AN OPTIMIZATION APPROACH TO SOLVE THE INVERSE KINEMATICS OF REDUNDANT MANIPULATOR

S. KUMAR, N. SUKAVANAM, AND R. BALASUBRAMANIAN

Abstract. In this paper, we propose an optimization based approach to solve inverse kinematics problem by converting it into a nonlinear optimization problem. An improved energy function is defined to solve the optimization problem even in the case when the matrix associated with objective function is not positive definite. The stability analysis has been done for the proposed algorithm using Lyapunov method. The result has been illustrated through simulation of inverse kinematics solution for a seven arm redundant manipulator.

Key Words. Gradient Descent; Dynamic Control; Inverse Kinematic; Jacobian; Optimization; Redundant Manipulator.

1. Introduction

A robot arm is the combination of links and joints in the form of a chain with one end is fixed while the other end is free. The joints are either prismatic or revolute, driven by actuators. In order to move the free end, also called the end effector, along a certain path, most, if not all, of the joints are to be moved. In doing so it is necessary to solve the inverse kinematics equation. In the case of redundant manipulator, inverse kinematics is much more difficult when compared to a non redundant manipulator whose kinematics is not so complicated.

Let \( r(t) = f(q(t)) \) be the direct kinematics equation of a given manipulator where \( q(t) \) is \( n \times 1 \) vector containing \( n \)-joint variables and \( r(t) \) is \( m \times 1 \) vector in \( \mathbb{R}^m \) representing the position and orientation of the end-effector and \( f \) is a continuous nonlinear function whose structure and parameters are known for a given manipulator. The inverse kinematics problem [3] is to find the joint variables given the desired positions and orientations of the end-effector through the inverse mapping

\[
q(t) = f^{-1}(r(t))
\]

The above equation may have infinitely many solutions \( q(t) \) for a given \( r(t) \). Usually, there are two classes of solution methods to solve the inverse kinematic problem: closed form solution and numerical solution. Closed form solution can be obtained by the spatial geometry of the manipulator, or by solving the matrix algebraic equation (1). Because of the complexity of equation (1) there are cases where a closed form solution does not exist. For non-redundant manipulators \( (m \geq n) \) which do not have a closed form solution, or for those manipulators which have redundant degrees of freedom \( (m < n) \), numerical methods are commonly used to derive the desired joint displacements [6]. Numerical methods are iterative in nature such as Newton-Raphson method. Because of their iterative nature, the numerical solutions are generally much slower than the corresponding closed form solution. In
order to overcome the drawbacks encountered in solving equation (1) an alternative technique based on the differential motion relationship between the joint displacements and the end-effector location is used. In this paper, we have used the relation between joint velocity $\dot{q}(t)$ and cartesian velocity $\dot{r}(t)$ which is a common indirect approach to solve the inverse kinematics problem. The two velocity vectors have the following relation:

$$\dot{r}(t) = J(q(t))\dot{q}(t)$$

where $J(q(t))$ is the $n \times m$ jacobian matrix and can be rank deficient some times. We have denoted the desired velocity by $\dot{r}_d(t)$ and final time by $T$. The corresponding joint position vector $q(t)$ is obtained by integration of $\dot{q}(t)$ for a given $q(0)$. The resulting $q(t)$ then is used for path planning.

To find the solutions of inverse kinematics of a redundant manipulator, many researchers have given various approaches. Some of them have used the Jacobian matrix for finding the solution of inverse kinematics. The inverse of the Jacobian matrix is broken down into manageable submatrices [11]. In [9], a method is proposed with a high speed computation process for the inverse Jacobian matrix. In [4], the pseudo inverse of the Jacobian is calculated. Other numerical techniques such as least square or Newton-Raphson method based approaches are given in [5, 7, 8] for solving the inverse kinematics problems.

In recent years, neural network becomes a very popular tool for solving the inverse kinematics problem and other problems related to robotics such as control. Many of the neural networks for robot kinematic control are feedforward network such as multilayer perceptron training via supervised learning using the backpropagation algorithm [15, 13]. Apart from feed forward neural network with classical backpropagation learning, several evolutionary algorithms based neural network learning algorithm have been used to control the robot inverse kinematics. The Bees algorithm was used to train multi-layer perceptron neural networks to model the inverse kinematics of an articulated robot manipulator arm [10]. In [14], a similar approach is used the only change was the use of particle swarm optimization in neural network training instead of Bee’s algorithm. Recently, some efforts have been made to speed-up the inverse kinematic control procedure. A technique to speedup the learning of the inverse kinematics of a robot manipulator by decomposing it into two or more virtual robot arms have been proposed in [2]. In [1], a practical trick has been proposed by decomposing the learning of the inverse kinematics into several independent and much simpler learning tasks. This was done at the expense of sacrificing generality. However, this trick works only for some robot models subject to certain types of deformations.

