**TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH**

**KHOA ĐIỆN – ĐIỆN TỬ**

**BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG**

**HÀ TIẾN TÀI**

**ĐỖ VŨ GIA LONG**

**LUẬN VĂN TỐT NGHIỆP**

**ĐIỀU KHIỂN ROBOT SCARA TRONG ỨNG DỤNG CNC ROUTING VÀ GẮP VẬT ĐỘNG**

**KỸ SƯ NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA**

**TP. HỒ CHÍ MINH, 2021**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH**

**KHOA ĐIỆN – ĐIỆN TỬ**

**BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG**

**HÀ TIẾN TÀI – 1713022**

**ĐỖ VŨ GIA LONG – 1710174**

**LUẬN VĂN TỐT NGHIỆP**

**ĐIỀU KHIỂN ROBOT SCARA TRONG ỨNG DỤNG CNC ROUTING VÀ GẮP VẬT ĐỘNG**

**APPLICATION OF CONTROL ROBOT SCARA IN CNC ROUNTING AND PICKING AND PLACING MOVING OBJECTS**

**KỸ SƯ NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA**

**GIẢNG VIÊN HƯỚNG DẪN**

**TS.NGUYỄN VĨNH HẢO**

**TP. HỒ CHÍ MINH, 2018**

|  |  |
| --- | --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH  **KHOA ĐIỆN – ĐIỆN TỬ**  **BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG** | CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM  Độc lập - Tự do - Hạnh phúc |
|  | *TP. HCM, ngày….tháng…..năm……..* |

**NHẬN XÉT LUẬN VĂN TỐT NGHIỆP**

**CỦA CÁN BỘ HƯỚNG DẪN**

|  |  |  |
| --- | --- | --- |
| **Tên luận văn:** | | |
| **ĐIỀU KHIỂN ROBOT SCARA TRONG ỨNG DỤNG CNC ROUTING VÀ GẮP VẬT ĐỘNG** | | |
| **Nhóm Sinh viên thực hiện:** | | **Cán bộ hướng dẫn:** |
| Hà Tiến Tài | 1713002 | TS. Nguyễn Vĩnh Hảo |
| Đỗ Vũ Gia Long | 1710174 | TS. Nguyễn Vĩnh Hảo |
| **Đánh giá Luận văn**   1. Về cuốn báo cáo:   Số trang Số chương  Số bảng số liệu Số hình vẽ  Số tài liệu tham khảo Sản phẩm  Một số nhận xét về hình thức cuốn báo cáo:           1. Về nội dung luận văn:            1. Về tính ứng dụng:            1. Về thái độ làm việc của sinh viên:           **Đánh giá chung:** Luận văn đạt/không đạt yêu cầu của một luận văn tốt nghiệp kỹ sư, xếp loại Giỏi/ Khá/ Trung bình  **Điểm từng sinh viên:**  <Tên sinh viên 1**>:………../10**  <Tên sinh viên 2**>:………../10** | | |
|  | | |

|  |  |
| --- | --- |
|  | **Cán bộ hướng dẫn**  (Ký tên và ghi rõ họ tên) |

|  |  |
| --- | --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH  **KHOA ĐIỆN – ĐIỆN TỬ**  **BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG** | CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM  Độc lập - Tự do - Hạnh phúc |
|  | *TP. HCM, ngày….tháng…..năm……..* |

**NHẬN XÉT LUẬN VĂN TỐT NGHIỆP**

**CỦA CÁN BỘ PHẢN BIỆN**

|  |  |  |
| --- | --- | --- |
| **Tên luận văn:** | | |
| **ĐIỀU KHIỂN ROBOT SCARA TRONG ỨNG DỤNG CNC ROUTING VÀ GẮP VẬT ĐỘNG** | | |
| **Nhóm Sinh viên thực hiện:** | | **Cán bộ phản biện:** |
| Hà Tiến Tài | 1713002 | TS. |
| Đỗ Vũ Gia Long | 1710174 | TS. |
| **Đánh giá Luận văn**   1. Về cuốn báo cáo:   Số trang Số chương  Số bảng số liệu Số hình vẽ  Số tài liệu tham khảo Sản phẩm  Một số nhận xét về hình thức cuốn báo cáo:           1. Về nội dung luận văn:            1. Về tính ứng dụng:            1. Về thái độ làm việc của sinh viên:           **Đánh giá chung:** Luận văn đạt/không đạt yêu cầu của một luận văn tốt nghiệp kỹ sư, xếp loại Giỏi/ Khá/ Trung bình  **Điểm từng sinh viên:**  Hà Tiến Tài **:………../10**  Đỗ Vũ Gia Long **:………../10** | | |

|  |  |
| --- | --- |
|  | **Người nhận xét**  (Ký tên và ghi rõ họ tên) |

**LỜI CẢM ƠN**

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến quý thầy cô trường Đại học Bách Khoa TP.HCM nói chung, và bộ môn Điều khiển Tự động nói riêng, đã truyền dạy cho chúng em những kinh nghiệm và kiến thức quý báu trong suốt quãng thời gian học tập tại trường Đại học Bách Khoa.

Chúng em xin chân thành cảm ơn thầy Nguyễn Vĩnh Hảo – Trưởng bộ môn Điều khiển và Tự động hóa là giảng viên hướng dẫn của bọn em. Thầy đã tận tình chỉ dạy, hỗ trợ kiến thức và kinh nghiệm cho chúng em để có thể hoàn thành đề tài luận văn này. Ngoài ra còn có sự nhiệt tình hỗ trợ, góp ý chân thành của thầy Trần Quốc Tiến Dũng và thầy Trần Hoàng Khôi Nguyên trong suốt thời gian bọn em thực hiện luận văn ở phòng thí nghiệm đã giúp đỡ bọn em nhận ra những thiếu sót, những bài học thực tiễn quý giá. Có được sự gặp gỡ và dẫn dắt của các thầy là một sự may mắn lớn trong con đường học tập và sự nghiệp của chúng em, đặc biệt là trong thời điểm đại dịch Covid 19 vô cùng khó khăn.

Cảm ơn gia đình, người thân đã quan tâm, chăm lo cho chúng con, tạo điều kiện giúp chúng con được ăn học. Cảm ơn cha mẹ đã dạy dỗ, nuôi nấng chúng con khôn lớn nên người.

Và cuối cùng, gửi lời cảm ơn đến những bạn bè, anh chị trong phòng thí nghiệm đã hỗ trợ, giúp đỡ nhau tận tinh, chia sẻ nhau những kinh nghiệm bổ ích trong thời gian học tập và làm việc tại phòng thí nghiệm.

TPHCM, ngày…tháng…năm

**Sinh viên**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH  **KHOA ĐIỆN – ĐIỆN TỬ**  **BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG** | CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM  Độc lập - Tự do - Hạnh phúc | |  | *TP. HCM, ngày….tháng…..năm……..* | |  |

**ĐỀ CƯƠNG CHI TIẾT**

|  |  |
| --- | --- |
| **TÊN LUẬN VĂN:** ĐIỀU KHIỂN ROBOT SCARA TRONG ỨNG DỤNG CNC ROUTING VÀ GẮP VẬT ĐỘNG | |
| **Cán bộ hướng dẫn:** TS. Nguyễn Vĩnh Hảo | |
| **Thời gian thực hiện:** Từ ngày 22/2/2021 đến ngày…………… | |
| **Sinh viên thực hiện:**  Hà Tiến Tài – 1713002  Đỗ Vũ Gia Long – 1710174 | |
| **Nội dung đề tài:**  *Mục tiêu:*  Nghiên cứu cách thức vận hành và thực hiện mô hình robot SCARA. Cùng với đó là kết hợp với thị giác máy tính để thực hiện ứng dụng gắp và phân loại vật chuyển động trên băng truyền. Bên cạnh đó, ứng dụng SCARA cho hoạt động CNC rounting. Thiết kế phần mềm điều khiển và hiện thị trên máy tính để dễ dàng cho người sử dụng cũng như cho nghiên cứu, học tập.  *Phương pháp thực hiện:*   * Nghiên cứu về cơ khí, lập trình, thiết kế của hệ thống robot, thiết kế băng truyền, đầu công tác. * Nghiên cứu các giải thuật cho CNC rounting. * Nghiên cứu giải thuật để xử lý ảnh RGB, phân loại và gắp vật thể trên băng truyền.   *Kết quả mong đợi:*   * Xây dựng thành công bộ thư viện xử lý ảnh, giải thuật gắp vật động và giải thuật CNC rounting. * Lập trình hệ thống và giao diện người dùng trực quan, dễ sử dụng. * Thực hiện thành công gắp và phân loại vật thể trên băng truyền động. * Cho ra kết quả CNC rounting đạt yêu cầu. | |
| **Kế hoạch thực hiện** | |
| **Hà Tiến Tài** | **Đỗ Vũ Gia Long** |
| **Giai đoạn 1: Nghiên cứu về mô hình SCARA**   * Xác định các thông số, kết cấu cơ khí của robot SCARA và bộ điều khiển từ các tài liệu đã có. * Kiểm tra thực tế các kết cấu có sẵn của phần điện tử.   **Giai đoạn 2: Thiết lập các nền tảng cơ sở cho robot SCARA**   * Hiệu chỉnh các thông số PID để động cơ đáp ứng nhanh hơn. * Thiết kế một protocol giao tiếp giữa PC và robot một cách linh hoạt và hiệu quả. * Tìm hiểu lại về các mô hình path planning và trajectory planning.   **Giai đoạn 3: Xây dựng mô hình gắp vật động**   * Thiết kế và xây dựng mô hình băng chuyền và các slot. * Tinh chỉnh lại các model LSPB và Scurve để giảm thiểu tối đa thời gian tính toán. * Thiết lập thuật toán gắp nhiều vật với T constraint và V constraint.   **Giai đoạn 4: Tìm hiểu và xây dựng mô hình CNC routing**   * Thiết kế đầu tool kẹp bút cho robot SCARA. * Tìm hiểu về Gcode và các phần mềm Inkspace để tạo ra các file Gcode từ file cad được thiết kế. * Xây dựng thuật toán giải mã file Gcode trên ứng dụng điều khiển. * Thiết kế các model phân tách tại các điểm gấp khúc của quỹ đạo file Gcode.   **Giai đoạn 5: Kết hợp với thị giác máy để triển khai thử nghiệm gắp vật**   * Nhận và xử lý các dữ liệu từ camera gửi xuống cho robot SCARA * Tinh chỉnh và giảm thiếu tối đa các delay trong việc xác định vị trí vật * Thử nghiệm với các vận tốc khác nhau   **Giai đoạn 6: Kiểm nhiệm lại model CNC routing từ file Gcode**   * Sửa các lỗi xảy ra với các thuật toán đã thiết kế ở giai đoạn 4. * Nhận xét và thử nghiệm với nhiều hình vẽ phức tạp.   **Giai đoạn 7: Kiểm tra và đánh giá kết quả**   * Đánh giá sự hiểu quả của các model di chuyển của robot SCARA. * Đánh giá các sai số của các model đã thiết kế. | **Giai đoạn 1: Lắp đặt cố định và các bước hiệu chuẩn camera**   * Lắp đặt giàn khung cho camera. * Thu thập các dữ liệu hiệu chuẩn, thu thập các thông số nội, ngoại.   **Giai đoạn 2: Nghiên cứu và chọn phương pháp nhận diện đối tượng**   * Tìm hiểu về CNN, các mô hình học sâu. * Tìm hiểu về các phương pháp nhận diện khác nhau: YOLO, SSD, … * Thu thập dữ liệu cho bộ dữ liệu. * Nghiên cứu và lựa chọn mô hình YOLO thích hợp.   **Giai đoạn 3: Ứng dụng mô hình YOLOv3 tiny PRN vào việc nhận diện**   * Xây dựng, sắp xếp bộ dữ liệu cùng với bước Data Augmentation. * Huấn luyện các mô hình khác nhau trên tập dữ liệu đã xây dựng. * Lựa chọn mô hình tốt nhất. * Áp dụng các phương pháp nhằm tăng tốc độ của mô hình sao cho ít ảnh hưởng đến độ chính xác.   **Giai đoạn 4: Xây dụng thuật toán thị giác máy xác định các thông tin 2D của đối tượng**   * Xác định tâm của đối tượng. * Xác định góc xoay của đối tượng.   **Giai đoạn 5: Chuyển các thông tin cho robot**   * Chuyển đổi hệ tọa độ camera sang robot. * Xây dựng thuật toán gửi.   **Giai đoạn 6: Quan sát kết quả và cải thiện thuật toán**   * Nghiên cứu và sử dụng thuật toán bộ lọc Kalman để lọc nhiễu và tracking vật thể. * Hiệu chỉnh các thuật toán cũng như mô hình nhận diện đối tượng để đạt hiệu suất cao.   **Giai đoạn 7: Kết hợp và hoàn chỉnh giao diện người dùng**   * Kết hợp các giao diện với nhau, hoàn thiện giao diện phần mềm. * Thu thập kết quả và viết báo cáo. |
| **Xác nhận của Cán bộ hướng dẫn**  (Ký tên và ghi rõ họ tên) | TP. HCM, ngày….tháng …..năm…..  **Sinh viên**  (Ký tên và ghi rõ họ tên) |

DANH SÁCH HỘI ĐỒNG BẢO VỆ LUẬN VĂN

Hội đồng chấm luận văn tốt nghiệp, thành lập theo Quyết định số …………………… ngày ………………….. của Hiệu trưởng Trường Đại học Bách khoa TP.HCM.

* 1. …………………………………………. – Chủ tịch.
  2. …………………………………………. – Thư ký.
  3. …………………………………………. – Ủy viên.
  4. …………………………………………. – Ủy viên.
  5. …………………………………………. – Ủy viên.

**MỤC LỤC**

[Chương 0. TÓM TẮT LUẬN VĂN 1](#_Toc73348189)

[Chương 1. GIỚI THIỆU ĐỀ TÀI 2](#_Toc73348190)

[1.1. Giới thiệu 2](#_Toc73348191)

[1.1.1. Giới thiệu robot công nghiệp và thị giác máy 2](#_Toc73348192)

[1.1.2. Đặt vấn đề 3](#_Toc73348193)

[1.2. Mục tiêu luận văn 4](#_Toc73348194)

[1.3. Nhiệm vụ của luận văn 5](#_Toc73348195)

[Chương 2. TỔNG QUAN VÀ PHƯƠNG PHÁP 7](#_Toc73348196)

[2.1. LÝ THUYẾT ĐIỀU KHIỂN ROBOT 7](#_Toc73348197)

[2.1.1. Động học thuận robot 7](#_Toc73348198)

[2.1.2. Động học ngược robot SCARA 9](#_Toc73348199)

[2.1.3. Động học vận tốc 11](#_Toc73348200)

[2.1.4. Điểm kì dị của robot SCARA 12](#_Toc73348201)

[2.1.5. Các phương thức di chuyển của robot 13](#_Toc73348202)

[2.1.6. Hoạch định quỹ đạo 15](#_Toc73348203)

[2.1.7. Hoạch định đường đi trong không gian làm việc 21](#_Toc73348204)

[2.2. GIỚI THIỆU PHƯƠNG PHÁP NHẬN DIỆN ĐỐI TƯỢNG 22](#_Toc73348205)

[2.2.1. Lựa chọn phương pháp nhận diện đối tượng 22](#_Toc73348206)

[2.2.2. Tổng quát về mạng neural tích chập (CNN – Convolutional Neural Network) 23](#_Toc73348207)

[2.2.3. Tổng quan về mạng YOLO 25](#_Toc73348208)

[Chương 3. THIẾT KẾ VÀ THỰC HIỆN 34](#_Toc73348209)

[3.1. TỔNG QUAN VỀ PHÂN CỨNG CỦA HỆ THỐNG 35](#_Toc73348210)

[3.1.1. Camera 36](#_Toc73348211)

[3.1.2. Robot SCARA 36](#_Toc73348212)

[3.1.3. Mô hình băng chuyền 40](#_Toc73348213)

[3.2. CHI TIẾT GIẢI THUẬT CHO ROBOT SCARA 43](#_Toc73348214)

[3.2.1. Hoạch định các mô hình path planning và trajectory planning cho robot SCARA 43](#_Toc73348215)

[3.2.2. Thuật toán gắp nhiều vật trên băng chuyền 46](#_Toc73348216)

[3.3. CHI TIẾT GIẢI THUẬT THỊ GIÁC MÁY 50](#_Toc73348217)

[3.3.1. Sơ lược, tóm tắt về giải thuật thị giác máy 50](#_Toc73348218)

[3.3.2. Giới thiệu thư viện OpenCV 53](#_Toc73348219)

[3.3.3. Huấn luyện mô hình YOLO 53](#_Toc73348220)

[3.3.4. Thuật toán xử lý ảnh kinh điển 62](#_Toc73348221)

[3.3.5. Chuyển đổi sang hệ tọa độ robot 69](#_Toc73348222)

[3.3.6. Camera calibration 70](#_Toc73348223)

[3.3.7. Bộ lọc Kalman 71](#_Toc73348224)

[3.3.8. Thuật toán gửi 74](#_Toc73348225)

[3.4. SORFTWARE CỦA HỆ THỐNG 77](#_Toc73348226)

[3.4.1. Protocol giao tiếp giữa máy tính và robot Scara 77](#_Toc73348227)

[3.4.2. Giải mã file Gcode trong ứng dụng CNC routing 79](#_Toc73348228)

[3.4.3. Cấu trúc giao diện chương trình 86](#_Toc73348229)

[Chương 4. KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ 91](#_Toc73348230)

[4.1. Kết quả điều khiển robot. 91](#_Toc73348231)

[4.1.1. Kết quả điều khiển dùng model LSPB và Scurve 91](#_Toc73348232)

[4.1.2. Kết quả điều khiển với 3 phương thức di chuyển (movL, movC, movJ) 92](#_Toc73348233)

[4.1.3. Đánh giá độ chính xác của robot 95](#_Toc73348234)

[4.1.4. Đánh giá độ chính xác trong điều khiển tốc độ băng chuyền 99](#_Toc73348235)

[4.1.5. Kết quả CNC routing 101](#_Toc73348236)

[4.2. Kết quả xử lí ảnh 106](#_Toc73348237)

[4.2.1. Đánh giá mô hình nhận diện đối tượng 106](#_Toc73348238)

[4.2.2. Đánh giá độ chính xác trong xác định vị trí và góc xoay 108](#_Toc73348239)

[4.3. Kết quả gắp và xếp vật 111](#_Toc73348240)

[Chương 5. KẾT LUẬN 114](#_Toc73348241)

[5.1. Nhận xét chung về luận văn 114](#_Toc73348242)

[5.2. Hướng phát triền của luận văn: 114](#_Toc73348243)

[Chương 6. PHỤ LỤC 116](#_Toc73348244)

[6.1. Phương pháp Denevit-Hartenbeg (DH) 116](#_Toc73348245)

[6.2. Ma trận Jacobian 118](#_Toc73348246)

[6.3. Điểm kì dị (Singularities) 119](#_Toc73348247)

[6.4. Các kiểu lớp trong mạng CNN 120](#_Toc73348248)

[6.4.1. Convolutional Layer 120](#_Toc73348249)

[6.4.2. Pooling Layer 121](#_Toc73348250)

[6.4.3. Fully Connected Layer 122](#_Toc73348251)

[6.5. Hàm tính IOU 123](#_Toc73348252)

[Chương 7. TÀI LIỆU THAM KHẢO 124](#_Toc73348253)

**DANH MỤC HÌNH VẼ**

[Hình 1.1 Hệ thống robot kết hợp thị giác máy 3](#_Toc73348254)

[Hình 2.1 Hệ toạ độ được chọn theo phương pháp D-H 7](#_Toc73348255)

[Hình 2.2 Hệ toạ độ được nhìn từ trên xuống 9](#_Toc73348256)

[Hình 2.3 Hai trường hợp nghiệm động học ngược 10](#_Toc73348257)

[Hình 2.4 Quỹ đạo vận tốc hình thang 16](#_Toc73348258)

[Hình 2.5 Đồ thị quỹ đạo S-curve 19](#_Toc73348259)

[Hình 2.6 Ví dụ mô hình CNN Mô hình thuật toán YOLO 24](#_Toc73348260)

[Hình 2.7 Sơ đồ kiến trúc tổng quát của mô hình YOLO [1] 25](#_Toc73348261)

[Hình 2.8 Các layer trong mạng darknet-53 [2] 26](#_Toc73348262)

[Hình 2.9 Cách hoạt động của mô hình [3] 26](#_Toc73348263)

[Hình 2.10 Minh họa cho kích thước output của mạng YOLO [3] 27](#_Toc73348264)

[Hình 2.11 Hàm mất mát YOLO [3] 28](#_Toc73348265)

[Hình 2.12 Sự khác biệt giữa khối thường (trái) và khối Residual networks (phải) 30](#_Toc73348266)

[Hình 2.13 Kết nối một phần dư thừa (Partial residual connection) 31](#_Toc73348267)

[Hình 2.14 Tính chất 1 của PRN 32](#_Toc73348268)

[Hình 2.15 Tính chất 2 của PRN 32](#_Toc73348269)

[Hình 2.16 Tính chất 3 của PRN 33](#_Toc73348270)

[Hình 3.1 Sơ đồ kết nối tổng quát của hardware hệ thống 35](#_Toc73348271)

[Hình 3.2 Cấu trúc phần cứng của robot 36](#_Toc73348272)

[Hình 3.3 Cấu trúc board điều khiển 37](#_Toc73348273)

[Hình 3.4 Mô hình nối dây giao tiếp bộ nhớ ngoại giữa STM32 và CPLD 39](#_Toc73348274)

[Hình 3.5 Bánh răng 20-5mm 41](#_Toc73348275)

[Hình 3.6 Bản vẽ solidwork của băng truyền 42](#_Toc73348276)

[Hình 3.7 Kết quả thiết kế băng truyền 43](#_Toc73348277)

[Hình 3.8 Sơ đồ hoạch định tổng thể 44](#_Toc73348278)

[Hình 3.9 Các biên giới ràng buộc của model LSPB(QVT) 45](#_Toc73348279)

[Hình 3.10 Các biên giới ràng buộc của model Scurve(QVT) 46](#_Toc73348280)

[Hình 3.11 Object buffer khi nhận diện được 1 vật 47](#_Toc73348281)

[Hình 3.12 Object buffer khi nhận diện được 2 vật 47](#_Toc73348282)

[Hình 3.13 Mô hình gắp vật ràng buộc thời gian 48](#_Toc73348283)

[Hình 3.14 Mô hình gắp vật ràng buộc vận tốc 49](#_Toc73348284)

[Hình 3.15 Sơ đồ thuật toán thị giác máy tổng quát 51](#_Toc73348285)

[Hình 3.16 Kết quả thuật toán YOLO phát hiện đối tượng 52](#_Toc73348286)

[Hình 3.17 Ảnh do camera thu được 57](#_Toc73348287)

[Hình 3.18 Ảnh sau khi dùng Data Augmentation 58](#_Toc73348288)

[Hình 3.19 Gán nhãn cho đối tượng 59](#_Toc73348289)

[Hình 3.20 Thông số GPU trên GC 61](#_Toc73348290)

[Hình 3.21 Kết quả sau huấn luyện 62](#_Toc73348291)

[Hình 3.22 Lưu đồ các thuật toán xử lí ảnh kinh điển 63](#_Toc73348292)

[Hình 3.23 Ảnh hệ màu RGB (trái) và ảnh kênh Saturation (phải) 64](#_Toc73348293)

[Hình 3.24 Ảnh sau khi nhị phân hóa 66](#_Toc73348294)

[Hình 3.25 Ảnh sau khi sử dụng thuật toán Canny 66](#_Toc73348295)

[Hình 3.26 Ảnh của đối tượng được xác định viền bao 67](#_Toc73348296)

[Hình 3.27 Tâm của đối tượng 69](#_Toc73348297)

[Hình 3.28 Chuyển đổi hệ tọa độ camera sang robot 69](#_Toc73348298)

[Hình 3.29 Mô hình bộ lọc Kalman 72](#_Toc73348299)

[Hình 3.30 Sơ đồ thuật toán gửi 75](#_Toc73348300)

[Hình 3.31 Cấu trúc gói tin Transmit 77](#_Toc73348301)

[Hình 3.32 Cấu trúc gói tin Response 77](#_Toc73348302)

[Hình 3.33 Thời điểm gọi hàm xử lý 79](#_Toc73348303)

[Hình 3.34 Ví dụ cấu trúc file Gcode 79](#_Toc73348304)

[Hình 3.35 Cấu trúc tổng thể xử lý một file Gcode 80](#_Toc73348305)

[Hình 3.36 Cấu trúc xử lý các chuỗi dòng lệnh từ Gcode 81](#_Toc73348306)

[Hình 3.37 Mô hình xử lý phân tách các điểm 82](#_Toc73348307)

[Hình 3.38 Biến đổi rời rạc đối với đường thẳng 82](#_Toc73348308)

[Hình 3.39 Biến đổi rời rạc đối với cung tròn 83](#_Toc73348309)

[Hình 3.40 Ảnh hưởng của việc thay đổi hướng đột ngột 83](#_Toc73348310)

[Hình 3.41 File Gcode vẽ hình chữ nhật 84](#_Toc73348311)

[Hình 3.42 Các nhóm quy hoạch đã được phân tách 85](#_Toc73348312)

[Hình 3.43 Bổ sung cung tròn tiếp tuyến với hai đường thẳng 85](#_Toc73348313)

[Hình 3.44 Bố cục của giao diện chương trình 87](#_Toc73348314)

[Hình 4.1 Đồ thị quãng đường, vận tốc và gia tốc 92](#_Toc73348315)

[Hình 4.2 Biểu đồ kết quả điều khiển với lệnh movL 93](#_Toc73348316)

[Hình 4.3 Biểu đồ kết quả điều khiển với lệnh movC 94](#_Toc73348317)

[Hình 4.4 Biểu đồ kết quả điều khiển với lệnh movJ 95](#_Toc73348318)

[Hình 4.5 Đánh giá độ chính xác của robot 96](#_Toc73348319)

[Hình 4.6 Sử dụng cảm biến quang đo bằng truyền 100](#_Toc73348320)

[Hình 4.7 Quỹ đạo của phương pháp LSPB cho từng nhóm phân tách 102](#_Toc73348321)

[Hình 4.8 Quỹ đạo của phương pháp phân tách và bo tròn tại các góc cạnh 103](#_Toc73348322)

[Hình 4.9 Mẫu kết quả CNC viết chữ 104](#_Toc73348323)

[Hình 4.10 Mẫu kết quả CNC vẽ hình 105](#_Toc73348324)

[Hình 4.11 Mẫu đánh giá sai số vị trí và góc xoay 108](#_Toc73348325)

[Hình 6.1 Minh hoạ phương pháp Denevit-Hartenberg 116](#_Toc73348326)

[Hình 6.2 Phép Convolution 120](#_Toc73348327)

[Hình 6.3 Mô hình Convolutional Layer tổng quát 121](#_Toc73348328)

[Hình 6.4 . Pooling Layer (2\*2) 122](#_Toc73348329)

[Hình 6.5 Bước chuyển ma trận 3 chiều thanh vector 1 chiều 122](#_Toc73348330)

[Hình 6.6 Hàm IOU 123](#_Toc73348331)

**DANH MỤC BẢNG**

[Bảng 2.1 Bảng thông số D-H 8](#_Toc73348332)

[Bảng 3.1 Các thông số cấu hình mạng YOLOv3-tiny 54](#_Toc73348333)

[Bảng 4.1 Sai số vị trí của robot SCARA 96](#_Toc73348334)

[Bảng 4.2 Sai số góc roll của robot SCARA 98](#_Toc73348335)

[Bảng 4.3 Sai số tốc độ băng truyền 100](#_Toc73348336)

[Bảng 4.4 Thống kê các lần nhận diện 106](#_Toc73348337)

[Bảng 4.5 Thống kê sai số vị trí 109](#_Toc73348338)

[Bảng 4.6 Thống kê sai số góc xoay 110](#_Toc73348339)

[Bảng 4.7 Số liệu gắp từng vật trên băng truyền 111](#_Toc73348340)

[Bảng 4.8 Số liệu gắp nhiều vật trên băng truyền 112](#_Toc73348341)

[Bảng 6.1 Công thức tính ma trận Jacobian 119](#_Toc73348342)

**DANH MỤC TỪ VIẾT TẮT**

SCARA (Selective Compliance Assembly Robot Arm): một dạng cánh tay robot

CNC (Computer Numerical Control): một dạng máy được điều khiển tự động bằng máy tính

AI (Artificial Intelligence): Trí tuệ nhân tạo

NN (Neural Network): Mạng nơ-ron

CNN (Convolutional Neural Network): Mạng nơ-ron tích chập

FC (**Fully Connected Layer): Lớp kết nối toàn phần**

**FCN (Fully Connected Neural Network): Mạng nơ-ron kết nối toàn phần**

**ROI (Region Of Interest) : Vùng quan tâm**

# TÓM TẮT LUẬN VĂN

Với nền công nghiệp hiện đại ngày nay, robot và thị giác máy tính ngày càng được sử dụng rộng rãi trong nhiều ứng dụng khác nhau vì độ chính xác cao và tính hiệu quả mà nó mang lại. Trong đó, cánh tay robot SCARA là robot phổ biến được ứng dụng trong việc phân loại và đóng gói các sản phẩm tinh hoặc động một cách tự động. Kết hợp cùng với thị giác máy tính, cánh tay SCARA trở thành một công cụ hữu ích có thể thay thế con người, đồng thời tăng tốc độ sản xuất cũng như tăng hiệu quả làm việc. Chính vì vậy, ý tưởng của luận văn này là sử dụng những kiến thức đã được học để sử dụng robot SCARA kết hợp thị giác máy tính để thực hiện nhiệm vụ gắp vật động cũng như nhiệm vụ CNC rounting.

Luận văn được được thực hiện dựa trên mô hình gồm 3 phần là robot SCARA, camera và máy tính. Đối với ứng dụng gắp vật động, máy tính sẽ lấy thông tin từ camera sau đó sử dụng các thuật toán để tìm ra vị trí, góc xoay của vật,… Sau đó, các thông tin này được gửi quay robot để thực hiện các lệnh gắp và phân loại vật bỏ vào các khay có sẵn. Đối với ứng dụng CNC rounting, sau khi đưa file Gcode đã được chuyển đổi từ các file cad-cam vào máy tính, phần mềm sẽ tính toán và hoạch định quỹ đạo cần thiết để robot vẽ ra các đường vẽ mong muốn.

Việc học cách điều khiển robot và camera cũng như kiểm tra lỗi trên mô hình hệ thống cũng là một trong những khó khăn đối với người sử dụng. Vì lẽ đó, để giải quyết vấn đề này, luận văn cũng xây dựng phần mềm điều khiển giám sát dựa trên nền tảng và thư viện Qt, OpenGL, OpenCV,… phù hợp với người sử dụng, giúp dễ dàng thao tác, sử dụng.

Trong đề tài luận văn này, các mục tiêu về cơ bản đã được hoàn thành, tuy nhiên vẫn có những hạn chế cần phải khắc phục và phát triển thêm để ứng dụng cho thực tế. Các ưu, nhược điểm, cách thức thực hiện các mục tiêu,… sẽ được trình bày chi tiết trong nội dung của luận văn.

# GIỚI THIỆU ĐỀ TÀI

## Giới thiệu

### Giới thiệu robot công nghiệp và thị giác máy

**Robot công nghiệp**

Robot công nghiệp là một loại máy móc tự động hóa được sử dụng trong những nhà máy công nghiệp, hoạt động tốt trong những điều kiện khắc nghiệt cũng như yêu cầu kĩ thuật cao. Chúng được lập trình sẵn để di chuyển, cử động, làm việc với 2 hay nhiều trục liên kết với nhau. Robot công nghiệp bao gồm thành phần khác nhau như: phần cơ khí, bộ phận chấp hanh cuối, nguồn động lực, bộ điều khiển,…

Sử dụng robot công nghiệp trong sản xuất mang lại nhiều lợi ích như: nâng cao năng suất, tiết kiệm không gian làm việc, hạn chế sai sót,… Chính vì điều đó, hiện nay, robot được con người nghiên cứu, phát triển, nâng cao chất lượng với mục đích thay thế con người thực hiện những công việc nặng nhọc hay những công việc có độ nguy hiểm cao.

