2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Real Time Forward Kinematics Solutions for General Stewart Platforms

Mahmoud Tarokh
Department of Computer Science
San Diego State University
San Diego, CA 92182-7720, U.S.A.
tarokh@cs.sdsu.edu

Abstract – A new paradigm is introduced for solving the forward kinematics of general Stewart platforms in real-time. It consists of an off-line preprocessing phase and an online real-time evaluation phase. In the preprocessing phase, the platform leg (link) space is decomposed into cells, and a large set of data is generated for the platform position/orientation (pose), and their corresponding link lengths are computed using the known inverse kinematics. Due to the existence of multiple solutions (poses) for a particular link vector, a data classification technique is employed to identify various solutions. The classified data are used to find the parameters of a simple model that represents the forward kinematics within a cell. These parameters are stored in a lookup table. During the online phase, given the link lengths, the appropriate cell is identified, the model parameters are retrieved from the lookup table and the poses are computed. The proposed method is tested on a Stewart platform, and the accuracy and online times are presented to show the effectiveness of the proposed method for real-time applications.

Index Terms –Stewart platforms, forward/direct kinematics.

I. INTRODUCTION

Hexapod robots, often referred to as Stewart Platforms, have been the subject of increased attention due to their applications in a variety of fields (see [1] for an extensive review). These applications are brought about mainly due to the rigidity of the structure relative to its size and weight. Areas of the applications include flight and other motion simulators, light weight and high precision machining, data driven manufacturing, dextrous surgery robots, and active vibration control systems for large space structures.

Although mechanical simplicity of Stewart Platforms provides potential for many engineering applications, their forward (direct) kinematics is very complex which limit their real-time applications. This is due to the requirements of solving a set of highly nonlinear equations or high degree polynomials for a general Stewart Platform. Broadly speaking, there are two approaches to the solution of the forward kinematics of the platform, namely analytical and numerical. In the first approach attempts are made to develop closed form solutions in special cases where some of the connection points at the platform or at the base are coalesced in groups of two or three. In this way the general 6-6 configuration, i.e. 6 separate joints at base and platform, is reduced to less than six at the base and/ or the platform [2]-[5]. Another approach in the analytical category is to place restriction on the geometry of the platform or the base, or assume a certain relationship (e.g. linear) between the base and platform coordinates at attachment points [6]-[9].

In the numerical approaches, Newton-Raphson method or a variation of it is used to solve iteratively the set of nonlinear forward kinematics equations. This approach finds only one solution assuming a good starting point, but no feasible solutions is guaranteed [10]-[11]. In order to find all solutions, it is necessary to formulate the problem in the form of polynomial equations and solve these equations numerically. It has been shown that the upper bound on the number of real and complex solutions for a general Stewart platform is forty [12]-[14]. In general, numerical procedures lead to heavy computation burden, and therefore are unsuitable for real-time applications.

The above difficulties have prevented the Stewart platform from being used in many high speed real-time engineering applications. In this paper, we propose a completely different approach to the forward kinematics solution of a general Stewart platform. The method consists of two phases, a preparatory off-line phase and a fast online evaluation phase. In the offline phase to be described in Sections II and III, the space of links (legs) is decomposed into cells and pose-link data sets are generated via the known inverse kinematics. Using this data, forward kinematics are accurately approximated in each cell by simple equations whose parameters are stored. In the online phase, to be explained in Section III, appropriate stored parameters are retrieved and the poses of the platform are determined. Due to the pre-processing offline phase, the online computation is extremely fast, making the method suitable for real-time applications.

