EFFICIENT COMPUTATION OF FORWARD KINEMATICS AND JACOBIAN MATRIX OF A STEWART PLATFORM-BASED MANIPULATOR

Charles C. Nguyen[*] **Zhen-Lei Zhou**[†] **Sami S. Antrazi**[†]
Robotics and Control Laboratory
Department of Electrical Engineering
Catholic University of America
Washington DC, 20064

Charles E. Campbell, Jr.
NASA/Goddard Space Flight Center
Greenbelt, Maryland 20771

Abstract

In this paper, we consider the problem of efficient computation of the forward kinematics of a 6 DOF robot manipulator built to study autonomous assembly of parts. The manipulator, designed based on the Stewart Platform mechanism consist mainly of 2 platforms and six linear actuators. The closed-form solution of the inverse kinematics is formulated in such a way to optimize the computation efficiency of the iterative solution of the forward kinematics using Newton-Raphson Method. A Modified Jacobian Matrix which relates length velocities to Cartesian translational velocities and time rates of change of Roll-Pitch-Yaw angles is introduced. Computer simulation is performed to evaluate the computation efficiency of the developed computation schemes.

Introduction

In the last several years, there has been an increasing interest in applications of the Stewart Platform [1] in developing robotic devices [2,10]. The Stewart Platform mechanism has been found to be suitable for situations where the requirements for accuracy and sturdiness outweigh those on workspace and maneuverability. Stewart Platform-based manipulators generally possess high positioning capability provided by their high structural rigidity and nonserial accumulation of actuator errors. Furthermore, they also have higher strength-to-weight ratios as compared to conventional open-kinematic chain manipulators because payload is proportionally distributed to their links.

Recently, based on the mechanism of the Stewart Platform, a robot manipulator was built at the Goddard Space Flight Center (NASA) to study the feasibility of autonomous assembly of parts in space. This paper considers the development of efficient computation schemes for the implementation of the manipulator forward kinematics. The paper is structured as follows. Next section gives a general description of the main components of the robotic manipulator. Then a kinematic analysis is performed to provide a closed-form solution to the inverse kinematics. After that, an efficient computation scheme using Newton-Raphson method is proposed for the manipulator forward kinematics and a Modified Jacobian Matrix is introduced. Finally we present results of computer simulation conducted to examine the efficiency and accuracy of the developed computation scheme.

The Robot Manipulator

This section is devoted to briefly describe the main components of the robot manipulator. As shown in Figure 1, the robot manipulator is designed based on the Stewart Platform mechanism and mainly consists of a lower base platform, an upper payload platform, a compliant platform, a gripper and six linear actuators. The movable payload platform is supported above the stationary base platform by six axially extensible rods with ballnuts and ballscrews providing the extensibility. Stepper motors were selected to drive the ballscrews to extend or shorten the actuator lengths whose variations will in turn produce the motion of the payload platform and consequently the motion of a gripper attached to the compliant platform. Each end of the actuator links is mounted to the platforms by 2 rotary joints whose axes intersect and are perpendicular to each other. Passive compliance is provided through the compliant platform, which is suspended from the payload platform by six spring-loaded pistons arranged in the Stewart Platform mechanism. The compliance is passively provided by permitting strain on two opposing springs acting in the pistons. Thus the pistons are compressed and extended when resistive and gravitational forces are applied to the gripper. The rotation of each stepper motor is controlled by sending out proper commands to an indexer which then transmits proper pulse sequences to the stepper motor drive. Therefore, precise gripper motion can be produced by properly controlling the motions of six manipulator legs.

The Inverse Kinematics

This section deals with the inverse kinematic transformation for the robot manipulator, which determines the required actuator lengths for a given configuration composed of Cartesian position and orientation of the payload platform with respect to the base platform. Frame assignment to the robot manipulator is illustrated in Figure 2 where two coordinate frames $\{\mathbf{P}\}$, and $\{\mathbf{B}\}$ are assigned to the payload and base platforms, respectively. The origin of Frame $\{\mathbf{P}\}$ is located at the centroid P of the payload platform, the $\mathbf{z}_P$-axis is pointing outward and the $\mathbf{x}_P$-axis is perpendicular to the line connecting the two attachment points P_1 and P_6. The angle between P_1 and P_2 is denoted by θ_P. A symmetrical distribution of joints on the payload platform is achieved by setting the angles between P_1 and P_3 and between P_3 and P_5 to 120^o. Similarly, Frame $\{\mathbf{B}\}$ has its origin at the centroid B of the base platform. The $\mathbf{x}_B$-axis is perpendicular to the line connecting the two attachment points B_1 and B_6 the angle between B_1 and B_2 is denoted by θ_B. Also the angles between B_1 and B_3 and between B_3 and B_5 are set to 120^o in order to symmetrically distribute the joints on the base platform. The Cartesian variables are chosen to be the relative position and orientation of Frame $\{\mathbf{P}\}$ with respect to Frame $\{\mathbf{B}\}$ where the position of Frame $\{\mathbf{P}\}$ is specified by the position of its origin with respect to Frame $\{\mathbf{B}\}$. Now if we denote the angle between PP_i and $\mathbf{x}_P$ by λ_i, and the angle between BB_i and $\mathbf{x}_B$ by Λ_i

