TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA TOÁN - CƠ - TIN HỌC

Điều khiển mô hình bàn tay bằng camera

Sinh viên:

Lê Bình Minh

Lớp K67A2 - Toán Tin

Giảng viên:

Ths. Hà Mạnh Toàn

Hà Nội - 2025

Mục lục

1 Giới thiệu vấn đề

1.1 Giới thiệu

Trong các lĩnh vực như đồ họa máy tính, hoạt hình 3D, thực tế ảo (VR/AR) hay điều khiển robot, một vấn đề cơ bản được đặt ra là làm thế nào để ánh xạ chuyển động của con người sang một hệ thống khác một cách tự nhiên và ổn định.

Dữ liệu thu được từ camera hoặc cảm biến thường chỉ bao gồm tập hợp các điểm mốc (landmarks) trong không gian, không mang thông tin về hướng quay hay cấu trúc xương. Bài toán được đặt ra là khôi phục các phép biến đổi hình học cứng (gồm quay và tịnh tiến) sao cho tư thế của mô hình khớp với dữ liệu quan sát.

Trong đồ họa game, phép quay này quyết định hướng của các xương trong khung xương của các mô hình 3D giúp điều khiển chuyển động. Trong robotics, nó là cơ sở để xác định tư thế của bộ phận chuyển động và đồng bộ hóa giữa không gian cảm biến và cơ cấu chấp hành. Vì vậy, việc hiểu rõ nền tảng toán học của bài toán căn chỉnh cứng đóng vai trò then chốt trong việc thiết kế các hệ thống điều khiển chính xác và khả chuyển.

1.1.1 Mục tiêu và phạm vi bài báo cáo

Bài báo cáo này trình bày một giải pháp hoàn chỉnh để điều khiển khung xương cứng của mô hình 3D dựa vào dữ liệu video từ camera. Hệ thống được phát triển sẽ bao gồm ba thành phần chính:

1. **Phát hiện và theo dõi landmarks 2D:** Sử dụng các mô hình học sâu để nhận diện vị trí các điểm mốc quan trọng (ví dụ: các khớp tay) trên từng khung hình video, sau đó theo dõi chúng qua các khung hình liên tiếp.
2. **Ước lượng chuyển động 3D:** Từ dữ liệu landmarks 2D, ước lượng quỹ đạo 3D của các điểm này và tính toán các phép biến đổi hình học (quay, tịnh tiến) phù hợp.
3. **Ánh xạ và điều khiển khung xương:** Ánh xạ các chuyển động 3D này lên hệ xương 3D của mô hình 3D, tính toán các góc khớp tương ứng, và áp dụng chúng để điều khiển chuyển động mô hình trong thời gian thực.

Mỗi thành phần sẽ được trình bày chi tiết ở các chương tiếp theo, bao gồm cơ sở toán học, các thuật toán được sử dụng, và kết quả thực nghiệm.

2 Kiến thức chuẩn bị

2.1 Không gian và hệ tọa độ

Xét không gian Euclid ba chiều $\mathbb{R}^3$ với hệ tọa độ Descartes có gốc tại $O(0, 0, 0)$. Mỗi điểm trong không gian được biểu diễn bởi một vectơ vị trí

$$p = (x,\, y,\, z)^\top.$$

Mọi chuyển động cứng trong không gian có thể được mô tả bằng hai phép biến đổi cơ bản:

- **Phép tịnh tiến**: dịch toàn bộ hệ theo một vectơ $t = (t_x, t_y, t_z)^\top$, làm thay đổi vị trí nhưng không làm thay đổi hướng hay hình dạng.
- **Phép quay**: quay toàn bộ hệ quanh một trục cố định đi qua gốc tọa độ, được mô tả bởi ma trận quay $R \in \mathbb{R}^{3\times 3}$.

Giả sử tại thời điểm t, tọa độ của một điểm là $p = (x, y, z)^\top$. Khi đó, tác động tổng hợp của phép quay R và phép tịnh tiến t được biểu diễn bằng hàm biến đổi:

$$f(p) = Rp + t.$$

2.2 Phép tịnh tiến

Xét không gian Euclid ba chiều $\mathbb{R}^3$ với hệ tọa độ Descartes $(Oxyz)$. **Phép tịnh tiến** là phép biến đổi dịch chuyển toàn bộ không gian theo cùng một hướng và cùng một độ dài, không làm thay đổi hình dạng, kích thước hay hướng của vật thể.

Phép tịnh tiến được xác định bởi một vectơ dịch chuyển

$$t = (t_x, t_y, t_z)^\top,$$

trong đó t_x, t_y, t_z lần lượt là độ dời theo các trục x, y, z.

Nếu một điểm ban đầu có tọa độ $p = (x, y, z)^\top$, thì sau khi thực hiện phép tịnh tiến, điểm mới p' được xác định bởi:

$$p' = p + t = \begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \end{bmatrix}.$$

2.3 Phép xoay

Xét không gian Euclid ba chiều $\mathbb{R}^3$ với hệ tọa độ Descartes $(Oxyz)$, trong đó $O(0, 0, 0)$ là gốc tọa độ.

Phép xoay (rotation) là phép biến đổi bảo toàn gốc tọa độ và độ dài của mọi vectơ, đồng thời thay đổi hướng của chúng. Một phép xoay được xác định bởi một trục quay và một góc quay quanh trục đó. Tập hợp tất cả các phép xoay trong $\mathbb{R}^3$ tạo thành nhóm đặc biệt trực giao $SO(3)$.

Trong thực hành, ta thường mô tả một phép xoay tổng quát bằng ba phép xoay liên tiếp quanh các trục tọa độ chính x, y và z. Ba góc quay tương ứng θ_x, θ_y, θ_z được gọi là **các góc Euler**, lần lượt biểu diễn:

- θ_x: góc quay quanh trục x, điều khiển sự *nghiêng* lên xuống ;
- θ_y: góc quay quanh trục y, điều khiển sự *quay ngang* sang hai bên ;
- θ_z: góc quay quanh trục z, điều khiển sự *xoay tròn* quanh trục dọc .

Ba phép quay cơ bản này được biểu diễn bằng các ma trận:

Phép quay quanh trục x

$$R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix}.$$

Phép quay quanh trục y

$$R_y(\theta_y) = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix}.$$

Phép quay quanh trục z

$$R_z(\theta_z) = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Phép quay tổng quát

Một phép quay tổng quát trong không gian 3 chiều có thể được biểu diễn dưới dạng hợp của ba phép quay cơ bản:

$$R = R_z(\theta_z)\, R_y(\theta_y)\, R_x(\theta_x),$$

trong đó thứ tự nhân ma trận xác định hệ quy chiếu (fixed-frame hoặc moving-frame).

