Robotica (2015) volume 33, pp. 705–720. © Cambridge University Press 2014
doi:10.1017/S0263574714000496

Humanoid robot upper body motion generation using B-spline-based functions

M. Ruchanurucks*

Electrical Engineering, Kasetsart University, Bangkok, Thailand

(Accepted February 7, 2014. First published online: March 10, 2014)

SUMMARY

This paper aims to represent a human's upper body motion using a humanoid robot. As a robot has different physical limitations than a human, we present a method that can filter the trajectories to meet the limitations. The filtering can be used directly and also can be used as a constraint for optimization. It will be shown to be applicable both offline and online. Many physical attributes, namely angle, collision, velocity, and dynamic torque, are represented as B-spline explicit functions. The B-spline coefficients are then calculated to limit the physical attributes. This explicit method can guarantee that many limits are met. Hence, it is better than many methods that only reduce such physical attributes using objective functions.

KEYWORDS: B-spline; Motion generation; Robot dynamics; Robot kinematics.

1. Introduction

Many areas are being explored in the field of humanoid robots, such as motion generation, balance control, man-machine interfaces, and artificial intelligence. However, this work focuses on space-time motion generation methods within a robot's physical constraints.

It is widely known that motion generation methods for robots have had limitations because of the physical capabilities of humanoid robots. There are limits relating to physical attributes, such as angle limits, while velocity is limited by back electromotive force (emf), and torque is limited by actuator power and collision.

For an industrial robot, the physical constraint problem can be solved by allowing the robot more time to finish a task, to satisfy the limits. The main requirement is the precision in the world coordinate system, the so-called "space constraint."

With the emergence of humanoid robots, there is a new problem called the "space-time constraint" in which the robot is required to finish a task within a certain time.

Currently, without considering balance control, many research groups are trying to solve the space-time problem within the physical limits. The significance of such motion generation is clear, because in many works, balance control is realized but methods to deal with other physical limits are not effective. If such limits are not satisfied, the trajectories will have a large error, which leads to collision problems and balance inconsistency. If these are not prevented, the robot will be damaged. However, no research has been successful in limiting all physical characteristics. We also do not claim that we have been successful; however, this research will show that we are closer than many previous works, namely, refs. [1]–[12], in some certain aspects.

Among motion generation methods, a filter-based approach is a fast way to retarget motion from captured or planned trajectories. Pollard *et al.*[1] realized dances using a humanoid robot without balance control by scaling the angle and filtering velocity. This method is successful in the sense that angle and velocity limits are guaranteed. However, since each joint is scaled separately, the overall motion may be different from the desired path.[2] Second, though the joint angle and velocity are limited to the actuator capabilities, collision and torques are not considered. Third, the method cannot be applied for optimization as constraints; since it processes each data input directly while optimization

* Corresponding author. E-mail: fengmtr@ku.ac.th

usually considers a trajectory as a function. Finally, it cannot be used for real-time motion generation since the velocity filter requires running it forward and backward iteratively. Hammam and Orin[3] focused on a zero moment point (ZMP) filter although an upper-body task was mentioned in that work.

Trajectory optimization is another approach for motion generation. Although it generally consumes more time than the filtering approach, it can ensure the smoothness and fidelity of the overall motion by tradeoffs between multiple objective functions and constraints. Furthermore, the smoothness is realized by representing the trajectory as a nonlinear function, such as a B-spline or wavelet. Lee and Shin[4] retargeted characters for computer graphics using a hierarchical B-spline, but this cannot be used in a real robot, because many physical limits are not checked. Ude *et al.*[5] performed trajectory optimization for a humanoid robot using B-spline wavelets that can represent the detailed motion well. However, their algorithm does not deal with physical limits effectively. Luo and Hauser[6] used a B-spline and considered physical limits as a set of box limits in an optimization scheme. They seemed to focus more on objective functions. Moulard *et al.*[7] relied also on box limits and further took ZMP into account. Perhaps the most advanced work that is close to the offline optimization part of our work is Khoury *et al.*[8] They considered many physical limits similar to ref. [6]. However, they further dealt with collision avoidance by considering it separately. However, there is no clear indication whether all physical limits conformed to each other.

A real-time approach also presents problems, as described in ref. [9]. Shin and Kim[10] used the virtual force of a spring-damper to mimic a human. Such research has mostly considered the relations between captured motion and a kinematics model. Attempts to limit physical characteristics rely on kinematics-based optimization to reduce values, rather than applying specific limitations, as described in ref. [11]. Even an advanced work like that of Erez *et al.*[12] still relied on cost functions to cope with physical limits. We argue that the physical attributes, in the case of real-time motion generation, should be limited using constraints.

The former methods tended to just reduce the values rather than reflecting the true limits. Even if hard limits are applied, there are problems with each constraint or conflicts between constraints.[13] This is a very difficult challenge that has caused considerable discussion.

This paper presents a method for solving angle, collision, velocity, and torque limits, based on the B-spline function, targeting both offline and online environments. While maintaining the B-spline's current strength in ensuring smoothness, value is added to the use of the B-spline by representing all physical constraints in terms of its coefficients.

It will be shown how to use B-spline coefficients in angle, velocity, and dynamic torque constraints, with specific limits guaranteed. The velocity and torque constraints are influenced by an iterative soft-constraint paradigm that makes limiting very effective. Self-collision avoidance is also considered as a constraint.

The proposed constraints can be directly used for an offline filtering approach. In this approach, a novel and effective method to decompose a trajectory to a B-spline is presented to replace a hierarchical decomposition algorithm.

In the offline optimization approach, the objective function's complexity is reduced to only mimicking a human trainer, which would enhance the convergence rate even when the function is used to do other jobs as well, such as reducing energy consumption. A hierarchical B-spline is selected since its hierarchical structure enables successive refinements of motion more precisely than a non-hierarchical B-spline. In order to locate the problem period that should be re-optimized in higher hierarchies, the traditional trajectory's error detector is used before the knot insertion process. Also, a knot density detector[13] is used to facilitate knot insertion.

In a real-time filtering approach, our angle, collision, and velocity constraints can be applied directly. Unfortunately, torque constraint, which requires running forward and backward iteratively, might not be suited to this approach. A different technique to limit torque is presented; however, it is not effective because it requires many changes in trajectories. At the present time, real-time torque constraint still needs improvement.