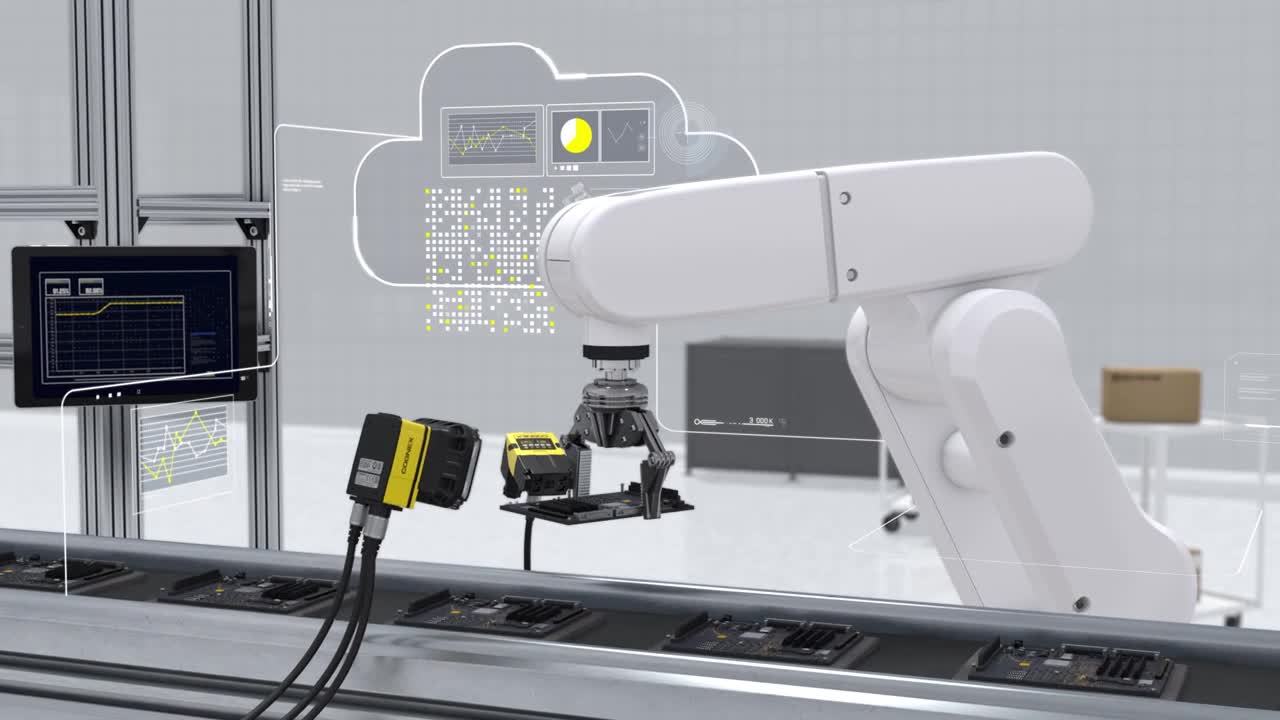
**Thị giác máy tính**

Thị giác máy tính là một thuật ngữ dùng để mô tả một lĩnh vực bao gồm nhiều phương pháp, công nghệ thu nhận, xử lý ảnh kỹ thuật số, phân tích, đánh giá, nhận dạng hình ảnh,… Mục đích chính của thị giác máy là tạo ra một hệ thống có thể sử dụng dữ liệu đa chiều từ thế giới thực để phân tích, xử lí cho ra các thông tin số hoặc một quyết định nào đó. Nhờ vào hệ thống này mà các thiết bị vô tri vô giác có khả năng “nhìn” và phản ứng với những thay đổi của thông tin mà chúng nhận được.

Thị giác máy khi được gắn với Trí tuệ nhân tạo (AI) tạo ra hệ thống giúp cho các thiết bị không chỉ phản ứng lại các thông tin thu được mà còn có khả năng phân tích, giải thích một cách phức tạp tương tự như con người với những yêu cầu cụ thể thông qua quá trình học tập trước đó.

**Hệ thống robot kết hợp với thị giác máy tính**

Cùng với sự hỗ trợ của thị giác máy tính, robot công nghiệp có thể thực hiện hàng loạt những tác vụ phức tạp. Từ đó, robot có thể thay thế con người để thực hiện các chuỗi hoạt động phức tạp. Các hệ thống này ngày càng được sử dụng một cách rộng rãi và cũng chính là xu hướng phát triển của công nghiệp nói riêng và cả thế giới nói chung.



Hình 1.1 Hệ thống robot kết hợp thị giác máy

### Đặt vấn đề

Hiện nay, hệ thống robot và thị giác máy ngày càng phát triển, ngày càng được cải tiến để nâng cao chất lượng nhằm thực hiện nhiều nhiệm vụ khác nhau. Chính vì vậy nên những yêu cầu của hệ thống ngày càng được đặt ra nhiều hơn, khắc khe hơn. Những yêu cầu mới mà mỗi hệ thống cần phải có là:

* Tốc độ di chuyển, thao tác của robot phải đáp ứng kịp tốc độ của đối tượng chạy trên băng truyền để đủ khả năng gắp vật di chuyển động.
* Hệ thống phải có khả năng phát hiện đối tượng trên băng truyền từ đó có thể xác định vị trí chính xác để gắp đối tượng, đồng thời giúp phân loại được các đối tượng khác nhau.
* Phải xác định được góc xoay của vật để có thể bỏ vào các vị trí cố định một cách chính xác.
* Đường đi của robot phải thật mượt, không bị gián đoạn để có thể thực hiện các tác vụ đòi hỏi tính liên tục trong ứng dụng CNC routing. Các góc cạnh của thay đổi đột ngột phải có phương pháp xử lí cụ thể để tránh hiện tượng jerk, dẫn đến sai đường hoạch định của file cad-cam …

Ngoài ra còn một số yêu cầu khác tùy theo mục tiêu và nhiệm vụ của từng hệ thống. Tuy nhiên, những yêu cầu trên cũng chính là những vấn đề mà nhóm em đã đặt ra để giải quyết. Hiện nay cũng có rất nhiều mô hình thực hiện rất tốt các việc này nhưng chi phí rất cao. Điều cần thiết là phải xây dụng được mô hình thực hiện được những điều trên nhưng với chi phí thấp. Vậy nên, với mục tiêu tìm hiểu kiến thức, nâng cao kĩ năng chuyên môn, nhóm sinh viên thực hiện luận văn quyết định chọn đề tài “Điều khiển robot SCARA trong ứng dụng CNC rounting và gắp vật động”.

## Mục tiêu luận văn

Từ những vấn đề đặt ra ở trên, luận văn sẽ hướng tới lần lượt các mục tiêu sau đây:

* Hệ thống camera, máy tính phải đảm bảo xác định được vị trí vật thể di chuyển động, cánh tay robot phải tiến hành gắp vật trên băng truyền và xếp đúng vị trí phân loại.
* Hệ thống phải xác định được hướng xoay vật thể trên mặt phẳng nằm ngang, cánh tay robot phải linh hoạt trong việc xoay gắp và đặt vật thể.
* Hệ thống robot phải di chuyển mượt, đều trên những quỹ đạo đã hoạch định trước thông qua máy tính.
* Xây dựng được giao diện giám sát hệ thống thân thiện với người dung bao gồm các thông tin từ camera, thông số của robot để tiện cho việc giám sát và theo dõi. Thuật toán giải mã file Gcode dễ sử dụng và hiệu quả

## Nhiệm vụ của luận văn

Dựa trên những mục tiêu trên, ta có thể chia luận văn thành 3 nhiệm vụ chính khác nhau, bao gồm:

**Đối với thị giác máy:**

* Xây dựng được mô hình học máy có thể phát hiện và phân loại đối tượng quan tâm với tốc độ realtime trên phần cứng chi phí thấp.
* Xây dựng được bộ thư viên có thể đọc vị trí và hướng xoay của đối tượng, từ đó chuyển đổi dữ liệu sang robot và gửi chúng đi.

**Đối với robot SCARA:**

* Lập trình di chuyển, thao tác ở tốc độ cao phù hợp với tốc độ băng truyền, đồng thời phải chính xác trên mô hình robot SCARA.
* Lập trình sao cho các đường quy hoạch phải đảm bảo không bị gián đoạn, bị đứt quãng bám theo đường đi đã quy định trước.
* Robot có thể xử lí tốt được các dữ liệu số lượng lớn từ các file Gcode đã giải mã từ máy tính.
* Xây dựng băng chuyền có tốc độ ổn định và chắc chắn

**Đối với việc kết hợp thành hệ thống hoàn chỉnh:**

* Phối hợp, trao đổi thông tin một cách chính xác, nhanh chóng. Giao thức của các gói thông tin phải rõ ràng, có quan hệ shakehand giữa máy tính và robot để kiểm soát các trạng thái hiện tại của hệ thống, giữa các thuật toán phải liên kết với nhau một cách chặt chẽ và linh hoạt.
* Xây dựng được phần mềm giao diện cho người dùng dễ sử dụng, thân thiện và hiệu quả.

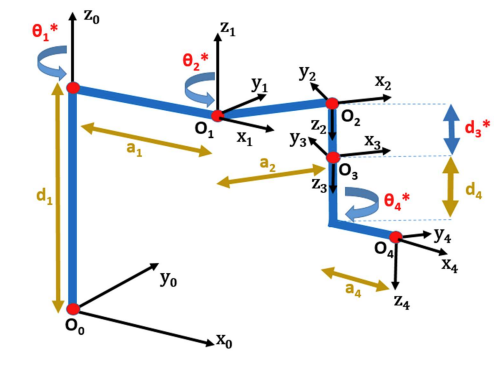
# TỔNG QUAN VÀ PHƯƠNG PHÁP

## LÝ THUYẾT ĐIỀU KHIỂN ROBOT

Trong phần này ta sẽ bàn về các lý thuyết chính về robot manipulator ứng dụng cụ thể cho robot SCARA.

### Động học thuận robot

Động học thuận là bài toán đi tìm vị trí và hướng xoay của các hệ tọa độ được đặt trên các khớp của robot, và quan trọng nhất là trên đầu end effector, khi biết trước giá trị của các biến khớp. Bài toán này được xây dựng dựa trên phương pháp Denevit-Hartenbeg. Chi tiết về phương pháp này được trình bày trong phần phụ lục. Dưới đây là phần tính toán động học thuận robot SCARA.



Hình 2.1 Hệ toạ độ được chọn theo phương pháp D-H

Ta đặt các hệ tạo độ như hình vẽ. Trong đó, hệ là hệ tọa độ chuẩn, hệ là hệ tọa độ tại đầu công tác (end-effector).

Dựa vào hình, ta lập được bảng thông số theo phương pháp Denavit-Hartenberg như sau:

Bảng 2.1 Bảng thông số D-H

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Khâu** |  |  |  |  |
| **1** |  |  |  | 0 |
| **2** |  | 0 |  | 180 |
| **3** |  |  | 0 | 0 |
| **4** |  |  |  | 0 |

Các ma trận chuyển vị trung gian:

Các ma trận chuyển vị từ các hệ về hệ chuẩn:

Vậy ta tìm được vị trí và hướng của đầu công tác như sau:

### Động học ngược robot SCARA

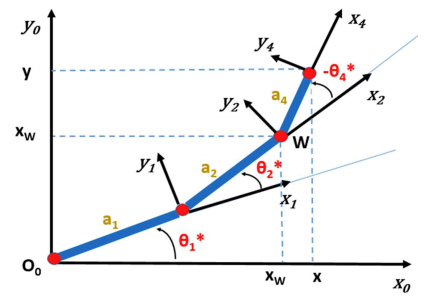
Trong việc hoạch định robot, ta hoạch định trước vị trí và hướng của những điểm robot mà sẽ đi tới, nên yêu cầu đặt ra là tìm các giá trị biến khớp tương ứng với vị trí và hướng đó.

Động học ngược là bài toán đi tìm giá trị của các biến khớp khi biết trước vị trí và hướng xoay của đầu công tác.

Vì góc và luôn cố định, nên ta chỉ quan tâm vào góc của robot.

Giả sử ta biết trước vị trí và góc của robot. Theo hệ phương trình động học thuận vừa tìm được ở trên, ta dễ dàng tính được:

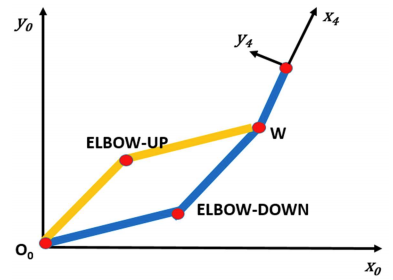
Khi nhìn robot theo hướng từ trên xuống (top view), ta được hình sau:

****

Hình 2.2 Hệ toạ độ được nhìn từ trên xuống

Ta có:

Ở đây ta thấy có 2 trường hợp nghiệm tương ứng robot sẽ có 2 dáng gọi là Elbow-up là Elbow-down. Khi tính ta sẽ chọn giá trị nào gần với giá trị hiện tại nhất. Nói cách khác, lựa chọn sao cho là nhỏ nhất.



Hình 2.3 Hai trường hợp nghiệm động học ngược

### Động học vận tốc

Trong điều khiển robot, giá trị vận tốc là một giá trị quan trọng cần tính toán. Khi hoạch định quỹ đạo trong không gian làm việc, vận tốc của đầu công tác bao gồm vận tốc dài và vận tốc góc. Do vậy, cần phải tìm quan hệ giữa vận tốc trong không gian làm việc và không gian biến khớp. Ma trận Jacobian là ma trận thể hiện quan hệ giữa các giá trị trong không gian biến khớp và không gian làm việc. Lý thuyết của ma trận Jacobian được thể hiện trong phần phụ lục. Ma trận Jacobian của robot SCARA là một ma trận , tương ứng với 4 bậc tự do : 3 khớp quay và 1 khớp tịnh tiến.

Ma trận Jacobian có dạng như sau:

Từng phần tử trong ma trận Jacobian được tính theo công thức ở bảng trên:

Vậy ta tìm được ma trận Jacobian:

### Điểm kì dị của robot SCARA

Dựa vào ma trận Jacobian đã tìm được ở trên, ta có:

Như vậy, giá trị tối đa mà hạng của ma trận **J** đạt được là 4. Điểm kì dị xảy ra khi và chỉ khi:

Thực hiện biến đổi sơ cấp trên bằng cách giao hoán các hàng, ta có được ma trận với hạng không thay đổi:

Gọi là ma trận con có kích thước của ma trận :

Ta có tính chất :

Điều này tương đương:

Ta khai triển định thức theo hàng thứ 3:

Ta biến đổi sơ cấp:

Sau đó biến đổi:

Vậy ta tìm được 2 điểm kì dị của robot. Tại vị trí trên thực tế đã được giới hạn đối với mô hình thật. Tại vị trí cần phải chú ý đến trong khi hoạch định quỹ đạo trong không gian làm việc.

### Các phương thức di chuyển của robot

Trong các ứng dụng của robot công nghiệp, ta thường gặp 2 trường hợp:

* Trường hợp 1: Đầu công tác cần đạt được vị trí và hướng tại các điểm nút. Tại các điểm này robot có thể thực hiện các thao tác cầm nắm hoặc buông nhả đối tượng. Trong trường hợp này, các điểm nút là mục tiêu quan trọng, còn dạng đường đi tới các điểm nút là thứ yếu
* Trường hợp 2: Đầu công tác cần phải đi theo một đường có hình dạng xác định ví dụ như đường thẳng, đường tròn. Trường hợp này công việc của robot là sơn, hàn, cắt kim loại, … Vì vậy, việc hoạch định quỹ đạo cho trường hợp này là rất quan trọng.

Tùy vào nhiệm vụ của robot mà chọn phương pháp hoạch định quỹ đạo phù hợp. Mục đích của việc hoạch định quỹ đạo là tạo tập giá trị đầu vào cho bộ điều khiển robot di chuyển theo quỹ đạo định trước. Thông thường, cần phải ràng buộc về vận tốc, gia tốc và thời gian di chuyển của robot. Vì thế, cần thiết lập một luật chuyển động theo thời gian để đáp ứng được các ràng buộc đó, đồng thời giúp robot di chuyển một cách mượt mà.

**Qui ước:** để tránh việc hiểu nhầm, ta thống nhất các khái niệm:

* Hoạch định quỹ đạo - Trajecroty planning: là việc thiết lập quan hệ vị trí biến khớp q, quãng đường đi được , vận tốc , gia tốc theo thời gian .
* Hoạch định đường đi - Path planning: là việc thiết lập quan hệ vị trí trong không gian làm việc theo quãng đường trong không gian.

Ta xây dựng 2 kiểu di chuyển cho robot riêng biệt:

* Kiểu 1: di chuyển trong không gian làm việc (Work space). Đầu tiên, ta lập phương trình của đường đi mong muốn trong không gian làm việc (Path planning), đường thẳng hoặc đường tròn. Từ đó, ta biết được chính xác độ dài quãng đường cần đi gọi là . Ta thiết lập luật quãng 20 đường, vận tốc dài, gia tốc dài theo thời gian , đó là hoạch định quỹ đạo (Trajectory planning). Đó là qui hoạch về vị trí 3D. Song song với việc đó, ta còn phải qui hoạch về hướng xoay theo thời gian để vị trí và hướng đồng bộ với nhau.
* Kiểu 2: di chuyển trong không gian biến khớp (Joint space). Kiểu di chuyển này quan tâm nhất là vị trí đầu và vị trí cuối, không quan tâm hình dạng đường đi. Dựa vào vị trí đầu và cuối, ta xác định được quãng đường mà của từng biến khớp cần đi trong không gian biến khớp, đó là một vector gọi là . Ta thiết lập luật quãng đường, vận tốc, gia tốc cho từng biết khớp theo thời gian (Trajectory planning).

### Hoạch định quỹ đạo

Gọi là quãng đường (quãng đường 3D, góc quay). Hoạch định quỹ đạo là việc xác định phương trình: với t là thời gian.

Khi hoạch định quỹ đạo, có 2 cách ràng buộc đầu vào mà người sử dụng cần cung cấp cho robot:

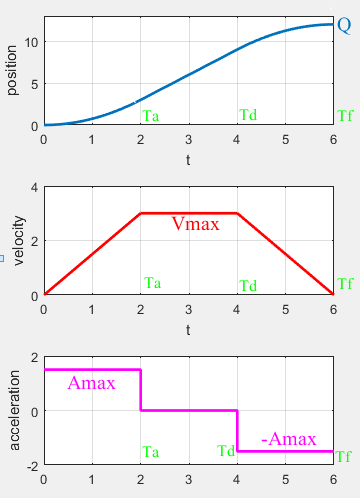
* Q\_V\_A: ràng buộc quãng đường 𝑞, vận tốc tối đa và gia tốc tối đa . Vận tốc và gia tốc này không được vượt quá vận tốc giới hạn và gia tốc giới hạn đã thiết kế sẵn cho robot. Cách này phù hợp với những thao tác không cần đòi hỏi thời gian thực thi xác định, chỉ cần tốc độ và gia tốc ở một mức độ phù hợp.
* Q\_V\_T: ràng buộc quãng đường 𝑞, vận tốc tối đa và thời gian thực thi . Cách này phù hợp với những thao tác cần thời gian thực thi xác định. Ví dụ khi robot chạy theo gắp vật trên băng chuyền, thời điểm robot kết thúc hành trình cần trùng khớp với thời điểm vật thể vừa đến vị trí gắp. Vì ràng buộc trước thời gian, nên cần khảo sát trước thời gian tối thiểu cần có để robot kịp di chuyển, vì bản thân robot đã có sẳn giới hạn vận tốc và giới hạn gia tốc.

Tuy nhiên, trong 1 trong 2 trường hợp trên người dùng cũng không nhất thiết phải điền đầy đủ 3 tham số vì nhiều lúc người dùng không chắc được thông số cuối sẽ thoả điều kiện khi quy hoạch trajectory.

Cho nên trong luận văn này, chương trình điều khiển được thiết kế để có thể cho người dùng nhập vào 2 trong 3 thông số của 1 trong 2 ràng buộc trên. Chương trình sẽ dựa vào các điều kiện của quỹ đạo nhập vào để tự động tính thông số còn lại cho phù hợp với các điều kiện quy hoạch. Điều này được đề cập sâu hơn ở chương 3.

#### Quỹ đạo vận tốc hình thang (LSPB)

Quỹ đạo có đặc trưng với đường vận tốc là hình thang theo thời gian, được chia làm 3 pha: tăng tốc, vận tốc không đổi, giảm tốc. Vì vận tốc đầu và vận tốc cuối bằng 0 nên thời gian tăng tốc và giảm tốc bằng nhau, hay .



Hình 2.4 Quỹ đạo vận tốc hình thang

Quãng đường, vận tốc và gia tốc trong suốt quá trình di chuyển tuân theo quy luật của các hệ phương trình sau:

* Pha tăng tốc: vận tốc tăng theo hàm tuyến tính
* Pha vận tốc không đổi: vận tốc giữ giá trị không đổi
* Pha giảm tốc: vận tốc giảm theo hàm tuyến tính

Đối với hai ràng buộc QVA và QVT ta sẽ có các điều kiện như sau:

* Với QVA ta có điều kiện:

Dấu bằng ở đây sẽ xảy khi không tồn tại pha vận tốc hằng, đồ thị vận tốc lúc này sẽ là một hình tam giác cân. Khi lựa chọn trước một trong 2 thông số thông số còn lại sẽ được tính với điều kiện trên.

* Với QVT ta có điều kiện:



Ở đây ta sẽ thấy khi chọn trước và , nếu nhỏ hơn giá trị , lúc này lượng diện tích của đồ thị vận tốc sẽ không đủ cho chiều dài quãng đường, đối với trường hợp ta sẽ gặp lại trường hợp mà đồ thị vận tốc là hình tam giác cân.

#### Quỹ đạo vận tốc S-curve

Quỹ đạo vận tốc S-curve có đồ thị vận tốc là 2 đường cong chữ S nên còn được gọi là Double S. Ưu điểm của quỹ đạo vận tốc S-curve so với quỹ đạo vận tốc hình thang là gia tốc liên tục, không thay đổi đột ngột giữa các pha.

Dạng đầy đủ của S-curve gồm có 7 pha nhưng trong luận văn này ta chỉ dùng dạng có tối đa 5 pha, bỏ đi 2 pha mà gia tốc đạt cực đại trong một khoảng thời gian. Đồ thị gia tốc được chọn có dạng 2 hình tam giác cân và pha gia tốc bằng 0 ở giữa.

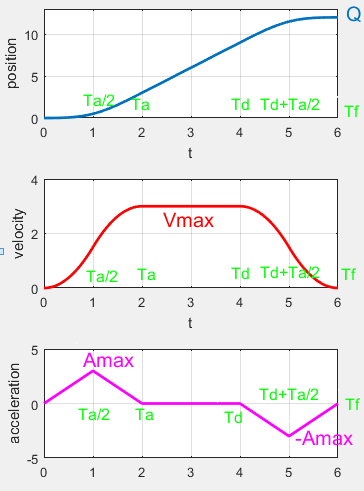
Gọi thời gian gia tốc tăng một là , thời gian gia tốc tăng và trở về 0 bằng , thời gian vận tốc không đổi là . Với ràng buộc đầu vào là quãng đường , vận tốc tối đa và gia tốc tối đa , ta tính được các giá trị thời gian:



Nếu Tc <= 0 thì pha thứ 3 không tồn tại và quỹ đạo còn 4 pha. Ta tính lại các giá trị thời gian như sau:



Giá trị sau khi tính lại sẽ nhỏ hơn ban đầu. Điều này kéo theo giá trị vận tốc mà robot đạt được sẽ nhỏ hơn giá trị mong muốn ban đầu.



Hình 2.5 Đồ thị quỹ đạo S-curve

Quãng đường, vận tốc và gia tốc trong suốt quá trình di chuyển tuân theo quy luật của các hệ phương trình sau:

Đặt 

* Pha 1: vận tốc tăng và gia tốc tăng theo chiều dương



* Pha 2: vận tốc tăng và gia tốc giảm theo chiều dương với



* Pha 3: vận tốc ở giá trị cực đại



* Pha 4: vận tốc giảm và gia tốc tăng theo chiều âm với



* Pha 5: vận tốc giảm và gia tốc giảm theo chiều âm



### Hoạch định đường đi trong không gian làm việc

Gọi là vector vị trí của đầu công tác trong không gian việc. Hoạch định đường đi là việc xác định phương trình: với là độ dài quãng đường đi trong không gian.

Gọi là điểm bắt đầu và là điểm kết thúc của hành trình.

Luận văn sử dụng 2 dạng đường đi trong không gian:

• Đoạn thẳng

• Cung tròn trên mặt phẳng XY

#### Đoạn thẳng

Gọi S là độ dài tổng quãng đường:

Vector vị trí được tính theo công thức:

#### Cung tròn trên mặt phẳng XY

Gọi là vị trí tâm của cung tròn, r là bán kính. Gọi là vector bắt đầu có gốc là tâm cung tròn, ngọn là vị trí bắt đầu. Gọi là vector kết thúc có gốc là tâm cung tròn, ngọn là vị trí kết thúc. Gọi 𝑑𝑖𝑟 là chiều di chuyển trên cung tròn, tương ứng với chiều dương và tương ứng với chiều âm.

Góc bắt đầu:

Góc kết thúc:

Vì hàm cho kết quả từ đến và quãng đường phụ thược và chiều di chuyển. Gọi là độ lớn góc di chuyển. Xét .

Nếu: thì . Nếu: thì

Gọi S là độ dài tổng quãng đường:

Vector vị trí được tính theo công thức:



## GIỚI THIỆU PHƯƠNG PHÁP NHẬN DIỆN ĐỐI TƯỢNG

### Lựa chọn phương pháp nhận diện đối tượng

Phát hiện đối tượng là một kĩ thuật máy tính liên quan đến thị giác máy tính và xử lý ảnh nhằm phát hiện ra đối tượng thuộc các lớp nhất định (ví dụ như con người, tòa nhà, xe hơi,...) trong hình ảnh hay video kỹ thuật số. Lĩnh vực của phát hiện đối tượng được nghiên cứu kỹ lưỡng thường là phát hiện khuôn mặt và phát hiện người đi đường. Phát hiện đối tượng được ứng dụng trong nhiều lĩnh vực của thị giác máy tính, bao gồm cả truy xuất hình ảnh và video giám sát. Các phương pháp phát hiện đối tượng thường là các phương pháp tiếp cận dựa trên mạng neural hoặc không dựa trên mạng neural.

Đối với phương pháp tiếp cận không dựa trên mạng neural, đầu tiên ta cần sử dụng phương pháp để xác định và trích xuất đặc trung như *Viola – Jones object detection framework based on Haar features, Scale – invariant feature transform (SIFT), Histogram of oriented gradients (HOG) features*. Sau đó sử dụng kỹ thuật như *Support vector machine (SVM)* để phân loại các đối tượng.

Đối với phương pháp dựa trên mạng neural, kỹ thuật mạng neural có khả năng thực hiện tác vụ từ đầu đến cuối trong việc nhận diện đối tượng mà không cần phải có bước xác định và trích xuất đặc trưng trung gian nào. Kỹ thuật này được dựa trên *Convolutional reural netwworks (CNN)* và được xây dựng với nhiều kỹ thuật và kiến trúc khác nhau như:

* Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN, cascade R-CNN)
* Single Shot MultiBox Detector (SSD)
* You Only Look Once (YOLO)
* Single-Shot Refinement Neural Network for Object Detection (RefineDet)
* Retina-Net
* Deformable convolutional networks.

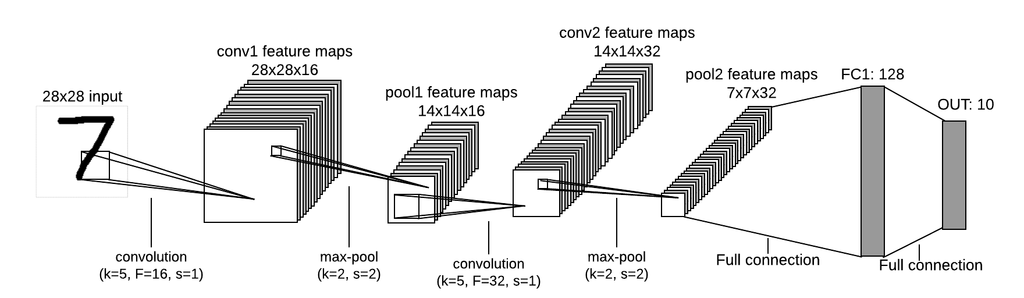
Các phương pháp học sâu khác nhau có ưu điểm và khuyết điểm khác nhau. Đối với các phương pháp *Object Detection Two-Stage* như họ các mô hình *R-CNN* có ưu điểm về độ chính xác cao nhưng có nhược điểm về mặt số lượng bounding box và thời gian training cũng như tốc độ xử lý cho mỗi khung ảnh cũng rất chậm. Còn các phương pháp Object Detection Single Stage như SSD, YOLO, Retina-Net có khả năng thực hiện việc xác định đối tượng và phân loại đối tượng cùng một lúc nên thời gian training cũng như tốc độ xử lý trên mỗi khung hình khá cao nhưng độ chính xác lại thấp hơn so với R-CNN. Đặc biệt là họ mô hình YOLO, vì không sử dụng region proposal nên tốc độ xử lí theo thời gian thực và độ chính xác ở mức cho phép nên được ứng dụng nhiều vào các bài toán thời gian thực như đếm số người, xác định phương tiện di chuyển,…

Chính vì điều này nên theo mục tiêu của đề tài luận văn này, ta thấy rằng mô hình *YOLO* phù hợp để thực hiện nhiệm vụ xác định và phân loại đối tượng trên bằng truyền.

### Tổng quát về mạng neural tích chập (CNN – Convolutional Neural Network)

Mô hình mạng nơ-ron tích chập (CNN) là mô hình kiến trúc mạng lưới thần kinh cải tiến từ Neural Network (NN). Ở mô hình tổng quát NN, các hidden layer được gọi là ***fully connected layer (FC)***, lí do có tên gọi đó là vì mỗi node trong FC được kết nối trực tiếp với tất cả các node trong layer trước. Cả mô hình được xây dựng với các FC được gọi là ***fully connected neural network (FCN)****.* Nhược điểm lớn nhất của *FCN* đó là để lấy được đặc trưng của ảnh ta cần phải có một số lượng lớn parameter, từ đó làm cho thời gian học và xử lý một ảnh của mạng *FCN* dài không đạt hiệu quả cao, tốn nhiều thời gian huấn luyện và chạy xử lí.

Trong khi đó ta thấy rằng, trong ảnh, các pixel ở càng gần nhau thì thường có liên kết với nhau hơn là những pixel ở xa. Dựa vào tính chất này của ảnh, ta tạo ra phép tính convolution. Ở phép tính này, chỉ 1 mặt nạ (kernel) sẽ được dùng trên toàn bộ bức ảnh tương ứng với 1 filter. Hay nói cách khác là cứ ứng với 1 filter thì các pixel trong ảnh sẽ cùng chia sẽ với nhau 1 trọng số. Áp dụng phép tính convolution vào layer trong neural network ta chỉ cần sử dụng, huấn luyện một lượng nhỏ các trọng số nhưng vẫn lấy được những đặc trưng cần thiết cho ảnh, từ đó giải quyết được vấn đề lượng lớn parameter mà FCN gặp phải. Chi tiết các lớp mạng CNN được trình bày ở phần phụ lục.



