多関節ロボットのパスの情報抽出手法の実装と評価

○渡辺 翼（北九州市立大学），　西田 健（北九州市立大学）

Implementation and evaluation of information extraction methods for paths of articulated robots

○Tsubasa WATANABE and Takeshi NISHIDA (The University of Kitakyushu)

Abstract: We propose a method for encoding and decoding paths of multi-joint robots. The proposed method encodes and stores the paths in the robot configuration space in a format that is independent of the robot's joint lengths and start and goal positions. It also enables fast, context-sensitive decoding of the stored path information to reconstruct the path. In this paper, we present an implementation of the proposed method using ROS and verify its effectiveness.

１．はじめに

多品種変量生産（HVP: high-mix, variable-volume production）の自動化には，ロボットのパスを迅速かつ適切に生成する機能が重要である[1,2]. 一方で，従来手法には調整パラメータが多く計算コストが高いなどの課題がある[2]. さらに HVP には，ロボットの運動学パラメータが頻繁に変化するタスクが多く存在する．例えば物体を把持して運搬するタスクでは，把持対象をロボットの一部としたパス生成が必要であるため，把持対象が変化する度に運動学パラメータが変化する．また，グリッパが対象物を解放する前後にも運動学パラメータが変化する場合がある．したがって， HVP に多関節ロボットを活用するためには，運動学パラメータの変化に頑健なパス生成手法が重要である．

一般に，多関節ロボットのパスは，ロボットの周辺環境の三次元空間（タスク空間もしくは T 空間），および各関節角度で張られる高次のロボットコンフィギュレーション空間（C 空間）の両方を複数回参照して計画する．障害物との衝突や自己干渉のチェックは T 空間で実施する必要があるが，衝突判定は計算コストが高いため，タスクが頻繁に変化する HVP では T 空間における衝突判定の回数を減ずる工夫が重要である．

これらの課題に対する従来研究には，ランダム探索手法[3]や，評価関数の勾配探索手法[4]などがある．しかし，これらは状況に応じたパスの発見的手法であり，環境やタスクの複雑さに応じてパスの生成時間が長くなる傾向がある．機械学習の汎化能力によって課題解決を試みる研究[5]も多くなされているが，環境・グリッパ・ワーク・タスクの組み合わせを網羅した膨大なパスのデータセットが必要になることや，学習結果にロボットの運動学パラメータが畳み込まれるため異なる構成のロボットへのデータ転用が困難であるといった課題がある．

著者らは現在までに，任意の開始・終了位置に対してパスの概形を復元するためには，C 空間のパスの開始・終了位置を結ぶ線分とウェイポイントの相関を規定する三種類の情報，①分割点，②偏差ノルム，③直交ベクトルが必要であることを見出している[6]. また，C 空間におけるパスのこれらの情報をエンコードする手法を提案している．さらに，エンコードされたパスの情報を，異なる開始・終了位置に適応させて高速にデコードする手法も併せて提案している．本論文では，本手法を ROS[7]を用いて実装し，その効果を検証する．

２．パスの情報抽出手法

2.1 定義

C空間の座標系を$\Sigma_C \subset \mathbb{R}^N$と表す．$N$はロボットの関節数を表す．所与のパスのスタート，ゴールおよびパスのウェイポイントを以下のように表す．

$$\boldsymbol{s} \triangleq [s_1 \quad \cdots \quad s_N]^T \in \mathbb{R}^N \tag{1}$$

$$\boldsymbol{g} \triangleq [g_1 \quad \cdots \quad g_N]^T \in \mathbb{R}^N \tag{2}$$

$$\boldsymbol{w}^{(m)} \triangleq \left[w_1^{(m)} \quad \cdots \quad w_N^{(m)}\right]^T \in \mathbb{R}^N \tag{3}$$

ここで，$m = 1, \cdots, M$はウェイポイントの番号，Mはウェイポイントのインデクスを表す．さらに，スタートからゴールへのベクトルは以下で表される．

$$\boldsymbol{v} \triangleq \boldsymbol{g} - \boldsymbol{s} \tag{4}$$

また，$\boldsymbol{v}$ の正規化ベクトルは以下で表される．

$$\boldsymbol{e} \triangleq |\boldsymbol{v}| \in \mathbb{R}^N \tag{5}$$

ここで$|\cdot|$はベクトルの正規化を表す．また，スタートとゴールを結ぶ線分をLとする．さらに，所与の$\boldsymbol{s}$と$\boldsymbol{g}$に対して，適切なパスプランニング手法の適用によりパ

