Mechanism and Machine Theory 107 (2017) 27–36

Contents lists available at ScienceDirect

Mechanism and Machine Theory

journal homepage: www.elsevier.com/locate/mechmachtheory

A dual quaternion solution to the forward kinematics of a class of six-DOF parallel robots with full or reductant actuation

XiaoLong Yang*, HongTao Wu, Yao Li, Bai Chen

College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics & Astronautics, 29 Yudao Street, Qinhuai District, Nanjing, China

ARTICLE INFO

Keywords:
Forward kinematics
Dual quaternion
Parallel robot
8-UPS
8-PUS

ABSTRACT

The forward kinematics is the basis of the design and control of the parallel robots. This paper aims to provide an efficient solution to the forward kinematics of a class of six-degrees-of-freedom parallel robots for real-time applications. With a unit dual quaternion used as the generalized coordinates of the robot system, the forward kinematic equations are derived to be a set of quadratic ones. An efficient algorithm is proposed to get the actual solution to them. The convergence and singularity problems of the new algorithm have been discussed. We have provided a convergence strategy and revealed the internal relation of the singularity with that of the parallel robot, proving the feasibility of the algorithm and giving the working condition in the practical applications. The new algorithm have been compared to the Newton's method for an 8-UPS parallel robot, resulting in the time consumptions of 0.2187 milliseconds and 14.25 milliseconds respectively. And then we perform a simulation of the state-feedback control for an 8-PUS parallel robot. The two examples present the applications of the new algorithm and demonstrate its validity and efficiency.

1. Introduction

Parallel robots, sometimes called parallel kinematic machines, are a kind of closed-loop mechanisms presenting very good performances in terms of high ratio of stiffness to mass, strong dynamic performances and high accuracy, which make themselves able to manipulate large loads and accomplish precise tasks. They have been gradually applied to the motion simulation systems, micro-displacement positioning devices, visualized haptic devices, industrial and medical robots, telescopes, multi-axis vibration isolation etc. [1]. A six-degrees-of-freedom (six-DOF) parallel robot has a mobile platform, also named as an end effector, and a fixed base connected by six or more kinematic chains, which can be RRPS, RPRS, PRRS, RRRS, etc. By switching the order of the joint, or replace R(P) by P(R), various six-DOF parallel robots can be generated, e.g., 6-UPS (Gough [2]), 6-RUS (Hunt [3]), 6-PUS (Merlet [4]) and 8-UPS (Allen [5]). Such robots are often called Stewart platforms, because this kind of parallel robots have attracted the researchers' attentions since the publication of [6] by Stewart in 1965. A comprehensive survey of the kinematics, singularity, work space and dexterity, dynamics and control, manufacturing and assembly are being carried out.

Although the advantages of parallel robots makes them an ideal solution to the high-speed and high-precision applications, it is real difficult to find the stable solution of the kinematics with an infinitesimal calculation error and time consumption due to its high degree of coupling. The inverse kinematics problem is defined as to find a set of actuated joint inputs on the basis of the desired pose (position and orientation) of the mobile platform. The solution to this problem is indeed not complex and usually can be computed

* Corresponding author.
E-mail address: yang_xiaolong@nuaa.edu.cn (X. Yang).

http://dx.doi.org/10.1016/j.mechmachtheory.2016.08.003
Received 21 March 2016; Received in revised form 7 July 2016; Accepted 2 August 2016
Available online 23 September 2016

analytically. However, on the other hand, there is no closed-form solution for the forward kinematics problem [1], which is defined as to find the pose of the mobile platform while the joint inputs are given. Solving the forward kinematics plays a most important and significant role in the feedback control and the workspace analysis.

There exist two categories of methods to solve the forward kinematics problem: analytical and numerical methods. In the aspect of analytical methods, the researchers formulated the forward kinematics with a set of nonlinear equations and struggled to find all the possible roots by means of algebraic elimination, continuation, interval analysis, etc. [7–10]. They have made some progress and those solutions are called assembly modes of the parallel robots. Unfortunately, the pose of the mobile platform has not been expressed in an explicit form so far [7–13]. Moreover, finding all possible solutions is not completely solving the forward kinematics problem. We still need some schemes to determine a unique actual pose among them, which is required for practical applications. In some cases, the use of auxiliary sensors is one commonly adopted scheme to further lead to a unique solution [14,15]. But such an approach is limited due to the expensive price and measurement errors. In the aspect of numerical methods, the unique actual solution can be obtained via a widely used algorithm named as Newton's method, also called Newton−Raphson method. If the initial guess is located in the domain of convergence, the exact solution will be obtained [16–18]. Some scholars use neural network algorithms to obtain the initial guess of Newton's method, to ensure the stability [19]. The unique solution also can be obtained by optimization algorithms such as genetic algorithms and neural network algorithms [20–25]. However, these algorithm take a long time, making them not very suitable for real-time applications. Thus, an efficient algorithm for the forward kinematics is challenging but essential on the field of the parallel robots.

This paper aims to provide an efficient solution to the forward kinematics of a class of six-DOF parallel robots for the real-time state-feedback control. The passive joints of the parallel robot discussed in this paper are designed with the compound hinges, whose rotation axes are intersecting at one point. The robot can be designed with full or reductant actuation. Inspired by Husty [26], WAMPLER [27], etc., who pioneered the use of dual quaternions in the forward kinematics of the Stewart platform, we now use dual quaternions to solve the forward kinematics numerically. Yang, etc. [28] proposed a fast algorithm based on quaternions for the forward kinematics of the 6-UPS parallel robot. The singularity problem had not been solved so that the Jacobian matrix of the algorithm had to be inspected in each iteration to check whether it was close to the singularity, making the time consumption of per iteration increased. Zhou, etc. [29], used dual quaternions to solve the forward kinematics of a 6−6R parallel robot numerically, where DH parameters are used to formulate the kinematic chains. The algorithm may be not efficient enough since the kinematic equations have lots of unknowns and the singularity problem of the algorithm has not been solved either, which limit the use in real-time applications. Based on these researches, we propose an efficient numerical solution to the forward kinematics of a class of six-DOF parallel robots, including fully actuated and redundant actuated ones, and solve the convergence and singularity problems. The method to model the forward kinematics of the six-DOF parallel robots based on dual quaternions will be later introduced. The forward kinematic equations are then derived to be a system of quadratic equations. We propose an efficient algorithm and discuss the convergence and singularity problems. Finally, we use two examples to show how the algorithm should be used and demonstrate its validity and efficiency.

2. The dual quaternion representation of the forward kinematics for the six-DOF parallel robots

The six-DOF parallel robot this paper discussed is composed of a mobile platform, six or more kinematic chains and the base, who are connected mutually by lower pairs. The mobile platform can perform any rotation and translation actuated by the independent single-axis joints. Due to the simplicity of the inverse kinematics of the parallel robot [1], the joint variables can be expressed explicitly in terms of the pose parameters of the mobile platform, which are often set as the generalized coordinates of the system. Here, we denote them as a vector $\mathbf{x}$. The actuated joints varibles are denoted as a vector $\boldsymbol{\theta}$. It is possible to establish the kinematic closed-loop equations [30] whose general form is

$$\mathbf{E}(\boldsymbol{\theta}, \mathbf{x}) = \mathbf{0}. \tag{1}$$

The forward kinematics is to solve $\mathbf{x}$ in Eq. (1) when $\boldsymbol{\theta}$ is given. To represent the vectors of Eq. (1) in terms of components, we build two Cartesian coordinate frames. One is a fixed frame attached to the base and the other is a mobile frame attached to the mobile platform. $\mathbf{x}$ is usually parameterized by the position vector $\mathbf{p}$ and the rotation matrix $\mathbf{R} \in \mathbb{SO}(3)$. Consequently Eq. (1) can be written as

$$\mathbf{E}_i(\boldsymbol{\theta}) = \mathbf{p} + \mathbf{R}\mathbf{a}_i \quad (i = 1, 2, \cdots, n) \tag{2}$$

where $\boldsymbol{P}$ is the position vector in the fixed frame of the origin of the mobile frame, $\boldsymbol{a}_i$ is the position vector in the mobile frame of the connection center of each kinematic chain and the mobile platform, and n is the number of the kinematic chains which is equal to or greater than 6. $n = 6$ stands for the fully actuated robot, and $n > 6$ stands for the redundant actuated robot.