[*]Associate Professor
[†]Graduate Research Assistant

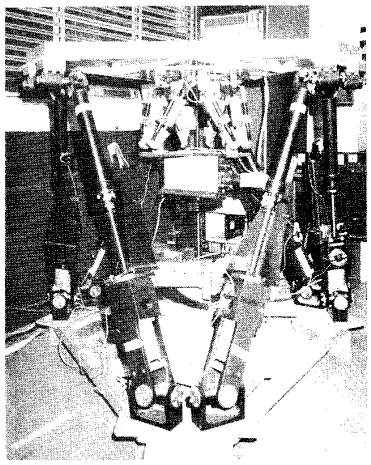

Figure 1: The robot manipulator

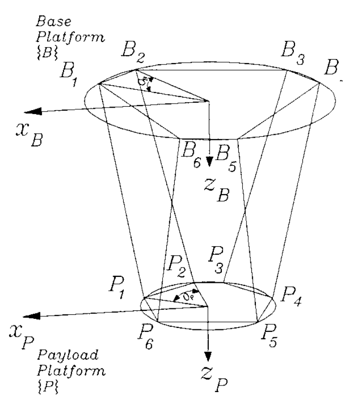

Figure 2: Frame assignment of the platforms

for i=1,2,...,6, then by inspection we obtain

$$\Lambda_i = 60i^o - \frac{\theta_B}{2};\ \lambda_i = 60i^o - \frac{\theta_P}{2}, \quad \text{for i} = 1,3,5 \tag{1}$$

and

$$\Lambda_i = \Lambda_{i-1} + \theta_B;\ \lambda_i = \lambda_{i-1} + \theta_P, \quad \text{for i} = 2,4,6. \tag{2}$$

Furthermore, if Vector ${}^P\mathbf{p}_i = (p_{ix}\ p_{iy}\ p_{iz})^T$ describes the position of the attachment point P_i with respect to Frame $\{\mathbf{P}\}$, and Vector ${}^B\mathbf{b}_i = (b_{ix}\ b_{iy}\ b_{iz})^T$ the position of the attachment point B_i with respect to Frame $\{\mathbf{B}\}$, then they can be written as

$$^P\mathbf{p}_i = \begin{bmatrix} r_P cos(\lambda_i) & r_P sin(\lambda_i) & 0 \end{bmatrix}^T \tag{3}$$

and

$$^B\mathbf{b}_i = \begin{bmatrix} r_B cos(\Lambda_i) & r_B sin(\Lambda_i) & 0 \end{bmatrix}^T \tag{4}$$

for i=1,2,...,6 where r_P and r_B represent the radii of the payload and base platforms, respectively.

We proceed to consider the vector diagram for an ith actuator given in Figure 3. The position of Frame $\{\mathbf{P}\}$ is represented by Vector ${}^B\mathbf{d} = [x\ y\ z]^T$ which contains the Cartesian coordinates x, y, z of the origin of Frame $\{\mathbf{P}\}$ with respect to Frame $\{\mathbf{B}\}$. The length vector ${}^B\mathbf{q}_i = (q_{ix}\ q_{iy}\ q_{iz})^T$, expressed with respect to Frame $\{\mathbf{B}\}$ can be computed by

$$^B\mathbf{q}_i = {}^B\mathbf{x}_i + {}^B\mathbf{p}_i \tag{5}$$

where

$$^B\mathbf{x}_i = {}^B\mathbf{d} - {}^B\mathbf{b}_i \tag{6}$$

$$= \begin{bmatrix} x - b_{ix} \\ y - b_{iy} \\ z - b_{iz} \end{bmatrix} = \begin{bmatrix} x - b_{ix} \\ y - b_{iy} \\ z \end{bmatrix} = \begin{bmatrix} \bar{x}_i \\ \bar{y}_i \\ \bar{z}_i \end{bmatrix} \tag{7}$$

which is a shifted vector of ${}^B\mathbf{d}$ and

$$^B\mathbf{p}_i = {}^B_P\mathbf{R}\ {}^P\mathbf{p}_i \tag{8}$$

$$= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} p_{ix} \\ p_{iy} \\ p_{iz} \end{bmatrix} = \begin{bmatrix} r_{11}p_{ix} + r_{12}p_{iy} \\ r_{21}p_{ix} + r_{22}p_{iy} \\ r_{31}p_{ix} + r_{32}p_{iy} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} \tag{9}$$

which is the representation of ${}^B\mathbf{p}_i$ in Frame $\{\mathbf{B}\}$ and ${}^B_P\mathbf{R}$ is the *Orientation Matrix* representing the orientation of Frame $\{\mathbf{P}\}$ with respect to Frame $\{\mathbf{B}\}$.