スPが得られているとする．

$$P \triangleq \left(\boldsymbol{s}, \boldsymbol{w}^{(1)}, \cdots, \boldsymbol{w}^{(m)}, \cdots, \boldsymbol{w}^{(M)}, \boldsymbol{g}\right) \tag{6}$$

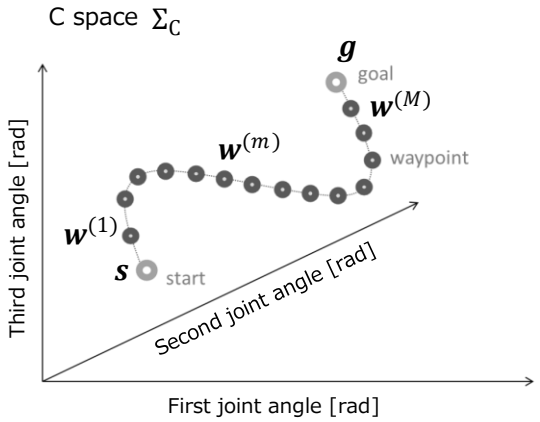
これらの概要を Fig.1(a)に示す．

2.2 パスのエンコード

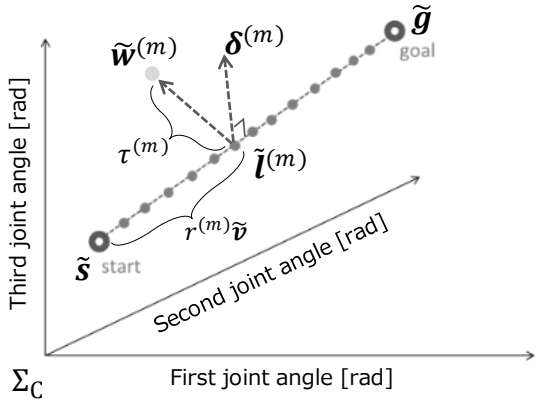
文献[1]で示されたアルゴリズムをTable 1に示す．この中の$\|\cdot\|$はベクトルのユークリッドノルムを表す．Step 1において，スタートからゴールに向かうベクトルを求める．Step 2では，ウェイポイント$\boldsymbol{w}^{(m)}$から線分Lに下した垂線の足$\boldsymbol{l}^{(m)}$（分割点）を求める．また，スタートからの$\boldsymbol{l}^{(m)}$の$r^{(m)}$，$\boldsymbol{w}^{(m)}$と$\boldsymbol{l}^{(m)}$の距離$\tau^{(m)}$（偏差ノルム），$\boldsymbol{l}^{(m)}$を起点としたベクトル$\boldsymbol{\delta}^{(m)}$（直交ベクトル）を求める．Step 3では，それらをまとめてパスシードSを出力する．このアルゴリズムのStep 2において，以下の外積演算を実施する．

$$\boldsymbol{\delta}^{(m)} \triangleq \left|\boldsymbol{e} \times \left(\boldsymbol{w}^{(m)} - \boldsymbol{l}^{(m)}\right)\right| \tag{7}$$

この $\boldsymbol{\delta}^{(m)}$ によって，L に対する $\boldsymbol{w}^{(m)}$ の存在する方向を特定することが可能になる．ただし，外積演算は $N = 0, 1, 3, 7$ のみに定義されるので，例えば $N = 6$ の場合に



(a) Approximate shape of a path in C-space (represented up to three axes).



(b) Path information extraction.

Fig.1 Overview of the path encoding process.