2.4 Phân rã giá trị riêng suy biến (SVD)

Với mọi ma trận $H \in \mathbb{R}^{m\times n}$, tồn tại các ma trận trực giao $U \in \mathbb{R}^{m\times m}$ và $V \in \mathbb{R}^{n\times n}$, cùng ma trận chéo $S \in \mathbb{R}^{m\times n}$ có các phần tử không âm sao cho:

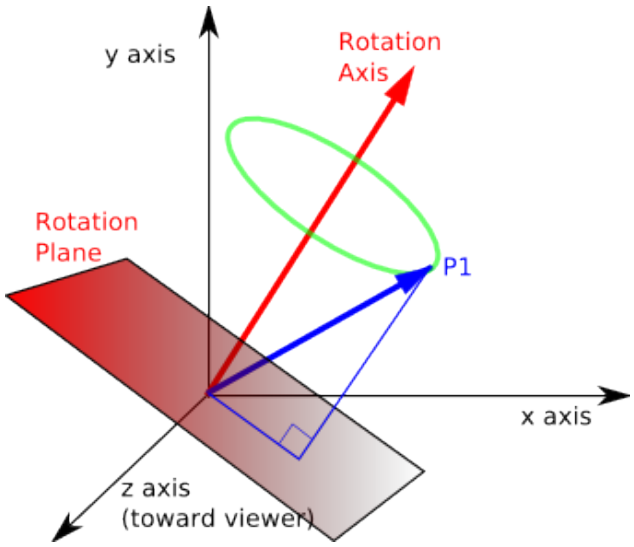
$$H = USV^{\top}.$$

Các phần tử trên đường chéo của S gọi là **các giá trị suy biến** (singular values) của H.

2.5 Biểu diễn bằng quaternion

Quaternion là phần mở rộng của số phức vào không gian ba chiều, có dạng tổng quát:

$$q = w + x\,\mathbf{i} + y\,\mathbf{j} + z\,\mathbf{k},$$



Hình 2.1: Ví dụ mô phỏng phép quay bằng quaternion.

trong đó $w, x, y, z \in \mathbb{R}$ và các đơn vị $\mathbf{i}, \mathbf{j}, \mathbf{k}$ thỏa mãn:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1.$$

Một **quaternion đơn vị** $q = (x, y, z, w)$ với $\|q\| = 1$ biểu diễn một phép quay trong không gian 3 chiều. Cụ thể, phép quay quanh trục đơn vị $\hat{u} = (u_x, u_y, u_z)$ với góc quay θ được viết dưới dạng:

$$q = \cos\frac{\theta}{2} + \hat{u}\,\sin\frac{\theta}{2} = \cos\frac{\theta}{2} + (u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\,\sin\frac{\theta}{2}.$$

3 Bài toán nhận diện các điểm mốc trên bàn tay

3.1 Không gian và hệ toạ độ

Trong quá trình mô hình hoá hình học và phát hiện landmarks, ta cần xác định rõ các không gian làm việc và mối quan hệ giữa chúng. Mỗi bước xử lý – từ thu nhận ảnh, phát hiện điểm đặc trưng, đến tái dựng hình học ba chiều – đều diễn ra trong một hệ toạ độ riêng. Các hệ này được liên kết với nhau thông qua các phép chiếu, phép quay, và tịnh tiến.

3.1.1 World Space

Không gian thế giới, ký hiệu $\mathcal{W}$, là không gian tuyệt đối cố định nơi các vật thể tồn tại. Một điểm $P \in \mathcal{W}$ có toạ độ:

$$P = (X, Y, Z)^\top \in \mathbb{R}^3.$$

Trong hệ thống tracking thời gian thực, gốc toạ độ O_W thường được đặt tại vị trí camera hoặc một điểm tham chiếu cố định trong môi trường. Để đơn giản hóa tính toán, ta có thể chọn hệ toạ độ local với gốc tại cổ tay (wrist), loại bỏ thành phần tịnh tiến tuyệt đối và chỉ giữ lại tư thế tương đối.

3.1.2 Camera Space

Khi camera quan sát thế giới, mỗi điểm $P = (X, Y, Z)$ được biểu diễn trong hệ toạ độ của camera $\mathcal{C}$:

$$P_C = R_{CW}(P - t_{CW}),$$

với R_{CW} là ma trận quay và t_{CW} là vectơ tịnh tiến từ thế giới sang camera. Trục z_C của camera hướng về phía trước ống kính, x_C hướng sang phải và y_C hướng xuống, tuân theo quy ước bàn tay phải của MediaPipe.

3.1.3 Image Space

Mỗi điểm trong không gian camera được chiếu lên mặt phẳng ảnh bằng mô hình chiếu phối cảnh:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim K \begin{bmatrix} X_C/Z_C \\ Y_C/Z_C \\ 1 \end{bmatrix}, \qquad K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

Ở đây (u, v) là toạ độ pixel, (c_x, c_y) là tâm ảnh, và (f_x, f_y) là tiêu cự tính theo đơn vị pixel. Mặt phẳng ảnh $\mathcal{I}$ do đó là không gian hai chiều $\mathbb{R}^2$, biểu diễn dữ liệu thu được từ camera.

3.1.4 Normalized Landmark Space

Trong MediaPipe, các toạ độ ảnh được chuẩn hoá theo kích thước khung hình:

$$x = \frac{u}{W}, \qquad y = \frac{v}{H}, \qquad z = z_{\text{rel}},$$

với W, H là chiều rộng và chiều cao ảnh. Giá trị z_{rel} thể hiện độ sâu tương đối so với cổ tay; nó không mang đơn vị vật lý nhưng bảo toàn tỉ lệ giữa các khớp.

Không gian này, ký hiệu $\mathcal{N}$, được dùng cho toàn bộ pipeline xử lý landmarks:

$$p_i = (x_i, y_i, z_i)^\top \in [0,1]^2 \times \mathbb{R}.$$

Tất cả các phép xoay, tịnh tiến, và căn chỉnh trong các chương sau đều được thực hiện trong $\mathcal{N}$, sau khi đã dịch về gốc cổ tay.

3.1.5 Từ không gian chuẩn hóa đến không gian làm việc 3D

Để phục vụ cho các thuật toán xử lý hình học, ta cần chuyển landmarks từ không gian chuẩn hóa sang không gian làm việc 3D.

