh xạ các điểm dữ liệu trong không gian ban đầu vào một không gian mới nhiều chiều hơn, để việc phân tách chúng trở nên dễ dàng hơn trong không gian mới.

Với các điểm tổng quan ở trên thì nhiệm vụ chính là phân loại thống kê:

- Thuật toán được cho trước một số điểm dữ liệu cùng với nhãn của chúng thuộc các lớp cho trước (Huấn luyện).
- Mục tiêu của thuật toán là xác định xem một điểm dữ liệu mới sẽ được thuộc về lớp nào (Phân loại).

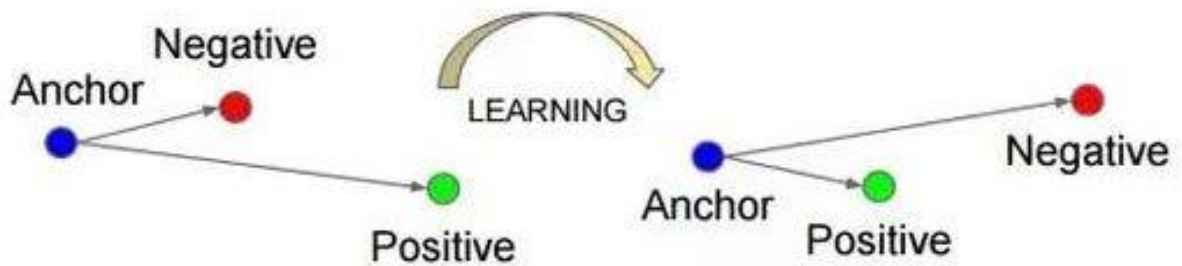
Các bước thực hiện cho mô hình SVM:

- Thiết lập dữ liệu huấn luyện.
- Thiết lập thông số của SVM
- Huấn luyện SVM
- Phân loại lớp (bản lề)

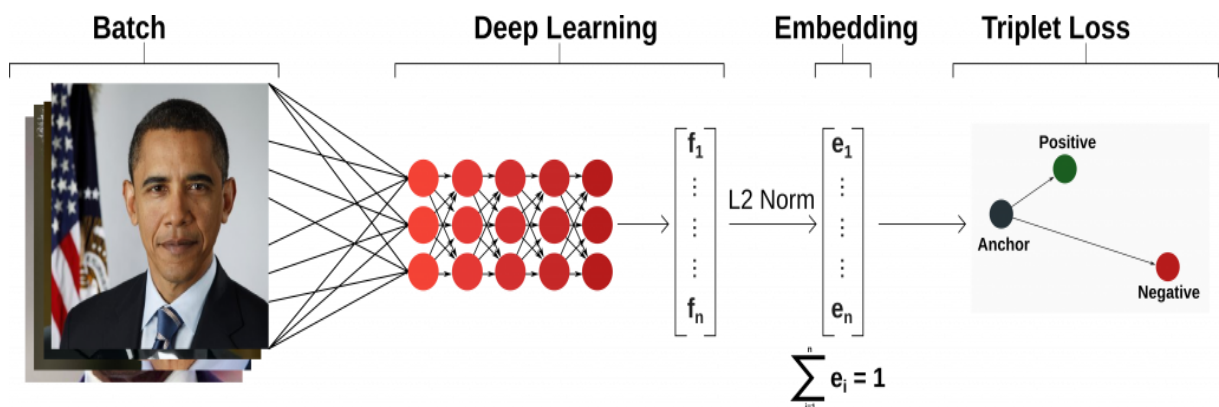
2.2.4.2 Face Recognition với Facenet [16]

Để hiểu cho đơn giản CNN hay Mạng neuron tích chập gồm các lớp tích chập sẽ thực hiện các thao tác tách đặc trưng của một hình ảnh ra và sau đó sử dụng một mô hình máy học khác như kNN hoặc SVM để phân biệt người này với người khác. Cách thực hiện này là one-shot learning, CNN chỉ đóng vai trò rút trích đặc trưng, việc huấn luyện CNN ở đây là khiến nó tách đặc trưng tốt

hơn. Ta cũng có thể thực hiện huấn luyện một CNN với số đầu ra là số người, nhưng điều đó rất khó khăn, đòi hỏi thay đổi cấu trúc mạng và việc thu thập dataset cũng mất thời gian hơn, còn có thể mất cân bằng bộ dataset.

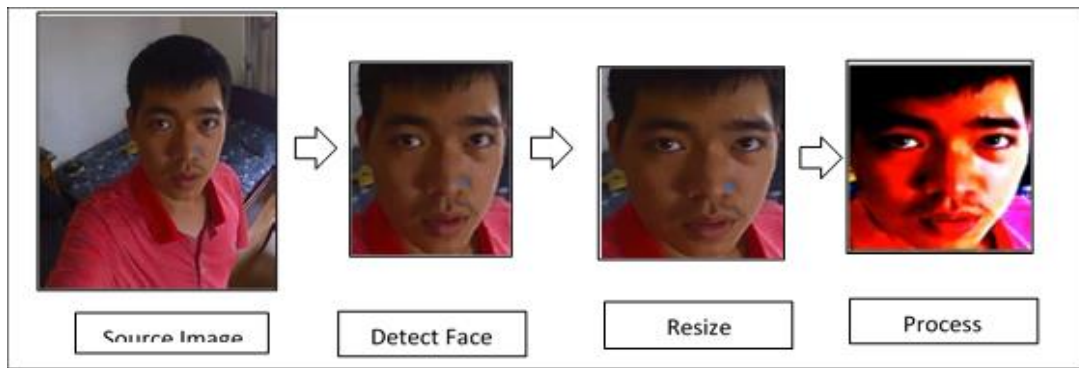


Hình 2.18 Tác dụng của hàm lỗi Triplet Loss [18] tối thiểu hóa khoảng cách đến phần dương, tối đa hóa khoảng cách đến phần âm



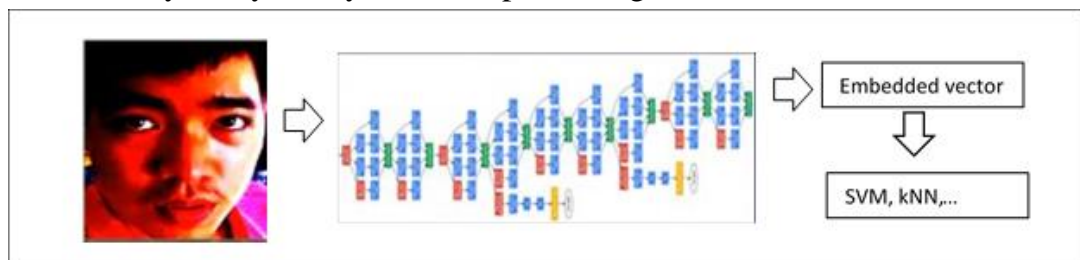
Hình 2.19 Cách Facenet (phần deep learning) hoạt động [38].

Facenet, thực chất là một CNN có nhiệm vụ tách các đặc trưng của một ảnh mặt. Điểm đặc biệt tạo nên sự khác biệt của Facenet là nó sử dụng hàm lỗi Triplet để tối thiểu hóa khoảng cách giữa các gương mặt tương đồng và tối đa hóa khoảng cách đến những gương mặt không tương đồng, vì vậy facenet có thể phân biệt rất chính xác người với người. Quá trình train CNN được thực hiện trên các bộ dataset lớn (ví dụ như vggface hoặc MS 1 million), kết quả ta muốn là tập embedding vector ra phải tách nhau ra thành từng cụm để thực hiện phân loại bằng kNN hoặc SVM. Quá trình training dựa trên triplet loss như sau: Tiền xử lý data: Chú ý là các bộ dataset kể trên thường có dạng là các thư mục với tên là tên của một người, trong thư mục là hình của người đó, mỗi người tầm 100-1000 hình, một bộ dữ liệu thì khoảng vài nghìn người.



Hình 2.20 Bước tiền xử lý của Facenet.

Sử dụng một bộ nhận diện khuôn mặt để tách phần ảnh có gương mặt người ra. Các phương pháp hiện đại sử dụng MTCNN (Multi-task Cascaded Convolutional Networks), ưu điểm là mô hình này xác định được gương mặt ở nhiều góc độ, và có thể nhận diện cả ranh giới khuôn mặt với thời gian xử lý khá tốt. Sau đó ta thay đổi kích thước hình lại một kích cỡ xác định (tùy chọn), facenet thì là 160x160 [19], gần đây thì arcface dùng cỡ 112x112 [20], việc chọn cỡ này là tùy theo yêu cầu về phần cứng hoặc độ chính xác.



Hình 2.21 Hoạt động của Facenet.

Thực hiện các bước “lọc” bộ dataset. Thực tế là các bộ dataset lớn như vggface hay MS 1m đều có nhiều “nhiều”, tức các hình không chứa gương mặt hoặc gương mặt khác với người cần xét. Vì vậy có thể dùng một mô hình nhận diện gương mặt khác để tìm và loại bỏ những hình “nhiều” ấy. Hoặc đơn giản là ta có thể lấy clean list (danh sách các hình chuẩn) đã được lọc sẵn.

Thực hiện bước normalize cho tập dataset đã chuẩn bị. Cách normalize cũng giống như cách normalize cho CNN bình thường. Các bước train: Đầu vào là tập dataset đã chuẩn bị, CNN có thể là các CNN như inception, mobilenet,... với đầu ra là embedding vector như đã nói, embedding vector này có thể có 128 chiều hoặc 512 chiều,... tùy ý. Tóm lại ta chỉ cần lấy cấu trúc

inception hoặc mobilenet, bỏ lớp cuối và thêm vào lớp embedding vector (không có activation).

Chạy trên toàn bộ tập data, lưu các vector nhúng đã tính trên minibatch lại.

Chia minibatch thành 3 phần là phần âm, phần dương và anchor, trong đó anchor là phần “neo” giống như “tâm” của cụm embedding vector, sau đó chọn phần dương là những embedding vector là cùng người với anchor, phần âm là khác người.

Tính Triplet loss là khoảng cách giữa các embedding vector anchor và phần dương trừ cho khoảng cách giữa các embedding vector anchor và phần âm (tức ta tìm cách làm giảm khoảng cách từ các vector dương tới anchor và ngược lại đẩy các vector âm đi xa khỏi anchor). Ngoài Triplet Loss còn có các biến thể của Softmax Loss như Center Loss, ArcLoss, Cosine Loss,... trong đó hàm lỗi được biến đổi cho mục đích phân cụm càng nhiều càng tốt.

Sau khi đã train trên các tập data lớn thì kết quả đạt được là một CNN có thể tách được các đặc trưng hữu ích để phân loại gương mặt. Lúc này để nhận diện người với người ta cần các bước huấn luyện:

1. Tạo một bộ dataset khác cho đối tượng mình muốn nhận diện.
2. Lưu bộ dataset đó theo cấu trúc như bộ dataset lớn đã train (để dễ tính lại các embedding vector), cấu trúc là thư mục chữ ảnh của cùng một người - một người tầm 10-20 ảnh, chú ý là đủ các góc độ sẽ cho hiệu quả hơn.
3. Tính embedding vector cho tập đã thu thập, lưu các embedding này vào file nào đó (tùy chọn)
4. Huấn luyện kNN hoặc SVM trên tập vector để phân loại người với người.

Thực hiện trong thực tế, ta có thể làm theo cách sau:

1. Sử dụng một mô hình phát hiện đối tượng nào đó để tìm hộp giới hạn của gương mặt (các mô hình hiện đại sử dụng MTCNN, còn OpenFace dùng Dlib detector) chú ý là mô hình phát hiện khuôn mặt phải giống với mô hình phát hiện khuôn mặt được dùng cho quá trình huấn luyện mới cho kết quả chính xác
2. Cắt phần gương mặt đã tìm được ra, thay đổi kích thước lại và thực hiện bước normalize data.
3. Đưa vào CNN đã train để tính ra được embedding vector.
4. Dùng kNN, SVM, hoặc so sánh khoảng cách để tìm “cụm” mà embedding vector đó thuộc về, từ đó suy ra danh tính người được chọn.

Tóm lại đó là cách facenet hay FaceID hoạt động, trích đặc trưng và so sánh. Tất nhiên nói luôn dễ hơn làm, để thực hiện một hệ thống an ninh bằng gương mặt cần đầu tư thời gian và công sức rất nhiều.

2.2.4.3 SSD dựa trên Resnet. [17]

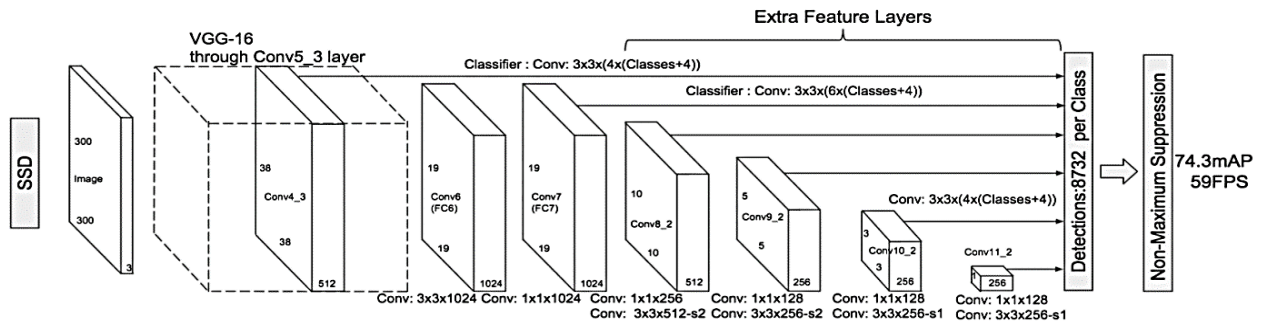
SSD được thiết kế để phát hiện đối tượng trong thời gian thực. Faster R-CNN sử dụng mạng đề xuất khu vực để tạo các hộp giới hạn và sử dụng các hộp đó để phân loại các đối tượng. Mặc dù được coi là khởi đầu chính xác, toàn bộ quá trình chạy ở 7 khung hình mỗi giây. Thấp hơn nhiều so với những gì cần cho một nhu cầu xử lý thời gian thực.

SSD tăng tốc quá trình bằng cách loại bỏ sự cần thiết của mạng đề xuất khu vực. Để giải quyết về vấn đề độ chính xác giảm, SSD áp dụng một vài cải tiến bao gồm các feature map đa kích thước và sử dụng các hộp anchor. Những cải tiến này cho phép SSD tiến gần được với độ chính xác của Faster R-CNN nhưng lại có thể sử dụng hình ảnh có độ phân giải thấp hơn, giúp đẩy tốc độ cao hơn.

Mô hình SSD được chia làm hai giai đoạn:

1. Trích xuất feature map (dựa vào mạng cơ sở VGG16) để tăng hiệu quả trong việc phát hiện => thì nên sử dụng ResNet, InceptionNet, hoặc MobileNet
2. Áp dụng các bộ lọc tích chập để có thể detect được các đối tượng.

Giai đoạn 1: mô hình SSD sử dụng mạng cơ sở VGG16 để trích xuất bản đồ đặc trưng.



Hình 2.22 Mô hình SSD.

SSD sử dụng VGG16 làm mạng cơ sở để trích xuất feature map.

Đầu vào: image kích thước (300 * 300)

Sau đó sử dụng VGG16 làm mạng cơ sở để trích xuất bản đồ tính năng. Lưu ý, đối với VGG16, chúng ta sẽ sử dụng lớp Conv4_3 là một trong những lớp dự đoán đối tượng của mạng, những lớp còn lại chỉ có tác dụng trích xuất các đặc trưng của hình ảnh đầu vào

Sau khi tạo ra feature map thì mỗi ô của feature map sẽ đưa ra 4 dự đoán đối tượng.

Mỗi dự đoán này bao gồm:

- 1 bounding box
- 21 score biểu thị cho độ tin cậy của mỗi lớp (bao gồm cả lớp background)

(score: là xác suất xuất hiện đối tượng được dự đoán có trong hộp giới hạn, 20 score là 20 xác suất phát hiện 20 loại đối tượng trong thực tế, 1 score còn lại là xác suất dự đoán trong hộp giới hạn là lớp nền).

Do đó, lớp Conv4_3 đưa ra tổng số lượng dự đoán là $38 \times 38 \times 4$ (4 bbb được dự đoán tại mỗi vị trí pixel).

Giai đoạn 2: Sử dụng trình dự đoán tích chập (Convolution predictors) cho phát hiện đối tượng:

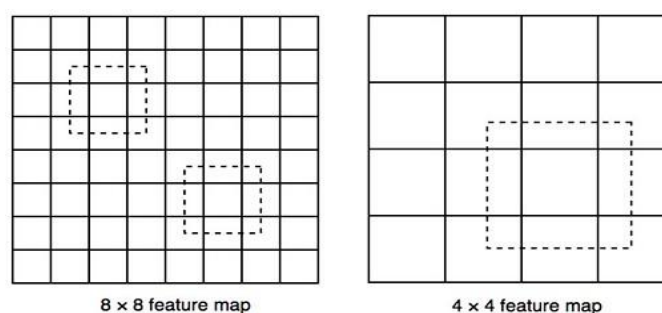
Sau khi trích xuất bản đồ đặc trưng dựa vào kiến trúc của mạng cơ sở (base network), mô hình SSD sẽ tiếp tục tính toán cả hai giá trị: đối với từng khu vực và score cho các lớp bằng việc sử dụng các bộ lọc tích chập nhỏ. Sau khi trích xuất bản đồ đặc trưng, SSD sử dụng một kernel 3×3 trên mỗi ô để tạo ra dự đoán. Mỗi đầu ra của bộ lọc là 25 kênh: 21 scores đối với mỗi lớp + 4 dự đoán.

Áp dụng bộ lọc 3×3 để tạo ra các predict đối với mỗi khu vực và các lớp. Quá trình sử dụng các trình phát hiện tích chập đơn giản cũng chỉ là quá trình predict ra các bbx dự đoán (với mỗi vị trí pixel sẽ là 4 bbx dự đoán giống như VGG16 đang làm) có điều nó khác bởi vì nó sử dụng các feature map có kích thước khác nhau với mục đích như sau:

1. Bản đồ đặc trưng nhiều tỷ lệ cho việc nhận dạng:
2. Vấn đề phối cảnh sẽ thay đổi kích thước của các vật thể.

Ở trên là việc phát hiện đối tượng ở mỗi một kích thước, trên thực tế thì việc phát hiện nhiều đối tượng với nhiều kích thước sẽ sử dụng quy trình tương tự nhưng ở các lớp khác nhau.

Cần chú ý một việc là đối với các bản đồ đặc trưng có độ phân giải thấp sẽ có khả năng phát hiện được đối tượng có kích thước khác nhau. Do đó, SSD thêm 6 lớp Conv phụ trợ vào sau VGG16, 5 trong số chúng sẽ được thêm vào để phát hiện đối tượng. Và trong 3 trong số 6 lớp này sẽ đưa ra dự đoán là 6 (bbx + score) thay vì 4 (bbx + score). Tổng cộng, SSD đưa ra 8732 dự đoán với việc sử dụng 6 lớp.



Hình 2.23 Độ phân giải nhỏ hơn của bản đồ đặc trưng giúp phát hiện vật lớn hơn.

Các điểm khác trong mô hình SSD

- SSD sử dụng non-maximum suppression (nms) để thực hiện loại bỏ các bbx dư thừa.
- SSD sử dụng lấy mẫu cứng (hard sample) để đào tạo làm tăng độ chính xác dự đoán của mô hình.
- SSD sử dụng các kỹ thuật tăng cường dữ liệu (data augmentation) để giúp tăng độ chính xác của thuật toán.

2.3 Bài toán theo vết đối tượng trong xử lý thời gian thực

2.3.1 Định nghĩa bài toán theo vết đối tượng

Khác với nhận diện đối tượng trong ảnh tĩnh, nhận dạng đối tượng trong video thời gian thực khó khăn hơn khi các đối tượng này có khả năng chuyển động liên tục các vị trí khác nhau trong khung hình. Việc này đòi hỏi nhiều tài nguyên hơn khi chỉ nhận diện đối tượng ở từng khung hình. Tuy nhiên, nếu ta chỉ tập trung theo dõi đối tượng đó, và theo dõi các vết của chúng thay đổi qua từng khung hình thì lượng tài nguyên được giảm thiểu rất nhiều cho việc duy trì nhận dạng liên tục.

Như vậy, nhiệm vụ của vấn đề theo vết đối tượng là chính xác hóa sự tương ứng của các vết đối tượng trong các khung hình liên tiếp, từ đó dự đoán đường đi, vận tốc, hướng chuyển động của các đối tượng.

Theo vết đối tượng còn là giám sát các thay đổi theo không gian và thời gian của đối tượng trong suốt chuỗi video như vị trí, kích thước hoặc hình dáng của đối tượng.

Quy trình theo vết đối tượng được mô tả như hình bên dưới:



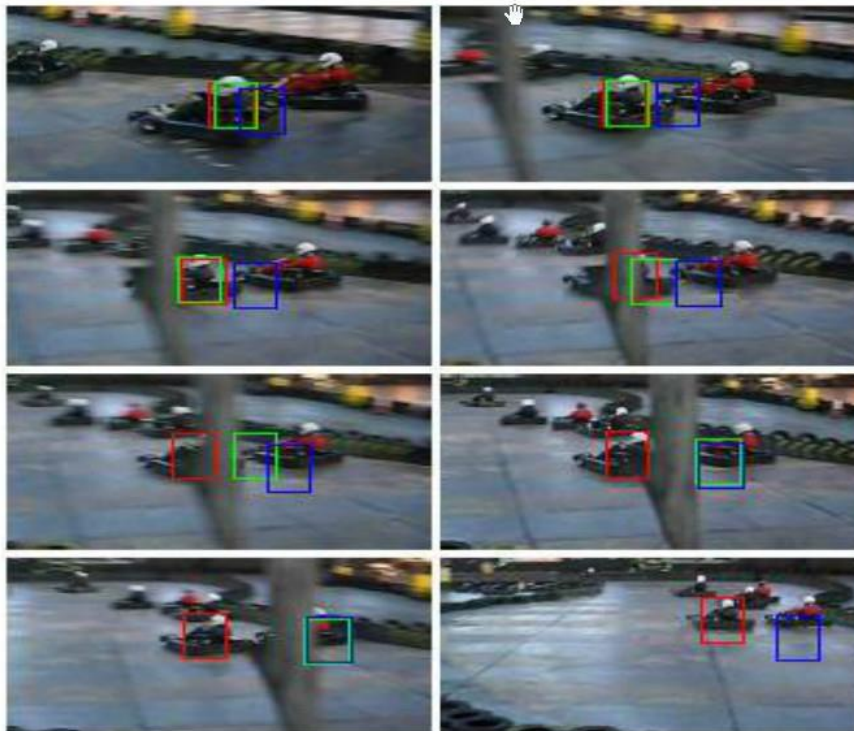
Hình 2.24 Sơ đồ thực hiện bài toán theo vết đối tượng trong video

2.3.2 Chính xác hóa đối tượng tương ứng (Object matching)

Chính xác hóa đối tượng tương ứng là module đầu tiên trong bài toán theo vết đối tượng chuyển động qua các khung hình video. Module này chịu trách nhiệm xác định chính xác đối tượng chỉ là chủ thể được lập lại qua các khung hình liên tiếp. Do đó, từ vết đối tượng đã được phát hiện, cần xác định sự tương ứng của các vết này trong các khung hình tiếp theo từ đó, xét xem các vết này có phải của cùng một đối tượng hay không.

Có một bất cập trong module này chính là nếu một đối tượng rời khung hình hoặc bị che khuất rồi xuất hiện trở lại, module này có thể đánh dấu sai đối tượng.

Hình bên dưới mô tả cho chức năng của module này.



Hình 2.25 Sự chính xác hóa đối tượng

2.3.3 Xử lý nhập nhằng (Occlusion)

Tuy đầu vào là các vết đã được phát hiện, vấn đề theo vết đối tượng vẫn gặp phải nhiều khó khăn khi xảy ra sự nhập nhằng giữa nhiều vết đối tượng với nhau. Cụ thể như, khi hai đối tượng tương tự nhau di chuyển nhập lại thành một, hay

một đối tượng tách thành hai rồi di chuyển ra xa nhau, sự phân biệt các vết đối tượng trở nên khó khăn hơn rất nhiều.

Bài toán theo vết đối tượng còn gặp phải khó khăn khi gặp các vấn đề như sau:

- Ảnh nền lộn xộn: Có nhiều đối tượng cùng xuất hiện.
- Ảnh nền động: Sự chuyển của camera.
- Cường độ chiếu sáng thay đổi: Thay đổi hướng và cường độ chiếu sáng.
- Thay đổi điểm nhìn: Thay đổi vị trí camera.
- Nhập nhằng: Đối tượng đang theo dõi biến mất hoặc bị che khuất.

Hầu hết các khó khăn ở trên đều gặp phải ở đề tài này. Việc camera có thể di chuyển để theo dõi đối tượng tạo nên tất cả các yếu tố làm ảnh hưởng đến việc theo vết đối tượng, do đó, xử lý nhập nhằng là vô cùng quan trọng trong đề tài giám sát đối tượng thông qua thiết bị bay này.

2.3.4 Dự đoán chuyển động

Mô hình sẽ được thực hiện của đề tài này phụ thuộc nhiều vào việc theo vết tuy nhiên, dự đoán chuyển động của đối tượng đang giám sát cũng là việc không kém phần quan trọng để điều khiển thiết bị bay mang theo camera theo dõi. Việc dự đoán trước chuyển động để đưa ra các thông tin như hướng di chuyển, vận tốc của đối tượng có thể giúp ta chuẩn trước trước các trường hợp để điều khiển thiết bị bay, giảm thời gian xử lý, thời gian trễ, tăng nhanh tốc độ của cả mô hình, việc vốn dĩ là vô cùng cần thiết để xử lý thông tin thời gian thực cần sự đáp ứng nhanh.

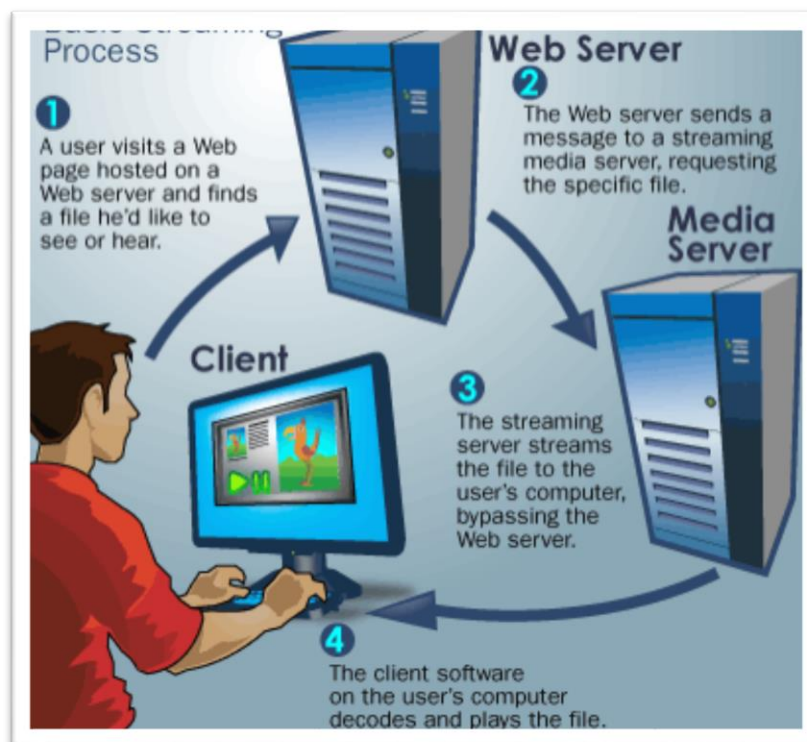
2.4 Kỹ thuật video streaming [39]

Streaming video là một kỹ thuật được sử dụng khá phổ biến trong các ứng dụng mạng. Rất nhiều các địa chỉ ứng dụng rộng rãi trong thực tế như: các phần mềm (media player, web browser, ...) trên các máy khách truy cập và xem video từ các máy chủ theo mô hình server/client; các ứng dụng hội họp trực tuyến, đào tạo từ xa;

giám sát, điều khiển từ xa qua hình ảnh thời gian thực, v.v... Đặc biệt là các ứng dụng áp dụng hệ thống nhúng đã dùng kỹ thuật streaming video để hoàn thiện ứng dụng mình hơn.

