Clarke Transform and Encoder-Decoder Architecture for Arbitrary Joints Locations in Displacement-Actuated Continuum Robots

Reinhard M. Grassmann and Jessica Burgner-Kahrs

***Abstract*— In this paper, we consider an arbitrary number of joints and their arbitrary joint locations along the center line of a displacement-actuated continuum robot. To achieve this, we revisit the derivation of the Clarke transform leading to a formulation capable of considering arbitrary joint locations. The proposed modified Clarke transform opens new opportunities in mechanical design and algorithmic approaches beyond the current limiting dependency on symmetric arranged joint locations. By presenting an encoder-decoder architecture based on the Clarke transform, joint values between different robot designs can be transformed enabling the use of an analogous robot design and direct knowledge transfer. To demonstrate its versatility, applications of control and trajectory generation in simulation are presented, which can be easily integrated into an existing framework designed, for instance, for three symmetric arranged joints.**

I. Introduction

Real-world tasks in medical and industrial applications do not impose symmetric manipulator designs for soft and continuum robots. However, almost all displacement-actuated continuum robots rely on a very narrow design space, *i.e.*, three or four symmetric arranged joints. Overcoming that design restriction has obvious benefits. In particular, increasing the joint number n significantly, *i.e.*, $n \gg 3$, and considering asymmetric joint arrangements: (i) enhances manipulability along the directions that coincide with joint locations and center-line; (ii) improves force absorption and delivery due to the distributed nature of the actuation forces; and (iii) increases safety through actuation redundancy. Desirable in medical applications [1], [2] and industrial settings [3], [4], the utilization of such mechanical design may lead to increased load capacity, better shape conformation, enhanced stability, and variable stiffness. Therefore, exploring and providing a computational inexpensive general approach is a pioneering step towards more capable displacement-actuated continuum robots that include a large set of continuum and soft robots.

To this day, displacement-actuated continuum robots with n joints are under-explored. Some attempts, *e.g.*, [5], [6] for n joints provide the solution for the robot-independent mapping, and the work by Dalvand *et al.* [7] considers n joints in their forward kinematics framework. Their framework includes an exhaustive list of 2^n combinations of passive and

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [RGPIN-2019-04846].

All authors are with Continuum Robotics Laboratory, Department of Mathematical and Computational Sciences, University of Toronto, Mississauga, ON L5L 1C6, Canada
reinhard.grassmann@utoronto.ca

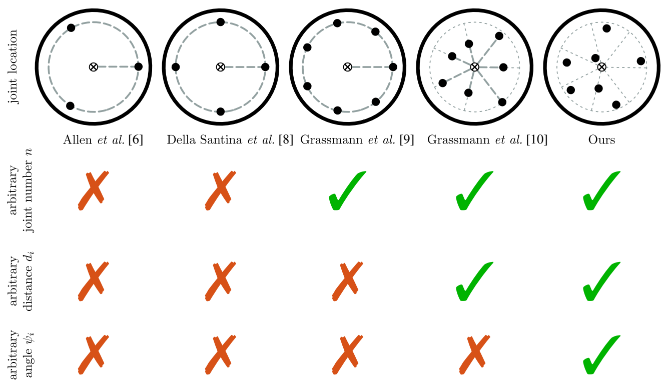


Fig. 1. Joint location and improved joint representation. The kinematics of a displacement-actuated continuum robot with fixed segment length is mainly influenced by the number of joints n and their location in the cross-section in terms of polar coordinates. For the i^{th} joint, the polar coordinates is described by the distance d_i to the center-line and the angle ψ_i. Joint representations, *e.g.*, [6], [8], [9], [10], have been proposed to consider various arrangement. Our approach generalizes and covers all cases.

active tendons as joints and several branches, *e.g.*, if-else-statements, as well as function calls to numerically solve an underlying beam model. However, a computed result might not be consistent, where a consistency between $96.1\,\%$ and $99.2\,\%$ is reported. While this approach [7] can be considered to account for arbitrary joint locations, it relies heavily on computing and heuristics.

In a recent work by Grassmann *et al.* [9], they show that the Clarke transform using a generalized Clarke transformation matrix can be used to map n joint values onto a two-dimensional manifold. Kinematics based on the Clarke transform are branchless, closed-form, and singularity-free. However, this approach [9] only applies to symmetric arranged joint location, see Fig. 1. The Clarke transform can also account for non-constant distances [10]. However, to the best of our knowledge, no approach can consider arbitrary joint location while only considering the kinematic design parameters of the target displacement-actuated continuum robot. The kinematic design parameters entail angular offset ψ_i, distance d_i to center-line in the cross-section, and length l of a displacement-actuated continuum robot as illustrated in Fig. 2.

Due to the Clarke transform's relationship to the Clarke transformation matrix, looking into generalized Clarke transformation matrices for n phases in the literature for electrical motors is worthwhile. Furthermore, based on the analogy to Kirchhoff's current law [9], the actuation constraint, *i.e.*, $\sum_{i=1}^{n} \rho_i = 0$, where ρ_i are the displacement values, inherent to a displacement-actuated continuum robot with symmetric arrangement can be considered as balanced system. Consequently, a displacement-actuated continuum robot with asymmetric arrangement can be considered an unbalanced system as the constraint is not necessarily zero akin to an unbalanced electrical system, where the sum of all electrical current in each phase is non-zero.

Up to a certain negative sign, a matrix similar to [10], [9] is derived by Janaszek [11]. In work by Willems [12] and by Rockhill & Lipo [13], the squared matrix is only useful for balanced systems. For symmetric phase arrangements, all variants should produce $(n-2)$ zeros resulting in unnecessary computation and large state representation. Another squared matrix is derived by Willems [12], which can consider unbalanced systems. However, for all representations derived by Willems [12], it is not guaranteed that its inverse produces always $(n-2)$ zeros. Therefore, the dimensionality cannot be reduced to two variables. To summarize, none of the generalized Clarke transformation matrices can be used directly.

In the present work, we propose a modified generalized Clarke transformation matrix to construct a Clarke transform that applies to arbitrary joint locations for displacement-actuated continuum robots. Furthermore, we propose an encoder-decoder architecture based on the Clarke transform to facilitate the use of those continuum robots. In particular, the contribution of this paper includes:

- Deriving a generalized Clarke transformation matrix for unbalanced systems
- Modifying the Clarke transform to consider arbitrary joint locations
- Proposing an encoder-decoder architecture to transform joint values between different robot designs
- Adapting a $\mathcal{C}^4$-smooth trajectory generator
- Generating feasible joint values

Due to the use of Clarke coordinates, this approach automatically contributes to providing robot-dependent mapping. Furthermore, all derived approaches are linear, compact, and closed-form.

