ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN TỬ - VIỄN THÔNG

****

BÁO CÁO CUỐI KỲ
MÔN: CHUYÊN ĐỀ 2

Xây dựng và so sánh các phương pháp thuật toán điều khiển robot vẽ trong bài toán lập kế hoạch và thực thi chuyển động

GVHS: TS. Nguyễn Văn Hiếu

SVTH:	Ngô Viết Huy Hoàng	106210215	21KTMT
	Nguyễn Minh Trung	106210256	21KTMT2
	Đào Văn Minh	106210244	21KTMT2

Đà Nẵng, 2025

SINH VIÊN THỰC HIỆN

Họ và tên	**Phần trăm đóng góp**
Ngô Viết Huy Hoàng	34%
Nguyễn Minh Trung	33%
Đào Văn Minh	33%

LỜI NÓI ĐẦU

Trong hành trình học tập và rèn luyện tại Trường Đại học Bách khoa – Đại học Đà Nẵng, nhóm chúng em luôn ý thức rằng mỗi học phần không chỉ là một yêu cầu học thuật, mà còn là cơ hội để tiếp cận sâu hơn với tri thức, công nghệ và những vấn đề thực tiễn của xã hội hiện đại. Báo cáo *Chuyên đề 2* với đề tài "**Xây dựng và so sánh các phương pháp thuật toán điều khiển robot vẽ trong bài toán lập kế hoạch và thực thi chuyển động**" là kết quả của quá trình học hỏi, tìm tòi và nỗ lực không ngừng của toàn nhóm trong suốt thời gian học tập và nghiên cứu.

Xuất phát từ niềm đam mê đối với lĩnh vực robot và tự động hóa, cùng với nền tảng kiến thức được tiếp cận thông qua các tài liệu chuyên ngành, đặc biệt là các công trình và giáo trình của Peter Corke, nhóm chúng em đã từng bước tiếp cận những khái niệm cốt lõi về động học robot, lập quỹ đạo chuyển động và điều khiển. Thông qua đề tài này, nhóm không chỉ mong muốn củng cố kiến thức lý thuyết đã học trên lớp, mà còn hướng đến việc hiện thực hóa mô hình robot trong thực tế, qua đó rèn luyện tư duy kỹ thuật, khả năng làm việc nhóm và kỹ năng giải quyết vấn đề.

Báo cáo này trình bày một cách hệ thống quá trình nghiên cứu và triển khai cánh tay robot 4 bậc tự do, bao gồm cơ sở lý thuyết về robot học, mô phỏng trên MATLAB & Simulink, cũng như quá trình thiết kế và xây dựng mô hình phần cứng. Mỗi nội dung đều là kết quả của sự trao đổi, thử nghiệm và điều chỉnh liên tục giữa các thành viên trong nhóm.

Với sức trẻ, tinh thần ham học hỏi và lòng nhiệt huyết của sinh viên Bách khoa Đà Nẵng, nhóm chúng em tin rằng những kiến thức và kinh nghiệm tích lũy được từ đề tài này sẽ là nền tảng quan trọng cho các nghiên cứu và ứng dụng sâu hơn trong tương lai. Chúng em hy vọng rằng, từ những bước đi nhỏ này, có thể góp phần lan tỏa tinh thần nghiên cứu khoa học, từng bước đưa các giải pháp robot và tự động hóa đến gần hơn với đời sống, sản xuất và sự phát triển chung của đất nước.

Nhóm chúng em xin chân thành cảm ơn sự hướng dẫn tận tình của giảng viên hướng dẫn, cùng sự hỗ trợ của nhà trường đã tạo điều kiện để nhóm hoàn thành báo cáo này.

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN 1

1.1. Lý do chọn đề tài 1

1.2. Mục tiêu và phạm vi đề tài 1

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 2

2.1. Cánh tay robot trong không gian 3 chiều 2

2.1.1. Giới thiệu robot cánh tay 3D 2

2.1.2. Biểu diễn hình học cơ bản cho robot 3D 3

2.1.3. Cấu trúc robot như một rigid-body tree 4

2.1.4. Mô tả robot 3D bằng Elementary Transform Sequence (ETS3) 4

2.2. Bài toán động học robot 5

2.2.1. Động học thuận (Forward Kinematics) 5

2.2.2. Động học nghịch (Inverse Kinematics) 7

2.3. Lập quỹ đạo chuyển động (Trajectory Planning) 13

2.3.1. Chuyển động trong không gian khớp (Joint-Space Motion) 13

2.3.2. Chuyển động trong không gian làm việc (Cartesian Space Motion) 15

2.4.2. Phân tích chuyển động qua điểm kỳ dị (Motion Through Singularity) 17

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG 22

3.1. Phân tích và Lựa chọn Cấu hình Robot 22

3.1.1. Phân tích Yêu cầu Không gian Nhiệm vụ 22

3.1.2. Điều kiện Cần và Kết luận Lý thuyết 22

3.1.3. Đề xuất Mô hình 22

3.2. Thiết kế hệ thống 23

3.2.1. Thiết kế cơ khí 23

3.2.2. Đồng bộ mô phỏng và cơ khí 34

CHƯƠNG 4: KẾT QUẢ - ĐÁNH GIÁ 38

4.1. Kết quả hiệu suất giữa hai thuật toán IK solver trong việc giải quyết bài toán ... 38

4.1.1. Dữ liệu 38

4.1.2. Các chỉ tiêu so sánh 38
4.1.3. Kết quả và nhận xét 38
4.2. Kết quả quỹ đạo vẽ và chuyển động robot 40
4.2.1. Kết quả mô phỏng 40
4.2.2. Kết quả mô hình thực tế 43
4.3. Nhận xét chung 43
KẾT LUẬN 45
TÀI LIỆU THAM KHẢO 46

CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN

1.1. Lý do chọn đề tài

- Trong bối cảnh cuộc Cách mạng Công nghiệp 5.0 đang diễn ra, các hệ thống tự động hóa đóng vai trò cốt lõi trong việc nâng cao năng suất, giảm chi phí vận hành và đảm bảo tính ổn định trong sản xuất. Robot công nghiệp ngày càng được ứng dụng rộng rãi trong dây chuyền lắp ráp, đóng gói, gia công cơ khí, hàn tự động và nhiều tác vụ lặp đi lặp lại khác.

- Trong lĩnh vực ứng dụng đặc thù, robot vẽ hình hay robot viết chữ là một ví dụ điển hình của bài toán điều khiển chuyển động chính xác theo quỹ đạo. Mặc dù thao tác vẽ có vẻ đơn giản đối với con người, việc tái hiện chuyển động này trên một cánh tay robot lại đòi hỏi sự nghiên cứu về động học, lập quỹ đạo và điều khiển servo. Đề tài cánh tay robot vẽ chữ bằng MATLAB và ESP32 vì vậy mang ý nghĩa thực tiễn và học thuật, giúp nhóm em tiếp cận cụ thể với mô hình robot công nghiệp thu nhỏ, góp phần vào việc nghiên cứu, thử nghiệm và mở rộng ứng dụng của robot tự động trong tương lai.

1.2. Mục tiêu và phạm vi đề tài

- Mục tiêu chính của đề tài là xây dựng một mô hình mô phỏng và điều khiển cánh tay robot 4 bậc tự do để vẽ chữ hoặc hình trên mặt phẳng thông qua việc:

    - Thiết kế mô phỏng robot trên MATLAB.
    - Xây dựng thuật toán tạo quỹ đạo cho đầu bút.
    - Áp dụng các thuật toán động học nghịch (Inverse Kinematics) để tính góc điều khiển từng khớp.
    - Truyền lệnh từ MATLAB xuống ESP32-C3 để điều khiển các servo thực tế.

- Phạm vi đề tài tập trung vào các thuật toán IK để tìm ra góc khớp cho bài toán cánh tay robot. Áp dụng thuật toán xử lý ảnh, lấy biên cho việc tìm quỹ đạo cho bài toán vẽ hình tĩnh, các quỹ đạo 2D đơn giản (chữ cái, đường cong cơ bản).

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Cánh tay robot trong không gian 3 chiều

2.1.1. Giới thiệu robot cánh tay 3D

- Trong lĩnh vực robot học, Không gian nhiệm vụ (Task Space), ký hiệu là τ, định nghĩa tập hợp tất cả các "tư thế" (pose) mà cơ cấu chấp hành cuối (end-effector) của robot có thể đạt được. Đối với các thao tác phức tạp trong môi trường 3 chiều, không gian nhiệm vụ được mô tả bằng biểu thức: $\mathcal{T} \subset R^3 \times S^3$ mô tả một không gian 6 bậc tự do (6-DoF), là yếu tố cơ bản để robot có thể thao tác tùy ý trong môi trường 3 chiều. Trong đó:
  - R^3 (Không gian Vị trí): Không gian (Ox, Oy, Oz)
  - S^3 (Không gian Hướng): Đại diện cho hướng (orientation) của cơ cấu chấp hành cuối.
- Tập hợp đầy đủ bao gồm cả vị trí và hướng của một vật thể rắn trong không gian được gọi là nhóm Euclid đặc biệt $SE(3)$.Trong tính toán kỹ thuật, một phần tử thuộc $SE(3)$ thường được biểu diễn dưới dạng ma trận biến đổi đồng nhất (Homogeneous Transformation Matrix) kích thước 4×4:

$$\mathrm{T} = \begin{pmatrix} \mathrm{R} & \mathrm{p} \\ 0 & 1 \end{pmatrix} \tag{1}$$

Trong đó:

- R: Ma trận quay 3×3 (đại diện cho hướng).
- p: Véc-tơ tịnh tiến 3×1 (đại diện cho vị trí).

- Để một robot có thể tiếp cận bất kỳ tư thế nào trong không gian làm việc này, không gian cấu hình C của robot phải thỏa mãn điều kiện:

$$\dim C \geq \dim \mathcal{T}$$

- Điều này giải thích lý do tại sao các tay máy đa năng thường được thiết kế với sáu bậc tự do (khớp) trở lên.

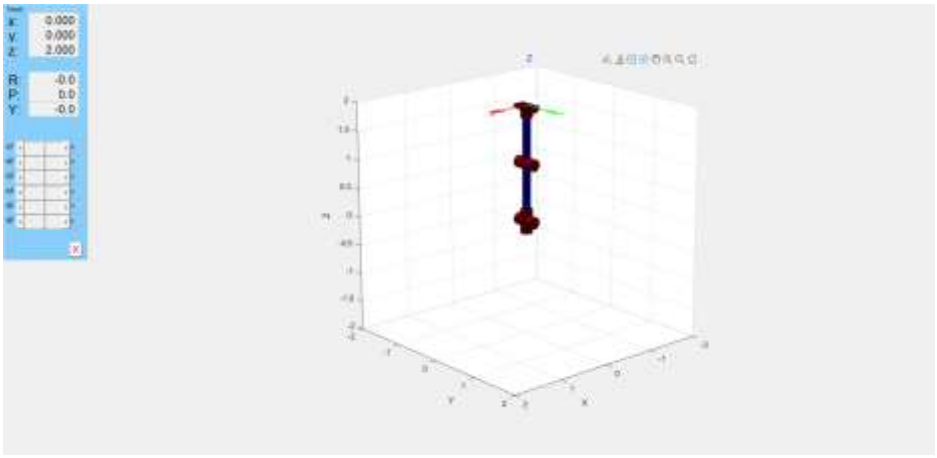
Ví dụ: ta có ma trận của cơ cấu chấp hành cuối như sau:

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

Ma trận kết quả T cho thấy:

$$R = I_{3\times3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

- Điều này chứng tỏ tại cấu hình khởi tạo, hệ tọa độ của cơ cấu chấp hành cuối có phương chiều trùng khớp hoàn toàn với hệ tọa độ gốc (Base Frame), không xuất hiện góc lệch (Orientation offset). Vị trí chỉ thay đổi tịnh tiến dọc theo trục Z một đoạn z = 2 đơn vị chiều dài, mô tả ở hình 1.

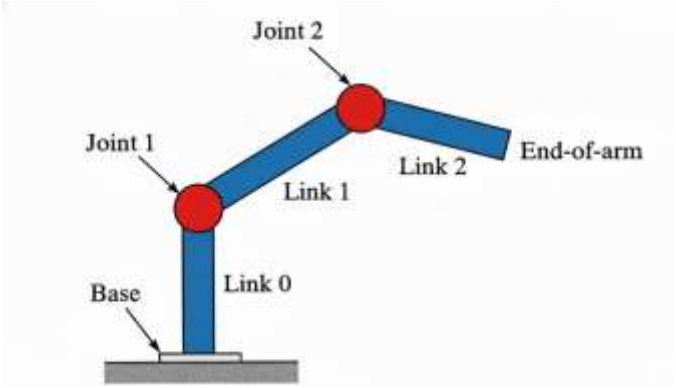


Hình 1. Mô tả cánh tay robot có cơ cấu chấp hành cuối giống như ma trận ví dụ trên

2.1.2. Biểu diễn hình học cơ bản cho robot 3D

- Robot được cấu thành bởi khớp (Joints) và Link (nối các khớp với nhau).
- Mối quan hệ giữa các Joints và các Link frames (hệ tọa độ gắn liền với link) được mô tả như sau:

- Frame cho Link l được ký hiệu là {l}.
- Pose (tư thế) của frame này là một hàm phụ thuộc vào các tọa độ khớp (joint coordinates) $\{q_j, j = 1, \dots, l\}$.
- Về mặt quan hệ hệ thống: Link l được xác định là "parent" (cha) của Link $l + 1$ và là "child" (con) của Link $l - 1$.
- Fixed Base (đế cố định) được quy ước là Link 0 và không có parent. Ngược lại, End Effector (cơ cấu chấp hành cuối) là thành phần cuối cùng và không có child.



Hình 2. Mô tả biểu diễn hình học cơ bản robot 3D bậc 2

2.1.3. Cấu trúc robot như một rigid-body tree

- Để giải quyết các bài toán phức tạp hơn về động lực học và điều khiển, chúng ta sử dụng các đối tượng chuyên dụng (dedicated objects) giúp đóng gói các khái niệm về khâu (link), khớp (joint) và cấu trúc robot. Phương pháp này sử dụng ba đối tượng cốt lõi trong các công cụ mô phỏng (như MATLAB Robotics System Toolbox): rigidBodyTree, rigidBody, và rigidBodyJoint, được mô tả ở bảng 1.

Bảng 1. Mô tả về rigidBodyTree, rigidBody, và rigidBodyJoint

Đối tượng (Object)	**Vai trò & Định nghĩa**	**Đặc điểm & Thuộc tính chính**	**Liên hệ với cơ thể con người**
rigidBodyTree	Cấu trúc tổng thể Là đối tượng chứa ("container") đại diện cho toàn bộ robot. Nó quản lý sự kết nối giữa các bộ phận.	- Hỗ trợ cấu trúc nối tiếp (serial) hoặc phân nhánh (tree). - Là đầu vào cho các thuật toán tính toán động học và động lực học.	Toàn bộ cơ thể người (bao gồm tất cả xương và khớp kết nối với nhau).
rigidBody	Khâu / Vật rắn (Link) Đại diện cho một phần tử vật lý rắn của robot. Trong tài liệu, thuật ngữ này đồng nghĩa với "Link".	- Chứa các tính chất vật lý: Khối lượng, Quán tính, Tâm khối. - Chứa thông tin về hình học va chạm.	Xương cánh tay hoặc Xương cẳng chân.
rigidBodyJoint	Khớp nối (Joint) Định nghĩa mối quan hệ chuyển động tương đối giữa một khâu con (rigidBody) và khâu cha của nó.	- Xác định loại chuyển động: Quay (Revolute), Tịnh tiến (Prismatic), hoặc Cố định (Fixed). - Xác định trục quay và giới hạn góc quay.	Khớp khuỷu tay (nối xương cánh tay và xương cẳng tay).