The test data are the Japanese traditional dances captured from professional dancers as test motions. These dances are very complex; their motions exceed many physical limits of our robot and cannot be easily performed by any humanoid robot using existing methods. If our algorithm could deal with such dances, it would benefit other kinds of motion research as well.

Data are acquired using an optical motion capturing system, which can record the time series properties of marker positions and trainers' motions. Thirty-three markers are attached to a trainer,

and the body motion is captured as a sequence of markers. Then, using inverse kinematics, related positions of markers are converted to joint angles, which are inputs to our system.

The remainder of the paper is organized as follows. We describe the physical constraint functions and explain how to use them in Section 2. Section 3 explains how to decompose a trajectory into a B-spline curve and how to use the constraints in a filtering approach. Section 4 focuses on issues related to using the constraints in an optimization approach and motion refinement based on a hierarchical B-spline. Section 5 explains a real-time approach and the issues related to adaptation of the constraints along with some drawbacks. Finally, a discussion and a conclusion are presented in Section 6.

2. Physical Constraint Functions

Presently, various problems occur when attempts are made to limit physical attributes. Four physical attributes govern the movement of the upper body of a humanoid robot, namely, angle, collision, velocity, and torque. These attributes must be limited for many vital reasons.

If angle and torque are not well limited, the robot trajectories would be different from the planned trajectories as the robot does not have the capability to follow the planned motion. At first glance, this seems to be a negligible problem, however, it is not. Error in trajectories could lead to two serious problems—collision and balance inconsistency. Whereas the collision problem due to a violated angle is clear and present, and inadequate torque also leads to an incorrect angle, balance inconsistency is more obscure. Balance inconsistency is related to the whole body control of the humanoid robot. There are various methods to control the balance of the robot. Practically, they usually require the upper body motion to be planned prior to adjusting the whole body posture, such as by re-calculating the waist angle.[14] However, if the upper body motion is different from the planned one, the center of gravity and zero moment would be different from the planned whole body motion, and the robot could collapse. This must be prevented by applying a method that effectively limits angle and torque. Though angle limit can be achieved easily, torque is often not well limited due to the complexity of the dynamic equation. Most previous works can only reduce the torque attribute, rather than giving it definite limits, as our method does.

Velocity limits pose another problem. Usually, for most kinds of electric actuators used in humanoid robots, there is a back emf that increases proportionally with velocity. Excessive emf could lead to undesired actuator wear.

Collision is a clear and present problem that must be avoided. Various collision avoidance methods can be found in the literature, such as: the one in ref. [15], which used an interaction mesh;another one in ref. [16], which showed whole body collision avoidance of a humanoid robot stressing how to integrate collision avoidance; and one more in ref. [17], which contained the notion of check points. The idea of check points placed on the robot's body is adopted in the current work. In this paper, the problem of collision avoidance is not considered as an objective function to be reduced but as a constraint to be limited. Our experiment proved this to be effective, when compared with methods in works such as ref. [18], which only uses an objective function.

In order to ensure the smoothness of trajectories and to create our constraint functions, the choice of the curve representation method is very important. The choice determines how the constraint functions work.

The following portions of this section start with a discussion of the curve representation. Then, based on the best representation method, the physical constraints of angle, collision, velocity, and torque from ref. [19] are explained. The velocity and torque constraints are influenced by an iterative soft-constraint paradigm that makes limiting very effective.

2.1. Data representation

In the robotics field, a curve representation method called B-spline is widely used for manipulator motion-planning[20] and even for intelligent control.[21] The important characteristics of B-spline are: first, changing a parameter of B-spline function affects only a limited range of a curve and, second, the method involves hierarchical refinement. In practice, a cubic B-spline, a third-order function of a trajectory, is usually employed in order to ensure continuity in the angle, velocity, and acceleration domains, and also by implication applies to the torque domain that is a function of these three parameters.

https://doi.org/10.1017/S0263574714000496 Published online by Cambridge University Press

Recently, a data representation method called wavelets[22] has been used in many fields. Wavelets have many variations that make them suitable for dealing with different problems. As wavelets share two important desirable properties with a B-spline (local adaptation and hierarchical refinement), some previous works[23] have compared the advantages of these techniques. In the robotics field, Ude *et al*.[24] proposed using B-spline wavelets for trajectory optimization. Their comprehensive work can be found in ref. [5]. The idea involves using a B-spline as a base curve and adding wavelets to sections that contain error compared to the original trajectory. It is understandable that the authors were trying to enhance the convergence characteristics of B-spline by adding wavelets, as the latter converge faster than the former if the trajectory contains an inadequate space constraint.[23] This, however, is not the case for motion generation, as a large amount of data can be derived from a human trainer or even from the trajectories the robot plans itself.

In order to select between these two curve representation methods, with the criterion in mind that it must be exploitable for constraint functions, B-spline is the method of choice as its angle, velocity, and acceleration functions (also implying the torque function) have a clear structure compared to wavelets. Furthermore, a B-spline is superior to wavelets not only for its simplicity but also for its usability in a real-time approach, as will be shown in Section 5. (Effecting an online torque limit faces a problem due to the B-spline acceleration function. But since a wavelet's velocity shares the similar physical structure as a B-spline's acceleration, it would be ineffective to use wavelets in real time for both velocity filtering and torque limiting.)

Industrial robots have long been using a B-spline with physical constraints in trajectory planning.[25,26] However, such methods apply limits by scaling the time domain, which is not applicable for a space-time constraint problem often found in the case of a humanoid robot. In this work, the B-spline-based constraints for a humanoid robot are presented as follows.

2.2. *Angle limits*

A cubic B-spline has as a characteristic that its amplitude will not be higher than the magnitude of a control point. Hence, angle limiting can be implemented directly by applying bounded constraints to the magnitude of the control points (p_n) in a B-spline function as shown below.

$$q(t) = \frac{1}{6}\{(-t^3 + 3t^2 - 3t + 1)p_0 + (3t^3 - 6t^2 + 4)p_1 + (-3t^3 + 3t^2 + 3t + 1)p_2 + (t^3)p_3\}, \quad (1)$$

where q is angle, t is time, and p is a control point.

2.3. *Self-collision avoidance*

In order to decide whether or not collision occurs, check points are placed on or inside a robot body and an arm is considered as a link of cylinders. Collision is detected if the distance between a cylinder and any check point is lower than a certain value derived from the physical structure of the robot.

