

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH

KHOA ĐIỆN - ĐIỆN TỬ

BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

TRẦN HOÀNG KHÔI NGUYỄN

ĐẶNG ĐÌNH NGUYỄN

LUẬN VĂN TỐT NGHIỆP

HỆ THỐNG ĐIỀU KHIỂN TAY MÁY SAO CHÉP CỬ ĐỘNG

SỬ DỤNG MẠNG LAN

KỸ SƯ NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

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.....

ĐỀ CƯƠNG CHI TIẾT

TÊN LUẬN VĂN:

HỆ THỐNG ĐIỀU KHIỂN TAY MÁY SAO CHÉP CỬ ĐỘNG THÔNG QUA MẠNG LAN

Cán bộ hướng dẫn: Cô Bùi Thanh Huyền

ThS. Huỳnh Hữu Phương

Thời gian thực hiện: 01/2018 đến 06/2018

Sinh viên thực hiện: Trần Hoàng Khôi Nguyên

Đặng Đình Nguyên

Nội dung đề tài:

Mục tiêu, phạm vi, đối tượng

- Mô hình hóa và mô phỏng động học thuận, động học ngược, hoạch định quỹ đạo khớp cho tay máy RV-2AJ
- Xây dựng hệ thống điều khiển từ xa tay máy Mitsubishi RV-2AJ sao cho điểm cuối tay máy sao chép chính xác vị trí và hướng bàn tay người dùng.
- Giao tiếp giữa các máy tính ở hướng người dùng và hướng tay máy thông qua mạng Ethernet/LAN.
- Xây dựng hệ thống môi trường ảo hỗ trợ điều khiển.
- Thiết kế và thi công dụng cụ công tác cho điểm cuối có hồi tiếp lực.
- Xây dựng phần mềm giao diện người dùng tích hợp hỗ trợ điều khiển.

Phương pháp thực hiện

- **Mô hình hóa và mô phỏng**

- Tính toán động học thuận bằng phương pháp ma trận DH
- Tính toán động học ngược bằng phương pháp đại số và Jacobian.
- Hoạch định quỹ đạo bằng phương pháp thực nghiệm.
- Dùng MATLAB mô phỏng kiểm chứng các mô hình toán đã xây dựng.

- **Xây dựng hệ thống điều khiển theo phương pháp sao chép cử động**

Hệ thống gồm có máy trạm Master (tiếp nhận thông tin và gửi tín hiệu điều khiển) và máy trạm Slave (nhận thông tin điều khiển và điều khiển Tay máy RV-2AJ).

Phần mềm cho các máy trạm được viết bằng ngôn ngữ C++ trên nền tảng Qt.

Tại máy trạm Master

- Sử dụng cảm biến Leap Motion và thư viện Orion để theo dõi và nhận diện cử chỉ bàn tay (bao gồm vị trí và hướng lòng bàn tay, hướng ngón tay, vị trí các khớp ngón tay).
- Lọc dữ liệu thu được và tính toán số liệu, chuyển đổi sang hệ tọa độ tay máy.
- Từ các số liệu đã tính toán, máy trạm Master sẽ gửi các khung dữ liệu điều khiển đến máy trạm Slave thông qua mạng LAN
- Đồng thời máy trạm Master tiếp nhận các phản hồi gửi từ máy trạm Slave để theo dõi hoạt động của Tay máy.
- Ghi lại thao tác điều khiển tay máy để có thể lặp lại.

Tại máy trạm Slave

- Tiếp nhận dữ liệu điều khiển từ máy trạm Master.
- Từ dữ liệu điều khiển vừa nhận được, tính toán các thông số điều khiển tay máy.
- Xét các thông số điều khiển mới với trạng thái hiện tại của tay máy, nếu phù hợp sẽ điều khiển tay máy thông qua Controller CR-571, nếu không sẽ báo tín hiệu lỗi về máy trạm Master.

- **Giao tiếp thông qua mạng LAN**

Sử dụng giao thức UDP để truyền dữ liệu hình ảnh, tín hiệu điều khiển giữa các trạm thông qua mạng LAN. Hình ảnh được nén để giảm kích thước, tăng tốc độ truyền nhận.

- **Xây dựng môi trường ảo**

Môi trường ảo bao gồm mô hình tay máy Mitsubishi RV-2AJ đặt trong Khu vực làm việc (bao gồm bàn làm việc, chân đế và hai thanh gắn Kinect) và mô hình các vật trên Bàn làm việc thu được bằng hai Kinect.

- Mô hình Tay máy và Khu vực làm việc được vẽ bằng phần mềm SolidWorks, được đưa vào code C++ bằng thư viện Asset Import, hiển thị ra giao diện người dùng. Các góc khớp tay máy sẽ được cập nhật đồng thời với mô hình thật theo tín hiệu phản hồi từ tay máy.
- Thu thập dữ liệu 3D từ hai Kinect bằng thư viện Nestk. Sử dụng thư viện PCL để xử lý dữ liệu thu được. Tính toán thông số ngoại bằng MATLAB để xoay tọa độ hai Kinect về tọa độ World của tay máy. Sử dụng thư viện PCL để tái tạo bề mặt từ dữ liệu đã qua xử lý và gửi về máy trạm điều khiển thông qua mạng LAN. Tại máy trạm, hình ảnh 3D sẽ được đưa vào code C++ bằng thư viện Asset Import, hiển thị ra thư viện người dùng.

- **Thiết kế và thi công dụng cụ công tác cho điểm cuối**

- Dụng cụ công tác được lựa chọn là tay kẹp – gripper để thao tác với nhiều đối tượng đa dạng. Tham khảo các bản vẽ tay kẹp và thiết kế bằng phần mềm SolidWorks. Thi công tay kẹp bằng phương pháp in 3D.
- Bố trí cảm biến lực tại điểm tác dụng của tay kẹp và hiệu chuẩn. Sử dụng động cơ RC Servo làm thiết bị động lực cho tay kẹp. Sử dụng Kit phát triển STM32F103C8T6 để điều khiển động cơ, đọc giá trị ADC từ cảm biến lực và giao tiếp với máy trạm Slave thông qua chuẩn truyền thông UART.

- **Xây dựng phần mềm giao diện người dùng**

Phần mềm giao diện người dùng được lập trình bằng ngôn ngữ C++ trên nền tảng Qt. Phần mềm điều khiển có hai chế độ RUN và SIMULATION

Chế độ RUN: Điều khiển thực tay máy RV-2AJ

- Hiển thị hệ thống trong môi trường ảo 3D để theo dõi chuyển động của tay máy từ xa.
- Hiển thị hình ảnh của camera 2D tại điểm cuối.
- Kết nối với tay máy, camera 2D, Kinect.
- Tùy chỉnh chế độ điều khiển Manual và Teleoperation.
- Hiển thị thông số vị trí góc khớp, tọa độ điểm cuối.

- Thông báo trạng thái làm việc và lỗi nếu có của tay máy và cảm biến Leap Motion.
- Ghi lại thao tác điều khiển và thực hiện lại.
- Vẽ đồ thị vị trí, vận tốc của góc khớp và điểm cuối theo thời gian.

Chế độ SIMULATION: Điều khiển mô phỏng tay máy.

- Điều khiển cơ bản các góc khớp và theo dõi tọa độ theo từng trục.
- Thiết lập tốc độ góc khớp, gia tốc, hãm tốc.
- Vẽ đồ thị vị trí, vận tốc của góc khớp và điểm cuối theo thời gian.
- Đồ hoạ 3D.

Kết quả mong đợi của đề tài

- Hệ thống hoạt động ổn định trong mạng LAN.
- Dữ liệu từ cảm biến Leap Motion có độ nhiễu thấp.
- Điều khiển Tay máy RV-2AJ bám sát theo chuyển động bàn tay một cách mượt mà, không bị ngắt quãng, trạng thái được phản hồi nhanh chóng.
- Dữ liệu từ camera Kinect đảm bảo khả năng theo dõi môi trường làm việc thực tế thông qua môi trường ảo 3D
- Giao diện người dùng thân thiện, dễ sử dụng, đầy đủ tính năng cần thiết.
- Tay máy có khả năng học lại thao tác điều khiển và tự vận hành.

Kế hoạch thực hiện:

Phân công công việc:

• Trần Hoàng Khôi Nguyên

- Mô hình hoá robot Động học thuận, Động học ngược, Hoạch định quỹ đạo, mô phỏng MATLAB.
- Viết chương trình máy trạm Slave điều khiển robot.
- Viết chương trình máy trạm Master xử lý cảm biến cử động Leap Motion.
- Viết chương trình giao tiếp giữa hai trạm.
- Viết giao diện người dùng.

• Đặng Đình Nguyên

- Thiết kế và thi công phần cứng, viết chương trình nhúng và chương trình máy tính điều khiển tay kẹp ở điểm cuối robot.
- Thiết kế Khu vực làm việc cho tay máy và bố trí camera Kinect, camera 2D.
- Viết chương trình máy tính đọc và xử lý đồng thời hai camera Kinect để xây dựng lại không gian 3D.
- **Cả hai**
 - Thi công bàn làm việc cho tay máy.
 - Xây dựng mô hình demo cho hệ thống.

Xác nhận của Cán bộ hướng dẫn

(Ký tên và ghi rõ họ tên)

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)

Sinh viên

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến cô Bùi Thanh Huyền, thầy Huỳnh Hữu Phương – giảng viên hướng dẫn. Thầy cô đã cho chúng em những chỉ dẫn quý báu để hoàn thành luận văn kịp thời và đầy đủ nhất.

Xin cảm ơn quý Thầy cô và ban chủ nhiệm Khoa Điện – Điện tử, đặc biệt là Bộ môn Tự động, đã tận tụy giảng dạy và truyền đạt những kiến thức, kinh nghiệm quý báu cho chúng em trong những năm qua.

Chân thành cảm ơn Ban giám đốc Phòng thí nghiệm Trọng điểm Điều khiển số và Kỹ thuật hệ thống đã hỗ trợ trang thiết bị để chúng em thực hiện luận văn. Cảm ơn thầy Trần Ngọc Huy đã tạo môi trường nghiên cứu rộng mở. Xin cảm ơn các anh, các bạn và các em là thành viên trong Phòng thí nghiệm đã giúp đỡ trong những lúc khó khăn nhất.

Cuối cùng, chúng con xin dành lời cảm ơn sâu sắc nhất cho ba, mẹ và các thành viên trong gia đình. Mọi người luôn là chỗ dựa tinh thần, là nguồn động viên và động lực không thể thiếu để chúng em có thể vượt qua những gian nan thử thách trong cuộc sống và hoàn thành Luận văn tốt nghiệp.

Sinh viên thực hiện

Trần Hoàng Khôi Nguyên

Đặng Đình Nguyên

TÓM TẮT

Phương pháp điều khiển từ xa - Teleoperation đã phát triển từ những ngày đầu và luôn đóng vai trò quan trọng trong lĩnh vực Robotic. Luận văn tập trung xây dựng một hệ thống Teleoperation với nhiệm vụ sao chép cử động bàn tay của người vận hành lên tay máy Mitsubishi RV-2AJ thông qua mạng LAN.

Từ Master, một máy tính chạy chương trình thu thập vị trí và hướng của bàn tay bằng cảm biến Leap Motion, biến đổi thành tín hiệu đặt và truyền đi phía Slave thông qua mạng LAN. Tại Slave, một máy tính khác có nhiệm vụ điều khiển robot bám theo tín hiệu đặt, phản hồi trạng thái, hình ảnh hai chiều về Master. Đồng thời, máy tính thứ ba sử dụng hai cảm biến Kinect dựng hình ảnh ba chiều nhằm phản hồi các vật thể được thao tác bởi robot.

Một gripper được chế tạo để nâng cấp đầu công tác hiện có của robot.

Môi trường ảo 3D được xây dựng để hỗ trợ người dùng hình dung đầy đủ không gian phía Slave. Chế độ mô phỏng hoạch định quỹ đạo giúp người dùng làm quen việc điều khiển robot.

Các kết quả đạt được cho thấy tính khả thi và thực tế của việc điều khiển, lập trình robot từ xa bằng cử chỉ tự nhiên với sự hỗ trợ của đồ họa ba chiều.

ABSTRACT

From the beginning of robotics, Teleoperation has developed and been an important method. This thesis focusses on building a Teleoperation system which copies human hand motion into a Mitsubishi RV-2AJ robot arm via Ethernet LAN.

From Master station, one computer collects human hand's position and orientation by using Leap Motion sensor, then transfers into designed signal of Slave robot. At Slave station, another computer controls the robot to move following designed signals, feedback robot states and 2D images to Master user. At the same time, a third computer uses two Kinect sensors to reconstruct 3D images of the objects in the robot workspace.

A gripper is designed and made to upgrade current endpoint tool of robot.

A 3D virtual environment is built to assist users to totally imagine space at Slave side. The simulation of trajectory planning helps users to practice controlling the robot.

The results achieved show practical application of tele control and programing robot by using human natural motion with 3D vision assistance

MỤC LỤC

Chương 1. TỔNG QUAN ĐỀ TÀI.....	1
1.1. Sơ lược về hệ thống Teleoperation và ứng dụng trong lĩnh vực Robot..	1
1.2. Các hệ thống điều khiển từ xa sao chép cử động người điều khiển hiện nay	3
1.3. Tổng quan đề tài	6
1.4. Cấu trúc trình bày luận văn.....	7
Chương 2. GIỚI THIỆU THIẾT BỊ VÀ PHẦN MỀM.....	9
2.1. Giới thiệu thiết bị.....	9
2.1.1. Tay máy Mitsubishi RV-2AJ	9
2.1.2. Cảm biến Leap Motion	16
2.1.3. Camera Microsoft Kinect	18
2.1.4. Camera Logitech C270.....	20
2.1.5. Kit phát triển STM32F103C8T6	21
2.1.6. Động cơ RC Servo TowerPro MG996R	22
2.1.7. Cảm biến lực Interlink FSR402	22
2.2. Giới thiệu phần mềm	24
2.2.1. Nền tảng Qt – C++	24
2.2.2. Thư viện xử lý ảnh 3D PointCloud Library	27
2.2.3. Thư viện xử lý ảnh 3D Nestk - RGBDemo	31
2.2.4. Thư viện Orion cho cảm biến Leap Motion.....	32
2.2.5. MATLAB Simulink và SimMechanics.....	33
2.2.5.1. MATLAB Simulink.....	33

2.2.5.2. SimMechanics	34
2.2.6. Thư viện đồ họa OpenGL	34
2.2.7. Thư viện Asset Import.....	35
Chương 3. ĐỘNG HỌC ROBOT RV-2AJ	36
3.1. Động học vị trí	36
3.1.1. Động học thuận.....	36
3.1.2. Động học ngược.....	38
3.2. Động học vận tốc	42
3.2.1. Ma trận Jacobi.....	42
3.2.2. Tìm động học vị trí ngược bằng ma trận Jacobi giả nghịch đảo...43	
3.3. Mô phỏng.....	44
3.3.1. Xây dựng mô phỏng với MATLAB m-file và Simmechanics.....	44
3.3.2. Kết quả mô phỏng.....	46
3.3.2.1. Kiểm tra tương quan động học thuận và động học ngược	46
3.3.2.2. So sánh mô hình toán với Simmechanics.....	48
Chương 4. HOẠCH ĐỊNH QUỸ ĐẠO ROBOT RV-2AJ.....	51
4.1. Thuật toán	51
4.1.1. Quỹ đạo vận tốc hình thang.....	51
4.1.2. Đồng bộ chuyển động giữa các khớp.....	53
4.2. Tìm thông số của hàm tính thời gian tăng tốc, hãm tốc.....	55
4.3. Mô phỏng.....	59
4.3.1. Xây dựng mô phỏng.....	59
4.3.2. Kết quả mô phỏng.....	59
4.3.2.1. Kết quả quỹ đạo một khớp	59

4.3.2.1. Kết quả quỹ đạo đồng bộ các khớp.....	60
Chương 5. THIẾT KẾ HỆ THỐNG	62
5.1. Tổng quan.....	62
5.2. Thiết kế và thi công dụng cụ công tác.....	63
5.2.1. Một số phương án thiết kế	63
5.2.1.1. Giác hút.....	63
5.2.1.2. Nam châm điện.....	63
5.2.1.3. Tay kẹp.....	64
5.2.2. Thiết kế và thi công tay kẹp.....	65
5.2.2.1. Mục đích và yêu cầu	65
5.2.2.2. Bản vẽ thiết kế.....	65
5.2.3. Hiệu chuẩn cảm biến lực.....	66
5.2.4. Mạch ra chân Kit STM32F103C8T6	68
5.2.4.1. Sơ đồ nguyên lý.....	68
5.2.4.2. Sơ đồ Layout.....	69
5.2.5. Giải thuật điều khiển Tay kẹp.....	69
5.2.6. Kết quả thực hiện Tay kẹp	71
5.3. Thiết kế và thi công Khu vực làm việc.....	71
5.3.1. Bố trí Kinect	71
5.3.2. Bố trí Camera điểm cuối	73
5.3.3. Thiết kế và thi công bàn làm việc.....	74
5.3.3.1. Thiết kế bàn làm việc.....	74
5.3.3.2. Thi công bàn làm việc.....	75
Chương 6. GIẢI THUẬT VÀ CHƯƠNG TRÌNH ĐIỀU KHIỂN	77

6.1.	Tổng quan.....	77
6.2.	Chương trình điều khiển Robot	79
6.2.1.	Điều khiển Robot trong chế độ Manual	79
6.2.2.	Điều khiển Robot trong chế độ Teleoperation	81
6.2.3.	Điều khiển Robot tự động bằng chức năng Program	82
6.2.4.	Điều khiển Robot mô phỏng.....	83
6.3.	Chương trình xử lý cảm biến Leap.....	84
6.3.1.	Khối xét cử chỉ.....	84
6.3.2.	Khối quy đổi tọa độ	85
6.3.3.	Lọc nhiễu	87
6.3.3.1.	Bộ xét mode chuyển động.....	88
6.3.3.2.	Bộ lọc thông thấp.....	91
6.3.3.3.	Bộ lọc Kalman	91
6.3.3.4.	Kết quả lọc.....	92
6.4.	Chương trình xử lý cảm biến Kinect	93
6.4.1.	Lưu đồ giải thuật.....	93
6.4.2.	Chức năng từng khối	94
6.4.2.1.	Khối Chụp ảnh 3D.....	94
6.4.2.2.	Khối Chuyển ảnh sang PointCloud.....	95
6.4.2.3.	Khối Lọc XY	95
6.4.2.4.	Khối Tách mặt bàn	95
6.4.2.5.	Khối Xoay tọa độ	96
6.4.2.6.	Khối Ghép hai PointCloud.....	97
6.4.2.7.	Khối Làm mượt và lọc nhiễu.....	97

6.4.2.8.	Khối Tái tạo bề mặt.....	98
6.4.2.9.	Khối Truyền dữ liệu bề mặt về trạm Master.....	99
6.5.	Mạng giao tiếp UDP	99
6.6.	Đọc Camera Tool	101
6.7.	Giao diện người dùng	102
6.7.1.	Chế độ RUN.....	102
6.7.1.1.	Khối Connection.....	103
6.7.1.2.	Khối Robot.....	103
6.7.1.3.	Khối Leap Sensor.....	105
6.7.1.4.	Khối 3D Environment.....	106
6.7.1.5.	Khối Tool Camera.....	106
6.7.1.6.	Khối Graph	107
6.7.2.	Chế độ SIMULATION.....	108
Chương 7.	KẾT QUẢ VÀ KẾT LUẬN	110
7.1.	Kết quả thực nghiệm.....	110
7.1.1.	Kết quả sao chép cử động.....	110
7.1.2.	Kết quả hoạt động tự động.....	112
7.1.3.	Kết quả tái tạo vật trong không gian ảo 3D.....	113
7.2.	Kết luận.....	115
7.2.1.	Nhiệm vụ đã hoàn thành.....	115
7.2.2.	Những điểm hạn chế.....	115
7.2.3.	Hướng phát triển đề tài.....	115

DANH MỤC HÌNH VẼ

Hình 1.1. Ứng dụng Teleoperation trong lĩnh vực thám hiểm không gian	2
Hình 1.2 Máy bay không người lái điều khiển từ xa Predator của Mỹ	2
Hình 1.3. Ứng dụng Teleoperation trong Phẫu thuật từ xa và thiết bị lặn ROV... 3	
Hình 1.4. Dự án VI-Bot đang vận hành	4
Hình 1.5. Dự án Robot Tìm kiếm và Cứu hộ Thông minh.....	5
Hình 1.6. Luận văn tốt nghiệp của Sinh viên Đại học Bách Khoa Hà Nội	5
Hình 1.7. Sơ đồ minh hoạ tổng quan hệ thống	6
Hình 2.1. Cấu hình các khâu và khớp của Tay máy RV-2AJ.....	9
Hình 2.2. Không gian làm việc của Tay máy RV-2AJ	11
Hình 2.3. Bộ điều khiển CR-571.....	12
Hình 2.4. Cảm biến Leap Motion	16
Hình 2.5. Cấu tạo cảm biến Leap Motion	17
Hình 2.6. Trục tọa độ của cảm biến Leap Motion.....	17
Hình 2.7. Không gian làm việc của cảm biến Leap Motion.....	18
Hình 2.8. Cấu tạo của camera Kinect	19
Hình 2.9. Sơ đồ khối của cảm biến Kinect	20
Hình 2.10. Camera Logitech C270	20
Hình 2.11. Kit phát triển STM32F103C8T6.....	21
Hình 2.12. Động cơ TowerPro MG996R.....	22
Hình 2.13. Cảm biến lực Interlink FSR402.....	22
Hình 2.14. Logo Qt C++	24
Hình 2.15. Hoạt động của Signal & Slot	25
Hình 2.16. So sánh chương trình đơn luồng và đa luồng.....	26
Hình 2.17. Cửa sổ lập trình giao diện QtDesigner.....	27
Hình 2.18. Cấu trúc thư viện PCL.....	27
Hình 2.19. Biểu diễn Vector mô tả điểm trong không gian.....	28

Hình 2.20. Hình ảnh dữ liệu trước và sau lọc nhiễu sử dụng module Filters	28
Hình 2.21. Dữ liệu đã được phân vùng sử dụng module Kdtree	29
Hình 2.22. Dữ liệu đã được phân vùng sử dụng module Octree.....	29
Hình 2.23. Dữ liệu đã được tách nhóm thành các hình dạng cụ thể sử dụng module SampleConsensus	30
Hình 2.24. Bề mặt tái tạo bằng phương pháp đa giác sử dụng module Surface	30
Hình 2.25. Hình ảnh được hiển thị sử dụng module Visualization.....	31
Hình 2.26. Hình ảnh từ Leap SDK sử dụng thư viện Orion	32
Hình 2.27. Logo MATLAB	33
Hình 2.28. Logo OpenGL	34
Hình 3.1 Hệ toạ độ của robot RV-2AJ	36
Hình 3.2. Các cấu hình khả thi của robot tại một vị trí.....	41
Hình 3.3. Sơ đồ khối mô phỏng robot RV-2AJ	45
Hình 3.4. Quá trình tính động học ngược bằng ma trận Jacobi	48
Hình 3.5. Đồ hoạ Simmechanics mô phỏng robot tại các vị trí tính toán.....	49
Hình 4.1. Đồ thị quỹ đạo LSPB.....	51
Hình 4.2. Thuật toán hoạch định quỹ đạo LSPB đồng bộ các khớp	54
Hình 4.3. Quỹ đạo thực tế khớp J1 của robot tại $V\% = 30$, $A\% = 4$, $D\% = 4$	56
Hình 4.4. Đồ thị kết quả ước lượng T_a và T_d	58
Hình 4.5. Đồ thị quỹ đạo mô phỏng và thực tế của robot RV-2AJ	60
Hình 4.6. Đồ thị mô phỏng quỹ đạo LBSP đồng bộ các khớp	61
Hình 5.1. Sơ đồ khối thiết kế hệ thống.....	62
Hình 5.2. Valve chân không và kí hiệu	63
Hình 5.3. Nam châm điện và nguyên lý hoạt động	63
Hình 5.4. Tay kẹp cho tay máy	64
Hình 5.5. Bản thiết kế tay kẹp.....	66
Hình 5.6. Thông số hiệu chuẩn cảm biến lực FSR402 từ nhà sản xuất	66
Hình 5.7. Sơ đồ nguyên lý mạch ra chân cho Kit STM32F103	68
Hình 5.8. Layout cho mạch ra chân kit STM32F103	69

Hình 5.9. Lưu đồ chương trình Vi điều khiển	70
Hình 5.10. Tay kẹp sau khi hoàn thành	71
Hình 5.11. Độ mở của Kinect theo chiều dọc và chiều ngang	72
Hình 5.12. Phác thảo vị trí bố trí Kinect.....	72
Hình 5.13. Bản thiết kế để gắn camera 2D	73
Hình 5.14. Bố trí camera 2D quan sát điểm cuối.....	74
Hình 5.15. Mô hình Khu vực làm việc được thiết kế bằng SolidWorks	75
Hình 5.16. Bàn làm việc của tay máy sau khi hoàn thành.	76
Hình 6.1. Sơ đồ tổng quan chương trình máy tính	77
Hình 6.2. Sơ đồ tổng quan chương trình chiều đi	78
Hình 6.3. Sơ đồ tổng quan chương trình chiều về	79
Hình 6.4. Sơ đồ khối điều khiển trong chế độ Manual	80
Hình 6.5. Sơ đồ khối điều khiển trong chế độ Teleoperation	81
Hình 6.6. Sơ đồ điều khiển tự động bằng chức năng Program.....	83
Hình 6.7. Sơ đồ chương trình mô phỏng robot	83
Hình 6.8. Sơ đồ chương trình xử lý cảm biến Leap	84
Hình 6.9. Các cử chỉ ngón tay điều khiển chương trình	85
Hình 6.10. Quy đổi tọa độ tâm bàn tay về điểm cuối robot	86
Hình 6.11. Cách xác định góc kẹp từ bàn tay.....	86
Hình 6.12. Giải thuật lọc nhiễu cho cảm biến Leap	87
Hình 6.13. Phân bố vị trí tay khi giữ yên tại $x = 0$	88
Hình 6.14. Các mode chuyển động được phân loại	89
Hình 6.15. Thuật toán xét mode di chuyển.....	90
Hình 6.16. Kết quả lọc nhiễu khi có gai nhọn.....	92
Hình 6.17. Kết quả lọc nhiễu khi di chuyển thông thường.....	92
Hình 6.18. Kết quả lọc khi nhiễu đứng yên.....	93
Hình 6.19. Giải thuật chương trình Kinect.....	94
Hình 6.20. Hình ảnh thô chụp từ hai Kinect	94
Hình 6.21. Dữ liệu vật đã được tách khỏi mặt bàn	96

Hình 6.22. Dữ liệu hai PointCloud đã được ghép và xoay	97
Hình 6.23. Dữ liệu sau khi được xử lý bằng Moving Least Square	98
Hình 6.24. Kết quả tái tạo bề mặt.....	99
Hình 6.25. Cấu trúc gói tin của giao thức UDP.....	99
Hình 6.26. Nén ảnh dùng cơ chế đa luồng.....	102
Hình 6.27. Giao diện chính của chế độ RUN	102
Hình 6.28. Giao diện khối Connection.....	103
Hình 6.29. Giao diện khối Robot trong chế độ Manual	103
Hình 6.30. Giao diện bảng điều khiển Manual.....	104
Hình 6.31. Giao diện khối Robot trong chế độ Teloperation.....	104
Hình 6.32. Giao diện cảm biến Leap.....	105
Hình 6.33. Giao diện đồ hoạ môi trường 3D.....	106
Hình 6.34. Giao diện hình ảnh Camera Tool.....	107
Hình 6.35. Giao diện đồ thị miền khớp.....	107
Hình 6.36. Giao diện đồ thị miền XYZ.....	108
Hình 6.37. Giao diện chính của chế độ SIMULATION	108
Hình 6.38. Giao diện bảng điều khiển trong chế độ SIMULATION	109
Hình 7.1. Đáp ứng vị trí X	110
Hình 7.2. Đáp ứng vị trí Y	110
Hình 7.3. Đáp ứng vị trí Z.....	110
Hình 7.4. Đáp ứng góc θ	111
Hình 7.5. Đáp ứng góc ψ	111
Hình 7.6. Ứng dụng gắp vật tự động.....	112
Hình 7.7. Kết quả gắp vật tự động	112
Hình 7.8. Các khối hộp được tái tạo bề mặt.....	113
Hình 7.9. Vật được tái tạo trong không gian ảo 3D	113
Hình 7.10. Nhiễu lẫn nhau giữa hai Kinect	114

DANH MỤC BẢNG

Bảng 2.1. Thông số kỹ thuật của Tay máy RV-2AJ	11
Bảng 2.2. Bảng thông số bộ điều khiển CR-571	13
Bảng 2.3. Khung truyền của ngôn ngữ giao tiếp robot với máy tính	14
Bảng 2.4. Tập lệnh của ngôn ngữ giao tiếp robot với máy tính.....	16
Bảng 3.1. Thông số DH của robot RV-2AJ	37
Bảng 3.2. Kết quả tính toán động học thuận dùng m-file.....	46
Bảng 3.3. Kết quả tính toán động học ngược bằng biến đổi đại số dùng m-file .	47
Bảng 3.4. Kết quả tính động học ngược bằng ma trận Jacobi dùng m-file.....	47
Bảng 3.5. Kết quả tính động học thuận robot dùng mô phỏng Simmechanics..	49
Bảng 4.1. Trích bộ dữ liệu khảo sát quỹ đạo robot RV-2AJ tại $V_{\%} = 30$	56
Bảng 4.2. So sánh quỹ đạo mô phỏng với thực tế của robot RV-2AJ.....	59
Bảng 4.3. Đầu vào thử nghiệm đồng bộ các khớp	60
Bảng 5.1. Kết quả hiệu chuẩn cảm biến lực FSR402	67
Bảng 5.2. Thông số tay kẹp sau khi hoàn thành.....	71
Bảng 5.3. Vật liệu thi công Bàn làm việc.....	76
Bảng 6.1. Cấu trúc mã lệnh giao tiếp giữa Master Client và Robot Server	100
Bảng 6.2. Tập mã lệnh giao tiếp giữa Master Client và Robot Server.....	101
Bảng 7.1. Độ trễ và sai số giữa tay người và robot.....	111

DANH MỤC TỪ VIẾT TẮT

LAN	Local Area Network
PCL	Point Cloud Library

Chương 1. TỔNG QUAN ĐỀ TÀI

1.1. Sơ lược về hệ thống Teleoperation và ứng dụng trong lĩnh vực Robot

Teleoperation là thuật ngữ chỉ việc vận hành một cơ cấu máy móc từ xa, thường được sử dụng trong các môi trường nghiên cứu, học thuật và kỹ thuật.

Hệ thống Teleoperation là hệ thống thiết bị có sự tương tác ở khoảng cách khác nhau. Hệ thống cho phép con người sử dụng khả năng tư duy, sự hiểu biết và hoạt động chân tay của mình tác động vào máy móc. Trong những thập niên gần đây, hệ thống teleoperation được sử dụng trong các công việc mang tính nguy hiểm cho con người khi trực tiếp thực hiện hay các công việc khó thực hiện như trong môi trường phóng xạ, độc hại, ngoài vũ trụ...

Teleoperation bao gồm một hệ thống chủ động gọi là “Master” và một hệ thống phụ thuộc gọi là “Slave”. Người điều khiển sử dụng một hệ thống tổng thể để gửi các yêu cầu cho hệ thống Teleoperation, thông qua hệ Master truyền tín hiệu đến Slave. Tùy thuộc vào kênh thông tin, hệ thống được gọi là Song phương hoặc Đơn phương. Trong Teleoperation đơn phương, không có phản hồi nào từ hệ Slave về hệ Master, hệ Slave được thúc đẩy làm việc nhờ các tín hiệu của hệ Master. Trong Teleoperation song phương, khi nhận được tín hiệu từ hệ Master, hệ Slave có thể phản hồi tín hiệu về cho trạm Master. Tín hiệu phản hồi có thể là hình ảnh, âm thanh, vị trí, nhiệt độ, lực, vận tốc, gia tốc,...

Trong lĩnh vực Robot, Teleoperation ngày càng phát triển và trở thành một phần tất yếu nhiều hệ thống Robot nào hiện nay. Các ứng dụng và ưu điểm của Teleoperation có nhiều điểm vượt trội hơn phương pháp điều khiển nào khác.

Trong lĩnh vực Không gian, sự hiện diện trực tiếp của con người để vận hành một chiếc xe hay thiết bị nào đó đòi hỏi nhiều nguồn lực, thậm chí hoàn toàn không thể (như việc thăm dò Mặt trời), do đó sử dụng robot điều khiển từ xa là bắt buộc.



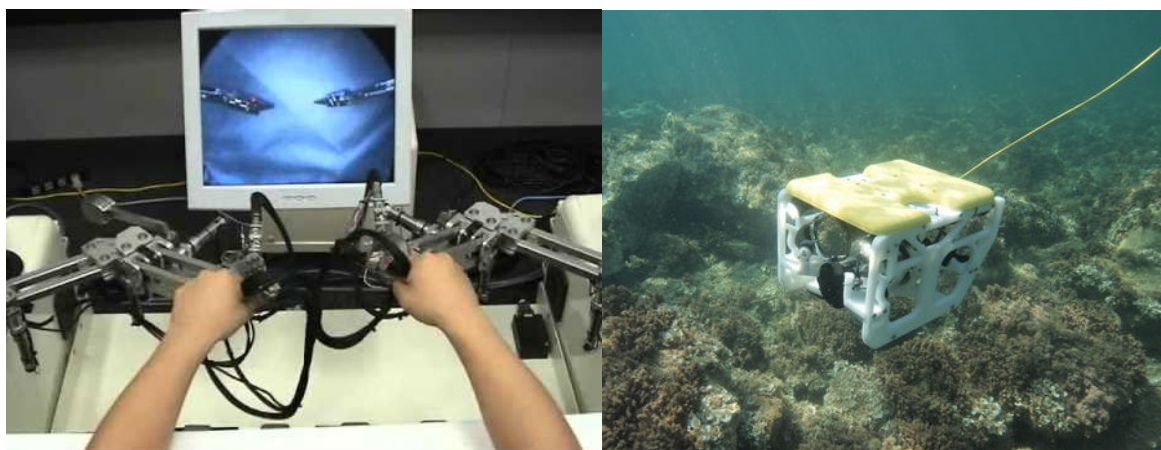
Hình 1.1. Ứng dụng Teleoperation trong lĩnh vực thám hiểm không gian

Trong lĩnh vực Quân sự, các robot điều khiển từ xa được sử dụng tham gia các nhiệm vụ trinh sát, rà phá bom mìn hay thậm chí là tham chiến. Việc này không chỉ giúp giảm đến tối thiểu các tổn thất về người mà còn tăng tính hiệu quả vì robot điều khiển từ xa có thể hoạt động ngày đêm, dễ dàng nguy trang và di chuyển linh hoạt.



Hình 1.2 Máy bay không người lái điều khiển từ xa Predator của Mỹ

Ngoài ra, các robot được điều khiển bằng phương pháp Teleoperation còn được sử dụng trong nhiều lĩnh vực khác như An ninh, Thám hiểm, Thiết bị lặn, Y học,...

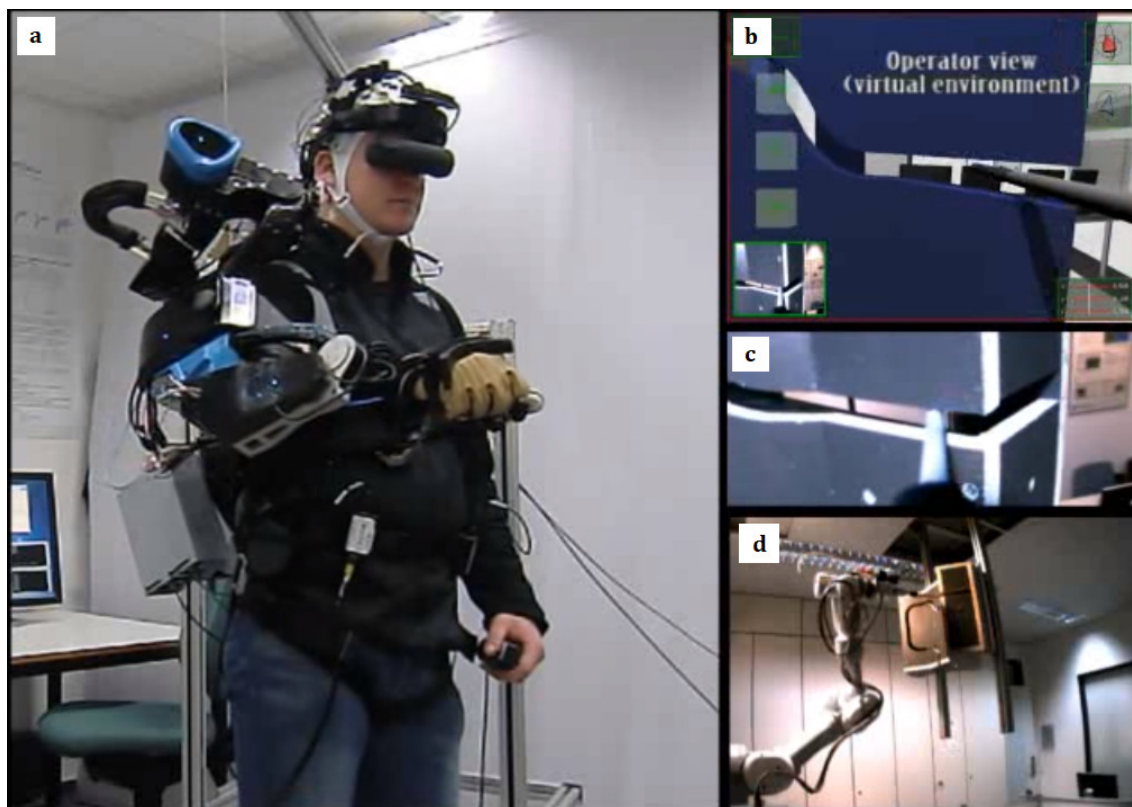


Hình 1.3. Ứng dụng Teleoperation trong Phẫu thuật từ xa và thiết bị lặn ROV

1.2. Các hệ thống điều khiển từ xa sao chép cử động người điều khiển hiện nay

Hiện nay trên thế giới, mảng Teleoperation, đặc biệt là Robotic Arm Teleoperation rất được chú trọng nghiên cứu và phát triển vì các ưu thế và ứng dụng trong thực tiễn.

Một ví dụ điển hình là dự án VI-Bot được phát triển bởi Trung tâm nghiên cứu Trí tuệ Nhân tạo Đức (German Research Center for Artificial Intelligence), được tài trợ bởi Bộ Giáo dục và Khoa học Liên bang thuộc Cộng Hòa Liên bang Đức. Hệ thống VI-Bot bao gồm một cánh tay cơ khí có thể gắn vào tay người điều khiển được kết nối với trạm Master, một tay máy thực thi được kết nối với trạm Slave. Ngoài ra tại trạm Slave còn có các camera được bố trí để thu thập dữ liệu, tái tạo lại môi trường ảo để người điều khiển tại trạm Master có thể quan sát khu vực làm việc của tay máy.



Hình 1.4. Dự án VI-Bot đang vận hành

- | | |
|------------------------------------|--------------------------------|
| a. Người vận hành | b. Môi trường thực tế ảo |
| c. Hình ảnh làm việc thu trực tiếp | d. Không gian Tay máy làm việc |

Một dự án khác được phát triển bởi Đại học Bắc Kinh, Trung Quốc – Hệ thống Robot Tìm kiếm và Cứu hộ Thông minh (Intelligent Search and Rescue Teleoperation Robot System) – là một ví dụ khác cho thấy sự đầu tư của các Quốc gia cho lĩnh vực này. Hệ thống nay bao gồm hai tay máy UR5 của hãng Universal Robot được gắn cố định trên thiết bị di chuyển GUARDIAN do Đại học Bắc Kinh chế tạo.



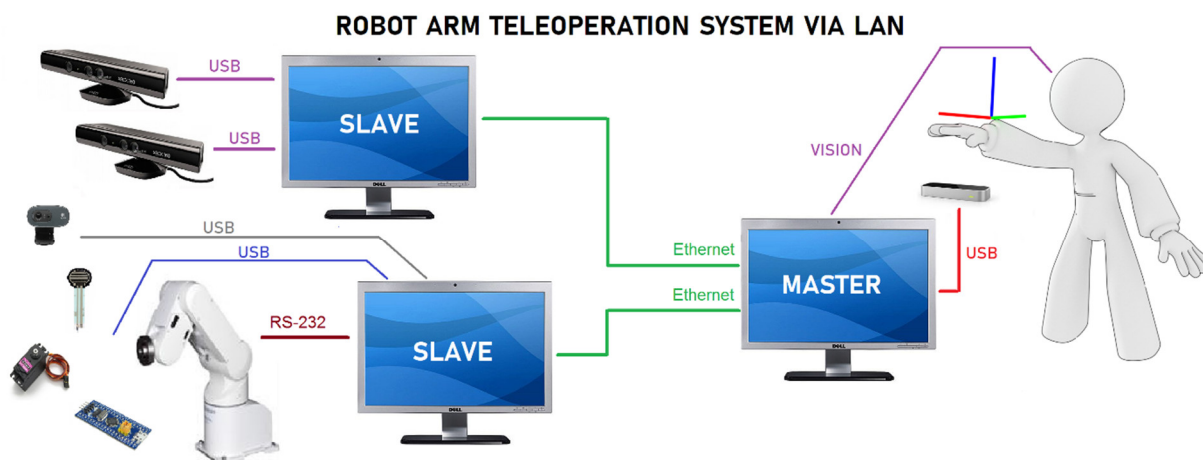
Hình 1.5. Dự án Robot Tìm kiếm và Cứu hộ Thông minh

Ở Việt Nam, mảng Teleoperation đang được nghiên cứu và phát triển mạnh. Các trường đại học khối kỹ thuật những năm gần đây có nhiều luận văn, đề tài nghiên cứu khoa học liên quan đến Teleoperation.



