applied sciences

MDPI

Article

Kinematic-Model-Free Redundancy Resolution Using Multi-Point Tracking and Control for Robot Manipulation

Ahmad AlAttar [1,2,*], Francesco Cursi [1,3] and Petar Kormushev [1]

[1] Robot Intelligence Lab, Dyson School of Design Engineering, Imperial College London, Exhibition Road, London SW7 2DB, UK; f.cursi17@imperial.ac.uk (F.C.); p.kormushev@imperial.ac.uk (P.K.)
[2] Dubai Future Lab, 77 Sheikh Zayed Road, Dubai, United Arab Emirates
[3] Hamlyn Centre, Imperial College London, Exhibition Road, London SW7 2BU, UK
* Correspondence: a.alattar19@imperial.ac.uk

Abstract: Robots have been predominantly controlled using conventional control methods that require prior knowledge of the robots' kinematic and dynamic models. These controllers can be challenging to tune and cannot directly adapt to changes in kinematic structure or dynamic properties. On the other hand, model-learning controllers can overcome such challenges. Our recently proposed model-learning orientation controller has shown promising ability to simultaneously control a three-degrees-of-freedom robot manipulator's end-effector pose. However, this controller does not perform optimally with robots of higher degrees-of-freedom nor does it resolve redundancies. The research presented in this paper extends the state-of-the-art kinematic-model-free controller to perform pose control of hyper-redundant robot manipulators and resolve redundancies by tracking and controlling multiple points along the robot's serial chain. The results show that with more control points, the controller is able to reach desired poses in fewer steps, yielding an improvement of up to 66%, and capable of achieving complex configurations. The algorithm was validated by running the simulation 100 times, and it was found that, in 82% of the times, the robot successfully reached the desired target pose within 150 steps.

Keywords: kinematic-model-free control; model-learning control; redundancy resolution; multi-point tracking; adaptive control

Citation: AlAttar, A.; Cursi, F.; Kormushev, P. Kinematic-Model-Free Redundancy Resolution Using Multi-Point Tracking and Control for Robot Manipulation. *Appl. Sci.* **2021**, *11*, 4746. https://doi.org/10.3390/app11114746

Academic Editor: Dario Richiedei

Received: 1 May 2021
Accepted: 20 May 2021
Published: 21 May 2021

1. Introduction

Conventional robot controllers require both the kinematic and dynamic models (heron cumulatively referred to as kinodynamic model) of the robot in order to perform accurate control [1]. These controllers require an accurate model of the robot. However, soft robots, morphing robots, malleable robots, transforming robots, and evolving robots can be challenging to model accurately and control. Model-based approaches have remained the dominant control method for over 50 years [2]. Model-based control of soft/continuum robots underperform because modeling these structures are achieved by means of approximations. Therefore, learning-based or model-free techniques may be preferred to overcome some of these challenges.

Recently, a new class of model-learning controllers emerged, showing promising ability to bypass the need for any prior robot model. These controllers directly learn a kinodynamic model online to control a robot. In our original work, the Encoderless controller [3] and Kinematic-Free controller [4], the end-effector position of a two-degree-of-freedom (2-degrees-of-freedom) robot manipulator was controlled in task space. The controller works by generating local linear models that represent the robot's local kinodynamics, which are used to actuate the robot's end effector towards a given target position. In our most recent work, the model-learning orientation controller [5] has successfully controlled the position and orientation of a 3-degrees-of-freedom robot manipulator's end effector in

task space without any predefined kinodynamic model of the robot. However, this controller underperforms, taking longer than required to converge to the desired pose, when higher-degrees-of-freedom robots are to be controlled, leaving redundancies unresolved, as well.

The Kinematic-Model-Free (KMF) Multi-Point controller presented in this paper extends the state-of-the-art kinematic-model-free orientation controller, enabling the control of hyper-redundant robot manipulators and resolving redundancies in these manipulators by tracking and controlling multiple points along the kinematic chain of such robots.

1.1. Contributions

The contributions of this papers are as follows: (1) the novel kinematic-model-free multi-point controller presented in this paper expands the state-of-the-art kinematic-model-free controller to control the pose of multiple points along a robot's kinematic chain; (2) the controller allows for the control of robots with high degrees-of-freedom, scaling up from the 3-degrees-of-freedom manipulator in our previous work; and (3) the controller is also capable of resolving redundancies in highly redundant system, such as continuum robots.

1.2. Paper Structure

Section 2 introduces work related to the research presented in this paper. Section 3 summarizes the fundamental mathematical background required to understand the working principles of the novel kinematic-model-free multi-point controller. Section 4 introduces the proposed controller along with its mathematical equations. Section 5 presents experimental results. Finally, Section 6 concludes the paper.

2. Related Work

Model-based controllers have been the predominant method of control for robot manipulation. The Jacobian of a robot, which relates the joint velocities to the end-effector velocity, is calculated prior to control [6], after which it is controlled using forward and inverse kinematics [7]. Using such models imposes many limitations [8] and assumes the Denavit-Hartenberg parameters, that define the robots kinematics, to be static. One example of model-based control is visual servoing that uses an external camera to calculate the required end-effector velocity to reach a desired target [9]. However, this assumption falls apart in the case of soft robots, when robots are damaged, or when robots transform or evolve in any way, such as altering the end effector by changing its gripper. Therefore, model-based conventional controllers are accurate but not adaptable.

Other researchers performed model-free control by approximating a Jacobian that is later used to actuate a robot. Researchers in Reference [10] controlled a deformable manipulator and in Reference [11] controlled a soft manipulator by approximating its Jacobian model. Other researchers controlled a dynamically-undefined robot by first estimating the robot's dynamic parameters and then controlled a robot using conventional methods [12]. Learning methods, such as motor babbling, that start with no predefined model of a robot's kinodynamics are used to approximate a model to control robots [13]. Other researchers use machine learning [14] and reinforcement learning algorithms [15] to control robots.

On the other hand, researchers in Reference [16] directly controlled current motors to actuate an unmanned underwater vehicle by means of deterministic artificial intelligence where uncertainties are ignored by making use of first principles. Researchers in Reference [17] controlled a SCARA robot's position through motor voltages using extended state observer to estimate system dynamics. Other researchers studied hybrid model-free and model-based approaches to increase the performance of robot navigation [18]. However, most of these learning approaches require encoder feedback for estimating the robot's pose.