II. Extending Clarke Transform

In this section, we briefly revisit the Clarke transform and state the relationship to arc parameters. Afterward, an alternative derivation is stated that does not rely on the connection to a Clarke transformation matrix. Based on the alternative derivation, we derive a Clarke transform and its inverse to consider asymmetric arranged joint locations.

A. Clarke Transform for Displacement-Actuated Joint

Using the representation proposed in [9] for i^{th} entry of displacement-actuated joints $\boldsymbol{\rho}$ given by ρ_i, the index notation for $\boldsymbol{\rho}$ is

$$\boldsymbol{\rho} = \left(\rho_{\text{Re}} \cos\psi_i + \rho_{\text{Im}} \sin\psi_i\right)_i \subset \mathbb{R}^n, \tag{1}$$

whereas Clarke coordinates, *i.e.*, ρ_{Re} and ρ_{Im}, being the free parameter of (1), can be combined into

$$\overline{\boldsymbol{\rho}} = \left[\rho_{\text{Re}}, \rho_{\text{Im}}\right]^\top \in \mathbb{R}^2. \tag{2}$$

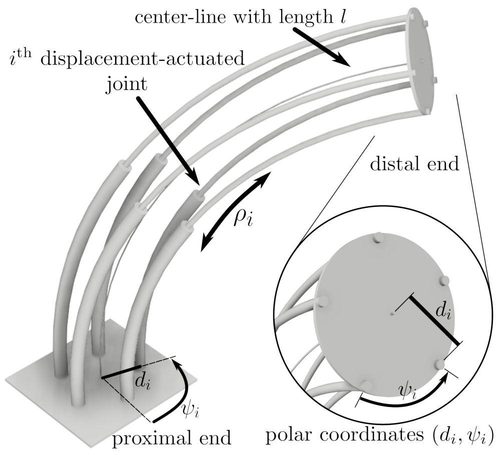

center-line with length l
i^{th} displacement-actuated joint
distal end
ρ_i
d_i
d_i
ψ_i
ψ_i
proximal end
polar coordinates (d_i, ψ_i)

Fig. 2. Kinematic design parameters of a displacement-actuated continuum robot.

Note that (1) is not an element of $\mathbb{R}^n$ since all joints are interdependent and constrained.

Both representations can be transformed into each other using

$$\overline{\boldsymbol{\rho}} = \boldsymbol{M}_{\mathcal{P}} \boldsymbol{\rho} \quad \text{and} \tag{3}$$

$$\boldsymbol{\rho} = \boldsymbol{M}_{\mathcal{P}}^{-1} \overline{\boldsymbol{\rho}}, \tag{4}$$

where $\boldsymbol{M}_{\mathcal{P}}^{-1}$ is the right-inverse of $\boldsymbol{M}_{\mathcal{P}}$ and $\boldsymbol{M}_{\mathcal{P}}$ is given by the generalized Clarke transformation matrix. For a symmetric arranged joint location, we kindly refer to [9] for a derivation and specific realization of both matrices.

B. Relation to Arc Parameters

As pointed out by Grassmann & Burgner-Kahrs [10], it is possible to normalize the Clarke coordinates with respect to kinematic design parameters, *i.e.*, distance d_i and segment length l, which leads to the curvature-curvature representation if constant curvature is assumed, *i.e.*,

$$\begin{bmatrix} \kappa\cos(\theta) \\ \kappa\sin(\theta) \end{bmatrix} = \underbrace{1/l}_{\text{removes } l} \overbrace{\boldsymbol{M}_{\mathcal{P}}}^{\text{removes } \psi_i} \underbrace{\operatorname{diag}(1/d_i)}_{\text{removes } d_i} \boldsymbol{\rho}. \tag{5}$$

For its inverse, the parameters l, d_i, and ψ_i are added, *i.e.*,

$$\boldsymbol{\rho} = \underbrace{l}_{\text{adds } l} \overbrace{\operatorname{diag}(d_i)}^{\text{adds } d_i} \underbrace{\boldsymbol{M}_{\mathcal{P}}^{-1}}_{\text{adds } \psi_i} \begin{bmatrix} \kappa\cos(\theta) \\ \kappa\sin(\theta) \end{bmatrix}. \tag{6}$$

For both formulations, the assumption $d_i = d$ has been removed. The kinematic design parameters fully describe the displacement-actuated continuum robot in the kinematic sense, see Fig. 2

C. Alternative Derivation of $\boldsymbol{M}_{\mathcal{P}}$

Grassmann *et al.* in [9] generalize the Clarke transformation matrix motivated by an analogy between tendon-driven continuum robot and the control of brushless motors. In fact, the more important part is joint representation (1). To show this, we present an alternative approach to derive $\boldsymbol{M}_{\mathcal{P}}$.

First, the Clarke coordinates in (1) are factored out, *i.e.*,

$$\begin{bmatrix}\rho_1\\ \rho_2\\ \vdots\\ \rho_n\end{bmatrix} = \underbrace{\begin{bmatrix}\cos(\psi_1) & \sin(\psi_1)\\ \cos(\psi_2) & \sin(\psi_2)\\ \vdots & \vdots\\ \cos(\psi_n) & \sin(\psi_n)\end{bmatrix}}_{\boldsymbol{M}_{\mathcal{P}}^{-1}} \begin{bmatrix}\rho_{\mathrm{Re}}\\ \rho_{\mathrm{Im}}\end{bmatrix}, \tag{7}$$

which resembles (4). As a result, a generalized inverse Clarke transformation matrix denoted by $\boldsymbol{M}_{\mathcal{P}}^{-1}$ can be identified. For the sake of clarification, the notation used for $\boldsymbol{M}_{\mathcal{P}}^{-1}$ does not indicate a matrix inverse of $\boldsymbol{M}_{\mathcal{P}}$.

Second, constructing the Moore-Penrose pseudoinverse for solving undetermined linear systems leads to (3), where

$$\boldsymbol{M}_{\mathcal{P}} = \left(\left(\boldsymbol{M}_{\mathcal{P}}^{-1}\right)^{\top} \boldsymbol{M}_{\mathcal{P}}^{-1}\right)^{-1} \left(\boldsymbol{M}_{\mathcal{P}}^{-1}\right)^{\top}. \tag{8}$$

Note that (8) leads to an exact solution and it is not an approximation.

