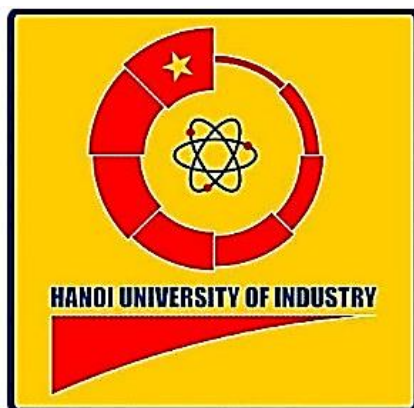


TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI
TRƯỜNG CƠ KHÍ – Ô TÔ
KHOA CƠ ĐIỆN TỬ



BÁO CÁO ĐỒ ÁN ĐO LƯỜNG VÀ ĐIỀU KHIỂN ROBOT

ĐỀ TÀI
NGHIÊN CỨU ỨNG DỤNG CẢM BIẾN CHIỀU SÂU PHÂN LOẠI SẢN PHẨM TRONG SẢN XUẤT CÔNG NGHIỆP

Giáo viên hướng dẫn: ThS. Bùi Huy Anh

Nhóm sinh viên thực hiện:	Nguyễn Thành Đạt	2022607446
	Phạm Đức Duy	2022602556
	Lê Doãn Sơn	2022603879

Mã lớp học phần: 20241ME6133001

Hà nội - 2024

MỤC LỤC

MỤC LỤC	1
DANH MỤC HÌNH ẢNH	3
DANH MỤC BẢNG BIỂU	5
LỜI MỞ ĐẦU	6
CHƯƠNG I: GIỚI THIỆU CHUNG	7
1.1 Tổng quan về ứng dụng của đề tài.	7
1.1.1 Ý tưởng.	7
1.1.2 Mục tiêu nghiên cứu.	7
1.1.3 Phương pháp nghiên cứu.	7
1.1.4 Đối tượng nghiên cứu.	8
1.2 Kết quả dự kiến, ý nghĩa.	8
1.2.1 Kết quả dự kiến	8
1.2.2 Ý nghĩa của nghiên cứu	8
CHƯƠNG II: TỔNG QUAN CẤU TRÚC HỆ THỐNG	9
2.1. Hệ thống cơ khí.	9
2.1.1. Hệ thống băng chuyền.	9
2.1.2. Hệ thống phân loại.....	10
2.1.3. Khung giá đỡ.	11
2.1.4. Cảm biến chiều sâu Visionary-T.	12
2.2. Cơ sở lý thuyết	16
2.2.1. Giới thiệu về Deep Learning.	16
2.2.2. Giới thiệu chung về họ gia đình YOLO.	18
2.2.3. YOLOv8 và YOLOv11	19
2.2.4. Giới thiệu về API.....	23
2.2.5. Giới thiệu về GG Colab.....	25
2.2.6. Giới thiệu về RoboFlow.	27
CHƯƠNG III: THIẾT KẾ CHẾ TẠO VÀ THỬ NGHIỆM MÔ HÌNH HỆ THỐNG.	29
3.1 Xây dựng chương trình lập trình.	29
3.1.1. Chuẩn bị dữ liệu	29
3.1.2. Gán nhãn và đa dạng hóa dữ liệu bằng Roboflow.....	29
3.1.2 Train model với Google Colab	29
3.1.3 Thiết kế lập trình API	31
3.2 Thiết kế chế tạo mô hình cơ khí.	31
3.2.1. Chế tạo khung giá đỡ.....	31
3.2.2. Chế tạo hệ thống phân loại.	33

3.2.3. Hệ thống băng chuyền.	38
3.2.4. Cảm biến chiều sâu	40
3.2.5 Hoàn thiện mô hình.	42
3.3 Thử nghiệm hệ thống	43
3.4 Kết quả và đánh giá.....	43
CHƯƠNG IV: KẾT LUẬN	45
TÀI LIỆU THAM KHẢO	46
PHỤ LỤC 1: LƯU ĐỒ THUẬT TOÁN.....	47
PHỤ LỤC 2: CODE PYTHON VÀ VI ĐIỀU KHIỂN	48
PHỤ LỤC 3: BẢN VẼ SOLIDWORKS BĂNG CHUYỀN VÀ KHUNG	55

DANH MỤC HÌNH ẢNH

Hình 2. 1. Mô hình băng chuyền.....	9
Hình 2. 2. Mô hình khung giá đỡ.....	11
Hình 2. 3. Cảm biến chiều sâu Visionary-T CX V3S100-2X	13
Hình 2. 4. Field of view (Trường nhìn)	14
Hình 2. 5. Sơ đồ nguyên lý hoạt động của Visionary-T	16
Hình 2. 6. Cơ chế hoạt động Machine learning	17
Hình 2. 7. Cơ chế hoạt động Deep Learning	17
Hình 2. 8. Các phiên bản yolo tính đến năm 2023	18
Hình 2. 9. Ví dụ về cách hoạt động của YOLO.....	19
Hình 2. 10. Hiệu suất các phiên bản YOLOv8 phát hiện đối tượng trên COCO	21
Hình 2. 11. Hiệu suất của yolov11 so với các phiên bản cũ.....	22
Hình 2. 12. Hiệu suất các phiên bản YOLOv11 phát hiện đối tượng trên COCO.	23
Hình 2. 13. Tổng quan giao diện API trên máy tính và điện thoại.....	25
Hình 2. 14. Google Colab.	26
Hình 3. 1. Giao diện Sopas	29
Hình 3. 2. Đồ thị biểu hiện các chỉ số đánh giá mô hình yolov8.....	30
Hình 3. 3. Đồ thị biểu hiện các chỉ số đánh giá mô hình yolov11.....	30
Hình 3. 4. Khung giá đỡ.....	33
Hình 3. 5. Arduino UNO R3	34
Hình 3. 6. Sơ đồ chân của Arduino UNO R3	35
Hình 3. 7. Động cơ giảm tốc MG996R.....	35
Hình 3. 8. Sơ đồ chân của động cơ giảm tốc MG996R.....	36
Hình 3. 9. Sơ đồ kết nối Arduino và động cơ Servo.....	37

Hình 3. 10. Thanh gạt.....	37
Hình 3. 11. Hộp đựng vật phẩm.....	38
Hình 3. 12. Hệ thống tay gạt.....	38
Hình 3. 13. Băng chuyền.....	40
Hình 3. 14. Cảm biến chiều sâu Visionary-T CX V3S100-2X	41
Hình 3. 15. Mô hình hoàn thiện	42
Hình 3. 16. Sơ đồ khối mô hình hóa hệ thống	42
Hình 3. 17. Tiến hành thử nghiệm	43
Hình 3. 18. Kết quả thử nghiệm Yolov8 và Yolov11	43

DANH MỤC BẢNG BIỂU

Bảng 2. 1. Các chi tiết cơ khí trong hệ thống băng truyền	10
Bảng 2. 2. Các chi tiết cơ khí của hệ thống phân loại	10
Bảng 2. 3. Chi tiết cơ khí trong khung giá đỡ.....	12
Bảng 2. 4. Thông số kỹ thuật của cảm biến Visionary-T	14
Bảng 2. 5. Phạm vi quét của cảm biến tại từng khoảng làm việc.....	15
Bảng 3. 1. Các bộ phận cấu tạo khung giá đỡ.....	31
Bảng 3. 2. Thông số kỹ thuật của Arduino UNO R3.....	34
Bảng 3. 3. Thông số kỹ thuật động cơ giảm tốc MG996R.....	36
Bảng 3. 4. Các bộ phận cấu tạo băng truyền.....	39
Bảng 3. 5. Thông số kỹ thuật của động cơ giảm tốc GB37-520 24V22RPM	40
Bảng 3. 6. Các bộ phận kết nối và cấp nguồn của cảm biến	41

LỜI MỞ ĐẦU

Cảm biến chiều sâu (depth camera) là một thiết bị công nghệ tiên tiến được thiết kế để thu thập thông tin về khoảng cách giữa camera và các vật thể trong không gian ba chiều. Khác với camera truyền thống chỉ ghi nhận hình ảnh 2D, cảm biến chiều sâu có khả năng đo đạc và tái tạo chi tiết cấu trúc 3D của môi trường, nhờ vào việc sử dụng các công nghệ như ánh sáng hồng ngoại, phát hiện thời gian bay (Time-of-Flight), hoặc ánh sáng có cấu trúc (Structured Light).

Cảm biến chiều sâu được ứng dụng rộng rãi trong nhiều lĩnh vực, từ thực tế ảo tăng cường (AR/VR), robot và xe tự hành, đến nhận diện cử chỉ, quét 3D, và các hệ thống giám sát an ninh. Với khả năng cung cấp dữ liệu không gian trực quan và chính xác, cảm biến chiều sâu đã mở ra nhiều cơ hội cho sự phát triển công nghệ và cải thiện trải nghiệm người dùng trong thế giới số hóa ngày nay. Và đặc biệt còn được sử dụng nhiều trong hệ thống công nghiệp ví dụ như trên các băng chuyền sản xuất để phân loại hàng hóa.

Trong quá trình thực hiện đồ án này, nhóm nghiên cứu đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và chỉ đạo nhiệt tình của thầy cô và bạn bè. Nhóm xin gửi lời cảm ơn đến thầy **Ths. Bùi Huy Anh**, người đã hướng dẫn tận tình, giúp đỡ nhóm trong suốt quá trình thực hiện đồ án.

Nhóm cũng xin chân thành cảm ơn các thầy cô giáo trong trường Đại học Công Nghiệp Hà Nội nói chung, các thầy cô trong khoa Cơ điện tử nói riêng đã hướng dẫn nhóm kiến thức về các môn đại cương cũng như các môn chuyên ngành, giúp nhóm có được cơ sở lý thuyết vững vàng và tạo điều kiện giúp đỡ trong suốt quá trình học tập.

Nhóm xin gửi lời cảm ơn đến bạn bè đã hỗ trợ và chia sẻ kinh nghiệm cho nhóm trong thời gian qua.

Cuối cùng nhóm xin chúc quý thầy cô và cùng toàn thể các bạn sinh viên nhiều sức khỏe và thành công trong mọi công việc.

Xin chân thành cảm ơn!

CHƯƠNG I: GIỚI THIỆU CHUNG

1.1 Tổng quan về ứng dụng của đề tài.

Trong các hệ thống sản xuất công nghiệp hiện đại, băng chuyền là thành phần quan trọng để tự động hóa việc vận chuyển và xử lý sản phẩm. Việc phát hiện sản phẩm trên băng chuyền không chỉ hỗ trợ giám sát chất lượng mà còn giúp cải thiện hiệu suất, giảm chi phí vận hành, và nâng cao độ chính xác trong sản xuất. Tuy nhiên, các phương pháp truyền thống như cảm biến thông thường hoặc kiểm tra thủ công thường khó có thể đáp ứng với các sản phẩm ngày càng đa dạng, phức tạp.

Do đó, trong đề án lần này, nhóm nghiên cứu sẽ tập trung vào việc ứng dụng cảm biến công nghiệp hiện đại Visionary-T kết hợp với các thuật toán học sâu nhằm tối ưu hóa quy trình phát hiện và phân loại sản phẩm trong sản xuất công nghiệp. Điều này không chỉ mang lại hiệu quả cao hơn mà còn giúp khai thác tiềm năng của công nghệ hiện đại trong ngành sản xuất tự động hóa.

1.1.1 Ý tưởng.

Đề tài: “**Nghiên cứu ứng dụng cảm biến cảm biến chiều sâu phân loại sản phẩm trong sản xuất công nghiệp.**”

Hệ thống được đề xuất sẽ sử dụng cảm biến Visionary-T để thu thập dữ liệu từ sản phẩm trên băng chuyền và áp dụng các thuật toán học sâu để phân loại sản phẩm theo yêu cầu. Hệ thống này sẽ được đánh giá về tính chính xác, hiệu suất, và khả năng triển khai trong môi trường thực tế.

1.1.2 Mục tiêu nghiên cứu.

Mục tiêu của nghiên cứu:

- Hiểu rõ và phân tích hoạt động của cảm biến Visionary-T, bao gồm cấu tạo, nguyên lý hoạt động và khả năng ứng dụng trong công nghiệp.
- Ứng dụng Visionary-T kết hợp với các mô hình học sâu (Deep Learning) như YOLO, Faster R-CNN để phát hiện và phân loại sản phẩm trên băng chuyền.
- Đánh giá hiệu suất các mô hình học sâu trong việc ứng dụng vào sản xuất công nghiệp, từ đó.

1.1.3 Phương pháp nghiên cứu.

Nghiên cứu sẽ được thực hiện theo các phương pháp sau:

a) Nghiên cứu tài liệu

Tìm hiểu về cảm biến Visionary-T, các thuật toán học sâu phổ biến và phương pháp tích hợp chúng vào sản xuất công nghiệp.

b) Mô phỏng và triển khai thực nghiệm

Thu thập dữ liệu từ cảm biến Visionary-T trong môi trường băng chuyền mô phỏng cùng với huấn luyện và thử nghiệm các mô hình học sâu trên tập dữ liệu thu thập được.

c) Đánh giá và so sánh

So sánh hiệu suất (độ chính xác, tốc độ, khả năng triển khai) của các thuật toán học sâu khi kết hợp với cảm biến Visionary-T trong môi trường thực tế.

d) Đề xuất giải pháp

Đưa ra khuyến nghị về ứng dụng phù hợp trong các hệ thống sản xuất công nghiệp.

1.1.4 Đối tượng nghiên cứu.

- **Cảm biến Visionary-T:** Một loại cảm biến 3D tiên tiến dùng để nhận diện hình dạng và đặc điểm của sản phẩm trên băng chuyền.
- **Các thuật toán học sâu:** Các mô hình như YOLO, Faster R-CNN, hoặc Mask R-CNN, được sử dụng để phát hiện và phân loại sản phẩm dựa trên dữ liệu từ cảm biến.
- **Hệ thống băng chuyền mô phỏng:** Môi trường thử nghiệm với các sản phẩm mẫu, giúp kiểm chứng tính khả thi của hệ thống.

1.2 Kết quả dự kiến, ý nghĩa.

1.2.1 Kết quả dự kiến

- Xây dựng được hệ thống tích hợp cảm biến Visionary-T và các thuật toán học sâu để phân loại sản phẩm trên băng chuyền.
- Đánh giá chi tiết hiệu suất của các thuật toán học sâu, xác định mô hình phù hợp nhất với các yêu cầu công nghiệp.
- Đưa ra báo cáo đầy đủ về khả năng ứng dụng của cảm biến Visionary-T trong môi trường sản xuất công nghiệp thực tế.

1.2.2 Ý nghĩa của nghiên cứu

a) Ý nghĩa thực tiễn

- Hỗ trợ tự động hóa trong sản xuất công nghiệp, giúp cải thiện năng suất và giảm chi phí lao động.
- Đưa ra giải pháp ứng dụng cảm biến hiện đại để nâng cao chất lượng sản phẩm và giảm tỷ lệ lỗi trong sản xuất.

