Báo cáo Nghiên cứu Chuyên sâu: Kiến trúc Hệ thống Scan-to-Path cho Robot Hàn 6 Trục - Tích hợp Thị giác máy 3D và Điều khiển Thời gian thực

1. Tóm tắt Điều hành (Executive Summary)

Trong kỷ nguyên Công nghiệp 4.0, sự chuyển dịch từ các dây chuyền robot hàn được lập trình cứng nhắc (hard-coded teaching) sang các hệ thống hàn thích ứng thông minh (adaptive welding) là một bước tiến tất yếu nhằm đảm bảo chất lượng mối hàn trong bối cảnh dung sai gá lắp và biến dạng nhiệt phức tạp. Báo cáo này cung cấp một phân tích kỹ thuật toàn diện và chuyên sâu về việc phát triển tính năng "Scan-to-Path" (Quét để tạo đường dẫn) cho robot công nghiệp 6 trục, sử dụng cảm biến biên dạng laser 2D (2D Laser Profiler) công nghệ Blue Laser của Hikrobot theo cấu hình Eye-in-Hand.

Mục tiêu cốt lõi của nghiên cứu này là giải quyết triệt để ba thách thức kỹ thuật lớn nhất mà các kỹ sư thị giác máy và điều khiển robot phải đối mặt: sự đồng bộ hóa dữ liệu chính xác giữa cảm biến và vị trí robot, toán học phức tạp trong việc ghép nối các hệ quy chiếu động để tạo ra quỹ đạo hàn 6 bậc tự do (6-DOF), và việc lựa chọn kiến trúc phần mềm tối ưu trên nền tảng C#/.NET để xử lý dữ liệu thời gian thực.

Phân tích chỉ ra rằng đối với các bề mặt kim loại có độ phản xạ cao như thép không gỉ hoặc nhôm, việc sử dụng laser bước sóng xanh (405nm) là bắt buộc để giảm thiểu nhiễu quang học do hiện tượng tán xạ dưới bề mặt. Tuy nhiên, độ chính xác của cảm biến sẽ trở nên vô nghĩa nếu không có chiến lược đồng bộ hóa quyết định (deterministic synchronization). Báo cáo khẳng định việc sử dụng kích hoạt phần cứng (Hardware Trigger) thông qua bộ mã hóa trục (Shaft Encoder) là giải pháp vượt trội hoàn toàn so với các phương pháp nội suy dựa trên thời gian (Time-based Interpolation), giúp loại bỏ biến dạng hình học do thay đổi vận tốc robot.

Về mặt thuật toán, báo cáo đi sâu vào việc xây dựng ma trận định hướng cho mỏ hàn. Việc chỉ sử dụng vector pháp tuyến bề mặt là không đủ để xác định tư thế 6 trục; cần phải áp dụng quy trình trực giao hóa Gram-Schmidt kết hợp với vector di chuyển (Travel Vector) để xác định đầy đủ các góc Euler (Rx, Ry, Rz), đảm bảo mỏ hàn duy trì góc công tác và góc đẩy/kéo chính xác.

Cuối cùng, trong cuộc tranh luận giữa việc lựa chọn thư viện PCL (Point Cloud Library) hay Open3D cho môi trường phát triển C#/.NET, báo cáo đề xuất một kiến trúc lai (Hybrid Architecture). Mặc dù Open3D mang lại sự tiện lợi trong tạo mẫu nhanh, PCL vẫn giữ vị thế độc tôn về hiệu năng và độ sâu thuật toán cho các tác vụ công nghiệp nặng. Giải pháp tối ưu

được đề xuất là xây dựng lõi xử lý bằng C++ sử dụng PCL, được bao bọc (wrapped) và gọi từ C# thông qua cơ chế P/Invoke, giúp cân bằng giữa hiệu suất xử lý thời gian thực và khả năng tích hợp hệ thống linh hoạt.

---

2. Kiến trúc Phần cứng và Vật lý Quang học trong Hàn Robot

Việc lựa chọn phần cứng cảm biến và thiết lập cấu hình vật lý là nền tảng quyết định sự thành bại của hệ thống Scan-to-Path. Trong môi trường hàn công nghiệp, cảm biến không chỉ phải chịu được nhiệt độ cao, khói hàn và nhiễu điện từ, mà còn phải đối mặt với thách thức quang học lớn nhất: bề mặt kim loại phản xạ.