Finally, we assume that the angle ψ_i represented a symmetric arrangement of the joint locations, *i.e.*, $\psi_i = 2\pi(i-1)/n$. For this specific case, we can write

$$\begin{aligned}\overline{\boldsymbol{\rho}} &= \left(\left(\boldsymbol{M}_{\mathcal{P}}^{-1}\right)^{\top} \boldsymbol{M}_{\mathcal{P}}^{-1}\right)^{-1} \left(\boldsymbol{M}_{\mathcal{P}}^{-1}\right)^{\top} \boldsymbol{\rho}\\ &= \left(\begin{bmatrix}n/2 & 0\\ 0 & n/2\end{bmatrix}\right)^{-1} \left(\boldsymbol{M}_{\mathcal{P}}^{-1}\right)^{\top} \boldsymbol{\rho}\\ &= \frac{2}{n}\left(\boldsymbol{M}_{\mathcal{P}}^{-1}\right)^{\top} \boldsymbol{\rho},\end{aligned}$$

where the trigonometric identity, *i.e.*, $\sum_{i=1}^{n} \sin^2(\psi_i) = n/2$, $\sum_{i=1}^{n} \cos^2(\psi_i) = n/2$, and $\sum_{i=1}^{n} \sin(\psi_i)\cos(\psi_i) = 0$, derived and stated in [10], [9] are used to rewrite $\left(\boldsymbol{M}_{\mathcal{P}}^{-1}\right)^{\top} \boldsymbol{M}_{\mathcal{P}}^{-1}$. Per definition, $\boldsymbol{M}_{\mathcal{P}}$ is set to $2/n\left(\boldsymbol{M}_{\mathcal{P}}^{-1}\right)^{\top}$, which aligns with the property stated in [9].

D. Asymmetric arranged joint locations

In order to consider asymmetric designs such as illustrated in Fig. 3, the Clarke transform [9] can be modified. The pseudoinverse (8) and inverse robot-dependent mapping (6) point towards the possibility to relax assumption $\psi_i = 2\pi(i-1)/n$ and $d_i = d$, respectively.

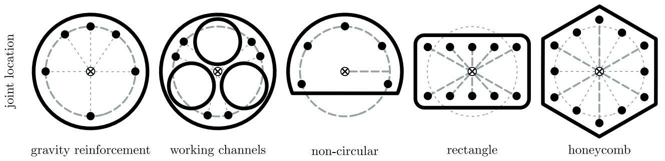


Fig. 3. Possible designs of arbitrary asymmetric joint location.

A modification leads to

$$\widehat{\boldsymbol{M}}_{\mathcal{P}}^{-1} = \frac{1}{f(d_i)} \operatorname{diag}(d_i)\, \boldsymbol{M}_{\mathcal{P}}^{-1},$$

where $\widehat{\boldsymbol{M}}_{\mathcal{P}}$ can be found using the Moore-Penrose pseudoinverse considering the whole right-hand side of expression akin to (8). The diagonal matrix $\operatorname{diag}(d_i) \in \mathbb{R}^n$ is used to rescale $\overline{\boldsymbol{\rho}}$, *cf.* (5) and (6). To account for the normalization and change of unit due to $\operatorname{diag}(d_i)$, the function $f(d_i)$ is introduced and outputs a scalar for a given list or vector of all d_i. The unit of the scalar is a unit of length. However, many different choices can be justified that are dependent on a specific application. To overcome the choice of a suitable function $f(d_i)$, we propose an encoder-decoder architecture. For the sake of clarification, $1/f(d_i) \neq l$.

III. Encoder-Decoder Architecture

The Clarke transform is applicable to a wide variety of different robot morphologies. Furthermore, the Clarke coordinates are generalized improved state representations [9]. With that in mind, the joint values $\boldsymbol{\rho}_{(\text{robot A})}$ of *robot A* with specific number of joints can be mapped to the same Clarke coordinates $\overline{\boldsymbol{\rho}}$, that correspond to the joint values $\boldsymbol{\rho}_{(\text{robot B})}$ of *robot B* with a different number of joints. This can be expressed by

$$\overline{\boldsymbol{\rho}} = \boldsymbol{M}_{\mathcal{P}}^{-1}{}_{(\text{robot A})} \boldsymbol{\rho}_{(\text{robot A})} = \boldsymbol{M}_{\mathcal{P}}^{-1}{}_{(\text{robot B})} \boldsymbol{\rho}_{(\text{robot B})}$$

Rearranging the above equation leads to an *encoder-decoder architecture* illustrated in Fig. 4 and given by

$$\boldsymbol{\rho}_{(\text{robot A})} = \underbrace{\boldsymbol{M}_{\mathcal{P}(\text{robot A})}}_{\text{decoder}} \overbrace{\boldsymbol{M}_{\mathcal{P}}^{-1}{}_{(\text{robot B})}}^{\text{encoder}} \boldsymbol{\rho}_{(\text{robot B})}, \tag{9}$$

which assumed symmetric arranged joint locations. It should be highlighted that $\boldsymbol{M}_{\mathcal{P}}^{-1}{}_{(\text{robot A})}$ is the right-inverse of $\boldsymbol{M}_{\mathcal{P}(\text{robot A})}$. Therefore, if *robot A* and *robot B* have the same kinematic structure, the inputs are mathematically the same, *i.e.*, $\boldsymbol{\rho}_{(\text{robot A})} = \boldsymbol{\rho}_{(\text{robot B})}$.

To relax the assumption, (5) and (6) can be used to remove and add the kinematic design parameters, respectively. In this case, (5) is the encoder, whereas (6) is the decoder. This results in

$$\begin{aligned}\boldsymbol{\rho}_{(\text{robot A})} = {} & \overbrace{l_{(\text{A})} \operatorname{diag}\left(d_{i,(\text{A})}\right) \boldsymbol{M}_{\mathcal{P}}^{-1}{}_{(\text{A})}}^{\text{adds kinematic design parameters of robot A}}\\ & \cdot \underbrace{\frac{1}{l_{(\text{B})}} \boldsymbol{M}_{\mathcal{P}(\text{B})} \operatorname{diag}\left(\frac{1}{d_{i,(\text{B})}}\right)}_{\text{removes kinematics design parameters of robot B}} \boldsymbol{\rho}_{(\text{robot B})},\end{aligned} \tag{10}$$

where, for brevity, the subscript *A* and *B* is short for *robot A* and *robot B*, respectively. The generalized Clarke transformation matrix $\boldsymbol{M}_{\mathcal{P}}$ in (10) is defined by (8).

IV. Demonstration in Simulation

We evaluate our proposed methods on a control problem in simulation. For this, we state a two-staged method to generate feasible joint values, generate trajectories, and synthesize a PD controller scheme. Figure 5 illustrates the workflow.