Hình 1.6. Luận văn tốt nghiệp của Sinh viên Đại học Bách Khoa Hà Nội

1.3. Tổng quan đề tài



Hình 1.7. Sơ đồ minh họa tổng quan hệ thống

Hệ thống điều khiển từ xa tay máy RV-2AJ chia thành hai phía: trạm Master và trạm Slave. Các máy tính giữa hai trạm liên kết với nhau thông qua mạng Ethernet LAN, giao thức UDP.

Tại trạm Master có:

- Máy tính “Master” hiển thị giao diện điều khiển, hiển thị môi trường ảo khu vực làm việc của tay máy, nhận và xử lý dữ liệu từ cảm biến Leap Motion, truyền dữ liệu điều khiển tới trạm Slave.
- Cảm biến Leap Motion theo dõi hành động bàn tay của người dùng và gửi về máy tính “Master”.

Tại trạm Slave có:

- Máy tính “Robot” giao tiếp với Controller CR1-571 điều khiển tay máy RV-2AJ, thu thập dữ liệu từ camera 2D, điều khiển tay kẹp và gửi dữ liệu trạng thái và lỗi của tay máy, dữ liệu hình ảnh về trạm Master.
- Máy tính “Kinect” thu thập và xử lý dữ liệu từ hai Kinect, truyền hình ảnh 3D về trạm Master.

- Tay kẹp – dụng cụ công tác tại điểm cuối của tay máy – cấu tạo từ các chi tiết được in 3D, Kit phát triển STM32F103C8T6, động cơ RC Servo TowerPro MG996R và cảm biến lực Interlink FSR402.
- Camera 2D Logitech 270 thu thập hình ảnh từ điểm cuối tay máy gửi về máy tính “Robot”.
- Hai camera Microsoft Kinect V1 thu thập hình ảnh 3D gửi về máy tính “Kinect”.

Sau khi hoàn thành, hệ thống cho phép người dùng điều khiển tay máy từ xa tại trạm Master. Người dùng có thể quan sát được khu vực làm việc của tay máy thông qua môi trường ảo 3D.

1.4. Cấu trúc trình bày luận văn

Chương 1: TỔNG QUAN ĐỀ TÀI

Giới thiệu hệ thống Teleoperation, các nghiên cứu trong và ngoài nước và tổng quan hệ thống sinh viên đã xây dựng.

Chương 2: GIỚI THIỆU THIẾT BỊ VÀ PHẦN MỀM

Giới thiệu các thiết bị phần cứng và các phần mềm, thư viện được sử dụng khi thực hiện luận văn.

Chương 3: ĐỘNG HỌC ROBOT RV-2AJ

Trình bày các tính toán Động học thuận, Động học ngược cho tay máy Mitsubishi RV-2AJ, mô phỏng MATLAB để kiểm chứng kết quả tính toán.

Chương 4: HOẠCH ĐỊNH QUỸ ĐẠO ROBOT RV-2AJ

Trình bày thuật toán hoạch định quỹ đạo, đồng bộ chuyển động giữa các góc khớp, thực hiện mô phỏng MATLAB để kiểm chứng.

Chương 5: THIẾT KẾ VÀ THI CÔNG THIẾT BỊ HỖ TRỢ

Trình bày các thiết kế và kết quả thi công các thiết bị hỗ trợ cho hệ thống như dụng cụ công tác (tay kẹp), Bàn làm việc, bố trí camera Kinect.

Chương 6: CHƯƠNG TRÌNH MÁY TÍNH

Trình bày các chương trình máy tính điều khiển tay máy RV-2AJ, chương trình xử lý cảm biến Leap Motion, chương trình xử lý cảm biến Kinect, mạng giao tiếp UDP, giới thiệu giao diện người dùng.

Chương 7: KẾT QUẢ VÀ KẾT LUẬN.

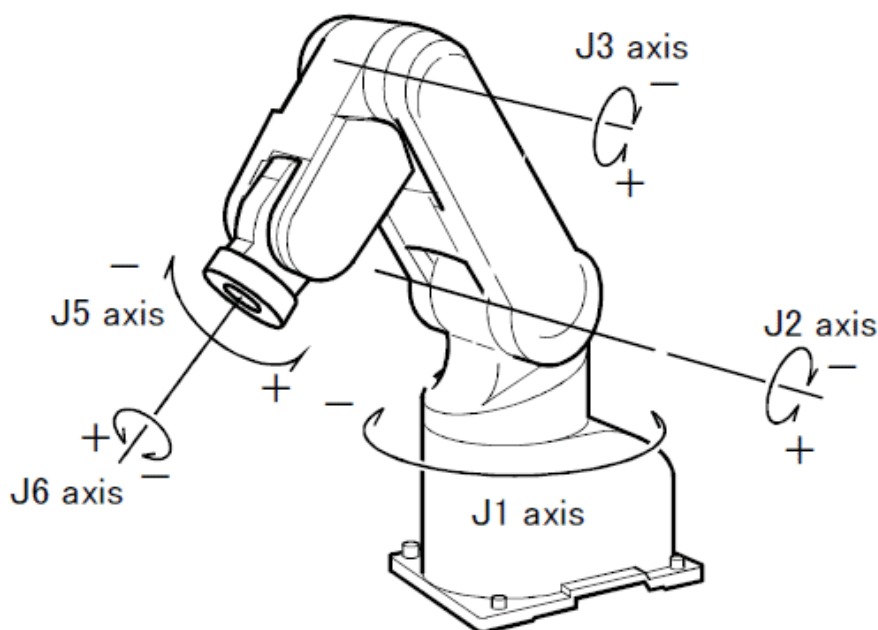
Trình bày các hình ảnh, số liệu kết quả demo của hệ thống, một số đánh giá, kết luận và hướng phát triển của đề tài.

Chương 2. GIỚI THIỆU THIẾT BỊ VÀ PHẦN MỀM

2.1. Giới thiệu thiết bị

2.1.1. Tay máy Mitsubishi RV-2AJ

RV-2AJ là cánh tay robot 5 bậc tự do của hãng Mitsubishi, Nhật Bản sản xuất. Đi kèm với RV-2AJ là bộ điều khiển CR1-571 và bộ lập trình bằng tay R28TB.



Hình 2.1. Cấu hình các khâu và khớp của Tay máy RV-2AJ

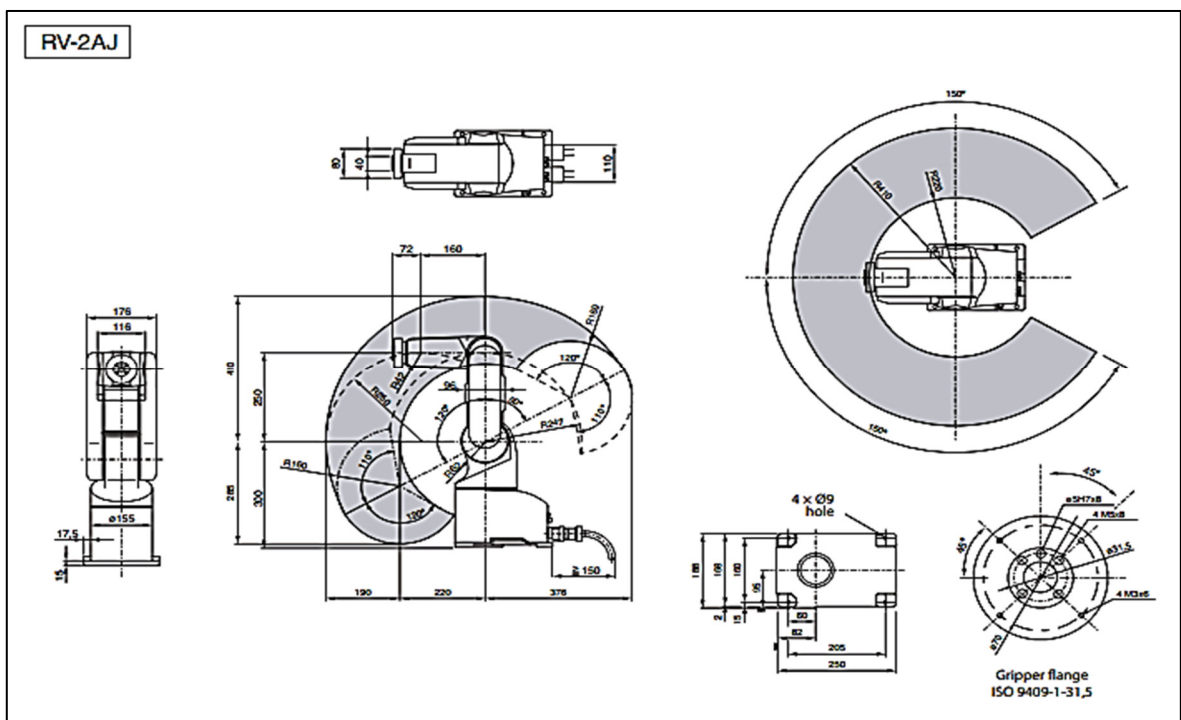
RV-2AJ được điều khiển bởi bộ điều khiển CR1-571 trang bị CPU kiến trúc RISC/DSP 64bit ngôn ngữ lập trình MELFA-BASIC IV hoặc MOVEMASTER COMMANDS. Phần mềm giao diện đi kèm là COSIROP dùng để giao tiếp giữa robot với máy tính và lập trình điều khiển. Trên dây chuyền FMS, các tọa độ vị trí và chuyển động của robot RV-2AJ được lập trình trước bằng phần mềm COSIROP và nạp vào bộ điều khiển.

Characteristics/Functions	RV-2AJ
Degrees of freedom (no. of axes)	5
Installattion posture	On floor, hanging

Structure		Vertical multiple-joint type
Drive system		AC servo (J1, J2, J3 and J5-axies: with brake; J4 and J6 axes: without brake)
Position detection method		Absolutely encoder
Arm length (mm)	Shoulder shift	0
	Upper arm	250
	Fore arm	160
	Elbow shift	0
	Wrist length	72
Operating range (degree)	Waist (J1)	300 (-150 to +150)
	Shoulder (J2)	180 (-60 to +120)
	Elbow(J3)	230 (-110 to +120)
	Wrist twist (J4)	-----
	Wrist pitch (J5)	180 (-90 to +90)
	Wrist roll (J6)	400 (-200 to +200)
Maximum speed (degree/s)	Waist (J1)	180
	Shoulder (J2)	90
	Elbow (J3)	135
	Wrist twist (J4)	-----
	Wrist pitch (J5)	180
	Wrist roll (J6)	210
Maximum composite speed (mm/s)		2200
Payload capacity (kg)	Rated	1.5
	Maximum	2
Position repeatability (mm)		±0.02
Ambient temperature (°C)		0 to 40
Weight (kg)		17
Tolerable moment (Nm)	Wrist twist (J4)	-----
	Wrist pitch (J5)	2.16

	Wrist roll (J6)	1.10
Tolerable inertia (kgm ²)	Wrist twist (J4)	-----
	Wrist pitch (J5)	3.24×10^{-2}
	Wrist roll (J6)	8.43×10^{-3}
Arm reachable radius (to the center point of the J5 axis) (mm)		410
Potection rating		IP 30
Robot controller		CR1

Bảng 2.1. Thông số kỹ thuật của Tay máy RV-2AJ

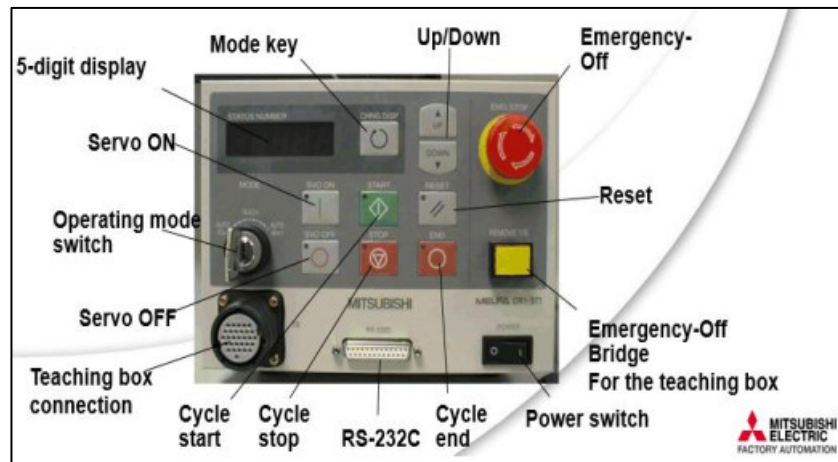


Hình 2.2. Không gian làm việc của Tay máy RV-2AJ

- Bộ điều khiển CR-571**

Bộ điều khiển CR-571 là bộ điều khiển đi kèm tay máy RV-2AJ, có kích thước nhỏ gọn, tích hợp nhiều chức năng, cung cấp các thao tác từ đơn giản đến phức tạp. Các chức năng của bộ điều khiển CR-571 không chỉ gói gọn trong việc điều khiển chính xác tay máy RV-2AJ, mà còn tương thích với hầu hết tay máy MELFA.

Không những vậy, bộ điều khiển cho phép tay máy tăng thêm 2 trục so với thiết kế, đồng thời điều khiển thêm 6 động cơ độc lập.



Hình 2.3. Bộ điều khiển CR-571

Bộ điều khiển CR-571 hỗ trợ 2 ngôn ngữ lập trình là MELFA BASIC IV và MOVEMASTER COMMAND, bộ nhớ chương trình lớn, tốc độ xử lý nhanh.

Về phần cứng, bộ điều khiển CR-571 cho phép kết nối 16 input và 16 output ngoài, có thể mở rộng thêm 240 input và 240 output với Expansion Box, phục vụ nhu cầu đồng bộ dây chuyền sản xuất và mở rộng chức năng.

Đặc điểm		Thông số
Số trục có khả năng điều khiển		Robot 5 trục và 6 trục
Bộ xử lý (CPU)		Main CPU: 64 bits RISC
		Servo CPU: DSP
Bộ nhớ	Số điểm tối đa	2500
	Số bước chương trình	5000
	Số chương trình tối đa	88
Ngôn ngữ lập trình		MELFA BASIC IV MOVEMASTER COMMAND
Cổng nhập/ cổng xuất	GPIOs	16 ngõ vào và 16 ngõ ra
	Đóng mở tay gấp	8 ngõ vào và 4 ngõ ra

Đặc điểm		Thông số
	Ngắt khẩn cấp	1
	Công tắc cửa	1
Giao tiếp	RS232C/USB	1 cổng cho PC
	RS442 Ethernet cho thiết bị cầm tay	1 cổng cho Teaching Box
	Robot I/O link	1 kênh (tối đa 240 ngõ vào và 240 ngõ ra)
Nguồn điện		1 pha; 90-132 VAC; 50-60 Hz; 0.7 kVA
		1 pha; 180-253 VAC; 50-60 Hz; 0.7 kVA
Kích thước		212 x 166 x 290 mm
Khối lượng		8 kg

Bảng 2.2. Bảng thông số bộ điều khiển CR-571

- **Ngôn ngữ giao tiếp giữa robot và máy tính**

Ngôn ngữ giao tiếp robot và máy tính được thiết kế dựa trên ngôn ngữ MELFA BASIC, nhưng không hoàn toàn giống MELFA BASIC, được dùng để gửi lệnh hoặc gửi chương trình xuống robot.

Ngôn ngữ giao tiếp quy định khung truyền giữa robot và máy tính, gồm các lệnh bật tắt động cơ servo, xoay góc khớp, tịnh tiến điểm cuối theo trục XYZ, dịch chuyển điểm cuối đến tọa độ yêu cầu, tải chương trình điều khiển.

Ngôn ngữ giao tiếp không được nhà sản xuất công bố. Do đó, để xây dựng phần mềm điều khiển robot bằng máy tính, ta cần giải mã khung truyền và tập lệnh của ngôn ngữ này. Việc này được thực hiện bằng cách quan sát các gói tin giao tiếp giữa phần mềm điều khiển bằng máy tính do nhà sản xuất cung cấp và robot thông qua cổng COM. Ta thu thập được giao thức giữa robot và máy tính như trong các bảng sau.

Gửi lệnh : PC → Robot	Phản hồi: Robot → PC
1;<k>;<command>\r	Thành công: QoK<info>\r Có lỗi: QeR<error>\r
Ý nghĩa	
<k> Là vị trí của chương trình đích trên controller mà lệnh gửi đến <command> Là lệnh cần thực thi. \r Là kí tự kết thúc khung (mã ASCII là 0x0D)	<info> Là thông tin phản hồi bổ sung, tùy vào lệnh từ máy tính. <error> Là mã số của lỗi xảy ra. \r Là kí tự kết thúc khung (mã ASCII là 0x0D)

Bảng 2.3. Khung truyền của ngôn ngữ giao tiếp robot với máy tính

Nội dung	<command>	<info>	<error>
Initialize the communication protocol. Get the robot information.	OPEN=NARCUSR	1F;37;6,0;3,5,A,1E ,32,46,64;MB4;PR M;RV-2AJ;CRn- 5xx;MELFA;05- 06- 14;Ver.K4;ENG;CO PYRIGHT(C)1999- 2005 MITSUBISHI ELECTRIC CORPORATION ALL RIGHTS RESERVED;1;1;8;	none
Get the status of robot	STATE	<program>;140;70; 4;<servo	none

		<i>status>1060;6;1;;;;; ;;;CYC;START;30;0</i>	
Reset alarm to reactive robot	RSTALRM	<i>none</i>	<i>none</i>
Turn ON/ OFF remote control over serial interface	CNTLON CNTLOFF	<i>none</i>	<i>none</i>
Turn ON/ OFF the servo motors.	SRVON SRVOFF	<i>none</i>	<i>none</i>
Close/Open the gripper	EXECHCLOSE <tool> EXECHOPEN <tool>	<i>none</i>	<i>none</i>
Establish JCOSIROP variable with coordinate values.	EXECJCOSIROP=(J1,J2,J3,J4,J5,J6)	<i>none</i>	<i>none</i>
Establish PCOSIROP variable with position values.	EXECPCOSIROP=(X,Y,Z,A,B,C)(<i><footer></i>)	<i>none</i>	<i>none</i>
Move joints to JCOSIROP value	EXECMOV JCOSIROP	<i>none</i>	564000000: commands are sent so quickly that robot haven't finish the last command.
Move endpoint linearly to PCOSIROP value	EXECMVS PCOSIROP	<i>none</i>	
Set the speed of linear movement	EXECSPD <mm/s>	<i>none</i>	<i>none</i>
Set the speed of joint movement	EXECJOVRD <%>	<i>none</i>	<i>none</i>

Get the current joint coordinate of robot	JPOSF	J1;<j1>;J2;<j2>;J3;<j3>;J4;****;J5;<j5>;J6;<j6>;****,****;70;0.00;00000000	none
Get the current XYZ position of robot	PPOSF	X;<x>;Y;<y>;Z;<z>;A;<roll>;B;<pitch>;6,0;70;0.00;000000000	none

Bảng 2.4. Tập lệnh của ngôn ngữ giao tiếp robot với máy tính

2.1.2. Cảm biến Leap Motion

Leap Motion là thiết bị cảm biến cử chỉ bàn tay và ngón tay, kích thước bằng một chiếc USB, được phát triển bởi công ty Leap Motion, ra mắt lần đầu tiên vào năm 2013.

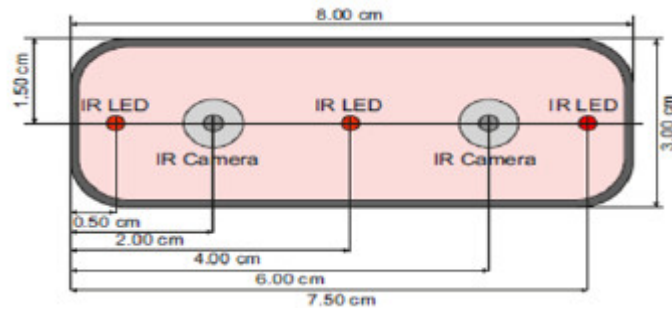


Hình 2.4. Cảm biến Leap Motion

- **Cấu tạo cảm biến Leap Motion**

Cảm biến Leap Motion được đóng gói trong vỏ hợp kim nhôm, mặt trên được che phủ bởi tấm kính có màu tối. Đây chính là bề mặt hoạt động của cảm biến.

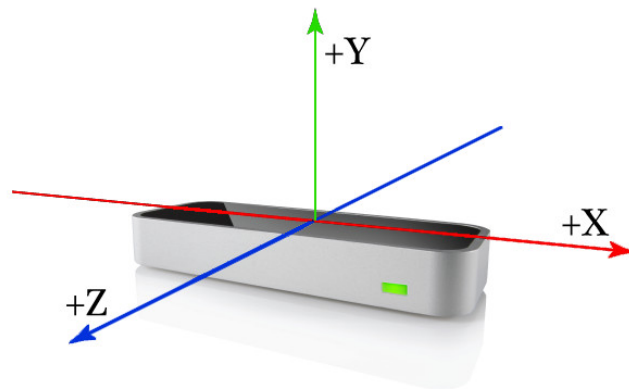
Bên trong cảm biến Leap Motion là 2 camera hồng ngoại và 3 đèn LED hồng ngoại bố trí xen kẽ nhau hướng lên mặt kính.



Hình 2.5. Cấu tạo cảm biến Leap Motion

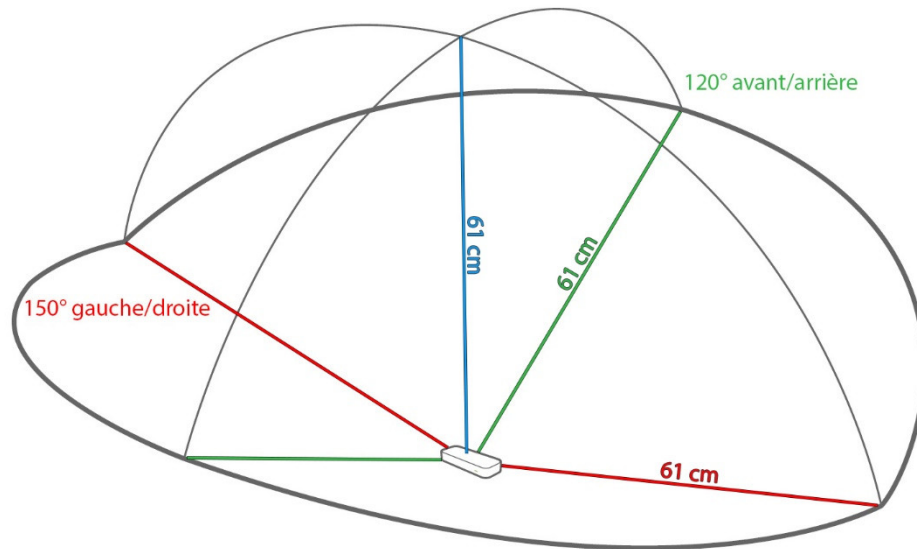
- **Trục tọa độ và không gian làm việc của cảm biến Leap Motion**

Cũng như mọi cảm biến độ sâu, Leap Motion có 3 trục tọa độ XYZ. Các trục tọa độ được bố trí như hình sau, với trục Y vuông góc mặt trên cảm biến, trục X theo phương chiều dài cảm biến và trục Z theo phương chiều rộng cảm biến.



Hình 2.6. Trục tọa độ của cảm biến Leap Motion

Không gian làm việc của cảm biến nằm phía trên mặt kính. Leap Motion cung cấp một không gian làm việc khá rộng, gần một nửa bán cầu, bán kính lên đến 0.6 mét và góc mở lên đến 150 độ.



Hình 2.7. Không gian làm việc của cảm biến Leap Motion

- **Nguyên lý hoạt động của cảm biến Leap Motion**

Cảm biến Leap Motion sử dụng 3 đèn LED hồng ngoại và 2 camera hồng ngoại để thu hình ảnh trong không gian 3 chiều.

Thuật toán trong cảm biến Leap Motion sử dụng mô hình bàn tay để cung cấp khả năng theo dõi, tiên đoán vị trí ngay cả khi không nhìn thấy được toàn bộ các phần của bàn tay. Từ các phần nhìn thấy của bàn tay và vị trí đã quan sát trong quá khứ, thiết bị sẽ tính toán các vị trí có khả năng nhất của các phần bị che khuất.

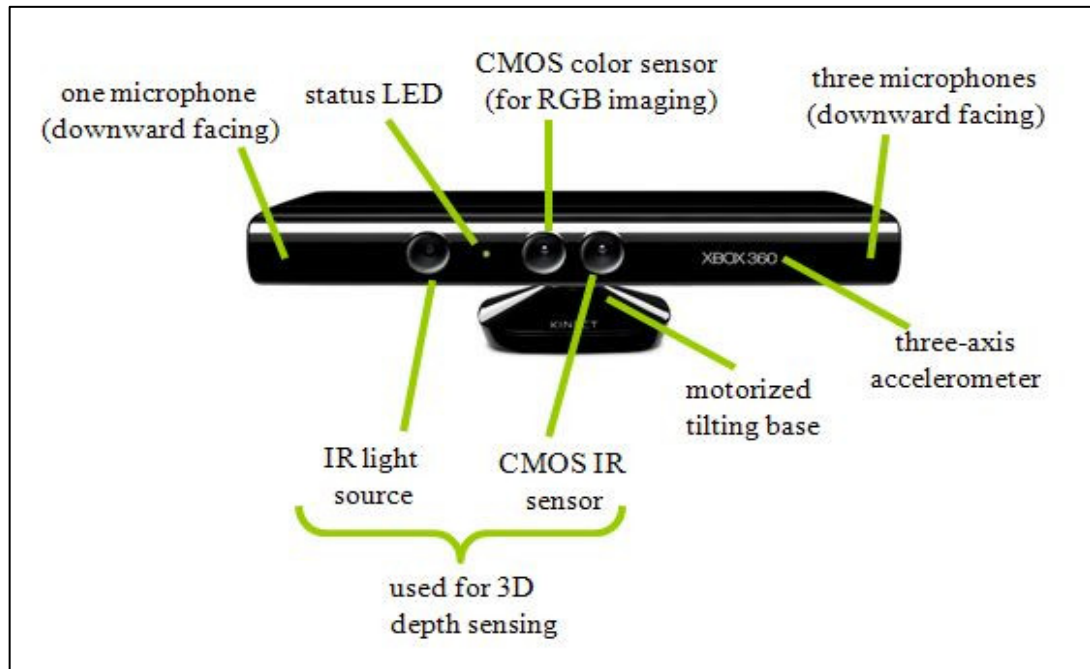
Hơn nữa, nhà phát triển – công ty Leap Motion – cung cấp hai bộ thư viện nguồn mở V2 và Orion, hỗ trợ cho lập trình viên nghiên cứu và phát triển các ứng dụng riêng biệt trên nhiều nền tảng thông dụng như C, C#, C++, Unity, Python, Java, Objective-C, ...

2.1.3. Camera Microsoft Kinect

Kinect là thiết bị cảm biến ngoại vi thu chuyển động phát triển bởi Microsoft dành cho máy console Xbox 360 và Windows. Kinect cho phép người

dùng có thể điều khiển và tương tác với máy console mà không cần thiết bị cầm tay, thông qua trực tiếp sử dụng động tác tay chân, thân thể hoặc lệnh thoại.

- **Cấu tạo của camera Kinect**



Hình 2.8. Cấu tạo của camera Kinect

Một bộ phát hồng ngoại (Infrared light) và một camera hồng ngoại (Depth image CMOS): Thu thập dữ liệu độ sâu của ảnh theo kỹ thuật Structure light – phát hiện độ sâu bằng cách chiếu một dải mẫu lên vật và dựa vào độ méo của dải mẫu để tính khoảng cách tới vật.

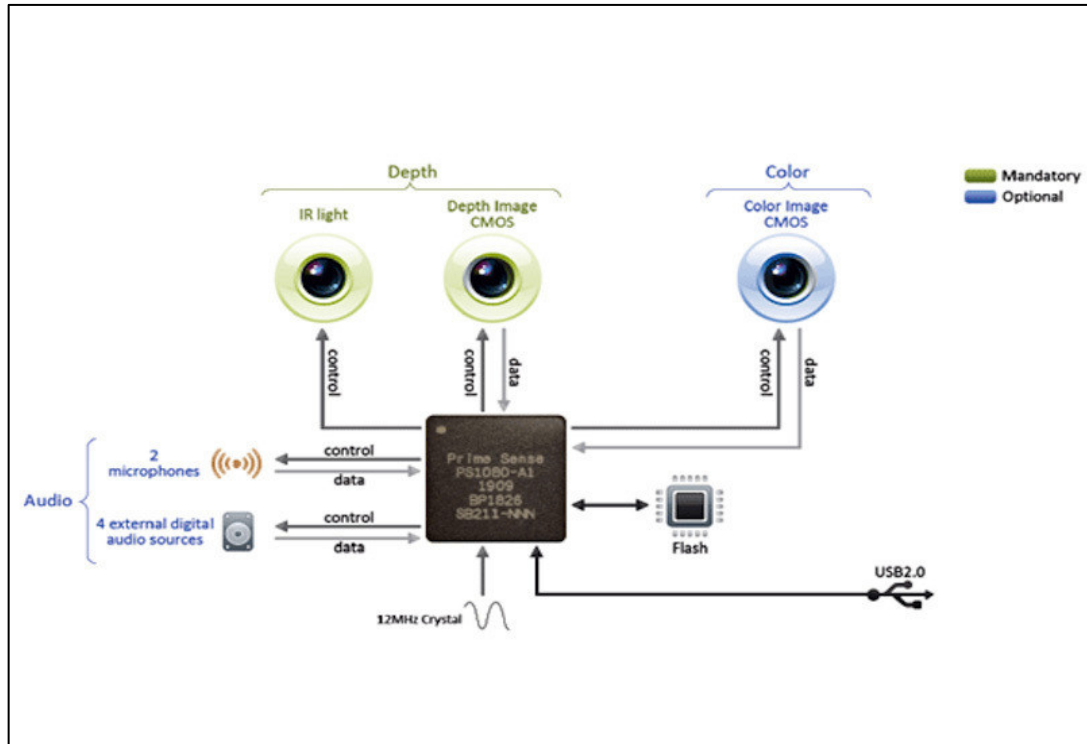
Một camera RGB: Thu thập dữ liệu màu của ảnh. Chất lượng dữ liệu Kinect thu thập có thể được tùy chỉnh dựa vào mục đích của người dùng.

- Kích thước 320x240 pixels, tốc độ 60 khung hình/giây.
- Kích thước 640x480 pixels, tốc độ 30 khung hình/giây.
- Kích thước 1280x960 pixels, tốc độ 60 khung hình/ giây.

Động cơ điều hướng: Được đặt tại điểm nối camera Kinect và phần đế, giúp Kinect có thể hướng đầu lên cao và hạ xuống thấp một góc 27 độ.

Máy đo gia tốc: Dùng để xác định tốc độ của các chuyển động.

Micro: Bao gồm 4 micro nhỏ, hoạt động như nhau ở 16 bits và tốc độ lấy mẫu là 16kHz.



Hình 2.9. Sơ đồ khối của cảm biến Kinect

2.1.4. Camera Logitech C270



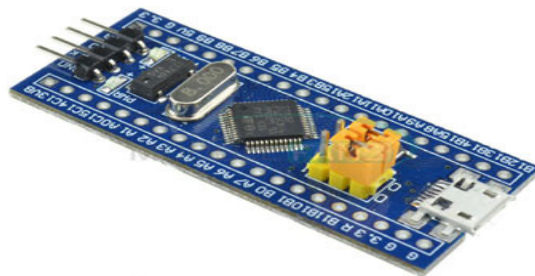
Hình 2.10. Camera Logitech C270

Logitech C270 là dòng webcam được sản xuất bởi Logitech – tập đoàn hàng đầu thế giới về các sản phẩm phụ kiện cho máy tính cá nhân và máy tính xách tay.

Camera C270 mang những ưu điểm vượt trội so với các dòng camera khác cùng tầm giá như:

- Độ phân giải ảnh tĩnh lên đến 3.0 Megapixels
- Độ phân giải video lên đến 1280x720 pixels, đạt chuẩn HD
- Cảm biến hình ảnh công nghệ RightLight cung cấp hình ảnh chất lượng cả trong điều kiện thiếu sáng.
- Micro hạn chế tạp âm từ môi trường.
- Kết nối USB 2.0
- Cài đặt driver và phần mềm hỗ trợ dễ dàng.

2.1.5. Kit phát triển STM32F103C8T6



Hình 2.11. Kit phát triển STM32F103C8T6

Kit phát triển STM32F103C8T6 là Kit ra chân đầy đủ sử dụng vi điều khiển STM32F103C8T6. Trên Kit có tích hợp sẵn thạch anh 8MHz, 32kHz, cổng USB để cấp nguồn 5V, IC nguồn chuyển đổi 5V sang 3V3 cung cấp cho vi điều khiển.

2.1.6. Động cơ RC Servo TowerPro MG996R



Hình 2.12. Động cơ TowerPro MG996R

TowerPro MG996R là dòng động cơ thiên về tốc độ xử lý nhanh và lực kéo mạnh, thường được sử dụng trong các thiết kế Robot như xe dò line, cánh tay Robot,...

Thông số hoạt động của động cơ:

- Chủn g loại: Analog RC Servo
- Điện áp hoạt động: 4.8 – 6.6 Volts
- Kích thước: 40x19x43 milimets
- Lực kéo: 11kg-cm ở 6 Volts
- Tốc độ quay không tải: 0.15 giây/60 độ

2.1.7. Cảm biến lực Interlink FSR402



Hình 2.13. Cảm biến lực Interlink FSR402

Cảm biến lực Interlink FSR402 là cảm biến lực dùng để đo áp lực tại một điểm thông qua sự thay đổi trở kháng giữa hai miếng tiếp xúc của cảm biến.

Thông số cảm biến:

- Bề mặt cảm ứng: Đường kính tròn 13 milimets
- Tín hiệu trả về: Điện trở
- Tầm đo: 0.1-10 Newton
- Độ dày: 0.45 milimets
- Độ bền: 10 triệu lần nhấn

2.2. Giới thiệu phần mềm

2.2.1. Nền tảng Qt – C++

C++ là một ngôn ngữ lập trình bậc trung dùng để viết các ứng dụng máy tính, được dùng phổ biến trong thương mại. Về cơ bản, C++ là sự nâng cao của ngôn ngữ C, được bổ sung các khái niệm lớp (class), thừa kế, đa hình, tính đóng gói, tính trừu tượng... Về tư duy tiếp cận, C++ vừa được tiếp cận theo lập trình cấu trúc (lập trình hướng thủ tục), vừa được tiếp cận theo lập trình hướng đối tượng.

Các ưu điểm của C++ là thực thi nhanh, đa nền tảng (chạy được trên nhiều hệ điều hành: window, linux, mac OS...), quy mô phát triển lớn và nhiều thư viện hỗ trợ đồ hoạ.

Qt là một nền tảng (framework) sử dụng ngôn ngữ C++, có mã nguồn mở, hướng đối tượng, đa nền tảng. Qt được dùng để lập trình giao diện (GUI) bằng cách kéo thả với nhiều class đồ hoạ trực quan, mỗi class chứa nhiều biến và hàm thực thi được viết sẵn. Bên cạnh các ứng dụng máy tính, Qt hỗ trợ viết ứng dụng trên các thiết bị màn hình cảm ứng.



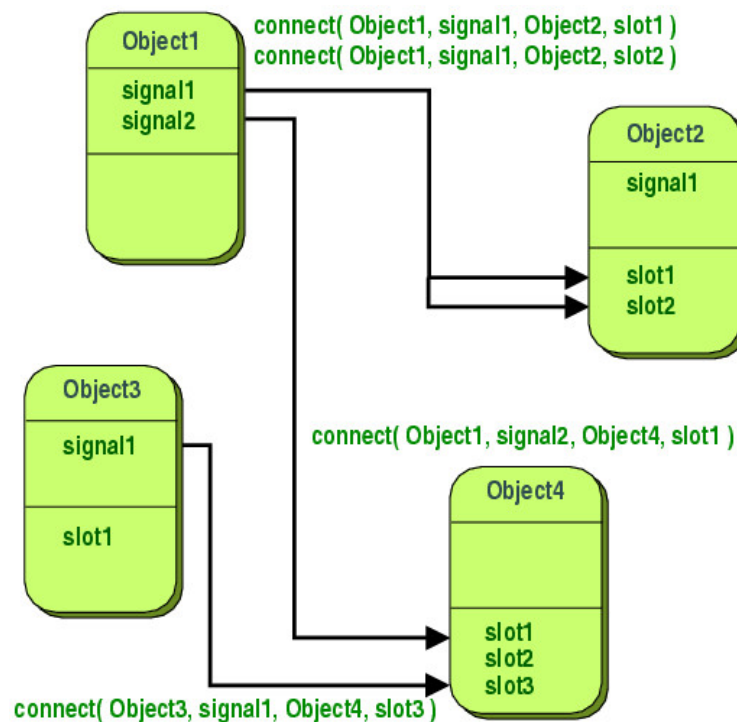
Hình 2.14. Logo Qt C++

- **Signal và Slot**

Một trong những điểm mạnh của Qt là khái niệm Signal và Slot. Signal và Slot của Qt là tính năng phục vụ việc rẽ nhánh chương trình. Signal là một tín

hiệu được kích để thông báo một sự kiện xảy ra. Slot là một hàm được gọi để đáp ứng cho Signal mà nó có liên kết. Signal liên kết với Slot bằng phương thức Connect().

Tính năng Signal & Slot rất linh hoạt. Đối tượng kích Signal và đối tượng gọi Slot có thể là hai đối tượng khác nhau. Một Signal có thể liên kết với nhiều Slot và ngược lại. Liên kết giữa Signal và Slot có thể đi xuyên từ backend đến frontend, từ một thread đến thread khác.

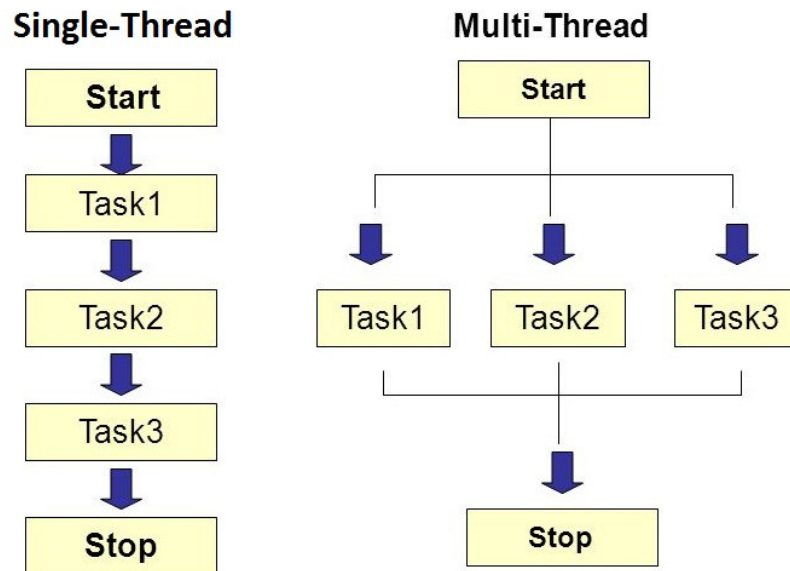


Hình 2.15. Hoạt động của Signal & Slot

- **Multithread**

Multithread - đa luồng là cơ chế lập trình nhằm thực hiện các tác vụ một cách song song, nhờ đó tăng tốc độ xử lý chương trình. Khi máy tính thực hiện giao tiếp với những thiết bị khác, Multithread giúp tránh việc chương trình chính bị dừng trong thời gian chờ truyền nhận thông tin. Nhờ vậy, chương trình có thể đáp ứng trong thời gian thực (realtime).

Trong Qt, cơ chế đa luồng được thực hiện bằng lớp QThread. Trước tiên một luồng mới được khởi tạo ở dạng QThread, sau đó đối tượng cần phân luồng được chuyển vào trong luồng. Việc gọi các tác vụ của đối tượng bên trong luồng từ những luồng bên ngoài được thực hiện thông qua liên kết Signal & Slot.