All angles responsible for moving such a cylinder are then searched to see which one requires the least change to avoid collision, which implies the maximum sensitivity in (2).

$$j = \arg\max_j(d/\Delta p_{n,j}), \quad (2)$$

where j is the joint number, d is the distance between a cylinder and a check point, and n is the index of control point.

For example, to avoid collision on the lower arm, three joints in the shoulder and a joint in the elbow are explored. Searching occurs until an angle is reached that can avoid collision which is preferable to just checking the sensitivity of each angle's ability to move away from a collision check point, since every joint provides sensitivity but may reach the angle limit before being able to avoid collision. The result of avoiding collision of the head is shown in Fig. 1.

In practice, we tried to do collision avoidance using an objective function proposed in ref. [18], and found that solving it using a constraint was more effective, since any objective function tends to just reduce a value instead of hard limiting it. It is also straightforward to choose critical distances from collision check points.

https://doi.org/10.1017/S0263574714000496 Published online by Cambridge University Press

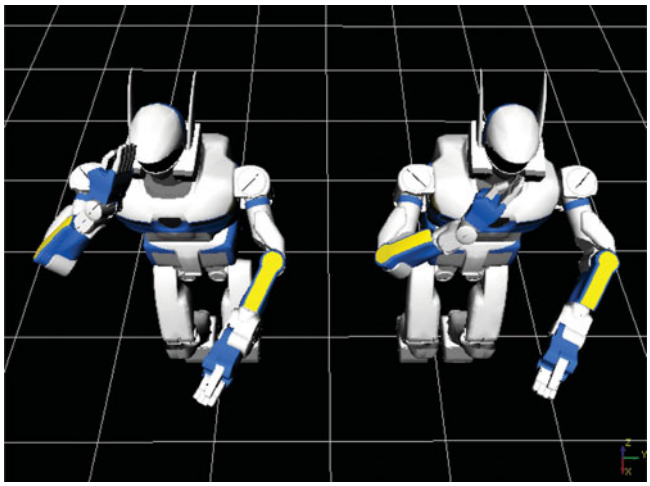


Fig. 1. A collision check point is placed at the center of mass of a robot's head: (left) without collision avoidance; (right) with collision avoidance.

2.4. Velocity limits

Velocity must be constrained mainly due to the back emf of an electric motor, which increases proportionally with velocity and needs to be suppressed to avoid damage to the actuator.

From (1), velocities at the beginning of a period and at the middle of the previous period are:

$$\dot{q}(\text{present} : \text{begin}) = \frac{p_2 - p_0}{2T}, \tag{3}$$

$$\dot{q}(\text{previous} : \text{middle}) = \frac{p_2 + 5p_1 - 5p_0 - p_{-1}}{8T}, \tag{4}$$

where T is the length of the knot period, *present/previous* represents a time consideration, the current/former set of four control points, in (1), respectively, begin/middle represents a spatial consideration, the *start/middle* of a section, of the trajectory, that is generated using the *present/previous* set of four control points, respectively.

Interestingly enough, (3) must be checked before (4) to avoid divergence, by altering p_2 and p_1, respectively. This is based on two criteria. The first criterion is the sensitivity of altering the control point (a higher sensitivity control point affects the shape of a curve more). This must be high for checking the present period and low for past or future periods.

It can be seen that there are similar ways that meet the criterion; however, these will also lead to divergence or ineffective limiting. Thus, the second criterion is required: after changing p_2 in (3), altering p_1 in (4) does not affect (3) and affects previous checking only in a supportive way.

It is likely that peak velocity would not occur at the beginning or the middle of a knot period. By checking at acceleration's zero crossing point, maximum velocity is shown in (5).

$$\dot{q}_{\max} = \frac{1}{2T}\left[(p_1 - p_{-1}) + \frac{(p_1 - 2p_0 + p_{-1})^2}{-p_2 + 3p_1 - 3p_0 + p_{-1}}\right]. \tag{5}$$

However, we found it was adequate to use only (3) and (4); (5) is not required.

2.5. Dynamic torque limits

In a humanoid robot, the actuator's power is limited for many reasons such as size and the capability of the actuator. Thus, to represent abrupt human motions, a dynamic torque limit is needed.

The torque in each joint can be calculated from the inverse dynamics equation in (6).

$$\tau_i = \sum_j M_{ij}(Q)\ddot{Q}_j + I\ddot{Q}_i + \sum_{j,k} C_{ijk}(Q)\dot{Q}_j\dot{Q}_k + G_i(Q), \tag{6}$$

https://doi.org/10.1017/S0263574714000496 Published online by Cambridge University Press

where τ is the applied torque, M is the inertia matrix, I is the actuator's inertia of the present joint, C is a centripetal and Coriolis forces matrix, and G is gravitational loading.

The above equation is used to limit values by altering a control point to alter Q and its derivatives, first inserting velocity and acceleration in the form of a B-spline. Note that since the dynamic equation's parameters are nonlinear in Q, it can be considered to be constants that are updated recursively.

$$\tau_i = \sum_j M_{ij} \left(\frac{p_2 - 2p_1 + p_0}{T^2} \right)_j + I \left(\frac{p_2 - 2p_1 + p_0}{T^2} \right)_i + \sum_{j,k} C_{ijk} \left(\frac{p_2 - p_0}{2T} \right)_j \left(\frac{p_2 - p_0}{2T} \right)_k + G_i. \tag{7}$$

Since the acceleration is a straight line, its peak values are located at the beginning of each knot period, and the acceleration multiplies M. M greatly influences the dynamics equation; therefore, limiting torque only at the beginning of a knot period is sufficient.

As stated above, one of our criteria was choosing to change a control point that requires the least change to limit torque. From (7), it can be seen that the term multiplies M, changing p_1 provides the highest sensitivity. Thus, if one rearranges the function to isolate terms that contain p_1 from those that do not,

$$\tau_i = \sum_j (M_{ij} + I) \left(\frac{-2p_1}{T^2} \right)_j + \sum_j (M_{ij} + I) \left(\frac{p_2 + p_0}{T^2} \right)_j + \sum_j \sum_k C_{ijk} \left(\frac{p_2 - p_0}{2T} \right)_j \left(\frac{p_2 - p_0}{2T} \right)_k + G_i. \tag{8}$$

This is a set of nonlinear equations that cannot be solved explicitly. Consequently, they are solved by recursively searching for the value p_1 of joint i alone that satisfies a limit. The hill climbing method is used for this recursive search. A new problem arises upon the use of this method—changing p_1 often results in the torque of a previous period being larger than the limit, making torque constraint ineffective. This can be solved using the method described next.