2.1 Vật lý của Công nghệ Blue Laser trên Bề mặt Kim loại

Sự lựa chọn cảm biến Hikrobot 2D Blue Laser (ví dụ dòng MV-DP) thay vì các dòng Red Laser truyền thống là một quyết định kỹ thuật chính xác dựa trên các đặc tính vật lý lượng tử của tương tác ánh sáng và vật chất. Các laser đỏ thông thường, hoạt động ở bước sóng 635nm - 660nm, thường gặp vấn đề nghiêm trọng khi quét trên kim loại gia công cơ khí hoặc kim loại bóng.[1]

Vấn đề chính của laser đỏ là hiện tượng tán xạ dưới bề mặt (sub-surface scattering). Do bước sóng dài, các photon đỏ có xu hướng xâm nhập sâu hơn vào lớp oxit hoặc cấu trúc tinh thể bề mặt của kim loại trước khi phản xạ trở lại. Điều này tạo ra một hiệu ứng quang học gọi là "speckle" (nhiễu đốm), khiến cho đường laser thu được trên cảm biến CMOS không phải là một đường mảnh sắc nét mà là một vệt mờ rộng. Khi thuật toán trích xuất tâm đường (line center extraction) hoạt động trên vệt mờ này, nó tạo ra nhiễu tần số cao trên trục Z (độ sâu), thường dao động trong khoảng ±0.05mm đến ±0.1mm, làm giảm độ chính xác đo lường.

Ngược lại, laser xanh (405nm) có năng lượng photon cao hơn và bước sóng ngắn hơn. Đặc tính này giúp chùm tia tương tác chủ yếu ngay tại bề mặt vật liệu, hạn chế tối đa sự xâm nhập vào lớp dưới bề mặt. Kết quả là hình ảnh đường laser thu được trên cảm biến có độ sắc nét cao, biên dạng rõ ràng, cho phép thuật toán đạt độ chính xác "sub-pixel" tốt hơn đáng kể. Hơn nữa, trong quá trình hàn, hồ quang hàn phát ra một lượng lớn bức xạ hồng ngoại (IR) và ánh sáng khả kiến ở dải bước sóng dài. Laser xanh (405nm) nằm ở cực tím của phổ khả kiến, cách xa phổ phát xạ chính của hồ quang hàn, giúp các bộ lọc quang học (bandpass filters) trên camera hoạt động hiệu quả hơn, loại bỏ nhiễu từ ánh sáng hàn.[1]

2.2 Động lực học của Cấu hình Eye-in-Hand

Cấu hình "Eye-in-Hand" (Mắt gắn trên tay), nơi cảm biến được gắn cố định vào mặt bích (flange) của trục thứ 6 của robot, mang lại sự linh hoạt tối đa cho hệ thống Scan-to-Path. Khác với cấu hình Eye-to-Hand (camera cố định, vật di chuyển), cấu hình này cho phép robot

quét các đường hàn dài vô tận hoặc các chi tiết có hình học phức tạp bằng cách di chuyển cảm biến dọc theo đường hàn dự kiến.

Tuy nhiên, cấu hình này đặt ra một bài toán động lực học khắt khe. Độ chính xác của đám mây điểm 3D thu được (P_{cloud}) là hàm số phụ thuộc vào cả độ chính xác của cảm biến (E_{sensor}) và độ chính xác động học của robot (E_{robot}).

$$E_{total} = E_{sensor} + E_{robot}(t)$$

Trong đó $E_{robot}(t)$ bao gồm sai số vị trí khớp, độ rơ (backlash) của hộp số, và rung động cơ khí khi robot di chuyển. Nếu robot rung động với biên độ 0.5mm khi di chuyển, đám mây điểm thu được sẽ bị biến dạng lượn sóng (wavy pattern) tương ứng. Do đó, thuật toán xử lý hậu kỳ bắt buộc phải bao gồm các bộ lọc làm trơn (smoothing filters) để tách biệt tín hiệu hình học thực tế của chi tiết khỏi nhiễu động học của robot.[3]

2.3 Giao diện Kết nối và Yêu cầu Băng thông

Cảm biến Hikrobot DP series thường sử dụng giao thức GigE Vision để truyền dữ liệu hình ảnh và đám mây điểm. Với tốc độ quét cao (lên tới hàng nghìn profile mỗi giây), băng thông mạng là yếu tố không thể xem nhẹ.

Một profile điển hình có thể chứa 2048 điểm chiều rộng (X-axis). Mỗi điểm bao gồm dữ liệu độ sâu (Z) 16-bit và dữ liệu cường độ sáng (Intensity) 8-bit hoặc 16-bit.