2.1.4. Mô tả robot 3D bằng Elementary Transform Sequence (ETS3)

- Để máy tính có thể hiểu và tính toán được chuyển động của robot, ta cần định nghĩa các thành phần vật lý của nó dưới dạng các "đối tượng" (objects) trong lập trình. Theo tài liệu kỹ thuật, một robot được cấu thành từ 3 đối tượng cốt lõi:

- Vật rắn / Khâu (rigidBody): Đại diện cho các đoạn link của robot. Đối tượng này chứa các thuộc tính vật lý như khối lượng, quán tính và hình dáng.
- Khớp nối (rigidBodyJoint): Là thành phần định nghĩa sự chuyển động tương đối (quay hoặc tịnh tiến) giữa một khâu con và khâu cha của nó.
- Cây Robot (rigidBodyTree): Là đối tượng bao trùm ("container"), đóng vai trò quản lý sự kết nối của tất cả các khâu và khớp lại với nhau thành một hệ thống nhất (có thể là chuỗi nối tiếp hoặc dạng cây).

- Vấn đề: Việc tạo thủ công từng đối tượng rigidBody, định nghĩa từng rigidBodyJoint, sau đó lắp ráp chúng lại bằng lệnh addBody thường rất phức tạp, tốn thời gian và dễ sai sót, đặc biệt với robot nhiều bậc tự do. Để giải quyết sự phức tạp trên, ta sử dụng phương pháp mô tả Chuỗi biến đổi cơ bản (ETS3). Phương pháp này coi việc mô hình hóa robot giống như mô tả một "hành trình" (kinematic journey) đi từ đế (Base) đến điểm tác động cuối (Tip).
- Nguyên lý hoạt động: Thay vì lắp ráp từng khối, ta "bước đi" dọc theo cấu trúc hình học của robot từ gốc đến ngọn và ghi lại các phép biến đổi:

(1) Di chuyển (Translation): Đi dọc theo các trục tọa độ một khoảng bằng chiều dài khâu L.
(2) Thực hiện khớp (Rotation/Action): Thực hiện phép quay khớp q tại các điểm nối.

Công thức tổng quát của hành trình này là tích của các biến đổi sơ cấp:

$$E = T_{L_1} \cdot R_{q_1} \cdot T_{L_2} \cdot R_{q_2} \dots \cdot R_{q_n} \quad (4)$$

Pipeline hoàn chỉnh xây dựng Robot tự động:

(1) Bước 1: Định nghĩa Thông số Hình học: Xác định các kích thước cố định của robot (chiều dài các Link) dựa trên bản vẽ thiết kế.
(2) Bước 2: Viết Chuỗi Hành trình (ETS Expression): Mô tả cấu trúc robot. Tại bước này, ta kết hợp các phép tịnh tiến (Tx, Ty, Tz) cho kích thước vật lý và các phép quay (Rx, Ry, Rz) cho các khớp động.
(3) Bước 3: Tự động hóa Khởi tạo (ets2rbt): Sử dụng hàm ets2rbt (ETS to Rigid Body Tree) để chuyển đổi chuỗi biểu thức ở Bước 2 thành đối tượng robot hoàn chỉnh.

2.2. Bài toán động học robot

2.2.1. Động học thuận (Forward Kinematics)

- Forward Kinematics (FK) là quá trình xác định vị trí và hướng (pose) end-effector dựa trên các thông số góc khớp của robot.
- Trong một hệ robot có n bậc tự do (DOF) với góc khớp $q = [q_1, q_2, \dots, q_n]$ và các thông số hình học (kích thước cánh tay, chiều dài liên kết, trục xoay), FK trả về vị trí và hướng của end-effector trong hệ tọa độ gốc. Nói cách khác, FK là quá trình dự đoán vị trí của cơ cấu chấp hành cuối của robot khi biết các góc khớp. Bài toán được mô tả như sau:

$$T = \mathcal{K}(q) \tag{5}$$

Trong đó:

- T là ma trận đồng nhất 4×4 biểu diễn vị trí và hướng của end-effector trong hệ tọa độ cơ sở.
- $q \in SE(3)$ là vector khớp.
- $SE(3)$ là nhóm các phép biến đổi trong không gian 3D.

- Ma trận chuyển biểu diễn bằng ma trận Homogeneus Transformation:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{6}$$

Trong đó:

- $R \in SO(3)$là ma trận xoay $3x3$.
- $t = [x\ y\ z]^T$ là vector tịnh tiến.

Ý nghĩa:

- Mô tả dáng của robot (pose).
- Có thể nhân chuỗi để tạo ra chuyển động phức tạp.

- Giải quyết bài toán bằng Denavit-Hartenberg (DH parameters):

- Mỗi khớp được mô tả bởi 4 tham số:

Bảng 2. Các tham số biểu diễn góc khớp

Ký hiệu	**Ý nghĩa**
a_i	Độ dài link (link length)
α_i	Xoay quanh X (link twist)
d_i	Tịnh tiến theo Z (link offet)
θ_i	Xoay quanh Z (joint angle)

  - Với khớp Revolute: θ_i thay đổi, d_i cố định.
  - Với khớp Prismatic: d_i thay đổi, θ_i cố định.
- Ma trận biến đổi DH giữa frame i-1:

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

- Giải quyết bài toán bằng ETS (Elementary Transform Sequence): Mỗi robot được mô tả như chuỗi:

$$T(q) = E_1(q_1)E_2(q_2) \dots E_k(q_k) \quad (8)$$

- Mỗi E_i có dạng:
  - $Tx(d)$: tịnh tiến theo X
  - $Ty(d)$: tịnh tiến theo Y
  - $Tz(d)$: tịnh tiến theo Z
  - $Rx(\theta)$: quay quanh X
  - $Ry(\theta)$: quay quanh Y
  - $Rz(\theta)$: quay quanh Z

- Bài toán FK luôn có nghiệm duy nhất với tập giá trị khớp đã cho. Thường được ứng dụng trong mô phỏng robot và kiểm tra chuyển động, kiểm tra va chạm, kiểm tra giới hạn khớp, ngoài ra cũng có thể dùng để chuẩn hóa dữ liệu cho IK.

2.2.2. Động học nghịch (Inverse Kinematics)

- Inverse Kinematics (IK) là bài toán xác định cấu hình khớp (q) cần thiết để đạt được một tư thế (pose) mong muốn (ξ_E), tư thế này bao gồm vị trí và hướng của robot:

$$q = \mathcal{K}^{-1}(T_d) \quad (9)$$

- IK là bài toán có thể vô nghiệm, có thể có một nghiệm, nhiều nghiệm hoặc vô số nghiệm vì ràng buộc hình học trong không gian, giới hạn khớp, và liên quan đến cấu trúc robot.

- Phương pháp Closed-form để giải quyết IK: Phương pháp này sử dụng các kỹ thuật hình học hoặc đại số để tìm ra công thức đóng cho các góc khớp.

- Điều kiện cần: Độ phức tạp của nghiệm giải tích tăng rất nhanh theo số lượng bậc tự do của robot. Phương pháp giải closed-form đặc biệt phù hợp với các robot công nghiệp sáu bậc tự do có cổ tay cầu (spherical wrist). Cơ cấu cổ tay cầu bao gồm ba khớp quay có trục trực giao và giao nhau tại một điểm, tạo ra chuyển động thuần xoay mà không làm thay đổi vị trí của khâu chấp hành cuối. Nhờ đó, bài toán IK có thể tách thành hai phần độc lập:
  (1) Xác định vị trí của cổ tay (wrist center) bằng ba khớp đầu tiên.
  (2) Xác định hướng của khâu cuối thông qua ba khớp của cổ tay cầu.
- Cách phân tách này làm đơn giản hóa đáng kể việc tìm nghiệm giải tích và là cơ sở cho đa số thuật toán closed-form trong robot công nghiệp.

- Phương pháp Numerical để giải quyết IK: Phương pháp này sử dụng cho các robot mà lời giải giải tích không khả thi hoặc không có sẵn, hoặc khi xử lý các trường hợp đặc biệt mà phương pháp closed-form không giải được (ví dụ: tại điểm kỳ dị).

- Nguyên tắc là tối ưu hóa: Bài toán IK được xem là một bài toán tối ưu hóa nhằm giảm thiểu lỗi giữa tư thế khâu cuối hiện tại ($\mathcal{K}(q)$) và tư thế mong muốn (ξ_E^d):

$$q^* = arg\min_q \left\| \mathcal{K}(q) \ominus \xi_E^d \right\| \tag{10}$$

- Dựa trên quan hệ:

$$x = \dot{J}(q)\dot{q} \tag{11}$$

Trong đó:

  - x: pose của end-effector.
  - $J(q)$: Jacobian.

- Phương pháp lặp:

$$q_{k+1} = q_k + J^{\#}(q_k)\big(x_d - f(q_k)\big) \tag{12}$$

- Phương pháp Numerical chậm hơn đáng kể so với phương pháp giải tích. Tuy nhiên, thời gian chạy của nó phụ thuộc mạnh vào ước tính ban đầu của tọa độ khớp.

- IK là thành phần cốt lõi trong điều khiển robot, cho phép chuyển đổi yêu cầu chuyển động trong không gian làm việc thành các giá trị góc khớp hoặc tịnh tiến tại từng khớp. IK được ứng dụng rộng rãi trong nhiều lĩnh vực của robot công nghiệp cũng như robot dịch vụ: Điều khiển vị trí và định hướng của robot công nghiệp, robot humanoid, robot legged, robot phục hồi chức năng, hệ thống in 3D và gia công bằng robot, …

- Bài toán cánh tay robot là một trường hợp điển hình cho việc áp dụng Inverse Kinematics (IK) vì mục tiêu điều khiển robot thường là đưa end-effector đến một pose mong muốn trong không gian ba chiều. IK không chỉ giúp robot xác định chính xác các góc khớp để đạt dáng (pose) mong muốn mà còn kết hợp với nội suy và gán thời gian, cho phép lập quỹ đạo mượt, tránh các kỳ dị và đáp ứng các yêu cầu ứng dụng trong công nghiệp, y tế, teleoperation và mô phỏng.

2.2.3. Thuật toán IK Numerical giải quyết để giải quyết bài toán

- Có nhiều thuật toán được sử dụng để giải quyết bài toán hay còn gọi là IK solver, nhưng trong Matlab đã hỗ trợ sẵn 2 thuật toán mạnh, tối ưu để thực hiện bài toán là:

- Broyden-Fletcher-GoldFarb-Shanno Gradient Projection (BFGS Gradient Projection)

- Levenberg-Marquardt (LM).

Vậy nên trong bài nghiên cứu này, nhóm sẽ tập trung vào hai thuật toán này để giải quyết bài toán vẽ cánh tay robot.

Thuật toán BFGS Gradient Projection

- Trong bài toán Numerical IK mục tiêu của nó là tìm cầu hình khớp q ($q = [q_1, q_2, \dots]^T$) sao cho end-effector đạt tới pose mong muốn: $f(q) = x_d$. Đây là hệ phi tuyến → chuyển thành bài toán tối ưu:

$$\min_q \Phi(q) = \frac{1}{2}\|f(q) - x_d\|^2 \tag{13}$$

- Vấn đề khó khăn là bài giải cơ bản không tuyến tính mạnh, thuật toán Gradient Descent (GD) hội tụ chậm, thuật toán Gauss Newton yêu cầu ma trận Hessian nhưng khó tính và mất thời gian. Bên cạnh đó, khớp robot trong thực tế còn có ràng buộc: $q_{min} \le q \le q_{max}$, buộc thuật toán phải tránh cập nhật vượt giới hạn.

- Vậy nên, nhóm chọn BFGS Gradient Projection để giải quyết các vấn đề trên, khi kết hợp ta sẽ được một thuật toán tối ưu nhanh, bỏ qua bước tính Hessian, không vượt quá giới hạn khớp và ổn định trong môi trường robotics.

Công thức cập nhật:

(1) Gradient của hàm lỗi

$$g = J^T e \tag{14}$$

Trong đó:

- $e = f(q) - x_d$: Sai số của end-effector.
- J: Jacobian.
- g: Hướng mà lỗi tăng nhanh nhất.

⇨ Ta sẽ đi ngược hướng g để giảm lỗi.

(2) Gradient Projection

$$g_p = P(g) \tag{15}$$

Trong đó:

- $P(\cdot)$: Toán tử chiếu.

⇨ Khi một joint đang ở giới hạn, thành phần gradient hướng vượt giới hạn sẽ bị loại bỏ. Giúp q luôn nằm trong khoảng giới hạn.

(3) Bước cập nhật joint

$$\Delta q = -H_k^{-1} g_p \tag{16}$$

Trong đó:

- $-H_k^{-1}$: Xấp xỉ nghịch đảo Hessian.
  ⇨ Cho phép uốn cong hướng đi theo local curvature của bài toán.
  ⇨ Cập nhật thông min hơn gradient descent

(4) Cập nhật Hessian theo BFGS

$$H_{k+1} = H_k + \frac{q\langle k\rangle q\langle k\rangle^T}{q\langle k\rangle^T p\langle k\rangle} - \frac{H_k p_k p_k^T H_k}{p_k^T H_k p_k} \tag{17}$$

Trong đó:

- $p_k = q_{k+1} - q_k$: Vector thay đổi joint.
- $q_k = g_{k+1} - g_k$: Sự thay đổi gradient do bước đi trước.
- $\frac{q\langle k\rangle q\langle k\rangle^T}{q\langle k\rangle^T p\langle k\rangle}$: Thêm thông tin mới vào Hessian; điều chỉnh theo gradient mới.
- $\frac{H_k p_k p_k^T H_k}{p_k^T H_k p_k}$: Loại bỏ ảnh hưởng sai của xấp xỉ cũ.
  ⇨ Ý nghĩa: Làm Hessian tiệm cận Hessian thật mà không cần tính trực tiếp.

(5) Cập nhật giá trị joint

$$q_{k+1} = q_k + \alpha \Delta q \tag{18}$$

Trong đó:

- α: Step size giúp điều chỉnh tốc độ hội tụ.
  ⇨ Đảm bảo cập nhật mượt, tránh overshoot, luôn giữ q trong vùng khả thi.

Thuật toán LM

- Thuật toán LM là một phương pháp lặp mạnh mẽ được sử dụng để giải quyết các bài toán tối ưu hóa bình phương tối thiểu phi tuyến (non-linear least squares problems). Trong ngữ

cảnh động học ngược (Numerical Inverse Kinematics), LM được sử dụng để tìm cấu hình khớp q nhằm giảm thiểu sai số giữa phép động học thuận $\mathcal{K}(q)$ và tư thế mục tiêu.

- Thuật toán LM được phát triển để cải thiện tính ổn định của các phương pháp tối ưu hóa dựa trên gradient, đặc biệt là phương pháp Gauss-Newton.

- Vấn đề cốt lõi là khi robot ở gần các điểm kỳ dị (singularities), ma trận Hessian *(được trình bày ở phần 2.4.1)* gần đúng trở nên điều kiện kém (poorly conditioned) hoặc kỳ dị (singular). Việc đảo ngược ma trận này để tính toán bước cập nhật trở nên không ổn định hoặc không khả thi.

- LM giải quyết vấn đề này bằng cách thêm vào một số giảm chấn tương ứng (λ) vào ma trận cần đảo ngược, việc thêm như vậy đảm bảo ma trận kết quả luôn là ma trận xác định dương và không bị kỳ dị. LM là sự kết hợp hai phương pháp cổ điển Gauss-Newton và Gradient Descent:

- Khi hệ số giảm chấn λ nhỏ (hoặc bằng 0), công thức LM gần giống với Gauss-Newton, cho phép tốc độ hội tụ nhanh khi cấu hình khớp đã gần nghiệm tối ưu.
- Khi lớn, việc thêm làm cho các phần tử đường chéo của ma trận chi phối, khiến công thức cập nhật gần giống với phương pháp Gradient Descent (giảm dần gradient), đảm bảo sự hội tụ ngay cả khi cấu hình hiện tại ở xa nghiệm hoặc gần điểm kỳ dị.