2.6. *Iterative soft limits*

Suppose that a value that needs to be reduced is 1. The easiest way to do this is $1 - 1 = 0$; however, this poses the problem stated above, in case such a value represents torque. Another way to do this is:

$$1 - x \sum_{i=0}^{\inf} (1 - x)^i = 1 - 1 = 0; \quad x < 1. \tag{9}$$

In the case of torque constraints, instead of limiting torque to the desired value, reduce it only by amount x of the needed amount iteratively, say, 0.5 with 46 iterations.

$$1 - 0.5 \sum_{i=0}^{45} (1 - 0.5)^i = 1 - 0.99999999999999 \approx 0. \tag{10}$$

In doing so, we can solve such a problem as ineffective torque constraint.

This method is also useful for a velocity constraint, as the shape of trajectories after applying such a constraint leans on the right side because limit checking is performed forward. Symmetry can be achieved by using these iterative soft limits.

Furthermore, if both the velocity and torque constraints are to be used separately, the resulting trajectories would meet the torque limits while the velocity limits are often violated. Fortunately, if both are put under the iterative soft constraint, they will gradually converge to limit values.

https://doi.org/10.1017/S0263574714000496 Published online by Cambridge University Press

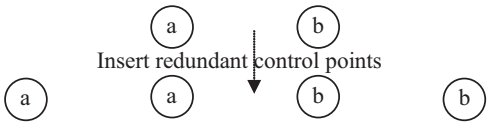


Fig. 2. Insertion of a redundant control point at the beginning and the end of the trajectory.

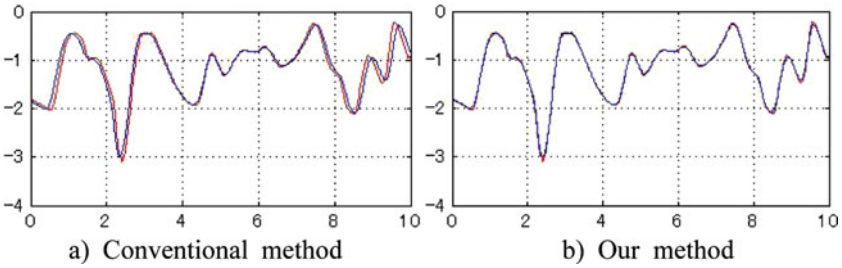


Fig. 3. Decomposition from 200 fps to 20 control points per second: (a) conventional method, (b) our method. Red, blue, and black lines represent original data, generated data, and limit lines, respectively. *X*-axis is time and *Y*-axis is each value specified in each subfigure. Angle is in radians, velocity is in radian/s, and torque is in kg m^2/s^2 .

It must be noted that these iterative soft limits are applicable only in offline cases, as it requires running forward and backward iteratively.

3. Filtering Approach

The fastest and easiest way to retarget motion to the robot with physical limits is filtering. Compared to well-known optimization works, optimizing a large dataset could be time consuming.

Since all of the constraints operate on a B-spline function in order to ensure smoothness, it is necessary to decompose any joint trajectory to a B-spline curve prior to limiting physical attributes.

3.1. Decomposition from raw data

Unfortunately, it is inappropriate to use B-spline hierarchical decomposition (see ref. [27] or [28]) in cases where it is extensively used for optimization approach. Let us explain the differences. For optimization, many works optimize a curve on the lowest hierarchy, with the smallest number of control points, first. Then, if the precision of the derived curve does not satisfy the error criteria, such a curve is hierarchically decomposed to a curve with a higher number of control points, using (18), and is re-optimized only in the period that contains the problem.

Differing from the optimization approach, where the appropriate value of control points is unknown at first, the filtering approach needs only a set of control points that can precisely represent the original trajectory. It is more like one-shot filtering on an already decomposed B-spline curve that first represents the original data.

On the other hand, if a trajectory is downsampled and then the derived samples are used as control points for a B-spline, as in the field of path planning, the decomposed curve will have a large error around the end of it as shown in Fig. 3(a). Many trajectory-planning works solved this problem by increasing the sampling rate. However, higher order trajectories such as those of velocity or acceleration would be jerkier.

By careful inspection of the B-spline structure in (1), we observed that the downsampling error occurs due to the lack of synchronization between the number of control point periods and the number of knot periods. In other words, from (1), it is shown that a knot period consists of three control point periods (the periods between four control points) instead of only one control point period. To synchronize these two domains, a redundant control point is placed at each end point. Using an example to prove this method, from two control points in Fig. 2, adding a redundancy control point to each end will result in four control points overall. Thus, the result is like a control point period, between a and b, corresponding to a knot period, generated from the four control points, as required. The result of this decomposition method is shown in Fig. 3(b).

https://doi.org/10.1017/S0263574714000496 Published online by Cambridge University Press

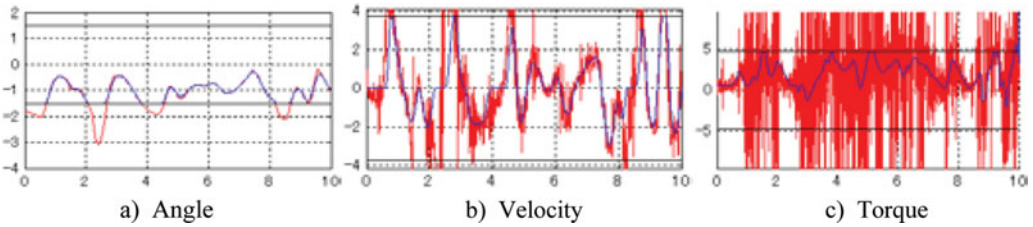


Fig. 4. Application of constraints simultaneously with fixed end posture: (a) angle [+/−1.5184], (b) velocity [+/−3.74129], and (c) torque [+/−4.8112]. Red, blue, and black lines represent original data, generated data, and limit lines, respectively. *X*-axis is time and *Y*-axis is each value specified in each subfigure. Angle is in radians, velocity is in radian/s, and torque is in kg m^2/s^2 .