式(7)の演算のためには工夫が必要となる．本研究では，スタートからゴールへ向かうベクトル$\boldsymbol{e}$，ウェイポイント$\boldsymbol{w}^{(m)}$，および垂線の足$\boldsymbol{l}^{(m)}$を以下のように二つの三次元ベクトルに分割して計算することとした．

$$\boldsymbol{e}_{1,2,3} \triangleq [e_1 \quad e_2 \quad e_3]^T \tag{8}$$

$$\boldsymbol{e}_{4,5,6} \triangleq [e_4 \quad e_5 \quad e_6]^T \tag{9}$$

$$\boldsymbol{w}_{1,2,3}^{(m)} \triangleq \left[w_1^{(m)} \quad w_2^{(m)} \quad w_3^{(m)}\right]^T \tag{10}$$

$$\boldsymbol{w}_{4,5,6}^{(m)} \triangleq \left[w_4^{(m)} \quad w_5^{(m)} \quad w_6^{(m)}\right]^T \tag{11}$$

$$\boldsymbol{l}_{1,2,3}^{(m)} \triangleq \left[l_1^{(m)} \quad l_2^{(m)} \quad l_3^{(m)}\right]^T \tag{12}$$

$$\boldsymbol{l}_{4,5,6}^{(m)} \triangleq \left[l_4^{(m)} \quad l_5^{(m)} \quad l_6^{(m)}\right]^T \tag{13}$$

これによって，以下の外積演算が成立する．

$$\boldsymbol{\delta}_{1,2,3}^{(m)} \triangleq \left|\boldsymbol{e}_{1,2,3} \times \left(\boldsymbol{w}_{1,2,3}^{(m)} - \boldsymbol{l}_{1,2,3}^{(m)}\right)\right| \in \mathbb{R}^3 \tag{17}$$

$$\boldsymbol{\delta}_{4,5,6}^{(m)} \triangleq \left|\boldsymbol{e}_{4,5,6} \times \left(\boldsymbol{w}_{4,5,6}^{(m)} - \boldsymbol{l}_{4,5,6}^{(m)}\right)\right| \in \mathbb{R}^3 \tag{18}$$

これらを統合することでベクトル$\boldsymbol{\delta}^{(m)}$が算出できる．

$$\boldsymbol{\delta}^{(m)} \triangleq \begin{bmatrix} \boldsymbol{\delta}_{1,2,3}^{(m)} \\ \boldsymbol{\delta}_{4,5,6}^{(m)} \end{bmatrix} \in \mathbb{R}^6 \tag{19}$$

また，垂線の足$\tau^{(m)}$は，二種類のユークリッドノルムを算出して統合することとした．

$$\tau_{1,2,3}^{(m)} \triangleq \left\|\boldsymbol{w}_{1,2,3}^{(m)} - \boldsymbol{l}_{1,2,3}^{(m)}\right\| \tag{14}$$

$$\tau_{4,5,6}^{(m)} \triangleq \left\|\boldsymbol{w}_{4,5,6}^{(m)} - \boldsymbol{l}_{4,5,6}^{(m)}\right\| \tag{15}$$

$$\boldsymbol{\tau}^{(m)} \triangleq \left[\tau_{1,2,3}^{(m)} \quad \tau_{4,5,6}^{(m)}\right]^T \tag{16}$$

以上で求めた情報を Step 3 に示すような行列の集合Sで保持する．これをパスシードと呼ぶ．

ロボットの関節数が$N = 3, 7$の場合には，Step 2 の外

Table 1. Algorithm of path encoding.

Given: $P \triangleq \left(\boldsymbol{s}, \boldsymbol{w}^{(1)}, \cdots, \boldsymbol{w}^{(m)}, \cdots, \boldsymbol{w}^{(M)}, \boldsymbol{g}\right)$

Step 1: Calculate as follows:

$\boldsymbol{v} = \boldsymbol{g} - \boldsymbol{s}$,

$\boldsymbol{e} = |\boldsymbol{v}|$.

Step 2: Calculate for all m follows:

$\boldsymbol{l}^{(m)} = \boldsymbol{s} + \left\{\left(\boldsymbol{w}^{(m)} - \boldsymbol{s}\right)^T \boldsymbol{e}\right\} \boldsymbol{e}$,

$r^{(m)} = \left\|\boldsymbol{l}^{(m)} - \boldsymbol{s}\right\| / \|\boldsymbol{v}\|$,

$\tau^{(m)} = \left\|\boldsymbol{w}^{(m)} - \boldsymbol{l}^{(m)}\right\|$,

$\boldsymbol{\delta}^{(m)} = \left|\boldsymbol{e} \times \left(\boldsymbol{w}^{(m)} - \boldsymbol{l}^{(m)}\right)\right|$.