In this paper, the inverse kinematics problem is formulated as a nonlinear optimization problem. Instead of assuming that the matrix associated with quadratic objective function of nonlinear optimization problem is positive definite [13], the proposed algorithm is able to solve inverse kinematics problem when associated matrix is not positive definite. The obtained nonlinear optimization problem is solved by using gradient descent method. Applying the gradient method, we form the update equations. Using the proposed algorithm, simulation is carried out for kinematic control of PA-10 manipulator during the tracking of a given trajectory.

The rest of this paper is organized as follows. In section 2, the architecture of PA-10 manipulator, the proposed algorithm to solve the inverse kinematics, its convergence and stability analysis are given. In section 3, the simulation results are presented together with error computation. Finally, section 4 concludes the paper.
2. Proposed Algorithm to Solve Inverse Kinematics

Due to infinite number of solutions of inverse kinematics problem, a kinematic redundant manipulator has many useful properties such as planning optimized path, avoiding singularities and obstacles. But unfortunately it is very difficult to get the solution of inverse kinematic for a redundant manipulator. To achieve a unique solution in case of redundant manipulators, the inverse kinematics problem is formulated as an energy minimization problem. The update equations for joint velocities are obtained by partially differentiating the energy function with respect to time. In this section a detailed formulation is presented for proposed approach. The convergence and stability analysis for proposed algorithm is explained by using Lyapunov function approach.

2.1. Arm Matrix of Redundant Manipulator. The redundant manipulator used in proposed study (PA-10 Manipulator: Portable general purpose Intelligent Arm) made by Mitsubishi, has seven degrees of freedom as shown in Figure 1. The key specifications of the PA-10 manipulator are as follows: the weight of the manipulator is 35 Kg, the maximum combined speed with all axis is 155 m/sec, the payload weight is 10 Kg, and the output torque is 9.8 nm.

The homogeneous transformation matrix (arm matrix) $T_{07}$ for employed seven arms manipulator, which represents the final position and orientation of end effector with respect to the base coordinate system, can be obtained by chain product of successive coordinate transformation matrices. Let $T_{i-1,i}$ for $i = 1, 2, ..., 7$, be the transformation matrices between successive arms, the final arm matrix can be
expressed as
\[
T_{07} = \prod_{i=1}^{7} T_{i-1,i} = \begin{pmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{pmatrix}
\]

where \( n \in \mathbb{R}^3 \) is the normal vector of the end-effector in the base coordinate system, \( s \in \mathbb{R}^3 \) is the sliding vector of the end-effector, \( a \in \mathbb{R}^3 \) is the approach vector of the end-effector, \( p = [x, y, z]^T \) is the position vector of the end-effector and \( T_{i-1,i} \) for \( i = 1, 2, ..., 7 \) are given below:

\[
T_{01} = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
T_{12} = \begin{pmatrix} c_2 & -s_2 & 0 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
T_{23} = \begin{pmatrix} c_3 & -s_3 & 0 & 0 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
T_{34} = \begin{pmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
T_{45} = \begin{pmatrix} c_5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
T_{56} = \begin{pmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
T_{67} = \begin{pmatrix} c_7 & -s_7 & 0 & 0 \\ s_7 & c_7 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\]

where \( c_i = \cos q_i \), and \( s_i = \sin q_i \). The arm matrix is obtained by multiplying the above transformation matrices successively.

### 2.2. Proposed Optimization Algorithm

In order to find out \( q(t) \) for given \( r_d(t) \), we have to solve the following time varying nonlinear optimization programming problem with equality constraints:

\[
\text{minimize} \quad \frac{1}{2} \dot{q}(t)^T W \dot{q}(t)
\]

\[
\text{subject to} \quad J(q(t))\dot{q}(t) = \dot{r}_d(t)
\]

where \( \dot{q}(t)^T \) denotes the transpose of \( \dot{q}(t) \) and \( W \) is an \( m \times m \) symmetric weighting matrix may or may not be positive definite. In particular, if \( W \) is an identity matrix, then the objective function to be minimized is equivalent to the 2-norm of joint velocity \( \| \dot{q}(t) \|_2^2 \). If \( W \) is an inertia matrix, then the objective function to be minimized is equivalent to the kinetic energy.