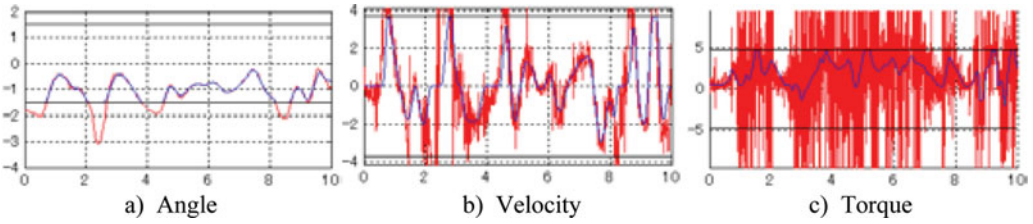


Fig. 5. Application of constraints simultaneously without fixing end posture: (a) angle [+/−1.5184], (b) velocity [+/−3.74129], and (c) torque [+/−4.8112]. Red, blue, and black lines represent original data, generated data, and limit lines, respectively. *X*-axis is time and *Y*-axis is each value specified in each subfigure. Angle is in radians, velocity is in radian/s, and torque is in kg m^2/s^2 .

An appropriate sampling rate was acquired using the error detector shown below.

$$e(n) = |q(n, p) - \theta(n)| > \text{thErr}, \tag{11}$$

where p is the vector of the present angles' control points, $q(n, p)$ is the curve magnitude of the B-spline, $\theta(n)$ is the angle from inverse kinematics, and thErr is the threshold for error determined heuristically by the user.

If the curve is not precise enough, the sampling rate is increased.

3.2. Filtering

After the B-spline curve has been derived, a set of constraints in Section 2 can be applied to the B-spline curve directly. For a combination of constraints, after angle limiting, collision avoidance is checked prior to the other constraints. This order is adopted because collision avoidance may pose discontinuity in trajectories, which can be solved by torque and velocity constraints. Collision may occur, but it can be solved automatically by increasing the critical distance for those periods that have collisions.

Users might prefer to set the initial and final postures of joint trajectories to be the same as the original data. This might be preferred in an application such as using a humanoid robot as a demonstrator. This can be achieved by fixing the first and the last three control points. However, due to such forcing, angle and collision problems might occur. Consequently, angle and collision constraints must be given higher priority than fixing the end point to the initial condition, as can be seen in Fig. 4(a). Although it is possible to fix the final posture, the initial angle must obey constraints.

Due to forcing of the end point position, an abrupt surge in the torque domain can be seen around the end of the trajectory, which could be larger than the torque limit. However, practically, this is not a problem since the characteristic of the actuator usually allows instantaneous torque to be more or less substantially higher than the constant torque limit.

If the end point is not fixed, an open-end curve results as shown in Fig. 5.

The effect of approximating the nonlinear dynamic equation of torque to be linear, (7), can be seen more clearly when the limit is set to be stricter, for example, set to half the real limit as shown

https://doi.org/10.1017/S0263574714000496 Published online by Cambridge University Press

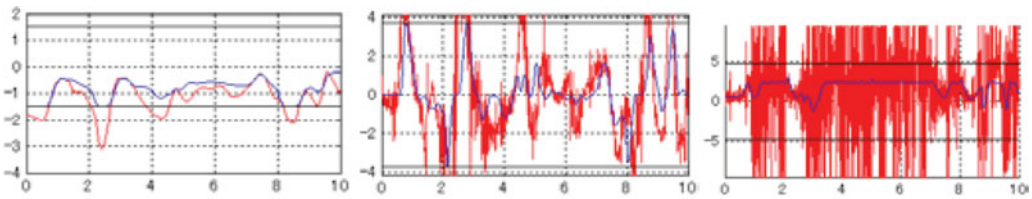


Fig. 6. Application of constraints simultaneously with strict torque limit: (a) angle [+/−1.5184], (b) velocity [+/−3.74129], and (c) torque half of [+/−4.8112]. Red, blue, and black lines represent original data, generated data, and limit lines, respectively. *X*-axis is time and *Y*-axis is each value specified in each subfigure. Angle is in radians, velocity is in radian/s, and torque is in kg m^2/s^2 .

in Fig. 6. Some small spikes presented in the torque domain are a little higher than the set limit due to such linearization, which is not a problem since the characteristic of the actuator usually allows instantaneous torque.

Compared to the optimization result in the next section, the filtering approach gives a more precise trajectory since there is no problem due to the optimization process. However, the disadvantage of this approach is that multiple objectives may not be realized, such as when an end-effector position cannot be corrected to a hand position. In that case, the optimization method is required.

4. Optimization Approach

Optimization is used, for example, to preserve the characteristics of motion by imposing objective functions, and then to transform the motion to the capabilities of the humanoid robot by constraints. Using a B-spline, instead of optimizing a joint angle directly, control points will be altered so the function could represent the trajectory based on objective functions and constraints.

Traditionally, using an objective function to reduce physical characteristics, as described in ref. [5] or [29], is inadequate since it does not apply specific limits to those values. Limit violation can occur, requiring a user to manually adjust trajectories.

Hence in this study, physical attributes are limited using constraints, as shown in Section 2, while an objective function is used to preserve the essence of original input motion and will be explained in this section.

After performing optimization to the B-spline curve, though an objective function is optimized and constraints are checked, the curve may still not be as precise as expected. In that case, optimization is done hierarchically, first using a small number of control points to represent a curve. Then if the precision is not satisfied, the original set of control points is decomposed to a larger number of control points and re-optimized. Prior to decomposition, two methods are used to detect the problem of the present hierarchy curve—knot density approximation[13] (which will be explained here) and error detection (11).

4.1. Optimization

Based on the B-spline function, optimizing the joint angles is usually enough for a robot to represent human motion, so an acceptable objective function is

$$f(P) = \sum_{n=1}^{N} ||Q(n, P) - \Theta(n)||_2, \tag{12}$$

where P is the vector of all angle control points, N is the sampling length of a trajectory, Q is the vector of all angles calculated from the B-spline function, and Θ is the vector of all angles derived from inverse kinematics.

The markers attached to the body of the human via an elastic suit may be not balanced, or they may slip during the capture process, so some of the sequences of angles computed by inverse

https://doi.org/10.1017/S0263574714000496 Published online by Cambridge University Press

kinematics may have large errors. Therefore, another objective function that considers both angles and end-effector is used for such motions:

$$f(P) = \sum_{n=1}^{N} \{||Q(n, P) - \Theta(n)||_2 + w||\text{F.K.}(Q) - Pos(n)||_2\}, \tag{13}$$

where w is weight selected from experiment, F.K. is forward kinematics, and *Pos* is the hand position of the trainer. The weight w is derived from experiment and fixed for each trainer. If a trainer's arm length is substantially different from that of the robot, *Pos* can be scaled to match the robot size.