For the evaluation, we consider five different displacement-actuated continuum robots with one type-0 segment, *i.e.*, each continuum robot has a single segment with a fixed length. The segment length is $l = 0.1\,\mathrm{m}$ for all continuum robots, whereas the distance d_i and angle ψ_i are different. Furthermore, the number of joints n for each target continuum robot is different. All kinematic design parameters are listed in Table I.

TABLE I

KINEMATIC DESIGN PARAMETERS OF EACH CONSIDERED DISPLACEMENT-ACTUATED CONTINUUM ROBOT.

robot	n	ψ_i in rad	asymmetric ψ_i	d_i in mm	non-constant d_i	l in m
robot_0	3	$2\pi\,[0,\ 1/3,\ 2/3]$	✗	$[10,\ 10,\ 10]$	✗	0.1
robot_A	4	$2\pi\,[0,\ 0.25,\ 0.5,\ 0.75]$	✗	$[10,\ 10,\ 10,\ 10]$	✗	0.1
robot_B	3	$2\pi\,[0,\ 1/3,\ 2/3]$	✗	$[10,\ 7,\ 5]$	✓	0.1
robot_C	5	$2\pi\,[0,\ 0.2,\ 0.4,\ 0.6,\ 0.8]$	✗	$[10,\ 8.7,\ 5,\ 9.5,\ 6.5]$	✓	0.1
robot_D	7	$2\pi\,[0.05,\ 0.18,\ 0.51,\ 0.63,\ 0.76,\ 0.87,\ 0.91]$	✓	$[10,\ 1,\ 8.7,\ 5,\ 5.6,\ 9.5,\ 6.5]$	✓	0.1

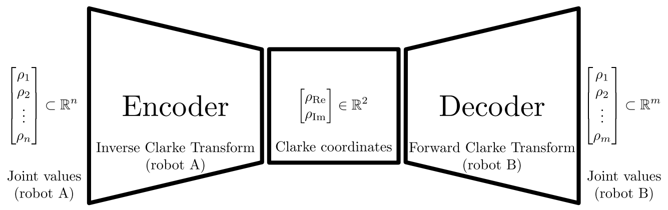

$\begin{bmatrix}\rho_1\\ \rho_2\\ \vdots\\ \rho_n\end{bmatrix} \subset \mathbb{R}^n$
Joint values (robot A)
Encoder
Inverse Clarke Transform (robot A)
$\begin{bmatrix}\rho_{\mathrm{Re}}\\ \rho_{\mathrm{Im}}\end{bmatrix} \in \mathbb{R}^2$
Clarke coordinates
Decoder
Forward Clarke Transform (robot B)
$\begin{bmatrix}\rho_1\\ \rho_2\\ \vdots\\ \rho_m\end{bmatrix} \subset \mathbb{R}^m$
Joint values (robot B)

Fig. 4. Encoder-decoder architecture. Joint values of one robot type (robot A) with n-dimensional joint space can be transformed into joint values of a different robot type (robot B) with m-dimensional joint space. The latent space representation is encoded as Clarke coordinates. It is worth noticing that the compression is a lossless compression that allows joint values to be uniquely reconstructed from the Clarke coordinates.

A. Feasible Joint Values

One of the used robot designs is a surrogate robot denoted by *robot_0*. To sample $m+1$ set of $n_{(\text{robot\_0})}$ feasible joint values, we use a two-stage method.

First, sample Clarke coordinates utilizing a direct sampling method [9]. This rejection-free sampling method is vectorized. Using the index notation, it can be expressed by

$$\begin{pmatrix}\rho^{(i)}_{\mathrm{Re},\mathcal{U}}\\ \rho^{(i)}_{\mathrm{Im},\mathcal{U}}\end{pmatrix}_i = \begin{pmatrix}L^{(i)}_{\mathcal{U}}\cos\left(\theta^{(i)}_{\mathcal{U}}\right)\\ L^{(i)}_{\mathcal{U}}\sin\left(\theta^{(i)}_{\mathcal{U}}\right)\end{pmatrix}_i \in \mathbb{R}^{2\times m+1},$$

where all magnitude $L^{(i)}_{\mathcal{U}}$ and all angle $\theta^{(i)}_{\mathcal{U}}$ are sampled via

$$L_{\mathcal{U}} = \pi d_{(\text{robot\_0})}\sqrt{\mathcal{U}^{m+1}\,[0;1]} \quad \text{and}$$
$$\theta_{\mathcal{U}} = \pi\,\mathcal{U}^{m+1}\,[-1;1)\,,$$

respectively. The distance $d_{(\text{robot\_0})}$ is $0.01\,\mathrm{m}$ as stated in Table I. The expression $\sqrt{\mathcal{U}^{m+1}\,[0;1]}$ samples $m+1$ values from an uniform distribution $\mathcal{U}$ with the interval $[0;1]$ and, afterwards, the square root is computed. This leads to a uniformly distributed disk [14]. Note that, assuming constant curvature assumption, $\max L_{\mathcal{U}} = \pi d_{(\text{robot\_0})}$ corresponds to a half-circle. This formulation allows specifying the maximal value without defining the maximal curvature and using segment length l.

Second, transform all sampled Clarke coordinates using (4), which can be vectorized as well. This step is illustrated as the decoder in Fig. 5.

B. Trajectory Generation

To provide smooth trajectories for n displacements, we adapted a $\mathcal{C}^4$-smooth trajectory generator [15] for via poses capable of respecting kinematic limits, *i.e.*, considering the maximum velocity and maximum acceleration. Here, we use the fact that $\boldsymbol{\rho} \subset \mathbb{R}^n$ can be treated as $n\,$dof position in Euclidean space.

TABLE II

TRAJECTORY STATE REPRESENTATION.