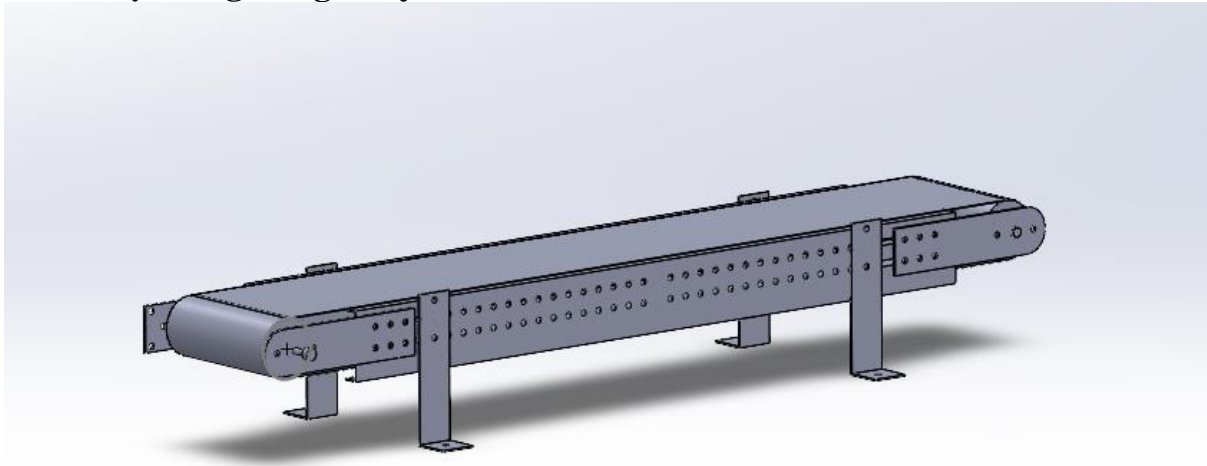
b) Ý nghĩa khoa học

- Đóng góp vào nghiên cứu ứng dụng công nghệ cảm biến 3D và học sâu trong lĩnh vực sản xuất.
- Làm rõ vai trò của các công nghệ hiện đại trong việc giải quyết các thách thức thực tế của ngành công nghiệp 4.0.

CHƯƠNG II: TỔNG QUAN CẤU TRÚC HỆ THỐNG

2.1. Hệ thống cơ khí.

2.1.1. Hệ thống băng chuyền.



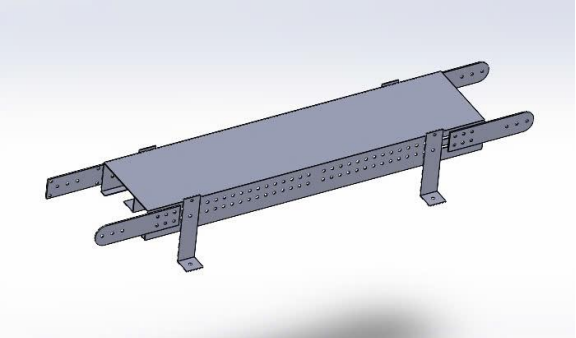
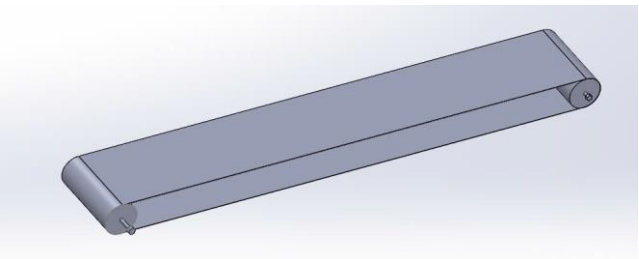
Hình 2. 1. Mô hình băng chuyền

Băng chuyền có nhiệm vụ đưa sản phẩm từ vị trí ban đầu, qua phạm vi quét của cảm biến chiều sâu. Qua quá trình xử lý, vật đến với các hộp phân loại sau khi bị tay gạt tác động.

Băng chuyền có chiều dài 60 cm, chiều rộng 10 cm và bao gồm các bộ phận: Khung băng tải, con lăn, động cơ, dây đai băng tải.

- Khung băng tải: Là nơi chịu toàn bộ tải trọng của cơ cấu vận chuyển. Khung phải có kết cấu phù hợp với không gian cho phép.
- Trục lăn: Con lăn giúp việc di chuyển các sản phẩm mặt đáy phẳng có trọng lượng lớn nhanh gọn và dễ dàng.
- Động cơ: Dùng để chuyển tốc độ tới băng tải qua bộ truyền đai.
- Dây đai băng tải: Dùng để nối giữa 2 đầu con lăn và dùng vận chuyển vật trên băng tải.

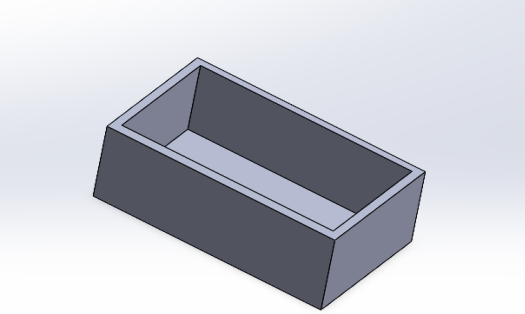
Bảng 2. 1. Các chi tiết cơ khí trong hệ thống băng truyền

1	Khung băng tải	
2	Trục lăn và băng tải	

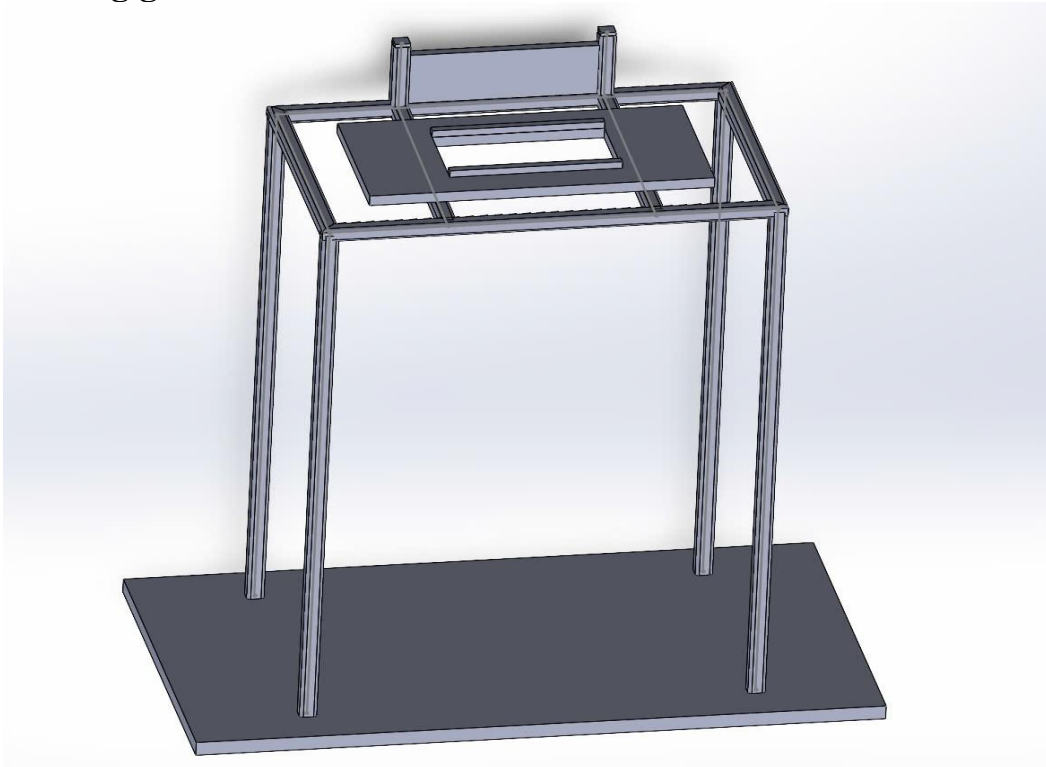
2.1.2. Hệ thống phân loại.

Hệ thống phân loại được sử dụng để phân loại các đồ vật sau khi qua quá trình xử lý. Hệ thống này gồm hai phần chính là tay gạt và hộp chứa đồ vật.

Bảng 2. 2. Các chi tiết cơ khí của hệ thống phân loại

1	Thanh gạt	
2	Hộp chứa đồ vật	

2.1.3. Khung giá đỡ.



Hình 2. 2. Mô hình khung giá đỡ


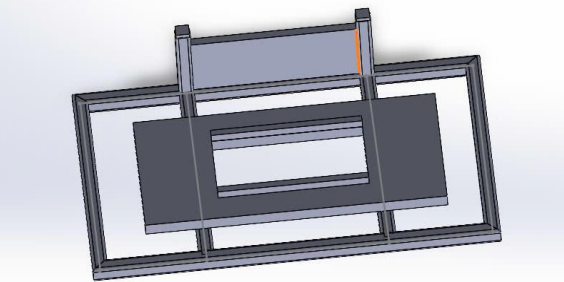
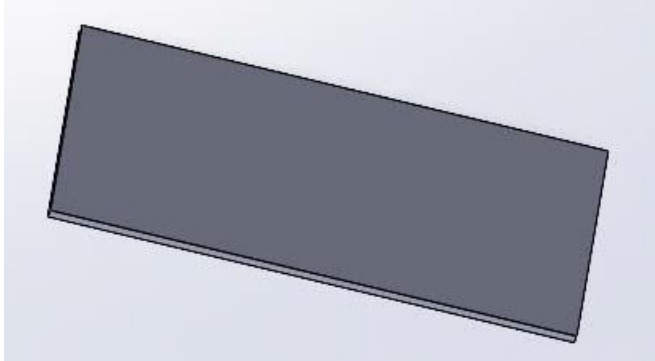
Khung giá đỡ được thiết kế để giữ cảm biến chiều sâu với độ cao phù hợp, giúp thực hiện hiệu quả nhiệm vụ thu thập dữ liệu phục vụ phân loại đồ vật.

Khung giá đỡ gồm hai phần: Giá đỡ cảm biến và phần đế.

- Phần giá đỡ cảm biến có chiều cao 74 cm, được làm bằng cách ghép nối các thanh nối lại với nhau. Phần mặt gá cảm biến là một tấm hình chữ nhật có chiều dài 40 cm, chiều rộng 29 cm và có độ dày là 0.3 cm.

- Phần đế là một tấm hình chữ nhật có chiều dài 80 cm, chiều rộng 29 cm và có độ dày là 1 cm.

Bảng 2. 3. Chi tiết cơ khí trong khung giá đỡ

1	Khung đỡ	
2	Mặt gá cảm biến	
3	Đế khung	

2.1.4. Cảm biến chiều sâu Visionary-T.

Tổng quan

Visionary-T sử dụng công nghệ snapshot 3D dựa trên đo lường thời gian bay (time-of-flight) giúp cảm biến ghi nhận và cung cấp thông tin về độ sâu của từng điểm ảnh trong thời gian thực. Cảm biến này lý tưởng cho các ứng dụng trong nhà, cung cấp thông tin không gian ba chiều chính xác, bao gồm cả khoảng cách và cường độ sáng cho mỗi ảnh chụp.

Các cảm biến Visionary-T của SICK có khả năng truyền dữ liệu dạng thô hoặc đã được xử lý tùy vào yêu cầu ứng dụng, đảm bảo cung cấp dữ liệu phù hợp để hỗ trợ các quyết định tự động.



Hình 2. 3. Cảm biến chiều sâu Visionary-T CX V3S100-2X

Cấu tạo và tính năng

Visionary-T gồm 2 bộ phận chính đó là 2 đầu phát tia hồng ngoại và 1 đầu thu tia phản xạ lại. Đồng thời Visionary-T còn có các tính năng chi tiết :

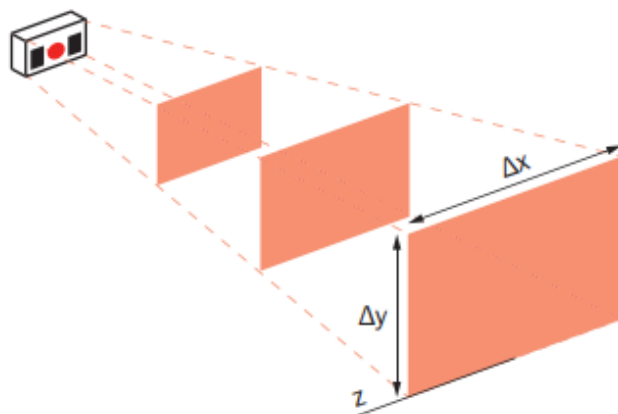
- **Tốc độ chụp ảnh cao:** Cảm biến có thể ghi nhận tới 50 hình ảnh 3D mỗi giây, hỗ trợ tối ưu cho các ứng dụng đòi hỏi phản ứng nhanh như điều khiển robot hoặc giám sát dây chuyền sản xuất.
- **Độ phân giải và dữ liệu chi tiết:** Với 176 x 144 điểm ảnh trên mỗi lần chụp, Visionary-T cung cấp hơn 25.000 giá trị khoảng cách và cường độ ánh sáng, hỗ trợ nhiều ứng dụng đa dạng từ cảnh báo va chạm đến phát hiện đối tượng.
- **Khả năng tùy biến dữ liệu:** Dữ liệu có thể được định dạng theo nhu cầu cụ thể của ứng dụng, với các tùy chọn dữ liệu như dạng cực hay tọa độ, cho phép tích hợp dễ dàng vào hệ thống công nghiệp.
- **Dải nhiệt độ hoạt động và độ bền cao:** Hoạt động tốt trong khoảng từ 0 °C đến 50 °C, cùng với chuẩn bảo vệ IP67, giúp cảm biến chống chịu bụi và nước tốt trong môi trường khắc nghiệt.