Step 3: Output $\boldsymbol{S} = \{\boldsymbol{R}, \boldsymbol{T}, \boldsymbol{\Delta}\}$:

$\boldsymbol{R} = \left[\boldsymbol{r}^{(1)} \cdots \boldsymbol{r}^{(m)} \cdots \boldsymbol{r}^{(M)}\right] \in \mathbb{R}^{M}$,

$\boldsymbol{T} = \left[\boldsymbol{\tau}^{(1)} \cdots \boldsymbol{\tau}^{(m)} \cdots \boldsymbol{\tau}^{(M)}\right] \in \mathbb{R}^{M}$,

$\Delta = [\delta^{(1)} \quad \cdots \quad \delta^{(M)}] \in \mathbb{R}^{6 \times M}$.

積演算が可能であるが，それ以外の軸数の場合には，上述のように各ベクトルを分割して外積演算を実施する必要がある．

2.3 パスのデコード

前述のパスシードSを利用してパスを生成するアルゴリズムをTable 2に示す．

まず，新しいスタート位置$\tilde{\boldsymbol{s}}$，新しいゴール位置$\tilde{\boldsymbol{g}}$，およびパスシードSが与えられる．次に，Step 1において，スタートからゴールへ向かうベクトルを求め，Step 2において，スタートとゴールを結ぶ線分$\tilde{L}$上に，$r^{(m)}$を利用して$\tilde{\boldsymbol{l}}^{(m)}$を以下のように配置する．

$$\tilde{\boldsymbol{l}}^{(m)} = \tilde{\boldsymbol{s}} + \boldsymbol{r}^{(m)}\tilde{\boldsymbol{v}} \tag{17}$$

次に，$\boldsymbol{\delta}^{(m)}$と$\tilde{\boldsymbol{e}}$の外積によって，$\tilde{\boldsymbol{l}}^{(m)}$を足として$\tilde{L}$に直交するベクトルを求め，ノルム$\tau^{(m)}$を用いて$\tilde{\boldsymbol{w}}^{(m)}$の位置を以下のように算出する．

$$\tilde{\boldsymbol{w}}^{(m)} = \tilde{\boldsymbol{l}}^{(m)} + \tau^{(m)}\left(\boldsymbol{\delta}^{(m)} \times \tilde{\boldsymbol{e}}\right) \quad (\forall m) \tag{18}$$

ここで，$N = 3,7$以外の場合にはベクトルを分割して外積演算を実施する必要がある．

これらの手順によって，新しいウェイポイントが求まり，パス$\tilde{P}$が生成される．

$$\tilde{P} \triangleq \left(\tilde{\boldsymbol{s}}, \tilde{\boldsymbol{w}}^{(1)}, \cdots, \tilde{\boldsymbol{w}}^{(m)}, \cdots, \tilde{\boldsymbol{w}}^{(M)}, \tilde{\boldsymbol{g}}\right) \tag{19}$$

2.4 パスの衝突チェック

デコードされたパスを実行する際には，周囲環境や自己との衝突チェックを実施し，衝突が予想される場合には，パスの修正を実施する．本研究では，式(19)を初期軌道としてSTOMP[5]を実行する．デコードによってパスの概形はすでに生成されているため，STOMPによるパスの修正は短時間で実行される．

3．シミュレーション

ここでは，上述の手法のシミュレーションにより各処理の詳細を示す．シミュレーションにはROS（robot operation system）[7]の各種機能を用いた．

Table 2. Algorithm of path decoding.

Given: $\tilde{\boldsymbol{s}}$, $\tilde{\boldsymbol{g}}$, S

Step 1: Calculate for all m as follows:

$$\tilde{\boldsymbol{v}} = \tilde{\boldsymbol{g}} - \tilde{\boldsymbol{s}},$$
$$\tilde{\boldsymbol{e}} = |\tilde{\boldsymbol{v}}|,$$

Step 2: Calculate for all m as follows:

$$\tilde{\boldsymbol{l}}^{(m)} = \tilde{\boldsymbol{s}} + r^{(m)}\tilde{\boldsymbol{v}},$$
$$\tilde{\boldsymbol{w}}^{(m)} = \tilde{\boldsymbol{l}}^{(m)} + \tau^{(m)}\left(\boldsymbol{\delta}^{(m)} \times \tilde{\boldsymbol{e}}\right).$$