The energy function of the time-varying nonlinear optimization programming problem subject to equality constraints described in equations (4) and (5) is defined as follows:

\[
E(\dot{q}(t), \lambda(t)) = \frac{1}{2} \dot{q}(t)^T W \dot{q}(t) + \lambda(t)^T [J(q(t))\dot{q}(t) - \dot{r}_d(t)] + \frac{K}{2} |J(q(t))\dot{q}(t) - \dot{r}_d(t)|^T [J(q(t))\dot{q}(t) - \dot{r}_d(t)]
\]
where $\lambda(t)$ is an $n$-dimensional column vector belongs to $R^{n \times 1}$ and $K$ is the penalty rate parameter which is always greater than or equals to zero. Applying a gradient method, we can perform the network update equation as

$$ (7) \quad \dot{q}(k+1) = \dot{q}(k) - \mu \nabla_{\dot{q}} E(\dot{q}, \lambda) $$

$$ (8) \quad \lambda(k+1) = \lambda(k) + \eta \nabla_{\lambda} E(\dot{q}, \lambda) $$

where $\mu$ and $\eta$ are the learning rate parameter. After determining the gradient in equations (7) and (8), the resulting learning rules are

$$ (9) \quad \dot{q}(k+1) = \dot{q}(k) - \mu [W \dot{q}(k) + J(q(t))^T \lambda(k) + KJ(q(t))^T (J(q(t))) \dot{q}(t) - \dot{r}_d(t)] $$

and

$$ (10) \quad \lambda(k+1) = \lambda(k) + \eta [J(q(t)) \dot{q}(t) - \dot{r}_d(t)] $$

The optimal solution for joint velocities $\dot{q}$ as well as the parameters $\lambda$ are obtained by above defined updating equations. A suitable threshold can be fixed to decide the final solution for the joint velocities. Finally, the joint positions/variables $q(t)$ are obtained by integrating the joint velocities $\dot{q}(t)$ using a given initial joint position vector.

2.3. Convergence and Stability Analysis. Consider the Hessian matrix associated with the energy function defined in (6)

$$ (11) \quad H = \frac{\partial^2 E(\dot{q}, \lambda)}{\partial \dot{q}^2} = W + KJ^T J $$

It can be seen from equation (11), that the Hessian matrix of the energy function is positive semidefinite and symmetric, for $K = 0$ if $W$ is positive semidefinite. This is the case in many existing methods like Cichicki (1993), Wang (1999). In our algorithm, it is assumed that $W$ is only symmetric. Then the Hessian matrix can be forced into positive semidefinite by choose a sufficiently large positive value for the parameter $K$. $ \frac{\partial^2}{\partial \dot{q}^2} [J(q(t)) \dot{q}(t) - \dot{r}_d(t)]^T [J(q(t)) \dot{q}(t) - \dot{r}_d(t)]$ is the penalty term in objective function of quadratic programming problem, to improve the convergence properties of the network in cases when some of the eigenvalues are relatively small positive numbers or even in case, when $W$ is not positive semidefinite.

Let us assume that the vectors $v(t)$ and $u(t)$ represent the network estimated values of $\dot{q}$ and $\lambda(t)$ respectively, then the dynamics equations of the presented system represent as follows:

$$ (12) \quad \dot{v}(t) = -(W + KJ(q(t))^T J(q(t))) v(t) $$

$$ -J(q(t))^T u(t) + KJ(q(t))^T \dot{r}_d(t) $$

$$ (13) \quad \dot{u}(t) = J(q(t)) v(t) - \dot{r}_d(t) $$

Written in combined format, the equations (12) and (13) are given by the following time-varying linear system

$$ (14) \quad \dot{\psi}(t) = A(t) \psi(t) + B(t) $$

where $[\psi(t)]^T = [v(t)^T, u(t)^T]$, $B(t)^T = [KJ(q(t))^T \dot{r}_d(t), -\dot{r}_d(t)]$, and

$$ (15) \quad A(t) = \begin{pmatrix} -(W + KJ(q(t))^T J(q(t))) & -J(q(t))^T \\ KJ(q(t)) & 0 \end{pmatrix} $$

The solution $\psi(t)$ of the system starting from $\psi_0$ is said to be stable if for given positive real number $\epsilon > 0$, there exists a positive real number $\delta > 0$, such that
for any initial point $\psi(0)$ in the $\epsilon$-nhbd of $\psi_0$, the corresponding solution of (14) remains in the $\delta$-nhbd of $\psi(t)$ for $t \in [0, \infty)$. Since the system defined in (14) is linear, we will analyze the stability without consideration of $B(t)$. The following theorems guarantees the stability of the solution of (14).