Kỹ thuật video streaming ra đời đã được áp dụng vào các phần mềm giải trí và cả những ứng dụng hỗ trợ cho việc quản lý, hội họp, quan sát và điều khiển trong các doanh nghiệp, cơ quan, tổ chức trở nên dễ dàng và hiệu quả.

Streaming video sử dụng cách thức phát lại các đoạn video được lưu trữ trên các máy tính trên mạng tới người dùng đầu cuối muốn xem đoạn video mà không cần tải đoạn video đó về trên máy tính. Về bản chất, streaming video là quá trình chia nhỏ file video thành các frame, rồi lần lượt gửi từng frame tới một bộ đệm trên máy tính của người xem và hiển thị nội dung frame đó. Và quá trình này tuân thủ chặt chẽ về ràng buộc theo thời gian, nói khác là tuân thủ chặt chẽ theo giao thức RTSP, RTP và RTCP. Với đặc tính như vậy thì streaming video là kỹ thuật cũng khá phức tạp để triển khai. Nhưng với những lợi ích mà kỹ thuật streaming video đem lại, chúng ta hoàn toàn có thể triển khai được kỹ thuật này trên thực tế.



Hình 2.26 Kỹ thuật Streaming Video

Các bước thực hiện kỹ thuật streaming video:

- Phần mềm máy khách (media player, web browser, ...) cần kết nối được và xác định file video trên máy streaming server muốn xem.
- Yêu cầu streaming file video đó sẽ được gửi tới streaming server để tìm file video đó.
- Chương trình thực hiện streaming chạy trên máy streaming server sẽ chia file video thành các frame rồi gửi các frame đó tới máy yêu cầu sử dụng các giao thức ràng buộc về thời gian (RTSP, RTP, RTCP).
- Khi các frame về máy khách, sẽ được lưu trữ trong vùng đệm và nội dung các frame sẽ được giải mã (decode) và hiển thị thông qua các chương trình chơi video (ví dụ VLC).

Một số khái niệm được sử dụng trong streaming video:

- Streaming video (luồng video) thực chất là quá trình truyền các frame của file video tới người nhận.
- Demand streaming (stream theo yêu cầu) là quá trình streaming một file video có sẵn (đã được lưu trên ổ cứng) tới người nhận.
- Live streaming (stream từ một nguồn tạo video) là quá trình streaming trực tiếp từ các frame video được tạo ra từ các thiết bị thu nhận video (như camera) tới người nhận.
- H.264, VP8 là các thuật toán mã hóa cho các luồng video.
- Bitstream là khái niệm ám chỉ một luồng video từ máy chủ streaming tới máy khách nhận các frame video dựa vào giao thức MMS hay RTP.
- Codec: thuật ngữ ám chỉ chung cho các thuật toán mã hóa đường truyền trong quá trình streaming audio hay video.
- RTSP (Real Time Streaming Protocol) là giao thức mạng điều khiển quá trình streaming video hay streaming audio.

- RTP (Real-time Transport Protocol) là giao thức chuẩn định dạng cho gói tin (packet) video hay audio được truyền trên mạng.

Ưu điểm của tính năng video streaming:

- Chủ động thực hiện: Thời điểm trước, khi bạn thực hiện các thông điệp hay chiến dịch quảng bá thông thường nhờ vào các kênh nhà đài để phát sóng lên trực tiếp. Ngày nay với công nghệ live stream bạn có thể tự hoàn toàn làm chủ cuộc chơi, bất chấp mọi giới hạn về không gian và thời gian mà không phải lệ thuộc vào bất kỳ các yếu tố nào khác.

- Tối ưu hóa chi phí: Sự xuất hiện của công nghệ phát live stream đã giúp chúng ta có thể kết nối với nhau qua một thiết bị phổ biến nhất hiện nay, đó chính là smartphone. Khi đó, không cần bỏ nhiều chi phí quảng cáo, thuê khu vực hoặc các kênh truyền thống khác mà có thể tự mình thực hiện trên chính sản phẩm và địa điểm tại chỗ.

- Hiệu quả tương tác cao: Thời đại phát triển xã hội hóa mạnh mẽ, việc tiếp cận đến các đối tượng mà bạn mong muốn đã trở nên dễ dàng hơn bao giờ hết. Tính năng live stream giúp cho ta tăng khả năng nhận diện thương hiệu và đồng thời có thể tương tác, trao đổi trực tiếp với người dùng một cách nhanh chóng.

2.4.1 Giao thức RTSP

RTSP (Real Time Streaming Protocol) là một giao thức điều khiển trên mạng được thiết kế để sử dụng giao tiếp giữa máy client và máy streaming server. Giao thức này được sử dụng để thiết lập và điều khiển phiên giao dịch giữa các máy tính (end points).

Về hình thức giao thức RTSP cũng có nét tương đồng với giao thức HTTP, RTSP định nghĩa một bộ các tín hiệu điều khiển tuần tự, phục vụ cho việc điều khiển quá trình playback. Trong khi giao thức HTTP là giao thức không có trạng thái thì RTSP là giao thức có xác định trạng thái. Một định danh được sử dụng khi cần thiết để theo dõi các phiên giao dịch hiện tại của quá trình streaming video gọi là số hiệu session. Cũng giống như HTTP, RTSP sử dụng TCP là giao thức để duy trì một kết

nối đầu cuối tới đầu cuối và các thông điệp điều khiển của RTSP được gửi bởi máy client tới máy server. Nó cũng thực hiện điều khiển lại các đáp trả từ máy server tới máy client. Cổng mặc định được sử dụng bởi giao thức này là 554.

Để thực hiện kỹ thuật streaming video theo giao thức RTSP nhất thiết máy client phải gửi lên máy server (streaming server) những request sau và phải theo một trình tự nhất định.

Đầu tiên, máy client sẽ gửi yêu cầu OPTIONS kèm với đường link trỏ tới file video cần xem tới máy server, để máy server chấp nhận đường link này.

Nếu máy server trả về mã chấp nhận đường link trên thì máy client tiếp tục gửi yêu cầu DESCRIBE tới máy server để máy server phân tích đường link. Một yêu cầu DESCRIBE bao gồm một đường link RTSP có dạng (rtsp://) và kiểu dữ liệu đáp trả từ phía server. Cổng mặc định được sử dụng cho giao thức RTSP là 554 và cổng này được sử dụng cho cả giao thức của tầng giao vận UDP và TCP. Thông điệp đáp trả từ máy server cho yêu cầu DESCRIBE của máy client bao gồm bản tin miêu tả chi tiết phiên giao dịch (Session Description Protocol – SDP). Ngoài ra trong thông điệp trả về từ máy server còn liệt kê các đường link thích hợp hơn tới file video cần chơi khi mà trong file video đó có trộn lẫn giữa phụ đề và âm thanh. Và điều quan trọng nhất ở trong bản tin miêu tả phiên giao dịch này là streamid của luồng video và streamid của luồng âm thanh khi mà đoạn video đó có lồng âm thanh vào trong các frame.

Sau khi máy client nhận được thông điệp đáp trả từ máy server sau yêu cầu DESCRIPTION thì máy client sẽ tiếp tục gửi tiếp yêu cầu SETUP tới máy server. Một yêu cầu SETUP sẽ chỉ ra cách mà một dòng dữ liệu (single media stream) bắt buộc phải được truyền đi như thế nào. Và yêu cầu SETUP bắt buộc phải được hoàn thành trước khi một yêu cầu PLAY được gửi từ máy client. Yêu cầu SETUP bao gồm một đường link tới file video cần streaming và một thông tin đặc tả cho phần giao vận. Đặc tả này bao gồm 2 cổng trong đó có một cổng cục bộ trên máy client dành cho việc nhận các gói tin RTP (audio và video) và cổng còn lại dùng để nhận các gói tin RTCP (meta information). Máy server sẽ đáp trả lại bằng các xác nhận

các tham số đã được lựa chọn, và điền vào các phần còn thiếu ví dụ như máy server có thể chọn lại cổng của mình. Mỗi luồng dữ liệu sẽ được cấu hình cụ thể sau khi yêu cầu SETUP được hoàn tất trước khi máy client gửi yêu cầu PLAY.

Sau khi hoàn tất yêu cầu SETUP, cấu hình được các luồng dữ liệu để chuẩn bị streaming, máy client sẽ gửi yêu cầu PLAY để thực hiện truyền các frame dữ liệu thật sự từ máy server tới máy client, và các frame dữ liệu này sẽ được lưu trong một bộ đệm của máy client, các frame này sẽ được giải mã (decode), rồi được hiển thị bởi trình chơi file video và âm thanh (VLC). Yêu cầu PLAY bao gồm một đường dẫn trỏ tới file video cần phát giống như các yêu cầu trước đó. Đường link này có thể là đường tổng hợp (để phát các luồng dữ liệu) hoặc là một đường link đơn lẻ (chỉ phát một luồng dữ liệu duy nhất). Trong yêu cầu PLAY, máy client cũng sẽ chỉ ra một dải (range) chỉ rõ một cách cụ thể số hiệu frame bắt đầu được gửi và số hiệu frame kết thúc, Nếu như không chỉ rõ tham số này, thì toàn bộ các frame sẽ được gửi tới máy client. Và nếu như luồng dữ liệu có bị tạm dừng (pause) thì luồng dữ liệu này cũng sẽ được phục hồi ở frame mà nó tạm dừng truyền.

Trong quá trình streaming video, nếu như người dùng muốn tạm dừng quá trình streaming thì sẽ gửi yêu cầu PAUSE tới máy server, yêu cầu này sẽ làm tạm dừng một hay nhiều luồng dữ liệu đang truyền các frame về máy client. Máy server sẽ tạm dừng gửi các frame dữ liệu tới máy client.

Trong quá trình streaming video, nếu như người dùng muốn dừng hẳn quá trình streaming thì sẽ gửi yêu cầu TEARDOWN để dừng truyền và kết thúc một phiên giao dịch của giao thức RTSP. Máy server sẽ đáp trả lại thông điệp xác nhận cho yêu cầu TEARDOWN và sẽ dừng gửi các frame tới máy client.

2.4.2 Giao thức RTP

RTP (Real-time Transport Protocol) định dạng một gói tin RTP được dùng để truyền trên luồng dữ liệu video hay audio dựa trên địa chỉ IP. RTP được sử dụng trong phiên giao dịch giữa các hệ thống giải trí hoặc giao tiếp mà có triển khai kỹ thuật streaming video như là telephony, ứng dụng hội họp từ xa, hệ thống giám sát bằng hình ảnh dựa trên IP.

RTP được sử dụng kết hợp với giao thức RTCP (RTP Control Protocol). Trong đó, RTP được sử dụng để đóng gói các frame dữ liệu (audio và video) để truyền trên luồng dữ liệu thì RTCP được sử dụng để giám sát chất lượng của dịch vụ (QoS) hoặc để thống kê theo các tiêu chí trong quá trình truyền tải. Thường thì giao thức RTP sử dụng cổng có số hiệu chẵn còn giao thức RTCP sử dụng cổng có số hiệu lẻ.

RTP được thiết kế cho quá trình streaming theo thời gian thực từ theo kiểu điểm tới điểm. Giao thức này cung cấp tiện ích để dò ra những gói tin RTP đã quá hạn. Trên thực tế, gói tin RTP sử dụng địa chỉ IP trên mạng để định danh các máy tính gửi và nhận. RTP cũng hỗ trợ truyền dữ liệu tới nhiều điểm đích thông qua địa chỉ IP multicast.

RTP được phát triển bởi tổ chức Audio/Video Transport của tổ chức tiêu chuẩn IETF. RTP được sử dụng kết hợp với các giao thức khác và giao thức RTSP. Chuẩn RTP định nghĩa một cặp giao thức làm việc với nhau đó là RTP và RTCP. RTP được sử dụng để truyền tải dữ liệu đa phương tiện và giao thức RTCP được sử dụng để gửi các thông tin điều khiển với các tham số QoS.

Các giao thức thành phần: Đặc tả RTP gồm 2 giao thức con là RTP và RTCP.

Giao thức truyền, RTP, quy định cách thức truyền dữ liệu theo thời gian thực. Thông tin được cung cấp bởi giao thức này bao gồm thời gian đồng bộ (timestamps), số thứ tự gói tin (phục vụ cho việc tìm gói tin bị lạc) và chi phí cho việc mã hóa định dạng dữ liệu.

Giao thức điều khiển, RTCP được sử dụng cho việc kiểm tra chất lượng (QoS) luồng dữ liệu và thực hiện đồng bộ giữa các luồng dữ liệu. So với RTP, thì băng thông của RTCP sẽ nhỏ hơn, vào cỡ 5%.

Một giao thức cho phép miêu tả dữ liệu đa phương tiện nhưng không bắt buộc phải kèm theo là giao thức miêu tả phiên (Session Description Protocol – SDP).

Phiên (Session): Một phiên RTP được thiết lập cho mỗi luồng dữ liệu. Một phiên bao gồm một địa chỉ IP với một cặp cổng của giao thức RTP và RTCP. Ví dụ, các luồng video và audio sẽ có các phiên RTP khác nhau, bên nhận sẽ nhận một cách

riêng biệt giữa dữ liệu video và audio thông qua 2 cổng khác nhau cho 2 giao thức RTP và RTCP. Thường thì số hiệu cổng của RTP là một số chẵn trong khoảng 1024 tới 65535 và cổng của RTCP là một số lẻ kế tiếp.

Hình vẽ dưới đây là hình ảnh của một header của gói tin RTP

RTP packet header							
bit offset	0-1	2	3	4-7	8	9-15	16-31
0	Version	P	X	CC	M	PT	Sequence Number
32	Timestamp						
64	SSRC identifier						
96	CSRC identifiers						
	...						
96+32×CC	Profile-specific extension header ID					Extension header length	
128+32×CC	Extension header						
	...						

Hình 2.27 Header của RTP Packet.

Kích thước nhỏ nhất của một header của gói tin RTP là 12 bytes. Sau phần header chính, là phần header mở rộng và không cần thiết phải có phần header này. Chi tiết các trường trong một header như sau:

- Version (2 bits): Cho biết phiên bản của giao thức này. Phiên bản hiện tại là phiên bản 2.
- P (Padding) (1 bit): Cho biết số các byte mở rộng cần thêm vào cuối của gói tin RTP. Ví dụ trong trường hợp ta muốn sử dụng các thuật toán mã hóa, ta có thể thêm vào một số byte vào phần kết thúc của gói tin để tiến hành mã hóa frame trên đường truyền.
- X (Extension) (1bit): Cho biết có thêm phần header mở rộng vào sau phần header chính hay không.
- CC (CSRC Count) (4 bit): Chứa con số định danh CSRC cho biết kích thước cố định của header.
- M (Marker) (1 bit): Cho biết mức của ứng dụng và được định nghĩa bởi một profile. Nếu được thiết lập, có nghĩa là dữ liệu hiện tại đã được tính toán chi phí một cách thích hợp

- PT (Payload Type) (7 bit): Cho biết định dạng của file video. Đây là một đặc tả được định nghĩa bởi một profile RTP.
- Sequence Number (16 bits): số hiệu của frame. Và sẽ được tăng lên 1 đơn vị cho mỗi gói tin RTP trước khi gửi và được sử dụng bởi bên nhận để dò ra các gói bị lạc và có thể phục hồi lại gói có số thứ tự đó.
- Timestamp (32 bits): Được sử dụng thông báo cho bên nhận biết để phát lại frame này trong khoảng thời gian thích hợp.
- SSRC (32 bits): Định danh cho nguồn streaming. Mỗi nguồn cho phép streaming video sẽ định danh bởi một phiên RTP duy nhất.

2.5 Thiết bị bay không người lái.

2.5.1 Tổng quan về thiết bị bay không người lái

Với sự phát triển vượt bậc của khoa học công nghệ ngày nay, máy bay không người lái (UAV) là một trong những chủ đề nổi bật của khoa học công nghệ trên thế giới cũng như tại Việt Nam.



Hình 2.28 Thiết bị bay không người lái với rất nhiều hình dạng và cấu trúc.

Phương tiện bay không người lái hay Máy bay không người lái, viết tắt tiếng Anh là UAV (Unmanned Aerial Vehicle) là tên gọi chỉ chung cho các loại máy bay mà không có người lái ở buồng lái, hoạt động tự lập và thường được điều khiển

từ xa từ trung tâm hay máy điều khiển [40]. Một chiếc UAV được định nghĩa là một phương tiện di chuyển trong không trung, không có người lái, sử dụng lực khí động để cung cấp lực nâng, có thể bay tự hành hoặc được điều khiển từ xa, có thể thu hồi tái sử dụng hoặc không, có thể mang theo tải trọng hoặc không. Do đó, tên lửa không được coi là UAV vì chính nó được sử dụng làm vũ khí chứ không phải phương tiện vận chuyển, và không thể thu hồi để tái sử dụng, mặc dù nó cũng không có người lái và một số loại có thể điều khiển từ xa.

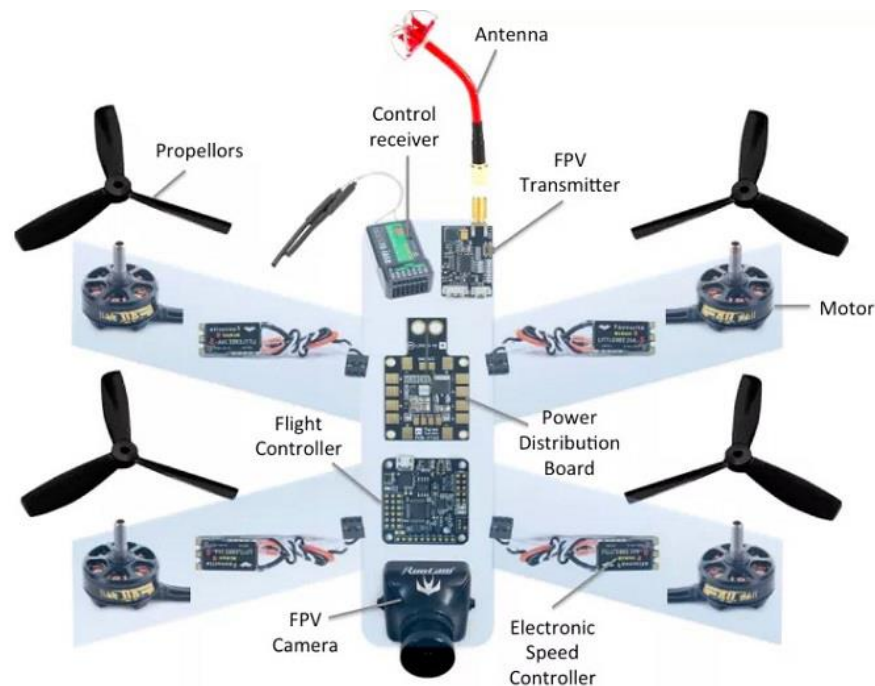
UAV hiện nay đang được sử dụng trong nhiều loại nhiệm vụ khác nhau. Có thể nói sự xuất hiện của các thiết bị bay không người lái UAV thực sự là cuộc cách mạng trong lĩnh vực thu thập số liệu, khảo sát, giám sát và theo dõi các đối tượng trên thực địa.

Theo sự phát triển công nghệ hiện có các dạng UAV:

- Máy bay theo nghĩa truyền thống được trang bị hệ thống điều khiển và lái tự động, được gọi là UAS (unmanned aircraft system), xuất hiện từ những năm 1950 và đã từng phục vụ việc do thám và trinh sát chiến trường. Loại tổ hợp máy bay này có khả năng tự động hóa các hoạt động của máy bay cao, không đòi hỏi những trang thiết bị hàng không đặc chủng, giá thành khai thác sử dụng và bảo trì hệ thống để phục vụ lâu dài rẻ, trong quân sự loại máy bay này có đặc tính tấn công chớp nhoáng.
- Phương tiện bay kiểu mới, được chế tạo rất đa dạng, có kích thước và công suất động cơ nhỏ đến trung bình, được gọi là drone.
- Các drone có lắp camera để quan sát, và thường được gọi là flycam. Để thuận tiện điều khiển thao tác thì drone có nhiều cánh quạt, thường là 4.

Ứng dụng của UAV drone đang tăng lên mạnh mẽ, từ các mục đích quân sự cho đến nghiên cứu khoa học, điện ảnh - truyền hình, nông nghiệp, thương mại, vận chuyển, giải trí. Tuy nhiên vấn đề an toàn bay UAV đang đặt ra cấp thiết, gồm sự uy hiếp của nó tới khu dân cư, và đến không gian hoạt động của hàng không quân dân sự hiện nay.

2.5.2 Cấu trúc một UAV cơ bản.



Hình 2.29 Cấu tạo Drone cơ bản [41]

Về cơ bản, một thiết bị bay không người lái sẽ có các thành phần sau:

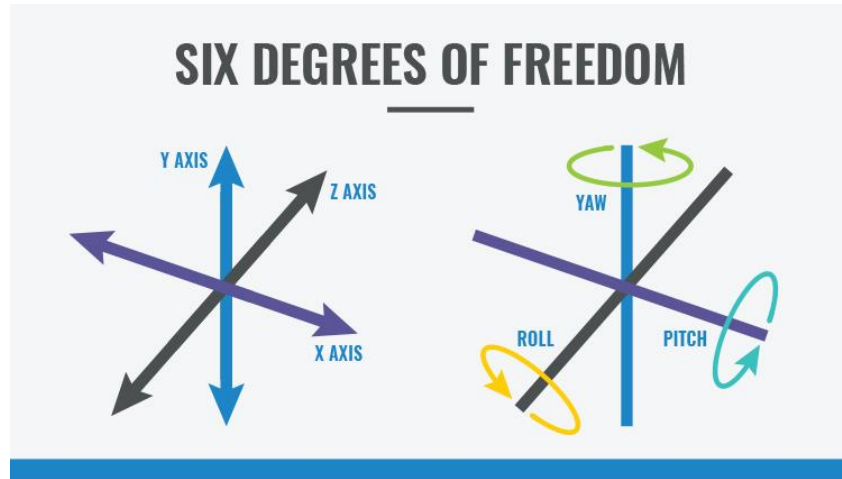
- Bộ điều khiển R/C: Bộ điều khiển này về lý thuyết khi không có vật cản nó có khả năng điều khiển trong bán kính 1km. Để điều khiển cho UAV cần phải có điều khiển 4 kênh trở lên. Các cần điều khiển gồm có qua trái, qua phải, lên, xuống, xoay trái máy bay, xoay phải máy bay, tự cân bằng...
- Board điều khiển chính: Board này chịu trách nhiệm nhận tín hiệu điều khiển từ joystick, đọc tín hiệu cảm biến cân bằng trên thân nó để điều khiển bốn động cơ ở 4 góc cho phù hợp, cảm biến cân bằng trên Board này rất hữu ích, nó giúp ta điều chỉnh cần điều khiển lúc ban đầu sao cho máy bay sẽ cân bằng. Các board hiện đại người ta còn tích hợp cả GPS trên đó giúp định vị máy bay hay tự động đáp...
- Bộ khung sườn: Bộ khung này được thiết kế sao cho 4 góc là hoàn toàn cân đối cả về khối lượng và kích thước, vị trí bắt ốc để máy bay được cân bằng. Động cơ được đặt ở bốn góc này phải hoạt động đồng bộ. Khi ta lắp ráp khung sườn cần phải đảm bảo tính đối xứng của nó.

- Động cơ và bộ điều xung: Sau khi board điều khiển nhận tín hiệu từ điều khiển và phát xung điều khiển ra board chính, bộ điều xung sẽ căng cứ vào độ rộng xung phát ra mà điều khiển tốc độ động cơ. Động cơ dành cho thiết bị bay là loại động cơ 1 chiều có tốc độ vòng tua lớn, có thể lên đến 750~1000 vòng/phút. Tiêu thụ dòng lên đến 35A. Đây cũng chính là lý do các loại thiết bị bay này chỉ có thể bay trong vòng 10~15 phút vì tiêu thụ công suất ở các động cơ là rất lớn, năng lượng nguồn cũng bị hao hụt nhanh hơn.
- Cánh máy bay: Đây là bộ phận cực kỳ dễ hư hỏng nếu bạn điều khiển không tốt làm cho máy bay bị rơi. Cánh máy bay thường làm bằng nhựa hoặc sợi Carbon. Bạn nên sử dụng loại cánh máy bay sợi carbon vì nó cực kỳ chuẩn trong thiết kế giúp cho máy bay cân bằng tốt và độ cứng rất cao sẽ giúp ít hư hỏng khi máy bay rơi.
- Năng lượng cung cấp: Pin Lipo với dòng lớn cỡ 30A hoặc pin Li-ion 5A.

Một thành phần cần được nhắc đến đặc biệt của một thiết bị bay cơ bản đó chính là bộ cảm biến IMU (Inertial Measurement Unit).

IMU là thiết bị được kết hợp từ hai bộ cảm biến là Accelerometer (cảm biến gia tốc) và Gyroscope (cảm biến con quay hồi chuyển) có chức năng cung cấp thông tin về góc quay và độ nghiêng giúp cho việc giữ thăng bằng của hệ thống tự động như robot vượt địa hình, máy bay [41]... Để có thể nắm được nguyên tắc hoạt động và cấu tạo của IMU chúng ta cần tìm hiểu cấu tạo và nguyên tác hoạt động của hai bộ cảm biến:

- Accelerometer (gọi tắt là accel): như tên gọi của nó, accel đơn giản là một cảm biến đo gia tốc của bản thân module và thường sẽ có 3 trục xyz ứng với 3 chiều không gian (loại 1 và 2 trục ít dùng). Lưu ý là accel đo cả gia tốc của trọng lực nên giá trị thực khi đo sẽ bao gồm cả trọng lực.
- Gyroscope (gọi tắt là gyro): là một loại cảm biến đo tốc độ quay của nó quanh một trục. Tương tự với accel, gyro cũng thường có 3 trục xyz.



Hình 2.30 Trục dịch chuyển của IMU (3 trục phẳng và 3 trục xoay) [43]

Một ví dụ đơn giản, khi đặt một con chip IMU thẳng đứng và để im không chuyển động, giá trị trả về sẽ là $\text{accel} = [0.0, -9.8, 0.0]$ và $\text{gyro} = [0.0, 0.0, 0.0]$ do chỉ có trọng lực trái đất tác dụng lực và không có bất cứ chuyển động quay nào cả. Lưu ý rằng gyro chỉ đo tốc độ quay chứ không đo trực tiếp góc quay, nên khi quay module một góc nào đó rồi dừng, giá trị của gyro sẽ tăng lên rồi hạ xuống về 0.