Thông số kỹ thuật

Bảng 2. 4. Thông số kỹ thuật của cảm biến Visionary-T

Working distance (Phạm vi hoạt động)	0.5m → 60m
Detection angle (góc phát hiện)	$69^\circ \times 56^\circ$ (góc ngang 69° và góc thẳng đứng 56°)
Pixel count (độ phân giải)	176 x 144 pixels
Light sensitivity (độ nhạy sáng của cảm biến)	< 50 kLux (sunlight)
Repeatability (độ lặp lại của cảm biến)	≥ 2 mm, at a range of 1 m (ở cùng một khoảng cách đo 1m sai số giữa các lần đo không vượt quá 2mm) ≥ 7 mm, at a range of 7 m
Supply voltage (điện áp cung cấp)	24V DC
Power consumption (điện năng tiêu thụ)	≤ 22 W
Ambient Temperature (nhiệt độ môi trường)	Operation (hoạt động): $0^\circ\text{C} \rightarrow 50^\circ\text{C}$ Storage (lưu trữ): $-20^\circ\text{C} \rightarrow 70^\circ\text{C}$

Field of view của Visionary-T được mô tả qua hình và bảng như dưới đây:



Hình 2. 4. Field of view (Trường nhìn)

Bảng 2. 5. Phạm vi quét của cảm biến tại từng khoảng làm việc

Axial working distance (z)	Range (Δx)	Range (Δy)
0.5 m	0.7 m	0.5 m
1.0 m	1.4 m	1.0 m
1.5 m	2.1m	1.6 m
2.0 m	2.8 m	2.1 m
3.0 m	4.1 m	3.1 m
4.0 m	5.5 m	4.2 m
5.0 m	6.8 m	5. m
10.0 m	13.7 m	10.6 m
15.0 m	20.6 m	15.9 m
20.0 m	27.4 m	21.2 m
40.0 m	54.9 m	42.5 m

Nguyên lý hoạt động

Cảm biến Visionary-T là một máy ảnh 3D hoạt động dựa trên nguyên tắc ToF (Time-of-Flight), một kỹ thuật đo khoảng cách dựa trên thời gian mà sóng ánh sáng hoặc sóng siêu âm di chuyển từ nguồn phát đến đối tượng và phản xạ trở lại cảm biến.

1. Visionary-T phát ra sóng ánh sáng hồng ngoại từ nguồn sáng tích hợp.
2. Ánh sáng này chiếu đến vật thể và bị bề mặt vật thể phản xạ trở lại cảm biến
3. Bộ phận tiếp nhận ánh sáng của cảm biến sẽ nhận lại ánh sáng phản xạ và đo thời gian từ lúc sóng ánh sáng được phát ra đến khi nhận lại.
4. Thời gian này (time-of-flight) được sử dụng để tính toán khoảng cách từ cảm biến đến đối tượng.

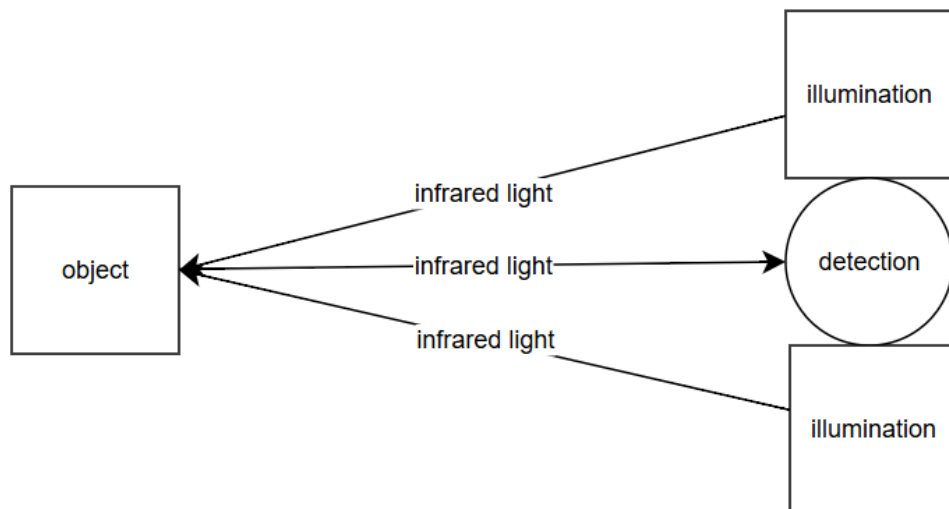
$$d = \frac{c * t}{2} \quad (1.1)$$

Trong đó:

- d: khoảng cách từ điểm phát sóng ánh sáng của cảm biến đến đối tượng (đơn vị: m) .
- c: tốc độ của ánh sáng ($\sim 3.10^8$ m/s).
- t: thời gian bay của ánh sáng (đơn vị s).

Sau khi tính toán được khoảng cách từ cảm biến đến mỗi điểm trong không gian, các giá trị khoảng cách này được sắp xếp và biểu diễn thành một bức ảnh chiều sâu và ảnh intensity. Trong ảnh chiều sâu này, mỗi pixel sẽ tương ứng với một giá trị khoảng cách từ cảm biến đến vật thể trong không gian. Còn ảnh intensity thì mỗi pixel của ảnh sẽ là giá trị cường độ sáng tương ứng.

2.2. Cơ sở lý thuyết



Hình 2. 5. Sơ đồ nguyên lý hoạt động của Visionary-T

2.2.1. Giới thiệu về Deep Learning.

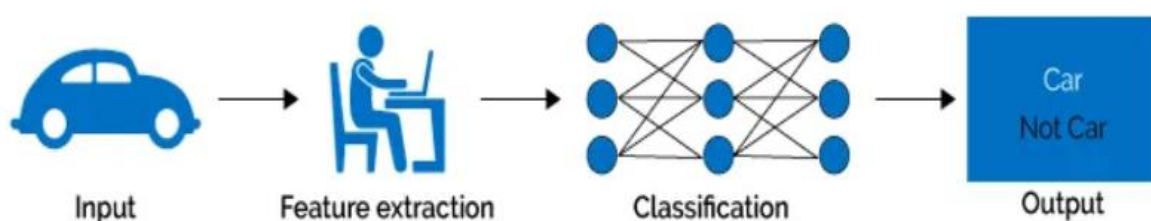
Deep Learning (học sâu) có thể được xem là một lĩnh vực con của Machine Learning – ở đó các máy tính sẽ học và cải thiện chính nó thông qua các thuật toán. Deep Learning được xây dựng dựa trên các khái niệm phức tạp hơn rất nhiều, chủ yếu hoạt động với các mạng nơ-ron nhân tạo để bắt chước khả năng tư duy và suy nghĩ của bộ não con người.

Cơ chế hoạt động

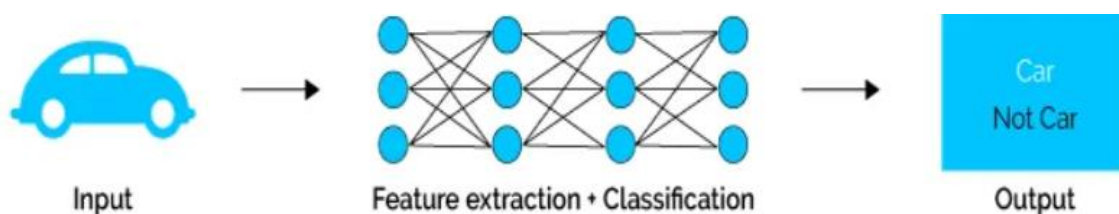
Deep Learning là một phương pháp của Machine Learning. Mạng nơ-ron nhân tạo trong Deep Learning được xây dựng để mô phỏng khả năng tư duy của bộ não con người.

Một mạng nơ-ron bao gồm nhiều lớp (layer) khác nhau, số lượng layer càng nhiều thì mạng sẽ càng “sâu”. Trong mỗi layer là các nút mạng (node) và được liên kết với những lớp liền kề khác. Mỗi kết nối giữa các node sẽ có một trọng số tương ứng, trọng số càng cao thì ảnh hưởng của kết nối này đến mạng nơ-ron càng lớn.

Mỗi nơ-ron sẽ có một hàm kích hoạt, về cơ bản thì có nhiệm vụ “chuẩn hoá” đầu ra từ nơ-ron này. Dữ liệu được người dùng đưa vào mạng nơ-ron sẽ đi qua tất cả layer và trả về kết quả ở layer cuối cùng, gọi là output layer.



Hình 2. 6. Cơ chế hoạt động Machine Learning



Hình 2. 7. Cơ chế hoạt động Deep Learning

Trong quá trình huấn luyện mô hình mạng nơ-ron, các trọng số sẽ được thay đổi và nhiệm vụ của mô hình là tìm ra bộ giá trị của trọng số sao cho phán đoán là tốt nhất.

Các hệ thống Deep Learning yêu cầu phần cứng phải rất mạnh để có thể xử lý được lượng dữ liệu lớn và thực hiện các phép tính phức tạp. Nhiều mô hình Deep Learning có thể mất nhiều tuần, thậm chí nhiều tháng để triển khai trên những phần cứng tiên tiến nhất hiện nay.

Ưu điểm và nhược điểm

- Ưu điểm:

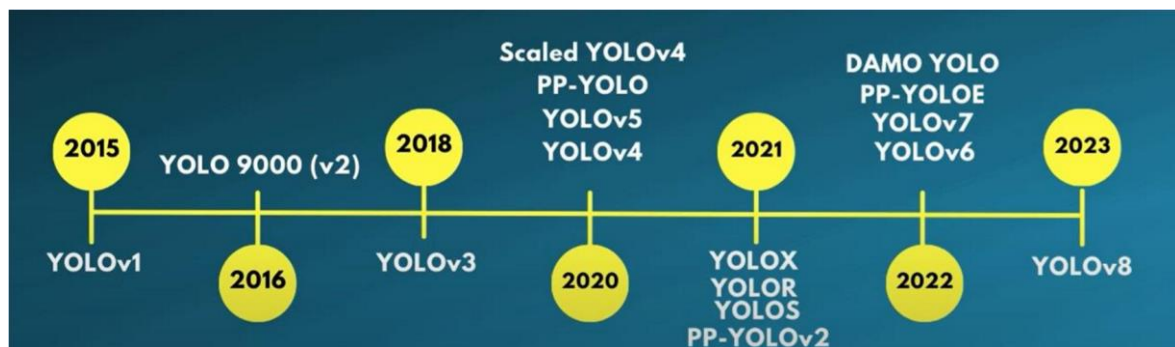
- Kiến trúc mạng nơ-ron linh hoạt, có thể dễ dàng thay đổi để phù hợp với nhiều vấn đề khác nhau.
- Có khả năng giải quyết nhiều bài toán phức tạp với độ chính xác rất cao.
- Tính tự động hoá cao, có khả năng tự điều chỉnh và tự tối ưu.
- Có khả năng thực hiện tính toán song song, hiệu năng tốt, xử lý được lượng dữ liệu lớn.

- Nhược điểm:

- Cần có khối lượng dữ liệu rất lớn để tận dụng tối đa khả năng của Deep Learning.
- Chi phí tính toán cao vì phải xử lý nhiều mô hình phức tạp.
- Chưa có nền tảng lý thuyết mạnh mẽ để lựa chọn các công cụ tối ưu cho Deep Learning.

2.2.2. Giới thiệu chung về họ gia đình YOLO.

YOLO (You Only Look Once) là các thuật toán phát hiện đối tượng thời gian thực, nổi bật với tốc độ xử lý nhanh và độ chính xác cao. Được Joseph Redmond và các cộng sự đề xuất vào năm 2015, đến nay đã có tất cả 11 phiên bản, được đặt tên là YOLOv_x (với $x = \{1,2,3,4,5,6,7,8,9,10,11\}$) . Với những ưu điểm vượt trội, YOLO đã trở thành một trong những phương pháp phát hiện đối tượng được sử dụng rộng rãi trong lĩnh vực thị giác máy tính.



Hình 2. 8. Các phiên bản yolo tính đến năm 2023

2.2.2.1. Nguyên tắc hoạt động của YOLO

Các bước hoạt động của YOLO:

a) Phân chia ảnh đầu vào thành các lưới (grids).

- YOLO chia ảnh đầu vào thành một lưới gồm $S \times S$ ô vuông. Mỗi ô sẽ chịu trách nhiệm dự đoán các đối tượng có tâm nằm trong ô đó.

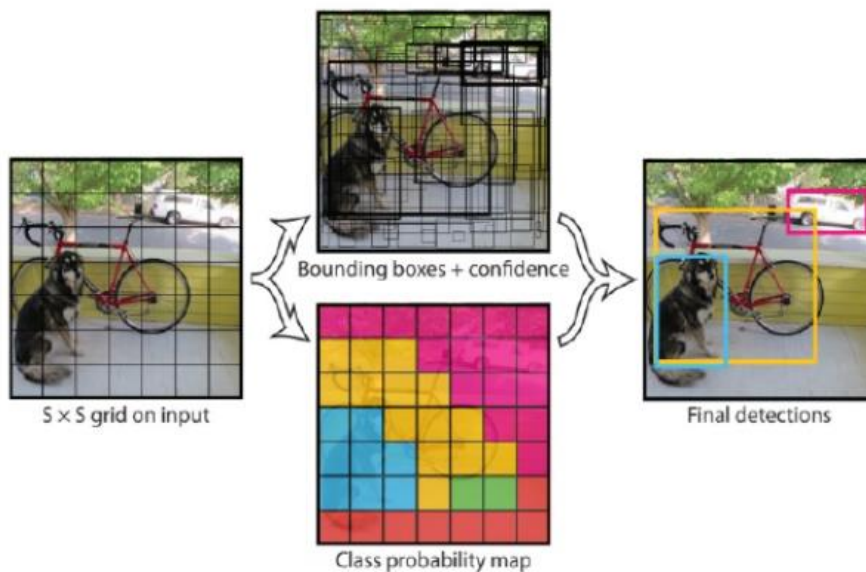
b) Dự đoán các bounding box.

Mỗi ô lưới dự đoán một số bounding box với các thông tin sau:

- **Tọa độ của bounding box:** x, y, w, h trong đó (x, y) là tọa độ tương đối của tâm bounding box so với ô lưới, (w, h) là chiều rộng và chiều cao của bounding box.
- **Độ tin cậy (confidence score):** Phản ánh mức độ chắc chắn rằng bounding box chứa một đối tượng và độ chính xác của dự đoán.
- **Xác suất (Class probabilities):** Các xác suất cho từng lớp đối tượng có thể có trong hình ảnh.

c) Kết quả trả về.

- YOLO kết hợp thông tin về vị trí của bounding box và xác suất lớp đối tượng để phân loại và xác định vị trí chính xác của từng đối tượng trong ảnh.



Hình 2. 9. Ví dụ về cách hoạt động của YOLO

Ưu điểm và nhược điểm của YOLO

a) Ưu điểm:

- Tốc độ xử lý nhanh: YOLO có khả năng xử lý khung hình ở tốc độ 45 fps (đối với mạng lớn) đến 150 fps (đối với mạng nhỏ), đảm bảo khả năng hoạt động tốt với thời gian thực.
- Có khả năng khái quát hóa tốt hơn khi xử lý các hình ảnh mới hoặc trong điều kiện ánh sáng, góc chụp khác nhau.
- Dễ triển khai.

b) Nhược điểm:

- Độ chính xác thấp với các đối tượng phức tạp.
- Khó phát hiện các vật thể ở gần nhau hoặc chồng lấn lên nhau vì mỗi lưới chỉ có thể đề xuất hai hộp giới hạn.
- Khó khăn trong việc phát hiện các vật thể nhỏ.