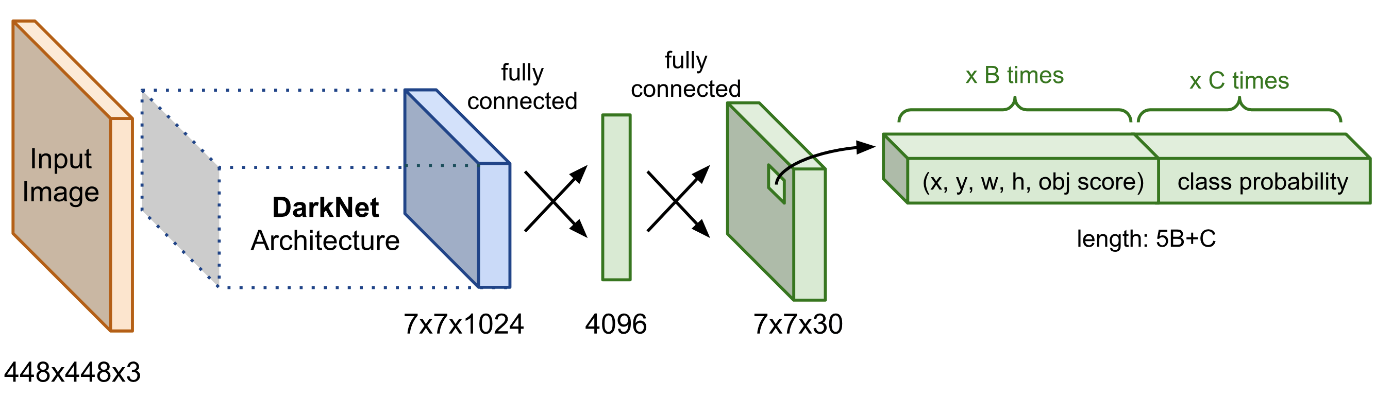
Hình 2.6 Ví dụ mô hình CNN Mô hình thuật toán YOLO

YOLO trong object detection có nghĩa là “You only look once”. Tức là chỉ cần nhìn một lần là có thể phát hiện ra vật thể. Đó chính xác là cách hoạt động của YOLO. Về độ chính xác của YOLO thì có thể không phải là tốt nhất nhưng YOLO là một trong những thuật toán nhanh nhất trong các lớp mô hình object detection. Nó có thể đạt được tốc độ thời gian thực mà độ chính xác không quá giảm so với các model thuộc top đầu. YOLO là thuật toán object detection nên mục tiêu của mô hình không chỉ là dự đoán nhãn cho vật thể như các bài toán classification mà nó còn xác định bounding box, vị trí của vật thể trong ảnh. Do đó YOLO có thể phát hiện được nhiều vật thể có nhãn khác nhau trong một bức ảnh thay vì chỉ phân loại duy nhất một nhãn cho một bức ảnh.

### Tổng quan về mạng YOLO

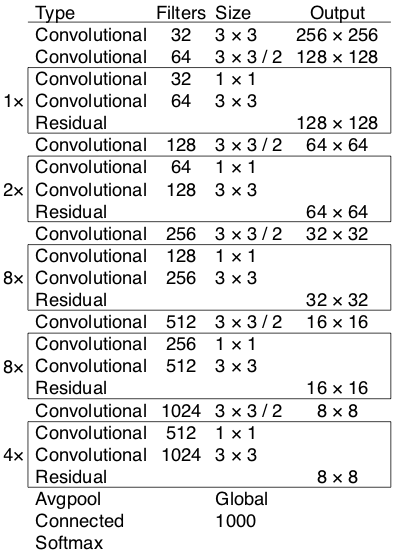
#### Kiến trúc của mạng YOLO

Kiến trúc YOLO [1] bao gồm: *Base network* là các mạng *Convolutional Layer* và *Fully Conntected Layer* làm nhiệm vụ trích xuất đặc trưng của ảnh đầu vào và liên kết chúng lại với nhau. Phần phía sau là những *Extra Layers* được áp dụng để dự đoán ra xác xuất, độ tin cậy và tọa độ phát hiện vật thể trên feature map của *Base network* đã tạo ra. Các kiến trúc YOLO cũng khá đa dạng và có thể tùy biến thành các phiên bản khác nhau với những kích thước ảnh đầu vào khác nhau.



Hình 2.7 Sơ đồ kiến trúc tổng quát của mô hình YOLO [1]

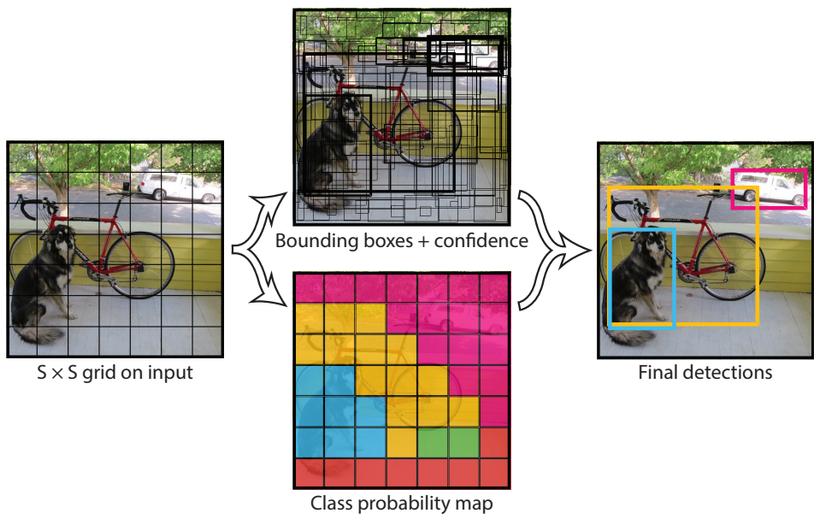
Ở mỗi phiên bản khác nhau của YOLO, kiến trúc của mạng *Feature Extractor* cũng khác nhau cùng với những tác dụng nhằm ứng các mục đích khác nhau. Ví dụ như trong YOLO version 3 (YOLOv3) [2], tác giả áp dụng một mạng *Feature Extractor* là *darknet-53*. Mạng này gồm 53 convolutional layers kết nối liên tiếp, mỗi layer được theo sau bởi một *Batch Normalization* và một *Activation function* là *Leaky Relu*. Tác giả đã down sample bằng các filter với kích thước là 2 sau mỗi convolution layer để giảm kích thước của kết quả output. Điều này giúp giảm thiểu số lượng tham số cho mô hình.



Hình 2.8 Các layer trong mạng darknet-53 [2]

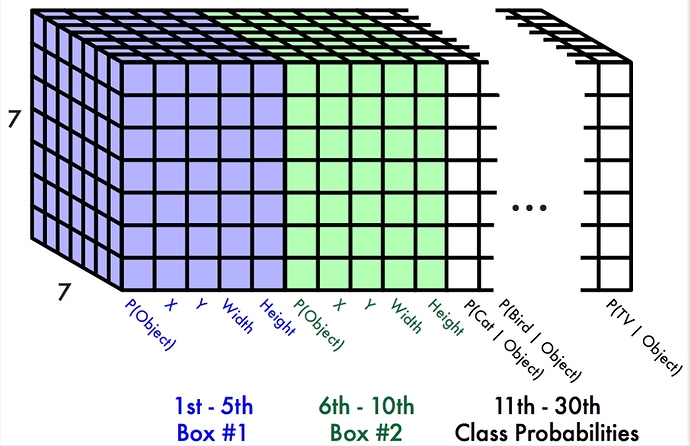
#### Cách YOLO hoạt động

Ảnh đầu vào được chia thành Lai S×S ô thường thì sẽ là 3×3, 7×7, 9×9... việc chia ô này có ảnh hưởng tới việc mô hình phát hiện đối tượng.



Hình 2.9 Cách hoạt động của mô hình [3]

YOLO sẽ dự đoán xem trong mỗi ô xem liệu có object mà điểm trung tâm rơi vào ô đó. Đồng thời, dự đoán điểm trung tâm, kích thước của object đó và xác xuất object đó là object nào trong các objects cần xác định. Với Input là 1 ảnh, đầu ra mô hình là một ma trận 3 chiều có kích thước với **B** và **C** lần lượt là số lượng Box và Class mà mỗi ô cần dự đoán. Dự đoán mỗi bounding box gồm 5 thành phần: với là tọa độ tâm của bounding box, lần lượt là chiều rộng và chiều cao của bounding box, prediction được định nghĩa ***)***. Ví dụ như phân chia hình ảnh thành một mạng lưới 7x7 ô, nếu mỗi ô có trách nhiệm dự đoán 2 hộp (B =2) bao quanh. Mỗi 1 hộp mô tả hình chữ nhật bao quanh một object. Giả sử ta đang huấn luyên YOLO nhận dạng 20 objects khác nhau (C = 20). Sau khi qua các layers, image input sẽ được biến đổi thành 1 tensor kích thước 7x7x30.



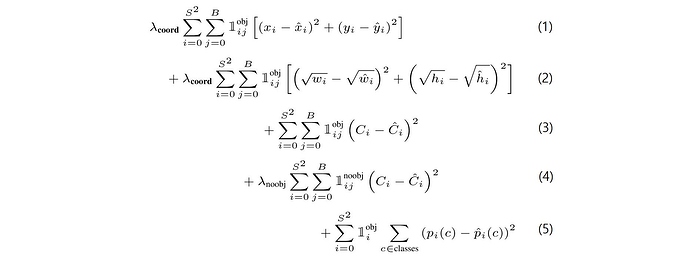
Hình 2.10 Minh họa cho kích thước output của mạng YOLO [3]

Như vậy, Neural Network có nhiệm vụ huấn luyện các trọng số của các layers để có được mô hình tốt cuối cùng. Dựa vào các tham số trong grid 7x7x30 ta sẽ xác định được các box chứa object với xác xuất cao. (các box đè lên nhau sẽ được loại bằng phương pháp NMS, chỉ giữ lại box có xác xuất cao nhất).

#### Hàm Loss function

Hàm lỗi trong YOLO được tính trên việc dự đoán và nhãn mô hình để tính [3]. Cụ thể hơn nó là tổng độ lỗi của 3 thành phần lỗi sau :

* Hàm lỗi của việc dự đoán loại nhãn của object - Classification loss
* Hàm lỗi của việc dự đoán tọa độ tâm, chiều dài, rộng của boundary box (x, y, w, h) - Localization loss
* Hàm lỗi của việc dự đoán bounding box đó chứa object so với nhãn thực tế tại ô vuông đó - Confidence loss



Hình 2.11 Hàm mất mát YOLO [3]

(1) Tính toán loss (tổn thất) của điểm trung tâm cho hộp  của ô nơi object tồn tại. Chú ý là  là tham số của tensor output của neural net còn  là của ground-truth. Tương tự cho tất cả các biến khác.

(2) Tính toán tổn thất width và height của hộp    của ô   nơi object tồn tại.

(3) Đối với các hộp  của ô   nơi object tồn tại, tính tổn thất của xác xuất object tồn tại. Chú ý  luôn = 1.

(4) Đối với hộp  của ô   và nơi không có object, tính tổn thất của xác xuất này. Chú ý   luôn = 0.

(5) Tính tổn thất của xác suất có điều kiện cho ô  nơi object tồn tại. Chú ý luôn = 1 nếu đúng lớp  với ground-truth, ngược lại thì   luôn = 0.

 thông số cân bằng để cân bằng tổn thất tọa độ (x, y, w, h) với các tổn thất khác.

 thông số cân bằng để cân bằng giữa hộp có và không có object. (Nói chung, đa số các ô trong image không có object, rất ít ô có object).

#### Anchor Boxes

Anchor Boxes là một thay đổi quan trong ở phiên bản thứ hai của YOLO [4]. Anchor boxes là các box được định nghĩa trước về hình dạng (width, height). YOLOv1 dự đoán trực tiếp các thông số của hộp chứa object (hình chữ nhật, bounding box) dựa vào connected layers. YOLOv2 loại bỏ connected layers và các convolutional layers, dự đoán các tham số của hộp chứa object dựa vào anchor boxes rồi tinh chỉnh x, y, width, height cũng như các xác xuất   và . Để có anchor boxes, YOLOv2 sử dụng k-means clustering trên các ground-truth boxes (thông số các objects trong các ảnh dùng để huấn luyện).

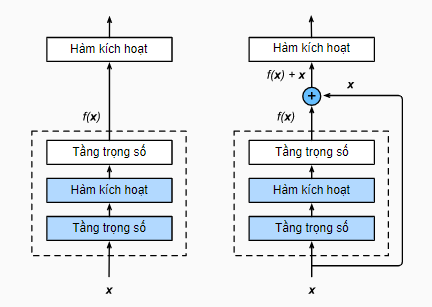
#### Non-max suppression

Do thuật toán YOLO dự báo ra rất nhiều bounding box trên một bức ảnh nên đối với những cell có vị trí gần nhau, khả năng các khung hình bị trùng lên nhau là rất cao. Trong trường hợp đó YOLO sẽ cần đến thuật toán giúp giảm bớt số lượng các khung hình được sinh. Từ như nhu cầu đó thuật toán Non-max Suppression được sinh ra giúp giảm bớt tình trạng đó một cách đáng kể [4].

#### Partial Residual Networks

Kể từ khi AlexNet giành chiến thắng trong cuộc thi the *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* vào năm 2012, kỷ nguyên của *Deep Convolutional Networks (DCN)* đã được mở ra. Tiếp theo đó là kiến trúc của *Plain networks (PlainNet)* đã làm cho *VGG-16* và *VGG-19* đạt được hiệu quả tuyệt vời. Tuy nhiên, các nhà nghiên cứu cũng đã phát hiện ra rằng khi DCN đạt đến một độ sâu nhất định thì độ chính xác của PlainNet bắt đầu giảm khi mà chiều sâu tăng. Vấn đề này không dễ dàng để có thể giải quyết, kể cả khi *GoogLeNet* được tạo với sự áp dụng chiến lược tổn thất phụ trợ *(the strategy of auxiliary loss)* vẫn không thể giải quyết được vấn đề. Sau đó, trong thiết kế của *Residual networks (ResNet),* tác giả đã giới thiệu khái niệm về kết nối tắt nhận dạng, thứ có thể làm cho gradients lan truyền ngược tới tất cả các layers một cách hiệu quả. Bằng cách sử dụng khái niệm này, họ đã xây dựng thành công một số kiến trúc rất sâu như *ResNet-50, ResNet-101, and ResNet-152*.

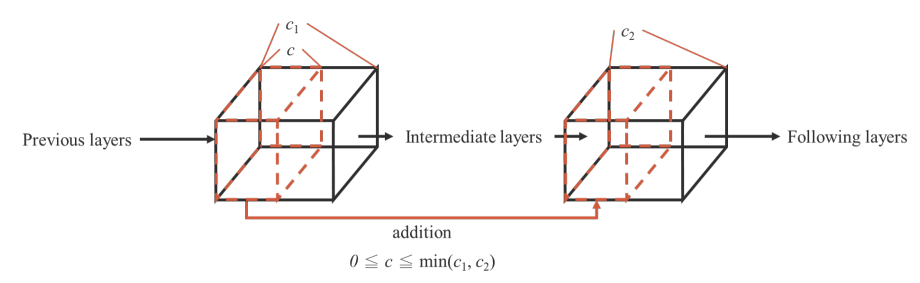
Ý tưởng trọng tâm của ResNet là mỗi tầng được thêm vào nên có một thành phần là hàm số đồng nhất. Điều này có nghĩa rằng, nếu ta huấn luyện tầng mới được thêm vào thành một ánh xạ đồng nhất , thì mô hình mới sẽ hiệu quả ít nhất bằng mô hình ban đầu. Vì tầng được thêm vào có thể khớp dữ liệu huấn luyện tốt hơn, dẫn đến sai số huấn luyện cũng nhỏ hơn. Sau đây là mô hình của *Residual networks* [5] *.*



Hình 2.12 Sự khác biệt giữa khối thường (trái) và khối Residual networks (phải)

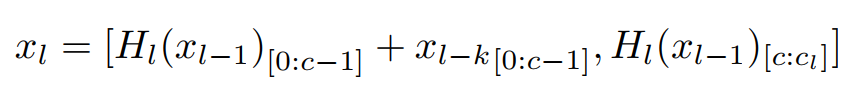
Ký hiệu đầu vào là . Giả sử ánh xạ lý tưởng muốn học được là và được dùng làm đầu vào của hàm kích hoạt. Phần nằm trong viền nét đứt bên trái phải khớp trực tiếp với ánh xạ  Điều này có thể không đơn giản nếu chúng ta không cần khối đó và muốn giữ lại đầu vào . Khi đó, phần nằm trong viền nét đứt bên phải chỉ cần tham số hoá độ lệch khỏi giá trị , bởi vì ta đã trả về Trên thực tế, ánh xạ phần dư thường dễ tối ưu hơn, vì chỉ cần đặt . Nửa hình bên phải mô tả khối phần dư cơ bản của ResNet. Về sau, những kiến trúc tương tự đã được đề xuất cho các mô hình chuỗi (sequence model).

Dựa trên mô hình của mạng ResNet, mạng *Partial Residual Networks (PRN)* được tạo ra [6]. Với mạng PRN, phần kết nối dư thừa *(the residual connection)* được chuyển đổi thành một đường dẫn tạo ra sự kết hợp giữa các gradients. Từ dó, mạng PRN không còn lan truyền sự kết hợp các đặc trưng mà là sự kết hợp các gradients. Và điều này làm cho mạng PRN có khối lượng nhẹ hơn vì kết hợp đặc trưng thì tạo ra các layer mới nhưng kết hợp gradient thì không. Chính đặc điểm này đã giúp cho mạng PRN phát huy hiệu quả tốt hơn đối với các ứng dụng real-time trên các thiết bị nhúng nhưng lại cho một chất lượng đáng mong đợi.

****

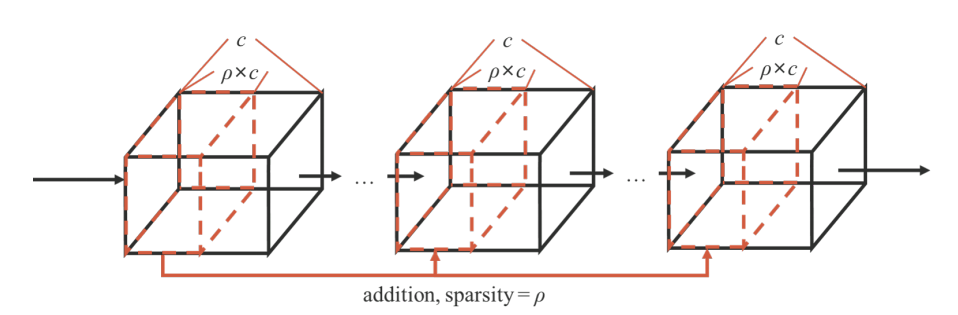
Hình 2.13 Kết nối một phần dư thừa (Partial residual connection)

Kiến trúc của PRN được thể hiện như sau: Khi ta thực hiện kết nối dư thừa một phần trên feature maps của kênh của lớp layer thứ và kênh của lớp layer thứ thì ta sẽ thực hiện phép cộng cho kênh đầu tiên của lớp thứ với kênh đầu tiên của lớp thứ . Sau khi kênh được cộng hoàn tất thì các kênh tương ứng này được chuyển tiếp tới lớp tiếp theo. Cách thức hoạt động của PRN được thể hiện như sau:



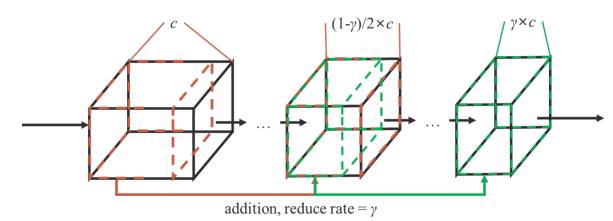
Kiến trúc PRN rất linh hoạt trong thiết kế, cho phép kết hợp các bản đồ đặc trưng *(feature maps)* của các lớp tùy ý với số lượng kênh tùy ý. Điều này khác với kiến trúc ResNet bởi vì ResNet giới hạn các khối trong cùng một giai đoạn phải có cùng số kênh. Sau đây là các tính chất và lợi ích của kiến trúc PRN:

* Đặc điểm 1: Không thay đổi cấu trúc liên kết của ResNet, chỉ thay thế kết nối phần dư *(the residual connection)* thành kết nối dư một phần *(the partial residual connection)* với độ thưa *.* Độ phức tạp tính toán của kiến trúc này gần giống như của ResNet.



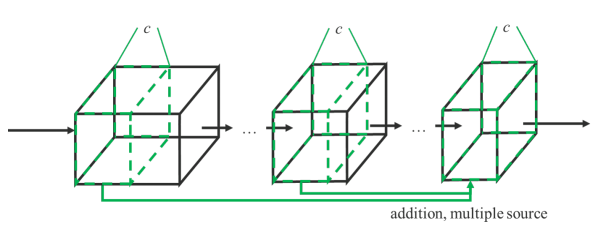
Hình 2.14 Tính chất 1 của PRN

* Đặc điểm 2: Số kênh của khối trong cùng một giai đoạn sẽ giảm tuyến tính từ tới , với tỷ lệ giảm = . Kiến trúc này có thể được sử dụng để chứng minh tính linh hoạt của PRN khi kết hợp các số kênh khác nhau.



Hình 2.15 Tính chất 2 của PRN

* Đặc điểm 3: Kiến trúc của PRN rất linh hoạt vì nó có thể kết hợp với nhiều bộ bản đồ đặc trưng riêng biệt *(sets of distinct-channel feature maps).*



Hình 2.16 Tính chất 3 của PRN

# THIẾT KẾ VÀ THỰC HIỆN

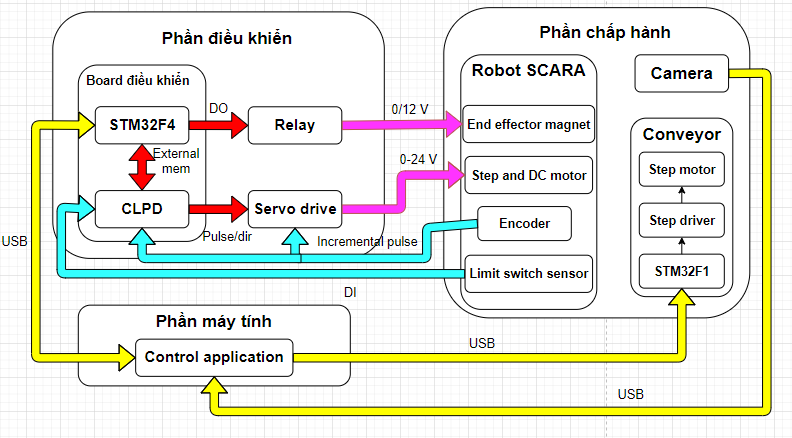
Khi tiếp nhận mô hình đã dùng của khoá trước, ta nhận thấy để đạt được hai mục tiêu mới ở trong đề tài lần này là gắp vật động và CNC routing thì ta cần cải tiến một số điều cần sau khi xem xét qua kết quả mà ở luận văn trước tác giả đã ghi.

* Các hệ số điều khiển cần phải được tune lại để các khớp đáp ứng nhanh hơn.
* Các model di chuyển như là LSPB và Scurve đã đúng về mặt toán học, tuy nhiên các công thức vẫn còn chưa được rút gọn làm tổn hao CPU cũng như các điều kiện ràng buộc vẫn chưa rõ ràng. Ở luận văn này ta sẽ hoàn chỉnh lại các model này.
* Protocol giao tiếp giữa máy tính và robot của luận văn cũ sơ sài chưa chặt chẽ, data vẫn còn gửi dưới dạng string gây tổn hao đường truyền. Nên để đáp ứng cho ứng dụng CNC routing khi gửi một lượng lớn data xuống robot, ta cần thiết kế một protocol linh hoạt và chặt chẽ hơn. Áp dụng giao thức shakehand cho từng packet.
* Phần mềm cũ của tác giả cũng đã thực hiện được những tác vụ cơ bản điều khiển robot, tuy nhiên với protocol giao tiếp mới và những cải tiến ở model xử lý, ta sẽ thiết kế lại toàn bộ phần software cho luận văn lần này. Đồng thời bổ sung thêm phần giải mã file Gcode trong software để cho mục đích CNC routing.
* Phần Hardware ta sẽ tạo dựng bổ sung thêm mô hình băng chuyền cũng như mạch điều khiển của nó.

Sau những điều đã nêu trên, ta có thể thấy để đáp ứng các yêu cầu đặt ra của luận văn lần này, ta sẽ thay đổi hầu như toàn bộ phần firmware và phần software của robot.

## TỔNG QUAN VỀ PHÂN CỨNG CỦA HỆ THỐNG

Phần hardware cơ bản của mô hình robot SCARA từ luận văn trước hầu như đã hoàn chỉnh. Ở luận văn lần này, ta lắp lại các mô hình phần cứng đã tháo dỡ, kèm theo đó là tạo dựng lên thêm mô hình băng chuyền để phục vụ cho việc gắp vật động.



Hình 3.1 Sơ đồ kết nối tổng quát của hardware hệ thống

Phần cứng hệ thống bao gồm các phần tử chính:

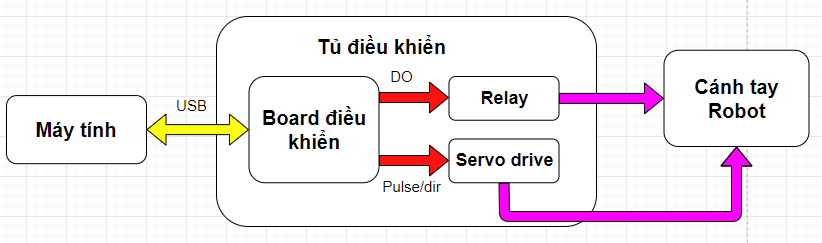
* Camera: là cảm biến thu nhận hình ảnh để xử lí.
* Robot SCARA bao gồm cánh tay, động cơ, encoder và các cảm biến hổng ngoại được kết nối và điều khiển thông qua tủ điện.
* Băng chuyền bao gồm khung sườn, dây đai, và động cơ bước kết nối với driver điều khiển.
* Máy tính: là trung tâm xử lí thông tin. Máy tính xử lí hình ảnh, ra lệnh cho robot chấp hành và điều khiển băng chuyền.

### Camera

Camera hệ thống đang dùng là loại webcam Logitech C720 với các thông số kỹ thuật sau.

|  |  |
| --- | --- |
|  | * Công nghệ tinh thể lỏng cho chất lượng gọi video và quay HD 720p. * Chụp ảnh với độ phân giải lên tới 3MP * Chuẩn USB 2.0. * Hệ điều hành tương thích: Window Vista, Window 7-8-10. |

### Robot SCARA

****

Hình 3.2 Cấu trúc phần cứng của robot

Phần cứng robot bao gồm các thành phần:

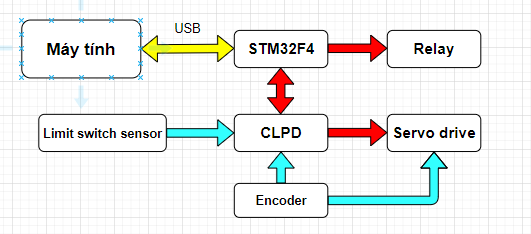
* Cánh tay robot: là kết cấu cơ khí cần được điều khiển.
* Board điều khiển: có nhiệm vụ hoạch định đường đi, hoạch định quỹ đạo cho robot, xử lí các toạ độ file Gcode, xuất xung ra các Driver để điều khiển động cơ, điều khiển đóng mở Relay để bật tắt nam châm điện trên robot. …
* Servo driver: nhận tín hiệu xung đặt vào của board điều khiển, dựa vào đó điều khiển động cơ đáp ứng vị trí mong muốn.
* Relay: được kích từ board điều khiển để bật tắt nam châm điện.

#### Mô hình cơ khí cánh tay robot

Mô hình cơ khí cánh tay trong luận văn được thừa kế từ những luận văn trước đó. Nên ở đây ta sẽ chỉ tóm tắt một số thông số chính của chúng.

Mô hình có 4 động cơ làm động lực cho 4 khớp, trong đó 3 khớp đầu tiên dùng động cơ DC Servo, khớp thứ tư dùng động cơ bước. Các cảm biến limit switch được đặt tại các góc giới hạn đã biết trước của cánh tay.

#### Board điều khiển

****

Hình 3.3 Cấu trúc board điều khiển

Board điều khiển có 2 thành phần chính là vi điều khiển STM32F407VGT6 và CLPD MAXII EPM570.

STM32f407VGT6 là vi điều khiển được thiết kế dựa trên processor ARM Cortex-M4 core. Với một số đặc điểm chính sau.

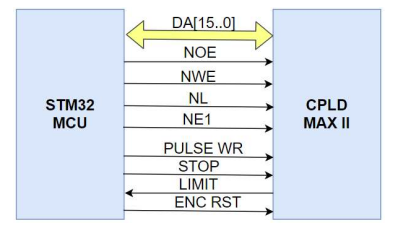
|  |  |
| --- | --- |
|  | * Up to 1 Mbyte of Flash memory * 192 Kbytes SRAM * Frequency up to 168Mhz * Equipped with multiple interfaces and peripherals * I2C, USART, CAN, FSMC, SPI, DAC, ADC,… |

Vi điều khiển trên board mạch có chức năng giao tiếp với máy tính thông qua đường USB để nhận lệnh và gửi phản hồi. Tính toán hoạch định đường đi; quỹ đạo; cuối cùng tính ra số xung cần xuất ra cho robot, ghi số xung vào CPLD bằng cách truy cập bộ nhớ ngoài, từ đó CLPD băm xung ra đưa vào các servo driver.

CPLD (Complex Programmable Logic Device) là một vi mạch mà người dùng có thể lập trình được. Ngôn ngữ lập trình cho dòng này chủ yếu là các ngôn ngữ lập trình phần cứng như là VHDL hay là Verilog. CLPD MAXII EPM570 có các đặc điểm chính sau:

|  |  |
| --- | --- |
|  | * Up to 570 logic elements (Les). * Đầu vào có thể tích hợp cổng Schimmitt trigger để giảm nhiễu * Tần số hoạt động từ 0-304Mhz |

CLPD trên board mạch đóng vai trò như một ngoại vi của vi điều khiển. Giao tiếp với STM32 thông qua giao thức đọc/ghi bộ nhớ ngoại. Chúng được dùng ở đây để có thể tiếp quản việc đưa xung tới các servo driver một cách nhanh chóng và đồng bộ. Ngoài ra cũng cập nhập các giá trị từ encoder của động cơ và giá trị các limit switch một cách đồng thời.



Hình 3.4 Mô hình nối dây giao tiếp bộ nhớ ngoại giữa STM32 và CPLD

STM32 dùng phương thức truy cập bộ nhớ ngoài FSMC để đọc và ghi dữ liệu và các thanh ghi của CPLD. Các đường tin hiệu FSMC bao gồm: DA[15..0], NOE, NWE, NL, NE1. Tín hiệu PULSE\_WR dùng để báo hiệu cho CPLD bắt đầu chu kì rãi xung mới. Tín hiệu STOP để dừng việc rãi xung ngay lập tức. Tín hiệu LIMIT để CPLD báo hiệu khi có một cảm biến hành trình được kích. Tín hiệu ENC\_RST để xóa các thanh ghi đếm encoder trên CPLD.

#### Driver điều khiển động cơ

Mô hình cánh tay robot sử dụng 2 loại động cơ khác nhau là DC Servo và động cơ bước nên trong luận văn này ta sử dụng 2 loại driver khác nhau.