IMU - cũng như nhiều cảm biến khác - cần được hiệu chỉnh trước khi sử dụng nếu muốn có một kết quả đáng tin cậy. Sau đây là những vấn đề cần lưu tâm khi sử dụng một module IMU [44]:

Accelerometer: accel luôn có offset trên mỗi trục làm cho giá trị đo được thường lệch đi so với thực tế một chút. Ngoài ra, giá trị đo được theo accel thường khá nhiễu khiến cho việc đọc trở nên khó khăn. Với offset là hằng số thì đơn giản chỉ là đo lại giá trị đó và trừ vào giá trị đo. Với việc tín hiệu bị nhiễu thì có thể dùng một bộ lọc điện tử tần số thấp

Gyroscope: cũng như accel, gyro cũng có offset (hay còn gọi là bias) làm lệch các giá trị đo. Một vấn đề khác nữa có thể gặp phải của gyro là drift, có nghĩa là bias thay cũng thay đổi chậm theo thời gian. Dù vậy, điểm cộng là gyro lại ít bị nhiễu hơn accel. Vấn đề drift của gyro thì có thể dùng bộ lọc cao tần (digital high-pass filter) do gyro drift khá chậm.

Một cách chuyên nghiệp hơn, nếu các bạn muốn đo một cách chính xác góc quay thì có một giải pháp rất thông dụng là sensor fusion, có nghĩa là sẽ dùng cả gyro và accel để đo góc và dùng một số loại thuật toán để gộp 2 giá trị với nhau, bù trừ nhau để đưa ra kết quả chính xác nhất. Một số thuật toán thường dùng hiện nay là [44]:

- Complementary filter: loại này đơn giản nhất nhưng cũng khá hiệu quả.
- Kalman filter: hơi phức tạp nhưng phổ biến nhất và rất hiệu quả, kể cả trong những ứng dụng chuyên nghiệp.
- Mahony filter: khả năng ngang với Madgwick filter.
- Madgwick filter: cái này khá mới so với những cái trên nhưng rất hiệu quả.

IMU là thành phần không thể thiếu đối với một board điều khiển chính. Nó vừa giúp điều chỉnh độ cân bằng trong suốt quá trình bay cũng như thu thập các thông số hỗ trợ việc điều khiển bay.

Mỗi khi cất cánh, IMU sẽ gửi các thông số sang board điều khiển chính giúp thiết bị bay có thể cân bằng lúc ban đầu. Mọi hư hại cũng như sai số của IMU đối với thiết bị bay đều gây hại rất nhiều. Ví dụ như khi cất cánh, thay vì bay thẳng lên, thiết bị bay sẽ bay xiên góc hoặc thậm chí va chạm và không thể điều khiển được.

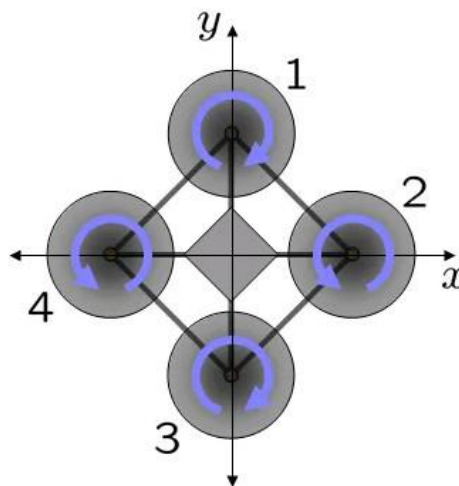
Sau khi hoàn tất một lệnh nào đó từ tay cầm điều khiển, ví dụ như bay sang trái, hoặc sang phải, hoặc xoay,.. khi ta thả cần điều khiển ra, IMU sẽ đóng vai trò chủ đạo trong việc đảo hướng điều khiển, tức là khi ta dừng lệnh bay sang trái, thiết bị bay vẫn sẽ còn quán tính để bay tiếp tục sang trái, điều thiết bị bay cần làm là phải điều chỉnh làm sao cho thiết bị triệt tiêu đi quán tính đó, nghĩa là phải bay đảo hướng sang phải hoặc ít nhất lực sinh ra cho việc bay ngược lại phải đủ để chống lại quán tính. Quá trình triệt tiêu quán tính có thể nhanh hay chậm tùy thuộc vào độ nhạy của thiết bị bay cũng như IMU và vận hành của board điều khiển chính. Vì thế đối với một số loại UAV vẫn sẽ bay thêm một đoạn nữa rồi mới dừng lại dù ta đã không điều khiển chúng bay hướng đó nữa.

2.5.3 Lý thuyết về hành vi bay của thiết bị bay UAV

Đa số các thiết bị bay cơ bản được cấu tạo có 4 cánh quạt, chúng đối xứng nhau cả về kích thước lẫn trọng lượng, tạo nên một sự cân bằng để chúng có thể dễ dàng bay lên. Tuy nhiên, khi chúng mang trên mình nhiều linh kiện hơn, làm cho sự cân bằng đó không được đảm bảo, vậy thì cánh quạt sẽ làm thay đổi điều đó, thiết lập lại cân bằng khi bay.

Vậy các thiết bị bay này hoạt động như thế nào? Ở phần này sẽ đề cập đến các thiết bị bay gồm 4 cánh quạt.

Trong 4 động cơ, 2 động cơ đối diện nhau sẽ quay ngược với 2 cái còn lại để tạo nên moment xoắn, giúp thiết bị có thể bay lên. Điều này có thể mô tả như hình bên.



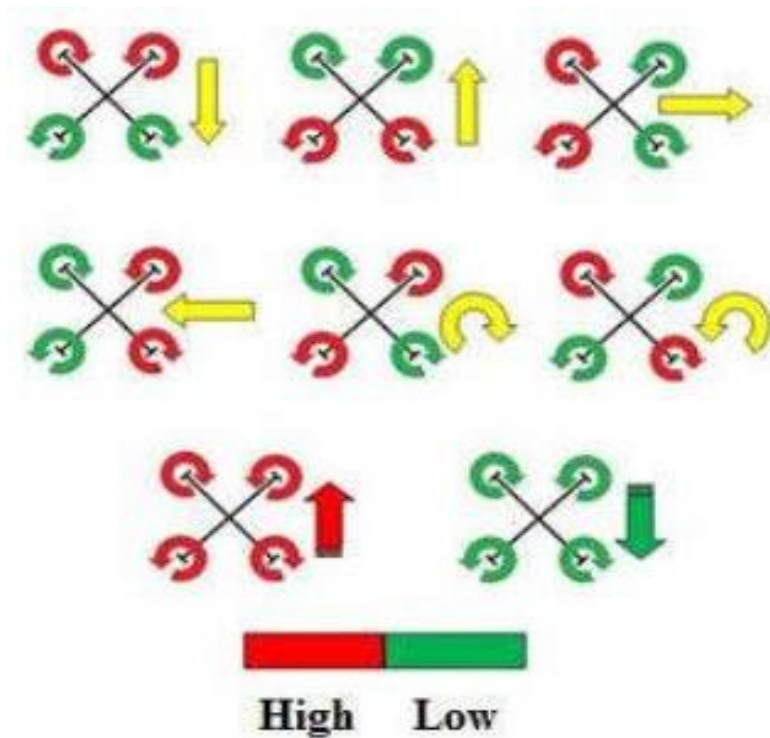
Hình 2.31 Hoạt động đối xứng nhau của động cơ giúp thiết bị có khả năng bay

Về hành vi bay của UAV, ta có thể chia thành 3 hướng cơ bản được mô tả ở hình 35:

- Độ cao (hai hình dưới cùng): Ở đây, máy bay sẽ bay điều chỉnh độ cao, nghĩa là bay lên cao, hoặc bay xuống thấp. Để làm được điều này, ta có thể tăng lực đẩy của toàn bộ cả 4 động cơ để bay lên và giảm lực đẩy của toàn bộ 4 động cơ để thiết bị bay xuống. Điều cần lưu ý của việc tăng giảm lực đẩy khi điều khiển tự động bằng chương trình là cần phải tiến hành chậm, tránh trường hợp tăng giảm không có bước đệm, ví dụ

như động cơ đang quay mức 1, cần nâng dần lên mức 2 3 4 rồi mới tăng lên 5, không nên từ 1 trực tiếp lên 5, dòng xả của động cơ thay đổi đột ngột có thể khiến hư hỏng hoặc cháy nổ không cần thiết. Tất nhiên, điều này ở điều khiển bay bằng tay cầm điều khiển ít khi xảy ra.

- Độ lệch theo hướng cố định: Ta có thể điều khiển thiết bị bay sang trái, sang phải bằng cách tăng lực đẩy vào hai động cơ để điều chỉnh cho chúng bay theo ý muốn. Ở hình 35, bốn hình đầu thể hiện điều này, theo thứ tự là bay lùi, tiến, bay sang phải và bay sang trái.
- Độ xoay: Thiết bị bay có thể xoay tại chỗ nhờ việc tăng lực đẩy ở một động cơ đồng thời giảm lực đẩy ở động cơ đối diện. Hai hình tiếp theo ở hình 35 (hình thứ năm và hình thứ sáu) thể hiện điều này. Xoay phải là hình thứ năm, còn xoay trái là hình thứ sáu.



Hình 2.32 Nguyên lý di chuyển của mô hình Quacopter [45]

Việc phối hợp một cách hợp lý và nhuần nhuyễn các động tác này của thiết bị làm cho hành vi bay của thiết bị có thể nhẹ nhàng, mềm mại hơn, giúp cho việc ghi nhận hình ảnh trở nên dễ dàng.

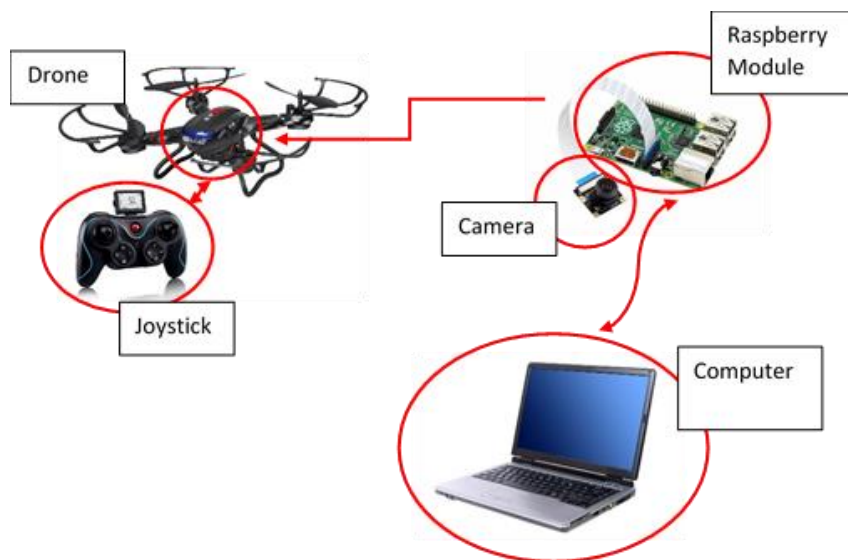
CHƯƠNG 3: THIẾT KẾ VÀ THỰC HIỆN MÔ HÌNH

3.1 Mô hình với drone tự lắp ráp, cân chỉnh hệ thống cân bằng và ổn định bay

3.2 Mô hình với drone hoàn thiện bởi nhà sản xuất DM107s

3.1 Mô hình với drone tự lắp ráp, cân chỉnh hệ thống cân bằng và ổn định bay

3.1.1 Tổng quan mô hình và ý tưởng thực hiện



Hình 3.1 Sơ đồ hệ thống dự tính thực hiện

Dựa trên việc tìm hiểu, tổng kết các vấn đề đặt ra và lần lượt giải quyết các vấn đề này, chúng em dự tính tạo ra một mô hình cơ bản như hình trên.

Hệ thống sẽ có 2 chế độ hoạt động: Chế độ điều khiển bằng tay, hoặc chế độ tìm kiếm và giám sát đối tượng tự động.

Đầu tiên, một thiết bị bay có bộ điều khiển chính kết nối với Raspberry Pi Module có trang bị camera chuyên dụng của module này. Raspberry Pi sẽ có trách nhiệm ghi nhận và gửi hình ảnh thu được từ camera về máy tính để máy tính xử lý, nhận diện đối tượng và gửi tín hiệu điều khiển trở về Raspberry Pi từ đó điều khiển drone. Như vậy, Raspberry Pi được coi như module truyền tải tín hiệu giữa drone và máy tính vì mô hình này không có bộ phát wifi.

Từ máy tính, giao diện người dùng có thể điều khiển bằng tay hoặc chế độ tìm kiếm tự động.

Joystick luôn sẵn sàng cho kết nối với thiết bị bay bằng switch trên thiết bị này phòng khi tín hiệu kết nối với máy tính bị ngắt hoặc chương trình điều khiển bị treo khiến drone mất kiểm soát. Từ đây, chế độ failsafe có thể được kích hoạt nhằm đảm bảo drone hạ cánh an toàn.

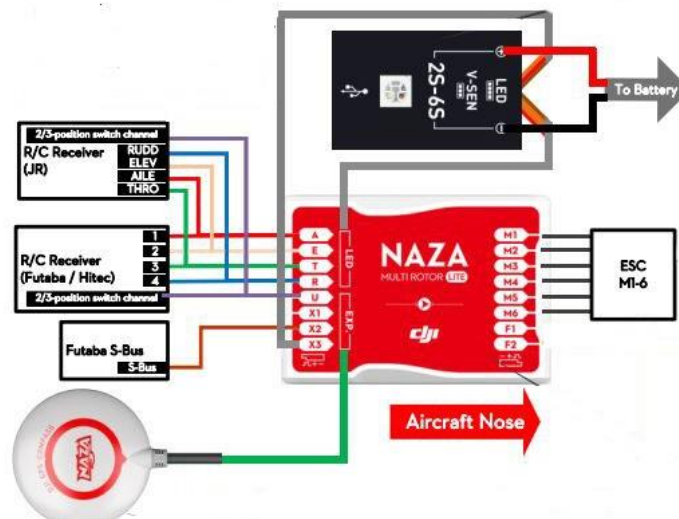
Bảng 2. Thông số hệ thống

<i>Các thành phần</i>	<i>Thông số kỹ thuật</i>
<i>Khung Drone</i>	Tarot Iron man 650
<i>Mạch chính</i>	Naza M-lite
<i>Động cơ</i>	980 KV
<i>Máy phát</i>	RX701
<i>Nguồn năng lượng</i>	Pin Lipo 5200 mAh
<i>Cánh quạt</i>	Emax 10x4.5
<i>Camera</i>	Raspberry Pi Camera Module V2 8 Megapixel, 1080p
<i>CPU</i>	Raspberry Pi 3
<i>Kết nối máy tính</i>	Kết nối không dây (Wifi)

Cụ thể, đề tài này sử dụng các linh kiện sản xuất bởi NAZA, bộ NAZA M-LITE:

- Bộ điều khiển chính (MC): Bộ điều khiển chính (MC) là bộ não của hệ thống, nó giao tiếp với tất cả các ESC và bộ phát RC để thực hiện chức năng lái tự động. Nó có một Đơn vị đo lường quán tính (IMU) tích hợp bao gồm một gia tốc kế 3 trục, một con quay hồi chuyển 3 trục và một áp kế để cảm nhận trạng thái và độ cao. (Có phần mềm hỗ trợ cài đặt thông số cho hệ thống bay do NAZA phát triển).
- Bộ đa năng (VU): Được thiết kế đặc biệt cho NAZA-M LITE. Nó giải quyết vấn đề tiêu thụ điện năng cao của hệ thống nhiều cánh quạt, cung cấp và giám sát nguồn điện cho NAZA-M LITE và các thiết bị điện tử khác. Nó cũng có đèn LED để chỉ ra các trạng thái hoạt động khác nhau của NAZA-M LITE và giao diện USB để định cấu hình đơn vị NAZA-M LITE và nâng cấp phần mềm.

- Mô-đun GPS & La bàn: Mô-đun GPS / La bàn dùng để cảm nhận vị trí và hướng. Vì GPS & La bàn nhạy cảm với nhiễu từ trường, vì thế nên sử dụng giá đỡ để gắn mô-đun GPS lên cao tách ra khỏi bộ phận nguồn.
- Động cơ: Dùng 4 động cơ không chổi than Sunnysky A2212 - 980KV, điện áp làm việc 3S-11.1V Lipo. Kèm theo đó là cánh quạt phù hợp với động cơ.
- Bộ khung TAROT IRON MAN 650 dùng chất liệu sợi carbon, vừa có độ dẻo và độ cứng phù hợp cho quá trình thực nghiệm các thí nghiệm vừa dễ dàng thay thế nếu có hư hại.
- RX701 và Devo 7: Tay cầm điều khiển DEVO 7 là bộ tay điều khiển 6 kênh, điều khiển từ xa một điều khiển đơn mà không có mạch nhận, khi kết hợp có thể thêm RX 601, 701, 801 hoặc RX1002.



Hình 3.2 Cách kết nối các linh kiện theo hướng dẫn của nhà sản xuất.



Hình 3.3 Mô hình UAV dùng cho đề tài.

3.1.2 Thử nghiệm và tiến hành cân chỉnh hệ thống

Với mô hình thiết bị bay được dùng cho đề tài, chúng em bắt đầu tìm hiểu cấu trúc, các vấn đề có liên quan đến thiết bị bay này, đối chiếu mô hình thực tế và các thành phần trong mô tả. Đây là thiết bị bay được lắp ráp bằng các thành phần rời, đã được dùng cho đề tài “Encoding Blocks for Digital Video of DVB-S2 Standard with Drone-based Communications” [46] của các sinh viên khóa trước thuộc trường Đại học Bách Khoa TP.HCM.

Vốn dĩ là một thiết bị của một đề tài khác vì thế nên thiết bị bay này được trang bị thêm một board mạch nặng với khung bảo vệ khiến thiết bị bay khá cồng kềnh. Hệ thống cân chỉnh thăng bằng của máy bay đã bị can thiệp điều chỉnh sao cho phù hợp với đề tài đó. Các thành phần được thêm vào là không cần thiết ở đề tài của chúng em, vì thế, việc loại bỏ là điều cần thiết.

Tuy nhiên, với sự can thiệp điều chỉnh hệ thống cân bằng bay của đề tài trước, việc tháo gỡ board mạch cũ làm cho thiết bị bay mất cân bằng, điều cần thiết lúc này là làm sao để drone có thể bay cân bằng trước khi thực hiện can thiệp vào hệ thống cơ bản của nó.

Nỗ lực tìm kiếm cách giải quyết cho vấn đề cân bằng của thiết bị bay tốn khá nhiều thời gian. Từ việc điều chỉnh 4 động cơ, đến cân chỉnh lại mạch chính nhờ cài đặt mạch điều khiển chính bằng phần mềm phát triển của hãng cùng với kiểm tra lại bộ khung của thiết bị bay cùng các bố trí của thành phần cho cân đối nhất, nhưng bộ khung của thiết bị bay khá mỏng manh, giòn và dễ gãy, sau khi được cho bay thử vài lần và rơi xuống đất khi đang bay do sự mất cân bằng của toàn bộ thiết kế, một vài chi tiết của khung đã bị hỏng và cần thay thế.

Nhờ sự giúp đỡ của một vài bạn của khoa Kỹ thuật Hàng không gợi ý rằng vấn đề có được tinh chỉnh tay cầm điều khiển, đổi vị trí của bộ GPS hoặc thiết lập lại chế độ bay. Sự giúp đỡ này thật sự có ích và giúp đỡ chúng em rất nhiều trong quá trình tìm hiểu và thực hiện luận văn, vì thế, chúng em thật sự muốn gửi lời cảm ơn chân thành đến các bạn ở khoa Kỹ thuật Hàng không của trường Đại học Bách Khoa Tp.HCM vì sự giúp đỡ của các bạn.

Với các gợi ý như trên, chúng em tìm hiểu kỹ hơn về cách giải quyết hiện trạng theo 3 hướng:

- Điều chỉnh lại tay cầm điều khiển.
- Cân chỉnh bộ GPS của máy bay.
- Các chế độ bay của máy bay.

a) *Điều chỉnh lại tay cầm điều khiển.*



Hình 3.4 Tay cầm điều khiển DEVO 7

Tay cầm điều khiển DEVO 7 là bộ tay điều khiển 6 kênh, điều khiển từ xa một điều khiển đơn mà không có mạch nhận, khi kết hợp có thể thêm RX 601, 701, 801 hoặc RX1002 (mô hình hiện tại được kết hợp với bộ thu RX701).

DEVO 7 là một trong các dòng TX RX thương mại của Walkera muốn tung ra để tham gia thị trường. Thiết kế lấy lại từ dòng 26xx, thay bằng màu đen, cầm nặng tay, các stick sử dụng bearing, có thể tùy chỉnh chiều dài stick. Màn hình rộng có đèn nền, có buzz nghe, pin 12v, ...

Thông số kỹ thuật bộ điều khiển DEVO 7 [47]:

- Bộ mã hóa: hệ thống vi điều khiển ARM
- Tần số: 2.4G (các DSSS)
- Công suất đầu ra: $\leq 100\text{mW}$
- Tín hiệu kiểm soát khoảng cách: 200 – 400m (tùy RX)
- Dòng tiêu thụ: $\leq 220\text{mA}$
- Tiêu chuẩn pin: 1.2V*8 pin nickel-cadmium hoặc 1.5V*1 pin 8AA
- Đầu ra xung: 900-2100mS (1500mS điểm trung lập)

Ưu điểm của DEVO 7:

- Thiết kế Bốn Chiều: Tiện nghi rocker đã được cải thiện hơn nữa, mỗi rocker đã thêm bốn vòng bi, mang lại cảm giác mềm mại hơn khi chỉ đạo kiểm soát.
- Sử dụng công nghệ cốt lõi của DEVO 12, tỷ lệ thực hiện đầy đủ tính năng, giá cả phải chăng.
- Để cung cấp máy bay trực thăng, cánh cố định hai loại của các loại mô hình. Cải thiện hơn trộn thiết kế và giao diện, như trộn chương trình, ga trộn.
- Điểm kiểm soát đường cong ga, mang lại tính tế hơn xử lý cảm giác.
- Có thể lưu trữ 15 máy bay mô hình có sẵn.
- Phần mềm nâng cấp trực tuyến qua mạng, rất đơn giản và dễ dàng để nâng cấp tính năng phần mềm miễn phí và các dịch vụ khác để tải về bản cập nhật các thông số mô hình.

Tinh chỉnh DEVO 7:

Việc tinh chỉnh cho tay cầm này khá phức tạp, về cơ bản, để hỗ trợ cho vấn đề bay của thiết bị ta cần phải điều chỉnh min và max của nút điều khiển để cải thiện độ nhạy của tay cầm. Song song với đó, việc điều chỉnh độ trôi của máy bay cũng là điều thật sự cần thiết. Để tinh chỉnh đầy đủ và cụ thể hơn có thể tham khảo bài viết [25].

b) Cân chỉnh bộ GPS của thiết bị bay [48]

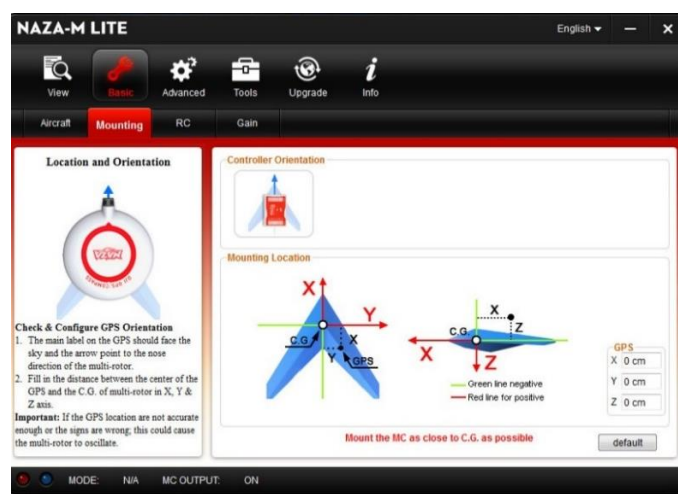


Hình 3.5 GPS Module của Naza.

Module GPS của Naza là thành phần đi kèm của bộ điều khiển bay DJI NAZA-M LITE, nó giúp cho việc điều khiển bay ổn định, tăng khả năng cân

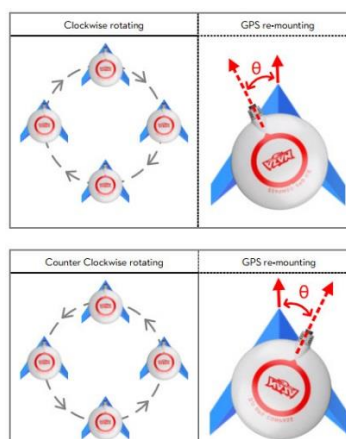
bằng của máy bay, ổn định vị trí chính xác sẽ giúp nâng cao hiệu suất chụp ảnh trên không. Điều khiển bay với module GPS giúp ta có thể bay ở chế độ GPS với nhiều chức năng an toàn khi điều khiển. Chẳng hạn như chức năng trở về điểm Home được thiết lập sẵn khi sắp hết pin, mất kết nối tay điều khiển,...

Trong quá trình sử dụng module này, trước hết cần phải thiết lập chính xác vị trí nó chính xác đối với bộ điều khiển chính, việc đặt này có hỗ trợ trên ứng dụng DJI NAZA-M Lite Assistant của hãng sản xuất upload cho người dùng của họ trên website.



Hình 3.6 Cài đặt GPS cho thiết bị bay [48].

Sau khi cài đặt thông số chỉ vị trí của GPS module so với bộ điều khiển chính, tiến hành bay thử nghiệm ở chế độ bay với GPS và tinh chỉnh lại vị trí của GPS module nếu phát hiện máy bay bị quay hoặc bị trôi trong khi lơ lửng giữa không trung trong thời gian không điều khiển.

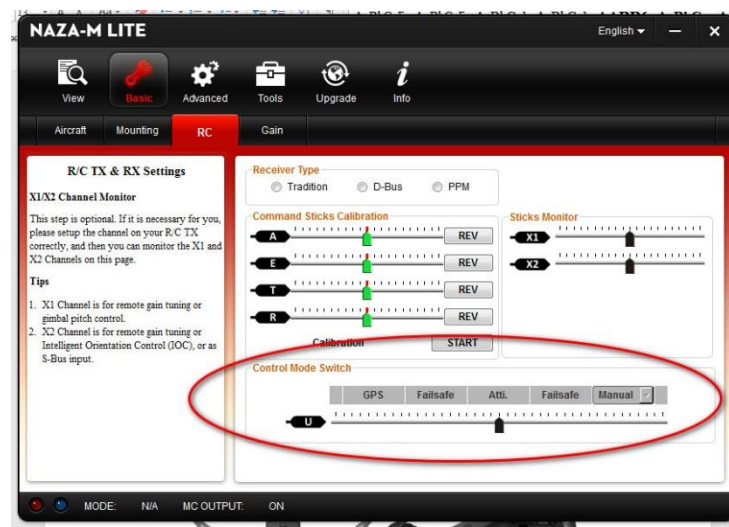


Hình 3.7 Tinh chỉnh hướng của GPS Module [48]

Khi máy bay bị quay cùng chiều kim đồng hồ, chỉnh hướng của GPS module lệch trái góc $10-30^0$, ngược lại, khi máy bay bị trôi ngược chiều kim đồng hồ, tinh chỉnh hướng của GPS Module lệch phải góc $10-30^0$. Cố gắng tinh chỉnh góc phù hợp đến khi thấy thiết bị bay đứng yên tại chỗ khi cho máy bay lơ lửng trên không.

c) Các chế độ bay của thiết bị bay

Bộ xử lý chính của thiết bị bay cung cấp cho người dùng 4 chế độ bay: Chế độ bay với GPS (GPS mode), chế độ bay Attitude (Atti. mode), chế độ bay thủ công (Manual mode) và chế độ bay hạ cánh an toàn (Failsafe mode).