2.2.3. YOLOv8 và YOLOv11

YOLOv8

a) Tổng quan

YOLOv8 là một mô hình nhận dạng đối tượng dựa trên mạng convolutional neural network (CNN) được phát triển bởi Joseph Redmon và nhóm nghiên cứu của ông tại Đại học Washington.

YOLOv8 là phiên bản nâng cấp của YOLOv7, với khả năng nhận diện đối tượng nhanh hơn và chính xác hơn. Điều này được đạt được thông qua một số cải tiến, bao gồm mạng kim tự tháp đặc trưng, các mô-đun chú ý không gian và các kỹ thuật tăng cường dữ liệu tiên tiến.

Mô hình YOLOv8 sử dụng một mạng neural kiến trúc darknet-53 để trích xuất đặc trưng của hình ảnh và áp dụng thuật toán nhận dạng đối tượng YOLOv8 trên các đặc trưng đó.

b) Các tính năng chính của YOLOv8

- Tăng cường mô hình bằng cách thêm các kênh phân tán để tăng tốc độ tính toán.
- Sử dụng kỹ thuật Attention để cải thiện khả năng nhận dạng đối tượng của mô hình.
- Áp dụng phương pháp đào tạo mới để tăng tốc độ hội tụ.
- Sử dụng kiến trúc mạng nơ-ron mới: Sử dụng kiến trúc YOLOv4 làm cơ sở để tăng hiệu suất và độ chính xác của mô hình.
- Tích hợp cơ chế tự động điều chỉnh tỷ lệ tăng kích thước của hình ảnh đầu vào (AutoScale).
- Hỗ trợ giám sát bằng video (Video Supervision): Mô hình có khả năng phát hiện và giám sát vật thể trong các video và đưa ra dự đoán liên tục trên toàn bộ video.
- Tích hợp công nghệ Ensemble.
- Tính năng điều chỉnh tỷ lệ tự động (AutoAnchor): Cải thiện việc phát hiện đối tượng với nhiều tỷ lệ khác nhau.

c) Nhiệm vụ YOLOv8 hỗ trợ

- Phát hiện đối tượng.
- Theo dõi đối tượng .
- Phân đoạn phiên bản.

d) Các phiên bản của YOLOv8

YOLOv8 cung cấp nhiều phiên bản mô hình khác nhau, phù hợp với từng yêu cầu cụ thể về tốc độ và độ chính xác.

- YOLOv8n (Nano): Phiên bản nhỏ gọn, nhanh nhất, phù hợp cho các ứng dụng yêu cầu thời gian xử lý nhanh như trên các thiết bị nhúng.
- YOLOv8s (Small), YOLOv8m (Medium): Phiên bản cân bằng giữa tốc độ và độ chính xác, thích hợp cho các bài toán vừa và nhỏ.
- YOLOv8l (Large), YOLOv8x (Extra Large): Phiên bản lớn hơn, phù hợp cho các ứng dụng yêu cầu độ chính xác cao nhưng có thể chấp nhận tốc độ xử lý chậm hơn.

Model	size (pixels)	mAP ^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Hình 2. 10. Hiệu suất các phiên bản YOLOv8 phát hiện đối tượng trên COCO

e) Ưu điểm và hạn chế của YOLOv8.

- Ưu điểm:

- **Tốc độ:** YOLOv8 được đánh giá là nhanh chóng và thời gian phản hồi thấp, giúp xử lý các tác vụ nhận diện đối tượng và phân-segment ảnh trong thời gian thực.
- **Độ chính xác:** YOLOv8 được xây dựng trên các tiến bộ về học sâu và thị giác máy tính, đảm bảo độ chính xác cao trong việc nhận diện đối tượng.
- **Sự linh hoạt:** YOLOv8 hỗ trợ việc nhận diện đối tượng và phân-segment trên cả GPU và CPU, tận dụng các công nghệ như TensorRT của Nvidia và OpenVino của Intel.

- Hạn chế:

- Phải được huấn luyện trên một tập dữ liệu đủ lớn và đa dạng để đạt được hiệu quả cao nhất.
- Yêu cầu các tài nguyên tính toán cao để đạt được tốc độ xử lý nhanh và chính xác.
- YOLOv8 có thể không hoạt động tốt trong tất cả các môi trường và có thể cần thêm điều chỉnh hoặc tối ưu hóa để đạt được hiệu suất tối ưu.

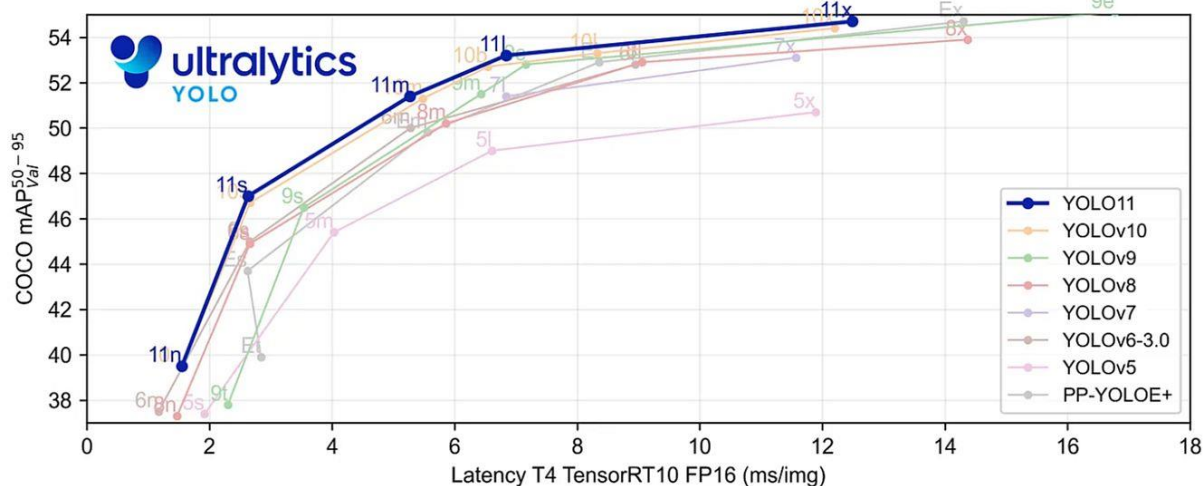
YOLOv11

a) Tổng quan

YOLOv11 là phiên bản mới nhất của YOLO, một phiên bản phát hiện đối tượng thời gian thực nâng cao. Gia đình YOLO bước vào một chương mới với YOLO11, một mô hình có khả năng và khả năng thích ứng hơn, vượt qua ranh giới của thị giác máy tính.

Mô hình hỗ trợ các tác vụ thị giác máy tính như ước tính tư thế và phân đoạn cá thể. YOLOv11 được đánh giá cao vì hiệu quả tốt hơn và kiến trúc được tối ưu hóa.

YOLOv11 sở hữu thiết kế cải tiến, cho phép phát hiện chính xác hơn các chi tiết tinh tế - ngay cả trong những tình huống khó khăn. Có tính năng trích xuất tốt hơn và có thể trích xuất nhiều mẫu và chi tiết từ ảnh.



Hình 2. 11. Hiệu suất của yolov11 so với các phiên bản cũ

b) Các tính năng của YOLOv11

- Phát hiện đối tượng: Định vị đối tượng trong hình ảnh hoặc video bằng cách vẽ các hộp giới hạn cùng với điểm tin cậy. Hữu ích cho các ứng dụng như lái xe tự động, camera giám sát hoặc trạm thu phí giao thông.
- Phân đoạn phiên bản: Xác định và phân đoạn các đối tượng hoặc cá nhân trong một hình ảnh. Hữu ích cho việc chụp ảnh y tế, sản xuất và hơn thế nữa.
- Phân loại hình ảnh: Kỹ thuật này phân loại hình ảnh thành các nhóm được thiết lập trước. Nó làm cho nó trở nên hoàn hảo cho các mục đích sử dụng như phân loại sản phẩm thương mại điện tử hoặc quan sát động vật.
- Theo dõi đối tượng: Tính năng này, rất quan trọng đối với nhiều ứng dụng thời gian thực, theo dõi và giám sát chuyển động của các đối tượng trên một loạt khung hình video.
- Ước tính tư thế: Xác định các điểm chính trong khung hình ảnh hoặc video để theo dõi chuyển động hoặc cử chỉ của cơ thể, giúp nó hữu ích cho các ứng dụng như thực tế ảo, tập khiêu vũ và vật lý trị liệu.
- Phát hiện đối tượng định hướng (OBB): Phát hiện đối tượng có góc định hướng, cho phép định vị chính xác hơn các vật phẩm bị nghiêng hoặc xoay. Tính năng này đặc biệt hữu ích cho các ứng dụng như lái xe tự động, kiểm tra công nghiệp và phân tích hình ảnh từ máy bay không người lái hoặc vệ tinh.

c) Các phiên bản của YOLOv11

Giống với YOLOv8, YOLOv11 cũng có các phiên bản n, s, m, l, x, phân biệt theo kích thước của mô hình.

Model	size (pixels)	mAP ^{val} ₅₀₋₉₅	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n	640	39.5	56.1 ± 0.8	1.5 ± 0.0	2.6	6.5
YOLO11s	640	47.0	90.0 ± 1.2	2.5 ± 0.0	9.4	21.5
YOLO11m	640	51.5	183.2 ± 2.0	4.7 ± 0.1	20.1	68.0
YOLO11l	640	53.4	238.6 ± 1.4	6.2 ± 0.1	25.3	86.9
YOLO11x	640	54.7	462.8 ± 6.7	11.3 ± 0.2	56.9	194.9

Hình 2. 12. Hiệu suất các phiên bản YOLOv11 phát hiện đối tượng trên COCO

d) Ưu điểm của YOLOv11

- Độ chính xác tốt hơn với ít thông số hơn: ít tham số hơn làm cho mô hình nhanh hơn mà không ảnh hưởng đáng kể đến tính chính xác của v11.
- Hỗ trợ nhiều tác vụ khác nhau: YOLOv11 có khả năng thực hiện một loạt các tác vụ bao gồm ước tính tư thế, nhận dạng đối tượng, phân loại hình ảnh, phân đoạn phiên bản và phát hiện đối tượng định hướng (OBB).
- Cải thiện tốc độ và hiệu quả: Tốc độ xử lý nhanh hơn đạt được thông qua các thiết kế kiến trúc được cải thiện và quy trình đào tạo nhằm thỏa hiệp giữa độ chính xác và hiệu suất.
- Cải thiện trích xuất đặc trưng: YOLOv11 có kiến trúc neck và backbone tốt hơn để cải thiện khả năng trích xuất đặc trưng, từ đó dẫn đến việc phát hiện đối tượng chính xác hơn.
- Khả năng thích ứng giữa các ngữ cảnh: YOLOv11 có thể thích ứng với nhiều ngữ cảnh, chẳng hạn như nền tảng đám mây, thiết bị biên và hệ thống tương thích với GPU NVIDIA.

2.2.4. Giới thiệu về API.

API là gì?

API (Application Programming Interface) là một tập hợp các giao diện và giao thức cho phép các ứng dụng phần mềm giao tiếp với nhau. API hoạt động như một cầu nối, giúp các hệ thống khác nhau trao đổi dữ liệu và thực hiện các chức năng mà không cần phải hiểu chi tiết nội bộ của từng hệ thống.

Ví dụ: Trong ứng dụng phân loại hàng hóa, API có thể kết nối giữa cảm biến (như camera hoặc máy quét) và hệ thống phần mềm phân loại, giúp xử lý và truyền tải dữ liệu nhanh chóng, hiệu quả.

Vai trò của API trong ứng dụng cảm biến phân loại hàng hóa.

API đóng vai trò quan trọng trong việc xây dựng hệ thống phân loại hàng hóa bằng cảm biến, bao gồm:

- **Truy xuất dữ liệu cảm biến:** API cho phép lấy dữ liệu từ cảm biến (hình ảnh, thông số kỹ thuật, v.v.) để xử lý.

- **Tích hợp mô hình phân loại:** API truyền dữ liệu đầu vào đến mô hình học máy hoặc thuật toán và nhận kết quả phân loại.
- **Tối ưu hóa quy trình làm việc:** Thay vì xử lý dữ liệu thủ công, API tự động hóa việc trao đổi và xử lý dữ liệu giữa các thành phần trong hệ thống.

API giúp dễ dàng mở rộng hệ thống để hỗ trợ thêm các cảm biến hoặc chức năng mới.

Streamlit

a) Giới thiệu về Streamlit

Streamlit là một framework Python mã nguồn mở, hỗ trợ xây dựng các ứng dụng giao diện người dùng (UI) trực quan và dễ sử dụng. Trong ứng dụng này, Streamlit đóng vai trò là cầu nối giữa người dùng và các API cảm biến, cung cấp một nền tảng tương tác trực quan để nhập, xử lý và hiển thị dữ liệu phân loại hàng hóa.

b) Mục tiêu thiết kế Streamlit

- Hiển thị dữ liệu cảm biến (ví dụ: hình ảnh hoặc thông số đo lường).
- Hỗ trợ người dùng tải lên dữ liệu đầu vào.
- Phân loại hàng hóa và hiển thị kết quả theo thời gian thực.
- Tạo giao diện trực quan, dễ sử dụng mà không yêu cầu người dùng phải có kiến thức lập trình.

c) Kết quả thiết kế API ứng dụng cho cảm biến Visionary-T

Truy cập vào API bằng cách chạy lệnh ‘Streamlit run main.py’ trên terminal. Sau khi chạy lệnh sẽ trả về Local URL và Network URL và chọn một trong 2 để truy cập.

Giao diện ứng dụng gồm 4 tính năng chính:

- Visionary-T: Giới thiệu về thiết bị cảm biến Visionary-T
- STL File Viewer: Hiển thị trực quan các bản vẽ cơ khí cho phép xem STL file tải lên từ máy tính
- Train Model: Kiểm tra cấu hình máy tính.

Nhập số Epochs và Batch size mong muốn để bắt đầu quá trình Train Model.