The classical kinematic-model-free position controllers, the Encoderless [3] and Kinematic-Free [4] controllers, do not require any prior model of the robot. They operate by generating psuedo-random actuation signals that actuate a planar 2-degrees-of-freedom

robot's joints causing a positional displacement on the robot's end effector. The effect on the robot's end effector is observed using an external camera and is recorded. The recorded data are then used to generate a local linear model that actuates the robot's end effector towards a given target. Any discrepancies will re-trigger another exploratory phase. The classical kinematic-model-free controller was then extended to simultaneously control the position and orientation of a 3-degrees-of-freedom robot's end effector [5] using locally-weighted dual quaternions. These kinematic-model-free controllers, however, are unable to resolve redundancies in redundant robots, such as manipulators with high degrees of freedom [19] and continuum robots [20].

3. Mathematical Background

3.1. Quaternions

Quaternions, introduced by Hamilton in 1844 [21], are extend the complex number system used to represent rigid-body rotations. A quaternion $\mathbf{q}$ is the sum of a scalar s and a vector $\mathbf{v} = (q_1, q_2, q_3)$:

$$\mathbf{q} = s + \mathbf{v} = s + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}, \tag{1}$$

where $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ are unit vectors pointing in the x, y, z directions.

The conjugate of a quaternion $\mathbf{q}$ is defined as:

$$\mathbf{q}^* = s - \mathbf{v} = s - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k}, \tag{2}$$

which is used to calculate the norm of a quaternion as follows:

$$\| \mathbf{q} \| = \sqrt{\mathbf{q}^*\mathbf{q}}. \tag{3}$$

Unit quaternions satisfy $s^2 + q_1^2 + q_2^2 + q_3^2 = 1$ or simply $\| \mathbf{q} \| = 1$.

More information on quaternions can be found in Reference [22].

3.2. Dual Quaternions

Dual quaternions, presented by Clifford in 1871 [23], are extend the dual number system. A dual quaternion $\hat{\mathbf{q}}$ is defined as the sum of two quaternions:

$$\hat{\mathbf{q}} = \mathbf{p} + \epsilon\mathbf{q}, \tag{4}$$

where $\mathbf{p} = s_p + \mathbf{v}_p$ is the real part, and $\mathbf{q} = s_q + \mathbf{v}_q$ is the dual part. The dual element ϵ is nilpotent, i.e., $\epsilon^2 = 0$.

The conjugate of a dual quaternion is defined as:

$$\hat{\mathbf{q}}^* = \mathbf{p}^* + \epsilon\mathbf{q}^*, \tag{5}$$

which is used to calculate the dual quaternion norm:

$$\| \hat{\mathbf{q}} \| = \sqrt{\hat{\mathbf{q}}^*\hat{\mathbf{q}}}. \tag{6}$$

A unit dual quaternion satisfies $s_p^2 + p_1^2\& + p_2^2 + p_3^2 = 1$ and $s_p s_q + p_1 q_1\& + p_2 q_2 + p_3 q_3 = 0$, or simply $\| \hat{\mathbf{q}} \| = 1$ and $\langle \mathbf{p}, \mathbf{q} \rangle = 0$. The vec_8 operator maps the elements of a dual quaternion to an 8-dimensional vector as follows:

$$vec_8(\hat{\mathbf{q}}) = (s_p, p_1, p_2, p_3, s_q, q_1, q_2, q_3). \tag{7}$$

3.3. Rigid Transformation

Rigid-body transformations are represented using dual quaternions as follows:

$$\hat{\mathbf{q}} = \mathbf{r} + \frac{\epsilon}{2}\mathbf{t}\mathbf{r}, \tag{8}$$

where $\mathbf{r}$ is the rotation quaternion, and $\mathbf{t}$ is the translation quaternion [24]. Note that the scalar part is always zero for the translation quaternion, i.e., $t = 0 + \mathbf{v}_t = \mathbf{v}_t$.

3.4. Interpolation

Shoemake introduced the Spherical Linear Interpolation to interpolate between two quaternions [25]:

$$\mathbf{m} = \Phi(\mathbf{p}, \mathbf{q}, t) = \mathbf{p}(\mathbf{p}^*\mathbf{q})^t, \tag{9}$$

where $t \in [0, 1]$. On the other hand, Screw Linear Interpolation for dual quaternions is defined similarly as $\hat{\mathbf{p}}(\hat{\mathbf{p}}^*\hat{\mathbf{q}})^t$.

3.5. Relative Dual Quaternions

The relative rotational displacement, $\mathbf{b}_{rel}$, between two quaternions, $\mathbf{p}$ and $\mathbf{q}$ is found as follows:

$$\mathbf{b}_{rel} = \mathbf{p}^*\mathbf{q}. \tag{10}$$

On the other hand, the relative dual quaternion between two dual quaternions, $\hat{\mathbf{p}}$ and $\hat{\mathbf{q}}$, is [5]:

$$DQrel(\hat{\mathbf{p}}, \hat{\mathbf{q}}) = \hat{\mathbf{b}}_{rel} = \mathbf{r}_1^*\mathbf{r}_2 + \frac{\epsilon}{2}(\mathbf{t}_1^* + \mathbf{t}_2)\mathbf{r}_1^*\mathbf{r}_2. \tag{11}$$

Note that $\hat{\mathbf{b}}_{rel}$ must be normalized if not.

3.6. Dual Quaternion Distance

The same distance metric defined in Reference [5] will be adopted for the proposed controller, which provides a measure of how close a control point is to a target point:

$$DQdist(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2) = \| \, vec8(\hat{\mathbf{0}}) - vec8(DQrel(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2)) \, \|, \tag{12}$$

where $\hat{\mathbf{0}}$ is the null dual quaternion.

3.7. Dual Quaternion Regression

To find the regression coefficients or weights, $\mathbf{w} = [w_1, w_2, ..., w_n]$, of the dual quaternions $\hat{\mathbf{Q}} = [\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, ..., \hat{\mathbf{q}}_n]$ that yield some desired dual quaternion $\hat{\mathbf{q}}_d$, the following cost function is optimized:

$$DQreg(\hat{\mathbf{q}}_d, \hat{\mathbf{Q}}) = \min_{\mathbf{w}} DQdist(\hat{\mathbf{q}}_d, \hat{\mathbf{q}}_r), \tag{13}$$

where

$$\hat{\mathbf{q}}_r = DQLC(\hat{\mathbf{Q}}, \mathbf{w}), \tag{14}$$

which can be optimized using a generic optimization algorithm, such as MATLAB's FMINCON optimization algorithm. The Dual Quaternion Linear Combination (DQLC) function is defined as:

$$DQLC(\hat{\mathbf{Q}}, \mathbf{w}) = DQLC(\hat{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_n, w_1, \ldots, w_n), \tag{15}$$

$$= \prod_{i=1}^{n} \mathbf{r}_i^{w_i} + \frac{\epsilon}{2} \sum_{i=1}^{n} w_i \mathbf{t}_i \prod_{i=1}^{n} \mathbf{r}_i^{w_i}. \tag{16}$$

