A Real-Time Method for Solving the Forward Kinematics of a Tripod With Fixed-Length Legs

Younan Xu
School of Mechanical & Electrical Engineering,
East China Jiaotong University,
Nanchang, Jiangxi 330013 PRC
e-mail: xyn@ecjtu.jx.cn

Fengfeng Xi[1]
Department of Aerospace Engineering,
Ryerson University,
Toronto, ON M5B 2K3, Canada
e-mail: fengxi@ryerson.ca

This paper presents a real-time method for solving the forward kinematics of a tripod with fixed-length legs. The basic idea is to model the problem at hand based on a spatial four-bar linkage through which three sliding legs can be interrelated by choosing one link as a driving variable and other two links as driven variables. As a result, the original multivariable nonlinear problem with three variables can be reduced to one variable problem. A complete approach is provided to solve the unitary nonlinear programing problem. This includes a method for solving the implicit functions in terms of the driving and driven variables, and an approximation method for selecting an initial value leading to a fast solution. The simulation results show that (i) the method is effective, (ii) can reach very accurate results within five iterations for an error bound of 10^{-10}*, and (iii) numerically very stable. The experiment results show that the proposed forward kinematic method is fast enough to be implemented in real time to provide an accurate prediction of the tool pose from the joint encoder measurement.*
[DOI: 10.1115/1.2114928]

1 Introduction

Parallel kinematic machines (PKMs) represent the most recent radical change in machine-tool history. Unlike traditional machine tools, which are based on the serial structure, PKMs are developed utilizing the parallel structure. The initial development of PKMs involved simply inverting the *Gough-Stewart platform*, which is a prismatic parallel mechanism with extensible legs. Because these PKMs have six legs, they are called hexapods. Examples of this type of hexapods include *VARIAX* from Gidding & Lewis; *Tornado* from Hexel Corp; and Geodetic from Geodetic Technology Ltd. Later development resulted in the hexapods with fixed-leg lengths. Examples of this type of hexapods include *Hexaglide* of the Swiss Federal Institute of Technology, *LINAPID* of Stuttgart University, and *HEXAM* of Toyoda [1].

Since machining operation requires a maximum of five axes, recent development has focused on tripods, i.e., three-axis PKMs. Examples include *Triaglide* and *Tricept* [2]. Tripods can be combined with two-axis serial systems, such as *x-y* stages, to form five-axis machines that can offer advantages from both serial and parallel structures [1].

Kinematics plays an important role in PKM motion control. The inverse kinematics solves the actuator motions based on that of the moving platform. For parallel mechanisms, the inverse kinematics is relatively easy to solve, as usually there exist closed-form solutions. However, it is relatively difficult to obtain closed-form solutions for the forward kinematics that solves the motion of the moving platform based on the actuator motions. For machining applications, the forward kinematics is particularly useful for predicting the position and orientation of the tool tip or even forming a Cartesian-based control system [1].

The forward kinematics is a classical problem that has been studied for many years. The methods archived in the literature generally can be classified into analytical method and numerical method. The former include closed-form solutions [3,4] and polynomial equations [5–7]. The numerical method includes linear and nonlinear programing [8–11]. Each of these methods has advantages and limitations compared to the others. The closed-form solutions only exist for very simple structures. Though polynomial equations would provide all the solutions, they are usually in higher orders and difficult to solve. The nonlinear programing method is easy for modeling, but may encounter convergent problems. The linear programing method is based on the Jacobian Matrix, and it may have stability problems at singular configurations [12].

Recent papers [6,7] focused on the methodology that solves the forward kinematics for 3-PRS and 3-RS mechanisms, respectively. They used Bezout's elimination method to solve 3-PRS and 3-RS nonlinear constraint functions and derived 16th-order polynomials with only one variable solved numerically. In [13], Merlet proposed a method to reduce this problem to eighth-order polynomials, and also solved numerically. Since these methods generate all possible solutions, an elimination procedure is required to delete imaginary solutions while retaining real ones.

In this paper a new method is presented for real-time implementation of the forward kinematics for a tripod with fixed-length legs. The basic idea is to model the problem at hand based on a spatial four-bar linkage through which three sliding legs can be interrelated by choosing one link as a driving variable and the other two links as driven variables. As a result, the original multivariable nonlinear problem with three variables can be reduced to one variable problem, a similar idea to [6,7,13]. The difference, however, is that the proposed method solves second order polynomials with only real solutions that are sufficiently fast for real-time applications. In what follows, a complete approach is described, including a method for solving the implicit functions in terms of the driving and driven variables, and an approximation method for selecting an initial value leading to a fast solution. A number of tests are carried out to show the effectiveness of the method for real-time implementation.

2 Forward Kinematics Modeling

2.1 Hybrid Machine. The system under consideration is a tripod (parallel structure) with three fixed-length legs that can be combined with an *x-y* stage (serial structure) to form a five-axis machine. Figure 1(*a*) shows the design of the hybrid machine

[1]Corresponding author who completed this work while visiting the East China Jiaotong University.

Contributed by the Manufacturing Engineering Division of ASME for publication in the JOURNAL OF MANUFACTURING SCIENCE AND ENGINEERING. Manuscript received April 9, 2004; final manuscript received April 4, 2005. Review conducted by G. Wiens.

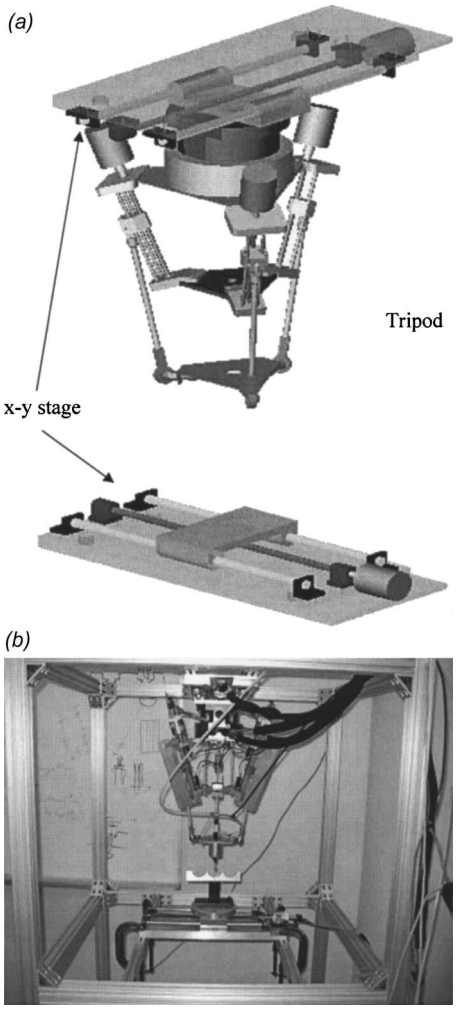

Fig. 1 (*a*) Design model of the hybrid machine and (*b*) Prototype of the hybrid machine at Ryerson University

developed at Ryerson University. This machine has been constructed and fully implemented under computer control, as shown in Fig. 1(*b*). For manufacturing applications, such as polishing, the real-time forward kinematics is an important enabling technology. With this technology, the pose of a tool can be determined in real time from online joint encoder measurement. The manufacturing functions arising from this technology include part-tool alignment and part profile measurement (reverse engineering). These functions have been implemented on the hybrid machine developed at Ryerson University. The key to this accomplishment is the real-time forward kinematics method, which is to be described in the following sections. The kinematics analysis of an *x-y* stage is trivial.