In this paper, we will yet use a unit dual quaternion to parameterize $\mathbf{x}$. The set of dual quaternions is a Clifford algebra and a dual quaternion is an ordered pair of quaternions, in which the real part is used to represent the rotation and the dual part is used to represent the translation. We now need to transform $\mathbf{p}$ and $\mathbf{R}$ into the unit dual quaternion.

The rotation of the mobile platform can be represented by a unique rotation matrix $\mathbf{R}$, which is a linear operate determined by the rotation axis, defined by a unit vector $\boldsymbol{n}$ and its amplitude α. The rotation is denoted as the operator $\mathbf{R}(\alpha, \mathbf{n})$.

We firstly parameterize $\mathbf{R}$ by the real part of the unit dual quaternion. To describe the rotation of any vector $\mathbf{y} \in \mathbb{R}^3$ about the axis $\mathbf{n}$, we decompose $\mathbf{y}$ into two parts, one of which is parallel to $\mathbf{n}$ and the other is perpendicular to $\mathbf{n}$ as follows:

$$\mathbf{y} = (\mathbf{y}\cdot\mathbf{n})\mathbf{n} + (\mathbf{n}\times\mathbf{y})\times\mathbf{n}. \tag{3}$$

Due to the orthogonality of $\mathbf{n}$, $\mathbf{n}\times\mathbf{y}$ and $(\mathbf{n}\times\mathbf{y})\times\mathbf{n}$ and $\mathbf{Rn} = \mathbf{n}$, we get

$$\mathbf{Ry} = (\mathbf{y}\cdot\mathbf{n})\mathbf{n} + \mathbf{R}[(\mathbf{n}\times\mathbf{y})\times\mathbf{n}] = (\mathbf{y}\cdot\mathbf{n})\mathbf{n} + (\mathbf{n}\times\mathbf{y})\sin\alpha + [(\mathbf{n}\times\mathbf{y})\times\mathbf{n}]\cos\alpha. \tag{4}$$

Eq.(4) can be written as the well-known Euler–Rodrigues formula:

$$\mathbf{Ry} = \mathbf{y} + (\mathbf{n}\times\mathbf{y})\sin\alpha + [\mathbf{n}\times(\mathbf{n}\times\mathbf{y})](1 - \cos\alpha). \tag{5}$$

A quaternion is a linear combinations of the basis elements 1, i, j and k. the quaternion product for i, j and k is often written as

$$\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1. \tag{6}$$

A convenient way to work with the quaternion product is to write a quaternion as the sum of a three dimensional vector and a scalar. Let us consider the set $\mathbb{R}^3\times\mathbb{R}$, whose element is a pair of a vector $\mathbf{q}$ and a scalar q_0, denoted as $q = q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k} + q_0 = (\mathbf{q}\quad q_0) = (q_1\quad q_2\quad q_3\quad q_0)$. The law of quaternion product

$$(q, r)\to qr = (q_0\mathbf{r} + r_0\boldsymbol{q} + \boldsymbol{q}\times\mathbf{r}\quad q_0r_0 - \boldsymbol{q}\cdot\mathbf{r}) \tag{7}$$

is bilinear of q and r. It is easy to verify that the composition operation is associative but not commutative since it involves a cross product, which makes $\mathbb{R}^3\times\mathbb{R}$ an associative algebra. The set $\mathbb{R}^3\times\mathbb{R}$ is designated as $\mathbb{H}$, whose elements are called quaternions. The components $\mathbf{q}$ and q_0 are considered as the imaginary part $\mathcal{Im}(q)$ and the real part $\mathcal{Re}(q)$. A vector in $\mathbb{R}^3$ is exactly a quaternion with zero real part, which conforms to all the theorems of quaternions.

The quaternion $q^* = (-\mathbf{q},\quad q_0)$ is called the conjugate of $q = (\mathbf{q},\quad q_0)$. The mapping $q\to q^*$ is an automorphism of the vector space $\mathbb{H}$, but antiautomorphism of the algebra structure as $(qr)^* = r^*q^*$. Since

$$qq^* = q^*q = (\|\mathcal{Re}(q)\|^2 + \|\mathcal{Im}(q)\|^2\quad 0) = \|\mathcal{Re}(q)\|^2 + \|\mathcal{Im}(q)\|^2 \tag{8}$$

is a sum of two positive numbers, it makes sense to define the norm of a quaternion as the scalar $\|q\| = (qq^*)^{1/2}$. Obviously, $\|q\| = 0$ if and only if q=0. Furthermore, for any r, $q\in\mathbb{H}$,

$$\|rq\|^2 = (rq)(rq)^* = (rq)(q^*r^*) = r(qq^*)r^* = (rr^*)\|q\|^2 = \|r\|^2\|q\|^2 \tag{9}$$

which means that the norm $q\to\|q\|$: $\mathbb{H}\to\mathbb{R}^+$ makes $\mathbb{H}$ a normed algebra.

q is called a unit quaternion if $\|q\| = 1$. The set of unit quaternions is the 3-sphere denoted as $\mathbb{S}^3$. And $\mathbb{S}^3$ is a group in terms of the quaternion product. We define a unit quaternion $\varsigma = (\boldsymbol{n}\sin\theta/2\quad\cos\theta/2)\in\mathbb{S}^3$. For any $\mathbf{y}\in\mathbb{R}^3$, the product $\varsigma\mathbf{y}\varsigma^*\in\mathcal{R}^3$. According to the following computation:

$$\varsigma\mathbf{y}\varsigma^* = ((\mathbf{y}\cdot\boldsymbol{n})\boldsymbol{n} + (\boldsymbol{n}\times\mathbf{y})\sin\alpha + [(\boldsymbol{n}\times\mathbf{y})\times\boldsymbol{n}]\cos\alpha\quad 0) \tag{10}$$

one can see Eq. (10) is identical to Eq. (4). Since ς describes the same rotation with $-\varsigma$, we choose the quaternion whose $\varsigma_0\geq 0$ when carrying out the computations. Thus, we can now transform the rotation matrix into a quaternion. Eq. (2) can be written as

$$\boldsymbol{E}_1(\theta) = \boldsymbol{P} + \varsigma\boldsymbol{a}_i\varsigma^*\quad (i = 1, 2,\cdots, n). \tag{11}$$

To establish the relation between the translation of the mobile platform and another quaternion, we construct the dual part $\lambda\in\mathbb{H}$ from the translation vector $\mathbf{p}$ and the rotation quaternion ς.

Here we recall some important rules of dual quaternions. ς and λ constitute a dual quaternion $\hat{\varsigma} = \varsigma + \varepsilon\lambda$, where ε is the dual unit ($\varepsilon\neq 0$ and $\varepsilon\varepsilon = 0$) and commutes with every element of the algebra.

The conjugate of a dual quaternion is the extension of the conjugate of a quaternion:

$$\hat{\varsigma}^* = \varsigma^* + \varepsilon\lambda^* \tag{12}$$

As for quaternions, the norm of the dual quaternion can be defined as $\|\hat{\varsigma}\| = \sqrt{\hat{\varsigma}\hat{\varsigma}^*}$. This is a dual number called the magnitude of the dual quaternion. When $\hat{\varsigma}$ is used to represent the spatial Euclidean displacements, the magnitude of it should be 1 [31–33]:

$$\|\hat{\varsigma}\|^2 = \varsigma\varsigma^* + \varepsilon(\varsigma\lambda^* + \lambda\varsigma^*) = \|\varsigma\|^2 + 2\varepsilon\mathcal{Re}(\varsigma\lambda^*) = 1. \tag{13}$$

This introduces two constraint equations on the components of $\hat{\varsigma}$:

$$\begin{cases}\|\varsigma\| = 1\\ \mathcal{Re}(\varsigma\lambda^*) = 0\end{cases}. \tag{14}$$

λ can be constructed from the vector $\mathbf{p}$ and the quaternion ς:

$$\lambda = c_1\mathbf{p}\varsigma,\quad \text{or}\quad \lambda = c_2\varsigma\mathbf{p} \tag{15}$$

where c_1 and c_2 can be arbitrary nonzero constants.

