Solving the forward kinematics problem of six-DOF Stewart platform using multi-task Gaussian process

Jianjun He, Hong Gu and Zhelong Wang

DOI: 10.1177/0954406212444508

The online version of this article can be found at:
http://pic.sagepub.com/content/227/1/161
Solving the forward kinematics problem of six-DOF Stewart platform using multi-task Gaussian process

Jianjun He, Hong Gu and Zhelong Wang

Abstract
The forward kinematics problem of general six-degree-of-freedom Stewart platform is addressed in this article. Unlike the convention taking positional variables as independent ones and solving them individually, this article presents an alternative approach which takes the positional variables as multiple-related tasks and exploits the commonality between them using a multi-task Gaussian process learning method, as a result, a simple adaptive algorithm, which may satisfy the requirements for high accuracy and real-time processing, is established. Moreover, the proposed algorithm can achieve the desired accuracy using 1000 training samples at most, which is far less than those of other algorithms. Simulation results on a Stewart platform used in aircraft flight simulation system show that the proposed algorithm can achieve superior performance.

Keywords
Stewart platform, forward kinematics, multi-task learning, Gaussian process

Date received: 22 June 2011; accepted: 14 March 2012

Introduction
The six-degree-of-freedom (DOF) Stewart platform is a parallel mechanism that generally consists of six identical telescopic links, a mobile platform, and a base platform. As shown in Figure 1, each link connects the base and the mobile platforms with universal and spherical joints respectively, and the link length is controlled by a prismatic joint. The base platform is generally fixed as a base and the mobile platform has six-DOF motion with respect to the base platform. Over the past years, six-DOF Stewart platform has received more and more attentions from researchers due to its extensive applications in various fields such as machine tool technology, medical robots, crane technology, underwater research, flight simulation systems, air-to-sea rescue, and telescopes.

The forward kinematics problem (FKP) of Stewart platform is to compute the position and orientation (pose) of mobile platform when link lengths are given. Because it involves a set of highly coupled nonlinear equations and in general there is no closed form and unique solution, this problem has never been fully solved and researchers have to explore new algorithms.

Approaches for solving FKP may be classified into two categories.

The first category mainly focuses on finding all the possible solutions of FKP for a certain Stewart platform. Some representative approaches of this category are polynomial, continuation, and interval analysis methods. Polynomial methods\(^1\),\(^2\) usually use different variable elimination techniques to generate a high degree of polynomial equation with a single unknown. Also, all the possible solutions can be found by solving this polynomial equation. In continuation methods,\(^3\),\(^4\) a solvable start system of equations is transformed into a target system of equations incrementally, while all the solutions of the target system can be found by tracking the solutions of start system. The main idea of interval analysis method\(^5\) is to obtain convergence region of all
solutions by subdividing the solution space of equations.

The other category tries to find a unique actual pose of mobile platform in real time using numerical techniques. Neural network is one of the popular approaches of this category. In Yurt et al., a feedforward neural network was trained using backpropagation algorithm to solve the FKP of 6-3 Stewart platform. Parikh and Lam proposed a neural network based hybrid approach where they obtained an initial solution to the Newton–Raphson method through a neural network. Considering the hybrid approach may be time consuming, they developed an iterative neural network approach which consists of a trained neural network and an error compensation algorithm in the feedback loop. Some other approaches also have been introduced. Wang presented a numerical method that utilizes the relationship between the small change in link lengths and the small motion of mobile platform to generate a unique solution. Liu et al. discussed the solution of FKP of general Stewart platform using hybrid immune genetic algorithm. In Chen et al., the FKP was solved using a nonlinear observer designed to estimate the system states including three-axis translations and rotations. Principal component analysis (PCA) was used to establish an iterative algorithm for FKP by Wang et al. Since the PCA-based algorithm can hardly be used in real-time situation, recently, they presented an improved algorithm using independent component analysis. Though many approaches have been developed and promising results have been achieved for certain simplified configurations of Stewart platform, it is still a challenging problem to devise a methodology that can solve the FKP for general Stewart platform in real time and with high accuracy.