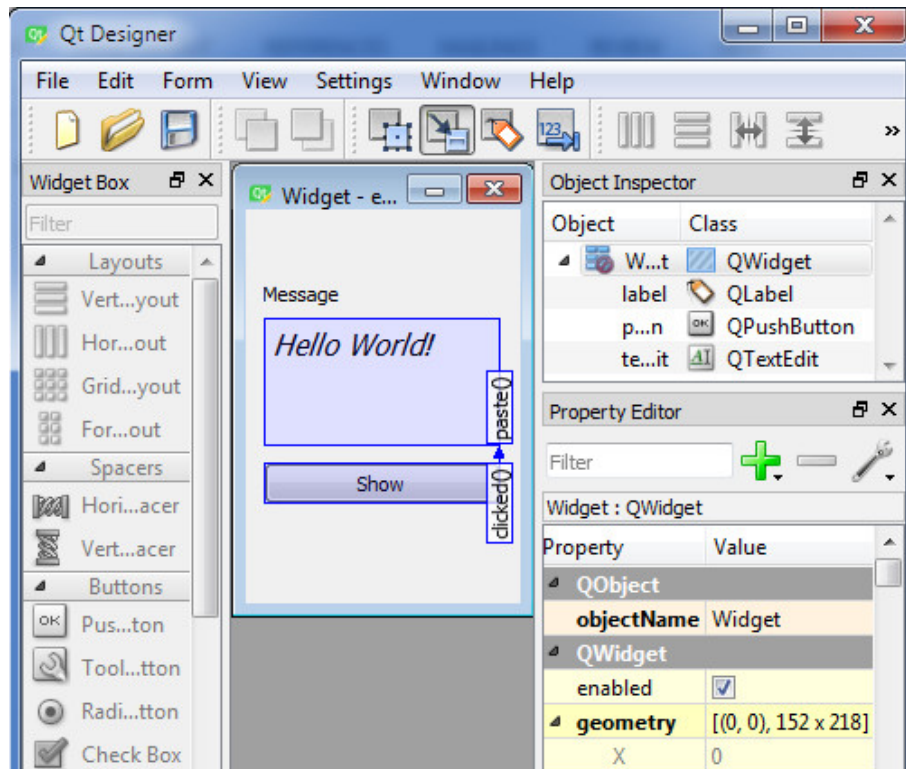


Hình 2.16. So sánh chương trình đơn luồng và đa luồng

- **Giao diện**

Qt hỗ trợ đầy đủ cho việc lập trình giao diện (GUI). Mỗi đối tượng trong giao diện được thừa kế một lớp quản lý các thuộc tính của mình. Cửa sổ giao diện có lớp quản lý chung là QWidget hoặc QMainWindow. Các phần tử trong giao diện có lớp quản lý riêng, ví dụ với nút nhấn là QPushButton, nhãn là QLabel, ô chữ là QTextEdit chữ...

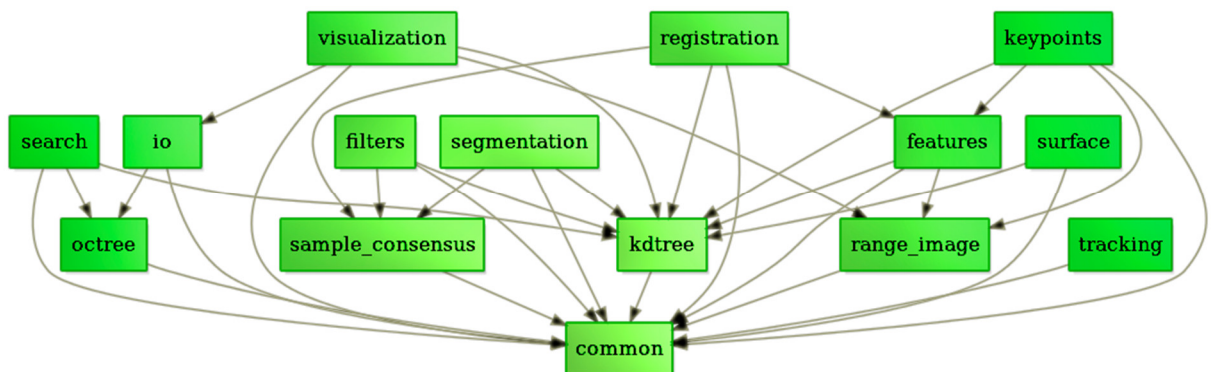
Về phía người dùng, Qt cung cấp cửa sổ QtDesigner giúp bố trí các phần tử trên giao diện theo kiểu kéo thả. Tính năng dàn trang (layout) cho phép các phần tử di chuyển mà vẫn giữ vị trí tương đối với nhau khi cửa sổ bị thu phóng. Tính năng Edit Signal & Slot cho phép nhanh chóng thực hiện việc liên kết giữa các signal & slot mặc định của các phần tử.



Hình 2.17. Cửa sổ lập trình giao diện QtDesigner

2.2.2. Thư viện xử lý ảnh 3D PointCloud Library

PointCloud Library (PCL) là thư viện mã nguồn mở hỗ trợ việc xử lý dữ liệu 3D rất mạnh được phát triển từ năm 2010, được xây dựng với nhiều module thực hiện các thuật toán như: Lọc (filtering), Khôi phục bề mặt (Surface reconstruction), Phân vùng (Segmentation), Ước lượng đặc tính (Feature estimation)...

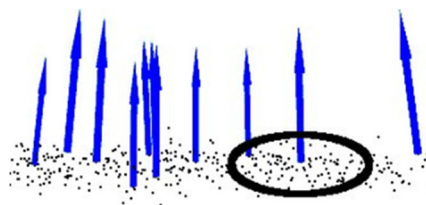


Hình 2.18. Cấu trúc thư viện PCL

- **Tổng quan các module thuộc thư viện PCL**

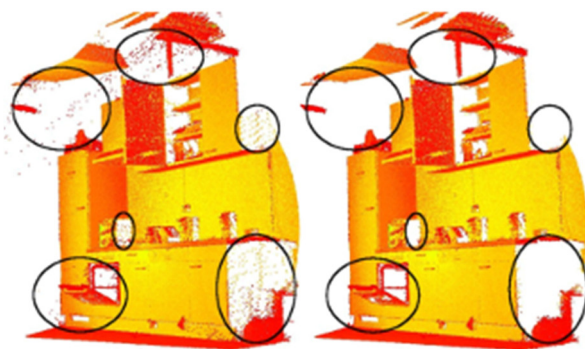
- **Module Common:** Chứa cấu trúc dữ liệu và phương thức được sử dụng bởi phần lớn các thư viện trong PCL. Cấu trúc dữ liệu cốt lõi trong PCL là các class PointCloud, các loại dữ liệu biểu diễn điểm, bề mặt, giá trị màu, mô tả tính năng,...

- **Module Features:** Chứa các cấu trúc dữ liệu và cơ chế tính toán, ước lượng 3D từ các dữ liệu điểm PCD. 3D Features biểu diễn chính xác điểm 3D hoặc vị trí trong không gian để mô tả phần hình khối dựa vào thông tin có được xung quanh điểm. Vùng dữ liệu được chọn lân cận điểm truy vấn thường được gọi là K-neighborhood.



Hình 2.19. Biểu diễn Vector mô tả điểm trong không gian

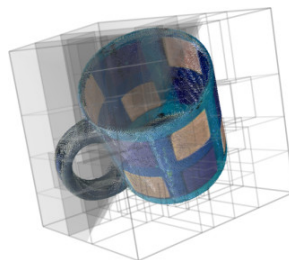
- **Module Filters:** Chứa các kỹ thuật loại bỏ nhiễu, bộ lọc tăng số lượng mẫu, giảm số lượng mẫu.



Hình 2.20. Hình ảnh dữ liệu trước và sau lọc nhiễu sử dụng module Filters

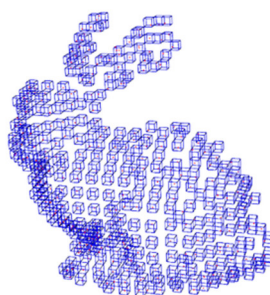
- **Module Geometry:** Chứa tất cả các cấu trúc dữ liệu và giải thuật để tính toán hình học.

- **Module IO:** Chứa các hàm và các lớp để đọc và ghi dữ liệu dạng PCD, có thể thu thập dữ liệu từ nhiều nguồn khác nhau.
- **Module Kdtree:** Module cung cấp cấu trúc dữ liệu Kd-tree, sử dụng FLANN giúp nhanh chóng tìm kiếm vùng gần nhất (nearest neighbors searches). Kd-tree là một cấu trúc dữ liệu dùng để phân vùng không gian lưu trữ tập K-dimension điểm dưới dạng cây do đó dễ dàng phân loại và tìm kiếm.



Hình 2.21. Dữ liệu đã được phân vùng sử dụng module Kdtree

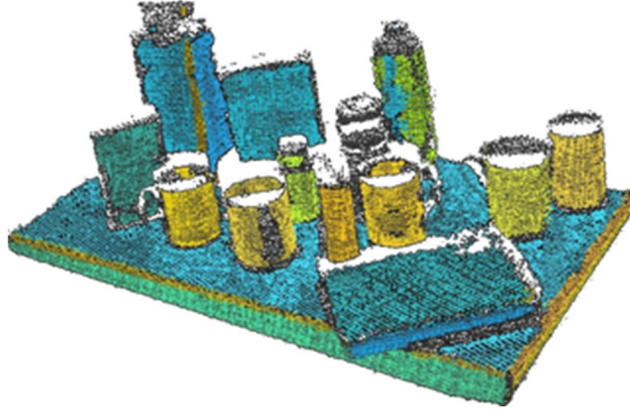
- **Module Keypoint:** Module chứa thực thi của hai thuật toán nhận dạng “PointCloud keypoint”. Keypoint (hay còn gọi là interest point) là các điểm trong ảnh hoặc trong PointCloud mà có tính chất ổn định, riêng biệt và có thể dễ dàng phát hiện ra.
- **Module Octree:** Module chứa các thuật toán hiệu quả để tạo nên một cấu trúc dữ liệu phân cấp từ dữ liệu PointCloud như phân vùng không gian, giảm số lượng mẫu, thực hiện các phép toán tìm kiếm trong tập dữ liệu PointCloud, đồng thời cung cấp các chương trình tìm kiếm lân cận hiệu quả.



Hình 2.22. Dữ liệu đã được phân vùng sử dụng module Octree

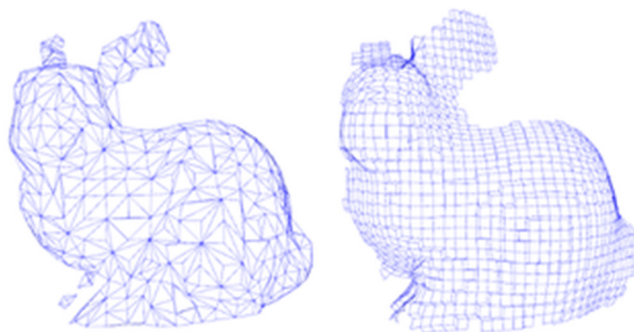
- **Module Registration:** Module cung cấp các hàm kết hợp các bộ dữ liệu vào một mô hình chung, thống nhất.

- **Module SampleConsensus:** Module cung cấp các khả năng tách các nhóm điểm có cùng tính chất dựa trên các thuật toán như RANSAC. Các nhóm điểm có thể là mặt phẳng, mặt cầu, mặt trụ.



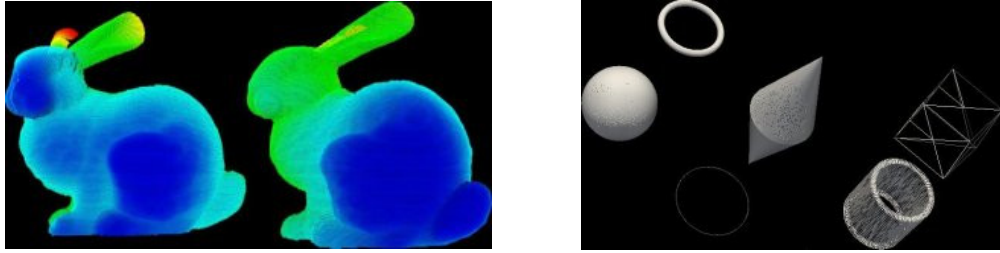
Hình 2.23. Dữ liệu đã được tách nhóm thành các hình dạng cụ thể sử dụng module SampleConsensus

- **Module Search:** Module cung cấp các phương pháp tìm kiếm lân cận bằng các sử dụng các cấu trúc dữ liệu khác nhau như Kd-tree, Octree, Brute force,...
- **Module Segmentation:** Module chứa các thuật toán để phân chia PointCloud thành các nhóm riêng biệt. Thích hợp khi xử lý các PointCloud trong các vùng không gian cô lập.
- **Module Surface:** Module cung cấp giải thuật cho việc xây dựng lại các bề mặt từ dữ liệu PointCloud.



Hình 2.24. Bề mặt tái tạo bằng phương pháp đa giác sử dụng module Surface

- **Module Visualization:** Module dùng để hiển thị nhanh chóng các kết quả thuật toán trên dữ liệu 3D.



Hình 2.25. Hình ảnh được hiển thị sử dụng module Visualization

PCL được viết dựa trên các thư viện mã nguồn mở hỗ trợ đi kèm được chia nhỏ và có thể biên dịch độc lập. Các thư viện này gồm có:

- Eigen: Hỗ trợ các phép tính toán tuyến tính, dùng vào hầu hết các tính toán toán học của PCL.
- FLANN (Fast Library for Approximate Nearest Neighbors): Tìm kiếm nhanh các điểm lân cận trong không gian 3D.
- Boost: Giúp chia sẻ con trỏ trên tất cả các module và thuật toán trong PCL để tránh sao chép và trùng dữ liệu đã lấy về trong hệ thống.
- VTK (Visualization Toolkit): Hỗ trợ nhiều platform trong việc thu về dữ liệu 3D, hỗ trợ hiển thị, ước lượng thể tích vật thể.
- CminPack: Thư viện mở giúp giải quyết phép toán tuyến tính và phi tuyến.

2.2.3. Thư viện xử lý ảnh 3D Nestk - RGBDemo

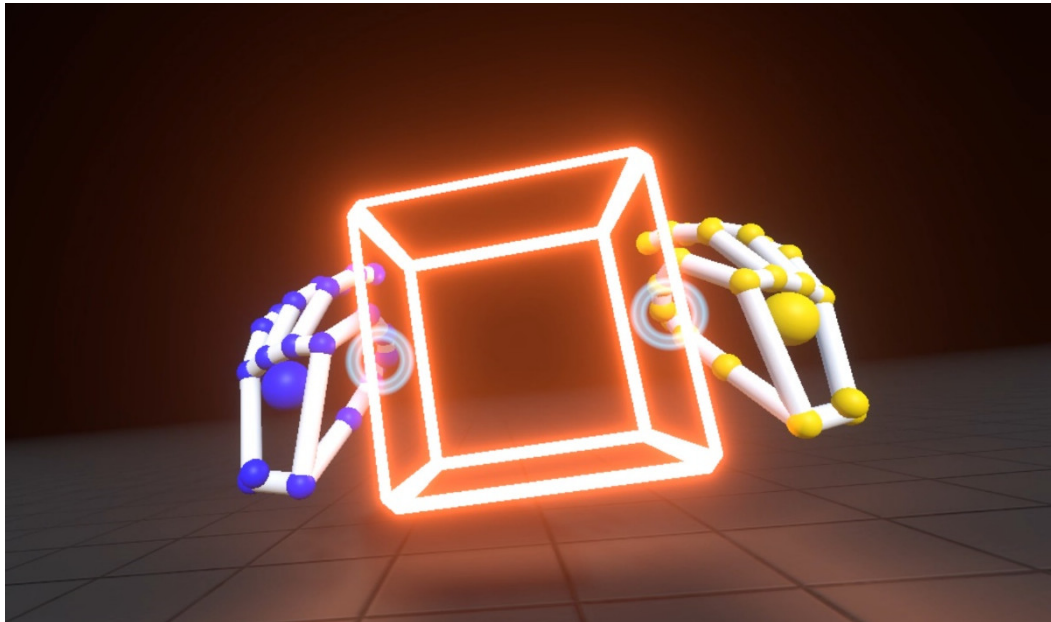
RGBDemo là phần mềm mã nguồn mở được phát triển nhằm mục đích cung cấp một bộ công cụ đơn giản để bắt đầu tiếp cận với dữ liệu Kinect và phát triển các chương trình ứng dụng Thị giác máy tính độc lập mà không gặp rắc rối trong việc tích hợp các thư viện hiện có.

Nestk là thư viện cốt lõi được phát triển trong dự án RGBDemo. Thư viện Nestk được thiết kế để dễ dàng tích hợp vào phần mềm dựa trên nền tảng phần mềm Cmake và cung cấp khả năng truy cập các tính năng Kinect một cách đơn giản, nhanh chóng.

Thư viện Nestk được xây dựng dựa trên ưu điểm về mảng đồ họa của thư viện OpenCV và Qt, đồng thời mảng thuật toán và xử lý của thư viện PCL.

2.2.4. Thư viện Orion cho cảm biến Leap Motion

Orion là bộ thư viện dành cho lập trình viên hệ thứ ba được công ty Leap Motion phát hành. Bộ thư viện hỗ trợ nhiều nền tảng thông dụng hiện nay như C, C#, C++, Unity, Python, Java, Objective-C đồng thời được phát hành dưới dạng mã nguồn mở, cho phép người dùng tùy biến phù hợp với mục đích và yêu cầu sử dụng.



Hình 2.26. Hình ảnh từ Leap SDK sử dụng thư viện Orion

Thư viện Orion định nghĩa từng lớp (class) đại diện cho từng đối tượng (object) cơ bản được theo dõi, bao gồm:

- Frame: Đối tượng Frame là gốc của mô hình dữ liệu và cung cấp quyền truy cập vào tất cả các đối tượng khác được theo dõi.
- Hand: Đối tượng Hand mô tả vị trí và hướng của bàn tay; theo dõi cử động của bàn tay giữa các khung hình và chứa danh sách các ngón tay được liên kết với bàn tay đó.

- Arm: Đối tượng Arm mô tả vị trí, trục và hướng của cánh tay của một đối tượng Hand. Đối tượng chỉ có thể được truy cập thông qua đối tượng Hand.
- Pointable, Finger: Đối tượng Pointable mô tả các đặc tính thông dụng của ngón tay. Đối tượng Pointable bổ sung thông tin cụ thể cho các đối tượng Finger đã liên kết với Hand.
- Bone: Đối tượng Bone mô tả vị trí và hướng của xương trên tự ngón tay.

2.2.5. MATLAB Simulink và SimMechanics

2.2.5.1. MATLAB Simulink



Hình 2.27. Logo MATLAB

MATLAB (MATrix LABoratory) là một phần mềm khoa học được thiết kế để cung cấp việc tính toán số và hiển thị đồ họa bằng ngôn ngữ lập trình cấp cao. MATLAB cung cấp các tính năng tương tác tuyệt vời cho phép người sử dụng thao tác dữ liệu linh hoạt dưới dạng mảng ma trận để tính toán và quan sát. Các dữ liệu vào của MATLAB có thể được nhập từ "Command line" hoặc từ "mfiles", trong đó tập lệnh được cho trước bởi MATLAB.

MATLAB cung cấp cho người dùng các toolbox tiêu chuẩn tùy chọn. Người dùng cũng có thể tạo ra các hộp công cụ riêng của mình gồm các "mfiles" được viết cho các ứng dụng cụ thể. Chúng ta có thể sử dụng các tập tin trợ giúp của MATLAB cho các chức năng và các lệnh liên quan với các toolbox có sẵn.

Simulink là một công cụ trong Matlab dùng để mô hình, mô phỏng và phân tích các hệ thống động với môi trường giao diện sử dụng bằng đồ họa. Việc xây dựng mô hình được đơn giản hóa bằng các hoạt động nhấp chuột và kéo thả. Simulink bao gồm một bộ thư viện khối với các hộp công cụ toàn diện cho cả việc

phân tích tuyến tính và phi tuyến. Simulink là một phần quan trọng của MATLAB và có thể dễ dàng chuyển đổi qua lại trong quá trình phân tích, và vì vậy người dùng có thể tận dụng được ưu thế của cả hai môi trường.

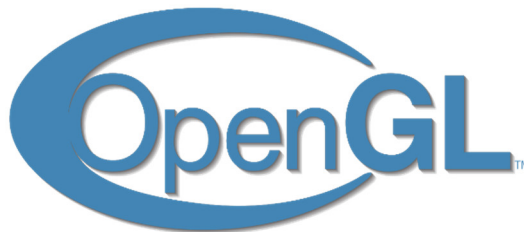
2.2.5.2. SimMechanics

SimMechanics là một công cụ của MATLAB cho phép người dùng mô hình hóa được các chi tiết cơ khí, từ đó xây dựng được mô hình các bộ phận máy, các máy cơ khí.

SimMechanics gồm có các thư viện sau:

- Thư viện Bodies: Dùng để biểu diễn một vật thể cứng có các thuộc tính khối lượng, mô-men quán tính, tọa độ trọng tâm, hệ tọa độ và biểu diễn các điểm trong hệ tọa độ tuyệt đối World.
- Thư viện Constraints & Driver: Dùng để định rõ góc giữa hai vector trục, khoảng cách giữa hai gốc tọa độ trục, thành phần vector nối hai trục, vận tốc, quỹ đạo chuyển động giữa hai Body, định dạng quỹ đạo song song, quỹ đạo tròn của Body.
- Thư viện Joints: Dùng để biểu diễn sự liên kết của một khớp với các bậc tự do tịnh tiến hoặc xoay, lực tác động, vị trí và vận tốc ban đầu, ma sát phải chịu của khớp.
- Thư viện Utilities: Dùng để kết nối hệ thống mô phỏng với mô hình vật lý.

2.2.6. Thư viện đồ họa OpenGL



Hình 2.28. Logo OpenGL

OpenGL (Open Graphics Library) là một tiêu chuẩn kỹ thuật đồ họa có mục đích định ra một giao diện lập trình ứng dụng (Application Programming Interface - API) đồ họa 3 chiều. Giao diện lập trình này chứa khoảng 250 hàm để vẽ các cảnh phức tạp từ những hàm đơn giản. Thư viện được dùng rộng rãi trong các ứng dụng CAD, thực tế ảo, mô phỏng khoa học, mô phỏng thông tin, trò chơi điện tử...

OpenGL là một thư viện đồ họa độc lập và tương thích với mọi nền tảng. Thư viện giúp đơn giản hóa việc tương tác giữa các mô hình 3D bằng một giao diện lập trình thống nhất, ép buộc các phần cứng 3D khác nhau phải tương thích để hỗ trợ hoàn toàn cho thư viện và có thể sử dụng thêm sức mạnh phần mềm để xử lý.

2.2.7. Thư viện Asset Import

Asset Import là thư viện mã nguồn mở hỗ trợ giải mã và nhập mô hình 3D đa nền tảng với mục đích cung cấp một giao diện lập trình ứng dụng chung cho các định dạng tệp 3D khác nhau. Thư viện được phát triển nhằm mục đích tạo ra một phương thức chuyển đổi dữ liệu 3D đầy đủ để sử dụng trong các Game engine cũng như các ứng dụng thực tế ảo.

Thư viện Asset Import có các đặc điểm nổi bật như:

- Viết bằng C++, giao diện lõi/API được cung cấp cho cả C và C++.
- Cấu hình và tùy biến dễ dàng thông qua Cmake.
- Giao diện Command-line để thực hiện các tác vụ phổ biến.
- Nhập nhiều kênh màu UV và vertex
- Hỗ trợ vật liệu nhiều lớp phức tạp.
- Không phụ thuộc thư viện bên thứ ba.

Nhờ có Asset Import, các mô hình 3D được vẽ bằng các phần mềm chuyên dụng như Solidworks, AutoCad có thể được nhập trực tiếp vào chương trình đồ họa, giúp tiết kiệm chi phí lập trình.

Chương 3. ĐỘNG HỌC ROBOT RV-2AJ

3.1. Động học vị trí

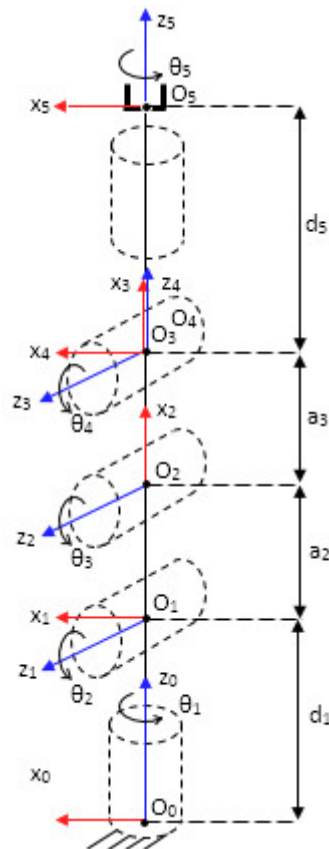
3.1.1. Động học thuận

Động học thuận là bài toán từ vị trí các khớp đã biết, xác định vị trí và hướng điểm cuối của robot trong không gian Descartes. Công thức tổng quát của bài toán là

$$\mathbf{q} \rightarrow \mathbf{R}_e^0, \mathbf{p}_e^0 \quad (3.1)$$

trong đó $\mathbf{q} = [q_1 \ q_2 \ \dots \ q_n]^T$ là vector vị trí các khớp; $\mathbf{p}_e^0$ là vector vị trí đầu công tác; $\mathbf{R}_e^0$ là ma trận gồm ba vector đơn vị $[\mathbf{u}_e^0 \ \mathbf{v}_e^0 \ \mathbf{w}_e^0]$ biểu diễn hướng của điểm cuối so với hệ tọa độ gốc.

Quy tắc Denavit-Hartenberg (DH) thường được dùng để chọn hệ tọa độ cho các khớp robot. Đối với robot RV-2AJ, hệ tọa độ được chọn như sau.



Hình 3.1 Hệ tọa độ của robot RV-2AJ

Khâu i	a_i (m)	α_i (deg)	d_i (m)	θ_i (deg)
1	0	-90	d_1	θ_1
2	a_2	0	0	$-90 + \theta_2$
3	a_3	0	0	θ_3
4	0	90	0	$90 + \theta_4$
5	0	0	d_5	θ_5

Bảng 3.1. Thông số DH của robot RV-2AJ

Trong đó d_1, a_2, a_3, d_5 là các độ dài cố định; $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ là các biến khớp xoay của robot.

Ma trận biến đổi thuần nhất giữa các hệ toạ độ được tính bởi

$$\begin{aligned}
 \mathbf{T}_i^{i-1} &= Rot_{z,\theta} Trans_{z,d} Trans_{x,\alpha} Rot_{x,\alpha} \\
 &= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & -s_{\theta_i}s_{\alpha_i} & 0 \\ s_{\theta_i} & -c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned} \tag{3.2}$$

Khai triển trên từng khớp robot, ta có

$$\mathbf{T}_1^0 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$\mathbf{T}_2^1 = \begin{bmatrix} s_2 & c_2 & 0 & a_2s_2 \\ -c_2 & s_2 & 0 & -a_2c_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$\mathbf{T}_3^2 = \begin{bmatrix} c_3 & -s_3 & 0 & a_3c_3 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

$$\mathbf{T}_4^3 = \begin{bmatrix} -s_4 & 0 & c_4 & 0 \\ c_4 & 0 & s_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.6)$$

$$\mathbf{T}_5^4 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.7)$$

Suy ra ma trận thuần nhất của hệ tọa độ điểm cuối so với hệ tọa độ gốc là

$$\begin{aligned} \mathbf{T}_5^0 &= \mathbf{T}_1^0 \mathbf{T}_2^1 \mathbf{T}_3^2 \mathbf{T}_4^3 \mathbf{T}_5^4 \\ &= \begin{bmatrix} -s_1 s_5 + c_1 c_5 c_{234} & -s_1 c_5 - c_1 s_5 c_{234} & c_1 s_{234} & c_1(a_2 s_2 + a_3 s_{23} + d_5 s_{234}) \\ c_1 s_5 + s_1 c_5 c_{234} & c_1 c_5 - s_1 s_5 c_{234} & s_1 s_{234} & s_1(a_2 s_2 + a_3 s_{23} + d_5 s_{234}) \\ -c_5 s_{234} & s_5 s_{234} & c_{234} & d_1 + a_2 s_2 + a_3 c_{23} + d_5 c_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (3.8)$$

Kết quả động học thuận của robot

$$\mathbf{R}_e^0 = \begin{bmatrix} -s_1 s_5 + c_1 c_{234} c_5 & -s_1 c_5 - c_1 c_{234} s_5 & c_1 s_{234} \\ c_1 s_5 + s_1 c_{234} c_5 & c_1 c_5 - s_1 c_{234} s_5 & s_1 s_{234} \\ -s_{234} c_5 & s_{234} s_5 & c_{234} \end{bmatrix} \quad (3.9)$$

$$\mathbf{p}_e^0 = \begin{bmatrix} c_1(a_2 s_2 + a_3 s_{23} + d_5 s_{234}) \\ s_1(a_2 s_2 + a_3 s_{23} + d_5 s_{234}) \\ d_1 + a_2 s_2 + a_3 c_{23} + d_5 c_{234} \end{bmatrix} \quad (3.10)$$

3.1.2. Động học ngược

Động học ngược là bài toán cho biết trước vị trí và hướng điểm cuối, yêu cầu xác định vị trí các góc khớp của robot. Đây là bài toán thiết yếu để điều khiển robot đi đến điểm mong muốn. Công thức tổng quát của bài toán là

$$\mathbf{R}_e^0, \mathbf{p}_e^0 \rightarrow \mathbf{q} \quad (3.11)$$

Trước hết, nhận xét cấu trúc của robot đang khảo sát, ta thấy ba khớp liên tiếp là 2, 3 và 4 có trục xoay cùng hướng với nhau. Do đó, ta có thể tổng hợp các

góc $\theta_2, \theta_3, \theta_4$ thành một góc duy nhất là $(\theta_2 + \theta_3 + \theta_4)$. Khi đó, không khó để nhận ra phép xoay đến điểm cuối của robot chính là phép xoay Euler theo thứ tự ZYZ với trục hiện tại. Bằng chứng là sự đồng dạng giữa ma trận xoay $\mathbf{R}_e^0$ và ma trận xoay Euler $\mathbf{R}_{ZYZ}$.

$$\begin{aligned} \mathbf{R}_{ZYZ} &= \mathbf{R}_{z,\phi} \mathbf{R}_{y,\theta} \mathbf{R}_{z,\psi} \\ &= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} -s_\phi s_\psi + c_\phi c_\theta c_\psi & -s_\phi c_\psi - c_\phi c_\theta s_\psi & c_\phi s_\theta \\ c_\phi s_\psi + s_\phi c_\theta c_\psi & c_\phi c_\psi - s_\phi c_\theta s_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \end{aligned} \quad (3.12)$$

$$\text{Suy ra} \quad \phi = \theta_1, \quad \theta = \theta_2 + \theta_3 + \theta_4, \quad \psi = \theta_5 \quad (3.13)$$

Tiếp theo, ta thấy số bậc tự do của robot là 5, thiếu một bậc để robot có khả năng đi đến mọi vị trí với mọi hướng trong không gian làm việc. Để có vị trí chính xác, ta chọn việc không thoả mãn một trong ba góc xoay Euler, tức là tính ra góc đó mà không sử dụng ma trận xoay $\mathbf{R}_e^0$. Xét vector vị trí $\mathbf{p}_e^0$

$$\mathbf{p}_e^0 = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} c_1(a_2 s_2 + a_3 s_{23} + d_5 s_{234}) \\ s_1(a_2 s_2 + a_3 s_{23} + d_5 s_{234}) \\ d_1 + a_2 s_2 + a_3 c_{23} + d_5 c_{234} \end{bmatrix} \quad (3.14)$$

Dễ dàng tìm được ϕ , có 2 nghiệm ứng với cấu hình tay phải và tay trái

$$\phi = \theta_1 = \begin{cases} \text{atan2}(p_y, p_x), & \text{right arm} \\ \text{atan2}(-p_y, -p_x), & \text{left arm} \end{cases} \quad (3.15)$$

Nếu vị điểm cuối rơi vào điểm kì dị $p_x = p_y = 0$, ϕ có vô số nghiệm.

Hai góc Euler còn lại được tìm dựa trên ϕ và $\mathbf{R}_e^0$

$$\mathbf{R}_{y,\theta} \mathbf{R}_{z,\psi} = (\mathbf{R}_{z,\phi})^{-1} \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \quad (3.16)$$

$$\begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & s_\theta \\ s_\psi & c_\psi & 0 \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} = \begin{bmatrix} c_\phi & s_\phi & 0 \\ -s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \quad (3.17)$$

Suy ra

$$\theta = \text{atan2}(c_\phi w_x + s_\phi w_y, w_z) \quad (3.18)$$

$$\psi = \text{atan2}(-s_\phi u_x + c_\phi u_y, c_\theta(c_\phi u_x + s_\phi u_y) - s_\theta u_z) \quad (3.19)$$

Từ đây, ta còn việc tìm cụ thể $\theta_2, \theta_3, \theta_4$ để thoả mãn vị trí điểm cuối. Điều này có thể thực hiện bằng phương pháp biến đổi đại số. Gọi C là vị trí tâm cổ tay robot

$$C_x = c_1(a_2 c_2 + a_3 c_{23}) = p_x - c_1 s_{234} d_5 \quad (3.20)$$

$$C_y = s_1(a_2 c_2 + a_3 c_{23}) = p_y - s_1 s_{234} d_5 \quad (3.21)$$

$$C_z = d_1 + a_2 s_2 + a_3 s_{23} = p_z - d_5 c_{234} \quad (3.22)$$

Biến đổi đại số tìm được hai nghiệm θ_3 ứng với cấu hình khuỷu tay trên và khuỷu tay dưới

$$C_x^2 + C_y^2 + (d_1 - C_z)^2 = a_2^2 + a_3^2 + 2a_2 a_3 c_3 \quad (3.23)$$

$$c_3 = \frac{C_x^2 + C_y^2 + (d_1 - C_z)^2 - a_2^2 - a_3^2}{2a_2 a_3} \quad (3.24)$$

$$\theta_3 = \begin{cases} \text{atan2}(c_3, \sqrt{1 - c_3^2}), & \text{above elbow} \\ \text{atan2}(c_3, -\sqrt{1 - c_3^2}), & \text{below elbow} \end{cases} \quad (3.25)$$

Đặt hệ phương trình tìm được θ_2

$$\begin{cases} a_2 c_2 + a_3 c_{23} = c_1 C_x + s_1 C_y \\ a_2 s_2 + a_3 s_{23} = C_z - d_1 \end{cases} \quad (3.26)$$

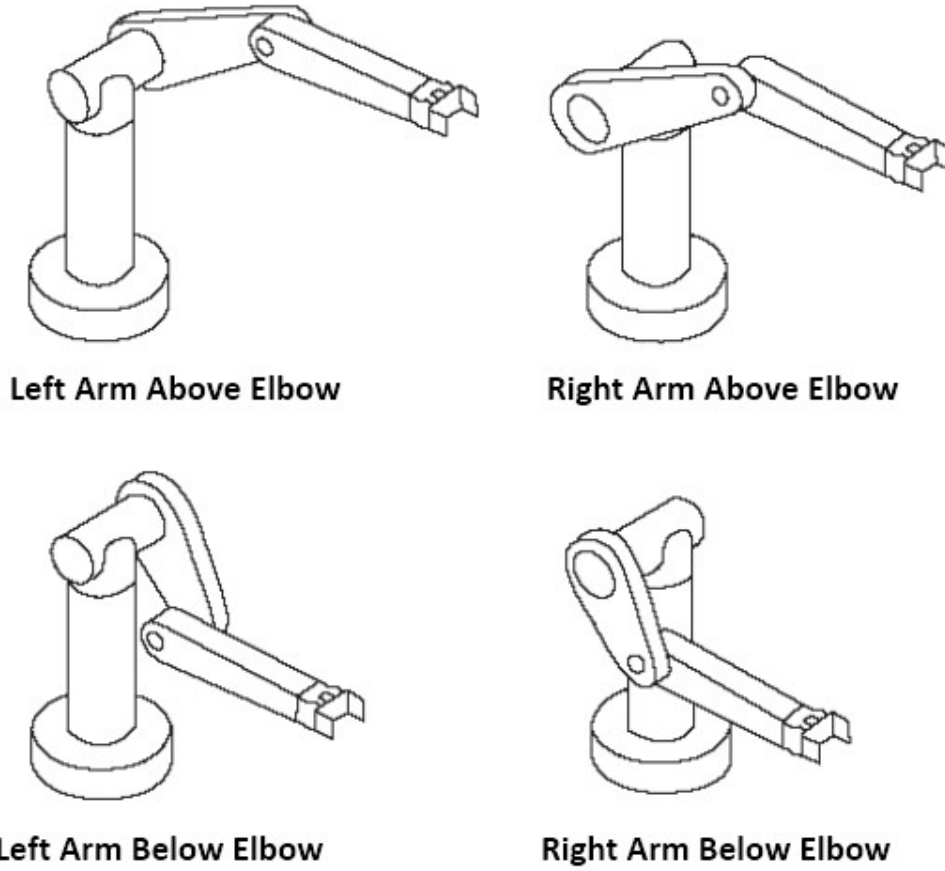
$$c_2 = \frac{(c_1 C_x + s_1 C_y) a_3 s_3 + (C_z - d_1)(a_2 + a_3 c_3)}{(a_2 + a_3 c_3)^2 + (a_3 s_3)^2} \quad (3.27)$$

$$s_2 = \frac{(c_1 C_x + s_1 C_y)(a_2 + a_3 c_3) - (C_z - d_1)a_3 s_3}{(a_2 + a_3 c_3)^2 + (a_3 s_3)^2} \quad (3.28)$$

$$\theta_2 = \text{atan2}(c_2, s_2) \quad (3.29)$$

Cuối cùng

$$\theta_4 = \theta - (\theta_2 + \theta_3) \quad (3.30)$$



Hình 3.2. Các cấu hình khả thi của robot tại một vị trí

3.2. Động học vận tốc

3.2.1. Ma trận Jacobi

Ma trận Jacobi biểu thị mối quan hệ giữa vận tốc dài, vận tốc góc điểm cuối với vận tốc các khớp của robot như sau

$$\begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix} = \mathbf{J}\dot{\mathbf{q}} \quad (3.31)$$

trong đó $\mathbf{J}$ là ma trận Jacobi kích thước $6 \times n$, với n là số bậc tự do của robot.

Mỗi cột của của ma trận được xác định bằng công thức

- Nếu khớp i là khớp xoay

$$\mathbf{J}_i = \begin{bmatrix} \mathbf{z}_{i-1}^0 \times (\mathbf{p}_n^0 - \mathbf{p}_{i-1}^0) \\ \mathbf{z}_{i-1}^0 \end{bmatrix} \quad (3.32)$$

- Nếu khớp i là khớp tịnh tiến

$$\mathbf{J}_i = \begin{bmatrix} \mathbf{z}_{i-1}^0 \\ 0 \end{bmatrix} \quad (3.33)$$

Áp dụng vào robot RV-2AJ với 5 khớp xoay

$$\begin{aligned} \mathbf{J} &= \begin{bmatrix} \mathbf{z}_0^0 \times (\mathbf{p}_5^0 - \mathbf{p}_0^0) & \mathbf{z}_1^0 \times (\mathbf{p}_5^0 - \mathbf{p}_1^0) & \mathbf{z}_2^0 \times (\mathbf{p}_5^0 - \mathbf{p}_2^0) & \mathbf{z}_3^0 \times (\mathbf{p}_5^0 - \mathbf{p}_3^0) & \mathbf{z}_4^0 \times (\mathbf{p}_5^0 - \mathbf{p}_4^0) \\ \mathbf{z}_0^0 & \mathbf{z}_1^0 & \mathbf{z}_2^0 & \mathbf{z}_3^0 & \mathbf{z}_4^0 \end{bmatrix} \\ &= \begin{bmatrix} -s_1(a_2s_2 + a_3s_{23} + d_5s_{234}) & c_1(a_2c_2 + a_3c_{23} + c_{234}d_5) & & & \\ c_1(a_2s_2 + a_3s_{23} + d_5s_{234}) & s_1(a_2c_2 + a_3c_{23} + c_{234}d_5) & & & \\ 0 & -a_2s_2 - a_3s_{23} - d_5s_{234} & & & \\ 0 & -s_1 & & & \\ 0 & c_1 & & & \\ 1 & 0 & & & \end{bmatrix} \\ &\quad \begin{bmatrix} c_1(a_3c_{23} + c_{234}d_5) & c_1c_{234}d_5 & 0 \\ s_1(a_3c_{23} + c_{234}d_5) & s_1c_{234}d_5 & 0 \\ -a_3s_{23} - s_{234}d_5 & s_{234}d_5 & 0 \\ -s_1 & -s_1 & c_1s_{234} \\ c_1 & c_1 & s_1s_{234} \\ 0 & 0 & c_{234} \end{bmatrix} \end{aligned} \quad (3.34)$$

3.2.2. Tìm động học vị trí ngược bằng ma trận Jacobi giả nghịch đảo

Trong bài toán động học vị trí ngược, xảy ra hai trường hợp đối với tọa độ điểm cuối: nằm trong hoặc nằm ngoài vùng làm việc khả thi của robot. Tuy nhiên, phương pháp giải bằng biến đổi đại số như trên chỉ cho nghiệm trong trường hợp thứ nhất. Để giải quyết cả hai trường hợp, có thể sử dụng phương pháp ma trận Jacobi giả nghịch đảo. Ta chọn

$$\Delta \mathbf{q} = \mathbf{J}^+ \mathbf{e} \quad (3.35)$$

với $\Delta \mathbf{q}$ là sai phân của vector biến khớp, $\mathbf{e} = \mathbf{x}_d - \mathbf{x}$ là sai số giữa vector tọa độ điểm cuối cần tính và điểm cuối hiện tại, $\mathbf{J}^+$ là ma trận Jacobi giả nghịch đảo. Lời giải động học ngược vị trí được tìm bằng thuật toán sau.

Bước 1 : Chọn $\boldsymbol{\eta}$. Khởi tạo $\mathbf{q}_0$ và $\mathbf{x}_0$ tương ứng.

Bước 2 : Tính sai số tọa độ điểm cuối so với giá trị đặt

$$\mathbf{e}_k = \mathbf{x}_d - \mathbf{x}_k \quad (3.36)$$

Bước 3 : Nếu $\mathbf{e}_k$ nhỏ hơn sai số tối thiểu, hoặc k lớn hơn số vòng lặp tối đa thì kết thúc.

Bước 4 : Tính sai phân của vector biến khớp

$$\Delta \mathbf{q}_k = \mathbf{J}(\mathbf{q}_k)^+ \mathbf{e}_k \quad (3.37)$$

Bước 5 : Cập nhật vị trí khớp và tọa độ điểm cuối

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \boldsymbol{\eta} \Delta \mathbf{q}_k \quad (3.38)$$

$$\mathbf{x}_{k+1} = f(\mathbf{q}_k) \quad (3.39)$$

Bước 6 : Quay lại bước 2.