Ví dụ tính toán băng thông:

- Độ rộng dữ liệu: 2048 điểm $\times$ 4 bytes (2 byte Depth + 2 byte Intensity) = 8 KB/profile.
- Tần số quét: 2.000 Hz.
- Băng thông yêu cầu: $8\ \text{KB} \times 2000 = 16\ \text{MB/s}$. Mặc dù con số này nằm trong giới hạn lý thuyết của Gigabit Ethernet (125 MB/s), nhưng trong môi trường công nghiệp thực tế, sự ổn định của luồng dữ liệu là quan trọng hơn băng thông đỉnh. Bất kỳ sự gián đoạn nào (packet loss) do hệ điều hành Windows xử lý các tác vụ nền đều dẫn đến mất dữ liệu profile, gây ra các "lỗ hổng" trong đường hàn được tái tạo. Do đó, kiến trúc hệ thống yêu cầu sử dụng thẻ mạng (NIC) chuyên dụng cho camera, hỗ trợ Jumbo Frames (9000 bytes MTU) để giảm tải ngắt CPU (CPU interrupts).[4]

---

3. Chiến lược Đồng bộ hóa: Nội suy vs. Kích hoạt Phần cứng

Vấn đề cốt lõi được nêu trong yêu cầu của người dùng là lựa chọn giữa "Interpolation" (Nội suy) và "Hardware Trigger" (Kích hoạt phần cứng). Phân tích sâu sắc cho thấy đây không chỉ

là sự lựa chọn về phương pháp, mà là sự lựa chọn giữa "ước lượng" và "chính xác".

3.1 Cạm bẫy của Nội suy Tuyến tính (Time-based Interpolation)

Trong phương pháp nội suy dựa trên thời gian, quy trình hoạt động như sau:

1. Camera laser được đặt ở chế độ "Free Run", chụp liên tục ở tần số cố định, ví dụ 500Hz ($\Delta t_{cam} = 2ms$).
2. Máy tính đọc vị trí robot từ bộ điều khiển (thông qua Ethernet) ở một tần số khác, ví dụ 250Hz ($\Delta t_{robot} = 4ms$).
3. Khi nhận được một profile laser tại thời điểm t, phần mềm tìm hai vị trí robot gần nhất P_{t1} và P_{t2} để nội suy vị trí của robot tại thời điểm t.

Công thức nội suy tuyến tính cơ bản:

$$P_{robot}(t) = P_{t1} + \frac{t - t1}{t2 - t1}(P_{t2} - P_{t1})$$

Tại sao phương pháp này thất bại trong hàn robot:

- **Giả định sai lầm về vận tốc:** Công thức trên giả định vận tốc robot là hằng số giữa $t1$ và $t2$. Thực tế, robot hàn liên tục tăng tốc và giảm tốc, đặc biệt là khi vào cua hoặc bắt đầu/kết thúc đường hàn. Sự thay đổi vận tốc phi tuyến tính này làm cho phép nội suy tuyến tính bị sai lệch, dẫn đến đám mây điểm bị co giãn không đều.
- **Độ trễ mạng ngẫu nhiên (Network Jitter):** Thời gian t mà phần mềm ghi nhận thường là thời gian gói tin *đến* máy tính, không phải thời gian *thu thập* thực tế. Hệ điều hành Windows không phải là hệ điều hành thời gian thực (RTOS). Độ trễ ngẫu nhiên của ngăn xếp mạng (network stack) có thể dao động từ 1ms đến 20ms. Ở tốc độ hàn 20mm/s, độ trễ 20ms tương đương sai số vị trí 0.4mm – một con số quá lớn cho hàn chính xác.[5]

3.2 Giải pháp Tối ưu: Kích hoạt Phần cứng (Hardware Trigger)

Để loại bỏ hoàn toàn các biến số về thời gian và vận tốc, hệ thống Scan-to-Path chuyên nghiệp bắt buộc phải sử dụng phương pháp **Kích hoạt theo Không gian (Spatial Triggering)** thông qua bộ mã hóa (Encoder).

Trong kiến trúc này, việc thu thập dữ liệu không được điều khiển bởi đồng hồ (clock) mà được điều khiển bởi quãng đường di chuyển thực tế.

1. **Nguồn tín hiệu:** Robot cung cấp tín hiệu xung (pulse) tương ứng với quãng đường di chuyển của TCP (Fast Output / Path Output) hoặc sử dụng một bộ mã hóa vòng quay vật lý gắn trên trục chuyển động.

2. **Cơ chế:** Tín hiệu này được đưa trực tiếp vào cổng I/O tốc độ cao của camera Hikrobot (thường là Line 0 hoặc Line 2).
3. **Cấu hình:** Camera được cài đặt để chụp một profile sau mỗi N xung nhận được.

