Simultaneous Optimization of Timing and Trajectory in Sequential and Parallel Tasks of Humanoid Robots

*Shuichi SUZUKI, Yuichi TAZAKI and Tatsuya SUZUKI

Abstract—This paper presents a motion planning technique capable of scheduling multiple sub-tasks that can be achieved in different orders and at various timings. A robotic motion is defined as a timed event sequence of multi-body dynamical system. Schedules of the motion planning is determined by robotic mechanism and motion requirements. Motion planning problem including task scheduling is then formulated as a class of constrained optimization problems, which can be solved efficiently by means of an iterative algorithm that exploits the sparsity of the multi-body system. The proposed method is demonstrated in object-catching tasks of a standing humanoid in numerical simulation.

I. INTRODUCTION

Humanoids are required to perform complex tasks by utilizing their large degrees of freedom and versatility to help humans in their daily life. Complex tasks are essentially classified into parallel tasks that execute multiple sub-tasks simultaneously and sequential tasks that execute them in a certain order. For example, moving an object by a hand while walking is the former, and grasping and putting a cup into a cupboard is the latter. In such complex tasks, the optimal motion is determined subject to physical laws, kinematic constraints and various performance criteria.

Conventional motion planning problem for robots including humanoids is defined as a problem to design a trajectory from an initial state to a goal state [1]. To date, several planning methods have been proposed. One major approach is to perform a search in the configuration space of the robot [2]. Some studies have focused on parallel tasks. Vahrenkamp et al proposed a planning for dual-arm manipulation of humanoids by using RRT [3]. Yoshida et al proposed a PRM-based whole-body motion planning method of humanoids for conveying a large object [4]. Sequential tasks, on the other hand, have been treated by planning motions for sub-tasks separately and putting them together. In [5], Tsukagoshi addressed a sequential task of moving towards an object and grasping it. In this study, not only the walking and grasping motions but also the standing position at which the robot grasps the object is planned.

In motion planning involving multiple sub-tasks, we need to plan not only trajectories but also optimal order and timing for executing sub-tasks. Keith et al proposed a method that determines the time-optimal execution of overlapping tasks [6]. One shortcoming in their approach is that the scheduling is treated purely in the task-space, and therefore the kinematics and dynamics of the robot is not fully taken into account. Moreover, the order of task-execution is predetermined. Liu et al proposed a technique for multi-character motion generation that optimizes the placement and timing of motion constraints simultaneously based on time-warping. The intention of timing adjustment in their research is in constructing multi-character motion by combining simple motion clips stored in the database. Since the time-warping function must be monotonic, it seems difficult to adopt this technique to motion planning problems in which task execution order must be decided by the planner.

Motivated by the above backgrounds, the authors have proposed a method that enables simultaneous planning of motion and task-achievement timing for sequential tasks of humanoid robots [8]. This method enables planning both trajectories and timings simultaneously, by introducing timing variables expressing task-achievement timing. In this setting, the motion planner plays a role of a task scheduler, which determines the optimal execution order and timing of multiple sub-tasks. In the previous work, however, we have addressed sequential tasks only, and moreover, each sub-task was limited to a punctual task. Here, punctual task ([6]) is a type of tasks that a certain condition must hold at a point of time. In this paper, the method is extended to handle both sequential and parallel tasks where each sub-task may be an interval task. An interval task is expressed as a constraint that must hold for a duration of time.

The rest of this paper is organized as follows. In Section II, multi-body system is introduced for modeling the robot and the workspace and kinematic and physical constraints are formulated based on multi-body dynamics. In Section III, tasks are formulated as constraint conditions on the trajectory of multi-body system and the timing variables. Next, in Section IV, the motion planning problem is formulated as a constraint satisfaction problem. In Section V, we briefly review an algorithm for solving this problem proposed in the previous study. In Section VI, the proposed method is applied to the object catching task of a humanoid robot and tested in numerical simulations. Finally, Section VII summarizes this paper with comments on further extensions.

Notation The symbol $i : j$ denotes a sequence of integers $i, i+1, \ldots, j$.