- Công thức cập nhật LM:

$$\delta_q\langle k\rangle = J^T(q\langle k\rangle)MJ(q\langle k\rangle + \lambda 1_{N\times N})^{-1}J^T(q\langle k\rangle)M\Delta(\mathcal{K}q\langle k\rangle, x_E^*) \qquad (19)$$

Trong đó:

- $q\langle k\rangle$: Vector cập nhật cấu hình khớp tại bước lặp thứ k. Đây là giá trị cần tìm.
- $J(q\langle k\rangle)$: Ma trận Jacobian tại cấu hình khớp hiện tại $q\langle k\rangle$. Ma trận này chuyển đổi vận tốc khớp thành vận tốc trong không gian tác vụ.
- M: Ma trận trọng số (Weighting matrix), thường là một ma trận xác định dương, được sử dụng để điều chỉnh mức độ quan trọng của các thành phần lỗi trong véc-tơ lỗi.
- $J(q\langle k\rangle)^T MJ(q\langle k\rangle)$: Ma trận Hessian gần đúng của hàm chi phí.
- λ: Hệ số giảm chấn.
- $I_{N\times N}$: Ma trận đơn vị.
- $\Delta(\mathcal{K}q\langle k\rangle, x_E^*)$: Vector lỗi giữa tư thế $\mathcal{K}q\langle k\rangle$ và x_E^*.

So sánh hai thuật toán

Bảng 3. So sánh lý thuyết hai thuật toán BFGS Gradient và LM

Tiêu chí	BFGS Gradient Projection	Levenberg–Marquardt (LM)
Mục tiêu chính	Tối ưu hóa có ràng buộc khớp $q_{\min} \leq q \leq q_{\max}$	Giải bài toán bình phương tối thiểu phi tuyến, ổn định hóa ma trận Hessian gần kỳ dị
Cơ sở toán học	Kết hợp BFGS (quasi-Newton) với Gradient Projection để xử lý ràng buộc	Kết hợp Gauss–Newton và Gradient Descent, điều chỉnh bằng hệ số giảm chấn λ
Xử lý ràng buộc khớp	Có – dùng phép chiếu gradient $g_p = P(g)$ để loại bỏ thành phần đẩy khớp vượt giới hạn	Không trực tiếp – thường cần kết hợp với phương pháp khác (ví dụ: penalty function, barrier method)
Tính Hessian	Không cần tính trực tiếp – xấp xỉ Hessian bằng BFGS qua các lần lặp	Gần đúng bằng $J^T M J$ (Gauss–Newton), không cần tính đạo hàm bậc hai
Xử lý kỳ dị (singularity)	Không xử lý đặc biệt, nhưng BFGS có thể ổn định hơn Newton thuần do xấp xỉ Hessian	Xử lý tốt – thêm λI để đảm bảo ma trận khả nghịch ngay cả khi J suy biến
Công thức cập nhật	$\Delta q = -H_k^{-1} g_p$ H_{k+1} cập nhật theo BFGS	$\delta_q = (J^T M J + \lambda I)^{-1} J^T M \Delta$
Tốc độ hội tụ	Siêu tuyến tính (quasi-Newton) – nhanh hơn Gradient Descent, gần Newton	Nhanh gần nghiệm (khi $\lambda \to 0$ giống Gauss–Newton), chậm hơn khi λ lớn (giống Gradient Descent)

Tiêu chí	BFGS Gradient Projection	Levenberg–Marquardt (LM)
Điều chỉnh tham số	Cần chọn step size α và khởi tạo H_0 (thường là ma trận đơn vị)	Cần điều chỉnh hệ số giảm chấn λ động (tăng/giảm theo sai số)
Ưu điểm	1. Xử lý ràng buộc khớp trực tiếp. 2. Không cần tính Hessian. 3. Tốc độ hội tụ nhanh nhờ BFGS. 4. Ổn định với bài toán phi tuyến mạnh.	1. Ổn định với kỳ dị nhờ λ. 2. Hội tụ đáng tin cậy ngay cả khi khởi tạo xa nghiệm. 3. Được tích hợp sẵn trong nhiều thư viện (MATLAB, SciPy).
Nhược điểm	1. Không xử lý kỳ dị Jacobian một cách tường minh. 2. Cần lưu trữ/làm mới H_k (cỡ $N \times N$). 3. Phức tạp cài đặt hơn LM.	1. Không xử lý ràng buộc khớp. 2. Cần điều chỉnh λ động để cân bằng tốc độ/ổn định. 3. Có thể chậm nếu λ lớn.

2.3. Lập quỹ đạo chuyển động (Trajectory Planning)

2.3.1. Chuyển động trong không gian khớp (Joint-Space Motion)

- Quy hoạch quỹ đạo trong không gian khớp (Joint-space trajectory planning) là bài toán tìm hàm số q(t) mô tả vị trí của khớp theo thời gian, sao cho khớp chuyển động từ vị trí đầu q_0 đến vị trí cuối q_f trong khoảng thời gian T. Giả sử robot có n bậc tự do, vector biến khớp tại thời điểm t là:

$$q(t) = [q_1(t), q_2(t), \dots, q_n(t)]^T \quad (20)$$

- Trong đó, $t \in [0, T]$. Bài toán đặt ra yêu cầu đảm bảo tính liên tục và trơn (smoothness) không chỉ về vị trí mà còn về vận tốc $\dot{q}(t)$ và gia tốc $\ddot{q}(t)$.

Nội suy đa thức bậc 5 (Quintic Polynomial)

- Trong các hình ảnh mô phỏng (hàm quinticpolytraj), quỹ đạo được tạo ra từ đa thức bậc 5. Đây là phương pháp phổ biến nhất để đảm bảo các điều kiện biên về vị trí, vận tốc và gia tốc đều bằng 0 tại điểm đầu và điểm cuối (giúp robot khởi động và dừng lại êm ái, không bị giật). Hàm quỹ đạo cho một khớp đơn lẻ có dạng:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \quad (21)$$

- Đạo hàm bậc nhất (vận tốc) và bậc hai (gia tốc) tương ứng là:

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \quad (22)$$

$$\ddot{q}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \quad (23)$$

- Để tìm 6 hệ số $(a_0, \dots, a_5)$, ta sử dụng 6 điều kiện biên (Boundary Conditions):

- Tại $t =: q(0) = q_0, \quad \dot{q}(0) = 0, \quad \ddot{q}(0) = 0$
- Tại $t =: q(T) = q_f, \quad \dot{q}(T) = 0, \quad \ddot{q}(T) = 0$

- Giải hệ phương trình này, ta thu được quỹ đạo "Minimum Jerk" (tối thiểu độ giật) cơ bản, phù hợp với chuyển động tự nhiên.

Biên dạng vận tốc hình thang (Trapezoidal Velocity Profile - LSPB)

- Phương pháp thứ hai được nhắc đến (hàm trapveltraj) là Linear Segments with Parabolic Blends (LSPB). Thay vì thay đổi vận tốc liên tục như đa thức bậc 5, phương pháp này chia quỹ đạo làm 3 giai đoạn:

(1) Giai đoạn tăng tốc đều (Constant Acceleration): Vận tốc tăng tuyến tính từ 0 đến v_{max}.
(2) Giai đoạn vận tốc đều (Constant Velocity): Duy trì v_{max}.
(3) Giai đoạn giảm tốc đều (Constant Deceleration): Vận tốc giảm tuyến tính về 0.

- Công thức vận tốc được xác định như sau:

$$\dot{q}(t) = \begin{cases} \alpha t & 0 \le t < t_b \quad (Tăng\ tốc) \\ v_{max} & t_b \le t < T - t_b \quad (Vận\ tốc\ đều) \\ \alpha(T - t) & T - t_b \le t \le T \quad (Giảm\ tốc) \end{cases} \quad (24)$$

- Trong đó α là gia tốc không đổi và t_b là thời gian pha tăng tốc. Phương pháp này thường được dùng trong công nghiệp vì tận dụng tối đa khả năng vận tốc của động cơ nhưng vẫn tránh được sự thay đổi đột ngột.

Động học thuận và sai lệch trong không gian Cartesian

- Mặc dù quỹ đạo q(t) đảm bảo tính trơn trong không gian khớp, chuyển động thực tế của khâu cuối P(t) trong không gian làm việc lại chịu chi phối bởi hàm động học thuận phi tuyến:

$$P(t) = FK\big(q(t)\big) \quad (25)$$

- Do $FK(\ )$ chứa các hàm lượng giác, ánh xạ từ đường thẳng trong không gian khớp sẽ tạo thành đường cong trong không gian Cartesian. Như thể hiện trong kết quả mô phỏng, quỹ đạo thực tế bị uốn cong thành hình cung và các góc Euler thay đổi phi tuyến dọc theo đường đi. Điều này đòi hỏi việc kiểm tra va chạm phải được thực hiện trên toàn bộ quỹ đạo nội suy P(t), chứ không chỉ tại các điểm nút.

Tối ưu hóa tiêu chuẩn độ giật (Minimum Jerk)

- Cơ sở lý thuyết cho việc ưu tiên sử dụng đa thức bậc cao nằm ở việc tối ưu hóa độ giật (Jerk) – đạo hàm bậc ba của vị trí:

$$j(t) = \frac{d^3q(t)}{dt^3} = \dddot{q}(t) \tag{26}$$

- Quỹ đạo đa thức bậc 5 thỏa mãn điều kiện giảm thiểu hàm mục tiêu năng lượng về độ giật:

$$J = \int_0^T \|\ddot{q}(t)\|^2 dt \rightarrow min \tag{27}$$

- Việc cực tiểu hóa J giúp triệt tiêu các thay đổi gia tốc đột ngột, từ đó giảm thiểu rung động cơ khí và đảm bảo bộ điều khiển bám quỹ đạo chính xác hơn.

2.3.2. Chuyển động trong không gian làm việc (Cartesian Space Motion)

- Trong nhiều ứng dụng thực tế như hàn, sơn, cắt laser,… robot được yêu cầu di chuyển khâu chấp hành cuối (end-effector) theo một đường thẳng tuyệt đối nối giữa hai điểm trong không gian làm việc (Task Space). Khác với chuyển động trong không gian khớp (nơi quỹ đạo thực tế bị cong do tính phi tuyến), phương pháp quy hoạch quỹ đạo Cartesian thực hiện nội suy trực tiếp trên các ma trận biến đổi đồng nhất SE(3).

- Tuy nhiên, việc nội suy tuyến tính đơn thuần giữa hai tư thế đầu T_{start} và cuối T_{end} thường dẫn đến vận tốc không đổi trong suốt hành trình, gây ra hiện tượng gián đoạn vận tốc (vận tốc thay đổi tức thời từ 0 lên hằng số) tại các điểm khởi động và dừng lại. Điều này tạo ra gia tốc vô cùng lớn về mặt lý thuyết và gây rung động mạnh cho robot.

Kỹ thuật biến đổi tỷ lệ thời gian (Time Scaling)

- Để giải quyết vấn đề về tính mượt của chuyển động, báo cáo áp dụng kỹ thuật Biến đổi tỷ lệ thời gian (Time Scaling). Thay vì sử dụng thời gian thực t, quỹ đạo được tham số hóa thông qua một biến vô hướng chuẩn hóa $s(t) \in [0,1]$. Hàm nội suy tư thế T(s) được xác định dựa trên tham số s:

$$T(t) = \text{Interpolate}\big(T_{start}, T_{end}, s(t)\big) \tag{28}$$

- Trong đó, hàm s(t) không phải là hàm tuyến tính ($s \neq t/T_{total}$) mà được tạo ra từ các bộ tạo quỹ đạo trơn như Minimum Jerk (đa thức bậc 5) hoặc Biên dạng hình thang. Khi đó, vận tốc và gia tốc trong không gian làm việc được kiểm soát thông qua chuỗi đạo hàm (Chain Rule):

$$v(t) = \frac{dT}{ds} \cdot \dot{s}(t) \tag{29}$$

$$a(t) = \frac{d^2T}{ds^2} \cdot \dot{s}(t)^2 + \frac{dT}{ds} \cdot \ddot{s}(t) \tag{30}$$

- Việc sử dụng $\dot{s}(0) = \dot{s}(T) = 0$ và $\ddot{s}(0) = \ddot{s}(T) = 0$ đảm bảo robot khởi động và dừng êm ái ngay cả khi di chuyển theo đường thẳng.

- Xử lý đa nghiệm trong Động học ngược liên tục.

- Sau khi xác định được quỹ đạo Cartesian T(t), thách thức tiếp theo là chuyển đổi ngược lại sang không gian khớp để điều khiển động cơ:

$$q(t) = IK\big(T(t)\big) \tag{31}$$

- Do bài toán Động học ngược (Inverse Kinematics - IK) thường tồn tại đa nghiệm (multiple solutions) cho cùng một tư thế, việc chọn nghiệm ngẫu nhiên có thể dẫn đến hiện tượng "nhảy khớp" (joint jumping) – robot thay đổi cấu hình đột ngột gây nguy hiểm.

- Để khắc phục, giải thuật lựa chọn nghiệm tối ưu được áp dụng dựa trên nguyên lý cực tiểu hóa khoảng cách Euclidean trong không gian khớp. Tại mỗi bước thời gian t_k, nghiệm $q(t_k)$ được chọn sao cho sai lệch so với cấu hình trước đó $q(t_{k-1})$ là nhỏ nhất:

$$q(t_k) = \arg \min_{q \in \text{Solutions}} \big|\big|q - q(t_{k-1})\big|\big|_2 \tag{32}$$

- Trong mô phỏng, hàm ikineTraj (hoặc các tùy chọn trong inverseKinematics) thực hiện cơ chế này tự động, đảm bảo chuỗi giá trị khớp q(t) thu được là liên tục và khả thi về mặt vật lý.

2.4. Manipulator velocity

2.4.1. Ma trận Jacobian

- Ma trận Jacobian là một khái niệm toán học mở rộng từ đạo hàm, dùng để mô tả mối quan hệ vi phân giữa vận tốc trong không gian cấu hình (joint space) và vận tốc trong không gian tác vụ (task space).

- Trong robot học, Jacobian của robot (manipulator Jacobian) được định nghĩa là:

$$J(q) = \frac{\partial \xi}{\partial q} \tag{33}$$

Trong đó:

- $q \in \mathbb{R}^N$: vector tọa độ khớp.
- $\xi \in \mathbb{R}^M$: vector biểu diễn vị trí và hướng của đầu công tác trong không gian tác vụ.
- N: số bậc tự do của robot.
- M: chiều của không gian tác vụ (thường $M = 6$ trong không gian 3D).

- Mối quan hệ giữa vận tốc khớp và vận tốc đầu công tác được mô tả bởi phương trình:

$$v = J(q)\dot{q} \tag{34}$$

Trong đó:

- $v \in \mathbb{R}^M$: vận tốc không gian (spatial velocity) của đầu công tác, bao gồm:
  - Thành phần quay: $\omega = (\omega_x, \omega_y, \omega_z)$
  - Thành phần tịnh tiến: $\dot{p} = (v_x, v_y, v_z)$
- $\dot{q} \in \mathbb{R}^N$: vận tốc khớp.

- Ma trận Jacobian biểu diễn các thông số như:

- Mỗi cột của Jacobian tương ứng với ảnh hưởng của một khớp lên vận tốc đầu công tác.
- Nếu cột thứ i của $J(q)$ là J_i, thì:

$$v = \sum_{i=1}^{N} J_i(q)\dot{q}_i \tag{35}$$

- Hạng (rank) của Jacobian quyết định số bậc tự do có thể điều khiển được tại vị trí hiện tại.

2.4.2. Phân tích chuyển động qua điểm kỳ dị (Motion Through Singularity)

- Trong không gian cấu hình của các robot chuỗi động học mở, tồn tại những cấu hình đặc biệt được gọi là điểm kỳ dị. Về mặt toán học, đây là trạng thái mà ma trận Jacobian J(q) – ma trận ánh xạ vận tốc khớp sang vận tốc Cartesian – bị suy biến hạng (rank deficient):

$$\det\big(J(q)\big) = 0 \tag{36}$$

- Tại các cấu hình này, robot mất đi ít nhất một bậc tự do, đồng nghĩa với việc khâu chấp hành cuối không thể di chuyển theo một hướng nhất định trong không gian làm việc, bất kể các khớp chuyển động như thế nào. Các dạng kỳ dị phổ biến bao gồm kỳ dị biên (boundary singularity - khi robot vươn hết tầm) và kỳ dị nội tại (internal singularity - khi các trục quay của robot thẳng hàng nhau, gây ra hiện tượng khóa Gimbal).