If a quaternion is considered as an element of $\mathbb{R}^4$, then the quaternion norm has the meaning of the Euclidean norm in $\mathbb{R}^4$. Define the inner product $\cdot$: $\mathbb{H}\times\mathbb{H}\to\mathbb{R}^+$ with the law

$$(q, r)\to q\cdot r = q_0r_0 + q_1r_1 + q_2r_2 + q_3r_3. \tag{16}$$

As the quaternion conjugate, norm and inner product are all linear operators, we obtain the transformation formulas required in

the later handling with the kinematic equations:

$$\begin{cases}(q+r)(q^*+r^*)=(q+r)\cdot(q+r)\\ r\cdot(qs)=(q^*r)\cdot s=(rs^*)\cdot q\end{cases}. \tag{17}$$

We now use the generalized coordinates $\mathbf{x}=(\varsigma \quad \lambda)^{\mathrm{T}}$ to parameterize the pose of the mobile platform. According to Eq.(15), we simply let $\lambda=\mathbf{p}\varsigma$ and transform Eq.(11) into

$$\mathbf{E}_1(\boldsymbol{\theta})\varsigma=\lambda+\varsigma\mathbf{a}_i \quad (i=1,2,\cdots,n). \tag{18}$$

For the forward kinematics problem, only ς and λ are unknowns in Eq.(18). It is sometimes necessary to separate the geometric variables from $\mathbf{E}_1(\boldsymbol{\theta})\varsigma$. Thus, we have

$$\mathbf{E}_2(\boldsymbol{\theta})\varsigma=\lambda+\varsigma\mathbf{a}_i+\mathbf{c}_i\varsigma \quad (i=1,2,\cdots,n) \tag{19}$$

where $\mathbf{c}_i$ is the constant vector separated from $\mathbf{E}_1(\boldsymbol{\theta})$. Note that for a six-DOF parallel robot in practical engineering, the compound joints, including the universal joints and the spherical joints, are always designed as the combination of the revolute joints. For example, the 6-UPS structure is actually a 6-2RP3R one. The 2R should be designed with intersecting rotation axes and so should 3R such that the passive joint varibles will not be introduced in $\mathbf{c}_i$.

By computing the inner product of (19) with itself, we get

$$(\mathbf{E}_2(\boldsymbol{\theta})\varsigma)\cdot(\mathbf{E}_2(\boldsymbol{\theta})\varsigma)=(\lambda+\varsigma\mathbf{a}_i+\mathbf{c}_i\varsigma)\cdot(\lambda+\varsigma\mathbf{a}_i+\mathbf{c}_i\varsigma) \quad (i=1,2,\cdots,n). \tag{20}$$

According to Eq.s (17) and (20) can be simplified as

$$\|\mathbf{E}_2(\boldsymbol{\theta})\|^2=\|\lambda\|^2+\|\mathbf{a}_i\|^2+\|\mathbf{c}_i\|^2+2(\varsigma^*\lambda)\cdot\mathbf{a}_i+2(\lambda\varsigma^*)\cdot\mathbf{c}_i+2\mathbf{a}_i\cdot\ \mathbf{c}_i \quad (i=1,2,\cdots,n). \tag{21}$$

Eq. (21) is composed of only quadratic terms of ς, λ and constants. Thus, it can be written in the quadratic form of $\mathbf{x}$ as below:

$$f_i(\mathbf{x})=\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}_i\mathbf{x}-\mathrm{C}_i=0 \quad (i=1,2,\cdots,n) \tag{22}$$

where C_i is a constant determined by the actuated joint inputs and the geometric parameters, and $\mathbf{Q}_i$ is a constant symmetric matrix determined by the geometric parameters. Besides, Eq. (14) can be written in the quadratic form of $\mathbf{x}$ as well:

$$\begin{cases}f_{n+1}(\mathbf{x})=\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}_{n+1}\mathbf{x}-1=0\\ f_{n+2}(\mathbf{x})=\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}_{n+2}\mathbf{x}=0\end{cases},\ \mathbf{Q}_{n+1}=\begin{pmatrix}2\boldsymbol{I}_{4\times4} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}_{4\times4}\end{pmatrix},\ \mathbf{Q}_{n+2}=\begin{pmatrix}\boldsymbol{0}_{4\times4} & \boldsymbol{I}\\ \boldsymbol{I} & \boldsymbol{0}_{4\times4}\end{pmatrix}. \tag{23}$$

Eqs. (22) and (23) together constitute the forward kinematic equations of the six-DOF parallel robot. The number of the equations are usually eight in total for the fully actuated parallel robots, while it is more than eight for the redundant actuated parallel robots. The most important step to generate the forward kinematic equations is to extract the constant symmetric matrix $\mathbf{Q}_i$ and constant scalar C_i from the closed-loop equations. The efficient numerical algorithm will hereinafter be constructed on the basis of this set of quadratic equations.

3. An efficient algorithm for the forward kinematic equations

The been transformed into the problem of dealing with a set of nonlinear algebraic equations and there exist many feasible methods [34]. However, we notice that the forward kinematic equations based on the unit dual quaternion have generally a quadratic form and this characteristic can be fully utilized. Therefore, in this section we will discuss the numerical solution to the quadratic equations as well as the convergence and singularity problems for the algorithm.

3.1. Construction of the algorithm

For any of Eqs. (22) and (23), if let $\mathbf{a}$, $\mathbf{b}\in\mathbb{R}^{n+2}$, then

$$f_i(\mathbf{a})-f_i(\mathbf{b})=\mathbf{b}^{\mathrm{T}}\mathbf{Q}_i(\mathbf{a}-\mathbf{b})+\frac{1}{2}(\mathbf{a}-\mathbf{b})^{\mathrm{T}}\mathbf{Q}_i(\mathbf{a}-\mathbf{b}) \quad (i=1,2,\cdots,n+2). \tag{24}$$

Assume $\mathbf{x}_d\in\mathbb{R}^{n+2}$ is a real solution to $f_i(\mathbf{x})=\mathbf{x}^{\mathrm{T}}\mathbf{Q}_i\mathbf{x}/2-\mathrm{C}_i \quad (i=1,2,\cdots,n+2)$ and $\mathbf{x}_k\in\mathbb{R}^{n+2}$ is an approximation of $\mathbf{x}_d$. Making $\mathbf{a}=\mathbf{x}_d$, $\mathbf{b}=\mathbf{x}_k$, $\Delta\mathbf{x}=\mathbf{x}_d-\mathbf{x}_k$ in Eq. (24) and considering $f_i(\mathbf{x}_d)=\mathbf{0}$, one can get

$$-f_i(\mathbf{x}_k)=\mathbf{x}_k{}^{\mathrm{T}}\mathbf{Q}_i(\mathbf{x}_d-\mathbf{x}_k)+\frac{1}{2}\Delta\mathbf{x}^{\mathrm{T}}\mathbf{Q}_i\Delta\mathbf{x} \quad (i=1,\cdots,n+2). \tag{25}$$

By omitting $(\Delta\mathbf{x}^{\mathrm{T}}\mathbf{Q}_i\Delta\mathbf{x})/2$, which implies in the geometric sense to replace the quadric surface with the tangent plane at $\mathbf{x}_k$, one can get

$$-f_i(\mathbf{x}_k)\approx\mathbf{x}_k{}^{\mathrm{T}}\mathbf{Q}_i(\mathbf{x}_d-\mathbf{x}_k) \quad (i=1,\cdots,n+2). \tag{26}$$

Then the iterative sequence is

$$\mathbf{x}_{k+1} = \Phi(\mathbf{x}_k) = \mathbf{x}_k - (\mathbf{J}_k^{\mathrm{T}}\mathbf{J}_k)^{-1}\mathbf{J}_k^{\mathrm{T}}\mathbf{F}(\mathbf{x}_k) \quad (k = 0, 1, 2\ldots) \tag{27}$$

where $\mathbf{J}_k = (\mathbf{Q}_1\mathbf{x}_k \quad \mathbf{Q}_2\mathbf{x}_k \ \ldots \ \mathbf{Q}_{n+2}\mathbf{x}_k)^{\mathrm{T}}$ and $\mathbf{F}(\mathbf{x}_k) = (f_1(\mathbf{x}_k) \quad f_2(\mathbf{x}_k) \ \ldots f_{n+2}(\mathbf{x}_k))^{\mathrm{T}}$.