II. DECOMPOSITION AND DATA GENERATION

Consider a fixed base B and attach a coordinate frame XYZ to it. Similarly put a coordinate frame xyz to the moving platform A, and let R be the 3×3 rotation matrix which defines the rotating angles of this frame with respect to the fixed frame *XYZ*. Denote by D the displacement vector of the frame *xyz* relative to *XYZ*. Let the position of the links (legs) at the attachment to the base relative to the coordinates *XYZ* be $B_i = (B_{x,i}\ B_{y,i}\ B_{z,i})^T$; $i = 1,2,\cdots,6$. Similarly denote the position of the links at the attachment to the platform with respect to the coordinates *xyz* by $a_i = (a_{x,i}\ a_{y,i}\ a_{z,i})^T$. We can now write the length of each link connecting the base to the platform as

$$\begin{aligned}\ell_i &= \sqrt{\|R\,a_i + D - B_i\|} \qquad i = 1,2,\cdots,6 \\ &= f_i(p)\end{aligned} \tag{1}$$

where $f_i(p)$ is the inverse kinematics function, and p is the platform *pose*. The latter is defined by the vector $p = (x\ y\ z\ \alpha\ \beta\ \gamma)^T$ where (x, y, z) is the coordinates of the origin of the platform and (α, β, γ) is its orientation, both with respect to the base frame. The *pose space* is the six dimensional (6-D) space whose coordinates are the components of p. The *link vector* is defined as $\ell = (\ell_1\ \ell_2 \cdots \ell_6)^T$, and the *link space* is the 6-D space whose coordinates are the components of ℓ. The forward kinematics problem maybe stated as follows: Given a link vector ℓ, determine the set of poses p of the platform that satisfy (1).

In order to solve the forward kinematics of a general Stewart platform, we propose a two phase strategy. In an off-line phase the link space is decomposed into 6-D hypercubes (cells), and the forward kinematics is approximated in each cell by a simple model. This is followed by an online phase that finds the poses for a given link vector. In the following paragraphs and in Section III, we describe the first phase. The link space is divided or decomposed into regular 6-D cells. The cell side length ℓ_c is a design parameter which determines the granularity of the decomposition. The number of divisions along an axis of the link space is

$$N_i = \frac{\ell_{i,max} - \ell_{i,min}}{\ell_c} \qquad i = 1,2,\cdots,6 \tag{2}$$

where $\ell_{i,min}$ and $\ell_{i,max}$ are the minimum and maximum lengths of the i-th link.

We now generate a large number of pose vectors by assigning values randomly within the ranges of x, y, z, α, β and γ . It is noted that regularly spaced intervals within the range of pose components will not produce uniform link space coverage due to the nonlinear forward kinematics mapping. On the other hand random generation of poses provides better coverage of the link space. The ranges of pose components $x, y, z, \alpha, \beta, \gamma$ are either specified for the application for which the platform is used, or in the absence of such data, they can be gross estimates of the ranges values. It is noted that some combinations of the pose vector components may result in infeasible link values. There can be two types of such infeasible values. The first type occurs when substituting a generated pose vector in (1) produces link lengths that are outside their physical range of $\ell_{i,min}$ to $\ell_{i,max}$. The second type is due to mechanical constraints, e.g. limitation on the joints connecting the links to the base or platform, links crossing each other, etc. Mechanical constraints are platform specific and can generally be checked through the knowledge of the mechanical design. In both cases, the infeasible link data and their corresponding poses are removed from the data set and will not be used in modelling

At the conclusion of pose-link data generation and rejection of infeasible data, we have n cells in the link space that contain data points. The set of link data in a cell C_j have corresponding set of poses in a region R_j in the pose space, $j = 1,2,\cdots,n$. Note that while link space cells are regular hypercubes and have the same volume, the corresponding regions in the pose space do not have the same shape or volume. The data must now be processed and stored for the online phase. This will be discussed in the following section.

III. Classification and Curve Fitting

Typical generated data points in a pose region R_j corresponding to a link cell C_j form clusters as shown in Fig. 1. The 6-D pose space is shown in two subspaces, i.e. position and orientation subspaces, for ease of visualization. The units in these two subspaces are milammeters and degrees, respectively. Each cluster shown in a color represents a forward kinematic solution. These solutions must be classified before modelling the forward kinematics in a region.

