https://doi.org/10.21289/KSIC.2020.23.4.549

A Study on Track Record and Trajectory Control of Robot Manipulator with Eight Joints Based on Monitoring Simulator for Smart Factory

Hee-jin Kim[1], Gi-won Jang[1], Dong-ho Kim[1], Sung-hyun Han[2]

〈Abstract〉

We describe a new approach to real-time implementation of track record and trajectory control of robotic manipulator with eight joints based on monitoring simulator. Trajectory generator uses the kinematic equations of the arm to convert the task description into a series of set points for each of the joint control loops, while the joint controllers, with simple algorithms for just one joint can move at a fast sampling rate, guaranteeing a smooth motion. The proposed control scheme is robust, fast in computation, and suitable for real-time control. Moreover, this scheme does not require any accurate parameter information, nor values of manipulator parameters and payload. Reliability of the proposed technology is veriefied by monitoring simulation and experimental of robot manipulator for the smart factory with eight degrees of freedom.

Keywords : Track Record, Trajectory control, Robot manipulator, Monitoring simulator, Smart factory, Real Time Implementation

1 Dept.of Mechanical Engineering., Graduate School, Kyungnam University, Korea
2 Dept. of Mechanical Engineering., Kyungnam Univ. Korea

1. Introduction

Current industrial approaches to the design of robot arm control systems treat each joint of the robot arm as a simple servomechanism. This approach models the time varying dynamics of a manipulator inadequately because it neglects the motion and configuration of the whole arm mechanism. The changes in the parameters of the controlled system are significant enough to render conventional feedback control strategies ineffective. This basic control system enables a manipulator to perform simple positioning tasks such as in the pick-and-place operation. However, joint control- lers are severely limited in precise tracking of fast trajectories and sustaining desirable dynamic performance for variations of payload and parameter uncertainties (R. Ortega et al., 1989; P. Tomei, 1991). In many servo control applications the linear control scheme proves unsatisfactory, therefore, a need for nonlinear techniques is increasing.

Once the arm type is given, the designer of the control system faces several problems: First, there are the limited capabilities of the arm, such as a maximum speed for each joint and a limited force or torque output of each actuator. Secondly, there are constraints imposed by the user, such as maximum allowed deviations from the specified path at the tool tip. And finally, there is the highly nonlinear relationship between the motions of the individual joints and the path in space, further complicated by the fact that the dynamic parameters of the arm change along with its configuration.

This solution is complex, however, and would require enormous computing power to be able to perform the computations, in particular the simulation, in real time. Sampling intervals cannot be lengthened to deal with this, since we are operating at high speed, close to limits, and want a smooth motion of the arm.

These problems can be avoided if we resort to using a trajectory generator combined with closed-loop control of each individual joint, as shown in Fig. 1.

Except for monitoring of position, velocity, and current limits for emergency actions (usually power shut-off), there is no feedback and thus no way to modify the trajectory should the robot fall behind. Therefore, if one wants to run the robot close to its limits, the real problem becomes setting up the trajectory to be sent to the joint controllers.

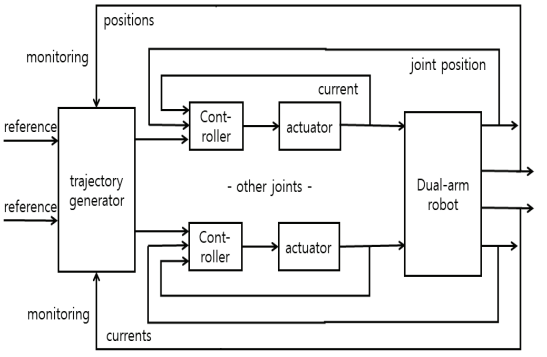


Fig. 1 The control scheme of robot system

Some way had to be found to arrive at an optimal set of transition and segment times that would minimize overall execution time without violation of any velocity or actuator current limit.

All points of the path were assumed to be known before the start of any motion. It was allowed to run the robot several times, monitoring currents and positions, changing the set of transition and segment times iteratively based on the performance in previous runs.

The study was restricted to joint interpolated motion, since it is faster than cartesian motion, and easier to optimize.