Lợi ích vượt trội:

- **Độc lập với Vận tốc:** Dù robot di chuyển nhanh (100mm/s) hay chậm (5mm/s), mật độ điểm quét trên bề mặt vật thể luôn không đổi (ví dụ: 1 profile mỗi 0.5mm). Điều này đảm bảo đám mây điểm 3D không bị biến dạng hình học theo phương dọc trục.[5]
- **Độ chính xác tất định:** Loại bỏ hoàn toàn sai số do độ trễ truyền thông và jitter của hệ điều hành. Vị trí của mỗi profile được gắn cứng với vị trí cơ khí của robot.

Triển khai trên Hikrobot MVS: Để thiết lập chế độ này, cần cấu hình các tham số SDK như sau [4]:

- TriggerMode: On
- TriggerSource: ShaftEncoderModuleOut (Sử dụng mô-đun xử lý encoder tích hợp trong camera).
- EncoderSourceA / EncoderSourceB: Chọn chân tín hiệu vật lý (Line 0, Line 3...).
- EncoderDivider: Tham số quan trọng nhất. Đây là bộ chia tần số. Ví dụ, nếu bộ mã hóa phát ra 10.000 xung/mm nhưng ta chỉ cần độ phân giải quét 0.1mm, ta cần thiết lập bộ chia để camera chỉ chụp sau mỗi 1.000 xung.

$$\text{Divider} = \frac{\text{Encoder Resolution (pulse/mm)}}{\text{Desired Scan Density (profile/mm)}}$$

Kết luận cho phần này: Đối với hệ thống hàn 6 trục, tuyệt đối không sử dụng phương pháp nội suy thời gian. Hãy sử dụng tính năng "Distance Output" hoặc "Position Compare" của bộ điều khiển robot để xuất xung kích hoạt phần cứng tới camera Hikrobot.

---

4. Cơ sở Toán học: Ghép nối Tọa độ và Tạo Quỹ đạo 6-DOF

Đây là phần trọng tâm giải quyết yêu cầu "vấn đề toán học ghép nối tọa độ" của người dùng. Chúng ta cần thiết lập chuỗi biến đổi từ điểm ảnh 2D sang tư thế hàn 6D hoàn chỉnh.

4.1 Định nghĩa Hệ tọa độ

Chúng ta làm việc với 4 hệ quy chiếu (frames) chính:

1. **Hệ tọa độ Cảm biến** $\{S\}$**:** Gắn liền với camera. Trục Z_S là chiều sâu, X_S là chiều rộng tia laser.

2. **Hệ tọa độ Mặt bích $\{F\}$ (Flange/Tool0):** Điểm gắn kết cơ khí cuối cùng của robot.
3. **Hệ tọa độ Cơ sở $\{B\}$ (Base/World):** Gốc cố định của robot.
4. **Hệ tọa độ Mỏ hàn $\{T\}$ (Tool/TCP):** Điểm làm việc thực tế (đầu dây hàn).

4.2 Hiệu chuẩn Hand-Eye: Giải phương trình $AX = ZB$

Để biến đổi một điểm đo được từ camera $\{S\}$ về hệ cơ sở $\{B\}$, ta cần biết ma trận biến đổi cố định giữa mặt bích robot và camera, ký hiệu là ${}^{F}T_{S}$.

Phương trình biến đổi cho một điểm P:

$$P_B = {}^{B}T_{F}(t) \cdot {}^{F}T_{S} \cdot P_S$$

Trong đó:

- P_B: Tọa độ điểm trong không gian robot (cần tìm).
- P_S: Tọa độ điểm do camera Hikrobot trả về (dữ liệu thô).
- ${}^{B}T_{F}(t)$: Vị trí hiện tại của robot (nhận từ controller).
- ${}^{F}T_{S}$: Ma trận hiệu chuẩn Hand-Eye (chưa biết).

Để tìm ${}^{F}T_{S}$, ta thực hiện quy trình hiệu chuẩn bằng cách di chuyển robot đến nhiều vị trí khác nhau và quan sát một vật thể mẫu cố định. Bài toán này dẫn đến phương trình ma trận kinh điển $AX = ZB$ (hoặc $AX = XB$ tùy ký hiệu):

- A: Chuyển động tương đối của mặt bích robot giữa hai vị trí.
- B: Chuyển động tương đối của camera (được tính toán từ việc quan sát vật mẫu).
- X: Ma trận cần tìm ${}^{F}T_{S}$.