Thus the length l_i of Vector ${}^B\mathbf{q}_i$ can be computed from its components as

$$l_i = \sqrt{q_{ix}^2 + q_{iy}^2 + q_{iz}^2} \tag{10}$$

or

$$l_i = \sqrt{(\bar{x}_i + u_i)^2 + (\bar{y}_i + v_i)^2 + (\bar{z}_i + w_i)^2} \tag{11}$$

Using (3)-(4) and properties of orientation matrix, (10) can be rewritten as

$$\begin{aligned} l_i^2 &= x^2 + y^2 + z^2 + r_P^2 + r_B^2 + 2(r_{11}p_{ix} + r_{12}p_{iy})(x - b_{ix}) \\ &\quad +2(r_{21}p_{ix} + r_{22}p_{iy})(y - b_{iy}) \\ &\quad +2(r_{31}p_{ix} + r_{32}p_{iy})z - 2(xb_{ix} + yb_{iy}), \end{aligned} \tag{12}$$

for i=1,2,...,6.

Equation (12) represents the *closed-form* solution to the inverse kinematic problem in the sense that required actuator lengths l_i for i=1,2,...,6 can be determined using (12) to yield a given Cartesian configuration composed of Cartesian position and orientation of Frame $\{\mathbf{P}\}$ with respect to Frame $\{\mathbf{B}\}$.

The orientation of Frame $\{\mathbf{P}\}$ with respect to Frame $\{\mathbf{B}\}$ can be described by the orientation matrix ${}^B_P\mathbf{R}$ as shown in (9) which requires nine variables r_{ij} for i,j=1,2,3 from which six are redundant because only three are needed to specify an orientation. There exist several ways to specify an orientation by three variables, but the most widely used one is the Roll-Pitch-Yaw angles α, β, and γ, which represent the orientation of Frame $\{\mathbf{P}\}$, obtained after the following sequence of rotations from Frame $\{\mathbf{B}\}$, first rotate Frame $\{\mathbf{B}\}$ about the $\mathbf{x}_B$-axis an angle γ (*Yaw*), then rotate the resulting frame about the $\mathbf{y}_B$-axis an angle β (*Pitch*), finally rotate the resulting frame about the $\mathbf{z}_B$-axis an angle α (*Roll*).

The orientation represented by the above Roll-Pitch-Yaw angles is given by [1]

$$^B_P\mathbf{R} = \mathbf{R}_{RPY} = \begin{bmatrix} c\alpha\, c\beta & c\alpha\, s\beta\, s\gamma - s\alpha\, c\gamma & c\alpha\, s\beta\, c\gamma + s\alpha\, s\gamma \\ s\alpha\, c\beta & s\alpha\, s\beta\, s\gamma + c\alpha\, c\gamma & s\alpha\, s\beta\, c\gamma - c\alpha\, s\gamma \\ -s\beta & c\beta\, s\gamma & c\beta\, c\gamma \end{bmatrix}. \tag{13}$$

The Forward Kinematics

This section considers the development of the forward transformation which transforms the actuator lengths l_i for i=1,2,...,6 into the Cartesian position and orientation of the payload platform with respect to the base platform. The forward kinematic problem can be formulated as to find a Cartesian position specified by x, y, z and an orientation specified by Roll-Pitch-Yaw angles α, β, and γ to satisfy Equation (12) for a given set of actuator lengths l_i for i=1,2,...,6. In general, there exists no closed-form solution for the above problem since Equation

[1] $c\alpha \equiv \cos\alpha$, and $s\alpha \equiv \sin\alpha$.

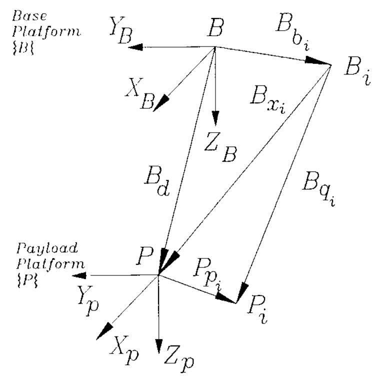

Figure 3: Vector diagram for the ith actuator

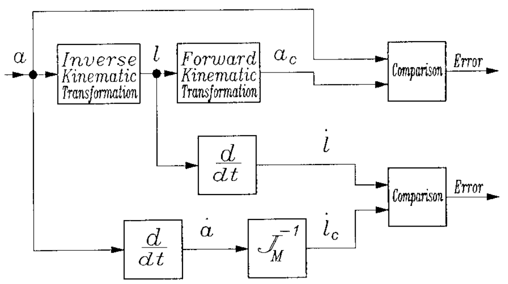

Figure 4: Computer simulation scheme

(12) represents a set of 6 highly nonlinear simultaneous equations with 6 unknowns. Consequently iterative numerical methods must be employed to solve the above set of nonlinear equations. In the following we will present the implementation of Newton-Raphson method for solving the forward kinematic problem.