Sự bất ổn định của phương pháp Động học ngược giải tích

- Khi quy hoạch quỹ đạo Cartesian đi qua hoặc lân cận vùng kỳ dị, việc sử dụng phương pháp Động học ngược giải tích (Analytical Inverse Kinematics) thường dẫn đến sự bất ổn định nghiêm trọng. Xét phương trình vi phân động học:

$$\dot{x} = J(q)\dot{q} \Rightarrow \dot{q} = J^{-1}(q)\dot{x} \tag{37}$$

- Khi robot tiến gần điểm kỳ dị ($\det(J) \to 0$), các phần tử của ma trận nghịch đảo J^{-1} sẽ tiến tới vô cùng. Hệ quả là để duy trì một vận tốc đầu cuối $\dot{x}$ hữu hạn và không đổi, bộ điều khiển yêu cầu các khớp phải quay với vận tốc vô cùng lớn ($\dot{q} \to \infty$). Trên thực tế, hiện tượng này biểu hiện qua việc các khớp dao động mạnh hoặc quay ngược chiều nhau với tốc độ cao (dù đầu robot gần như đứng yên), dẫn đến bão hòa mô-men xoắn và gây nguy hiểm cho hệ thống cơ khí.

2.4.3. Đánh giá độ linh hoạt (Manipulability Analysis)

- Để định lượng khả năng vận hành và phát hiện sớm nguy cơ kỳ dị trên quỹ đạo, báo cáo sử dụng chỉ số độ linh hoạt (Manipulability measure) w, được định nghĩa bởi Yoshikawa:

$$w(q) = \sqrt{\det(J(q)J(q)^T)} \tag{38}$$

- Khi robot ở trạng thái bình thường, w đạt giá trị cao, thể hiện khả năng di chuyển linh hoạt theo mọi hướng.
- Khi $w \to 0$, robot tiến gần đến điểm kỳ dị.

- Việc giám sát chỉ số w dọc theo quỹ đạo cho phép bộ quy hoạch động lực học chủ động chuyển đổi thuật toán điều khiển hoặc điều chỉnh đường đi để tránh các vùng suy biến, đảm bảo tính an toàn và liên tục cho hệ thống robot công nghiệp.

2.4.4. Phân tích vấn đề ma trận Jacobian không vuông

- Ma trận Jacobian là ma trận có kích thước $\dim \mathcal{T} \times \dim \mathcal{C}$, trong đó:

- $\dim \mathcal{T}$: số chiều không gian tác vụ (task space), thường 6 trong không gian 3D (3 tịnh tiến + 3 quay).

- $\dim \mathcal{C}$: số chiều không gian cấu hình (số khớp N).

- Vì thế, ma trận Jacobian không vuông khi:

$$N \neq \dim \mathcal{T}$$

- Ma trận Jacobian không vuông dẫn đến các hệ quả như:

1. Robot thiếu cơ cấu chấp hành (Underactuated): $N < \dim \mathcal{T}$
2. Robot dư thừa cơ cấu chấp hành (Overactuated/Redundant): $N > \dim \mathcal{T}$

Bảng 4. Hệ quả của robot với các trường hợp khớp khác nhau dẫn đến ma trận Jacobian khác nhau, từ đó gây ra các trường hợp xấu trong việc tìm nghiệm.

Loại robot	Số khớp N	Kích thước Jacobian	Đặc điểm
Underactuated	$N < 6$	$6 \times N$	Thiếu khớp để điều khiển đầy đủ 6 bậc tự do trong không gian tác vụ. Hệ phương trình quá xác định (over-determined).
Fully-actuated	$N = 6$	6×6	Jacobian vuông, có thể điều khiển đầy đủ 6 bậc tự do nếu không kỳ dị.
Overactuated/Redundant	$N > 6$	$6 \times N$	Thừa khớp so với yêu cầu tối thiểu. Hệ phương trình thiếu xác định (under-determined), có vô số nghiệm.

- Từ bảng trên ta có thể thấy, với ma trận Jacobian không vuông thì hệ quả ảnh hưởng xấu đến việc điều khiển và xác định vị trí của robot.

- Vì vậy, với các bài toán ma trận Jacobian không vuông, ta phải có các hướng giải quyết khác để phù hợp tình huống.

- Xét phương trình cơ bản:

$$v = J(q)\dot{q} \tag{39}$$

- $v \in \mathbb{R}^6$: vận tốc không gian mong muốn.
- $\dot{q} \in \mathbb{R}^N$: vận tốc khớp cần tìm.

a. Trường hợp Underactuated ($N < 6$)

- Bài toán quá xác định: số phương trình > số ẩn.
- Không có nghiệm chính xác cho mọi v tùy ý.
- Giải pháp tối ưu theo bình phương tối thiểu (least-squares):

$$\dot{q} = J^+ v \tag{40}$$

với $J^+ = (J^T J)^{-1} J^T$ là giả nghịch đảo trái (left pseudoinverse).

- Nhược điểm: Nghiệm này không thỏa mãn chính xác v, mà chỉ tối thiểu hóa sai số $\| J\dot{q} - v \|^2$.
- Ví dụ:
  - Robot 2 khâu phẳng ($N = 2$) với không gian tác vụ 3 chiều (ω_z, v_x, v_y).
  - Nếu chỉ quan tâm đến v_x, v_y, có thể xóa hàng tương ứng với ω_z để Jacobian trở thành vuông 2×2.

b. Trường hợp Overactuated ($N > 6$)

- Bài toán thiếu xác định: số phương trình < số ẩn → vô số nghiệm.
- Nghiệm có chuẩn nhỏ nhất (minimum-norm solution):

$$\dot{q} = J^+ v \tag{41}$$

với $J^+ = J^T (J J^T)^{-1}$ là giả nghịch đảo phải (right pseudoinverse).

- Nghiệm tổng quát hơn (có thể thêm thành phần null space):

$$\dot{q} = J^+ v + P(q)\dot{q}_0$$

Trong đó:

  - $P(q) = I_{N\times N} - J^+ J$ là ma trận chiếu lên không gian null.
  - $\dot{q}_0$ là vận tốc khớp tùy ý.
  - $P(q)\dot{q}_0$ không ảnh hưởng đến vận tốc đầu công tác (vì $J \cdot P\dot{q}_0 = 0$).

Không gian Null (Null Space) và điều khiển dư thừa

- Null space của J là tập các $\dot{q}$ sao cho:

$$J\dot{q} = 0 \tag{42}$$

- Ý nghĩa: Các chuyển động trong null space không làm thay đổi vị trí/hướng đầu công tác, nhưng có thể dùng để:
  - Tránh vật cản.
  - Tránh giới hạn khớp.
  - Tối ưu hóa tư thế robot.

Phương pháp "bình phương hóa" Jacobian (Squaring Up)

- Khi Jacobian không vuông, có thể xóa hàng hoặc cột để tạo ma trận vuông:

Bảng 5. Phương pháp bình phương hóa với 2 loại robot không vuông

Phương pháp	Thực hiện	Ứng dụng
Xóa hàng (Underactuated)	Bỏ bớt một số bậc tự do trong không gian tác vụ không quan trọng.	Ưu tiên kiểm soát một số hướng chuyển động cụ thể.
Xóa cột (Overactuated)	Khóa một số khớp (coi như cố định).	Giảm bậc tự do dư thừa, đơn giản hóa bài toán.

- Áp dụng vào Robot Overactuated Panda 7 khâu

- Jacobian 6×7, null space 1 chiều.
- Nghiệm vận tốc khớp:

$$\dot{q} = J^{+}v + N\alpha \tag{43}$$

với N là cơ sở null space, α là hệ số tự do.

- Có thể dùng α để tránh vật cản mà không ảnh hưởng đến đầu công tác.

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

3.1. Phân tích và Lựa chọn Cấu hình Robot

3.1.1. Phân tích Yêu cầu Không gian Nhiệm vụ

- Để xác định số bậc tự do tối thiểu cần thiết cho robot vẽ chữ, ta phân tích các thành phần chuyển động độc lập mà nhiệm vụ yêu cầu:

Yêu cầu về Vị trí (x, y, z):

- Tọa độ (x, y): Bắt buộc phải thay đổi liên tục để tạo hình nét chữ trên mặt phẳng 2D.
- Tọa độ (z): Cần thiết cho thao tác nhấc bút (pen-up) và hạ bút (pen-down) giữa các nét vẽ.

→ Số biến vị trí cần thiết: 3.

Yêu cầu về Hướng (Orientation):

- Nhiệm vụ vẽ chữ yêu cầu cây bút luôn giữ phương thẳng đứng (hoặc một góc nghiêng cố định) so với mặt giấy.
- Do cây bút có tính chất đối xứng trục (Axial Symmetry), góc xoay quanh thân bút (Yaw) không ảnh hưởng đến kết quả đầu ra.

→ Hướng là một ràng buộc cố định, không phải là biến số cần điều khiển trong quỹ đạo nhiệm vụ.

- Từ phân tích trên, ta xác định được số chiều tối thiểu của không gian nhiệm vụ chức năng ($\mathcal{T}_{min}$):

$$\dim \mathcal{T}_{min} = 3 \quad (\mathrm{x}, \mathrm{y}, \mathrm{z}) \tag{44}$$

3.1.2. Điều kiện Cần và Kết luận Lý thuyết

- Dựa trên điều kiện bao phủ không gian làm việc: $\dim \mathcal{C} \geq \dim \mathcal{T}$.Ta có thể kết luận rằng: Để thực hiện được nhiệm vụ vẽ, một robot cần có ít nhất 3 bậc tự do $\dim \mathcal{C} \geq 3$.Về mặt lý thuyết, các cấu hình robot 3-DoF đã đủ điều kiện cần để hoàn thành tác vụ này.

3.1.3. Đề xuất Mô hình

- Mặc dù cấu hình 3-DoF là đủ về mặt lý thuyết, nhóm thực hiện quyết định lựa chọn thiết kế Robot 4 Bậc Tự do (4-DoF). Việc bổ sung thêm một bậc tự do so với mức tối thiểu mang lại các ưu điểm kỹ thuật sau:

- Tách biệt chuyển động (Decoupling): Với các tay máy nối tiếp dạng khớp quay (Revolute), chuyển động vươn xa/thu gần của cánh tay thường kéo theo sự thay đổi góc nghiêng của khâu cuối. Bậc tự do thứ 4 đóng vai trò bù trừ chuyển động này.
- Đảm bảo ràng buộc hướng: Khớp thứ 4 (thường là khớp cổ tay - Pitch) cho phép robot chủ động điều chỉnh góc nghiêng của bút độc lập với vị trí đầu bút. Điều này đảm bảo cây bút luôn vuông góc với mặt bàn tại mọi vị trí trong vùng làm việc, giúp nét vẽ đồng đều và ổn định hơn so với robot 3-DoF khớp quay thuần túy.

➔ **Kết luận: Nhóm chọn mô hình 4-DoF để đảm bảo sự cân bằng giữa tính khả thi của nhiệm vụ và chất lượng của nét vẽ.**

3.2. Thiết kế hệ thống

3.2.1. Thiết kế cơ khí

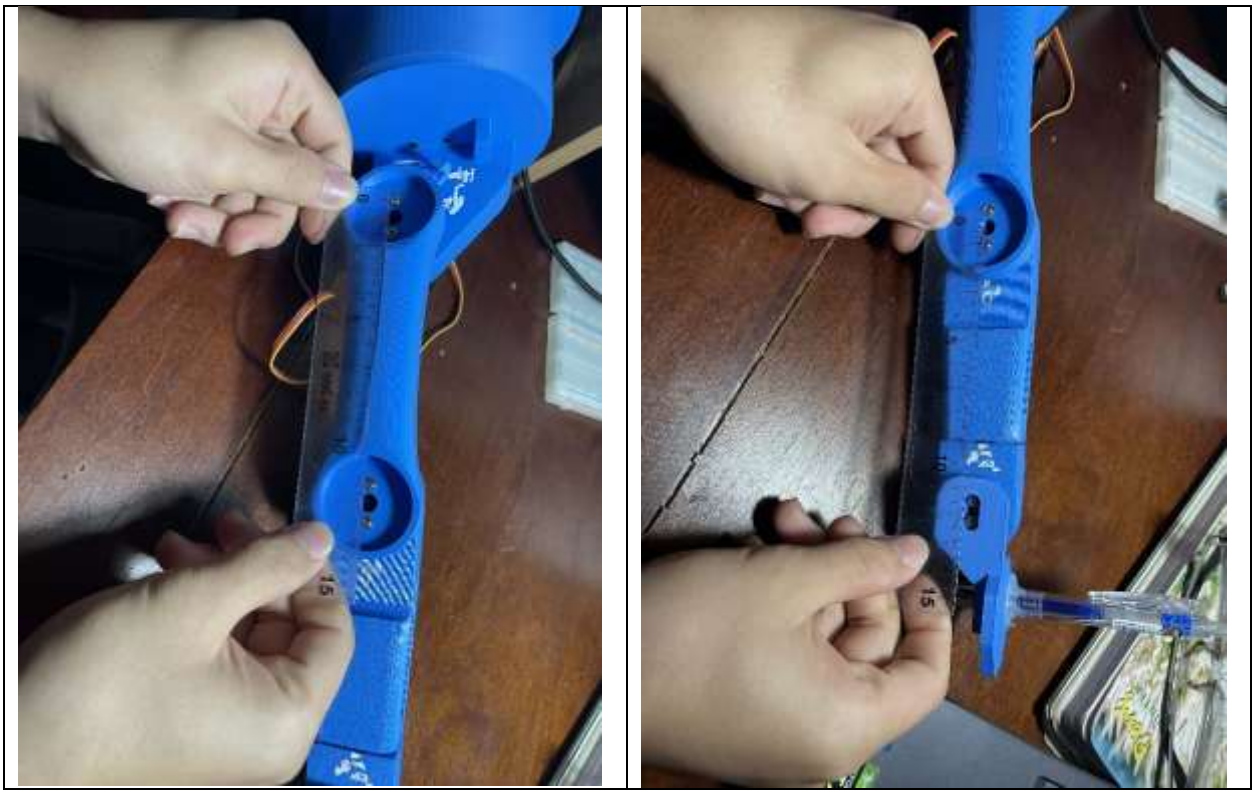
- Dựa vào model 3D có sẵn nhóm tiến hành đo lại từng link và tạo ra mô hình trong Matlab phục vụ việc mô phỏng.

- Độ dài từng link của robot:

Bảng 6. Độ dài giữa các khớp robot

Link 1	Link 2	Link 3	Link 4
0.12 m	0.118 m	0.083 m	0.045m

- Thực hiện: Độ dài mỗi link nhóm đo từ tâm khớp (n) đến tâm khớp (n+1).