II. MULTI-BODY SYSTEM

In the proposed motion planning method, the entire scene including the robot and the workspace are expressed as a multi-body system. A multi-body system is composed of two kinds of elements: rigid bodies and joints. We use the subscripts $i$ and $j$ for indexing rigid bodies and joints,
Kinematic and Physical Laws

The motion of a rigid body is expressed by the trajectories of its position and velocity, both having translational and rotational components, and it is governed by physical laws and geometric constraints imposed by joints connected to the rigid body.

Suppose the $i_1$-th and the $i_2$-th rigid bodies are connected by the $j$-th joint. The kinematic constraints imposed by the joint are given by

$$ p_{i_1}(t) + R_{i_1}(t) \tilde{p}_{i_1,j} = p_{i_2}(t) + R_{i_2}(t) \tilde{p}_{i_2,j}, $$

$$ R_{i_1}(t) \hat{R}_{i_1,j} R(\eta_j, \theta_j(t)) = R_{i_2}(t) \hat{R}_{i_2,j}. $$

Here, $R(\eta, \theta)$ denotes a matrix expressing a rotation along the axis $\eta$ in the angle $\theta$. Moreover, $\tilde{p}_{i,j}$ and $\hat{R}_{i,j}$ are constant parameters expressing the configuration of the joint relative to the rigid body’s coordinate frame (see Fig. 1).

For the $i$-th rigid body, whose mass and inertia matrix are $m_i$ and $I_i$, the following equations of motion about the acceleration $a_i$ and the angular acceleration $\alpha_i$ hold:

$$ m_i a_i(t) = f_{i,ext}^e(t) + \sum_{j} f_{j}(t) - \sum_{j} f_{j}(t), $$

$$ I_i \alpha_i(t) = \tau_{i,ext}^e(t) + \sum_{j} [\tau_j(t) + (R_i(t) \hat{p}_{i,j}) \times f_{j}(t)] $$

$$ - \sum_{j} [\tau_j(t) + (R_i(t) \hat{p}_{i,j}) \times f_{j}(t)]. $$

The above equations require that total force applied to a rigid body is the sum of all forces applied by joints connected to it plus the external force. Here, $f_{i,ext}^e$ and $\tau_{i,ext}^e$ denote the external force (most typically the gravitational force) and the external moment applied to the rigid body, respectively. The trajectories of external force and moment are specified a priori and are not subject to planning. Here, for simplicity, we assume that the moment of inertia of the rigid body is constant regardless of the rotation axis. This is solely for simplicity and it is possible to handle asymmetric inertia by minor extension.

III. Tasks

The trajectory of a multi-body system must be not only kinematically and physically consistent but also satisfying the required task specification. A task specification is expressed by a set of task constraints, indexed by the subscript $n \in \{1, N\}$. Each task constraint is imposed on a pair of rigid bodies to let a certain variable of these rigid bodies be equal for a duration of time.

In this paper, we focus on one typical type of tasks; the position-matching task. The task constraint condition imposed between the $i_1$-th and the $i_2$-th rigid bodies for the time interval $[t_n^e, t_n^f]$ by a position-matching task is expressed as follows:

$$ p_{i_1}(t) - p_{i_2}(t) = 0 \quad \forall t \in [t_n^e, t_n^f] $$

Fig. 2 illustrates the position trajectories of the $i_1$-th and the $i_2$-th rigid bodies satisfying the constraint (3). Note here that not only the position trajectory of each rigid body but also the task achievement time interval $[t_n^e, t_n^f]$ is subject to planning. A task is an interval task when $t_n^e < t_n^f$ or a punctual task when $t_n^e = t_n^f$. Although not explained in detail, it is not difficult to put additional constraints to explicitly specify the tasks starting time $t_n^e$, the task finishing time $t_n^f$, and even the task duration $t_n^f - t_n^e$.