Dầu tiên là Driver DC Servo MSD, là driver được phát triển bởi CC-Smart Lab, driver điều khiển được một động cơ DC Servo và chứa các thông số sau.

|  |  |
| --- | --- |
|  | * 10-27VDC, 0-10A, 20-200W. * Điều khiển vị trí, vận tốc, gia tốc. * Encoder x4, tích hợp bộ lọc nhiễu encoder. * Phương thức giao tiếp: Pulse/Dir, Rs232(option), USB. * Tích hợp các bộ điều khiển vòng kín: PID, PI, State feedback. |

Và thứ hai là Driver động cơ bước DRV8825, một driver phổ biến dành cho động cơ bước với các đặc điểm chính như:

|  |  |
| --- | --- |
|  | * Điện áp điều khiển hoạt động từ 8.2-45V, dòng max 1.8A * Điều chỉnh được dòng giới hạn * Có thể chỉnh lên tới vi bước (1/32 bước) |

### Mô hình băng chuyền

Đối với mô hình bằng chuyền, yêu cầu đặt ra khi thiết kế mô hình này là bằng chuyền phải di chuyển với một tốc độ ổn định và có thể điều chỉnh được thông qua máy tính. Dựa vào yêu cầu thiết kế, chúng ta sẽ chọn động cơ bước làm cơ cấu truyền động cho băng chuyền để tiết kiệm chi phí khi phải đầu tư với động cơ DC.

Board điều khiển băng chuyền được thiết dùng board stm32f103c8t6 bluepill.

|  |  |
| --- | --- |
|  | * Up to 128 Kbyte Flash memory * 20 Kbyte of SRAM * Frequency up to 72Mhz * Equipped with multiple interface: CAN, I2C, SPI, USART, USB,… |

Driver động cơ bước A4988: là một driver phổ biến dành cho động cơ bước chỉ sau DRV8824 đã đề cập ở phần trên. Ngoài công suất tải của board chỉ đạt được cỡ 70% so với board DRV8825, tuy nhiên các feature thì vẫn đáp ứng tương tự.

|  |  |
| --- | --- |
|  | * Điện áp điều khiển hoạt động từ 8.2-45v, dòng max 1.3A * Điều chỉnh được dòng giới hạn * Có thể chỉnh lên tới vi bước (1/32 bước) |

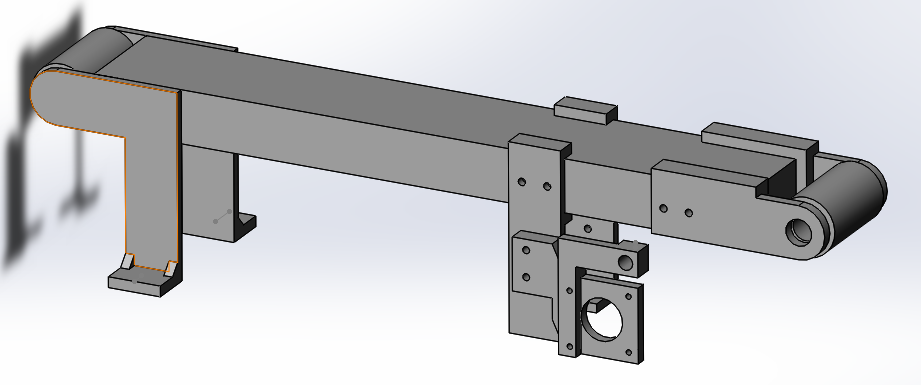
Động cơ bước ta chọn là loại động cơ với các thông số như sau:

|  |  |
| --- | --- |
|  | * Độ phân giải 1.8 độ full step * Size 42x42mm * Điện áp định mức 12v ở 1.2A |

Ta dùng cơ cấu băng đai để truyền động với 1 bánh răng 20-5mm gắn vào trục quay băng tải.



Hình 3.5 Bánh răng 20-5mm



Hình 3.6 Bản vẽ solidwork của băng truyền

Dùng solidwork để thiết kế các bộ phận của băng truyền, từ đó tách ra và in 3d các phần nhỏ ghép vào.

Ta gọi tốc độ quay của động cơ là , tốc độ của bánh lăn là . Ta có số răng của trục động cơ là 21, của trục bánh lăn là 20.

Với đường kính bánh lăn là 35.5mm, ta có thể tính được tốc độ dài của băng truyền bằng công thức sau:

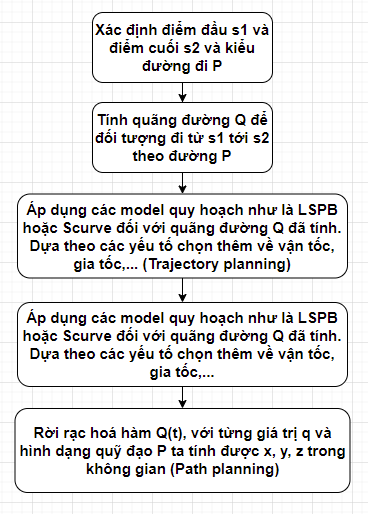


Hình 3.7 Kết quả thiết kế băng truyền

## CHI TIẾT GIẢI THUẬT CHO ROBOT SCARA

### Hoạch định các mô hình path planning và trajectory planning cho robot SCARA

Hai khái niệm **path planning** và **trajectory planning** khi dịch ra tiếng Việt đều cùng một ý nghĩa là quy hoạch quỹ đạo hoặc quy hoạch đường đi. Tuy nhiên chúng là hai hình thức khác nhau. Khi đề cập tới **trajectory planning** ta sẽ nhắc tới với việc quy hoạch các yếu tố về vật lý của đối tượng đang xét (quãng đường, vận tốc, gia tốc, lực, …). Còn đối với **path planning** ta sẽ chỉ để ý về hình dạng quỹ đạo của đối tượng (đường thẳng, đường tròn, đường spline, …). Và trong một quy trình quy hoạch tổng thể, **trajectory planning** phải xảy ra trước **path planning.** Quy trình hoạch định được cụ thể hoá theo sơ đồ như sau:



Hình 3.8 Sơ đồ hoạch định tổng thể

Ví dụ ta quy hoạch cánh tay di chuyển từ điểm tới điểm theo một đường thẳng với các bước sau đây

Tính tổng quãng đường di chuyển từ A tới B. Ở đây là đường thẳng nên ta dễ dàng tính được . Quy hoạch với model QVT cho và ta có thể tính ra được :

Với từng ta có thể tính toạ độ từng điểm bằng

Như trong phần (2.1.6.1) đã đề cập thì ta thấy được điều kiện ràng buộc của cả hai model, tuy nhiên ở đây khi ứng dụng vào gắp nhiều vật động di chuyển trên băng chuyền ta phải dùng model QVT để hoạch định có thời gian chuyển động ràng buộc, từ đó ta có thể tính được vị trí cuối để gắp của vật. Nên ta xét tới điều kiện sau của loại hình QVT:





Hình 3.9 Các biên giới ràng buộc của model LSPB(QVT)

Ta có thể thấy rằng về khía cạnh đồ thị không thể vượt quá hai giới hạn này, vì diện tích của hình thang chính là quãng đường di chuyển được. Hình thể của nó có thể thu lại hoặc dãn ra nhưng tổng diện tích của nó phải là . Đó chính là lý do tại sao nếu bằng thì nó là hình chữ nhật với diện tích và nếu vmax bằng thì nó là hình tam giác cũng có diện tích là .

Thế nên với QVT ta có thể ràng buộc Q và T từ đó cho hoặc ngược lại đối với Q và V thì để có thể đảm bảo end effector di chuyển tới mục tiêu đúng thời gian xét trong cả hai trường hợp. Điều này sẽ ứng dụng cho model gắp các vận động trên băng chuyền.

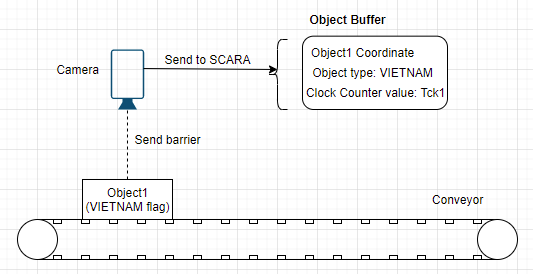
Ta cũng có thể thấy điều tương tự cho model Scurve:



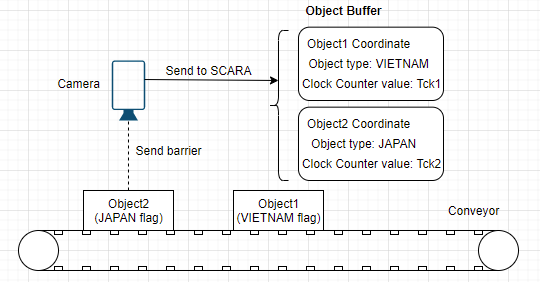
Hình 3.10 Các biên giới ràng buộc của model Scurve(QVT)

### Thuật toán gắp nhiều vật trên băng chuyền

Về phần xử lí ảnh bên phía camera, đổi với mỗi vật khi được nhận diện sẽ được gửi toạ độ lên robot chỉ một lần. Trong chương trình của robot SCARA ta sẽ tạo sẵn một ring buffer gồm các struct object để chứa thông tin về toạ độ các vật và thời điểm mà robot Scara nhận được vật đó dựa vào một biến đếm timer.



Hình 3.11 Object buffer khi nhận diện được 1 vật



Hình 3.12 Object buffer khi nhận diện được 2 vật

Các dữ liệu trong object buffer sẽ được chờ để xử lí lần lượt, những object nào đã được xử lý xong sẽ được xoá đi. Quá trình gắp vật tổng cộng 8 giai đoạn tính từ vị trí home của một slot nào đó bao gồm:

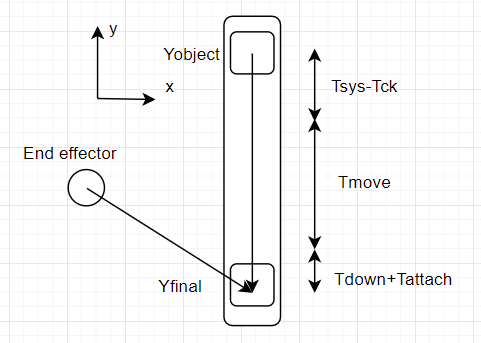
1. Di chuyển tới vị trí gắp với độ cao di chuyển định trước.
2. Hạ trục tới độ cao của vật.
3. Bật nam châm.
4. Nâng trục lên độ cao di chuyển.
5. Di chuyển tới vị trí thả vật.
6. Hạ trục tới độ cao thả.
7. Tắt nam châm.
8. Nâng trục tới độ cao di chuyển.

Tất cả 8 giai đoạn trên đều có thời gian định trước cụ thể, riêng giai đoạn 1 sẽ là then chốt để ta có thể xác định được vị trí cuối cùng để gắp vật, ta dùng theo hai phương pháp sau đây áp dụng với giai đoạn 1.

* **Xác định vị trí gắp vật với thời gian cố định**

Tốc độ băng chuyền được xem là cố định, băng chuyền được canh chỉnh đặt song song với trục y của robot. Chính vì thế ở đây dựa vào toạ độ tại thời điểm camera gửi, ta chỉ cần tính ra vị trí y của vật tại thời điểm robot thực thi một nhiệm vụ trong object buffer là có thể gắp được. Ta tính như sau:

Với



Hình 3.13 Mô hình gắp vật ràng buộc thời gian

Trong đó :

* Điểm y mà ta sẽ ra lệnh robot tới gắp.
* Điểm y của object tương ứng tại thời điểm khi camera vừa gửi.
* vận tốc hiện tại của băng chuyền.
* giá trị đếm của thanh ghi hiện tại.
* giá trị đếm của thanh ghi của object tương ứng tại thời điểm camera gửi.
* thời hằng di chuyển.
* thời gian để hạ trục xuống vị trí gắp.
* thời gian bật nam châm.
* **Xác định vị trí gắp vật với vận tốc cố định**

Đối với việc đặt vận tốc cố định cho end effector, ta chưa thể xác định được ngay vị trí vì chưa có , ta phải tính vị trí thông qua các yếu tố trung gian.



Hình 3.14 Mô hình gắp vật ràng buộc vận tốc

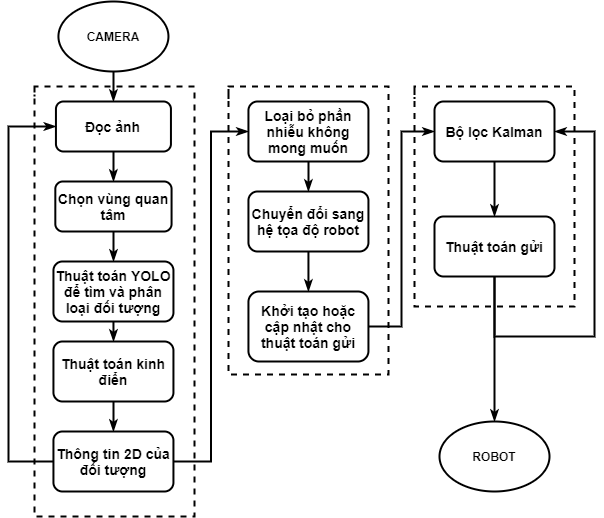
Với . Ta có thể tính:

Ta gọi vận tốc cố định để gắp vật là . Xét tam giác ABC như hình ta có:

## CHI TIẾT GIẢI THUẬT THỊ GIÁC MÁY

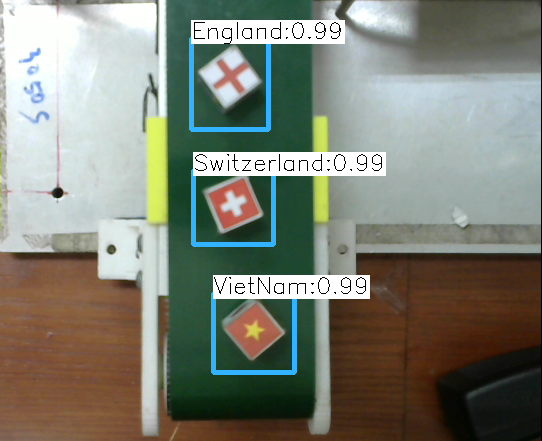
### Sơ lược, tóm tắt về giải thuật thị giác máy

Giải thuật chạy trong giao diện chương trình được chia làm 3 giai đoạn khác nhau và có thể chạy song loan với nhau. Cấu trúc giải thuật được thể hiện ở lưu đồ sau đây:



Hình 3.15 Sơ đồ thuật toán thị giác máy tổng quát

Ở giai đoạn 1, giải thuật bắt đầu với việc thu thập ảnh từ camera. Ảnh sau đó được chỉnh sửa sao cho chỉ giữ lại vùng quan tâm trước khi sử dụng thuật toán YOLO. Giải thuật YOLO giúp ta phát hiện và phân loại đối tượng.



Hình 3.16 Kết quả thuật toán YOLO phát hiện đối tượng

Đầu vào của thuật toán YOLO là ảnh và đầu ra là danh sách các đối tượng đã được phát hiện kèm theo đó là thông tin về danh tính (ID), độ tin cậy (confidence), vị trí, kích thước bounding box… YOLO là một công cụ hữu ích trong việc phát hiện và phân loại đối tượng. Tuy nhiên, thông tin về vị trí của đối tượng mà YOLO xuất ra không phải lúc nào cũng có độ chính xác cao. Nhưng hình ảnh ở trên, tâm của vật nhiều lúc nằm lệch ra nhiều so với tâm của bounding box nên nếu sử dụng thông tin vị trí này để tiến hành gắp vật thì kết quả sai số sẽ rất cao, thậm chí không thể gắp được. Chính vì điều này nên ta cần thêm một bài toán để xác định chính xác tâm của vật dựa vào hình ảnh cắt từ bouding box của đối tượng.

Sau khi sử dụng các thuật toán kinh điển để tìm thông tin 2D của vật thể thì ta sẽ chuyển thông tin này sang giai đoạn 2 để xử lí tiếp. Ở giai đoạn 2, ta sẽ loại bỏ các giá trị không mong muốn bằng cách xét các giới hạn và điều kiện. Sau đó, đưa giá trị vị trí của đối tượng qua ma trận chuyển đổi để đưa từ hệ tọa độ camera sang hệ tọa độ robot. Tiếp theo, ta khởi tạo các giá trị ban đầu nếu đối tượng xuất hiện lần đầu hoặc cập nhật giá trị cho bộ lọc Kalman theo giá trị đã chuyển đổi được.

Tiếp nói đó là giai đoạn 3 với bộ lọc Kalman được chạy lấy mẫu mỗi 3 ms dựa trên giá trị đã thiết lập hoặc cập nhật ở giai đoạn 2. Nếu các giá trị thông tin này thỏa điều kiện gửi thì sẽ gửi cho robot để tiến hành gắp vật và xóa thông tin của đối tượng đó ra khỏi bộ nhớ.

### Giới thiệu thư viện OpenCV

OpenCV (*Open Source Computer Vision Library*) là một thư viện chứa những hàm được lập trình sử dụng trong xử lí thời gian thực chứa hàng ngàn thuật toán tối ưu hoá, trong đó cung cấp một bộ công cụ phổ biến cho các ứng dụng về thị giác máy tính, được phát triển ban đầu bởi Intel, sau này được hỗ trợ bởi  Willow Garage và Itseez. Đây là một thư viện cross-platform , miễn phí cho người sử dụng dưới bản quyền open-source BSD. OpenCV có thể chạy trên nhiều hệ điều hành khác nhau như: Window, Linux, macOS, FreeBSD, NetBSD, OpenBSD,… và được viết bằng nhiều ngôn ngữ như C++, Python… Thư viện OpenCV bao gồm một số tính năng nổi bật như: Bộ công cụ hỗ trợ 2D và 3D, nhận diện khuôn mặt, nhận diện cử chỉ, nhận dạng chuyển động, đối tượng, hành vi, tương tác giữa con người và máy tính, điều khiển Robot, hỗ trợ thực tế tăng cường,…Trong luận văn, thư viện OpenCV là được dùng để triển khai các kĩ thuật xử lí ảnh và chạy mô hình YOLO.

### Huấn luyện mô hình YOLO

Như đã phân tích phía trên, mô hình YOLO được ứng dụng trong luận văn là mô hình YOLOv3-tiny-prn với chức năng nhận dạng, phân loại, xác định vị trí của các vật thể. Cách thức và các bước huấn luyện được trình bày như sau:

#### Cài đặt mã nguồn Darknet

Darknet là một neural netwwork framework với mã nguồn mở giống như tensorflow hay pytorch để huấn luyện các mạng thần kinh ở lĩnh vực deeplearning phục vụ cho các mục đích như xác định đối tượng, phân loại phức đối tượng … Darknet được viết bằng C và CUDA, nó nhanh hơn, dễ cài đặt hơn và có thể hỗ trợ tính toán cho cả CPU và GPU. Để cài đặt ta cần phải clone the Darknet git repository với địa chỉ: https://github.com/AlexeyAB/darknet.git và lưu thư mục vừa tải với tên darknet.

Bước quan trọng nhất khi huấn luyện mô hình YOLO là biên dịch file cài đặt. Như ta đã nói, ta sẽ dùng file yolov3-tiny-prn.cfg gồm 3 lớp “net, convolutional, maxpool, yolo’’ để cấu hình cho mô hình huấn luyện.

Bảng 3.1 Các thông số cấu hình mạng YOLOv3-tiny

|  |  |
| --- | --- |
| **[net] Layer** | |
| batch=64 | Số lượng hình ảnh sẽ được lấy ra từ tập dữ liệu trong một lần huấn luyện. |
| subdivisions=8 | Một batch sẽ được chia thành nhiều khối, subdivisions là số lượng hình ảnh trong một khối, những ảnh nào nằm trong cùng một khối sẽ được huấn luyện cùng một lúc |
| width=416 | Chiều rộng của ngõ vào của CNN, mọi hình ảnh sẽ được thay đổi kích thước thành kích thước mạng trong quá trình huấn luyện và phát hiện. Kích thước này là bội số của 32 và có thể thay đổi tùy theo người dùng. |
| height=416 | Chiều dài của ngõ vào của CNN, mọi hình ảnh sẽ được thay đổi kích thước thành kích thước mạng trong quá trình huấn luyện và phát hiện. Kích thước này là bội số của 32 và có thể thay đổi tùy theo người dùng. |
| channels=3 | Số kênh của ngõ vào của CNN. |
| momentum=0.9 | Hằng số tích lũy chuyển động (accumulation of movement), thể hiện sự ảnh hưởng của kết quả ở quá khứ đến kết quả của hiện tại khi tính toán các trọng số lúc thay đổi từ lần huấn luyện này sang lần huấn luyện khác (hay từ batch này sang batch khác). Trọng số góp phần giúp kết quả thoát khỏi cực trị cục bộ. |
| decay=0.0005 | Hệ số làm cho việc cập nhật trọng số cho các đặc trưng yếu đi từ đó loại bỏ sự mất cân bằng trong tập dữ liệu, giúp tránh tình trạng overfitting. |
| angle=0 | Góc quay của ảnh trong thời gian huấn luyện. |
| saturation = 1.5 | Thay đổi ngẫu nhiên độ bão hòa của ảnh trong thời gian huấn luyện. |
| exposure = 1.5 | Thay đổi ngẫu nhiên độ phơi sáng của ảnh trong thời gian huấn luyện. |
| hue=0.1 | Ngẫu nhiên thay đổi màu sắc (theo kênh màu HSV) của ảnh trong thời gian huấn luyện. |
| learning\_rate=0.001 | Tốc độ học là tham số kiểm soát mức độ thay đổi của trọng số trong mạng theo hàm gradient mất mát (the loss gradient) trong quá trình học tập. |
| burn\_in=1000 | Là số lượng steps ban đầu sao cho learning\_rate được tính với công thức current\_learning rate = learning\_rate \* pow (iterations / burn\_in, 4). Như vậy, ta thấy rằng giá trị này sẽ tăng dần từ 0 đến learning\_rate. Sau đó learning\_rate sẽ được giữ ổn định. Thực nghiệm cho thấy thiết lập learning\_rate bé ở những steps đầu sẽ giúp cho thuật toán hội tụ nhanh hơn. |
| max\_batches= 12000 | Số lần huấn luyện tối đa. Max\_batches = classes\*2000 nhưng không được ít hơn số hình ảnh cần huấn luyện. |
| policy=steps | Quy luật thay đổi tỉ lệ học learning rate như constant, sgdr, steps, step, sig, exp, poly, random. |
| steps=9600,10800 | Tại những thứ lần huấn luyện này thì learning rate sẽ được cập nhật theo learning rate\*scales. Thường ta chọn Steps = 80%\*max\_batches hoặc 90%\*max\_batches. |
| scales=0.1,0.1 | Giá trị cập nhật cho learning rate sau các steps đã cài đặt trước đó. Với thông số hiện tại, nếu policy = 11000 thì current\_learning\_rate = learning\_rate \* scales [0] \* scales [1] = 0.001 \* 0.1 \* 0.1 = 0.00001 |
| **[convolutional] Layer** | |
| batch\_normalize | Nếu = 1 thì sử dụng batch normalize cho layer, nếu = 0 thì sẽ không dùng. |
| filters | Số lượng bộ lọc kernel filters |
| size | Kích thước của mỗi kernel filters |
| stride | Bước di chuyển của kernel. |
| pad | Thêm các ô ở viền ngoài kernel. |
| activation=leaky | Hàm kích hoạt sau khi convolution: logistic (by default), loggy, relu, elu, selu, relie, plse, hardtan, lhtan, linear, ramp, leaky, tanh, stair, relu6, swish, mish. |
| **[maxpool] Layer** | |
| size | Kích thước của max-pooling kernel. |
| stride | Bước di chuyển max-pooling kernel. |
| **[yolo] Layer** | |
| mask | Danh sách id của các Anchors mà lớp sử dụng. |
| anchors | Thông số này là một bộ các bounding box được xác định trước với chiều rộng– cao cụ thể ở mỗi cell gọi là các anchors như đã giới thiệu ở trên. Một số box trong đó (gần nhất với kích thước đối tượng) sẽ được thay đổi kích thước thành kích thước để phù hợp với đối tượng. |
| classes = 6 | Số lượng đối tượng cần detect. |
| num = 6 | Tổng số lượng Anchors. |
| ignore\_thresh = 0.7 | Giữ các bounding box nếu IoU (detect, truth) > ignore\_thresh, được sử dụng trong thuật toán NMS |
| truth\_thresh = 1 | Điều chỉnh các bounding box nếu IoU (detect, truth) > truth thresh, được sử dụng trong thuật toán NMS |

#### Chuẩn bị tập dữ liệu huấn luyện

*Bước 1: Xây dựng bộ dữ liệu*

Ta thu thập dữ liệu hình ảnh của đối tượng thông qua camera và đặt các đối tượng dưới những phông nền khác nhau, đặt dưới những điều kiện sáng khác nhau, lần lượt thay đổi vị trí của các đối tượng trong ảnh. Sau cùng ta thu thập được khoảng 300 tấm ảnh gốc khác nhau chứa cả 6 đối tượng cùng lúc.

Hình 3.17 Ảnh do camera thu được

Khi đã thu thập được một lượng ảnh gốc nhất định ta tiến hành dùng *Data Augmentation* với các phép xoay ảnh, thay đổi độ bão hòa, màu sắc, độ sáng, làm mờ ảnh, làm nhiễu ảnh… Đây là một kỹ thuật giúp tăng được một lượng lớn số lượng data trong bộ dữ liệu, bên cạnh đó còn giúp cho model trong thời gian huấn luyện trong bị overfitting, nâng cao chất lượng xử lí. Sau khi áp dụng kỹ thuật này, ta thu được một bộ dữ liệu khoảng 1800 ảnh.

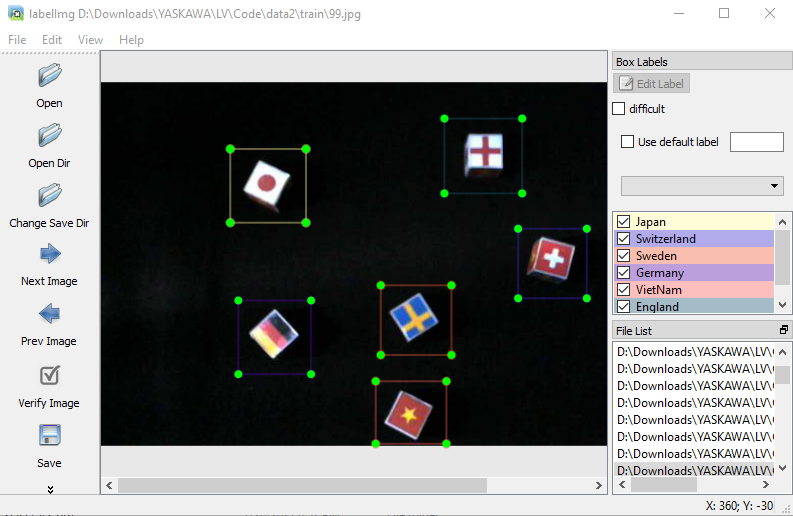
 

Hình 3.18 Ảnh sau khi dùng Data Augmentation

*Bước 2: Gán nhãn cho các đối tượng trong ảnh*

Đầu tiên ta cần tạo ra file với tên *custom.names* và *classes.txt* chứa tên của đối tượng cần nhận diện. Tên của đối tượng được viết theo từng dòng và khi huấn luyện cũng như khi dán nhãn, thực hiện chạy model thì số thứ tự của đối tượng (id) sẽ được tự động quy ước từ trên xuống dưới với giá trị từ 0 đến (số đối tượng – 1).

Tiếp theo, ta tiến hành gán nhãn tên bằng phân mềm *Imglabel* bằng cách xác định khung chữ nhật bao lấy đối tượng sau đó đặt tên nhãn cho đối tượng theo danh sách đã ghi ở file *classes.txt*.



Hình 3.19 Gán nhãn cho đối tượng

Sau khi gán nhãn ta sẽ thu được file có định dạng đuôi là txt, bên trong có chứa thông của các bounding box đã gán nhãn với cấu trúc như sau:

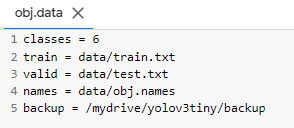
*<id-class> <center-x> <center-y> <bbox-width> <bbox-height>*

Trong đó:

* *<id-class>:* giá trị index của các classes đã được quy định theo file classes.txt.
* *<center-x> <center-y> :* lần lượt là giá trị x, y của tọa độ tâm của bounding box được chuẩn hóa bằng cách chia cho chiều rộng và cao của ảnh.
* *<bbox-width> <bbox-height>:* lần lượt là chiều rộng và chiều cao của bounding box được chuẩn hóa bằng cách chia cho chiều rộng và chiều cao của ảnh.

*Bước 3: Tạo các file chuẩn bị trước khi huấn luyện*

Tiếp theo, ta tạo file *train.txt* và *test.txt* chứa các đường dẫn đến từng ảnh phục vụ cho việc huấn luyện. Bên cạnh đó, ta tạo thêm file *obj.data* với cấu trúc như sau:



Với: *classes* là số đối tượng cần phân loại; *train, valid, names* là đường dẫn đến các file chuẩn bị ở phai trên; *backup* là đường dẫn đến folder mong muốn để chứa kết quả huấn luyện.

*Bước 4: Tải file pre-trained chứa các trọng số đã đào tạo trước*

Quá trinh học chuyển giao *(Transfer Learning)* là một kỹ thuật cho phép tận dụng những mô hình tốt đã được huấn luyện trước đó, có sẵn dựa trên những tập dữ liệu lớn để huấn luyện mô hình của tập dữ liệu khác. Cụ thể hơn, ta tận dụng trọng số của mô hình trước để là trọng số khởi đầu cho việc huấn luyện. Lợi ích của kỹ thuật này là giúp cho việc huấn luyện đạt được chất lượng tốt hơn, đồng thời sẽ giảm thời gian huấn luyện nếu so với việc huấn luyện cho mạng từ đầu. Với mô hình YOLOv3 tiny PRN ta cần tải file chứa trọng số *yolov3-tiny.conv.11.*

#### Tiến hành huấn luyện

Trong đề tài luận văn này, ta sử dụng dịch vụ máy ảo trên Google Colab (GC) để tiến hành huấn luyện cho model YOLO. GC có cung cấp một dịch vụ máy ảo chạy bằng GPU Tesla miễn phí cho người dùng.

*Bước 1: Khởi động và kích hoạt GPU*

Ta kích hoạt bằng cách vào thanh công cụ của GC, click vào Runtime > Change runtime type > GPU >Save. Sau đó ta sử dụng câu lệnh *!nvidia-smi* để kiểm tra GPU được sử dụng trên GC.



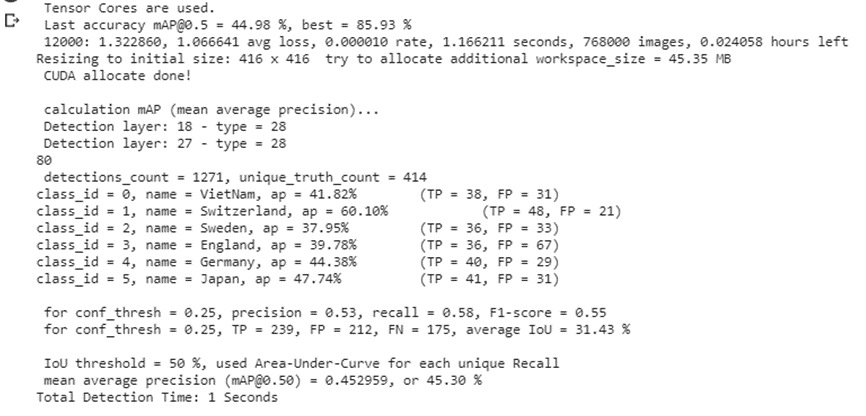
Hình 3.20 Thông số GPU trên GC

Ở hình ảnh trên, ta thấy rằng GC cung cấp cho ta một máy ảo với GPU là Tesla K80 và bộ nhớ là 11441 MiB. Như vậy với GPU này ta hoàn toàn có thể thực hiện huấn luyện các mô hình vừa và nhỏ.