2. Robot Dynamics

The dynamic model of a manipulator-plus-payload is derived and the tracking control problem is stated in this section.

Let us consider a nonredundant joint robotic manipulator in which the $n\times1$ generalized joint torque vector $T(t)$ is related to the $n\times1$ generalized joint coordinate vector $q(t)$ by the following nonlinear dynamic equation of motion

$$I(q)\ddot{q}+C(q,\dot{q})+G(q)=T(t) \tag{2.1}$$

where $I(q)$ is the $n\times n$ symmetric positive- definite inertia matrix, $C(q)$ is the $n\times1$ coriolis and centrifugal torque vector, and $G(q)$ is the $n\times1$ gravitational loading vector.

Equation (2.1) describes the manipulator dynamics without any payload. Now, let the $n\times1$ vector X represent the end-effector position and orientation coordinates in a fixed task-related cartesian frame of reference.

Let us now consider payload in the manipulator dynamics. Suppose that the manipulator end- effector is firmly grasping a payload represented by the point mass δP_p. For the payload to move with acceleration $\ddot{X}(t)$ in the gravity field, the end-effector must apply the $n\times1$ force vector $\tau(t)$ given by

$$\tau(t)=\delta P_p[\ddot{X}(t)+g] \tag{2.2}$$

where g is the $n\times1$ gravitational acceleration vector.

The end-effector requires the additional joint torque

$$T_f(t)=J(q)^T\tau(t) \tag{2.3}$$

where superscript τ denotes transposition. Hence, the total joint torque vector can be obtained by combining equations (2.1) and (2.3) as

$$J(q)^T\tau(t)+I(q)\ddot{q}+C(q,\dot{q})+G(q)=T(t) \tag{2.4}$$

Substituting equations (2.2) into equation (2.4) yields

$$\begin{aligned}&\delta P_p J(q)^T[J(q)\ddot{q}+\dot{J}(q,\dot{q})\dot{q}+g]\\&+I(q)\ddot{q}+C(q,\dot{q})+G(q)=T(t)\end{aligned} \tag{2.5}$$

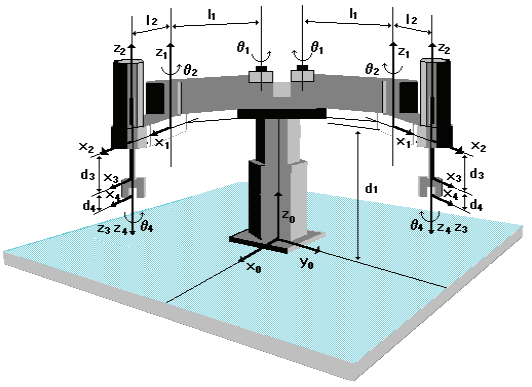


Fig. 2 Link coordinates of robot

Equation (2.5) shows explicitly the effect of payload mass δP_p on the manipulator dynamics. This equation can be written as

$$[I(q)+\delta P_p J(q)^T J(q)]\ddot{q}+[C(q,\dot{q})+\delta P_p J(q)^T \dot{J}(q,\dot{q})\dot{q}]+G(q)+\delta P_p J(q)^T g=T(t) \quad (2.6)$$

where the modified inertia matrix $[I(q)+\delta P_p J(q)^T J(q)]$ is symmetric and positive-definite. Equation (2.6) constitutes a nonlinear mathematical model of the manipulator-plus- payload dynamics.

In modeling the robot, we start with the general form for the dynamics equation of a six-link manipulator with revolute joints that can be derived using Lagrangian mechanics [1,2]:

$$\tau_j=\sum_{k=1}^{8} I_{jk}\ddot{q}_k+D\alpha_j\ddot{q}_j+\sum_{k=1}^{8}\sum_{l=1}^{8} I_{jkl}\dot{q}_k\dot{q}_l+I_j \quad (2.7)$$

where:

τ_j = torque in joint j

I_{jj} = effective inertia, joint j

I_{jk} = coupling inertia, joints j and k

$D\alpha_j$ = actuator inertia in q_i coord.system

I_{jkk} = coefficients for centripetal forces

I_{jkl} = coefficients for coriolis forces

I_j = gravity loading terms

The expressions for the I-coefficients are fairly complex (refer to [1,2]). They depend on the arm configuration, so that the six equations (2.7) are nonlinear and strongly coupled.