IV. FORMULATION OF MOTION PLANNING

A. Primary Problem Formulation

Let us denote by $x$ the state of the multi-body system obtained by aggregating the variables of all rigid bodies and joints listed in Table I into a single vector. Then, motion planning is to compute a trajectory $x(t)$ over a finite time interval $[0, T]$ that satisfies a set of constraint conditions described in the previous sections.
 Kinematic and physical constraints are expressed as a single equality constraint on \( x \):

\[
e^i(x(t)) = 0 \quad \forall t \in [0, T].
\] (4)

Note that this type of constraints restricts the trajectory over the entire planning horizon \([0, T]\) (Fig. 3(a)).

Task constraints, including but not limited to the position matching task, are expressed as follows.

for every \( n \in 1 \ : \ N, \)

\[
e^i(x(t)) = 0 \quad \forall t \in [t^e_n, t^s_n], \quad 0 \leq t^e_n \leq t^s_n \leq T.
\] (5)

Each task constraint restricts the trajectory over the subinterval \([t^e_n, t^s_n]\) (Fig. 3(b)), which is also subject to planning.

To summarize, the motion planning problem is stated as follows:

find \( x(t) (t \in [0, T]) \) and \( \{t^s_n, t^e_n\} \ (n \in 1 \ : \ N) \)

that satisfies (4) and (5). (6)

**B. Trajectory Representation and Constraint Approximation**

Theoretically, the motion of a multi-body system \( x(t) \) is defined as a trajectory in continuous time \( t \in [0, T] \). It is technically difficult, however, to compute \( x(t) \) satisfying the constraints (4) and (5). To cope with this problem, we employ quadratic interpolation to express continuous-time trajectories by a finite number of variables. First, let us introduce a sequence of time instants \( \{t_k\} \ (t_0 = 0 < \cdots < t_k < \cdots < t_K = T) \). In contrast to the timing variables, this time instants are treated as constants in the planning algorithm. This is an essential difference from the time-warping technique [7], in which the sequence \( \{t_k\} \) itself is modified through optimization. Indeed, separating the timing variables from trajectory representation enables the timing variables of different sub-tasks to change their order during the optimization process and thus enables the planning of task execution order.

The position trajectory of the \( i \)-th rigid body over \([t_k, t_{k+1}]\) is given by

\[
p_i(t) = (1 - s^2)p_{i,k} + s^2 p_{i,k+1} + s(1 - s)h_k v_{i,k} =: [p_{i,k} \ v_{i,k} \ p_{i,k+1}] S_k(t)
\] (7)

where

\[
h_k = t_{k+1} - t_k, \quad s = (t - t_k)/h_k,
\]

\[
S_k(t) = \begin{bmatrix} 1 - s^2 \\ s(1 - s)h_k \\ s \end{bmatrix}
\] (8)

and \( p_{i,k} \) and \( v_{i,k} \) are the values of position and velocity \( p_i(t) \) must take at \( t = t_k \). The \( k \)-th trajectory segment is illustrated in Fig. 4. In this manner, the continuous-time position trajectory is uniquely determined when positions and velocities at all \( t_k \)s are specified. However, the first derivative of \( p_i(t) \) may be discontinuous at some \( t_k \) if the \( \{p_{i,k}, v_{i,k}\} \) are given arbitrarily. In order to ensure the first derivative to be continuous over \([0, T]\), \( \{p_{i,k}, v_{i,k}\} \) must be set in a way that first derivatives of \( p_i(t) \) at each \( t_k \), from both directions, take the same value. This constraint condition is given by

\[
\frac{p_{i,k+1} - p_{i,k}}{t_{k+1} - t_k} = \frac{v_{i,k+1} + v_{i,k}}{2} = 0
\] (9)

for all \( k \in 0 \ : \ K - 1 \). The orientation of a rigid body is also expressed as a piecewise quadratic curve and a similar continuity constraints are imposed on rotational trajectories.

Once the trajectories are represented as piece-wise quadratic curves, one can make the trajectories to approximately satisfy the physical constraints by manipulating their keypoints. More precisely, the equality constraints (1) and (2) are treated only on each \( t_k \).