*Bước 2: Tiến hành huấn luyện mô hình*

Sau khi đã chuẩn bị hết tất cả các bước cần thiết cho việc huấn luyện, ta tiến hành khởi động huấn luyện cho model bằng câu lệnh:

!./darknet detector train data/obj.data cfg/yolov3prn-obj.cfg yolov3-tiny.conv.11 -dont\_show -map

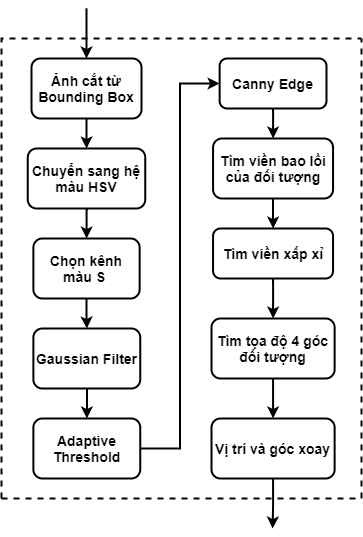


Hình 3.21 Kết quả sau huấn luyện

### Thuật toán xử lý ảnh kinh điển

Đây là bước tiếp theo sau khi ta sử dụng model YOLO để nhận diện và phân loại đối tượng. Các thuật toán xử lí ảnh kinh điển được sử dụng một cách lần lượt với mục đích cuối cùng là xác định được tâm và góc xoay của vật trong khung bounding box đã phát hiện. Phương pháp được sử dụng sau đây dựa trên phương pháp xử lý ảnh đếm số vật trên băng truyền [7].

Sau khi đã có ảnh cắt từ bounding box, ta làm cho ảnh không chịu ảnh hưởng của nhiễu. Tiếp theo ta chuyển ảnh sang nhị phân xác định vùng hiển thị của đối tượng và tiến hành xác định các cạnh bằng thuật toán Canny. Dựa vào hình ảnh của cạnh ta tìm được ta xác định các góc của vật từ đó ta biết được vị trí của tâm vật và góc xoay.



Hình 3.22 Lưu đồ các thuật toán xử lí ảnh kinh điển

#### Chuyển đổi hệ màu

RGB là không gian màu phổ biến ở hầu hết tất cả các hình ảnh kỹ thuật số hiện tại. Với 3 màu sắc đơn giản đỏ (red), xanh lá (green), xanh dương (blue), ta hoàn toàn có thể biểu diễn bất kì màu sắc nào. Không gian màu RGB được thực hiện dựa trên ý tưởng đó. Với mỗi hình ảnh màu, ta sẽ biểu diễn thành 3 ma trận khác nhau ứng với ba màu sắc cơ bản.

Bên cạnh không gian màu RGB, ta còn có không gian màu HSV, một không gian màu dựa trên 3 thông số: Màu sắc (Hue), Độ bão hòa (Saturation), Độ sáng (Value).

Trong nội dung của luận văn này, ta sẽ chuyển đổi ảnh từ hệ màu RGB sang hệ màu HSV. Sau đó, khi có được ảnh ở hệ màu HSV, ta chỉ lấy kênh màu duy nhất là Độ bão hòa (Saturation) làm mục tiêu để xử lí ảnh. Mục đích của bước chuyển đổi này là giúp ta thu được những chi tiết của hình ảnh một cách rõ ràng hơn, không phụ thuộc vào điều kiện ánh sáng.

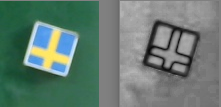
Để thực hiện việc chuyển đổi hệ màu RGB sang HSV ta sử dụng câu lệnh [8]:

*void cv::cvtColor( InputArray src, OutputArray dst, COLOR\_BGR2HSV, int dstCn=0)*

Sau đó ta tách lấy kênh Saturation bằng lệnh:

*void cv::split( InputArray m, OutputArrayOfArrays mv)* [9]

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |



Hình 3.23 Ảnh hệ màu RGB (trái) và ảnh kênh Saturation (phải)

#### Lọc nhiễu

Lọc ảnh là một kỹ thuật rất quan trọng trong xử lý ảnh với nhiều tác dụng khác nhau như loại bỏ nhiễu không mong muốn, giúp các thuật toán khác hoạt động tốt hơn, tìm biên,…

Trong số các bộ lọc thì bộ lọc Gauss là bộ lọc được sử dụng nhiều nhất trong các quy trình xử lý ảnh vì tính hiệu quả của nó. Ý tưởng của bộ lọc là việc các pixel gần nhau thường liên hệ với nhau nhiều hơn là các pixel nằm ở xa. Chính vì vậy, bộ lọc Gauss sẽ đặt ra các trọng số cho các pixel với trọng số của pixel nào càng gần tâm thì càng có giá trị cao theo quy luật phân phối chuẩn Gauss. Ta áp dụng bộ lọc này với câu lệnh:

*void cv::GaussianBlur( InputArray input, OutputArray output, , Size ksize, double Xsigma, double Ysigma)* [10]

Trong đó, ta sử dụng *ksize = (3,3)* và *Xsigma = 0, Ysigma = 0*.

#### Nhị phân hóa hình ảnh

Nhị phân hóa là quá trình chuyển đổi ảnh xám thành ảnh nhị phân. Ảnh nhị phân là ảnh mà giá trị của các pixel chỉ được biểu diễn bằng 0 (đen) hoặc 255 (trắng). Ảnh nhị phân được áp dụng cho nhiều ứng dụng khác nhau như: phân đoạn ảnh (segmentation), tiền xử lý cho các giải thuật nhận diện chữ viết tay/ chữ số, tạo mặt nạ cho ROI, loại bỏ nền,… Gọi là giá trị pixel của ảnh tại tọa độ , là ngưỡng cho trước, là giá trị pixel của ngõ ra ở tọa độ . Vậy nhị phân hóa thường được áp dụng nhờ công thức:

Giá trị pixel của ảnh ngõ ra sẽ phụ thuộc rất nhiều vào giá trị ngưỡng . Tuy nhiên, không phải bước ảnh nào màu sắc, giá trị biểu diễn, cường độ ánh sáng cũng như nhau. Chính vì vậy nên giá trị của cũng sẽ thay đổi theo nếu ta muốn nổi bật 1 đối tượng cụ thể nào đó. Để có thể phân đoạn đối tượng đạt được kết quả tốt mà không quan tâm đến sự thay đổi của , ta cần áp dụng thuật toán *Adaptive Threshold. Adaptive Threshold* là một thuật toán lấy ngưỡng động, trong đó, ngưỡng xét của 1 pixel trong ảnh đầu vào sẽ được quyết định bằng những giá trị pixel ở một vùng xung quanh.

Trong OpenCV, để ta thực hiện được thuật toán *Adaptive Threshold,* ta sẽ sử dụng câu lệnh:

*void cv::adaptiveThreshold(InputArray src, OutputArray dst, double maxValue, int adaptiveMethod, int thresholdType, int blockSize, double C)* [11]

Trong phạm vi luận văn này ta sẽ sử dụng các giá trị *blockSize* và *C* sẽ được tùy chỉnh theo từng lại đối tượng trong phạm vi luận văn này.



Hình 3.24 Ảnh sau khi nhị phân hóa

#### Thuật toán xác định cạnh \_ Canny Edge Detection

Trong một bức ảnh, cạnh là nơi mã hóa và chứa nhiều thông tin nhất. Nó có thể chứa thông tin về hình dạng, kích thước của một hay nhiều đối tượng. Chính vì vậy, thuật toán phát hiện cạnh là thuật toán được ứng dụng nhiều trong những bài toán detection. Hiện nay có nhiều thuật toán dùng để phát hiện cạnh khác nhau như: *Sobel, Laplacian,…* Trong đó, thuật toán *Canny Edge Detection* được sử dụng nhiều nhất vì tính tiện lợi, nhanh chông và độ chính xác cao. Thuật toán này được thực hiện thông qua 4 bước khác nhau và kết quả cho ra các đường biên khá chính xác và khá mỏng [12]. Áp dụng thuật toán *Canny* vào trong đối tượng trong hình ảnh đã được nhị phân hóa ở bước trên ta sẽ thu được những cạnh viền của đối tượng. Thuật toán *Canny* được sử dụng trong thư viện OpenCV bằng câu lệnh:

*void cv::Canny(Mat img, Mat edges, int lower\_threshold, int upper\_threshold, int aperture)* [13]

Ở đây, ta sử dụng *lower\_threshold=66*, *upper\_threshold=133*, *aperture = (3,3)*.



Hình 3.25 Ảnh sau khi sử dụng thuật toán Canny

#### Phát hiện viền bao (Contour)

Ta có thể hiểu rằng các viền bao (Contour) là các tập điểm liên tục tạo thành các vòng cung (curve or boundary) kín, không hở. Đặc điểm các điểm của một viền bao là có cùng/ xấp xỉ một giá trị màu hoặc một mật độ. Viền bao là một công cụ hữu ích được dùng để phân tích hình dạng đối tượng, phát hiện và nhận dạng đối tượng.

Trong OpenCV, câu lệnh thực hiện phát hiện viền bao là:

*void cv::findContours(InputOutputArray img, OutputArrayOfArrays contours,, int mode, int method)* [14]

Sau khi đã tìm thấy viền bao trong hình ảnh, ta sẽ cho điều kiện: diện tích của viền bao để lọc các nhiễu và chỉ chọn lại phần có viền to là đối tượng cần phân tích. Tiếp theo ta lần lượt thực hiện thao tác tìm viền lồi (Convex Hull) và tìm viền xấp xỉ (Contour Approximation Algorithm) để tìm được hình dạng đa giác chính xác của đối tượng. Câu lệnh thực hiện thuật toán trên là:

*void cv::convexHull(InputArray points, OutputArray hull)* [15]

và

*void cv::approxPolyDP(InputArray contour, OutputArray approxContour ,double epsilon, bool closed true)* [15]



Hình 3.26 Ảnh của đối tượng được xác định viền bao

#### Xác định tọa độ tâm và góc xoay của đối tượng

Sau khi đã xác định được viền bao đa giác của đối tượng, ta sẽ xét điều kiện để tiếp tục phân tích: Nếu viền bao được phát hiện là hình tứ giác thì tiếp tục, nếu không thì sẽ bỏ qua. Khi đã xác nhận viền bao là hình tứ giác, ta sẽ trích xuất tọa độ của 4 góc của viền bao tứ giác. Gọi lần lượt A, B, C, D lần lượt là 4 góc của viền bao. Vậy trọng tâm O của đối tượng được tính bằng công thức:

Khi xác định viền bao, trong nhiều trường hợp, các thuật toán có tìm ra nhiều hơn một viền bao. Điều này không chỉ không làm cho việc xác định tâm bị sai đi mà còn làm cho chính xác hơn. Để làm được điều này,ta cần phải loại nhiễu một cách cẩn thận, đồng thời sau khi xác định tâm của từng viền bao thì cần phải lấy trung bình các tâm đã tìm được để xác định tọa độ tâm cuối cùng:

Đối với góc xoay của đối tượng, đầu tiên ta cần tìm góc xoay của từng viền bao. Sau khi đã có tọa độ của bốn góc của viền bao, ta sẽ tính góc lệch của 4 cạnh của viền bao. Gọi A, B lần lượt là 2 điểm liên tiếp của viền bao, vậy công thức tính góc lệch là:

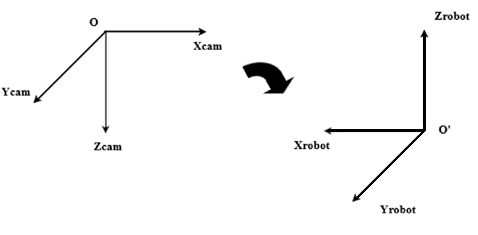
Vì đối tượng có dạng hình vuông nên đối tượng chỉ có thể xoay 1 góc . Vậy giá trị góc xoay cần phải được điều chỉnh như sau:

Khi đã có giá trị góc xoay đã được điều chỉnh, ta sẽ tính trung bình góc xoay của 4 cạnh để ra được giá trị góc xoay của viền bao đó. Và cũng như tính tâm của đối tượng, trong nhiều trường hợp ta sẽ thu được nhiều góc xoay của nhiều viền bao vậy ta chỉ cần lấy trung bình cộng góc xoay của các viền bao xác định được để tính ra góc xoay cuối cùng của đối tượng:



Hình 3.27 Tâm của đối tượng

### Chuyển đổi sang hệ tọa độ robot



Hình 3.28 Chuyển đổi hệ tọa độ camera sang robot

Chuyển hệ tọa độ camera sang hệ tọa độ robot dựa vào công thức sau:

***RobotP = RobotRCam CamP + RobotPCamOrg***

Đầu tiên ta sẽ tìm ma trận xoay ***RobotRcam*** khi ta xoay trục tọa độ của camera sao cho giống với trục tọa độ của robot. Ta thực hiện xoay trục tọa X một góc , sau đó xoay trục Z một góc . Vậy, ma trận xoay ta có được là:

***RobotRcam***

Sau đó ta dịch chuyển hệ tọa độ camera tới vị trí của hệ tọa độ robot. Dựa vào dữ liệu tính toán khoảng cách giữa 2 trục tọa độ, ta sẽ thu được ma trận dịch chuyển:

***RobotPCamOrg***

Vậy ma trận chuyển đổi hệ tọa độ như sau:

### Camera calibration

Một số máy ảnh, đặc biệt là dạng pinhole, có thể tạo ra một sự méo dạng (distortion) đáng kể cho hình ảnh. Có hai dạng méo dạng chính là méo xuyên tâm (radial distortion) và méo tiếp tuyến (tangential distortion).

Méo xuyên tâm làm cho các đường thẳng bị cong và điểm ở càng xa tâm của hình ảnh thì độ méo càng tăng. Tương tự như vậy, hiện tượng méo tiếp tuyến xảy ra khi dải lấy ảnh không được căn chỉnh hoàn toàn song song với mặt phẳng ảnh. Điều này làm cho một số khu vực của ảnh nhìn có vẻ gần hơn bình thường. Độ méo dạng được biểu diễn bằng hệ số méo dạng sau, trong đó là hệ số méo dạng xuyên tâm, là hệ số méo dạng tiếp tuyến:

Việc hiệu chỉnh camera được thực hiện nhằm mục đích làm giảm sự méo dạng từ đó giúp cho hình ảnh đầu ra được chính xác hơn. Tuy nhiên, ngoài các hệ số méo dạng ra thì để có thể thực hiện được hiệu chỉnh loại bỏ các méo dạng ta cần thêm ma trận thông số nội (bao gồm các tiêu cự và tọa độ ) và thông số ngoại. Để có thể tìm được các thông số trên ta sử dụng câu lệnh của thư viện OpenCV với những ảnh mẫu đã chụp sẵn:

[*cv::calibrateCamera*](https://docs.opencv.org/master/d9/d0c/group__calib3d.html#ga687a1ab946686f0d85ae0363b5af1d7b)*(InputArrayofArrays objpoints, InputArrayofArrays imgpoints, Size size, None, None)* [16]

Tuy nhiên, khi hiệu chỉnh thì ta sẽ tìm lại ma trận camera để phù hợp với với các méo dạng rồi tính toán lại hình ảnh. Chính vì vậy, việc hiệu chỉnh toàn bộ hình ảnh sẽ tốn một lượng thời gian đáng kể. Nhưng camera Logitech C270 có độ méo dạng khá nhỏ nên việc thực hiện hiệu chỉnh toàn bộ hình ảnh là không cần thiết. Để tăng hiệu quả của mô hình ta sẽ chỉ hiệu chỉnh một điểm pixel cũng chính là tâm của đối tượng. Để thực hiện điều này ta sử dụng câu lệnh:

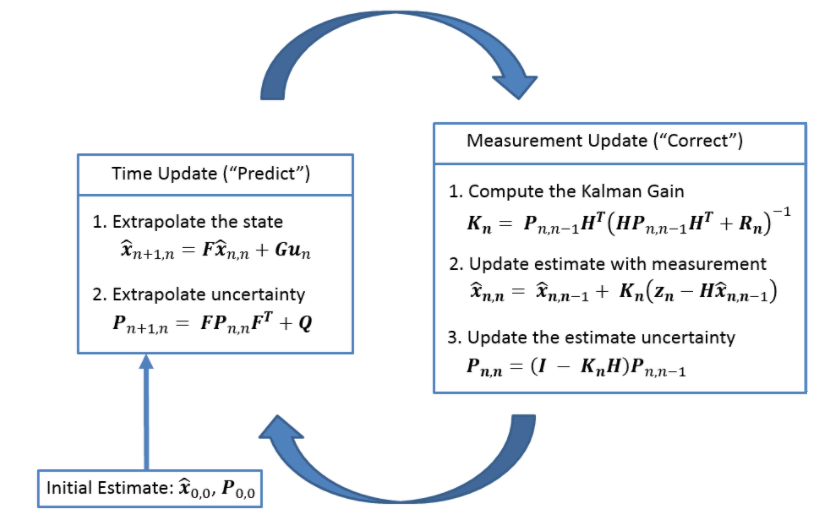
[*cv::undistortPoints*](https://docs.opencv.org/master/d9/d0c/group__calib3d.html#ga55c716492470bfe86b0ee9bf3a1f0f7e)*(*[*InputArray*](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga353a9de602fe76c709e12074a6f362ba)*src,*[*OutputArray*](https://docs.opencv.org/master/dc/d84/group__core__basic.html#gaad17fda1d0f0d1ee069aebb1df2913c0)*dst,*[*InputArray*](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga353a9de602fe76c709e12074a6f362ba)*cameraMatrix,*[*InputArray*](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga353a9de602fe76c709e12074a6f362ba)*distCoeffs,*[*InputArray*](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga353a9de602fe76c709e12074a6f362ba)*R,*[*InputArray*](https://docs.opencv.org/master/dc/d84/group__core__basic.html#ga353a9de602fe76c709e12074a6f362ba)*P)* [16]

### Bộ lọc Kalman

Bộ lọc Kalman là thuật toán sử dụng một chuỗi các giá trị đo lường, bị ảnh hưởng bởi nhiễu hoặc chứa sai số, nhằm đưa ra các ước lượng về biến số từ đó cung cấp một kết quả chính xác hơn khi chỉ sử dụng một kết quả duy nhất. Bộ lọc Kalman áp dụng phương pháp hồi quy đối với các giá trị bị nhiễu nhằm tối ưu hóa các giá trị ước lượng trạng thái của hệ thống.

Bộ lọc Kalman được ứng dụng trong nhiều lĩnh vực khác nhau như: Tracking, Process Control, Location & Navigation System,… Thông thường, bộ lọc Kalman được sử dụng cho mục đích là uớc lượng trạng thái của hệ thống động.

Bộ lọc Kalman được chia làm 2 phần chính [17]. Phần đầu là dự đoán, dựa vào giá trị hiện tại, bộ lọc sẽ dự đoán giá trị tương lại tiếp theo dựa vào phương trình của hệ thống. Phần thứ hai chính là phần cập nhật, sau khi đã có giá trị dự đoán, thuật toán sẽ so sánh nó với giá trị đo được từ đó đưa ra những điều chỉnh phù hợp cho ra kết quả rồi tiếp tục quá trình quay lại với phần đầu. Dưới đây là mô hình của bộ lọc Kalman.



Hình 3.29 Mô hình bộ lọc Kalman

Trong đó:

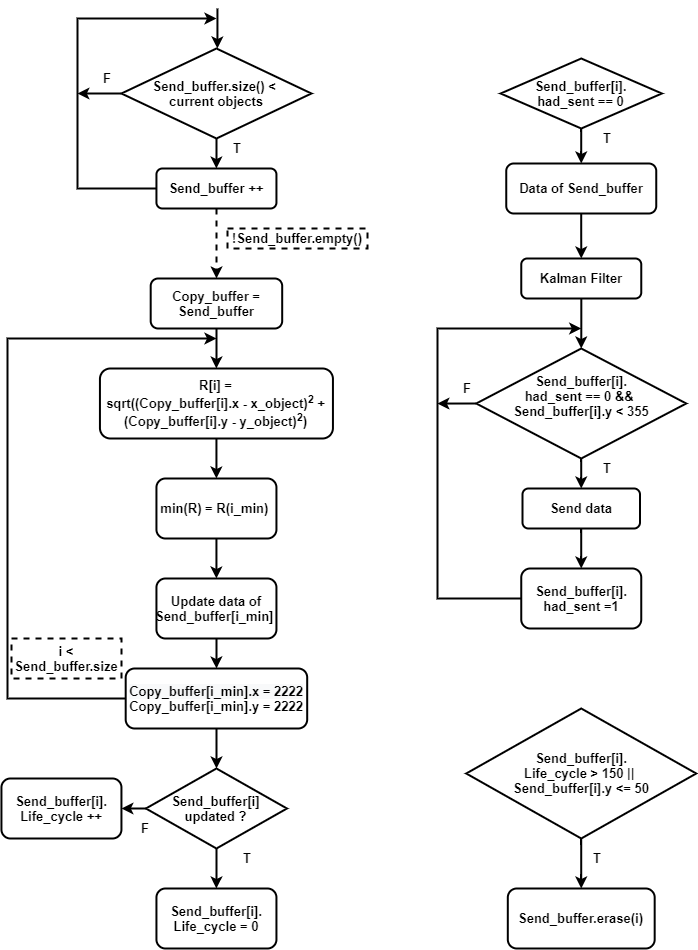
* lần lượt là giá trị ước tính của các biến trạng thái ở thời điểm n sau khi thực hiện phép đo và ở thời điểm n được dự đoán từ trạng thái . Tại phạm vi luận văn này, giá trị của biến là một vector có độ dài là 5, bao gồm các giá trị trạng thái sau: tọa độ , vận tốc và góc xoay . Trong đó được xác định bằng với giá trị đo được ở thời điểm đầu tiên.
* : vector chứa các giá trị trạng thái đo được ở thời điểm *n*. Ở trong luận văn này, vector là một vector có độ dài 3 với các giá trị xác định được từ camera của tọa độ và góc xoay .
* : tín hiệu điều khiển.
* : ma trận hiệp phương sai trong đó mỗi phần tử của ma trận biểu diễn giá trị hiệp phương sai của 2 biến. Ma trận này dùng để xác định sự phân tán của hệ trạng thái tại thời điểm *n.* Ta chọn tại thời điểm đầu tiên.
* : Ma trận biểu diễn mối quan hệ giữa các trạng thái của hệ thống tại thời điểm *n* và *n-1,* được dùng để dự đoán trạng thái tiếp theo. Ta chọn ma trận với giá trị là thời gian sau mỗi vòng lặp như sau:
* : Ma trận biểu diễn mối quan hệ giữa tín hiệu điều khiển và các biến trạng thái. Ta chọn .
* lần lượt là ma trận hiệp phương sai của nhiễu của phần ước lượng và của phần đo lường (Process Noise Covariance Matrix and Measures Noise Covariance Matrix). Ta chọn
* : Ma trận của Observation model, sẽ giúp ta hiệu chỉnh giá trị dự đoán theo giá trị đo được. Vì ta chỉnh đo được biến trong số 5 biến của bộ lọc nên ma trận sẽ có dạng:

Hiện tại, thư viện OpenCV có hỗ trợ mô hình thuật toán của bộ lọc Kalman với class *cv::KalmanFilter* [18]*.* Dựa vào những giá trị mà ta đã xác định trước, ta áp dụng vào thuật toán có sẵn của OpenCV để chạy ra kết quả mong muốn.

### Thuật toán gửi

Vì mục tiêu của luận văn là thực hiện gắp nhiều vật chạy trên băng truyền, tức là sẽ có nhiều vật xuất hiện cùng lúc trên băng truyền. Tuy nhiên, để có thể sử dụng bộ lọc Kalman, đặc biệt là trong việc tracking đối tượng, việc nhập chính xác các giá trị thông tin của từng đối tượng vào đúng bộ lọc của chúng là một việc vô cùng cần thiết. Chính vì vậy, ta cần phải có một thuật toán giúp ta quy hoạch và xác định đúng thông tin đối tượng cho từng bộ lọc, cũng như kiểm soát việc gửi thông tin đó cho robot để gắp vật.

Trong thuật toán gửi này được chia làm 4 phần khác nhau và có thể chạy riêng biệt với nhau bao gồm: Tạo mới buffer, Cập nhật thông tin, Cập nhật giá trị cho bộ lọc Kalman và gửi cho robot, Xóa buffer. Phía dưới là sơ đồ thuật toán gửi của luận văn. Trong đó, *send\_buffer* là một vector 2 chiều với số lượng vector ở chiều đầu tiên thể hiện số lượng đối tượng xuất hiện trong khung hình và ở chiều thứ hai gồm có 7 thông tin của đối tượng. Nó bao gồm thông tin về ID của đối tượng (chạy từ 0 đến 5 tương ứng với 6 loại đối tượng đã quy định), vị trí 3D của đối tượng (), góc xoay , cờ khởi tạo giá trị, cờ đếm số lần không cập nhật và cờ xác nhận gửi. Dưới đây là sơ đồ thuật toán gửi của luận văn.



Hình 3.30 Sơ đồ thuật toán gửi

Ở phần đầu tiên trong thuật toán là tạo buffer mới. Khi xuất hiện đối tượng mới trên khung hình, tức là số đối tượng hiện có lớn hơn số buffer của *Send\_buffer* thì ta sẽ tạo thêm một buffer mới trong vector *Send\_buffer.*

Tiếp theo đó là phần hai của thuật toán. Nếu như *Send\_buffer* không trống thì ta sẽ tạo thêm một vector *Copy\_buffer* giống hệt *Send\_buffer* để thuận tiện cho việc đánh dấu sau này. Tới đây ta sẽ thực hiện cập nhật thông tin cho từng buffer ứng với từng đối tượng. Với mỗi giá trị của đối tượng trong khung hình ta sẽ tính toán sai lệch *R* với từng thông tin có trong từng buffer. Nếu như buffer nào cho giá trị sai lệch nhỏ nhất tức là buffer này thể hiện thông tin của đối tượng đó trong quá khứ. Ta liền cập nhật thông tin hiện tại của đối tượng vào buffer đó trong vector *Send\_buffer* và đồng thời đánh dấu rằng buffer tại đây đã được cập nhật bằng cách đổi giá trị thông tin trong *Copy\_buffer* tại index đó bằng một giá trị lớn để khi tính toán *R* tiếp theo cho ra giá trị cao hơn các số còn lại. Cứ tiếp tục như vậy cho đến khi cập nhật hết cho các đối tượng. Bên cạnh đó, nếu như buffer nào không được cập nhật (tức là đối tượng đã rời khỏi khung hình hoặc bị lỗi nhận diện), ta sẽ cộng số vào cờ *Life\_cycle* của buffer đó. Ngược lại nếu buffer được cập nhật ta sẽ đặt lại giá trị *Life\_cycle* bằng 0.

Phần ba của thuật toán đó là cập nhật giá trị cho bộ lọc Kalman và gửi thông tin cho robot. Đối với phần cập nhật giá trị, ta xét xem giá trị buffer này có gửi cho robot chưa. Nếu rồi thì không cần thiết phải cập nhật giá trị nữa, nếu chưa ta sẽ tiến hành bước tiếp theo. Sau đó, ta kiểm tra cờ khởi tạo, nếu cờ khởi tạo bằng 0, ta sẽ tiến hành khởi tạo giá trị cho bộ lọc Kalman còn nếu bằng 1, ta tiến hành cập nhật vào phần giá trị đo được của bộ lọc. Sau mỗi thời gian quy định, bộ lọc Kalman sẽ tạo ra một giá trị dự đoán. Ta sẽ kiểm tra xem cờ *had\_sent* có bằng 0 và giá trị dự đoán có vượt quá ngưỡng gửi. Nếu thỏa ta sẽ tiến hành gửi thông tin cho robot và đồng thời bật cờ *had\_sent* lên 1.

Phần cuối cùng là xóa buffer. Ta sẽ xét xem nếu *Life\_cycle* lớn hơn ngưỡng cho phép hoặc giá trị *y* của buffer bé hơn 50 tức là đã vượt qua khỏi khung hình. Khi đó ta sẽ xóa buffer khỏi vector *Send\_buffer.*

## SORFTWARE CỦA HỆ THỐNG

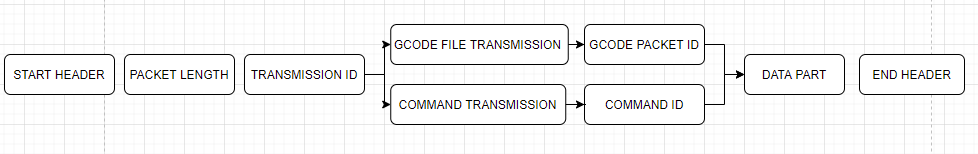
Chương trình điều khiển robot trên máy tính trong luận văn được viết bằng ngôn ngữ C++ trên nền tảng Qt và kèm theo thư viện OpenCV hỗ trợ cho việc xử lí ảnh.

Cấu trúc của chương trình bao gồm 3 luồng chính (thread):

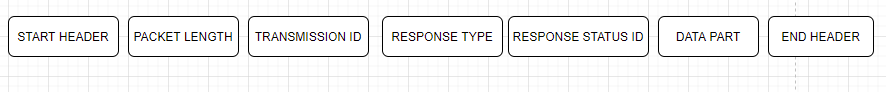
* Main thread: luồng này quản lý giao diện của chương trình điều khiển, giao tiếp với người dùng thông qua các nút nhấn và console, thực hiện các thao tác lệnh và xử lí file.
* Serial thread: luồng này dùng để nhận và xử lí các gói dữ liệu đến và đi từ robot.
* Image processing thread: luồng này dùng riêng cho việc xử lý ảnh từ camera, chạy các thuật toán nhận diện được các vật cũng như toạ độ của chúng để gửi cho robot.

### Protocol giao tiếp giữa máy tính và robot Scara

Các gói tin Transmit và Response được định nghĩa như sau:



Hình 3.31 Cấu trúc gói tin Transmit



Hình 3.32 Cấu trúc gói tin Response

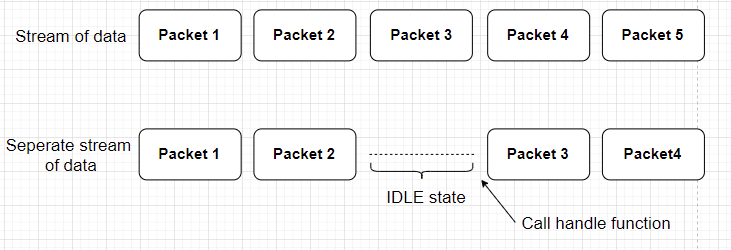
Trong đó:

* Start header và end header đều có kích thước là 2 byte và có giá trị cố định (start header: 0x28 0x7B, end header: 0x7D 0x29).
* Packet length là chiều dài tổng gói tin không bao gồm start header và end header. Packet length chiếm 2 byte.
* Transmission ID có hai kiểu là Command và Gcode file, Command dùng cho những thao tác user command từ máy tính như các lệnh di chuyển, set output, …Theo sau là một command ID. GCode File dùng để phân định các định nghĩa data của file Gcode đã encode gửi từ máy tính. Theo sau là một packet ID. Có tổng cộng 28 command ID và 5 Gcode packet ID.
* Phần data part có chiều dài thay đổi theo từng packet khác nhau.
* Response type sẽ cho máy tính quyết định nên làm gì với gói data này. Có tất cả 9 response type.
* Phần Response status ID hay DTC (diagnostic trouble codes) là phần để báo lên máy tính các trạng thái của robot bao gồm các lỗi từ command hoặc các trạng thái hiện tại của robot. Có tổng cộng 46 response code.

Các số double trong data part thay vì phải gửi tổng thể 8 bytes số chấm động, thì ta có thể scale chúng với một hệ số để đưa về int32\_t từ đó chỉ còn 4-byte cho một số, tăng tính hiệu quả của gói tin truyền.