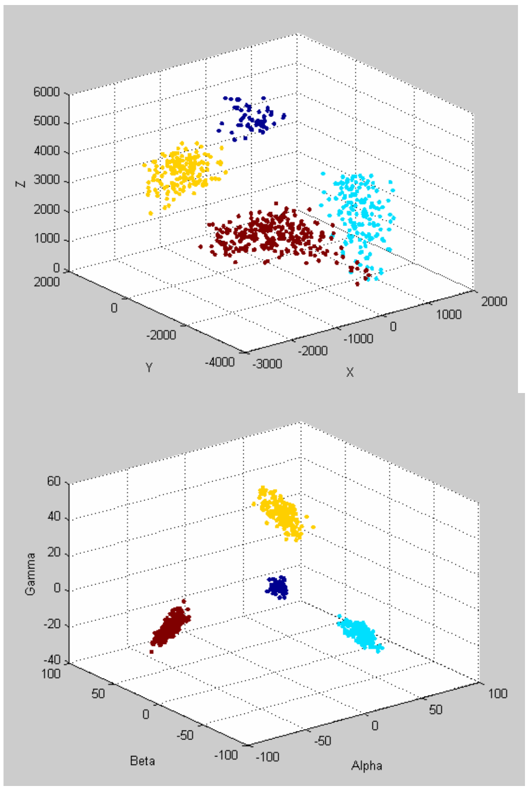

Fig. 1. Solution clusters in a pose region, top – position subspace and bottom – orientation subspace.

There are many techniques for classification of data [15]. Here we apply a modified "*K*-mean" clustering, tailored for our application. The procedure starts with an estimate $K_{\max}$, the maximum number of clusters (real solutions). This estimate is based on the configuration of the platform. It is known that the more general cases of Stewart platforms, i.e. 6-6 configurations, have more solutions than specific cases such as 3-3, or 6-3 configurations. A value of $K_{\max}$ higher than that required produces some identical solutions (poses) as will be seen in Section V. On the other hand, a low value of $K_{\max}$, gives rise to errors when modelling the forward kinematics by simple equations. These considerations can be used during the off-line phase to adjust $K_{\max}$ accordingly. Once $K_{\max}$ is chosen, the stored pose regions $R_j, i = 1,2,\cdots,n$ are searched to find the one with the widest ranges of position and orientation. This region is assigned the maximum number of solution clusters $K_{\max}$. The number of solution clusters K_i in other regions is then decreased in proportion to their ranges of position/orientations. In other words, wider spreading clusters are assumed to represent forward kinematics with more solutions.

We must now classify (assign) each point in a region to an appropriate solution cluster, i.e. identify different solution clusters. Consider a region R_j with n_j poses (points) whose estimated number of solutions is K_j, and pick a random pose $p_{j,1}$ in this region. This pose will be the nucleus of a solution cluster in the region. Now we find the pose in this region that is farthest away from the first picked pose. This second pose, say $p_{j,2}$, will be the nucleus of the second solution cluster. The idea is that far apart poses must belong to different clusters. Next we select the pose $p_{j,3}$ whose sum of distances to $p_{j,1}$ and $p_{j,2}$ is the largest among the $(n_j - 2)$ yet unprocessed poses. The pose $p_{j,3}$ forms the nucleus of the third cluster. We continue this process such that at step $k \le K_j$ the selected pose $p_{j,k}$ has the maximum sum of distances to $p_{j,1}, p_{j,2}, \dots, p_{j,k-1}$. This process yields the K_j nucleuses for the K_j solution clusters. We must assign the remaining $(n_j - K_j)$ poses to theses nucleuses. Now pick an unclassified pose and place it in the nearest cluster and find the mean position and orientation of the two poses in this cluster. The process of placing the remaining poses to the nearest cluster is continued. It is noted that after placing a new pose in a cluster, the mean of position and orientation of that cluster is updated. The closest cluster is the one that has minimum distance between its mean value and the pose being classified. At the conclusion of the classification, each region R_j will have K_j solution clusters. In the example of Fig. 1, four solution clusters have been identified. Note that some mixing that appear in the position clusters are due to the plotting of 3-D information on 2-D page. The two clusters are in fact distinct, with one placed behind the other.