In order to apply the Newton-Raphson method, first from (11) we define 6 scalar functions

$$f_i(\mathbf{a}) = (\bar{x}_i + u_i)^2 + (\bar{y}_i + v_i)^2 + (\bar{z}_i + w_i)^2 - l_i^2 = 0 \qquad (14)$$

for i=1,2,...,6, where the vector $\mathbf{a}$ is defined as

$$\mathbf{a} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{bmatrix}^T = \begin{bmatrix} x & y & z & \alpha & \beta & \gamma \end{bmatrix}^T, \qquad (15)$$

and then employ the following algorithm [11] to solve for $\mathbf{a}$:

Forward Kinematic Algorithm

Step 1: Select an initial guess $\mathbf{a}$.

Step 2: Compute the elements r_{ij} of ${}^B_P\mathbf{R}$ using (13) for i,j = 1,2,3.

Step 3: Compute $\bar{x}_i$,$\bar{y}_i$, $\bar{z}_i$, using (7) and u_i, v_i, w_i using (9) for i=1,2,...,6.

Step 4: Compute $f_i(\mathbf{a})$ and $A_{ij} = \frac{\partial f_i}{\partial a_j}$ using (14) for i, j=1,2,...,6.

Step 5: Compute B_i = -$f_i(\mathbf{a})$ for i=1,2,...,6. If $\sum_{j=1}^{6} \mid B_j \mid < tolf$ (tolerance), stop and select $\mathbf{a}$ as the solution.

Step 6: Solve $\sum_{j=1}^{6} A_{ij}\delta a_j = B_i$ for δa_j for i,j=1,2,...,6 using LU decomposition. If $\sum_{j=1}^{6} \delta a_j < tola$ (tolerance), stop and select $\mathbf{a}$ as the solution.

Step 7: Select $\mathbf{a}^{new} = \mathbf{a} + \delta\mathbf{a}$ and repeat Steps 1-7.

Computation of Partial Derivatives

In order to minimize the computational time of the Forward Kinematic Algorithm, the expressions for computing the partial derivatives in Step 4 of the algorithm should be simplified. First using (9) and (13), the partial derivatives of u_i, v_i, and w_i with respect to the angles α, β, and γ can be computed as follows:

$$\frac{\partial u_i}{\partial \alpha} = -v_i; \quad \frac{\partial u_i}{\partial \beta} = c\alpha\, w_i; \quad \frac{\partial u_i}{\partial \gamma} = p_{iy}\, r_{13}, \qquad (16)$$

$$\frac{\partial v_i}{\partial \alpha} = u_i; \quad \frac{\partial v_i}{\partial \beta} = s\alpha\, w_i; \quad \frac{\partial v_i}{\partial \gamma} = p_{iy}\, r_{23}, \qquad (17)$$

$$\frac{\partial w_i}{\partial \alpha} = 0; \quad \frac{\partial w_i}{\partial \beta} = -(c\beta\, p_{ix} + s\beta\, s\gamma\, p_{iy}); \quad \frac{\partial w_i}{\partial \gamma} = p_{iy}\, r_{33}. \qquad (18)$$

From (7), we note that

$$\frac{\partial \bar{x}_i}{\partial x} = \frac{\partial \bar{y}_i}{\partial y} = \frac{\partial \bar{z}_i}{\partial z} = 1. \qquad (19)$$

Employing (16)-(19), we obtain after intensive simplification

$$\frac{\partial f_i}{\partial a_1} = \frac{\partial f_i}{\partial x} = \frac{\partial f_i}{\partial \bar{x}_i} = 2(\bar{x}_i + u_i), \qquad (20)$$

$$\frac{\partial f_i}{\partial a_2} = \frac{\partial f_i}{\partial y} = \frac{\partial f_i}{\partial \bar{y}_i} = 2(\bar{y}_i + v_i), \qquad (21)$$

$$\frac{\partial f_i}{\partial a_3} = \frac{\partial f_i}{\partial z} = \frac{\partial f_i}{\partial \bar{z}_i} = 2(\bar{z}_i + w_i), \qquad (22)$$

$$\frac{\partial f_i}{\partial a_4} = \frac{\partial f_i}{\partial \alpha} = 2(-\bar{x}_i v_i + \bar{y}_i u_i), \qquad (23)$$

$$\frac{\partial f_i}{\partial a_5} = \frac{\partial f_i}{\partial \beta} = 2[(-\bar{x}_i\, c\alpha + \bar{y}_i\, s\alpha) w_i - (p_{ix}\, c\beta + p_{iy}\, s\beta\, s\gamma)\bar{z}_i] \qquad (24)$$

$$\frac{\partial f_i}{\partial a_6} = \frac{\partial f_i}{\partial \gamma} = 2p_{iy}(\bar{x}_i r_{13} + \bar{y}_i r_{23} + \bar{z}_i r_{33}). \qquad (25)$$