2.2 Notation for Modeling the Tripod. As shown in Fig. 2, the tripod consists of a moving platform (MP) to which a tool is attached and three legs sliding along the guideways that are mounted on the support structure, including the base platform (BP). Each leg is connected at one end to the guideway by a revolute joint and at the other end to the MP by a spherical joint. The coordinate system used is given in Fig. 2. The fixed global coordinate system O-xyz is attached to the BP, and a local coordinate system C-$x'y'z'$ is attached to the MP. The symbols used are given in Table 1.

2.3 Problem and Its Equations. The problem of the forward kinematics can be stated as the input variables, $s_i (i=1,2,3)$, are known, the output variables, $p_i (i=1,2,3)$, need to be solved. Once p_i is obtained, the position and orientation of the moving platform can be readily determined using the three-point method.

Referring to Figure 2, p_i can be expressed as

$$p_i = b_i + s_i + l_i \quad (\text{for } i = 1-3) \tag{1}$$

and l_i can be expressed as

$$l_i = \{-l_i \cos(\beta_i + \beta_{i0})\cos\alpha_i \; -l_i \cos(\beta_i + \beta_{i0})\sin\alpha_i \; l_i \sin(\beta_i + \beta_{i0})\} \quad (\text{for } i = 1-3) \tag{2}$$

It can be seen from Eqs. (1) and (2) that solving p_i, in fact, requires solving β_i, because the rest are either given or constants.

Furthermore, p_i is confined in the moving platform defined as

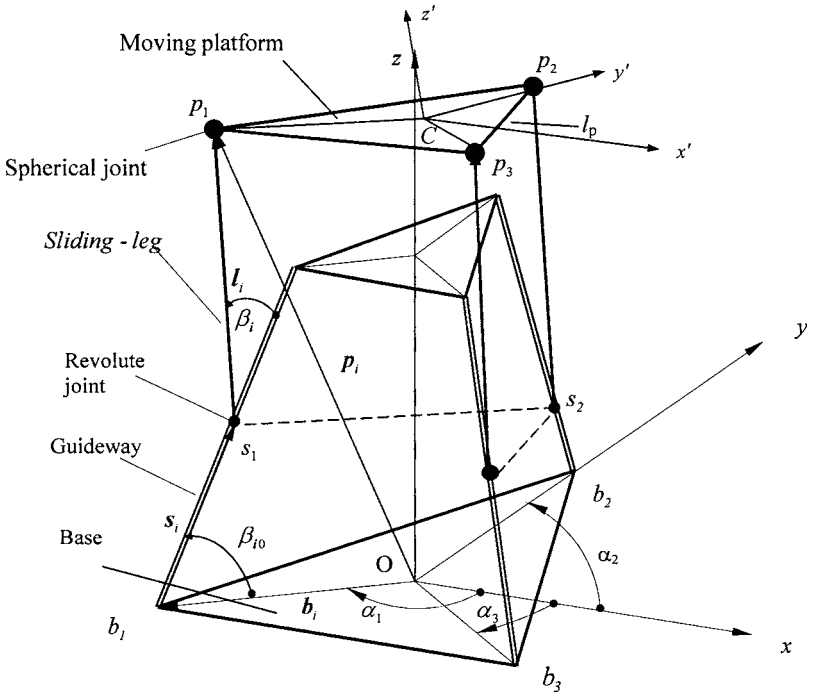

Fig. 2 Kinematic model for a tripod with fixed-length legs

Table 1 Definition of the symbols used in Fig. 1

Symbol	Definition
b_i	Indicates the position of one end of the ith guideway attached to the base
s_i	Indicates the position of the ith revolute joint
p_i	Indicates the position of the ith spherical joint
$\boldsymbol{b}_i$	Vector from O to b_i
$\boldsymbol{s}_i$	Vector from b_i to s_i
$\boldsymbol{l}_i$	Vector from s_i to p_i
$\boldsymbol{p}_i$	Vector from O to p_i
α_{I}	Represents the relative angle between the ith guideway and the x axis
β_{I}	Represents the angle between the ith sliding-leg and the ith guideway
β_{io}	Represents the angle between the ith guideway and the BP

$$\|p_i - p_j\| = l_p \quad (\text{for } i,j = 1-3, \quad j \neq i) \tag{3}$$

where l_p is the side length of the triangle plate of the moving platform.

If Eq. (2) is substituted into Eq. (1), which is then substituted into Eq. (3), then this will yield three nonlinear equations in terms of three unknowns β_i. Though conventional nonlinear programing methods could be used to solve this problem, a more effective method is presented here that utilizes the special features of the tripod.

2.4 Problem Simplification. Inspired by the way of solving the planar four-bar linkages, two spatial four-bar linkages $s_2p_2p_1s_1$ and $s_2p_2p_3s_3$, are introduced and they are indicated by the dashed lines in Fig. 2. Here s_2p_2 is assumed to be the driving bar, and p_1s_1 and p_3s_3 are driven by it. With this approach, the output angle β_1 and β_3 can be expressed as a function of input angle β_2; that is,

$$\beta_1 = \beta_1(\beta_2) \tag{4}$$

$$\beta_3 = \beta_3(\beta_2) \tag{5}$$

Equations (4) and (5) are implicit functions, which will be discussed in Sec. 3.

To obtain β_2, it is only required to solve the following one of the three equations given in Eq. (3); that is, $i=1$, $j=3$

$$\|p_1 - p_3\| = l_p \tag{6}$$

Once β_2 is solved, β_1 and β_3 can be determined from Eqs. (4) and (5), and subsequently all three p_i can be solved. With this formulation, instead of solving three nonlinear equations, it solves one nonlinear equation.

2.5 Programing Model. To this end, a unitary nonlinear programing model is defined for the forward kinematics in terms of β_2

$$\min_{x=\beta_2} f(x) = (\|p_1 - p_3\| - l_p)^2 \tag{7}$$

where

$$p_i = p_i(\beta_i) = p_i(\beta_2), \quad \beta_i = \beta_i(\beta_2) \quad (\text{for } i = 1 \text{ and } 3) \tag{8}$$

subjected to

$$0^\circ < \beta_{2,\min} \leqslant \beta_2 \leqslant \beta_{2,\max} < 180^\circ \tag{9}$$

where $\beta_{2,\min}$ and $\beta_{2,\max}$ represent the angle range of the second sliding leg.

