robotics

Article

Kinematic-Model-Free Orientation Control for Robot Manipulation Using Locally Weighted Dual Quaternions

Ahmad AlAttar * and Petar Kormushev

Robot Intelligence Lab, Dyson School of Design Engineering, Imperial College London, London SW7 2DB, UK; p.kormushev@imperial.ac.uk

* Correspondence: a.alattar19@imperial.ac.uk

Received: 30 July 2020; Accepted: 22 September 2020; Published: 24 September 2020

Abstract: Conventional control of robotic manipulators requires prior knowledge of their kinematic structure. Model-learning controllers have the advantage of being able to control robots without requiring a complete kinematic model and work well in less structured environments. Our recently proposed Encoderless controller has shown promising ability to control a manipulator without requiring any prior kinematic model whatsoever. However, this controller is only limited to position control, leaving orientation control unsolved. The research presented in this paper extends the state-of-the-art kinematic-model-free controller to handle orientation control to manipulate a robotic arm without requiring any prior model of the robot or any joint angle information during control. This paper presents a novel method to simultaneously control the position and orientation of a robot's end effector using locally weighted dual quaternions. The proposed novel controller is also scaled up to control three-degrees-of-freedom robots.

Keywords: orientation control; model learning; adaptive control; kinematic-model-free control; locally weighted dual quaternions

1. Introduction

Since the emergence of modern robots in the early 1950s, robot control became a thriving field of research as interest grew [1]. Conventional control of robotic systems, such as manipulators, is based on the kinematics and dynamics of rigid bodies that use joint-space to task-space transformations to calculate their poses [2]. This approach leaves the controller heavily dependent on the robot's configuration, which assumes a static arrangement of joints and links. This kinematic-model-based approach has remained the prevalent control method for over half a century [3].

Over the past 70 years, several control approaches, such as model-approximating and model-learning, have emerged. Nevertheless, true model-free control remains a challenging problem. In our previous work, the Encoderless Robot Control [4] and Kinematic-Free Control [5] (which we propose to call "Kinematic-Model-Free" (KMF) Controllers) successfully controlled a robot manipulator's end effector in task space without requiring any prior kinematic model of the robot by learning local linear models. However, these controllers only managed to control the position of a two-degrees-of-freedom (DOF) planar robot, leaving orientation control unsolved.

Orientation control imposes challenges due to the discontinuities or singularities that exist in the orientation spectrum, such as the gimbal lock in Euler angles or angle wrapping that occurs after a full rotation. Without resolving these issues, the robot will misinterpret movements when crossing the point of discontinuity or can take a longer route to reach the desired target, causing the robot to behave undesirably and inefficiently. Quaternions, discovered by Hamilton around 1844 [6], are an extension to the complex number system and are able to represent rotations without suffering from discontinuities. Furthermore, dual quaternions (DQs), introduced by Clifford around 1873 [7], provide a unified representation of a rigid link's pose, both position and orientation. Linear combinations of dual quaternions are used in this research to learn a local kinematics and dynamics (what we call 'kinodynamics') model of the robot.

The contributions of this paper are the following: (1) the controller presented in this paper extends the state-of-the-art KMF controller to control both the position and orientation of a robot's end effector; (2) the robot controlled is a 3-DOF robot that allows multiple orientation solutions for given target positions, scaling up from the 2-DOF robot manipulator in our previous work; (3) this paper also presents the equations for finding relative dual quaternions and for taking a weighted sum of dual quaternions, both of which are key in developing our KMF-DQ controller.

2. Related Work

In any introductory course about robotics, the first type of controller taught is the conventional controller in which a Jacobian, which is characteristic to the robot's configuration, is populated [8]. Robots are controlled using forward and inverse kinematics and dynamics. Most of the literature in robot control assumed that the robot's Jacobian matrix is known. This assumption imposes many limitations [9], which implies that the link lengths and joint offsets must be known and static. However, this assumption breaks when the robot is soft or damaged and if the robot picks up a tool, modifying its end effector [10]. Any change in the robot's configuration will cause the controller to fail and a new model has to be generated. Thus, conventional controllers are accurate but not flexible.

Model-based controllers use the kinematic and dynamic model of the robot [11]. A well-known example is visual servoing [12]. Visual servoing uses a camera to calculate the velocity of the end effector required to reach a target. The controller sends the calculated velocity commands to a conventional velocity controller to move the robot [4]. However, as with all model-based controllers, any change to the robot's kinematics will result in undesirable behaviour and failure. Recent advances in robot control include approximating a model of the robot. Researchers in [13–15] controlled a deformable manipulator by approximating a Jacobian. Some researchers addressed the issue of controlling a dynamically undefined robot by identifying and estimating the system's dynamic parameters, which were then used alongside conventional controllers to control the robot [16].

Other controllers start with no prior knowledge about the robot's kinematic/dynamic properties and try to learn a model for controlling the robot. Examples include motor babbling [17] and reinforcement learning [18], which uses machine learning techniques to control a robot. However, all of these approaches rely on joint position feedback (through encoders) for estimating the robot's kinematic pose.