Adding more objective functions is possible, but the weighting of each parameter in the function must be taken into account.

Constraints can be applied very much like the filtering approach, except that there is no redundancy control point at the end of each curve, as in Section 3. Also, the criterion for a fixed- or open-end is the same with the filtering approach.

Note that the global optimization program described in ref. [30] was used in this research with a small adaptation so that many constraints could be used.

4.2. Motion refinement

For many reasons, optimized trajectories may contain errors caused by such factors as the fact that the number of control points is not appropriate, or under- or over-optimization occurs, etc. Fortunately, the B-spline function allows a hierarchical structure so that a trajectory can be refined, that is, be re-optimized, using a larger number of B-spline bases by the process called knot insertion. Prior to knot insertion, there must be criteria to decide whether or not the present hierarchy's B-spline curve needs to be re-optimized. Two methods are used here—knot density approximation[13] and error detection.

First, the convergence could be made faster with less error by assigning an appropriate hierarchy to each part of a trajectory using a knot density detector, explained as follows. B-spline density problems comprise two issues. First, a B-spline has the limitation that it cannot represent a curve that swings more frequently than some certain value. Second, a too-dense B-spline can result in an over-optimized curve that swings more frequently than the original curve. A B-spline has a characteristic that its knot period of T can represent data with a peak-to-peak range as small as T itself. Hence, detecting the peak-to-peak distance of each original joint angle and piecewise-assignment of an appropriate knot density would help in solving density problems.

In order to locate the local peak value in an angle, a zero crossing point in the velocity domain should first be detected. However, the original curve's velocity has spikes of noise, so a frequency-based filter is used for smoothing. It is important to consider the characteristic of the filter to be used, especially the group delay, that is, the delay of each frequency component of the output signal after passing the filter:

$$\text{Group Delay}(\omega) = \frac{\text{d}}{\text{d}\omega}[\arg H(e^{j\omega})], \tag{14}$$

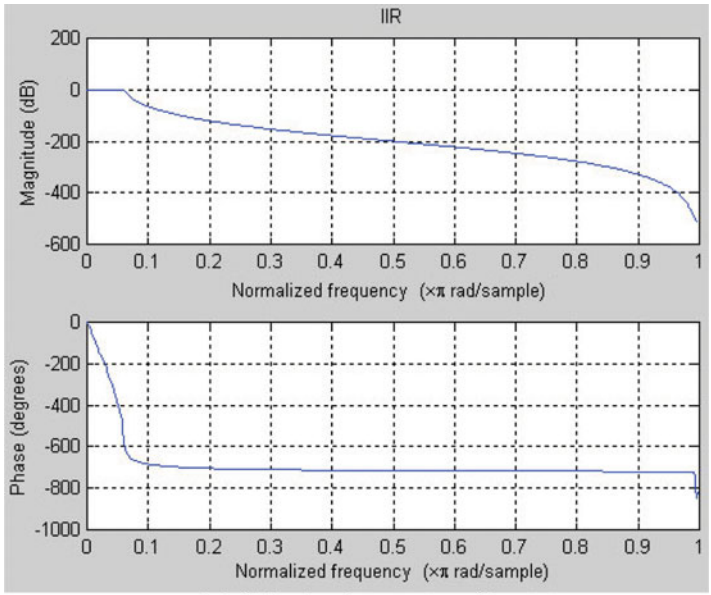
where $H(e^{j\omega})$ is the transfer function of the filter.

On this basis, a filter with a constant group delay or linear phase would produce an output with all the frequency components going out at the same time, and this is required for the positioning problem. Thus, an infinite impulse response (IIR) filter is inappropriate because of its frequency-varying group delay, as its phase is not linear. Hence, the filter of choice would be a finite impulse response (FIR) filter which has constant group delay, as its phase is linear. Though normally an FIR requires a higher order than an IIR, in zero crossing detection, the cut-off need not be sharp. Consequently, a low-order FIR filter that has a small delay can be selected. The filtering result from these two filters is shown in Fig. 7. It can also be seen that the FIR has a smaller delay than the IIR, which is desirable since there would be less data loss at the end of the trajectory.

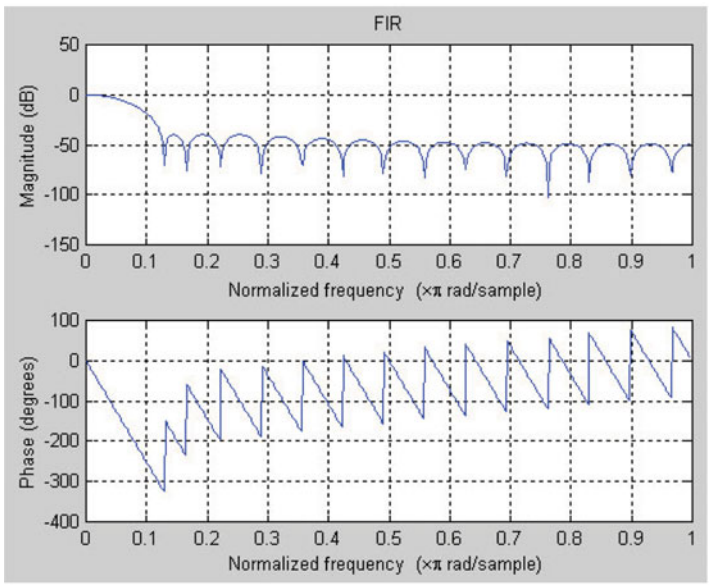
Then, the actual zero crossing can be calculated by

$$\text{Actual Position} = \text{Observed Position} - \text{Group Delay}. \tag{15}$$

https://doi.org/10.1017/S0263574714000496 Published online by Cambridge University Press



a) Infinite impulse response filter



b) Finite impulse response filter

Fig. 7. IIR and FIR filter magnitude and phase.

In the present hierarchy, if two zero crossing points of velocity are detected proactively within the range below the present knot period, which implies a swing pattern, that part of the curve needs a greater number of control points, and it would be marked to be re-optimized in the next hierarchy.