Step 3: Output

$$\tilde{\boldsymbol{P}} = \left(\tilde{\boldsymbol{s}}, \tilde{\boldsymbol{w}}^{(1)}, \cdots, \tilde{\boldsymbol{w}}^{(m)}, \cdots, \tilde{\boldsymbol{w}}^{(M)}, \tilde{\boldsymbol{g}}\right).$$

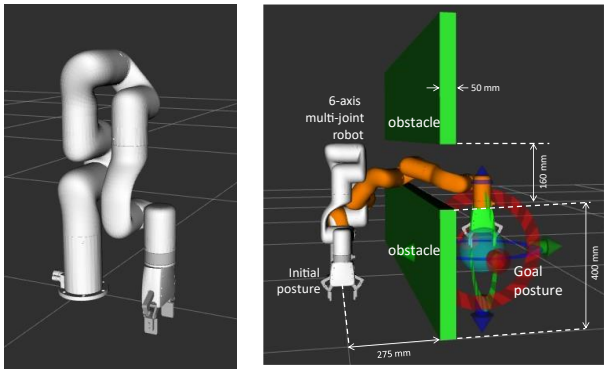
3.1 条件

6軸多関節ロボットとして，xArm6を利用した．目標とする最終姿勢を，手先がリングを潜り抜けた箇所に設定した．それらの概要をFig.2に示す．

まず，RRT-Connectによってパスプランを実施した．探索時間の上限は5 sとした．探索に要した時間の分布についてFig.3に示す．50回の探索で20回はパスを発見することができなかった．経路が発見された30回のうち，最短は0.032 s，最長は2.933 sであり，平均は0.706 sであった．以降のシミュレーションでは，この探索で得られたパスを利用した．

3.2 エンコード

まず，得られたパスについて，第1軸から第3軸，第4軸から第6軸に二分割してパスのエンコードを実施した．エンコードによって



(a) A 6-axis robot (b) obstacles

Fig.2 Overview of a 6-axis multi-joint robot and obstacles in a simulation.

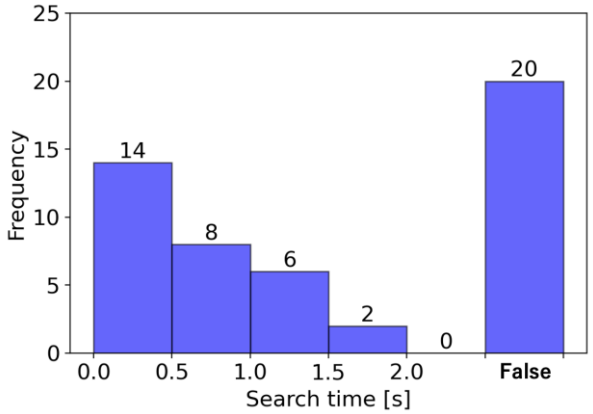


Fig.3 Distribution of the search time taken to find a path by RRT-Connect.

得られた情報についてFig.4に示す．これらの図からわかるように，スタートとゴールを結ぶ線分Lに対して，ウェイポイントの情報が抽出された様子が分かる．また，二つに分割したベクトルによって，外積が成立していることも分かる．

3.3 デコード

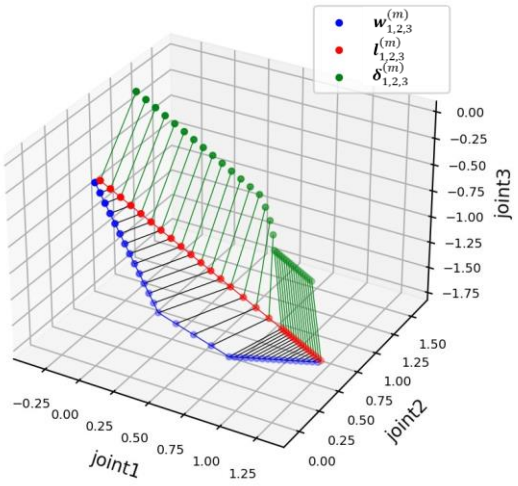
ここではエンコードの手順で利用したパスに対して，提案手法を適用してデコードをした場合の具体例を示す．