Manipulators usually have significant friction terms, so extra terms have to be added on the right hand side of equation (2.7). The most adequate form has to be found experimentally.

As a basis for the design of a simple algorithm, equation (2.5) in its general form is far too complex. For the first attempt, radical simplifications were made as follows:

All coupling inertias were ignored, arguing that the combined magnitude of effective inertias and actuator inertias makes the coupling terms relatively unimportant [1].

Only velocity and load independent friction terms were kept, i.e., friction was modeled as constant · sign($\dot{q}$). Test runs showed that this was feasible.

much during a transition, effective inertias and gravity loadings were assumed to be constants for each transition.

Finally, by substituting from the approximate relationship between output torque τ, and input current D for the DC electric drives

$$\tau = kD \tag{2.8}$$

for some constant k, we get a linear relation-ship between current and acceleration:

$$D_j = \alpha_{ij}\ddot{q}_j + \beta_{ij} \tag{2.9}$$

The constants α_{ij} and β_{ij} are different for each transition and each joint. Basically, represents the sum of effective and actuator inertia and β_{ij} accounts for gravity loading and friction. If a joint changes direction during a transition, there will be two values for β_{ij}, since the friction term changes its sign. What counts, however, is the form of the relationship and that it allows us to estimate the necessary motor currents for a given acceleration without involved computations, once the constants have been found in some way.

In the case of the joint control loop the simplest possible approach was taken: Assume that the controller works perfectly, i.e., desired and true accelerations are identical. The less-than-ideal controller performance in practice later caused some problems (see result section).

The assumed equality of desired and true accelerations makes it possible to combine equations (3.4) and (2.9):

$$D_{ij,\max} = 0.92\alpha_{ij}\left[\frac{\delta q_{i+1}}{\tau_{i+1}} - \frac{\delta q_i}{\tau_i}\right]\frac{1}{T_i} + \beta_{ij} \tag{2.10}$$

where $D_{ij,\max}$ is the peak current observed during a transition. There is one such equation for each transition and each joint in this model.

From the way RCCL trajectories are set up, the maximum velocity in any segment is completely defined by the distance that a joint has to move and the segment time for this motion:

$$S_{ij,\max} = \delta q_{ij}/\tau_i \tag{2.11}$$

Again, the assumption of a perfect controller makes modeling easier, since we can use (2.11) directly to estimate the actual velocities.

The models developed above allow us to replace constraints (2.12a) and (2.12b) with expressions in T_i and τ_i:

$$\tau_i < \tau_{i,\text{limit}} = \max_j\left\{\frac{\delta q_{ij}}{S_{j,\text{limit}}}\right\} \tag{2.12a}$$

$$D_{ij,\max} = 0.92\alpha_{ij}\left[\frac{\delta q_{i+1}}{\tau_{i+1}} - \frac{\delta q_i}{\tau_i}\right]\frac{1}{T_i} + \beta_{ij} < D_{j,\text{limit}} \tag{2.12b}$$

Unfortunately the second constraint is nonlinear, ruling out linear programming, and contains unknown constants. What algorithm could deal with this? We propose the following solution:

Intuitively, it should be clear that a minimum- time path is found if the joints run at maximum velocity wherever possible or else start decelerating as soon as they

finish accelerating (this minimizes segment times), while accelerations are made as high as possible (reducing transition times). That is, we have a minimum-time solution when the arm gets as close to its velocity and current limits as possible. This can be used as a new objective for the algorithm.

As mentioned earlier, an algorithm based on a simpler model may exhibit poor convergence properties. For the first version, however, we did use a very simple model, dropping β_{ij} from (2.12b). Then, using ratios of observed currents and limiting currents, we could do without α_{ij} as well.

A strategy that computes new transition times based on the old segment times and then updates the segment times finally made the algorithm very compact.