Trong chương trình sẽ có một hàm để gọi phân loại các packet được gửi tới từ máy tính từ đó thực thi tác vụ của packet đó. Chính vì thế trong một luồng data gửi từ robot đến máy tính ta phải phân tách được các packet rõ ràng từ đó không gây sai sót trong việc truyền dữ liệu.

Chu kì kiểm tra các packet không phụ thuộc vào vào chu kì của thread Serial, nếu làm như vậy sẽ làm tổn hao tài nguyên khác của chương trình điều khiển. Ở đây nếu ta xét buffer Serial Idle trong một khoảng thời gian một gói tin ta sẽ gọi hàm xử lý.

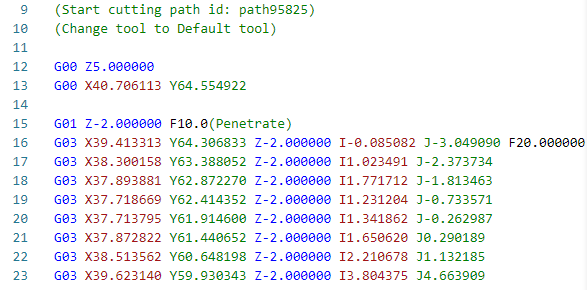


Hình 3.33 Thời điểm gọi hàm xử lý

### Giải mã file Gcode trong ứng dụng CNC routing

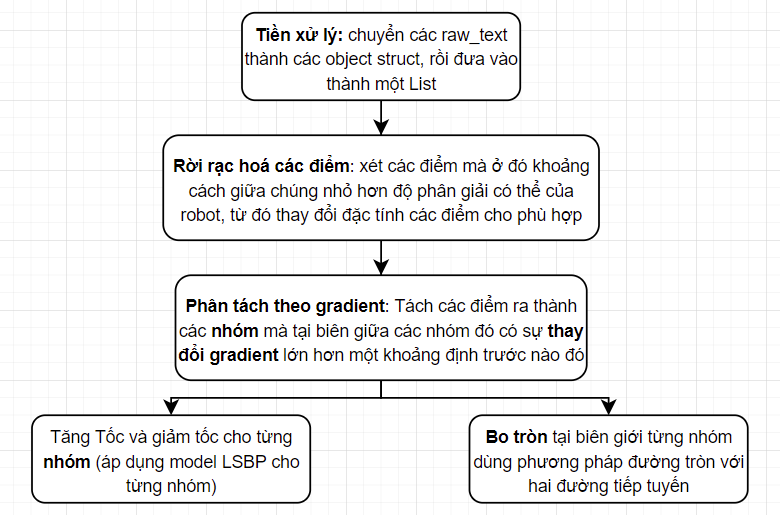
G-Code được hiểu đơn giản là ngôn ngữ lập trình để chỉ thị vị trí cho máy CNC đi đến đâu và cách thức di chuyển. Cấu trúc của một file Gcode thường quay quanh các mã lệnh căn bản như G00, G01, G02, G03, trong đó:

* G00: di chuyển theo đường thẳng
* G01: di chuyển theo đường thẳng nhưng với một federate
* G02: di chuyển theo một cung tròn thuận chiều kim đồng hồ
* G03: di chuyển theo một cung tròn ngược chiều kim đồng hồ



Hình 3.34 Ví dụ cấu trúc file Gcode

Ta sẽ giải mã các file Gcode này trên ứng dụng điều khiển từ đó gửi các toạ độ điểm xuống robot SCARA. Ở đây ta phân ra các bước tổng thể để có thể đưa một file Gcode cho robot SCARA hoạch định theo đúng được quỹ đạo đó.

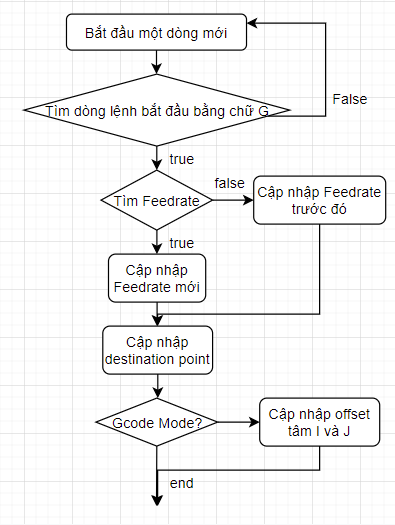


Hình 3.35 Cấu trúc tổng thể xử lý một file Gcode

#### Tiền xử lý

Ở bước này, ta sẽ viết giải thuật để xử lý các chuỗi text này thành các toạ độ với một toạ độ là một struct định nghĩa cho một dòng lênh Gcode, tất cả các object này sẽ chứa trong một ObjectList để chờ xử lý.

Về giải thuật, ta sẽ xét chạy từng dòng trong file Gcode để có thể lấy được data của các câu lệnh.



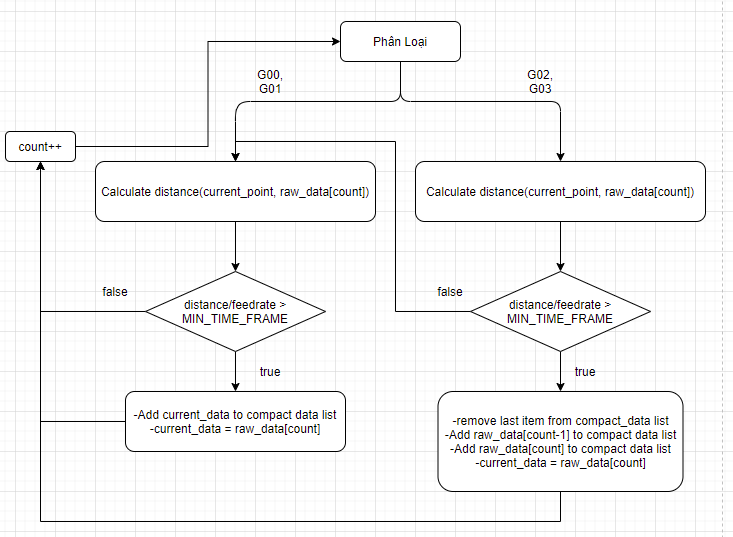
Hình 3.36 Cấu trúc xử lý các chuỗi dòng lệnh từ Gcode

Sau khi giải mã các dòng lệnh về các object để xử lý, ta cũng chưa thể gửi ngay những điểm này cho robot vì những điểm này chỉ mới là hoạch định về hình dạng, các yếu tố quỹ đạo như vận tốc và gia tốc của end effector vẫn chưa được đưa vào xử lý.

#### Rời rạc hoá các điểm

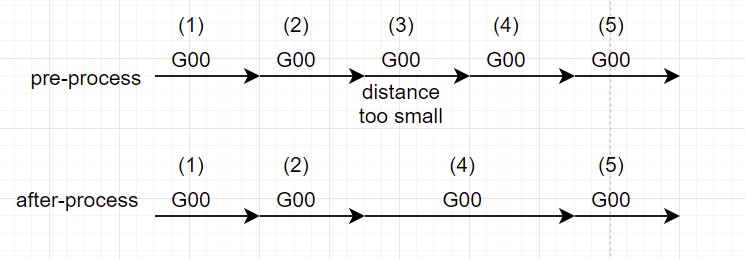
Ta nhận thấy rằng một số khoảng cách giữa các điểm trong file Gcode rất nhỏ so với độ phân giải mà robot có thể cho di chuyển được. Nên nếu vẫn giữ những điểm này và gửi cho robot thì những khoảng thực thi này sẽ không có gây ảnh hưởng cho bố cục toàn bộ của quỹ đạo cần vẽ. Thậm chí chúng còn tăng kích thước tổng thể của gói data khi gửi xuống robot dẫn đến gây tổn hao đường truyền.

Chính vì lý do đó ta sẽ phải lọc bỏ các điểm này đi. Tuy nhiên việc chỉ bỏ các điểm này đi sẽ gây ra các lỗ hỗng trong tập lệnh của file Gcode. Thế nên từ đây ta cần đưa ra các quy luật và thuật xử lý để bù đắp các lỗi này.



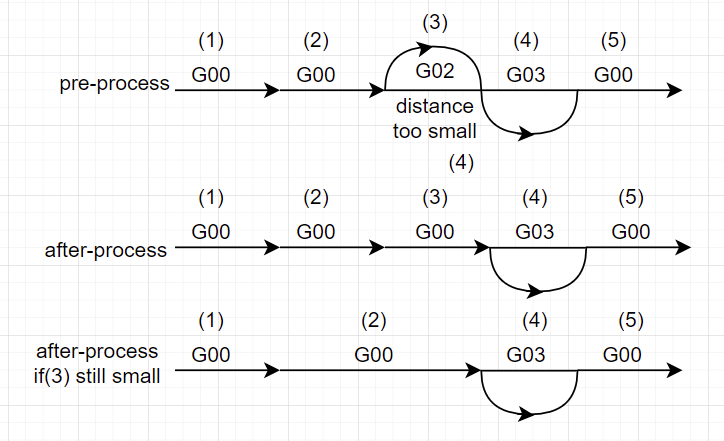
Hình 3.37 Mô hình xử lý phân tách các điểm

Ta sẽ đi qua một số ví dụ ở bên dưới để dễ hình dung hơn một số trường hợp cần chuyển đổi.



Hình 3.38 Biến đổi rời rạc đối với đường thẳng

Ở hình trên câu lệnh G00 (3) được bỏ đi và sau khi thực hiện câu lệnh (2) sẽ đi tiếp tới câu lệnh (4).

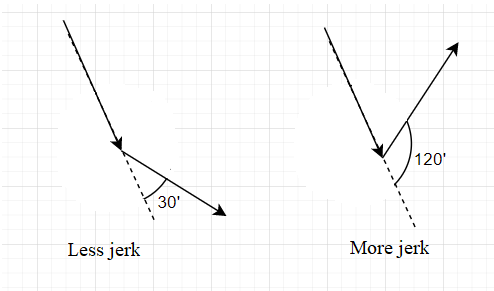
****

Hình 3.39 Biến đổi rời rạc đối với cung tròn

Ví dụ trên cho thấy lệnh G02 (3) đã được chuyển thành lệnh G00, tuy nhiên tại bước này nếu câu lệnh G00 (3) vẫn không đáp ứng về distance sẽ quay về điều kiện như hình ban đầu.

#### Phân tách theo gradient

Đối với các lệnh move Linear, tại các điểm gấp khúc hay tại các điểm mà vector gradient thay đổi đột ngột có thể gây ra jerk lớn ảnh hưởng đến việc quy hoạch cách tay robot.



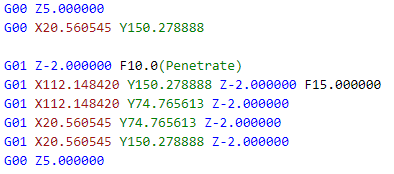
Hình 3.40 Ảnh hưởng của việc thay đổi hướng đột ngột

Chính vì thế ở luận văn này, ta sẽ đưa ra hai hướng giải quyết để xử lý ngay tại các điểm gấp.

* **Áp dụng model LSPB cho từng nhóm đã phân tách**

Với phương pháp này tại mỗi điểm gấp khúc, ta có thể đưa vận tốc end effector về không. Tránh được tình trạng xuất hiện các gai nhọn ngay tại điểm gấp khúc. Tuy nhiên nhược điểm của phương pháp này sẽ làm tăng thời gian để thực thi một file Gcode. Các bước thực hiện được trình bày như sau:

* Phân tách các điểm đã rời rạc hoá thành các nhóm khác Z (gọi là nhóm alpha).
* Trong từng nhóm đã tách ở trên, tách các nhóm này ra thành các nhóm nhỏ nữa mà denta gradient của từng điểm nhỏ hơn một mức setting trước (gọi là nhóm beta).
* Trong từng nhóm beta này, tính tổng độ dài của các thành phần điểm từ đó ta sẽ quy hoạch được các thông số của model LSPB của từng nhóm beta này gửi cho robot.



Hình 3.41 File Gcode vẽ hình chữ nhật

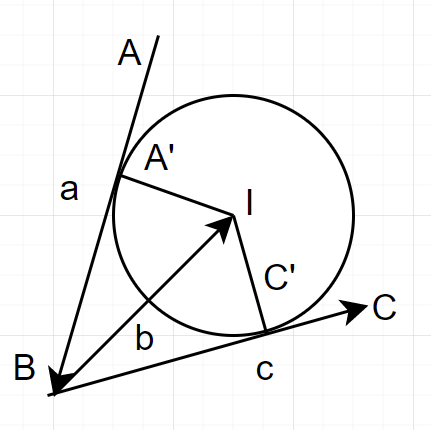
Ví dụ như đoạn Gcode vẽ một hình vuông như ở trên. 4 cạnh của hình vuông sẽ được chia làm 4 nhóm beta cho nhóm alpha thứ 2. Và có tổng cộng 3 nhóm alpha với Z khác nhau. Suy ra với trường hợp này ta phân ra 7 nhóm cần quy hoạch LSPB.



Hình 3.42 Các nhóm quy hoạch đã được phân tách

* **Phương pháp đường cung tròn với 2 tiếp tuyến**

Tại mỗi điểm gấp khúc ta đặt một cung tròn với bán kính R tiếp tuyến với hai đường thằng. Điều này sẽ làm cho vector vận tốc của end effect tránh thay đổi hướng một cách đột ngột, từ đó giảm tình trạng gai nhọn tại các điểm gấp khúc. Tuy nhiên hình vẽ quy hoạch sẽ không còn giống với hình vẽ ban đầu như đã thiết kế.



Hình 3.43 Bổ sung cung tròn tiếp tuyến với hai đường thẳng

Với phương pháp này điều ta cần là loại bỏ được điểm B, thêm vào điểm A’ và C’, tìm được tâm I để tạo câu lệnh G02 hoặc G03 cho chúng tuỳ vào trường hợp. Các bước của phương pháp này được tiến hành như sau:

* Gọi góc giữa a và c là , gradient của các vector AB, BI, BC lần lượt là , , . là một trọng số giới hạn từ 0 tới 1.
* Tính lần lượt các góc , , .
* Xét độ lớn của hai vector AB và BC, gọi L là đại diện cho độ lớn của vector nhỏ hơn.
* Ta tính a = c = L – L.
* Ta tính được độ lớn của đường phân giác
* Tâm I được tính bằng
* Điểm A’ được tính bằng
* Điểm C’ được tính bằng

Ngoài ra phương pháp này cũng sẽ làm tăng kích thước số điểm của file Gcode so với mặc định, càng nhiều điểm phân tách thì số lượng điểm cũng càng tăng. Làm giảm thiểu hiệu suất truyền file.

### Cấu trúc giao diện chương trình

Giao diện chương trình có chức năng cung cấp môi trường cho người sử dụng giám sát và điều khiển, theo dõi khi robot hoạt động.

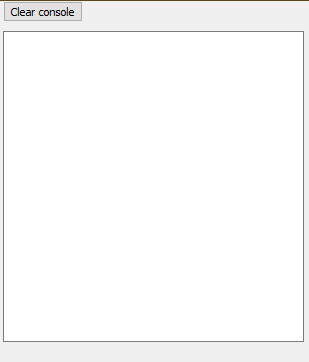
Giao diện chương trình được phân bổ với bố cục như sau:



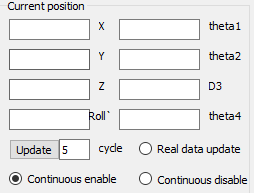
Hình 3.44 Bố cục của giao diện chương trình

Trong đó:

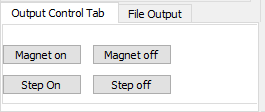
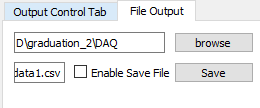
* Serial port configuration là phần lựa chọn port để kết nối với máy tính 
* Program console là chỗ để log các trạng thái của hiện tại của chương trình cũng như các respond từ robot.



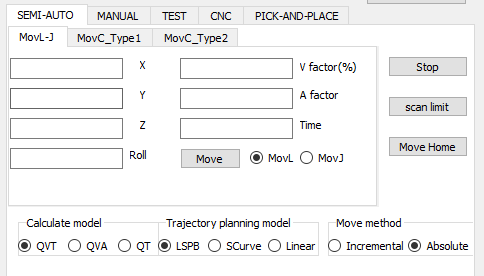
* Scara position data là phần hiển thị các thông số về vị trí hiện tại của robot.



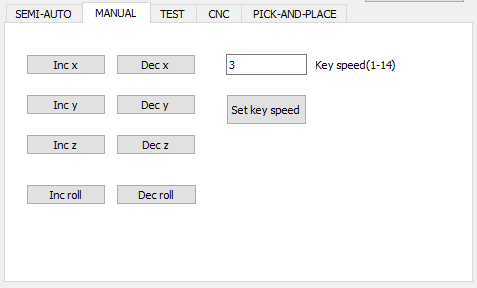
* Output control tab là phần để bật tắt bằng tay các DO trên robot, đồng thời có phần xuất dữ liệu thu thập được về vị trí ra file.

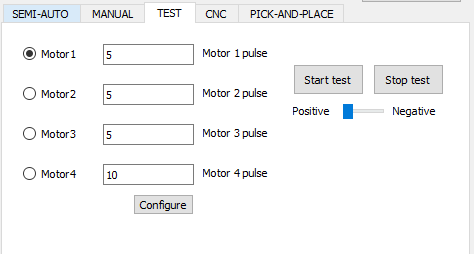
* Phần Scara control panel là phần chứa các chức năng điều khiển chính cho robot Scara. Có tất cả 5 method điều khiển chính cho robot SCARA.



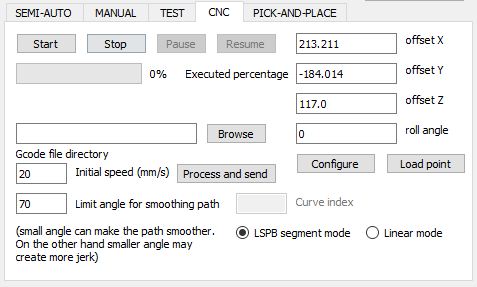
Method SEMI-AUTO: chứa giao diện để ta điều khiển robot với các lệnh chính là movL, movC, movJ theo các model LSPB. Ngoài ra còn các phím chức năng để scan limit, move home cho robot SCARA.



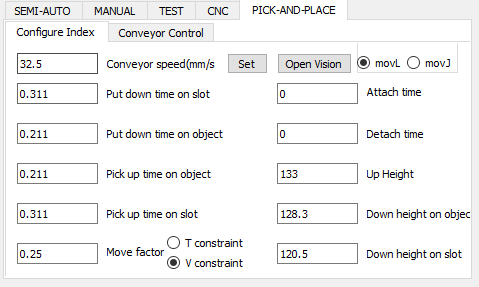
Method MANUAL: có các nút chức năng để ta cho robot di chuyển tịnh tiến theo các trục toạ độ điều khiển được của robot SCARA.



Method TEST: để test việc đưa xung đến các trục xoay của các động cơ.



Method CNC: method này chứa các thông số và phím chức năng cho phép ta cấu hình một file Gcode để gửi xuống robot SCARA.



Method PICK-AND-PLACE: method cho phép ta cấu hình các thông số của mô hình gắp vật động trên băng chuyền để gửi xuống robot SCARA.

# KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ

## Kết quả điều khiển robot.

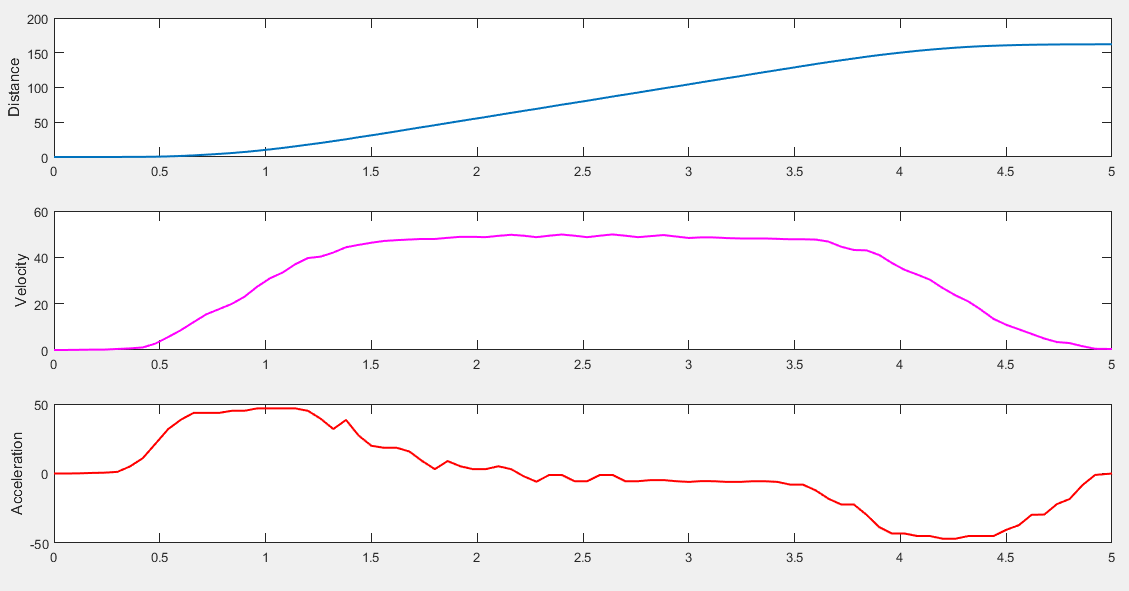
Robot có thể di chuyển theo 3 phương thức: di chuyển trong miền biến khớp (move joint), di chuyển theo đường thẳng trên miền không gian (move line), di chuyển theo cung tròn trên mặt phẳng XY trong miền không gian (move circle) với 2 loại quỹ đạo là LSPB và S-curve.

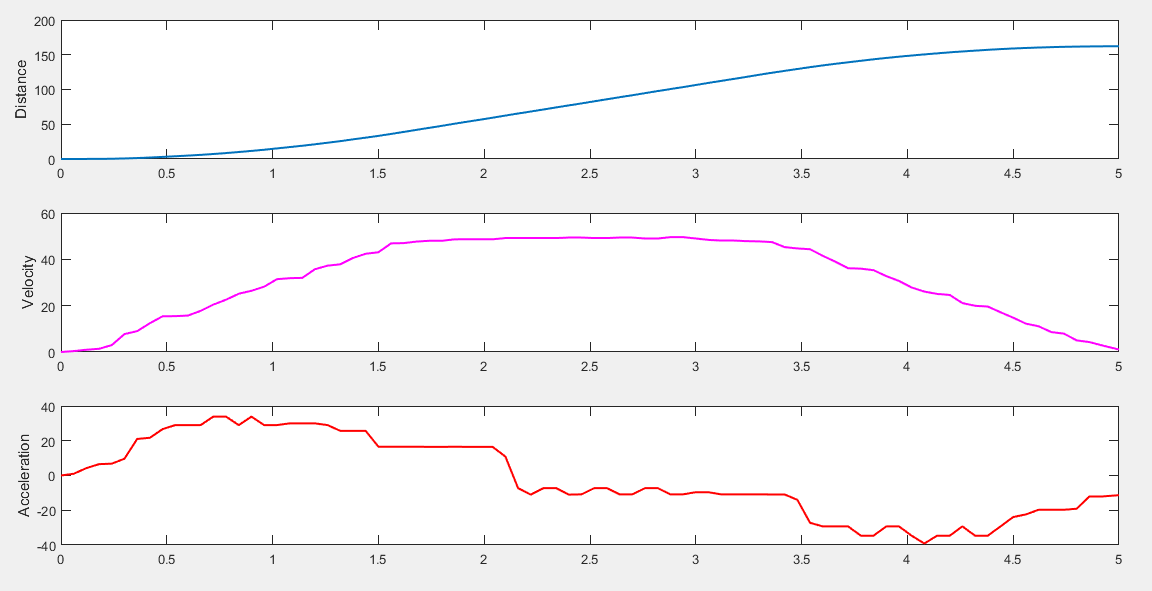
Vị trí đầu công tác trong phần này được xác định một cách gián tiếp thông qua việc đo vị trí của các biến khớp và sử dụng phương trình động học thuận để tính ra vị trí hiện tại của đầu công tác.

### Kết quả điều khiển dùng model LSPB và Scurve

Ta thu thập dữ liệu khi cho robot di chuyển từ điểm x=200, y = -75, z = 133.32 cho tới điểm x=250, y=75, z=90

Đồ thị quãng đường, vận tốc và gia tốc được vẽ trong matlab như sau:





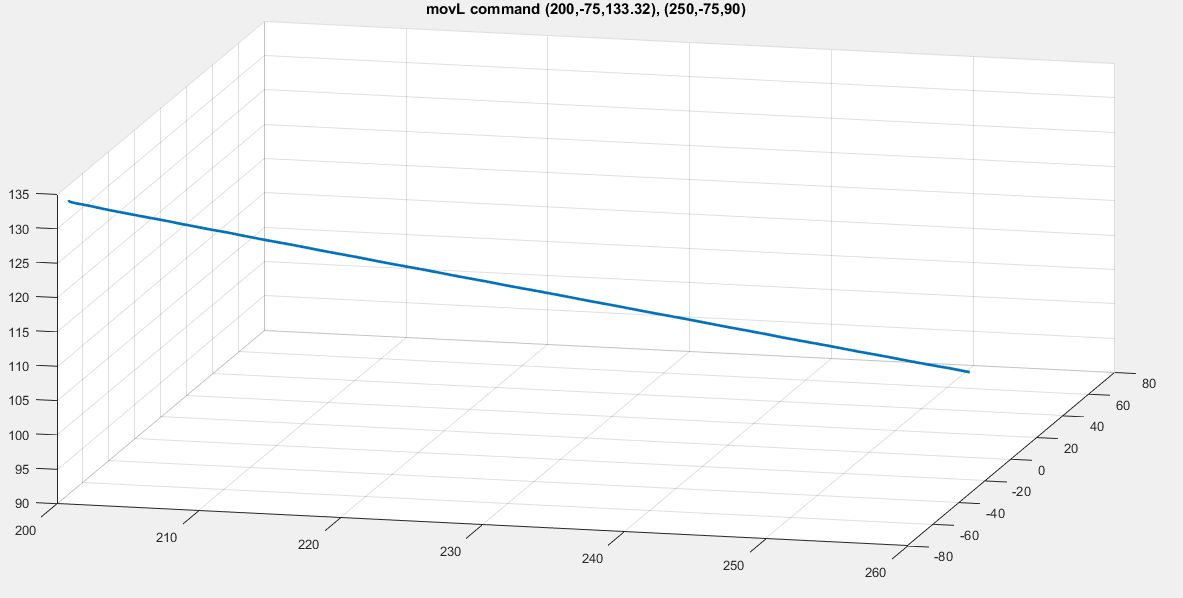
Hình 4.1 Đồ thị quãng đường, vận tốc và gia tốc

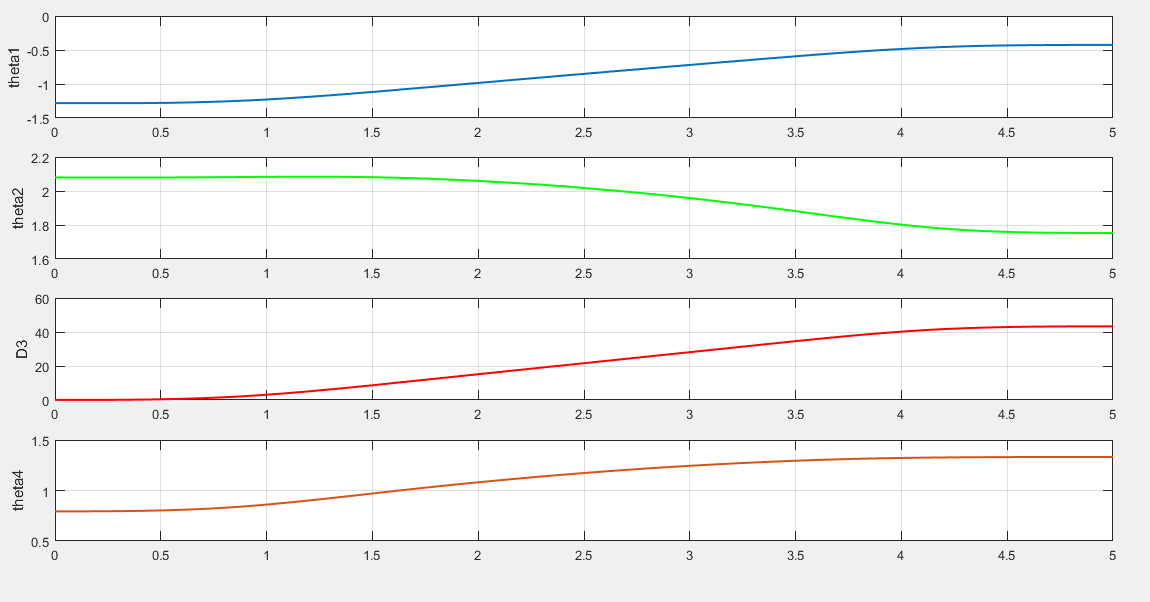
Ta nhận thấy dữ liệu thu thập được từ quỹ đạo robot di chuyển cho ra xấp xỉ với tính toán của lý thuyết đã thiết kế.