Furthermore, even if the density of a control point is appropriate, error can occur for reasons such as the optimization process does not fully converge, which often happens because the stop criterion of the optimization process is set before convergence is met. Hence, along with knot density approximation, a traditional error detector (11) is applied.

Now, if a density problem or error is detected, that set of B-spline coefficients must be fed into a hierarchical B-spline decomposition to generate a new B-spline that has a greater number of control points.

$$[\mathrm{hp}] = (\mathrm{hp}_0, \mathrm{hp}_1, \ldots, \mathrm{hp}_{\mathrm{hcp}}) = \mathrm{Hie}(p_0, p_1, \ldots, p_{\mathrm{cp}}), \tag{16}$$

https://doi.org/10.1017/S0263574714000496 Published online by Cambridge University Press

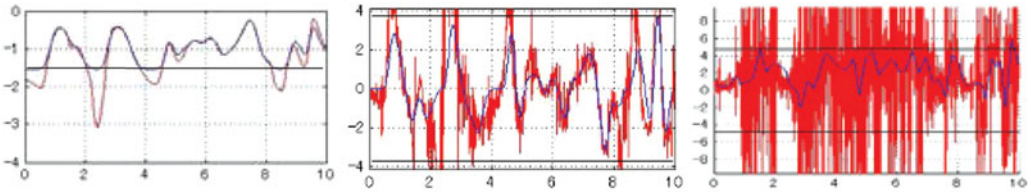


Fig. 8. Optimization result: (a) angle [+/−1.5184], (b) velocity [+/−3.74129], and (c) torque [+/−4.8112]. Red, blue, and black lines represent original data, generated data, and limit lines, respectively. *X*-axis is time and *Y*-axis is each value specified in each subfigure. Angle is in radians, velocity is in radian/s, and torque is in kg m^2/s^2 .

where [hp] is a set of hcp + 1 hierarchical control points hp and hcp > cp, Hie denotes the decomposition function.

The decomposition function can be easily implemented based on ref. [27] or [28] which show that the basis of a B-spline can be represented by five half-width bases with the coefficients:

$$h = \frac{1}{8}\{1, 4, 6, 4, 1\}. \tag{17}$$

Then decomposition can be performed by the following equation:

$$\mathrm{hp}_i = \sum_{j=0}^{i} h_{i-2j}\, p_j. \tag{18}$$

This new set of control points, at the period of the joint angle trajectory that has a problem, is optimized again to find a new optimum set of points for the B-spline (Fig. 8). The objective functions and constraints can be applied in a higher hierarchy without any adaptation required.

5. Real-Time Approach

As opposed to filtering or optimization approaches where the entire input trajectories could be derived *a priori* from a human trainer or from a robot's self-generated path, real-time motion generation focuses on how to tele-operate the robot with time delay,[31] plan its path in a dynamic environment where pertinent data are not available in advance,[32] or on force-feedback.[33] Attempts to limit physical characteristics rely on kinematics-based optimization to reduce values, rather than supplying specific limitations, as described in ref. [11].

The current research focuses on how to control the robot in real time using constraint functions. This has both advantages and drawbacks. An advantage is that since the trajectory is represented by a B-spline, the output trajectories' smoothness will always be ensured, as opposed to conventional methods that process each new time frame separately. However, since all the constraint functions operate on a B-spline parameter, new input time frames must be decomposed into the B-spline, and this introduces delay into output trajectories, which will be explained next.

Even though angle, collision, and velocity constraints can be used directly, let us consider limiting the torque constraint, which requires running forward and backward iteratively, and cannot be used in a real-time scheme. A method to limit torque in each time frame is proposed in this section.

5.1. Real-time decomposition from raw data

In a manner much like the decomposition technique presented in Section 2, new time-frame data are downsampled and their amplitude values are used as control point values. There are two interesting aspects different from offline processing. The first is how to decide the downsampling rate. Offline decomposition can choose the appropriate sampling rate from an error criterion (11). However, an online method does not know the data in advance. Hence, the sampling rate is subjectively set based on the trainer's capabilities. For example, in the case where original data are derived from a human

https://doi.org/10.1017/S0263574714000496 Published online by Cambridge University Press

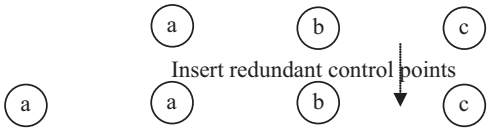


Fig. 9. Insertion of a redundant control point at the beginning (and, although not shown here, at the end) of a trajectory.

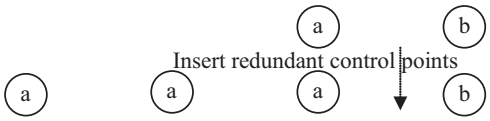


Fig. 10. Insertion of two redundant control points at the beginning of a trajectory.

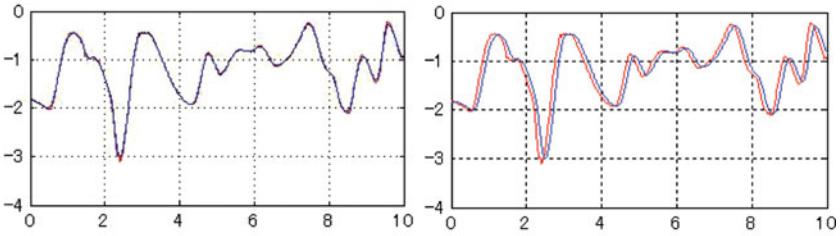


Fig. 11. Real-time decomposition from 200 fps to 20 control points per second. Red, blue, and black lines represent original data, generated data, and limit lines, respectively. *X*-axis is time and *Y*-axis is each value specified in each subfigure. Angle is in radians, velocity is in radian/s, and torque is in kg m^2/s^2 .

sampling rate of 30 control points per second, the same rate with a commercial video capture device would be appropriate. In our experiment, a sampling rate of 20 control points was set to compare with a filtering approach.

Second, it is important to consider what kind of model of decomposition would be appropriate. Formerly, in a filtering approach, inserting a redundant control point at the beginning and at the end of the trajectory was used. In a real-time approach, this introduces delay in processing. Figure 9(a) shows that it is necessary to wait for the third sampled control point, c, because a knot of B-spline consists of four control points. Hence, the processing delay introduced is a sampling rate lag. However, the original curve is well preserved, as can be seen in Fig. 11(a).