Nghiên cứu [10] chỉ ra rằng các phương pháp giải tích như của **Park & Martin** hoặc **Tsai** (sử dụng đại số Lie hoặc Quaternion kép) mang lại độ ổn định cao nhất. Trong thực tế triển khai C#, ta có thể sử dụng thư viện OpenCvSharp với hàm Cv2.CalibrateHandEye() để giải quyết vấn đề này một cách chính xác.

4.3 Dẫn xuất Tư thế 6-DOF (Rx, Ry, Rz) từ Pháp tuyến và Tiếp tuyến

Một sai lầm phổ biến khi lập trình Scan-to-Path là chỉ sử dụng vector pháp tuyến bề mặt để định hướng mỏ hàn. Một vector pháp tuyến chỉ khóa được 2 bậc tự do (Pitch và Yaw). Mỏ hàn vẫn có thể xoay tự do quanh trục pháp tuyến (Roll) mà không thay đổi quan hệ với bề mặt. Để hàn chính xác, ta cần khóa cả 6 bậc tự do, đảm bảo mỏ hàn hướng theo đường hàn (Travel angle) và nghiêng đúng góc (Work angle).

Quy trình toán học đề xuất sử dụng thuật toán **Trực giao hóa Gram-Schmidt**:

Bước 1: Xác định Vector Pháp tuyến (n)

Sử dụng thuật toán PCA (Principal Component Analysis) trên vùng lân cận của điểm hàn để tìm vector pháp tuyến bề mặt n.

Giả sử ta muốn trục Z của mỏ hàn (z_{tool}) trùng với phương pháp tuyến (hướng vào bề mặt):

$$z_{tool} = -n$$

Bước 2: Xác định Vector Di chuyển thô (v)

Vector di chuyển được xác định bởi điểm hàn hiện tại P_i và điểm tiếp theo P_{i+1}:

$$v = P_{i+1} - P_i$$

Bước 3: Trực giao hóa (Gram-Schmidt) để tìm Vector Tiếp tuyến (x_{tool})

Vector v có thể không vuông góc với n (ví dụ khi leo dốc). Ta cần chiếu v lên mặt phẳng tiếp tuyến:

$$x_{proj} = v - (v \cdot n)n$$

Chuẩn hóa để có vector đơn vị trục X của mỏ hàn:

$$x_{tool} = \frac{x_{proj}}{||x_{proj}||}$$

Lúc này, x_{tool} chỉ phương di chuyển của robot, vuông góc với pháp tuyến.

Bước 4: Xác định Vector trùng pháp tuyến (y_{tool})

Trục Y được xác định bằng tích có hướng (cross product) để đảm bảo hệ tọa độ thuận:

$$y_{tool} = z_{tool} \times x_{tool}$$

Bước 5: Tạo Ma trận Quay và Chuyển đổi sang Euler (Rx, Ry, Rz)

Ma trận quay R được tạo thành từ 3 vector cột:

$$R = \begin{bmatrix} x_{tool} & y_{tool} & z_{tool} \end{bmatrix}$$

Từ ma trận R, ta cần trích xuất các góc Euler (Rx, Ry, Rz) theo quy ước của hãng robot.

- **Lưu ý quan trọng:** Mỗi hãng robot sử dụng một quy ước khác nhau.
  - **Fanuc/Yaskawa:** Sử dụng quy ước W, P, R tương ứng với quay quanh $Z - Y - X$ (Extrinsic).
  - **Universal Robots:** Sử dụng Axis-Angle (Vector quay).
  - **KUKA:** Sử dụng Euler $Z - Y - X$ (A, B, C).

Giả sử quy ước $Z - Y - X$ (Rz, Ry, Rx), công thức trích xuất là [12]:

$$Ry = \text{atan2}(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2})$$

$$Rz = \text{atan2}(r_{21} / \cos(Ry), r_{11} / \cos(Ry))$$

$$Rx = \text{atan2}(r_{32} / \cos(Ry), r_{33} / \cos(Ry))$$

Xử lý Điểm Kỳ dị (Singularity Handling):

Nếu $Ry = \pm 90^\circ$, $\cos(Ry) = 0$, dẫn đến lỗi chia cho 0 (Gimbal Lock). Trong thuật toán, bắt buộc phải kiểm tra điều kiện:

if (Abs(r31) > 0.9999): Thiết lập $Rx = 0$ và tính Rz theo công thức thay thế. Đây là bước sống còn để tránh robot bị lỗi dừng đột ngột khi hàn ở tư thế thẳng đứng.