Handling task constraints, which involve timing variables, is more complicated. The following discussion is focused on the position-matching task. Note that other types of tasks that can essentially be expressed as matching two trajectories of certain physical variables (e.g., velocity matching) can be treated in a similar manner.

The position-matching task is expressed as (3) and the trajectory of each rigid body is now given by (7). Therefore, the \( n \)-th position-matching task is an equality constraint on the following set of variables:

\[
t^e_n, t^s_n, \ \{p_{i_1,k}, v_{i_1,k}, p_{i_2,k}, v_{i_2,k}\} \ (k \in \mathcal{K}(t^e_n, t^s_n)).
\]

Here, \( i_1 \) and \( i_2 \) are the subscripts of the target rigid bodies. The function \( \mathcal{K}(t^e, t^s) \) returns an integer sequence \( k^e : k^e \)

satisfying

\[
t_k^e \leq t^e < t_{k+1}^e, \quad t_{k-1}^e \leq t^e < t_k^e.
\]
C. Modified Problem Formulation

Observe that the set of variables involved in the position-matching task changes according to the values of $t_n^p$ and $t_n^o$ (see Fig. 5). The set of constraint conditions is given as follows:

$$
\begin{align*}
[p_{11,k} & \quad v_{11,k} & \quad p_{11,k+1}] S_k(t_n^p)
= [p_{21,k} & \quad v_{21,k} & \quad p_{21,k+1}] S_k(t_n^o), \quad (10a) \\
[p_{11,k-1} & \quad v_{11,k-1} & \quad p_{11,k}] S_{k-1}(t_n^p),
& \quad (10b) \\
[p_{11,k-1} & \quad v_{11,k-1} & \quad p_{11,k}] S_{k-1}(t_n^o),
\end{align*}
$$

$p_{11,k} = p_{21,k} \quad \forall k \in [k^e + 1, k^e - 1]$ \quad \quad (10c)

As a result, position matching task is expressed as a set of equality constraints on the trajectory keypoints and the timing variables.

C. Modified Problem Formulation

Now, let $z$ be a vector variable containing all trajectory keypoints (i.e., the values of $t_k$ at all $t_k$, $k \in 0 : K$, where $x(t)$ is defined in Subsection IV-A) and all timing variables (i.e., $t_n^p$ and $t_n^o$ for all $n \in 1 : N$). Then, after approximating the constraint conditions as described in the previous subsection, the problem (6) is expressed as a constraint satisfaction problem on $z$:

$$
\text{find } z \text{ that satisfies } c(z) = 0. \quad (11)
$$

V. CONSTRAINT SATISFACTION ALGORITHM

In this section, we explain the overview of the optimization algorithm used for motion planning. More detailed explanation is given in [9]. An algorithm described below is an iterative algorithm, which starts from an arbitrary set of initial values of $z$ and updates it repeatedly until $c(z) = 0$ is satisfied. Define the total constraint error as

$$
e = \| y \|, \quad y = c(z).$$

If we were able to manipulate $y$ directly, the simplest way to reduce $e$ is to move each $y$ in the steepest descent:

$$
dy = -\mu y$$

where $0 < \mu < 1$ is the step size. The problem is not so easy, however, since we can change the value of $z$ only. From partial differentiation, we have

$$
dy = J(z)dz, \quad J(z) = \frac{\partial c}{\partial z}(z).$$

Therefore, we would like to obtain $\delta z$ satisfying

$$
J\delta z = -\mu y.
$$

This is a linear equation of $\delta z$ and it is normally under-determined. Therefore we try to find the value of $\delta z$ that minimizes the cost $\|W\delta z\|^2$. Here, $W$ is a diagonal matrix for weighting. Then $\delta z$ minimizing this cost is given by

$$
\delta z = J^T \lambda
$$

where $J^T$ is the transpose of $J$ and $\lambda$ is given as a solution to the following linear equation:

$$(JW^{-2}J^T)\lambda = -\mu y.$$

The key idea of this method is to determine $\delta y$ first based on the steepest descent, and then compute $\delta z$ that realizes $\delta y$. Of course, it is possible to perform the steepest descent in the $z$-space, but then finding an appropriate step size becomes quite severe due to the strong nonlinearity of $c$, making it difficult to achieve fast convergence while avoiding instability. Although not explained here, the method can handle constraints in multiple priority-levels by making a small modification to the algorithm.