Note that Eq. (7) is in the form of second-order polynomials. The main steps for programing are given as follows in quasi codes:

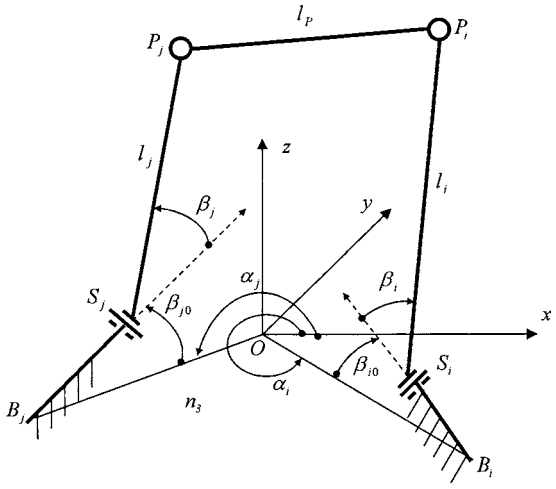

Fig. 3 Spatial four-bar linkage model of the tripod

Begin forward-kinematic-method algorithm:
- Define s_1, s_2, and s_3 as input variables;
- Define p_1, p_2, and p_3 as output variables;
- Assign initial values to s_1, s_2, and s_3;
- Set the iteration time k to 0, i.e., let $k=0$;
- Compute initial value $\beta_2^{(0)}$ (method given in Sec. 4);
- Begin iteration-outer-loop
  - Compute $\beta_1^{(k)}$ and $\beta_3^{(k)}$ based on Eqs. (4) and (5) (method given in Sec. 3);
  - Compute $p_1^{(k)}$ and $p_3^{(k)}$ based on Eqs. (1) and (2);
  - Compute $f(x^{(k)})$ based on Eq. (7);
  - Check result: if $f(x^{(k)})$ is less than or equal to $f_{\max}$, then finish outer-loop iteration;
  - Select $\Delta\beta_2^{(k)}$ [set to 0.1 deg first, then is adjusted according to $f(x^{(k)})$];
  - Begin iteration-inner-loop
    - Compute $\beta_2^{(k+1)}$ using the equation $\beta_2^{(k+1)} = \beta_2^{(k)} + \Delta\beta_2^{(k)}$;
    - Check constrains: if $\beta_2^{(k+1)}$ satisfies Eq. (9), then finish inner-loop iteration;
    - Reset $\Delta\beta_2^{(k)}$;
  - End iteration-inner-loop
  - Reset the iteration-time k to $k+1$, i.e., let $k=k+1$;
- End iteration-outer-loop
- Output $\beta_1^{(k)}$, $\beta_2^{(k)}$, and $\beta_3^{(k)}$ as final results;
- Determine output variables p_1, p_2, and p_3 based on Eqs. (1) and (2);
- Determine the pose of the tripod using the three-point method;

End of forward-kinematic-method algorithm.

3 Analysis of Spatial Four-Bar Linkage

The purpose here is to relate β_1 and β_3 to β_2 as implicitly expressed in Eqs. (4) and (5). As mentioned before, the concept of a spatial four-bar linkage is used.

3.1 Spatial Four-Bar Linkage Model. Figure 3 represents a spatial four-bar linkage $S_jP_jP_iS_i$, where S_i and S_j are rotation joints, and P_i and P_j are spherical joints. S_jP_j is a driving bar, and S_iP_i is a driven bar. The lengths of the bars are l_j,l_p, and l_i. The position S_i and S_j, and input angle β_j are all known. The output angle β_i is to be determined.

It is known that the degrees of freedom (DOF) of this mechanism is equal to 1, so β_i can be expressed as

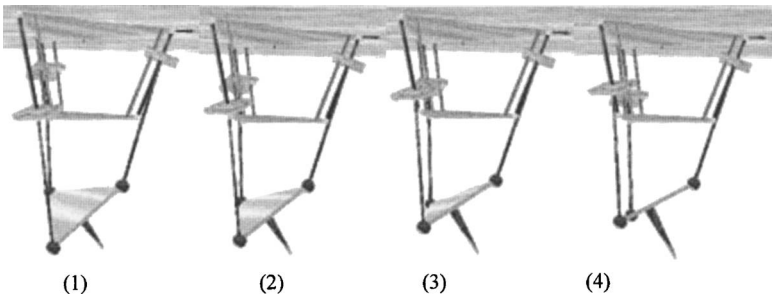

Fig. 4 Relationship between β_2 and the pose of MP

$$\beta_i = \beta_i(\beta_j) \tag{10}$$

Equation (10) is a general form of Eqs. (4) and (5).

By substituting Eq. (1) into Eq. (3), yielding

$$\|b_i + s_i + l_i - p_j\| = l_p \tag{11}$$

Let

$$m_{ij} = b_i + s_i - p_j = m_{ij}(\beta_j) \tag{12}$$

Then Eq. (11) becomes

$$\|l_i + m_{ij}\|^2 = l_p^2 \tag{13}$$

where $m_{ij}=\{m_{ijx}, m_{ijy}, m_{ijz}\}$, and m_{ij} is an explicit function of β_j. Substituting Eq. (2) into Eq. (13) results in

$$l_i^2 + m_{ij}^2 - 2m_{ijx}l_i\cos(\alpha_i)\cos(\beta_i+\beta_{i0}) - 2m_{ijy}l_i\sin(\alpha_i)\cos(\beta_i+\beta_{i0}) + m_{ijz}l_i\sin(\beta_i+\beta_{i0}) = l_p^2 \tag{14}$$

Equation (14) is the equation that only β_i is to be solved.

3.2 Solving the Equations. Set

$$\left.\begin{aligned} a &= -2m_{ijx}l_i\cos(\alpha_i) - 2m_{ijy}l_i\sin(\alpha_i) \\ b &= 2m_{ijz}l_i \\ c &= l_p^2 - l_i^2 - m_{ij}^2 \end{aligned}\right\} \tag{15}$$

Rewriting Eq. (14) leads to

$$a\cos(\beta_i+\beta_{i0}) + b\sin(\beta_i+\beta_{i0}) = c \tag{16}$$

Solving Eq. (16) gives the following solutions:

$$\left.\begin{aligned} \varphi &= tg^{-1}(b/a) \in [0,\ 360) \\ \chi &= \cos^{-1}\left(\frac{c}{\sqrt{a^2+b^2}}\right) \in (0,180] \\ \beta_i &= \varphi - \beta_{i0} \pm \chi \in (-180,180] \end{aligned}\right\} \tag{17}$$

There are two solutions to $\beta_i(\beta_{i1}, \beta_{i2})$. Here, a larger value of the two is selected based on the reason given in Appendix B

$$\beta_i = \max(\beta_{i1}, \beta_{i2}) \tag{18}$$

3.3 Computation Steps. When β_j is given, the steps to compute β_i are summarized as follows:

1. Given β_j
2. Compute l_j based on Eq. (2)
3. Compute p_j based on Eq. (1)
4. Compute m_{ij} based on Eq. (12)
5. Compute β_i based on Eqs. (15)–(18)

4 Determination of Initial Value $\beta_2^{(0)}$

4.1 Approximation Solution. Selecting an initial value of $\beta_2^{(0)}$ closer to final β_2 can reduce iteration times for solving the nonlinear programing defined in Eqs. (7)–(9). To do so, a special position of *MP* is chosen, in which the second leg is located on its real position, while the positions of the first sliding-leg and the third's are at the average value of the two, i.e.,

$$\left.\begin{aligned} s_2^{(0)} &= s_2 \\ s_1^{(0)} &= s_3^{(0)} = \tfrac{1}{2}(s_1+s_3) \end{aligned}\right\} \tag{19}$$

where the superscript 0 represents an initial value, and $s_i^{(0)}$ represents the initial value of s_i.