Hình 3.8 Các chế độ bay của thiết bị [48]

Đề bay ở chế độ bay với GPS, tín hiệu GPS phải đủ, đèn tín hiệu ở trạng thái chớp xanh hoặc xanh đỏ tùy lượng tín hiệu GPS. Người dùng được khuyến khích bay ở chế độ này vì nó là an toàn và ổn định khi bay nhất. Chế độ này ít bị ảnh hưởng bởi gió, bay khá chậm và tiết kiệm pin nhất trong ba chế độ có thể bay.

Chế độ bay Atti. không phụ thuộc vào GPS, nó điều thiết bị bay nhanh hơn chế độ GPS, tối ưu hóa tốc độ động cơ vì thế dễ tăng tốc. Tuy nhiên, nó cũng có nhược điểm là không ổn định, dễ bị trôi khi gặp trời gió vì không có GPS; nếu điều khiển động cơ bay nhanh liên tục kéo theo lượng năng lượng tiêu hao sẽ cực kỳ lớn khiến cho pin nhanh chóng bị tổn hao, thời gian bay ngắn (2-5 phút). Ở chế độ này, đèn hiệu chớp vàng hoặc đỏ vàng tùy tín hiệu GPS.

Chế độ thủ công (Manual mode) dành cho người chuyên nghiệp, đã điều khiển máy bay thuần thục, có kinh nghiệm bay và biết cách giải quyết nhanh các tình huống khi bay. Khi vào chế độ này, đèn tín hiệu không bật, hoặc chớp đỏ tùy tín hiệu GPS. Chế độ này rất khó điều khiển thiết bị bay. Và trùng hợp, ban đầu, thiết bị bay được đặt ở chế độ này khi bàn giao thiết bị cho chúng em, từ đó xảy ra nhiều sự cố trong quá trình làm quen thiết bị, ảnh hưởng lớn đến tiến độ của luận văn.

Chế độ bay cuối cùng là hạ cánh khẩn cấp khi không an toàn (Failsafe). Ở chế độ này, đèn tín hiệu chuyển sang chớp vàng liên tục, tay cầm điều khiển mất quyền điều khiển, bộ điều khiển chính bắt đầu chương trình tự hạ cánh an toàn. Trong chương trình này, có hai hướng để máy bay hạ cánh (tùy người dùng cài đặt trong phần Failsafe setting của DJI NAZA Lite Assistant). Hướng đầu tiên, trở về nhà (Home Point) và hạ cánh (Go-Home and Landing), trong cài đặt này, thiết bị bay sẽ được nâng độ cao khoảng 10m, nhận tín hiệu GPS và quay trở về điểm Home được thiết lập sẵn hoặc quay trở lại địa điểm bắt đầu cất cánh. Hướng cài đặt này khuyến khích khi tín hiệu GPS là đủ và khi cần phải bay thiết bị ở nơi khuất tầm nhìn hay phải bay xa để mất điều khiển đối với drone. Khi tín hiệu GPS không đủ hoặc không có, độ lệch địa điểm tùy thuộc vào vị trí của drone. Hướng thứ hai, chỉ hạ cánh (Landing), ở cài đặt này, thiết bị bay hạ dần độ cao rồi hạ cánh an toàn. Hướng này được khuyến khích khi thiết bị bay chỉ bay trong tầm nhìn, có thể theo dõi drone khi bay, phòng trường hợp thiết bị hạ cánh và ta không thể tìm thấy chúng.

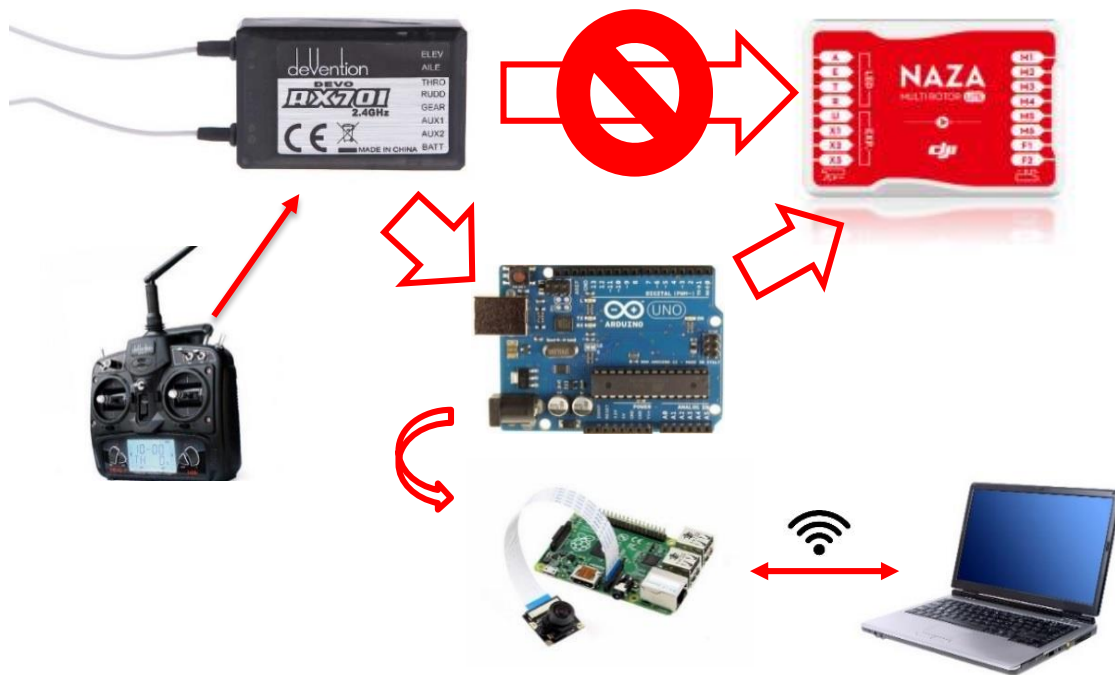
Control Mode (GPS)				
	Manual	Att.	GPS Att.	IOC
GPS satellites < 5	● ● ●	● ● ● ●	● ● ● ●	● ● ● ●
GPS satellites = 5	● ●	● ● ●	● ● ●	● ● ● ●
GPS satellites = 6	●	● ●	● ●	● ● ●
GPS satellites > 6	No	●	●	● ●
Attitude status bad	■	■ ●	■ ●	■ ●
Control Mode				
	Manual	No		
Att.	● ● ● ● ● ● ● ●			

Hình 3.9 Đèn tín hiệu khi bay ở các chế độ [48]

Như vậy, để đảm bảo an toàn cho cả người và thiết bị bay, trước khi cất cánh, ta phải kiểm tra xem thiết bị bay đang ở chế độ nào, điều chỉnh đúng chế độ cần thiết rồi mới được phép cất cánh.

3.1.3 Can thiệp quá trình truyền tín hiệu giữa RX701 và MC

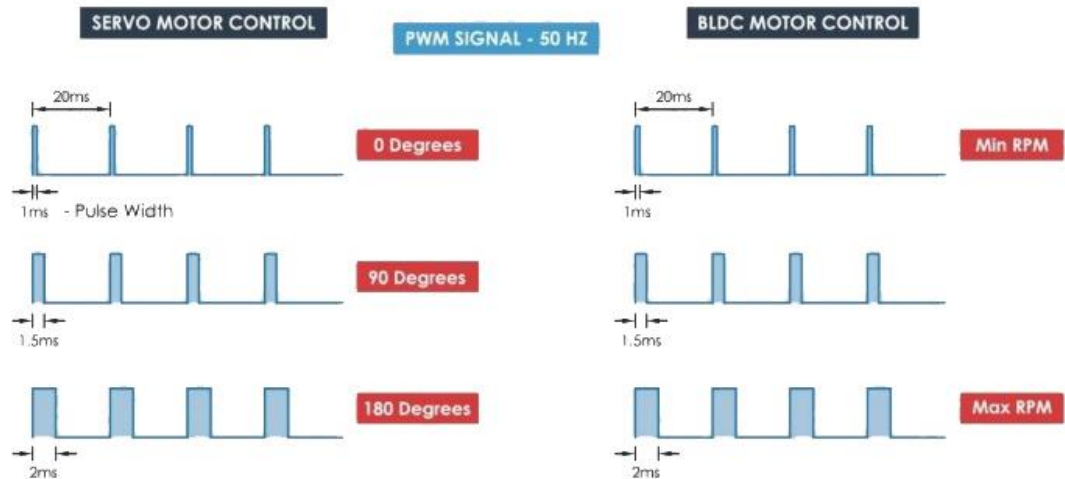
Ý tưởng ban đầu cho mô hình là dùng Raspberry Pi xử lý tất cả mọi thứ, kể cả truyền tín hiệu video, xử lý ảnh và điều khiển bay. Tuy nhiên, có 2 vấn đề cản trở việc này, thứ nhất, nếu chỉ dùng Raspberry Pi, khối lượng công việc là quá lớn dẫn tới tốc độ xử lý chậm chạp; thứ hai, mức điện áp hoạt động của bộ RX701 và các port của Raspberry Pi là khác nhau đưa đến việc khó khăn trong chuyển tiếp tín hiệu.



Hình 3.10 Sơ đồ cho việc truyền tín hiệu

Do đó, hướng giải quyết được đưa ra là giao việc chuyển tiếp tín hiệu (ở chế độ điều khiển bằng tay) hoặc giả tín hiệu từ tay cầm điều khiển để điều khiển hành vi bay, hay việc chọn mode hoạt động cho mô hình cho kit Arduino Uno, Raspberry Pi sẽ chỉ chịu trách nhiệm truyền hình ảnh về máy tính, nhận lệnh sau khi xử lý xong truyền đến arduino rồi đưa đến bộ điều khiển MC.

Bắt đầu thực hiện với việc đọc các chân tín hiệu của RX701 khi sử dụng joystick. Hiểu càng rõ về cách thức hoạt động của nó ta càng dễ dàng giả các tín hiệu này để điều khiển tự động theo ý muốn.



Hình 3.11 Tín hiệu PWM điều khiển động cơ tương tự tín hiệu nhận được từ tay cầm điều khiển

Tín hiệu truyền đến từ joystick và điều khiển động cơ đều dùng tín hiệu điều khiển PWM. Joystick về cơ bản gồm 4 kênh tương ứng với 2 cần gạt, mỗi bên đều có thể đẩy lên xuống hay trái phải tương ứng với tăng giảm cho các kênh. Cần gạt bên trái điều khiển 2 kênh làm cho drone bay lên, xuống và xoay trái phải; cần gạt bên phải điều khiển 2 kênh còn lại bay tiến, lùi và bay sang trái, phải. Ngoài ra còn các nút điều khiển các kênh khác như Gear, Mux,... ta có thể dùng các kênh này để chuyển đổi chế độ bay cho mô hình (bằng tay hoặc tự động).

Đo giá trị của các kênh thu được kết quả như sau:

Bên trái:

Khi đẩy lên cao nhất: THRO trên RX701 cho giá trị PWM là 1900us.

Khi đẩy xuống thấp nhất: THRO trên RX701 cho giá trị PWM là 1100us.

Khi đẩy qua trái nhất: RUDD trên RX701 cho giá trị PWM là 1900us.

Khi đẩy qua phải nhất: RUDD trên RX701 cho giá trị PWM là 1100us.

Bên phải:

Khi đẩy lên cao nhất: ELEV trên RX701 cho giá trị PWM là 1900 us.

Khi đẩy xuống thấp nhất: ELEV trên RX701 cho giá trị PWM là 1100us.

Khi đẩy qua trái nhất: AILE trên RX701 cho giá trị PWM là 1900 us.

Khi đẩy qua phải nhất: AILE trên RX701 cho giá trị PWM là 1100 us.

Các nút gạt:

GEAR có 3 mức: 1860 us cho vị trí cao nhất, 1384 us ở giữa, 1192 us vị trí dưới cùng.

AUX1: 1900 us ở vị trí cao nhất, 1500 us ở vị trí chính giữa và 1100 us cho vị trí dưới cùng.

Mux (tương ứng với AUX2): vị trí cao là 1100 us (1.1 ms), vị trí thấp là 1900 us (1.9 ms)

Từ các giá trị này, có thể mô phỏng các hành vi bay của drone bằng cách phối hợp các kênh tương ứng.

3.1.3.1 Tạo mạng Wifi cục bộ để truyền tín hiệu với Raspberry Pi

Ta cần một cách nào đó để giao tiếp điều khiển giữa máy bay và máy tính đang dùng để xử lý nhận dạng. Raspberry Pi có vẻ là lựa chọn lý tưởng vì dễ tìm kiếm, dễ lập trình trên nền ngôn ngữ Python và nhỏ gọn.

a) Tạo mạng Wifi cục bộ

Raspberry Pi sử dụng chipset BCM43438 để thực hiện các tác vụ liên quan đến wifi. Chip này hỗ trợ mạng wifi 2.4 GHz và 5 GHz theo các chuẩn b/g/n/ac với cấu hình anten đơn. Ta sẽ tạo một điểm truy cập bằng cách sử dụng hostapd, tạo một tệp cấu hình trong /etc/hostapd/hostapd.conf chứa các định nghĩa tự giải thích sau:

```
interface=wlan0
```

```
ssid=dronehq  
  
hw_mode=g  
  
ieee80211n=1  
  
channel=11  
  
wmm_enabled=1  
  
macaddr_acl=0  
  
ignore_broadcast_ssid=0
```

Interface là thiết bị để phát wifi, ssid là tên wifi mà điện thoại sẽ nối vào, channel là kênh wifi, *hw_mode* và *ieee80211n* kết hợp tạo chế độ của wifi là chuẩn n. Sau đó ta cần tạo ra một máy chủ DHCP để cung cấp địa chỉ IP cho bất kỳ máy nào kết nối tới Raspberry Pi. Một chương trình ta có thể dùng là dnsmasq [52].

Cho drone bay lên và ta kết nối vào mạng này qua máy tính để kiểm tra đường truyền video thông qua OpenCV hoặc mở qua môi trường dòng lệnh (FFMPEG, GStreamer). Lúc ở dưới đất, hình ảnh được truyền tốt, không bị đứt đoạn. Tuy nhiên khi bay lên trời, hình ảnh được truyền về luôn bị chậm hay bị vỡ hạt, khựng.

Ta còn có thể thay thế wifi có sẵn trên Raspberry Pi bằng 1 thiết bị wifi ngoài, trường hợp ở đây sử dụng USB wifi. Ta sử dụng USB wifi Samsung WIS12ABGNIX từ Smart TV thế hệ 2012.



Hình 3.12 USB wifi của Samsung Smart TV

USB wifi này sử dụng chipset Ralink RT3572, hỗ trợ 2 mạng wifi 2.4 GHz và 5 GHz, hỗ trợ luôn các chuẩn b/g/n/ac. Khác với Raspberry Pi, USB này có cấu hình anten đôi, tăng gấp đôi băng thông [53]. Vì được cắm ở ngoài, nó có thể phát wifi tốt hơn là mạch bên trong Pi. Với cách làm như ban đầu nhưng thay wlan0 thành wlan1, vì wlan1 là tên gọi của thiết bị trong Raspbian, hệ điều hành của Raspberry Pi. Ta vẫn tạo được mạng giống như trước, nhưng kết quả không khả quan hơn. Dù cho được vùng phủ sóng rộng hơn, khi cho drone bay lên, hình ảnh truyền về vẫn vỡ hạt, hay khựng. Chuyện này xảy ra có thể do các anten bên trong không thể truyền xa được hay truyền tốt khi máy bay lên cao.

Ta có thể chuyển sang cách khác là sử dụng thiết bị chuyên về phát sóng wifi và tạo bộ lặp wifi để kết nối tất cả các thiết bị với nhau (cách tạo bộ lặp wifi sẽ đề cập đến ở phần dưới, mục 3.2). Tenda AC10 được lựa chọn vì có độ phủ sóng rộng hơn so với các router khác. Nó hỗ trợ cả 2 mạng 2.4 GHz và 5 GHz với các chuẩn b/g/n/ac. Ta sẽ lựa chọn mạng 5 GHz với chuẩn ac để đạt được tốc độ cao nhất và chống nhiễu do trên băng tần không bị nghẽn như băng tần 2.4 GHz.

Sử dụng USB wifi để kết nối với mạng tốt hơn so với dùng chip wifi bên trong Raspberry Pi vì anten bên ngoài sẽ dễ bắt sóng hơn.



Hình 3.13 Router Tenda AC10

b) Truyền tín hiệu điều khiển thông qua Raspberry

Một chương trình được viết để nhận thông tin điều khiển từ máy tính thông qua Wifi và chuyển sang cho Arduino Uno thông qua cổng USB.

Chương trình này bao gồm 2 luồng chạy song song. Luồng thứ nhất `recvmsg()` là luồng đợi thông tin được gửi đến Raspberry Pi qua các gói tập tin UDP.

```
class UDPMsg:

    [...]

    def recvmsg(self):

        while not self.shutdown_thread:

            self.received = False

            data, addr = self.sock.recvfrom(1024)

            self.received = True

            self._udp_timer = time()

            self.string = data

            self.string_print = data.decode()

        def check_msg(self):

            if (not self.received and (time() - self._udp_timer >= 2)):

                self.in_contact = False

            else:

                self.in_contact = True

        return self.in_contact
```

Nếu lệnh `recvfrom(1024)` không nhận được gói UDP nào từ máy tính gửi sang, nguyên cả chương trình con `recvmsg()` sẽ được giữ lại và biến `received` sẽ là `False`, dẫn đến `in_contacted` được `False`.

```
get_text = UDPMsg(5005).start()
```



```

while True:

    if get_text.check_msg():

        print(time(), get_text.string_print)

        strsend = get_text.string

    else:

        strsend = bytes('8888','ascii')

    por.write(strsend)

    por.flush()

    if por.in_waiting > 0:

        strrecv = por.readline().decode('ascii','ignore')

        print(strrecv)

    sleep(0.05)

```

Luồng thứ hai là luồng chính, quyết định thông tin được gửi đến Arduino. Theo như biến từ luồng thứ nhất là *in_contacted* được False, luồng thứ hai sẽ truyền “8888” sang cho Arduino, tức là giữ drone ở vị trí cân bằng.

Nếu lệnh *recvfrom(1024)* có nhận được gói UDP nào từ máy tính gửi sang, nó sẽ được pass trực tiếp sang cho Arduino qua lệnh *write()*. Lệnh *flush()* là để đợi cho write hoàn tất.

3.1.3.2 Mô phỏng tín hiệu PWM điều khiển bay thông qua Arduino Uno

Để có thể đọc được các giá trị từ bộ thu RX701, ta bật chế độ ngắt của Arduino Uno.

```
PCICR |= (1 << PCIE0);
```

```
PCMSK0 |= (1 << PCINT0);
```

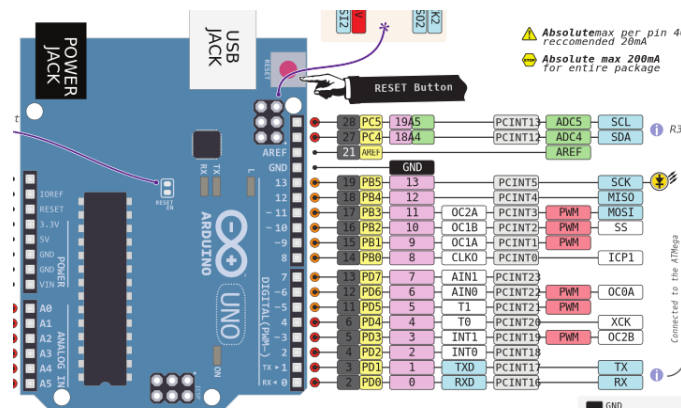
```
PCMSK0 |= (1 << PCINT1);
```

```
PCMSK0 |= (1 << PCINT2);
```

```
PCMSK0 |= (1 << PCINT3);
```

```
PCMSK0 |= (1 << PCINT4);
```

Dòng đầu để bật chế độ Interrupt (ngắt) khi giá trị trong chân thay đổi. Các dòng kế theo để xác định chân nào sẽ dùng để ngắt, cụ thể là PCINT0 đến 4 tương ứng với chân 8 đến 12 trên Arduino Uno R3.



Hình 3.14 Chi tiết vị trí chân trên Arduino Uno R3 (Đen: GND; Vàng: chân theo PORTD, PIND, PORTB, PINB; Hồng: chân theo chương trình IDE; Xám: chân theo interrupt ngoài) [50]

Mỗi khi có một chân có giá trị thay đổi, từ 0 lên 1 hay ngược lại, nó sẽ chạy một chương trình ngắt.

```
ISR(PCINT0_vect){

    current_time = micros();

    //Channel 1=====

    if(PINB & B00000001){

        if(last_channel_1 == 0){

            last_channel_1 = 1;
```

```
        timer_1 = current_time;

    }

}

else if(last_channel_1 == 1){

    last_channel_1 = 0;

    receiver_input_1 = current_time - timer_1;

    if(receiver_input_1 > 2500) receiver_input_1 = center;

}

//Channel 2=====

if(PINB & B00000010 ){

    if(last_channel_2 == 0){

        last_channel_2 = 1;

        timer_2 = current_time;

    }

}

else if(last_channel_2 == 1){

    last_channel_2 = 0;

    receiver_input_2 = current_time - timer_2;

    if(receiver_input_2 > 2500) receiver_input_2 = center;

}

[...]
```

Chương trình ISR (Interrupt Service Routine – trình dịch vụ ngắt) là chương trình sẽ chạy khi các chân từ 8 đến 13 thay đổi giá trị. Khi chạy, nó lấy số giây của thời gian hiện tại vào biến `current_time`. Các chân từ 8 đến 12 sẽ có giá trị trong PINB (do chân 8 đến 12 thuộc bank B của chip ATmega328). PINB sẽ dùng phép AND với từng bit trong 5 bit cuối, tương đương chân 12, 11,... 8 để kiểm tra xem chân nào đổi sang 1. Chân đó sẽ được lưu lại số bit tạm thời (`last_channel_x`) cùng với thời gian thu được bit đó (`timer_x`). Lần kích kế theo sẽ lấy số bit ngược lại cùng với hiệu số thời gian giữa hai lần lấy số bit (`receiver_input_x`). Nếu như hiệu số vượt quá ngưỡng (khoảng 2500) thì được dời về chính giữa điều khiển là 1500 để tránh ảnh hưởng đến việc bay.

Ta nối các chân từ 8 đến 12 với các chân tín hiệu của RX701 cho chế độ bay bằng tay:

- Chân 8 nối với THRO
- Chân 9 với RUDD
- Chân 10 với ELEV
- Chân 11 với AILE
- Chân 12 với MUX2

Từ các giá trị `receiver_input_x` ở chân từ 8 đến 11, ta tiến hành lặp lại tín hiệu trên các chân từ 4 đến 7. Riêng chân 12 được dùng để xác định chế độ bay theo Serial từ Raspberry Pi hay theo chân 8 đến 11.

```
if (receiver_input_5 > 1600){

    PORTB /= B00100000;

    getFromSerial();

}

else {

    PORTB &= B11011111;

    throttle = receiver_input_1;
```

```

yaw = receiver_input_2;

pitch = receiver_input_3;

roll = receiver_input_4;

}

loop_timer = micros();

PORTD |= B11110000;

timer_channel_1 = throttle + loop_timer;

timer_channel_2 = yaw + loop_timer;

timer_channel_3 = pitch + loop_timer;

timer_channel_4 = roll + loop_timer;

while(PORTD >= 16){

    esc_loop_timer = micros();

    if(timer_channel_1 <= esc_loop_timer) PORTD &= B11101111;

    if(timer_channel_2 <= esc_loop_timer) PORTD &= B11011111;

    if(timer_channel_3 <= esc_loop_timer) PORTD &= B10111111;

    if(timer_channel_4 <= esc_loop_timer) PORTD &= B01111111;

}

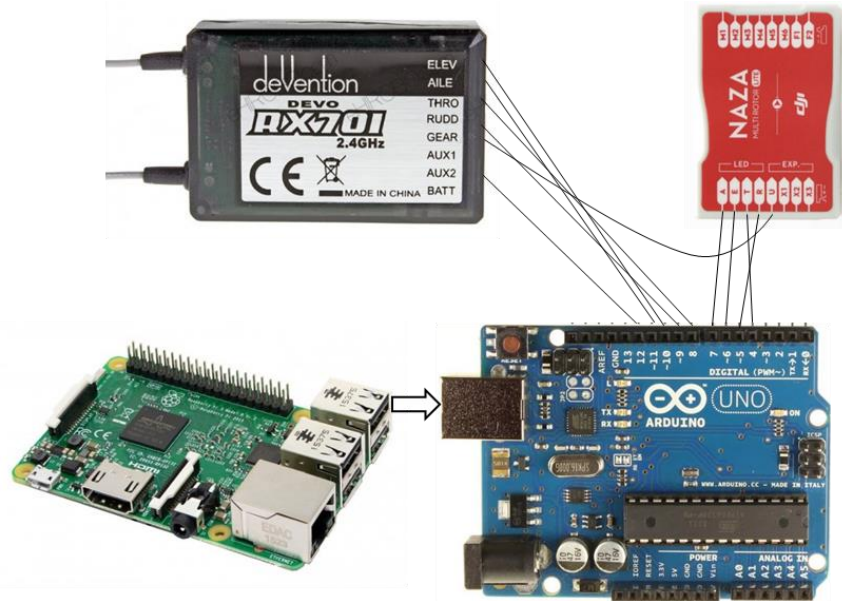
}

```

Ta lấy trường hợp ở chế độ bay bằng tay. Các giá trị receiver_input_x được cộng vào với số giây thời gian hiện thời, tạo thành timer_channel_x. timer_channel_x là thời điểm để chuyển các bit trên PORTD từ 1 sang 0, với PORTD là giá trị ra của bank D của chip ATmega328, tương ứng chân 0 đến 7 của Arduino Uno R3. Bit cao nhất của PORTD là bit ra của chân 7,

kế theo là 6 và 5... Việc đẩy bit lên xuống theo receiver_input_x tạo ra tín hiệu PWM trên chân 4 đến 7 để nối tới mạch điều khiển bay, thay thế cho việc nối với RX701. Với cả 2 chế độ:

- Chân 4 là throttle, nối với chân T của mạch điều khiển bay.
- Chân 5 là yaw, nối với chân R của mạch điều khiển bay.
- Chân 6 là pitch, nối với chân E của mạch điều khiển bay.
- Chân 7 là roll, nối với chân A của mạch điều khiển bay.



Hình 3.15 Nối Arduino với RX701 và mạch điều khiển bay [51]

Nếu chân 12 đặt ở chế độ bay theo Raspberry Pi, Arduino sẽ chạy một chương trình con để giải mã tin nhắn gửi từ Raspberry Pi sang.

```
void getFromSerial(){
    if (Serial.available() > 0) {
        incoming = Serial.readString();
        incoming.toCharArray(inctochar, 5);
        if (inctochar[0] == 's'){
            throttle = 1100;
            yaw = 1100;
```

```
pitch = 1100;

roll = 1900;

}

else {

for (int i=0; i < 4; i++) {

if (inctochar[i] >= 'a' & inctochar[i] <= 'f'){

tempvalue[i] = int(inctochar[i]) - 'a' + 10;

}

else if (inctochar[i] >= '0' & inctochar[i] <= '9') {

tempvalue[i] = int(inctochar[i]) - '0';

}

else tempvalue[i] = 8;

}

throttle = tempvalue[0]*25 + 1300;

yaw = 1700 - tempvalue[1]*25;

pitch = tempvalue[2]*25 + 1300;

roll = 1700 - tempvalue[3]*25;}}}
```

Raspberry Pi sẽ luôn truyền cho Arduino 4 chữ số thập lục phân như “0000”, “98ab”, “45db”. Chương trình con này dịch các tin nhắn nhận từ cổng Serial/USB của Arduino và chuyển nó thành các giá trị giống như các giá trị receiver_input_x để cho ra chân từ 4 đến 7. Nếu được nhận tin là “st”, Arduino dịch ra lệnh cho máy bay khởi động 4 động cơ lên.