4. Proposed Approach

After specifying the control points that are to be controlled via markers or trackers, the controller first goes through an actuators discovery phase. Each actuator, $j \in \mathbb{J}$, is moved

one after another but in no particular order, where $\mathbb{J}$ is the set of all actuators. For each control point $i \in \mathbb{I}$, the actuators that can contribute to displacing it are assigned to that point, where $\mathbb{I}$ is the set of all control points. Thus, actuators could be assigned to multiple control points. The control point that can be displaced by all actuators, most probably being the end effector of the robot, is assigned an index $i = n$, where n is the cardinality of set $\mathbb{I}$. The discovery phase produces the following Boolean matrix:

$$S = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} = \begin{bmatrix} \delta_1^1 & \delta_2^1 & \cdots & \delta_m^1 \\ \delta_1^2 & \delta_2^2 & \cdots & \delta_m^2 \\ \vdots & \vdots & \ddots & \vdots \\ \delta_1^n & \delta_2^n & \cdots & \delta_m^n \end{bmatrix}, \tag{17}$$

where m is the cardinality of set $\mathbb{J}$. The function above is defined as: $\delta_j^i = 1$ if actuator j can displace control point i and $\delta_j^i = 0$ otherwise. To find the number of actuators that contribute to some control point i, the dot product can be used $a_i = s_i \cdot s_i$. A vector of the number of such actuators can be defined as $a = [a_1, a_2, \cdots, a_n]$.

The kinematic-model-free multi-point controller, just like any other kinematic-model-free controller, assumes that the local kinodynamics information of the manipulator is attainable. This information is attained by generating pseudo-random actuation signals to actuate the robot joints and observing their result on the robot's control point(s). A high-level diagram summarizing the kinematic-model-free multi-point approach is shown in Figure 1.

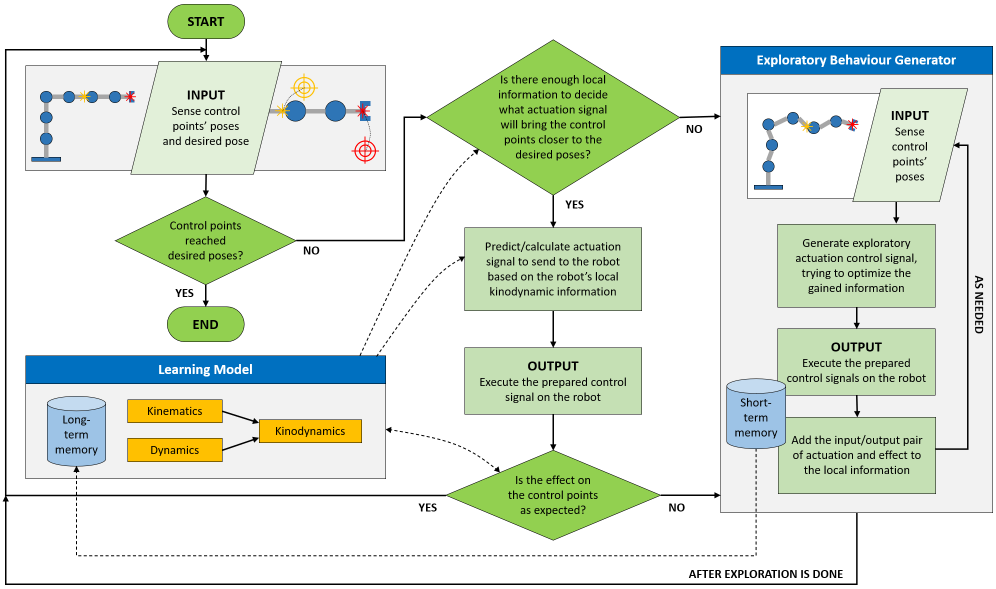

Figure 1. Flowchart of kinematic-model-free multi-point control.

The actuation primitives produce signals $\tau(t)$ which are sent to each of the robot's actuators and are defined as:

$$\tau(t) = \begin{cases} \tau_p & t \in [t_0, t_0 + \frac{d_p}{2}) \\ -\tau_p & t \in [t_0 + \frac{d_p}{2}, t_0 + d_p] \\ 0 & \text{otherwise} \end{cases}. \tag{18}$$

Note that actuation primitives could be motor torques, motor currents, joint positions, etc. The controller tracks the poses of the control points, $\hat{\mathbf{q}}_i$, using dual quaternion representation which yield the controlled points vector:

$$\hat{\boldsymbol{\phi}} = [\hat{\mathbf{q}}_i, \cdots, \hat{\mathbf{q}}_n]. \tag{19}$$

The controller attempts to bring the points $\hat{\mathbf{q}}_i$ towards desired target poses $\hat{\mathbf{d}}_i^f$, yielding the desired poses vector as follows:

$$\hat{\boldsymbol{\gamma}} = [\hat{\mathbf{d}}_i^f, \cdots, \hat{\mathbf{d}}_n^f]. \tag{20}$$

By controlling multiple points, redundancies can be easily resolved by specifying the desired poses of other points along the robot's serial chain. The controller terminates when $\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{\gamma}}$, i.e., when the poses of the controlled points of interest coincide with their corresponding desired poses. This condition can be relaxed by allow the controlled points' poses to reach within some deviation σ of their respective desired target poses. Otherwise, the controller is to actuate the control points towards a set of intermediate poses $\hat{\mathbf{d}}_i^t$ that lead to the final desired target poses $\hat{\mathbf{d}}_i^f$ in discrete steps. The intermediate poses are found using screw linear interpolation interpolation between the current poses of the controlled points, $\hat{\mathbf{q}}_i$, and desired target poses, $\hat{\mathbf{d}}_i^f$. At each step, the kinematic-model-free multi-point controller estimates actuation primitives, b_i, to move each controlled point $i \in \mathbb{I}$ towards their respective desired poses:

$$b_i = \begin{bmatrix} \tau_p^1(\hat{p}) \\ \tau_p^2(\hat{p}) \\ \vdots \\ \tau_p^m(\hat{p}) \end{bmatrix}, \tag{21}$$

where $\tau_p^j(\hat{p})$ is the magnitude of the actuation primitive of actuator j. After each step where the robot joints have been actuated, the displacement of the control points are recorded as observations, which are calculated using the *DQrel* function. The nearest k observation dual quaternions to each point $i \in \mathbb{I}$ form the observation matrix:

$$\hat{\boldsymbol{\psi}} = \begin{bmatrix} \hat{\boldsymbol{\psi}}_1 \\ \hat{\boldsymbol{\psi}}_2 \\ \vdots \\ \hat{\boldsymbol{\psi}}_n \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{o}}_1^1 & \hat{\mathbf{o}}_2^1 & \dots & \hat{\mathbf{o}}_k^1 \\ \hat{\mathbf{o}}_1^2 & \hat{\mathbf{o}}_2^2 & \dots & \hat{\mathbf{o}}_k^2 \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{o}}_1^n & \hat{\mathbf{o}}_2^n & \dots & \hat{\mathbf{o}}_k^n \end{bmatrix}, \tag{22}$$

where $\hat{\mathbf{o}}_k^i$ is the k^{th} closest observation dual quaternion of controlled point i. The actuation primitives corresponding to these observations form the actuation matrix:

$$A_i = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \cdots & \tau_p^1(p_k) \\ \tau_p^2(p_1) & \tau_p^2(p_2) & \dots & \tau_p^2(p_k) \\ \vdots & \vdots & \ddots & \vdots \\ \tau_p^{a_i}(p_1) & \tau_p^{a_i}(p_2) & \cdots & \tau_p^{a_i}(p_k) \end{bmatrix}. \tag{23}$$

The desired actuation primitives, b_i, is assumed to be a linear combination of the k nearest actuation primitives such that:

$$A_i w_i = b_i, \tag{24}$$

where $w_i = [w_1^i, w_2^i, ..., w_k^i]^T$ is a set of unknown weights. The weights are estimated by minimizing the following:

$$DQreg(\hat{\mathbf{d}}_i^t, \hat{\boldsymbol{\psi}}_i) = \min_{w_i} DQdist(\hat{\mathbf{d}}_i^t, DQLC(\hat{\boldsymbol{\psi}}_i, w_i)). \tag{25}$$

With the weights for each control point, the desired actuation is then computed using Equation (24) for each control point. The respective actuation matrix is used in the calculation. After the primitive for each control point has been obtained, the following averaging is done to dampen and stir the actuation of joints that contribute to the displacement of the control points:

$$b'_i = \frac{\sum_{r=1}^{r=i} \eta_r^i Z(b_r)}{\sum_{r=1}^{r=i} \eta_r^i}, \tag{26}$$

where the operator Z adds zero padding to a vector to keep the length equal to m, and the averaging weights matrix is a lower triangular matrix:

$$H = \begin{bmatrix} \eta_1^1 & 0 & 0 & 0 & \cdots & 0 \\ \eta_1^2 & \eta_2^2 & 0 & 0 & \cdots & 0 \\ \eta_1^3 & \eta_2^3 & \eta_3^3 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \eta_1^n & \eta_2^n & \eta_3^n & \eta_4^n & \ddots & \eta_n^n \end{bmatrix}. \tag{27}$$

In the case that the averaging weights for each control point is normalized, Equation (26) would simply be:

$$b'_i = \sum_{r=1}^{r=i} \eta_i^r Z(b_r). \tag{28}$$

Alternatively, to reduce the number of optimizations required, it is possible to fuse the optimization of each control point together into a wholesome expression as follows:

$$DQreg(\hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\psi}}) = \min_{w} \sum_{i=1}^{i=n} DQdist(\hat{\mathbf{d}}_i^t, DQLC(\hat{\boldsymbol{\psi}}_i, w)), \tag{29}$$

where the weights are later used in $A_n w = b$ to estimate the required actuation. After each discrete actuation, the resulting displacement on the pose of each control point is compared to the expected displacement. Should the difference be significant, a new exploratory phase is triggered to collect more data about the robot's local kinodynamics as shown in Figure 1. The controller leaves the exploratory phase once it has completed a predefined number of exploratory steps. The controller terminates once all control point poses have reached the desired poses or close enough to them, $DQdist(\hat{\mathbf{q}}, \hat{\mathbf{d}}) < \sigma$.

This controller, just like its other kinematic-model-free predecessors, does not require any prior model of the robot's kinodynamics. As such, the controller is agnostic to any kinodynamic changes in the robot, such as alteration in the robot's link dimensions. Since the kinematic-model-free multi-point controller is not based on any accurate robot model, the accuracy of the controller is less than conventional controllers. Moreover, this controller operates in discrete steps, making it significantly slower than other controllers and the motion can be relatively jerky. As with other complex models, such as those found in Reference [26,27], it is difficult to proof the convergence of the kinematic-model-free controller due to the nonlinear functions that are involved.

5. Simulation Results

The controller is tested on a simulated redundant 9-degrees-of-freedom continuum robot, shown in Figure 2.

The robot was simulated using Matlab's Robotics Toolbox [28] and DQ Robotics Toolbox [29]. The actuation primitives used were joint position command signals to drive the control points towards the target poses. The Denavit–Hartenberg parameters of the robot is listed in Table 1.

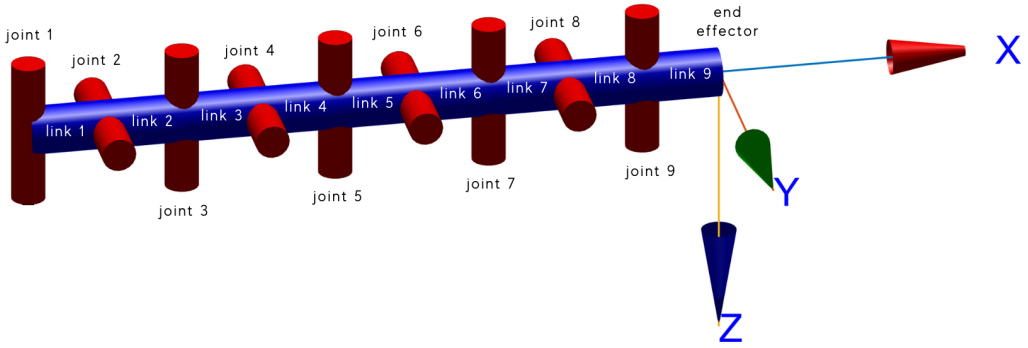

Figure 2. The simulated 9-degrees-of-freedom continuum robot.

Table 1. Denavit–Hartenberg parameters of redundant robot.