Modified Jacobian Matrix

Conventionally the Manipulator Jacobian Matrix is defined as a matrix relating joint velocities to Cartesian velocities composed of translational velocities and rotational velocities. For the robot manipulator, since actuator lengths are selected as joint variables, the time rates of change of actuator lengths $\dot{l}_1$, $\dot{l}_2$,..., $\dot{l}_6$ are joint velocities. However in order to utilize the partial derivatives computed for the forward kinematic transformation, we define here the velocities of Cartesian positions of the payload platform with respect to Frame {**B**}, namely $\dot{x}$, $\dot{y}$ and $\dot{z}$ as the *translational velocities* and the velocities of the Roll-Pitch-Yaw angles $\dot{\alpha}$, $\dot{\beta}$, and $\dot{\gamma}$ as the *rotational velocities*. The matrix which relates the length velocities to translational velocities and Roll-Pitch-Yaw angle velocities is therefore called *The Modified Jacobian Matrix*. Denoting

$$\dot{\mathbf{a}} = (\dot{a}_1\ \dot{a}_2\ \dot{a}_3\ \dot{a}_4\ \dot{a}_5\ \dot{a}_6)^{\mathrm{T}} = (\dot{x}\ \dot{y}\ \dot{z}\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma})^{\mathrm{T}}, \qquad (26)$$

and

$$\dot{\mathbf{l}} = (\dot{l}_1\ \dot{l}_2\ \dot{l}_3\ \dot{l}_4\ \dot{l}_5\ \dot{l}_6)^{\mathrm{T}}, \qquad (27)$$

we obtain

$$\dot{\mathbf{a}} = \mathbf{J}_M\, \dot{\mathbf{l}}, \qquad (28)$$

or

$$\dot{\mathbf{l}} = \mathbf{J}_M{}^{-1}\,\dot{\mathbf{a}} \tag{29}$$

where $\mathbf{J}_M$ is the Modified Jacobian Matrix. Calling $k_{ij} = \frac{\partial l_i}{\partial a_j}$, the ij-element of $\mathbf{J}_M{}^{-1}$, from (29) we have

$$\dot{l}_i = \sum_{j=1}^{6} k_{ij}\dot{a}_j = \sum_{j=1}^{6} \frac{\partial l_i}{\partial a_j}\dot{a}_j. \tag{30}$$

Now solving for l_i^2 in (14) yields

$$l_i^2 = (\bar{x}_i + u_i)^2 + (\bar{y}_i + v_i)^2 + (\bar{z}_i + w_i)^2 = \bar{f}_i \tag{31}$$

for i=1,2,...,6. Recognizing that $\bar{f}_i$ is a function of $\bar{x}_i$, $\bar{y}_i$, $\bar{z}_i$, α, β, and γ, and using (19), we differentiate both sides of (30) with respect to time to obtain

$$2l_i\,\dot{l}_i = \sum_{j=1}^{6} \frac{\partial \bar{f}_i}{\partial a_j}\dot{a}_j \tag{32}$$

from which solving for $\dot{l}_i$ yields

$$\dot{l}_i = \sum_{j=1}^{6} \frac{1}{2l_i}\frac{\partial \bar{f}_i}{\partial a_j}\dot{a}_j. \tag{33}$$

Now comparing (30) and (33) and noting from (31) and (14) that $\frac{\partial f_i}{\partial a_j} = \frac{\partial \bar{f}_i}{\partial a_j}$, we arrive at

$$k_{ij} = \frac{1}{2l_i}\frac{\partial f_i}{\partial a_j} \tag{34}$$

where $\frac{\partial f_i}{\partial a_j}$ can be obtained from Step 4 of the Forward Kinematic Algorithm using (20)-(25). In other words, we just showed that the inverse of the Modified Jacobian Matrix can be computed using the results of the forward kinematic transformation.

Computer Simulation Results

In this section we will report results obtained from the computer simulation conducted to study the efficiency of the developed inverse and forward kinematic transformations as well as the Modified Jacobian matrix. The simulation scheme employed in the study is illustrated in Figure 4. In the upper loop, a set of Cartesian test trajectories comprised by Vector **a** are converted to the corresponding actuator length trajectories comprised by Vector **l** via the inverse kinematic transformation. The Forward Kinematic Algorithm implementing the forward kinematic transformation is then applied to convert **l** to $\mathbf{a}_c$, a vector composed of computed Cartesian trajectories corresponding to **l**. The computed Cartesian trajectories are then compared with the Cartesian test trajectories and the resulting errors are recorded. In addition, the test length velocities contained by $\dot{\mathbf{l}}$ are obtained by differentiating **l** with respect to time. In the lower loop, the Cartesian test velocities comprised by Vector $\dot{\mathbf{a}}$ are obtained by differentiating **a** with respect to time. The Cartesian test velocities $\dot{\mathbf{a}}$ are then converted to the corresponding length velocities, $\dot{\mathbf{l}}_c$ using the Inverse Modified Jacobian Matrix $\mathbf{J}_M{}^{-1}$. Errors in length velocities are then obtained by comparing the computed length velocities with the test length velocities. The developed transformations are implemented in C and graphical facility is provided by MATLAB. Computer simulation results for two test cases are presented and discussed below. The manipulator parameters used in the computer simulation are r_P = 22.238 inches, r_B = 29.267 inches, θ_p = 95.908° θ_b = 15.722°.