Variable	Description of the variable in $\mathcal{T}^j_i$
$\Delta\rho^j_i$	Distance to overcome. It is define as $\Delta\rho^j_i = \rho^j_i - \rho^{j-1}_i$
ρ^{j-1}_i	Start point of the displacement
ρ^j_i	Goal point of the displacement
v^j_i	Maximum velocity of the velocity profile
a^j_i	Maximum acceleration of the lift-off phase
d^j_i	Maximum deacceleartion of the set-down phase
$t^j_{\mathrm{lo},i}$	Duration for the lift-off phase
$t^j_{\mathrm{cr},i}$	Duration for the curse phase
$t^j_{\mathrm{sd},i}$	Duration for the set-down phase
$t^j_{\mathrm{enb},i}$	Switching time at which the j^{th} trajectory starts and the blending into the $(j+1)^{\mathrm{th}}$ trajectory is happening

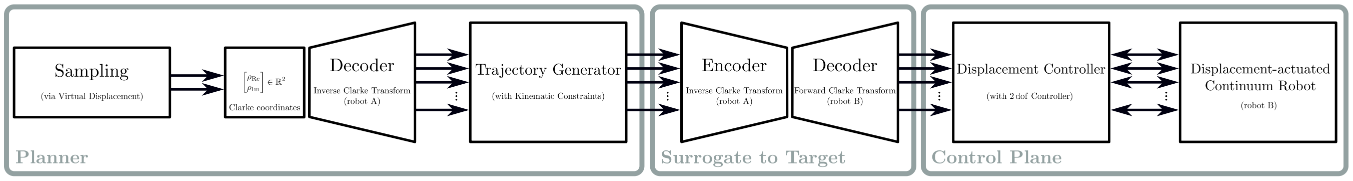
For $m-1$ intermediate points, one start point, and a goal point, we define $m\times n$ trajectories for n displacements. Each trajectory is composed of three phases for the trapezoidal-like velocity profile. They are denoted by *lo*, *cr*, and *sd* for *lift-off*, *cruise*, and *set-down* phase, respectively. The $m\times n$ trajectories are blended into n trajectories for each displacement ρ_i. We kindly refer to [15] for details on the blending.

In contrast to [15], the used j^{th} trajectory state $\mathcal{T}^j_i$ of the m trajectories of the i^{th} displacement is defined as

$$\mathcal{T}^j_i = \left(\Delta\rho^j_i,\ v^j_i,\ a^j_i,\ d^j_i,\ t^j_{\mathrm{lo},i},\ t^j_{\mathrm{cr},\ i},\ t^j_{\mathrm{sd},i},\ t^j_{\mathrm{enb},i}\right), \quad (11)$$

where the description of each variable is listed in Table II. An advantage of (11) is the tracking of each duration, which simplifies the synchronization between the n trajectories, reduces re-computation of quantities, and avoids checking of zero divisions. Related to the time-memory trade-off in computer science, this comes at the expense of a slightly larger trajectory state formulation, *i.e.*, eight variables in (11) instead of five variable, *cf.* [15].

For the evaluation, the kinematic constraints are set to $v^j_i = 0.01\pi\,\mathrm{m/s}$, $a^j_i = 0.125\pi\,\mathrm{m/s^2}$, and $d^j_i = 0.125\pi\,\mathrm{m/s^2}$ for all i and j. The generated paths consist of $m = 5$ blended velocity profiles. Figure 6 shows the resulted velocity profiles for the surrogate robot.



Sampling
(via Virtual Displacement)
Clarke coordinates
Decoder
Inverse Clarke Transform (robot A)
Trajectory Generator
(with Kinematic Constraints)
Planner
Encoder
Inverse Clarke Transform (robot A)
Decoder
Forward Clarke Transform (robot B)
Surrogate to Target
Displacement Controller
(with 2dof Controller)
Displacement-actuated Continuum Robot
(robot B)
Control Plane

Fig. 5. Workflow of the evaluation.

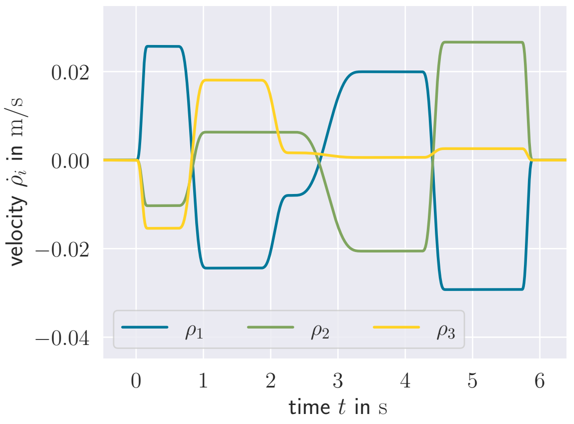


Fig. 6. Velocity profile of the surrogate robot with $n = n_{(\text{robot\_0})} = 3$. The trapezoidal-like velocity profiles and blending of the velocity profiles of each displacement are $\mathcal{C}^4$-smooth.

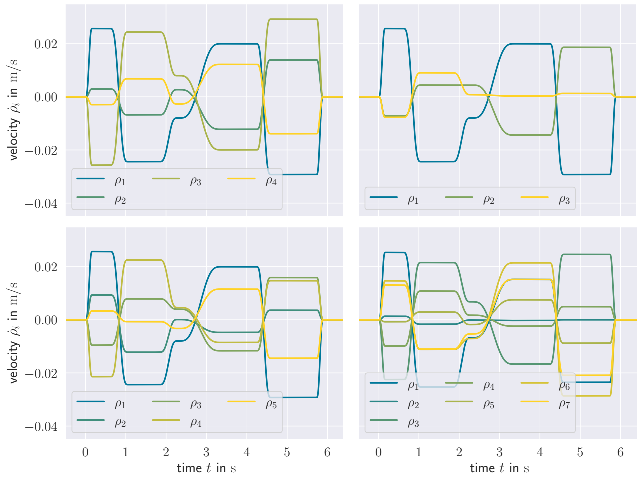


Fig. 8. Transformed velocity profiles as the output of a used encoder-decoder architecture. The number of displacements unambiguously identifies the corresponding target robot listed in Table I. As can be seen, the kinematic constraint on the velocity, *i.e.*, $v_i^j = 0.01\pi\,\text{m/s} \approx 0.03\,\text{m/s}$, is fulfilled.

C. Control of Joints

The setup is similar to [9], where the control frequency is $1\,\text{kHz}$, each actuator is modelled as an independent first-order proportional delay element (PT_1) system with a time constant equal to $250\,\text{ms}$, and the additive measurement noise is drawn from a uniform distribution $\mathcal{U}^n[-\epsilon, \epsilon]$ with $\epsilon = 2.5\,\text{mm}$ every $1\,\text{ms}$. Here, all PD control gains are $\boldsymbol{K}_\text{p} = 75$ and $\boldsymbol{K}_\text{d} = 0.0015$. The controller scheme is illustrated in Fig. 7.

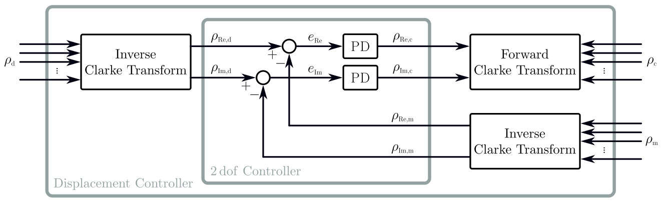


Fig. 7. Control of Clarke Coordinates. Using the Clarke transform, only two simple PD controllers are necessary to control n displacement-actuated joints.