One iteration involves the following steps, after peak currents for each transition and peak velocities for each segment have been found:

Estimate new transition times from

$$T_{i,est} = \max_j \left\{ \frac{D_{ij,max}}{D_{j,limit}} \right\} T_{i,old} \qquad (2.13)$$

Do not reduce the transition times quite as far, compute the new ones as

$$T_{i,new} = T_{i,old} + s(T_{i,est} - T_{i,old}) \qquad (2.14)$$

where the *step size s* has a value between 0 and 1.

Reduce the segment times as far as possible

$$\tau_i = T_{i-1,new} + T_{i,new} \qquad (2.15)$$

Where this time is likely to cause violation of the velocity limits, correct it:

$$\tau_{i,new} = \max \{\tau_i, \tau_{i,limit}\} \qquad (2.16)$$

The step size parameter had to be introduced to account for the errors introduced by computing transition times based on previous segment times and by the simplifications in the model. It has a decisive influence on the speed of convergence and the likelihood of overshoots beyond the limits. The experiments showed, however, that good perfor- mance of the algorithm could be achieved with just one value of s, independent of the path to be optimized. In any case, the algorithm stops when it is not possible to improve either overall execution time or position accuracy.

3. Trajectory generation

As is apparent from the introduction, the problem has already been reduced to the question how the best set of transition and segment times can be found. If the robot follows its set points accurately, this is equivalent to minimizing the performance index

$$p_{tot} = T_0 + \sum_{i=1}^{n} t_i + T_n \qquad (3.1)$$

subject to the constraints that no velocity limit and no actuator current limit is violated for any joint and that the times are consistent:

$$S_{ij} < S_{j,\,limit} \tag{3.2a}$$

$$D_{ij} < D_{j,\,limit} \tag{3.2b}$$

$$T_i \geq 0 \tag{3.2c}$$

$$t_i \geq T_{i-1} + T_i \tag{3.2d}$$

S is velocity, T_i is transition time, t_i is segment time.

How can we arrive at a solution? An analytical approach seems hopeless, because of the great number of free parameters. Some kind of iteration is necessary, and one way to do this is to add an optimization program that will provide the trajectory generator with sets of times to evaluate, based on an examination of the performance of the robot in the previous attempt (Fig. 3).

The optimization program has to be provided with the observed currents, velocities and true segment and transition times.

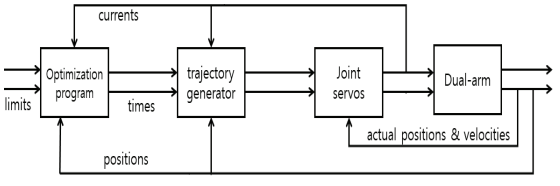


Fig. 3 The scheme of trajectory control

The next step is the selection or design of an algorithm to perturb the segment and transition times towards an optimal or near-optimal solution. Standard search procedures that do not require any knowledge about the system to be optimized are likely to need a large number of iterations, especially if there are many free parameters.

Modeling the trajectory generator requires some rearrangement of trajectory equations in order to find an expression for the peak acceleration during a transition in terms of segment and transition times. For the type of transitions built into RCCL, which are polynomials that make the resulting trajectory continuous up to its second derivative [1], the result is a parabola:

$$\ddot{q} = 0.92\left[\frac{\delta q_i}{\tau_i} - \frac{\delta q_{i+1}}{\tau_{i+1}}\right]\frac{\tau^2 - T_i^2}{T_i^3} \tag{3.3}$$

That is, we have zero acceleration at beginning and end of the transition (at $\pm T$) and a peak acceleration of

$$\ddot{q}_{max} = 0.92\left[\frac{\delta q_i}{\tau_i} - \frac{\delta q_{i+1}}{\tau_{i+1}}\right]\frac{\tau^2 - T_i^2}{T_i^3} \tag{3.4}$$

4. Track Record Simulator

Fig. 4 shows the structure of monitoring simulator. And Fig. 5 shows experimental set-up.