Researchers have found that in many situations, learning multiple-related tasks simultaneously can achieve better performance than learning each of them independently. In machine learning field, this subject is approached through multi-task learning, which aims to improve the performance of learning tasks using a shared model. Since many real-world problems can be solved in the framework of it, multi-task learning has received extensive attentions of researchers and many algorithms have been developed using the strategies such as sharing hidden nodes in neural networks, placing a common prior in Bayesian models, sharing parameters of Gaussian processes, and structured regularization in kernels methods. In the previous approaches for solving FKP of Stewart platform, the six positional variables (i.e. variables represent the position and orientation of the mobile platform) are taken as independent ones and individually solved. However, it is attractive if one can exploit commonality in these variables to improve performance. The aim of this article is to show how this can be carried out. Using a multi-task Gaussian process (MTGP) learning method, link lengths are transformed into other six variables called intermediate variables, each of which is only related to one of the positional variables. Then, six one-to-one relationships between these intermediate and positional variables may be identified. As a result, a simple iterative algorithm for solving the FKP can be established using these one-to-one relationships.

The structure of the remainder of this article is as follows: ‘The kinematic model of Stewart platform’ is illustrated in the following section. Next, ‘Multi-task Gaussian process’ is used to get a set of intermediate variables. Thereafter, ‘Iterative algorithm’ is presented. Then, ‘Simulations’ are carried out. The ‘Conclusions’ make up the final section.

The kinematic model of Stewart platform

For the convenience of a establishing kinematic model, two right-handed Cartesian coordinate systems were set up for the Stewart platform. As shown in Figure 1, the B-frame $Oxyz$ has its origin at the centroid of the base platform and the M-frame $Ox'y'z'$ at the centroid of the mobile platform. The $z$- and $z'$-axes are vertically directed out of the plane of base and mobile platforms, respectively. The axes of the two coordinate systems are
aligned when the mobile platform is at its initial position. The terms $b_i$ and $t_i$ are the joints of $i$th link on the base and mobile platforms, respectively. The pose of the mobile platform can be represented by six positional variables $(x, y, z, \alpha, \beta, \gamma)$. Here, $(x, y, z)$ are the coordinates of $O'$ in the B-frame and $\alpha, \beta, \gamma$ the rotational angles of the mobile platform relative to $z'$, $y'$, and $x'$-axes of B-frame, respectively. Let $(b_1, b_2, b_3)\T$ denote the coordinate of joint $b_i$ in B-frame and $(t_1, t_2, t_3)\T$ be the coordinate of $t_i$ in M-frame. Here, $l_i$ is the length of $i$th link $(i = 1, \ldots, 6)$. The inverse kinematic model of Stewart platform may be established as follows

$$||{(x, y, z)}\T + R(t_1, t_2, t_3)\T - (b_1, b_2, b_3)\T|| = l_i, \quad i = 1, \ldots, 6$$  \hspace{1cm} (1)

where $R$ is a rotation matrix defined as

$$R = \begin{pmatrix}
\cos \alpha \cos \beta & \cos \alpha \sin \beta \sin \gamma - \sin \alpha \cos \gamma & \cos \alpha \sin \beta \cos \gamma + \sin \alpha \sin \gamma \\
\sin \alpha \cos \beta & \sin \alpha \sin \beta \sin \gamma + \cos \alpha \cos \gamma & \sin \alpha \sin \beta \cos \gamma - \cos \alpha \sin \gamma \\
-\sin \beta & \cos \beta \sin \gamma & \cos \beta \cos \gamma
\end{pmatrix}$$  \hspace{1cm} (2)

Thus, the FKP of Stewart platform is to find a set of functions $(x = f_1(L), \ y = f_2(L), \ldots, \gamma = f_6(L))$ using equation (1); here, $L = (l_1, \ldots, l_6)\T$.

The proposed algorithm will be described based on a real-world six-DOF Stewart platform used in aircraft flight simulation system.\textsuperscript{21} This Stewart platform includes a fixed base and a payload platform upon which a cockpit and a visual display are mounted. The payload platform is a six-DOF motion system driven by six hydraulically powered position servomechanisms. Its geometrical parameters are given in Dieudonne et al.\textsuperscript{21}

**Multi-task Gaussian process**

It can be seen from the above section that the FKP of Stewart platform is to find functions $\{f_1, \ldots, f_6\}$. In this section, MTGP is used to find an approximation function $\hat{f}_i$ of $f_i$ $(i = 1, \ldots, 6)$, which will be used to establish an iterative algorithm for solving FKP.