The desired displacements $\boldsymbol{\rho}_\text{d}$ of the target robot for each time step are provided by the encoder-decoder architecture, see Fig. 5. The encoder-decoder architecture illustrated in Fig. 4 transforms the trajectory generated for surrogate robot *robot_0* to the respective target robot, *e.g.*, *robot_A* or *robot_D*. Therefore, the encoder part is constant through the evaluation, whereas the decoder part depends on the target robot listed in Table I. The transformed velocity profiles for each target robot are shown in Fig. 8.

D. Results

Figure 9 shows the open-loop behavior without noise, open-loop behavior with added noise, and the closed-loop behavior for all five displacement-actuated continuum robots listed in Table I. The controller output follows the desired transformed displacement of the respective target robot. While the results in Fig. 9 utilize the more general definition of the encoder-decoder architecture (10), the results in Fig. 10 are achieved using (9).

V. Discussion and Future Work

The alternative derivation highlights the importance of the displacement representation (1) over the relation to the standard Clarke transformation matrix $\boldsymbol{M}_\text{Clarke} \in \mathbb{R}^{3\times3}$ commonly used in the literature. While no analogy to electrical engineering is used for the alternative derivation, possible synergy effects could be overlooked, and the relation between $\boldsymbol{M}_\text{Clarke}$ and $\boldsymbol{M}_\mathcal{P}$ is left in the dark. Possible synergy effects are discussed in [10], [9].

Generating joint values via rejection sampling has been shown in [9] to be inefficient even for the most simple symmetric case with $n = 3$. To overcome the inefficiency, we combine two strategies; exploiting the geometric meaning of Clarke coordinates described in [9] and utilizing a simple surrogate displacement-actuated continuum robot. Using an encoder-decoder architecture, feasible joint values for a target robot can be sampled. This sampling method is branch-less, vectorizable, and rejection-free, *i.e.*, all samples are $100\,\%$ correct.

Transforming generated trajectories for a simple surrogate robot like *robot_0* using the encoder-decoder architecture is computationally cheaper than recomputing trajectories for

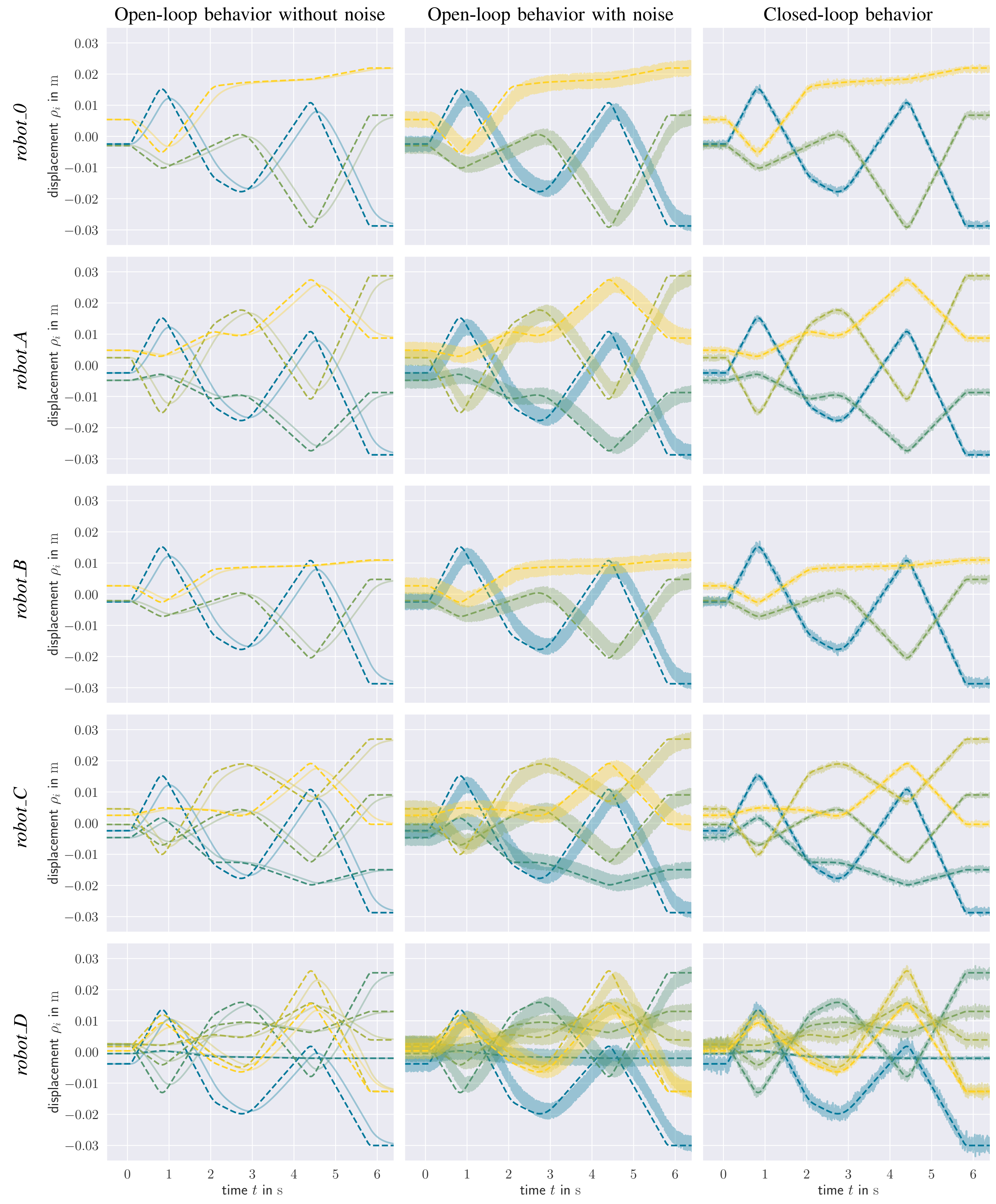


Fig. 9. Displacement-control. (left column) Desired path versus open-loop behavior of the noise-free PT_1 system. (middle column) Desired path versus measured displacement with noise. Due to the high frequency, the measurement noise appears to be a band. (right column) Desired path versus closed-loop behavior. (first row) surrogate robot *robot_0* with $n = 3$ symmetric joint location. (second row) target robot *robot_A* with $n = 4$ with symmetric joint location. (third row) target robot *robot_B* with $n = 3$ with non-constant distant d_i. (fourth row) target robot *robot_C* with $n = 5$ with non-constant distant d_i. (last row) target robot *robot_D* with $n = 7$ with non-constant distant d_i and asymmetric ψ_i