Test Case 1: Straight Line Motion

Figures 5-7 present the computer simulation results of the case in which the Cartesian test trajectories specify a straight line in the x-y plane of the base frame. The straight line motion is described by

$$x(t) = x_0 + 18[1 + 3\exp(-\frac{3.5}{7.5}t) - 4\exp(-\frac{3.5}{10}t)] \tag{35}$$

and

$$y(t) = y_0 + 21.6[1 + 3\exp(-\frac{3.5}{7.5}t) - 4\exp(-\frac{3.5}{10}t)] \tag{36}$$

where the initial position is denoted by x_0 = -9 inches, y_0 = -10 inches. The computer simulation was conducted with a sampling time of 0.05 second on a SUN workstation for 10 seconds. According to Figure 5 which presents the errors in Cartesian positions x, y, z, a maximum error of 4.292 microinch and a maximum Root-Mean-Square (**RMS**) error of 1.5 microinch both in x-position. The errors in Roll-Pitch-Yaw (**RPY**) angles are showed in Figure 6 where a maximum error of -0.177 microradian and a maximum RMS error of 0.073 microradian occur in the Roll angle. According to Figure 7 which presents the errors in length velocities, relatively large errors exist at the beginning of the simulation and settle down almost to zero after about 1sec. A maximum error of 0.1913 inch/sec and a maximum RMS error of 0.0365 inch/sec occur in the second actuator length.

Test Case 2: Sinusoidal Motion

Computer simulation results of the case in which the Cartesian test trajectories specify a sinusoidal motion are presented in Figures 8-10. The sinusoidal motion consists of 3 segments described by

$$x(t) = x_0 + \frac{1}{2}at^2 \text{ for } t_0 \le t \le t_1 \tag{37}$$

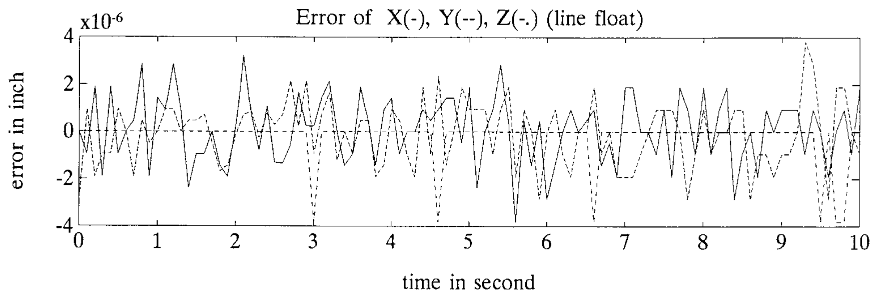

Figure 5: Straight line motion, x, y, z, errors
x (solid), y (dashed), z (dashed-dotted)

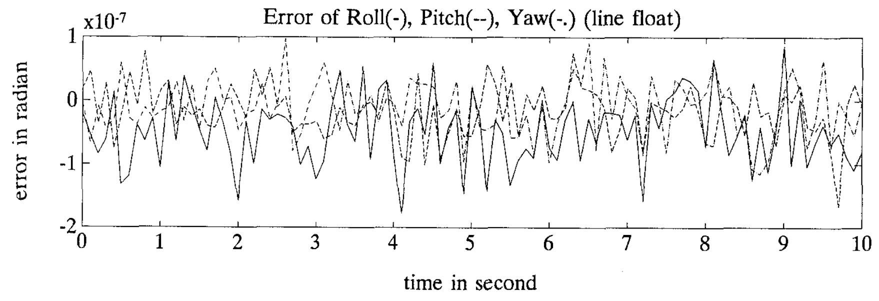

Figure 6: Straight line motion, RPY angle errors
α (solid), β (dashed), γ (dashed-dotted)

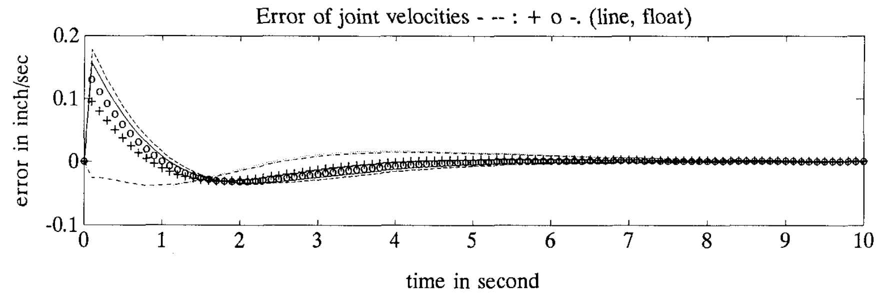

Figure 7: Straight line motion, length velocity errors
$\dot{l}_1$ (solid), $\dot{l}_2$ (- -), $\dot{l}_3$ (..), $\dot{l}_4$ (++), $\dot{l}_5$ (oo), $\dot{l}_6$ (-.)