To further increase the efficiency of the algorithm, we simplify and refine the iterative sequence. Considering the relation between $\mathbf{F}(\mathbf{x}_k)$ and $\mathbf{J}_k$:

$$\mathbf{F}(\mathbf{x}_k) = \frac{1}{2}\mathbf{J}_k\mathbf{x}_k - \mathbf{C}, \quad \mathbf{C} = (C_1 \ C_2 \cdots C_n 1 \ 0)^{\mathrm{T}} \tag{28}$$

we can consequently eliminate $\mathbf{F}(\mathbf{x}_k)$ in Eq.(27). The iterative function can be written as

$$\Phi(\mathbf{x}_k) = \frac{1}{2}\mathbf{x}_k + (\mathbf{J}_k^{\mathrm{T}}\mathbf{J}_k)^{-1}\mathbf{J}_k^{\mathrm{T}}\mathbf{C}. \tag{29}$$

Now we needn't to compute $\mathbf{F}(\mathbf{x}_k)$ in each iteration. Moreover, we prefer to solve the linear equations other than to solve the inverse matrix since solving the inverse matrix of $\mathbf{J}_k^{\mathrm{T}}\mathbf{J}_k$ is time consuming. Thus, we obtain

$$\begin{cases} \mathbf{x}_{k+1} = \frac{1}{2}\mathbf{x}_k + \Delta\mathbf{x}_k \\ (\mathbf{J}_k^{\mathrm{T}}\mathbf{J}_k)\Delta\mathbf{x}_k = \mathbf{J}_k^{\mathrm{T}}\mathbf{C} \end{cases} (k = 0, 1, 2\cdots). \tag{30}$$

In fact, the algorithm (30) gives the least square solution of the forward kinematic equations. When the parallel robot does not have the redundant actuation, the least square solution is the exact solution and the Jacobian matrix $\mathbf{J}_k$ is a 8×8 square matrix. At this moment the algorithm (30) can also be written as

$$\begin{cases} \mathbf{x}_{k+1} = \frac{1}{2}\mathbf{x}_k + \Delta\mathbf{x}_k \\ \mathbf{J}_k\Delta\mathbf{x}_k = \mathbf{C} \end{cases} (k = 0, 1, 2\cdots). \tag{31}$$

3.2. Convergence and singularity analysis of the algorithm

According to the iteration function (27), one can easily solve its derivative with respect to $\mathbf{x}$:

$$\Phi'(\mathbf{x}) = \boldsymbol{E}^{8\times8} - ((\mathbf{J}^{\mathrm{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathrm{T}})'\boldsymbol{F}(\mathbf{x}) - (\mathbf{J}^{\mathrm{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathrm{T}}\mathbf{J} = -((\mathbf{J}^{\mathrm{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathrm{T}})'\boldsymbol{F}(\mathbf{x}) \tag{32}$$

where $\mathbf{E}^{8\times8}$ is a 8×8 identity matrix. The value of $\Phi'(\mathbf{x})$ at $\mathbf{x}_d$ is

$$\Phi'(\mathbf{x}_d) = -((\mathbf{J}^{\mathrm{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathrm{T}})'\mathbf{F}(\mathbf{x}_d) = \mathbf{0}. \tag{33}$$

Therefore, the spectral radius of $\Phi'(\mathbf{x}_d)$ certainly less than 1 since it is valued as zero. Then according to the local convergence theory, there exists a neighborhood of $\mathbf{x}_d$, denoted as $S_\delta = \{\mathbf{x} \in \mathbb{R}^8 | \|\mathbf{x}_d - \mathbf{x}\| < \delta\}$, such that the sequence converges to $\mathbf{x}_d$ for any initial values $\mathbf{x}_0 \in S_\delta$. S_δ is called the converging interval of $\mathbf{x}_d$.

According to Eqs.(25) and (27), one can get

$$\mathbf{x}_d - \mathbf{x}_{k+1} = (\mathbf{J}_k^{\mathrm{T}}\mathbf{J}_k)^{-1}\mathbf{J}_k^{\mathrm{T}}\mathbf{H}_k/2 \tag{34}$$

where $\mathbf{H}_k = ((\mathbf{x}_d - \mathbf{x}_k)^{\mathrm{T}}\mathbf{Q}_1(\mathbf{x}_d - \mathbf{x}_k)\ldots(\mathbf{x}_d - \mathbf{x}_k)^{\mathrm{T}}\mathbf{Q}_{n+2}(\mathbf{x}_d - \mathbf{x}_k))^{\mathrm{T}}$. If $\mathbf{J}_k$ has full column rank, then

$$\|\mathbf{x}_d - \mathbf{x}_{k+1}\| \le c_3\|\mathbf{x}_d - \mathbf{x}_{k+1}\|^2 \tag{35}$$

where c_3 is the supremum of the coefficient of $\|\mathbf{x}_d - \mathbf{x}_{k+1}\|^2$. Therefore, the algorithm has a quadratic convergence.

In general, it is not easy to determine a good initial value. Fortunately, we are aware of that $\boldsymbol{\theta}$ is a continuous function of $\mathbf{x}$ for the parallel robots. There exists a neighborhood of the real actuated joints varibles $\boldsymbol{\theta}_d$, denoted as $S_\Delta = \{\boldsymbol{\theta} \in \mathbb{R}^{n+2} | \|\boldsymbol{\theta}_d - \boldsymbol{\theta}\| < \Delta\}$, such that $\mathbf{x} \in S_\delta$ when $\boldsymbol{\theta} \in S_\Delta$. Moreover, the Jacobian matrix of the parallel robot in the planned workspace should be well conditioned, which means a small change of $\mathbf{x}$ results in a small change of $\boldsymbol{\theta}$. Therefore, $\mathbf{x}_0$ can be set in S_δ by regulating the change of $\boldsymbol{\theta}$.

In the real-time control, the sensors in the legs measure a sequence of joint varibles: $\boldsymbol{\theta}^0$, $\boldsymbol{\theta}^1$, $\cdots$. During the i^{th} control cycle, the initial value for the iteration is set as the results of the previous control cycle, denoted as $\mathbf{x}_d^{i-1}$. And the results of this cycle, denoted as $\mathbf{x}_d^i$, will be used as the next initial value. If $\mathbf{x}_d^{i-1}$ is located out of the converging interval of $\mathbf{x}_d^i$, we will use a convergence strategy here: artificially insert a sample point, $\boldsymbol{\theta}^i + (\boldsymbol{\theta}^{i+1} - \boldsymbol{\theta}^i)/2$, between $\boldsymbol{\theta}^i$ and $\boldsymbol{\theta}^{i+1}$ to halve the change of $\boldsymbol{\theta}$. This strategy can be used tautologically until the convergence.

On the other hand, for the parallel robots, the pose of the mobile platform $\mathbf{x}$ has multi solutions for the given actuated joint variables $\boldsymbol{\theta}$. When $\boldsymbol{\theta}$ changes continuously, the forward kinematic equations have several solution spaces, which are usually not intersecting [1]. The aim of the new algorithm is to find the actual solution in these spaces. However, when the Jacobian matrix $\mathbf{J}_k$ is singular or close to singularity, it will be difficult to converge, or the iterative sequence will jump from one solution space into another and converge to an unexpected solution [34], making the real-time control failed. Fortunately, we have discovered that the singularity of the algorithm has a deep relation with the kinematic singularity of the parallel robot.

Differentiate Eq.(1) with respect to time and we get

$$\frac{\partial \mathbf{E}}{\partial \boldsymbol{\theta}}\dot{\boldsymbol{\theta}} + \frac{\partial \mathbf{E}}{\partial \mathbf{x}}\dot{\mathbf{x}} = \mathbf{J}_\theta\dot{\boldsymbol{\theta}} + \mathbf{J}_x\dot{\mathbf{x}} = \mathbf{0}. \tag{36}$$

In Ref. [35], Gosselin and Angeles defined and discussed the singularity of the parallel mechanisms. They distinguished serial singularity obtained when $\mathbf{J}_\theta$ is singular and parallel singularity obtained when $\mathbf{J}_x$ is singular. $\mathbf{J}_\theta$ is named as the inverse-kinematics Jacobian matrix and $\mathbf{J}_x$ is named as the forward-kinematics Jacobian matrix. We observe that $\mathbf{J}_x$ is related to $\mathbf{J}_k$, which can be revealed by computing the time derivatives of Eqs.(22) and (23). Both of them are the coefficient matrix in front of $\dot{\mathbf{x}}$, the generalized speeds of the system. The relation is obtained as

$$\mathbf{J}_k = \begin{pmatrix} \mathbf{J}_x \\ 2\varsigma \quad \mathbf{0}_4 \\ \lambda \quad \varsigma \end{pmatrix}. \tag{37}$$