The reason for choosing the average value of s_1 and s_3 is that β_2 changes little. Figure 4 shows four poses where the average value of s_1 and s_3 is the same. As you can see from Fig. 4, β_2 has little change. Also, this approximate position is easy to analyze.

To solve the forward kinematic model, the spatial four bar linkage model needs to be analyzed, but the conditions are different from the ones introduced in Sec. 4, and the method is discussed in Sec. 4.2.

4.2 Spatial Four-Bar Linkage Model. According to the condition in Eq. (19), at the approximate position, the following holds:

$$\beta_3^{(0)} = \beta_1^{(0)} \tag{20}$$

In light of Eqs. (1)–(3), the approximate position can be expressed as

$$\begin{aligned} \boldsymbol{l}_i^{(0)} &= \{-l_i\cos(\beta_i^{(0)}+\beta_{i0})\cos\alpha_i \\ &\quad - l_i\cos(\beta_i^{(0)}+\beta_{i0})\sin\alpha_i l_i\sin(\beta_i^{(0)}+\beta_{i0})\} \\ p_i^{(0)} &= b_i + s_i^{(0)} + \boldsymbol{l}_i^{(0)} \quad (\text{for } i=1,3) \\ \|p_1^{(0)} - p_3^{(0)}\| &= l_p \end{aligned} \tag{21}$$

where $p_i^{(0)}$ is the approximate position to p_i, and $\boldsymbol{l}_i^{(0)}$ is the approximate vector to l_i Equation (21) is the equation for the spatial four bar linkage model. By solving it, $\beta_1^{(0)}$ and $\beta_3^{(0)}$ can be obtained, subsequently, $\beta_2^{(0)}$ can be obtained by using the method introduced in Sec. 3.

4.3 Solving the Equations. Let

$$q = \{q_x, q_y, q_z\} = (b_3 + s_3^{(0)}) - (b_1 + s_1^{(0)}) \tag{22}$$

Simplifying Eq. (21) yields

$$\|\{q_x,0,0\} + \{-\sqrt{3}l_1\cos(\beta_1^{(0)}+\beta_{10}),0,0\}\| = l_p \tag{23}$$

It is obvious that

$$q_x > \sqrt{3}l_1\cos(\beta_1^{(0)}+\beta_{10}) \tag{24}$$

Solving Eq. (23) results in

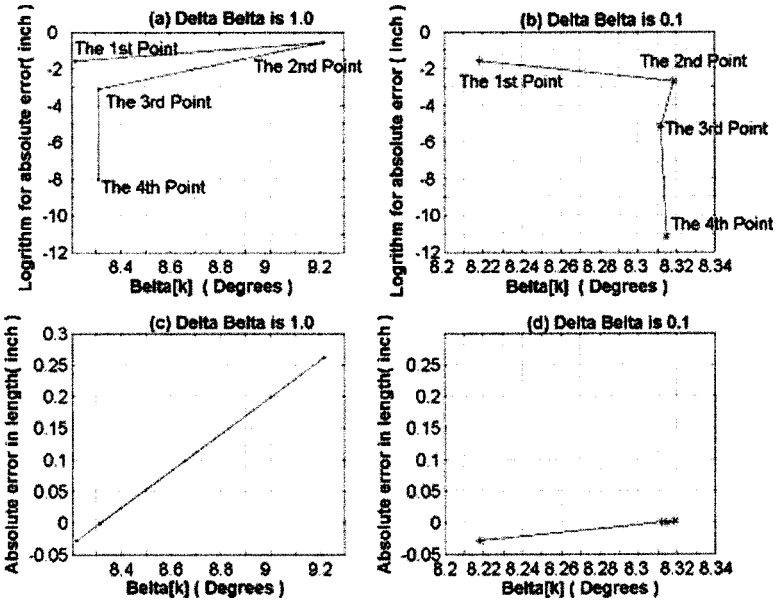

Fig. 5 Convergence for computing β_2

$$\left.\begin{aligned}\psi = \cos^{-1}\left(\frac{q_x - l_p}{\sqrt{3}l_1}\right) \in [0,180]\\ \beta_1 + \beta_{10} = \pm\psi\end{aligned}\right\} \tag{25}$$

Thus,

$$\beta_1 = \pm\psi - \beta_{10} \tag{26}$$

This leads two answers, and the positive operating sign should be chosen. The reason is given in Appendix B.

4.4 Computation Steps.

1. Assign $s_1^{(0)}$, $s_2^{(0)}$, and $s_3^{(0)}$ based on Eq. (19)
2. Compute q using Eq. (22)
3. Compute $\beta_1^{(0)}$ using Eqs. (25) and (26)
4. Determine $\beta_3^{(0)}$ using Eq. (20)
5. Determine $\beta_2^{(0)}$ as described in Sec. 3

5 Testing

A number of tests are carried out on the tripod available at Ryerson University with the sizes given in Appendix A. These tests include convergence, stability, accuracy from forward to inverse, and accuracy from inverse to forward.

5.1 Convergence. In this test, several test points are chosen, but because of space limitations only two results of computing β_2 are shown in Fig. 5. In Figs. 5(a) and 5(b), the polylines are drawn in logarithm scale, and in Figs. 5(c) and 5(d), the straight lines are drawn in linear scale. The left-most points in each figure are the initial points. Along each line, there are four points, namely, $(\beta_2^{(0)},\varepsilon_2^{(0)})$, $(\beta_2^{(1)},\varepsilon_2^{(1)})$, $(\beta_2^{(2)},\varepsilon_2^{(2)})$, and $(\beta_2^{(3)},\varepsilon_2^{(3)})$, representing the initial, first iteration, second iteration, and third iteration, respectively. The last point is the final solution.

In Figs. 5(a) and 5(c), $\Delta\beta_2^{(0)}$ is set to 1.0 and the initial error is −0.028. After the first iteration, the error becomes bigger at 0.265. However, after the line-acceleration is applied for the second iteration, the error is reduced to 1.0×10^{-3}. Furthermore, after the parabola-acceleration is applied for the third iteration, the error is further reduced to 1.0×10^{-7}, taken as the final solution.