Trong quá trình thực hiện thuật toán, sai số $\mathbf{e}_k$ giảm dần và $\mathbf{q}_k \rightarrow \mathbf{q}_d$. Nếu điểm đặt ở trong vùng làm việc khả thi, robot đi đến điểm đặt với sai số gần như bằng 0. Trong trường hợp ngược lại, robot vẫn tiến đến điểm gần nhất có thể so với điểm đặt. Đây là ưu điểm của phương pháp ma trận Jacobi.

Bên cạnh đó, với $\Delta \mathbf{q}$ được tính bằng ma trận Jacobi giả nghịch đảo, hàm sai số xấp xỉ động học $E = \|\mathbf{J}\Delta \mathbf{q} - \mathbf{e}\|$ đạt giá trị cực tiểu. Nhờ đó thuật toán tìm được kết quả chính xác nhất.

Áp dụng phương pháp trên vào robot, bắt đầu từ đoạn xác định $\theta_2, \theta_3, \theta_4$ để thoả mãn vector vị trí điểm cuối p_e^0 sau khi đã biết 3 góc Euler (ϕ, θ, ψ) . Vì $\theta = \theta_2 + \theta_3 + \theta_4$ nên thực chất chỉ còn hai ẩn cần tìm. Ma trận Jacobi có thể được viết gọn chỉ gồm thành phần vận tốc dài ứng với khớp 2 và 3.

$$\mathbf{J} = [\mathbf{J}_{v_2} \quad \mathbf{J}_{v_3}] = \begin{bmatrix} c_1(a_2c_2 + a_3c_{23} + c_{234}d_5) & c_1(a_3c_{23} + c_{234}d_5) \\ s_1(a_2c_2 + a_3c_{23} + c_{234}d_5) & s_1(a_3c_{23} + c_{234}d_5) \\ -a_2s_2 - a_3s_{23} - d_5s_{234} & -a_3s_{23} - s_{234}d_5 \end{bmatrix} \quad (3.40)$$

Khai triển ma trận Jacobi giả nghịch đảo, tìm được sai phân các biến khớp

$$\begin{bmatrix} \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix} = [(\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T] \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} \quad (3.41)$$

Áp dụng thuật toán lặp, ta tìm được θ_2, θ_3 và θ_4 .

3.3. Mô phỏng

3.3.1. Xây dựng mô phỏng với MATLAB m-file và Simmechanics

MATLAB là phần mềm tính toán, mô phỏng mạnh mẽ và phổ biến. Trong phần này, tác giả sử dụng MATLAB để mô phỏng và kiểm chứng mô hình toán của robot. Đầu tiên, các công thức động học thuận và ngược được viết vào các chương trình m-file, gồm có

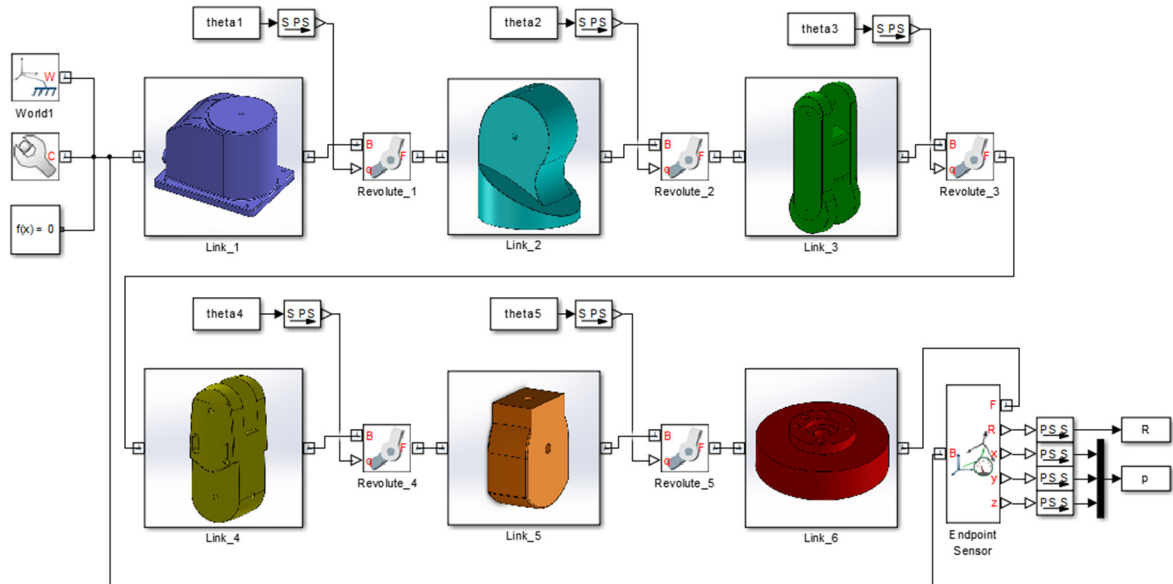
- *mitsu_forward_kinematic.m*
- *mitsu_inverse_kinematic.m*
- *mitsu_inverse_kinematic_Jacobian.m*

Các chương trình trên sẽ được kiểm tra với một số đầu vào cụ thể. Yêu cầu đặt ra là cả hai chương trình động học ngược phải có đầu ra bằng nhau, và bằng với đầu vào của chương trình động học thuận.

Tiếp theo, robot được mô phỏng sử dụng Simulink/Simmechanics. Đây là công cụ chuyên mô phỏng các hệ cơ khí dưới dạng các khâu và khớp. Chuyển động của cơ hệ được tự động tính toán theo các thông số độ dài, khối lượng, quán tính, hệ toạ độ... dưới tác động của đầu vào là vị trí hoặc lực trên các khớp. Simmechanics còn có đồ hoạ 3D hiển thị được các file bản vẽ CAD.

Thực hiện mô phỏng robot bằng Simmechanics nhằm hai mục đích: so sánh mô hình toán động học robot trong m-file với một công cụ tính toán độc lập, đồng thời thể hiện trực quan hình dáng và tư thế của robot trong không gian ba chiều.

Sơ đồ khối mô phỏng robot gồm 5 khớp và 6 khâu. Đầu vào là giá trị các góc khớp $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ [deg]. Đầu ra là vị trí và hướng của điểm cuối (p_e^0 [mm], R_e^0). Thông số độ dài các khâu của robot được chọn theo giá trị thực tế của là $d_1 = 300$ mm, $a_2 = 250$ mm, $a_3 = 160$ mm, $d_5 = 72$ mm.



Hình 3.3. Sơ đồ khối mô phỏng robot RV-2AJ

3.3.2. Kết quả mô phỏng

3.3.2.1. Kiểm tra tương quan động học thuận và động học ngược

Các chương trình m-file tính động học thuận và ngược được kiểm nghiệm bằng các bộ số như nhau, kết quả thu được là

Chương trình: <i>mitsu_forward_kinematic.m</i>		
Lần	Đầu vào $\mathbf{q} = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]^T$	Đầu ra $\mathbf{R}_e^0 = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix}, \mathbf{p}_e^0 = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$
1	$[0.00, 0.00, 90.00, 0.00, 0.00]^T$	$\begin{bmatrix} 0.00 & 0.00 & 1.00 \\ 0.00 & 1.00 & 0.00 \\ -1.00 & 0.00 & 0.00 \end{bmatrix}, \begin{bmatrix} 232.00 \\ 0.00 \\ 550.00 \end{bmatrix}$ right-above
2	$[30.00, 45.00, 60.00, -30.00, 60.00]^T$	$\begin{bmatrix} -0.321 & -0.444 & 0.837 \\ 0.815 & 0.321 & 0.483 \\ -0.483 & 0.837 & 0.259 \end{bmatrix}, \begin{bmatrix} 347.16 \\ 200.44 \\ 454.00 \end{bmatrix}$ right-above
3	$[30.00, 90.00, -60.00, 30.00, 60.00]^T$	$\begin{bmatrix} -0.217 & -0.625 & 0.750 \\ 0.875 & 0.217 & 0.433 \\ -0.433 & 0.750 & 0.500 \end{bmatrix}, \begin{bmatrix} 339.79 \\ 196.18 \\ 474.56 \end{bmatrix}$ right-below

Bảng 3.2. Kết quả tính toán động học thuận dùng m-file

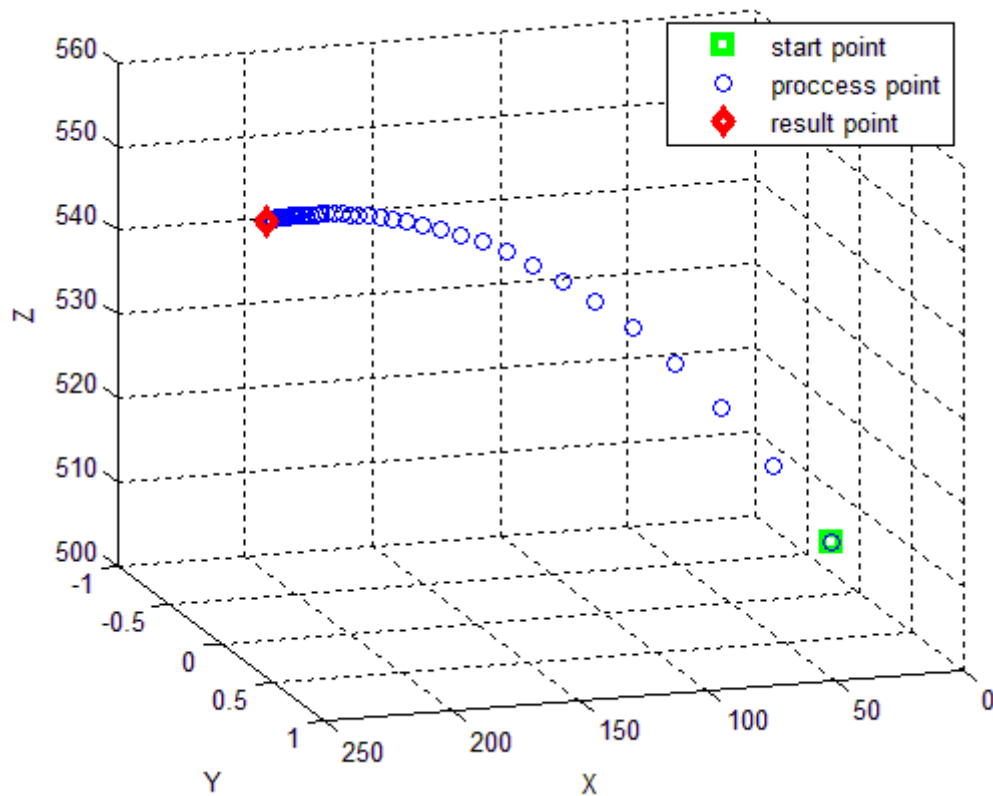
Chương trình: <i>mitsu_inverse_kinematic.m</i>		
Lần	Đầu vào $\mathbf{R}_e^0 = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix}, \mathbf{p}_e^0 = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$	Đầu ra $\mathbf{q} = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]^T$
1	$\begin{bmatrix} 0.00 & 0.00 & 1.00 \\ 0.00 & 1.00 & 0.00 \\ -1.00 & 0.00 & 0.00 \end{bmatrix}, \begin{bmatrix} 232.00 \\ 0.00 \\ 550.00 \end{bmatrix}$ right-above	$[0.00, 0.00, 90.00, 0.00, 0.00]^T$

2	$\begin{bmatrix} -0.321 & -0.444 & 0.837 \\ 0.815 & 0.321 & 0.483 \\ -0.483 & 0.837 & 0.259 \end{bmatrix}, \begin{bmatrix} 347.16 \\ 200.44 \\ 454.00 \end{bmatrix}$ right-above	$[30.00, 45.00, 60.00, -30.00, 60.00]^T$
3	$\begin{bmatrix} -0.217 & -0.625 & 0.750 \\ 0.875 & 0.217 & 0.433 \\ -0.433 & 0.750 & 0.500 \end{bmatrix}, \begin{bmatrix} 339.79 \\ 196.18 \\ 474.56 \end{bmatrix}$ right-below	$[30.00, 90.00, -60.00, 30.00, 60.00]^T$

Bảng 3.3. Kết quả tính toán động học ngược bằng biến đổi đại số dùng m-file

Chương trình: <i>mitsu_inverse_kinematic_Jacobian.m</i>		
Lần	Đầu vào $\mathbf{R}_e^0 = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix}, \mathbf{p}_e^0 = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$	Đầu ra $\mathbf{q} = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]^T$
1	$\begin{bmatrix} 0.00 & 0.00 & 1.00 \\ 0.00 & 1.00 & 0.00 \\ -1.00 & 0.00 & 0.00 \end{bmatrix}, \begin{bmatrix} 232.00 \\ 0.00 \\ 550.00 \end{bmatrix}$ right-above	$[0.00, 0.00, 90.00, 0.00, 0.00]^T$
2	$\begin{bmatrix} -0.321 & -0.444 & 0.837 \\ 0.815 & 0.321 & 0.483 \\ -0.483 & 0.837 & 0.259 \end{bmatrix}, \begin{bmatrix} 347.16 \\ 200.44 \\ 454.00 \end{bmatrix}$ right-above	$[30.00, 45.00, 60.00, -30.00, 60.00]^T$
3	$\begin{bmatrix} -0.217 & -0.625 & 0.750 \\ 0.875 & 0.217 & 0.433 \\ -0.433 & 0.750 & 0.500 \end{bmatrix}, \begin{bmatrix} 339.79 \\ 196.18 \\ 474.56 \end{bmatrix}$ right-below	$[30.00, 90.00, -60.00, 30.00, 60.00]^T$

Bảng 3.4. Kết quả tính động học ngược bằng ma trận Jacobi dùng m-file



Hình 3.4. Quá trình tính động học ngược bằng ma trận Jacobi

Kết quả cho thấy đầu ra của động học ngược khớp với đầu vào của động học thuận, đồng thời hai phương pháp tính động học ngược có đầu ra bằng nhau. Như vậy các chương trình tính động học tương thích với nhau.

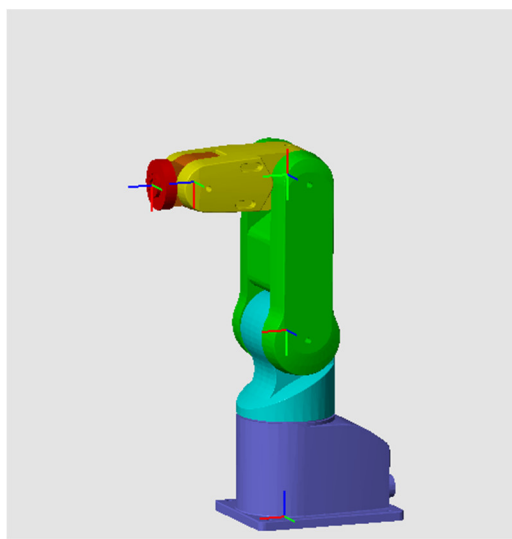
3.3.2.2. So sánh mô hình toán với Simmechanics

Chạy chương trình mô phỏng Simmechanics với đầu vào bằng với đầu vào chương trình m-file tính động học thuận, cho kết quả sau

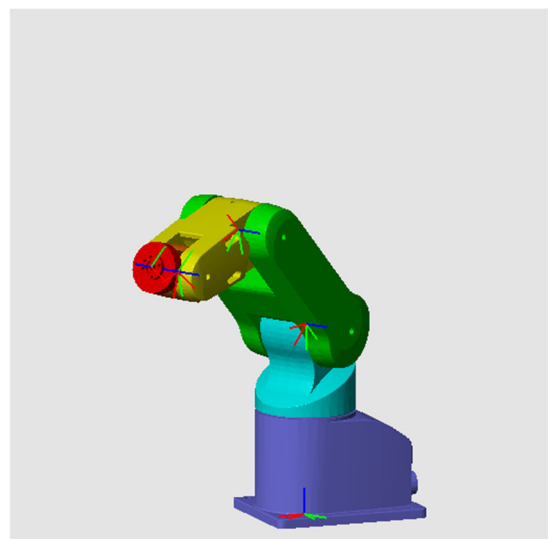
Chương trình: <i>mitsu_kinematic_simmechanics.slx</i>		
Lần	Đầu vào $\mathbf{q} = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]^T$	Đầu ra $\mathbf{R}_e^0 = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix}, \mathbf{p}_e^0 = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$
1	$[0.00, 0.00, 90.00, 0.00, 0.00]^T$	$\begin{bmatrix} 0.00 & 0.00 & 1.00 \\ 0.00 & 1.00 & 0.00 \\ -1.00 & 0.00 & 0.00 \end{bmatrix}, \begin{bmatrix} 232.00 \\ 0.00 \\ 550.00 \end{bmatrix}$

		right-above
2	$[30.00, 45.00, 60.00, -30.00, 60.00]^T$	$\begin{bmatrix} -0.321 & -0.444 & 0.837 \\ 0.815 & 0.321 & 0.483 \\ -0.483 & 0.837 & 0.259 \end{bmatrix}, \begin{bmatrix} 347.16 \\ 200.44 \\ 454.00 \end{bmatrix}$
		right-above
3	$[30.00, 90.00, -60.00, 30.00, 60.00]^T$	$\begin{bmatrix} -0.217 & -0.625 & 0.750 \\ 0.875 & 0.217 & 0.433 \\ -0.433 & 0.750 & 0.500 \end{bmatrix}, \begin{bmatrix} 339.79 \\ 196.18 \\ 474.56 \end{bmatrix}$
		right-below

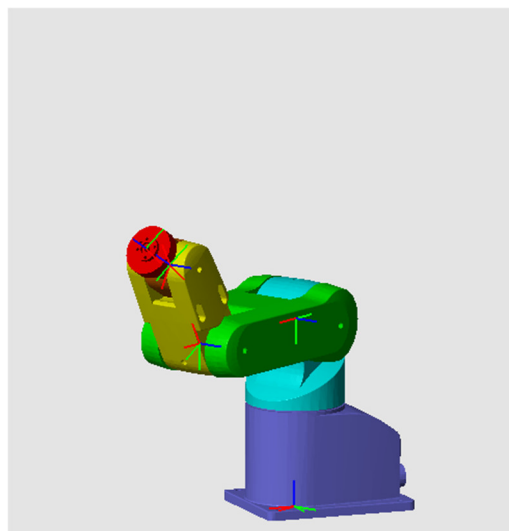
Bảng 3.5. Kết quả tính động học thuận robot dùng mô phỏng Simmechanics



(1)



(2)



(3)

Hình 3.5. Đồ họa Simmechanics mô phỏng robot tại các vị trí tính toán

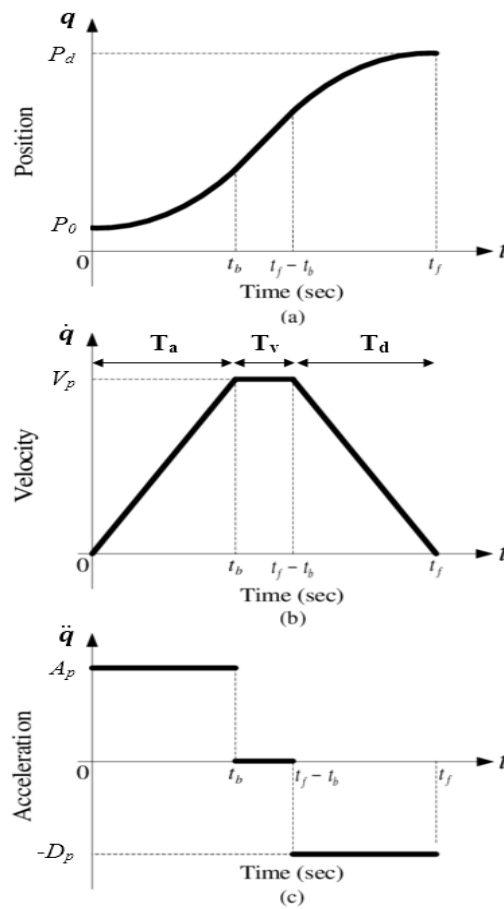
Kết quả mô phỏng động học thuận Simmechanics trùng khớp với kết quả tính theo mô hình toán. Mặt khác, mô hình toán đã được chứng tỏ có sự tương thích giữa động học thuận và động học ngược. Nhờ vậy, ta kết luận mô hình động học của robot đã được xây dựng chính xác.

Chương 4. HOẠCH ĐỊNH QUỸ ĐẠO ROBOT RV-2AJ

4.1. Thuật toán

4.1.1. Quỹ đạo vận tốc hình thang

Hoạch định quỹ đạo là bài toán tìm vị trí của các khớp theo thời gian và không gian trong quá trình robot di chuyển. Robot RV-2AJ được hoạch định quỹ đạo vận tốc hình thang, còn gọi là quỹ đạo LSBP (Linear Segment with Parabolic Blend). Với quỹ đạo này, đồ thị vị trí khớp theo thời gian $q(t)$ có dạng parabol ở hai đầu và tuyến tính ở giữa. Chuyển động được chia thành 3 giai đoạn: tăng tốc, chuyển động đều, hãm tốc với các khoảng thời gian tương ứng là T_a, T_v, T_d . Vị trí đầu và cuối là P_0, P_d . Vận tốc tăng dần từ 0, khi chuyển động đều là V_p , rồi giảm dần về 0. Gia tốc khi tăng tốc là A_p , khi giảm tốc là $-D_p$.



Hình 4.1. Đồ thị quỹ đạo LSBP

Giả sử các ràng buộc cho trước là P_0, P_d, V_p, T_a, T_d , bài toán hoạch định quỹ đạo cho một khớp của robot được giải như sau.

Ta có gia tốc và hãm tốc

$$A_p = \frac{V_p}{T_a}, \quad D_p = \frac{V_p}{T_d} \quad (4.1)$$

Thời gian chuyển động đều

$$T_v = \frac{P_d - P_0}{V_p} - \frac{1}{2}(T_a + T_d) \quad (4.2)$$

- Trường hợp 1: $T_v \geq 0$, quỹ đạo ở dạng đầy đủ.

Vận tốc theo thời gian

$$\dot{q}(t) = \begin{cases} (V_p/T_a)t, & accel \\ V_p, & constant \\ -(V_p/T_d)(t - T_a - T_v) + V_p, & decel \end{cases} \quad (4.3)$$

- Trường hợp 2: $T_v < 0$, quỹ đạo ở dạng hình tam giác, do khoảng cách vị trí không đủ lớn để vận tốc có thời gian đạt đỉnh V_p .

Vận tốc đỉnh được tính lại

$$V_{p_{tri}} = \text{sign}(P_d - P_0) \sqrt{\frac{P_d - P_0}{\frac{1}{A_p} + \frac{1}{D_p}}} \quad (4.4)$$

Lúc này thời gian gia tốc và hãm tốc bằng

$$T_{a_{tri}} = \frac{V_{p_{tri}}}{A_p}, \quad T_{d_{tri}} = \frac{V_{p_{tri}}}{D_p} \quad (4.5)$$

Quỹ đạo vận tốc

$$\dot{q}(t) = \begin{cases} \left(\frac{V_{ptri}}{T_{atri}}\right)t, & 0 < t \leq T_{atri} \\ -\left(\frac{V_{ptri}}{T_{dtri}}\right)(t - T_{atri}) + V_{ptri}, & T_{atri} < t \leq T_{atri} + T_{dtri} \\ 0, & otherwise \end{cases} \quad (4.6)$$

Sau khi tìm được $\dot{q}(t)$, dễ dàng suy ra $q(t)$ và $\ddot{q}(t)$.

4.1.2. Đồng bộ chuyển động giữa các khớp

Đồng bộ chuyển động giữa các khớp được hiểu là làm cho tất cả các khớp bắt đầu và kết thúc chuyển động một cách đồng thời. Phương pháp này giúp tăng tính nhất quán và ổn định cho robot.

$$T_{f_1} = T_{f_2} = \dots = T_{f_n} \quad (4.7)$$

Trở ngại đầu tiên là với khoảng cách cần đi khác nhau, nếu ta hoạch định rời rạc từng khớp thì tất yếu thời gian chuyển động ở mỗi khớp sẽ lệch nhau.

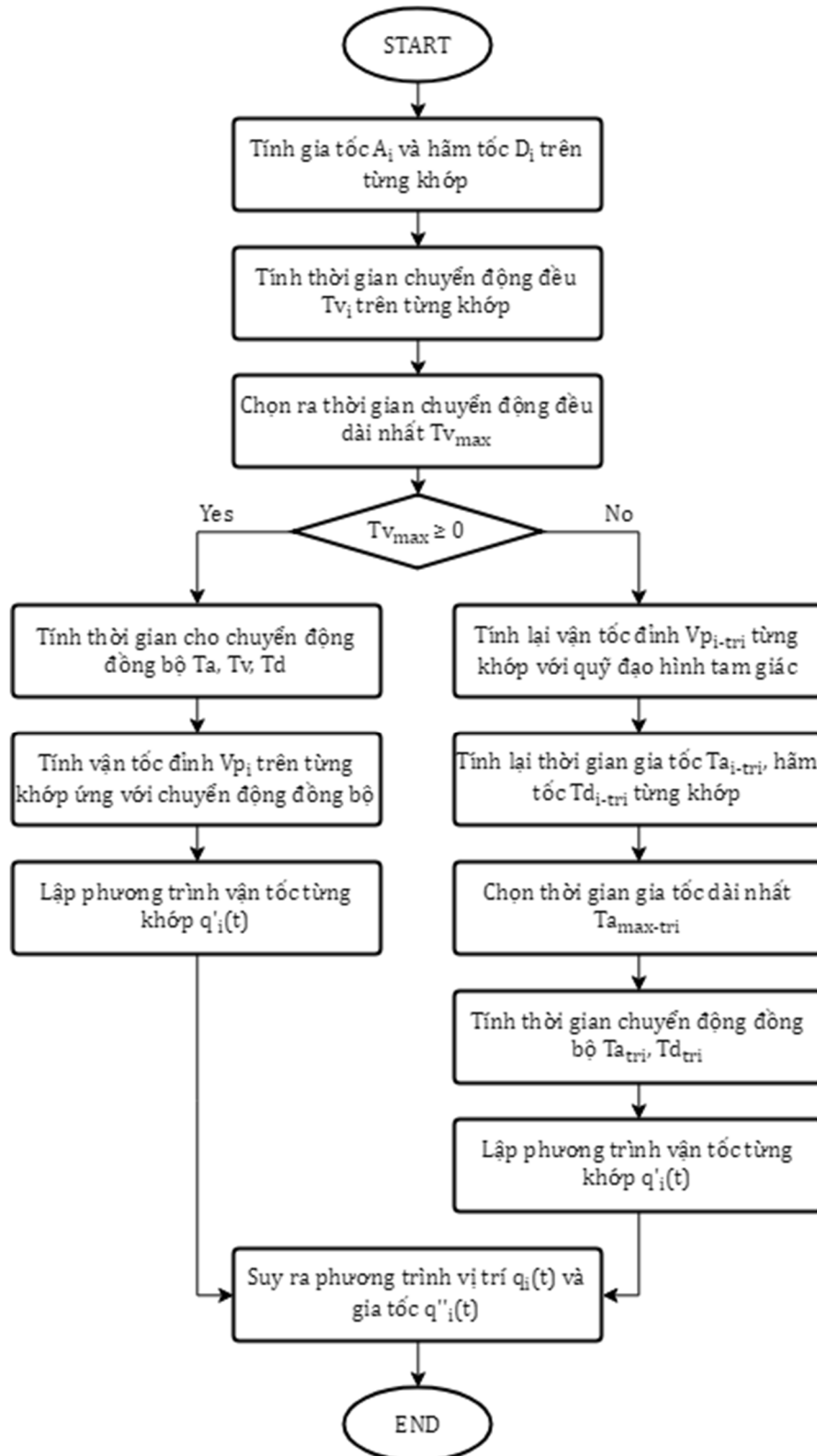
$$T_{f_1} \neq T_{f_2} \neq \dots \neq T_{f_n} \quad (4.8)$$

Mặt khác, khi chuyển động, mỗi khớp của robot phải chịu lực quán tính tỉ lệ với vận tốc. Để đảm bảo không phá huỷ cho cơ hệ, các khớp luôn được giới hạn vận tốc tối đa. Tuy nhiên, do đặc tính động lực học khác nhau nên mỗi khớp cũng có vận tốc tối đa khác nhau. Việc hoạch định vận tốc không được vượt quá giá trị này.

$$V_{p_i} \leq V_{m_i} \quad (4.9)$$

Từ đó, ý tưởng để giải quyết bài toán là trong những khoảng thời gian chuyển động khác nhau tính bởi khoảng cách cần đi của mỗi khớp, ta chọn khoảng thời gian dài nhất làm thời gian quy hoạch chung để đồng bộ mọi khớp. Chọn thời gian dài nhất sẽ đảm bảo không góc khớp nào vượt quá tốc độ tối đa.

Tổng hợp lại, ta có thuật toán hoạch định quỹ đạo hoàn chỉnh.



Hình 4.2. Thuật toán hoạch định quỹ đạo LSPB đồng bộ các khớp

4.2. Tìm thông số của hàm tính thời gian tăng tốc, hãm tốc

Trong phần trên, bài toán hoạch định quỹ đạo được xét với đầu vào là các thông số P_0, P_d, V_p, T_a, T_d . Tuy nhiên đối với nhiều loại tay máy, để thuận tiện cho người dùng, nhà sản xuất sẽ ràng buộc các thông số vận tốc đỉnh V_p và thời gian tăng tốc, hãm tốc T_a, T_d một cách gián tiếp thông qua các biến tỉ lệ phần trăm vận tốc, gia tốc và hãm tốc $V_{\%}, A_{\%}, D_{\%}$. Công thức quy đổi là

$$V_p = V_{\%} V_m \quad (4.10)$$

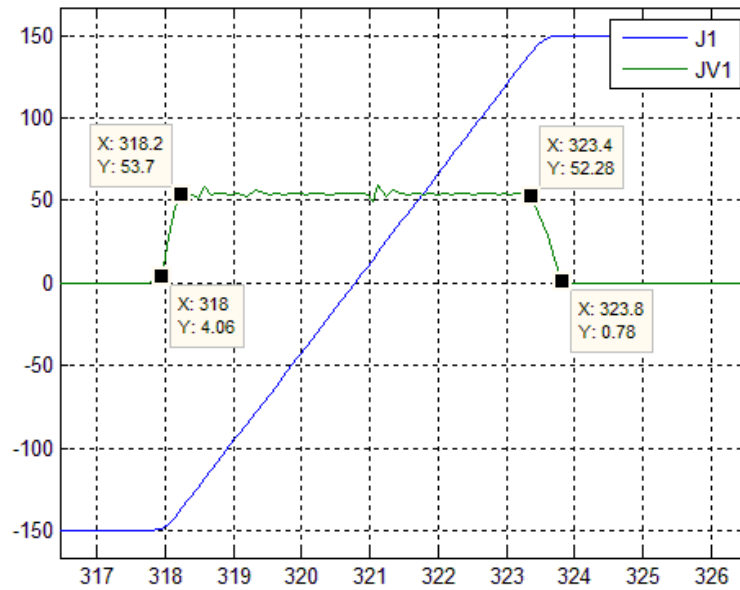
$$T_a = T_{a0} + K_a \left(\frac{V_{\%}}{A_{\%}} \right) \quad (4.11)$$

$$T_d = T_{d0} + K_d \left(\frac{V_{\%}}{D_{\%}} \right) \quad (4.12)$$

trong đó, V_m là hằng số vận tốc tối đa của khớp, T_{a0}, T_{d0} là hằng số thời gian nhỏ nhất mà robot cần để gia tốc, hãm tốc, K_a, K_d là hằng số.

Trong các giá trị V_p, T_a, T_d , ta có thể tính được V_p sử dụng hằng số V_m được công bố trong tài liệu của robot. Ngược lại, không dễ tính được T_a, T_d do các hằng số T_{a0}, T_{d0}, K_a, K_d không được công bố.

Do vậy để tính được T_a, T_d , ta tiến hành khảo sát bằng thực nghiệm. Đầu tiên, ta thu thập dữ liệu quỹ đạo của robot để tìm được thời gian T_a, T_d ứng với các tổ hợp đầu vào $(V_{\%}, A_{\%}, D_{\%})$. Dữ liệu được lấy 3 lần rồi tính trung bình cộng để tăng độ chính xác. Do hạn chế về tốc độ giao tiếp của controller với máy tính, chu kì lấy mẫu khá chậm (vào khoảng 50 ms) gây ra sai số nhất định cho dữ liệu thu thập.



Hình 4.3. Quỹ đạo thực tế khớp J1 của robot tại $V_{\%} = 30$, $A_{\%} = 4$, $D_{\%} = 4$

Thông số đặt $V_{\%}, A_{\%}, D_{\%}$	Thời gian gia tốc T_a [s] – lần 1, 2, 3	Thời gian giảm tốc T_d [s] – lần 1, 2, 3
30, 1, 1	1.0928, 1.0051, 1.0494	1.7925, 1.7831, 1.7015
30, 2, 2	0.5740, 0.6238, 0.5224	0.9078, 0.8750, 0.9143
30, 3, 3	0.3908, 0.4547, 0.4237	0.6064, 0.6074, 0.5968
30, 4, 4	0.2439, 0.2714, 0.2374	0.4862, 0.4734, 0.4771
30, 5, 5	0.2447, 0.2469, 0.2428	0.2971, 0.3220, 0.3242
30, 10, 10	0.1764, 0.1935, 0.1810	0.1627, 0.1536, 0.1974
30, 20, 20	0.1472, 0.1425, 0.1125	0.1243, 0.1338, 0.1174
30, 30, 30	0.1056, 0.1168, 0.1645	0.1387, 0.1471, 0.1481
30, 40, 40	0.1242, 0.1281, 0.1168	0.1386, 0.1128, 0.1412
30, 50, 50	0.1166, 0.1317, 0.1168	0.1210, 0.1390, 0.1039
30, 60, 60	0.1298, 0.1524, 0.1511	0.1170, 0.1151, 0.1556
30, 70, 70	0.1256, 0.0922, 0.0829	0.1452, 0.1412, 0.1202
30, 80, 80	0.0998, 0.1066, 0.1743	0.1333, 0.1167, 0.1419
30, 90, 90	0.1302, 0.1312, 0.1298	0.1418, 0.1566, 0.1185
30, 100, 100	0.1560, 0.1799, 0.1911	0.0927, 0.1308, 0.1357

Bảng 4.1. Trích bộ dữ liệu khảo sát quỹ đạo robot RV-2AJ tại $V_{\%} = 30$

Vì T_a, T_d là các hàm tuyến tính nên ta có thể ước lượng hệ số của chúng bằng phương pháp hồi quy tuyến tính. Dưới đây là lời giải cho T_a , thực hiện tương tự cho T_d .

Đặt

$$\Theta = \begin{bmatrix} T_{a0} \\ K_a \end{bmatrix}, X = \begin{bmatrix} 1 & V_{\%1}/A_{\%1} \\ 1 & V_{\%2}/A_{\%2} \\ \vdots & \vdots \\ 1 & V_{\%m}/A_{\%m} \end{bmatrix}, Y = \begin{bmatrix} T_{a1} \\ T_{a2} \\ \vdots \\ T_{am} \end{bmatrix} \quad (4.13)$$

Khi đó

$$Y = X\Theta \quad (4.14)$$

$$\Theta = (X^T X)^{-1} X^T Y \quad (4.15)$$

Thu được kết quả các hàm ước lượng tuyến tính

$$T_a = 0.0960 + 0.0310 \left(\frac{V_{\%}}{A_{\%}} \right)$$

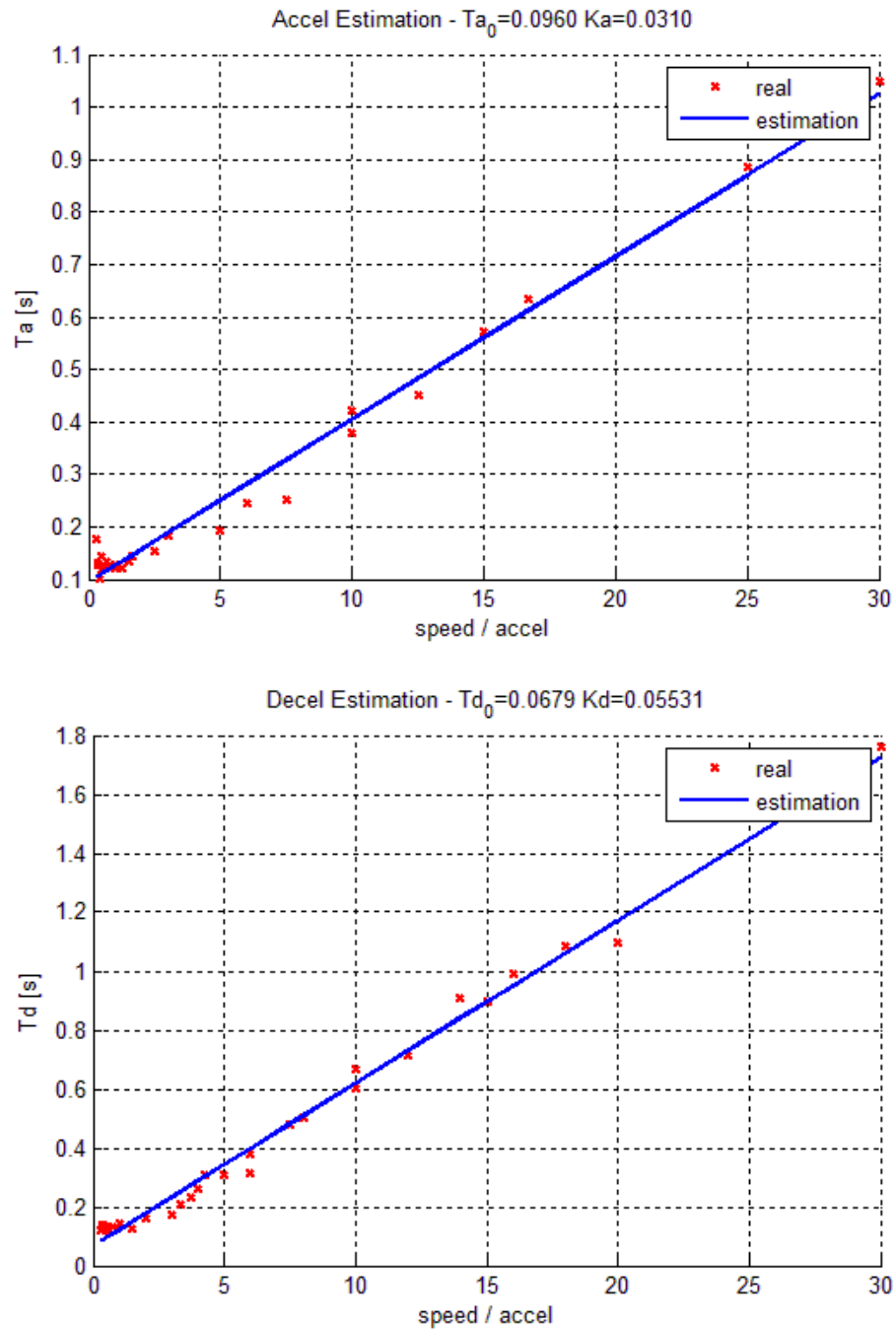
$$T_d = 0.0679 + 0.0553 \left(\frac{V_{\%}}{D_{\%}} \right)$$

Sai số ước lượng

$$RMSE_{T_a} = 0.0277 [s]$$

$$RMSE_{T_d} = 0.0390 [s]$$

Sai số trên là chấp nhận được vì nhỏ hơn chu kỳ lấy mẫu (50 ms). Thêm vào đó, biểu diễn bằng đồ thị cho nhận thấy kết quả ước lượng đạt yêu cầu.