$\mathbf{J}_k$ is singular if and only if $\mathbf{J}_x$ is singular. Therefore, the algorithm is always effective as long as the robot is not in a parallel singular state. Sometimes the generalized speeds of the system are set as the twist of the mobile platform, $\mathbf{w}$, composed of its translational and angular velocity. Then Eq.(36) with be

$$\frac{\partial \mathbf{E}}{\partial \boldsymbol{\theta}}\dot{\boldsymbol{\theta}} + \frac{\partial \mathbf{E}}{\partial \mathbf{x}}\dot{\mathbf{x}} = \mathbf{J}_\theta\dot{\boldsymbol{\theta}} + \mathbf{J}_w\mathbf{w} = \mathbf{0}. \tag{38}$$

The forward-kinematics Jacobian matrix now is $\mathbf{J}_w$. It is easy to see $\mathbf{J}_x\dot{\mathbf{x}} = \mathbf{J}_w\mathbf{w}$ by comparing Eq.(36) with Eq.(38). In order to derive the relation between $\mathbf{J}_k$ and $\mathbf{J}_w$, we need to find the transformation matrix $\mathbf{H}$ relating $\dot{\mathbf{x}}$ and $\mathbf{w}$, making $\mathbf{w} = \mathbf{H}\dot{\mathbf{x}}$ and $\mathbf{J}_x = \mathbf{J}_w\mathbf{H}$. Thus,

$$\mathbf{J}_k = \begin{pmatrix} \mathbf{J}_w\mathbf{H} \\ 2\varsigma \quad \mathbf{0}_4 \\ \lambda \quad \varsigma \end{pmatrix}. \tag{39}$$

To obtain $\mathbf{H}$, it is necessary to figure out the relations between $\mathbf{v}$, $\boldsymbol{\omega}$ and $\dot{\varsigma}$, $\dot{\lambda}$. We differentiate $\mathbf{R}\boldsymbol{y} = \varsigma\boldsymbol{y}\varsigma^*$ with respect to time and obtain

$$\dot{\mathbf{R}}\boldsymbol{y} = \dot{\varsigma}\boldsymbol{y}\varsigma^* + \varsigma\boldsymbol{y}\dot{\varsigma}^* = \dot{\varsigma}\boldsymbol{y}\varsigma^* - (\dot{\varsigma}\boldsymbol{y}\varsigma^*)^*. \tag{40}$$

Since ς is a unit quaternion: $\varsigma\varsigma^* = 1$ and $\dot{\mathbf{R}}\boldsymbol{y} = \boldsymbol{\omega} \times \mathbf{R}y = \boldsymbol{\omega} \times (\varsigma\boldsymbol{y}\varsigma^*)$, Eq. (40) can be transformed into

$$\boldsymbol{\omega} \times (\varsigma\boldsymbol{y}\varsigma^*) = \dot{\varsigma}\varsigma^*(\varsigma\boldsymbol{y}\varsigma^*) - (\dot{\varsigma}\varsigma^*(\varsigma\boldsymbol{y}\varsigma^*))^*. \tag{41}$$

Differentiate $\varsigma\varsigma^* = 1$ with respect to time and we get $\dot{\varsigma}\varsigma^* + (\dot{\varsigma}\varsigma^*)^* = \dot{\varsigma}\varsigma^* + \varsigma\dot{\varsigma}^* = 0$, from which we can conclude $Re(\varsigma\varsigma^*) = 0$. Thus, $\varsigma\varsigma^*$ is a pure vector. Moreover, for any two quaternions r and s, one has $rs - (rs)^* = 2r_0\mathbf{s} + 2s_0\mathbf{r} + 2\mathbf{r} \times \mathbf{s}$. Especially when both r and s are pure vectors, one can get $rs - (rs)^* = 2\mathbf{r} \times \mathbf{s}$. Consequently, Eq. (41) is simplified to be

$$\boldsymbol{\omega} \times (\varsigma\boldsymbol{y}\varsigma^*) = 2(\dot{\varsigma}\varsigma^*) \times (\varsigma\boldsymbol{y}\varsigma^*). \tag{42}$$

One can obtain the relation of $\boldsymbol{\omega}$ and $\dot{\varsigma}$ from Eq. (42):

$$\boldsymbol{\omega} = 2\dot{\varsigma}\varsigma^*. \tag{43}$$

According to the relation of λ and $\mathbf{p}$: $\lambda = \mathbf{p}\varsigma$, we have $\mathbf{p} = \lambda\varsigma^*$. Differentiate it with respect to time and we get

$$\mathbf{v} = \dot{\mathbf{p}} = \dot{\lambda}\varsigma^* + \lambda\dot{\varsigma}^*. \tag{44}$$

Eqs. (43) and (44) can be written in the matrix form since the product of two quaternions is bilinear of themselves. Thus,

$$\mathbf{H} = \begin{pmatrix} \overset{+}{\lambda}\mathbf{G} & \bar{\varsigma} \\ 2\bar{\varsigma} & \mathbf{0}_{3\times4} \end{pmatrix} \tag{45}$$

where

$$\overset{+}{\lambda} = \begin{pmatrix} \lambda_0 & -\lambda_3 & \lambda_2 & \lambda_1 \\ \lambda_3 & \lambda_0 & -\lambda_1 & \lambda_2 \\ -\lambda_2 & \lambda_1 & \lambda_0 & \lambda_3 \end{pmatrix} \text{ and } \bar{\varsigma} = \begin{pmatrix} \varsigma_0 & \varsigma_3 & -\varsigma_2 & \varsigma_1 \\ -\varsigma_3 & \varsigma_0 & \varsigma_1 & \varsigma_2 \\ \varsigma_2 & -\varsigma_1 & \varsigma_0 & \varsigma_3 \end{pmatrix} \tag{46}$$

The rows of $\overset{+}{\lambda}$ and $\bar{\varsigma}$ are orthogonal mutually. $\mathbf{G} = diag(-1 \ -1 \ -1 \ 1)$, is an orthogonal matrix. We discover that $\mathbf{H}$ is of full row rank by computing $\mathbf{H}\mathbf{H}^{\mathrm{T}}$. It makes $rank(\mathbf{J}_x) = rank(\mathbf{J}_w\mathbf{H})$. $\mathbf{J}_k$ is singular if and only if $\mathbf{J}_w$ is singular. Therefore, in this situation we conclude likewise that the algorithm is effective as long as the robot is not in a parallel singular state. In other words, the algorithm is always valid in the singularity-free workspace of the parallel robot.

It is necessary to carry out the singularity analysis and obtain the singularity-free workspace, denoted as S_x, before using the algorithm to control the parallel robot. The pose of the mobile platform is figured out in each cycle by (30) or (31), where the initial guess is set as the result of the previous cycle. If the result in one iteration exceeds S_x, the convergence strategy will work.

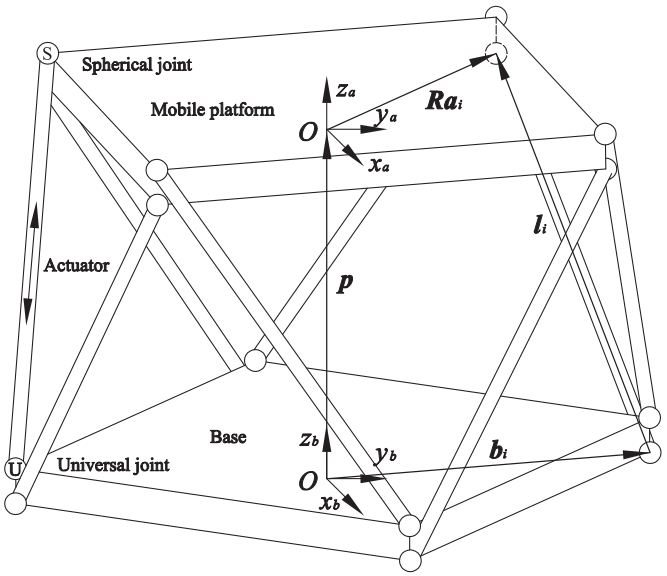

Fig. 1. the geometric description of 8-UPS robot.

4. Verification

To verify the validity and effectiveness of the new algorithm for the redundant actuated parallel robots, an 8-UPS and an 8-PUS parallel robots with non-planar platforms are set as examples. Figs. 1 and 2 describe the geometry of the 8-UPS and 8-PUS parallel robots respectively. $O-x_ay_az_a$ is a mobile coordinate frame attached to the mobile platform. $O-x_by_bz_b$ is a fixed coordinate frame attached to the base. $\mathbf{a}_i(i=1,\ \ 2\cdots8)$ is the position vector in $O-x_ay_az_a$ of the connection center of the mobile platform and each kinematic chain. $\mathbf{b}_i(i=1,\ \ 2\cdots8)$ is the position vector in $O-x_by_bz_b$ of the connection center of the base and each kinematic chain. The physical units the paper has not identified adopt the International System of Units, SI.