3.1.4 Truyền video thời gian thực từ Raspberry về máy tính.

3.1.4.1 Thư viện truyền FFMPEG.

Raspberry Pi Camera là module được sản xuất chuyên dùng cho các dòng Raspberry, với các phiên bản khác nhau, tốc độ ghi hình, chất lượng hình ảnh là khác nhau. Tùy thuộc mục đích sử dụng có thể chọn cho phù hợp. Thư viện truyền video cơ bản của OpenCV là FFMPEG.

Raspberry Pi Camera có thể xuất video theo 2 định dạng: MJPEG và H264. Mỗi định dạng có đặc điểm riêng biệt.

+ MJPEG chất lượng hình ảnh tốt, dung lượng cao, truyền rất tốn băng thông, video stream theo giao thức http, đạt chuẩn thời gian thực.

Sử dụng chương trình raspivid để lấy hình ảnh từ camera theo định dạng MJPEG, sau đó đẩy qua VLC truyền qua giao thức http:

```
raspivid -n -cd MJPEG -t 0 -v -w 1280 -h 720 -fps 30 -b 8000000 -o -  
/ cvlc stream:///dev/stdin --sout '#standard{access=http, mux=mpjpeg,  
dst=:8081}'
```

Bên PC cần nhập vào OpenCV: 'http://địa_chi_Pi:8081/'

Tuy nhiên, điểm yếu của cách này là cần băng thông truyền rất cao, 8 Mbps (Megabit/s). Khi cho drone bay lên cao, hình ảnh bị giật và khựng do truyền không kịp, mặc dù truyền dưới đất đạt được thời gian thực.

+ H264 chất lượng hình ảnh giảm, dung lượng thấp, tốc độ truyền nhanh, truyền theo giao thức http hoặc rtsp, bị delay không thể truyền theo thời gian thực.

Sử dụng chương trình raspivid để lấy hình ảnh từ camera theo định dạng H264, sau đó đẩy qua VLC truyền qua giao thức http hay rtsp:

```
RTSP: raspivid -n -t 0 -v -w 1280 -h 720 -fps 30 -b 2000000 -o - / cvlc  
-vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8081/}' --demux=h264 -  
-h264-fps=30
```



```
HTTP: raspivid -n -t 0 -v -w 1280 -h 720 -fps 30 -b 2000000 -o - | cvlc
-vvv stream:///dev/stdin --sout '#standard{access=http, mux=ts,
dst=:8081}' --demux=h264 --h264-fps=30
```

Bên PC cần nhập vào OpenCV: 'http://địa_chỉ_Pi:8081/' hay 'rtsp://địa_chỉ_Pi:8081'

Cách này giúp giảm băng thông cần để truyền video. Tuy nhiên, điểm yếu của cách này là không đạt được phản ứng theo thời gian thực, có thể vì VLC không thể truyền đủ nhanh.

3.1.4.2 Thư viện truyền GStreamer

GStreamer là một khuôn khổ đa phương tiện dựa trên đường ống liên kết với nhiều hệ thống xử lý phương tiện khác nhau để hoàn thành các quy trình công việc phức tạp. Ví dụ, GStreamer có thể được sử dụng để xây dựng một hệ thống đọc các tệp ở một định dạng, xử lý chúng và xuất chúng ở một định dạng khác. Qua thực hiện khảo sát cho thấy, Gstreamer truyền tốt hơn nhiều so với khi sử dụng thư viện FFMPEG.

Ta chuyển sang thư viện GStreamer để truyền hình ảnh từ drone cho PC. Raspivid sẽ được truyền qua Gstreamer để tạo một server trên Pi để được kết nối và nhận hình ảnh ra.

Để có thể sử dụng GStreamer, ta cần các thư viện này được cài sẵn: gstreamer1.0-tools, gstreamer1.0-plugins-good, gstreamer1.0-plugins-bad, gstreamer1.0-plugins-ugly, gstreamer1.0-libav. Trong Ubuntu, ta dùng lệnh `sudo apt update <tên_phần_mềm>` để cài đặt.

Có thể thông qua 3 giao thức là TCP, UDP hoặc RTSP để truyền video.

a) Thông qua TCP

Có thể dùng 2 dòng lệnh cơ bản để thực hiện truyền video theo thư viện Gstreamer như sau:

```
raspivid -n -t 0 -v -w 1280 -h 720 -fps 30 -b 2000000 -o - | gst-launch-
1.0 -v fdsrc ! h264parse ! rtph264pay config-interval=1 pt=96 ! gdppay
! tcpserver sink host=0.0.0.0 port=8081
```

Qua dòng lệnh này, GStremaer sẽ tạo một kết nối đến Pi để xem video.

```
gst-launch-1.0 tcpclientsrc host=< địa_chỉ_Pi > port=8081 !
gdpdepay ! rtph264depay ! avdec_h264 ! videoconvert ! autovideosink
sync=false
```

Qua cách này, video truyền không bị trễ nhiều như trước nữa.

b) Thông qua UDP

```
raspivid -n -rot 180 -t 0 -a 512 -v -w $width -h $height -fps $fps -b $br
-o - / gst-launch-1.0 -v fdsrc ! h264parse ! rtph264pay config-interval=1
pt=96 ! udpsink host==< địa_chỉ_máy_nhận > port=8081
```

GStreamer sẽ tạo một kết nối đến Pi để xem video.

```
gst-launch-1.0 udpsrc port=8081 ! rtph264depay ! decodebin !
autovideosink sync=false
```

Với giao thức truyền UDP, video truyền nhanh hơn giao thức TCP một chút, giảm delay rất nhiều, có thể coi như truyền song song với thời gian thực. Tuy nhiên, cách này yêu cầu phải biết địa chỉ của PC nhận, không như cách TCP sẽ tạo một server để nối vào.

Trong OpenCV, không có cách để sử dụng thư viện GStreamer để lấy hình ảnh. Do vậy ta cần compile lại OpenCV cho phù hợp với nhu cầu [52]. Từ đây ta dùng các dòng lệnh này để sử dụng OpenCV với thư viện GStreamer:

```
cv2.VideoCapture(src, cv2.CAP_GSTREAMER)
```

với

```
src = "tcpclientsrc host=192.168.4.101 port=8081 ! gdpdepay !
rtph264depay ! avdec_h264 ! videoconvert ! appsink sync=false" đối với
TCP.
```

```
src = "udpsrc port=8081 ! application/x-rtp, payload=96 !
rtppjitterbuffer ! rtph264depay ! avdec_h264 ! appsink sync=false" đối với
UDP.
```

Tuy nhiên, khi đi qua OpenCV chương trình có bị một chút delay về hình ảnh.

c) Thông qua RTSP

Còn cách nữa mà ta có thể thực hiện là tạo lại RTSP thông qua GStreamer. Thông qua [53] để compile ra được GStreamer RTSP Server, ta tạo một máy chủ (server) RTSP trên Raspberry Pi.

Nhập dòng này để kích hoạt server RTSP:

```
./test-launch --gst-debug=3 "( rpicamsrc bitrate=2000000
preview=false ! video/x-h264, width=1280, height=720, framerate=30/1
! h264parse ! rtph264pay name=pay0 pt=96 )"
```

Sau đó với OpenCV ta có cách truy cập bằng FFMPEG như trước: 'rtsp:// địa_chỉ_Pi:8554/test'; hoặc sử dụng thư viện GStreamer:

```
rtspsrc location=rtsp://< địa_chỉ_Pi >:8554/test latency=0 buffer-
mode=auto ! decodebin ! videoconvert ! appsink sync=false
```

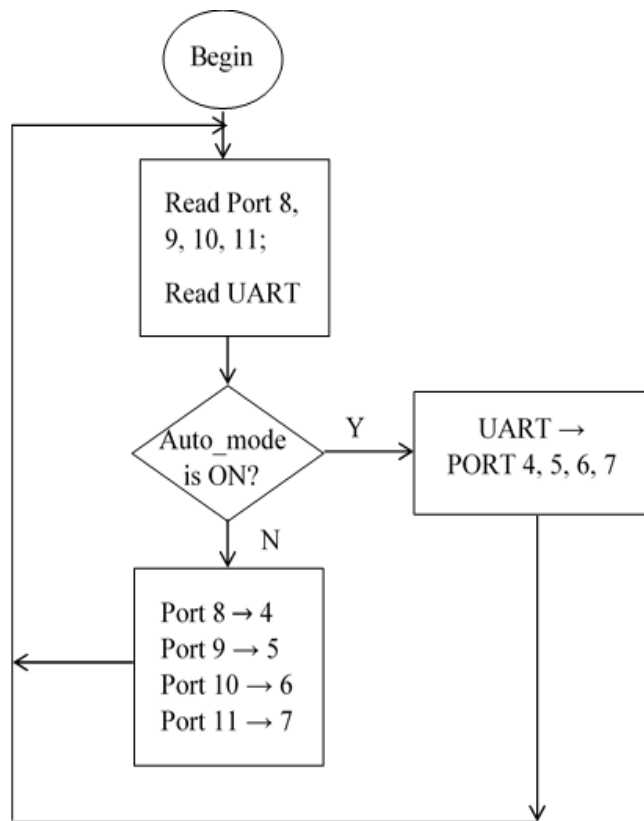
Thông qua RTSP từ GStreamer, video được truyền theo thời gian thực.

Ta cũng có một cách khác

```
gst-launch-1.0 -v rpicamsrc bitrate=2000000 preview=false !
video/x-h264, width=1280, height=720, framerate=30/1 ! h264parse !
rtph264pay config-interval=1 pt=96 ! gdppay ! tcpserver sink
host=0.0.0.0 port=8081
```

Tuy nhiên, hình ảnh luôn bị vỡ ảnh có thể do PC nhận sai thông tin quá nhiều làm cho hình ảnh không còn nguyên vẹn nữa. Kết hợp với tăng nhanh tốc độ truyền dữ liệu bằng module wifi có thể giảm thiểu vấn đề này. Trong dòng lệnh, tùy theo chất lượng hình ảnh mong muốn cũng có thể tăng hoặc giảm bitrate theo mục đích sử dụng. Định dạng áp dụng ở đây là .h264.

3.1.5 Lưu đồ giải thuật



Hình 3.16 Lưu đồ giải thuật cho việc chuyển tiếp dữ liệu

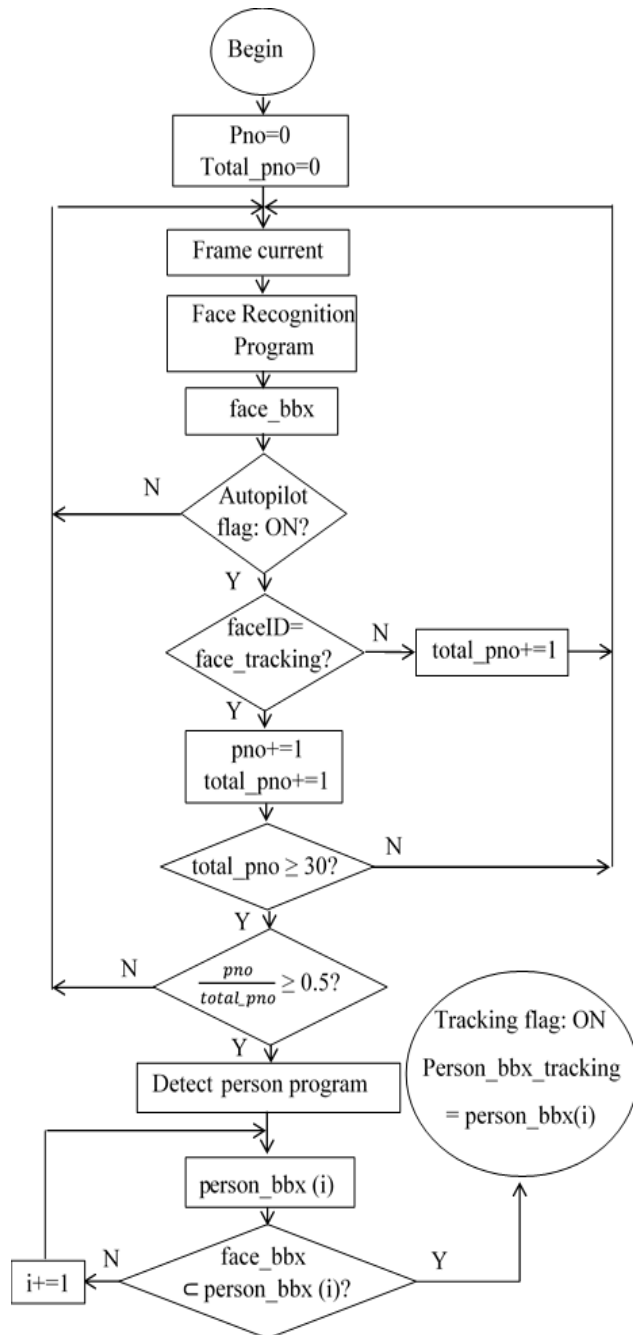
Ở vấn đề chuyển tiếp dữ liệu, ta cần phải xem xét chế độ bay hiện thời đang là chế độ nào. Với chế độ bay bằng tay, Auto_mode tắt, RX701 nhận tín hiệu điều khiển từ DEVO 7 sau đó truyền sang Arduino, Arduino sẽ chỉ chuyển tiếp sang board điều khiển chính MC để điều khiển thiết bị bay. Với chế độ bay tự động, Auto_mode bật, Arduino ngừng nhận tín hiệu của 4 kênh điều khiển chính từ RX701 thay vào đó, nó sẽ dịch code nhận được ở UART từ Raspberry Pi sau đó truyền vào các PORT 4, 5, 6, 7 để đưa sang board điều khiển chính MC để điều khiển bay.

Việc chuyển tiếp dữ liệu không thật sự ổn định như trong dự tính. Các tín hiệu ban đầu từ điều khiển chuyển sang MC có sự ổn định cơ bản để không xảy ra sự cố khi bay. Tuy nhiên, mức điện áp làm việc của Arduino đối với các tín hiệu PWM nhận trong thời gian thực cũng như việc tạo ra xung PWM ở PORT 4 5 6 7 để điều khiển bay không thật sự ổn định, con số hiển thị mức độ xung tạo ra dao động trong một khoảng lớn hơn rất nhiều so với tín hiệu điều khiển thật. Về cơ bản

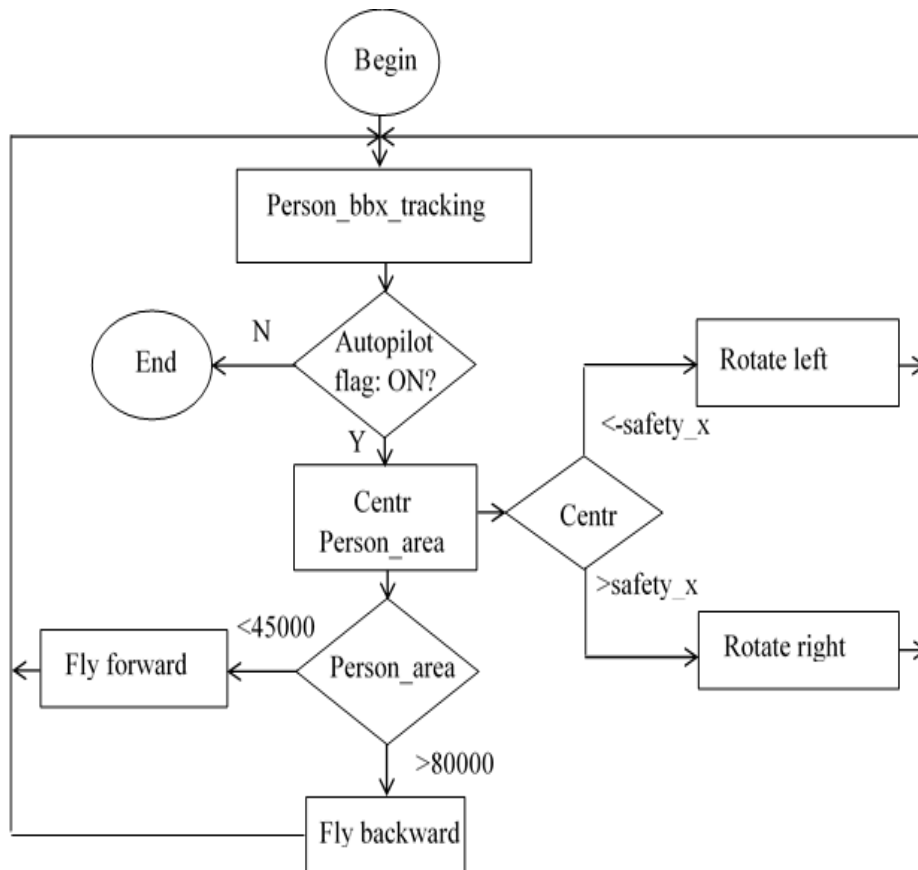
thì xung này vẫn có thể điều khiển thiết bị bay, tuy nhiên, thỉnh thoảng vẫn sẽ gặp trường hợp bay không ổn định do điều này.

Nhận dạng đối tượng và điều khiển bay là hai vấn đề cốt lõi của đề tài này. Trong việc nhận dạng đối tượng, lại có hai việc cần phải làm. Trước hết, việc đầu tiên cần làm nhận dạng khuôn mặt, xem thử đối tượng này có phải đối tượng cần theo dõi hay không. Mạng FaceNet sẽ làm việc với khuôn mặt đã được huấn luyện sẵn, so sánh và đưa ra ID cho đối tượng này. Việc nhận dạng có xác suất chính xác khá cao, tuy nhiên, trong một vài trường hợp vẫn xảy ra việc nhận dạng sai, vì thế ta cần phải lấy mẫu trong một số khung hình đã xử lý, nếu kết quả nhận dạng là đúng đối tượng với xác suất cao hơn ngưỡng đã đặt (ở đây là 50% số khung hình xử lý) thì chuyển sang bước thứ hai, nhận dạng người. Mạng YOLOv3 sẽ được khởi động và đưa ra các hộp giới hạn khoảng vùng các đối tượng được xác định

là “người”. Từ kết quả của FaceNet, ta xét xem tọa độ của khuôn mặt đã được nhận dạng có nằm trong hộp giới hạn người nào hay không. Nếu có, khóa đối tượng và tiến hành tracking.



Hình 3.17 Lưu đồ giải thuật nhận dạng



Hình 3.18 Lưu đồ giải thuật điều khiển bay

Điều khiển bay luôn là vấn đề khó khăn nhất được xác định từ khi bắt đầu đề tài này. Để giải quyết vấn đề này, trước hết cần phải trả lời được các câu hỏi đặt ra: Điều khiển bay dựa trên bao nhiêu kênh? Dùng phương pháp nào để điều khiển? Chỉ tiêu đặt ra là gì?

Trước hết ta phải xác định khoảng an toàn với mục đích là vừa luôn giữ hình ảnh của đối tượng trong khung hình của mình, vừa có thể giữ được an toàn cho người đang được giám sát cũng như an toàn cho thiết bị bay (khoảng này được cho là ở giữa khung hình). Từ tọa độ hộp giới hạn của người đang được tracking, so sánh với tọa độ của khoảng an toàn, kết hợp với diện tích hộp giới hạn của người này (vấn đề này liên quan đến xử lý hình ảnh với độ sâu), ta có thể tiến hành tạo xung PWM gửi đến các PORT của board điều khiển chính MC trực tiếp điều khiển thiết bị bay theo yêu cầu của đề tài.

Tuy phương pháp thực hiện điều khiển chỉ dựa trên những yếu tố cơ bản nhưng vẫn đạt kết quả khá khả quan, hoàn thành được mục tiêu đề ra.

3.2 Mô hình với drone hoàn thiện bởi nhà sản xuất DM107s

3.2.1 Drone DM107s



Hình 3.19 Drone DM107s

Đây là một máy bay không người lái có thể gấp lại. Nó có mức tăng tốc gấp đôi để chụp ảnh trên không rất nhanh. Các 2 ống kính được trang bị trên các UAV cho phép bạn chuyển đổi quan điểm của bạn bất cứ lúc nào. Hơn thế nữa, nó hỗ trợ VR, chỉ cần một chiếc điện thoại di động và phần mềm điều khiển của hãng, ta có thể kết nối trực tiếp thiết bị bay và theo dõi video truyền về điện thoại cũng như điều khiển bay bằng điện thoại không cần thông qua tay cầm.

Các tính năng chính:

- Giữ độ cao: cho phép UAV lơ lửng trong không trung, cho phép chụp ảnh rõ nét dễ dàng.
- 2 mức tăng tốc: cho phép chụp ảnh tốc độ rất cao.
- 2 camera: một camera trước và một camera dưới, dễ dàng chuyển đổi góc nhìn.
- Chức năng lật 360 độ: bay nhào lộn trên không.
- Quay trở lại vị trí ban đầu: drone sẽ trả lại vị trí bay ban đầu, chuyển bay an toàn và dễ dàng hơn.
- Tải xuống APP: Mã QR trên sách hướng dẫn.
- Thiết kế có thể gấp lại, dễ dàng mang theo.
- Kích thước: 17 x 14,5 x 6cm (gấp lại), 40 x 30 x 6cm (mở ra).

Chiếc drone DM107s nhỏ gọn, dung lượng pin khá lớn thuận lợi cho việc sử dụng cho đề tài (thời gian bay khoảng 10 phút bay liên tục cho 1 viên pin 1000mAh, pin rời, vì thế nên dễ dàng thay đổi khi hết pin cũng như sạc lại cho các pin cũ). Điều cần thiết ở thiết bị này chính là nó đã được phát triển ứng dụng điều khiển bay bằng điện thoại, điều này là mấu chốt cho ý tưởng thiết lập mô hình thứ hai này.

3.2.2 Ý tưởng cho mô hình.



Hình 3.20 Ý tưởng cho mô hình 2

Mô hình này liên quan mật thiết đến thiết lập kết nối có sẵn đã được phát triển giữa drone và ứng dụng giao tiếp trên điện thoại di động. Ta có thể hiểu rằng để kết nối thiết bị bay và điện thoại, thiết bị bay sẽ phát ra tín hiệu wifi, điện thoại sẽ kết nối với wifi đó và giao tiếp giữa chúng sẽ được thiết lập, đồng thời kết nối yếu này cũng cho phép truyền gửi tín hiệu điều khiển từ điện thoại đến thiết bị bay cũng như truyền video từ thiết bị bay về điện thoại.

Ý tưởng cho mô hình ở đây sẽ là ta can thiệp kết nối giữa chúng, tạo một trạm trung chuyển tín hiệu là máy tính xách tay. Từ máy tính ta có thể làm cách nào đó để thay điện thoại điều khiển thiết bị bay cũng như ghi nhận tín hiệu video truyền về. Dựa trên video nhận được, ta phân tích đối tượng trong video, xử lý để đưa về đề tài của ta, điều khiển thiết bị bay theo dõi đối tượng được nhận dạng. Như vậy ta có thể hoàn thành mục tiêu đề tài thông qua mô hình này.

3.2.3 Can thiệp kết nối giữa drone và điện thoại

Đối tượng thực hiện ở đây là các drone có sẵn, đã phát triển chương trình điều khiển trên điện thoại di động. Các drone chỉ có điều khiển bằng tay cầm điều khiển hoặc các drone chỉ phát triển chương trình để nhận video từ camera gắn trên drone trên điện thoại di động mà không có điều khiển bay trực tiếp trên điện thoại thì không nằm trong đề tài nghiên cứu. Cũng chính vì thế nên ta chọn Drone DM107s.

Việc đầu tiên cần quan tâm là kết nối giữa drone và điện thoại, làm sao để bắt được các lệnh được gửi từ điện thoại đến drone và luồng video theo hướng ngược lại từ drone về điện thoại.

Ý tưởng đầu tiên là sử dụng bộ lập wifi để mở rộng phạm vi kết nối của drone. Điều này có nghĩa là máy tính sẽ đóng vai trò là một bộ lập wifi, đứng giữa kết nối của drone và điện thoại di động, từ đó ta có thể kiểm tra, phân tích các lệnh được gửi đi từ điện thoại cũng như gửi về từ drone, đồng thời, ta có thể tìm cách nào đó để giả tạo các lệnh này để điều khiển drone trực tiếp từ máy tính, từ đó phát triển chương trình điều khiển tự động drone theo ý muốn.

Việc tạo một bộ lập wifi khá đơn giản [49].

Đầu tiên, đóng trình quản lý mạng bằng cách thực hiện:

```
# /etc/init.d/network-manager stop
```

Sau đó, ta có thể làm cho máy tính xách tay của mình kết nối với drone bằng giao diện có sẵn trên Ubuntu.

Hiện tại có 2 thiết bị nối wifi trong máy tính: wlo1(trong) và wlxc83a35c239b0 (ngoài). Ta dùng wlo1 nối với drone qua wifi có AP M8_1080P[...]ccaef ([...] là không rõ, mỗi lần bật lên lại khác), còn lại để phát truy cập.

Bước tiếp theo là tạo một điểm truy cập bằng thẻ Wi-Fi thứ hai trong máy tính xách tay. Điều này có thể được thực hiện bằng cách sử dụng hostapd. Cái này không được cài đặt trong máy tính theo mặc định, vì vậy phải cài đặt thông qua

apt install hostapd. Sau đó, tạo một tệp cấu hình trong /etc/hostapd/hostapd.conf chứa các định nghĩa tự giải thích sau:

```
interface=wlsx83a35c239b0  
ssid=ForQuad  
hw_mode=g  
ieee80211n=1  
channel=11  
wmm_enabled=1  
macaddr_acl=0  
ignore_broadcast_ssid=0
```

Interface là thiết bị để phát wifi, ssid là tên wifi mà điện thoại sẽ nối vào, channel là kênh wifi, hw_mode và ieee80211n kết hợp tạo chế độ của wifi.

Tệp cấu hình hoàn toàn không tồn tại và ta phải sửa đổi một tệp cấu hình khác để làm cho hostapd tải cấu hình. Điều này được thực hiện bằng cách chỉnh sửa tệp /etc/default/hostapd để bao gồm dòng sau:

```
DAEMON_CONF = "/ etc / hostapd / hostapd.conf"
```

Khi đã xong, có thể khởi động mạng Wi-Fi "lắp lại" bằng cách khởi động daemon hostapd theo cách Debian:

```
#/etc/init.d/hostapd start
```

Có thể kiểm tra trạng thái của hostapd bằng cách sử dụng lệnh "status" thay vì "start" trong init.d. Đầu ra dự kiến sẽ nói rằng dịch vụ đang hoạt động không có lỗi. Khi điều này được kích hoạt, mạng "ForQuad" của mạng Wi-Fi sẽ hiển thị cho thiết bị di động và người ta có thể kết nối với nó. Tuy nhiên, không có máy chủ DHCP nào được thiết lập, do đó điện thoại sẽ không nhận được bất kỳ địa chỉ IP nào được chỉ định và một lỗi sẽ được hiển thị trong điện thoại. Vấn đề này có thể được khắc phục bằng cách cài đặt máy chủ DHCP, ta chỉ cần cấu hình IP của bộ

điều hợp Wi-Fi từ laptop và tự cấu hình IP của điện thoại di động từ điện thoại di động. Bước đầu tiên, từ máy tính xách tay là:

```
#ifconfig wlan0 192.168.0.1
```

Sau đó, trong điện thoại di động, chọn "static" thay vì "dhcp" trong cài đặt nâng cao khi kết nối với Wi-Fi và chọn 192.168.10.2 làm ip của điện thoại và 192.168.10.1 làm IP cổng.