Hình 4.4. Đồ thị kết quả ước lượng T_a và T_d

4.3. Mô phỏng

4.3.1. Xây dựng mô phỏng

Từ thuật toán và các thông số tìm được ở trên, chương trình mô phỏng hoạch định quỹ đạo robot được xây dựng bằng phần mềm MATLAB m-file: *mitsu_trajectory_planning.m*

4.3.2. Kết quả mô phỏng

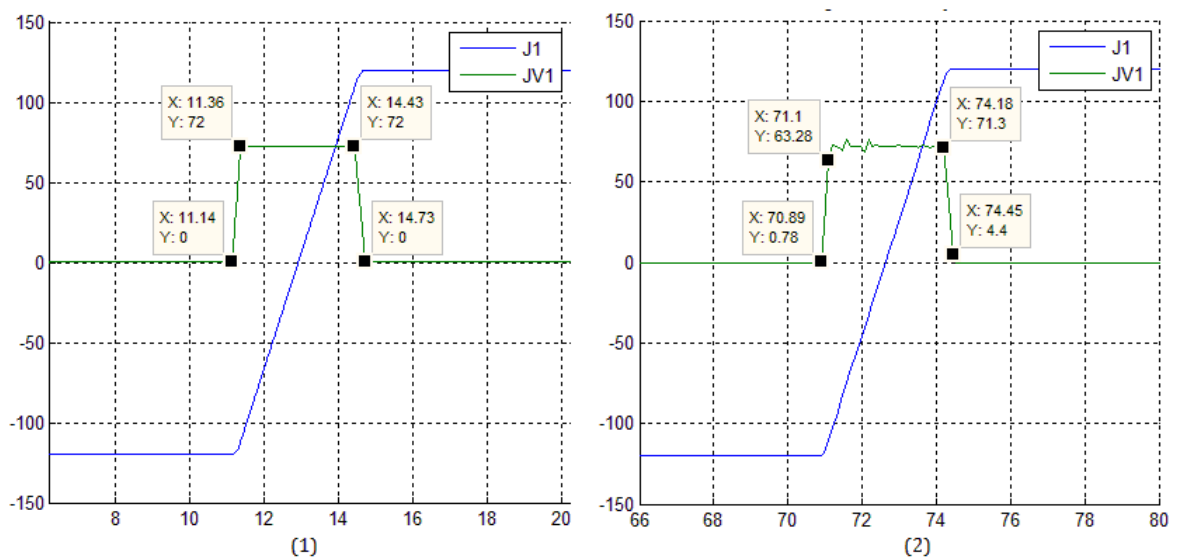
4.3.2.1. Kết quả quỹ đạo một khớp

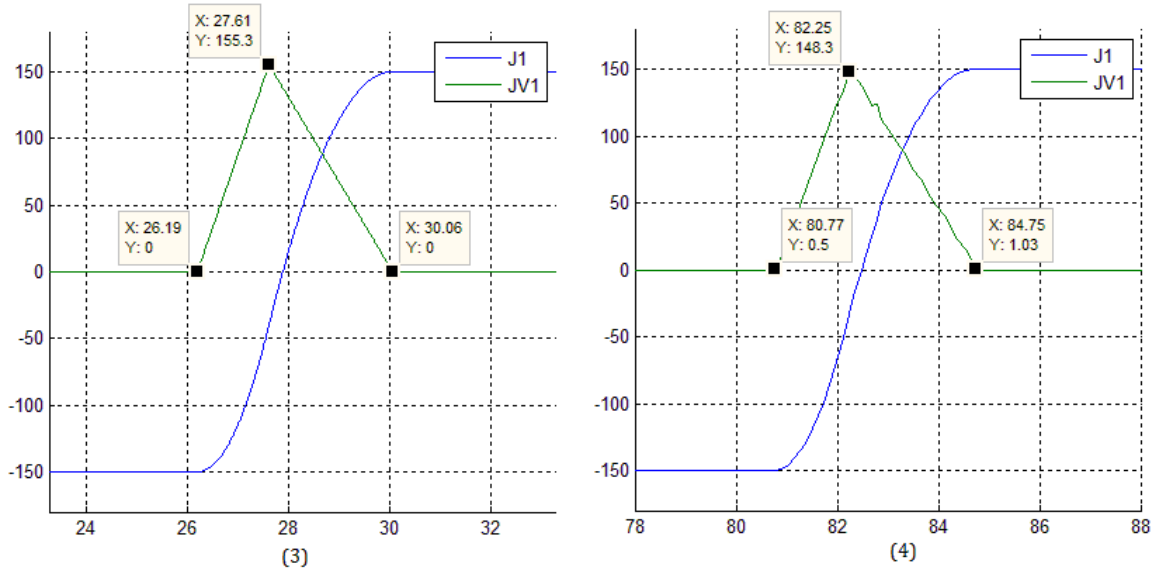
Để kiểm chứng kết quả mô phỏng so với thực tế, ta lấy một số mẫu dữ liệu nằm ngoài tập dữ liệu tính toán hồi quy. Kiểm tra đại diện hai trường hợp quỹ đạo hình thang và quỹ đạo hình tam giác

Đầu vào $P_0, P_d, V_{\%}, A_{\%}, D_{\%}$	Mô phỏng $T_a, T_v, T_d [s]$	Thực tế $T_a, T_v, T_d [s]$
-120°, 120°, 40, 10, 10	0.220, 3.070, 0.300	0.247, 3.045, 0.295
-150°, 150°, 100, 2, 2	1.420, 0.000, 2.500	1.478, 0.000, 2.450

Bảng 4.2. So sánh quỹ đạo mô phỏng với thực tế của robot RV-2AJ

So sánh cho thấy kết quả chương trình mô phỏng quỹ đạo một khớp là rất sát với thực tế.





Hình 4.5. Đồ thị quỹ đạo mô phỏng và thực tế của robot RV-2AJ

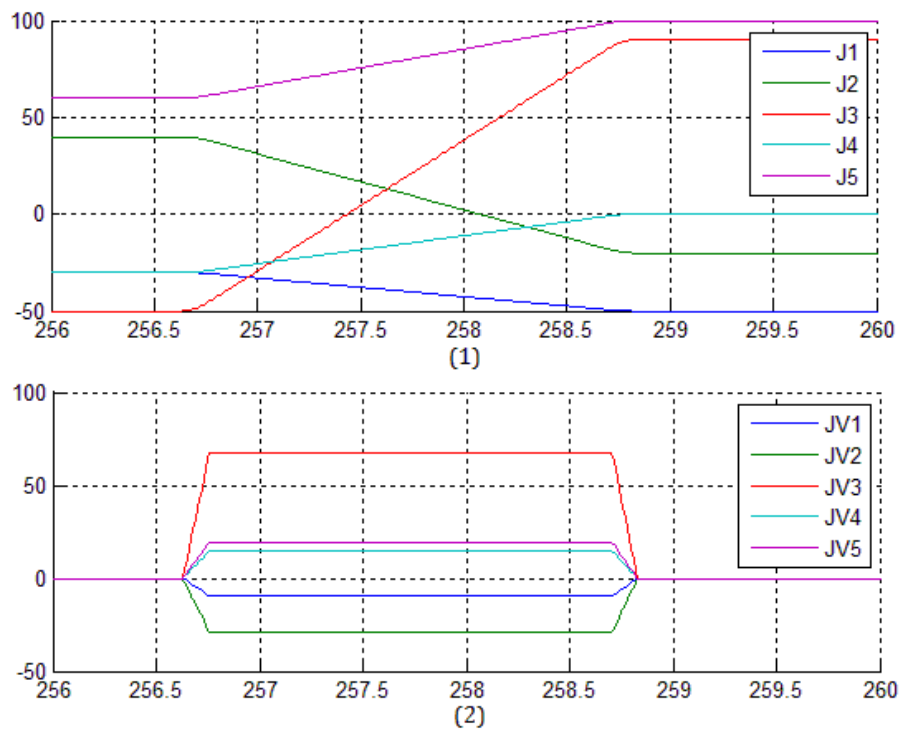
4.3.2.1. Kết quả quỹ đạo đồng bộ các khớp

Tính năng đồng bộ chuyển động các khớp của chương trình được thử nghiệm với một tập đầu vào như sau.

$V_{\%} = 50$	$J1_0 = -30$	$J2_0 = 40$	$J3_0 = -50$	$J4_0 = -30$	$J5_0 = 60$
$A_{\%} = 50$	$J1_d = -50$	$J2_d = -20$	$J3_d = 90$	$J4_d = 0$	$J5_d = 100$
$D_{\%} = 50$					

Bảng 4.3. Đầu vào thử nghiệm đồng bộ các khớp

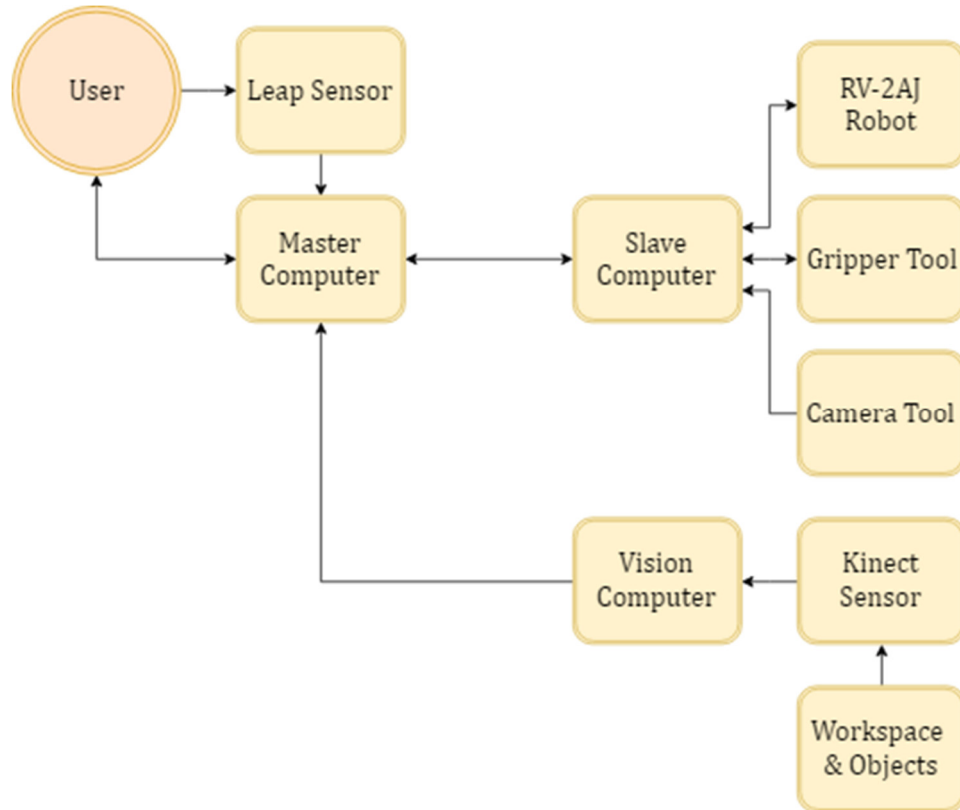
Kết quả vị trí và vận tốc các khớp theo thời gian cho thấy với các khớp robot đã được đồng bộ đúng yêu cầu.



Hình 4.6. Đồ thị mô phỏng quỹ đạo LBSP đồng bộ các khớp

Chương 5. THIẾT KẾ HỆ THỐNG

5.1. Tổng quan



Hình 5.1. Sơ đồ khối thiết kế hệ thống

Về tổng quan, hệ thống gồm ba trạm chủ (Master), tớ (Slave) và thị giác (Vision). Trạm Master đọc dữ liệu từ cảm biến Leap Sensor và nhận lệnh từ người điều khiển, rồi truyền đến trạm Slave. Tại đây, trạm Slave điều khiển cánh tay Robot và dụng cụ Gripper, nhận phản hồi trạng thái từ chúng, đồng thời đọc hình ảnh 2D từ Camera Tool rồi gửi về Master. Bên cạnh đó, trạm Vision ghi lại hình ảnh 3D quan sát không gian làm việc - Workspace của robot thông qua Kinect Sensor, truyền về Master để dựng thành môi trường ảo 3D.

Để xây dựng hệ thống, ngoài các phần cứng mua từ nhà sản xuất như máy tính, cảm biến Leap, cảm biến Kinect, camera, tác giả đã tự thiết kế và thi công dụng cụ Gripper, khu vực bàn làm việc và các giá cảm biến.

5.2. Thiết kế và thi công dụng cụ công tác

5.2.1. Một số phương án thiết kế

5.2.1.1. Giác hút

Giác hút – hay cụ thể hơn là giác hút chân không – là loại cơ cấu chấp hành của tay máy, dùng khí nén hoặc thủy lực để tạo ra lực hút và giữ vật.



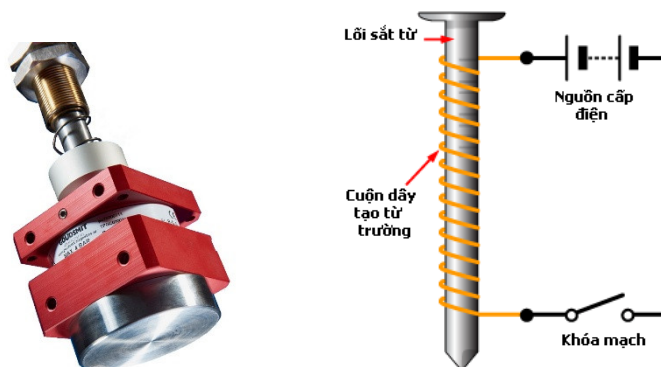
Hình 5.2. Valve chân không và kí hiệu

Van chân không hoạt động dựa trên nguyên lý ống Venturi: Khi có dòng khí áp suất cao di chuyển từ cổng 1 sang cổng 3, dòng khí này di chuyển ngang cổng 2 gây sụt áp và làm cho áp suất tại cổng 2 thấp hơn áp suất khí quyển.

Ưu điểm của thiết kế này là đơn giản, phù hợp với các dạng vật phẳng. Tuy nhiên khuyết điểm là không sử dụng được cho các vật có hình dạng phức tạp, đồng thời sinh ra tiếng ồn lớn trong lúc hoạt động.

5.2.1.2. Nam châm điện

Nam châm điện là loại cơ cấu chấp hành dùng lực từ để hút và giữ vật.



Hình 5.3. Nam châm điện và nguyên lý hoạt động

Nam châm điện tạo ra từ trường bằng cách cho một dòng điện có cường độ lớn chạy qua một cuộn dây. Từ trường này có thể được khuếch đại bằng một lõi dẫn từ.

Thiết kế này có ưu điểm là quá trình gắp thả vật nhanh, có thể sử dụng cho các vật không phẳng, đồng thời không gây tiếng ồn khi làm việc. Tuy nhiên nam châm điện chỉ có thể sử dụng cho các vật liệu có từ tính.

5.2.1.3. Tay kẹp

Tay kẹp (Gripper) là cơ cấu chấp hành phổ biến cho tay máy hiện nay. Tay kẹp có nhiều hình dáng, kích thước, chủng loại và có thể hoạt động bằng điện hoặc khí nén. Đối với đa số các tác vụ của tay máy, tay kẹp đều có thể giải quyết được.



Hình 5.4. Tay kẹp cho tay máy

Tay kẹp có ưu điểm là linh hoạt, có thể dùng cho hầu hết các hình dạng vật thể. Khuyết điểm là đòi hỏi thiết kế cơ cấu nhỏ gọn để gắn được lên điểm cuối tay máy.

Căn cứ theo mục đích và đối tượng thao tác của luận văn, phương án tay kẹp được lựa chọn thực hiện.

5.2.2. Thiết kế và thi công tay kẹp

5.2.2.1. Mục đích và yêu cầu

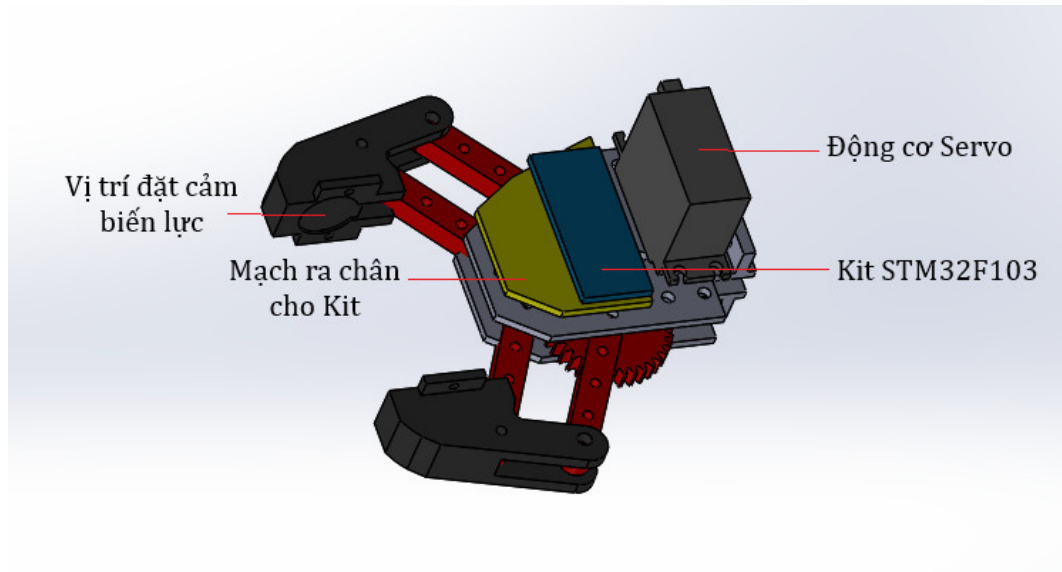
Dụng cụ công tác hiện có của tay máy là giác hút, không phù hợp với đối tượng cần thao tác có hình dạng và chất liệu bất kỳ, vì vậy cần một thiết bị mới thay thế. Thiết bị được chọn là tay kẹp.

Tay kẹp được thiết kế cần thỏa mãn các yêu cầu sau:

- Hoạt động ổn định.
- Độ mở tối thiểu 50 mm.
- Trọng lượng vật có thể thao tác đạt 1kg.
- Hai mặt tiếp xúc vật của Tay kẹp luôn song song với nhau.
- Có cảm biến lực hồi tiếp và tự điều chỉnh lực kẹp cho phù hợp với yêu cầu cho trước.
- Hình dáng, kích thước phù hợp với tổng thể tay máy.

5.2.2.2. Bản vẽ thiết kế

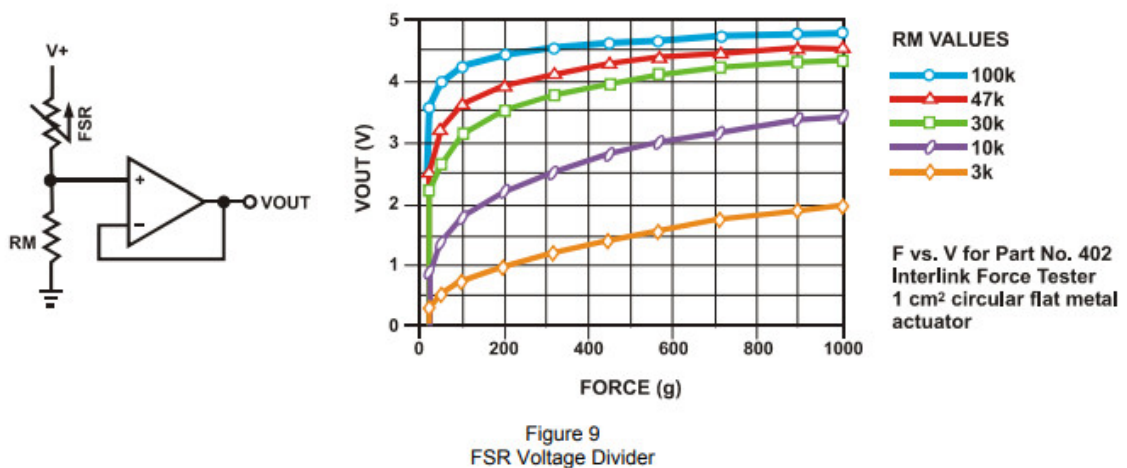
Bản thiết kế dựa trên nguyên mẫu Vex Robot Gripper, thực hiện bằng phần mềm SolidWorks. Sau khi hoàn thành bản vẽ, các chi tiết được gia công bằng phương pháp in 3D.



Hình 5.5. Bản thiết kế tay kẹp

5.2.3. Hiệu chuẩn cảm biến lực

Cảm biến khi được sản xuất đã được hiệu chuẩn theo các thông số của nhà sản xuất. Tuy nhiên, do điều kiện làm việc khác nhau nên khi sử dụng cần hiệu chuẩn lại. Hiệu chuẩn được thực hiện nhằm mục đích xác định được các giá trị điện trở tương ứng với các cường độ lực khác nhau, từ đó nội suy tuyến tính để được giá trị điện trở tại các cường độ lực bất kỳ.



Hình 5.6. Thông số hiệu chuẩn cảm biến lực FSR402 từ nhà sản xuất

Các dụng cụ dùng để hiệu chuẩn gồm có:

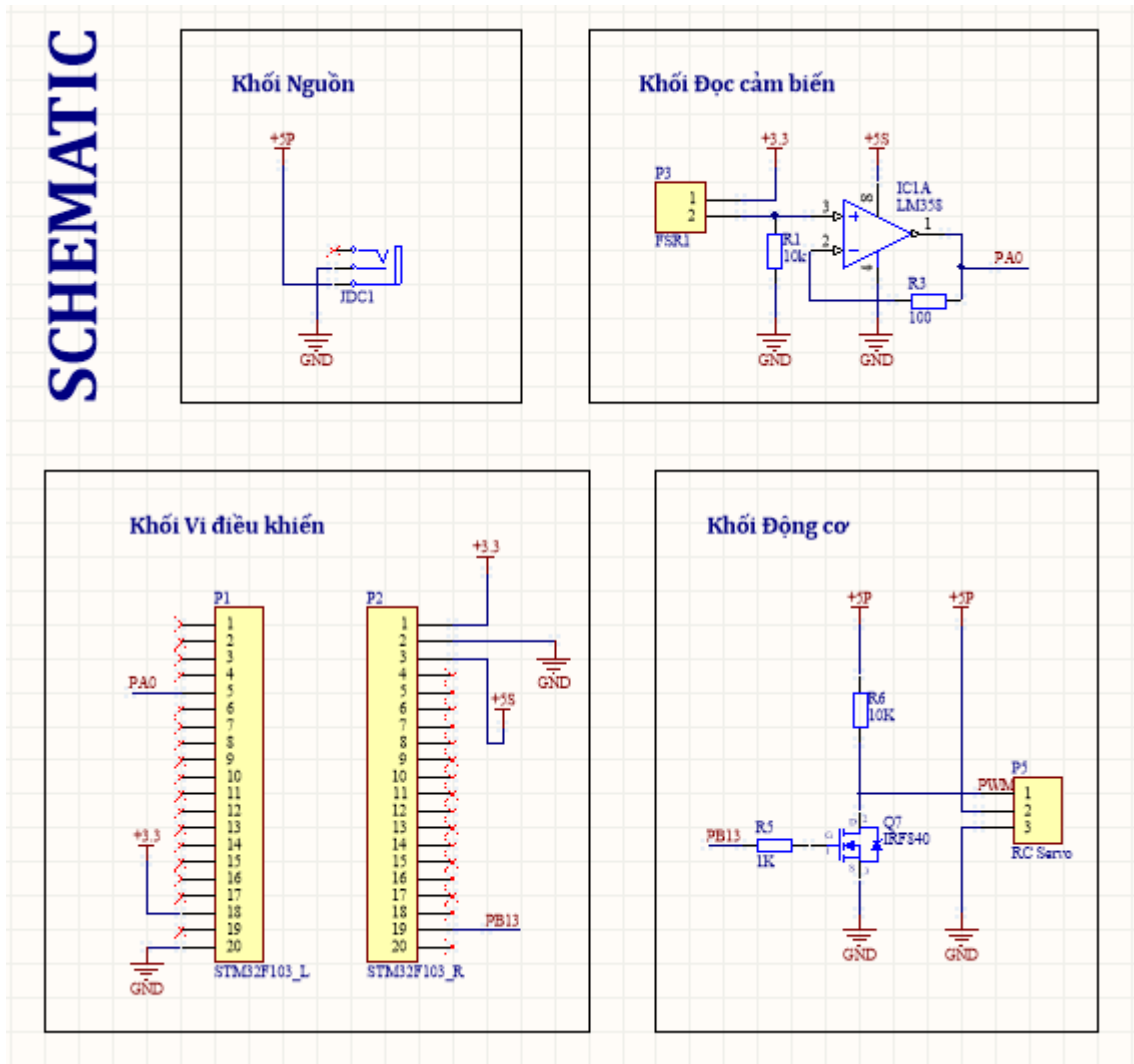
- Các quả nặng có khối lượng khác nhau.
- Cân điện tử dùng để xác định chính xác trọng lượng quả nặng.

Khối lượng [kg]	Trọng lượng [N]	Điện trở [k Ω]
0	0	0
0.04	0.39	1.78
0.09	0.88	4.91
0.14	1.37	7.84
0.54	5.30	19.33
0.60	5.89	31.09
0.91	8.92	44.85
1.37	13.44	56.40

Bảng 5.1. Kết quả hiệu chuẩn cảm biến lực FSR402

5.2.4. Mạch ra chân Kit STM32F103C8T6

5.2.4.1. Sơ đồ nguyên lý



Hình 5.7. Sơ đồ nguyên lý mạch ra chân cho Kit STM32F103

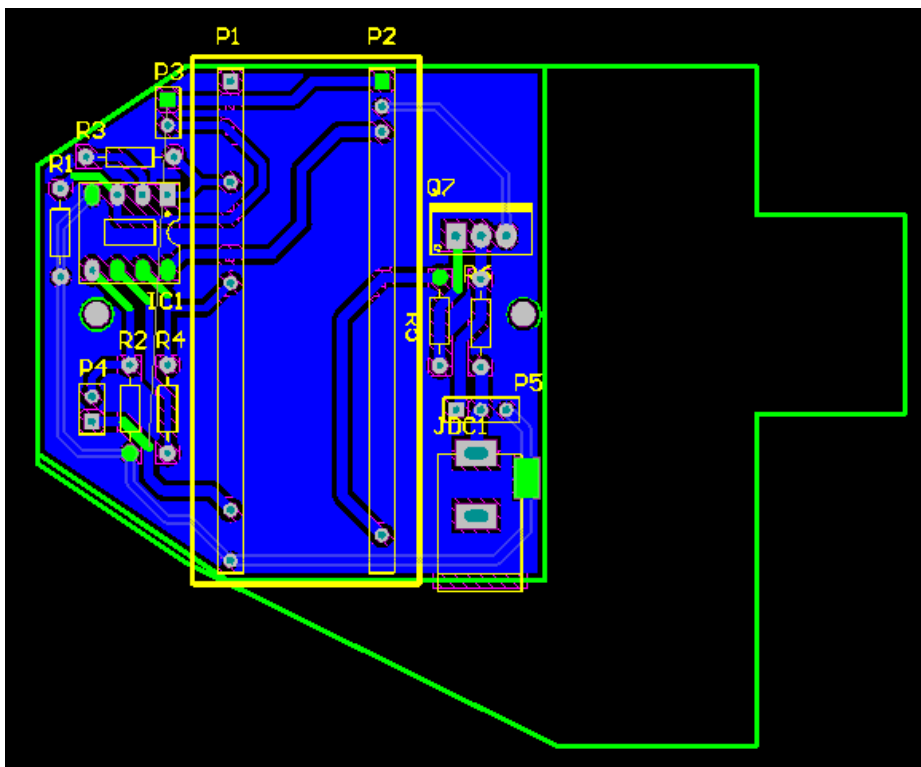
Khối Nguồn: Nhận nguồn 5VDC từ jack cắm và cung cấp cho mạch.

Khối Đọc cảm biến: Gồm chân cắm và mạch phân cực cho cảm biến lực, chuyển đổi giá trị điện trở sang giá trị điện áp để Vi điều khiển đọc ADC.

Khối Vi điều khiển: Gồm hai header cắm Vi điều khiển và chân nối sẵn nguồn cho Vi điều khiển hoạt động.

Khối Động cơ: Gồm chân cắm nguồn cho động cơ và MOSFET làm nhiệm vụ khuếch đại tín hiệu PWM từ vi điều khiển điều khiển động cơ.

5.2.4.2. Sơ đồ Layout



Hình 5.8. Layout cho mạch ra chân kit STM32F103

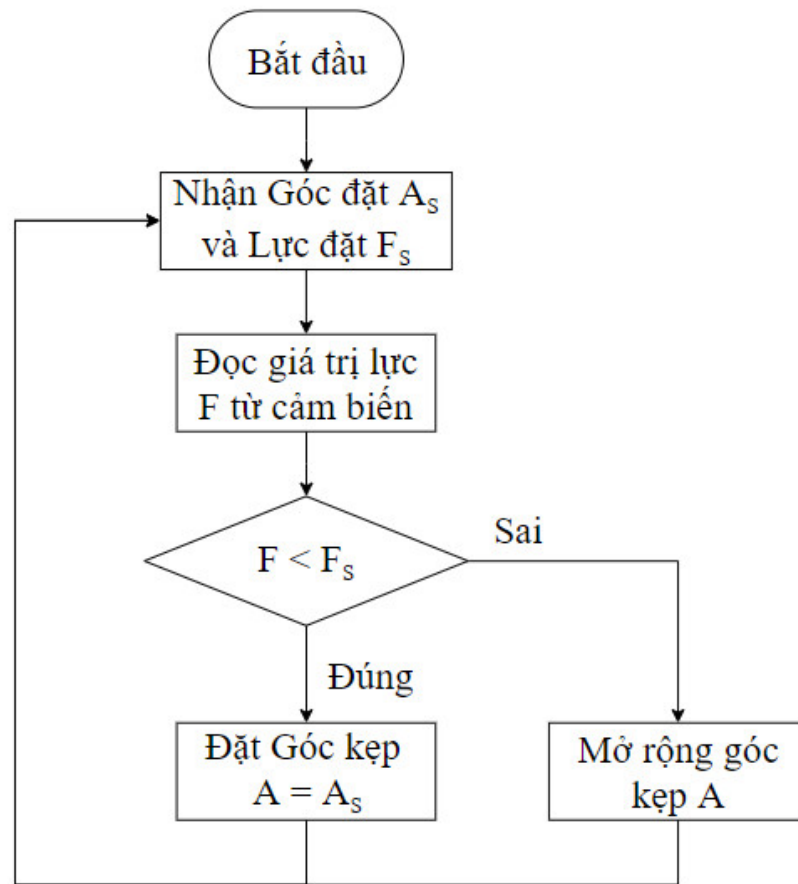
5.2.5. Giải thuật điều khiển Tay kẹp

Kit STM32F108 điều khiển Tay kẹp bằng động cơ Servo và giao tiếp với máy tính thông qua chuẩn giao tiếp nối tiếp RS232.

Khi bắt đầu chạy, máy tính gửi giá trị lực đặt F_s (là giá trị lực kẹp tối đa cho phép) đến vi điều khiển làm giá trị tham chiếu. Mỗi khi có tín hiệu góc gửi từ máy tính, vi điều khiển sẽ so sánh lực hiện tại F với lực đặt F_s .

- Nếu $F < F_s$ (tức lực hiện tại nhỏ hơn giá trị đặt): Vi điều khiển sẽ tuân theo máy tính, đặt góc kẹp là góc nhận được từ máy tính.

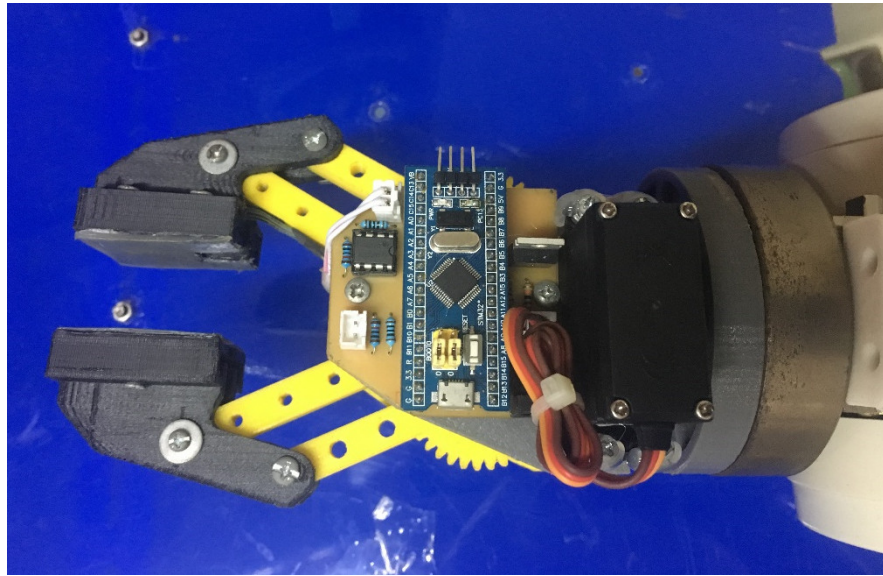
- Nếu $F > F_s$ (tức lực hiện tại lớn hơn giá trị đặt): Vi điều khiển sẽ không tuân theo máy tính. Thay vào đó vi điều khiển sẽ mở góc kẹp rộng ra nhằm giảm lực kẹp.



Hình 5.9. Lưu đồ chương trình Vi điều khiển

Việc điều khiển có hồi tiếp lực theo cơ chế trên sẽ giúp điều chỉnh được lực kẹp của Tay kẹp, phù hợp với các đối tượng cần thao tác khác nhau. Đồng thời, với tín hiệu điều khiển góc liên tục nhận được từ máy tính, lực kẹp sẽ nhanh chóng hội tụ và ổn định tại giá trị đặt.

5.2.6. Kết quả thực hiện Tay kẹp



Hình 5.10. Tay kẹp sau khi hoàn thành

Yêu cầu	Kết quả
Hoạt động ổn định	Đạt
Độ mở	0 cm – 55 cm
Trọng lượng vật gấp	Tối đa 1kg
Hồi tiếp lực	1N – 13N
Kích thước	9 cm x 16 cm x 8 cm

Bảng 5.2. Thông số tay kẹp sau khi hoàn thành.

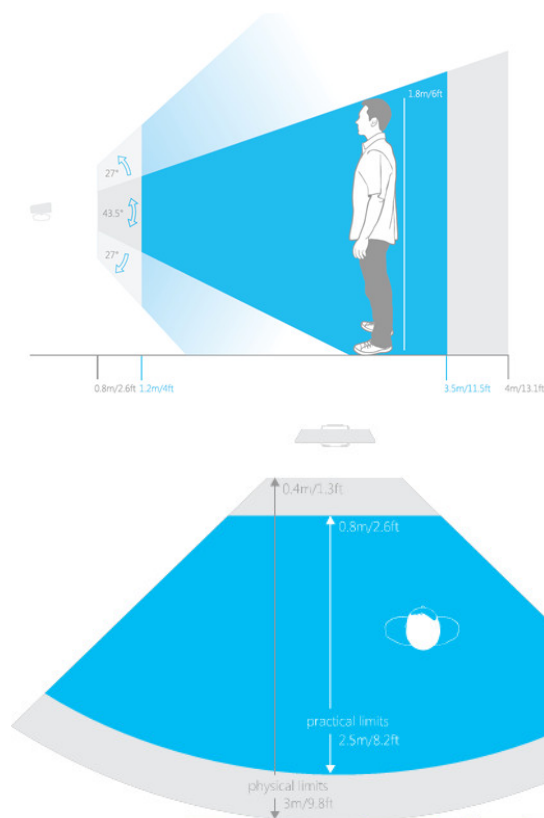
Thông số của tay kẹp thành phẩm đạt yêu cầu thiết kế.

5.3. Thiết kế và thi công Khu vực làm việc

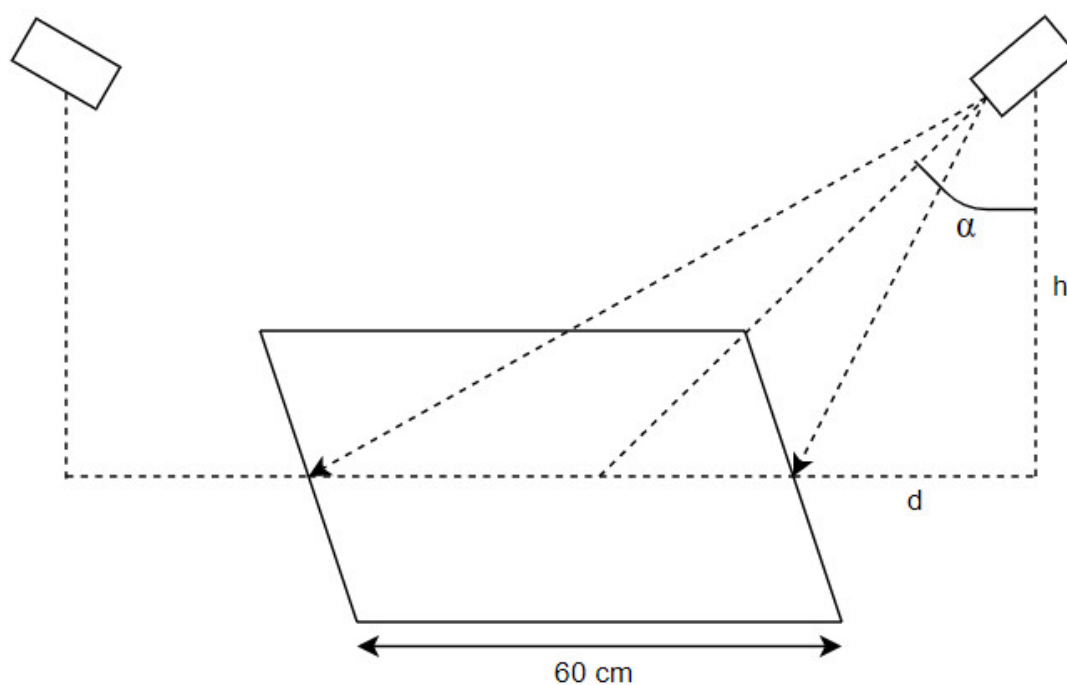
5.3.1. Bố trí Kinect

Camera Kinect có trường quan sát giới hạn, vì vậy cần bố trí thích hợp để có thể qua sát được khu vực làm việc của Tay máy.

Camera Kinect có thể quan sát theo chiều dọc với độ mở 43 độ, theo chiều ngang với độ mở 57 độ và tầm đo của cảm biến độ sâu từ 0.4 mét đến 3 mét.



Hình 5.11. Độ mở của Kinect theo chiều dọc và chiều ngang



Hình 5.12. Phác thảo vị trí bố trí Kinect

Với bàn làm việc có kích thước 600 mm x 400 mm, camera Kinect cần được bố trí tại vị trí có độ cao h so với mặt bàn, cách mép bàn khoảng cách d và nghiêng so với phương thẳng đứng góc α sao cho:

$$\begin{cases} h \cdot \tan\left(\alpha + \frac{43}{2}\right) \geq d + 0.6 \\ h \cdot \tan\left(\alpha - \frac{43}{2}\right) \leq d \end{cases} \quad (5.1)$$

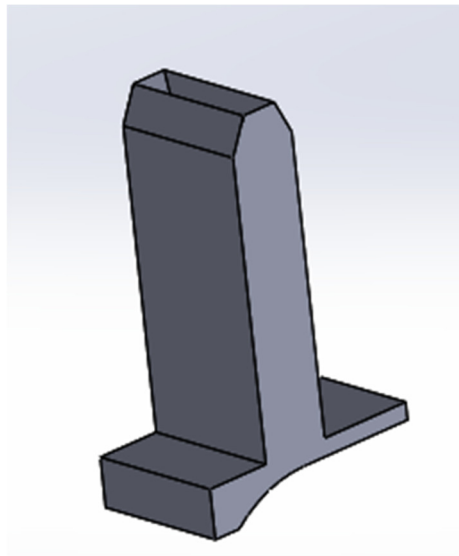
Chọn $h = 0.7m$, $d = 0.3m$ để tính α thỏa hệ phương trình trên, ta được

$$\begin{cases} \alpha \geq 30.6^\circ \\ \alpha \leq 44.7^\circ \end{cases}$$

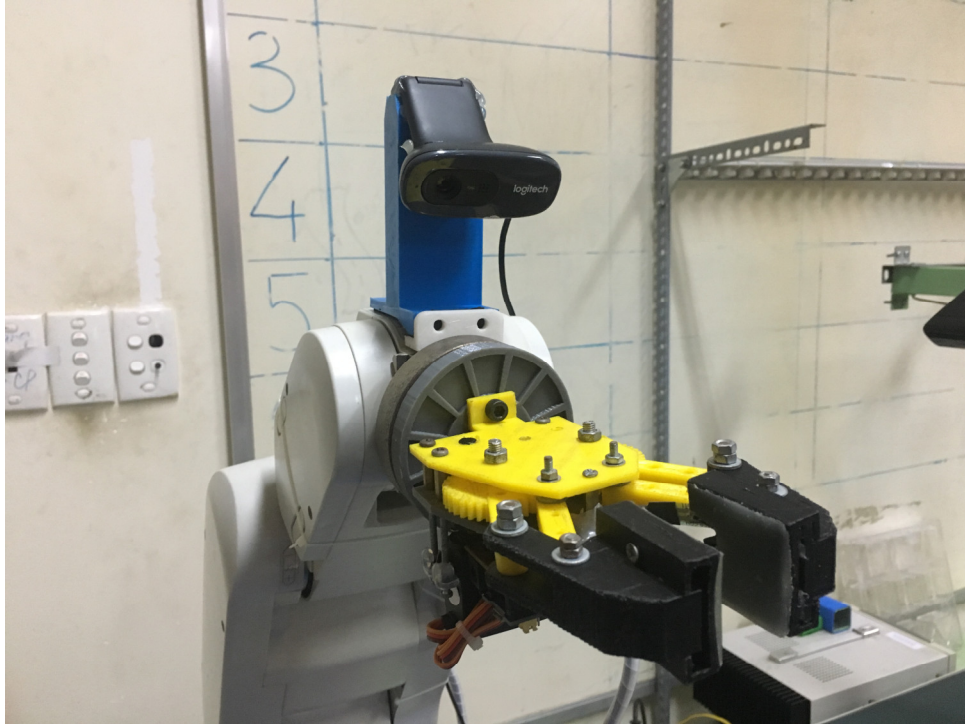
5.3.2. Bố trí Camera điểm cuối

Do thực hiện điều khiển từ xa, người dùng không thể quan sát rõ quá trình thao tác tại điểm cuối tay máy nên cần thiết phải bố trí camera theo dõi hoạt động của điểm cuối.

Bản thiết kế phải đảm bảo có thể cố định được camera 2D khi tay máy hoạt động và quan sát rõ thao tác khi gấp vật của tay kẹp.