4.1. 8-UPS parallel robot

The geometry of 8-UPS parallel robot can be uniquely determined by the parameters: $\mathbf{a}_i(i=1,\ \ 2\cdots8)$, $\mathbf{b}_i(i=1,\ \ 2\cdots8)$, and the

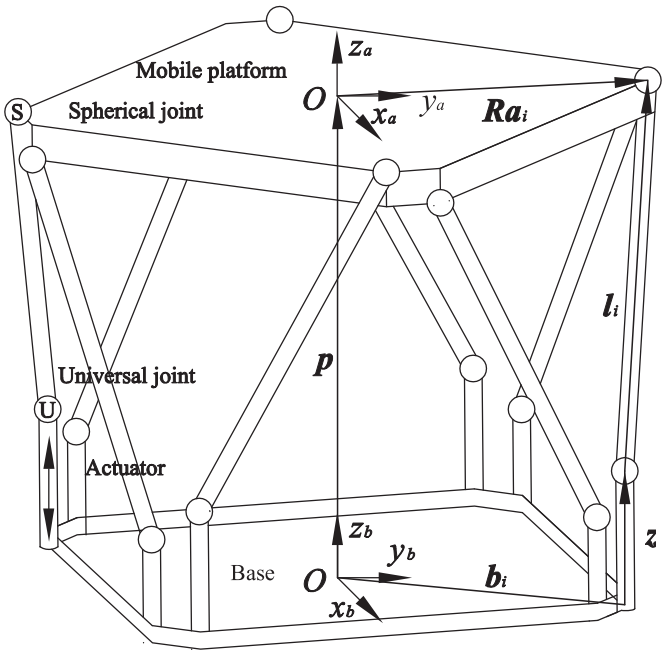

Fig. 2. the geometric description of 8-PUS robot.

initial length of each leg. The spherical and universal joints are located symmetrically in four circles. Thus

$$\begin{cases} \mathbf{a}_{2i-1} = 0.9(\cos(\pi(i-1)/2) \;\; \sin(\pi(i-1)/2) \;\; 0.1)^{\mathrm{T}} \\ \mathbf{a}_{2i} = 0.9(\cos(\pi i/2) \;\; \sin(\pi i/2) \;\; 0)^{\mathrm{T}} \\ \mathbf{b}_{2i-1} = 0.9(\cos(\pi/4+\pi(i-1)/2) \;\; \sin(\pi/4+\pi(i-1)/2) \;\; 0.1)^{\mathrm{T}} \\ \mathbf{b}_{2i} = 0.9(\cos(-\pi/4+\pi i/2) \;\; \sin(-\pi/4+\pi i/2) \;\; 0)^{\mathrm{T}} \end{cases} \quad (i = 1, 2, 3). \tag{47}$$

Sincs $\mathbf{Q}_9$, $\mathbf{Q}_{10}$, C_9 and C_{10} are known to us and keep constant, we only need to generate $\mathbf{Q}_i (i = 1, \;\; 2\cdots 8)$ and $\mathrm{C}_i (i = 1, \;\; 2\cdots 8)$ from the kinematic closed-loop equations for preparation. We can get the kinematic equations from the closed-loop vectors in Fig. 1:

$$\mathbf{l}_i + \mathbf{b}_i = \mathbf{p} + \mathbf{R}\mathbf{a}_i \quad (i = 1, \;\; 2\cdots 8). \tag{48}$$

Based on the method introduced in Section2, $\mathbf{Q}_i$ and C_i are generated as

$$\begin{aligned} \mathbf{Q}_i &= \begin{pmatrix} \mathbf{Q}_{i11} & \mathbf{Q}_{i12} \\ \mathbf{Q}_{i21} & \mathbf{Q}_{i22} \end{pmatrix} \quad (i = 1, 2\cdots 8) \\ \mathrm{C}_i &= \mathbf{l}_i^2 \end{aligned} \tag{49}$$

where $\mathbf{Q}_{i11}$ is 4×4 symmetric matrix, $\mathbf{Q}_{i12}$, $\mathbf{Q}_{i21}$ are both 4×4 antisymmetric matrices and $\mathbf{Q}_{i22} = \mathbf{I}_{4\times 4}$. We set a rule here: the coordinate presentation of any vector $\mathbf{u} \in \mathbb{R}^3$ is $(u_x \;\; u_y \;\; u_z)^{\mathrm{T}}$. For more compact expression, we introduce the intermediate varibles $\mathbf{A}_i = \mathbf{a}_i + \mathbf{b}_i$ and $\mathbf{B}_i = \mathbf{a}_i - \mathbf{b}_i$. The block matrices in Eq. (49) are finally written as

$$\mathbf{Q}_{i11} = \begin{pmatrix} 2(B_{xi}^2 + A_{yi}^2 + A_{zi}^2) & -4(a_{ix}b_{iy} + a_{iy}b_{ix}) & -4(a_{ix}b_{iz} + a_{iz}b_{ix}) & -4(a_{iy}b_{iz} - a_{iz}b_{iy}) \\ -4(a_{ix}b_{iy} + a_{iy}b_{ix}) & 2(A_{xi}^2 + B_{yi}^2 + A_{zi}^2) & -4(a_{iy}b_{iz} + a_{iz}b_{iy}) & -4(a_{ix}b_{iz} - a_{iz}b_{ix}) \\ -4(a_{ix}b_{iz} + a_{iz}b_{ix}) & -4(a_{iy}b_{iz} + a_{iz}b_{iy}) & 2(A_{xi}^2 + A_{yi}^2 + B_{zi}^2) & -4(a_{ix}b_{iy} - a_{iy}b_{ix}) \\ -4(a_{iy}b_{iz} - a_{iz}b_{iy}) & -4(a_{ix}b_{iz} - a_{iz}b_{ix}) & -4(a_{ix}b_{iy} - a_{iy}b_{ix}) & 2(B_{xi}^2 + B_{yi}^2 + B_{zi}^2) \end{pmatrix} \tag{50}$$

$$\mathbf{Q}_{i21} = \mathbf{Q}_{i12}^{\mathrm{T}} = \begin{pmatrix} 0 & 2A_{iz} & -2A_{iy} & 2B_{ix} \\ -2A_{iz} & 0 & 2A_{ix} & 2B_{iy} \\ 2A_{iy} & -2A_{ix} & 0 & 2B_{iz} \\ -2B_{ix} & -2B_{iy} & -2B_{iz} & 0 \end{pmatrix}. \tag{51}$$

Assume that the 8-UPS parallel robot's mobile platform moves from the initial state

$$\mathbf{p}_0 = (0 \;\; 0 \;\; 1)^{\mathrm{T}}, \quad \mathbf{R}_0 = \mathbf{I}_{3\times 3}. \tag{52}$$

to the final state

$$\mathbf{p}_{\mathrm{f}} = (0.1 \;\; 0.1 \;\; 1.1)^{\mathrm{T}}, \quad \mathbf{R}_{\mathrm{f}} = \mathbf{R}(\pi/12, \mathbf{x}_a).\, \mathbf{R}(\pi/12, \mathbf{y}_a).\, \mathbf{R}(\pi/12, \mathbf{z}_a) = \begin{pmatrix} 0.933012701892 & -0.25 & 0.258819045102 \\ 0.314704761275 & 0.915675113362 & -0.25 \\ -0.174494158464 & 0.314704761275 & 0.933012701892 \end{pmatrix}. \tag{53}$$

The generalized coordinates $\mathbf{x}_0 = (\varsigma_0 \;\; \lambda_0)^{\mathrm{T}}$ and $\mathbf{x}_{\mathrm{f}} = (\varsigma_{\mathrm{f}} \;\; \lambda_{\mathrm{f}})^{\mathrm{T}}$, who correspond to the initial and final states, are respectively

$$\begin{aligned} \varsigma_0 &= (0 \;\; 0 \;\; 0 \;\; 1) \\ \lambda_0 &= (0 \;\; 0 \;\; 1 \;\; 0) \\ \varsigma_{\mathrm{f}} &= (0.145193738361 \;\; 0.111411073930 \;\; 0.145193738361 \;\; 0.972329743084) \\ \lambda_{\mathrm{f}} &= (-0.0107998331791 \;\; 0.242426712670 \;\; 1.06618445095 \;\; -0.185373593427) \end{aligned} \tag{54}$$

The termination condition of the algorithm is set as $\|\Delta\mathbf{x}\| < 10^{-8}$ and the initial guess is $\mathbf{x}_0$. We perform the simulation under the condition of Win 10/ Intel Core i7 3.4 GHz/ 16GB RAM/ Mathematica 9. The new algorithm converges by 5 iterations and the results in each iteration are shown in Table 1, where the difference of $\mathbf{x}_5$ and $\mathbf{x}_{\mathrm{f}}$ stays within the range of the allowable error, demonstrating the validity of the algorithm.