Bước cuối cùng là định cấu hình iptables trong máy tính xách tay để chuyển tiếp lưu lượng truy cập mà nó nhận được trong wlan0 để điện thoại có thể kết nối với drone thông qua bộ lặp, điều này có thể được thực hiện bằng các lệnh sau:

```
#echo 1 | tee /Proc/sys/net/ipv4/ip_forward  
#iptables -t nat -A POSTROUTING -s 192.168.10.0/16 -o wlan0 -j  
MASQUERADE
```

Bây giờ, ứng dụng HFun có thể được chạy từ điện thoại và nó có thể kết nối với drone, như thể điện thoại đã được kết nối thẳng với drone. Như vậy máy tính trở thành trạm trung chuyển truyền tín hiệu, từ đây ta có thể thay thế tín hiệu điều khiển để truyền sang thiết bị bay.

Nếu quy trình nói trên không hoạt động vì chuyển tiếp không hoạt động, toàn bộ thiết lập có thể được gỡ lỗi bằng cách cho máy tính xách tay kết nối với điểm phát wifi của Internet thông thường và kiểm tra xem mạng internet có hoạt động hay không, rồi sau đó đưa trở lại cách kết nối trong mô hình.

Cách quay lại thiết lập mạng ban đầu trong máy tính xách tay tương đối dễ thực hiện: dừng hostapd, chuyển đổi /etc/network/interfaces với nội dung ban đầu của nó và khởi động trình quản lý mạng như không có gì xảy ra.

Vấn đề cần được xem xét ở đây là độ trễ do việc kết nối gián tiếp này gây ra. Tuy nhiên, độ trễ dường như không quá tệ, việc điều khiển drone vẫn khá mượt và có thể chấp nhận được.

Như vậy, việc kết nối điện thoại, máy tính và drone coi như cơ bản hoàn thành. Việc này là hết sức quan trọng vì chỉ có như vậy ta mới dễ dàng phân tích,

nắm bắt lưu lượng cũng như cấu trúc các lệnh điều khiển được gửi từ điện thoại về drone từ đó đưa ra hướng tiếp cận của đề tài.

3.2.4 Phân tích gói dữ liệu gửi từ điện thoại về drone.

a) Điều khiển bay thông qua máy tính.

Đối với kết nối giữa drone và điện thoại di động, nhà sản xuất thường dùng giao thức UDP. UDP (User Datagram Protocol) là một trong những giao thức cốt lõi của giao thức TCP/IP. Dùng UDP, chương trình trên mạng máy tính có thể gửi những dữ liệu ngắn được gọi là datagram tới máy khác. UDP không cung cấp sự tin cậy và thứ tự truyền nhận mà TCP làm; các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. Tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của nó nên nó hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.

Việc sử dụng UDP là hợp lý để gửi các gói tin nhanh từ điện thoại về drone để điều khiển hành vi bay.

Từ việc đã thiết lập kết nối điện thoại di động – máy tính xách tay – thiết bị bay không người lái như ở trên, ta sẽ bắt đầu phân tích các gói dữ liệu được gửi đi bằng cách theo dõi lưu lượng trên Wireshark [49].

Tín hiệu điều khiển từ ứng dụng điều khiển được UDP gửi đến cổng 5353 cứ sau 100 ms nhưng thông báo dường như khác. Trong khi ở chế độ chờ, điện thoại sẽ gửi liên tục các gói giống nhau 20 byte từ cổng 9961 -> 19798 của drone qua giao thức UDP.

Làm theo các bước theo trình tự khởi động bình thường của thiết bị bay, đầu tiên bật chế độ giữ độ cao, sau đó bật bộ điều khiển trên màn hình và "điều khiển" cánh quạt bằng cách nhấn nút mũi tên lên trên màn hình. Điều này làm cho thiết bị bay sẵn sàng cất cánh. Tiếp tục theo dõi lưu lượng các tín hiệu truyền đi để hiểu rõ hơn về các quy luật của gói tin UDP điều khiển thiết bị bay.

Các gói tin qua UDP 20 byte đều có một quy luật chung như sau:

- Byte 0 và 1: 0x660A
- Byte 2 đến 6: Roll, pitch, throttle, yaw (ngiên trái phải, đi thẳng lùi, tăng giảm ga, xoay tại trục)
- Byte 7 và 8: 0x0002, bay bình thường, 0x0102
- Byte 9 đến 17: Mặc định là 0xFFFF000...
- Byte 19: là XOR Checksum của Byte2^Byte3^Byte4^...^Byte8 [4]
- Byte 20: 0x99

Lặp đi lặp lại nhiều lần, ta dễ dàng nhận thấy rằng mỗi mã lệnh gửi đi đều có mục đích nhất định. Tổng hợp các UDP này ta có bảng như sau:

Bảng 3. Gói UDP và công dụng.

UDP	Tác dụng
66:0a:80:80:80:80:00:00:ff:ff:00:00:00:00:00:00:00:99	Mới bật chương trình
66:0a:80:80:80:80:01:02:ff:ff:00:00:00:00:00:00:00:99	Nút cất cánh / hạ cánh
66:0a:80:80:80:80:02:02:ff:ff:00:00:00:00:00:00:00:99	Dừng NGYA LẬP TỨC
66:0a:80:80:80:80:04:00:ff:ff:00:00:00:00:00:00:00:99	Tinh chỉnh cân bằng
66:0a:80:80:80:80:00:02:ff:ff:00:00:00:00:00:00:00:99	Đã kết nối với điện thoại / Bình thường
66:0a:80:80:80:80:00:03:ff:ff:00:00:00:00:00:00:00:99	Đèn nhấp nháy chỉ dẫn
66:0a:80:80:fd:80:00:02:ff:ff:00:00:00:00:00:00:00:99	Throttle max
66:0a:80:80:00:80:00:02:ff:ff:00:00:00:00:00:00:00:99	Throttle min
66:0a:00:80:80:80:00:02:ff:ff:00:00:00:00:00:00:00:99	Pan (roll) trái
66:0a:fe:80:80:80:00:02:ff:ff:00:00:00:00:00:00:00:99	Pan (roll) phải

Việc phân tích được gói UDP gửi từ ứng dụng điều khiển bay trên điện thoại cũng như cơ chế gửi của nó giúp ta có thể bước đầu bắt tay vào việc điều khiển thiết bị bay thông qua máy tính và sau đó là phát triển để thiết bị bay có thể bay tự động.

Từ máy tính, “mạo danh” ứng dụng trên điện thoại bằng một vài dòng lệnh miêu tả đầy đủ các thông tin ta phân tích được như ở phía trên để gửi đến thiết bị bay và thử xem có thể điều khiển nó hay không:

Nếu các dòng lệnh trên hoạt động tốt thì điều này đồng nghĩa với việc ý tưởng điều khiển ban đầu là khả thi và đề tài đang phát triển theo đúng hướng. May mắn thay, mọi thứ diễn ra tốt đẹp, thiết bị bay được điều khiển thông qua máy tính mà không cần dùng ứng dụng trên điện thoại. Như vậy, vấn đề đầu tiên trong đề tài có thể coi như được giải quyết.

b) Truy xuất luồng video theo thời gian thực đồng thời với điều khiển bay.

Quay lại các kết nối UDP, tuy nhiên là theo chiều từ thiết bị bay về điện thoại di động. Dữ liệu của các khung hình thuộc video vẫn sẽ được gửi thông qua UDP.

Video được gửi bởi RTSP thông qua kết nối được bắt đầu bởi ứng dụng. Thiết bị bay gửi một chuỗi tin nhắn chứa video, cho khoảng 12 khung hình/giây.

Ta có thể chạy thử một script python để lấy video và điều khiển drone qua đó. Khi thiết bị bay kết nối máy tính và gửi luồng video thời gian thực thông qua luồng dữ liệu truyền trong luồng `rtsp://192.168.100.1/encavc0-stream`, sau một thời gian (khoảng 15 giây), đường truyền bị dừng lại. Sau đó chương trình bị treo, các gói tập tin UDP thông số điều khiển máy bay không thể gửi được nên thiết bị bay tự hạ cánh. Như vậy, nếu chỉ bay được khoảng 15 giây thì không có giá trị cho đề tài. Thách thức ở đây là làm như thế nào để kiểm soát song song việc đọc luồng video và điều khiển bay mà không bị dừng chương trình.

Tệp thô có phần mở rộng `.h264`, có thể được chơi bởi hầu hết các phần mềm có sẵn.

Kiểm tra wireshark khi mở cùng chương trình điều khiển ở Python, có thể thấy kết nối RTSP mở ở cổng 554 dưới dạng TCP. Với chương trình HFun, nó chạy bình thường. Nhưng khi chạy trên OpenCV-Python, kết nối giống vậy bị treo như trên. Nhập địa chỉ vào VLC hay ffplay lại chạy bình thường như HFun, nhưng kết nối cổng khác (không phải 554) với giao thức UDP. Tuy nhiên, có vẻ như VLC sẽ không chơi được nó ở tốc độ khung hình phù hợp, độ trễ nằm ở khoảng từ 100 đến 300 ms với tốc độ khung hình khoảng 10 khung hình mỗi giây.

Ta có thể lặp lại điều này trên OpenCV-Python với dòng lệnh:

```
os.environ["OPENCV_FFMPEG_CAPTURE_OPTIONS"] =  
"rtsp_transport;udp"  
  
cap = cv2.VideoCapture(src,cv2.CAP_FFMPEG)#
```

Ta chỉnh chế độ truyền RTSP sang giao thức UDP. Tuy nhiên hiện tại, điều này chỉ được thực hiện trên Linux (Lubuntu 18.04). Chưa kể khi chuyển sang chế độ mới, hình truyền sang hay bị ô vuông với độ phân giải kém. Lúc đầu giải quyết bằng chạy hai lần lệnh lấy frame của cv2:

```
(ret, _) = cap.read()
(ret, frame) = cap.read()
sleep(0.005)
```

Vấn đề được giải quyết nhanh chóng, hai chương trình điều khiển bay và đọc luồng video có thể chạy song song và không bị gián đoạn làm thiết bị ngừng bay.

Như vậy, vấn đề cuối cùng có thể được triển khai – nhận dạng người – trước khi thực hiện bước cuối cùng là thực hiện chương trình điều khiển bay tự động.

c) Nhận dạng người.

Dùng các mô hình có sẵn để nhận dạng người và nhận dạng khuôn mặt là việc khá thuận lợi trong các thuật toán nhận dạng, tuy nhiên, đối với nhận dạng đối tượng ở thời gian thực còn khá khó khăn ở đề tài này.

Ở đây, việc nhận dạng sẽ được chia làm hai phần riêng biệt: Nhận dạng khuôn mặt và nhận dạng thân hình người.

Tại sao lại phải qua nhiều bước như vậy?

Trước hết, việc nhận dạng khuôn mặt là cần thiết nhất, tuy nhiên, trong đề tài này, thiết bị bay cần phải bay theo người được giám sát, vậy nếu người đó quay ngược lại và chạy đi thì thiết bị bay rất khó khăn để luôn bám theo khuôn mặt.

Ý tưởng được đưa ra ở đây là: Trước hết, ta sẽ nhận dạng khuôn mặt, đúng với đối tượng cần giám sát. Sau đó nhận dạng thân hình người đó, và gán khuôn mặt đó cho thân hình mà khuôn mặt đó nằm trong, như vậy, ta sẽ bỏ qua việc nhận dạng khuôn mặt ở các khung hình sau, chỉ cần bám theo thân hình người này dù họ có xoay như thế nào đi nữa thì ta vẫn dễ dàng điều khiển thiết bị của ta bay theo họ.

Dựa theo ý tưởng này, tốc độ xử lý cần ưu tiên hàng đầu, vì ta ban đầu ta cần chạy hai chương trình riêng biệt và đây là xử lý video ở thời gian thực nên không có bất cứ điều gì tốt hơn việc tận dụng tốt thời gian.

Có 3 hướng có thể thử nghiệm:

- Yolov3 (mặt) + Yolov3 (người)
- Yolov2 (mặt) + Yolov3 (người)
- MTCNN (mặt) + Yolov3 (người)

Nhận dạng người là có sẵn ở các phiên bản YOLO, tuy YOLOv3 khá nặng về mặt cấu hình máy tính nhưng nhận dạng người là tối ưu nhất để xử lý ở thời gian thực. Nhận dạng người cần sự chính xác cao để tránh việc mất frame, mất đối tượng khi nhận dạng dẫn đến phải lặp lại quá trình nhận dạng mặt từ đầu nên ta sẽ chọn YOLOv3.

Khác với nhận dạng người, nhận dạng khuôn mặt gặp khó khăn ở việc tập dữ liệu để huấn luyện và nó là phần cốt lõi của giám sát nên cần độ chính xác khá cao. Để không trở ngại cho quá trình xử lý cũng như điều khiển bay, ta chọn mô hình MTCNN để dùng nhận dạng khuôn mặt. MTCNN + Facenet có điểm yếu là mặt phải khá lớn nên drone phải bay gần.

Cho Yolov3 chạy để nhận dạng người với GTX 1060, ta nhận được tốc độ khoảng 10 – 12 fps. Yolov3 có thêm thuật toán Deep sort để đánh số ID. Tuy nhiên nếu mình ra khỏi khung hình, hay mình lắc máy bay mạnh quá sẽ mất số ID. Sử dụng một chương trình biến điện thoại Android thành IP Webcam qua cổng http, một vấn đề nữa xuất hiện. Tốc độ camera là 30 fps nhưng tốc độ nhận dạng thấp hơn gấp 3 lần gây ra tình trạng trễ hoặc dồn frame. Đây là việc frame đang nhận dạng là frame sau khi IP Webcam đã thu đến frame khác mà chương trình nhận dạng không lấy, chỉ lấy frame theo thứ tự được thu. Thay các câu lệnh nhận dạng bằng lệnh sleep trong 0.1 giây sẽ lặp lại được tình trạng.

Một cách để giải quyết là sử dụng frameskip. Cách frameskip được thực hiện là nó sẽ đọc một số frame nhất định (Ví dụ 3 frame) và chỉ lấy 1 frame ra làm nhận dạng. Vì vậy có thể cắt video truyền từ 30 fps xuống 10 fps. Điều này giúp giữ cho chương trình vẫn đạt tốc độ thời gian thực.

Tuy nhiên, có một vấn đề sẽ vẫn tồn tại ở đây là frameskip cần có tốc độ chính xác. Nghĩa là chúng ta bỏ qua bao nhiêu frame phải được xác định từ trước. Trong khi đó, tốc độ xử lý nhận dạng cũng như tốc độ khung hình từ video thường không ổn định. Việc thiết lập sẵn frameskip là không khả thi, tình trạng dồn frame hoặc thiếu frame có thể vẫn sẽ bị mắc phải.

Một cách giải quyết có thể được xem xét đó là xử lý đa luồng. Ở đây, ta phân các lượng công việc thành các luồng khác nhau, xử lý song song. Giả sử, ta có 3 luồng với 3 nhiệm vụ như sau:

- Luồng thứ nhất chịu trách nhiệm điều khiển drone. Ở luồng này, nhận kết quả từ 2 luồng còn lại để gửi tín hiệu điều khiển drone theo mục tiêu đề ra.
- Luồng thứ hai, ta dùng để nhận dạng đối tượng thiết lập sẵn. Luồng này sử dụng thông tin từ luồng cuối cùng để nhận dạng. Lượng tài nguyên cần thiết cho luồng này là khá lớn, đòi hỏi độ chính xác và tốc độ xử lý trong mức chấp nhận được.
- Luồng thứ ba chỉ phụ trách nhận tín hiệu video, lưu các frame liên tục vào biến thiết lập sẵn. Mỗi khi luồng thứ 2 xử lý xong mỗi frame hình, frame tiếp theo luôn luôn được sẵn sàng để tiếp tục xử lý mà không cần biết tốc độ chênh lệch giữa tín hiệu video và tốc độ nhận dạng.

Ta có thể chọn giải pháp này cho vấn đề của chúng ta. Vừa giải quyết được việc trễ frame nhận dạng vừa tối ưu hóa được thời gian xử lý của máy tính, giúp cải thiện tốc độ nhận dạng.

Tuy đã giải quyết được hiện tượng dồn frame khiến treo chương trình nhưng trong quá trình thực hiện một vấn đề lớn được nữa được đặt ra: Tốc độ xử lý hình ảnh là quá chậm (1-4 fps) ảnh hưởng trực tiếp đến tốc độ xử lý cho điều khiển bay.

Có 2 nguyên nhân trực tiếp dẫn đến tình trạng này. Thứ nhất, mô hình nhận dạng thân hình người được sử dụng quá nặng, quá trình tính toán cho mỗi frame là quá lớn khiến cho tốc độ xử lý cho CPU của máy tính không thể giải quyết nhanh chóng được. Thứ hai, cấu hình máy tính luôn bị hạn chế bởi hiệu suất sử dụng năng lượng, hay cụ thể hơn, việc sử dụng nguồn năng lượng được dự trữ trong pin cho laptop sẽ khiến hiệu năng bị giới hạn không thể hoạt động toàn bộ công suất của máy tính.

Để giải quyết tình trạng này có 2 hướng có thể được áp dụng: Thứ nhất, thay đổi mô hình nhận dạng bằng các mô hình nhỏ gọn hơn với khối lượng tính toán ít hơn từ đó cải thiện được tốc độ xử lý. Thứ hai, đổi cấu hình máy tính, nâng cấp phần cứng để tăng nhanh tốc độ và thay đổi nguồn năng lượng sử dụng từ pin sang cắm trực tiếp vào nguồn điện để sử dụng.

Mô hình nhận dạng đối tượng mới được đề xuất ở đây là MobileNet [24].

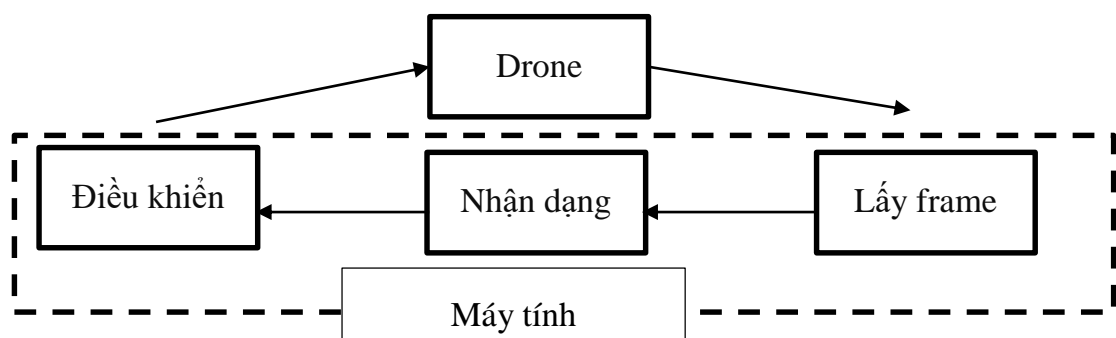
Mô hình nhận dạng đối tượng MobileNet được tối ưu hóa bằng cách giảm khối lượng tính toán thông số đến mức tối đa trong khi giữ độ chính xác đến mức có thể chấp nhận được. Mức giảm thông số tính toán là cực kỳ ấn tượng so với toàn bộ các mô hình đã được giới thiệu trước đó. Cơ sở của MobileNet đã được đề cập ở phần 2 cơ sở lý thuyết.

Thay thế mô hình YOLO các version bằng MobileNet, tốc độ nhận dạng được cải thiện cực kỳ khả quan. Song song với việc đó, máy tính sử dụng cho đề tài cũng được thay đổi tăng mạnh cấu hình máy với bộ xử lý CPU Intel Core I7-9750H 6 nhân 12 luồng, RAM 8GB, GPU NVIDIA Geforce GTX 1660Ti Mobile cài hệ điều hành Ubuntu 19.2.

Các thay đổi này đưa tốc độ xử lý lên mức 25 fps đối với nhận diện mặt và 15 fps đối với nhận dạng người. Vấn đề cơ bản được giải quyết.

Như vậy, các vấn đề hiện tại được giải quyết khá hoàn thiện để bắt đầu kết xuất đưa ra chương trình cuối cùng cho đề tài.

Sơ đồ xử lý như sau:



Hình 3.21 Sơ đồ xử lý của mô hình 2

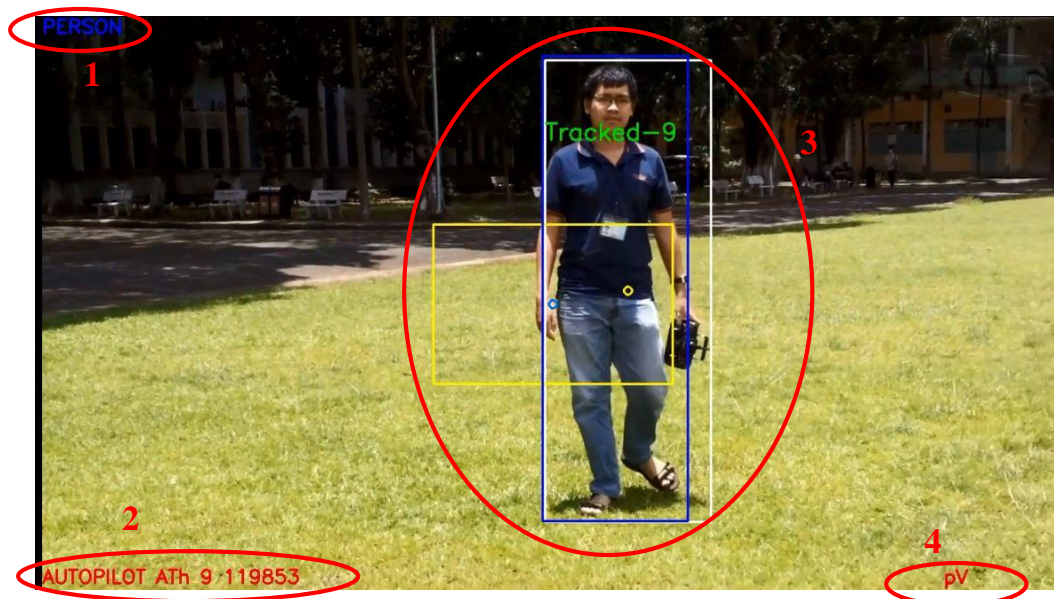
CHƯƠNG 4: KẾT QUẢ THỰC HIỆN VÀ HƯỚNG PHÁT TRIỂN

- 4.1 Kết quả thực hiện
- 4.2 Nhận xét ưu khuyết điểm
- 4.3 Hướng phát triển của đề tài

4.1 Kết quả thực hiện

4.1.1 Giao diện giám sát và các phím chức năng

Để triển khai thực hiện cả hai mô hình tốn khá nhiều thời gian, vì thế về vấn đề giao diện người dùng vẫn chưa được chú ý trao chuốt nhiều, chỉ dừng lại ở mức có thể dễ dàng nhìn thấy và phân biệt các chức năng cũng như hành vi điều khiển bay của máy tính nhằm mục đích kiểm tra và cải thiện hoạt động của chương trình.



Hình 4.1 Giao diện người dùng

Ở giao diện người dùng này có 4 vùng cần được lưu ý được đánh số như hình 4.1 ở trên.

Vùng 1, vùng góc trên bên trái màn hình, ở đây sẽ ghi chú về thao tác nhận dạng. Có 2 chế độ có thể được ghi chú ở vùng này là FACE hoặc PERSON tương

ứng với việc nhận dạng người hay nhận diện khuôn mặt mà chương trình đang xử lý. Phím chức năng để đổi chế độ nhận dạng là phím R.

Vùng 2 là vùng góc dưới bên trái màn hình giám sát. Ở đây có 4 mục được hiển thị phân tách nhau bằng dấu cách. Mục đầu tiên là chế độ bay gồm 2 lựa chọn AUTOPILOT hoặc MANUAL (tương ứng với bay tự động hoặc bay bằng tay cầm điều khiển), phím chức năng chuyển chế độ bay là phím O. Mục thứ 2 là có điều khiển độ cao hay không, nếu có sẽ hiện Ath, nếu không không hiện. Ath sẽ luôn hiện ở thời điểm kết thúc đề tài vì điều chỉnh độ cao là phần thêm vào để điều chỉnh chất lượng bay của việc giám sát. Mục thứ 3 là số chỉ ID của đối tượng đang được theo dõi do chương trình nhận dạng tự tạo. Mục thứ 4 là diện tích hộp giới hạn của khuôn mặt hoặc người đang hiện hành theo dõi.

Vùng 3 là vùng trung tâm của màn hình. Ở đây có 3 thứ cần lưu ý. Hộp chữ nhật màu vàng là vùng an toàn (safety area), hộp này sẽ chỉ hiện ra ở chế độ AUTOPILOT, chế độ MANUAL không hiện. Thiết bị bay sẽ cố gắng điều khiển bay sao cho tâm của hộp giới hạn mặt hoặc người nằm trong vùng này. Hộp chữ nhật màu xanh mực là hộp giới hạn bao quanh người. Bất cứ người nào trong màn hình giám sát được chương trình nhận dạng phát hiện ra đều có hộp giới hạn riêng và được đánh số tự động, việc theo chân đối tượng sẽ được tiến hành dựa trên số chỉ này vì thế khi thiết bị đã khóa chỉ định ID này trong hộp giới hạn này sẽ hiện lên dòng chữ “Tracked-ID” với ID là số chỉ của đối tượng. Hộp chữ nhật màu trắng là hộp bao quanh đối tượng đang được theo chân. Mọi đối tượng là “người” đều có hộp giới hạn của riêng mình và chỉ riêng “người” có hộp chữ nhật trắng bao quanh được theo chân.

Vùng 4 là vùng góc phải bên dưới màn hình. Ở đây sẽ hiển thị hành vi bay của thiết bị ở chế độ AUTOPILOT. Ở đây xuất hiện t, y, p, tương ứng với các kênh của drone và mũi tên tương ứng với hành động đang được thực hiện.

4.1.2 Mô hình với drone tự lắp ráp

Trước hết, trong mảng nhận dạng đối tượng, các mô hình nhận dạng khuôn mặt cũng như nhận người được áp dụng vào đề tài đều cho kết quả khá khả quan.

Tốc độ xử lý cũng như độ chính xác khá cao tuy điều này vẫn còn phụ thuộc nhiều vào điều kiện môi trường (ánh sáng, nắng, khoảng cách,...) đó là vấn đề chung khi xử lý hình ảnh bằng camera động. Trong đó, khoảng cách đối với người được theo dõi thật sự là một vấn đề nan giải. Nếu quá xa, camera đưa về hình ảnh quá mờ, tỉ lệ khuôn mặt quá nhỏ, mạng Neuron không thể nhận được khuôn mặt. Nếu quá gần, điều đó thật sự không an toàn. Chưa đề cập đến điều có thể ảnh hưởng đến người được giám sát hay không, chỉ riêng về sự ổn định của hệ thống bay và sự mất kiểm soát (nếu có thể xảy ra) đều rất nguy hiểm cho người lần thiết bị bay (dù đã có rất nhiều phương án dự phòng cho các trường hợp này). Việc chuyển đổi từ nhận dạng khuôn mặt sang nhận dạng thân hình người vẫn có một vấn đề nhỏ ở sự chuyển tiếp, nó liên quan đến sự triển khai mạng Neuron, việc khởi động mạng tạo gánh nặng cho máy tính khiến cho quá trình khựng lại trong một khoảnh khắc. Tuy nhiên nó không ảnh hưởng lớn đến toàn bộ quá trình.