Link	d_i	θ_i	a_i	α_i
1	0	0	0.05	1.5708
2	0	0	0.05	1.5708
3	0	0	0.05	−1.5708
4	0	0	0.05	1.5708
5	0	0	0.05	−1.5708
6	0	0	0.05	1.5708
7	0	0	0.05	−1.5708
8	0	0	0.05	1.5708
9	0	0	0.05	0

The controller was programmed as described in Section 4 using $k = 4$ nearest neighbors that comprise of actuation primitive and observation pairs. Since the experiments were conducted in simulation, a tracking camera was not needed to determine the poses of the control points. In addition, actuators discovery was also omitted in the simulation. Instead, the simulator provided these information directly. Furthermore, gravity was omitted in the simulation to keep the focus on multi-point pose control, deferring gravity compensation to future work.

Three experiments were performed to study the capabilities of the proposed kinematic-model-free multi-point controller. The first experiment involved end-effector pose control for the continuum robot with increasing number of degrees of freedom enabled. This experiment shows the degrading performance of the controller when increasing the number of degrees of freedom using a single control point. In the second experiment, the 9-degrees-of-freedom continuum robot was controlled using 1, 2, and 3 control point(s). This experiment highlights the capability of the kinematic-model-free multi-point controller to control robots of high degrees of freedom and resolve redundancies. Lastly, the robustness of the kinematic-model-free multi-point controller was measured by executing the simulation experiment 100 times with randomly-generated configurations.

5.1. Single-Point Control

In the first experiment, a single control point situated at the end effector of the continuum robot was used to actuate the robot towards a desired target pose. Initially, only the last 2 joints of the robot were enabled. Gradually, an extra joint was enabled, increasing the robot's degrees of freedom. The robot began from the home pose where the control-point's task-space position was (0, 0, 0.4500), and orientation was (0, 0, 3.1416). Orientations in this paper are specified in ZYX Euler angle representation for simplicity. The target position was (0.4218, 0.0240, 0.0605) and orientation was (0.6202, −0.6009, 2.7578). The results are shown in Figure 3 and tabulated in Table 2.

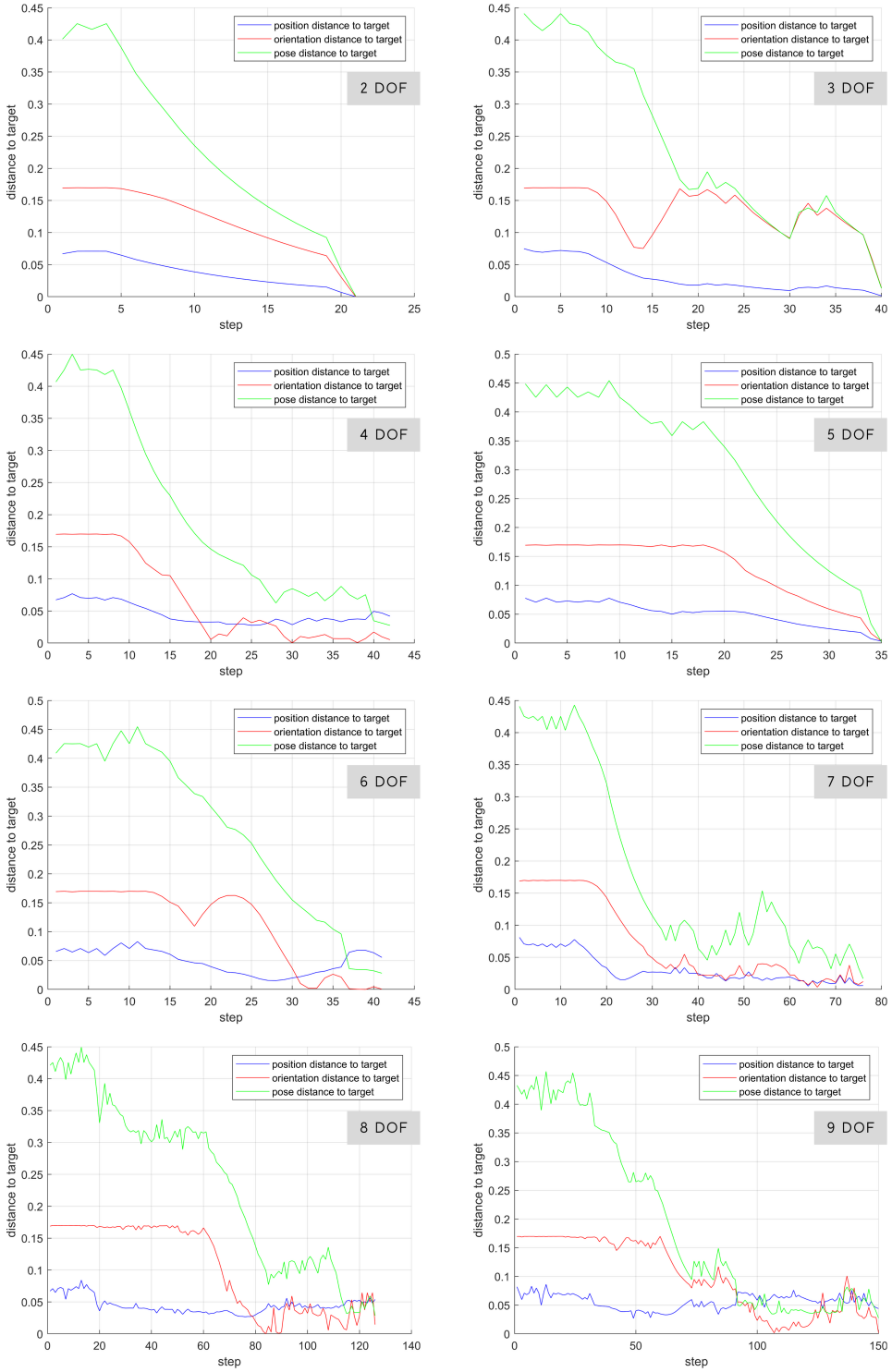

Figure 3. The controller performed the task of reaching, bringing a single control point at the end effector to a desired target pose. As more joints were enabled, the performance was degraded. In particular, in the 2-degrees-of-freedom case, the robot reaches the target in 21 steps, while, in the 9-degrees-of-freedom case, the robot reaches the target in 150 steps.

Table 2. Reaching steps for different degrees of freedom enabled.

Degrees of Freedom	Steps to Reach
2	21 steps
3	40 steps
4	42 steps
5	35 steps
6	41 steps
7	76 steps
8	127 steps
9	150 steps

In the simple 2-degrees-of-freedom case, the control point converged to the target pose within 25 steps. When increasing the degrees of freedom, it is noted that the controller takes longer to converge. Although it is evidently possible to reach the target pose within 25 steps, the added redundancies introduced additional nonlinearities that require excessive exploratory behavior. In the case where all 9 joints of the robot were enabled, the control point reached the target pose in 150 steps, a significant decrease in performance. With the stopping condition ($DQdist < \sigma$) applied, it can be seen that the controller can terminate before fully reaching the target pose.