Bước 1: Dịch gốc về wrist. Đặt wrist (landmark 0) làm gốc toạ độ local:

$$p'_i = (x_i - x_0,\, y_i - y_0,\, z_i - z_0)^\top, \quad i = 0, 1, \ldots, 20.$$

Phép dịch này loại bỏ thành phần tịnh tiến tuyệt đối, giúp tập trung vào hình dạng và tư thế tương đối của bàn tay.

Bước 2: Chuyển về pixel khi cần thiết. Đối với các thuật toán xử lý ảnh như optical flow, toạ độ chuẩn hóa cần được chuyển về pixel:

$$(u_i, v_i) = (x_i \cdot W,\, y_i \cdot H),$$

với $W = 1280$, $H = 720$ là kích thước khung hình.

Sau các bước xử lý, dữ liệu được chuyển ngược về không gian chuẩn hóa để

đảm bảo tính nhất quán trong pipeline.

3.1.6 Tóm tắt quan hệ giữa các không gian

Chuỗi ánh xạ giữa các hệ toạ độ được mô tả bởi:

$$P_W \xrightarrow{R_{CW}, t_{CW}} P_C \xrightarrow{\text{chiếu phối cảnh}} (u, v) \xrightarrow{\text{chuẩn hoá}} (x, y, z_{\text{rel}}).$$

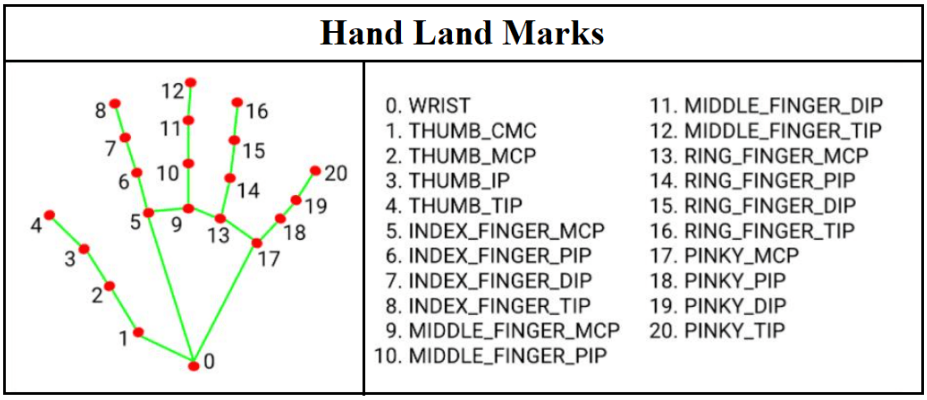
Mỗi tầng mô tả một mức trừu tượng khác nhau của cùng một điểm, từ vị trí vật lý trong thế giới thật đến biểu diễn số học dùng trong mô hình học sâu.

3.2 Bài toán phát hiện landmarks trên ảnh

Mục tiêu của bài toán là xác định vị trí của một tập hợp các điểm đặc trưng (landmarks) trên ảnh hai chiều thu được từ camera, chẳng hạn các khớp trên bàn tay, các đặc trưng khuôn mặt, hoặc vị trí các bộ phận cơ thể.

3.2.1 Cấu trúc hình học của bàn tay

Cấu trúc xương của bàn tay nhận diện từ ảnh được biểu diễn dưới dạng một khung xương gồm 21 điểm. Cấu trúc này bắt đầu từ điểm gốc tại cổ tay (wrist) và phân nhánh thành năm ngón tay. Mỗi ngón tay được biểu diễn bởi bốn điểm khớp nối tiếp: khớp đốt bàn ngón (MCP), khớp liên đốt gần (PIP), khớp liên đốt xa (DIP) và đầu ngón tay (TIP). Cách biểu diễn này giúp nắm bắt đầy đủ các bậc tự do chuyển động, cho phép suy luận chính xác về tư thế và cử chỉ của bàn tay từ dữ liệu hình ảnh (xem Hình **??**).



Hình 3.1: Sơ đồ cấu trúc các điểm đặc trưng trên bàn tay.

3.2.2 Đầu vào và đầu ra của mô hình

Cho một ảnh màu thu được tại thời điểm t:

$$I_t(x, y) : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^3,$$

với (x, y) là toạ độ điểm ảnh, giá trị $I_t(x, y)$ là vector màu trong hệ RGB.

Mô hình học sâu f_θ nhận ảnh này làm đầu vào và ước lượng vị trí của n landmarks trên mặt phẳng ảnh:

$$f_\theta : \mathbb{R}^{H\times W\times 3} \longrightarrow \mathbb{R}^{n\times 2}, \qquad f_\theta(I_t) = \hat{\mathcal{L}}_t = \{\, \hat{p}_i = (\hat{x}_i, \hat{y}_i) \mid i = 1, \ldots, n \,\}.$$

Đầu ra $\hat{\mathcal{L}}_t$ là toạ độ tương đối (thường được chuẩn hoá về miền $[0, 1]^2$) biểu diễn vị trí của từng landmark trong ảnh.

3.2.3 Hàm mục tiêu huấn luyện

Bài toán được huấn luyện theo khuôn khổ học có giám sát với tập dữ liệu

$$\mathcal{D} = \{(I_t, \mathcal{L}_t^{\text{gt}}) \mid t = 1, \ldots, N\},$$

trong đó $\mathcal{L}_t^{\text{gt}} = \{(x_i^{\text{gt}}, y_i^{\text{gt}})\}$ là vị trí thật (ground truth) của các landmarks. Hàm mục tiêu cần cực tiểu hoá sai số giữa dự đoán và giá trị gán nhãn:

$$\min_{\theta} \mathcal{L}(\theta) = \sum_{t=1}^{N} \sum_{i=1}^{n} \left\| f_\theta(I_t)_i - (x_i^{\text{gt}}, y_i^{\text{gt}}) \right\|^2.$$

Một cách khác thường dùng trong thực hành là dự đoán *heatmap* $H_i(x, y)$ cho từng landmark, biểu diễn xác suất hiện diện tại mỗi điểm ảnh, rồi lấy cực đại để thu được toạ độ $(\hat{x}_i, \hat{y}_i)$. Trong trường hợp này, hàm mất mát thường là sai số bình phương giữa heatmap dự đoán và heatmap gán nhãn Gaussian.