**Theorem 1:** The energy function defined in (14) is globally stable and $v$ is globally asymptotically stable.

**Proof:** Let us consider the homogeneous system

$$\dot{\psi}(t) = A\psi$$

and define a Lyapunov function

$$L(\psi) = \psi(t)^T C \psi$$

where $C = \text{diag}[1, 1]$. The time derivative of $L(\psi)$ is given by

$$\frac{dL}{dt} = 2\psi(t)^T A(t) \psi(t) = -v(t)^T H v(t)$$

where $H = W + K J^T J$ is the Hessian matrix. Since $W$ is symmetric and $K$ is a positive scalar, $H$ is also a symmetric matrix as $J^T J$ is symmetric. Since $J^T J$ is diagonally dominant and positive definite, so $H$ is also positive definite for large positive value of $K$. So from (18)

$$L(\psi(t)) \leq L(\psi(0))$$

$$\|v(t)\|^2 \leq \|v(0)\|^2 e^{-\sigma t} \to 0,$$ as $t \to +\infty$

where $\sigma$ is the minimal eigenvalue of $H$. Hence $v(t)$ is asymptotically stable for all values of $t$.

In this way, the proposed algorithm is guaranteed to converge with asymptotically stability without the condition that the matrix $W$ associated with the energy function should be positive definite matrix.

3. Results and Discussions

The simulation studies have been performed on PA-10 manipulator and the simulation results are presented to demonstrate the performance of the proposed method. For the simulation purpose, the matrix $W = \text{diag}(1, -1, -1, 1, 1, 1)$. The objective is to simulate the redundant manipulator in the following closed and curvilinear trajectory in 3-D workspace. Figure 3(a) represents the simulated trajectory in the 3-D space while figure 2(b), (c) and (d) show 2-D orthographic projections of 2(a) in $xy$, $yz$ and $xz$ planes respectively. These projections are virtually identical to the desired path. The values of various parameters of manipulator, i.e., $d_3$, $d_5$ and $d_7$ are 1.8, 3.6 and 0.6 respectively. The values of the parameters $\mu$, $\eta$ and $K$ used in gradient descent method are 0.02, 0.02 and 100.0 respectively.

Figure 3 represents the transient behaviors of the PA-10 manipulator in the proposed method. Fig. 3(a) and (b) show the desire coordinates $r_d(t) = (x, y, z, x_\omega, y_\omega, z_\omega)$ and the desired velocity $\dot{r}_d(t) = (\dot{x}, \dot{y}, \dot{z}, \dot{x}_\omega, \dot{y}_\omega, \dot{z}_\omega)$.

Here the cartesian coordinates of the end-effector with respect to the world coordinate frame is denoted by $(x, y, z)$ in meters and the orientation coordinates is denoted by $(x_\omega, y_\omega, z_\omega)$ in radians. $(\dot{x}, \dot{y}, \dot{z})$ denotes the translational velocity variables in m/s, and $(\dot{x}_\omega, \dot{y}_\omega, \dot{z}_\omega)$ denotes the angular velocity variables in rad/s. The initial angular vector is given by $q(0) = [0.20, 0.0, 0.0, -0.28, 0.0, -0.10, -0.03]$. 
Figure 2. Simulated trajectory of the PA-10 redundant manipulator.

Figure 3. Behaviors of the proposed method based on simulation results.
Fig. 3(c) shows the simulated velocity variables $\dot{q}(t)$ corresponding to time obtained using the proposed method. Fig. 3(d) represents the simulated joint variables $q(t)$ which we find out after integrating the joint velocity vector from initial to final time. It can be seen that $\dot{q}(0) = \dot{q}(T) = 0$, which is required for the desired trajectory.