MTGP is a multi-task learning model recently proposed by Bonilla et al.\textsuperscript{19} in the context of Gaussian processes. Let $\{g_i(x)\}_{i=1}^m$ be $m$ related task functions that cannot be described by any explicit expression. It is typical for more realistic modeling situations that we do not have access to function value $g_i(x)$ itself, but only noisy version thereof $y_i(x) = g_i(x) + \varepsilon_i$, $i = 1, \ldots, m$. Let $X = \{x_1, \ldots, x_n\}$ be a set of inputs and $Y = \{Y_1, \ldots, Y_m\}$ be the set that consists of the observations of $\{g_i(x)\}_{i=1}^m$ on the input set $X$, where $Y_k = \{y_i(x_k)\}_{i=1}^m$. For notational convenience, $S = \{(x_1, Y_1), \ldots, (x_n, Y_n)\}$ is named ‘training set,’ and $Y$ is represented by a vector $(y_1(x_1), \ldots, y_m(x_1), \ldots, y_m(x_n))\T$ rather than the set $\{Y_1, \ldots, Y_n\}$. The purpose of multi-task learning is to predict the function values $\{g_i(x*)\}_{i=1}^m$ for a new input $x*$ based on the training set $S$. To this end, the basic idea of MTGP is to place a Gaussian process prior over the functions $\{g_i(x)\}_{i=1}^m$, i.e., suppose $\{g_i(x)\}_{i=1}^m$ are Gaussian processes with zero mean and the following covariance

$$\text{cov}(g_i(x), g_j(x*)) = K^g_{ij}(x, x*), \quad i, j = 1, \ldots, m$$  \hspace{1cm} (3)

where $K^g$ is a positive semi-definite matrix that specifies the correlations among $\{g_i(x)\}_{i=1}^m$ and $k(\cdot, \cdot)$ a covariance function over input $x$; $k(x, x*) = \exp(-\frac{\|x - x*\|^2}{\sigma^2})$ is used in this study.

Let $G = (g_1(x_1), \ldots, g_m(x_1), \ldots, g_m(x_n))\T$ and $G* = (g_1(x*), \ldots, g_m(x*))\T$ be two vectors that, respectively, consist of the values of functions $\{g_i(x)\}_{i=1}^m$ on $X$ and $x*$. The following prior distributions over $G$ and $G*$ can be obtained using equation (3),

$$p(G|X, K^g, \theta) = \mathcal{N}(G|0, K^g \otimes K)$$  \hspace{1cm} (4)

$$p(G*, G|X, x*, K^g, \theta) = \mathcal{N}
\left[
\begin{array}{c}
G*\\
G
\end{array}
\right]
\left[
\begin{array}{c}
G*\\
G
\end{array}
\right]
N
\left[
\begin{array}{c}
K^g \otimes K^g \\
K^g \otimes K
\end{array}
\right]$$  \hspace{1cm} (5)

where $K$ is the covariance matrix of $X$, $K^g = k(x*, x*)$, $K^g$ is the covariance vector between $X$ and $x*$. Thus, the conditional prior

$$p(G*|G, X, x*, K^g, \theta) = \mathcal{N}(G*|(K^g \otimes K^g)^{-1}G, \times K^g \otimes (K^g - K^g K^g K^g)^{\dagger}K^g)$$  \hspace{1cm} (6)

may be analytically deduced, where $K^g$ is an identity matrix with the same order as $K^g$.

We have assumed that the observation $y_i(x)$ differs from the function value $g_i(x)$ by additive noise $\varepsilon_i$, and we will further assume that this noise follows an independent, identically distributed Gaussian distribution with zero mean and variance $\sigma^2_i$, i.e.,

$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2_i), \ i = 1, \ldots, m$$  \hspace{1cm} (7)
Thus, we can obtain the likelihood
\[ p(Y|G, D) = N(Y|G, D \otimes I) \]  
(8)
where \( D = \text{diag}(\sigma_1^2, \ldots, \sigma_n^2) \) and \( I \) is an identity matrix.

Equipped with the prior (4) and the likelihood (8), a close-form of the posterior distribution \( p(G|Y, X, K^*, \theta, D) \) can be deduced using Bayes’ rule
\[
p(G|Y, X, K^*, \theta, D) = \frac{p(Y|G, D) p(G|X, K^*, \theta)}{p(Y|X, K^*, \theta, D)}
\]
\[= N(G|K^* \otimes K)\Sigma^{-1} Y, (K^* \otimes K)\Sigma^{-1}(D \otimes I)) \]
(9)
where \( \Sigma = K^* \otimes K + D \otimes I \), and
\[
p(Y|X, K^*, \theta, D) = \int p(Y|G, D) p(G|X, K^*, \theta) dG \]
(10)
denotes the marginal likelihood for the hyperparameter \( \theta, K^* \) and \( D \).