- Validation Results

Hiển thị kết quả quá trình Validation trong 1 Batch, bao gồm:

- Predicts
- Loss Plot
- Confusion Matrix
- Test Model

Bao gồm 2 mode để lựa chọn:

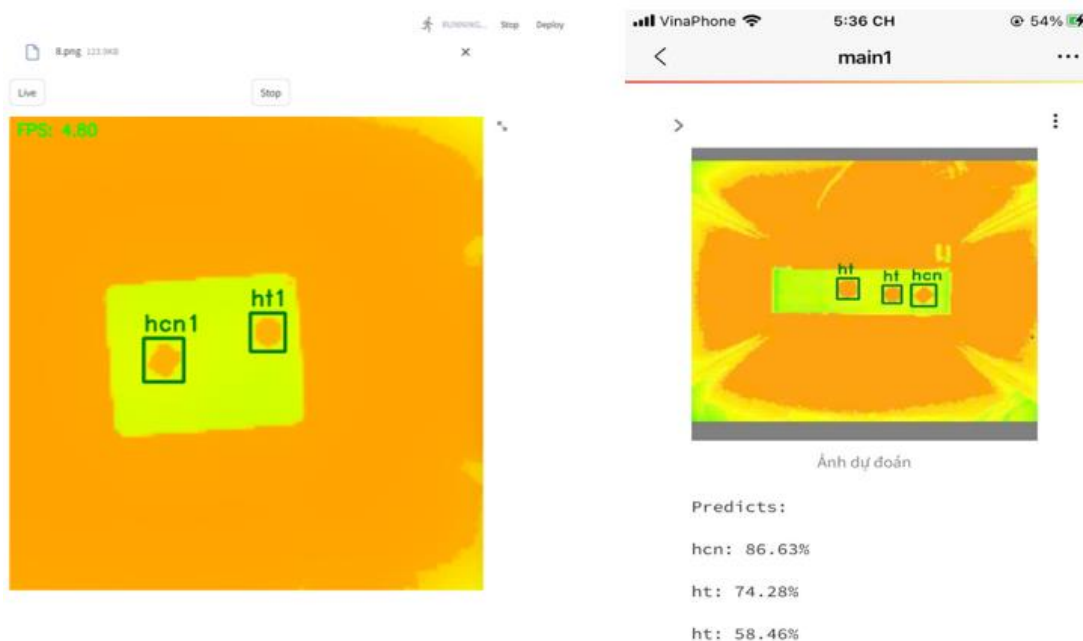
Upload file: Tải ảnh lên từ thiết bị, API sẽ tự động truyền ảnh được upload vào model để đưa ra kết quả dự đoán.

Real-time Detection (Máy tính cá nhân phải kết nối trực tiếp với Visionary-T):

API sẽ tự động tìm và kết nối real-time với Visionary-T trung gian qua máy tính cá nhân.

Đối với Real-Time Detection, sau khi detect được vật thể, API sẽ kết nối với Arduino và trả về hành động tương ứng.

Đồng thời nếu người dùng chỉ muốn dùng thiết bị ngoại vi khác để xem kết quả hoạt động trực tiếp của cảm biến mà không cần thiết gì thêm ta có thể sử dụng thiết bị khác truy cập vào Network URL để xem trực tiếp kết quả từ thiết bị khác có chung Network.



Hình 2.13. Tổng quan giao diện API trên máy tính và điện thoại

2.2.5. Giới thiệu về GG Colab.

Colaboratory hay còn gọi là **Google Colab**, là một sản phẩm từ Google Research, nó cho phép thực thi Python trên nền tảng đám mây, đặc biệt phù hợp với Data analysis, machine learning và giáo dục.

Colab không cần yêu cầu cài đặt hay cấu hình máy tính, mọi thứ có thể chạy thông qua trình duyệt, bạn có thể sử dụng tài nguyên máy tính từ CPU tốc độ cao và cả GPUs và cả TPUs đều được cung cấp cho bạn.

Sử dụng **Google Colab** có những lợi ích ưu việt như: sẵn sàng chạy Python ở bất kỳ thiết bị nào có kết nối Internet mà không cần cài đặt, chia sẻ và làm việc nhóm dễ dàng, sử dụng miễn phí GPU cho các dự án về AI.



Hình 2. 14. Google Colab.

Bên cạnh đó, Google Colab còn cung cấp cho bạn trải nghiệm lập trình Python tuyệt vời với những nâng cấp cực kỳ hữu ích không có trong Jupyter Notebook, Jupyterlab đơn thuần.

Một số tính năng có thể kể đến như sau:

- Tạo mục lục dựa trên các Heading viết bằng ngôn ngữ markdown giúp bạn dễ dàng cấu trúc Notebook làm việc của mình. Bạn cũng có thể thu gọn (Collapse) hay Mở rộng (Expand) các phần nội dung khi soạn thảo cực kỳ tiện lợi.
- Thêm hình ảnh, biểu mẫu dễ dàng với markdown giúp bạn trình bày báo cáo hoặc làm dashboard cực tiện lợi. Thậm chí bạn có thể ẩn các dòng code để trông Notebook gọn gàng hơn với tính năng biểu mẫu.
- Kết nối dễ dàng với Google Drive, Google Sheets để bắt tay vào phân tích dữ liệu “trên mây” hoàn toàn.
- Chạy Python trên Cloud hay Local Runtime (Python trên máy tính cá nhân) của bạn đều cho trải nghiệm tốt. Bạn vẫn tận dụng được tính năng tuyệt vời của Google Colab khi chạy với Python trên Local Runtime trong khi không bị Google tự động xóa dữ liệu khi kết thúc phiên làm việc như khi chạy trên Cloud.
- Tự động lưu lịch sử chỉnh sửa thành các phiên bản giúp bạn dễ dàng khôi phục lại phiên bản gần nhất nếu cần khi bạn gặp lỗi. Tính năng này tương tự như trên Google Sheets hay Google Docs, bạn thậm chí không cần đến Github để lưu trữ các phiên bản chỉnh sửa này.
- Cho phép tìm kiếm và chèn các đoạn mã được soạn thảo sẵn trong các Template (bởi bạn) vào Notebook. Tính năng này rất hay bởi bạn không cần

phải mở thêm nhiều file lưu trữ để tìm lại các đoạn code mẫu mình đã biết khi cần. Workflow lập trình Python trở nên đơn giản và hiệu quả hơn rất nhiều.

- Tạo dashboard viết bằng Python và chia sẻ với team dễ dàng nếu cần tương tự như Google DataStudio nhưng linh hoạt và mạnh mẽ hơn rất nhiều.

Tuy nhiên Google Colab có 1 nhược điểm đó là dữ liệu (bộ nhớ tạm) của phiên làm việc sẽ bị xóa sau khi bạn không active trong 1 thời gian nhất định để Colab đảm bảo có thể cung cấp tài nguyên cho nhiều người. Do đó mỗi khi mở Google Colab, nếu bạn cần sử dụng các thư viện của bên thứ 3 thì bạn cần install và import lại từ đầu để có thể sử dụng. Phiên bản Colab Pro giúp khắc phục điều này nhưng hiện tại không áp dụng cho thị trường Việt Nam.

2.2.6. Giới thiệu về RoboFlow.

Roboflow là một nền tảng toàn diện dành cho việc xây dựng, quản lý, và triển khai các mô hình thị giác máy tính (computer vision). Nó cung cấp các công cụ và dịch vụ giúp đơn giản hóa quá trình xử lý dữ liệu, huấn luyện mô hình, và triển khai ứng dụng, đặc biệt trong lĩnh vực như nhận diện đối tượng (object detection), phân loại (classification), và phân vùng (segmentation).

Các tính năng chính của Roboflow

a) Quản lý dữ liệu hình ảnh

- Tải lên và tổ chức tập dữ liệu từ nhiều nguồn khác nhau
- Tự động phân chia tập dữ liệu thành các nhóm huấn luyện (training), kiểm tra (validation), và thử nghiệm (testing).
- Hỗ trợ chuyển đổi định dạng annotation giữa các định dạng phổ biến như COCO, YOLO, Pascal VOC, ...

b) Tiền xử lý và tăng cường dữ liệu

- Cung cấp các công cụ chỉnh sửa kích thước, lọc và làm sạch dữ liệu.
- Thực hiện tăng cường dữ liệu (data augmentation) với các kỹ thuật như xoay, lật, thay đổi độ sáng, làm mờ, thêm nhiễu, cắt xén, v.v.
- Giúp cải thiện chất lượng và tính đa dạng của tập dữ liệu để tối ưu hóa hiệu suất mô hình

c) Hỗ trợ huấn luyện mô hình

- Tích hợp dễ dàng với các framework học sâu phổ biến như TensorFlow, PyTorch, YOLO, MMDetection, và Detectron2

- Tự động hóa quy trình tạo tập dữ liệu huấn luyện
- Cung cấp các gợi ý tối ưu để huấn luyện mô hình tốt hơn.

d) Triển khai mô hình

- Hỗ trợ triển khai mô hình thông qua API RESTful, giúp tích hợp nhanh vào ứng dụng thực tế.
- Tích hợp triển khai trên thiết bị edge (như Raspberry Pi, NVIDIA Jetson) hoặc trên các nền tảng cloud (AWS, Google Cloud, Azure)
- Đảm bảo việc triển khai dễ dàng và hiệu quả trong nhiều môi trường khác nhau.

Lợi ích khi sử dụng Roboflow

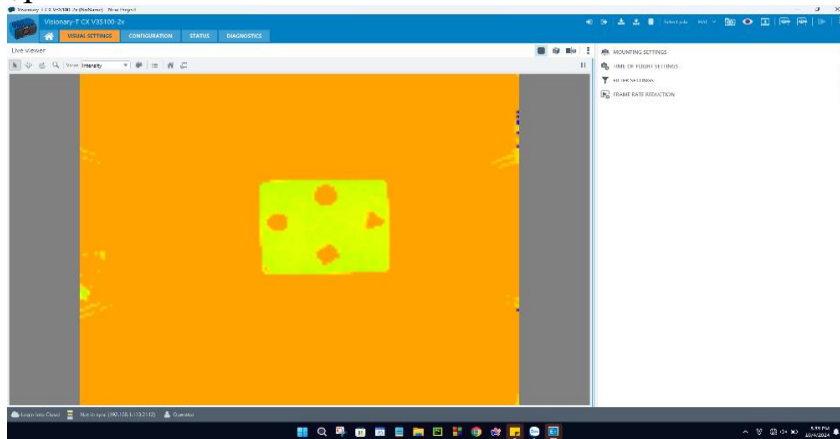
- Tiết kiệm thời gian: Giảm bớt các công việc phức tạp như tiền xử lý dữ liệu, tăng cường dữ liệu và chuyển đổi định dạng, hỗ trợ người dùng tập trung vào việc cải thiện mô hình.
- Đơn giản hóa quy trình: Giao diện thân thiện và trực quan, phù hợp cho cả người mới bắt đầu và chuyên gia trong lĩnh vực thị giác máy tính.
- Đồng bộ và chia sẻ: Hỗ trợ làm việc nhóm với tính năng chia sẻ tập dữ liệu và mô hình qua nền tảng cloud.
- Khả năng mở rộng: Dễ dàng mở rộng quy mô cho các dự án lớn với nhu cầu xử lý dữ liệu và triển khai trên nhiều thiết bị và nền tảng khác nhau.

CHƯƠNG III: THIẾT KẾ CHẾ TẠO VÀ THỬ NGHIỆM MÔ HÌNH HỆ THỐNG.

3.1 Xây dựng chương trình lập trình.

3.1.1. Chuẩn bị dữ liệu

- Bước 1: Chọn nguồn cài đặt từ device driver hoặc disk(nếu có).
- Bước 2: Sopas đề xuất các driver phù hợp với thiết bị. Lựa chọn driver phù hợp với nhu cầu.
- Bước 3: Tiến hành cài đặt driver.
- Bước 4: Sau khi cài xong driver ta có thể kết nối cảm biến với máy tính và bắt đầu thu thập data.



Hình 3. 1. Giao diện Sopas

3.1.2. Gán nhãn và đa dạng hóa dữ liệu bằng Roboflow.

- Bước 1: Tạo project và chọn bài toán của đề tài (Object Detection).
- Bước 2: Upload data đã tạo trước từ cảm biến.
- Bước 3: Chọn phần Annotate ở cột bên trái và bắt đầu gán nhãn dữ liệu. Bài toán là phân loại xem là hình tròn(ht), hình chữ nhật(hcn).
- Bước 4: Sau khi label xong bắt đầu gửi ảnh đã gán nhãn vào dataset và giao diện sẽ có thêm 2 mục xử lý ảnh nhằm đa dạng hóa dữ liệu.(Preprocessing và Augmentation). Chọn và Create.
- Bước 5: Tiếp sau đó sang phần Version (Train) ta có thể download dataset vừa được tạo về định dạng dữ liệu phù hợp mà ta cần sử dụng.(YOLOv11).

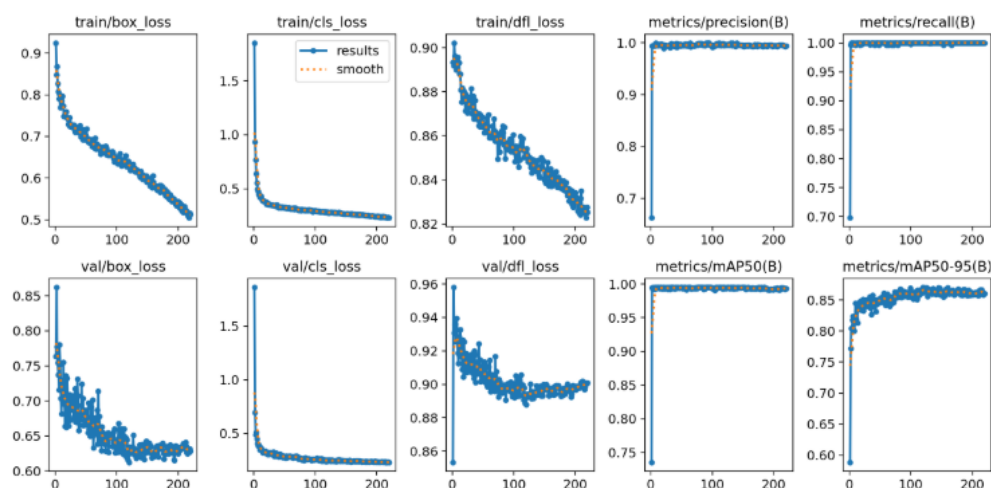
3.1.2 Train model với Google Colab

Các bước thực hiện:

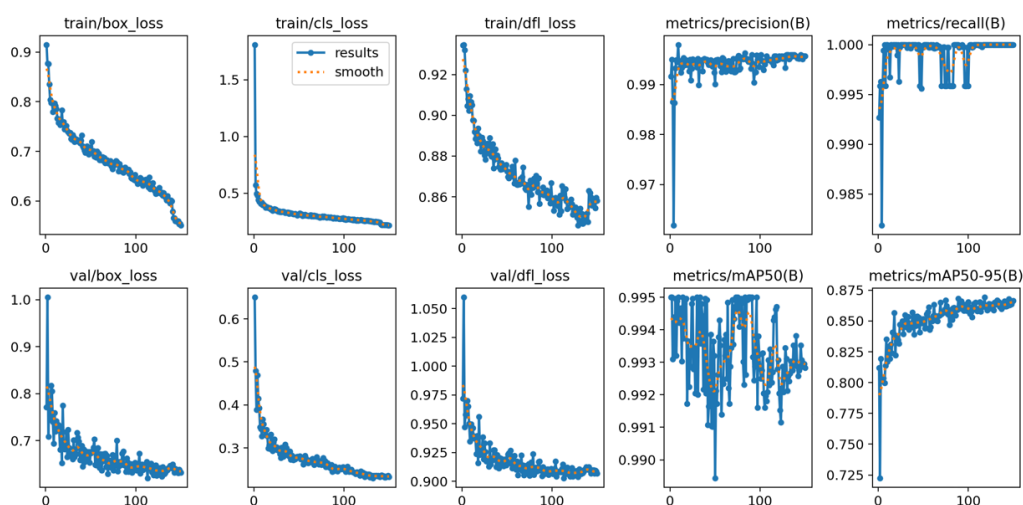
- Bước 1: Cài đặt thư viện Ultralytics của YOLO và download về model yolov8n.pt

- Bước 2: Kết nối Colab với GoogleDrive, rồi upload data đã tải từ Roboflow lên drive và tiến hành Train Model.
- Bước 3: Sau khi train xong kết quả trả về sẽ là một model mới với đường dẫn được trả về ở kết quả với đường dẫn là runs/detect/train. Lấy kết quả và tải về model 'best.pt' để sử dụng.
- Bước 4: Thực hiện đánh giá model thông qua các biểu đồ chỉ số Precision, Recall và Confidence, Confusion Matrix.