Hình 5.13. Bản thiết kế để gắn camera 2D



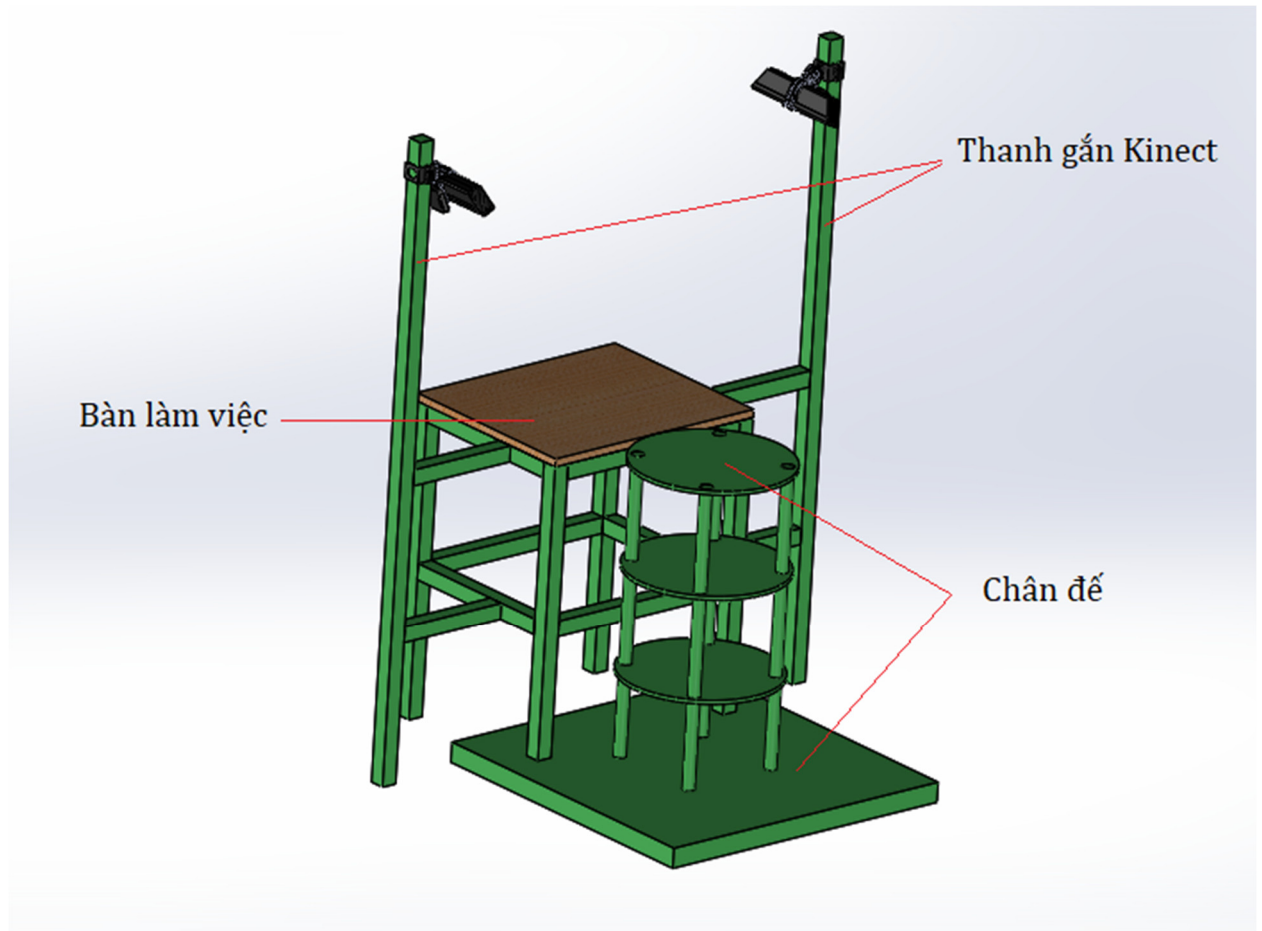
Hình 5.14. Bố trí camera 2D quan sát điểm cuối.

5.3.3. Thiết kế và thi công bàn làm việc

5.3.3.1. Thiết kế bàn làm việc

Thiết kế bàn làm việc nhằm mục đích tạo môi trường làm việc cho Tay máy, cần phù hợp với các đặc tính của Tay máy.

Tầm vươn của Tay máy ra khỏi chân đế theo phương ngang là 310 mm, vì vậy chọn kích thước bàn là 400 mm x 600 mm để tầm hoạt động của Tay máy đạt hơn một nửa bàn. Chiều cao bàn làm việc bằng chiều cao đế là 800 mm giúp việc tính toán đơn giản. Thanh gắn camera Kinect có chiều cao 1700 mm (cao hơn mặt bàn 900 mm), cách mép bàn 300 mm và được cố định vào bàn để tránh rung lắc trong quá trình hoạt động.



Hình 5.15. Mô hình Khu vực làm việc được thiết kế bằng SolidWorks

5.3.3.2. Thi công bàn làm việc

Bàn làm việc robot được thi công bằng một số máy cơ khí thông dụng, sử dụng các vật liệu sau.

STT	Vật liệu	Kích thước [mm]	Số lượng
1	Sắt hộp 4	780	2
		730	2
		520	4
		420	4
		300	4
		1700	2

2	Sắt tấm	800 x 800 x 10	1
3	Gỗ ghép cao su	600 x 400 x 20	1
4	Ke chữ L	-	-
5	Vít mang cá	-	-
6	Sơn	-	-

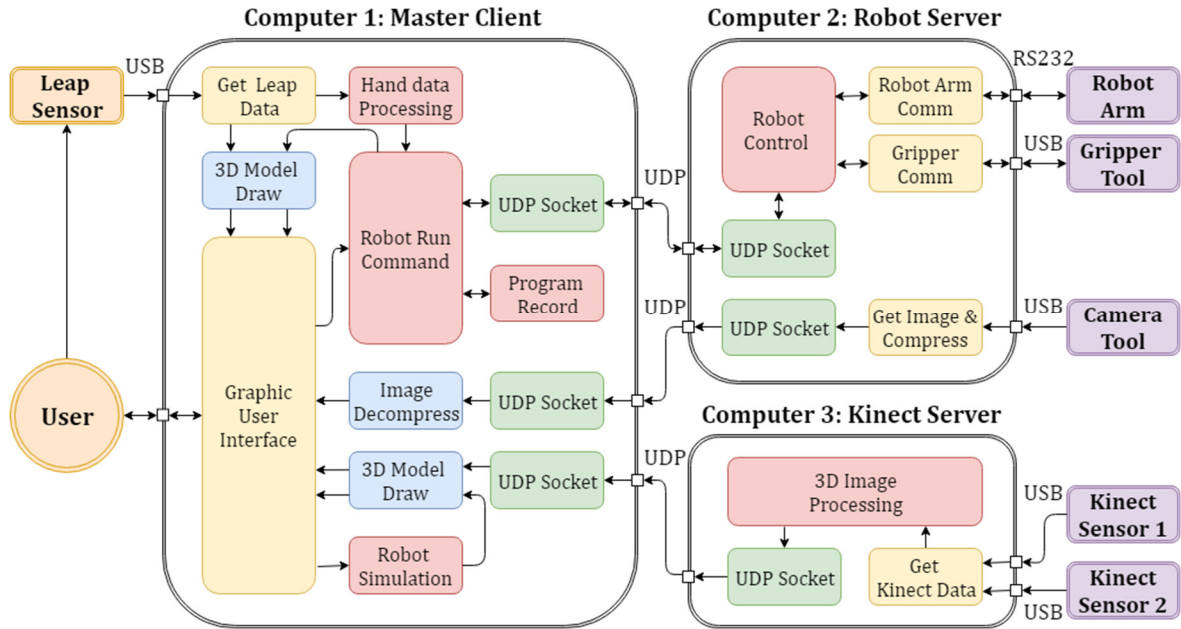
Bảng 5.3. Vật liệu thi công Bàn làm việc



Hình 5.16. Bàn làm việc của tay máy sau khi hoàn thành.

Chương 6. GIẢI THUẬT VÀ CHƯƠNG TRÌNH ĐIỀU KHIỂN

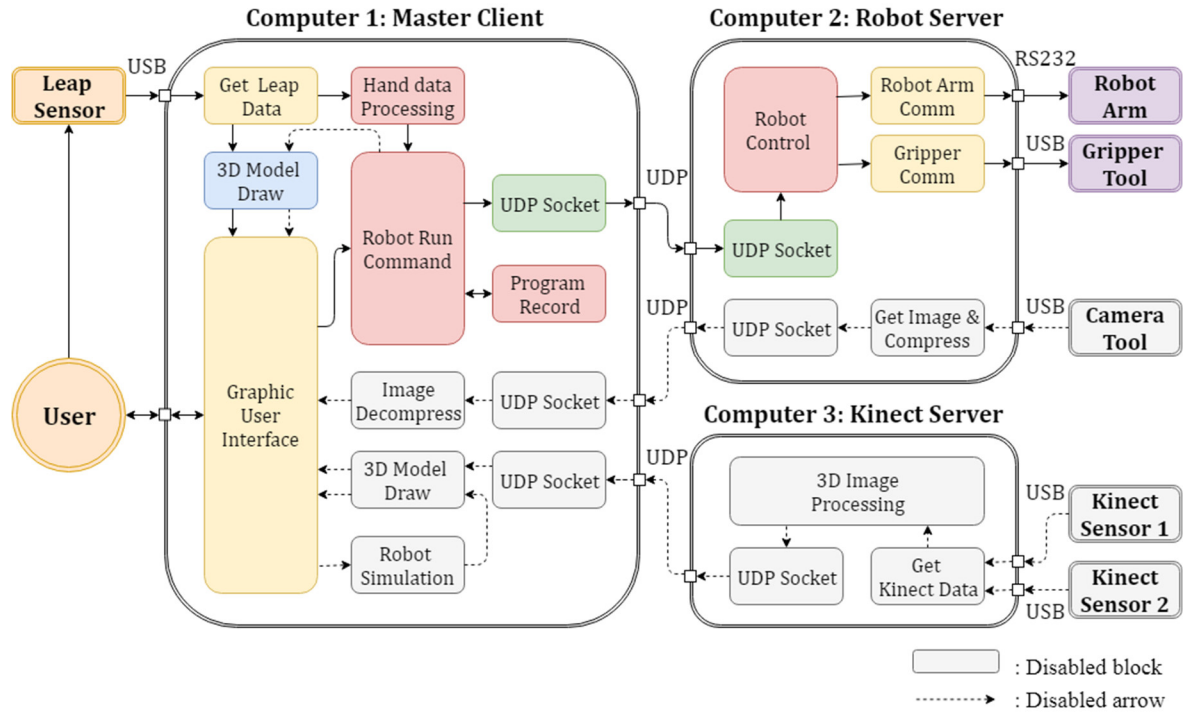
6.1. Tổng quan



Hình 6.1. Sơ đồ tổng quan chương trình máy tính

Hệ thống phần mềm chạy trên 3 máy tính. Trong đó, máy thứ nhất (Master Client) là máy ra lệnh, thuộc về phía người dùng. Máy thứ hai (Robot Server) và thứ ba (Kinect Server) là các máy thực thi lệnh và phản hồi thông tin, nằm ở phía robot. Các máy tính giao tiếp với nhau bằng mạng Ethernet LAN theo giao thức UDP để đảm bảo tải được dữ liệu lớn với độ trễ nhỏ và ổn định.

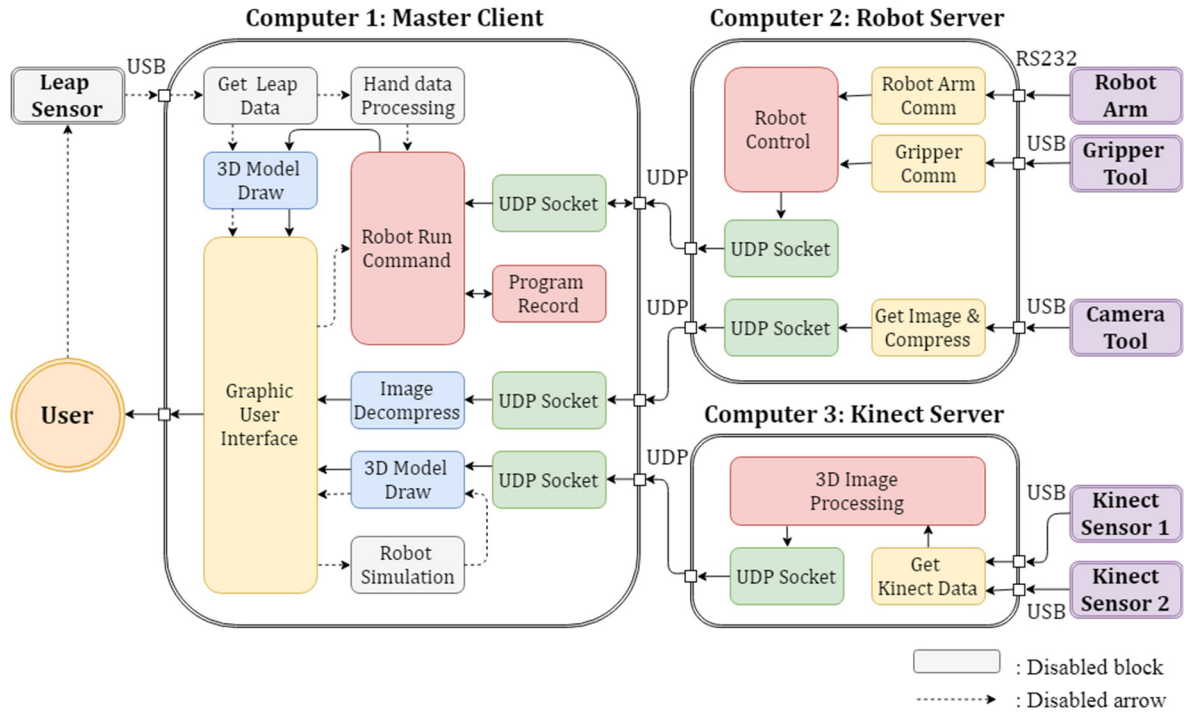
Master Client có nhiệm vụ nhận mệnh lệnh và thao tác từ người dùng thông qua giao diện (GUI) hoặc thông qua cảm biến cử chỉ bàn tay (Leap Sensor), sau đó xử lý và gửi các yêu cầu đến Robot Server dưới dạng tập lệnh giao tiếp quy ước giữa các máy tính. Robot Server xử lý yêu cầu nhận được, biên dịch sang các lệnh của Robot và Gripper và đẩy xuống để Robot và Gripper thực thi.



Hình 6.2. Sơ đồ tổng quan chương trình điều khiển

Ở chiều ngược lại, Robot Server liên tục lấy thông tin vị trí hiện tại của Robot Arm và Gripper Tool, đồng thời lấy hình ảnh tức thời từ Camera Tool quan sát ở đầu công tác truyền phản hồi về Master Client. Song song đó, Kinect Server cũng lấy từ hai cảm biến Kinect hình ảnh 3D tức thời của các vật thể trong khu vực robot làm việc truyền về Master Client. Từ tất cả dữ liệu phản hồi, phía Master hiển thị trực tiếp hình ảnh 2D và dựng lên một môi trường 3D ảo, phản ánh đến người dùng các sự kiện đang diễn ra trong không gian thực phía Robot.

Tóm lại, một vòng điều khiển kín đã được thành lập giữa người và robot thông qua mạng máy tính, phục vụ cho nhu cầu điều khiển từ xa bằng cử động tự nhiên.



Hình 6.3. Sơ đồ tổng quan chương trình chiều về

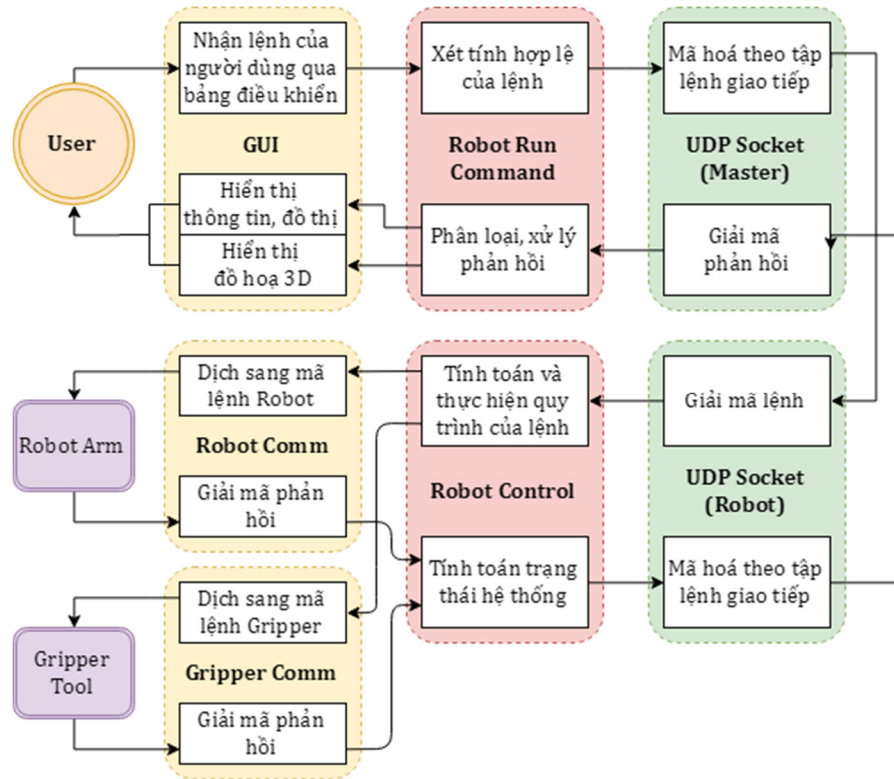
6.2. Chương trình điều khiển Robot

6.2.1. Điều khiển Robot trong chế độ Manual

Manual Control là chế độ điều khiển bằng tay, trong đó robot thực hiện theo các lệnh cơ bản được người dùng chọn trên bảng điều khiển. Các lệnh trong chế độ Manual bao gồm bật tắt động cơ, thiết lập vận tốc, gia tốc, vị trí di chuyển, đặt lực gắp, đặt cấu hình tay máy...

Chương trình thực hiện một lệnh Manual được trình bày theo sơ đồ sau. Khi máy tính Master Client nhận lệnh từ GUI, trước tiên lớp Robot Run Command sẽ xem xét các thông số và tính hợp lệ của lệnh đối với chế độ hiện tại của hệ thống, sau đó lớp UDP Socket mã hoá lệnh theo cú pháp của tập lệnh giao tiếp, rồi gửi qua môi trường mạng UDP. Tại Robot Server, sau khi được giải mã bởi UDP Socket, lệnh được lớp Robot Control phân loại, tính toán thông số cần thiết cho robot và thực hiện quy trình xử lý. Lớp Robot Comm và Gripper Comm chịu

trách nhiệm truyền đạt lệnh đến các bộ điều khiển tầng thấp hơn của Robot và Gripper theo mã giao tiếp riêng.



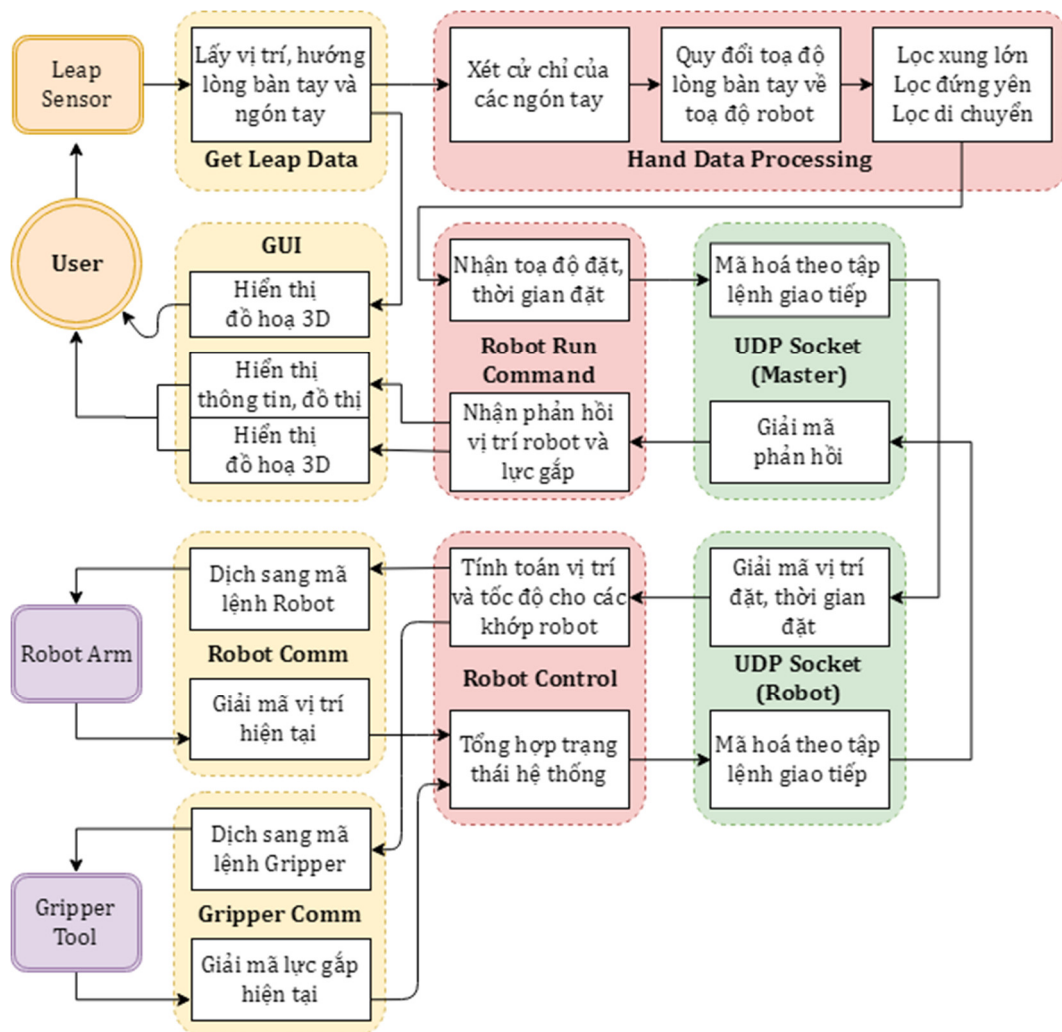
Hình 6.4. Sơ đồ khối điều khiển trong chế độ Manual

Vì người dùng không có mặt tại nơi đặt robot, để kiểm tra kết quả thực hiện lệnh, hệ thống phải phản hồi lại trạng thái trở lại cho người dùng. Từ các phản hồi của Robot và Gripper được giải mã bởi Robot Comm và Gripper Comm, trạng thái của hệ thống sẽ được lớp Robot Control tính toán. Trạng thái hệ bao gồm thông tin về vị trí hiện tại của các khớp và điểm cuối robot, vị trí và giá trị lực kẹp tại Gripper. Cùng với đó là thông báo thực thi lệnh thành công, hoặc có lỗi xảy ra như giá trị đặt vượt quá giới hạn làm việc, di chuyển khi đang tắt động cơ... Mặt khác, ta chú ý là robot chỉ phản hồi vị trí các khớp, giá trị tọa độ điểm cuối có được từ tính toán động học thuận trong lớp Robot Control.

Tất cả trạng thái được phản hồi, xử lý và hiển thị cho người dùng theo chiều ngược lại với chiều mà lệnh đã đi. Trên GUI, trạng thái hệ được hiển thị đồng thời ở ba dạng là thông tin, đồ thị và đồ họa 3D.

6.2.2. Điều khiển Robot trong chế độ Teleoperation

Teleoperation hay Master-Slave Control là chế độ điều khiển tay máy sao chép lại cử động của bàn tay người dùng trong thời gian thực. Bằng cách này, người dùng sẽ điều khiển robot một cách tự nhiên như thực hiện các thao tác hằng ngày bằng bàn tay của mình. Việc nhập dữ liệu bàn tay vào hệ thống được thực hiện thông qua cảm biến Leap Motion.



Hình 6.5. Sơ đồ khối điều khiển trong chế độ Teleoperation

Trước hết, ở phía Master, dữ liệu bàn tay được số hoá với độ chính xác cao bởi cảm biến Leap Motion. Sử dụng thư viện của nhà sản xuất, lớp Get Leap Data sẽ lấy các giá trị vị trí và hướng của lòng bàn tay, vị trí các khớp xương và hướng

của các ngón tay với tần số lấy mẫu là 30Hz. Dữ liệu tách ra theo hai hướng. Một là vào môi trường đồ họa, dựng lên thành mô hình 3D của bàn tay để hiển thị.

Hướng thứ hai, dữ liệu Leap đi vào lớp Hand Data Processing, trải qua 3 giai đoạn xử lý gồm xét cử chỉ, quy đổi tọa độ và lọc nhiễu, để cho ra dữ liệu điều khiển robot. Lớp Robot Run Command nhận tọa độ đặt, thời gian đặt, giá trị góc kẹp, thông qua lớp UDP Socket để truyền đến phía Slave.

Ở phía Slave, lớp Robot Control, tọa độ đặt được dùng để tính vị trí các khớp robot theo bài toán động học ngược, thời gian đặt được dùng để tính vận tốc robot cần di chuyển. Vị trí và vận tốc đặt được truyền xuống robot thông qua lớp Robot Comm. Mặt khác, lớp Gripper Comm truyền xuống gripper giá trị góc kẹp.

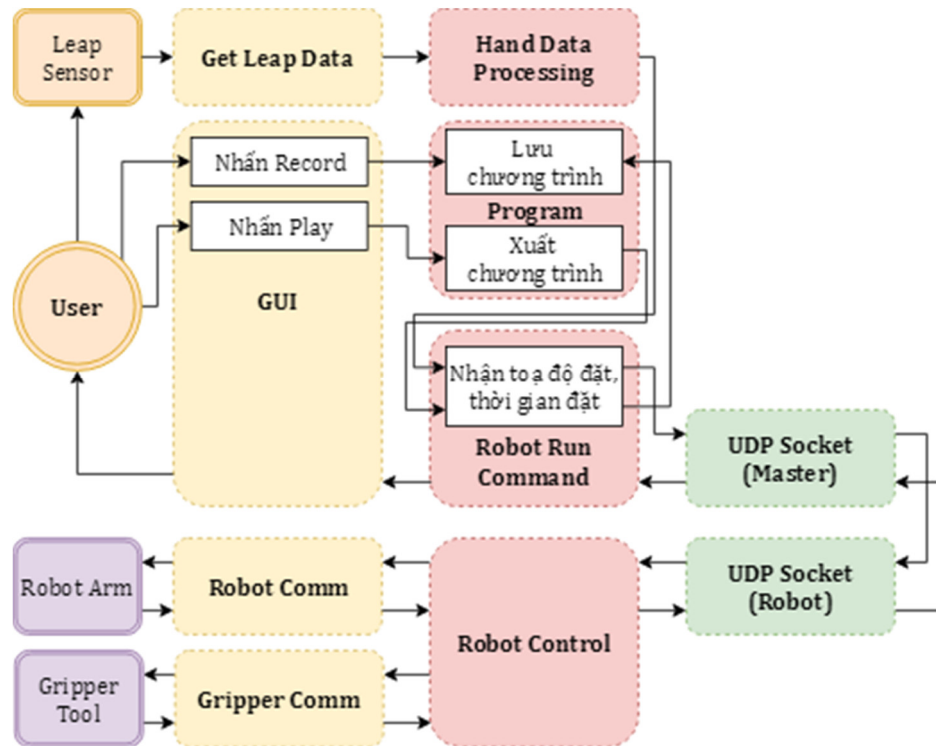
Phần còn lại, Robot, Gripper thực thi và phản hồi trạng thái tương tự như trong chế độ Manual Control.

6.2.3. Điều khiển Robot tự động bằng chức năng Program

Chức năng Program được tạo ra để lưu và lặp lại tự động các phiên làm việc trước của robot. Chức năng này phát huy hiệu quả khi robot cần lặp đi lặp lại rất nhiều lần một thao tác, ví dụ như trong dây chuyền sản xuất. Khi đó người dùng không điều khiển trực tiếp, mà chỉ cần “dạy” thao tác một lần, robot “học” xong sẽ thực hiện lại hoàn toàn tự động.

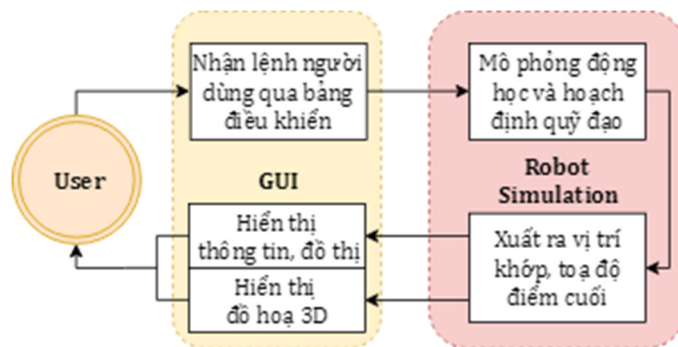
Cách thức hoạt động lớp Program khá đơn giản, có thể dùng trong cả hai chế độ Manual hoặc Teleoperation. Trước khi bắt đầu thao tác, người dùng nhấn nút Record trên GUI, một liên kết sẽ được thiết lập từ lớp Robot Run Command đến lớp Program, và tạo ra một chương trình mới. Khi người dùng thao tác, lớp Program lưu lại dữ liệu điều khiển theo thời gian, cùng lúc với việc truyền đi robot. Chương trình dừng lưu khi nhấn nút Stop Record. Đặc biệt, trong chế độ Teleoperation, lớp Program tự tạo chương trình mỗi khi Leap đọc được cử chỉ START, và dừng lưu khi Leap đọc được cử chỉ STOP.

Để lặp lại, người dùng chọn chương trình và nhấn nút Play trên GUI. Dữ liệu tọa độ theo thời gian được từ lớp Program đến lớp Robot Run Command. Từ đó các thao tác được thực hiện và phản hồi hoàn toàn tương tự như khi điều khiển trực tiếp.



Hình 6.6. Sơ đồ điều khiển tự động bằng chức năng Program

6.2.4. Điều khiển Robot mô phỏng



Hình 6.7. Sơ đồ chương trình mô phỏng robot

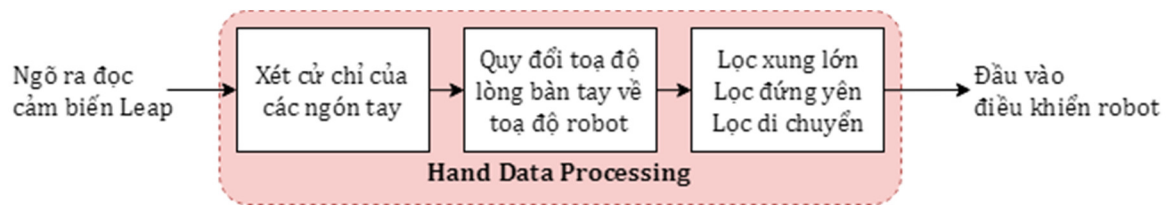
Mô phỏng là một chức năng tách biệt với các module chạy thực, có nhiệm vụ làm một robot ảo giúp người dùng thực tập, làm quen với các thao tác điều

khuyến. Trong chế độ mô phỏng, GUI cũng có hình thức và các tính năng tương tự chế độ chạy thực.

Người dùng nhập lệnh bằng bảng điều khiển trên GUI, đi vào lớp Robot Simulation. Tại đây, mô hình động học và hoạch định quỹ đạo với số liệu thực của robot được triển khai tính toán. Từ đó tìm được thông số vị trí khớp, tọa độ điểm cuối theo thời gian gần giống với hoạt động của robot thực. Dữ liệu phản hồi về GUI được hiển thị dưới dạng thông tin, đồ thị và đồ họa 3D.

6.3. Chương trình xử lý cảm biến Leap

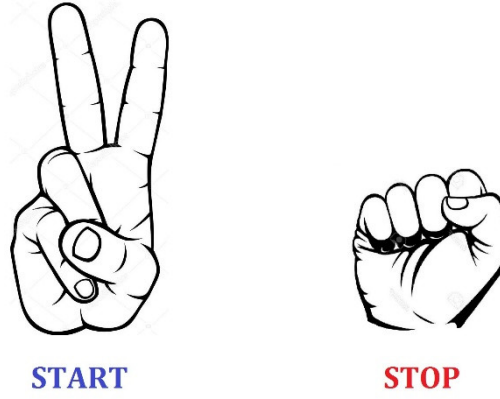
Module xử lý cảm biến Leap có mục tiêu là biến đổi dữ liệu ngõ ra của cảm biến Leap thành tín hiệu điều khiển thích hợp cho robot. Việc biến đổi gồm 3 công đoạn: xét cử chỉ, quy đổi tọa độ và lọc nhiễu.



Hình 6.8. Sơ đồ chương trình xử lý cảm biến Leap

6.3.1. Khởi xét cử chỉ

Ở công đoạn thứ nhất, dữ liệu các ngón tay được xem xét để phát hiện cử chỉ điều khiển. Khi phát hiện cử chỉ START (giơ ngón thứ hai và thứ ba) hoặc STOP (không giơ ngón nào) liên tiếp 10 lần, công đoạn này sẽ kích hoạt việc cho phép hoặc cấm việc thực hiện các công đoạn tiếp theo. Việc xét cử chỉ nhiều lần nhằm tăng sự chắc chắn, tránh nhầm lẫn với cử chỉ trong giai đoạn chuyển tiếp và các loại nhiễu.



Hình 6.9. Các cử chỉ ngón tay điều khiển chương trình

6.3.2. Khối quy đổi tọa độ

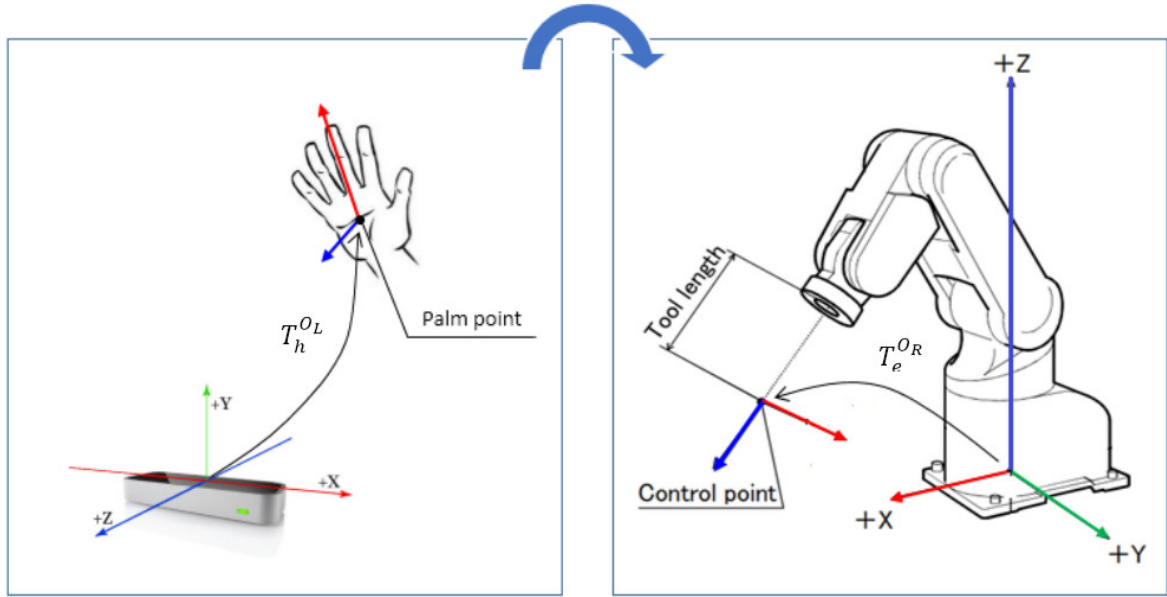
Nếu được cho phép, công đoạn thứ hai của Hand Data Processing sẽ tiến hành quy đổi dữ liệu của bàn tay thành dữ liệu đầu vào cho robot. Cụ thể, diễn ra việc quy đổi từ vị trí, hướng của lòng bàn tay trong hệ quy chiếu tọa độ của cảm biến Leap T_h^{OL} sang vị trí, hướng của điểm cuối trong hệ quy chiếu tọa độ của Robot T_e^{OR} . Việc quy đổi xuất phát từ khác biệt trong cách chọn hệ tọa độ giữa các nhà sản xuất, cũng như nhằm mục đích tối ưu tầm nhìn cho cảm biến Leap. Công thức quy đổi

$$T_e^{OR} = T_{OL}^{OR} T_h^{OL} T_e^h \quad (6.1)$$

với

$$T_{OL}^{OR} = \begin{bmatrix} 0 & 0 & -1 & 300 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -200 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_e^h = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Do ma trận T_e^{OR} chứa đến 12 thông số, để giảm dung lượng phải gửi qua đường truyền, ta áp dụng phần tìm đầu của lời giải bài toán động học ngược (xem chương 3) để rút gọn ma trận xoay R_e^{OR} gồm 9 thông số về 2 thông số góc xoay (θ, ψ) .



Hình 6.10. Quy đổi toạ độ tâm bàn tay về điểm cuối robot

Không chỉ xác định đầu vào cho điểm cuối của cánh tay robot, công đoạn này còn cho ra giá trị đặt cho góc kẹp của gripper. Tận dụng vector chỉ phương của các ngón tay, ta lấy góc hợp bởi hình chiếu của ngón tay thứ nhất (ngón cái) và thứ tư (ngón trỏ) lên mặt phẳng bàn tay làm giá trị góc kẹp.

$$j_g = \arccos \left(\frac{\vec{f}_{1c} \cdot \vec{f}_{4c}}{\|\vec{f}_{1c}\| \cdot \|\vec{f}_{4c}\|} \right) \quad (6.2)$$

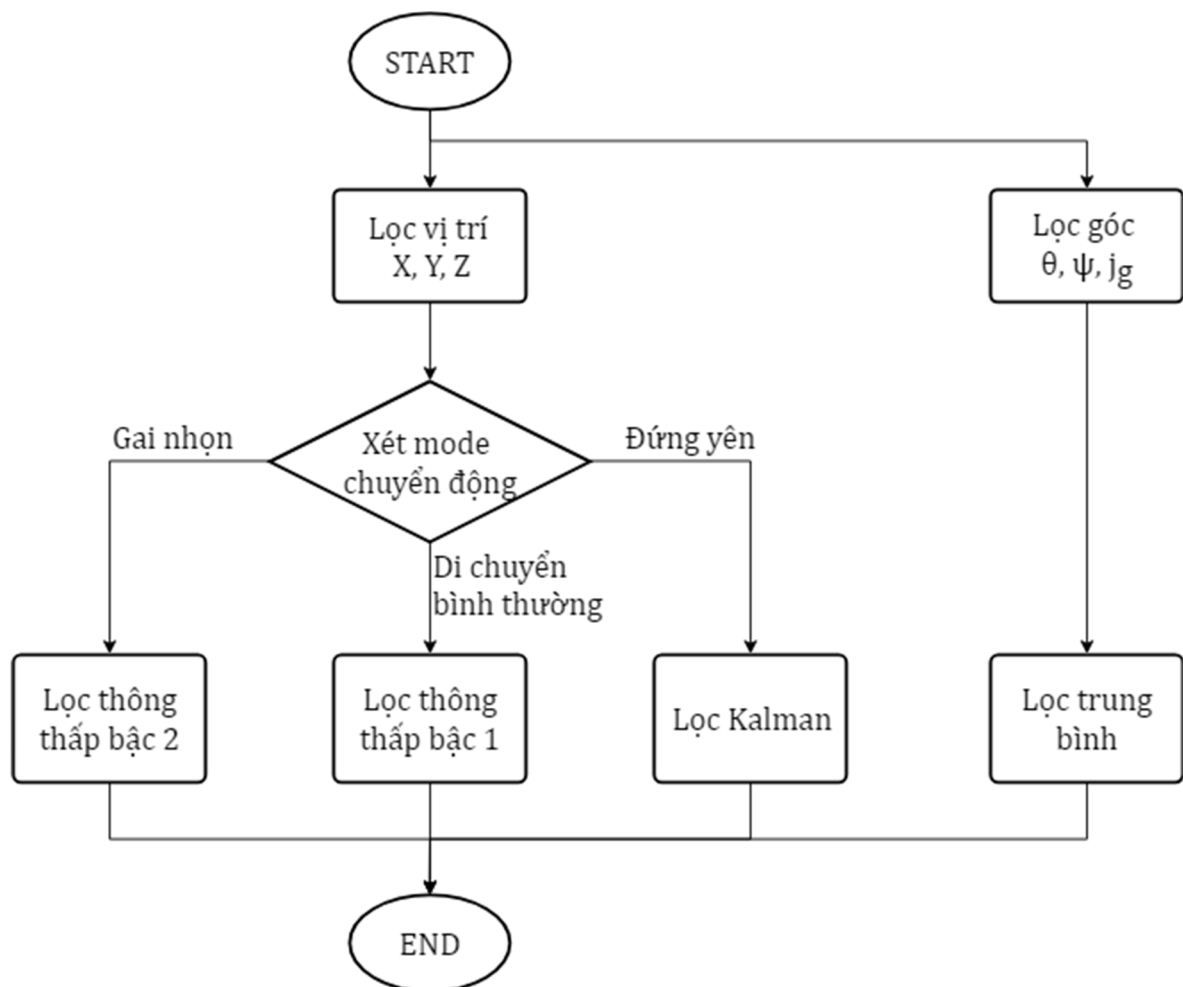


Hình 6.11. Cách xác định góc kẹp từ bàn tay

6.3.3. Lọc nhiễu

Công đoạn thứ ba của lớp Hand Data Process là lọc nhiễu. Công đoạn lọc nhằm giảm ảnh hưởng của sự run tay sinh học và các loại nhiễu.