In Figs. 5(b) and 5(d), $\Delta\beta_2^{(0)}$ is set smaller at 0.1 and the initial error is the same as the first case. After the first iteration, the error becomes smaller at 0.0025. After the line acceleration is applied for the second iteration, the error is reduced to 1.0×10^{-5}. After the parabola acceleration is applied for the third iteration, the error is further reduced to 1.0×10^{-11}, taken as the final solution.

From the test results given above, the following observations are obtained. First, the first iteration may degrade the solution; that is, the second point $(\beta_2^{(1)},\varepsilon_2^{(1)})$ may be worse than the initial point $(\beta_2^{(0)},\varepsilon_2^{(0)})$. However, the acceleration algorithms (line and parabola) would quickly improve the results. In order words, the third point $(\beta_2^{(2)},\varepsilon_2^{(2)})$ is much better than the previous solutions, and the fourth point $(\beta_2^{(3)},\varepsilon_2^{(3)})$ is much better than the third point $(\beta_2^{(2)},\varepsilon_2^{(2)})$. Second, the initial value has a great influence on the results. However,$\Delta\beta_2^{(0)}=0.1$ is recommended to achieve a good convergence regardless of initial values.

The convergence time tested on a PC of Inter P4 2.8 GHz CPU, 256 MB DDR Memory is 24.45 μs using Borland C++ Builder 6.0. This time would vary when it is tested under different programing languages or machines.

5.2 Stability. The test on stability is to investigate how the proposed method behaves at singular configurations. To do so, singular configurations should be chosen. One such configuration is to set $s_1=s_3$ for the first and the second sliding legs to form a plane. As shown in Fig. 6, a singular configuration occurs when the moving platform is coplanar with the aforementioned plane. At this configuration, even given $\omega_y\neq0$, $\dot{s}_1=\dot{s}_2=\dot{s}_3=0$, i.e., the tripod is singular. If the Jacobian-based numeric method were used, it would encounter singularity problem, leading to instable solutions. For the proposed method, it performs very quickly and stably because in this case the initial values are the final solutions, and there is no need for iteration at all.

5.3 Accuracy From Forward to Inverse. In Sec. 5.1, it has been shown that the proposed method requires only four to five iterations to reach convergent solutions with a tolerance of 10^{-10}. In this test, accuracy is discussed, first by looking at the problem from forward kinematics to inverse kinematics. Here, s_i is given first, and then p_i is determined by the proposed method. After that, the inverse kinematics is used to determine s_i'. The absolute error is defined as

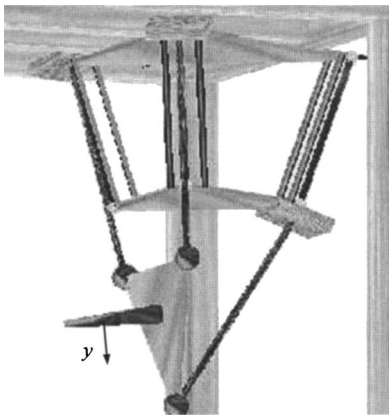

Fig. 6 Singular configuration for testing

$$\varepsilon_i = |s_i' - s_i| \quad (\text{for } i = 1-3) \tag{27}$$

where ε_i represents the absolute error of s_i; because the error is relatively small, Eq. (27) is defined in logarithmic form

$$\xi_i = \log(\varepsilon_i) = \log(|s_i' - s_i|) \text{ (for } i = 1-3) \tag{28}$$

In Eq. (28), ξ_i represents the logarithmic absolute error of s_i.

Simulations are carried out and the results are shown in Fig. 7. Here, the logarithm for absolute error is shown. The motion range of s_i is 6 in., and the error is expressed with respect to position *ID* numbered from 0 to 20.

In Figs. 7(*a*)–7(*c*) the error for termination of iterations is $[f_{\max}]=10^{-6}$. The number of the average iterations is 4.5. In Fig. 6(*d*) $[f_{\max}]=10^{-10}$, the number of the average iteration times is 4.8. For Figures 7(*a*)–7(*d*), s_i is evaluated by considering the range of the other two. In other words, the whole evaluations are done by considering the entire work space of the tripod. As it can be seen, the method is effective and can reach accurate results within five iterations.

5.4 Accuracy From Inverse to Forward. In this test, the pose of the moving platform $\boldsymbol{p}_t$ and $\boldsymbol{n}_t$ are given first, where $\boldsymbol{p}_t$ and $\boldsymbol{n}_t$ denote the position and normal vectors of the tool attached to the moving platform. Using the inverse kinematics method [1], s_i can be determined first. After that using the proposed method, $\boldsymbol{p}_i'$ is determined. Finally, the three-point method is applied to determine $\boldsymbol{p}_t'$ and $\boldsymbol{n}_t'$. The absolute error is defined as

$$\varepsilon_t = \|\boldsymbol{p}_t' - \boldsymbol{p}_t\| \tag{29}$$

where ε_t represent the absolute error for the tool position. Because this error is very small, it is also defined in logarithmic form

$$\xi_t = \log(\varepsilon_t) = \log(\|\boldsymbol{p}_t' - \boldsymbol{p}_t\|) \tag{30}$$

were ξ_t represent logarithm for the absolute error ε_t.

In the test, the tool path used is set as

$$\left.\begin{aligned} R &= 6 \\ n_{tx}^{(i)} &= \cos(\varphi_i)\cdot\sin(\psi_i) \\ n_{ty}^{(i)} &= \sin(\varphi_i)\cdot\sin(\psi_i) \\ n_{tz}^{(i)} &= \cos(\psi_i) \\ \mathbf{p}_t^{(i)} &= -R\cdot\mathbf{n}_t^{(i)} + \{0\ 0\ 20\} \\ \varphi_i &= i\cdot 2 \text{ deg } (i = 0,1,\ldots,360) \\ \psi_i &= (360 - i\cdot)0.125 \text{ deg } (i = 0,1,\ldots,360) \end{aligned}\right\} \tag{31}$$

This is, in fact, the outer surface of a ball. Here, $\boldsymbol{n}_t^{(i)} = \{n_{tx}^{(i)} n_{ty}^{(i)} n_{tz}^{(i)}\}$ represents the tool direction vector, it is opposite to the normal vector of the surface, $\boldsymbol{p}_t^{(i)}$ represents the *i*th point located on the surface, φ_i is the angle around *z* axis, ψ_i is the angle between $\boldsymbol{n}_t^{(i)}$ and *z* axis. The error for termination of iterations is