Each cluster in the pose region R_j associated with the link cell C_j must be represented using a simple forward kinematics model. In other words, given a link vector ℓ in C_j, we must express the pose vector p in terms of ℓ for each cluster in R_j. The simplicity or complexity of the required model depends on the size of the link cell ℓ_c. Smaller size cells will require simpler models to accurately represent forward kinematics in them, but the number of cells will be higher since the link pace volume is constant. The opposite is true for larger cell sizes, which require more complex models. We have explored several models including polynomial and neural networks. It was found that a second order (quadratic) polynomial model provides a compromise between complexity of the model and the desired accuracy. The model is expressed as

$$p_i = \ell^T A_i \ell + b_i^T \ell + c_i \qquad i = 1,2,\cdots,6 \tag{3}$$

where p_i are the elements of the pose vector, i.e. $p_1 = x$, $p_2 = y, \cdots, p_6 = \gamma$; A_i is a 6×6 symmetric parameter matrix, b_i is a 6×1 parameter vector, and c_i is a scalar parameter. Since A_i is symmetric, it has only 21 independent parameters, giving a total of 28 parameters in (3). These parameters can be found with a least square (regression) method using the link and pose cluster data. Suppose that the link and pose data for a cluster are denoted by (p_k, ℓ_k), $k = 1,2, \dots, d;$ where d is the number of data points, $P_k = (p_{k1}, \ \cdots, \ p_{k6})^T$ and $L_k = (\ell_{k1}, , \ell_{k2}, \cdots, \ell_{k6})^T$ are pose and link vectors of k-th data point, respectively. In order to determine the parameters of the linear model, the following error function is minimized

$$E = \sum_{k=1}^{d} \sum_{i=1}^{6} (P_{ki} - L_k^T A_i L_k - b_i^T L_k - c_i)^2 \tag{4}$$

A closed form solution to the above minimization exits and can be obtained by setting to zero the derivatives of E with respect to the elements of A_i, b_i and c_i. The resulting linear set of equations is solved to obtain closed-form solutions for A_i, b_i and c_i for each solution cluster in each pose region.

The model parameters are stored as records in a file or for the subsequent online retrieval. Each record has a unique address in the file where the parameters of the forward kinematics model are stored. The address can be encoded in a number of ways. One method is to encode the cell address as

$$C_{adrs} = \sum_{i=0}^{5} e_i \left(\prod_{j=1}^{i} N_j \right) \tag{5}$$

where e_i is in the range in the range of 0 to N_i, the latter is given by (2) and $N_0 = 1$. The encoding (5) is similar to decimal number encoding except that instead of base 10 for all digits we have bases $N_1, N_2, \cdots, N_6$, respectively, for the least to most significant digits. At each address representing a link cell, there is a solution number, followed by the values of model parameters A_i, b_i and c_i for the particular cell. Note that the offline phase is done only once for a platform.

IV. Online Forward Kinematics Computation

During the online phase, given the desired link vector ℓ, the indices needed for computing the address in the file, where the parameters are stored, are computed from

$$e_i = ceil\left(\frac{\ell_i - \ell_{i,min}}{\ell_c}\right) \tag{6}$$

The parameters for each solution region are retrieved and the platform pose is computed using (3). Since computing the address and determining the poses via (3) does not involve trigonometric or inverse trigonometric functions, the online computation is extremely fast, as will be seen in Section V.