5.2. Multi-Point Control

In this experiment, multiple control points were added to the continuum robot. In the first arrangement, only a single control point was added that was situated at the end effector. In the second arrangement, 2 control points were added, one at the end effector and one at the end of link 5. Finally, in the third arrangement, the 3 control points were added that were evenly distributed throughout the robot's kinematic chain. The placements of the control point for each arrangement are summarized in Table 3.

Table 3. Control points placement.

Arrangement	Control Point 1	Control Point 2	Control Point 3
1	end effector	-	-
2	end of link 5	end effector	-
3	end of link 3	end of link 6	end effector

These arrangements are depicted in Figure 4. Three reaching tasks were preformed with each arrangement. All desired poses were within the robot's configuration space. See Figure 5. The reaching tasks increased in complexity in terms of dexterity required. We note that arrangement 1 corresponds to kinematic-model-free control of the end-effector pose. Thus, this experiment also serves as a comparison with the kinematic-model-free orientation controller in Reference [5]. The results are shown in Figure 6 and tabulated in Table 4.

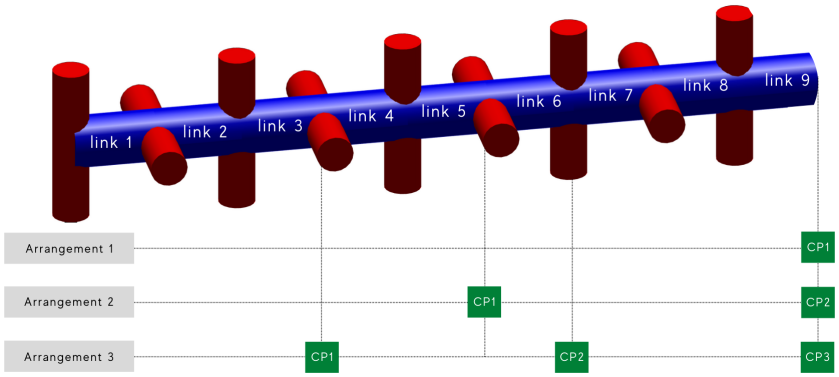

Figure 4. The three arrangements of the control points (CP's) along the robot's kinematic chain. In the first arrangement, only one control point was used which was placed at the end effector. In the second and third arrangements, two and three equally spaced control points were used, respectively.

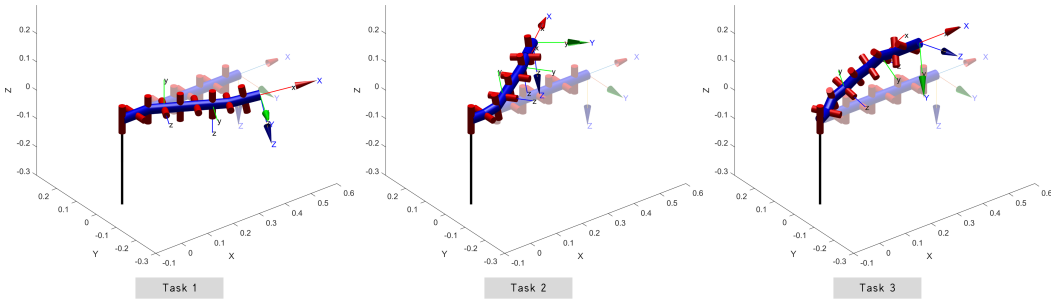

Figure 5. The three reaching tasks to be performed. The translucent pose is the initial configuration from which the robot is to actuate towards the opaque pose.

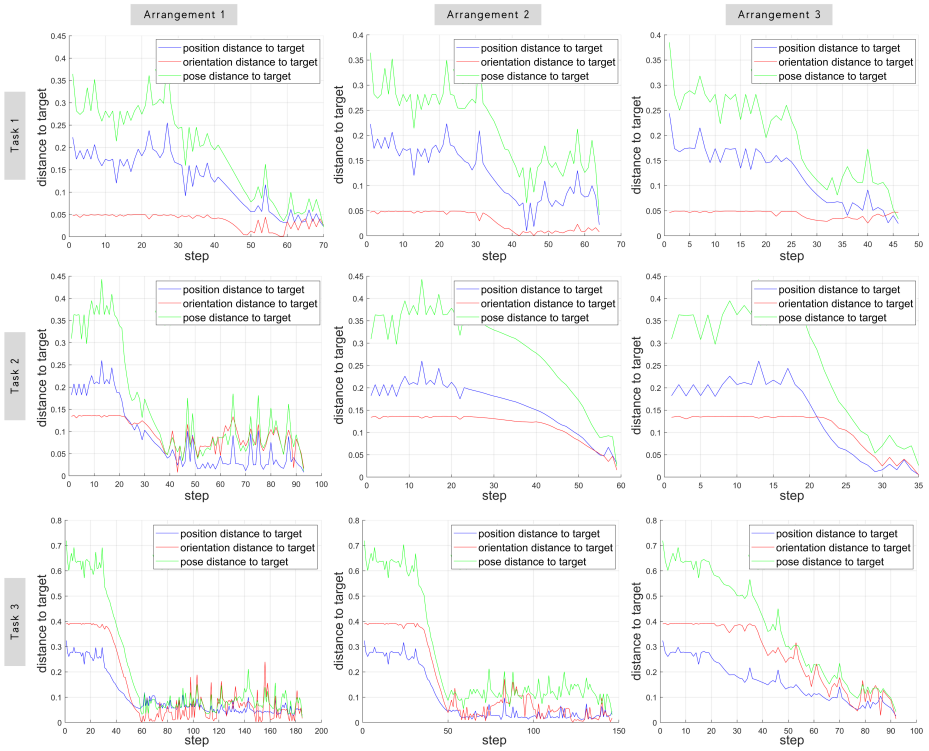

Figure 6. The controller performed three reaching tasks. In all 3 tasks, the controller converged faster when using more control points (CPs).

Table 4. Steps required to complete three reaching tasks using three arrangements of control points.

Task	Arrangement		
	1	**2**	**3**
1	70 steps	64 steps	46 steps
2	93 steps	59 steps	35 steps
3	185 steps	147 steps	92 steps

Only the pose distance of the control point situated at the end effector is shown. It can be seen that the controller converges faster to the desired pose when using more control points situated along the robot's kinematic chain. In task 1, we note that a 66% improvement in convergence speed when using 3 control points. In task 2 and 3, there was a 38% and 50% improvement was measured, respectively. These results indicate that adding more control points will improve convergence and resolve redundancies in hyper-redundant systems. As for accuracy, the termination criteria was set to DQdist < 0.03, achieving a close enough reaching accuracy.