The model-learning approach presented in our preliminary work in [4,5] are KMF controllers because they start with no prior knowledge about the robot's configuration and learn a model without requiring kinematic information about the robot's joint position during control. Instead, these approaches use exteroception, similar to visual servoing to learn a local linear model that is used to drive the end effector towards a reference target. The controller starts by recording end effector displacements by sending random actuation primitives in the form of square torque waveforms using a camera. The recorded information about the joint torques and their resulting end-effector displacements are then used to calculate a new primitive that will move the end effector towards a given target. The state-of-the-art KMF controller only addressed position control of a 2-DOF planar robot, leaving orientation control and control of manipulators of higher degrees-of-freedom unsolved.

3. Mathematical Background

3.1. Quaternions

Quaternions are an extension to the complex number system used to represent rotations, first described by Hamilton around 1844 [6]. 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 along the x-,y-,z-axes. The addition and subtraction of quaternions are element-wise:

$$\mathbf{p} \pm \mathbf{q} = (s_p \pm s_q) + (v_p \pm v_q) = (p_0 \pm q_0) + (p_1 \pm q_1)\mathbf{i} + (p_2 \pm q_2)\mathbf{j} + (p_3 \pm q_3)\mathbf{k} \tag{2}$$

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

$$\mathbf{q}^* = s - \mathbf{v} = s - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k} \tag{3}$$

from which the norm of a quaternion can be calculated as $\parallel \mathbf{q} \parallel = \sqrt{\mathbf{q}^*\mathbf{q}}$. A quaternion $\mathbf{q}$ is unit when $s^2 + q_1^2 + q_2^2 + q_3^2 = 1$ or simply $\parallel \mathbf{q} \parallel = 1$. The vec_4 operator maps the elements of a quaternion $\mathbf{q}$ to a 4-dimensional vector (s, q_1, q_2, q_3). A detailed examination of quaternions can be found in [19].

3.2. Dual Quaternions

Dual quaternions, introduced by Clifford [7], are an extension to the dual number system used to represent rigid transformations by unifying rotational and translational information into a single mathematical object. A dual quaternion $\hat{\mathbf{q}}$ is 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, $\epsilon^2 = 0$. The addition and subtraction of two dual quaternions are performed separately on the real and dual parts:

$$\hat{\mathbf{q}}_1 \pm \hat{\mathbf{q}}_2 = (\mathbf{p}_1 \pm \mathbf{p}_2) + \epsilon(\mathbf{q}_1 \pm \mathbf{q}_2) \tag{5}$$

The conjugate of a dual quaternion is $\hat{\mathbf{q}}^* = \mathbf{p}^* + \epsilon\mathbf{q}^*$. The norm of a dual quaternion is defined as $\parallel \hat{\mathbf{q}} \parallel = \sqrt{\hat{\mathbf{q}}^*\hat{\mathbf{q}}}$. A dual quaternion $\hat{\mathbf{q}}$ is unit when the real part is unit and is perpendicular to the dual part, $s_p^2 + p_1^2 + p_2^2 + p_3^2 = 1$ and $s_p s_q + p_1 q_1 + p_2 q_2 + p_3 q_3 = 0$ or simply when $\parallel \hat{\mathbf{q}} \parallel = 1$ and $\langle \mathbf{p}, \mathbf{q} \rangle = 0$. The vec_8 operator maps the elements of a dual quaternion $\hat{\mathbf{q}}$ to an 8-dimensional vector $(s_p, p_1, p_2, p_3, s_q, q_1, q_2, q_3)$.

3.3. Rigid Transformation

A rigid transformation can be represented using a dual quaternion:

$$\hat{\mathbf{q}} = \mathbf{r} + \frac{\epsilon}{2}\mathbf{tr} \tag{6}$$

where $\mathbf{r}$ is the rotation quaternion and $\mathbf{t}$ is the translation quaternion [20]. From a unit dual quaternion $\mathbf{p} + \epsilon\mathbf{q}$, the rotation and translation quaternions are derived as: $\mathbf{r} = \mathbf{p}$ and $\mathbf{t} = 2\mathbf{q}\mathbf{p}^*$, where $\mathbf{r}$ is the rotation quaternion and $\mathbf{t}$ is the translation quaternion. For the translation quaternion, the scalar part is always zero, $t = 0 + \mathbf{v}_t = \mathbf{v}_t$. The dual quaternion transformation represents a pure rotation when the dual part is zero, $\mathbf{q} = 0$.

3.4. Interpolation

Interpolation between two 3D rotations represented by two quaternions is achieved using Spherical Linear Interpolation (SLERP) which, was introduced by Shoemake in 1985 [21]. This method of interpolation is popular because it has the following desirable properties: constant speed, shortest path, and coordinate system invariant [22]. Let $\Phi(\mathbf{p}, \mathbf{q}, t)$ denote an interpolation between $\mathbf{p}$ and $\mathbf{q}$, where $t \in [0, 1]$. The interpolated quaternion is then

$$\mathbf{m} = \Phi(\mathbf{p}, \mathbf{q}, t) = \mathbf{p}(\mathbf{p}^*\mathbf{q})^t \tag{7}$$

Screw Linear Interpolation (ScLERP) is a generalization of SLERP that extends it to interpolate between dual quaternions and is defined similarly,

$$\hat{\mathbf{m}} = \Phi(\hat{\mathbf{p}}, \hat{\mathbf{q}}, t) = \hat{\mathbf{p}}(\hat{\mathbf{p}}^*\hat{\mathbf{q}})^t \tag{8}$$

where $t \in [0, 1]$.