The new algorithm has the advantages that the derivatives (Jacobian matrix) is a linear function of the dual quaternion and the equations needn't be evaluated in each iteration, making almost no time consumption of Jacobian matrix updating in each control cycle. The time consumption of the new algorithm in one control cycle is 0.2187 milliseconds. We construct Newton's method in the same environment and solve the forward kinematic equations of the 8-UPS parallel robot, whose unknowns are $\mathbf{p}$ and $\mathbf{R}$, parameterized by Euler angles. The time consumption of Newton's method is 14.25 milliseconds. If one want to obtain a univariate polynomial equation firstly by analytical methods, for example, a polynomial equation of degree 14 in Ref. [7] or 17 in Ref. [12], and then solve it by a numerical method, it will be inefficient for control because one need to firstly update polynomial equation or the coefficients of the univariate in each control cycle, which is very complicated and time consuming.

One can see that the new algorithm is efficient enough for real-time control. When a parallel robot is used for industrial or medical devices, six-axis vibration isolation, etc., the frequency of the control cycle is always set as hundreds to thousands of Hertz. The new algorithm only takes a small part of the control cycle, meeting the requirements of the real-time control system.

Table 1
the iterations of the new algorithm for the 8-UPS robot.

k	The values of $\mathbf{x}_i = (\varsigma_i \ \ \lambda_i)^{\mathrm{T}}$ in each iteration
0	$\varsigma_0 = (0 \ \ 0 \ \ 0 \ \ 1)\ \lambda_0 = (0 \ \ 0 \ \ 1 \ \ 0)$
1	$\varsigma_1 = (0.155666437146 \ \ 0.0862941629103 \ \ 0.144061596061 \ \ 1)$
	$\lambda_1 = (0.0147122185276 \ \ 0.262909843185 \ \ 1.15502723518 \ \ -0.144061596061)$
2	$\varsigma_2 = (0.145382998508 \ \ 0.109364125511 \ \ 0.145080889594 \ \ 0.973206458664)$
	$\lambda_2 = (-0.0104604833615 \ \ 0.244178188541 \ \ 1.07101668723 \ \ -0.184695501937)$
3	$\varsigma_3 = (0.145192504368 \ \ 0.11140123422 \ \ 0.145193221895 \ \ 0.972333426917)$
	$\lambda_3 = (-0.0107996527373 \ \ 0.242430729237 \ \ 1.06619953688 \ \ -0.185376164588)$
4	$\varsigma_4 = (0.145193738655 \ \ 0.11141099968 \ \ 0.145193737067 \ \ 0.972329751788)$
	$\lambda_4 = (-0.0107998290183 \ \ 0.242426713867 \ \ 1.06618445136 \ \ -0.185373572711)$
5	$\varsigma_5 = (0.145193738699 \ \ 0.111410999828 \ \ 0.145193737078 \ \ 0.97232975171)$
	$\lambda_5 = (-0.0107998290222 \ \ 0.242426713862 \ \ 1.06618445117 \ \ -0.185373572686)$

In the real-time control, the motion from the initial state to the final state is generally divided into a plurality of control cycles, which means to interpolate form $\boldsymbol{\theta}_0$ to $\boldsymbol{\theta}_{\mathrm{f}}$. The pose of the mobile platform is figured out and fed back in each control cycle. We will perform this kind of simulation on the 8-PUS parallel robot in the next subsection 4.2.

4.2. 8-PUS parallel robot

The geometric parameters of the 8-PUS parallel robot in Fig. 2 is composed of $\mathbf{a}_i (i = 1, \ \ 2\cdots 8)$,$\mathbf{b}_i (i = 1, \ \ 2\cdots 8)$ and the leg length $\|\mathbf{l}_i\| (i = 1, \ \ 2\cdots 8) = 1.2143$, and we assume that the actuated joint initiatives $\|\mathbf{z}_i\| = 0 (i = 1, \ \ 2\cdots 8)$.As the 8-UPS parallel robot, the spherical joints and universal joints are symmetrically located in four circles, and $\mathbf{a}_i$, $\mathbf{b}_i$ remain the same.

The derivation of $\mathbf{Q}_i$ and C_i is omitted here since it is similar with that of the 8-UPS parallel robot. Although the sampling rate in real controllers need to be very large, we just use a very small one (5 Hz) in this paper for demonstration. The movement of the robot is set according to Eqs. (52) and (53). The real-time simulation lasts 1 second which is divided into 5 control cycles evenly. We need to plan the motion of the mobile platform firstly. To obtain the constant translational and angular speeds, we make the linear interpolation for the translation and the spherical linear interpolation for the rotation:

$$\varsigma(t) = \varsigma_0 \frac{\sin((1-t)\phi)}{\sin\phi} + \varsigma_{\mathrm{f}} \frac{\sin(t\phi)}{\sin\phi}, \quad \lambda(t) = ((1-t)\ \mathbf{p}_0 + t\ \ \mathbf{p}_{\mathrm{f}})\,\varsigma(t) \tag{55}$$

where $\phi = \arccos(\varsigma_0 \cdot \varsigma_{\mathrm{f}})$ and $t \in [0, 1]$.

Then according to the closed-loop equations of the 8-PUS parallel robot

$$\mathbf{l}_i + \mathbf{b}_i + \mathbf{z}_i = \mathbf{p} + \mathbf{R}\mathbf{a}_i \ \ (i = 1, \ \ 2\cdots 8) \tag{56}$$

we figure out the displacements of the actuated joints

$$\|\mathbf{z}_i\| = \mathbf{d}_{iz}^2 - \sqrt{\mathbf{l}_i^2 - \mathbf{d}_{ix}^2 - \mathbf{d}_{iy}^2}, \qquad \mathbf{d}_i = \lambda\varsigma^* + \varsigma\boldsymbol{a}_i\varsigma^* - \boldsymbol{b}_i \quad (i = 1, 2\cdots 8). \tag{57}$$

The discretion of $\|\mathbf{z}_i(t)\|$ is performed according to the sampling rate. In each sampling period, the algorithm gives the pose of the mobile platform, where the initial value is the result of the previous period, and the termination condition of the iteration is set as $\|\Delta\mathbf{x}\| < 10^{-8}$. The process and results are shown in Table 2, where the difference of $\mathbf{x}_5$ and $\mathbf{x}_{\mathrm{f}}$ stays within the range of the allowable error.

Table 2
the simulation of the 8-PUS robot.

Control cycles	Results in the sequential sampling periods $\mathbf{x}_i = (\varsigma_i \ \ \lambda_i)^{\mathrm{T}}$
0	$\varsigma_0 = (0 \ \ 0 \ \ 0 \ \ 1)\ \lambda_0 = (0 \ \ 0 \ \ 1 \ \ 0)$
1	$\varsigma_1 = (0.153105379505 \ \ 0.115265782727 \ \ 0.175248993052 \ \ 0.960870138908)$
	$\lambda_1 = (-0.064575434607 \ \ 0.26200663647 \ \ 1.0975712739 \ \ -0.221322117894)$
2	$\varsigma_2 = (0.152031050766 \ \ 0.114788123132 \ \ 0.170732747621 \ \ 0.962391747993)$
	$\lambda_2 = (-0.0540186315342 \ \ 0.261520997342 \ \ 1.09253201857 \ \ -0.21647940037)$
3	$\varsigma_3 = (0.150402727804 \ \ 0.114116014616 \ \ 0.16464013534 \ \ 0.96467501163)$
	$\lambda_3 = (-0.0405702528376 \ \ 0.259249609234 \ \ 1.08531689159 \ \ -0.209572522333)$
4	$\varsigma_4 = (0.148109357088 \ \ 0.113013189218 \ \ 0.156431930785 \ \ 0.967909436761)$
	$\lambda_4 = (-0.0255104643303 \ \ 0.253687710095 \ \ 1.07638653524 \ \ -0.199680812427)$
5	$\varsigma_5 = (0.145193738229 \ \ 0.111411073776 \ \ 0.145193738161 \ \ 0.972329743152)$
	$\lambda_5 = (-0.0107998330767 \ \ 0.242426712526 \ \ 1.06618444995 \ \ -0.185373593007)$

5. Conclusions

This paper studies the method to solve the forward kinematics of a class of six-DOF parallel robots. A unit dual quaternion is set as the generalized coordinates of the robot system and the forward kinematic equations are derived to be a set of quadratic equations about the unit dual quaternion. These equations together with the constraint equations of the unit dual quaternion constitute the whole system of forward kinematic equations of the parallel robots.