3.3 Thư viện MediaPipe

3.3.1 Giới thiệu

MediaPipe [1] là thư viện mã nguồn mở do Google phát triển, nhằm cung cấp các mô hình học sâu và pipeline xử lý thời gian thực cho các bài toán thị giác máy tính. Thư viện này được thiết kế đa nền tảng (Python, C++, Android, Web) và hỗ trợ nhiều tác vụ khác nhau như phát hiện khuôn mặt, bàn tay, theo dõi tư thế (pose tracking), và phân tích cử chỉ.

Trong phạm vi nghiên cứu này, MediaPipe được sử dụng để phát hiện và trích xuất các điểm đặc trưng của bàn tay từ ảnh hoặc video đầu vào thu được qua camera.

3.3.2 Luồng hoạt động và mô hình sử dụng

MediaPipe hoạt động theo kiến trúc *đồ thị xử lý* (graph pipeline), trong đó mỗi bước là một nút (node) đảm nhiệm một tác vụ cụ thể. Đối với bài toán bàn tay, pipeline gồm hai mô hình chính:

- **Palm Detection Model:** Mô hình CNN phát hiện vùng chứa bàn tay trong ảnh, đầu ra là hộp bao (bounding box) bao quanh lòng bàn tay.
- **Hand Landmark Model:** Mô hình hồi quy đa đầu ra dự đoán vị trí của 21 landmarks trong vùng ảnh đã cắt, ký hiệu:

$$\mathcal{L} = \{\, p_i = (x_i, y_i, z_i) \mid i = 1, \ldots, 21 \,\}.$$

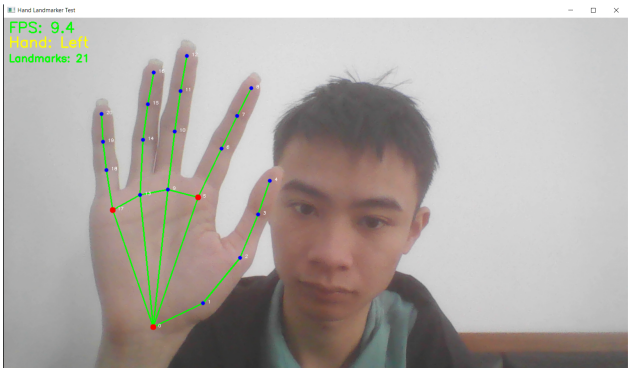
Các toạ độ (x_i, y_i) được chuẩn hoá trong miền $[0, 1]$ theo kích thước ảnh, còn z_i biểu diễn độ sâu tương đối so với mặt phẳng ảnh (giá trị nhỏ hơn nghĩa là gần camera hơn). Luồng xử lý khung hình có thể tóm tắt như sau:

$$\text{Ảnh đầu vào} \xrightarrow{\text{Phát hiện}} \text{Vùng bàn tay} \xrightarrow{\text{Hồi quy landmarks}} \mathcal{L}.$$

3.3.3 Đánh giá và nhận xét

MediaPipe đạt tốc độ xử lý cao và hoạt động ổn định trên CPU, phù hợp cho các ứng dụng thời gian thực như nhận dạng cử chỉ, điều khiển robot, hoặc giao tiếp người–máy.

Về độ chính xác, sai số trung bình của các landmarks thường nhỏ hơn 5% so với kích thước bàn tay trong ảnh. Tuy nhiên, giá trị z chỉ mang tính tương đối nên chưa thể tái dựng hình học 3D tuyệt đối. Hơn nữa, mô hình có thể gặp



Hình 3.2: Ví dụ phát hiện landmarks bàn tay sử dụng MediaPipe.

khó khăn khi bàn tay bị che khuất, xoay nghiêng mạnh hoặc có điều kiện chiếu sáng kém.

4 Trích xuất các tác động lên các điểm

4.1 Trích xuất chuyển động từ ảnh

Sau khi các landmarks đã được xác định bởi MediaPipe (như trình bày trong Chương 3), bước kế tiếp là trích xuất thông tin về sự thay đổi vị trí của chúng theo thời gian. Mục tiêu của giai đoạn này là xây dựng một biểu diễn định lượng cho chuyển động của bàn tay, thông qua các đại lượng như **vector dịch chuyển**, **vận tốc**, và **hướng chuyển động**.

Quá trình này được thực hiện bằng thuật toán **Lucas–Kanade Optical Flow** [2] trong không gian pixel. Trước khi áp dụng, landmarks chuẩn hóa từ MediaPipe được chuyển về toạ độ pixel:

$$(u_i, v_i) = (x_i \cdot W,\, y_i \cdot H),$$

với $W = 1280$, $H = 720$ là kích thước khung hình.

4.1.1 Nguyên lý và mô hình toán học

Giả sử $I(x, y, t)$ là cường độ sáng của điểm ảnh tại thời điểm t. Khi bàn tay di chuyển, cùng một điểm vật lý trong không gian sẽ xuất hiện tại vị trí $(x+u, y+v)$ ở khung hình kế tiếp $t + \Delta t$, với (u, v) là vector dịch chuyển cục bộ. Giả định

cường độ sáng không đổi dẫn tới ràng buộc:

$$I(x, y, t) = I(x + u, y + v, t + \Delta t).$$

Khai triển Taylor và bỏ qua các bậc cao hơn, ta thu được phương trình quang học cơ bản:

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0.$$

Đây là nền tảng của phương pháp Lucas–Kanade, trong đó (u, v) được coi là không đổi trong một cửa sổ nhỏ quanh mỗi điểm mốc, và được xác định bằng cách cực tiểu hoá sai số bình phương giữa hai khung hình liên tiếp.

4.1.2 Phương pháp Lucas–Kanade cục bộ

Tại mỗi landmark (x_i, y_i), ta xét tập hợp các điểm (x_j, y_j) trong cửa sổ kích thước $k \times k$ xung quanh nó. Hệ phương trình tuyến tính được thiết lập:

$$A_i \begin{bmatrix} u_i \\ v_i \end{bmatrix} = -b_i, \qquad A_i = \begin{bmatrix} \frac{\partial I}{\partial x}(x_1, y_1) & \frac{\partial I}{\partial y}(x_1, y_1) \\ \vdots & \vdots \\ \frac{\partial I}{\partial x}(x_n, y_n) & \frac{\partial I}{\partial y}(x_n, y_n) \end{bmatrix}, \quad b_i = \begin{bmatrix} \frac{\partial I}{\partial t}(x_1, y_1) \\ \vdots \\ \frac{\partial I}{\partial t}(x_n, y_n) \end{bmatrix}.$$

Vector dịch chuyển (u_i, v_i) được tính bằng nghiệm tối thiểu bình phương:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = -(A_i^\top A_i)^{-1} A_i^\top b_i.$$

Các giá trị (u_i, v_i) biểu diễn chuyển động biểu kiến của landmark i trên mặt phẳng ảnh giữa hai khung hình liên tiếp.