3.5. Relative Dual Quaternions

The relative rotational displacement, $\mathbf{b}_{rel}$, between two quaternions, $\mathbf{p}$ and $\mathbf{q}$, can be easily found as $\mathbf{b}_{rel} = \mathbf{p}^*\mathbf{q}$. Finding the relative dual quaternion, $\hat{\mathbf{b}}_{rel}$, between two dual quaternions, $\hat{\mathbf{p}}$ and $\hat{\mathbf{q}}$, is not as straightforward. The dual quaternions are split into their rotational and translational quaternions and then recombined after calculating the relative rotation and translation as follows:

$$DQrel(\hat{\mathbf{p}},\, \hat{\mathbf{q}}) = \hat{\mathbf{b}}_{rel} = \mathbf{r}_1^*\mathbf{r}_2 + \frac{\epsilon}{2}(\mathbf{t}_1^* + \mathbf{t}_2)\mathbf{r}_1^*\mathbf{r}_2 \tag{9}$$

Note that $\hat{\mathbf{b}}_{rel}$ must be a unit dual quaternion and normalized if not.

3.6. Dual Quaternion Distance

The KMF-DQ approach requires distance metrics between two dual quaternions, which will provide a measure of how close the end effector is to the target pose that reduces to zero as the end effector approaches the target. The *vec*8 and *DQrel* functions are used to find the distance between two dual quaternions as follows,

$$DQdist(\hat{\mathbf{q}}_1,\, \hat{\mathbf{q}}_2) = \parallel vec8(\hat{\mathbf{0}}) - vec8(DQrel(\hat{\mathbf{q}}_1,\, \hat{\mathbf{q}}_2)) \parallel \tag{10}$$

where $\hat{\mathbf{0}}$ is the null dual quaternion. This measure of distance satisfies the following three properties:

$$DQdist(\hat{\mathbf{p}},\, \hat{\mathbf{q}}) = 0 \iff \hat{\mathbf{p}} = \hat{\mathbf{q}}$$

$$DQdist(\hat{\mathbf{p}},\, \hat{\mathbf{q}}) = DQdist(\hat{\mathbf{q}},\, \hat{\mathbf{p}})$$

$$DQdist(\hat{\mathbf{p}},\, \hat{\mathbf{q}}) \leq DQdist(\hat{\mathbf{p}},\, \hat{\mathbf{z}}) + DQdist(\hat{\mathbf{z}},\, \hat{\mathbf{q}})$$

Although unifying both translational and rotational distances into a single distance metrics can be an ambiguous measure of distance between poses, we can still see that $DQdist \rightarrow 0$ as $\hat{\mathbf{q}}_1 \rightarrow \hat{\mathbf{q}}_2$, which is suitable for our control purposes; to know when the end effector has reached the target and to end execution.

3.7. Dual Quaternion Scaling

Vectors are scaled by multiplying them by a constant. To develop a similar scaling of dual quaternions by some factor c, they must be broken into their translational and rotational quaternions and then recombined after scaling as follows:

$$DQscale(\hat{\mathbf{q}}, c) = \hat{\mathbf{s}}_{scale} = \mathbf{r}^c + \frac{\epsilon c}{2}\mathbf{t}\mathbf{r}^c \tag{11}$$

where the exponential of some quaternion $\mathbf{q}^c = exp(ln(\mathbf{q})c)$. The definitions of exp and ln can be found in [23]. Note that $\hat{\mathbf{s}}_{scale}$ must be a unit dual quaternion and normalized if not. For example, $c = 2$ would result in a dual quaternion composed of twice the rotation and translation.

3.8. Dual Quaternion Linear Combination

To find a weighted sum of dual quaternions $\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \ldots, \hat{\mathbf{q}}_n$ using weights $w_1, w_2, \ldots, w_n$, we decompose the dual quaternions into their rotational quaternions, $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n$, and translational quaternions, $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$. The weighted sum of the rotational quaternions is as follows:

$$\mathbf{r}' = \mathbf{r}_1^{w_1}\mathbf{r}_2^{w_2}\ldots\mathbf{r}_n^{w_n} \tag{12}$$

which can be written more compactly:

$$\mathbf{r}' = \prod_{i=1}^{n}\mathbf{r}_i^{w_i}$$

The weighted sum of the translational quaternions is calculated as follows:

$$\mathbf{t}' = w_1\mathbf{t}_1 + w_2\mathbf{t}_2 + \cdots + w_n\mathbf{t}_n \tag{13}$$

or more compactly:

$$\mathbf{t}' = \sum_{i=1}^{n} w_i\mathbf{t}_i$$

As a result, the weighted sum of the dual quaternions is computed as:

$$DQLC(\hat{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_n, w_1, \ldots, w_n) = \mathbf{r}' + \frac{\epsilon}{2}\mathbf{t}'\mathbf{r}' \tag{14}$$

$$= \prod_{i=1}^{n}\mathbf{r}_i^{w_i} + \frac{\epsilon}{2}\sum_{i=1}^{n} w_i\mathbf{t}_i\prod_{i=1}^{n}\mathbf{r}_i^{w_i} \tag{15}$$

3.9. Dual Quaternion Regression

Regression can be seen as the inverse problem of calculating the weighted sum. The functions $DQLC$ and $DQdist$ are used to find the regression coefficients, $\mathbf{w} = [w_1, w_2, \ldots, w_n]$, of the dual quaternions $\hat{\mathbf{Q}} = [\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \ldots, \hat{\mathbf{q}}_n]$ that yield some desired dual quaternion $\hat{\mathbf{q}}_d$:

$$DQreg(\hat{\mathbf{q}}_d, \hat{\mathbf{Q}}) = \min_{\mathbf{w}} DQdist(\hat{\mathbf{q}}_d, \hat{\mathbf{q}}_r) \tag{16}$$

where

$$\hat{\mathbf{q}}_r = DQLC(\hat{\mathbf{Q}}, \mathbf{w}) \tag{17}$$

which can be optimized using some optimization algorithm. In our implementation of the KMF-DQ, MATLAB's FMINCON optimization algorithm was used.