Due to the nonlinearity of the planning problem, the method may converge to a local optimum dependent on initial conditions of decision variables. From the motion planning perspective, this means different combination of initial timing variables might result in completely different motion. Achieving the globally optimal solution regardless of the initial condition is one of the goals of our future study.

VI. SIMULATION RESULTS

In this section, we consider the motion planning of a humanoid to catch flying objects by its hands. The specification of object catching is expressed by imposing position-matching tasks between the hands of the robots and the objects. The planning method is implemented in the C++ programming language and executed on a computer equipped with 3.0GHz CPU and 3GB RAM.

A. Object Catching Task

Each object moves from its initial position in a constant velocity; the position trajectory of the $n$-th object is given by

$$
p_n^o(t) = p_{n,0}^o + v_{n,0}^o t
$$

where $p_{n,0}^o$ and $v_{n,0}^o$ are the initial position and the initial velocity of the $n$-th object, respectively.

For each object, a position-matching task is assigned between the object and the robot’s left or right hand. Moreover, the amount of time the hand must be kept touching the object is specified as follows:

$$
t_n^c = t_n^s + T_n^f.
$$

In object catching, it is reasonable to ask the robot to catch all object as soon as possible. This requirement is implemented by the following constraint:

$$
t_n^o = 0.
$$
The initial values of the timing variables that are provided together with the trajectory of the robot.

A. Sequential Catching

A sequential task is to execute multiple sub-tasks one after another. Note that an appropriate execution order is not given a priori. In this example, we plan the robot motion to catch two objects sequentially with its left hand.

The initial positions and velocities of the objects are set as

\[
p_{1,0} = \begin{bmatrix} 3.0 \\ -2.0 \end{bmatrix}, \quad v_{1,0} = \begin{bmatrix} -0.25 \\ -0.25 \end{bmatrix}, \quad p_{2,0} = \begin{bmatrix} -3.0 \\ 6.0 \end{bmatrix}, \quad v_{2,0} = \begin{bmatrix} 0.75 \\ -0.5 \end{bmatrix}
\]

and the task durations are set as \( \{T_1^1, T_2^1\} = \{0, 0\} \). Initial values of the timing variables are set as \( t_1^1 = t_2^1 = 2.4 \) and \( t_3^1 \). The algorithm produced \( t_1^1 = t_1^2 = 1.84 \) and \( t_2^2 = 4.37 \) after 1774 iterations. In Fig. 7, we observe that the distance between the hand the balls become near 0 at the respective catching time. Fig. 9 shows the planned motions. At first, the humanoid catches the object 1 (ball) at \( t = t_1^1 \) with its left hand. Next, it catches the object 2 (ball) at \( t = t_2^2 \).

D. Discussion

The results of both parallel and sequential tasks show that the proposed method plans the humanoids motion and timings that achieve the given tasks. Thanks to timing variables, motion planning of structurally different tasks, parallel and sequential tasks, can be handled in the same constraint satisfaction framework. At the moment, planned motions are more or less unrealistic when seen as human motions, and adding motion realism is a long-term goal. One quick recipe would be to put additional constraints, for example, on the range of joint angles and the center of mass.

VII. CONCLUSION

In this study, motion planning involving both trajectory planning and task scheduling is formulated as a constraint satisfaction problem. In the early part of this paper, constraints are classified into two categories: physical constraints and task constraints. It has been shown that a position-matching task can be expressed as a set of equality constraints on the trajectory keypoints inside the time interval defined by a pair of timing variables, so that it can be incorporated into the existing constraint solving algorithm.

There are challenges for the future study:

- handling collision avoidance between rigid bodies as constraints,
• improving the convergence speed of the planning algorithm, and
• handling more complex task specifications including logical conditions.

REFERENCES