5. Phân tích Thư viện: PCL vs. Open3D cho C#/.NET

Câu hỏi đặt ra là lựa chọn công cụ nào để hiện thực hóa các thuật toán trên trong môi trường.NET. Dưới đây là bảng so sánh chi tiết dựa trên các tiêu chí kỹ thuật [14]:

Tiêu chí	PCL (Point Cloud Library)	Open3D
Ngôn ngữ lõi	C++ thuần túy, sử dụng Template nặng.	C++11/14, nhưng ưu tiên Python binding.
Hỗ trợ C#/.NET	Kém. Các wrapper như PclSharp đã ngừng bảo trì hoặc thiếu tính năng.	Trung bình/Thấp. Open3D.Net đang phát triển nhưng chưa ổn định cho công nghiệp (Experimental).
Thuật toán chuyên sâu	Rất mạnh. Có sẵn Segmentation (V-groove), Registration, Filtering, Features (PFH/FPFH).	Mạnh về Deep Learning và Visualization. Thiếu một số thuật toán hình học cổ điển chuyên sâu.
Hiệu năng Real-time	Rất cao (KD-Tree FLANN tối ưu hóa tốt).	Tốt, nhưng tối ưu hóa cho Batch processing (Python) hơn là Real-time loop đơn lẻ.
Kiến trúc đề xuất	**C++ Native DLL** (Viết code C++ gọi PCL, xuất hàm C cho C#).	Sử dụng cho Visualization (WPF) hoặc Prototyping bằng Python.

Khuyến nghị Kiến trúc Lai (Hybrid Architecture):

Đối với một ứng dụng công nghiệp yêu cầu xử lý thời gian thực (<30ms/frame), việc phụ thuộc vào các thư viện wrapper C# không chính chủ (như PclSharp) là rủi ro lớn về bảo trì và hiệu năng (chi phí marshalling dữ liệu lớn).

Kiến trúc tối ưu nhất là:

1. **Lớp Lõi (Core Layer - C++):** Viết một thư viện liên kết động (DLL) bằng C++ sử dụng PCL gốc. Thư viện này thực hiện các tác vụ nặng: Lọc nhiễu, RANSAC tìm mặt phẳng, tính

toán pháp tuyến.
2. **Lớp Giao diện (Interop Layer):** Xuất các hàm C đơn giản từ DLL này, ví dụ: ProcessPointCloud(float* data, int size, float* outputPose).
3. **Lớp Ứng dụng (App Layer - C#):** Sử dụng `` để gọi hàm C++ và truyền con trỏ dữ liệu thô (raw pointer) từ SDK Hikrobot vào thẳng DLL C++. Cách tiếp cận "Zero-copy" này đảm bảo tốc độ xử lý tối đa.[16]

---

6. Chiến lược Xử lý Đám mây điểm & Giải thuật Cốt lõi

Để trích xuất đường hàn từ dữ liệu thô, chúng ta cần một quy trình xử lý (pipeline) tuần tự và chặt chẽ.

6.1 Lọc Nhiễu Phản xạ (Reflection Removal)

Trước khi áp dụng bất kỳ thuật toán hình học nào, cần loại bỏ nhiễu do kim loại bóng gây ra.

Tận dụng tính năng của Hikrobot: Cảm biến trả về hai luồng dữ liệu - Bản đồ độ sâu (Depth Map) và Bản đồ cường độ sáng (Brightness/Intensity Map).

- **Thuật toán:** Duyệt qua từng điểm ảnh. Nếu giá trị cường độ sáng nằm ngoài ngưỡng an toàn (quá tối < 10 hoặc quá sáng bão hòa > 250), hãy gán giá trị độ sâu tương ứng là NaN (Not a Number). Đây là bộ lọc rẻ tiền nhưng hiệu quả nhất để loại bỏ các tia phản xạ đa đường (multipath reflection) và nhiễu đốm.[17]

6.2 Phân đoạn Đường hàn V-Groove

Đối với loại mối hàn phổ biến là V-Groove (vát mép chữ V), thuật toán hiệu quả nhất là **RANSAC (Random Sample Consensus)**.

1. **Bước 1:** Cắt profile laser thành hai phần trái và phải.
2. **Bước 2:** Áp dụng RANSAC để tìm mô hình đường thẳng (Line Model) tốt nhất cho sườn trái (L_{left}) và sườn phải (L_{right}), tự động loại bỏ các điểm nhiễu ngoại lai (outliers).
3. **Bước 3:** Tính giao điểm của L_{left} và L_{right} . Giao điểm này chính xác là đáy mối hàn (Root Point) cần tìm.[19]

6.3 Làm trơn Quỹ đạo (Path Smoothing)

Dữ liệu thô từ cảm biến, dù đã lọc, vẫn sẽ có độ rung (jitter) nhất định. Nếu gửi trực tiếp tọa độ này cho robot, các động cơ servo sẽ bị rung giật.