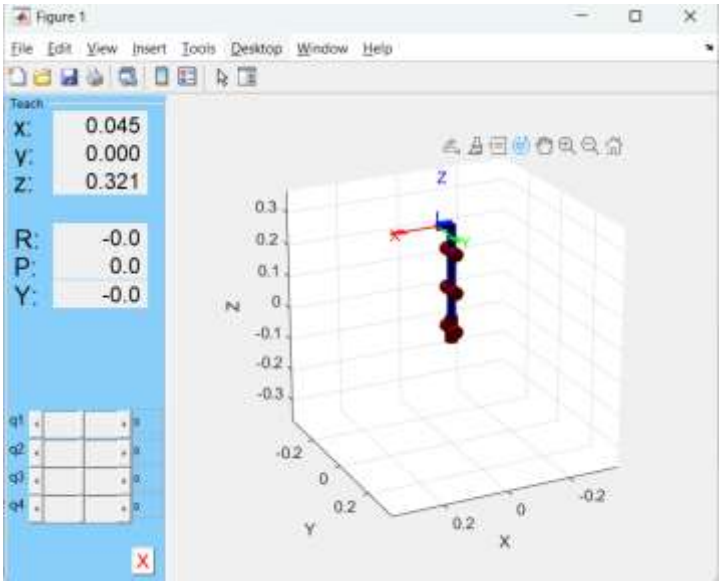


Hình 3: Quá trình đo từng link lấy số liệu

Thiết kế mô phỏng

- Tạo mô hình Kinematic cho robot: Để xây dựng mô phỏng động học cho robot 4 bậc tự do (4-DOF), ta cần thiết lập mô hình Kinematic dựa trên chuỗi các phép biến đổi hình học của hệ khớp. Vì robot được mô tả bằng chuỗi ETS3 và các tham số chiều dài khâu a1, a2, a3 và a4 được xác định dựa trên đặc tính thiết kế của hệ thống, nên có thể đảm bảo mô hình

phản ánh đúng cấu trúc cơ khí của robot. Hình 4 minh họa chuỗi các phép biến đổi được sử dụng để xây dựng mô hình động học.



Hình 4. Trình bày mô hình kinematic của robot được xây dựng từ chuỗi ETS3.

- Sau khi xây dựng chuỗi ETS3 hoàn chỉnh, mô hình được chuyển đổi sang dạng rigidBodyTree để phục vụ cho việc tính toán động học thuận/nghịch cũng như mô phỏng chuyển động trong MATLAB. Việc sử dụng định dạng này giúp các vector trạng thái khớp được biểu diễn theo dạng [q1 q2 q3 q4], phù hợp với hầu hết các hàm xử lý của MATLAB. Đồng thời, đảm bảo sự tương thích với các thuật toán quỹ đạo, hiển thị mô hình và các hàm tính toán động học, từ đó tránh các lỗi định dạng khi truyền dữ liệu.
- Biểu diễn chuỗi biến đổi ETS3 dưới dạng công thức:

$$e = R_z(q_1)\, R_y(q_2)\, T_z(a_1)\, R_y(q_3)\, T_z(a_2)\, R_y(q_4)\, T_z(a_3)\, T_x(a_4) \quad (45)$$

Trong đó:

- $R_z(q_1)$: quay quanh trục zmột góc q_1.
- $R_y(q_2), R_y(q_3), R_y(q_4)$: quay quanh trục ymột góc tương ứng.
- $T_z(a_1), T_z(a_2), T_z(a_3)$: tịnh tiến theo trục z.
- $T_x(a_4)$: tịnh tiến theo trục x.

- Với các tham số hình học:

$$a_1 = 0.12 \text{ m}, a_2 = 0.118 \text{ m}, a_3 = 0.083 \text{ m}, a_4 = 0.045 \text{ m}$$

- Dạng ma trận thu gọn

$$
{}^{0}T_{EE} = R_z(q_1)\ R_y(q_2) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_y(q_3) \dots
$$

$$
\dots \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & a_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_y(q_4) \begin{bmatrix} 1 & 0 & 0 & a_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{46}
$$

a) Dữ liệu

Dữ liệu đầu vào là chữ:

- Dữ liệu đầu vào là các ký tự cần vẽ, được biểu diễn dưới dạng các nét (stroke) theo font Hershey trong RVC Toolbox, gồm các điểm 2D (x, y) trong khoảng [0,1]. Với NaN dùng để phân tách các nét.

- Chương trình tải dữ liệu font, truy xuất ký tự qua biến ch, sau đó co giãn các điểm theo hệ số scale để phù hợp kích thước robot và mở rộng sang dạng 3D bằng cách thêm z = 0.

- Tại các vị trí NaN, tọa độ điểm trước đó được giữ nguyên và gán z = penUp để robot nâng bút khi di chuyển giữa các nét. Kết quả là quỹ đạo 3D hoàn chỉnh của ký tự, sẵn sàng cho nội suy và điều khiển robot.

Dữ liệu đầu vào là hình ảnh:

- Trong quá trình chuyển đổi ảnh line-art thành đường đi cho robot vẽ, bước tiền xử lý quyết định độ chính xác của hình dạng tái tạo.

- Mục tiêu là trích xuất các nét cốt lõi dưới dạng tập điểm liên tục và loại bỏ nhiễu như độ dày nét, thay đổi độ sáng hay cấu trúc pixel không đều.

- Ảnh được chuyển sang grayscale và nhị phân hóa bằng phương pháp Otsu để tách nền và nét vẽ, sau đó áp dụng phép toán hình thái bwmorph(thin) nhằm làm mỏng nét về dạng skeleton 1 pixel nhưng vẫn giữ nguyên hình dạng.

- Cuối cùng, các nét được truy vết thành từng đường liên tục bằng thuật toán bwboundaries với liên kết 8-láng giềng, thu được các contour độc lập có thể chuyển trực tiếp sang tọa độ robot.

- Pipeline này dựa trên ba trụ cột chính: grayscale & Otsu, bwmorph(thin) và bwboundaries, tạo nền tảng chuyển ảnh 2D thành đường đi robot trong không gian vật lý.

Mô tả Thuật toán Phân ngưỡng Toàn cục (Global Thresholding) trong graythresh

- Giả sử ảnh đầu vào được lượng tử hóa thành L mức xám (thông thường L=256, với các mức từ 0 đến L-1).

- Gọi n_i là số lượng điểm ảnh có mức xám i.
- Tổng số điểm ảnh trong ảnh là $N = \sum_{i=0}^{L-1} n_i$.
- Xác suất xuất hiện (probability) của mức xám i được tính bằng biểu đồ histogram chuẩn hóa:

$$p_i = \frac{n_i}{N}, \quad p_i \geq 0, \quad \sum_{i=0}^{L-1} p_i = 1 \tag{47}$$

Phân chia Lớp (Class Separation): Giả sử ta chọn một ngưỡng k($0 \leq k < L-1$) để chia các điểm ảnh thành hai lớp:

- Lớp C_1 (Nền/Background): Chứa các điểm ảnh có mức xám từ [0, k].
- Lớp C_2 (Đối tượng/Foreground): Chứa các điểm ảnh có mức xám từ [k+1, L-1].

Tính toán Trọng số và Trung bình Lớp: Tại mỗi ngưỡng k, thuật toán tính toán các đại lượng thống kê cho hai lớp:

- Trọng số (Xác suất tích lũy) của lớp $C_1 v$à C_2:

$$\omega_1(k) = \sum_{i=0}^{k} p_i \quad \text{và} \quad \omega_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - \omega_1(k) \tag{48}$$

- Giá trị trung bình (Mean) của lớp C_1 và C_2:

$$\mu_1(k) = \frac{\sum_{i=0}^{k} i \cdot p_i}{\omega_1(k)} \quad \text{và} \quad \mu_2(k) = \frac{\sum_{i=k+1}^{L-1} i \cdot p_i}{\omega_2(k)} \tag{49}$$

- Giá trị trung bình toàn cục (Global Mean) của toàn bộ ảnh:

$$\mu_T = \sum_{i=0}^{L-1} i \cdot p_i = \omega_1(k)\mu_1(k) + \omega_2(k)\mu_2(k) \tag{50}$$

Tiêu chí Tối ưu hóa (Optimization Criterion): Mục tiêu của Otsu là tìm ngưỡng k^* để tối đa hóa phương sai giữa hai lớp (Between-class Variance), ký hiệu là σ_B^2. Việc tối đa hóa đại lượng này tương đương với việc tối thiểu hóa phương sai nội bộ lớp:

Công thức tính phương sai giữa hai lớp tại ngưỡng k:

$$\sigma_B^2(k) = \omega_1(k)\omega_2(k)[\mu_1(k) - \mu_2(k)]^2 \quad (51)$$

Thuật toán sẽ quét qua mọi giá trị k có thể (từ 0 đến L-1) và chọn k^* sao cho:

$$\sigma_B^2(k^*) = \max_{0 \le k < L-1} \sigma_B^2(k) \quad (52)$$

Chuẩn hóa Kết quả (MATLAB Specific): Sau khi tìm được ngưỡng tối ưu k^* (là một giá trị nguyên trong khoảng [0, L-1]), hàm graythresh của MATLAB sẽ chuẩn hóa giá trị này về khoảng [0, 1] để tương thích với các định dạng dữ liệu double hoặc logical:

$$T = \frac{k^*}{L-1} \quad (53)$$

Mô tả Thuật toán Làm mỏng (Thinning Algorithm) trong bwmorph

- Hàm bwmorph với tùy chọn 'thin' thực hiện việc làm mỏng các đối tượng trong ảnh nhị phân xuống còn các đường nét đơn (bề rộng 1 pixel). Thuật toán này đảm bảo tính liên thông của đối tượng và bảo toàn cấu trúc tô pô (số Euler) của ảnh.

- Thuật toán này vận hành theo phương pháp lặp, trong đó mỗi vòng lặp lớn bao gồm hai tiểu bước lặp (sub-iterations). Quá trình sẽ tiếp diễn cho đến khi ảnh không còn thay đổi (trạng thái hội tụ) hoặc đạt đến số vòng lặp n quy định.

- Ký hiệu và Định nghĩa Lân cận: Giả sử p là điểm ảnh (pixel) đang xét có giá trị bằng 1. Xét lân cận 3×3 quanh p, các điểm lân cận được ký hiệu từ x_1đến x_8 theo chiều ngược chiều kim đồng hồ, bắt đầu từ lân cận phía Đông:

$$\begin{matrix} x_4 & x_3 & x_2 \\ x_5 & p & x_1 \\ x_6 & x_7 & x_8 \end{matrix}$$

- x_1: Đông (East)
- x_2: Đông Bắc (North-East)
- x_3: Bắc (North)
- ...
- x_8: Đông Nam (South-East)

- Quy trình Lặp: Thuật toán quyết định việc xóa một điểm ảnh p (chuyển giá trị từ 1 thành 0) dựa trên các điều kiện kiểm tra. Một vòng lặp đầy đủ bao gồm 2 bước:

(1) Tiểu bước 1: Điểm p bị xóa nếu và chỉ nếu thỏa mãn đồng thời ba điều kiện: G1, G2, và G3.

(2) Tiểu bước 2: Điểm p bị xóa nếu và chỉ nếu thỏa mãn đồng thời ba điều kiện: G1, G2, và G3'.

- Các Điều kiện Kiểm tra (Conditions):

- Điều kiện G1: Kiểm tra tính liên thông cục bộ, điều kiện này dựa trên số lượng sự chuyển tiếp giữa các điểm ảnh lân cận (crossing number).

$$G1: X_H(p) = 1 \tag{54}$$

Trong đó:

$$X_H(p) = \sum_{i=1}^{4} b_i \tag{55}$$

Với b_i được xác định như sau:

$$b_i = \begin{cases} 1, & nếu\ x_{2i-1} = 0\ và\ x_{2i} = 1\ hoặc\ x_{2x+1} = 1 \\ 0, & trường\ hợp\ khác \end{cases} \tag{56}$$

(Lưu ý: Khi i=4, x_{2i+1} tương ứng với x_1).

- Điều kiện G2: Giới hạn số lượng lân cận. Điều kiện này đảm bảo điểm bị xóa không phải là điểm cuối (endpoint) quan trọng hoặc điểm cô lập, nhưng cũng không phải là điểm bên trong khối quá đặc.

$$G2: 2 \le \min\{n_1(p), n_2(p)\} \le 3 \tag{57}$$

Trong đó:

- $n_1(p) = \sum_{k=1}^{4}(x_{2k-1} \vee x_{2k})$
- $n_2(p) = \sum_{k=1}^{4}(x_{2k} \vee x_{2k+1})$

(Phép toán $\vee$ là phép hoặc logic).

- Điều kiện G3 và G3': Các ràng buộc hướng, hai điều kiện này được sử dụng luân phiên trong hai tiểu bước để bào mòn đối tượng từ các hướng khác nhau mà không làm đứt gãy khung xương.

  - Trong tiểu bước 1 (G3):

$$(x_2 \vee x_3 \vee \overline{x_8}) \wedge x_1 = 0 \tag{58}$$

(Điều này ngụ ý: Nếu x_1=1, thì các lân cận x_2, x_3 phải bằng 0 và x_8 phải bằng 1).

- Trong tiểu bước 2 (G3'):

$$(x_6 \vee x_7 \vee \overline{x_4}) \wedge x_5 = 0 \tag{59}$$

(Tương tự cho hướng đối diện).

Mô tả Thuật toán Dò biên (Boundary Tracing) trong bwboundaries

- Hàm bwboundaries trong MATLAB được sử dụng để xác định tọa độ các điểm biên của đối tượng và các lỗ hổng (holes) bên trong đối tượng trên ảnh nhị phân. Thuật toán cốt lõi được sử dụng là Thuật toán Dò biên lân cận Moore (Moore-Neighbor Tracing) được tinh chỉnh bởi Tiêu chuẩn dừng Jacob (Jacob's stopping criteria).

Ký hiệu và Định nghĩa

- Ảnh nhị phân đầu vào: Các điểm ảnh có giá trị 1 (foreground) là đối tượng, giá trị 0 (background) là nền.
- Lân cận Moore (Moore Neighborhood): Với mỗi điểm ảnh p, lân cận Moore bao gồm 8 điểm ảnh xung quanh nó (tương tự như lân cận 8 điểm - 8-connectivity):

$$\begin{matrix} n\_4 & n\_3 & n\_2 \\ n\_5 & p & n\_1 \\ n\_6 & n\_7 & n\_8 \end{matrix}$$

Trong đó các điểm được quét theo chiều kim đồng hồ.

Quy trình Dò biên: Thuật toán hoạt động theo cơ chế "người mù dò đường" (bám theo bờ tường bên trái hoặc phải). Quy trình cụ thể như sau:

(1) Tìm điểm khởi đầu (s): Quét ảnh theo từng hàng (từ trên xuống dưới, từ trái sang phải) để tìm điểm ảnh có giá trị 1 đầu tiên. Điểm này được gọi là điểm khởi đầu s.

(2) Xác định hướng lùi (Backtracking): Để bắt đầu dò, thuật toán cần một điểm mốc nằm trong vùng nền (giá trị 0) ngay trước điểm s. Điểm này được gọi là b (backtrack pixel). Thông thường nếu quét từ trái sang, b là điểm bên trái của s.