まず，エンコードで与えたスタートとゴールと同じものを与えた場合について，Fig.5に示すように，誤差なく同じパスがデコードされることを確認した．

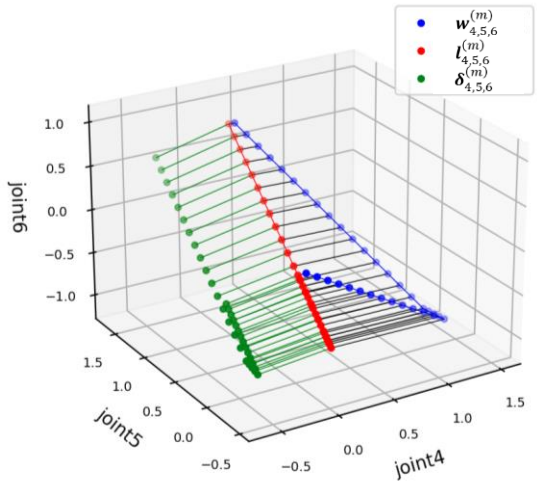
次に，異なるスタートとゴールを与えてデコードしてパスを生成した場合のウェイポイントをFig.6に示す．さらに，デコードによって生成されたパスにおけるエンドエフェクタの軌道をFig.7に示す．これらの結果より，提案手法によって，異なるスタートとゴールを与えた場合も，エンコード前のパスの概形が再現されたパスが適切に生成されたことが確認された．

3.4 計算量と実行時間

100通りの異なるスタートとゴールに対して，同一のパスシードからデコードによるパスの生成に要した時間を計測した結果をFig.8に示す．平均0.00232 sでパスを再現できたことを確認した．

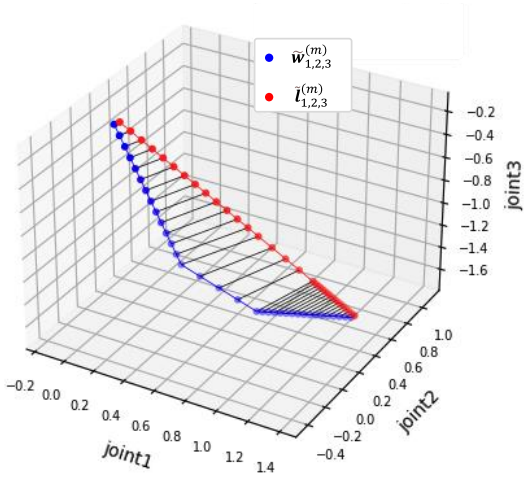


(a) Joint 1 to joint 3.

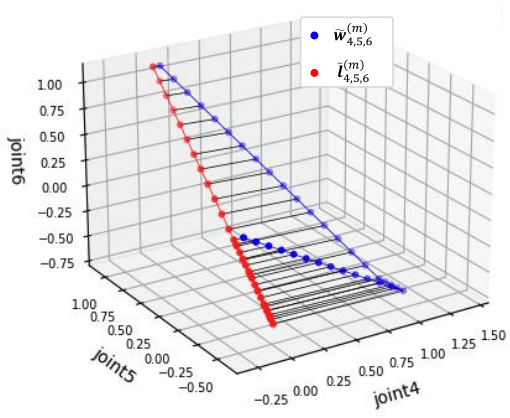


(b) Joint 4 to 6.

Fig.4 Overview of the 6-axis multi-joint robot and obstacles used in the simulation.

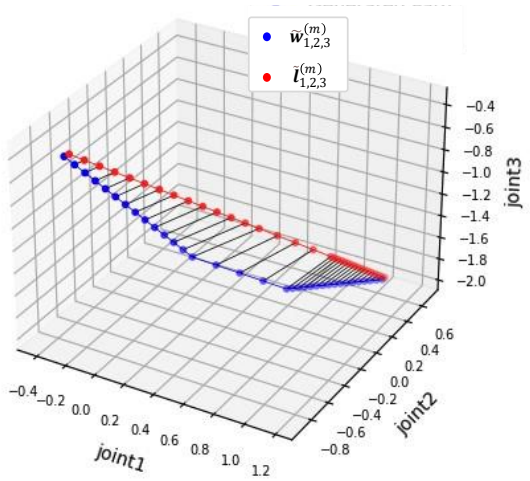


(a) Joint 1 to joint 3.