Using the conditional prior (6) and the posterior (9), we can obtain the predictive distribution of \( G \).
\[
p(G_*,|Y, X, x_*, K^*, \theta, D)
\]
\[= \int p(G_*,|G, Y, X, K^*, \theta) p(G|Y, X, K^*, \theta, D) dG
\]
\[= N(G_*|K_*^* \otimes K_* \Sigma^{-1} Y, K_*^* \otimes K_*)
\]
\[- (K_*^* \otimes K_*)^{-1} \Sigma^{-1} (K^* \otimes K_*) \]
(11)
Thus, the mean prediction on a new input \( x_* \) for the \( i \)th task \( g(x) \) is
\[
\hat{g}(x_i) = (K^*_i \otimes K_*)^{-1} \Sigma^{-1} Y, \Sigma = K^* \otimes K + D \otimes I
\]
(12)
where \( K^*_i \) selects the \( i \)th column of \( K^* \), the hyperparameter \( \theta, K^* \) and \( D \) can be obtained by maximizing the marginal likelihood \( p(Y|X, \theta, K^*, D) \) (10) using a gradient-based method. Until now, the whole MTGP algorithm has been presented.

Let \( \{Q_k = (x_k, y_k, z_k, \alpha_k, \beta_k, \gamma_k)^T \mid k = 1, \ldots, n\} \) be a sample set of positional variables, the corresponding sample set \( \{L_k = (l_{k1}, \ldots, l_{kn})^T \mid k = 1, \ldots, n\} \) of link lengths can be computed using equation (1). Taking \( S = \{L_k, Q_k\} \mid k = 1, \ldots, n\} \) as a training set, we can obtain an approximation function \( \hat{f}_i \) of \( f_i \) using equation (12), \( i = 1, \ldots, 6 \), i.e.
\[
\hat{f}_i(L_k) = (K^*_i \otimes K_*)^{-1} \Sigma^{-1} Y, \Sigma = K^* \otimes K + D \otimes I
\]
(13)
It can be seen that the computational cost of equation (13) is in proportion to \( n \). Although \( \{f_i|i = 1, \ldots, 6\} \) may also be taken as a solution of FKP, it usually cannot satisfy the requirement of real-time processing because it needs a big training set to obtain the desired accuracy. In the following section, we will first analyze the relationships between \( \{f_i|i = 1, \ldots, 6\} \) and \( \{x, y, z, \alpha, \beta, \gamma\} \), and then use the obtained relationships to establish a simple iterative algorithm for solving FKP, which can satisfy the requirements for real-time processing and high accuracy.

### Iterative algorithm

Equation (14) may be obtained by substituting equation (1) into equation (13),
\[
\hat{f}_i = h_i(x, y, z, \alpha, \beta, \gamma)
\]
(14)
For the convenience of expression, \( \{\hat{f}_i|i = 1, \ldots, 6\} \) are called intermediate variables. In order to analyze the relationships between \( \{f_i|i = 1, \ldots, 6\} \) and \( \{x, y, z, \alpha, \beta, \gamma\} \), two sample sets \( Q(1) = \{Q_k(k = 1, \ldots, n_1) \} \) and \( Q(2) = \{Q_k(k = 1, \ldots, n_2) \} \) of positional variables are generated from the uniform distribution on the range of these positional variables; here, \( n_1 = 500 \) and \( n_2 = 1000 \). A sample set \( \{L_k|k = 1, \ldots, n_1\} \) of link lengths corresponding to \( Q(1) \) can be computed using equation (1). Taking \( \{L_k, Q(1)\} \mid k = 1, \ldots, n_1\} \) as a training set of equation (13), we can obtain a sample set \( F = \{F_k|k = 1, \ldots, n_2\} \) of \( \{\hat{f}_i|i = 1, \ldots, 6\} \) corresponding to \( Q(2) \) from equation (14); here, \( F_k = (\hat{f}_k, \ldots, \hat{f}_k) \). Figure 2 visualizes the relationships between intermediate variables \( \{f_i|i = 1, \ldots, 6\} \) and positional variables \( \{x, y, z, \alpha, \beta, \gamma\} \) by plotting all the samples \( \{F_k, Q(2)\} \mid k = 1, \ldots, n_2\} \). The relationship between \( f_1 \) and \( x \) may be concluded as follows: (1) \( f_1 \) is mainly related to \( x \) and the other five positional variables do not have much effect on it. (2) \( f_1 = h_1(x, y, z, \alpha, \beta, \gamma) \) is a monotone increasing continuous function of \( x \) if other positional variables are given. The relationships \( \{f_2, y\}, \{f_3, z\}, \ldots \) and \( \{f_6, y\} \) are the same as \( \{f_1, x\} \) and will not be repeated here.