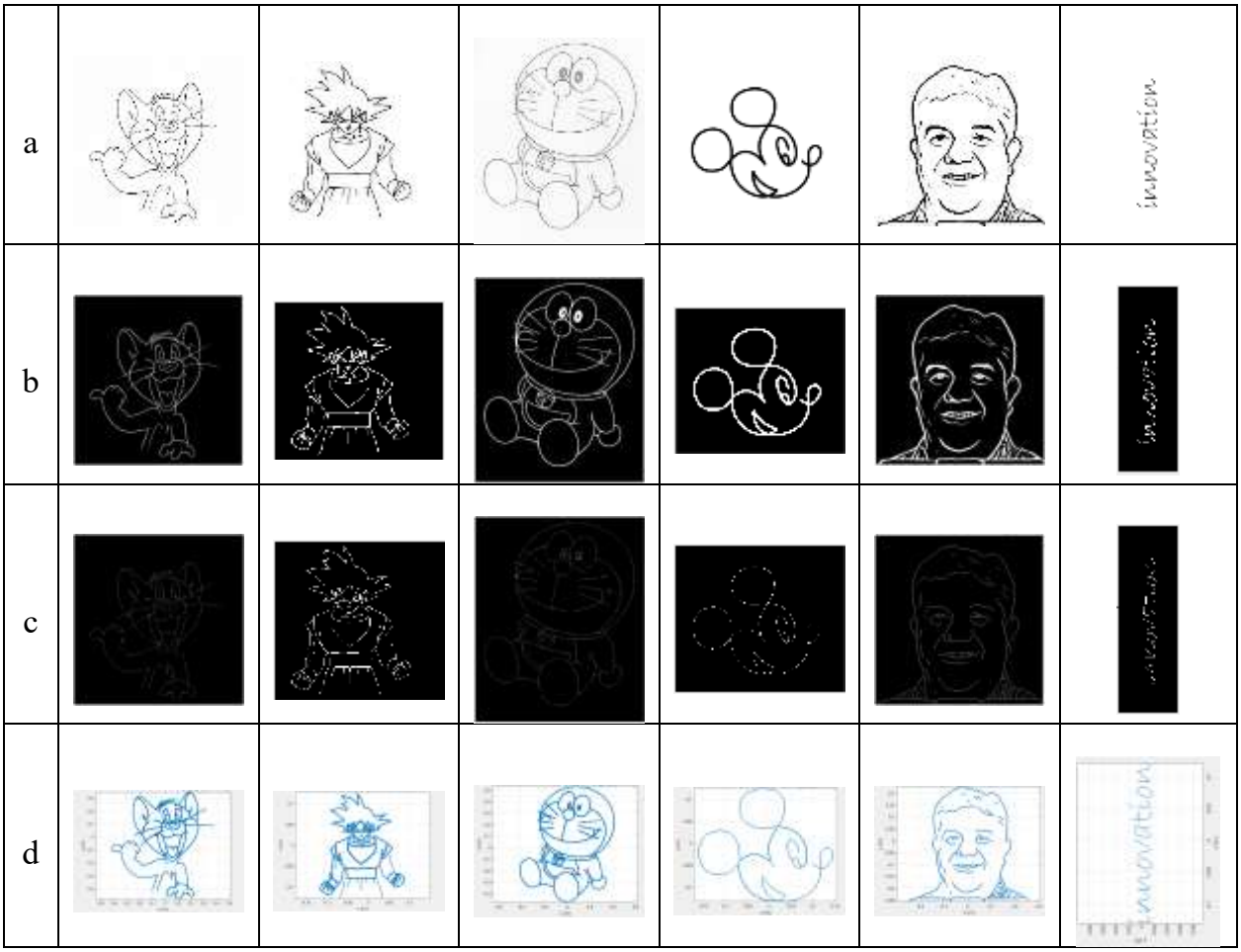
(3) Vòng lặp dò tìm: Từ vị trí hiện tại p (ban đầu p=s), xét các lân cận của nó bắt đầu từ hướng của b theo chiều kim đồng hồ:

- Quay vòng qua các điểm lân cận $n_1, n_2, \dots, n_8$.
- Điểm lân cận có giá trị 1 đầu tiên tìm thấy sẽ trở thành điểm biên tiếp theo.
- Cập nhật p thành điểm mới tìm thấy này, và cập nhật b thành điểm lân cận ngay trước đó (điểm 0 cuối cùng vừa quét qua).

- Tiêu chuẩn dừng Jacob (Jacob's Stopping Criteria): Điểm đặc biệt của bwboundaries so với thuật toán Moore cổ điển là cách nó quyết định khi nào dừng lại. Thuật toán Moore cơ bản có thể dừng ngay khi gặp lại điểm khởi đầu s, nhưng điều này có thể gây lỗi với các hình phức tạp

Tiêu chuẩn Jacob quy định: Thuật toán chỉ dừng lại khi điểm khởi đầu s được ghé thăm lần thứ hai theo cùng một hướng đi vào như lần đầu tiên.

Kết quả sau khi tiền xử lý:



Hình 5. Một số đầu ra của ảnh sau khi tiền xử lý

(a. Đầu vào ảnh nét vẽ; b. Ảnh được chuyển sang nhị phân; c. Ảnh được làm mỏng nét với độ mỏng 1 pixel; d. Đường path xy được trích xuất)

b) Trajectory

- Sau khi xây dựng được đường đi (path) rời rạc trong không gian 3D và lưu trong ma trận p, bước tiếp theo là sinh ra quỹ đạo theo thời gian (time-parameterized trajectory) để robot

bám theo. Trong nghiên cứu này, quỹ đạo được tạo bằng hàm mstraj thuộc Robotics Toolbox for MATLAB của Peter Corke.

- Ma trận p chứa các điểm mốc (via-points) trên path, mỗi cột tương ứng với một tọa độ không gian (x, y, z). Khi gọi mstraj(p(:,2:end)', ...), ta sử dụng các điểm từ cột thứ 2 trở đi làm các via-point mà quỹ đạo phải đi qua, trong khi điểm xuất phát được lấy từ cột thứ nhất p(:,1)' (truyền vào đối số q0).

- Tham số thời gian và giới hạn động học:

- dt = 0.02 (s) là bước thời gian lấy mẫu, quyết định chu kỳ cập nhật quỹ đạo. Giá trị này càng nhỏ thì quỹ đạo thu được càng mịn, nhưng chi phí tính toán và lưu trữ tăng lên.
- vmax = [0.05 0.05 0.05] (m/s) là vận tốc tối đa cho từng trục x, y, z. Đây là giới hạn vận tốc mà quỹ đạo không được vượt quá, đảm bảo chuyển động nằm trong khả năng động học của robot.
- accTime = 0.2 (s) là thời gian tăng tốc, tức là khoảng thời gian để vận tốc tăng từ 0 lên đến giá trị làm việc (và tương tự khi giảm tốc). Tham số này giúp quỹ đạo có pha tăng/giảm tốc êm, tránh thay đổi vận tốc đột ngột.

- Theo định nghĩa của hàm mstraj, các giới hạn vận tốc (vmax), bước thời gian (dt) và thời gian tăng tốc (accTime) được sử dụng để tự động xác định số mẫu thời gian cũng như dạng profile vận tốc (thường là tuyến tính theo đoạn và có đoạn chuyển tiếp dưới dạng đa thức).

- Kết quả của lệnh:

traj = mstraj(p(:,2:end)', vmax, [], p(:,1)', dt, accTime);

là ma trận traj có kích thước N x 3, với:

- **N**: số bước thời gian của quỹ đạo (phụ thuộc số điểm trên path, giới hạn vận tốc, dt và accTime).
- Mỗi hàng của traj có dạng [x y z], biểu diễn vị trí trong không gian tại một thời điểm mẫu.
- Các điểm trong traj nối tiếp nhau tạo thành một chuỗi chuyển động liên tục, đi qua các via-point đã cho trong p với vận tốc không vượt quá vmax và có pha tăng/giảm tốc êm.

- Như vậy, sau bước này, biến traj chính là **quỹ đạo thời gian hóa** được suy ra từ path 3D ban đầu. Quỹ đạo này có thể được dùng trực tiếp để:

- Vẽ và mô phỏng chuyển động trong "frame vẽ".
- Làm đầu vào cho bộ điều khiển quỹ đạo của robot (ở mức Cartesian hoặc nội suy tiếp sang không gian khớp).

c) Transform matrix

- Sau khi đã có quỹ đạo bút trong khung chữ, ta sẽ tạo ma trận biến đổi 4×4 (homogeneous transformation matrix) cho từng điểm trên quỹ đạo chữ, nhằm đưa các điểm này từ khung tọa độ chữ sang khung tọa độ base của robot.
- Ta đã có $traj$ có kích thước $N \times 3$ tạo ra ở bước c, đầu ra của bước này là chuỗi các ma trận biến đổi Tp có kích thước $4 \times 4 \times N$ (N đã được giải thích ở bước c).

- Công thức tổng quát, với điểm quỹ đạo thứ i:

$$Tp = T_{offset} \cdot T_{traj,i} \cdot T_{orient} \tag{60}$$

Trong đó:

- $T_{traj,i}(4 \times 4)$: Ma trận biểu diễn vị trí điểm quỹ đạo trong frame chữ.
- $T_{offset}(4 \times 4)$: Ma trận biến đổi từ khung chữ sang robot base.
- $T_{orient}(4 \times 4)$: Ma trận biến đổi định hướng bút.

- Ma trận $T_{traj,i}$ biểu diễn vị trí điểm quỹ đạo frame chữ tại thời điểm i:

$$\begin{bmatrix} I_{3\times3} & \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{61}$$

- Ma trận T_{offset} được định nghĩa trong code như sau:

T_offset = trvec2tform([0.15 0 -0.055]);

➔ Bàn vẽ đặt thấp hơn base robot 5.5cm và dịch ra trước 15cm.

- Ma trận T_{offset} được định nghĩa trong code như sau:

T_orient = axang2tform([0 1 0 pi/2]);

➔ Bút cần vuông góc với mặt phẳng chữ, pitch = 90° quanh trục y.

- Kết quả cuối cùng là pose homogeneous của end-effector trong base robot, ứng với mỗi i trên quỹ đạo, ma trận có dạng:

$$Tp(:,:,i) = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{62}$$

d) Inverse Kinematics

- Trong bước này, mục tiêu là tính các góc khớp của robot 4dof sao cho end-effector đạt đúng các posr 4×4 trên quỹ đạo chữ đã tạo ($Tp(:,:,i)$). Mục tiêu là tìm ra các góc khớp q để đưa vào robot. Robot 4Dof không đủ điều kiện để sử dụng phương pháp close-form nên trong trường hợp này sẽ sử dụng phương pháp numerical để giải quyết. Áp dụng IK với ràng buộc pitch = 90°, còn các thành phần khác linh hoạt để dễ tìm nghiệm.

- Đầu vào của IK:

- Đối tượng robot cần điều khiển (cấu trúc) và tên end-effector trong robot đó.
- $Tp(:,:,i)$ là các ma trận đã được tạo ở bước d.
- $weights = [010111]$ là trọng số thành phần để xác định độ ưu tiên bám orientation và position.
- $cfg0$ là initial guess của cấu hình khớp. Dùng initial guess từ bước trước để làm đầu vào ($cfg0 = cfgSol$)

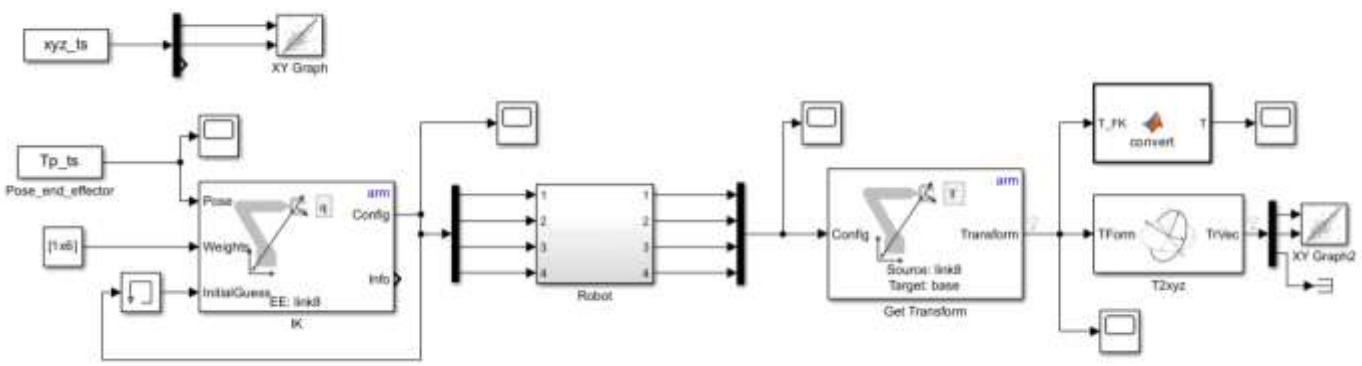
- Đầu ra của IK:

- Các ma trận qj lưu toàn bộ quỹ đạo joint-space.
- $cfgSol$ là nghiệm khớp tại i

➔ Numerical IK linh hoạt, phù hợp robot underactuated 4DOF.

e) Thực hiện

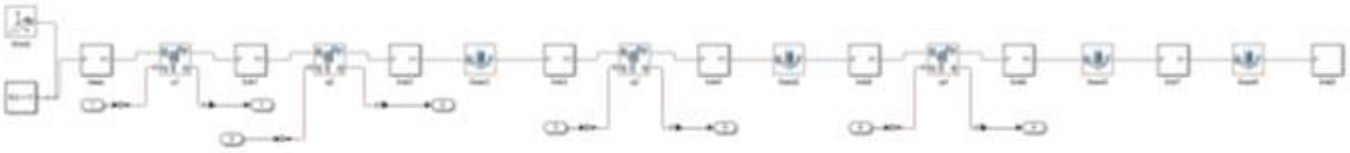
- Tiếp theo ta sẽ thực hiện mô phỏng bài toán bằng Matlab và Simulink. Dưới đây là sơ đồ kết nối của mô hình trong Simulink:



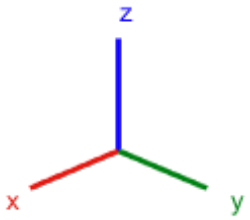
Hình 6. Mô hình trong Simulink

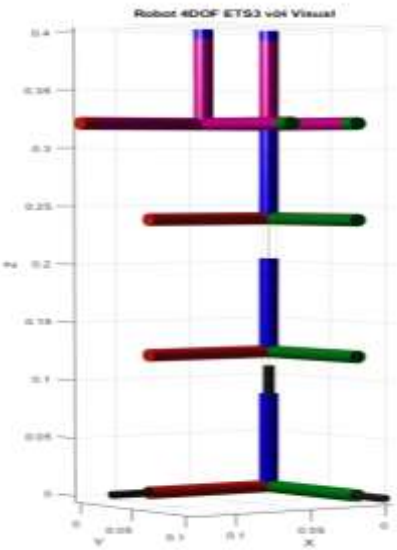
- Khối Tp_{ts}: Là Tp được lưu dưới dạng timeseries, Tp_{ts} được tính bằng Matlab code, kết quả sẽ được lưu vào Workspace. Ta có thể truy xuất dữ liệu bằng khối From Workspace. Khối xyz_{ts}: Là xyz (tọa độ) của end-effector được lưu dưới dạng timeseries, xyz_{ts} cũng được tính từ Matlab và truy xuất dữ liệu bằng khối From Workspace.

- Khối Robot (model): Được tạo ra bằng các liên kết đơn giản bằng Matlab code (model được lưu dưới dạng .slx), sau khi tạo thì ta có được biến chứa thông tin của robot là arm. Bên trong khối Robot:



Hình 7. Cấu trúc kết nối của robot





Hình 8. Cấu trúc robot sau khi hiển thị bằng ETS3

- Khối IK: Là khối Inverse Kinematics, nhận đầu vào là Pose (Tp_ts), Weights, và Initial Guess có đầu vào là nghiệm khớp của thời điểm trước đó. Bên cạnh đó nó còn nhận một đầu vào nữa là cấu trúc robot, ở đây nhóm sẽ cho giá trị bằng arm, tức là robot tự tạo của nhóm.

- Các khối sau đó có nhiệm vụ kiểm tra quỹ đạo của end-effector (xyz), ma trận Tp, vị trí khớp (q), sau khi đi qua IK và Robot arm.