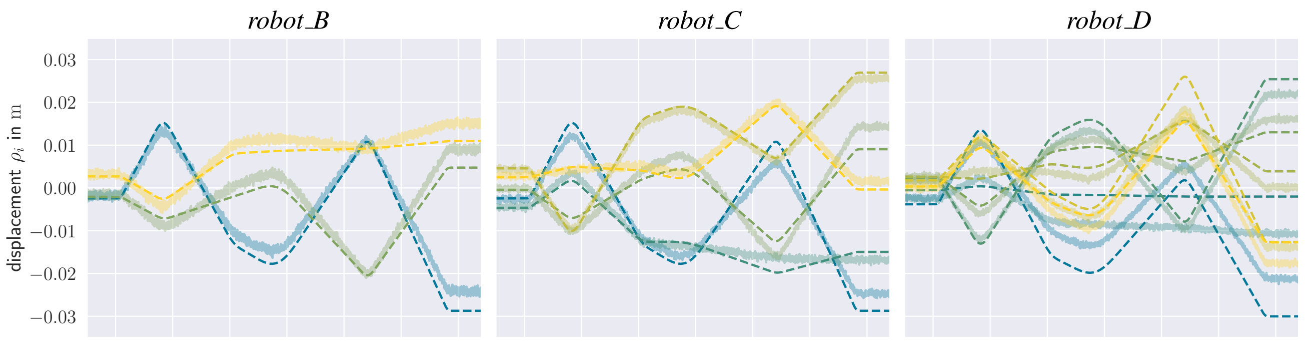


Fig. 10. Displacement-control without considering the non-constant distance d_i. The control outputs show that without using the appropriate encoder-decoder architecture, *i.e.*, (9) instead of (10), the desired path cannot be tracked accurately. Increasing the proportional gain or including an integrator might not be ideal, especially if this can be accomplished with (10) as shown in Fig. 9.

the target robot. The encoder-decoder architecture simplifies to a vector-matrix multiplication at each time step. In contrast, the trajectory generator [15] has five stages including the computation of octic polynomial functions, branching between three phases of the trapezoidal-like velocity profile, blending, and rescaling. To improve trajectory generation, generating the desired trajectories directly on the two-dimensional manifold using the Clarke coordinates is desirable, where the kinematic constraints must be expressed in relation to the Clarke coordinates.

Furthermore, since superposition, *i.e.*, scaling and addition, will not change the smoothness of the resulted trajectory, the $\mathcal{C}^4$-smooth trajectory for the surrogate robot remains a $\mathcal{C}^4$-smooth trajectory for the target robot. However, different robot types have different requirements for the smoothness of the trajectory. For example, for rigid serial-kinematic robots, $\mathcal{C}^3$ smoothness is required [16], whereas, for robots with flexible elements such as the Panda robot [17], a $\mathcal{C}^4$-smooth trajectory is mandatory [18]. The requirements for continuum and soft robots are unknown yet and more work in this direction is needed.

The controller scheme illustrated in Fig. 7 utilizes the forward and inverse Clarke transforms. For constant distances d_i and constant length l, the controller gains are scaled resulting in different steady-state errors and noise sensitivity. Constant factors in the Clarke transform can be taken into account in the controller gains. That is why only one value for length l in Table I is considered in the evaluation. Furthermore, due to $\mathrm{diag}\,(1/d_i)$, the sensitivity to noise is higher for *robot_B*, *robot_C*, and *robot_D* as shown in Fig. 9. Tuning the controller gains will meditate this. However, for uncompensated non-constant distance d_i, the steady-state error is significantly bigger and the PD controller scheme cannot accurately follow the trajectory as shown in Fig. 10, *cf.*, Fig. 9. To account for non-constant distance d_i, we utilize (10) instead of (9). For future work, a model-based PD controller based on the kinematic design parameters is desirable to circumvent tuning of the controller gains.

The use of the proposed encoder-decoder architecture conveniently allows to reuse of a well-established robot morphology to sample feasible joint values, generate trajectories, and control the joint values. It should be noted that the derivation of the encoder and decoder are model-based rather than learned using machine learning approaches. Therefore, the transformation is geometrically exact and relies only on the kinematic design parameters as listed in Table I and used in (10). Furthermore, note that, considering (8), forward (5) and inverse robot-dependent mapping (6) are provided too. Both representing the encoder and decoder part are useful in frameworks assuming constant curvature.

A. Limitations

It is assumed that the displacement-actuated joints can be used to pull and push. However, for tendon-driven continuum robots, for instance, tendons can only be pulled. Therefore, the location of the joints, *i.e.*, location of the tendon holes, influences the motion capabilities of this type of continuum robot. Further investigation on an appropriate clipping, shift, or storing approach is necessary to account for negative values related to pushing the tendons.

Regarding the encoder-decoder architecture, latent-space variables can be identified. Using (9), the latent-space variables are the Clarke coordinates. However, for the more general case using (10), the latent-space variables are not the Clarke coordinates. To reconstruct some sort of Clarke coordinates, a function $f(d_i)$ considering all distances d_i could be used. For future work, the choice of the function $f(d_i)$ should be investigated to link the latent space variables back to a virtual displacement. We kindly refer to [9], [19] on the concept of virtual displacement. The choice of function $f(d_i)$ is important to clip the Clarke coordinates, *e.g.*, to consider joint limits or saturation for PID controllers.

B. Possible Applications

Error propagation is a possible application. Our approach can be used to analyze the propagation of error due to uncertainty in the joint location, see Fig. 11 for visual aid. Furthermore, it would be possible to account for this error by accounting for the optimized value of the joint locations in the encoder-decoder architecture (10), within a piecewise constant curvature framework. However, an extension of Clarke transform to include twist to represent displacement-actuated continuum robot with $3\,\mathrm{dof}$ per segment, *e.g.,* [20], [21], would be desirable.

The consideration of asymmetric joint location opens new possibilities in designing physical hardware. For example, for certain applications, it would be necessary to integrate

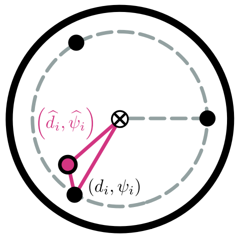


Fig. 11. Uncertainty of the joint location results in asymmetric joint location. The polar coordinates (d_i, ψ_i) refers to assumed location, whereas polar coordinates $\left(\widehat{d}_i, \widehat{\psi}_i\right)$ is the true location.

working channels to deploy instruments for instance. Those working channels cannot be used to realize a displacement-actuated joint. In this case, an asymmetric joint location is desirable. Another example is the use of additional displacement-actuated joints to increase the capability to withstand unwanted bending due to external forces or gravity. Both examples are not mutually exclusive. Figure 3 illustrates possible designs.