4. Proposed Approach

A high-level flowchart of the KMF-DQ controller is shown in Figure 1. At the core of the idea lies the assumption that it is possible to obtain information about the local kinodynamics of the robotic manipulator, which is gathered by generating pseudo-random actuation control signals and observing their effects on the robot's end effector. The torque actuation signals are square waves, as shown in Figure 2. The actuation primitive produces a torque signal $\tau(t)$ sent to each actuator in the robot and is defined as follows:

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

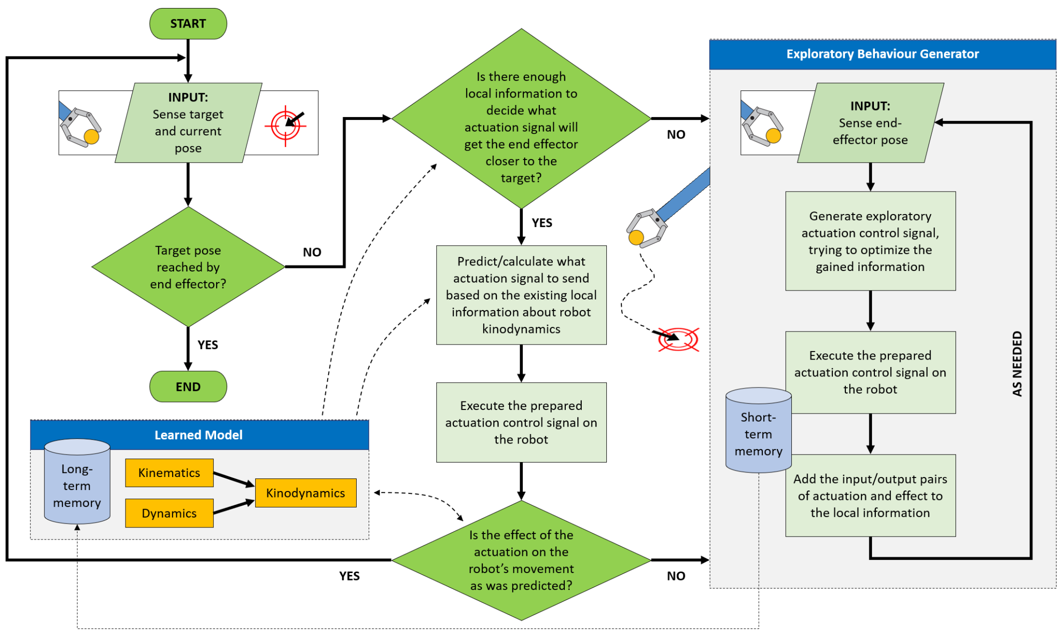
START
INPUT: Sense target and current pose
Target pose reached by end effector?
NO
YES
END
Is there enough local information to decide what actuation signal will get the end effector closer to the target?
NO
YES
Predict/calculate what actuation signal to send based on the existing local information about robot kinodynamics
Execute the prepared actuation control signal on the robot
Is the effect of the actuation on the robot's movement as was predicted?
YES
NO
Learned Model
Long-term memory
Kinematics
Dynamics
Kinodynamics
Exploratory Behaviour Generator
INPUT: Sense end-effector pose
Generate exploratory actuation control signal, trying to optimize the gained information
Execute the prepared actuation control signal on the robot
Add the input/output pairs of actuation and effect to the local information
Short-term memory
AS NEEDED

Figure 1. A high-level flowchart the Kinematic-Model-Free dual quaternions (KMF-DQ) approach.

As shown in the flowchart, if the end effector reached the target, the controller ends the execution. Otherwise, the end effector is to reach a set of intermediate targets $\hat{\mathbf{q}}_i$ that lead to the final desired target pose $\hat{\mathbf{q}}_t$ in discrete steps. These intermediate targets are found using ScLERP interpolation between the end effector's current pose, $\hat{\mathbf{q}}_c$, and the desired target pose, $\hat{\mathbf{q}}_t$. At each step, the controller tries to estimate an actuation primitive, $\mathbf{b}$, to move the end effector towards the desired goal pose:

$$\mathbf{b} = [\tau_p^1(\hat{p})\ \tau_p^2(\hat{p})\ \tau_p^3(\hat{p})]^T \tag{19}$$

where $\tau_p^j(\hat{p})$ is the magnitude of the actuation primitive of actuator j. After each actuation, the end effector displacement is recorded as an observation, which is calculated using the *DQrel* function. The nearest k observation dual quaternions form the observation vector:

$$\Psi = [\hat{\mathbf{o}}_1, \ldots, \hat{\mathbf{o}}_k] \tag{20}$$

These dual quaternions contain the rotational and translational displacements caused by the executed actuation primitives. The actuation primitives corresponding to these observations form the actuation matrix:

$$\mathbf{A} = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \ldots & \tau_p^1(p_k) \\ \tau_p^2(p_1) & \tau_p^2(p_2) & \ldots & \tau_p^2(p_k) \\ \tau_p^3(p_1) & \tau_p^3(p_2) & \ldots & \tau_p^3(p_k) \end{bmatrix} \tag{21}$$