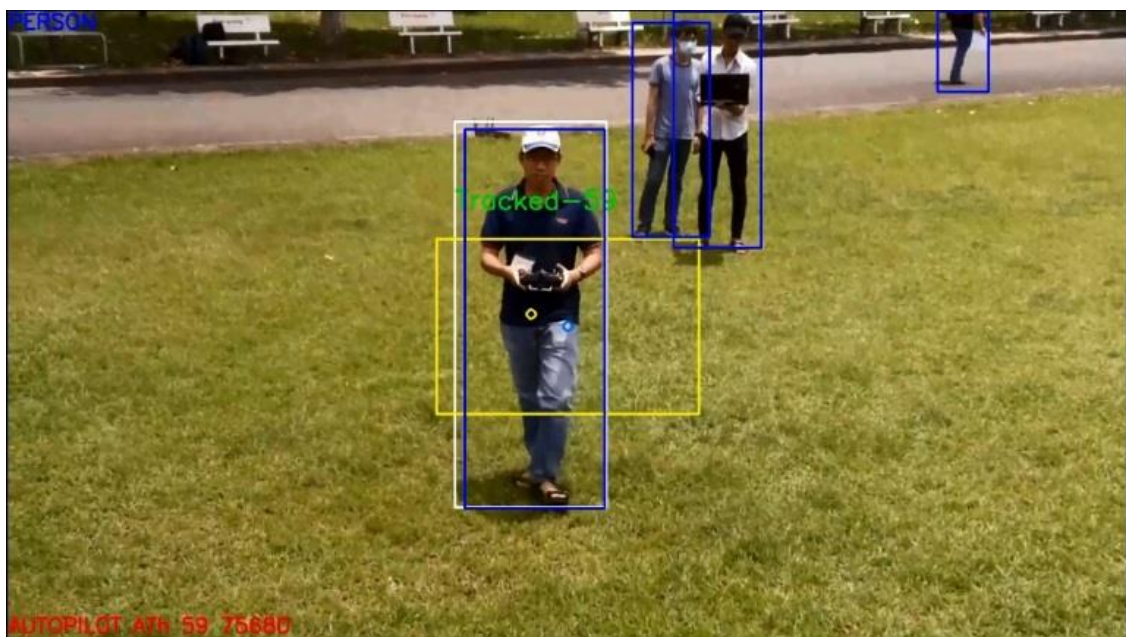


Hình 4.2 Kết quả nhận dạng khuôn mặt (Mô hình 1)

Trong vấn đề điều khiển bay theo vết đối tượng, tuy chỉ dùng 2/4 kênh điều khiển kết quả vẫn rất khả quan. Tầm nhìn từ camera luôn bám sát đối tượng cần theo dõi chỉ với 4 thao tác: tiến, lùi, xoay trái, xoay phải. Vấn đề tồn tại ở đây chính là việc độ nhạy của bộ xử lý không thật sự tốt và tiến trình xử lý vẫn bị delay so với quá trình mong muốn vì thế dẫn đến tình trạng trôi theo quán tính. Ví dụ như, khi cần tiến tới trước (do diện tích hộp giới hạn người nhỏ hơn ngưỡng đã

thiết lập) lệnh bay tiến tới trước được truyền đi và kéo dài một khoảng thời gian, tuy vậy khi tình huống xảy ra, hình ảnh gửi về máy đến lúc lệnh được gửi đi, dịch thành xung điều khiển động cơ, quá trình này chậm hơn so với tính toán, dẫn đến việc thiết bị bay tiến tới trước chậm hơn và kéo hơn cần thiết; quán tính bay của thiết bị bay cũng không được board điều khiển chính xử lý tốt vì vậy các lệnh bay thường kéo dài hơn so với dự tính (thay vì bay tới trước 20cm thì lại trôi thêm 3-5cm) điều này khiến cho vượt quá ngưỡng thiết lập dẫn đến thiết bị bay lại phải lùi lại; lùi lại quá mức lại tiếp tục tiến tới; tạo thành một vòng tuần hoàn tiến lùi tiến lùi khó có thể kiểm soát.

Để khắc phục lỗi này, chúng em đã tiến hành thêm một kênh điều khiển nữa (điều chỉnh độ cao, phụ thuộc vào tọa độ hộp giới hạn người và tọa độ khoảng an toàn), kết quả nằm ngoài mong đợi, khắc phục hoàn toàn ảnh hưởng của lỗi này. Thông qua đó, có thể thấy được khi tăng số kênh điều khiển có thể điều khiển thiết bị bay một cách chính xác, tuy nhiên, điều này đồng nghĩa với việc sẽ khó để cân chỉnh hơn lúc ban đầu. Đồng thời, do mỗi câu lệnh thực hiện tức thời và có sự sai số ở khâu truyền tín hiệu, vì vậy khi điều khiển một kênh sẽ là ổn định nhất, càng tăng kênh điều khiển, mỗi sự lệch lạc sai số của tín hiệu cũng sẽ tăng nhiều, không đơn giản là $1 + 1 = 2$ mà còn hơn thế nữa.



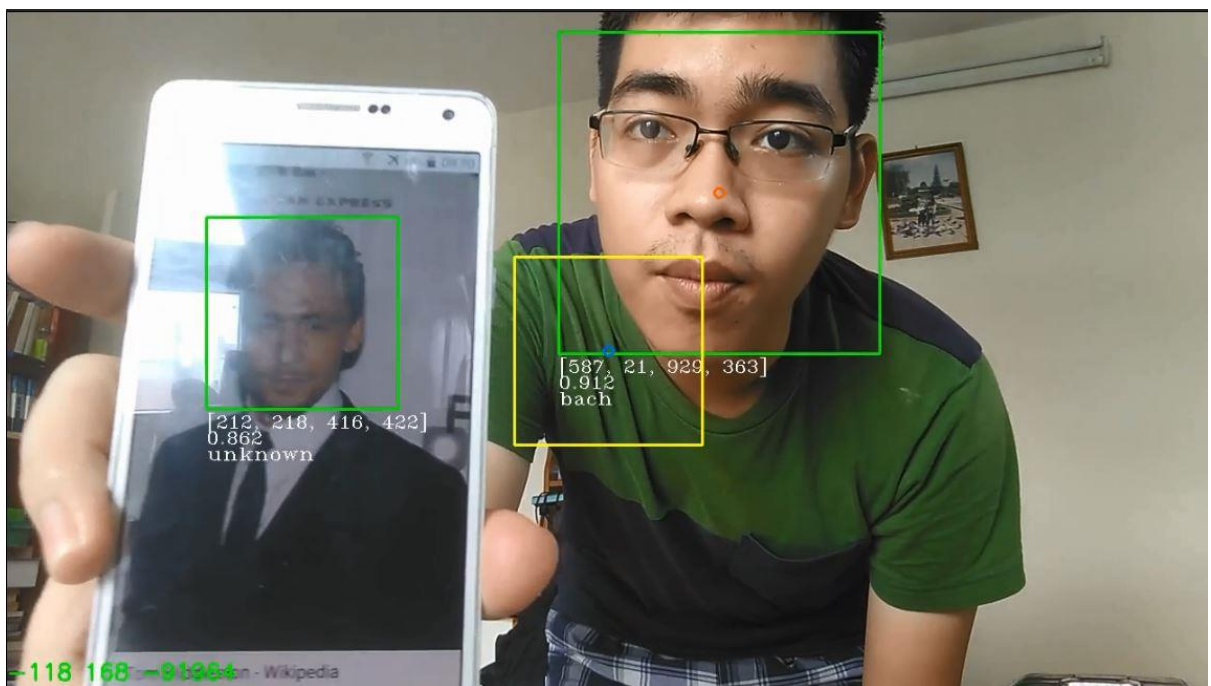
Hình 4.3 Kết quả bay theo sát đối tượng (Mô hình 1)

Thứ tự ưu tiên cho điều khiển cũng ảnh hưởng đến kết quả việc thực hiện. Cụ thể như, khi hình ảnh đối tượng vừa lệch lên trên vừa lệch bên trái, vậy thứ tự ưu tiên xử lý sẽ là bay lên cao hay là xoay trái? Điều này cũng là một nguyên nhân gây ra hiện tượng tiến lùi liên tục khi điều khiển 2 kênh. Hiện tại, trong đề tài sử dụng thứ tự ưu tiên là xoay, độ cao, cuối cùng là tiến lùi.

4.1.3 Mô hình với drone hoàn thiện bởi nhà sản xuất

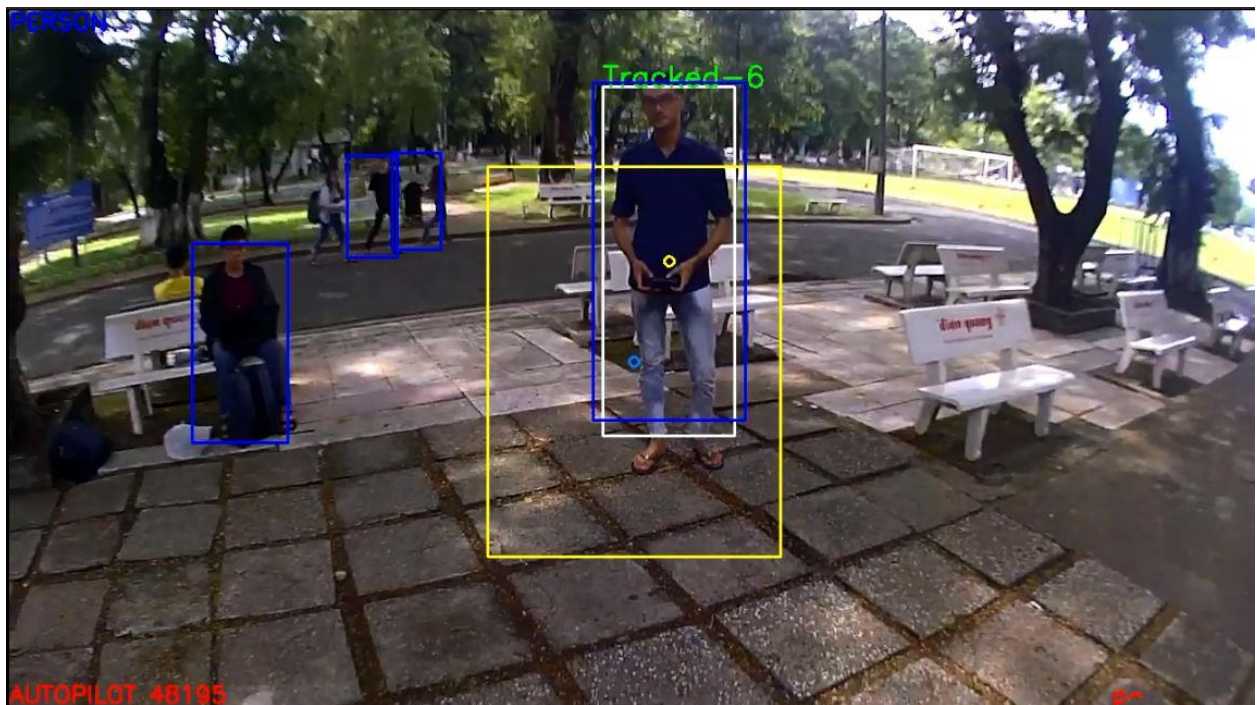
Ở mô hình này, kết quả đạt được khá hoàn thiện.

Trong mảng nhận dạng đối tượng, việc xử lý khá ổn, tốc độ nhận dạng cũng như độ chính xác tùy điều kiện môi trường. Tuy nhiên, do quá phụ thuộc vào môi trường (ánh sáng, khoảng cách,...) nên đây được coi là mặt hạn chế của mọi mô hình nhận dạng đối với camera động.



Hình 4.4 Kết quả nhận dạng khuôn mặt (Mô hình 2)

Trong mảng theo chân đối tượng, việc điều khiển bay đạt được thành công như mong đợi. Trước tiên, đối với tracking theo khuôn mặt, hộp giới hạn luôn theo sát khuôn mặt đồng thời xác định được đối tượng phải hay không phải là đối tượng được huấn luyện để nhận diện. Với việc điều khiển bay, thành công lập trình để hướng dẫn bay theo sát khuôn mặt đã được cài đặt sẵn.



Hình 4.5 Tracking người dựa trên diện tích hộp giới hạn (Mô hình 2)

Đối với tracking theo thân hình người, việc chuyển đổi từ nhận diện khuôn mặt sang nhận dạng thân hình người bước đầu hoàn thiện. Thiết bị có thể bay theo đối tượng đã được nhận dạng. Một vấn đề cần được cân đối ở đây chính là giữa độ chính xác khi theo sát thân hình (độ tin cậy của thuật toán nhận dạng đối tượng) và tốc độ xử lý (tốc độ điều khiển). Nếu tăng độ chính xác khi theo vết đối tượng (tránh cho thiết bị bay theo chân sai đối tượng trong tình trạng nhập nhằng giữa nhiều đối tượng chồng chéo, che lấp nhau), thì tốc độ xử lý (cả nhận dạng đối tượng và điều khiển bay) sẽ chậm lại đáng kể khiến cho chất lượng đề tài bị giảm đi.

4.1.4 So sánh các mô hình nhận dạng

Sử dụng lần lượt các mô hình nhận dạng để kiểm tra xem mô hình nào là tốt nhất cho đề tài. Các mô hình nhận dạng khi được kiểm tra thường chỉ test trên ảnh, không test trên video, cũng như là chỉ test với camera tĩnh chứ không test với camera được gắn trên thiết bị bay như đề tài này, vì tính đặc thù của vấn đề nên việc kiểm tra lại bằng thực tiễn để chọn một mô hình nhận dạng phù hợp nhất với nhận dạng dùng camera động là hoàn toàn cần thiết.

Có thể dễ dàng kiểm tra về tốc độ xử lý của các mô hình. Điều này có thể phụ thuộc nhiều vào cấu hình máy tính đã triển khai. Kết quả được ghi nhận như bảng sau:

Bảng 5. Tốc độ xử lý các mô hình nhận dạng thực tế	
Mô Hình	Nhận dạng người (FPS)
YOLOv3	8.632537369
YOLOv3-tiny	11.14020349
Mô Hình	Nhận diện khuôn mặt (FPS)
YOLOv2	25.14094281
Resnet	45.0316502

Qua bảng số liệu này ta có thể nhận ra rằng YOLOv3 xử lý chậm hơn YOLOv3-tiny ở mảng nhận dạng người, YOLOv2 nhận mặt chậm hơn gấp gần 2 lần so với Resnet. Không khó để giải thích cho điều này. YOLOv3-tiny là bản rút gọn của YOLOv3 vì thế nó chạy nhanh hơn. Resnet là phiên bản cải tiến xử dụng các nút kết nối tắt, cải thiện nhanh về tốc độ xử lý. Cả YOLOv3-tiny và Resnet đều sẽ đánh đổi độ chính xác để đẩy nhanh tốc độ xử lý. Ở đề tài này, tốc độ xử lý là cực kỳ quan trọng vì lượng xử lý cho máy tính là rất lớn (vừa phải truyền tín hiệu, chạy chương trình nhận dạng, vừa phải xử lý điều khiển bay và kiểm soát các kết nối), tốc độ xử lý càng cao thiết bị bay sẽ được điều khiển càng mượt. Độ chính xác có thể bù đắp bằng số lượng khung hình đã xử lý và có biện pháp khắc phục trong trường hợp xảy ra lỗi khi nhận dạng sai vì thế hoàn toàn có thể sử dụng mô hình có tốc độ cao là YOLOv3-tiny và Resnet.

4.1.5 So sánh 2 mô hình đã thực hiện

Việc lên ý tưởng cho hai mô hình với hướng tiếp cận hoàn toàn khác nhau khiến cho lượng công việc trở nên lớn hơn rất nhiều so với trong tưởng tượng. Kết quả đạt được có thể không thật sự hoàn hảo nhưng về cơ bản hoàn thành được mục tiêu đặt ra cho đề tài.

Một vài điểm có thể so sánh đối với 2 mô hình đã được thực hiện như khả năng thiết lập kết nối và chất lượng đường truyền cũng như tính ổn định của mô hình.

Bảng 6. So sánh 2 mô hình đã thực hiện

	Mô hình 1	Mô hình 2
Khả năng cân bằng vị trí và độ cao	Tốt. Trừ lúc khi pin sắp hết, drone sẽ tự bay thấp dần.	Kém. Drone không chịu gió tốt, dễ lung lay.
Lỗi trong cất cánh	Thỉnh thoảng nhiều trong kết nối giữa RX701 và Arduino Uno có thể khiến drone bay vòng tại chỗ hay tự tiến.	Nhiều lần cất cánh không cân bằng, đặc biệt qua máy tính, mặc dù lệnh được gửi lên không nói drone di chuyển. Lúc đó phải cân bằng lại IMU.
Khả năng kết nối wifi	Raspberry Pi sẽ nối vào router của trạm mặt đất mỗi khi khởi động. Cần truy cập vào router để tìm địa chỉ IP của Pi sau 1 phút khởi động.	Mỗi lần cần nối vào, drone sẽ đưa một tên wifi khác tên trước đó (M8_1080P[...]).
Chất lượng đường truyền	Tốt. Thỉnh thoảng vẫn bị mất frame hình khi bay, nhất là khi bay xa khỏi vùng phủ sóng của router.	Khá tệ. Mất frame hay xảy ra. Có lúc khiến chương trình thoát bất chợt, làm cho drone tự hạ cánh.
Hỗ trợ điều khiển bên ngoài khi chạy chương trình	Nút Mux trên điều khiển Devo-7 chuyển qua lại nhanh chóng giữa bay bằng điều khiển hay bằng Pi.	Drone luôn ưu tiên cho điều khiển từ xa. Nếu trong quá trình chạy mà bật điều khiển, lập tức drone sẽ theo điều khiển đến khi hạ cánh và tắt điều khiển.
Thư viện sử dụng trong OpenCV	FFMPEG: có độ trễ GStreamer: gần như realtime.	Sử dụng FFMPEG. GStreamer xảy ra lỗi.
Tinh chỉnh độ nhạy điều khiển drone trong chương trình	Ít cần chỉnh. Các chiều di chuyển luôn cân bằng lẫn nhau.	Drone có đặc điểm tiến và lùi không cân nhau, vì vậy thông số mức độ di chuyển phải luôn thay đổi.
Phần cứng	Dễ dàng triển khai thực hiện với mọi mô hình drone, dễ dàng thay thế linh kiện	Kén chọn với thiết bị, phải phát triển ứng dụng điều khiển drone trên điện thoại
Can thiệp kết nối truyền tín hiệu điều khiển	Mô hình này chỉ can thiệp duy nhất là kết nối từ bộ thu RX và board điều khiển chính vì thế có thể xem như mô phỏng hoàn toàn mọi hành vi của drone cũng như có thể dùng mọi tín hiệu truyền từ tay cầm điều khiển.	- Ở mô hình này can thiệp nhiều về việc gửi các gói lệnh điều khiển cho thiết bị bay, việc đọc lệnh để hiểu đó là lệnh nhằm mục đích gì rất khó khăn không thể tránh khỏi việc đọc hiểu sai lệnh dẫn đến những sự nhầm lẫn trong điều khiển bay cũng như thiếu

		các lệnh không thể mô phỏng đầy đủ bộ lệnh sẵn có của thiết bị. Đây có thể là một vấn đề lớn đối với chất lượng của mô hình.
--	--	--

4.2 Nhận xét ưu khuyết điểm

Ưu điểm:

- Mỗi mô hình đi theo một hướng điều khiển riêng nhưng đều đạt được kết quả cơ bản theo yêu cầu của đề tài: điều khiển thiết bị bay bay theo đối cần theo dõi.
- Do điều kiện đề tài không cho phép, đề tài chỉ mới triển khai huấn luyện cho mạng neuron để nhận dạng 1 người, tuy nhiên hoàn toàn có thể huấn luyện cho tập dữ liệu để nhận dạng nhiều người và chọn một trong các đối tượng để tiến hành giám sát.
- Sau khi nhận dạng được đối tượng và tiến hành tracking, thiết bị bay hoàn toàn chỉ bay theo đối tượng đó mặc dù có nhiều đối tượng là “người” xuất hiện trong hình.
- Phương hướng triển khai của cả hai mô hình đều rất đơn giản, dễ hiểu, đều có thể triển khai cho bất cứ ai có một chút kiến thức về drone, lập trình và có đam mê về lĩnh vực này.
- Drone đang rất thịnh hành trong cuộc sống hiện tại, đề tài có thể ứng dụng rộng rãi trong nhiều lĩnh vực.

Khuyết điểm:

- Yêu cầu về cấu hình máy tính xách tay là khá cao, sự chậm trễ trong truyền tín hiệu cũng như tốc độ xử lý kém sẽ ảnh hưởng trực tiếp đến kết quả của mô hình.
- Giới hạn hiệu suất xử dụng của máy tính (giữa việc dùng nguồn điện trực tiếp và dùng pin) cũng là vấn đề quan trọng khi triển khai thực hiện đề tài. Tốt nhất nên sử dụng nguồn điện trực tiếp từ mạng điện sẽ không bị vấn đề này.

- Khoảng cách nhận diện khuôn mặt khá hạn chế, các mô hình nhận diện khuôn mặt đều phụ thuộc nhiều vào kích thước khuôn mặt, góc mặt cũng như độ sáng của mặt, đồng thời, cải thiện độ chính xác khi nhận diện khuôn mặt sẽ tăng nhanh quá trình xử lý cho việc điều khiển.

- Nguồn năng lượng cho thiết bị bay bị giới hạn (mỗi pin chỉ có khoảng 5200mAh ở mô hình 1 và 1000mAh ở mô hình 2 tương đương 10-15 phút bay).

- Khoảng cách của tín hiệu wifi khá nhỏ, bị giới hạn trong một khuôn viên nhất định; thiết bị bay khá lớn và công kênh nên đề tài chỉ giới hạn ở ngoài trời, không gian thoáng, ít vật cản, khó triển khai trong phòng.

4.3 Hướng phát triển của đề tài

Đây là đề tài khá mới, vì thế có rất nhiều hướng có thể phát triển trong tương lai.

- Vì tín hiệu wifi từ router bị giới hạn trong khoảng không gian cụ thể, nên nếu muốn theo dõi đối tượng ở xa hoặc rất xa là không thể. Vì thế, nếu có thể chuyển mạng wifi trở thành mạng dữ liệu di động, truyền trực tiếp lên web sever thì dù ở bất cứ đâu ta vẫn có thể điều khiển theo dõi đối tượng như mong muốn.
- Nguồn năng lượng là khá hạn chế, nên nếu có thể tăng dung lượng pin nhưng cần cố gắng đảm bảo tải trọng của thiết bị bay.
- Kích thước của thiết bị bay này khá lớn (để đổi lại sự ổn định khi gió lớn hoặc thời tiết bất lợi), nếu có thể thu nhỏ kích thước mà vẫn đảm bảo kết cấu mạnh mẽ có thể ổn định khi gặp thời tiết bất lợi thì sẽ thuận tiện rất nhiều đối với mục tiêu theo dõi đối tượng như mong muốn.
- Công cụ thể thực hiện đề tài có vẻ khá phức tạp và dư thừa nhiều, vì thế khi tích hợp cả board điều khiển chính MC, arduino, Raspberry Pi,... vào chung một thiết bị duy nhất thì ta có thể giảm thiểu rất nhiều thể tích của thiết bị bay, từ đó tăng thời gian bay của thiết bị cũng như sự ổn định khi bay.

TÀI LIỆU THAM KHẢO

- [1] Khương Nha, Duy Tín, “Cách mạng Công nghiệp 4.0 là gì?”, <https://news.zing.vn/cach-mang-cong-nghiep-40-la-gi-post750267.html>, 2017.
- [2] Schwab Klaus, “Shaping the Fourth Industrial Revolution”, Portfolio Penguin, 2018, ISBN: 978-0-2413-6637-0.
- [3] Rituparna Chatterjee, “How image processing will change your world in future”, https://economictimes.indiatimes.com/tech/software/how-image-processing-will-change-your-world-infuture/articleshow/10394958.cms?utm_source=contentofinterest&utm_medium=text&utm_campaign=cppst, 2011.
- [4] Jain, B & Thakur, Shwetha & Suresh, K., “Visual Assistance for Blind Using Image Processing”, Hội nghị quốc tế về truyền thông và xử lý tín hiệu (ICCSP), 3-5/4/2018, Electronic ISBN: 978-1-5386-3521-6.
- [5] Vaibhaw Singh Chandel, “Selective Search for Object Detection (C++/Python)”, <https://www.learnopencv.com/selective-search-for-object-detection-cpp-python>, 2017.
- [6] Long Đỗ, “INTERSECTION OVER UNION (IOU) CHO OBJECT DETECTION”, <https://ai.hblab.vn/2017/10/intersection-over-union-iou-cho-object.html>, 2017.
- [7] Đỗ Hoàn, “Convolutional Neural Network | Cái nhìn tổng quan”, https://medium.com/@hon_14885/convolutional-neural-network-c%C3%A1inh%C3%ACn-t%E1%BB%95ng-quan-c6aaba265a39, 2019.
- [8] Chomba Bupe, “How does the region proposal network (RPN) in Faster R-CNN work?”, <https://www.quora.com/How-does-the-region-proposal-network-RPN-in-Faster-R-CNN-work>, 2017.
- [9] Joseph Redmon, Ali Farhadi, Better, *YOLO9000: Better, Faster, Stronger*, University of Washington, 2016, pp. 2-4, Electronic ISBN: 978-1-5386-0457-1, Print ISSN: 1063-6919.

- [10] Quoc Pham, “Tìm Hiểu Mô Hình YOLO Cho Bài Toán Object Detection - Understanding YOLO”, <https://pbcquoc.github.io/yolo>, 2018.
- [11] Long Đỗ, “YOLO: You Only Look Once”, <https://ai.hblab.vn/2017/10/yolo-you-only-look-once.html>, 2017.
- [12] Jonathan Hui, “Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3”, https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088, 2018.
- [13] Arthur Ouaknine, “Review of Deep Learning Algorithms for Object Detection”, <https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852>, 2018
- [14] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, Network Design, “You Only Look Once: Unified, Real-Time Object Detection”, University of Washington, 2015, Electronic ISBN: 978-1-4673-8851-1, Electronic ISSN: 1063-6919.
- [15] Trương Xuân Đạt, “Giới Thiệu Về Mô Hình SVM”, <https://www.stdio.vn/articles/gioi-thieu-ve-mo-hinh-svm-436>, 2015.
- [16] Tùng Trịnh, “Deep learning và bài toán Face Recognition”, <https://forum.machinelearningcoban.com/t/deep-learning-va-bai-toan-face-recognition/1776>, 2018.
- [17] Nguyễn Tuấn Anh, “Detect object sử dụng mô hình SSD”, <https://viblo.asia/p/detect-object-su-dung-mo-hinh-ssd-WAyK84rnKxX>, 2019.
- [18] Alexander Hermans, Lucas Beyer, Bastian Leibe, “In Defense of the Triplet Loss for Person Re-Identification”, 2017.
- [19] Florian Schroff, Dmitry Kalenichenko, James Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2015, 7-12/6/2015, Electronic ISBN: 978-1-4673-6964-0, Electronic ISSN: 1063-6919.