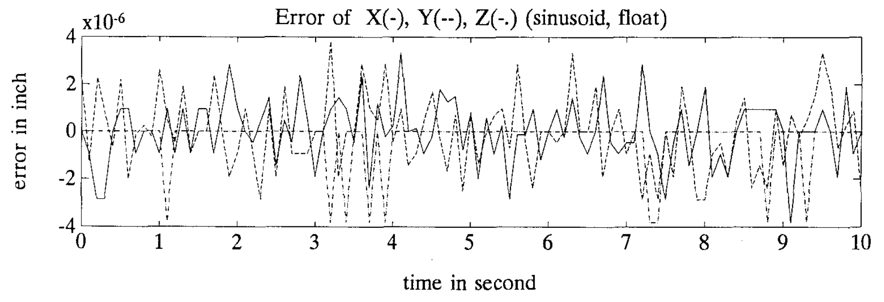

Figure 8: Sinusoidal motion, x, y, z, errors
x (solid), y (dashed), z (dashed-dotted)

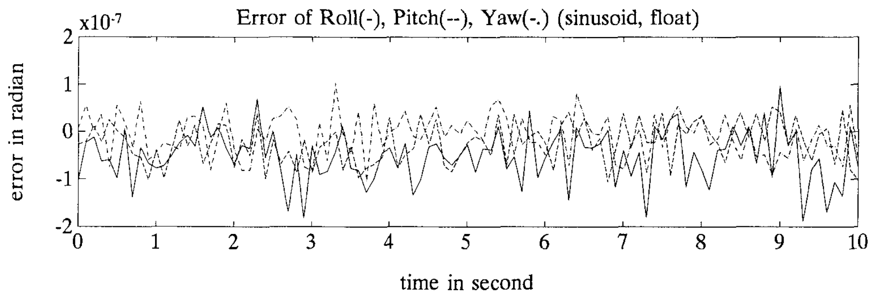

Figure 9: Sinusoidal motion, RPY angle errors
α (solid), β (dashed), γ (dashed-dotted)

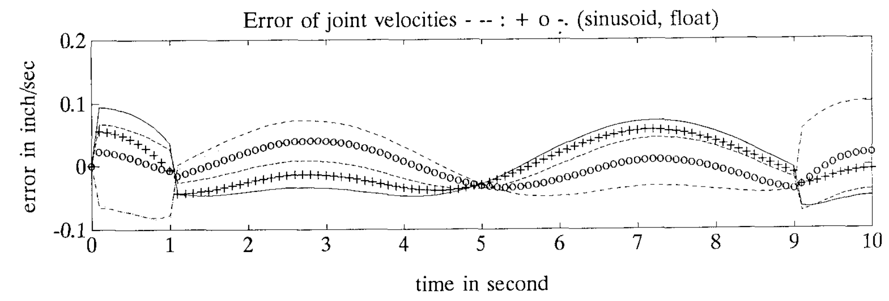

Figure 10: Sinusoidal motion, length velocity errors
$\dot{l}_1$ (solid), $\dot{l}_2$ (- -), $\dot{l}_3$ (..), $\dot{l}_4$ (++), $\dot{l}_5$ (oo), $\dot{l}_6$ (-.)

$$x(t) = x_1 + at_1(t - t_1) \text{ for } t_1 \le t \le t_2 \tag{38}$$

$$x(t) = x_f - \frac{1}{2}\, a(t_f - t)^2 \text{ for } t_2 \le t \le t_f \tag{39}$$

$$y(t) = y_0 + A\sin[\omega(x - x_0)] \text{ for all } t_0 \le t \le t_f \tag{40}$$

where t_0 = 0 sec, t_1 = 1sec, t_f = 10 sec, x_0 = -12.5 inches, x_1 = -11.11 inches, x_f = 12.5 inches, y_0 = inches, a = $(x_f\text{-}x_0)/(t_1 t_2)$ = 2.778 inches/sec^2, A = 10 inches, the angular velocity $\omega = 2\pi/(x_f\text{-}x_0)$ = 0.2513 radian/sec. The computer simulation was conducted on a SUN workstation with a sampling time of 0.05 second for 10 seconds. The errors in Cartesian positions x, y, z are showed in Figure 8 where there exist a maximum error of 4.053 microinch and a maximum RMS error of 1.58 microinch both in y-position. According to Figure 9 which presents the errors in RPY angles, a maximum error of -0.189 microradian and a maximum RMS error of 0.072 microradian occur in Roll angle. The errors in length velocities are reported in Figure 10 where a maximum error of 0.1018 inch/sec and a maximum RMS error of 0.0534 inch/sec occur both in the sixth actuator length. The complete simulation results are tabulated in Table 1.