It is noted that for some applications, such as flight or other motion simulators, finding the poses with a reasonably good approximation (say with less than 0.1% error) is sufficient. This type of approximation is achievable using the above method, as will be seen in Section V. In case of other applications, more exact values are needed. This can be achieved by finding the approximate solutions as above, and a correction step to arrive at the solution with high accuracy. Suppose that for a desired given link vector ℓ the approximate value of a pose found by the above method is $\hat{p}$. Substituting $\hat{p}$ in the inverse kinematic (1) gives a link vector approximation $\hat{\ell}$. The incremental change $\Delta p_i = (p_i - \hat{p}_i)$ needed in the pose element p_i to achieve the correction of the link vector $\Delta\ell = (\ell - \hat{\ell})$ is obtained from (3) as follows

$$\Delta p_i = \left(2\ell^T A_i + b^T\right)\Delta\ell \qquad i = 1,2,\cdots,6 \tag{7}$$

The correction requires one inverse kinematics evaluation using (1) and finding the increments (7). Since no inverse trigonometric functions, matrix inversion or nonlinear equation solving is involved, the computation of correction is also very fast, as will be seen in the next section. The total online time is $T_{on} = T_1 + T_2 + T_3$ where:

T_1 = Time to compute the lookup table cell address from (5)-(6) for a given link vector, and retrieve the model parameters.

T_2 = Time to compute the position and orientation for each solution via (3).

T_3 = Time to correct the acquired position and orientation using (7). This time consists of three subcomponents, i.e.times for substituting the acquired position and orientation in the inverse kinematics (1), determining the error $\Delta\ell$, and finally computing the correction via (7).

In the following section we will see that for a typical platform, each of the above times is only a few microseconds per solution using a standard PC.

V. Example

The example to be used for investigation of the proposed method is the one reported in [15]. It is a Stewart Platform configuration with six linear actuators which constitute the links. The base is a semiregular hexagon, and the platform is an equilateral triangle. The links are connected to the vertices of the base and platform with two and three degrees freedom universal joints. The side lengths of the base semi-regular hexagon are 15, 1, 15, 1, 15, 1 meters and those of the platform are 10, 10, 10 meters. The minimum and maximum lengths in meters of each link are respectively $\ell_{i,min} = 8$ and $\ell_{i,max} = 15$, $i = 1,2,\cdots,6$. This platform has a regular shape and geometry, but the proposed method does not use any simplifying geometric properties of this platform and treats it just it would a general platform. The reason for the choice of this particular platform was to check our results against those obtained in [16].

The decomposition method described in Section II was applied with the link cell side of $\ell_c = 1.4$ meters. Since the range of the motion of the each link is $\ell_{i,max} - \ell_{i,max} = 7$ meters, there are $7/1.4 = 5$ cells along each of the *six* dimensions of the link space. This produces $(5)^6 = 15625$ cells. Poses were generated randomly in the ranges $-2 < x < 2$, $-2 < x < 2$, $2 < z < 13$ (all ranges in meters), and each of the orientation angles α, β, γ between -60^o and 60^o. Higher pose ranges produced link lengths that were mostly outside limits. The number of generated valid poses in each pose region was about 1200. The maximum number of solution clusters in a region was set to $K_{max} = 8$. All computation was performed on a Pentium 4, 1.8 MHz PC with 512 MB of memory. The total off-line time was about seven hours, which is done only once.

We first compare the results of our method with those of [16] where exact analytical solutions were reported for extreme configurations. The results are summarized in Table 1 and in all case the correction mentioned in the previous section was implemented. The upper three numbers in Table 1 cells were obtained by the proposed method and the lower three numbers are the exact values reported in [16]. The "Lowest Position" in the table refers to the minimum link values, i.e. $\ell_1 = \ell_2 = \cdots = \ell_6 = 8$ meters. The values found by the proposed method are exactly the same as the results of the analytical solutions. The same conclusion is made for the case of "highest position" with $\ell_1 = \ell_2 = \cdots = \ell_6 = 15$ meters. The other two cases are the "Most Tilted" where $\ell_1 = \ell_2 = 15$, $\ell_3 = \ell_4 = \ell_5 = \ell_6 = 8$, and the "Most Twisted" where $\ell_1 = \ell_3 = \ell_5 = 8$ and $\ell_2 = \ell_4 = \ell_6 = 15$. In both these cases the results are also very close to the analytical solutions.