Đánh giá mô hình qua biểu đồ:



Hình 3. 2. Đồ thị biểu hiện các chỉ số đánh giá mô hình yolov8



Hình 3. 3. Đồ thị biểu hiện các chỉ số đánh giá mô hình yolov11

Có thể thấy các biểu đồ giữa train và val đều cùng một hướng giảm đồng thời chỉ số cũng không quá chênh lệch cho nên model hoạt động khá tốt.

3.1.3 Thiết kế lập trình API

Để thiết kế lập trình API, nhóm nghiên cứu đã sử dụng thư viện Streamlit. Streamlit là một framework Python mã nguồn mở, hỗ trợ xây dựng các ứng dụng giao diện người dùng (UI) trực quan và dễ sử dụng. Trong ứng dụng này, Streamlit đóng vai trò là cầu nối giữa người dùng và các API cảm biến, cung cấp một nền tảng tương tác trực quan để nhập, xử lý và hiển thị dữ liệu phân loại hàng hóa.

Mục tiêu thiết kế là:

- Hiển thị dữ liệu cảm biến (ví dụ: hình ảnh hoặc thông số đo lường).
- Hỗ trợ người dùng tải lên dữ liệu đầu vào.
- Phân loại hàng hóa và hiển thị kết quả theo thời gian thực.
- Tạo giao diện trực quan, dễ sử dụng mà không yêu cầu người dùng phải có kiến thức lập trình.






3.2 Thiết kế chế tạo mô hình cơ khí.



3.2.1. Chế tạo khung giá đỡ.

- Mặt gá cảm biến: Sử dụng tấm nhựa mica có chiều dài 40 cm, chiều rộng 29 cm, độ dày 0.3 cm. Ở trung tâm tấm mica, cắt một lỗ hình chữ nhật có chiều dài 16.5 cm, chiều rộng 6.5 cm.
- Phần đế khung: Sử dụng một tấm gỗ có chiều dài 80 cm, chiều rộng 40 cm, độ dày 1 cm làm phần đế cho khung giá đỡ. Dán kín mặt đế bằng tấm dán decan màu đen.
- Phần khung: lắp ráp các thanh nhôm định hình lại với nhau để tạo thành phần khung cho giá đỡ, đảm bảo chiều cao từ mặt đế lên đến đỉnh khung là 74 cm.
- Ghép nối các bộ phận gồm mặt gá cảm biến, phần đế, và khung lại với nhau để tạo thành khung giá đỡ hoàn chỉnh.

Bảng 3. 1. Các bộ phận cấu tạo khung giá đỡ

1	Tấm nhựa mica	
---	---------------	--

2	Tấm gỗ	
3	Tấm dán decan	
4	Thanh nhôm định hình	
5	Con trượt rãnh	
6	Ke góc vuông	

7	Bulong lục giác	
8	Chân đế nhôm định hình	

Khung giá đỡ hoàn chỉnh:



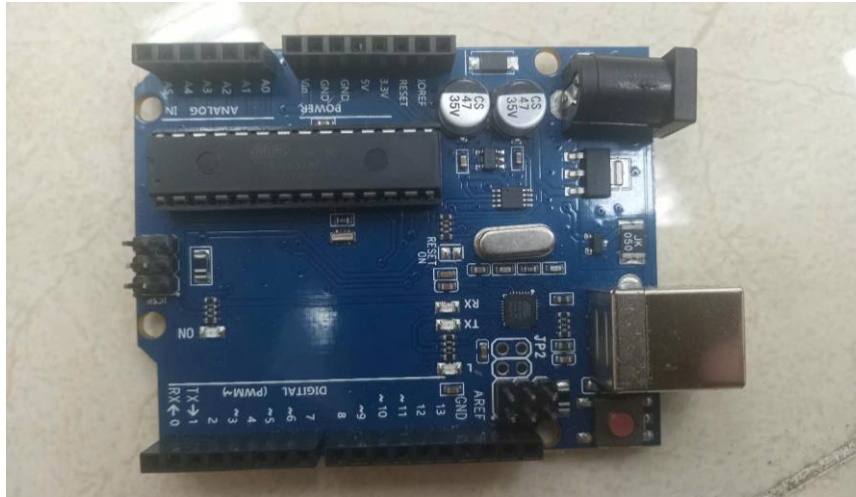
Hình 3. 4. Khung giá đỡ

3.2.2. Chế tạo hệ thống phân loại.

3.2.2.1. Thiết kế hệ thống điều khiển tay gạt.

Trong hệ thống phân loại của bài đồ án, Arduino được sử dụng làm vi điều khiển để gửi tín hiệu xung PWM điều khiển góc quay của động cơ servo, để có thể phân loại đồ vật vào đúng hộp đựng được chỉ định.

a) Arduino UNO R3



Hình 3. 5. Arduino UNO R3

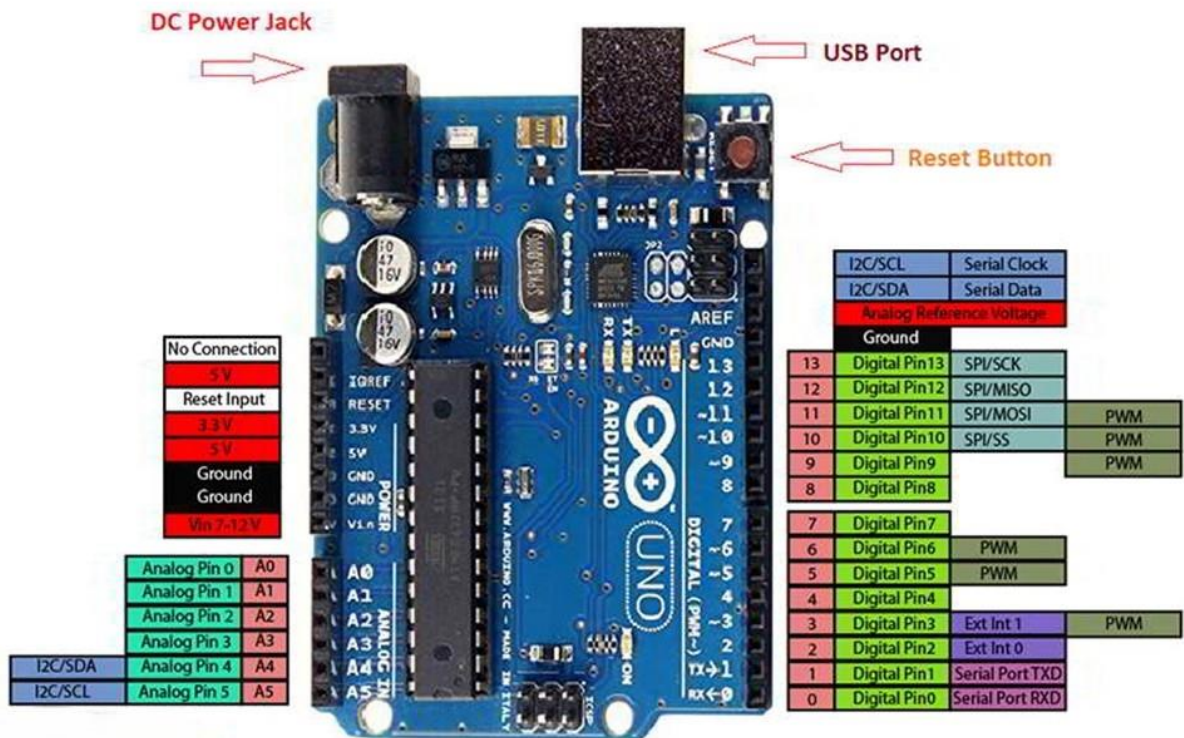
Arduino Uno R3 là một bảng mạch vi điều khiển nguồn mở dựa trên vi điều khiển Microchip ATmega328 được phát triển bởi Arduino.cc. Bảng mạch được trang bị các bộ chân đầu vào/ đầu ra Digital và Analog có thể giao tiếp với các bảng mạch mở rộng khác nhau.

Các thông số kỹ thuật của Arduino Uno R3

Bảng 3. 2. Thông số kỹ thuật của Arduino UNO R3

Chip điều khiển	Atmega328P
Điện áp hoạt động	5V
Điện áp đầu vào (khuyên dùng)	7 – 12V
Điện áp đầu vào (giới hạn)	6 – 20V
Số chân Digital	14
Số chân PWM Digital	6
Số chân Analog	6
Tốc độ thạch anh	16 MHz
Dòng điện DC trên mỗi chân I/O	20 mA
Dòng điện DC trên chân 3.3V	50 mA
Chiều dài	68.6 mm
Chiều rộng	53.4 mm
Cân nặng	25 g

Sơ đồ chân của Arduino UNO R3



Hình 3. 6. Sơ đồ chân của Arduino UNO R3

b) Động cơ Servo.

Sử dụng động cơ Servo MG996R trong hệ thống phân loại.



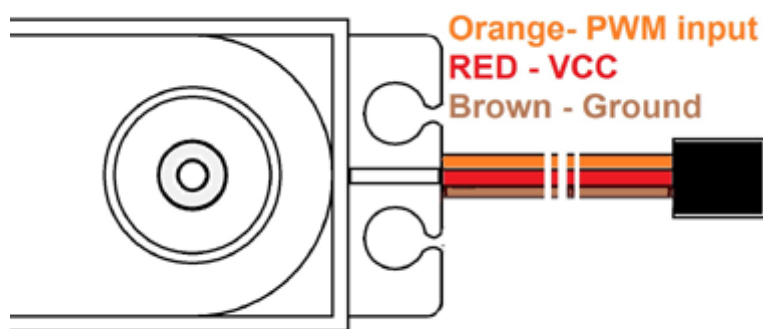
Hình 3. 7. Động cơ giảm tốc MG996R

Thông số kỹ thuật

Bảng 3. 3. Thông số kỹ thuật động cơ giảm tốc MG996R

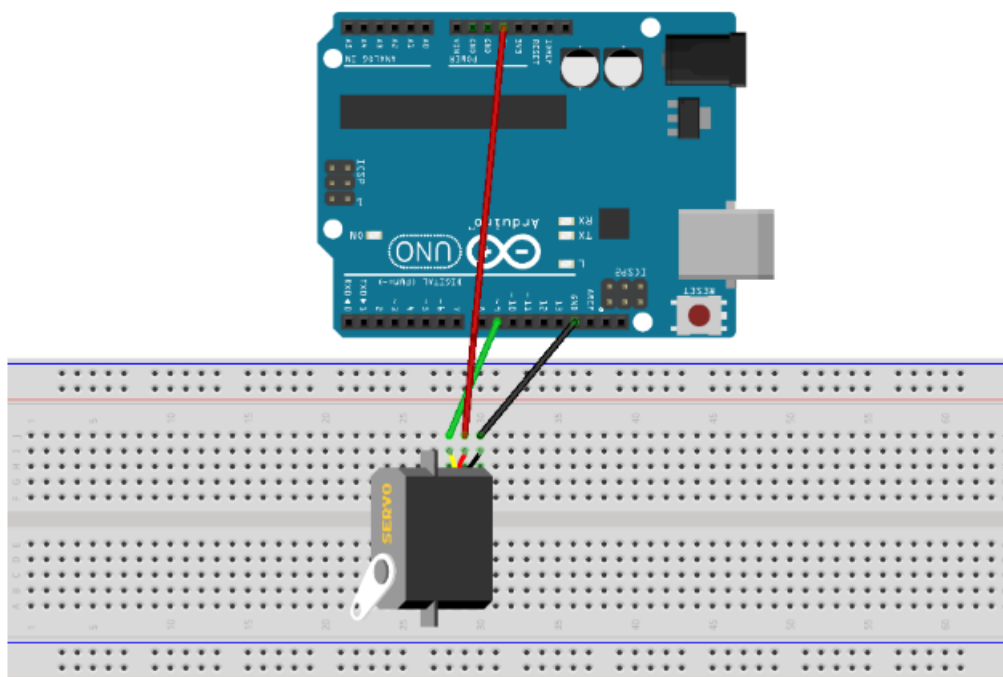
Kích thước	40.7 x 19.7 x 42.9 mm
Điện áp làm việc	4.8 ~ 7.2 V
Tốc độ xoay	0.17 giây / 60 độ (4.8 v) 0.14 giây / 60 độ (6 v)
Lực kéo	9.4kg / cm (4.8V) 11kg / cm (6V)
Nhiệt độ hoạt động	0 °C - 55 °C
Trọng lượng	55 g

Sơ đồ chân của động cơ Servo MG996R



Hình 3. 8. Sơ đồ chân của động cơ giảm tốc MG996R

c) Sơ đồ kết nối Arduino và động cơ Servo.



Hình 3. 9. Sơ đồ kết nối Arduino và động cơ Servo

Thiết kế hệ thống tay gạt và hộp chứa vật phẩm.

- **Thanh gạt:** Một thanh nhựa có chiều dài 17 cm, một đầu có lỗ để bắt vít vào đầu của động cơ servo để làm thanh gạt.