### Kết quả điều khiển với 3 phương thức di chuyển (movL, movC, movJ)

#### Kết quả điều khiển với lệnh movL

Ta thu thập dữ liệu khi cho robot di chuyển từ điểm x=200, y = -75, z = 133.32 cho tới điểm x=250, y=75, z=90. Dưới đây là biểu đồ kết quả:

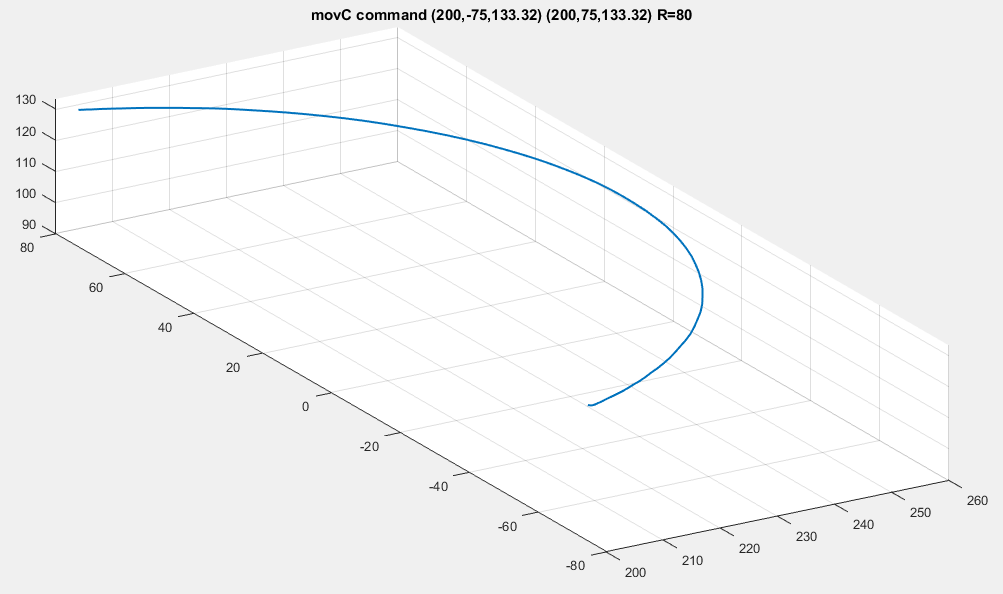
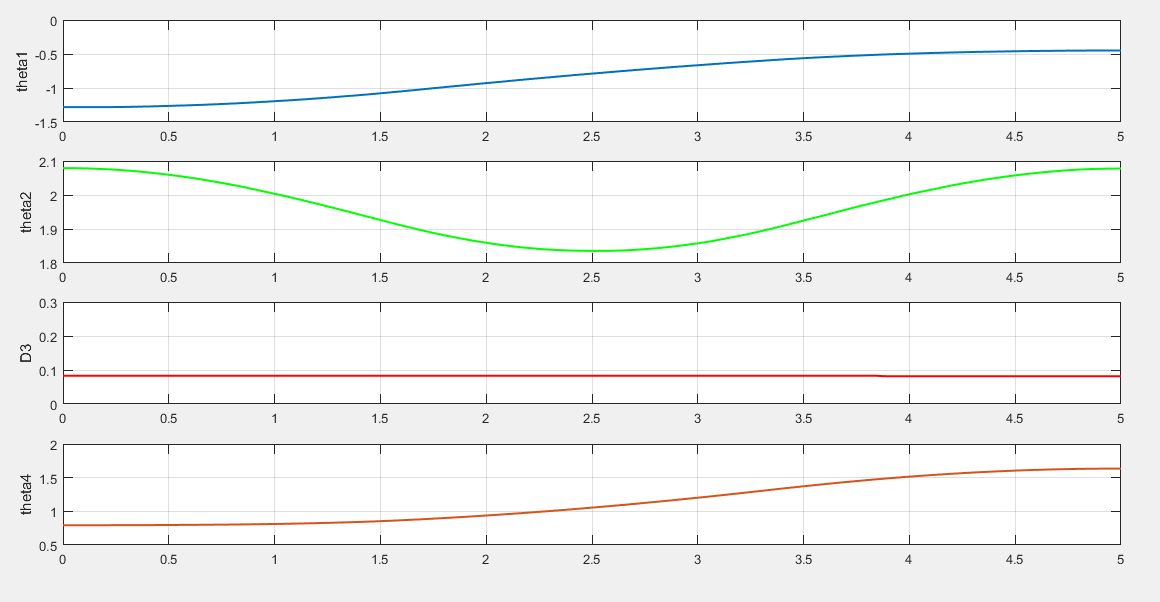




Hình 4.2 Biểu đồ kết quả điều khiển với lệnh movL

#### Kết quả điều khiển với lệnh movC

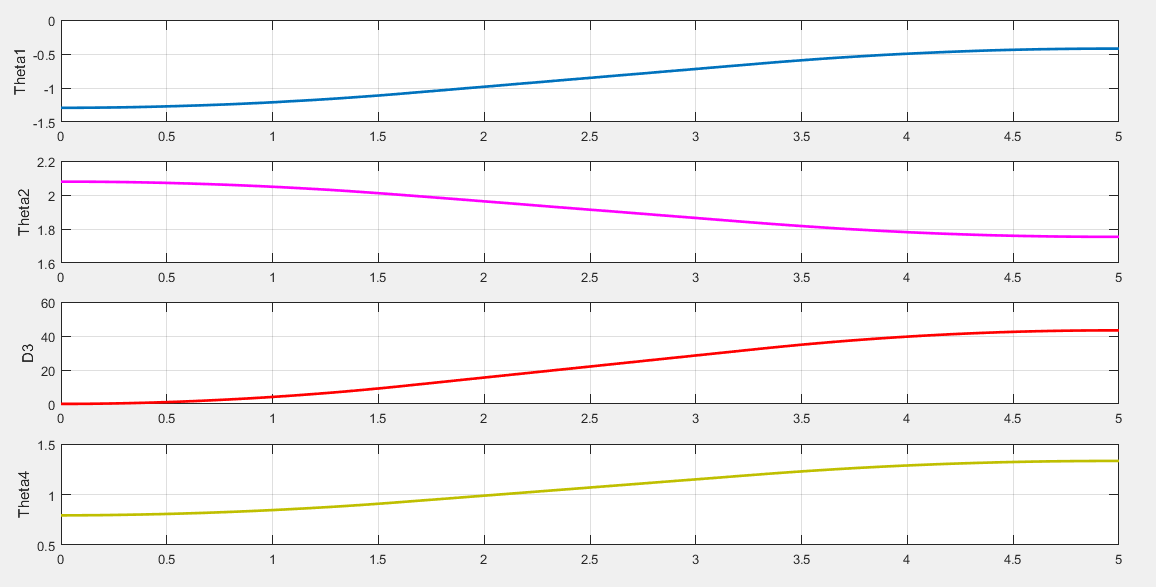
Ta thu thập dữ liệu khi cho robot di chuyển từ điểm x=200, y = -75, z = 133.32 cho tới điểm x=200, y=75, z=133.32 với R = 80. Dưới đây là kết quả thực hiện.

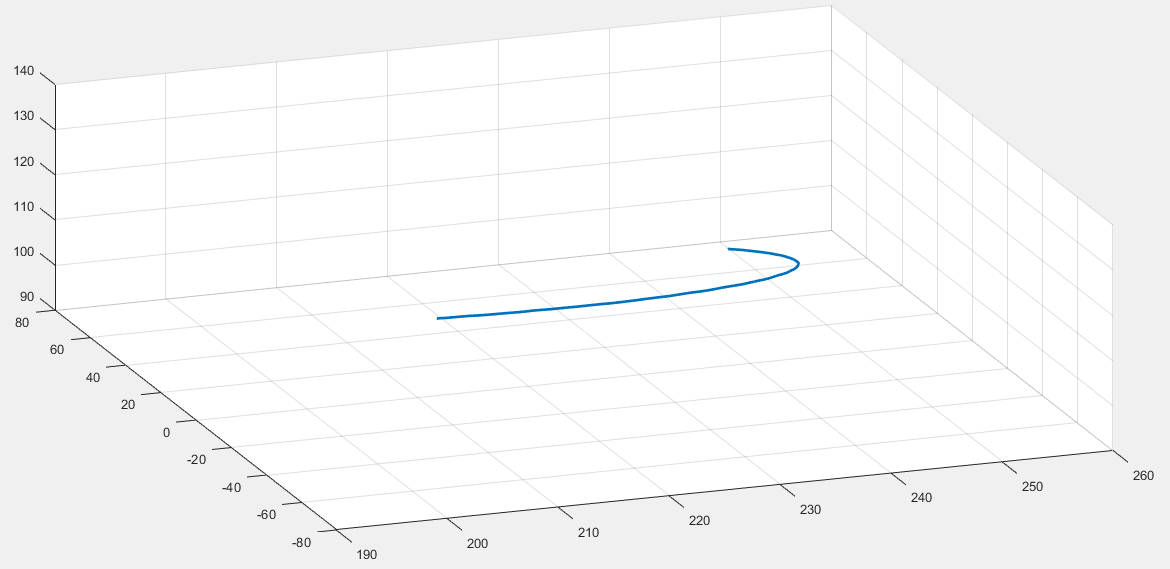
** **

Hình 4.3 Biểu đồ kết quả điều khiển với lệnh movC

#### Kết quả điều khiển với movJ

Ta thu thập dữ liệu khi cho robot di chuyển từ điểm x=200, y = -75, z = 133.32 cho tới điểm x=250, y=75, z=90. Sau đây là kết quả thực hiện.

****

****

Hình 4.4 Biểu đồ kết quả điều khiển với lệnh movJ

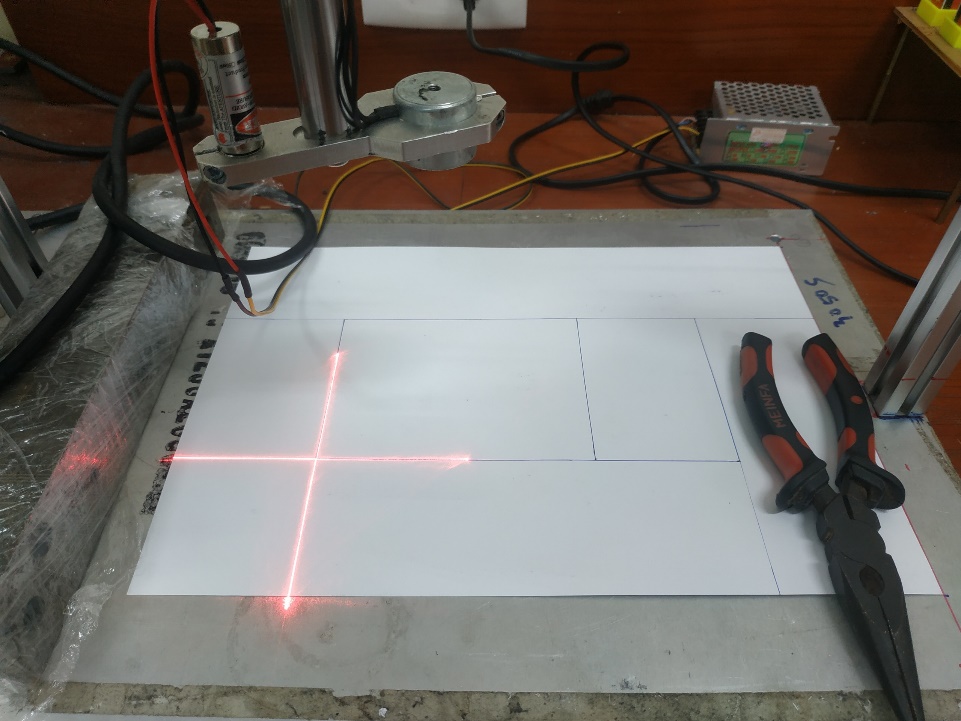
**Đánh giá:** Dựa vào các dữ liệu đã thu thập ở phía trên, ta thấy rằng quỹ đạo của movL, movC và movJ đều đạt được những kết quả mong muốn tương tự như lý thuyết đã quy hoạch.

### Đánh giá độ chính xác của robot

Để đánh giá độ chính xác của robot di chuyển trong không gian 3D, ta sẽ sử dụng phương pháp sau.

Ta sử dụng một miếng giấy A4 vẽ 4 góc của một hình chữ nhật, ta dùng các dụng cụ thước sắt để canh chỉnh tờ giấy đó vào trên mặt base của robot và biết được toạ độ của 4 điểm đó. End effector của robot được gắn một đầu lazer để xác định được toạ độ x và y của robot trên mặt giấy.

Cho robot di chuyển lần lượt qua 4 điểm đó, và tại mỗi điểm ta sẽ dùng thước sắt đo xem đầu lazer đó lệch so với 1 góc của hình chữ nhật một khoảng x và y bao nhiêu, từ đó đánh giá sai số.



Hình 4.5 Đánh giá độ chính xác của robot

Để tính độ chính xác về vị trí

Sai số vị trí:

Bảng 4.1 Sai số vị trí của robot SCARA

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Số lần | Vị trí mục tiêu | | Vị trí robot đạt được | | Sai số | | |
| X(mm) | Y(mm) | X(mm) | Y(mm) |  |  |  |
| 1 | 244.209 | -39.6372 | 243.583 | -39.608 | 0.626 | -0.0292 | 0.626681 |
| 2 | 244.209 | -39.6372 | 243.128 | -40.284 | 1.081 | 0.6468 | 1.259727 |
| 3 | 244.209 | -39.6372 | 243.678 | -38.321 | 0.531 | -1.3162 | 1.419276 |
| 4 | 244.209 | -39.6372 | 244.774 | -38.679 | -0.565 | -0.9582 | 1.112372 |
| 5 | 244.209 | -39.6372 | 243.102 | -38.178 | 1.107 | -1.4592 | 1.831588 |
| 6 | 244.209 | 50.3628 | 243.123 | 48.458 | 1.086 | 1.9048 | 2.192637 |
| 7 | 244.209 | 50.3628 | 243.542 | 48.984 | 0.667 | 1.3788 | 1.531659 |
| 8 | 244.209 | 50.3628 | 243.367 | 48.184 | 0.842 | 2.1788 | 2.335837 |
| 9 | 244.209 | 50.3628 | 243.223 | 49.009 | 0.986 | 1.3538 | 1.674805 |
| 10 | 244.209 | 50.3628 | 243.285 | 48.789 | 0.924 | 1.5738 | 1.824999 |
| 11 | 361.209 | 50.3628 | 363.178 | 51.356 | -1.969 | -0.9932 | 2.205313 |
| 12 | 361.209 | 50.3628 | 362.842 | 51.487 | -1.633 | -1.1242 | 1.982553 |
| 13 | 361.209 | 50.3628 | 363.156 | 51.213 | -1.947 | -0.8502 | 2.124535 |
| 14 | 361.209 | 50.3628 | 363.351 | 51.489 | -2.142 | -1.1262 | 2.420019 |
| 15 | 361.209 | 50.3628 | 362.854 | 51.984 | -1.645 | -1.6212 | 2.309613 |
| 16 | 361.209 | -39.6372 | 362.654 | -38.186 | -1.445 | -1.4512 | 2.047927 |
| 17 | 361.209 | -39.6372 | 362.152 | -38.469 | -0.943 | -1.1682 | 1.501313 |
| 18 | 361.209 | -39.6372 | 363.12 | -38.112 | -1.911 | -1.5252 | 2.445027 |
| 19 | 361.209 | -39.6372 | 362.481 | -38.748 | -1.272 | -0.8892 | 1.551986 |
| 20 | 361.209 | -39.6372 | 363.005 | -38.984 | -1.796 | -0.6532 | 1.911096 |
| Sai số RMSD | | | | |  | | |

Ngoài các đặc tính di chuyển trong không gian bao gồm x, y, z thì robot Scara còn một thông số về góc roll, do tia lazer này gồm có hai đường vuông góc nhau nên ta có thể dễ dàng canh chỉnh thước đo độ để đánh giá góc roll.

Bảng 4.2 Sai số góc roll của robot SCARA

|  |  |  |  |
| --- | --- | --- | --- |
| Số lần | Roll mục tiêu (độ) | Roll đạt được (độ) | (độ) |
| 1 | -45 | -43.158 | -1.842 |
| 2 | -45 | -43.785 | -1.215 |
| 3 | -45 | -44.005 | -0.995 |
| 4 | -45 | -43.365 | -1.635 |
| 5 | -45 | -43.897 | -1.103 |
| 6 | 0 | 1.231 | -1.231 |
| 7 | 0 | 1.365 | -1.365 |
| 8 | 0 | 1.587 | -1.587 |
| 9 | 0 | -1.698 | 1.698 |
| 10 | 0 | -0.741 | 0.741 |
| 11 | 45 | 44.123 | 0.877 |
| 12 | 45 | 44.632 | 0.368 |
| 13 | 45 | 46.982 | -1.982 |
| 14 | 45 | 44.741 | 0.259 |
| 15 | 45 | 44.024 | 0.976 |
| 16 | 90 | 90.785 | -0.785 |
| 17 | 90 | 88.325 | 1.675 |
| 18 | 90 | 88.485 | 1.515 |
| 19 | 90 | 88.348 | 1.652 |
| 20 | 90 | 88.842 | 1.158 |
| Sai số RMSD | |  | |

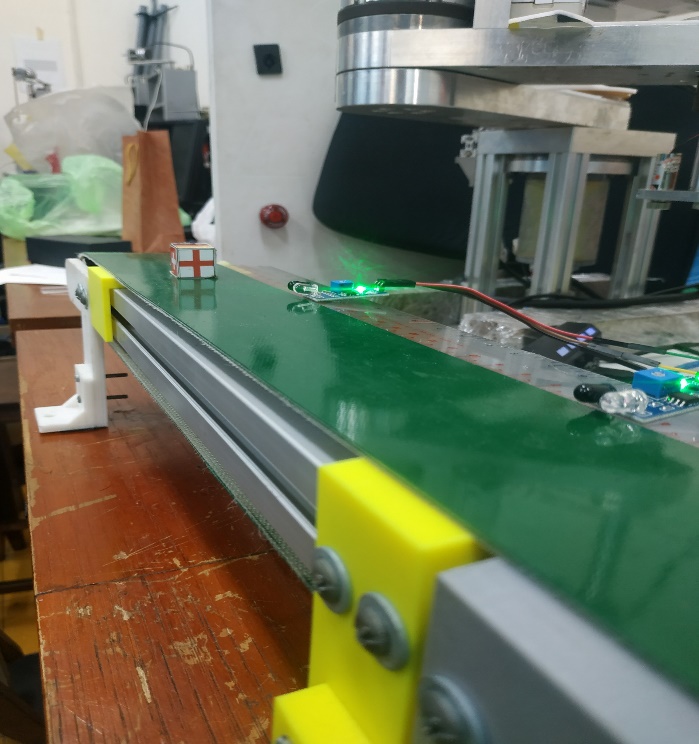
**Nhận xét:** những sai số đã đánh giá trên đến từ nhiều nguyên nhân, sai số vị trí góc quay của các driver, sai số trong mode scan với các limit switch, sai số trong mô hình cơ khí.

### Đánh giá độ chính xác trong điều khiển tốc độ băng chuyền

Ta gắn hai cảm biến hồng ngoại trên một cây thước, và tinh chỉnh hai đầu thu cách nhau 15mm, từ đó đặt thiết bị này song song với băng chuyền sao cho khi một vật đi qua một cảm biến có thể phát hiện được vật đó.

Phương pháp đo rất đơn giản, ta đặt băng chuyền ở những mức vận tốc khác nhau trên lý thuyết. Sau đó khi tốc độ băng chuyền đã ổn định ta đặt một vật lên băng chuyền để vật đi qua cảm biến thứ 1, lúc này vi điều khiển sẽ nhận biết được và bắt đầu đếm timer cho đến khi vật này đi qua cảm biến thứ 2. Ta sẽ có được thời gian vật di chuyển trong khoảng 150mm.

Từ thời gian đó ta sẽ dễ dàng tính được vận tốc đo đạc bằng cách lấy 150 chia cho khoảng thời gian đó.



Hình 4.6 Sử dụng cảm biến quang đo bằng truyền

Bảng 4.3 Sai số tốc độ băng truyền

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Số lần | Tốc độ lý thuyết | Tốc độ đo |  |  |
| 1 | 29.275 | 30.000 | 0.724 | 0.52461 |
| 2 | 29.275 | 29.940 | 0.664 | 0.441427 |
| 3 | 29.275 | 29.585 | 0.310 | 0.096162 |
| 4 | 29.275 | 29.644 | 0.368 | 0.135866 |
| 5 | 29.275 | 29.880 | 0.604 | 0.365783 |
| 6 | 29.275 | 29.940 | 0.664 | 0.441427 |
| 7 | 43.913 | 44.117 | 0.204 | 0.041616 |
| 8 | 43.913 | 44.378 | 0.465 | 0.216318 |
| 9 | 43.913 | 44.247 | 0.334 | 0.11169 |
| 10 | 43.913 | 44.117 | 0.204 | 0.041616 |
| 11 | 43.913 | 44.117 | 0.204 | 0.041616 |
| 12 | 43.913 | 44.510 | 0.596 | 0.35617 |
| 13 | 58.551 | 57.471 | -1.08 | 1.166616 |
| 14 | 58.551 | 57.034 | -1.5172 | 2.301896 |
| 15 | 58.551 | 57.251 | -1.299 | 1.6887 |
| 16 | 58.554 | 57.034 | -1.517 | 2.301896 |
| 17 | 58.551 | 55.970 | -2.581 | 6.66311 |
| 18 | 58.551 | 57.692 | -0.859 | 0.738053 |
| 19 | 73.189 | 70.422 | -2.766 | 7.655182 |
| 20 | 73.189 | 70.422 | -2.766 | 7.655182 |
| 21 | 73.189 | 70.422 | -2.766 | 7.655182 |
| 22 | 73.189 | 68.493 | -4.696 | 22.05336 |
| 23 | 73.189 | 68.807 | -4.382 | 19.20192 |
| 24 | 73.189 | 69.444 | -3.7449 | 14.02428 |
| Sai số RMSD | |  | | |

(đơn vị: mm/s)

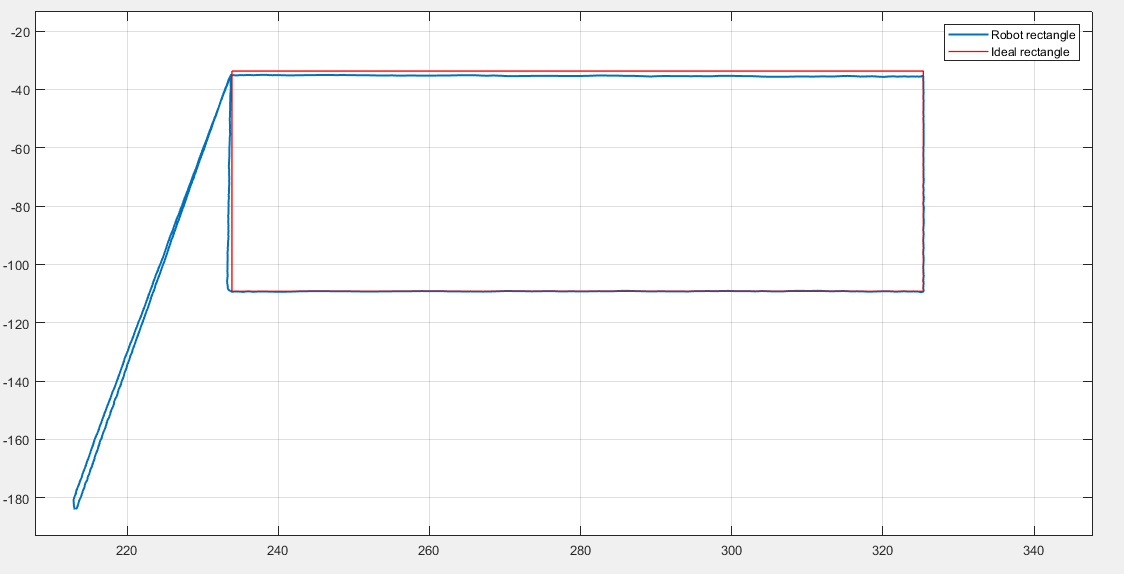
**Nhận xét:** sai số băng chuyền đến từ sai số trong mô hình cơ khí, trong cơ cấu truyền động băng tải, sai số trong độ trượt của dây đai.

### Kết quả CNC routing

Chất lượng của một quá trình cnc routing có thể được đánh giá bằng việc hình dạng bởi robot vẽ ra có càng khớp với model thiết kế hay không. Đối với những model file Gcode thông dụng thì số lượng điểm rất nhiều và hình dạng cũng phức tạp gây khó khăn để đánh giá sai số.

Chính vì thế ở đây ta sẽ dùng một file Gcode để vẽ một hình chữ nhật, từ đó có thể đơn giản trong việc đánh giá sai số hơn. Hình chữ nhật đây có 4 đỉnh với toạ độ lần lượt là (233.871, -33.735); (325.359, -33.735); (325.359, -109.248) (233.871, -109.248).

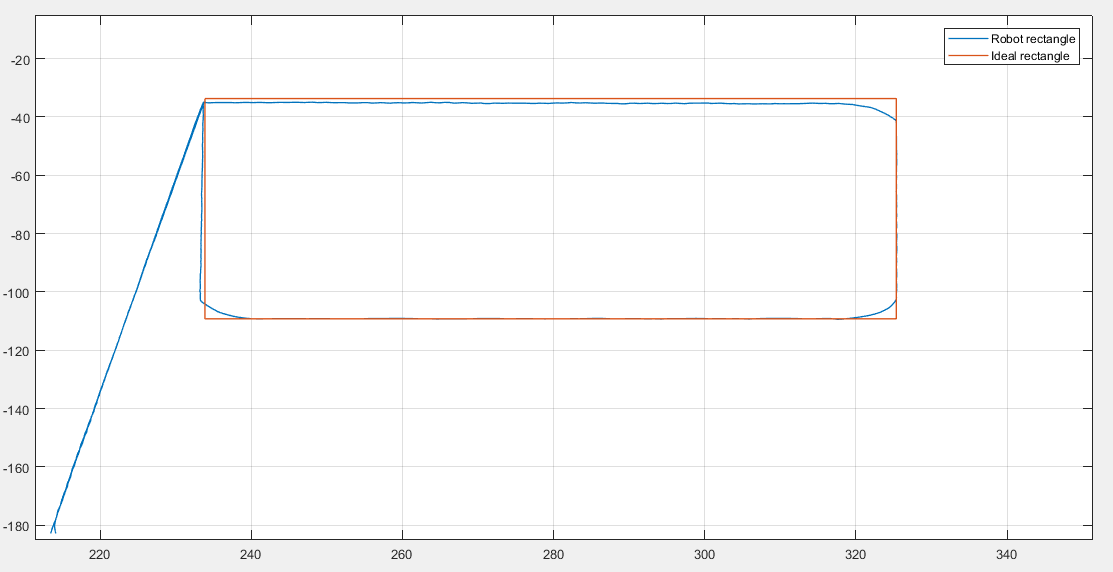
Đối với việc áp dụng LSPB cho từng nhóm phân tách ta có kết quả sau. Model được chạy với góc phân tách là .



Hình 4.7 Quỹ đạo của phương pháp LSPB cho từng nhóm phân tách

Ta cũng tính được RMSD (Root mean square deviation) của model trên từ dữ liệu trong file excel đã thu thập, tuy nhiên bảng dữ liệu dài nên không được hiển thị vào đây.

Đối với việc phân tách và bo tròn tại các góc cạnh ta có kết quả sau. Model được chạy với góc phân tách là và hệ số bo tròn .



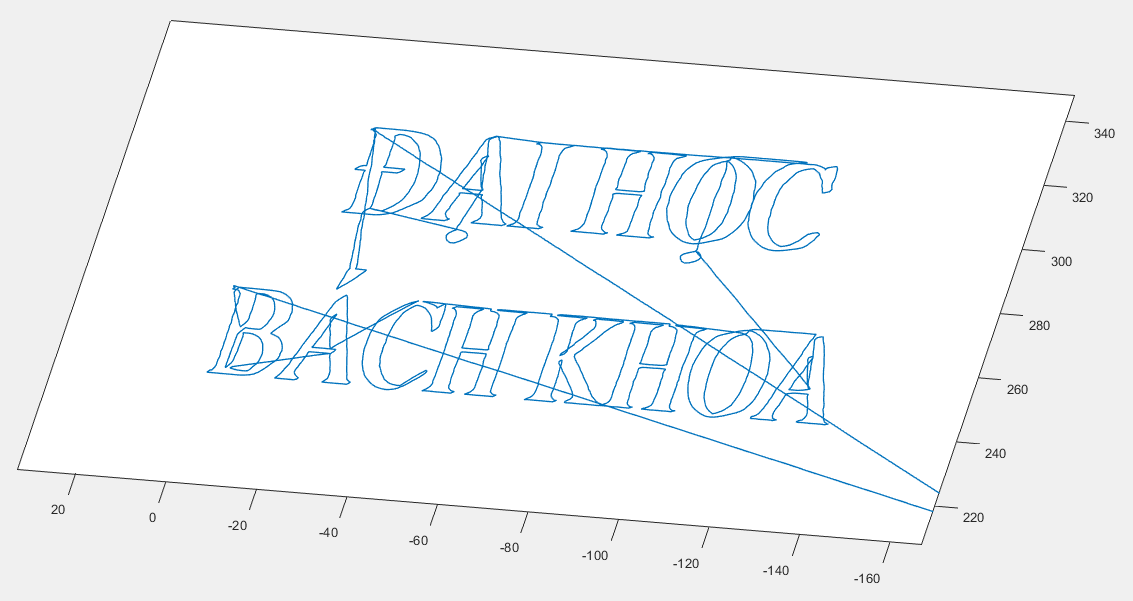
Hình 4.8 Quỹ đạo của phương pháp phân tách và bo tròn tại các góc cạnh

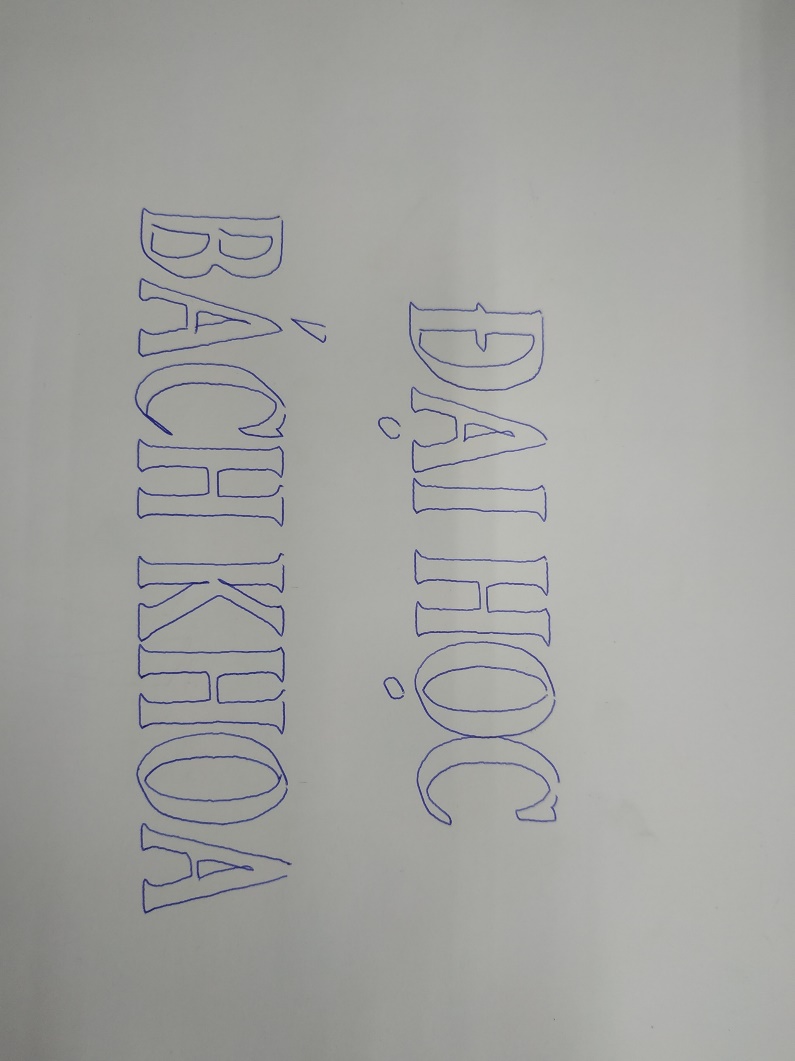
Tương tự như trên, bảng dữ liệu dài nên không được hiển thị vào đây

**Nhận xét:** ta có thể thấy sai số RMSD của model thứ hai lớn hơn nhiều so với model thứ nhất, đây cũng là điều đương nhiên bởi vì quỹ đạo lý tưởng ban đầu ở model 2 đã được thay đổi.

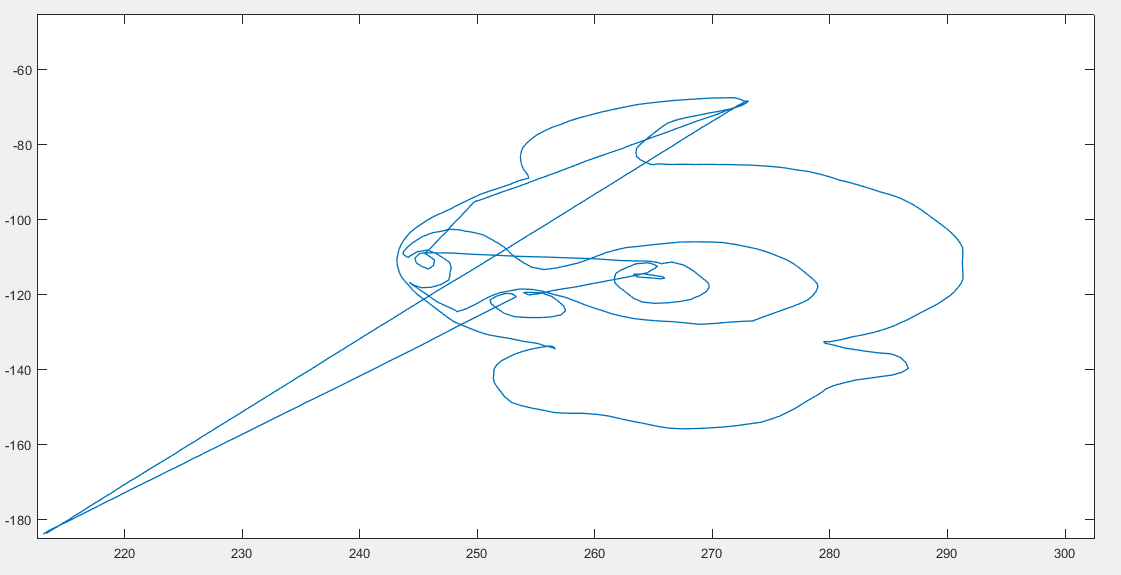
Sai số sẽ tập trung tại cái góc bo tròn, càng nhiều góc bo sai số càng lớn, tăng hệ số bo tròn có thể giảm thiểu được sai số này, tuy nhiên hệ số càng gần 1 thì jerk tại các điểm này sẽ càng lớn.