3.2.2. Đồng bộ mô phỏng và cơ khí

a) Sơ đồ tổng quát hệ thống

Hình 9. Sơ đồ tổng quát hệ thống robot

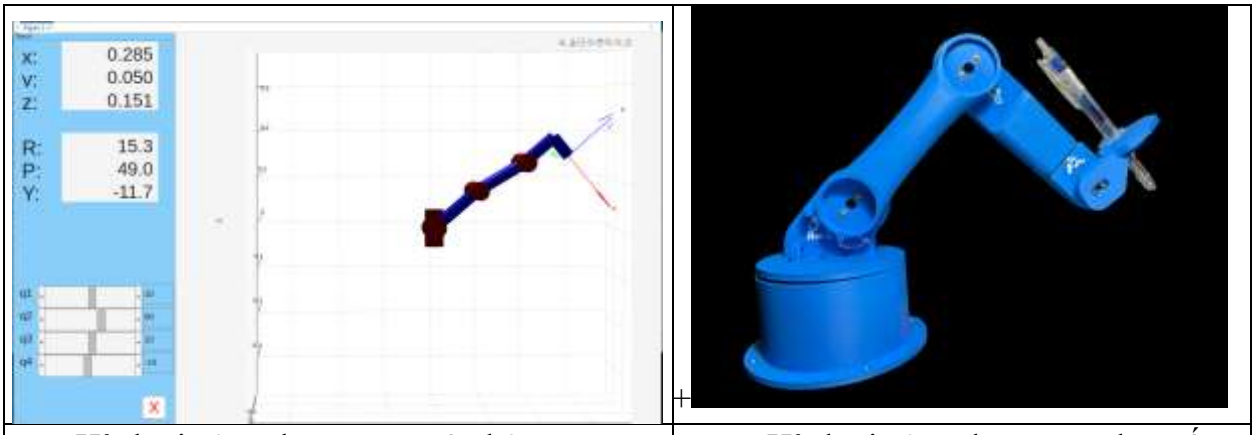
Hệ thống gồm các quy trình:

(1) Mô phỏng robot trên Matlab
(2) Truyền dữ liệu góc qua cổng serial để ESP32 thực hiện việc điều khiển các khớp.
(3) Robot vẽ hình.

b) Chi tiết quá trình thực thi

- Quá trình đưa ra thực tế gồm các bước:

Bước 1: Xét chiều khớp trong Matlab và xét chiều khớp trong thực tế



Hành vi của robot trong mô phỏng	Hành vi của robot trong thực tế

Hình 10. Hành vi của robot đối với các giá trị khớp tương ứng

- Thực hiện: Nhóm viết đoạn code trong Adruino IDE điều khiển các servo quay các góc tương ứng với các giá trị góc trên Matlab.

- Chú thích: Trong Matlab các giá trị góc các khớp lần lượt là (10; 50; 10; -10) nhưng trong Adruino IDE các giá trị góc các khớp lần lượt là (100; 140; 100; 80). Lý do vì trong Matlab

có góc khớp chạy từ đoạn [-90; 90] nhưng servo thực tế chạy từ đoạn [0; 180] nên nhóm đã có chỉnh sửa để các góc tương ứng với nhau.

- Nhận xét: Với các giá trị góc đồng bộ giữa mô phỏng và thực tế, hành vi của robot có sự chênh lệch với nhau. Xét về hướng thì các khớp chưa có sự đồng bộ, điều này gây ra sai lệch lớn trong dự án khi triển khai.

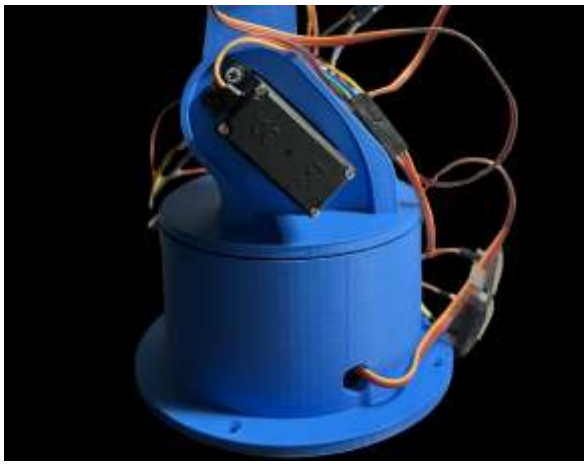
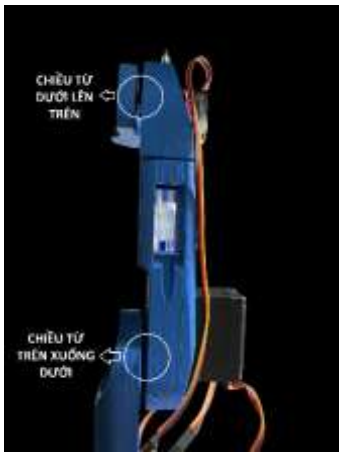
Bước 2: Thực hiện tinh chỉnh góc khớp của cánh tay thực tế (servo) để đồng bộ hành vi giữa mô phỏng và thực tế

- Thực hiện: Nhóm đánh dấu các khớp, sử dụng đoạn code ở bước 1 để thiết lập các góc servo về 0° và 180° để định vị cánh tay, sau đó kiểm thử với hành vi của robot trên mô phỏng.

- Kết quả: Các khớp có giá trị 0° và 180° cho hành vi không tương ứng với hành vi trên mô phỏng.

- Nhận xét:

- Với các giá trị góc tương ứng bằng nhau của các khớp thì hành vi trên mô phỏng và thực tế khác nhau, nhóm nhận xét điều đó xảy ra vì điều kiện khớp trong mô phỏng và điều kiện khớp trong thực tế (servo) là khác nhau.
- Trong mô phỏng các khớp được đặt cùng hướng và cùng chiều (xét theo điểm base khi chưa chuyển động). Còn trong thực tế, vị trí các servo khác nhau (có servo đặt cố định nghiêng góc 45°, có servo có chiều ngược chiều với các servo khác).

Servo đặt nghiêng góc 45°	Servo ngược chiều
	

Hình 11. Vị trí và góc đặt servo

➔ Vì thế, nhóm tinh chỉnh các góc về cả mặt cơ khí và mặt logic.

- Tinh chỉnh về mặt cơ khí: Nhóm thực hiện lắp các link sao cho vị trí link tương ứng với vị trí link trong mô phỏng. So sánh với các điểm 0° và 180° trước đó đã đánh dấu để xem với hành vi đó thì giá trị góc khớp thực tế khác như thế nào so với góc khớp mô phỏng.

- Tinh chỉnh về mặt logic: Sau khi tinh chỉnh về mặt cơ khí xong, nhóm sẽ kết nối đồng bộ cánh tay thực tế với đoạn code vẽ chữ B để so sánh hành vi của cánh tay mô phỏng và cánh tay thực tế. Đồng thời quan sát các góc được log ra trong terminal để xét trong thực tế, giúp việc tinh chỉnh đạt hiệu quả nhất. Khi đó nhóm sẽ biết được chắc chắn sự chênh lệch giữa góc khớp thực tế và góc khớp mô phỏng. Sau đó nhóm sẽ đặt điều kiện logic và kiểm tra lại sao cho hành vi giữa mô phỏng và thực tế là đồng bộ nhất.

Bước 3: Chỉnh sửa sai số, kiểm thử và nhận xét

- Sau khi tinh chỉnh xong, nhóm chạy code viết chữ B. Kết quả nhận được, hành vi của cánh tay robot thực tế tương ứng với hành vi của cánh tay robot mô phỏng, xét theo các trường hợp đặt bút, nhấc bút và cho ra kết quả là chữ B có khả năng nhận diện tốt. Tuy có sai lệch về giá trị thẩm mỹ vì các lý do khách quan như:

- Servo có sai số: Trong quá trình triển khai, nhóm nhận thấy servo có khoảng sai số ± 1°. Vì ảnh hưởng của quán tính, sau khi kết thúc 1 nét vẽ, servo có dịch chuyển thêm 1 góc rất nhỏ nhưng đủ tạo ra sai số ảnh hướng kết quả.
- Model cánh tay có sự rung lắc: Trong quá trình triển khai, nhóm nhận thấy model cánh tay có sự rung lắc do kiểu dáng thiết kế.
- Quán tính của cánh tay: Ở những nét vẽ, quán tính có tác động nhẹ cộng hưởng với khoảng sai số của servo làm cho nét vẽ có sự sai lệch. Quán tính cũng làm cho model cánh tay rung lắc.
- Trọng lực của trái đất: Vì servo có sai số ±1° mà trọng lực luôn hướng xuống nên cũng 1 phần làm cho servo không ở vị trí chính xác (vị trí tuyệt đối tương ứng với mô phỏng)

➔ Nhận xét về sự sai số: Các góc sai số từ servo, sự rung lắc do model, quán tính và trọng lực tuy nhỏ nhưng cánh tay có độ dài đủ lớn và khối lượng đủ nặng nên khi ánh xạ sự sai số từ tâm servo đến cơ cấu thực thi thì sự sai số đó ảnh hưởng rất lớn đến độ thẩm mỹ của kết quả.

CHƯƠNG 4: KẾT QUẢ - ĐÁNH GIÁ

4.1. Kết quả hiệu suất giữa hai thuật toán IK solver trong việc giải quyết bài toán

4.1.1. Dữ liệu

- Bộ dữ liệu đầu vào bao gồm 36 quỹ đạo hình học tương ứng với các ký tự a-z, 0-9. Mỗi ký tự được biểu diễn dưới dạng tập hợp các điểm rời rạc theo thứ tự thời gian (font chữ sử dụng là Hershey), tạo thành quỹ đạo mà robot cần bám theo.

- Các quỹ đạo bao phủ nhiều hình dạng cong, gấp khúc và độ dài khác nhau, giúp đánh giá solver trong nhiều tình huống hình học phức tạp.

- Tập dữ liệu này được xem như một benchmark để kiểm thử khả năng bám quỹ đạo và độ ổn định hội tụ của hai thuật toán BFGS Gradient Projection và LM.

4.1.2. Các chỉ tiêu so sánh

- Đánh giá thực hiện thông qua 5 chỉ số trong robot trajectory tracking, bao gồm:

- RMSE (Root Mean Square Error): Đo sai số trung bình pose giữa quỹ đạo gốc và quỹ đạo robot.
- Max Pose Error: Sai số lớn nhất xuất hiện dọc theo quỹ đạo, phản ánh độ ổn định và độ lệch tại những đoạn khó.
- Mean Manipulability: Giá trị trung bình của manipubility dọc theo quỹ đạo, phản ảnh mức độ tránh singularity của lời giải IK.
- Excution Time (s): Thời gian trung bình solver cần để tính toàn bộ IK cho một ký tự.
- Success Rate: Tỷ lệ ký tự mà solver hội tụ thành công (IK trả về nghiệm hợp lệ cho toàn bộ quỹ đạo).

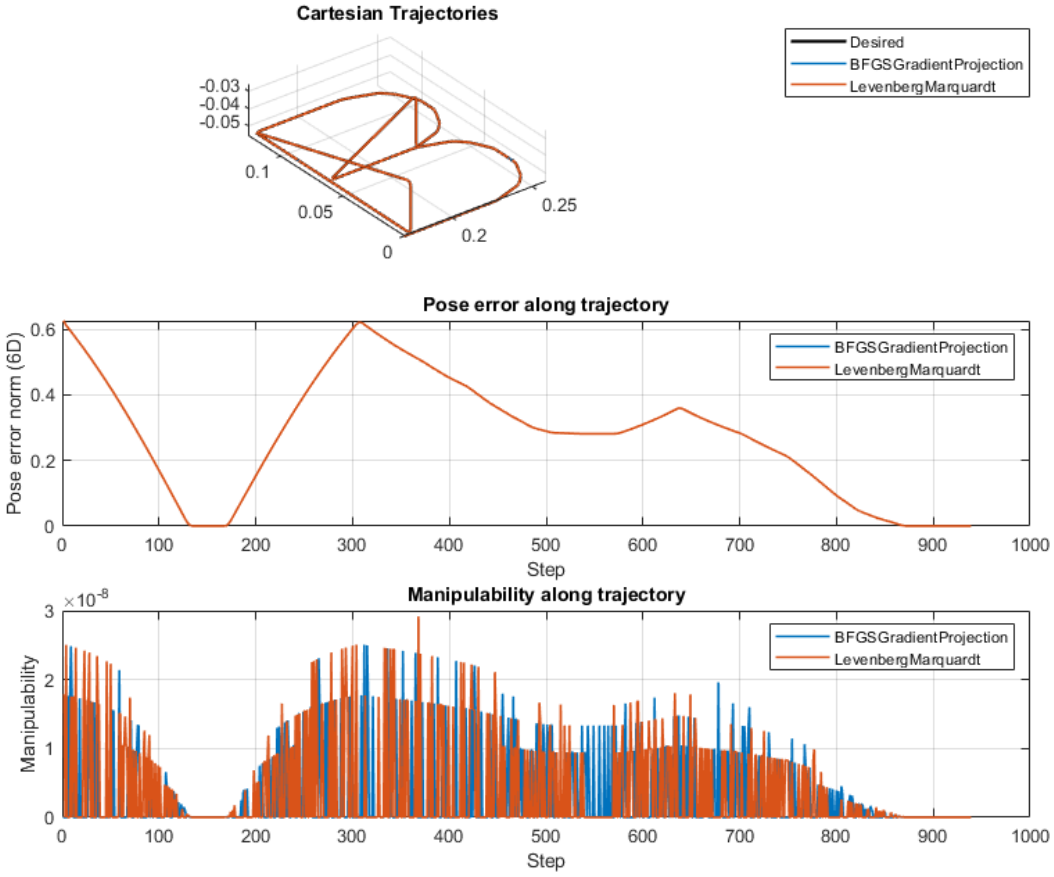
4.1.3. Kết quả và nhận xét

Bảng 7. Kết quả so sánh giữa hai thuật toán

Solver	Mean RMSE	Mean MaxPoseErr	Mean Manip	Mean Time (s)	SuccessRate
BFGS Gradient Projection	2.0461×10^{-8}	0.6039	3.1763×10^{-9}	5.0928	0.11792
LM	4.2666×10^{-8}	0.6039	3.4803×10^{-9}	3.2262	0.11792

Nhận xét:

- Về độ chính xác (RMSE): BFGS cho RMSE nhỏ hơn đáng kể, tức là bám quỹ đạo chính xác hơn LM. Đây là ưu điểm của phương pháp quasi-Newton với cơ chế hiệu chỉnh gradient tốt hơn.
- Về sai số cực đại (MaxPoseErr): Hai thuật toán có giá trị giống nhau, cho thấy những điểm khó của quỹ đạo đều bị ảnh hưởng như nhau.
- Về Manipubility: Hai thuật toán tạo ra quỹ đạo với manipubility rất thấp và tương đương nhau. Điều này chứng tỏ dữ liệu ký tự đưa vào robot (mô hình robot 4dof của nhóm) vào vùng singularity, và solver không có khác biệt rõ rệt trong tránh singularity.
- Về thời gian xử lý: LM nhanh hơn đáng kể.
- Về tỷ lệ hội tụ (Success Rate): Cả hai thuật toán đều có SuccessRate thấp, cho thấy bài toán IK cho các ký tự khó, nhiều điểm robot (mô hình robot 4dof của nhóm) không đạt được hoặc singularity.