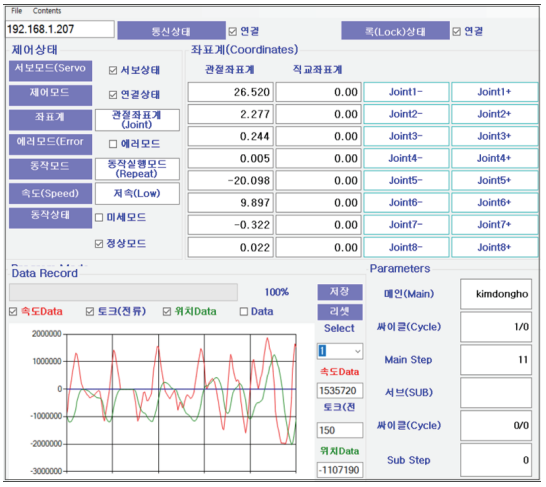


Fig. 4 The structure of monitoring simulator

Fig. 5 Experimental set-up

The performance test of the proposed track record and trajectory control has been performed for the dual-arm robot at the joint space and cartesian space. At the cartesian space, it has been tested for the peg-in-hole tasks, repeating precision tasks, and trajectory tracking for B-shaped reference trajectory. At the joint space, it has been tested for the trajectory tracking of angular position and velocity for a dual-arm robot.

Table 1. Link parameters of robot

Mass of link (kg)		Length of link (kg)		Inertia of link (kg)		Gear ratio of link	
m1	17.007	I1	0.53	I1	0.268	r1	1/110
m2	9.952	I2	0.51	I2	0.124	r2	1/85
m3	4.2	I3	0.36	I3	0.09	r3	1/210
m4	1.8	I4	0.01	I4	0.006	r4	1/80
m5	17.07	I5	0.53	I5	0.268	r5	1/110
m6	9.952	I6	0.51	I6	0.124	r6	1/85
m7	4.2	I7	0.36	I7	0.09	r7	1/210
m8	1.8	I8	0.01	I8	0.006	r8	1/80

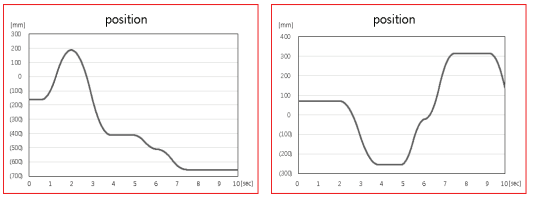
Table 2. Motor parameters of robot

Rotor inertia ($kg \cdot m^2$)		Torque constant (K m/a)		Back emf constant (V s/rad)		Amature-winding resistance (ohms)	
Jm1	7.03×10^{-5}	Ka1	25.94×10^{-2}	Kb1	259.47×10^{-3}	Ra1	2.6
Jm2	3.48×10^{-5}	Ka2	23.12×10^{-2}	Kb2	231.47×10^{-3}	Ra2	6.5
Jm3	2.89×10^{-5}	Ka3	23.12×10^{-2}	Kb3	231.47×10^{-3}	Ra3	11
Jm4	1.37×10^{-5}	Ka4	19.84×10^{-2}	Kb4	198.68×10^{-2}	Ra4	25
Jm5	7.03×10^{-5}	Ka5	25.94×10^{-2}	Kb5	259.47×10^{-3}	Ra5	2.6
Jm6	3.48×10^{-5}	Ka6	23.12×10^{-2}	Kb6	231.47×10^{-3}	Ra6	6.5
Jm7	2.89×10^{-5}	Ka7	23.12×10^{-2}	Kb7	231.47×10^{-3}	Ra7	11
Jm8	1.37×10^{-5}	Ka8	19.84×10^{-2}	Kb8	198.68×10^{-2}	Ra8	25

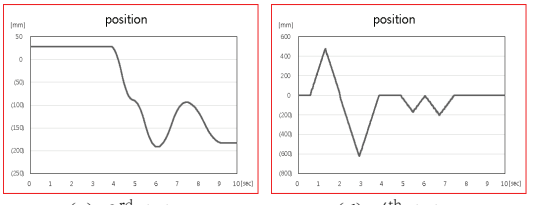
5. Experiment and results

This section represents the simulation results of the position and velocity and torque control of a four-link robotic manipulator by the proposed control algorithm, as shown in Fig. 2, and discusses the advantages of using joint controller based-on motion control of a dual-arm robot. The adaptive scheme developed in this paper will be applied to the control of a dual-arm robot with eighth axes.