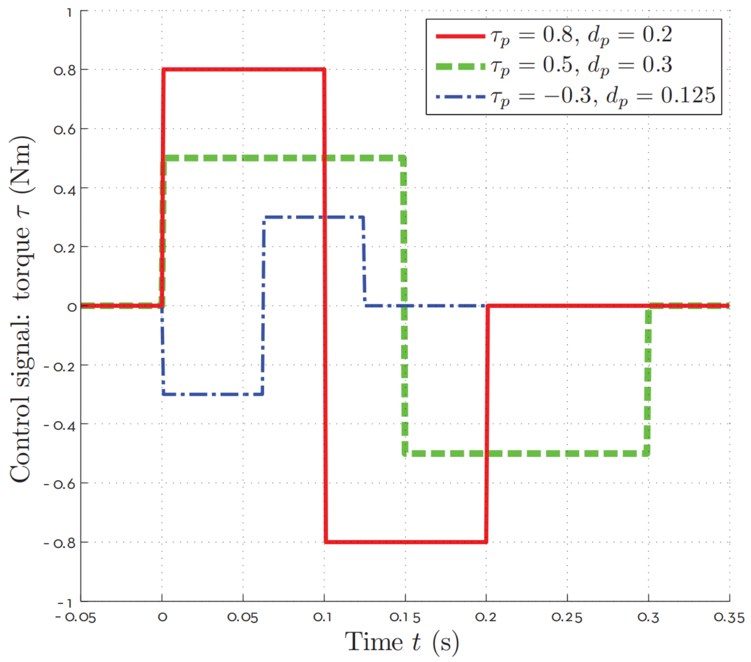

Figure 2. Example actuation primitives used by the proposed KMF-DQ approach. Each of the primitives shown has a different duration d_p and magnitude τ_p [4] (© 2015 IEEE).

The desired actuation primitive, $\mathbf{b}$, is assumed to be a linear combination of the k nearest actuation primitives:

$$\mathbf{b} = \mathbf{A}\mathbf{w} \tag{22}$$

where $\mathbf{w} = [w_1, w_2, \ldots, w_k]^T$ is a set of unknown weights. These weights are found by solving the local regression of the observation vector:

$$DQreg(\hat{\mathbf{q}}_i,\ \Psi) = \min_{\mathbf{w}} DQdist(\hat{\mathbf{q}}_i,\ DQLC(\Psi,\ \mathbf{w})) \tag{23}$$

With the weights, the desired actuation is then computed using Equation (13). After the completion of each finite actuation, the resulting effect on the end effector's state is compared with the anticipated effect. If the difference is significant, this means that the local linear model approximation of the local kinodynamics was not known precisely enough, which will trigger a new exploratory phase as shown in Figure 1. The controller escapes the exploratory phase once it has done a set number of explorations. Lastly, the algorithm terminates when $DQdist < \sigma$, where σ is the termination accuracy.

We stress that this approach does not require any joint angle information or prior knowledge about the robot's kinematics and dynamics. As such, this controller is able to adapt to kinematic and dynamic changes in the robot such as changes to robot links' sizes and extensions to the end effector. However, since the KMF-DQ controller is a discrete controller and is not based on an accurate model, the KMF-DQ controller can be used to control the robot where flexibility is favoured over precision and speed. As with other complex models such as [24,25], the convergence of the KMF-DQ controller is not guaranteed.

5. Simulation and Results

A dynamic simulation of a 3-DOF robot was performed using MATLAB's Robotics Toolbox [26] and DQ Robotics Toolbox [26]. The kinematic and dynamic specifications of the robot used for the experiments are shown in Table 1.

Table 1. Robot specifications.

Link	Length (m)	Mass (kg)
L_1	0.50	1.00
L_2	0.50	1.00
L_3	0.50	1.00

The information in Table 1 is not given to the KMF-DQ controller. The controller was implemented as described in Section 4 using $k = 4$ nearest neighbours comprising of actuation primitives and their corresponding observations. Since this experiment was conducted in simulation, there was no need to use a camera to obtain the end-effector's state. Instead, this was achieved internally from the simulator. We note that gravity is neglected in the simulation to keep the focus on simultaneous position and orientation control, deferring gravity compensation until future research.

The experiment involves a set of reaching tasks. The controller must move the robot's end effector to four target poses, drawing a polygon in 3D space. The position and orientation of the four target poses are listed in Table 2. The starting position of the robot was (−0.85059, 1.2148, −0.12941) and orientation was (−1.5708, 0.9599, 2.8798).

Table 2. Target poses.