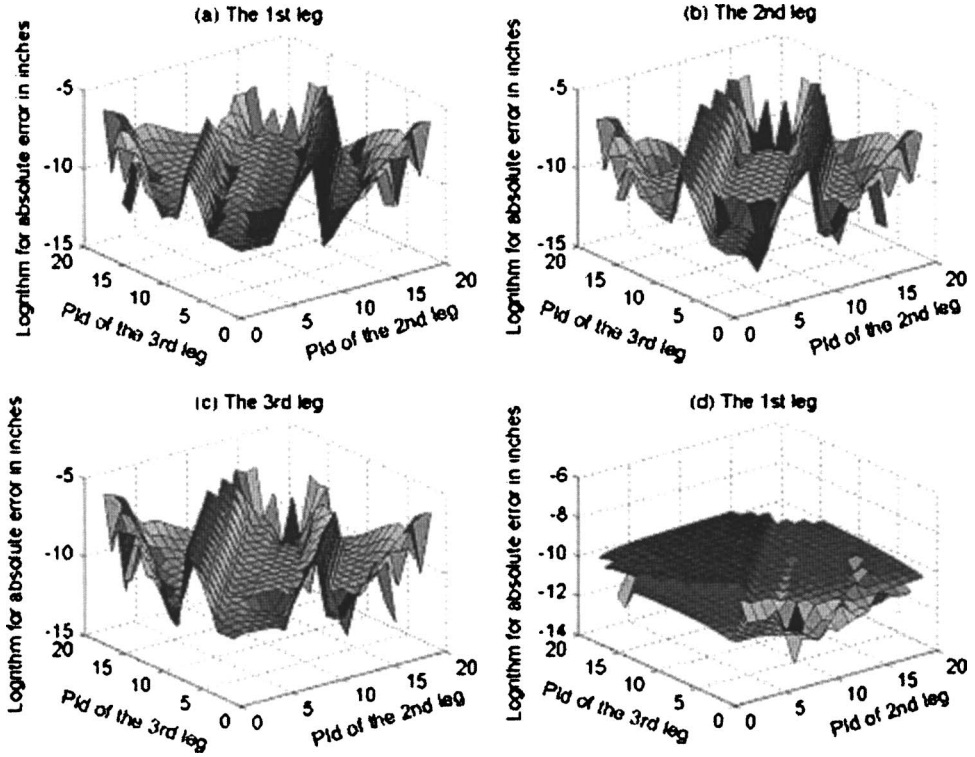

Fig. 7 Results for forward to inverse

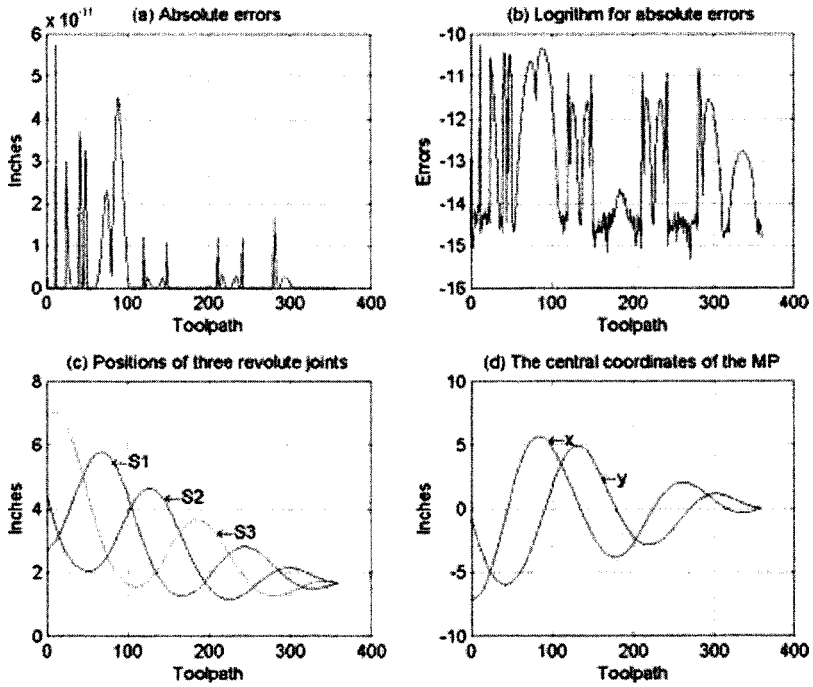

Fig. 8 Results for inverse to forward

set at 10^{-10}.

Figure 8 shows the results for this testing. Here the tool path is represented by point *ID* along the tool path numbered from 0 to 360, and in total there are 361 points. Figure 8(*a*) shows the absolute error along the given tool path. Figure 8(*b*) is the logarithm of the absolute error. It can be seen that the maximum error is within the set error bound of 10^{-10}. Figure 8(*c*) shows the position of the three revolute joints. Figure 8(*d*) shows the relative movements of the central point of the moving platform.

5.5 Experiment. The proposed method has been programed in C and implemented in the hybrid machine (tripod plus an *x*-*y* stage) available at Ryerson University. The program is used to compute the pose of the tool tip in real time based on the joint encoder measurement.

The experiment reported here was done by planning a real tool path as shown in Fig. 9. The tool is moving around a circle while keeping an angle of 15 deg from the vertical direction. As shown in Fig. 9, it forms a cone shape. The tool tip is pointing down. The smaller circle is the path of the tool tip that represents the tool position, whereas the larger circle is the path of the platform center. The straight lines represent the tool orientation.

The inverse kinematics method is used to compute the command joint variables, based on which the joints controllers drive the machine. The forward kinematics method is used to estimate the error in the pose of the moving platform.

Figure 10 shows the experiment results. Figure 10(*a*) is the command joint positions corresponding to the tool path in Fig. 9. Figure 10(*c*) represents the errors for all joints, i.e., the difference between the command values and the encoder reading. Figure 10(*b*) represents the absolute errors for the tool position, and Fig. 10(*d*) represents the absolute errors for the tool orientation. The actual pose of the tip is computed online using the proposed forward kinematic method and encoder measurement data. The sampling frequency is at 1 KHz. The results are compared to the planned tool path to determine the errors. As it may be seen from Fig. 9, the pose error is in the same magnitude of the joint errors, indicating that the major source of the error comes from the joints. In order words, the proposed forward kinematic method is fast enough to be implemented in real time to provide accurate results for predicting the tool pose.