4.1.3 Tính vận tốc và hướng chuyển động

Từ các giá trị dịch chuyển (u_i, v_i), ta có thể suy ra độ lớn và hướng của chuyển động:

$$\text{speed}_i = \sqrt{u_i^2 + v_i^2}, \qquad \text{angle}_i = \text{atan2}(v_i, u_i).$$

Để giảm nhiễu, các vector dịch chuyển được làm trơn bằng trung bình trượt trên N khung hình gần nhất:

$$\bar{u}_i = \frac{1}{N}\sum_{k=1}^{N} u_{i,k}, \qquad \bar{v}_i = \frac{1}{N}\sum_{k=1}^{N} v_{i,k},$$

với $N \leq 5$. Từ đó, vận tốc và hướng trung bình được xác định:

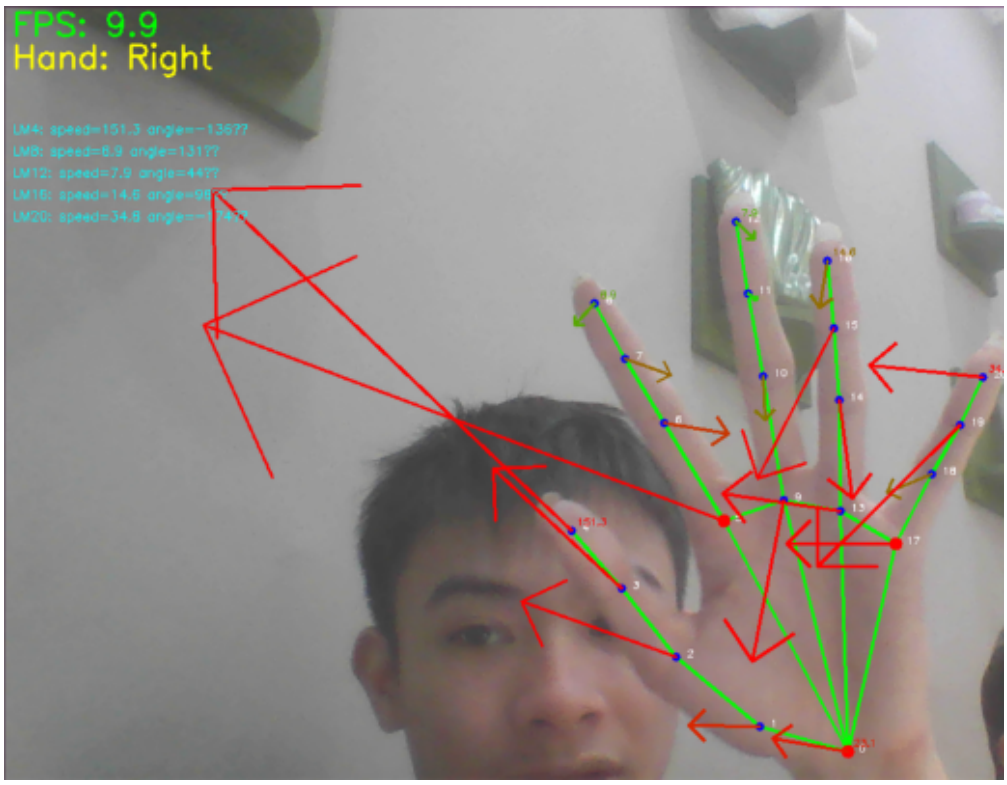
$$\overline{\text{speed}}_i = \sqrt{\bar{u}_i^2 + \bar{v}_i^2}, \qquad \overline{\text{angle}}_i = \text{atan2}(\bar{v}_i, \bar{u}_i).$$

Nếu tại một khung hình optical flow không thu được kết quả hợp lệ, giá trị của landmark được giữ nguyên hoặc thay bằng trung bình gần nhất để đảm bảo tính liên tục của quỹ đạo.

4.2 Trích xuất phép quay giữa các khung hình

Sau khi có landmarks 3D (đã dịch về gốc wrist như mô tả ở Chương 3), bước tiếp theo là ước lượng sự thay đổi định hướng (orientation) của bàn tay giữa hai khung hình liên tiếp.

Trong nghiên cứu này, phép quay toàn cục (root rotation) được trích xuất bằng phương pháp **Kabsch** [3], một kỹ thuật tuyến tính ổn định cho bài toán căn chỉnh cứng (rigid alignment). Để giảm nhiễu và tập trung vào cấu trúc lòng bàn tay, ta chỉ sử dụng **ba điểm đại diện** thay vì toàn bộ 21 landmarks.



Hình 4.1: Minh họa các vector dòng chảy quang học (optical flow vectors) được trích xuất từ các landmark của bàn tay. Các mũi tên biểu diễn hướng và độ lớn của chuyển động của từng landmark giữa hai khung hình liên tiếp.

4.2.1 Mô hình bài toán

Gọi $\mathcal{P} = \{p_W, p_I, p_P\}$ là tập ba điểm đại diện (wrist, index MCP, pinky MCP tương ứng với ID $\{0, 5, 17\}$) ở frame đầu tiên (bind pose), và $\mathcal{Q} = \{q_W, q_I, q_P\}$ là ba điểm tương ứng ở frame hiện tại, với các điểm $p_i, q_i \in \mathbb{R}^3$ đã được dịch về gốc wrist.

Mục tiêu là tìm ma trận quay $R \in SO(3)$ sao cho sai số bình phương giữa hai tập điểm được tối thiểu hoá:

$$E(R) = \sum_{i \in \{W,I,P\}} \|q_i - Rp_i\|^2.$$

Do đã dịch về gốc wrist, thành phần tịnh tiến $t = 0$.

4.2.2 Giải thuật Kabsch

Bài toán trên được giải bằng cách trước hết đưa hai tập điểm về cùng trọng tâm (centroid):

$$\tilde{p}_i = p_i - \bar{p}, \qquad \tilde{q}_i = q_i - \bar{q},$$

trong đó $\bar{p} = \frac{1}{3}(p_W + p_I + p_P)$ và $\bar{q} = \frac{1}{3}(q_W + q_I + q_P)$ là trọng tâm của tam giác tạo bởi ba điểm.