Hình 12. Kết quả 2 thuật toán với một ký tự bất kỳ (B)

4.2. Kết quả quỹ đạo vẽ và chuyển động robot

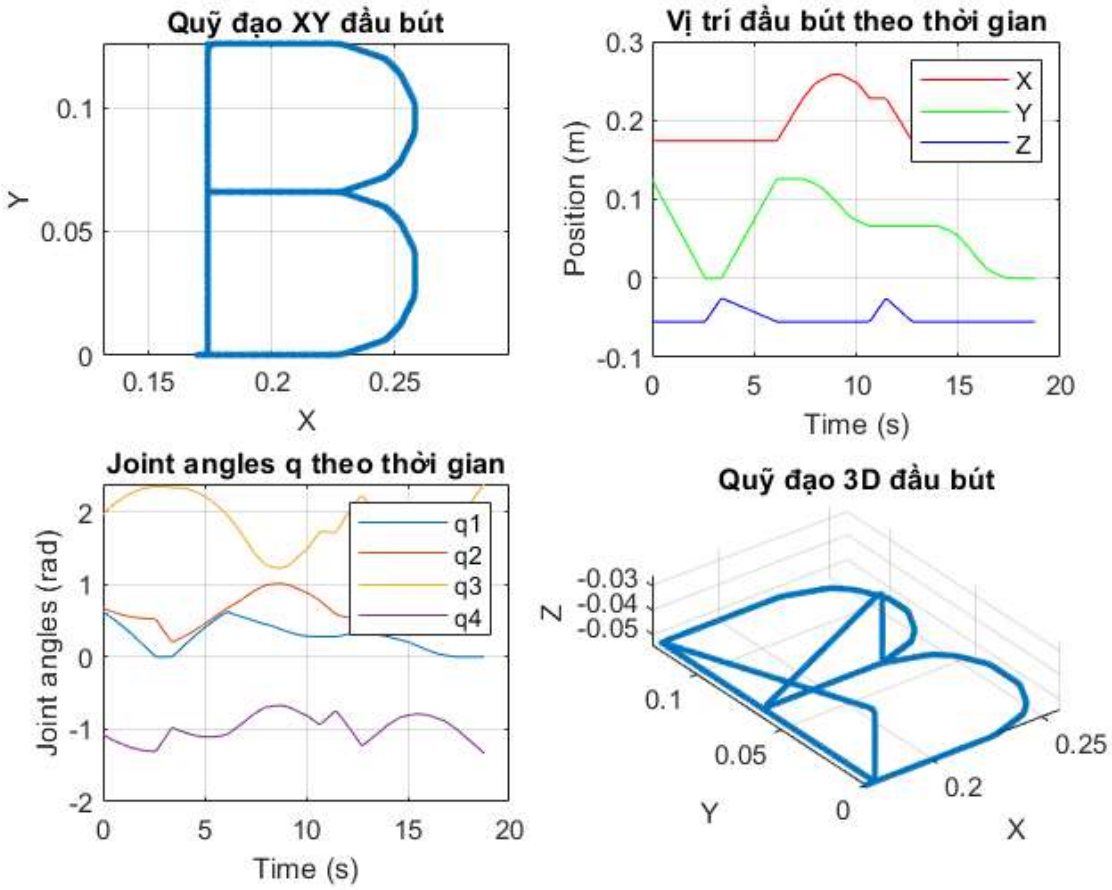
4.2.1. Kết quả mô phỏng

- Kết quả mô phỏng bao gồm dữ liệu từ bộ dữ liệu Hershey và dữ liệu chưa qua xử lí trên internet như: Chữ ‘B’ (best case) hình ảnh hoạt hình Mickey và chữ innovate (worst case).

Đánh giá với trường hợp best case

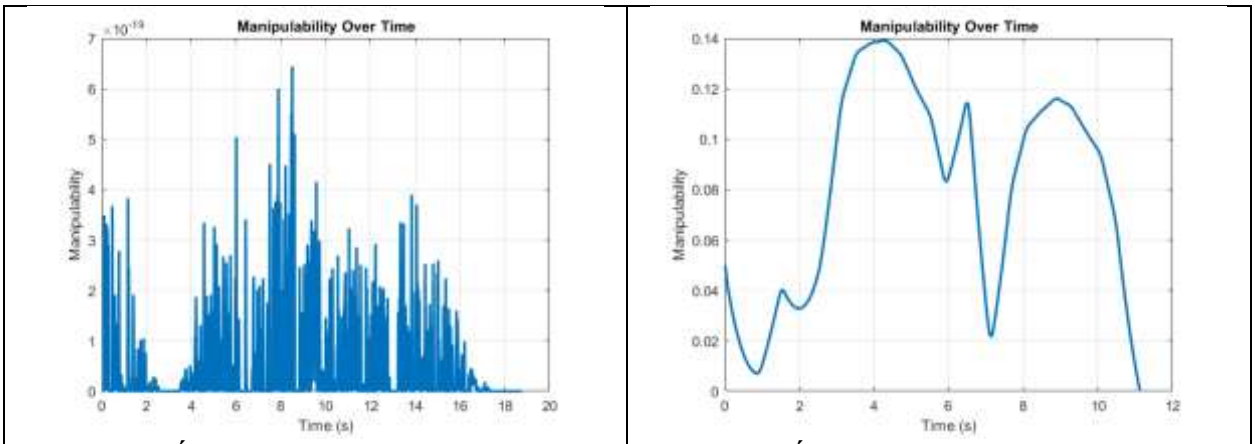
- Với quỹ đạo chữ ‘B’:

- Kết quả mô phỏng:



Hình 13. Đồ thị biểu diễn kết quả mô phỏng

- Đánh giá khả năng điều khiển:



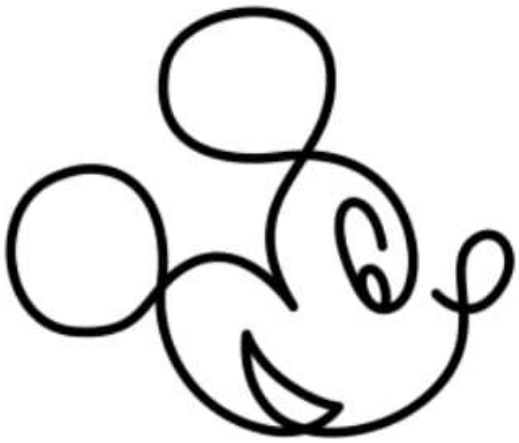
a. Kết quả với mô hình của nhóm b. Kết quả với mô hình abb – 6dof

Hình 14. Đồ thị đánh giá khả năng điều khiển của mô hình

Đánh giá với trường hợp worst case:

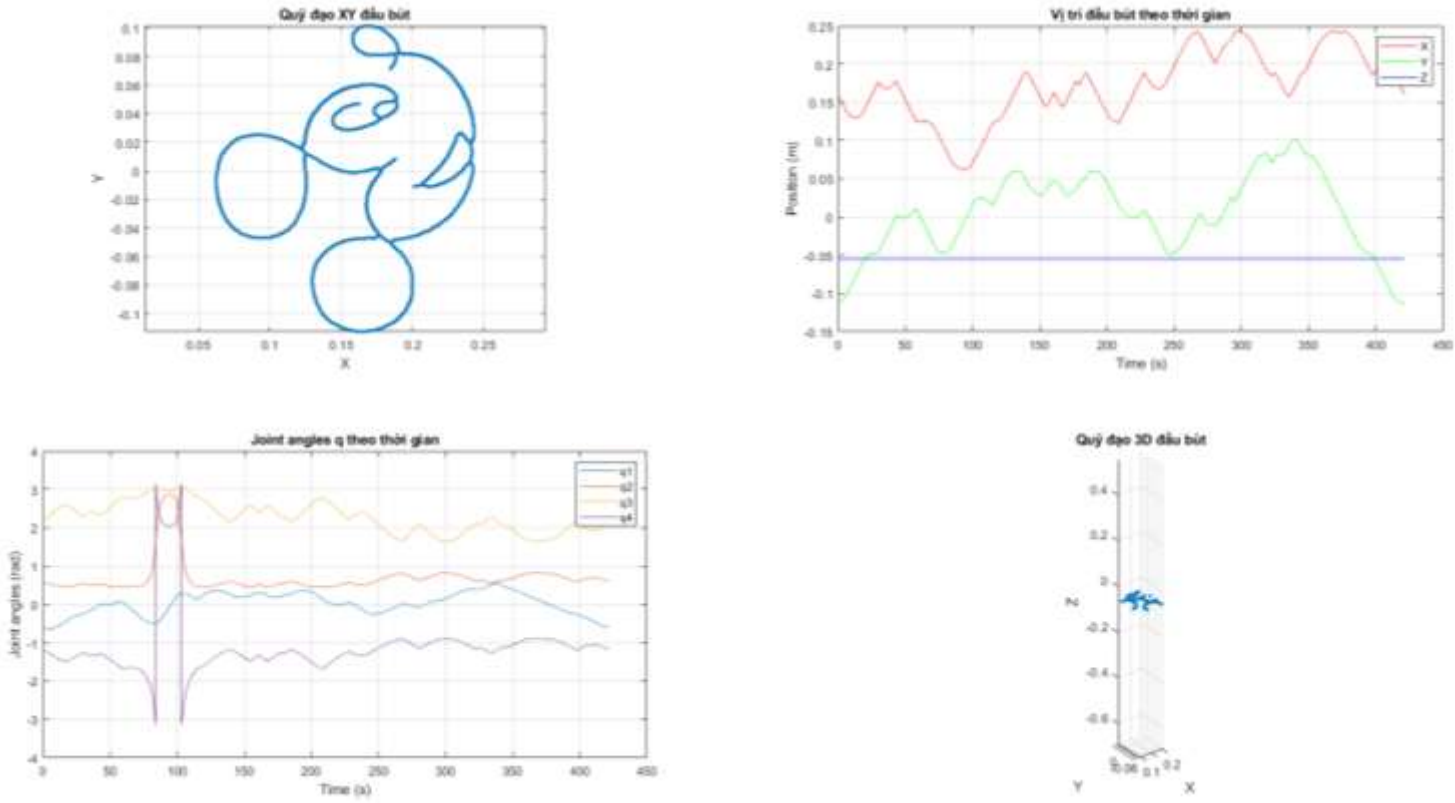
- Với quỹ đạo là hình ảnh Mickey:

- Hình ảnh đầu vào:



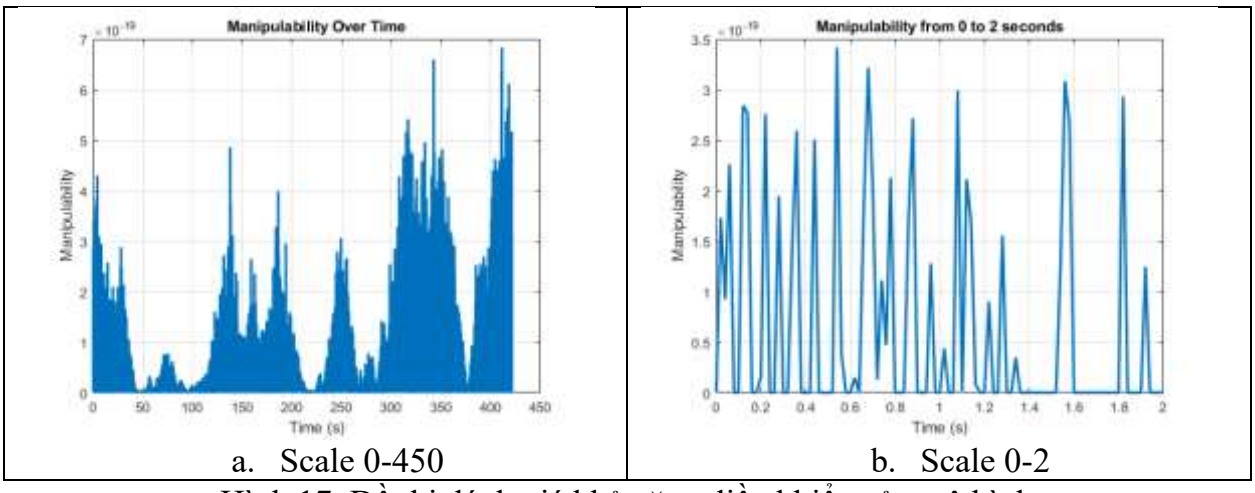
Hình 15. Hình ảnh đầu vào Mickey

- Kết quả mô phỏng



Hình 16. Đồ thị biểu diễn kết quả mô phỏng

- Đánh giá khả năng điều khiển quỹ đạo trong trường hợp tệ:



a. Scale 0-450 b. Scale 0-2

Hình 17. Đồ thị đánh giá khả năng điều khiển của mô hình

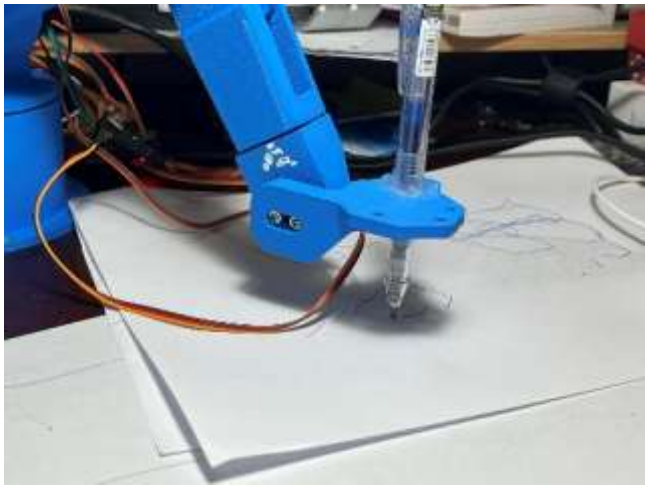
- Nhận xét:

- Đánh giá chủ quan: Robot hoạt động mượt trong chế độ Animate của Matlab.
- Đồ thị đánh giá khả năng điều khiển: Mô hình của nhóm trong suốt khoảng thời gian thực thi giá trị đạt giá 0 nhiều dẫn tới khớp ngoài thực tế không thể

thực thi được do bước nhảy dt (nội suy quỹ đạo) không phù hợp với mô hình của nhóm.

4.2.2. Kết quả mô hình thực tế

- Kết quả mô hình thực tế từ bộ dữ liệu Hershey:

Quá trình vẽ	Kết quả thu được
	
a. Hình robot	b. Kết quả vẽ thực tế

Hình 18. Kết quả thực thi khi vẽ chữ B

- Nhận xét:

- Hình dạng tổng thể giống chữ B: Robot đã đi được hai vòng lồi đặc trưng của chữ B.
- Đường bao được nối liền: Không bị đứt nét nhiều, chứng tỏ servo và chuyển động IK chạy đúng.
- Dựa vào nét vẽ, robot vẽ sai lệch ở các đoạn cong và các vị trí nối điểm:
  - Hai vòng cong của chữ B bị méo
  - Đường dọc ở giữa bị lệch
  - Các điểm nối bị lệch (jumps)

➔ Có thể thấy vài chỗ robot đổi hướng đột ngột, bị "giật" nét.

4.3. Nhận xét chung

- Hiệu năng chuyển động đồng bộ với mô phỏng, độ trễ thấp, tuy nhiên vì những nguyên nhân khách quan như quán tính, trọng lực, độ sai số của servo, độ chính xác của cơ khí nên nét vẽ còn rung và lệch so với quỹ đạo mô phỏng. Tuy nhiên, kết quả vẫn có độ nhận dạng.

- Ở đồ thị hình 12, thể hiện khả năng điều khiển robot trong thực tế, dễ chạm đến giới hạn của phần cơ khí với mô hình của nhóm, vậy nên, trong thực nghiệm cánh tay không thể vẽ giống như được trong mô phỏng.

Link code: https://github.com/Trung1510003/robotarm4dof

Link demo: https://drive.google.com/drive/folders/1VyqK3zCSToKBp9Pk3AyqR47Rt4BdU6k2?usp=sharing

KẾT LUẬN

Bài nghiên cứu này đã đào sâu vào một lĩnh vực trong cánh tay Robot: Sử dụng cánh tay robot 4 bậc để viết, vẽ. Bài nghiên cứu trình bày các khía cạnh về lý thuyết, quá trình thực nghiệm trên Matlab & Simulink, so sánh kết quả với mô hình chuẩn và quá trình triển khai mô hình thực tế.

Dựa vào ý tưởng và định hướng tương lai, nhóm em sẽ khắc phục những hạn chế về phần mềm và cơ khí. Sẽ cải tiến thêm những tính năng mới, tích hợp trí tuệ nhân tại AI và cánh tay để có thể ứng dụng vào cuộc sống hiện nay và nên công nghiệp nước nhà, bên cạnh đó còn đóng góp một nghiên cứu vào khoa học - kỹ thuật, giúp nước ta nắm vững một lĩnh vực quan trọng không thể thiếu trong cuộc cách mạng 5.0. Bổ sung vào sự chạy đua khoa học - kỹ thuật để cho đất nước không bị tụt lại phía sau.

TÀI LIỆU THAM KHẢO

[1] P. Corke, W. Jachimczyk, và R. Pillat, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB®*, 3rd ed. Cham, Switzerland: Springer, 2023.

[2] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms." *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. 9, No. 1, 1979, pp. 62–66.

[3] Lam, L., Seong-Whan Lee, and Ching Y. Suen, "Thinning Methodologies-A Comprehensive Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 14, No. 9, September 1992, page 879, bottom of first column through top of second column.

[4] Gonzalez, R. C., R. E. Woods, and S. L. Eddins, *Digital Image Processing Using MATLAB*, New Jersey, Pearson Prentice Hall, 2004.