Let \( L_\ast = (l_{1\ast}, \ldots, l_{6\ast})^T \) be the given values of link lengths, the FKP of Stewart platform is to find the corresponding values of positional variables. The proposed algorithm includes two steps: (1) computing the values \( F = (f_1, \ldots, f_6)^T \) of intermediate variables corresponding to \( L_\ast \) using equation (13). (2) Solving the equations \( \{f_i = h_i(x, y, z, \alpha, \beta, \gamma)i = 1, \ldots, 6\} \), the solution \( \{x, y, z, \alpha, \beta, \gamma\} \) is the desired values of positional variables. Because \( f_1, \ldots, f_6 \) are mainly related to \( x, y, z, \alpha, \beta, \gamma \), respectively, the equations \( \{f_i = h_i(x, y, z, \alpha, \beta, \gamma)i = 1, \ldots, 6\} \) can be solved using a simple iterative method. The flowchart of the proposed algorithm is shown in Figure 3. The termination condition of the algorithm is that the difference \( \{f_i - h_i(x_0, y_0, z_0, \alpha_0, \beta_0, \gamma_0)\} \) is smaller than \( \varepsilon_i \) for all \( i = 1, \ldots, 6 \). Here, \( \varepsilon_1, \ldots, \varepsilon_6 \) are the critical values of the allowable errors of \( x, y, z, \alpha, \beta, \gamma \).
respectively. Because $f_i = h_i(x, y, z, \alpha, \beta, \gamma)$ is a monotone increasing continuous function of $x$ if other positional variables are given, Nelder–Mead algorithm can be used to search $x^*$ that satisfies $f_{i1} = h_i(x^*, y_0, z_0, \alpha_0, \beta_0, \gamma_0)$. The detailed description of the algorithm searching for $x^*$ is shown in Figure 4. The algorithms searching for $y^*$, $z^*$, $\alpha^*$, $\beta^*$, and $\gamma^*$ are the same as the algorithm of $x^*$. Hence, they will not be repeated here.

**Simulations**

For investigating the performance of the proposed algorithm, a training set $\{(L_k^{(1)}, Q_k^{(1)})| k = 1, \ldots, n_1\}$

![Figure 2. The relationships between positional variables and $\{f_i\}_{i=1}^6$.](image-url)
and a test set \( \{(L_k^{(2)}, Q_k^{(2)})\} \) \( k = 1, \ldots, n_2 \) are created, where \( \{Q_k^{(1)}\} \) \( k = 1, \ldots, n_1 \) and \( \{Q_k^{(2)}\} \) \( k = 1, \ldots, n_2 \) are two sample sets of positional variables generated from the uniform distribution on their ranges; \( \{L_k^{(1)}\} \) \( k = 1, \ldots, n_1 \) and \( \{L_k^{(2)}\} \) \( k = 1, \ldots, n_2 \), which can be obtained using equation (1), are the sample sets of link lengths corresponding to \( \{Q_k^{(1)}\} \) \( k = 1, \ldots, n_1 \) and \( \{Q_k^{(2)}\} \) \( k = 1, \ldots, n_2 \), respectively. In the simulations, \( n_2 \) is set to 5000; the training and test sets, respectively, are generated 10 times, and the average performance is recorded. In addition, the simulations are conducted on a 1.5 GHz CPU, 256 MB RAM computer system based on a Matlab implementation of the proposed algorithm.

Table 1 presents the computed results of the proposed algorithm with different numbers of training samples when \( \epsilon_1, \epsilon_2, \epsilon_3 = 0.01 \text{mm} \) and \( \epsilon_4, \epsilon_5, \epsilon_6 = 0.003 \text{°} \). Here, MAE stands for mean absolute error and Perc stands for the percentage of samples that meet the allowable error. It can be seen that the performance of the proposed algorithm evidently improves when the number of training samples changes from 200 to 500 and then tends to level up in interval [500, 900]. Table 2 presents the computed results of the proposed algorithm with different allowable errors when \( n_1 = 800 \). It could be observed that Perc is mainly related to the ratio of \( \epsilon_1, \epsilon_2, \epsilon_3 \) to \( \epsilon_4, \epsilon_5, \epsilon_6 \) and the

---

**Figure 3.** The flowchart of the proposed iterative algorithm.

**Figure 4.** The algorithm for searching \( x_a \).
computing time, and average number of iterations are related to the scale of allowable errors.