Target	Position	Orientation
1	(−0.6827, 0.9525, 0.3565)	(−1.5710, 1.3350, −2.3476)
2	(−0.1410, 1.1470, 0.3830)	(1.5706, 1.3089, 0.8729)
3	(0.1216, 1.4198, 0.1710)	(1.5709, 1.3090, 0.3489)
4	(−0.8506, 1.2148, −0.1294)	(−1.5709, 0.9599, 2.8797)

The orientations are specified here in Euler XYZ angles for simplicity. The poses were selected to be within the robot's reachable workspace. The results are shown in Figure 3 in all three orthogonal views. The KMF-DQ controller generates intermediate targets to follow the yellow trajectory. The robot successfully reaches all four target poses, drawing the polygon in 3D space. These results prove that the local kinodynamic information of the robot is sufficient to create a local model that can be used to actuate the end effector towards a given target. Note that the specified target poses must lie within the robot's reachable workspace, otherwise a higher termination tolerance, σ, must be set.

To demonstrate the actuation primitives generated by the controller, Figure 4 shows the sequence of actuated primitives of the first reaching task (moving towards target 1). Since the KMF-DQ is a discrete controller, the figure shows the discrete primitives sent to the actuators. We note that as the end effector reaches the target, lower torques are sent to the actuators. The joint position of the three joints corresponding to the first reaching task is shown in Figure 5 to demonstrate the resulting displacements in the joints.

We can see that the joint angles of the three joints vary smoothly with no sudden jerks. Furthermore, Figure 6 shows the pose distance of the end effector to the targets using the *DQdist* function. We note that the pose distance approaches zero as the end effector approaches the targets.

Although not guaranteed, the simulation demonstrated convergence to the desired targets. The controller's robustness was measured by running the simulation 100 times, each with a random starting robot configuration and a randomly placed target pose within the robot's reachable workspace that is achievable within 100 steps. The target poses lie within the robot's achievable configuration space. It was found that 87% of the time, the controller successfully performed the reaching task. In the remaining 13%, the controller did not converge to solutions that actuate the robot towards the targets. These failures can occur because the optimizer gets stuck in a local minimum, which will yield inaccurate results for the weights and thus the calculated actuation primitives.

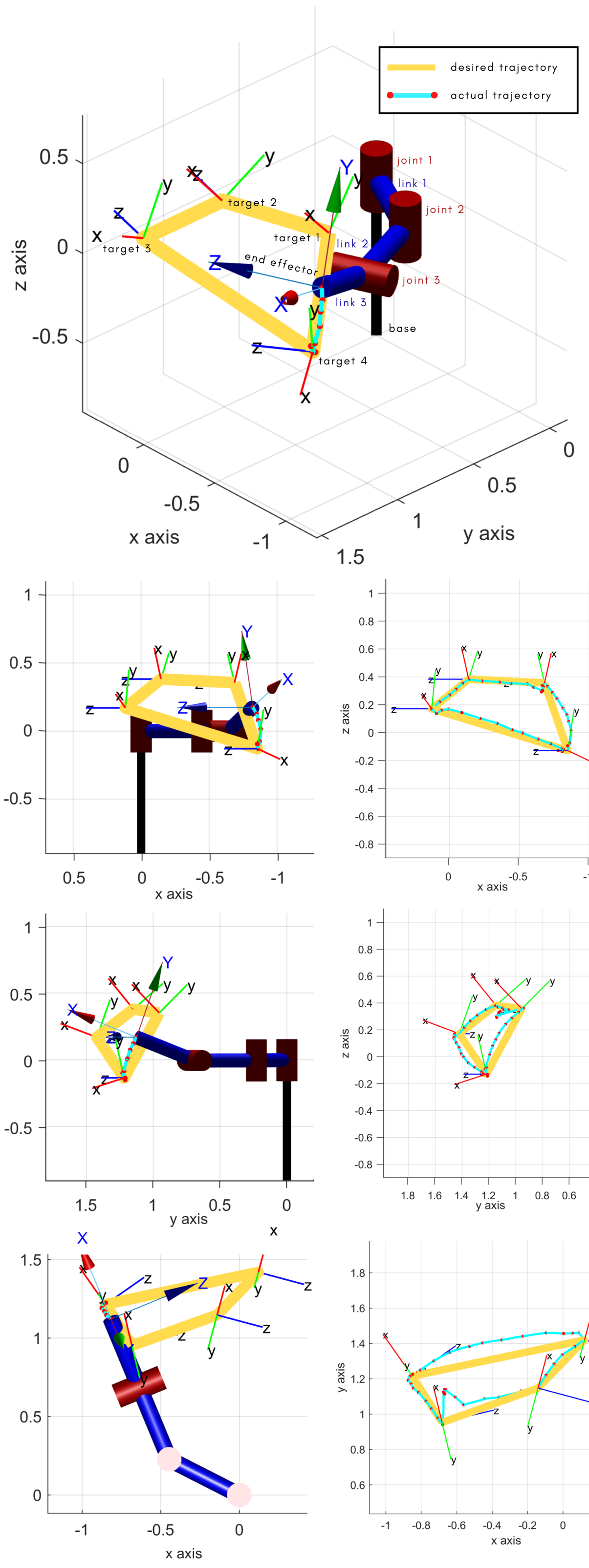

Figure 3. The KMF-DQ controller actuates the robot's joints to perform a set of reaching tasks to draw out a polygon in 3D space with each vertex having a pre-specified pose within the robot's reachable workspace. The bottom left three plots show the orthogonal views of the robot performing the experiment. The bottom right three plots show the final resulting trajectory of the end effector. The robot moves towards the next target only after reaching the current target.