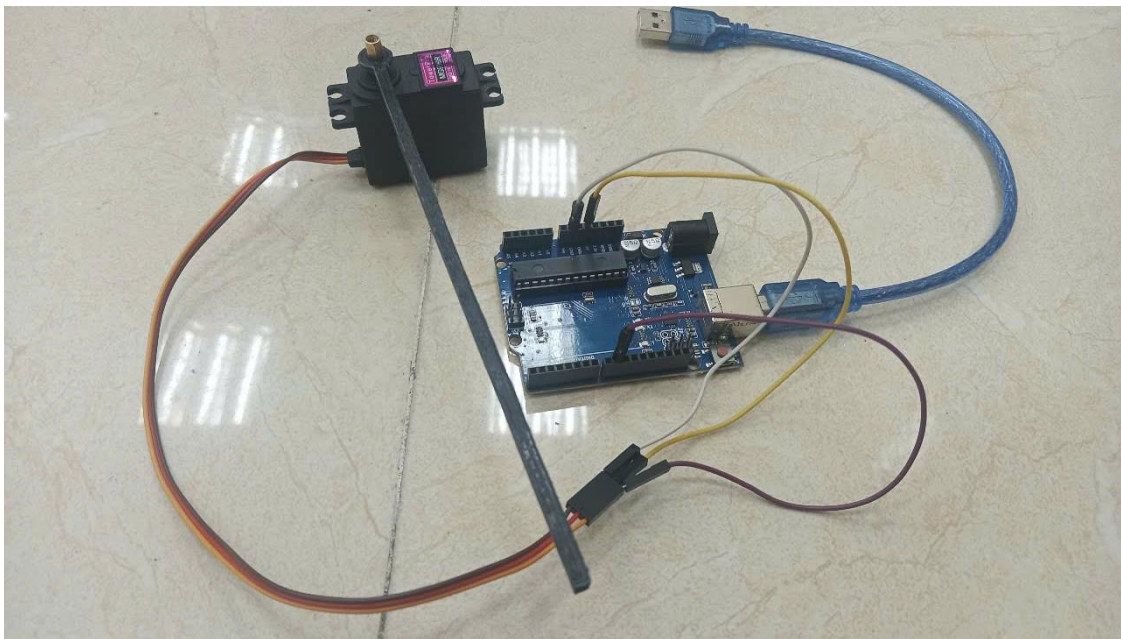
Hình 3. 10. Thanh gạt

- **Hộp chứa vật phẩm:** Hộp nhựa hình hộp chữ nhật có chiều dài mặt đáy x cm, chiều rộng mặt đáy y cm, chiều cao h cm. Gồm có hai hộp, một hộp được đặt bên đối diện với bên đặt cơ cấu tay gạt được chỉ định là hộp chứa vật phẩm hình chữ nhật, hộp còn lại đặt ở cuối băng truyền được chỉ định là hộp chứa vật phẩm hình tròn.



Hình 3. 11. Hộp đựng vật phẩm

Hoàn thiện hệ thống tay gạt





Hình 3. 12. Hệ thống tay gạt

3.2.3. Hệ thống băng chuyền.

Hệ thống băng chuyền bao gồm: khung băng truyền, băng tải, động cơ DC và con lăn.

Bảng 3. 4. Các bộ phận cấu tạo băng truyền

1	Khung băng truyền	
2	Băng tải	
3	Trục lăn	
4	Động cơ GB37-520 24V22RPM	

Động cơ DC được sử dụng là động cơ giảm tốc GB37-520 24V22RPM, có cấu tạo bằng kim loại cho độ bền và độ ổn định cao. Hộp giảm tốc của động cơ có nhiều tỉ số truyền để có thể lựa chọn giữa lực kéo và tốc độ (lực kéo càng lớn thì tốc độ càng chậm và ngược lại).

Bảng 3. 5. Thông số kỹ thuật của động cơ giảm tốc GB37-520 24V22RPM

Điện áp cấp	24 Vdc
Tốc độ quay	22 rpm
Đường kính trục	6 mm
Chiều dài động cơ	41 mm
Đường kính động cơ	37 mm
Chiều dài trục	12 mm

Động cơ được cấp trực tiếp nguồn 24 V từ pin cấp nguồn để có thể hoạt động bình thường.

Hệ thống băng truyền hoàn thiện:



Hình 3. 13. Băng chuyền



3.2.4. Cảm biến chiều sâu

Sử dụng cảm biến chiều sâu Visionary-T CX V3S100-2X để phân loại các vật phẩm. Các thông tin về cảm biến đã được đưa ra trong phần 2.1.4 của chương 2.



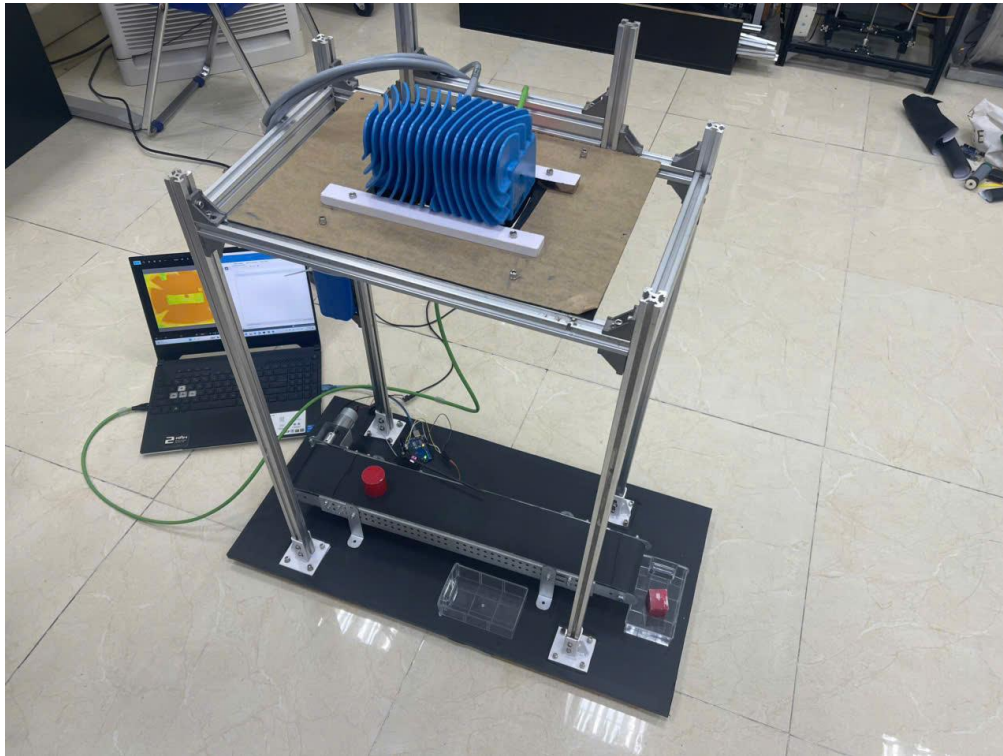
Hình 3.14. Cảm biến chiều sâu Visionary-T CX V3S100-2X

Bảng 3. 6. Các bộ phận kết nối và cấp nguồn của cảm biến

1	Dây nối nguồn 24 V	
2	Dây cổng lan kết nối với máy tính	
3	Pin cấp nguồn 24 V	

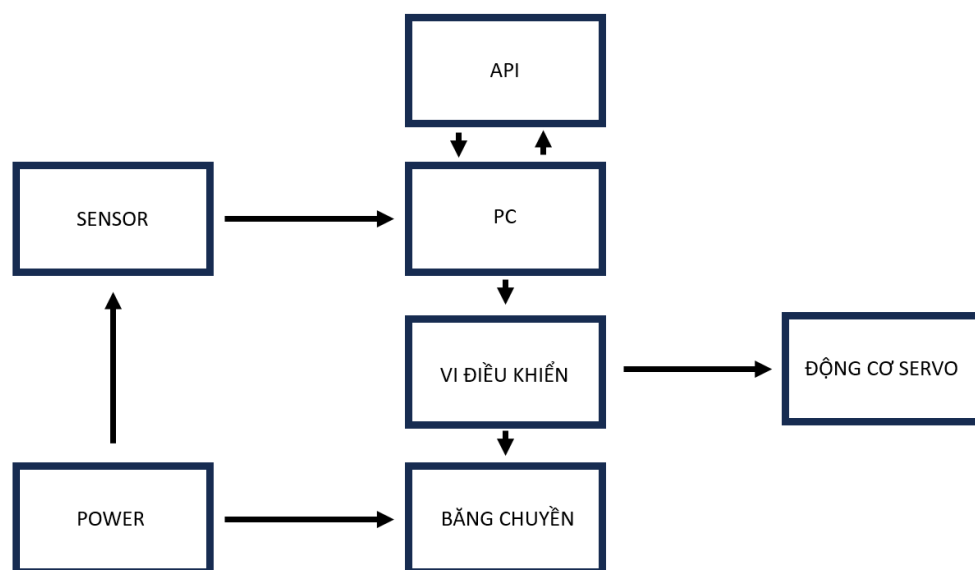
3.2.5 Hoàn thiện mô hình.

Sau khi hoàn thiện các phần khung giá đỡ, hệ thống phân loại và băng truyền, tiếp tục tiến hành lắp ráp các phần lại với nhau, sau đó đặt cảm biến lên mặt giá đỡ để hoàn thiện mô hình cơ khí.



Hình 3. 15. Mô hình hoàn thiện

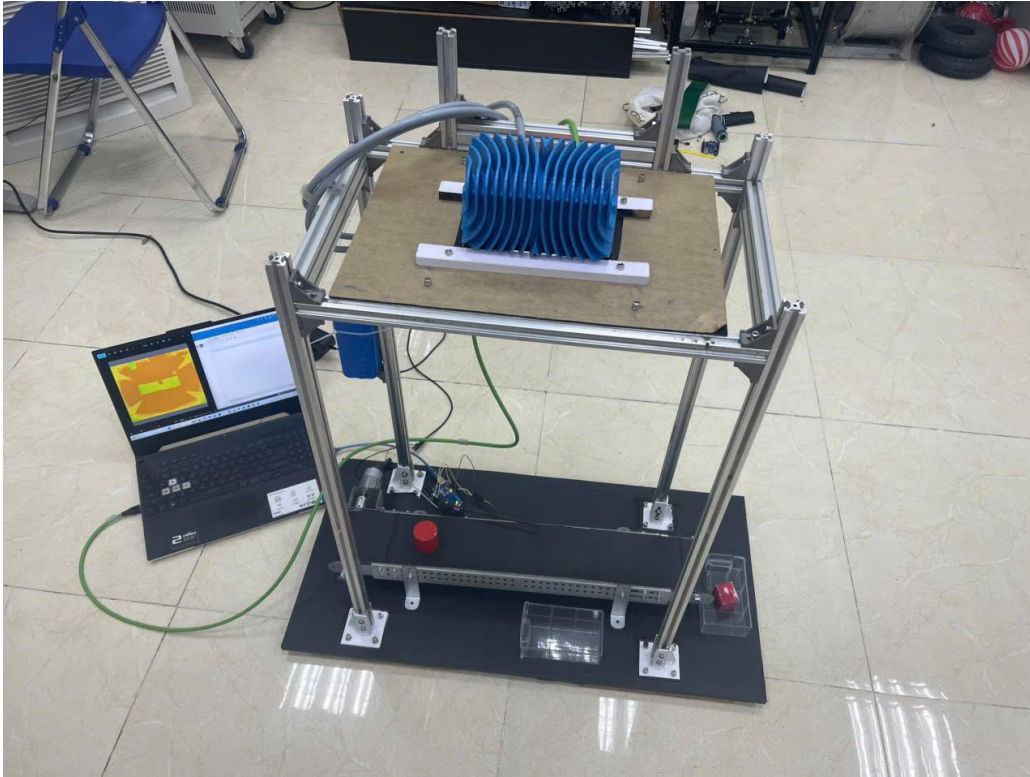
Mô hình hệ thống:



Hình 3. 16. Sơ đồ khối mô hình hóa hệ thống

3.3 Thử nghiệm hệ thống

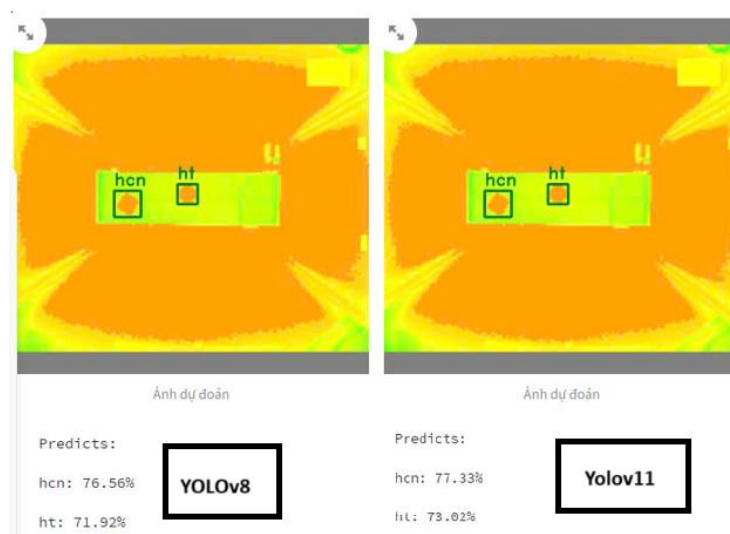
Kịch bản thử nghiệm: Cho 2 khối hình trụ và hình chữ nhật làm 2 sản phẩm cần phân loại. Sử dụng model đã được train bởi 2 phiên bản YOLOv8 và YOLOv11 để phân loại.



Hình 3. 17. Tiến hành thử nghiệm

3.4 Kết quả và đánh giá

Kết quả:



Hình 3. 18. Kết quả thử nghiệm YOLOv8 và YOLOv11.

Đánh giá:

Đồ án đã hoàn thành với đúng mục tiêu đề ra đó là khai thác dữ liệu từ cảm biến và ứng dụng nó vào mô hình sản xuất công nghiệp mô phỏng.

1. Mục tiêu đã đáp ứng:

- Sử dụng dữ liệu ảnh Intensity từ cảm biến Visionary-T để phục vụ bài toán Object Detection.
- Sự hiệu quả của ứng dụng, API trong dây chuyền sản xuất dễ dàng theo dõi, quản lý.
- Có thể thấy rõ được sự khác biệt giữa các phiên bản Yolo cũ và mới đều có những sự khác nhau về các ưu điểm.

2. Hạn chế:

- Trong sản xuất ngày nay còn đòi hỏi thêm rất nhiều yếu tố khác như màu sắc, đọc nhãn, ... dẫn đến cảm biến này không thể đáp ứng đầy đủ.
- Tốc độ phân loại chậm cơ cấu tay gạt là không đủ để đáp ứng tốt trong dây truyền sản xuất nhanh.

3. Từ các mục tiêu đã đáp ứng được và các hạn chế trên, đồ án sẽ còn có các phương án phát triển sau:

- Tích hợp thêm Camera RGBD nhằm đáp ứng nhận diện các yếu tố màu sắc, nhãn hàng, ...
- Thay đổi bộ phận cơ chấp hành từ tay gạt sang cơ cấu đẩy dùng pistol khí nén, hoặc sử dụng tay gấp robot,...

CHƯƠNG IV: KẾT LUẬN

Qua quá trình tìm hiểu và thực hiện đồ án đề tài ***“Nghiên cứu ứng dụng cảm biến chiều sâu phân loại sản phẩm trong sản xuất công nghiệp”*** với sự cố gắng của nhóm và sự hướng dẫn nhiệt tình của thầy **Ths.Bùi Huy Anh**, đề tài đã hoàn thành đúng thời gian yêu cầu đề ra.