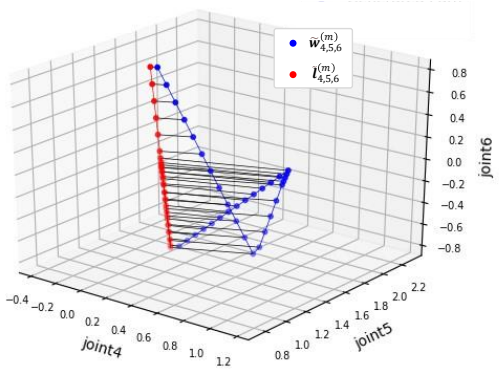


(b) Joint 4 to 6.

Fig.5 Overview of the 6-axis multi-joint robot and obstacles used in the simulation.

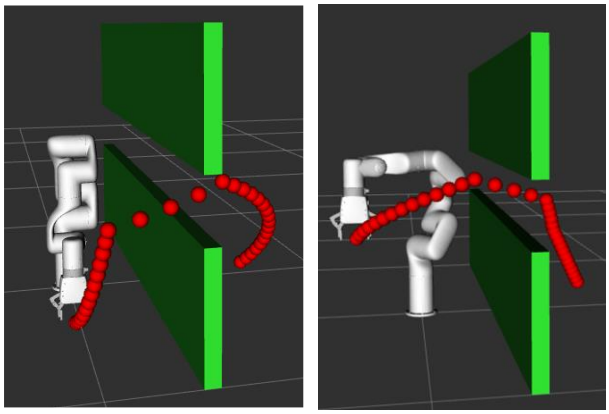


(a) Joint 1 to joint 3.



(b) Joint 4 to 6.

Fig.6 Waypoints for each joint of the decoded path for different start and goal.



(a) Original path. (b) Decoded path.

Fig. 7 The paths generated by the proposed method for the end-effector trajectory; (a) the original path, (b) the paths decoded by the proposed method given different starts and goals.

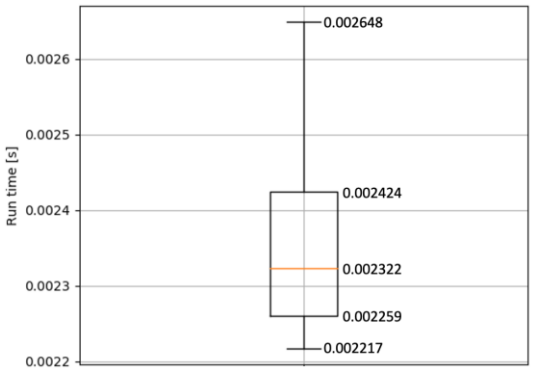


Fig. 8 Box-and-whisker plot of the time required to generate a path from a path seed by decoding using the proposed method for 100 possible start/goal combinations.

4. おわりに

任意の開始・終了位置に対して C 空間のパスの情報をエンコードする手法，およびエンコードされたパスの情報を異なる開始・終了位置に適応させて高速にデコードする手法を，ROS を用いて実装し効果を検証した．その結果，本手法は適切に 6 軸を有する多関節ロボットパスをエンコードおよびデコードすることが確認された．さらに，デコードによるパスの生成時間は従来手法よりも大幅に短くなることを確認した．今後は，ロボットが環境と衝突しないことを確認する処理と提案手法との連動について検討を進める．

参考文献

[1] 西田，松永，多田隈：“中小企業における産業用ロボット導入の障壁とその解決方法の提案”， 計測自動制御学会 *SI* 部門講演会 *SICE-SI* 予稿集(2014)
[2] 西田：“可逆な自動化”， 人工知能，Vol. 37，No. 3， pp. 286-291(2022)
[3] R. Kabutan, T. Nishida, “Motion Planning by T-RRT with Potential Function for Vertical Articulated Robots,” Electrical Engineering in Japan, DOI10.1002/eej.23103, 2018.
[4] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, S. Schaal, “STOMP: Stochastic trajectory optimization for motion planning,” IEEE Int. Conf. on Robotics and Automation, pp. 4569-4574, 2011.
[5] T. Barbis, R. Kabutan, R. Tanaka, T. Nishida, “Gaussian Mixture Spline Trajectory: Learning from a Dataset, Generating Trajectories Without One,” Advanced Robotics, Vol. 32, Issue 10, pp. 547-558, 2018.
[6] 西田，“多関節ロボットのパスのエンコードとデコード，” SI2023 予稿集 (2023)
[7] 西田，森田，岡田，原，山崎，垣内，田向，齋藤，大川，田中，有田，石田，「実用ロボット開発のための ROS プログラミング」， 森北出版，2018.