The difference between the actual translation position of the end-effector and simulated one over the whole simulation time is illustrated in Fig. 4(a). It can be seen that this difference is very small. Similar results are found in the case of orientation parameters as shown in Fig 4(b). Precisely, $\max \| r(t) - r_d(t) \|_2 = 3.6 \times 10^{-4}$ and $\max \sqrt{ (x_{err})^2 + (y_{err})^2 + (z_{err})^2 } = 2.16 \times 10^{-5}$ rad., which are negligible, and this suggests that our methodology is efficient.

4. Conclusions

An algorithm based on nonlinear optimization for solving inverse kinematics of redundant manipulator is presented, which is inspired by the results given by [13] in which the matrix $W$ was assumed to be symmetric and positive definite. Even in case when $W$ is positive definite, the proposed algorithm converges faster due to the penalty term. In the proposed method, the matrix $W$ need not be positive semi-definite. The problem of tracking an object moving along a closed curve which has
a parametric representation in time is considered. The problem is highly nonlinear and has more complexity due to seven unknown joint variables. The results show the smooth tracking of the desired path.

Acknowledgements

One of the authors N. Sukavanam acknowledges the support of DST, Government of India through its grant No. DST-347-MTD.

Appendix A. Jacobian for PA-10 Manipulator

The Jacobian of PA-10 manipulator is given by

\[
J = \begin{pmatrix}
J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & J_{16} & J_{17} \\
J_{21} & J_{22} & J_{23} & J_{24} & J_{25} & J_{26} & J_{27} \\
J_{31} & J_{32} & J_{33} & J_{34} & J_{35} & J_{36} & J_{37} \\
J_{41} & J_{42} & J_{43} & J_{44} & J_{45} & J_{46} & J_{47} \\
J_{51} & J_{52} & J_{53} & J_{54} & J_{55} & J_{56} & J_{57} \\
J_{61} & J_{62} & J_{63} & J_{64} & J_{65} & J_{66} & J_{67}
\end{pmatrix}
\]

and \(J^*_j\) are derived as follow

\[
J_{11} = -s_1v_{114} - c_1v_{234}, \quad J_{12} = c_1u_{23}, \quad J_{13} = s_1s_2u_{33} - c_2u_{32}, \\
J_{14} = v_{423}u_{43} - v_{433}u_{42}, \quad J_{15} = -v_{422}u_{53} + v_{423}u_{52}, \quad J_{16} = v_{522} \\
J_{17} = 0, \quad J_{21} = c_1v_{114} - s_1v_{234}, \quad J_{22} = s_1u_{23}, \quad J_{23} = c_2u_{32} - c_1s_2u_{33}, \\
J_{24} = v_{433}u_{41} - v_{413}u_{43}, \quad J_{25} = -v_{412}u_{51}, \\
J_{26} = v_{522}u_{61} - v_{512}u_{63}, \quad J_{27} = 0, \quad J_{31} = 0, \quad J_{32} = -s_1 \\
J_{33} = c_1s_2u_{32} - s_1s_2u_{31}, \quad J_{34} = v_{413}u_{42} - v_{423}u_{41}, \quad J_{35} = -v_{412}u_{52} + v_{422}u_{51}, \\
J_{36} = v_{522}u_{62} - v_{512}u_{61}, \quad J_{37} = 0, \quad J_{41} = 0, \\
J_{42} = -s_1, \quad J_{43} = c_1u_{23}, \quad J_{44} = v_{413}, \quad J_{45} = -v_{412}, \quad J_{46} = v_{512}, \quad J_{47} = -v_{612}. \\
J_{51} = 0, \quad J_{52} = c_1, \quad J_{53} = s_1s_2, \quad J_{54} = v_{433}, \quad J_{55} = -v_{422}, \\
J_{56} = v_{522}, \quad J_{57} = -v_{622}, \quad J_{61} = 1, \quad J_{62} = 0, \quad J_{63} = c_2, \quad J_{64} = v_{433}, \\
J_{65} = -v_{432}, \quad J_{66} = v_{532} \text{ and } J_{67} = -v_{632}
\]