Ma trận hiệp phương sai (cross-covariance) được xác định bởi:

$$H = \tilde{P}^\top \tilde{Q} = \begin{bmatrix} \tilde{p}_W^\top \\ \tilde{p}_I^\top \\ \tilde{p}_P^\top \end{bmatrix} \begin{bmatrix} \tilde{q}_W & \tilde{q}_I & \tilde{q}_P \end{bmatrix} \in \mathbb{R}^{3\times 3}.$$

Thực hiện phân rã giá trị rieng suy biến (SVD):

$$H = USV^\top.$$

Khi đó, ma trận quay tối ưu được tính bằng [3]:

$$R = VU^\top.$$

Nếu $\det(R) < 0$, cột cuối của V được đổi dấu để đảm bảo $R \in SO(3)$.

4.2.3 Ý nghĩa hình học

Ma trận R biểu diễn sự quay toàn cục (root rotation) của bàn tay giữa frame hiện tại so với bind pose ban đầu. Đây chính là thông tin định hướng (orientation) của toàn bộ bàn tay trong không gian 3D.

Trong pipeline của hệ thống:

- Ma trận R được chuyển thành quaternion $\mathbf{q}_{\text{root}}$ để truyền đến client 3D.
- Quaternion này được làm mượt bằng SLERP (Spherical Linear Interpolation) [4] để giảm nhiễu cao tần giữa các frame liên tiếp.
- Kết quả được sử dụng làm root rotation cho skeleton trong Chương 5.

5 Bài toán gán các điểm mốc vào mô hình bàn tay 3D

5.1 Không gian và cấu trúc khung xương của mô hình bàn tay

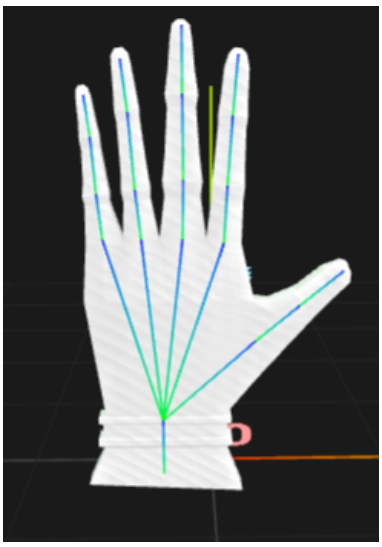
Mô hình bàn tay được biểu diễn dưới dạng hệ xương phân cấp, trong đó mỗi khớp được mô tả bởi ba thành phần: tên khớp, khớp cha (`parent`), vị trí tương đối so với khớp cha (`position`) và quaternion quay cục bộ (`quaternion`). Tập dữ liệu `skeleton_metadata.json` chứa tổng cộng 26 nút: một nút gốc `_rootJoint`, một nút trung tâm `Bone_00`, và 24 khớp còn lại thuộc năm ngón tay, trong đó mỗi ngón tay bao chia đều cho 5 ngón tay, riêng ngón cái chỉ có 4 khớp.

5.1.1 Hệ toạ độ cơ sở

Nút gốc `_rootJoint` có

$$\texttt{position} = (0, 0, 0), \qquad \texttt{quaternion} = (0, 0, 0, 1),$$

được dùng làm gốc của toàn bộ hệ toạ độ. Khớp `Bone_00` là con trực tiếp của `_rootJoint` với vị trí $(0, 0, 0)$ và một quaternion quay cố định, đóng vai trò là



Hình 5.1: Cấu trúc khung xương bàn tay trong mô hình 3D

khớp cổ tay trung tâm.

Mọi khớp còn lại được biểu diễn trong hệ toạ độ local của khớp cha: vector `position` cho biết hướng và độ dài của đoạn xương tiếp theo, còn `quaternion` mô tả hướng xoay của đoạn đó trong tư thế bind. Độ dài các đoạn xương (chuẩn hoá) được tính từ chuẩn Euclid của vector vị trí và nằm trong khoảng:

$$0.60 \lesssim \|\texttt{position}\| \lesssim 3.23.$$

Tổng độ dài từ cổ tay tới đầu ngón giữa (chuỗi `Bone008_08` đến `Bone011_end_022`) xấp xỉ 7.25 đơn vị ảo.

5.1.2 Cấu trúc phân cấp của khung xương

Các khớp được tổ chức thành năm chuỗi tương ứng với năm ngón tay, đều xuất phát từ `Bone_00`. Tên khớp mang phần hậu tố dạng `_NN`, trong đó `NN` tương ứng với chỉ số landmark trong hệ 3D bên ngoài.

Bảng **??** tóm tắt cấu trúc cho từng ngón tay, bao gồm số đoạn liên tiếp và

khoảng độ dài tương đối của các đoạn (theo dữ liệu trong `skeleton_metadata.json`).

Bảng 5.1: Khung xương bàn tay (tóm tắt từ `skeleton_metadata.json`)

Ngón tay	**Số đoạn (bao gồm end)**	**Khoảng chiều dài**
Ngón cái (Thumb)	4	0.91 – 2.11
Ngón trỏ (Index)	5	0.77 – 3.11
Ngón giữa (Middle)	5	0.93 – 3.11
Ngón áp út (Ring)	5	0.87 – 3.21
Ngón út (Pinky)	5	0.60 – 3.22

5.1.3 Định hướng khớp bằng quaternion

Mỗi khớp mang một quaternion đơn vị (x, y, z, w) mô tả phép quay local so với khớp cha trong tư thế bind. Ví dụ:

$$\begin{aligned} q_{\texttt{Bone001\_01}} &= (6.20 \times 10^{-8},\ 2.91 \times 10^{-8},\ -0.4092,\ 0.9124), \\ q_{\texttt{Bone002\_02}} &= (-8.5 \times 10^{-10},\ -5.2 \times 10^{-8},\ -0.0176,\ 0.9998). \end{aligned} \tag{5.1}$$

Từ các quaternion này, có thể khôi phục ma trận quay $R \in SO(3)$ của từng khớp và tính được hướng xương (bone direction) trong không gian mô hình.

5.1.4 Ý nghĩa trong không gian mô hình

Không gian của mô hình bàn tay hoàn toàn là không gian chuẩn hoá: tất cả các toạ độ và độ dài là tương đối, chỉ giữ tỉ lệ giữa các đốt tay và cấu trúc phân cấp giữa các khớp. Cấu trúc này cung cấp:

- khung tham chiếu hình học để ánh xạ landmarks thu được từ ảnh vào hệ xương 3D;
- thông tin động học (cha–con) để tính động học thuận;

- hướng khớp ban đầu để nội suy quay và thực hiện retargeting chuyển động.