In order to evaluate the relative performance of the proposed algorithm, it is compared with three published algorithms. Because the codes are difficult to be implemented or obtained, all the information of the published algorithms is taken from their original literatures. The second column in Table 3 presents the reported best accuracy of each algorithm. It can be seen, as far as the accuracy is concerned, the proposed algorithm can achieve superior performance to the published algorithms. The third and the fourth columns in Table 3, respectively, present the computed time of each algorithm and the configuration of the computer system on which the simulations are conducted, it can be seen that the proposed algorithm will spend more computing time than iterative artificial neural network algorithm8 and Parikh’s algorithm7. However, today, a

<table>
<thead>
<tr>
<th>Number of training samples</th>
<th>Time (s)</th>
<th>Perc (%)</th>
<th>x (10^{-2} mm)</th>
<th>y (10^{-2} mm)</th>
<th>z (10^{-2} mm)</th>
<th>α (10^{-3}°)</th>
<th>β (10^{-3}°)</th>
<th>γ (10^{-3}°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>0.3522</td>
<td>4.8683</td>
<td>77.26</td>
<td>0.6190</td>
<td>0.6793</td>
<td>0.8101</td>
<td>0.7603</td>
<td>0.7712</td>
</tr>
<tr>
<td>300</td>
<td>0.2940</td>
<td>4.1264</td>
<td>86.83</td>
<td>0.4443</td>
<td>0.4904</td>
<td>0.5788</td>
<td>0.7172</td>
<td>0.7370</td>
</tr>
<tr>
<td>400</td>
<td>0.2681</td>
<td>3.7785</td>
<td>90.93</td>
<td>0.3418</td>
<td>0.3877</td>
<td>0.4494</td>
<td>0.6930</td>
<td>0.7224</td>
</tr>
<tr>
<td>500</td>
<td>0.2593</td>
<td>3.5536</td>
<td>92.53</td>
<td>0.2938</td>
<td>0.3338</td>
<td>0.3849</td>
<td>0.6877</td>
<td>0.7130</td>
</tr>
<tr>
<td>600</td>
<td>0.2646</td>
<td>3.4640</td>
<td>93.26</td>
<td>0.2754</td>
<td>0.3230</td>
<td>0.3740</td>
<td>0.6924</td>
<td>0.7127</td>
</tr>
<tr>
<td>700</td>
<td>0.2546</td>
<td>3.2674</td>
<td>94.64</td>
<td>0.2545</td>
<td>0.2896</td>
<td>0.3319</td>
<td>0.6841</td>
<td>0.7151</td>
</tr>
<tr>
<td>800</td>
<td>0.2475</td>
<td>3.1198</td>
<td>95.89</td>
<td>0.2256</td>
<td>0.2639</td>
<td>0.2927</td>
<td>0.6842</td>
<td>0.7045</td>
</tr>
<tr>
<td>900</td>
<td>0.2447</td>
<td>2.9877</td>
<td>96.68</td>
<td>0.2180</td>
<td>0.2462</td>
<td>0.2781</td>
<td>0.6843</td>
<td>0.6928</td>
</tr>
</tbody>
</table>

MAE: mean absolute error.