Cần áp dụng bộ lọc **Moving Average** (Trung bình động) hoặc cao cấp hơn là **B-Spline Interpolation** trên chuỗi các điểm hàn ($P_t, P_{t-1}, P_{t-2} \cdots$) để tạo ra một quỹ đạo mượt mà cho robot bám theo.

---

7. Hướng dẫn Triển khai & Tối ưu hóa Hệ thống

Phần này cung cấp các hướng dẫn thực tế khi làm việc với SDK Hikrobot và tích hợp hệ thống.

7.1 Thiết lập SDK Hikrobot trong C#

Sử dụng MVS SDK (Machine Vision Software SDK) của Hikrobot. Lưu ý quan trọng về mô hình luồng (threading model):

- **Callback:** SDK cung cấp cơ chế RegisterImageCallBack. Hàm callback này được gọi từ luồng của driver camera. **Tuyệt đối không** xử lý nặng (như tính toán PCL) bên trong callback này, vì nó sẽ chặn luồng nhận dữ liệu, gây mất frame.
- **Producer-Consumer:** Trong callback, chỉ thực hiện việc copy dữ liệu vào một hàng đợi an toàn (ConcurrentQueue). Một luồng xử lý riêng biệt (Worker Thread) sẽ lấy dữ liệu từ hàng đợi và gọi sang C++ DLL để xử lý.

7.2 Bù Trễ Hệ thống (Latency Compensation)

Dù sử dụng Hardware Trigger, vẫn có độ trễ vật lý giữa thời điểm "chụp ảnh" và thời điểm "robot nhận lệnh".

Tổng độ trễ $T_{delay} \approx T_{exposure} + T_{readout} + T_{process} + T_{network}$.

Giả sử tổng độ trễ là 30ms. Nếu robot hàn ở tốc độ 20mm/s, vị trí tính toán được đã bị trễ $0.6mm$ so với thực tế.

- **Giải pháp:** Cần cộng thêm một vector bù vào vị trí gửi cho robot:

  $$P_{target} = P_{measured} + v_{robot} \times T_{delay}$$

  Trong đó v_{robot} là vận tốc hiện thời của robot.

8. Kết luận và Kiến nghị

Việc phát triển tính năng Scan-to-Path cho robot hàn 6 trục là một bài toán đa ngành phức tạp. Dựa trên các phân tích chuyên sâu, báo cáo đưa ra các kiến nghị chiến lược sau:

1. **Về Phần cứng:** Kiên quyết sử dụng cảm biến Blue Laser (405nm) để đảm bảo chất lượng dữ liệu đầu vào trên bề mặt kim loại. Thiết lập kết nối kích hoạt phần cứng (Hardware Trigger) thông qua bộ mã hóa để đảm bảo tính toàn vẹn hình học của dữ liệu quét, loại bỏ hoàn toàn phương pháp nội suy thời gian.
2. **Về Toán học:** Không sử dụng các phép tính pháp tuyến đơn giản. Hãy triển khai thuật toán trực giao hóa Gram-Schmidt để xây dựng đầy đủ khung tọa độ 6-DOF, đặc biệt chú trọng việc xử lý điểm kỳ dị (Singularity) khi chuyển đổi sang góc Euler.

3. **Về Phần mềm:** Áp dụng kiến trúc lai: Sử dụng C# cho giao diện và quản lý hệ thống, nhưng ủy thác toàn bộ việc xử lý toán học và đám mây điểm cho một module C++ Native sử dụng thư viện PCL. Đây là con đường duy nhất để đạt được hiệu năng thời gian thực và độ ổn định công nghiệp lâu dài.

Bằng cách tuân thủ các nguyên tắc kiến trúc này, hệ thống sẽ đạt được khả năng theo dõi đường hàn chính xác, thích ứng linh hoạt với sai số phôi, và đáp ứng các tiêu chuẩn khắt khe của môi trường sản xuất công nghiệp hiện đại.