The proposed controllers have several advantages over the analog control and the micro-computer based control. This allows instructions and data to be simultaneously fetched for processing. Moreover, most of the instructions, including multiplications, are performed in one instruction cycle. The tremendously increase speed of the controller and reduce computational delay, which allows for faster sampling operation. It is illustrated that can be used for the implementation of complex digital control algorithms, such as our adaptive control for robot systems.

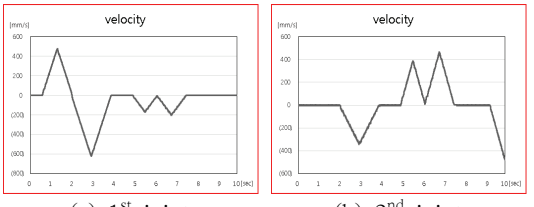


(a) 1st joint (b) 2nd joint

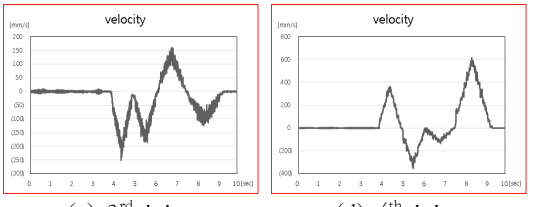


(c) 3rd joint (d) 4th joint

Fig. 7 (a)-(d) The results of position track record for each joint by monitoring system

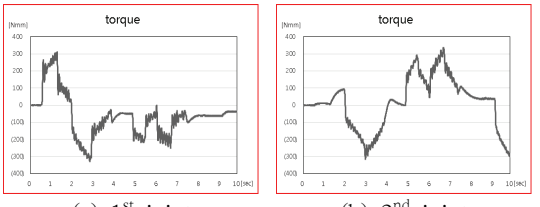


(a) 1st joint (b) 2nd joint

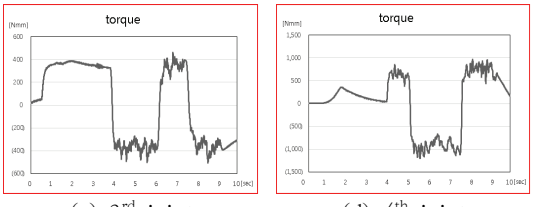


(c) 3rd joint (d) 4th joint

Fig. 6 (a)-(d) The results of velocity track record for each joint by monitoring system



(a) 1st joint (b) 2nd joint



(c) 3rd joint (d) 4th joint

Fig. 8 (a)-(d) The results of torque track record for each joint by monitoring system

6. Conclusion

This study propsed a new technology of track record and trajectory control of robot manipulator with eight joints based on monitoring simulator for smart factory. This monitoring technology can be used for performance evaluation of cooperative robot system. The perfomance is illustrated by simulation and experiment based monitoring simulator for robot manipulator with eight joints.

Acknowledgement

This study was supported by Industrial Technology Innovation Project (Project Number: 20005020).

References

[1] T.C. Hasi, 1986, “Adaptive Control Scheme for Robot Manipulators-A Review.” In Proceeding of the 1987 IEEE Conference on Robotics and Automation, San Fransisco, CA.
[2] S. Nicosia and P. Tomee, 1984, “Model Reference Adaptive Control Algorithm for Imdustrial Robots,” Automatica, Vol. 20, No. 5, pp. 635-644.
[3] A. Koivo and T. H. Guo, 1983, “Adaptive Linear Controller for Robot Manipulators.” IEEE Transactions and Automatic Control, Vol. AC-28, pp. 162-171.
[4] P. Tomei, Aug. 1991, “Adaptive PD Controller for Robot Manipulators,” IEEE Trans. Robotics and Automation, Vol. 7, No. 4.
[5] D. Koditschek, 1983, “Quadratic Lyapunov Functions for Mecanical Systems,” Technical Report No. 8703, Yale University, New Haven, CT.

(Manuscript received June 22, 2020;
revised July 6, 2020; accepted July 17, 2020)