Lúc này trục tọa độ được quy định theo robot. Đầu vào bộ lọc gồm 3 thông số vị trí (X, Y, Z) và 2 thông số góc hướng (θ, ψ) của điểm cuối robot, thêm vào đó là thông số góc kẹp (j_g) của gripper. Đối với vị trí (X, Y, Z), tác giả chọn phương án lọc theo 3 mode chuyển động, trong đó sử dụng bộ lọc thông thấp và bộ lọc Kalman. Đối với các góc (θ, ψ) và (j_g), áp dụng bộ lọc trung bình. Giải thuật như sau

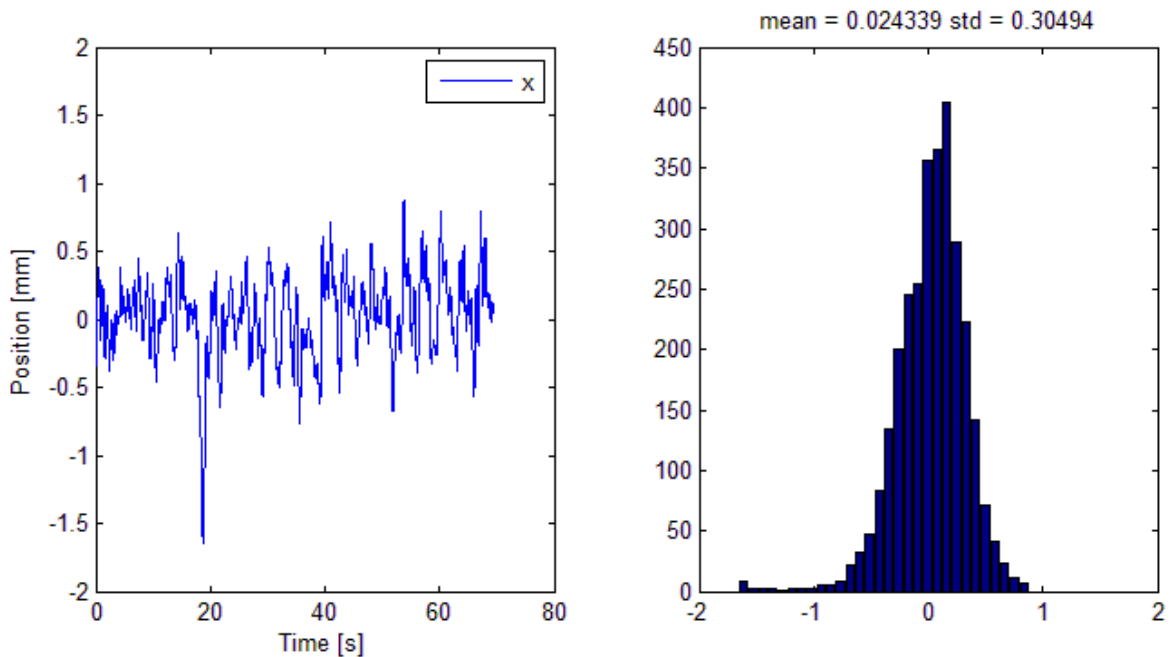


Hình 6.12. Giải thuật lọc nhiễu cho cảm biến Leap

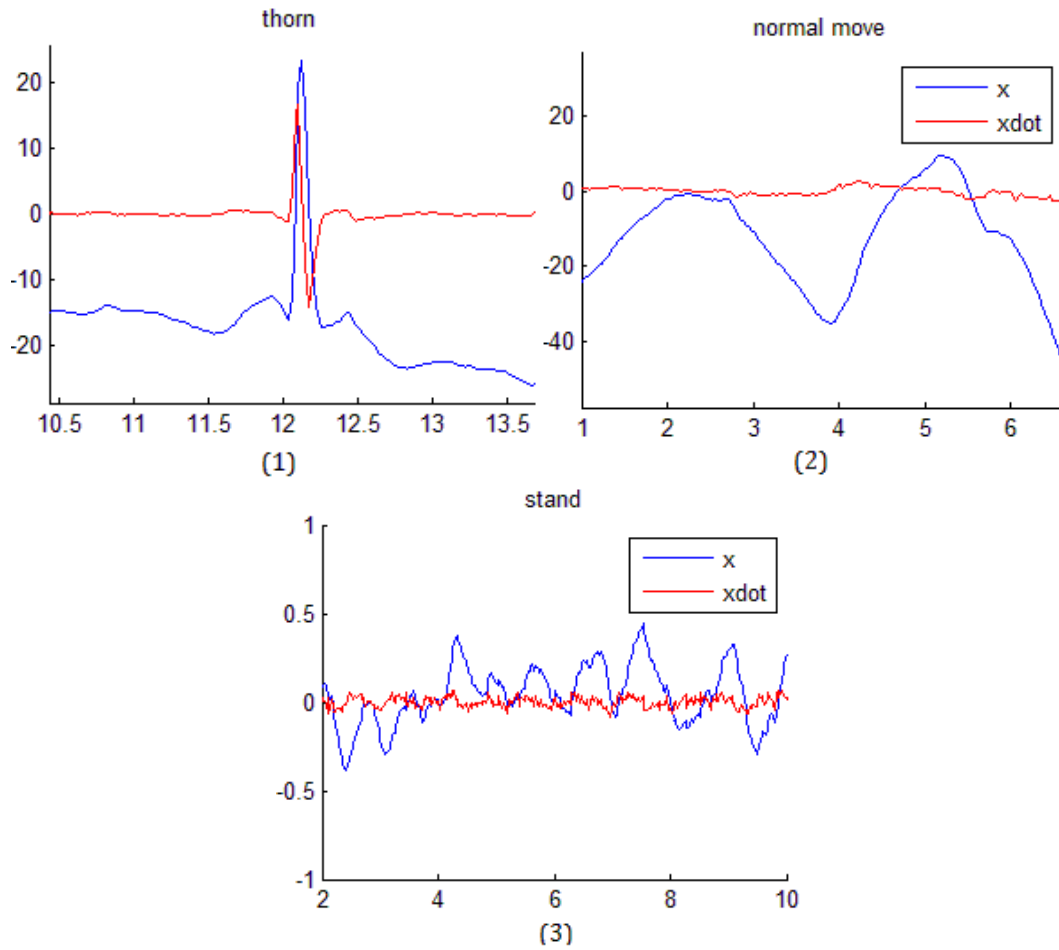
6.3.3.1. Bộ xét mode chuyển động

Bộ xét mode thực hiện phân loại chuyển động thành 3 mode: gai nhọn, di chuyển bình thường và đứng yên. Việc phân loại nhằm thoả mãn yêu cầu lọc riêng ở mỗi mode.

Cụ thể, với trường hợp có gai nhọn trong thời gian ngắn, chuyển động robot sẽ bị giật cục, kém mượt, do đó ta cần lọc mạnh bằng bộ lọc thông thấp bậc 2 để làm dịu gai. Với trường hợp di chuyển ở tốc độ bình thường, tay người có một ít độ run sinh học, nên ta chỉ lọc nhẹ bằng bộ lọc thông thấp bậc 1. Với trường hợp cố gắng giữ yên, tay người sẽ có độ run nhiều hơn và độ run lúc này tuân theo phân bố Gauss, vì vậy ta áp dụng bộ lọc Kalman.

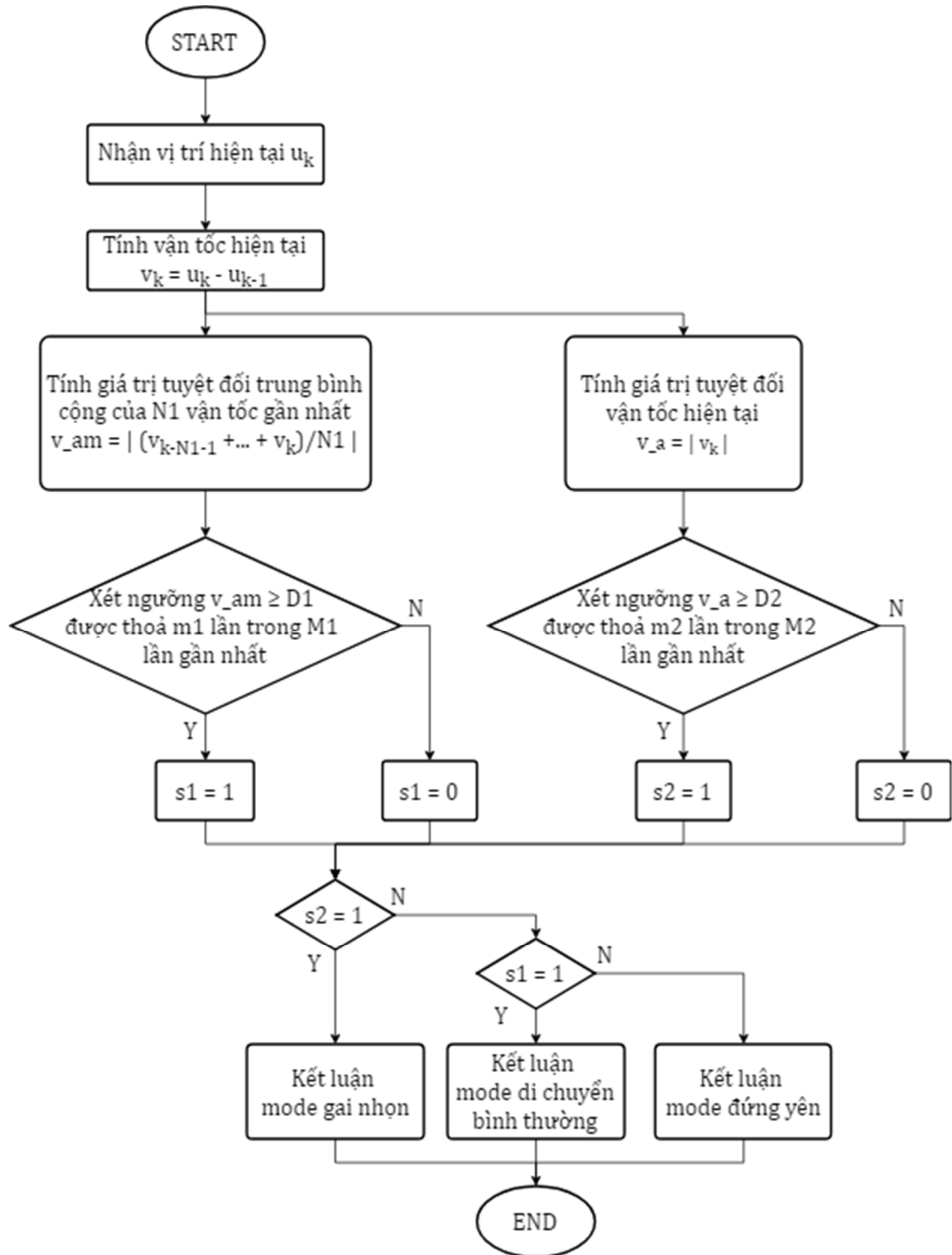


Hình 6.13. Phân bố vị trí tay khi giữ yên tại $x = 0$



Hình 6.14. Các mode chuyển động được phân loại
(1) – gai nhọn, (2) – di chuyển bình thường, (3) – đứng yên

Bộ xét mode sử dụng thuật toán xét ngưỡng vận tốc. Trong đó, các thông N_1, D_1, M_1, D_2, M_2 sẽ được chọn bằng phương pháp thử và sai.



Hình 6.15. Thuật toán xét mode di chuyển

6.3.3.2. Bộ lọc thông thấp

Bộ lọc thông thấp bậc 1 rời rạc có công thức

$$f(k) = K_1 f(k-1) + (1 - K_1)u(k) \quad (6.3)$$

trong đó u là ngõ vào, f là ngõ ra, $0 \leq K_1 \leq 1$ là độ lợi của bộ lọc.

Bộ lọc thông thấp bậc 2 rời rạc với hai cực trùng nhau là

$$f(k) = (K_2)^2 f(k-2) + 2K_2(1 - K_2)f(k-1) + (1 - K_2)^2 u(k) \quad (6.4)$$

trong đó u là ngõ vào, f là ngõ ra, $0 \leq K_2 \leq 1$ là độ lợi của bộ lọc.

K_1, K_2 càng lớn, tần số cắt của các bộ lọc càng nhỏ, ngõ ra càng trễ so với ngõ vào.

6.3.3.3. Bộ lọc Kalman

Khai triển bộ lọc Kalman rời rạc cho giá trị đo đạc như sau

Dự đoán

$$\bar{f}_k = f_{k+1} \quad (6.5)$$

$$\bar{P}_k = P_{k-1} + Q \quad (6.6)$$

Cập nhật

$$K_k = \frac{\bar{P}_k}{\bar{P}_k + R} \quad (6.7)$$

$$f_k = \bar{f}_k + K_k(u_k - \bar{f}_k) \quad (6.8)$$

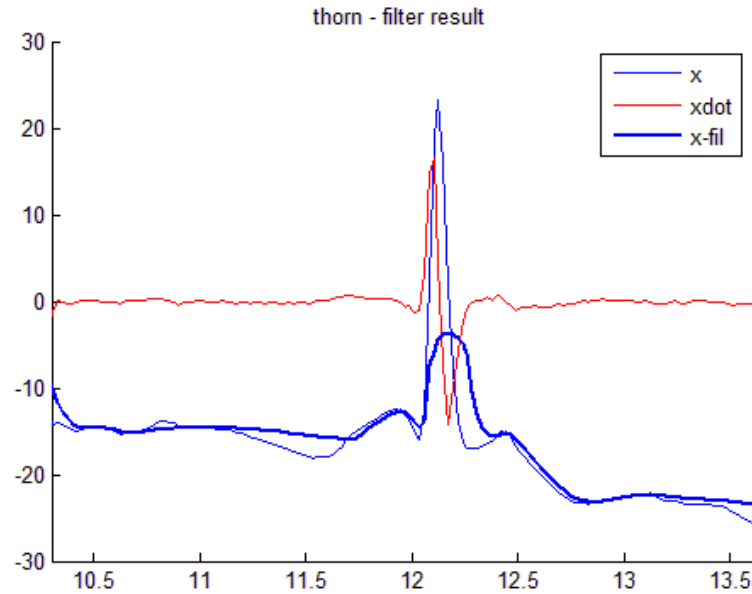
$$P_k = (1 - K_k)\bar{P}_k \quad (6.9)$$

trong đó u_k là ngõ vào, f_k là ngõ ra bộ lọc; Q, R là các ma trận hiệp phương sai nhiễu hệ thống và nhiễu đo đạc, P là ma trận hiệp phương sai sai số, $0 \leq K \leq 1$ là độ lợi của bộ lọc.

6.3.3.4. Kết quả lọc

- Trường hợp gai nhọn

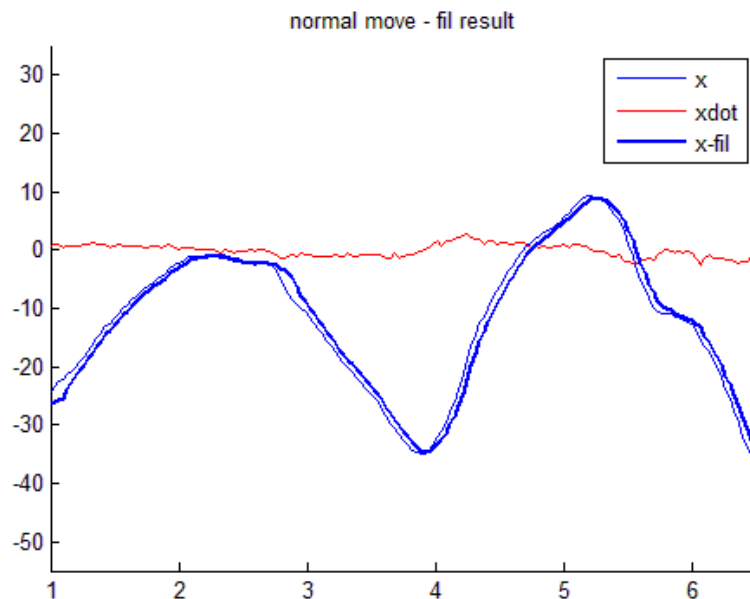
Gai nhọn đã được làm dịu xuống, đảm bảo khả năng đáp ứng của robot.



Hình 6.16. Kết quả lọc nhiễu khi có gai nhọn

- Trường hợp di chuyển bình thường

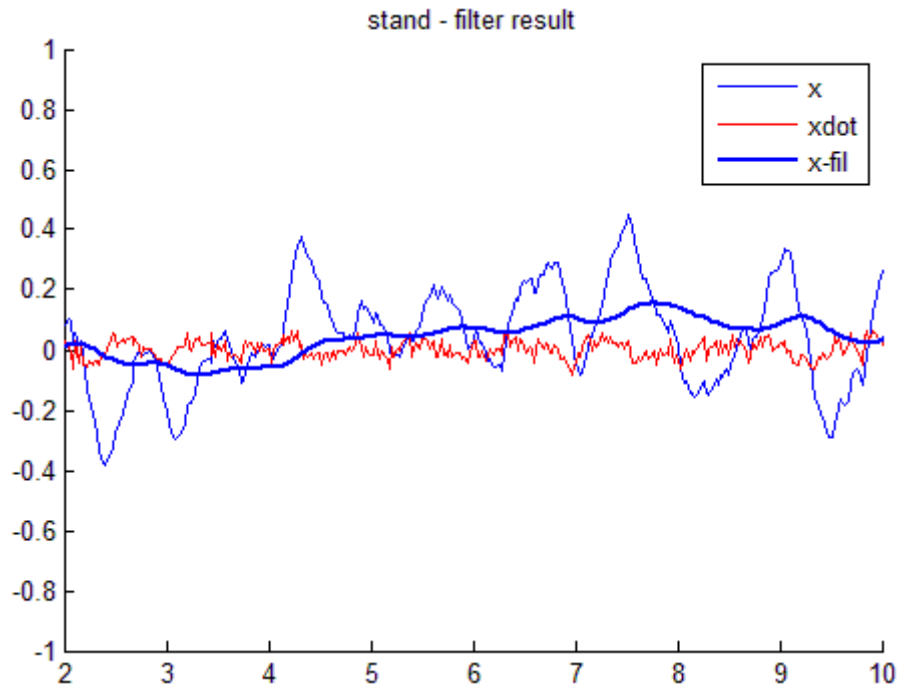
Tín hiệu sau khi lọc có phần mượt hơn và có độ trễ không đáng kể so với tín hiệu vào.



Hình 6.17. Kết quả lọc nhiễu khi di chuyển thông thường

- **Trường hợp đứng yên**

Biên độ run của tay được giảm đáng kể sau khi qua bộ lọc Kalman.

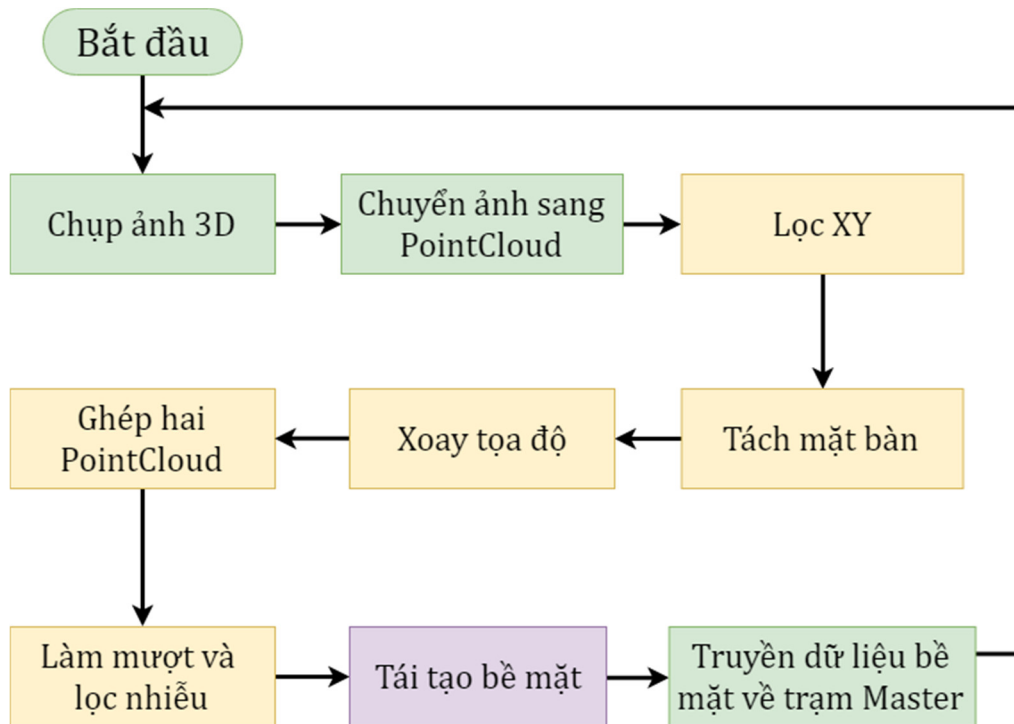


Hình 6.18. Kết quả lọc khi nhiều đứng yên

6.4. Chương trình xử lý cảm biến Kinect

6.4.1. Lưu đồ giải thuật

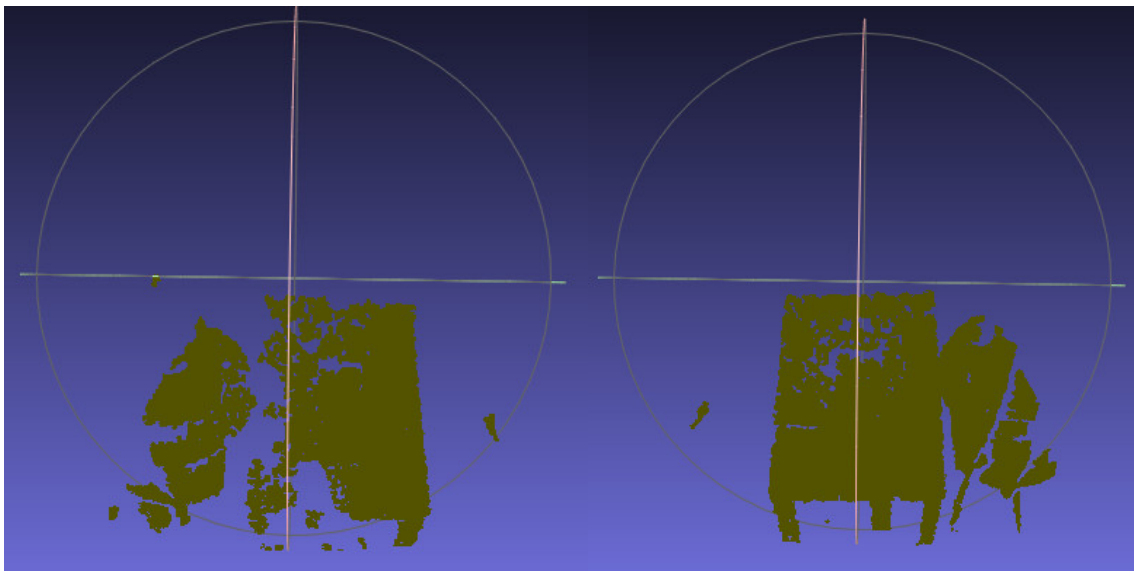
Chương trình Kinect nằm trên máy tính Kinect, thực hiện nhiệm vụ thu thập dữ liệu hình ảnh 3D từ hai camera Kinect, xử lý và tái tạo bề mặt sau đó truyền về trạm Master. Chương trình Kinect thực hiện theo sơ đồ giải thuật sau:



Hình 6.19. Giải thuật chương trình Kinect

6.4.2. Chức năng từng khối

6.4.2.1. Khối Chụp ảnh 3D



Hình 6.20. Hình ảnh thô chụp từ hai Kinect

Hai camera Kinect sẽ chụp ảnh Khu vực làm việc của tay máy. Ảnh này được thư viện Nestk hỗ trợ lấy dữ liệu và lưu lại với định dạng RGBDImage.

Trước khi bước vào giai đoạn xử lý ảnh 3D với thư viện PCL, ta phải tiến hành chuyển đổi định dạng ảnh sang định dạng của PCL là PointCloud.

6.4.2.2. Khối Chuyển ảnh sang PointCloud

Khối này thực hiện nhiệm vụ chuyển đổi định dạng dữ liệu RGBDImage sang định dạng PointCloud để có thể sử dụng thư viện PCL xử lý.

6.4.2.3. Khối Lọc XY

Khối Lọc XY thực hiện nhiệm vụ loại bỏ các đối tượng thừa không nằm trong khu vực làm việc của tay máy.

Bộ lọc loại bỏ các điểm không xác định và các điểm nằm ngoài vùng giới hạn được định nghĩa bởi hàm *setFiltersLimits()*, và chỉ áp dụng lọc đối với miền của trường dữ liệu được xác định bởi hàm *setFilterFieldName()* được định nghĩa trong lớp *pcl::PassThrough*.

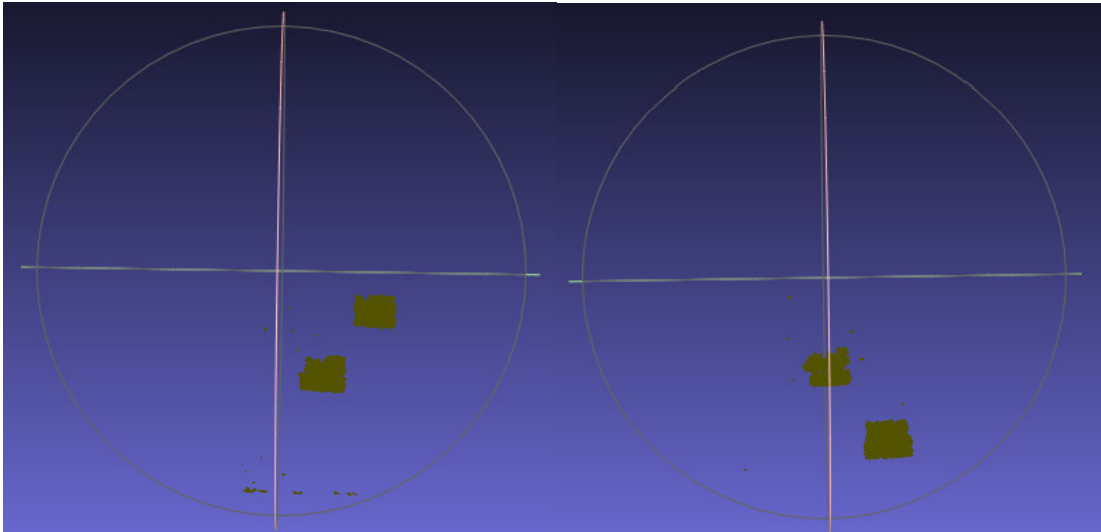
Tác dụng của Bộ lọc XY là rất cần thiết vì giúp giảm một số lượng đáng kể các điểm nằm ngoài khu vực làm việc của tay máy, đảm bảo chỉ các đối tượng cần tái tạo mô hình 3D được đưa vào xử lý.

6.4.2.4. Khối Tách mặt bàn

Dữ liệu PointCloud sau khi lọc XY vẫn còn một phần lớn điểm không cần thiết, đó là các điểm thuộc mặt bàn.

Khối Tách mặt bàn sử dụng thuật toán RANSAC (RANdom SAmple Consensus) để dò tìm mặt phẳng và Phân đoạn để có thể loại bỏ các điểm thuộc mặt bàn ra khỏi dữ liệu PointCloud. Thuật toán này thực hiện bằng cách lựa chọn ba điểm bất kỳ để thiết lập một mặt phẳng; xét toàn bộ khoảng cách từ các điểm còn lại trong PointCloud đến mặt phẳng, nếu thỏa điều kiện bé hơn một khoảng d_t cho trước thì xếp vào cùng một *inlier* (đường bao); sau cùng sẽ loại bỏ mặt

phẳng có *inlier* lớn nhất. Các hàm thực hiện thuật toán này nằm trong lớp *pcl::SACSegmentation* và lớp *pcl::ExtractIndices*.



Hình 6.21. Dữ liệu vật đã được tách khỏi mặt bàn

6.4.2.5. Khối Xoay tọa độ

Khối xoay tọa độ thực hiện nhiệm vụ xoay gốc tọa độ của hai PointCloud về gốc tọa độ World của tay máy RV-2AJ.

Các ma trận xoay được xác định theo các bước sau:

- Đánh dấu các điểm tham chiếu trên bàn làm việc của tay máy (tối thiểu 4 điểm, lấy nhiều điểm hơn để tăng độ chính xác).
- Xác định tọa độ XYZ của từng điểm tham chiếu trong hệ tọa độ của hai camera Kinect và hệ tọa độ World của tay máy.
- Sử dụng hàm *absor()* trong MATLAB để tính ma trận xoay từ tọa độ của các điểm tham chiếu.

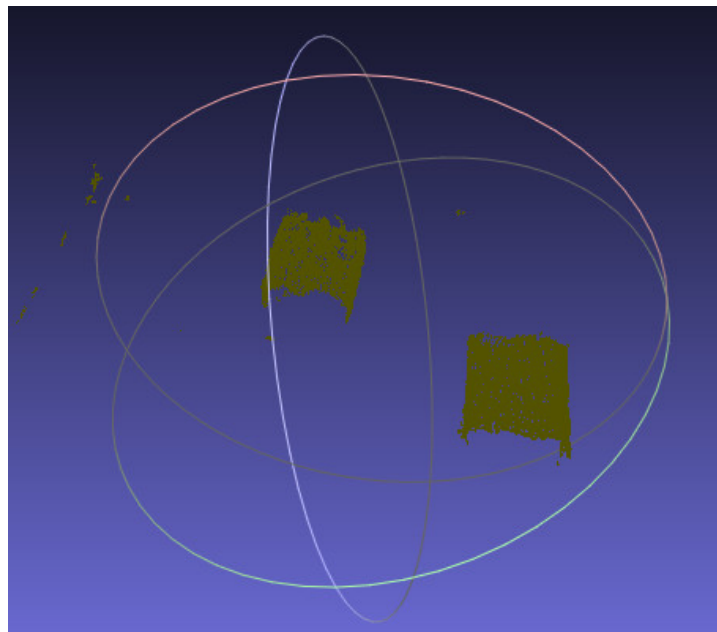
Các PointCloud sẽ được xoay theo các ma trận tương ứng bằng hàm *transformPointCloud()* thuộc lớp PointT theo trình tự Kinect 1 → Kinect 2 và Kinect 2 → World.

6.4.2.6. Khối Ghép hai PointCloud

Khối Ghép hai PointCloud thực hiện nhiệm vụ ghép hai PointCloud sau khi đã xoay thành một PointCloud thống nhất.

Phương pháp được sử dụng là cộng theo pointer của các PointCloud.

Phương pháp ICP không được sử dụng vì mặc dù có độ chính xác cao nhưng thời gian thực hiện dài, không phù hợp với mục đích theo dõi của đề tài. Hơn nữa dữ liệu PointCloud sau khi xử lý xoay tọa độ đã rất khớp với nhau nên chỉ cần cộng lại là đạt yêu cầu.

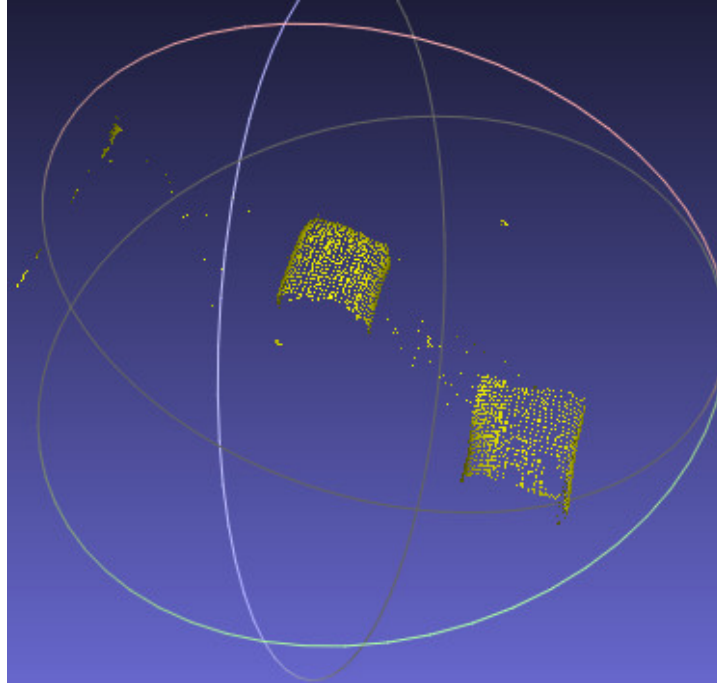


Hình 6.22. Dữ liệu hai PointCloud đã được ghép và xoay về gốc tọa độ tay máy

6.4.2.7. Khối Làm mượt và lọc nhiễu

Khối Làm mượt và lọc nhiễu thực hiện nhiệm vụ xây dựng lại một PointCloud liên tục và lọc nhiễu để chuẩn bị cho bước Tái tạo bề mặt.

Phương pháp được sử dụng là Moving Least Square. Đây là một phương pháp xây dựng lại các hàm liên tục từ một tập hợp các mẫu điểm chưa được tổ chức thông qua việc tính toán một phương pháp bình phương tối thiểu có trọng số đối với vùng xung quanh điểm mà tại đó giá trị cần xây lại được yêu cầu.



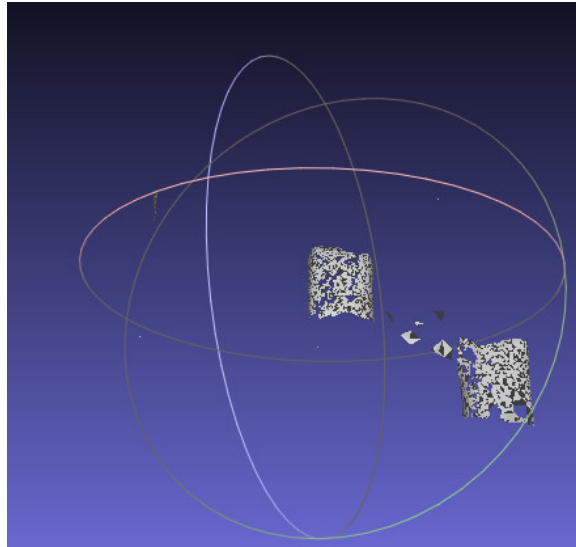
Hình 6.23. Dữ liệu sau khi được xử lý bằng phương pháp Moving Least Square

6.4.2.8. Khối Tái tạo bề mặt

Khối Tái tạo bề mặt thực hiện nhiệm vụ tái tạo (Reconstruction) mô hình vật trên bàn làm việc.

Các PointCloud sau quá trình xử lý sẽ được tái tạo bề mặt bằng phương pháp Chiếu tam giác hóa (Greedy projection triangulation). Đây là dạng thuật toán tái tạo bề mặt luôn luôn bảo toàn đám mây điểm đầu vào tạo lưới đa giác để liên kết các điểm. Lưới đa giác này có cùng số điểm với đám mây điểm đầu vào. Cùng dạng này còn có các thuật toán Concave Hull, Convex Hull, Organized Fast Mesh. Ưu điểm của các thuật toán dạng này là tốc độ xử lý nhanh, phù hợp với yêu cầu đề tài.

Ngoài ra còn có các thuật toán tạo bề mặt mới hoặc lớp điểm mới ở vị trí khác với vị trí các điểm trong PointCloud đầu vào như Grid Projection, Marching Cubes, Surfel Smoothing. Tuy nhiên, các phương pháp này không được sử dụng vì đòi hỏi thời gian xử lý dài, không phù hợp với yêu cầu về tốc độ của đề tài.



Hình 6.24. Kết quả tái tạo bề mặt

6.4.2.9. Khối Truyền dữ liệu bề mặt về trạm Master

Khối này thực hiện nhiệm vụ truyền dữ liệu bề mặt đã được tái tạo về trạm Master.

6.5. Mạng giao tiếp UDP

UDP (User Datagram Protocol) là một giao thức cho phép truyền dữ liệu trong mạng Ethernet mà không cần thiết lập kết nối hay xác nhận trở lại. Nhờ đó UDP có tốc độ nhanh hơn, gói tin ngắn hơn so với giao thức TCP. Đề tài chọn giao thức UDP nhằm giảm thời gian trễ do truyền dữ liệu giữa các máy tính. Mặc dù UDP có nhược điểm là không báo lỗi khi mất đường truyền, việc đó lại rất ít khi xảy ra đối với mạng nội bộ (LAN) mà đề tài sử dụng.

+	Bits 0 - 15	16 - 31
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

Hình 6.25. Cấu trúc gói tin của giao thức UDP

Để phục vụ giao tiếp điều khiển giữa người dùng và robot, một bộ mã lệnh được xây dựng. Các lớp UDPSocket ở hai đầu chịu trách nhiệm mã hoá và giải mã các yêu cầu hoặc phản hồi từ các lớp vận hành theo bộ mã lệnh này.

Chiều đặt (Master -> Slave)	Chiều phản hồi (Slave -> Master)
<Lệnh>;<Giá trị đặt>\r	<Kết quả>;<Lệnh>;<Giá trị phản hồi>\r
Ý nghĩa	
<Lệnh> là lệnh cần thực thi <Giá trị đặt> là thông số yêu cầu \r là kí tự kết thúc gói tin	<Kết quả> là kết quả thực hiện một lệnh, (thành công: OK, có lỗi: ER) <Lệnh> là lệnh đã thực hiện <Giá trị phản hồi> là thông tin phản hồi đối với lệnh \r là kí tự kết thúc gói tin

Bảng 6.1. Cấu trúc mã lệnh giao tiếp giữa Master Client và Robot Server

Lệnh	Nội dung yêu cầu	Nội dung phản hồi
CONNECT/ DISCONNECT	Kết nối/ ngắt kết nối giữa hai phía	Thành công/ lỗi
ROBCON	Kết nối với controller của robot	Thành công/ lỗi
EMGSTOP	Dừng khẩn cấp robot	Thành công/ lỗi
SRVON/ SRVOFF	Bật/ tắt động cơ robot	Thành công/ lỗi
RSTALRM	Reset lỗi trên robot	Thành công/ lỗi
TELESTART / TELESTOP	Bắt đầu/ kết thúc chế độ Teleoperation	Thành công/ lỗi
SETX;<X>,<Y>,<Z>,<T>,<K>,<JG> ;<time>	Di chuyển robot đến vị trí đặt của điểm cuối.	Thành công/ lỗi
SETJ;<J1>,<J2>,<J3>,<J4>,<J5>,<J6> ;<time>	Di chuyển robot đến vị trí đặt của các khớp	Thành công/ lỗi
JOGX;<index>,<step>	Di chuyển robot một khoảng cách theo một trục toạ độ	Thành công/ lỗi
JOGJ;<index>,<step>	Di chuyển robot một khoảng cách theo một khớp	Thành công/ lỗi

SETS;<V>	Đặt vận tốc robot	Thành công/ lỗi
SETA;<A>,<D>	Đặt gia tốc, hãm tốc robot	Thành công/ lỗi
SETC;<C>	Đặt cấu hình tay robot	Thành công/ lỗi
SETGF;<GF>	Đặt lực ngưỡng của gripper	Thành công/ lỗi
GETJX	Lấy giá trị góc khớp và tọa độ điểm cuối hiện tại của robot	Thành công/ lỗi Các thông số như yêu cầu
GETGF	Lấy giá trị lực kẹp hiện tại của gripper	Thành công/ lỗi Các thông số như yêu cầu

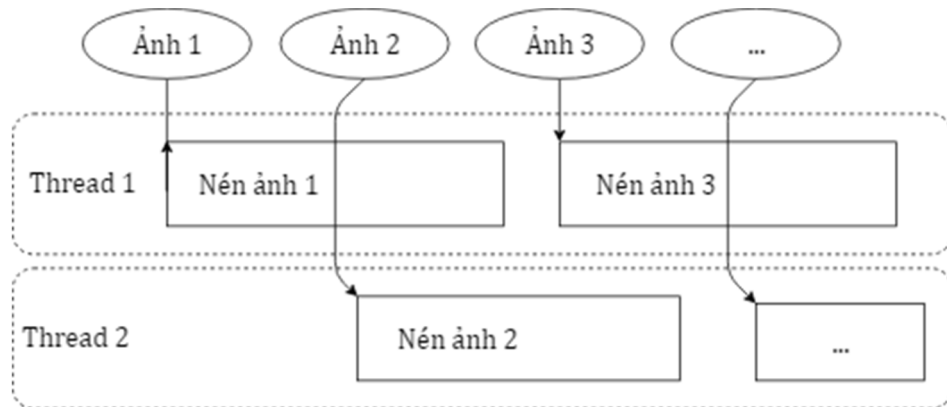
Bảng 6.2. Tập mã lệnh giao tiếp giữa Master Client và Robot Server

6.6. Đọc Camera Tool

Khi robot vận hành, Camera Tool ghi lại hoạt động của gripper gần tại điểm cuối robot. Mỗi chu kì, chương trình tại Robot Server lấy ảnh chụp từ camera. Ảnh sẽ được nén theo chuẩn JPEG để giảm dung lượng, rồi gửi dưới dạng chuỗi byte về Master Client. Khi chuỗi byte đến, Master Client lưu thành dạng ảnh, giải nén, rồi hiển thị lên màn hình.

Cú pháp gửi ảnh trên đường truyền là: IMG<ID><ByteArray>. Trong đó IMG là kí tự bắt đầu ảnh, <ID> là số thứ tự ảnh theo thời gian chụp, <ByteArray> là chuỗi dữ liệu chứa ảnh.

Thời gian cần để nén một ảnh có thể dài hơn chu kì chụp một ảnh, nhất là với ảnh kích thước lớn. Dẫn đến tích lũy độ trễ của ảnh gửi về Master so với thời gian thực. Để giải quyết việc này, công đoạn nén ảnh được thực hiện bằng cơ chế đa luồng (multithread). Nhờ các thread chạy song song, mỗi ảnh chụp sẽ được nén ngay lập tức mà không phải chờ các ảnh trước được nén xong. Sau khi hoàn tất nén một ảnh, thread sẽ được tái sử dụng cho các ảnh tiếp theo.