Nhờ đó, mô hình có thể nhận chuyển động ước lượng từ ảnh và biểu diễn lại trong không gian 3D một cách nhất quán.

5.2 Retargeting từ landmarks 3D sang mô hình xương

Các chương trước đã trình bày cách thu thập và xử lý dữ liệu từ camera:

- Chương 3: MediaPipe phát hiện 21 landmarks và trả về tọa độ chuẩn hóa.
- Chương 4: Optical flow trích xuất chuyển động; Kabsch tính root rotation từ 3 điểm đại diện.

Bước cuối cùng là ánh xạ toàn bộ dữ liệu này lên hệ xương phân cấp của mô hình 3D. Quá trình này gọi là **retargeting**, bao gồm:

1. Chuyển đổi không gian tọa độ (camera $\rightarrow$ scene).
2. Ánh xạ landmarks sang bones thông qua bảng mapping.
3. Tính quaternion local cho từng khớp dựa trên hướng của landmarks.
4. Áp dụng forward kinematics để dựng lại skeleton hoàn chỉnh.

5.2.1 Cấu trúc phân cấp và bảng ánh xạ

Skeleton của mô hình bao gồm hai nút gốc đặc biệt: `_rootJoint` (gốc tuyệt đối) và `Bone_00` (root bone chính). Nút `Bone_00` có quaternion khởi tạo không

trivial:

$$q_{\texttt{Bone\_00}} = (0.707, 0, 0, 0.707),$$

thể hiện phép quay 90° quanh trục x để căn chỉnh hệ toạ độ mô hình với hệ toạ độ làm việc.

Ánh xạ từ landmarks sang bones không phải là song ánh: chỉ một tập con các landmarks được chọn để điều khiển các khớp quan trọng. Bảng ánh xạ $\mathcal{M}$ xác định quan hệ:

$$\mathcal{M} : \{0, 1, 2, 4, 5, \ldots, 20\} \to \{\texttt{Bone\_00}, \texttt{Bone002\_02}, \ldots\},$$

ví dụ: $\mathcal{M}(0) = \texttt{Bone\_00}$, $\mathcal{M}(1) = \texttt{Bone002\_02}$, $\mathcal{M}(5) = \texttt{Bone013\_013}$.

Lưu ý rằng một số ID landmarks (như 3, 13, 14) không xuất hiện trong $\mathcal{M}$, và các bone trung gian (như `Bone001_01`, `Bone004_04`) được điều khiển gián tiếp qua chuỗi động học.

5.2.2 Chuỗi ngón tay và bind pose

Mỗi ngón tay được biểu diễn bằng một chuỗi các cặp (i, j), với i, j là ID của landmarks cha–con:

$$\begin{aligned}
\text{Thumb} &: \{(1,2), (2,3), (3,4)\},\\
\text{Index} &: \{(5,6), (6,7), (7,8)\},\\
\text{Middle} &: \{(9,10), (10,11), (11,12)\},\\
\text{Ring} &: \{(13,14), (14,15), (15,16)\},\\
\text{Pinky} &: \{(17,18), (18,19), (19,20)\}.
\end{aligned}$$

Tại frame đầu tiên khi phát hiện bàn tay, bộ ba điểm $\{L_0, L_5, L_{17}\}$ (wrist, index MCP, pinky MCP) được ghi lại làm bind pose cho thuật toán Kabsch. Các landmarks còn lại không tham gia vào tính toán root rotation, mà chỉ dùng để xác định hướng của từng đốt ngón.

5.2.3 Chuyển đổi không gian tọa độ

Landmarks từ MediaPipe nằm trong hệ camera với quy ước: trục $+x$ sang phải, $+y$ xuống, $+z$ ra khỏi camera. Để chuyển sang hệ scene (thường dùng $+Y$ lên trong Three.js/Unity), áp dụng ma trận:

$$R_{\text{scene}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Đối với tay trái, cần thêm phép gương theo trục x:

$$R_{\text{mirror}} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Phép biến đổi tổng quát từ không gian landmark sang không gian model:

$$\mathbf{p}_{\text{model}} = R_{\text{scene}} \cdot R_{\text{mirror}} \cdot \mathbf{p}_{\text{camera}},$$

với $R_{\text{mirror}} = I$ nếu là tay phải.

5.2.4 Tính quaternion local cho từng bone

Với mỗi cặp landmarks liên tiếp (i, j) trong chuỗi ngón tay, vector hướng được tính bằng:

$$\mathbf{d}_{ij} = \mathbf{p}_j - \mathbf{p}_i,$$

sau đó chuyển đổi sang không gian model:

$$\mathbf{d}'_{ij} = R_{\text{scene}} \cdot R_{\text{mirror}} \cdot \mathbf{d}_{ij}.$$

Quaternion local của bone tương ứng được xác định bằng phép quay ngắn nhất từ hướng chuẩn $\texttt{bone\_dir} = (0, 1, 0)^\top$ (hướng dương trục y trong local space) sang hướng quan sát $\hat{\mathbf{d}}'_{ij}$ (đã chuẩn hóa):

$$\begin{aligned}
\mathbf{a} &= \texttt{bone_dir} \times \hat{\mathbf{d}}'_{ij}, \\
\theta &= \arccos(\texttt{bone_dir} \cdot \hat{\mathbf{d}}'_{ij}), \\
\mathbf{q} &= \left[\mathbf{a} \cdot \sin\left(\tfrac{\theta}{2}\right), \cos\left(\tfrac{\theta}{2}\right)\right].
\end{aligned}$$

Trường hợp đặc biệt: nếu tay trái, ma trận quay của root bone cũng cần được mirror:

$$R^{\text{left}}_{\text{root}} = R_{\text{mirror}} \cdot R^{\text{right}}_{\text{root}} \cdot R_{\text{mirror}}.$$

5.2.5 Forward kinematics

Vị trí global của từng bone được tính tuần tự theo thứ tự phân cấp. Gọi ℓ_b là độ dài rest (trong bind pose) của bone b, vị trí của bone con được xác định:

$$\mathbf{p}^{\text{global}}_{\text{child}} = \mathbf{p}^{\text{global}}_{\text{parent}} + R^{\text{global}}_{\text{parent}} \cdot (\hat{\mathbf{d}}'_{\text{parent}} \cdot \ell_b),$$

trong đó $R_{\text{parent}}^{\text{global}}$ là tích lũy các quaternion từ root xuống parent, và $\hat{\mathbf{d}}'_{\text{parent}}$ là hướng đã chuẩn hóa của bone parent.