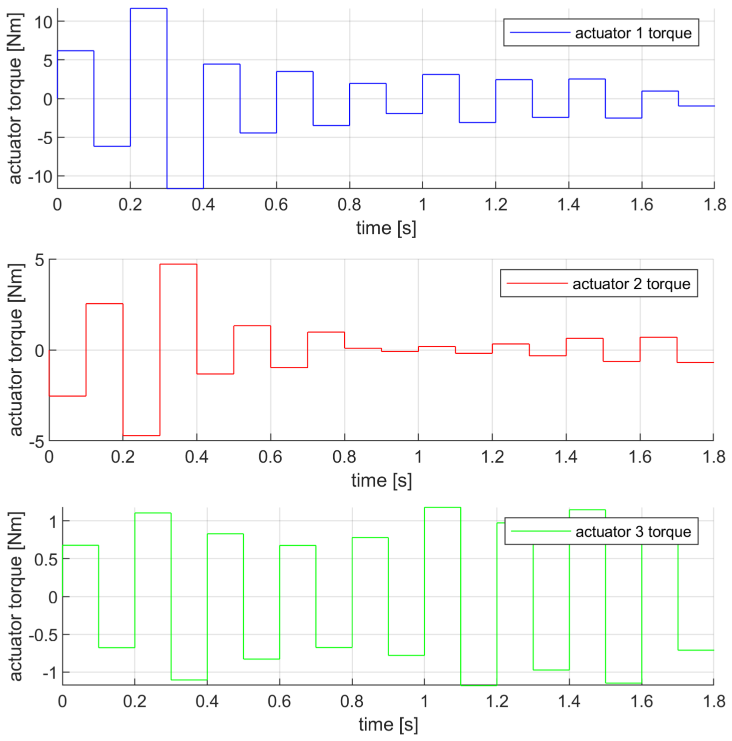

Figure 4. The control signals sent to the actuators generated by the controller. These actuation signals correspond to the first reaching task where the end effector moves towards target 1.

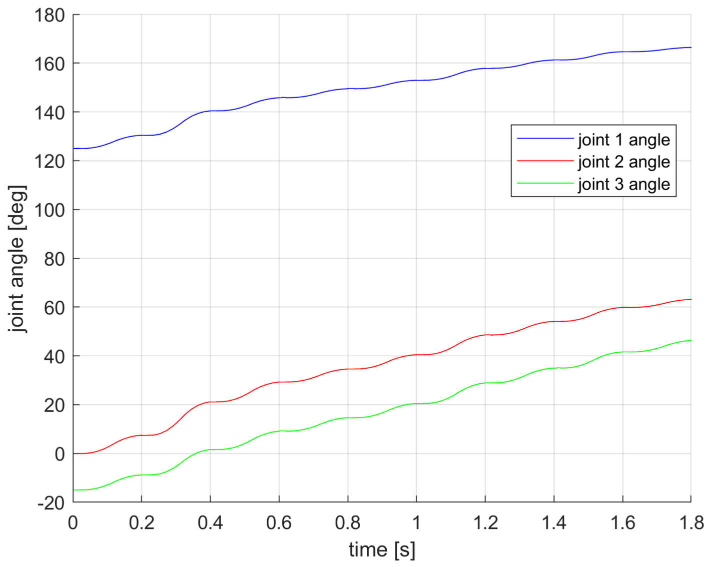

Figure 5. The joint positions of the three joints. These data correspond to the first reaching task where the end effector moves towards target 1.

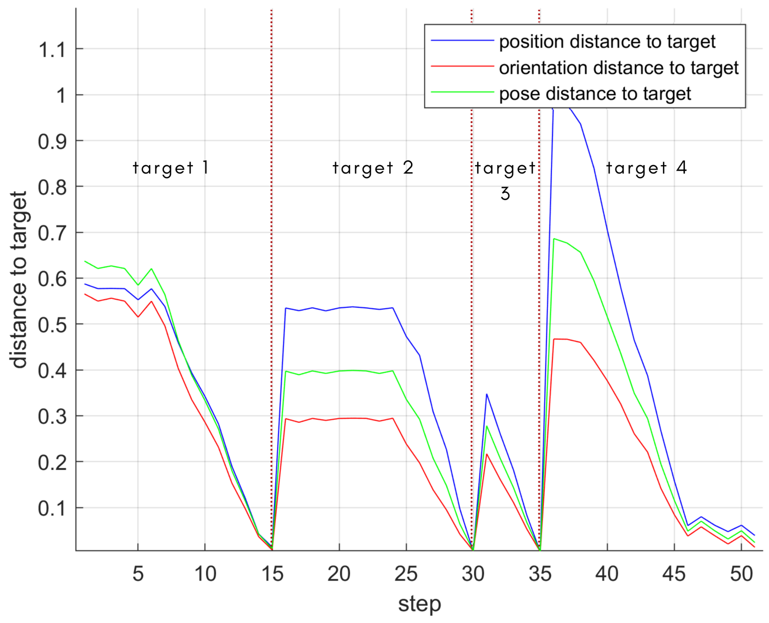

Figure 6. Distance of the end-effector pose to the target pose for each reaching task. The pose distance approaches zero as the end effector reaches the targets.