Furthermore, to demonstrate the joints' position during an execution, the joint position are shown in Figure 7. The larger jitters in joint positions correspond to exploratory behavior, which is triggered whenever there is deficiency in local kinodynamic information.

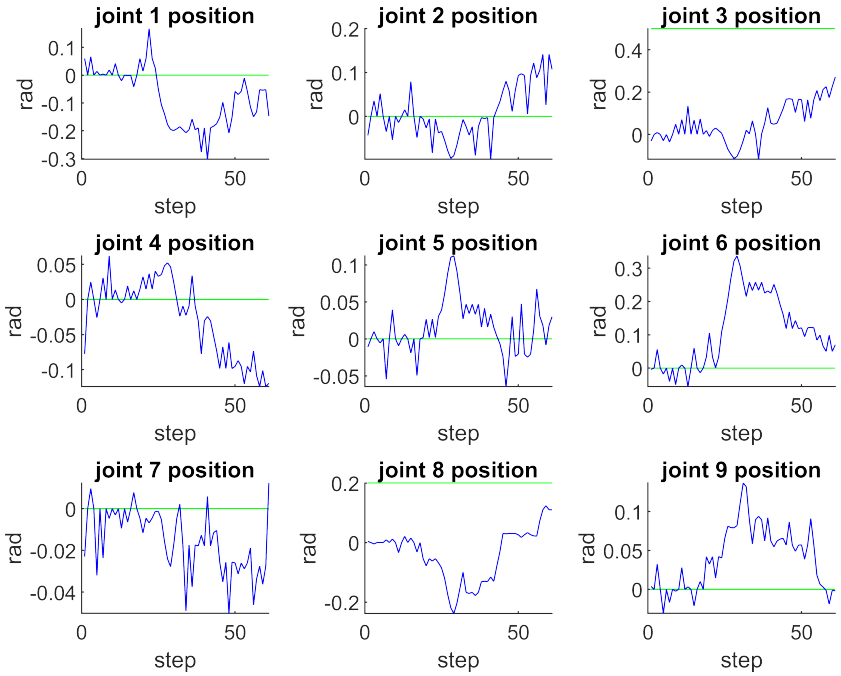

Figure 7. These joint positions correspond to reaching task 1. The larger jitters are due to exploratory behavior, which is triggered to collect more information about the robot's local kinodynamics.

5.3. Robustness

The robustness of the algorithm was measured by executing the simulation 100 times. Each time, the robot must reach a randomly-generated pose that is within its configuration space and it reachable within 150 steps. If the controller takes longer, the simulation is restarted. Three control points were used. One interesting configuration is shown in Figure 8. The controller managed to achieve this bent arch configuration within 150 steps.

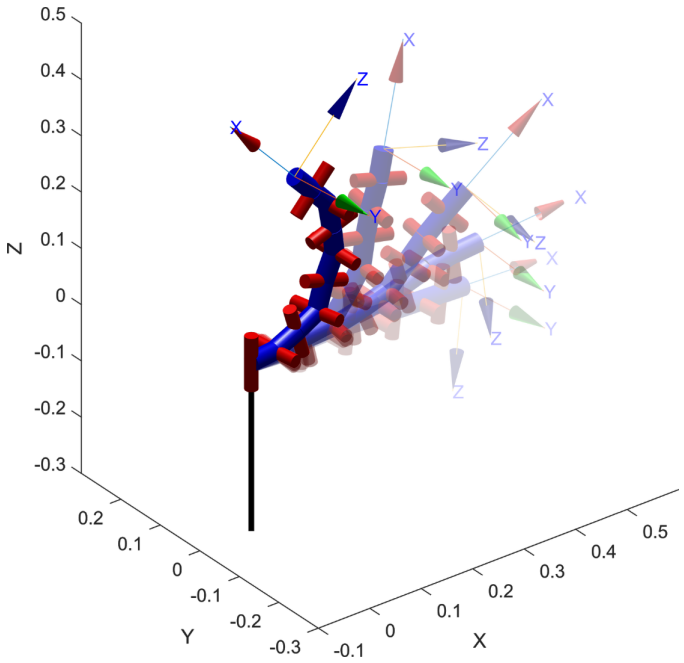

Figure 8. In this simulation run, the controller attempted to actuate the robot to a back arching configuration. It achieved to reach this configuration within 150 steps.

It was observed that 82% of the runs, the controller manages to reach to the target pose. However, in 18% of the runs, the controller did not successfully converge to the solutions that brought the robot's end effector to the desired poses. This can occur when the optimization is stuck in a local minimum, which results in an inaccurate actuation signal that does not optimally move the control points towards the target poses.

6. Conclusions

This paper presents a novel kinematic-model-free control for controlling the pose of multiple points along the kinematic chain of a continuum robot. The proposed controller is able to resolve redundancies and converge faster to a desired configuration. The controller works by actuating the robot using randomly-generated actuation primitives and observing their effects on the control points. This information is then used to build a local linear model using locally weighted dual quaternions. The local linear model is then used to calculate an actuation primitive and actuate the continuum robot towards a desired pose.

In the first experiment, it was evident that the performance of the controller worsened in terms of convergence when increasing the degrees of freedom of the robot. The higher the degrees of freedom, the longer it took to converge. This is due to the nonlinearities introduced, making the local linear model less accurate.

We presented the generalized kinematic-model-free multi-point controller and the method by which observations are used to calculate actuation primitives that will reduce the distance between a robot's current pose and its desired target pose. The simulation results in the second experiment show that this novel controller is capable of performing reaching tasks without any prior kinematic model of the robot, successfully actuating the poses of the control points towards some desired configuration.

Three arrangements for the control points were tested on a simulated 9-degrees-of-freedom continuum robot. Three different robot configurations were targeted, all of which are within the robot's configuration space. It was noted that, when more control points were used, the controller performed better and converged faster to the desired target robot pose. Controlling multiple points along a robot did not only resolves redundancies, but proved to converge faster to the desired pose.

Finally, the controller's robustness was tested by executing the simulation 100 times using random target configurations which the controller had to reach. The random target

configurations lied within the robot's configuration space. The kinematic-model-free multi-point controller succeeded in reaching the target poses within 150 steps 82% of the times.