The encoder-decoder architecture allows the resort to a surrogate robot with a well-established morphology. As a use case, one might reinforce a well-established design with $n = 4$ with two more displacement-actuated joints to withstand gravity, see Fig. 3. The encoder-decoder architecture is a key approach to reuse developed approaches for the well-established design. For instance, the joint values generated by a planner can be transformed to joint values of the new design without reworking the developed planner, *cf.* Fig. 5.

In fact, the encoder-decoder architecture can be used to translate frameworks for different robot morphologies. For example, the model-based controller by Della Santina *et al.* [8] is designed for a soft robot with four bellows. The proposed encoder-decoder architecture can be used to consider a soft robot with five bellows instead of four bellows. Obviously, certain parameters of the controller need to be tuned to the target robot. Another example, by investing slightly more work in the adaptation, the controller by Della Santina *et al.* [8] can be adapted to a tendon-driven continuum robot with five tendons. This example illustrates that the Clarke transform can facilitate knowledge transfer between research fields. A more straightforward example is the knowledge transfer between approaches for well-established designs with $n = 3$ and $n = 4$ symmetric arranged joints.

VI. Conclusions

We utilize the Clarke transform and relax the assumption on the joint locations to an arbitrary number of joints and arbitrary joint locations. We show how to generate feasible joint values for the target robot using a surrogate robot that can have a well-established robot morphology, *e.g.*, three joints symmetrically arranged around the center-line. One of the key approaches is the proposed encoder-decoder architecture to map the joint values of a surrogate robot to the target robot's joint values. Our work allows the consideration of asymmetrically arranged joints opening the possibility to extend mechanical designs and the investigation of error propagation due to uncertainties.

References

[1] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum Robots for Medical Applications: A Survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, 2015.

[2] P. E. Dupont, N. Simaan, H. Choset, and D. C. Rucker, "Continuum robots for medical interventions," *Proceedings of the IEEE*, vol. 110, no. 7, pp. 847–870, 2022.

[3] X. Dong, D. Palmer, D. Axinte, and J. Kell, "In-situ repair/maintenance with a continuum robotic machine tool in confined space," *Journal of Manufacturing Processes*, vol. 38, pp. 313–318, 2019.

[4] M. Russo, S. M. H. Sadati, X. Dong, A. Mohammad, I. D. Walker, C. Bergeles, K. Xu, and D. A. Axinte, "Continuum robots: An overview," *Advanced Intelligent Systems*, vol. 5, no. 5, p. 2200367, 2023.

[5] J. Lu, F. Du, F. Yang, T. Zhang, Y. Lei, and J. Wang, "Kinematic modeling of a class of n-tendon continuum manipulators," *Advanced Robotics*, vol. 34, no. 19, pp. 1254–1271, 2020.

[6] T. F. Allen, L. Rupert, T. R. Duggan, G. Hein, and K. Albert, "Closed-form non-singular constant-curvature continuum manipulator kinematics," in *IEEE International Conference on Soft Robotics*, 2020, pp. 410–416.

[7] M. M. Dalvand, S. Nahavandi, and R. D. Howe, "General Forward Kinematics for Tendon-Driven Continuum Robots," *IEEE Access*, vol. 10, pp. 60 330–60 340, 2022.

[8] C. Della Santina, A. Bicchi, and D. Rus, "On an improved state parametrization for soft robots with piecewise constant curvature and its use in model based control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1001–1008, 2020.

[9] R. M. Grassmann, A. Senyk, and J. Burgner-Kahrs, "Clarke Transform – A Fundamental Tool for Continuum Robotics," *arXiv preprint arXiv:2409.16501*, 2024.

[10] R. M. Grassmann and J. Burgner-Kahrs, "Clarke Transform and Clarke Coordinates – A New Kid on the Block for State Representation of Continuum Robots," in *40th Anniversary of the IEEE International Conference on Robotics and Automation*, 2024.

[11] M. Janaszek, "Extended Clarke transformation for n-phase systems," *Prace Instytutu Elektrotechniki*, no. 274, pp. 5–26, 2016.

[12] J. L. Willems, "Generalized Clarke components for polyphase networks," *IEEE Transactions on Education*, vol. 12, no. 1, pp. 69–71, 1969.

[13] A. Rockhill and T. Lipo, "A generalized transformation methodology for polyphase electric machines and networks," in *IEEE International Electric Machines & Drives Conference*, 2015, pp. 27–34.

[14] R. M. Grassmann, A. Senyk, and J. Burgner-Kahrs, "On the Disentanglement of Tube Inequalities in Concentric Tube Continuum Robots," in *IEEE International Conference on Robotics and Automation*, 2024.

[15] R. M. Grassmann and J. Burgner-Kahrs, "Quaternion-Based Smooth Trajectory Generator for Via Poses in SE(3) Considering Kinematic Limits in Cartesian Space," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4192–4199, 2019.

[16] L. Biagiotti and C. Melchiorri, *Trajectory planning for automatic machines and robots*. Springer Science & Business Media, 2008.

[17] S. Haddadin, S. Parusel, L. Johannsmeier, S. Golz, S. Gabl, F. Walch, M. Sabaghian, C. Jaehne, L. Hausperger, and S. Haddadin, "The Franka Emika Robot: A Reference Platform for Robotics Research and Education," *IEEE Robotics & Automation Magazine*, 2022.

[18] A. De Luca and W. J. Book, "Robots with flexible elements," in *Springer Handbook of Robotics*. Springer, 2016, pp. 243–282.

[19] M. Firdaus and M. Vadali, "Virtual Tendon-Based Inverse Kinematics of Tendon-Driven Flexible Continuum Manipulators," in *International Conference on Advances in Robotics*, 2023, pp. 1–5.

[20] R. M. Grassmann, P. Rao, Q. Peyron, and J. Burgner-Kahrs, "FAS – A Fully Actuated Segment for Tendon-Driven Continuum Robots," *Frontiers in Robotics and AI*, vol. 9, 2022.

[21] Y. Lei, Y. Sugahara, and Y. Takeda, "Design and inverse kinematics of a novel tendon-driven continuum manipulator capable of twisting motion," in *International Symposium on Advances in Robot Kinematics*. Springer, 2022, pp. 228–236.