TABLE 1 COMPARISON OF METHODS OF FINDING EXTREME CONFIGUARTIONS

Configuration	Position x,y,z (m)	Orientation α, β, γ (deg)
Lowest Position	0.000, 0.000, 2.646	0.00, 0.00, 0.00
	0.000, 0.000, 2.646	0.00, 0.00, 0.00
Highest Position	0.000, 0.000, 12.961	0.00, 0.00, 0.00
	0.000, 0.000, 12.960	0.00, 0.00, 0.00
Most Tilted	-1.239, -2.139, 5.506	58.99, -74.00, -46.07
	-1.236, -2.142, 5.503	58.99, -73.84, -46.06
Most Twisted	0.000, 0.000, 7.194	0.00, 0.00, 68.34
	0.000, 0.000, 7.192	0.00, 0.00, 68.36

In order to test the performance and accuracy of the method for a large set of data, we generated 1000 link vectors whose elements were selected randomly between the link length ranges of *8* and *15* meters. In order to find the accuracy of the solutions, the following procedure was adopted. For each of the 1000 desired link vector ℓ, the pose values were found using the proposed method. These were then substituted in (1) to get their corresponding approximated link value $\hat{\ell}$. The error E reported in Table 2 is the average absolute error $|\ell - \hat{\ell}|$ for the 1000 trials with and without correction using (7). The error standard deviation is shown under the column σ in Table 2. It is seen the errors are small especially with correction. In the latter case, the average absolute error is *0.00124* meters, or $(0.00124)/7 = 0.017\%$ of full range. Even without the correction, the errors are small. The memory needed to store the parameters was 18.2 MB, which is reasonable considering that a standard PC has 512 MB to 1 GB of memory. The success rate is the number of valid link values after finding the pose values and substituting them back into (1).

The times reported in Table 2, are defined in Section IV, i.e. T_1 is required for parameter retrieval from the lookup table (file), T_2 is for pose computation, and T_3 is the time for correction, all measured in μs (10^{-6} second). Most of the total time T_{on} in the case of correction is taken up by computing the inverse kinematics that involves trigonometric functions. Even with the correction the total online time T_{on} is extremely small. For example, even if the platform has 20 pose solutions for each link vector, all solutions can be computed in about 0.3 milliseconds.

TABLE 2 RESULTS OF 1000 TRIALS

Correction	E (m)	σ (m)	M (MB)	S (%)
Yes	0.00124	0.0038	18.2	97
No	0.00478	0.0089	18.2	96

Correction	T_1 (μs)	T_2 (μs)	T_3 (μs)	T_{on} (μs)
Yes	1.3	1.4	11.6	14.3
No	1.3	1.4	0	2.7

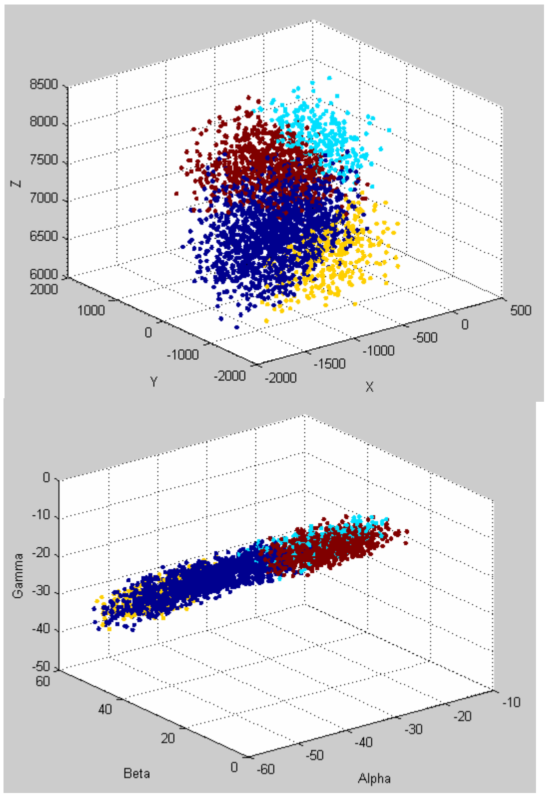

Fig. 2 Using $K_{max} = 8$, four clusters are identified in this region. Shown are position (top) and orientation subspaces