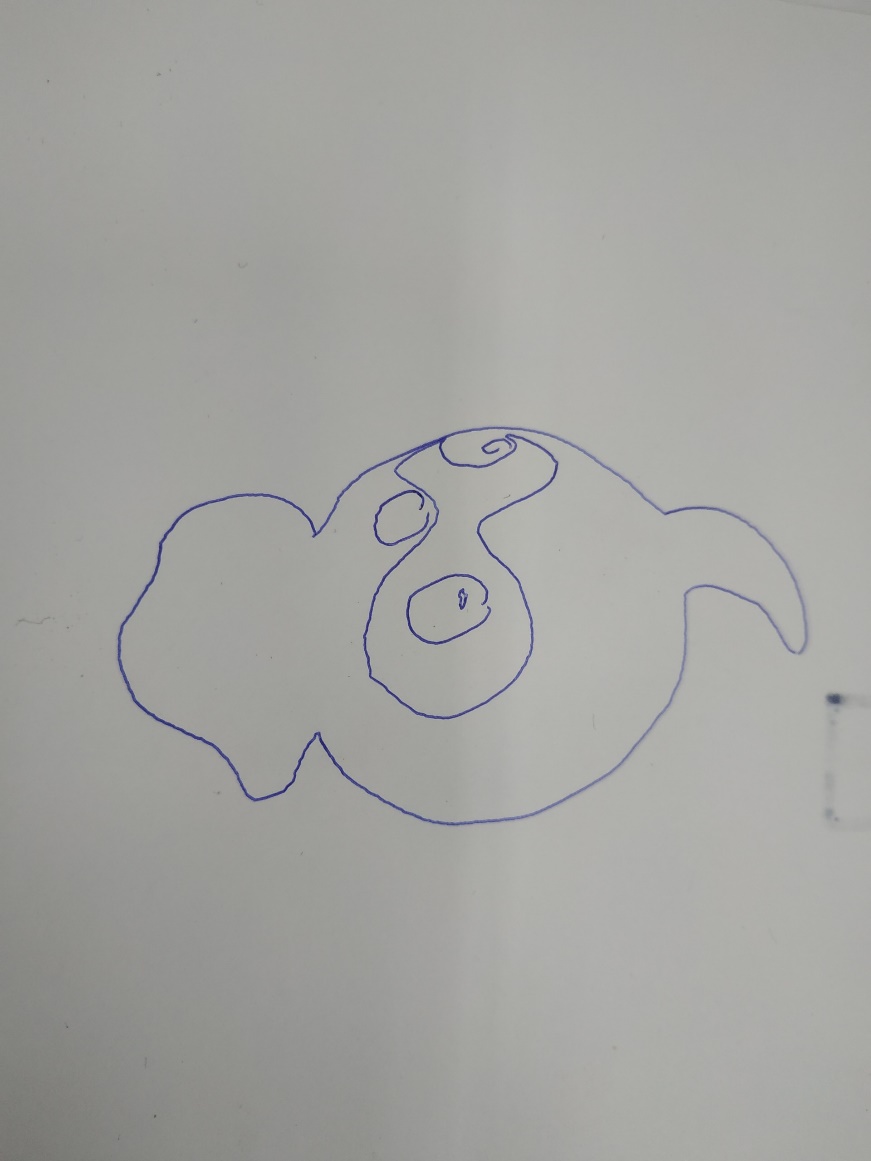
Và dưới đây là một số kết quả hình mẫu CNC khác.





Hình 4.9 Mẫu kết quả CNC viết chữ





Hình 4.10 Mẫu kết quả CNC vẽ hình

## Kết quả xử lí ảnh

### Đánh giá mô hình nhận diện đối tượng

Như đã trình bày ở phía trên, mô hình nhận diện đối tượng có chỉ số Mean average precision ở ngưỡng IoU 50% (mAP@0.50) là 0.4529. Ngoài ra, ta sẽ đánh giá mô hình bằng cách thực hiện đưa các đối tượng vào mô hình và đánh giá thông qua từng frame hình. Ta thực hiện 100 frame hình ứng với 6 đối tượng, tức là mỗi đối tượng sẽ thực hiện 100 lần ở 3 ngưỡng khác nhau là 0.5, 0.7, 0.85.

Bảng 4.4 Thống kê các lần nhận diện

|  |  |  |  |
| --- | --- | --- | --- |
| **VietNam** | | | |
| **Độ tin cậy**  **Số lần** | **0.5** | **0.7** | **0.85** |
| **Nhận diện đúng** | 71 | 90 | 96 |
| **Nhận diện sai** | 29 | 4 | 2 |
| **Không nhận diện** | 5 | 6 | 2 |
| **Switzerland** | | | |
| **Độ tin cậy**  **Số lần** | **0.5** | **0.7** | **0.85** |
| **Nhận diện đúng** | 79 | 84 | 95 |
| **Nhận diện sai** | 21 | 10 | 0 |
| **Không nhận diện** | 0 | 6 | 5 |
| **Sweden** | | | |
| **Độ tin cậy**  **Số lần** | **0.5** | **0.7** | **0.85** |
| **Nhận diện đúng** | 82 | 92 | 95 |
| **Nhận diện sai** | 18 | 2 | 0 |
| **Không nhận diện** | 0 | 6 | 5 |
| **England** | | | |
| **Độ tin cậy**  **Số lần** | **0.5** | **0.7** | **0.85** |
| **Nhận diện đúng** | 84 | 92 | 91 |
| **Nhận diện sai** | 14 | 4 | 0 |
| **Không nhận diện** | 2 | 4 | 9 |
| **Germany** | | | |
| **Độ tin cậy**  **Số lần** | **0.5** | **0.7** | **0.85** |
| **Nhận diện đúng** | 80 | 85 | 89 |
| **Nhận diện sai** | 18 | 5 | 0 |
| **Không nhận diện** | 2 | 10 | 11 |
| **Japan** | | | |
| **Độ tin cậy**  **Số lần** | **0.5** | **0.7** | **0.85** |
| **Nhận diện đúng** | 90 | 92 | 91 |
| **Nhận diện sai** | 8 | 0 | 0 |
| **Không nhận diện** | 2 | 8 | 9 |

**Đánh giá:** Ta thấy rằng với độ tin cậy thấp, mặc dù các đối tượng đều được nhận diện nhưng mô hình vẫn nhận diện nhầm một số vị trí trống là các đối tượng. Đối với độ tin cậy cao, đối tượng nhận diện được đều chính xác nhưng có một số đối tượng lại bị bỏ qua ở các vị trí khác nhau. Mô hình đồng thời cũng không thể nhận diện được các đối tượng nằm dính kề lấy nhau. Với kết quả mô hình luận văn mà nhóm đã thực hiện, ta thấy rằng với độ tin cậy là 0.85, mô hình đạt kết quả tốt nhất. Đồng thời, với tốc độ 15-17 fps, mô hình luận văn có thể chạy realtime phù hợp với mục tiêu gắp vật động.

### Đánh giá độ chính xác trong xác định vị trí và góc xoay

Việc đánh giá độ chính xác của vị trí đối tượng và góc xoay được được thực hiện bằng cách chạy thuật toán xác định của mô hình với hình ảnh của đối tượng được in trên giấy A4 với những vị trí và góc xoay được định trước. Tờ giấy này được đặt sao cho tâm của tờ giấy trùng với tâm của của camera ra và độ cao của tờ giấy bằng với độ cao của băng truyền và đối tượng. Từ đó, ta so sánh kết quả của mô hình và kết quả thực tế.



Hình 4.11 Mẫu đánh giá sai số vị trí và góc xoay

* **Đánh giá sai số xác định vị trí**

Sai số vị trí:

Việc đánh giá được thực hiện ở 8 vị trí khác nhau, ở mỗi vị trí ta thực hiện đo 6 lần ứng với 6 đối tượng. Vì các dữ liệu được thu thập được lớn nên việc thể hiện và trình bày trong bài luận văn này là không hợp lí. Chính vì vậy nên ta đưa dữ liệu vào file excel để thực hiện tính toán và chỉ thể hiện ở đây những dữ liệu tính toán cần thiết.

Bảng 4.5 Thống kê sai số vị trí

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Vị trí thứ | Vị trí đối tượng | | Vị trí trung bình đo được | | Sai số tối đa | | |
|  |  |  |  |  |  |  |
| 1 | -80 | 0 | 1.0631 | 0.7408 | 2.4 | 1.675 | 2.5703 |
| 2 | -80 | -60 | 1.1917 | 1.54 | 2.5 | 3.7 | 3.7106 |
| 3 | 0 | -60 | 1.3742 | 2.1733 | 1.94 | 4.06 | 4.3494 |
| 4 | 40 | -30 | 1.45 | 0.8683 | 2.95 | 1.42 | 3.265 |
| 5 | 80 | 0 | 0.4107 | 0.6193 | 1.25 | 0.98 | 1.271 |
| 6 | 80 | 60 | 0.395 | 0.8167 | 1.25 | 1.52 | 1.5318 |
| 7 | 0 | 60 | 0.728 | 1.411 | 1.55 | 2.98 | 3.3544 |
| 8 | -40 | 30 | 1.0217 | 1.3583 | 1.72 | 3.27 | 3.5377 |
| Sai số *RMSD* | | |  | | | | |

*(đơn vị: mm)*

**Nhận xét:** Sai số giữa việc xác định bằng các thuật toán thị giác máy so với giá trị vị trí thực tế là khoảng xấp xỉ 2 mm. Nguyên nhân có thể do phần nền ít có độ tương phản so với màu sắc của đối tượng, do nhiễu sáng, chói sáng, ánh sáng không được ổn định, … làm cho việc phát hiện các viền bị sai lệch. Lưu ý, phương pháp đánh giá sai số này có thể chứa sai số so với kết quả mà thuật toán đem lại khi chạy với băng truyền vì độ cao của giấy mẫu và băng truyền cộng thêm vật lệch nhau trên dưới 1mm và sai số từ việc đặt giấy vào trung tâm của camera. Tuy nhiên, những sai số này rất nhỏ và coi như có thể bỏ qua.

* **Đánh giá sai số góc xoay**

Vì góc xoay của đối tượng chỉ trong khoảng nên việc đánh giá được thực hiện ở 6 góc xoay khác nhau với mỗi góc xoay ta thực hiện đo 8 lần. Cũng như dữ liệu của vị trí, dữ liệu về góc xoay thu thập được lớn nên ta đưa dữ liệu vào file excel để thực hiện tính toán và chỉ thể hiện ở đây những dữ liệu tính toán cần thiết. Giá trị sai số được tính bằng:

Bảng 4.6 Thống kê sai số góc xoay

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Góc thứ | Góc xoay đối tượng | Trung bình góc xoay đo được | Sai số trung bình | Sai số lớn nhất |
| 1 | 0 | -0.3975 | 0.7982 | 1.465 |
| 2 | 15 | 15.0895 | 0.9495 | 2.575 |
| 3 | 30 | 30.3357 | 1.9367 | 5.7 |
| 4 | 45 | 43.889 | 1.203 | 2.5 |
| 5 | -30 | -29.604 | 0.901 | 2.69 |
| 6 | -15 | -15.4072 | 0.4997 | 1.06 |
| Sai số *RMSD* | |  | | |

*(đơn vị: độ)*

**Nhận xét:** Sai số góc xoay xấp xỉ . Sai số lớn nhất mà thuật toán ghi nhận được là . Nguyên nhân chủ yếu là do sự xuất hiện của nhiễu làm cho viền bao của thuật toán có thể bị sai lệch. Tuy vậy nhưng sự sai lệch này vẫn không ảnh hưởng đến việc gắp bỏ đối tượng vào các ô chứa.

## Kết quả gắp và xếp vật

Ở kết quả đánh giá này ta sẽ chạy thử với hai kiểu trường hợp

* Đặt từng vật lên băng chuyền

Bảng 4.7 Số liệu gắp từng vật trên băng truyền

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Vận tốc băng chuyền (mm/s) | Tổng số lần gắp | Số lần thành công | Số lần không thành công | Hiện tượng các lần không thành công | Nguyên nhân |
| 29.2757 | 18 | 16 | 2 | Vật bị chênh ra khỏi slot | Robot đã gắp đúng tâm của vật tuy nhiên khi thả xuống vật bị nảy lên nên bị trật |
| 36.5946 | 18 | 14 | 4 |
| 43.9136 | 18 | 13 | 5 | Robot gắp lật tâm vật nên thả xuống bị trật |
| 51.2325 | 18 | 10 | 8 | Vận tốc băng chuyền thay đổi đột ngột |

* Nhiều vật được đưa vào cùng một lúc

Bảng 4.8 Số liệu gắp nhiều vật trên băng truyền

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Vận tốc băng chuyền (mm/s) | Tổng số lần gắp | Số lần thành công | Số lần không thành công | Hiện tượng các lần không thành công | Nguyên nhân |
| 29.2757 | 18 | 14 | 4 | Vật bị chênh ra khỏi slot | Robot đã gắp đúng tâm của vật tuy nhiên khi thả xuống vật bị nảy lên nên bị trật |
| 36.5946 | 18 | 14 | 4 |
| 43.9136 | 18 | 10 | 8 | Robot gắp lật tâm vật nên thả xuống bị trật |
| Robot bỏ qua vật | Vận tốc băng chuyền thay đổi đột ngột |
| 51.2325 | 18 | 8 | 10 | Camera không xác định được tâm của vật khi quá sát nhau |
| Toạ độ gắp đi quá workspace của robot |

**Đánh giá:** Như ta thấy tại bảng số liệu trên, đối với gắp từng vật một trên băng truyền số lần chính xác nhiều hơn so với gắp nhiều vật liên tục. Đồng thời, với tốc độ băng truyền nhỏ hơn sẽ đem lại kết quả tốt hơn. Nguyên nhân của hiện tượng trên là do sai số tích lũy tăng dần sau mỗi lần gắp liên tục và sai số của tốc độ của băng truyền sẽ lớn khi ở giá trị lớn.

# KẾT LUẬN

## Nhận xét chung về luận văn

Những mục tiêu mà luận văn đã đạt được:

* Đáp ứng được khá tốt trong việc gắp và phân loại các vật động trên băng chuyền.
* Xây dựng hoàn chỉnh model CNC routing cho robot.
* Các hàm vận hành trên robot hoạt động ổn định.
* Chương trình điều khiển bao gồm đầy đủ các chức năng cơ bản của một robot công nghiệp.

Tuy nhiên luận văn còn tồn tại các hạn chế như:

* Không biết được chính xác tốc độ băng chuyền vì điều khiển vòng hở, gây ảnh hưởng tới độ chính xác của quá trình gắp vật.
* Sai số của robot vẫn gây ảnh hưởng lớn đến chất lượng đầu ra của mẫu CNC.
* Thuật toán thị giác máy chưa hoạt động tốt nếu nhưng các vật kề sát nhau cũng như chạy ở phần nền khác nhau.
* Hình dạng của các đối tượng giống nhau, chưa tạo ra được nhiều sự khác biệt, không mang tính tổng quát.

## Hướng phát triền của luận văn:

Phát triển về mô hình robot:

* Về phần công suất, bản thân các động cơ DC đã đủ momen để khiến cho robot di chuyển nhanh. Ta cần nâng cấp encoder với độ phân giải cao hơn kèm theo mạch servo drive điều khiển chính xác hơn.
* Thay thế động cơ bước ở mô hình băng chuyền thành động cơ DC có điều khiển hồi tiếp để kiểm soát được tốc độ băng chuyền.

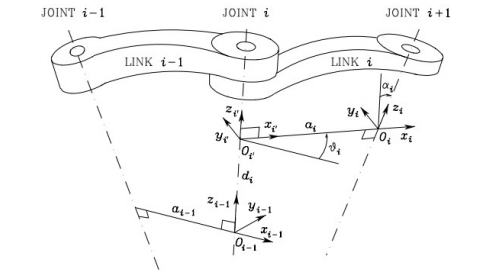
Phát triển thị giác máy:

* Cải thiện độ chính xác của thuật toán.
* Phát triển thuật toán có thể nhận diện các vật kề sát nhau cũng như ở nhưng phông nền băng truyền khác nhau.

# PHỤ LỤC

## Phương pháp Denevit-Hartenbeg (DH)

Bất kì một robot nào cũng có thể xem là tập hợp các khâu gắn liền với các khớp. Nếu đặt cho mỗi khâu một hệ trục tọa độ và sử dụng các phép biến đổi thì có thể mô tả vị trí và hướng tương đối giữa các hệ tọa độ.



Hình 6.1 Minh hoạ phương pháp Denevit-Hartenberg

Một Robot có khớp và khâu. Chỉ số các khớp bắt đầu từ đến , còn khâu bắt đầu 0 đến . Khớp kết nối khâu và khâu . Để chọn một hệ trục tọa độ cho các khâu thì phải tuân theo một số nguyên tắc chung sau:

* Trục nằm trên trục của khớp
* Trục nằm trên đường vuông góc chung hướng từ đến . Trong trường hợp 2 trục giao nhau thì , trục. Xác định bộ thông số Denavit-Hartenberg:
* Gọi là khoảng cách giữa trục và trục trên trục .
* Gọi là góc xoay quanh trục đến song song với .
* Gọi là khoảng cách dọc theo trục giữa và .
* Gọi là góc xoay quanh trục đến song song với . Xác định biến khớp:
* Nếu khớp là khớp quay (revolute) thì là biến khớp.
* Nếu khớp là khớp tịnh tiến (prismatic) thì là biến khớp.

Trên cơ sở các hệ tọa độ đã chọn cho tất cả các khâu của robot thì mối quan hệ giữa các hệ tọa độ nối tiếp nhau (i -1) và (i) được xác định bởi các phép quay và phép tịnh tiến sau đây.

* Quay quanh một góc .
* Tịnh tiến dọc theo một đoạn .
* Tịnh tiến dọc theo một đoạn .
* Quay quanh một góc xoắn .

Phép biến đổi thì hệ sang hệ được biểu diễn bằng ma trận (ma trận chuyển vị) được tính bằng công thức:

Khi robot có n khâu mà ma trận chuyển đổi từ tọa độ 0 đến n được xác định theo công thức:

Đây là phương trình động học thuận của robot. Ma trận  có dạng:

Ta tìm được biểu thị hướng của đầu công tác, biểu thị vị trí của đầu công tác.

## Ma trận Jacobian

Phương trình động học thuận của robot có thể biểu diễn bằng ma trận chuyển đổi giữa đầu công tác và gốc tọa độ.

Trong đó, là giá trị các biến khớp.

Mục đích của ma trận Jacobian là tìm mối quan hệ giữa vận tốc biến khớp và vận tốc trong không gian làm việc. Gọi lần lượt là vận tốc dài và vận tốc góc của đầu công tác. Mối quan hệ được biểu diễn như sau:

là ma trận biểu diễn quan hệ giữa vận tốc dài và vận tốc khớp, trong khi là ma trận biểu diễn vận tốc góc và vận tốc khớp. Đối với robot SCARA thì tương ứng với 4 bậc tự do.

Gọi là ma trận xoay nằm trong các ma trận chuyển đổi ứng giữa các hệ tọa độ, là vị trí của gốc tọa độ tương ứng. Công thức tính từng phần tử của ma trận Jacobian được trình bày ở bảng dưới.

Bảng 6.1 Công thức tính ma trận Jacobian

|  |  |  |
| --- | --- | --- |
|  | **Khớp tịnh tiến** | **khớp xoay** |
| **Vận tốc dài** |  |  |
| **Vận tốc góc** |  |  |

Có thể viết lại 2 phương trình trên như sau:

## Điểm kì dị (Singularities)

Ta có công thức:

Ma trận Jacobian có kích thước nên hạng của ma trận được tính là:

Hạng của ma trận không phải là hằng số. Thật vậy, hạng của ma trận Jacobian sẽ phụ thuộc vào cấu hình của . Tại những cấu hình mà hạng của ma trận **J** nhỏ hơn giá trị tối đa của nó được gọi là điểm kì dị (singularities).

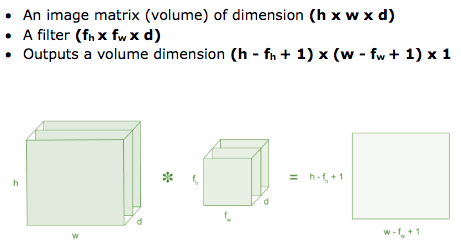
Điểm kì dị có những đặc điểm sau:

* Điểm kì dị biễu diễn một cấu hình mà tại đó, nhìn từ không gian làm việc, robot bị mất đi bậc tự do nên có một số hướng robot ko thể di chuyển được. Điều này thể hiện rõ ràng khi robot ở điểm kì dị biên.
* Khi robot di chuyển đến điểm kì dị thì phương trình động học ngược có thể có nhiều nghiệm.
* Trong lân cận của điểm kì dị, một sự biến nhỏ của vị trí trong không gian làm việc cũng có thể gây ra một vận tốc cực kì lớn cho các khớp.

## Các kiểu lớp trong mạng CNN

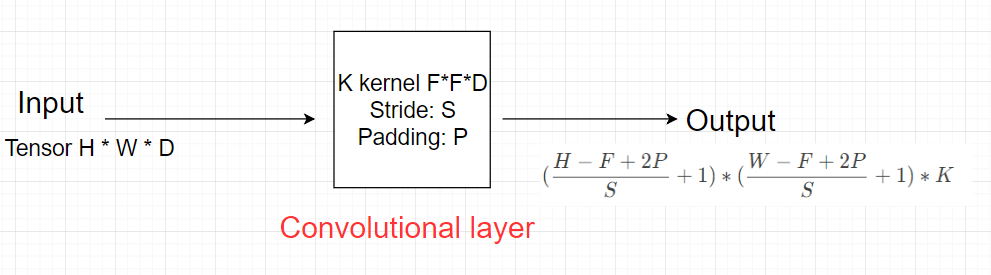
### Convolutional Layer

Lớp đầu tiên của mạng CNN là lớp tích chập với tính năng trích xuất các đặc trưng từ hình ảnh đầu vào. Lớp mạng này thể hiện mối quan hệ giữa các pixel thông qua việc tìm kiếm các đặc trưng của hình ảnh bằng cách sử dụng những cửa sổ trượt nhỏ của dữ liệu đầu vào. Nó là một phép toán có hai đầu vào là ma trận hình ảnh và bộ lọc hoặc hạt nhân (*Kernel*) với đầu ra là một ma trận thể hiện đặc trưng được trích xuất gọi là *Feature Map*.



Hình 6.2 Phép Convolution

Ví dụ như đầu vào là một ảnh màu có 3 channels red, green, blue kích thước 480 x 640 nên ảnh được biểu diễn dưới dạng tensor 3 chiều 480 x 640 x 3. Do đó kernel cũng là 1 tensor 3 chiều kích thước k x k x 3 (k thường là số lẻ 3,5,9,…). Sau đó thực hiện di chuyển kernel trên dữ liệu đầu vào, từ trái qua phải, trên xuống dưới (Stride = 1) và nhân tương ứng từng giá trị của dữ liệu đầu vào với nó rồi tính tổng lại, đưa qua *Activation funciton* (sigmoid, relu, elu, ...), kết quả sẽ là một con số cụ thể, tập hợp các con số này chính là *Feature map* (tensor đặc tính). Với mỗi kernel khác nhau ta sẽ học được những đặc trưng khác nhau của ảnh, nên trong mỗi convolutional layer ta sẽ dùng nhiều kernel để học được **nhiều thuộc tính** của ảnh.



Hình 6.3 Mô hình Convolutional Layer tổng quát

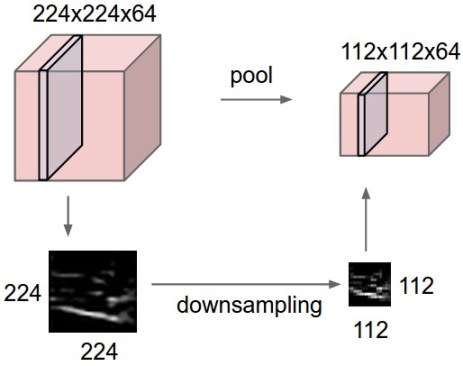
Giả sử input của một convolutional layer tổng quát là tensor kích thước . Kernel có kích thước (kernel luôn có depth bằng depth của input và là số lẻ), Stride: , Padding: . Convolutional layer áp dụng kernel. Vậy Output của layer là tensor 3 chiều có kích thước:

Tổng số parameter của layer: Mỗi kernel có kích thước và có 1 hệ số bias, nên tổng parameter của 1 kernel là . Mà convolutional layer áp dụng K kernel nên tổng số parameter trong layer này là

### Pooling Layer

*Pooling layer* là lớp thứ hai trong CNN, thường được dùng giữa các *Convolutional layer* với chức năng giảm kích thước dữ liệu đầu vào nhưng vẫn giữ được các đặc trưng quan trọng. Điều này là cần thiết cho các model để giúp tăng tốc độ xử lý cũng như tính toán.

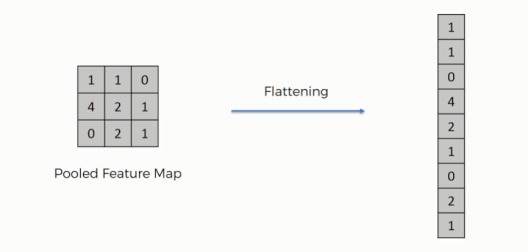
Có hai loại *Pooling Layer* phổ biến đó là *Max Pooling* và *Average Pooling*. Gọi pooling size kích thước . Input của pooling layer có kích thước , ta tách ra làm **D** ma trận kích thước . Với mỗi ma trận, trên vùng kích thước trên ma trận ta tìm maximum hoặc average của dữ liệu rồi viết vào ma trận kết quả. Quy tắc về stride và padding áp dụng như phép tính convolution trên.



Hình 6.4 . Pooling Layer (2\*2)

### Fully Connected Layer

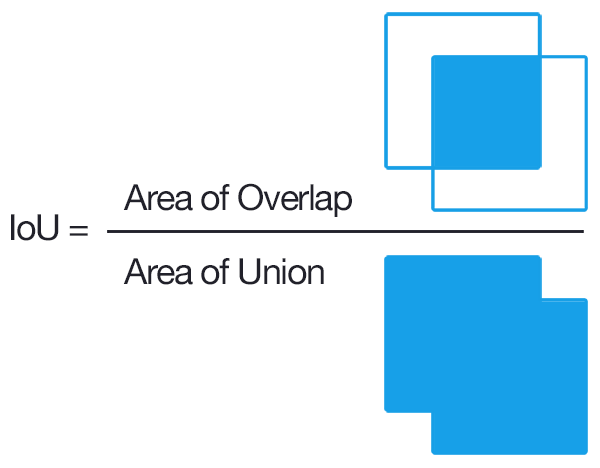
Lớp cuối cùng trong CNN là fully connected layer. Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer lần lượt thì model đã học được các đặc điểm của đối tượng (ví dụ hình dáng, góc cạnh,…) thì tensor của output của layer cuối cùng, kích thước , sẽ được chuyển về 1 vector có chiều dài bằng với mục đích tập hợp các đặc trưng ta đã tìm được thành một khối duy nhất. Sau đó dữ liệu này được đưa tới các fully connected layer có chức năng kết hợp, liên kết các đặc điểm này và phân chia chất lượng, đánh giá để ra được output của model.



Hình 6.5 Bước chuyển ma trận 3 chiều thanh vector 1 chiều

## Hàm tính IOU

IOU (INTERSECTION OVER UNION) là hàm đánh giá độ chính xác của object detector trên tập dữ liệu cụ thể, áp dụng kĩ thuật triệt tiêu không tối đa. Hàm IOU được tính bằng:



Hình 6.6 Hàm IOU

Trong đó Area of Overlap là diện tích phần giao nhau giữa predicted bounding box với grouth-truth bouding box , còn Area of Union là diện tích phần hợp giữa predicted bounding box với grouth-truth bounding box. Nếu bounding box nào có prediction tốt nhất được giữ lại và nếu IOU < 0.5 ta sẽ loại bỏ. Kỹ thuật triệt tiêu không tối đa nhằm mục đích loại bỏ các hộp giới hạn trùng lặp của cùng một đối tượng bằng cách chọn ra hộp tốt nhất.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | P. D. Khanh, "YOLO You Only Look Once," 9 March 2020. [Online]. Available:https://phamdinhkhanh.github.io/2020/03/09/DarknetAlgorithm.html. |
| [2] | J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018. |
| [3] | J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once:Unified, Real-Time Object Detection," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. |
| [4] | J. Hui, "Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3," 18 March 2018. [Online]. Available: https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088. |
| [5] | A. Zhang, Z. C. Lipton and M. L. a. A. J. Smola, "Dive into Deep Learning," 2021. |
| [6] | C.-Y. Wang, H.-Y. M. Liao, P.-Y. Chen and J.-W. Hsieh, "Enriching Variety of Layer-Wise Learning Information by Gradient Combination," 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 2019. |
| [7] | M. Baygin, M. Karakose, A. Sarimaden and E. Akin, "An Image Processing based Object Counting Approach for Machine Vision Application," International Conference on Advances and Innovations in Engineering (ICAIE), 2018. |
| [8] | "Color Space Conversions," OpenCV documentation, [Online]. Available: https://docs.opencv.org/3.4/d8/d01/group\_\_imgproc\_\_color\_\_conversions.html. |
| [9] | "Operations on arrays," OpenCV documentation, [Online]. Available: https://docs.opencv.org/3.4/d2/de8/group\_\_core\_\_array.html. |
| [10] | "Image Filtering," OpenCV documentation, [Online]. Available: https://docs.opencv.org/master/d4/d86/group\_\_imgproc\_\_filter.html. |
| [11] | "Image Thresholding," OpenCV documentation, [Online]. Available: https://docs.opencv.org/master/d7/d4d/tutorial\_py\_thresholding.html. |
| [12] | J. Canny, "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986. |
| [13] | "Feature Detection," OpenCV documentaion, [Online]. Available: https://docs.opencv.org/master/dd/d1a/group\_\_imgproc\_\_feature.html#ga04723e007ed888ddf11d9ba04e2232de. |
| [14] | "Contours : Getting Started," OpenCV documentation, [Online]. Available: https://docs.opencv.org/3.4/d4/d73/tutorial\_py\_contours\_begin.html. |
| [15] | "Structural Analysis and Shape Descriptors," OpenCV documentation, [Online]. Available: https://docs.opencv.org/3.4/d3/dc0/group\_\_imgproc\_\_shape.html. |
| [16] | "Camera Calibration and 3D Reconstruction," OpenCV documentaion, [Online]. Available: https://docs.opencv.org/master/d9/d0c/group\_\_calib3d.html. |
| [17] | A. Becker, "Kalman Filter Tutorial," [Online]. Available: https://www.kalmanfilter.net/default.aspx. |
| [18] | "cv::KalmanFilter Class Reference," OpenCV documentation, [Online]. Available: https://docs.opencv.org/master/dd/d6a/classcv\_1\_1KalmanFilter.html. |