- [20] Jiankang Deng, Jia Guo, Niannan Xue, Stefanos Zafeiriou, “ArcFace: Additive Angular Margin Loss for Deep Face Recognition”, IEEE, 15-20/6/2019, Electronic ISBN: 978-1-7281-3293-8, Electronic ISSN: 2575-7075.
- [21] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, Yu Qiao, “Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks”, IEEE, 2016, Print ISSN: 1070-9908.
- [22] Xingyi Zhou, Vladlen Koltun, Philipp Krähenbühl, “Tracking Objects as Points”, 2020.
- [23] Nguyễn Hùng Thái Sơn, Võ Nguyên Phúc, “Thiết kế và thi công mô hình bay Quacopter”, Tạp chí Khoa học Lạc Hồng Số 4 (12/2015), trang 28-32, 2015.
- [24] Andrew G. Howard, Menglong Zhu và các cộng sự, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”, Google Inc., 2017.
- [25] mujahidin, “DEVO 7 cài đặt full + Upgrade deviation firmware để sử dụng rx cam”, <http://myrcsaigon.com/devo-7-setup-full/>, 2017.
- [26] Oyster Bay, “Small Unmanned Aerial Systems Market Exceeds US\$8.4 Billion by 2019, Dominated by the Commercial Sector and Driven by Commercial Applications”, <https://www.abiresearch.com/press/small-unmanned-arial-systems-market-exceeds-us84-b/>, 2015
- [27] Rituparna Chatterjee, “How image processing will change your world in future”, <https://economictimes.indiatimes.com/tech/software/how-image-processin-g-will-change-your-world-in-future/articleshow/10394958.cms>, 2011
- [28] Federal Aviation Administration, “Fact Sheet – Small Unmanned Aircraft Regulations (Part 107)”, https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=22615, 2018.

- [29] Craig Whitlock, "Part One: War Zones", When drones fall from the sky, <https://www.washingtonpost.com/sf/investigative/2014/06/20/when-drones-fall-from-the-sky/>, 2014.
- [30] dl_ap, "3. Chi tiết thuật toán", YOLO: Real-Time Object Detection (phần 1), <https://dlapplications.github.io/2018-08-03-2018-07-31-yolo1/>, 2018.
- [31] Jason Brownlee, "A Gentle Introduction to Object Recognition With Deep Learning", <https://machinelearningmastery.com/object-recognition-with-deep-learning/>, 2019.
- [32] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, "Object detection with R-CNN", Rich feature hierarchies for accurate object detection and semantic segmentation, UC Berkeley, 2014, pp. 2-3, Electronic ISSN: 1063-6919.
- [33] Arthur Ouaknine, "Review of Deep Learning Algorithms for Object Detection", <https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852>, 2018.
- [34] Mauricio Menegaz, "Understanding YOLO", <https://hackernoon.com/understanding-yolo-f5a74bbc7967>, 2018.
- [35] Ayoosh Kathuria, "What's new in YOLO v3?", <https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>, 2018
- [36] Python Lessons, "YOLO v3 theory explained", <https://medium.com/@pythonlessons0/yolo-v3-theory-explained-33100f6d193>, 2019
- [37] Joseph Redmon, Ali Farhadi, "The Deal", YOLOv3: An Incremental Improvement, University of Washington, 2018, pp. 1-3.
- [38] Pietra F T Madio, "A FaceNet-Style Approach to Facial Recognition on the Google Coral Development board", <https://towardsdatascience.com/a-facenet-style-approach-to-facial-recognition-dc0944efe8d1>, 2019
- [39] "Tìm hiểu kỹ thuật video streaming", <https://sites.google.com/site/embedded247/npcourse/tim-hieu-ky-thuat-video-streaming>

- [40] "Phương tiện bay không người lái", https://vi.wikipedia.org/wiki/Ph%C6%B0%C6%A1ng_ti%E1%BB%87n_bay_kh%C3%B4ng_ng%C6%B0%E1%BB%9Di_l%C3%A1i
- [41] btv10, "Cấu tạo Drone", Tổng quan Flycam, Drone là gì, mua ở đâu?, <https://giaonhanlogistics.com/tong-quan-drone-flycam/>, 2019
- [42] Lê Hoàng Anh, "Tầm quan trọng về hiệu quả của bộ cảm biến IMU đối với cuộc thi Robocon 2013", <https://dangbo.lhu.edu.vn/201/17372/>
- [43] Stephanie Stocker, "6 Degrees of Freedom Graphic", <https://www.ceva-dsp.com/ourblog/what-is-an-imu-sensor/hil-what-is-an-imu-wk-1/>, 2019
- [44] Phan Vu Hoang, "Đôi điều về IMU (Inertial Measurement Unit) - Cảm biến góc quay + cảm biến gia tốc", <http://arduino.vn/bai-viet/960-doi-dieu-ve-imu-inertial-measurement-unit>, 2016
- [45] Nguyễn Hùng Thái Sơn, Võ Nguyên Phúc, "Thiết kế và thi công mô hình bay quadcopter", Tạp chí Khoa học Lạc Hồng Số 4, trang 28-32, 2015
- [46] Thuong Le-Tien, Khang Nguyen-Dinh, Khang Nguyen-Dinh, "Encoding Blocks for Digital Video of DVB-S2 Standard with Drone-based Communications", 2019, Electronic ISBN: 978-1-7281-5353-7
- [47] "Bộ điều khiển 7 kênh devo 7+rx nâng cấp 701", <https://shopee.vn/B%E1%BB%98-%C4%90I%E1%BB%80U-KHI%E1%BB%82N-7-K%C3%8ANH-DEVO-7-RX-N%C3%82NG-C%E1%BA%A4P-701-CH%C3%8DNH-H%C3%83NG.-i.29996360.402477839>
- [48] DJI, "NAZA-M LITE User Manual", 2014.04.21 Revision, For Firmware Version V1.00 & Assistant Software Version V1.00
- [49] adria.junyent-ferre, "Controlling a JJRC H37 Elfie quad from a PC", <https://hackaday.io/project/19680/logs>, 2017

[50] Wikimedia Commons, “Pinout of ARDUINO Board and ATmega328PU”, https://commons.wikimedia.org/wiki/File:Pinout_of_ARDUINO_Board_and_ATMega328PU.svg, 2017

[51] Gia dụng nhà Việt, “Bo mạch Arduino Uno R3 kèm cáp USB”, <https://giadungnhaviet.com/san-pham/bo-mach-arduino-uno-r3/>

[52] Galaktyk 01, “How to compile OpenCV with Gstreamer [Ubuntu&Windows]”, <https://medium.com/@galaktyk01/how-to-build-opencv-with-gstreamer-b11668fa09c>, 2019

[53] neilyoung, “RTSP streaming from Raspberry PI”, <https://gist.github.com/neilyoung/8216c6cf0c7b69e25a152fde1c022a5d>, 2020

PHỤ LỤC CODE SỬ DỤNG TRONG BÀI

Code điều khiển bay (Python)

```
import socket

from time import time, sleep

from threading import Thread

# Define drone

class dm107s():

# Default control value

def __init__(self):

    # 4 values for flight

    self.roll = 128

    self.pitch = 128

    self.throttle = 128
```

```
self.yaw = 128

# 0 - normal mode, 2 - emergency stop, 4 - gyroscope calibration

self.commands = 0

# Required for wifi control

self.onoff = 1

# Prevent multiple takeoff button presses

self._takeoff_flag = False

# Prevent multiple calibrate button presses

self._calibrate_flag = False

# Connect to UDP port

self.sess = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, 0)

#self.sess.connect(('192.168.100.1', 19798))

# Initialize timer value

self._takeoff_timer = 0

self._calibrate_timer = 0

# Flag to stop thread

self._stopped = False


# Start separated thread for drone control

def start(self):

    self._thread = Thread(target=self.send_ctrl, args=(), daemon=True)

    self._thread.start()

    return self


# Get command hex for drone

def get_hex(self):
```

```
# XOR is for checksum

self.command_out=((26122<<144)|self.roll<<136|self.pitch<<128|self.throt
tle<<120|self.yaw<<112|self.commands<<104|self.onoff*2<<96|65535<<80|(se
lf.roll^self.pitch^self.throttle^self.yaw^self.commands^(self.onoff*2))<
<8|153)

self.command_out = hex(self.command_out)[2::]

return self.command_out

# Turn hex to byte package

def _get_packet(self):

    self._hex_code = self.get_hex()

    self.package = bytes.fromhex(self._hex_code)

    return self.package

# Send control to drone

def send_ctrl(self):

    while not self._stopped:

        self._package = self._get_packet()

        #self.ssess.send(self._package)

        self.ssess.sendto(self._package, ('192.168.100.1', 19798))

        self.Flag_off()

        sleep(0.02)

# Close connection to drone

def close_connection(self):

    self._stopped = True
```

```
        if self._thread.daemon == False:

            self._thread.join()

            self.ssess.close()

# Return to default

def default(self):

    self.roll = 128

    self.pitch = 128

    self.throttle = 128

    self.yaw = 128

    self.commands = 0

    self.onoff = 1

    self._takeoff_flag = False

# Increment control

def incremt(self, rl, pt, th, yw):

    self._value_to_change = [128, 128, 128, 128]

    self._change_val = [rl, pt, th, yw]

    for x in range(len(self._value_to_change)):

        self._value_to_change[x] += self._change_val[x]

        if self._value_to_change[x] <= 0:

            self._value_to_change[x] = 0

        if self._value_to_change[x] >= 255:

            self._value_to_change[x] = 255

    [self.roll, self.pitch, self.throttle, self.yaw] =
    self._value_to_change
```

```
# Roll right

def roll_right(self):

    self.roll += 20

    if self.roll > 248:

        self.roll = 248


# Pitch forward

def pitch_fwd(self):

    self.pitch += 20

    if self.pitch > 248:

        self.pitch = 248


# Increase throttle

def throttle_up(self):

    self.throttle += 20

    if self.throttle > 248:

        self.throttle = 248


# Yaw right

def yaw_right(self):

    self.yaw -= 20

    if self.yaw < 18:

        self.yaw = 18


# Roll left
```

```
def roll_left(self):  
  
    self.roll -= 20  
  
    if self.roll < 18:  
  
        self.roll = 18  
  
  
# Pitch backward  
  
def pitch_bwd(self):  
  
    self.pitch -= 20  
  
    if self.pitch < 18:  
  
        self.pitch = 18  
  
  
# Decrease throttle  
  
def throttle_dwn(self):  
  
    self.throttle -= 20  
  
    if self.throttle < 18:  
  
        self.throttle = 18  
  
  
# Yaw left  
  
def yaw_left(self):  
  
    self.yaw += 20  
  
    if self.yaw > 248:  
  
        self.yaw = 248  
  
  
# Takeoff  
  
def takeoff(self):  
  
    if self._takeoff_flag == False:
```

```
        self.commands = 1

        self._takeoff_flag = True

        self._takeoff_timer = time()

# Landing

def land(self):

    if self._takeoff_flag == False:

        self.commands = 1

        self._takeoff_flag = True

        self._takeoff_timer = time()

# Flip takeoff flag

def Flag_off(self):

    if (self._takeoff_flag == True and (time() - self._takeoff_timer >=
1)):

        self.commands = 0

        self._takeoff_flag = False

    if (self._calibrate_flag == True and (time() - self._calibrate_timer
>= 3)):

        self.commands = 0

        self.onoff = 1

        self._calibrate_flag = False

# Stop IMMEDIATELY

def emergency_stop(self):

    self.roll = 128

    self.pitch = 128
```



```
self.throttle = 128

self.yaw = 128

self.commands = 2

self.onoff = 1

self._takeoff_flag = False


# Calibrate gyroscope

def calib_gyro(self):

    if self._calibrate_flag == False:

        self.roll = 128

        self.pitch = 128

        self.throttle = 128

        self.yaw = 128

        self.commands = 4

        self.onoff = 0

        self._calibrate_flag = True

        self._calibrate_timer = time()


class naza():

    # Default control value

    def __init__(self, ip, port):

        # 4 values for flight

        self.roll = 8

        self.pitch = 8

        self.throttle = 8

        self.yaw = 8
```

```
# Prevent multiple takeoff button presses

self._takeoff_flag = False

# Prevent multiple ignite button presses

self._ignite_flag = False

self._ignite_send = False

# Connect to UDP port

self.sess = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, 0)

self.ip = ip

self.port = port

#self.sess.connect((ip, port))

# Initialize timer value

self._ignite_timer = 0

self._takeoff_timer = 0

# Flag to stop thread

self._stopped = False


# Start separated thread for drone control

def start(self):

    self._thread = Thread(target=self.send_ctrl, args=(), daemon=True)

    self._thread.start()

    return self


# Get command hex for drone

def get_hex(self):

    # XOR is for checksum
```

```
self.command_out=(self.throttle<<12|self.yaw<<8|self.pitch<<4|self.roll)

self.command_out = hex(self.command_out)[2::]

return self.command_out


# Send control to drone

def send_ctrl(self):

    while not self._stopped:

        if self._ignite_send == True:

            ignite_msg = 'st'

            self._package = ignite_msg.encode()

        else:

            self._package = self.get_hex().encode()

        #self.ssess.send(self._package)

        self.ssess.sendto(self._package, (self.ip, self.port))

        self.Flag_off()

        sleep(0.05)


# Close connection to drone

def close_connection(self):

    self._stopped = True

    if self._thread.daemon == False:

        self._thread.join()

        self.ssess.close()


# Return to default
```

```
def default(self):

    self.roll = 8

    self.pitch = 8

    self.throttle = 8

    self.yaw = 8

    self._takeoff_flag = False

    self._ignite_flag = False


# Increment control

def incremt(self, rl, pt, th, yw):

    self._value_to_change = [8, 8, 8, 8]

    self._change_val = [rl, pt, th, yw]

    for x in range(len(self._value_to_change)):

        self._value_to_change[x] += self._change_val[x]

        if self._value_to_change[x] <= 0:

            self._value_to_change[x] = 0

        if self._value_to_change[x] >= 15:

            self._value_to_change[x] = 15

    [self.roll, self.pitch, self.throttle, self.yaw] =
self._value_to_change


# Roll right

def roll_right(self):

    if self.roll < 15:

        self.roll += 1
```

```
# Pitch forward

def pitch_fwd(self):

    if self.pitch < 15:

        self.pitch += 1


# Increase throttle

def throttle_up(self):

    if self.throttle < 15:

        self.throttle += 1


# Yaw right

def yaw_right(self):

    if self.yaw < 15:

        self.yaw += 1


# Roll left

def roll_left(self):

    if self.roll > 0:

        self.roll -= 1


# Pitch backward

def pitch_bwd(self):

    if self.pitch > 0:

        self.pitch -= 1


# Decrease throttle
```

```
def throttle_dwn(self):  
  
    if self.throttle > 0:  
  
        self.throttle -= 1  
  
  
# Yaw left  
  
def yaw_left(self):  
  
    if self.yaw > 0:  
  
        self.yaw -= 1  
  
  
  
# Start engine  
  
def ignite(self):  
  
    if self._ignite_flag == False:  
  
        self._ignite_flag = True  
  
        self._ignite_send = True  
  
        self._ignite_timer = time()  
  
  
  
  
# Takeoff  
  
def takeoff(self):  
  
    if self._takeoff_flag == False:  
  
        self.throttle = 12  
  
        self._takeoff_flag = True  
  
        self._takeoff_timer = time()  
  
  
  
  
# Flip takeoff flag  
  
def Flag_off(self):  
  
    if self._ignite_flag == True:
```

```
        if (time() - self._ignite_timer >= 1) and (time() -
self._ignite_timer < 1.5):

            self._ignite_send = False

            self.roll = 8

            self.pitch = 8

            self.yaw = 8

            self.throttle = 0

        # Warming up engine

        if (time() - self._ignite_timer >= 1.5) and (time() -
self._ignite_timer < 2):

            self.throttle = 2

        if (time() - self._ignite_timer >= 2) and (time() -
self._ignite_timer < 2.5):

            self.throttle = 4

        if (time() - self._ignite_timer >= 2.5) and (time() -
self._ignite_timer < 3):

            self.throttle = 6

        if (time() - self._ignite_timer >= 3) and (time() -
self._ignite_timer < 4):

            self.throttle = 8

        # After starting engine, takeoff after 4s

        if (time() - self._ignite_timer >= 4):

            self._ignite_flag = False

            self.takeoff()

        if (self._takeoff_flag == True and (time() - self._takeoff_timer >=
4)):

            self.throttle = 8

            self._takeoff_flag = False
```

Code chạy trên Raspberry Pi (Python)

```
from time import time, sleep

from threading import Thread

import serial

import socket

class UDPMsg:

    def __init__(self, UDP_PORT):

        self.shutdown_thread = False

        self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

        self.sock.bind(("0.0.0.0", UDP_PORT))

        self.string = None

        self.in_contact = False

        self.received = False

        self._udp_timer = 0

    def start(self):

        self._thread = Thread(target=self.recvmes, args=(), daemon=True)

        self._thread.start()

        return self

    def recvmes(self):

        while not self.shutdown_thread:

            self.received = False

            data, addr = self.sock.recvfrom(1024)

            self.received = True
```



```
        self._udp_timer = time()

        self.string = data

        self.string_print = data.decode()

def check_msg(self):

    if (not self.received and (time() - self._udp_timer >= 2)):

        self.in_contact = False

    else:

        self.in_contact = True

    return self.in_contact

def close(self):

    self.shutdown_thread = True

    if self._thread.daemon == False:

        self._thread.join()

        self.sock.close()

get_text = UDPMsg(5005).start()

por = serial.Serial('/dev/ttyACM0', 9600)

sleep(1)

try:

    while True:

        if get_text.check_msg():

            print(time(), get_text.string_print)

            strsend = get_text.string
```

```
    else:

        strsend = bytes('8888','ascii')

        por.write(strsend)

        por.flush()

        if por.in_waiting > 0:

            strrecv = por.readline().decode('ascii','ignore')

            print(strrecv)

        sleep(0.05)

except KeyboardInterrupt:

    get_text.close()

    print('exit.')
```

Code trên Arduino Uno (C)

```
//Declaring global variables

byte last_channel_1, last_channel_2, last_channel_3, last_channel_4,
last_channel_5;

unsigned long loop_timer;

unsigned long timer_channel_1, timer_channel_2, timer_channel_3,
timer_channel_4, timer_channel_5, esc_loop_timer;

int receiver_input_1, receiver_input_2, receiver_input_3,
receiver_input_4, receiver_input_5;

unsigned long timer_1, timer_2, timer_3, timer_4, timer_5, current_time;

int center = 1500;

int throttle=1500, yaw=1500, pitch=1500, roll=1500;

//For converting string to PWM value

String incoming;

char inctochar[5];
```

```
int tempvalue[5];

//Setup routine

void setup(){

    DDRB |= B00100000;
    //Configure digital port 13 as output.

    DDRD |= B11110000;
    //Configure digital port 4, 5, 6 and 7 as output.

    //Use the led on the Arduino for startup indication.

    PORTB |= B00100000;
    //Turn on the warning led.


    Serial.begin(9600);
    //Setup Serial to read data

    Serial.setTimeout(10);
    //Pull timeout to 10ms to have low delay on readString

    PCICR |= (1 << PCIE0);
    //Set PCIE0 to enable PCMSK0 scan.

    PCMSK0 |= (1 << PCINT0);
    //Set PCINT0 (digital input 8) to trigger an interrupt on state change.

    PCMSK0 |= (1 << PCINT1);
    //Set PCINT1 (digital input 9)to trigger an interrupt on state change.

    PCMSK0 |= (1 << PCINT2);
    //Set PCINT2 (digital input 10)to trigger an interrupt on state change.

    PCMSK0 |= (1 << PCINT3);
    //Set PCINT3 (digital input 11)to trigger an interrupt on state change.

    PCMSK0 |= (1 << PCINT4);
    //Set PCINT4 (digital input 12)to trigger an interrupt on state change.


    //When everything is done, turn off the led.
```

```
    PORTB &= B11011111;
//Turn off the warning led.

    loop_timer = micros();
//Set the timer for the next loop.

}

//Main program loop

void loop(){

    //The refresh rate is 100Hz. That means the esc's need there pulse
    every 10ms.

    while(micros() - loop_timer < 10000);
//We wait until 10000us are passed.

    //If channel 5 is higher than 1600, activate pilot by UART

    if (receiver_input_5 > 1600){

        PORTB |= B00100000;
//Turn on the warning led.

        getFromSerial();

    }

    else {

        PORTB &= B11011111;
//Turn off the warning led.

        throttle = receiver_input_1;

        yaw = receiver_input_2;

        pitch = receiver_input_3;

        roll = receiver_input_4;

    }

}
```

```
    loop_timer = micros();
//Set the timer for the next loop.

    PORTD |= B11110000;
//Set digital outputs 4,5,6 and 7 high.

    timer_channel_1 = throttle + loop_timer;
//Calculate the time of the falling edge of the esc-1 pulse.

    timer_channel_2 = yaw + loop_timer;
//Calculate the time of the falling edge of the esc-2 pulse.

    timer_channel_3 = pitch + loop_timer;
//Calculate the time of the falling edge of the esc-3 pulse.

    timer_channel_4 = roll + loop_timer;
//Calculate the time of the falling edge of the esc-4 pulse.


    while(PORTD >= 16){
//Stay in this loop until output 4,5,6 and 7 are low.

        esc_loop_timer = micros();
//Read the current time.

        if(timer_channel_1 <= esc_loop_timer) PORTD &= B11101111;
//Set digital output 4 to low if the time is expired.

        if(timer_channel_2 <= esc_loop_timer) PORTD &= B11011111;
//Set digital output 5 to low if the time is expired.

        if(timer_channel_3 <= esc_loop_timer) PORTD &= B10111111;
//Set digital output 6 to low if the time is expired.

        if(timer_channel_4 <= esc_loop_timer) PORTD &= B01111111;
//Set digital output 7 to low if the time is expired.

    }

}

//This routine is called every time input 8, 9, 10, 11 or 12 changed
state. This is used to read the receiver signals.

//More information about this subroutine can be found in this video:
```

```
//https://youtu.be/bENj11KQbvo

ISR(PCINT0_vect){

    current_time = micros();

    //Channel 1=====

    if(PINB & B00000001){
//Is input 8 high?

        if(last_channel_1 == 0){
//Input 8 changed from 0 to 1.

            last_channel_1 = 1;
//Remember current input state.

            timer_1 = current_time;
//Set timer_1 to current_time.

        }

    }

    else if(last_channel_1 == 1){
//Input 8 is not high and changed from 1 to 0.

        last_channel_1 = 0;
//Remember current input state.

        receiver_input_1 = current_time - timer_1;
//Channel 1 is current_time - timer_1.

        if(receiver_input_1 > 2500) receiver_input_1 = center;

    }

    //Channel 2=====

    if(PINB & B00000010 ){
//Is input 9 high?

        if(last_channel_2 == 0){
//Input 9 changed from 0 to 1.

            last_channel_2 = 1;
//Remember current input state.
```

```
        timer_2 = current_time;
//Set timer_2 to current_time.

    }

}

else if(last_channel_2 == 1){
//Input 9 is not high and changed from 1 to 0.

    last_channel_2 = 0;
//Remember current input state.

    receiver_input_2 = current_time - timer_2;
//Channel 2 is current_time - timer_2.

    if(receiver_input_2 > 2500) receiver_input_2 = center;
}

//Channel 3=====

if(PINB & B00000100 ){
//Is input 10 high?

    if(last_channel_3 == 0){
//Input 10 changed from 0 to 1.

        last_channel_3 = 1;
//Remember current input state.

        timer_3 = current_time;
//Set timer_3 to current_time.

    }

}

else if(last_channel_3 == 1){
//Input 10 is not high and changed from 1 to 0.

    last_channel_3 = 0;
//Remember current input state.

    receiver_input_3 = current_time - timer_3;
//Channel 3 is current_time - timer_3.

    if(receiver_input_3 > 2500) receiver_input_3 = center;
```

```
    }

    //Channel 4=====

    if(PINB & B00001000 ){
//Is input 11 high?

        if(last_channel_4 == 0){
//Input 11 changed from 0 to 1.

            last_channel_4 = 1;
//Remember current input state.

            timer_4 = current_time;
//Set timer_4 to current_time.

        }

    }

    else if(last_channel_4 == 1){
//Input 11 is not high and changed from 1 to 0.

        last_channel_4 = 0;
//Remember current input state.

        receiver_input_4 = current_time - timer_4;
//Channel 4 is current_time - timer_4.

        if(receiver_input_4 > 2500) receiver_input_4 = center;

    }

    //Channel 5=====

    if(PINB & B00010000 ){
//Is input 12 high?

        if(last_channel_5 == 0){
//Input 12 changed from 0 to 1.

            last_channel_5 = 1;
//Remember current input state.

            timer_5 = current_time;
//Set timer_5 to current_time.

        }

    }
```



```
    }

    else if(last_channel_5 == 1){
//Input 12 is not high and changed from 1 to 0.

        last_channel_5 = 0;
//Remember current input state.

        receiver_input_5 = current_time - timer_5;
//Channel 5 is current_time - timer_5.

        if(receiver_input_5 > 2500) receiver_input_5 = center;
    }
}

//Subroutine for reading signal
void read_signal(){

    Serial.print("8:");

    if(receiver_input_1 - (center - 20) < 0) Serial.print("vvv");
    else if(receiver_input_1 - (center + 20) > 0) Serial.print("^^^");
    else Serial.print("--");

    Serial.print(receiver_input_1);

    Serial.print(" 9:");

    if(receiver_input_2 - (center - 20) < 0) Serial.print(">>>");
    else if(receiver_input_2 - (center + 20) > 0) Serial.print("<<<");
    else Serial.print("--");

    Serial.print(receiver_input_2);

    Serial.print(" 10:");
```

```
    if(receiver_input_3 - (center - 20) < 0) Serial.print("vvv");

    else if(receiver_input_3 - (center + 20) > 0) Serial.print("^^^");

    else Serial.print("-+-");

    Serial.print(receiver_input_3);

    Serial.print(" 11:");

    if(receiver_input_4 - (center - 20) < 0) Serial.print(">>>");

    else if(receiver_input_4 - (center + 20) > 0) Serial.print("<<<");

    else Serial.print("-+-");

    Serial.print(receiver_input_4);

    Serial.print(" 12:");

    if(receiver_input_5 - (center - 20) < 0) Serial.print("vvv");

    else if(receiver_input_5 - (center + 20) > 0) Serial.print("^^^");

    else Serial.print("-+-");

    Serial.println(receiver_input_5);

}
```

```
//Subroutine for getting data from serial (for example "8888")
```

```
void getFromSerial(){

    if (Serial.available() > 0) {

        incoming = Serial.readString();

        incoming.toCharArray(inctochar, 5);

        if (inctochar[0] == 's'){

            throttle = 1100;

            yaw = 1100;
```

```
    pitch = 1100;

    roll = 1900;

}

else {

    for (int i=0; i < 4; i++) {

        if (inctochar[i] >= 'a' & inctochar[i] <= 'f'){

            tempvalue[i] = int(inctochar[i]) - 'a' + 10;

        }

        else if (inctochar[i] >= '0' & inctochar[i] <= '9') {

            tempvalue[i] = int(inctochar[i]) - '0';

        }

        else tempvalue[i] = 8;

    }

    throttle = tempvalue[0]*25 + 1300;

    yaw = 1700 - tempvalue[1]*25;

    pitch = tempvalue[2]*25 + 1300;

    roll = 1700 - tempvalue[3]*25;

}

}
```