Works cited

1. HIKROBOT Introduces 3D Laser Profile Sensor in India for Various Application Scenarios, accessed February 1, 2026, https://www.hikvisionindia.com/press-release/hikrobot-introduces-3d-laser-profile-sensor-india-various-application-scenarios
2. 【Exhibition】SPS 2024 – Advancing Smart Production with Hikrobot, accessed February 1, 2026, https://www.hikrobotics.com/en/news/newsdetail/?id=53
3. Simultaneous Hand–Eye and Intrinsic Calibration of a Laser Profilometer Mounted on a Robot Arm - PMC - PubMed Central, accessed February 1, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC7913545/
4. GigE Line Scan Camera User Manual - Hikrobot, accessed February 1, 2026, https://www.hikrobotics.com/en2/Hikrobotics/Machine%20Vision/01%20Product/%E5%B7%A5%E4%B8%9A%E7%BA%BF%E9%98%B5%E7%9B%B8%E6%9C%BA/%E7%94%A8%E6%88%B7%E6%89%8B%E5%86%8C/UD24388B_GigE%20Line%20Scan%20Camera%20User%20Manual_V3.4.32_20210621.pdf
5. The Complete Guide to Machine Vision Triggering for High-Speed Image Acquisition, accessed February 1, 2026, https://www.elementaryml.com/blog/the-complete-guide-to-machine-vision-triggering-for-high-speed-image-acquisition
6. Subframe-Level Synchronization in Multi-Camera System Using Time-Calibrated Video, accessed February 1, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC11548676/
7. Performance Line Scan Cameras, Software, and Systems for AI and Machine Learning Applications - Emergent Vision Technologies, accessed February 1, 2026, https://emergentvisiontec.com/tech-portal/performance-line-scan-cameras-software-and-systems-for-ai-and-machine-learning-applications/
8. Hikrobot Machine Vision Software User Manual - CNC Utilities (CNCU.CO.ZA), accessed February 1, 2026, https://cncu.co.za/FlatCUT-Flatbed-Cutting-Machine/MDR%20Oscillating%20Blade%20CNC%20Router/Hikrobot%20MVS%20(Machine%20Vision%20Software)%20STD%204.0.0%20(221208)%20Installation/Applications/Win32/doc/User_Manual_English/pdf.pdf
9. Camera Link Line Scan Camera, accessed February 1, 2026, https://cache.miancp.com:2083/data/www.szcco-vision.com/static/upload/file/20

250318/1742292418317958.pdf
10. Robotic Hand–Eye Calibration Method Using Arbitrary Targets Based on Refined Two-Step Registration - MDPI, accessed February 1, 2026, https://www.mdpi.com/1424-8220/25/10/2976
11. Laser Profiler and Camera Fusion for 3D Mobile Scanning System, accessed February 1, 2026, https://repository.dl.itc.u-tokyo.ac.jp/record/2004331/files/A36430.pdf
12. rotm2eul - Convert rotation matrix to Euler angles - MATLAB - MathWorks, accessed February 1, 2026, https://www.mathworks.com/help/robotics/ref/rotm2eul.html
13. Rotation Matrix To Euler Angles | LearnOpenCV #, accessed February 1, 2026, https://learnopencv.com/rotation-matrix-to-euler-angles/
14. Point Cloud Libraries: PCL vs. Open3D for Processing Efficiency - Patsnap Eureka, accessed February 1, 2026, https://eureka.patsnap.com/article/point-cloud-libraries-pcl-vs-open3d-for-processing-efficiency
15. Performance comparisons against PCL and Open3D in common operations.... - ResearchGate, accessed February 1, 2026, https://www.researchgate.net/figure/Performance-comparisons-against-PCL-and-Open3D-in-common-operations-Left-column-running_fig2_328374976
16. The Complete Guide to C++ and C# Integration using .NET | by TinTin Winata - Medium, accessed February 1, 2026, https://medium.com/@tintinwinata/bridging-worlds-a-practical-guide-to-c-and-c-integration-2ef6989f6ede
17. Calibration Settings for Optimal Quality From the Laser Line Probe (LLP), accessed February 1, 2026, https://knowledge.faro.com/Hardware/FaroArm_and_ScanArm/FaroArm_and_ScanArm/Calibration_Settings_for_Optimal_Quality_From_the_Laser_Line_Probe-LLP
18. 3D Laser Profile Sensor - Hikrobot - Machine Vision Products, accessed February 1, 2026, https://www.hikrobotics.com/en/machinevision/visionproduct/?typeId=99&id=157
19. Weld seam recognition algorithm based on a fast point cloud plane fitting method, accessed February 1, 2026, https://opg.optica.org/josaa/abstract.cfm?uri=josaa-43-2-307
20. Flowchart for weld seam feature points extraction of V-groove butt joint type - ResearchGate, accessed February 1, 2026, https://www.researchgate.net/figure/Flowchart-forweld-seam-feature-points-extraction-of-V-groove-butt-joint-type_fig1_372161768