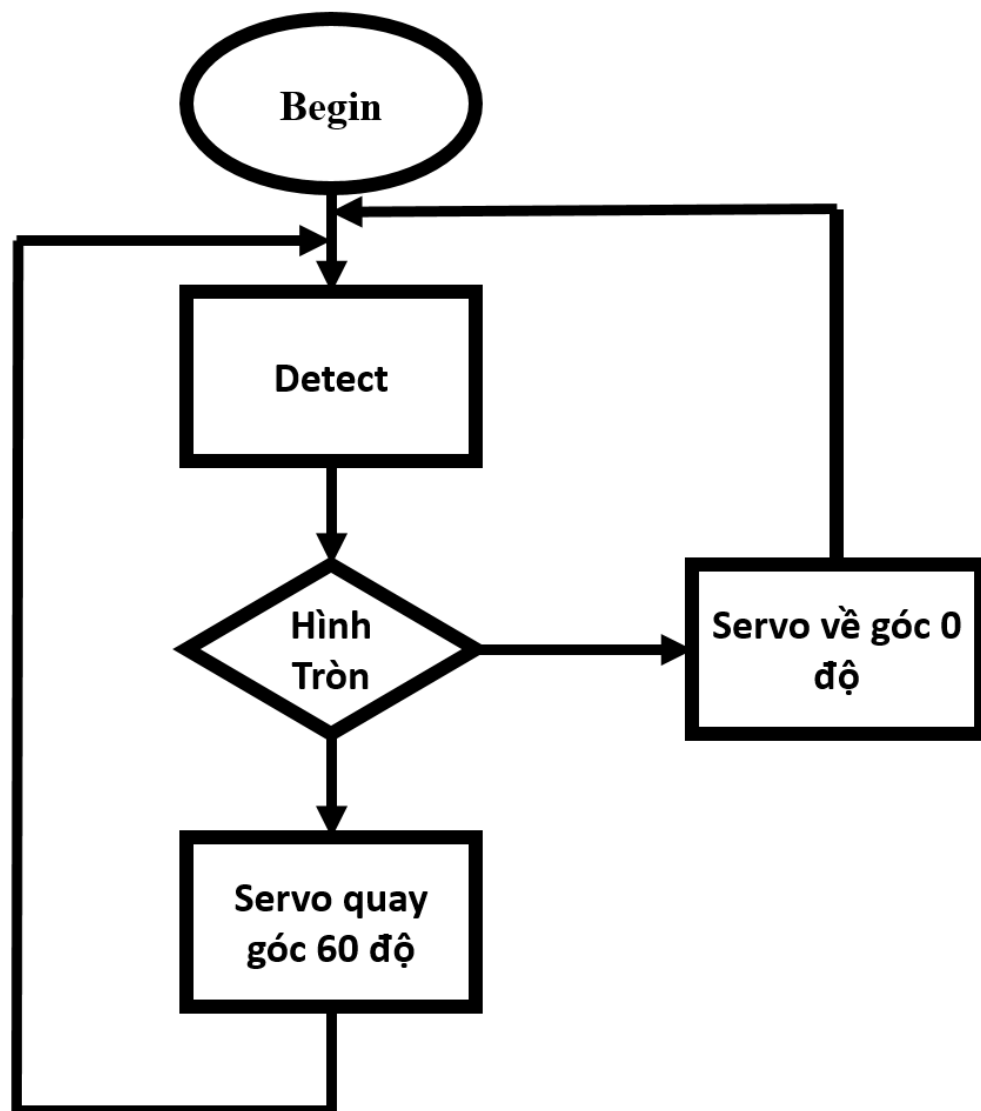
Song song với quá trình thực hiện, nhóm báo cáo cũng gặp một số hạn chế nhất định và các vấn đề đề cần khắc phục trong báo cáo đồ án. Vì vậy những ý kiến nhóm nghiên cứu sẽ tích cực, nghiêm túc tiếp thu những ý kiến đóng góp quý báu của thầy cô để tiếp tục phát triển.

Qua đây, nhóm nghiên cứu cũng xin được gửi lời cảm ơn đến các thầy cô trong Trường Cơ Khí – Ô Tô đã nhiệt tình hướng dẫn cho nhóm để hoàn thành tốt đồ án môn này. Một lần nữa nhóm nghiên cứu xin chân thành cảm ơn các thầy cô đã hỗ trợ nhóm trong thời gian học vừa qua.

TÀI LIỆU THAM KHẢO

- [1] <https://www.sick.com/media/docs>
- [2] [YOLOv8 - Ultralytics YOLO Tài liệu](#)
- [3] [Build Vision Models with Roboflow | Roboflow Docs](#)
- [4] [\[2211.12851\] A Streamlit-based Artificial Intelligence Trust Platform for Next-Generation Wireless Networks](#)
- [5] [\[1404.7828\] Deep Learning in Neural Networks: An Overview](#)

PHỤ LỤC 1: LƯU ĐỒ THUẬT TOÁN



PHỤ LỤC 2: CODE PYTHON VÀ VI ĐIỀU KHIỂN

```
import streamlit as st
from streamlit_option_menu import option_menu
import numpy as np
from io import BytesIO
import trimesh
import pandas as pd
from fonts import get_multiple_fonts_css
from train import test_image_model, train_model, load_stl, laptop_info,
preprocess_image, load_screen
import time
import cv2
import serial
import time
# CSS
st.markdown(get_multiple_fonts_css(), unsafe_allow_html=True)
st.markdown(
    """
    <style>
    /* Căn giữa các tab */
    [data-baseweb="tab-list"] {
        display: flex;
        justify-content: center;
    }

    [data-baseweb="tab"] {
        color: #2D32501;
    }

    /* Căn giữa caption */
    .caption {
        text-align: center;
        font-size: 16px;
        color: #5D6D7E;
        margin-top: 10px;
    }
    </style>
    """,
    unsafe_allow_html=True
)

# Sidebar
sidebar_1 = "Validation Results"
```

```

sidebar_2 = "Visionary-T"
sidebar_3 = "Test Model"
sidebar_4 = "STL file"
sidebar_5 = "Train Model"
with st.sidebar:
    selected = option_menu(
        "Menu",
        [sidebar_2, sidebar_4, sidebar_5, sidebar_1, sidebar_3],
        icons=['1-circle-fill', '2-circle-fill', '3-circle-fill', '4-circle-fill', '5-circle-fill'],
        menu_icon="app", default_index=0,
        styles={
            "container": {"padding": "0!important", "background-color": "#2D3250"},
            "icon": {"color": "#D4BDAC", "font-size": "16px"},
            "nav-link": {
                "font-size": "18px",
                "font-family": "Pacifico, cursive",
                "color": "#758694",
                "text-align": "left",
                "margin": "0px",
                "--hover-color": "#424769"
            },
            "nav-link-selected": {
                "background-color": "#7077A1",
                "color": "#F5E8C7",
                "font-weight": "bold"
            },
            "menu-title": {
                "font-size": "20px",
                "font-family": "Roboto, sans-serif",
                "color": "#F5E8C7",
                "text-align": "left",
                "font-weight": "bold",
                "padding": "8px",
            }
        }
    )

# Visionary-T Tab
if selected == sidebar_2:
    st.header("Visionary-T")

```

```

# st.markdown('<p class="handwriting">Visionary-T</p>',
unsafe_allow_html=True)
st.image("sick1.png")
st.text("Là 1 máy ảnh 3D hoạt động dựa trên nguyên tắc time-of-flight (ToF)")
st.text("Cung cấp dữ liệu 3D thời gian thực với tốc độ 50 fps (frames per second)")
st.text("Cấu tạo")
st.image("cautao_vst.png")

# Results Tab
if selected == sidebar_1:
    progress_text = "Operation in progress. Please wait."
    my_bar = st.progress(0, text=progress_text)

    for percent_complete in range(50):
        time.sleep(0.01)
        my_bar.progress(percent_complete + 1, text=progress_text)
    my_bar.empty()
    st.header("Results")
    # st.markdown('<p class="handwriting">Results</p>',
unsafe_allow_html=True)
    val_img = "runs/detect/train29/predicts.jpg"
    conf_matrix = "runs/detect/train29/conf_matrix.jpg"
    loss_img = "runs/detect/train29/loss_plot.png"

    tab1, tab2, tab3 = st.tabs(["Predicts", "Loss", "Confusion Matrix"])
    with tab1:
        st.image(val_img, width=800)
        st.markdown('<p class="caption">Kết quả thu được</p>',
unsafe_allow_html=True)
    with tab2:
        st.image(loss_img, width=1000)
        st.markdown('<p class="caption">Biểu đồ Loss sau mỗi epoch</p>',
unsafe_allow_html=True)
    with tab3:
        st.image(conf_matrix, width=1000)
        st.markdown('<p class="caption">Confusion Matrix</p>',
unsafe_allow_html=True)
    char_data = pd.DataFrame(np.random.rand(20, 3), columns=["A", "B", "C"])

# Test Model Tab

```

```

if selected == sidebar_3:
    # st.markdown('<p class="handwriting">YOLOv11</p>',
    unsafe_allow_html=True) # Đợi 2 giây để Arduino reset
    ser = serial.Serial('COM6', 9600) # Windows: COM6, Linux: /dev/ttyUSB0
    time.sleep(2) # Đợi 2 giây để Arduino reset
    st.header("YOLOv11")
    uploaded_file = st.file_uploader("Tải lên ảnh của bạn", type=["jpg", "png",
"jpeg"])
    col1, col2 = st.columns(2)
    with col1:
        button = st.button("Live")
    if button:
        with col2:
            button = st.button("Stop")
            frame_placeholder = st.empty()
            while True:
                frame = load_screen()
                start = time.time()
                results, conf, label, cnt = test_image_model(frame)
                end = time.time()
                totalTime = end - start
                fps = 1 / totalTime
                fps_disp = f"FPS: {fps:.2f}"
                results_fps = cv2.putText(frame, fps_disp, (10, 25),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
                frame_placeholder.image(results_fps, use_column_width=True)
                if ('ht' in label):
                    ser.write(b'1')
                elif ('hcn' in label):
                    ser.write(b'0')
            if button:
                break

if uploaded_file is not None:
    col1, col2 = st.columns(2)
    image = preprocess_image(uploaded_file)
    with col1:
        st.image(image, caption="Ảnh gốc", use_column_width=True)
        results, conf, label, cnt = test_image_model(image)
        st.text("Predicts:")
    with col2:
        st.image(results, caption="Ảnh dự đoán", use_column_width=True)
    for i in range(cnt):

```

```

        st.text(f"{label[i]}: {(conf[i]*100):.2f}%")
    st.toast("Done!", icon='😊')

# STL File Tab
if selected == sidebar_4:
    # st.markdown('<p class="handwriting">3D STL Model Viewer</p>',
    unsafe_allow_html=True)
    st.header("3D STL Model Viewer")
    tab_stl1, tab_stl2, tab_stl3 = st.tabs(["Miếng vuông", "Con trượt 1", "Con trượt
2"])
    with tab_stl1:
        stl_mesh1 = trimesh.load('miengvuongg4lo.STL', file_type='stl')
        fig1 = load_stl(stl_mesh1)
        st.plotly_chart(fig1, use_container_width=True, key='1')
    with tab_stl2:
        stl_mesh2 = trimesh.load('contruotnhua.STL', file_type='stl')
        fig2 = load_stl(stl_mesh2)
        st.plotly_chart(fig2, use_container_width=True, key='2')
    with tab_stl3:
        stl_mesh3 = trimesh.load('vithinhthang.STL', file_type='stl')
        fig3 = load_stl(stl_mesh3)
        st.plotly_chart(fig3, use_container_width=True, key='3')
    uploaded_file = st.file_uploader("Tải file STL của bạn", type=["stl"])
    if uploaded_file is not None:
        stl_mesh = trimesh.load(BytesIO(uploaded_file.read()), file_type='stl')
        fig = load_stl(stl_mesh)
        st.plotly_chart(fig, use_container_width=True)

# Train Model Tab
if selected == sidebar_5:
    progress_text = "Operation in progress. Please wait."
    my_bar = st.progress(0, text=progress_text)

    for percent_complete in range(50):
        time.sleep(0.01)
        my_bar.progress(percent_complete + 1, text=progress_text)
    my_bar.empty()
    st.toast(f"Cấu hình sử dụng: {laptop_info()}", icon='😊')
    # st.markdown('<p class="handwriting">Training Model</p>',
    unsafe_allow_html=True)
    st.header("Training Model")
    epochs = st.text_input("Nhập Epochs:", placeholder="Số lần duyệt qua toàn
bộ dữ liệu")
    batch_size = st.text_input("Nhập Batch size:",

```

placeholder=" Số mẫu được truyền qua mạng trong mỗi lần
cập nhật tham số ")

```
if st.button("Xác nhận"):
    if epochs.strip() and batch_size.strip():
        try:
            epochs = int(epochs)
            batch_size = int(batch_size)
            if epochs > 0 and batch_size > 0:
                train_model(epochs=epochs, batch=batch_size)
                st.success(f"T: {batch_size}")
            else:
                st.error("Giá trị Epochs và Batch size phải lớn hơn 0.")
        except ValueError:
            st.error("Vui lòng nhập một số nguyên hợp lệ. VD: Epochs = 10;  
Batch size = 16")
        else:
            st.warning("Vui lòng nhập giá trị!")
```

Sidebar footer

```
st.sidebar.text("@author: phamduyaaaa")
```

##Code Arduino:

```
#include <Servo.h> // Thêm thư viện Servo
```

```
Servo myServo; // Tạo đối tượng Servo
```

```
void setup() {
```

```
    // Khởi tạo giao tiếp nối tiếp với tốc độ 9600 bps
```

```
    Serial.begin(9600); // Cấu hình tốc độ truyền dữ liệu (9600 bps)
```

```
    // Kết nối servo với chân PWM 9
```

```
    myServo.attach(9);
```

```
    // Quay servo về vị trí 120 độ ban đầu
```

```
    // In thông báo khi servo ở 120 độ
```

```
}
```

```
void loop() {
```

```
    // Kiểm tra xem có dữ liệu gửi từ máy tính không
```

```
    if (Serial.available() > 0) {
```

```
        int value = Serial.read(); // Đọc giá trị nhận được từ máy tính
```

```
// Kiểm tra giá trị nhận được và điều khiển servo
if (value == '1') {
  myServo.write(65); // Quay servo đến góc 65 độ
  Serial.println("Servo quay đến 65 độ"); // In ra thông báo
} else if (value == '0') {
  myServo.write(120); // Quay servo đến góc 120 độ
  Serial.println("Servo quay đến 120 độ"); // In ra thông báo
} else {
  Serial.println("Giá trị không hợp lệ, vui lòng gửi '0' hoặc '1'");
}
}
}
```

PHỤ LỤC 3: BẢN VẼ SOLIDWORKS BĂNG CHUYỀN VÀ KHUNG