The expressions in \(u\) and \(v\) are as follows

\[
v_{414} = c_3s_6d_7, \quad v_{424} = -c_3d_7 - d_5, \quad v_{434} = s_5s_6d_7, \quad v_{434} = c_4v_{414} - s_4v_{424}, \quad v_{524} = v_{434}, \quad v_{434} = -s_4v_{414} - c_4v_{424}, \\
v_{214} = c_3v_{414} - s_3v_{424}, \quad v_{224} = -v_{334} - d_3, \quad v_{234} = s_3v_{314} + c_3v_{344}, \\
v_{114} = c_2v_{214} - s_2v_{224}, \quad v_{134} = -s_2v_{214} - c_2v_{224}, \quad v_{141} = c_1c_2c_3 \\
- s_1s_3c_4 - c_1s_2s_4, \quad v_{412} = -(c_1c_2c_3 - s_1s_3)s_4 - c_1s_2c_4, \quad v_{413} = -c_1c_2s_3 - s_1c_3, \\
v_{421} = (c_1c_2c_3 + c_1s_3)c_4 - s_1s_2s_4, \quad v_{422} = -(c_1c_2c_3 + c_1s_3)s_4 - s_1s_2c_4, \quad v_{423} = -s_1c_2s_3 + c_1c_3, \\
v_{431} = -s_2c_3c_4 - c_2s_4, \quad v_{432} = s_2c_3s_4 - c_2c_4, \quad v_{433} = s_2c_3, \\
v_{511} = v_{411}c_5 + v_{413}s_5, \quad v_{512} = -v_{411}s_5 + v_{413}c_5, \quad v_{513} = -v_{412}, \quad v_{521} = c_2c_5 \\
+ v_{423}s_5, \quad v_{522} = -v_{421}s_5 + v_{423}c_5, \quad v_{523} = -v_{422}, \quad v_{531} = v_{431}c_5 + v_{433}s_5, \quad v_{532} = -v_{431}s_5 + v_{433}c_5, \\
v_{533} = -v_{432}, \quad v_{612} = -v_{511}s_6 + v_{513}c_6, \quad v_{622} = -v_{521}s_6, \\
v_{422}c_6, \quad v_{632} = -v_{531}s_6 + v_{533}c_6, \quad u_{21} = c_1c_2v_{214} - c_1s_2v_{224} - s_1v_{234}, \quad u_{22} = c_1c_2v_{214} - s_1s_2c_4 + c_1s_2s_4, \\
u_{23} = -v_{214} - c_2v_{224}, \quad u_{31} = c_1c_2c_3 - s_1s_3, \quad u_{34} = (c_1c_2s_3 - s_1c_3)u_{34} + s_1s_2u_{33}, \\
u_{35} = (s_1c_2c_3 + c_1s_3)u_{34} - (s_1c_2s_3 + c_1c_3)d_7, \quad u_{55} = v_{521}s_6d_7 + v_{523}c_6d_7, \\
u_{53} = v_{531}s_6d_7 + v_{533}c_6d_7, \quad u_{61} = -v_{612}d_7, \quad u_{62} = -v_{622}d_7 \text{ and } u_{63} = -v_{632}d_7.
\]
References


Sanjeev Kumar, Department of Mathematics, Indian Institute of Technology Roorkee, Roorkee-247667, India, E-mail: malikfma@iitr.ernet.in, Has completed his Ph.D. from IIT Roorkee. Currently, he is working as an assistant professor at department of mathematics, IIT Roorkee from November 2010. Earlier, he worked as a post doctoral fellow in AVIRES lab, University of Udine, Italy from March 2008 to October 2010. His current research interests include Mathematical Optimization, Computer Vision and Robotic Control.

Nagarajan Sukavanam, Department of Mathematics, Indian Institute of Technology Roorkee, Roorkee-247667, India, E-mail: nsukvfma@iitr.ernet.in, is an Associate Professor in the Department of Mathematics at Indian Institute of Technology Roorkee. He has received his M.Sc. degree in Mathematics from the University of Madras, India, in 1979 and Ph.D. degree in Mathematics from the Indian Institute of Science Bangalore, India, in 1985. So far he has published more than fifty research papers in various International Journals and Conferences. His Areas of research are Robotics, Nonlinear Analysis, Control Theory and Computer Vision.

Balasubramanian Raman, Department of Mathematics, Indian Institute of Technology Roorkee, Roorkee-247667, India, E-mail: balarfma@iitr.ernet.in, is an Assistant Professor in the Department of Mathematics at Indian Institute of Technology Roorkee. He received his Ph.D. in Mathematics (2001) from Indian Institute of Technology, Madras, India. He has more than seventy research papers in various reputed international journals and conferences. His areas of research include Computer Vision, Graphics, Satellite Image Analysis, Scientific Visualization, Imaging Geometry and Reconstruction problems.