<table>
<thead>
<tr>
<th>Allowable errors</th>
<th>Time (s)</th>
<th>Perc (%)</th>
<th>x (mm)</th>
<th>y (mm)</th>
<th>z (mm)</th>
<th>α (°)</th>
<th>β (°)</th>
<th>γ (°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ε_x$, $ε_y$, $ε_z$ (mm)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>0.1244</td>
<td>2.2260</td>
<td>99.23</td>
<td>3.8 $\times$ 10^{-2}</td>
<td>4.2 $\times$ 10^{-2}</td>
<td>4.6 $\times$ 10^{-2}</td>
<td>2.5 $\times$ 10^{-3}</td>
<td>2.5 $\times$ 10^{-3}</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1526</td>
<td>2.4589</td>
<td>98.62</td>
<td>1.7 $\times$ 10^{-2}</td>
<td>2.0 $\times$ 10^{-2}</td>
<td>2.2 $\times$ 10^{-2}</td>
<td>2.5 $\times$ 10^{-3}</td>
<td>2.5 $\times$ 10^{-3}</td>
</tr>
<tr>
<td>0.05</td>
<td>0.1662</td>
<td>2.5780</td>
<td>97.83</td>
<td>9.8 $\times$ 10^{-3}</td>
<td>1.1 $\times$ 10^{-2}</td>
<td>1.2 $\times$ 10^{-2}</td>
<td>2.4 $\times$ 10^{-3}</td>
<td>2.4 $\times$ 10^{-3}</td>
</tr>
<tr>
<td>0.025</td>
<td>0.1849</td>
<td>2.7827</td>
<td>93.90</td>
<td>7.0 $\times$ 10^{-3}</td>
<td>7.8 $\times$ 10^{-3}</td>
<td>8.7 $\times$ 10^{-3}</td>
<td>2.4 $\times$ 10^{-3}</td>
<td>2.4 $\times$ 10^{-3}</td>
</tr>
<tr>
<td>0.01</td>
<td>0.2166</td>
<td>3.0461</td>
<td>78.50</td>
<td>5.2 $\times$ 10^{-3}</td>
<td>6.1 $\times$ 10^{-3}</td>
<td>6.2 $\times$ 10^{-3}</td>
<td>2.4 $\times$ 10^{-3}</td>
<td>2.4 $\times$ 10^{-3}</td>
</tr>
<tr>
<td>0.01</td>
<td>0.2859</td>
<td>3.1859</td>
<td>98.86</td>
<td>1.6 $\times$ 10^{-3}</td>
<td>1.9 $\times$ 10^{-3}</td>
<td>2.1 $\times$ 10^{-3}</td>
<td>2.3 $\times$ 10^{-4}</td>
<td>2.5 $\times$ 10^{-4}</td>
</tr>
<tr>
<td>0.001</td>
<td>0.4435</td>
<td>3.6863</td>
<td>98.93</td>
<td>1.6 $\times$ 10^{-4}</td>
<td>1.7 $\times$ 10^{-4}</td>
<td>1.9 $\times$ 10^{-4}</td>
<td>2.4 $\times$ 10^{-5}</td>
<td>2.4 $\times$ 10^{-5}</td>
</tr>
</tbody>
</table>

MAE: mean absolute error.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Accuracy</th>
<th>Time (s)</th>
<th>Computer systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parikh’s algorithm7</td>
<td>0.01 mm, 0.01°</td>
<td>0.02</td>
<td>0.5 GHz CPU, 128 MB RAM</td>
</tr>
<tr>
<td>PCA-based algorithm12</td>
<td>0.01 mm, 0.0003°</td>
<td>&gt; 1</td>
<td>1.8 GHz CPU, 512 MB RAM</td>
</tr>
<tr>
<td>Iterative ANN8</td>
<td>0.25 mm, 0.01°</td>
<td>0.1</td>
<td>0.5 GHz CPU, 128 MB RAM</td>
</tr>
<tr>
<td>The proposed algorithm</td>
<td>0.001 mm, 0.0001°</td>
<td>0.4</td>
<td>1.5 GHz CPU, 256 MB RAM</td>
</tr>
</tbody>
</table>

PCA: principal component analysis and ANN: artificial neural network.
common personal computer has been with 2.7 GHz CPU, 2 GB RAM, which is enough to make the proposed algorithm satisfy the requirements for real-time processing of many applications.

At last, in order to evaluate the performance of the proposed algorithm on the Stewart platform adopting other geometrical parameters, it is applied to a 6-3 Stewart platform used by Yurt et al.\textsuperscript{6} Table 4 presents some results of the proposed algorithm using 800 training samples. It can be seen that the proposed algorithm also can achieve superior performance.

### Conclusions

The forward kinematics is essential for the applications of Stewart platform. Although many approaches have been developed, it is still a challenge to devise a methodology for general Stewart platform that satisfies the requirements for high accuracy and real-time processing. Unlike the convention taking the positional variables as independent ones and solving them individually, this article presents an alternative approach which considers the variables as multiple-related tasks and solves them simultaneously. Simulation results show that the proposed algorithm can achieve superior performance.

### Funding

This study was supported by China Earthquake Research Funds (grant no. 200808075) and Liaoning Provincial Natural Science Foundation of China (grant no. 20102025).

### Conflict of interests

None Declared.