Kết quả retargeting bao gồm hai thành phần:

- $\mathcal{P}_{\text{global}} = \{\texttt{bone\_name} \mapsto \mathbf{p}^{\text{global}}\}$: toạ độ 3D của từng bone trong scene space,
- $\mathcal{Q}_{\text{local}} = \{\texttt{bone\_name} \mapsto \mathbf{q}\}$: quaternion local cho từng bone.

Dữ liệu này được tuần tự hóa thành JSON và truyền qua WebSocket đến client 3D (Three.js) với tần suất 30 khung hình/giây, cho phép hiển thị chuyển động thời gian thực.

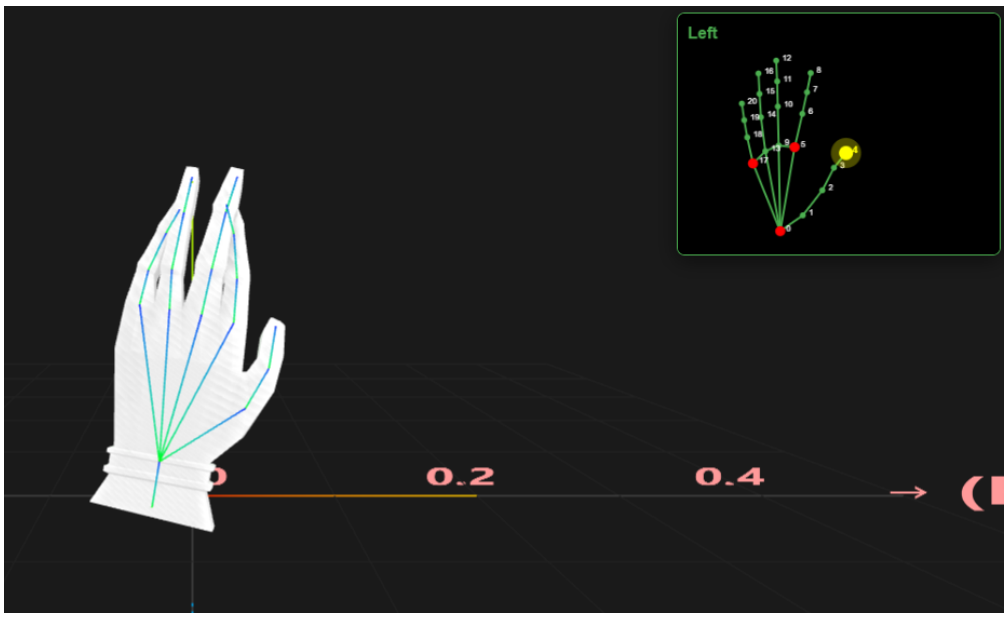
6 Nhận xét và kết luận

6.1 Tóm tắt kết quả

Báo cáo trình bày về quy trình xử lý để theo dõi và ánh xạ chuyển động bàn tay từ video sang mô hình 3D.

- **Phát hiện landmarks**: Sử dụng thư viện MediaPipe [1] để phát hiện 21 điểm đặc trưng trên bàn tay với tốc độ 30 khung hình/giây, đạt độ chính xác cao trong điều kiện chiếu sáng thông thường.
- **Trích xuất chuyển động**: Áp dụng thuật toán Lucas–Kanade optical flow [2] để tính toán vector dịch chuyển và vận tốc của từng landmark, kết hợp với bộ lọc trung bình trượt để giảm nhiễu.
- **Ước lượng root rotation**: Sử dụng thuật toán Kabsch [3] trên ba điểm đại diện (wrist, index MCP, pinky MCP) để tính ma trận quay toàn cục với độ ổn định cao. Quaternion thu được qua SLERP [4] để làm mượt chuyển động giữa các frame.
- **Retargeting**: Chuyển đổi landmarks sang không gian mô hình 3D, tính quaternion local cho từng khớp bằng phép quay ngắn nhất, và áp dụng forward kinematics để dựng lại skeleton hoàn chỉnh.

Hệ thống đã được triển khai với kiến trúc client-server: backend Python xử



Hình 6.1: Một số kết quả minh họa chuyển động bàn tay được theo dõi và ánh xạ lên mô hình 3D.

lý video và tính toán, frontend Three.js hiển thị mô hình 3D, giao tiếp qua WebSocket đảm bảo độ trễ thấp.

6.2 Đánh giá và hạn chế

Quan sát thực nghiệm

- **Độ ổn định hạn chế**: Chuyển động hiển thị thường dao động, xuất hiện jitter; SLERP [4] giúp giảm nhiễu nhưng chưa triệt để.
- **Sai số hình học**: Ánh xạ hướng các ngón tay từ landmarks sang quaternion local chưa bám sát tư thế thật, đặc biệt khi có xoay phức tạp.
- **Root rotation chưa chính xác**: Kabsch [3] với 3 điểm đại diện cho kết quả thiếu ổn định trong các góc nhìn khó hoặc che khuất.
- **Phụ thuộc điều kiện ảnh**: MediaPipe [1] suy giảm chất lượng khi ánh

sáng kém hoặc có bóng mạnh.

6.3 Kết luận

Hệ thống được xây dựng theo đúng nền tảng toán học (optical flow [2], Kabsch [3], quaternion/SLERP [4]) và pipeline MediaPipe [1]. Tuy nhiên, kết quả thực nghiệm chưa đạt chất lượng mong muốn: chuyển động tái hiện còn dao động, tư thế ngón tay chưa tự nhiên, và root rotation thiếu ổn định .

Những hạn chế này bắt nguồn từ đặc tính dữ liệu (độ sâu tương đối), lựa chọn điểm đại diện ít cho căn chỉnh, và ánh xạ động học đơn giản hoá. Do đó, phần kết luận tập trung ghi nhận các vấn đề kỹ thuật và giới hạn quan sát được, thay vì khẳng định hiệu quả ứng dụng.

Bibliography

[1] Camillo Lugaresi et al. "MediaPipe: A Framework for Building Perception Pipelines". In: *arXiv preprint arXiv:1906.08172* (2019).

[2] Bruce D Lucas and Takeo Kanade. "An iterative image registration technique with an application to stereo vision". In: *Proceedings of DARPA Image Understanding Workshop* (1981), pp. 121–130.

[3] Wolfgang Kabsch. "A solution for the best rotation to relate two sets of vectors". In: *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 32.5 (1976), pp. 922–923.

[4] Ken Shoemake. "Animating rotation with quaternion curves". In: *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. 1985, pp. 245–254.