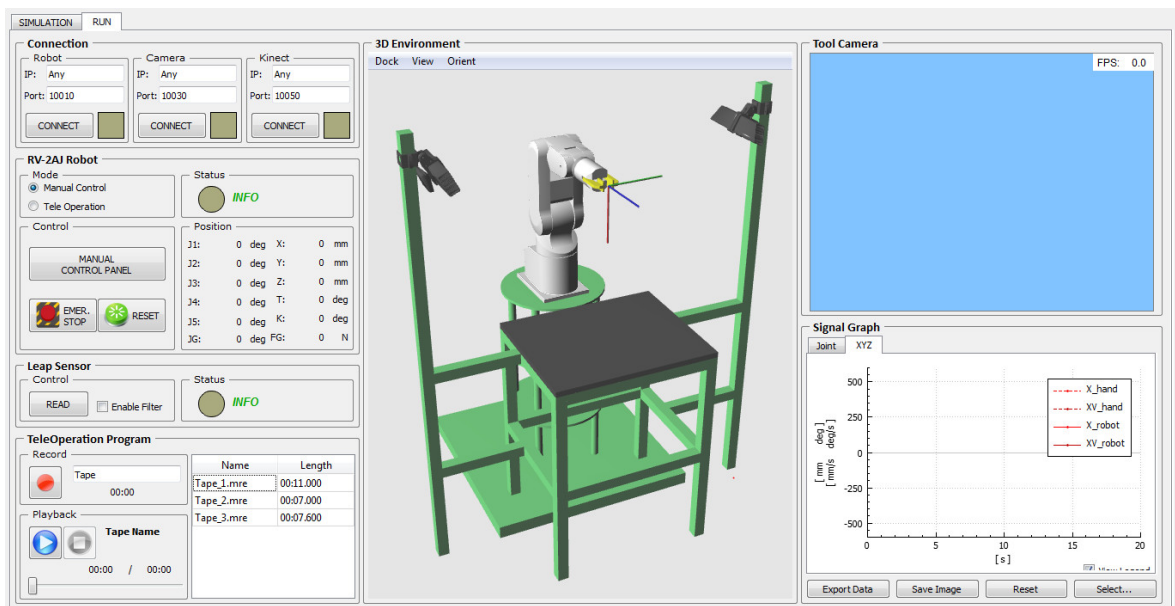


Hình 6.26. Nén ảnh dùng cơ chế đa luồng

6.7. Giao diện người dùng

6.7.1. Chế độ RUN

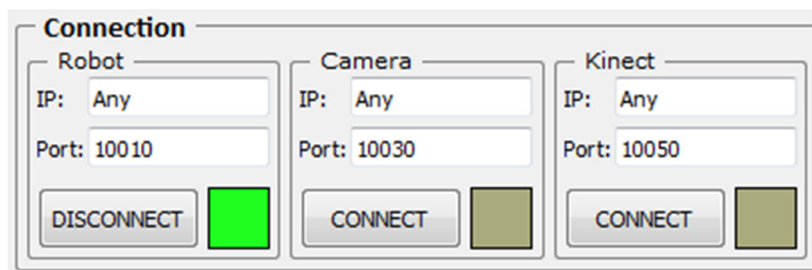
Giao diện trong chế độ RUN chứa tất cả tính năng để chạy hệ thống thực.



Hình 6.27. Giao diện chính của chế độ RUN

6.7.1.1. Khối Connection

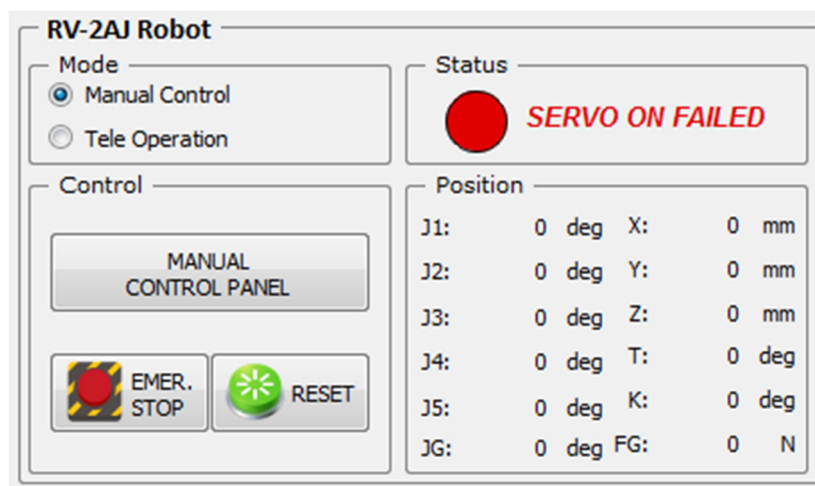
Khối Connection cho phép thiết lập và xem trạng thái kết nối mạng UDP đến Robot, Camera và Kinect.



Hình 6.28. Giao diện khối Connection

6.7.1.2. Khối Robot

Khối Robot gồm nhóm Mode để chọn giữa chế độ Manual và Teleoperation, nhóm Control để ra lệnh điều khiển robot, nhóm Status hiển thị trạng thái hoạt động và báo lỗi, nhóm Position hiển thị giá trị góc khớp và vị trí điểm cuối hiện tại của robot.



Hình 6.29. Giao diện khối Robot trong chế độ Manual

Với chế độ Manual, bảng điều khiển được dùng để nhập lệnh. Với chế độ Teleoperation, lệnh di chuyển sẽ được nhập thông qua Khối Leap Sensor. Mặt khác, nút dừng khẩn cấp (EMER. STOP) luôn có tác dụng trong cả hai chế độ.

Jog Mode

☒ JOINT Jog
☐ XYZ Jog

Control

SERVO: ☐ OFF

Set Configuration

☒ Auto/ Manual
☐ Right ☐ Left
☒ Above ☐ Below

SET

Set Gripper Force

FG N

SET

Jog Operator

J1

J2

J3

J4

J5

J6

Step 10.00

Set Position

J1 deg
J2 deg
J3 deg
J4 deg
J5 deg
J6 deg

SET

Set Speed, Acceleration

Joints Speed (%) ☒ Auto/ Manual

Accel (%) 50 Decel (%) 50

SET

Get Data

Robot Position:

GET

Hình 6.30. Giao diện bảng điều khiển Manual

RV-2AJ Robot

Mode

☐ Manual Control
☒ Tele Operation

Status

TELEOPERATION START FAILED

Control

TELEOPERATION START

EMER. STOP RESET

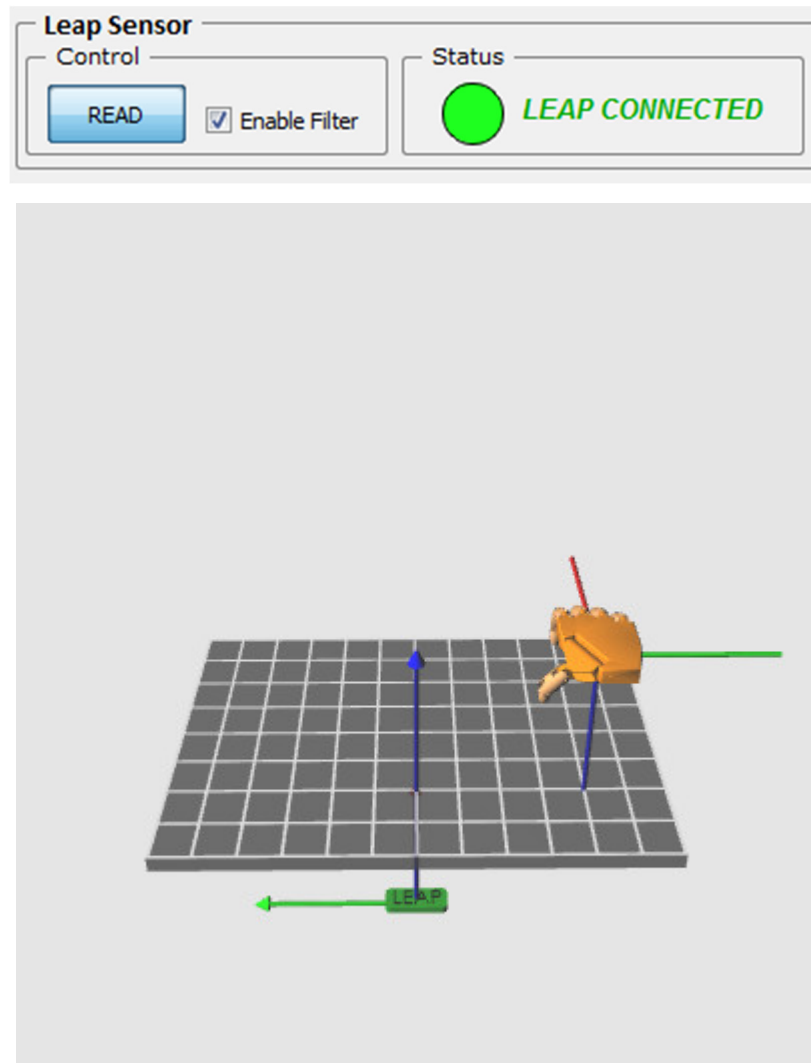
Position

J1: 0 deg X: 0 mm
J2: 0 deg Y: 0 mm
J3: 0 deg Z: 0 mm
J4: 0 deg T: 0 deg
J5: 0 deg K: 0 deg
J6: 0 deg FG: 0 N

Hình 6.31. Giao diện khối Robot trong chế độ Teloperation

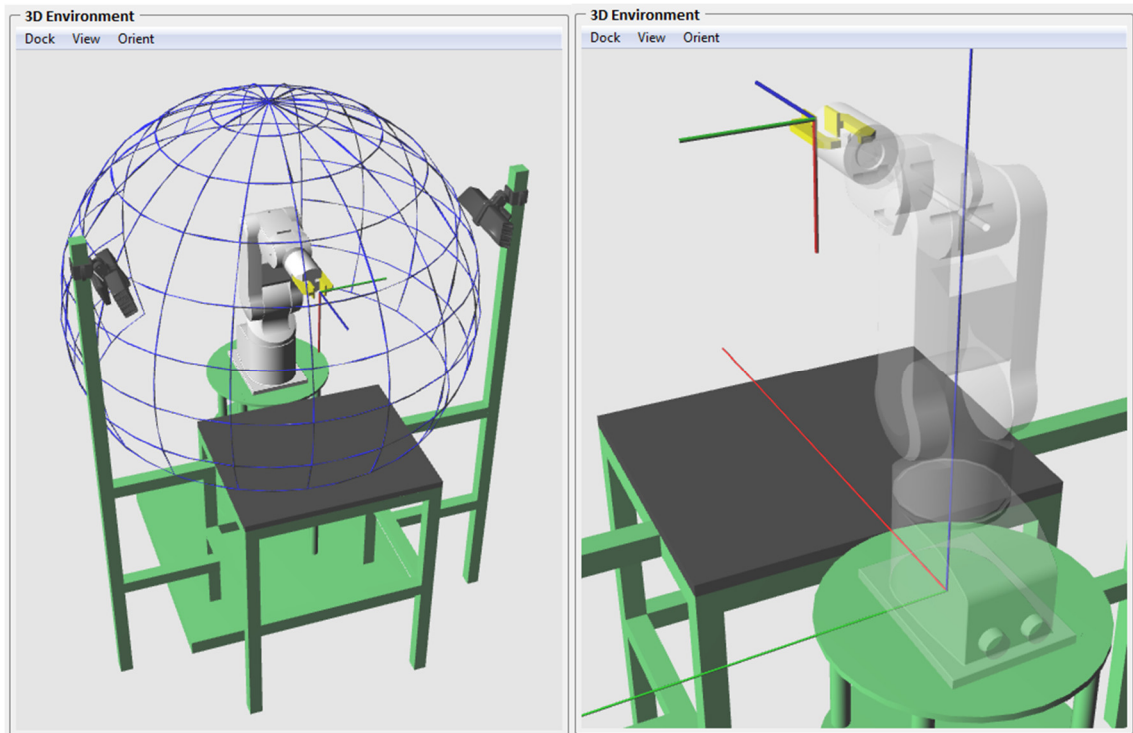
6.7.1.3. Khối Leap Sensor

Khối Leap bao gồm nhóm Control và nhóm Status. Khi nhấn READ, chương trình bắt đầu lấy dữ liệu từ cảm biến Leap, một cửa sổ đồ họa được bật lên để theo dõi cử động bàn tay.



Hình 6.32. Giao diện cảm biến Leap

6.7.1.4. Khối 3D Environment

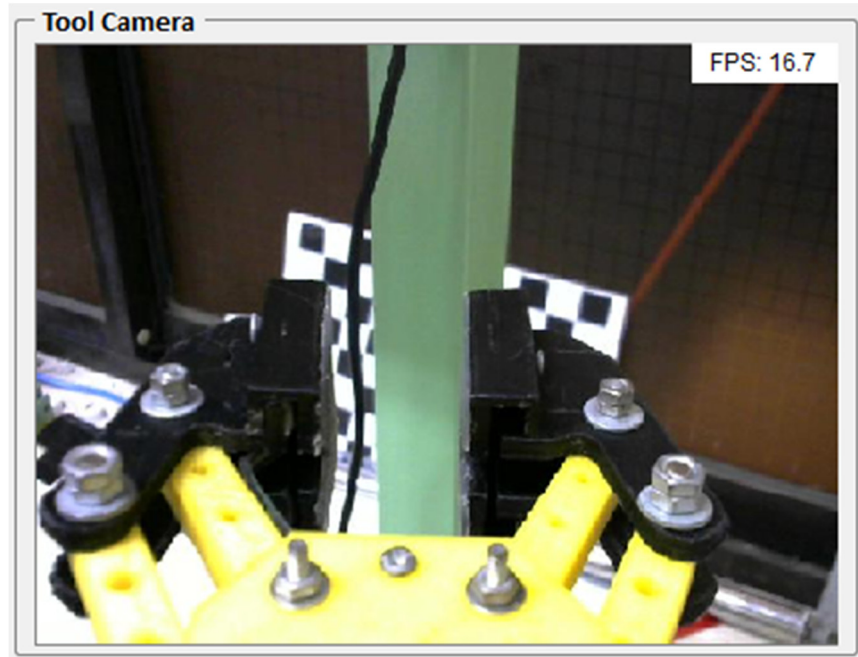


Hình 6.33. Giao diện đồ họa môi trường 3D

Khối 3D Environment thể hiện vị trí robot trong không gian thực tế ảo 3D. Ngoài ra, môi trường đồ họa có thể cho thấy giới hạn làm việc của cánh tay, hệ tọa độ tại gốc và tại gripper. Có thể chỉnh hiển thị tay máy thành trong suốt để quan sát rõ hơn bàn làm việc.

6.7.1.5. Khối Tool Camera

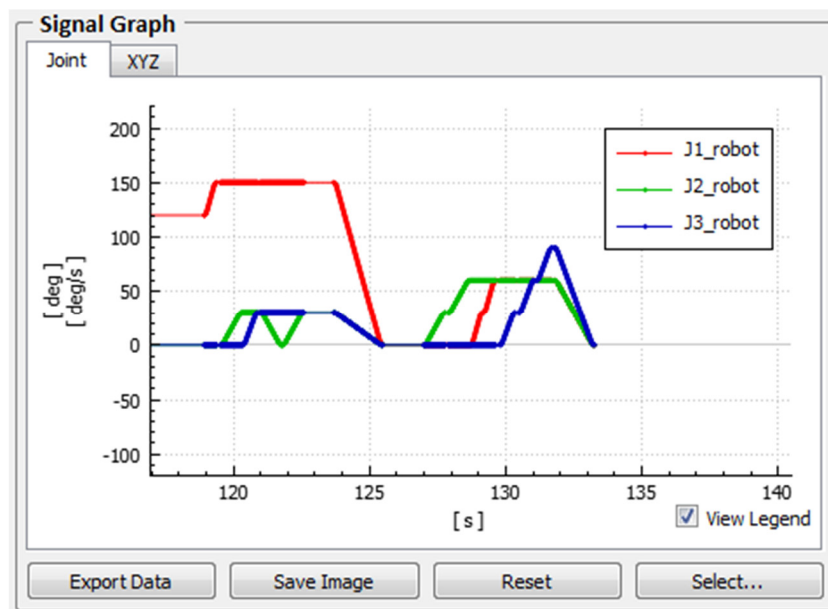
Khối Tool Camera hiển thị hình ảnh từ camera quan sát Gripper.



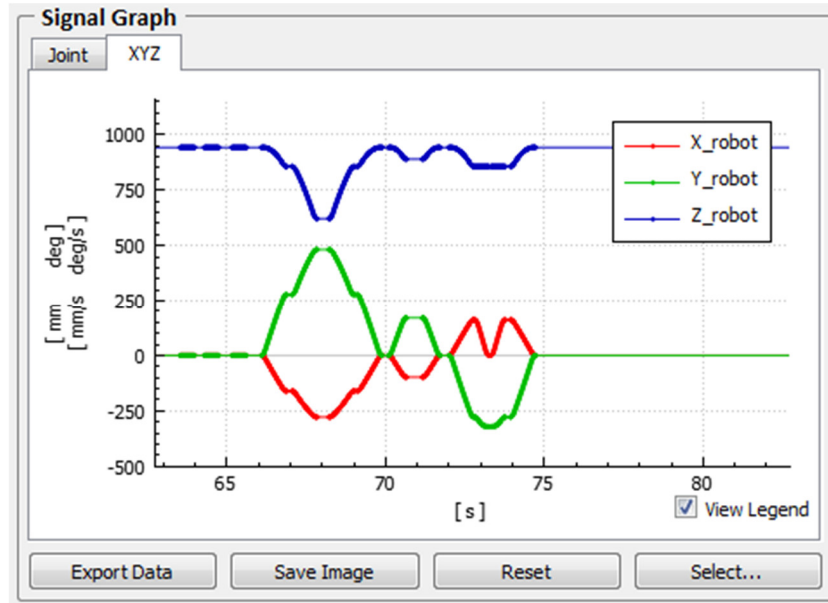
Hình 6.34. Giao diện hình ảnh Camera Tool

6.7.1.6. Khối Graph

Khối Graph làm nhiệm vụ vẽ đồ thị các tín hiệu đặt từ bàn tay người dùng và tín hiệu phản hồi từ robot trong miền khớp hoặc miền không gian.



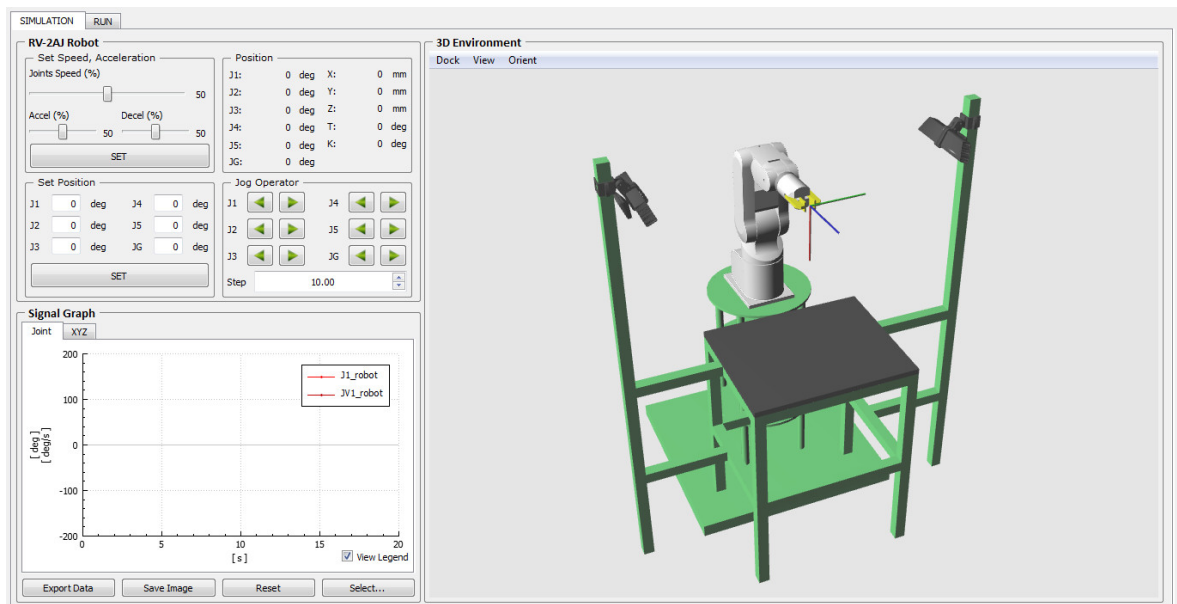
Hình 6.35. Giao diện đồ thị miền khớp



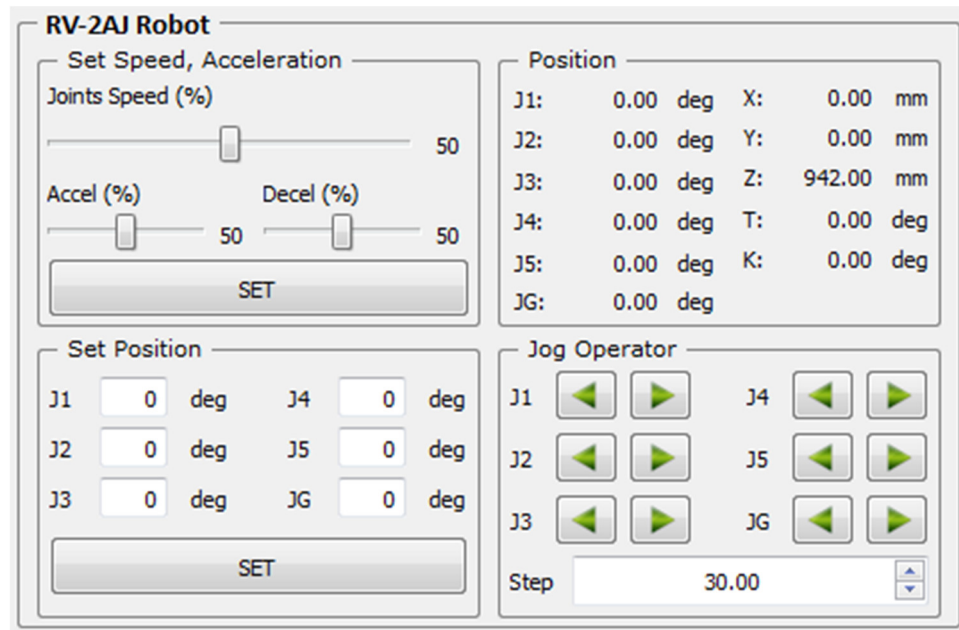
Hình 6.36. Giao diện đồ thị miền XYZ

6.7.2. Chế độ SIMULATION

Giao diện trong chế độ SIMULATION cho phép người dùng thực tập trên robot mô phỏng với các tính năng hoạt động giống như robot thực.



Hình 6.37. Giao diện chính của chế độ SIMULATION



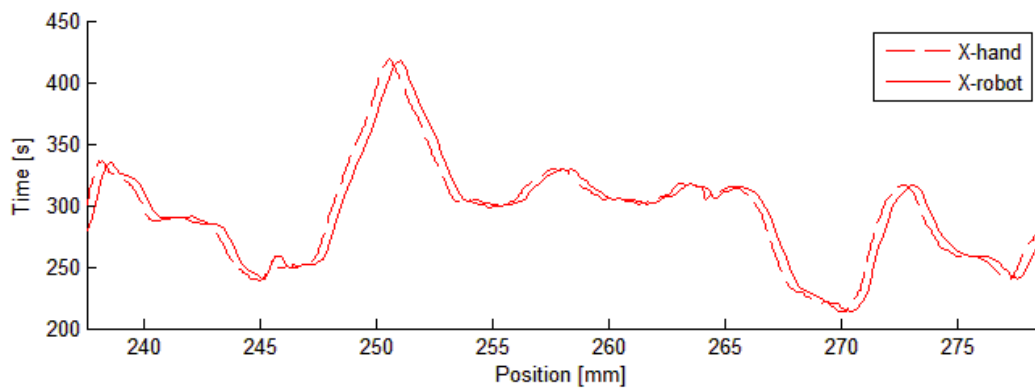
Hình 6.38. Giao diện bảng điều khiển trong chế độ SIMULATION

Chương 7. KẾT QUẢ VÀ KẾT LUẬN

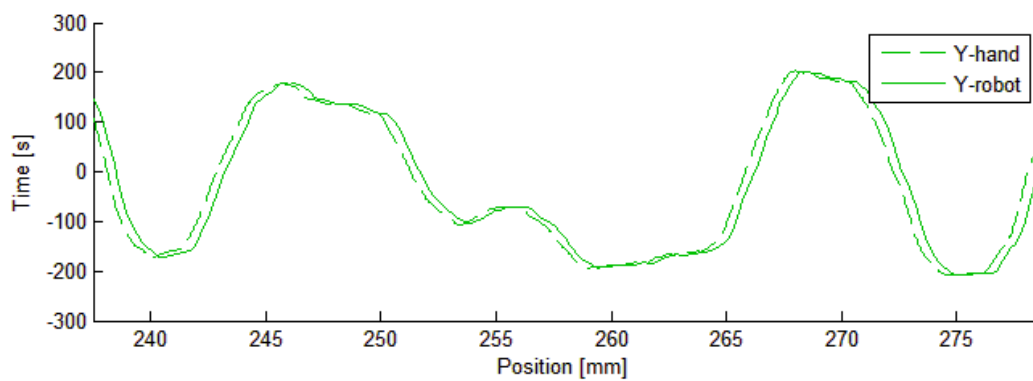
7.1. Kết quả thực nghiệm

7.1.1. Kết quả sao chép cử động

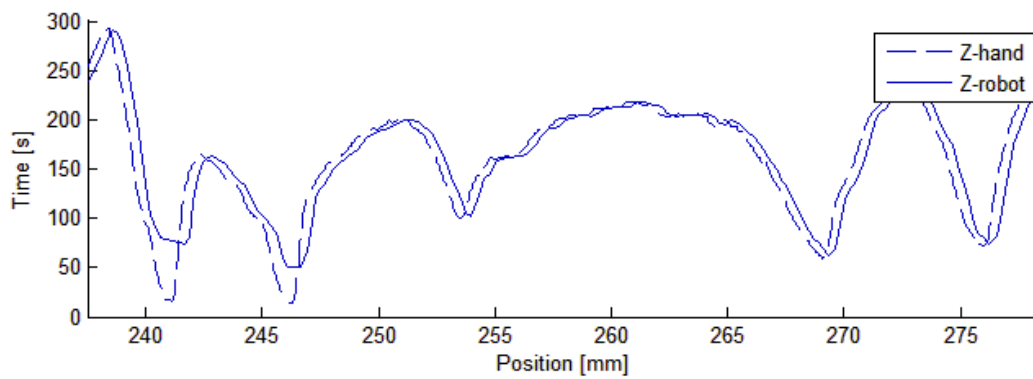
Hệ thống được thử nghiệm với một chuỗi cử động bình thường để kiểm tra độ bám của slave theo master.



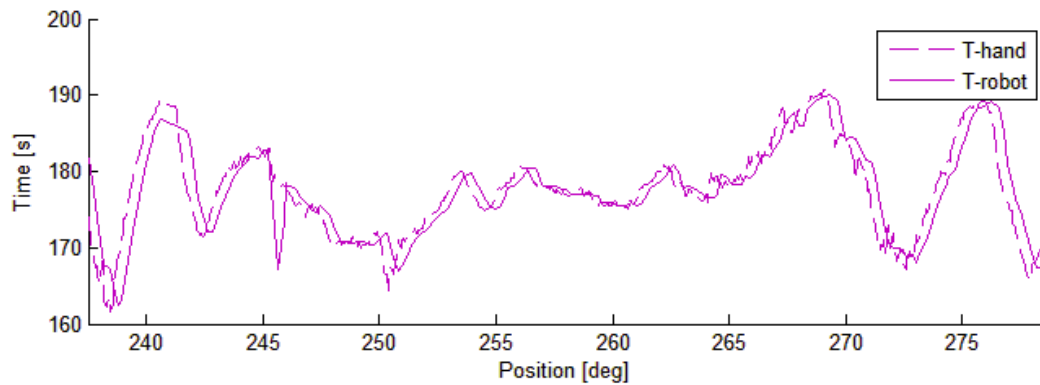
Hình 7.1. Đáp ứng vị trí X



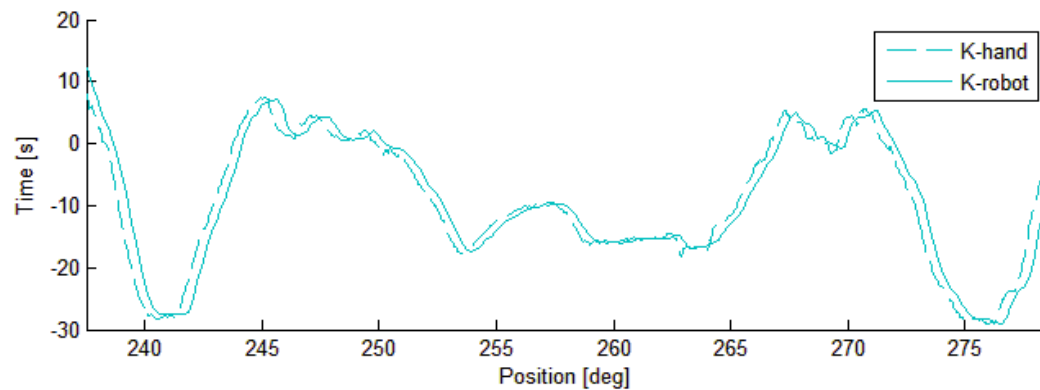
Hình 7.2. Đáp ứng vị trí Y



Hình 7.3. Đáp ứng vị trí Z



Hình 7.4. Đáp ứng góc θ



Hình 7.5. Đáp ứng góc ψ

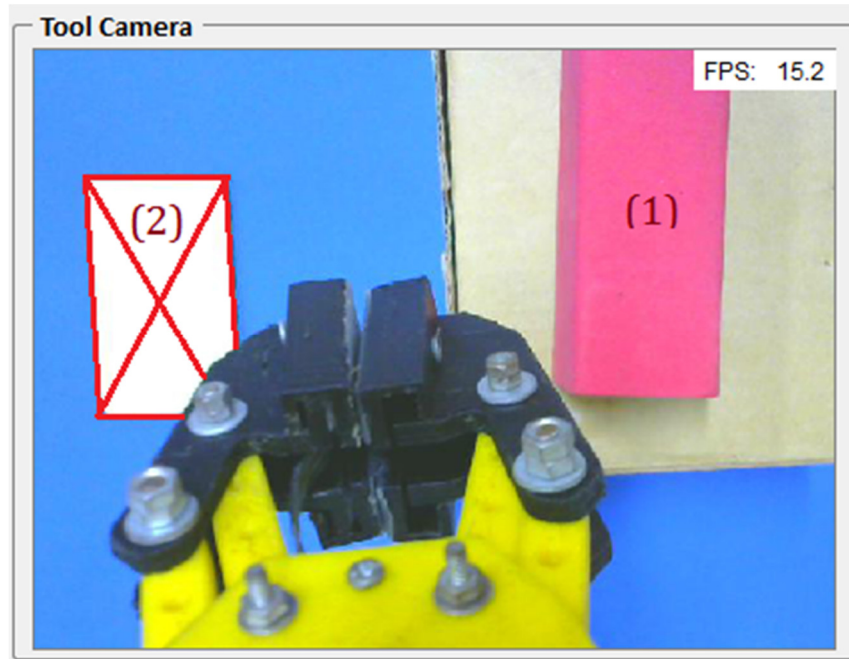
Thông số	Giá trị
Thời gian trễ	0.439 s
Sai số vị trí theo trục X	0.653 mm
Sai số vị trí theo trục Y	1.068 mm
Sai số vị trí theo trục Z	2.155 mm
Sai số góc hướng θ	0.538 deg
Sai số góc hướng ψ	0.163 deg

Bảng 7.1. Độ trễ và sai số giữa tay người và robot.

Kết quả cho thấy khả năng bám tốt của slave theo master. Sai số sao chép giữa tín hiệu điều khiển và đáp là nhỏ và thời gian trễ vừa phải. Thời gian trễ là tổng của thời gian đọc dữ liệu bàn tay, thời gian chuyển động quá độ của robot và thời gian truyền tín hiệu theo chiều đi và về.

7.1.2. Kết quả hoạt động tự động

Ứng dụng chức năng lập trình (Programming) của hệ thống, robot được huấn luyện để có thể tự động lặp lại thao tác điều khiển gắp vật từ của người dùng. Mục tiêu là gắp vật từ vị trí ban đầu (1), di chuyển rồi đặt lên vị trí đích (2).



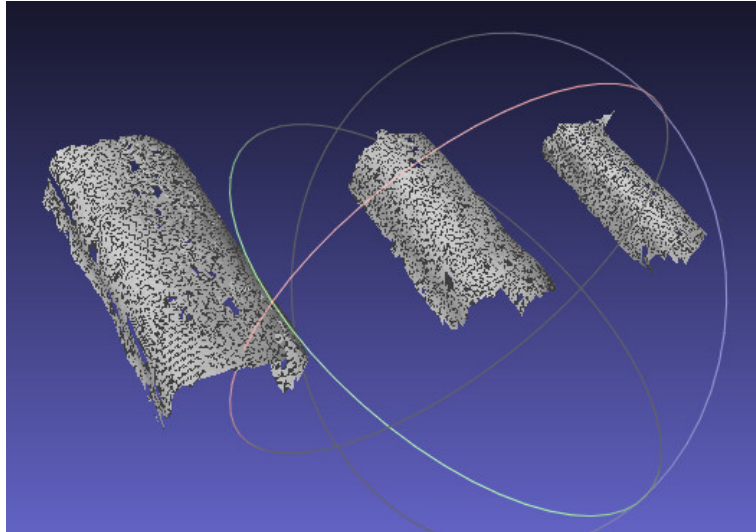
Hình 7.6. Ứng dụng gắp vật tự động



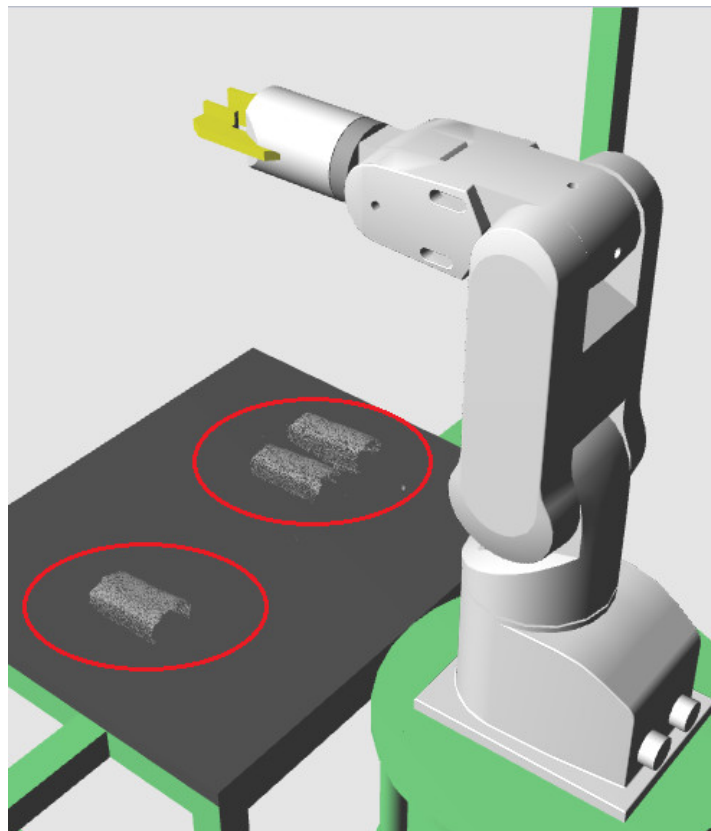
Hình 7.7. Kết quả gắp vật tự động

Sau khi được người dùng huấn luyện một lần, robot lặp lại thao tác gấp vật với độ chính xác cao.

7.1.3. Kết quả tái tạo vật trong không gian ảo 3D



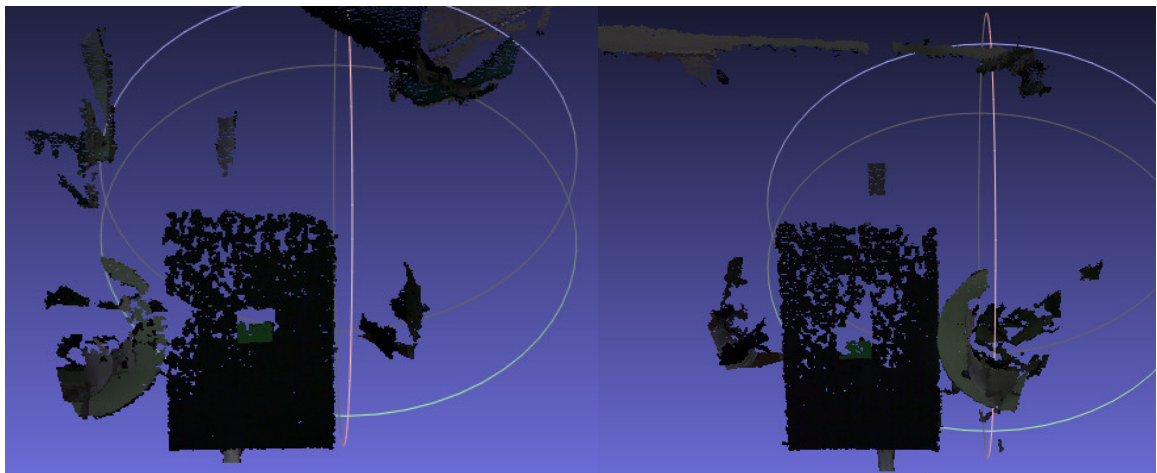
Hình 7.8. Các khối hộp được tái tạo bề mặt.



Hình 7.9. Vật được tái tạo trong không gian ảo 3D

Dữ liệu được xử lý và cập nhật mỗi 2 giây. Tuy nhiên khi số lượng vật mẫu tăng, dữ liệu lớn sẽ làm chậm lại tốc độ xử lý.

Việc sử dụng hai camera Kinect để quan sát hai phía của vật có hạn chế là tia hồng ngoại của mỗi Kinect gây nhiễu trên Kinect còn lại, dẫn đến các lỗ trống trên hình ảnh 3D. Tuy là nhiễu được bù trừ khi ghép hai ảnh lại, nhưng độ tin cậy của mỗi ảnh đơn lẻ đã giảm.



Hình 7.10. Nhiễu lẫn nhau giữa hai Kinect

7.2. Kết luận

7.2.1. Nhiệm vụ đã hoàn thành

Hoàn thành mô hình hoá và mô phỏng động học, quy hoạch quỹ đạo, đồng bộ chuyển động các góc khớp của tay máy.

Hoàn thành hệ thống cơ bản trạm Master và Slave, điều khiển chính xác tay máy theo cử động bàn tay và có tín hiệu phản hồi từ trạm Slave.

Hoàn thành dụng cụ công tác điểm cuối cho tay máy, có phản hồi lực và tự hạn chế lực kẹp.

Hoàn thành tái tạo bề mặt 3D cho vật trong không gian làm việc của robot.

Hoàn thành ứng dụng demo tự vận hành lặp lại cho tay máy.

7.2.2. Những điểm hạn chế

Do thời gian thực hiện Luận văn ngắn và hạn chế về phần cứng máy tính nên Luận văn vẫn còn một số mặt hạn chế:

Thời gian xử lý ảnh dài, không thể thử nghiệm các phương pháp đa dạng để thu được dữ liệu tốt nhất.

Quá trình cập nhật không gian 3D không đáp ứng được thời gian thực.

7.2.3. Hướng phát triển đề tài

Xây dựng một hệ thống phản hồi hoàn chỉnh bao gồm hình ảnh 3D, âm thanh, lực để người điều khiển có thể cảm nhận chính xác nhất về môi trường thực tế bên phía trạm Slave.

Phát triển cơ chế di chuyển cho tay máy giúp tăng tính linh hoạt, tăng mạnh tính ứng dụng cho hệ thống.

Hệ thống có thể được phát triển cho nhiều công việc như hỗ trợ lập trình cho tay máy trong công nghiệp, cứu hộ thiên tai, làm việc trong môi trường độc hại đối với con người.

TÀI LIỆU THAM KHẢO

Tài liệu Tiếng Việt

- [1] Huỳnh Thái Hoàng, *Bài giảng Cơ sở tự động*, Đại học Bách khoa TP.HCM, 2016.
- [2] Huỳnh Thái Hoàng, *Bài giảng Lý thuyết điều khiển nâng cao*, Đại học bách khoa TP.HCM, 2016.
- [3] Nguyễn Đức Thành, *Kỹ thuật Robot*, 2017.
- [4] Trần Trường Sơn và Võ Xuân Xăng, *Luận văn tốt nghiệp Ứng dụng giải thuật ICP kết hợp tái tạo mô hình 3D của vật thể từ ảnh depth map*, Đại học Bách Khoa TP.HCM, 2012.

Tài liệu Tiếng Anh

- [5] M. Spong, *Robot Modeling and Control*, Wiley Inc., 2005.
- [6] S. Schaal. Class Lecture. CS545. Topic: “*Jacobian methods for inverse kinematics and planning*”, University of Southern California, 2010.
- [7] Passenberg, C., Peer, A. and Buss, M. (2010) A survey of environment-, operator-, and task-adapted controllers for teleoperation systems. *Mechatronics*, 20 (7). pp. 787-801. ISSN 0957-4158 Available: eprints.uwe.ac.uk
- [8] Mitsubishi Electric Corp., RV-2AJ Industrial Robot Manual, 2007.
- [9] Leap Motion Inc., Leap Motion Documentation, 2018. [Online], Available: developer.leapmotion.com
- [10] MathWorks Inc., Simscape Multibody Documentation, 2018. [Online], Available: www.mathworks.com
- [11] Qt Company, Qt Documentation, 2018. [Online], Available : doc.qt.io
- [12] Open Perception, PCL Documentation, 2018 [Online], Available : pointclouds.org

[13] Nicolas Burrus, Nestk Documentation, 2018 [Online], Available:
<http://rgbdemo.org/index.php/Documentation/Nestk>