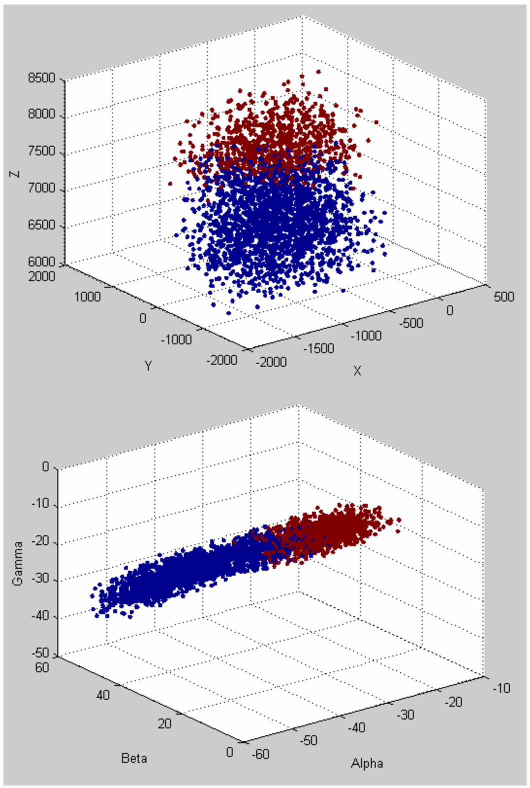

Fig. 3 Using $K_{max} = 4$, two clusters are identified. Shown are position (top) and orientation subspaces.

TABLE 3 COMPARISON OF RESULTS IN TWO CASES

$K_{\max}$	Position x,y,z (m)	Orientation α,β,γ (deg)
4	-1.7297 -0.8595 7.0496 -1.2639 0.3873 8.0563	-46.03 58.50 -39.46 -28.42 21.01 -14.85

8	-1.7296 -0.8588 7.0500
	-1.2638 0.3876 8.0565
	-1.7296 -0.8587 7.0501
	-1.2639 0.3873 8.0563
		-46.03 58.49 -39.45
		-28.41 21.00 -14.84
		-46.02 58.48 -39.45
		-28.42 21.01 -14.85

Finally, we discuss the effect of reducing the maximum number of clusters $K_{\max}$. In an experiment, we chose $K_{max} = 4$ instead of $K_{max} = 8$. In this case the accuracy of solutions was not much affected. In other words, the data presented in Tables 1 and 2 were similar for the two $K_{\max}$ values. This indicates that for the majority of link values there are no more than four solutions. An investigation of the solutions revealed that with $K_{max} = 8$, we obtained repeated (identical) solutions in many regions. A typical example of this situation is shown in Fig. 2 where the choice of $K_{max} = 8$ has produced four solution clusters in the region, and in Fig. 3 with $K_{max} = 4$ where only two clusters have been formed in the same region. This means that in the latter case, a pair of clusters has been combined into one cluster. Table 3 shows the poses found by the method for a randomly selected $\ell = (9.249, 8.574, 11.200, 13.270, 11.763, 13.327)^T$ in the link cell corresponding to the pose region shown in Fig. 2 or 3. It is seen that $K_{max} = 8$ produced four solutions in which solutions 1 and 3 were almost identical and so were solutions 2 and 4. In such a case, redundant solutions are eliminated. The error measured in the link value due to the approximation in all cases reported in Table 3 was less than *0.002* meter or *0.03%* of full range.