The proposed controller can be used in applications where simultaneous translational and rotational reaching of redundant robots is required. The controller is useful for resolving redundancies of continuum robots. The kinematic-model-free approach in general can be used for new robots where a model is not readily-available or for robots that are hard to model. In general, the controller is applicable where flexibility is favored over precision and speed. Examples of such applications include use of soft robots for reaching, drawing, picking-and-placing, etc. As for future work, continuous control, gravity compensation, joint limits, joint friction, and physical implementations, are promising research directions to be investigated. These challenges must be addressed before deploying the kinematic-model-free controller on any physical system.

Author Contributions: A.A. developed the theory, methodology, and software, performed the simulation experiments, and wrote the paper. F.C. prepared the robot simulation model and interface and reviewed the paper. P.K. supervised the research and proofread the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported through the resources at the Dubai Future Labs. Full financial support for A.A.'s PhD program was provided by the UAE's Ministry of Education.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cho, H. *Opto-Mechatronic Systems Handbook: Techniques and Applications*; CRC Press: Boca Raton, FL, USA, 2002.
2. Spong, M.W.; Fujita, M. Control in robotics. In *The Impact of Control Technology: Overview, Success Stories, and Research Challenges*; IEEE Control Systems Society: New York, NY, USA, 2011.
3. Kormushev, P.; Demiris, Y.; Caldwell, D.G. Encoderless position control of a two-link robot manipulator. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 943–949.
4. Kormushev, P.; Demiris, Y.; Caldwell, D.G. Kinematic-free position control of a 2-dof planar robot arm. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 5518–5525.
5. AlAttar, A.; Kormushev, P. Kinematic-model-free orientation control for robot manipulation using locally weighted dual quaternions. *Robotics* **2020**, *9*, 76. [CrossRef]
6. Nour, M.; Ooi, J.; Chan, K. Fuzzy logic control vs. conventional PID control of an inverted pendulum robot. In Proceedings of the 2007 International Conference on Intelligent and Advanced Systems, Kuala Lumpur, Malaysia, 25–28 November 2007; pp. 209–214.
7. Ghosal, A. *Robotics: Fundamental Concepts and Analysis*; Oxford University Press: New Delhi, India, 2006.
8. Cheah, C.C.; Liu, C.; Slotine, J.J.E. Adaptive Jacobian tracking control of robots with uncertainties in kinematic, dynamic and actuator models. *IEEE Trans. Autom. Control.* **2006**, *51*, 1024–1029. [CrossRef]
9. Huang, Y.; Su, J. Visual Servoing of Nonholonomic Mobile Robots: A Review and a Novel Perspective. *IEEE Access* **2019**, *7*, 134968–134977. [CrossRef]
10. Li, G.; Song, D.; Xu, S.; Sun, L.; Liu, J. Kinematic-free orientation control for a deformable manipulator based on the geodesic in rotation group so (3). *IEEE Robot. Autom. Lett.* **2018**, *3*, 2432–2438. [CrossRef]
11. Jin, Y.; Wang, Y.; Chen, X.; Wang, Z.; Liu, X.; Jiang, H.; Chen, X. Model-less feedback control for soft manipulators. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Guangzhou, China, 6–8 December 2017; pp. 2916–2922.
12. Wu, J.; Wang, J.; You, Z. An overview of dynamic parameter identification of robots. *Robot. Comput. Integr. Manuf.* **2010**, *26*, 414–419. [CrossRef]
13. Aoki, T.; Nakamura, T.; Nagai, T. Learning of motor control from motor babbling. *IFAC Papers OnLine* **2016**, *49*, 154–158. [CrossRef]
14. Cursi, F.; Mylonas, G.P.; Kormushev, P. Adaptive kinematic modelling for multiobjective control of a redundant surgical robotic tool. *Robotics* **2020**, *9*, 68. [CrossRef]
15. Kormushev, P.; Calinon, S.; Caldwell, D. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics* **2013**, *2*, 122–148. [CrossRef]

16. Sands, T. Development of Deterministic Artificial Intelligence for Unmanned Underwater Vehicles (UUV). *J. Mar. Sci. Eng.* **2020**, *8*, 578. [CrossRef]
17. Saleki, A.; Fateh, M.M. Model-free control of electrically driven robot manipulators using an extended state observer. *Comput. Electr. Eng.* **2020**, *87*, 106768. [CrossRef]
18. Dromnelle, R.; Renaudo, E.; Pourcel, G.; Chatila, R.; Girard, B.; Khamassi, M. How to reduce computation time while sparing performance during robot navigation? A neuro-inspired architecture for autonomous shifting between model-based and model-free learning. In *Conference on Biomimetic and Biohybrid Systems*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 68–79.
19. AlAttar, A.; Rouillard, L.; Kormushev, P. Autonomous air-hockey playing cobot using optimal control and vision-based bayesian tracking. In Proceedings of the 20th International Conference Towards Autonomous Robotic Systems (TAROS 2019), London, UK, 3–5 July 2019.
20. Cursi, F.; Modugno, V.; Kormushev, P. Model predictive control for a tendon-driven surgical robot with safety constraints in kinematics and dynamics. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October–24 January 2020; pp. 7653–7660.
21. Hamilton, W.R. On quaternions; or on a new system of imaginaries in algebra. *Philos. Mag.* **1844**, *25*, 489–495.
22. Goldman, R. Understanding quaternions. *Graph. Model.* **2011**, *73*, 21–49. [CrossRef]
23. Clifford. Preliminary Sketch of Biquaternions. *Proc. Lond. Math. Soc.* **1871**, *s1-4*, 381–395. [CrossRef]
24. Han, D.P.; Wei, Q.; Li, Z.X. Kinematic control of free rigid bodies using dual quaternions. *Int. J. Autom. Comput.* **2008**, *5*, 319–324. [CrossRef]
25. Shoemake, K. Animating rotation with quaternion curves. In Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, San Francisco, CA, USA, 22–26 July 1985; pp. 245–254.
26. Schilling, M. Universally manipulable body models—dual quaternion representations in layered and dynamic MMCs. *Auton. Robot.* **2011**, *30*, 399. [CrossRef]
27. Rakita, D.; Mutlu, B.; Gleicher, M. RelaxedIK: Real-time Synthesis of Accurate and Feasible Robot Arm Motion. In *Robotics: Science and Systems*; ACM: Pittsburgh, PA, USA, 2018; pp. 26–30.
28. Corke, P. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB® Second, Completely Revised*; Springer: Berlin/Heidelberg, Germany, 2017; Volume 118.
29. Adorno, B.V.; Marinho, M.M. DQ Robotics: A Library for Robot Modeling and Control. *IEEE Robot. Autom. Mag.* **2020**, 1–15. [CrossRef]