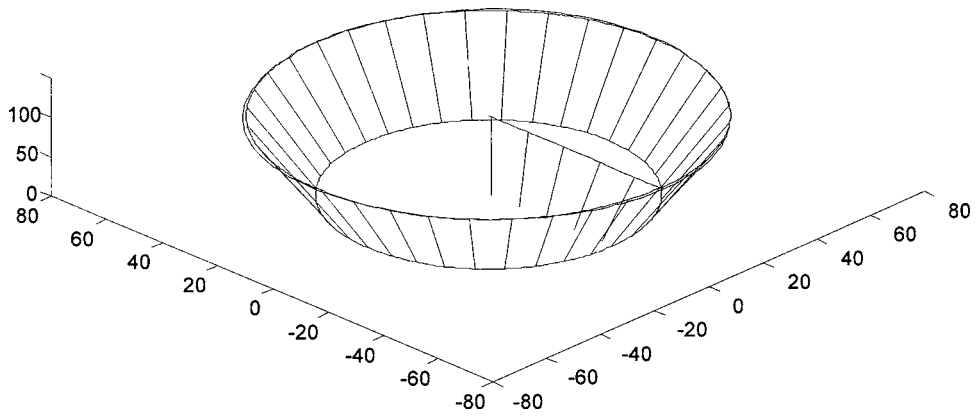

Fig. 9 Tool path for experiment test

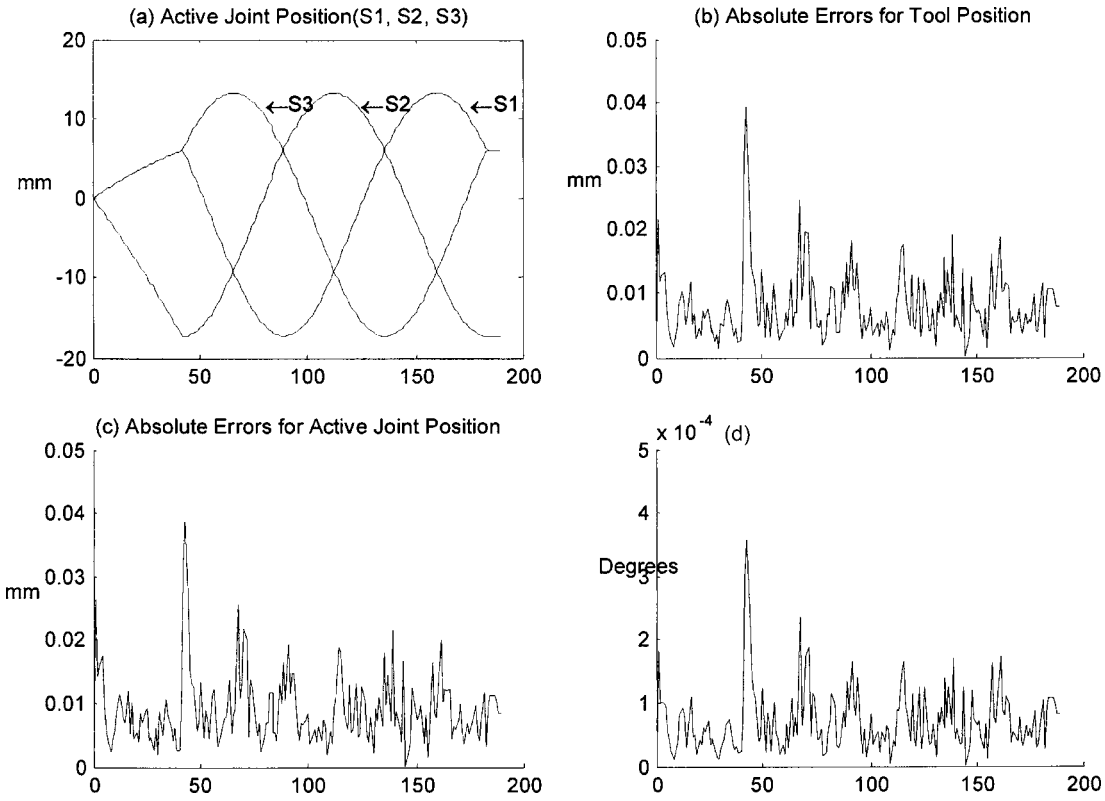

Fig. 10 Test results for the tripod at Ryerson University

6 Comparison With Other Methods

In this section, the method proposed in this paper is compared to the methods proposed by Merlet [13] and Zhang [14]. All the three methods follow the same idea, i.e., converting the underlying problem into one variable. However, the equations used in the first method are second-order polynomials with one explicit and two implicit variables, whereas the last two methods are in eighth-order polynomials with only one explicit variable. Though all the three methods are solved numerically, the solutions obtained from the last two methods contain all possible solutions, including imaginary ones. The first method is only aimed at finding the real solutions. Nevertheless, it can also find all possible solutions, when needed, by choosing another set of signs in Eqs. (17) and (25).

Computationally speaking, the first method is superior to the last two. This can be understood by counting the number of operations from Eqs. (1)–(26). To perform a forward kinematics analysis, 10 triangle functions, 50 multiply or division operations, and 5 square-root operations are calculated. Note that in Eq. (15), $l_i \cos(\alpha_i)$ and $l_i \sin(\alpha_i)$ only need to be calculated one time. For the last two methods, the resulting equations are extremely lengthy, expressed in terms of ~270 multiplication operations. After some simplification, it still needs ~60 multiplication and 8 triangle functions to calculate the first set of parameters. After that, an eighth-order polynomial equation must be solved numerically. Obviously the method proposed in this paper is computationally less complex and, therefore, more efficient—ideal for real-time applications [15].

7 Concluding Remarks

In this paper, a real-time computational method is developed for the forward kinematics analysis of a sliding-leg tripod. Through the introduction of a spatial four-bar linkage model, three sliding legs can be interrelated by choosing one as a driving variable and other two are driven variables. As such, the original multivariable nonlinear problem with three variables is reduced to one-variable problem. A complete approach is provided to solve the unitary nonlinear programing. The methods are provided to solve the implicit functions in terms of the driving and driven variables. An approximation method is put forward to select an initial value leading to a faster solution. A number of tests are carried out and the results have shown that (i) the method is effective, (ii) can reach very accurate results within five iterations, and (iii) numerically stable.

The method has been implemented on the tripod at Ryerson University for various applications, including prediction of the tool tip, part alignments, and part profile measurement. The experiment results show that the proposed forward kinematic method is fast enough to be implemented in real time with accurate results. The proposed method is also suitable for direct kinematic method for other types of tripods, including 3-RPS with fixed or nonfixed leg length, or 3-RS mechanisms, and applicable to 6-degree-of-freedom parallel mechanisms, in general.

Acknowledgment

The authors would also like to thank Steven Yang of Ryerson University for his assistance in the experiment.

Appendix A: Tripod Sizes Used in This Paper

1. **Sizes of Tripod**

$$l_p = 6.0 \text{ in.} \tag{A1a}$$

$$l_i = 8.67 \text{ in.} \quad (\text{for } i = 1-3) \tag{A1b}$$

$$l_g = 6.0 \text{ in.} \tag{A1c}$$

$$\beta_0 = \beta_{i0} = 20 \text{ deg} \quad (\text{for } i = 1-3) \tag{A1d}$$

$$l_b = 6.0 + \tfrac{1}{2} l_g \sin(\beta_0) = 10.444 \text{ in.} \tag{A1e}$$

where l_b and l_p are the side length of the triangle plates of the base and moving platform, respectively. l_g and l_i are the lengths of the ith guideway and the sliding legs. β_{i0} is the angle between the ith guideway and the base platform.