VI. Conclusions

A new approach to the forward kinematics solutions of a general Stewart platform is introduced that is extremely fast and is suitable for real-time applications. Unlike the analytical methods that are restricted to special types of platforms, the proposed approach is applicable to a general platform without placing any restrictions on the geometry or link connection points. Furthermore, it does not suffer from the problems associated with numerical methods of solving a set of nonlinear or polynomial equations, such as local minima, sensitivity to the initial values, finding only one solution, and solutions with imaginary part. In fact due to the particular method of data generation, only valid link-pose data are considered for modelling, and multiple solutions are generated and classified. The accuracy of the solutions is very good, and only modest memory is required for storing the model parameters. Finally, the method is very flexible in the sense that depending on the desired level of accuracy and online computation time, the cell size can be adjusted accordingly.

We have successfully applied the method to several platforms, one of which is reported in this paper. A possible limitation of the method is dealing with platforms that have a large number of solutions, e.g. 16 or more distinct real solutions. In such cases, more data must be generated so that each solution cluster has enough data points for curve fitting. This adds to the offline processing and requires more memory to store the model parameters. However, practical platforms with more than 16 distinct real solutions are rare.

References

[1] B. Dasgupta and T.S. Mruthyunjaya, "The Stewart Platform manipulators: A review", Mechanism and Machine Theory, vol. 35, no. 1, pp. 15-40, 2000.
[2] M. Griffis and J. Duffy, "A forward displacement analysis of a class of Stewart platforms," J. Robot. Syst. , vol. 6, no. 6, pp. 703-720, 1989.
[3] C. Innocenti, and V. Parenti-Castelli, "Direct position analysis of the Stewart platform mechanism, " Mech. Mach. Theory, vol. 26, no. 6, pp. 611-621, 1990.
[4] M.S. Tsai, T. N. Shiau and Y. J. Tsai, "Direct kinematic analysis of a 3-PRS parallel mechanism," Mech. Mach. Theory, vol. 38, pp. 71-83, 2003.
[5] I.D. Akcali and H. Mutlu, "A novel approach in the direct kinematics of Steward platform with planar platforms," ASME Trans. J. Mech. Design, vol. 128, pp. 252-263, 2006.
[6] G. Wang, "Forward displacement analysis of a class of 6-6 Stewart platforms," Robot., Spatial Mechan. Syst. , vol. 45, pp. 113-117, 1992.
[7] S.V. Sreenivasan, K. J. Waldron, and P. Nanua, "Closed form direct displacement analysis of a class of a 6-6 platform," Mechanisms and Machine Theory, vol. 29, no. 6, pp. 855-868, 1994.
[8] J. Yang and Z. J. Geng, "Closed form forward kinematics solution of a class of hexapod robots," IEEE Trans. Robot. Automat., vol. 14, pp. 503-508, 1998.
[9] P. Ji and H. Wu, "A closed form forward kinematics solution for 6-6P Stewart platform," IEEE Trans. Robotics and Automation, vol. 17, no. 4, pp. 522-526, 2001.
[10] D. M. Ku, "Direct displacement analysis of a Stewart platform mechanism," Mech. Mach. Theory, vol. 34, pp. 453-465, 1999.
[11] L.-C. Wang and C.C. Chen, IEEE Trans, Robot Automn., vol. 9, no. 3, pp. 272-285, 1993.
[12] M. Raghaven, "The Stewart platform of general geometry has 40 configurations," ASME J. Mech. Design, vol. 115, pp. 277-282, 1993.
[13] M.L. Husty, "An algorithm for solving the direct kinematics of general Stewart-Gough platforms," Mech. Mach. Theory, vol. 31, no. 4, pp. 365-379, 1996.
[14] P. Dietmair, "The Stewart-Gough platform of general geometry can have 40 real postures," Advances in Robot Kinematics: Analysis and Control, J. Lenarcic and M.L. Hust (eds), pp. 7-16, Kluwer Academic Publications, 1998.
[15] R. O. Duda, P.E. Hart and D. G. Stork, Pattern Classification, 2nd edition, Wiley 2001.
[16] K. Liu, J.M. Fitzgerald and F.L. Lewis, "Kinematics analysis of a Stewart platform Manipulator," IEEE Trans. Industrial Electronics, vol. 40, no. 2, pp. 282-293, 1993.