### References


### Table 4. The performance of the proposed algorithm on the Stewart platform used by Yurt et al.\textsuperscript{6}

<table>
<thead>
<tr>
<th>Allowable error</th>
<th>MAE of positional variables</th>
<th>Average number of iterations</th>
<th>Perc (%)</th>
<th>Time (s)</th>
<th>$\varepsilon_1, \varepsilon_2, \varepsilon_3$ (mm)</th>
<th>$\varepsilon_4, \varepsilon_5, \varepsilon_6$ (%)</th>
<th>$x$ (mm)</th>
<th>$y$ (mm)</th>
<th>$z$ (mm)</th>
<th>$\alpha$ (°)</th>
<th>$\beta$ (°)</th>
<th>$\gamma$ (°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.1)</td>
<td></td>
<td>0.0565</td>
<td>1.768</td>
<td>100</td>
<td>$1.4 \times 10^{-2}$</td>
<td>$1.5 \times 10^{-2}$</td>
<td>$1.2 \times 10^{-2}$</td>
<td>$2.5 \times 10^{-3}$</td>
<td>$2.5 \times 10^{-3}$</td>
<td>$2.4 \times 10^{-3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0.05)</td>
<td></td>
<td>0.0613</td>
<td>1.1573</td>
<td>100</td>
<td>$7.6 \times 10^{-3}$</td>
<td>$7.7 \times 10^{-3}$</td>
<td>$5.7 \times 10^{-3}$</td>
<td>$2.5 \times 10^{-3}$</td>
<td>$2.4 \times 10^{-3}$</td>
<td>$2.4 \times 10^{-3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0.01)</td>
<td></td>
<td>0.0907</td>
<td>1.6279</td>
<td>100</td>
<td>$1.6 \times 10^{-3}$</td>
<td>$1.6 \times 10^{-3}$</td>
<td>$1.4 \times 10^{-3}$</td>
<td>$2.5 \times 10^{-3}$</td>
<td>$2.4 \times 10^{-3}$</td>
<td>$2.4 \times 10^{-3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0.002)</td>
<td></td>
<td>0.1318</td>
<td>2.0172</td>
<td>99.03</td>
<td>$4.5 \times 10^{-4}$</td>
<td>$4.0 \times 10^{-4}$</td>
<td>$2.9 \times 10^{-4}$</td>
<td>$2.5 \times 10^{-3}$</td>
<td>$2.4 \times 10^{-3}$</td>
<td>$2.4 \times 10^{-3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0.001)</td>
<td></td>
<td>0.1445</td>
<td>2.0657</td>
<td>92.36</td>
<td>$3.3 \times 10^{-4}$</td>
<td>$3.1 \times 10^{-4}$</td>
<td>$1.9 \times 10^{-4}$</td>
<td>$2.5 \times 10^{-3}$</td>
<td>$2.4 \times 10^{-3}$</td>
<td>$2.4 \times 10^{-3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0.0001)</td>
<td></td>
<td>0.1635</td>
<td>2.065</td>
<td>100</td>
<td>$1.4 \times 10^{-4}$</td>
<td>$1.4 \times 10^{-4}$</td>
<td>$1.3 \times 10^{-4}$</td>
<td>$2.5 \times 10^{-4}$</td>
<td>$2.3 \times 10^{-4}$</td>
<td>$2.3 \times 10^{-4}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0.00001)</td>
<td></td>
<td>0.2523</td>
<td>2.3965</td>
<td>100</td>
<td>$1.5 \times 10^{-5}$</td>
<td>$1.6 \times 10^{-5}$</td>
<td>$1.4 \times 10^{-5}$</td>
<td>$2.3 \times 10^{-5}$</td>
<td>$2.4 \times 10^{-5}$</td>
<td>$2.4 \times 10^{-5}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MAE: mean absolute error.


**Appendix**

**Notation**

\[ \text{cov}(x, y) \] covariance between two random variables (processes) \( x \) and \( y \)

\[ \text{diag}(a_1, \ldots, a_n) \] diagonal matrix with \( a_1, \ldots, a_n \) on its diagonal

\[ \exp(\cdot) \] exponential function

\[ \mathcal{N}(x|\mu, \sigma) \] the variable \( x \) has a Gaussian distribution with mean \( \mu \) and covariance \( \sigma \)

\[ p(A|B) \] conditional probability of \( A \) given \( B \)

\[ ||v|| \] \( l^2 \)-norm (Euclidean norm) of the vector \( v \)

\[ v^T \] transpose of the vector (matrix) \( v \)

\[ \otimes \] Kronecker product