2. **Position of Base Platform (BP)**

$$\boldsymbol{b}_1 = \begin{bmatrix} b_{1x} \\ b_{1y} \\ b_{1z} \end{bmatrix} = \begin{bmatrix} \dfrac{-l_b}{2} \\ \dfrac{-\sqrt{3} l_b}{6} \\ 0 \end{bmatrix} \tag{A2a}$$

$$\boldsymbol{b}_2 = \begin{bmatrix} b_{2x} \\ b_{2y} \\ b_{2z} \end{bmatrix} = \begin{bmatrix} 0 \\ \dfrac{\sqrt{3} l_b}{3} \\ 0 \end{bmatrix} \tag{A2b}$$

$$\boldsymbol{b}_3 = \begin{bmatrix} b_{3x} \\ b_{3y} \\ b_{3z} \end{bmatrix} = \begin{bmatrix} \dfrac{l_b}{2} \\ \dfrac{-\sqrt{3} l_b}{6} \\ 0 \end{bmatrix} \tag{A2c}$$

3. **Position of Moving Platform (MP)**

Measured at local coordinate $C-x'y'z'$ attached to MP (see Fig. 2)

$$\boldsymbol{p}'_1 = \begin{bmatrix} p'_{1x} \\ p'_{1y} \\ p'_{1z} \end{bmatrix} = \begin{bmatrix} \dfrac{-l_p}{2} \\ \dfrac{-\sqrt{3} l_p}{6} \\ 0 \end{bmatrix} \tag{A3a}$$

$$\boldsymbol{p}'_2 = \begin{bmatrix} p'_{2x} \\ p'_{2y} \\ p'_{2z} \end{bmatrix} = \begin{bmatrix} 0 \\ \dfrac{\sqrt{3} l_p}{3} \\ 0 \end{bmatrix} \tag{A3b}$$

$$\boldsymbol{p}'_3 = \begin{bmatrix} p'_{3x} \\ p'_{3y} \\ p'_{3z} \end{bmatrix} = \begin{bmatrix} \dfrac{l_b}{2} \\ \dfrac{-\sqrt{3} l_b}{6} \\ 0 \end{bmatrix} \tag{A3c}$$

Appendix B: Solution Selection

In this section, a method is presented for selection of the solutions to $\beta_i (i=1,3)$. To do so, the plane of $O-b_i s_i p_i$ is inspected.

Figure 11 shows the plane of $O-b_i s_i p_i$. In this figure, $p_j^{(p)}$ is the projection of p_j, and

$$\left.\begin{aligned} \beta_{i1} &\geqslant 0 \\ \beta_{i2} &\leqslant 0 \end{aligned}\right\} \tag{B1}$$

From Fig. 11, it can be seen that β_{i1} is the real solution. On the contrary, β_{i2} is the imagery solution, p_i is the real position, and p'_i is the imagery position, for the simple reason that the two points $p_j^{(p)}$ and p_i are always mirrored each other about the line $b_i s_i$. Therefore, the solution given in Eq. (18) is valid.

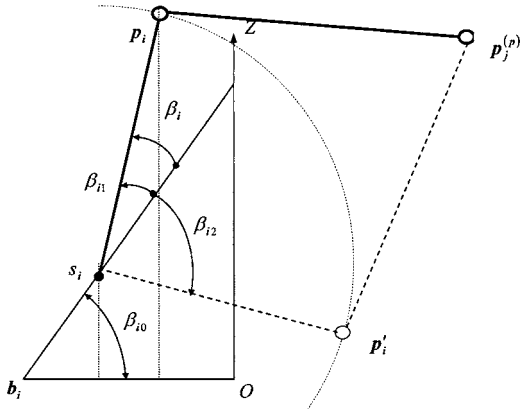

Fig. 11 Plane of $O-b_i s_i p_i$

Furthermore, from Fig. 10, it can also be seen that

$$\beta_i^{(0)} + \beta_{i0} > 0 \tag{B2}$$

Thus, Eq. (26) should always select a positive sign.

References

[1] Xi, F., Han, W., Marcel, V., and Ross, A., 2000, "Development of A Sliding-Leg Tripods as an Add-on Device for Manufacturing," Robotica, **18**, pp. 285–294.

[2] Xi, F., Zhang, D., Xu, Z., and Mechefske, C., 2003, "A Comparative Study on Tripod-Based Machine Tools," Int. J. Mach. Tools Manuf., pp. 721–703.

[3] Liu, Z., and Xiong, Y., 1999, "A Closed Form Forward Kinematics of 6-6 Stewart in Parallel Mechanisms," Chin. J. Huazong Univ. Sci. Technol.

[4] Sreenivasan, S. V., and Waldron, K. J., 1994, "Closed-Form Direct Displacement Analysis of a 6-6 Stewart Platform," Mech. Mach. Theory, **29**(6), pp. 855–864.

[5] Innocenti, C., 2001, "Forward Kinematics in Polynomial Form of the General Stewart Platform," J. Mech. Des., **123**, pp. 254–260.

[6] Tsai, M. S., Shiau, T. N., Tsai, Y. J., and Chang, T. H., 2003, "Direct Kinematic Analysis of a 3-PRS Parallel Mechanism," Mech. Mach. Theory, **38**, pp. 71–83.

[7] Jinwook, K., and Park, F. C., 2001, "Direct Kinematic Analysis of 3-RS Parallel Mechanisms," Mech. Mach. Theory, **36**, pp. 1121–1134.

[8] Lee, T.-Y., and Shim, J.-K., 2001, "Forward Kinematics of the General 6-6 Stewart Platform using Algebraic Elimination," Mech. Mach. Theory, **36**, pp. 1073–1085.

[9] Fan, S., Xu, L., and Zhou, Z., 2002, "Numeric-Symbolic Approach for Direct Displacement Solution of 4-DOF Spatial Parallel Mechanism," Chin. J. Mech. Eng., pp. 57–60.

[10] Liu, D., Gong, Z., and Wang, Q., 1998, "An Optimization Method for the Direct Position Kinematic Problem of a General Parallel Manipulator," Chin. J. Appl. Sci., (4).

[11] Wen, F. A., and Liang, C. G., 1994, "Displacement analysis of the 6-6 Stewart platform mechanisms," Mech. Mach. Theory, **29**(4), pp. 547–557.

[12] Monsarrat, B., and Gosselin, C., 2001, "Singularity Analysis of a Three-Leg Six-degree-of-Freedom Parallel Platform Mechanism Based on Grassman Line Geometry," Int. J. Robot. Res., **20**(4), pp. 312–328.

[13] Merlet, J.-P., 1992, "Direct Kinematics and Assembly Modes of Parallel Manipulators," Int. J. Robot. Res., **11**(2), pp. 150–162.

[14] Zheng, G., Leonada, S., Haynes, J., Lee, D., and Carroll, R. L., 1992, "On the Dynamic Model and Kinematic Analysis of a Class of Stewart Platform," Rob. Auton. Syst., **9**, pp. 237–254.

[15] Rao, Q., Cheng, N., and Bai, S., 1994, "Forward Position Analysis of the 6-6 Stewart Platform," Chin. J. Mech. Sci. Technol.