A new algorithm is constructed for the system of quadratic equations. It possesses a quadratic convergence speed since it is the further simplification and optimization of Newton's method. The new algorithm has the advantages that the Jacobian matrix is linear to the generalized coordinates and the equations needn't be evaluated in each iteration. To make the algorithm converge to the actual solution, the convergence and singularity problems haven been discussed. We have given a convergence strategy and revealed the internal relation of the singularity to that of the parallel robot. We conclude that the algorithm is always effective as long as the robot is not in a parallel singular state. These works provide a guidance for practical applications.

The new algorithm is verified by the application to two parallel robots: 8-UPS and 8-PUS. For the 8-UPS parallel robot, the actual solution to the forward kinematics is obtain by 5 iterations within a computational error 10^{-8}. The new algorithm takes 0.2187 milliseconds while Newton's method takes 14.25 milliseconds. For the 8-PUS robot, we simulate a process of the state-feedback control. The two examples present the applications of the new algorithm and demonstrate its validity and efficiency.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant nos. 51375230 and 51575256) and the National High Technology Research and Development Program of China (Grant no. 2013AA041004).

References

[1] J.-P. Merlet, Parallel Robots, Springer Science & Business Media, 2006.
[2] V. Gough, Contribution to discussion of papers on research in automobile stability, Control and Tyre Performance, Proc. Auto Div. Inst. Mech. Eng. (1956) 392–394.
[3] K. Hunt, Structural kinematics of in-parallel-actuated robot-arms, J. Mech. Trans. Autom. Des. 105 (1983) 705–712.
[4] J.-P. Merlet, C. Gosselin, Nouvelle architecture pour un manipulateur parallèle à six degrés de liberté, Mech. Mach. Theory 26 (1991) 77–90.
[5] J. Allen, Active rack isolation system (ARIS) users guide, NASA Johnson Space Center/Boeing, Houston, TX, USA, Tech. Rep. SSP, 57006, 1999
[6] D. Stewart, A platform with six degrees of freedom, Proceed. Inst. Mech. Eng. 180, 1965, pp. 371–386.
[7] X. Huang, Q. Liao, S. Wei, Closed-form forward kinematics for a symmetrical 6-6 Stewart platform using algebraic elimination, Mech. Mach. Theory 45 (2010) 327–334.
[8] D. Gan, Q. Liao, J.S. Dai, S. Wei, L. Seneviratne, Forward displacement analysis of the general 6–6 Stewart mechanism using Gröbner bases, Mech. Mach. Theory 44 (2009) 1640–1647.
[9] C. Innocenti, Forward kinematics in polynomial form of the general Stewart platform, J. Mech. Des. 123 (2001) 254–260.
[10] J.-P. Merlet, Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis, Int. J. Robot. Res. 23 (2004) 221–235.
[11] I. Akcali, H. Mutlu, A novel approach in the direct kinematics of Stewart platform mechanisms with planar platforms, J. Mech. Des. 128 (2006) 252–263.
[12] S. Cheng, H. Wu, Y. Yao, F. Liu, Q. Miao, C. Li, J. Zhu, An analytical method for the forward kinematics analysis of 6-SPS parallel mechanisms, Jixie Gongcheng Xuebao(Chin. J. Mech. Eng.) 46 (2010) 26–31.
[13] D. Gan, Q. Liao, J. Dai, S. Wei, L. Seneviratne, Forward Displacement Analysis of a New 1CCC–5SPS Parallel Mechanism Using Gröbner Theory, Proceed. Inst. Mech. Eng. Part C: J. Mech. Eng. Sci. 223, 2009, pp. 1233–1241.
[14] Y.-J. Chiu, M.-H. Perng, Forward kinematics of a general fully parallel manipulator with auxiliary sensors, Int. J. Robot. Res. 20 (2001) 401–414.
[15] V. Parenti-Castelli, R. Di Gregorio, A new algorithm based on two extra-sensors for real-time computation of the actual configuration of the generalized Stewart-Gough manipulator, J. Mech. Des. 122 (2000) 294–298.
[16] C.-f Yang, S.-t Zheng, J. Jin, S.-b Zhu, J.-w Han, Forward kinematics analysis of parallel manipulator using modified global Newton-Raphson method, J. Cent. South Univ. Technol. 17 (2010) 1264–1270.
[17] K. Liu, F. Lewis, M. Fitzgerald, Solution of nonlinear kinematics of a parallel-link constrained Stewart platform manipulator, Circuits Syst. Sig. Process. 13 (1994) 167–183.
[18] J.-P. Merlet, Direct kinematics of parallel manipulators, IEEE Trans. Robot. Autom. 9 (1993) 842–846.
[19] P.J. Parikh, S.S. Lam, A Hybrid Strategy to Solve the Forward Kinematics problem in Parallel manipulators, IEEE Trans. Robot. 21 (2005) 18–25.
[20] J. He, H. Gu, Z. Wang, J. He, Z. Wang, Solving the forward kinematics problem of six-DOF Stewart platform using multi-task Gaussian process, ARCHIVE Proceedings of the Institution of Mechanical Engineers Part C Journal of Mechanical Engineering Science 1989–1996 (vols. 203–210), 227, 2013, pp. 161–169.
[21] J. Angeles, Is there a characteristic length of a rigid-body displacement?, Mechan. Mach. Theory 41 (8) (2006) 884–896.
[22] A. Ghasemi, M. Eghtesad, M. Farid, Neural network solution for forward kinematics problem of cable robots, J. Intell. Robot. Syst. 60 (2010) 201–215.
[23] A. Omran, M. Bayoumi, A. Kassem, G. El-Bayoumi, Optimal forward kinematics modeling of Stewart manipulator using genetic algorithms, Jordan J. Mech. Ind. Eng. 3 (2009) 280–293.
[24] P.J. Parikh, S.S. Lam, Solving the forward kinematics problem in parallel manipulators using an iterative artificial neural network strategy, Int. J. Adv. Manuf. Technol. 40 (2009) 595–606.
[25] Z. Wang, J. He, H. Shang, H. Gu, Forward kinematics analysis of a six-DOF Stewart platform using PCA and NM algorithm, Ind. Robot: Int. J. 36 (2009) 448–460.
[26] M.L. Husty, An algorithm for solving the direct kinematics of general Stewart-Gough platforms, Mech. Mach. Theory 31 (1996) 365–379.
[27] C.W. Wampler, Forward displacement analysis of general six-in-parallel SPS (Stewart) platform manipulators using soma coordinates, Mech. Mach. Theory 31 (1996) 331–337.
[28] X.Y. Yang, H.T. Wu, B. Chen, et al., Fast numerical solution to forward kinematics of general stewart mechanism using quaternion, Trans. Nanjing Univ. Aeronaut. Astronaut. 31 (4) (2014) 377–385.
[29] W. Zhou, W. Chen, H. Liu, X. Li, A new forward kinematic algorithm for a general Stewart platform, Mech. Mach. Theory 87 (2015) 177–190.
[30] J.P. Merlet, Jacobian, manipulability, condition number and accuracy of parallel robots, J. Mech. Des. 128 (2005) 199–206.
[31] F. Thomas, Approaching dual quaternions from matrix algebra, IEEE Trans. Robot. 30 (2014) 1037–1048.
[32] M.O.F.B. Mahmoud Gouasmi, Robot Kinematics Using Dual Quaternions, Int. J. Robot. Autom. (IJRA) 1 (2012) 13–30.
[33] D. Gan, Q. Liao, S. Wei, J. Dai, S. Qiao, Dual Quaternion-Based Inverse Kinematics of the General Spatial 7 R Mechanism, Proceed. Inst. Mech. Eng. Part C: J. Mech. Eng. Sci. 222, 2008, pp. 1593–1598.
[34] R. Burden, J. Faires, A. Burden, Nelson Education, Nelson Education, 2011.
[35] C. Gosselin, J. Angeles, Singularity analysis of closed-loop kinematic chains, IEEE Trans. Robot. Autom. 6 (1990) 281–290.