6. Conclusions

This paper presented a novel model-free controller for robot manipulation using locally weighted combinations of dual quaternions. The approach does not require any joint angle information or prior knowledge about the robot's kinematics and dynamics. The controller works by executing random actuation primitives and perceiving their resulting effects on the end effector. These pieces of information are then used to build a local kinodynamic model. An actuation primitive is calculated using dual quaternion regression, which is executed on the robot to move the end effector towards a given target.

We presented the dual quaternion distance, relative dual quaternion, and dual quaternion regression functions used for building a local kinodynamic model. Simulation experiments were conducted in which a manipulator performed a set of reaching tasks. The simulation results show that this novel approach can reach a set of reference poses, successfully performing simultaneous position and orientation control. Furthermore, the robustness of the approach was tested by running the simulation 100 times with random target poses that lie within the robot's reachable workspace, which are achievable by some configuration of the robot. The controller managed to successfully reach the target poses 87% of the time within 100 steps.

The proposed approach looks promising and can be used in applications where simultaneous translational and rotational reaching is required. This approach can be used for new robot designs where a model is not present and where flexibility is favoured over precision and speed. With regards to future work, scalability to higher-degrees-of-freedom robot manipulators, continuous control, and practical real-world implementation issues such as joint limits and friction remain open problems that will be investigated in future research. Also, key issues that hindered the immediate experimentation of the proposed controller to physical experiments, such as gravity compensation and friction compensation, must be tackled before the experiments are performed since the current controller does not take into account these physical phenomena.

Author Contributions: A.A. developed the theory, performed the simulation experiments, and wrote 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 research received no external funding.

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

References

1. Niku, S.B. *Introduction to Robotics: Analysis, Control, Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2020.
2. Cho, H. *Opto-Mechatronic Systems Handbook: Techniques and Applications*; CRC Press: Boca Raton, FL, USA, 2002.
3. 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.
4. 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.
5. Kormushev, P.; Demiris, Y.; Caldwell, D.G. Kinematic-free position control of a 2-dof planar robot arm. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 5518–5525.
6. Hamilton, W.R. On Quaternions; or on a new System of Imaginaries in Algebra. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **1844**, *33*, 58–60. [CrossRef]
7. Clifford, W.K. Preliminary Sketch of Biquaternions. *Proc. Lond. Math. Soc.* **1871**, *s1–s4*, 381–395. [CrossRef]
8. 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.
9. 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]
10. Cheah, C.C.; Liu, C.; Slotine, J.J.E. Adaptive tracking control for robots with unknown kinematic and dynamic properties. *Int. J. Robot. Res.* **2006**, *25*, 283–296. [CrossRef]
11. Asensio, J.; Montano, L. A kinematic and dynamic model-based motion controller for mobile robots. *IFAC Proc. Vol.* **2002**, *35*, 427–432. [CrossRef]
12. Huang, Y.; Su, J. Visual Servoing of Nonholonomic Mobile Robots: A Review and a Novel Perspective. *IEEE Access* **2019**, *7*, 134968–134977. [CrossRef]
13. Li, G.; Sun, L.; Xu, S.; Song, D.; Liu, J. A hybrid model and kinematic-free control framework for a low-cost deformable manipulator using in home service. In Proceedings of the 2016 IEEE International Conference on Automation Science and Engineering (CASE), Fort Worth, TX, USA, 21–25 August 2016; pp. 1002–1007.
14. Li, G.; Xu, S.; Sun, L.; Liu, J. Kinematic-free position control for a deformable manipulator. In Proceedings of the 2016 35th Chinese Control Conference (CCC), Chengdu, China, 27–29 July 2016; pp. 10302–10307.
15. 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]
16. Wu, J.; Wang, J.; You, Z. An overview of dynamic parameter identification of robots. *Robot. Comput.-Integr. Manuf.* **2010**, *26*, 414–419. [CrossRef]
17. Aoki, T.; Nakamura, T.; Nagai, T. Learning of motor control from motor babbling. *IFAC-PapersOnLine* **2016**, *49*, 154–158. [CrossRef]
18. Kormushev, P.; Calinon, S.; Caldwell, D. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics* **2013**, *2*, 122–148. [CrossRef]
19. Goldman, R. Understanding quaternions. *Graph. Model.* **2011**, *73*, 21–49. [CrossRef]
20. 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]
21. Shoemake, K. Animating rotation with quaternion curves. In Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques: San Francisco, CA, USA, July 1985; pp. 245–254.
22. Kavan, L.; Collins, S.; O'Sullivan, C.; Zara, J. *Dual Quaternions for Rigid Transformation Blending*; Tech. Rep. TCD-CS-2006-46; Trinity College Dublin: Dublin, Ireland, 2006.
23. Särkkä, S. *Notes on Quaternions*; Internal Technical Document; Helsinki University of Technology: Espoo, Findland, 2007.
24. Schilling, M. Universally manipulable body models—Dual quaternion representations in layered and dynamic MMCs. *Auton. Robot.* **2011**, *30*, 399. [CrossRef]

25. Rakita, D.; Mutlu, B.; Gleicher, M. RelaxedIK: Real-time Synthesis of Accurate and Feasible Robot Arm Motion. In Proceedings of the Robotics: Science and Systems, Pittsburgh, PA, USA, 26–30 June 2018; pp. 26–30.
26. Corke, P. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB® Second, Completely Revised*; Springer: Berlin/Heidelberg, Germany, 2017; Volume 118.