	Straight Line Motion		Sinusoidal Motion	
	Max Error	RMS Error	Max Error	RMS Error
x [μin]	4.292	1.500	3.815	1.455
y [μin]	-3.815	1.393	4.053	1.580
z [μin]	-3.815	0.932	-3.815	1.076
α [μrad]	-0.177	0.073	-0.189	0.072
β [μrad]	-0.165	0.046	-0.138	0.046
γ [μrad]	-0.115	0.042	-0.104	0.043
$\dot{l}_1$ [$\frac{in}{sec}$]	0.1690	0.0321	0.0944	0.0499
$\dot{l}_2$ [$\frac{in}{sec}$]	0.1913	0.0365	-0.0744	0.0326
$\dot{l}_3$ [$\frac{in}{sec}$]	-0.0449	0.0183	0.0882	0.0448
$\dot{l}_4$ [$\frac{in}{sec}$]	0.1019	0.0199	0.0560	0.0327
$\dot{l}_5$ [$\frac{in}{sec}$]	0.1396	0.0265	0.0399	0.0220
$\dot{l}_6$ [$\frac{in}{sec}$]	-0.0376	0.0159	0.1018	0.0534

Table 1: Computer simulation results

Concluding Remarks

This paper presented a computationally efficient solution for the forward kinematics of 6 DOF robot manipulator built at Goddard Space Flight Center (NASA) to investigate the feasibility of autonomous robotic assembly in space. Designed based on the mechanism of the Stewart Platform, the manipulator mainly consists of two platforms and six linear actuators driven by stepper motors and ballscrews. A closed-form solution was obtained for the inverse kinematic transformation to convert Cartesian variables into required actuator lengths. The inverse kinematic equations were then extensively simplified and then applied to develop an iterative solution for the forward kinematic transformation which converts actuator lengths to Cartesian variables using the Newton-Raphson method. It was proved that a Modified Jacobian Matrix relating length velocities to translational velocities and velocities of RPY angles can be obtained as part of the forward kinematic transformation. Results of computer simulation conducted to evaluate the developed transformations and Modified Jacobian matrix showed that the conversion accuracies were excellent with very negligible errors. Current research activities focus on solving the manipulator forward kinematics using the Powell's direction set numerical method [11] and implementing it on a SGI Personal Iris for real-time graphic interaction between users and manipulator.

Acknowledgement

The research presented in this paper was conducted at the Catholic University of America under a research grant, Grant Number NAG 5-780, funded by Goddard Space Flight Center (NASA).

References

[1] Stewart, D., "A Platform with Six Degrees of Freedom," *Proc. Institute of Mechanical Engineering*, Vol. 180, part 1, No. 5, pp. 371-386, 1965-1966.

[2] Dieudonne, J.E. et al, "An Actuator Extension Transformation for a Motion Simulator and an Inverse Transformation Applying Newton-Raphson's Method," *NASA Technical Report D-7067*, 1972.

[3] Sugimoto, K. and Duffy, J., "Application of Linear Algebra to Screw Systems," *Mech. Mach. Theory*, Vol. 17, No. 1, pp. 73-83, 1982.

[4] Hunt, K. H., "Structural Kinematics of in-parallel-actuated Robot Arms," *Trans. ASME, J. Mech., Transmis., Automa. in Des.*, Vol. 105, pp. 705-712, 1983.

[5] Premack, Timothy et al, "Design and Implementation of a Compliant Robot with Force Feedback and Strategy Planning Software," *NASA Technical Memorandum 86111*, 1984.

[6] Yang, D. C. and Lee, T. W., "Feasibility Study of a Platform Type of Robotic Manipulators from a Kinematic Viewpoint," *Trans. ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 106. pp. 191-198, June 1984.

[7] Fichter, E.F., "A Stewart Platform-Based Manipulator: General Theory and Practical Construction," *Int. Journal of Robotics Research*, pp. 157-182, Summer 1986

[8] Nguyen, C.C., and Pooran, F.J., "Kinematic Analysis and Workspace Determination of A 6 DOF CKCM Robot End-Effector," *Journal of Mechanical Working Technology*, Vol. 20, pp. 283-294, 1989.

[9] Nguyen, C.C., and Pooran, F.J., "Dynamical Analysis of 6 DOF CKCM Robot End-Effector for Dual-Arm Telerobot Systems," *Journal of Robotics and Autonomous Systems*, Vol. 5, pp. 377-394, 1989.

[10] Sharon, A., Hogan, N., and Hardt, D.E., "High Bandwidth Force Regulation and Inertia Reduction Using a Macro/Micro Manipulator System," *Proc. IEEE International Conf. on Robotics and Automation*, Philadelphia, pp. 126-132, 1988.

[11] Press, W.H., et al, "Numerical Recipes in C: The Art of Scientific Computing," *Cambridge University Press*, 1988.