On the other hand, if the two redundant control points are placed only at the beginning of the curve as shown in Fig. 10, after the second control point, b, arrives and two redundant control points are added, filtering could start immediately.

It is not surprising that these inserts introduce a delay in the output trajectory as shown in Fig. 11(b). The delay, in this case, is not a processing delay as in the former decomposition method but is a physical delay that occurs in the output trajectory and must be avoided.

Hence, the notion of inserting a control point at the beginning and the end would be used, which is the same kind of decomposition that is used in the filtering approach.

5.2. Real-time limiting

After the B-spline curve is derived, most of the constraints in Section 2 can be applied to the B-spline curve directly, namely, angle, collision, and velocity. The result after passing through these filters is shown in Fig. 12.

However, for torque equation (8), changing the highest sensitivity control point affects the former period in an unwanted manner. This can be solved in an offline case using the iterative soft limits. Unfortunately, it cannot be applied in real time. Hence, if (8) is solved directly, the resulting torque trajectory is not well limited as shown in Fig. 13.

Thus, it is necessary to modify the torque constraint function. From (7), considering the term that multiplies M, instead of changing the highest sensitivity control point, p_1, choose to change the

https://doi.org/10.1017/S0263574714000496 Published online by Cambridge University Press

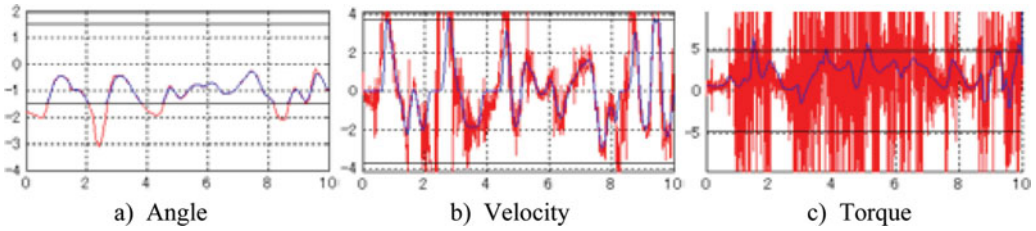


Fig. 12. On-the-fly application of angle and velocity constraints simultaneously: (a) angle [+/−1.5184], (b) velocity [+/−3.74129], and (c) torque. Red, blue, and black lines represent original data, generated data, and limit lines, respectively. *X*-axis is time and *Y*-axis is each value specified in each subfigure. Angle is in radians, velocity is in radian/s, and torque is in kg m^2/s^2 .

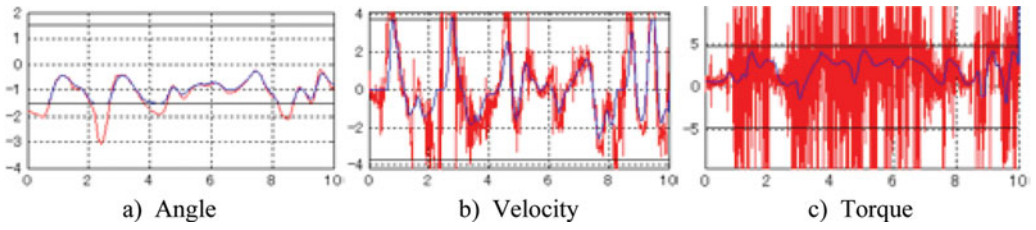


Fig. 13. On-the-fly application of angle, velocity, and torque constraints simultaneously: (a) angle [+/−1.5184], (b) velocity [+/−3.74129], and (c) torque half of [+/−4.8112] cannot be well limited. Red, blue, and black lines represent original data, generated data, and limit lines, respectively. *X*-axis is time and *Y*-axis is each value specified in each subfigure. Angle is in radians, velocity is in radian/s, and torque is in kg m^2/s^2 .

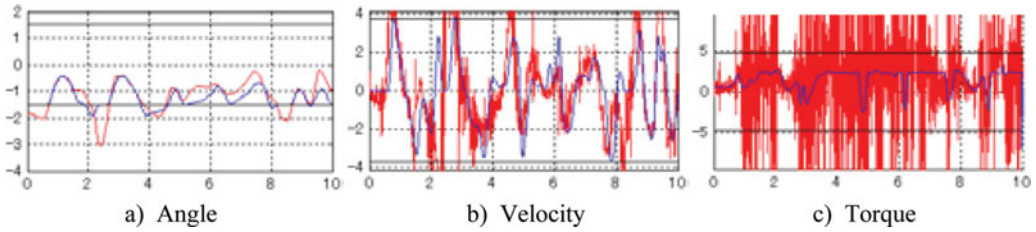


Fig. 14. On-the-fly application of angle, velocity, and real-time torque constraints simultaneously: (a) angle [+/−1.5184] is violated since real-time torque limiting, (b) velocity [+/−3.74129], and (c) torque half of [+/−4.8112]. Red, blue, and black lines represent original data, generated data, and limit lines, respectively. *X*-axis is time and *Y*-axis is each value specified in each subfigure. Angle is in radians, velocity is in radian/s, and torque is in kg m^2/s^2 .

control point that does not affect the torque of the previous period, which is p_2. Then, rearrange the function to isolate terms that contain p_2 from those that do not. As a result, (8) would be superseded by (19).

$$\tau_i = \sum_j \left[\frac{2M_{ij}}{T} + \sum_k C_{ijk} \left(\frac{p_2 - p_0}{2T} \right)_k \right] \left(\frac{p_2 - p_0}{2T} \right)_j + \frac{2I}{T} \left(\frac{p_2 - p_0}{2T} \right)_i + \sum_j M_{ij} \left(\frac{-2p_1 + 2p_0}{T^2} \right)_j + I \left(\frac{-2p_1 + 2p_0}{T^2} \right)_i + G_i. \quad (19)$$

Using this new constraint function, exactly the same way as with (8), torque can be well limited as shown in Fig. 14(c). Nevertheless, since p_2 is not the highest sensitivity control point, a considerable change in its value is needed, so that joint trajectory (or velocity trajectory) violates its limit, as can be seen in Fig. 14(a). This is the drawback of real-time torque constraint.

https://doi.org/10.1017/S0263574714000496 Published online by Cambridge University Press

6. Conclusion

The proposed physical constraints based on the B-spline function bring many benefits into motion generation, not only at the offline level, but also at the online level. This is possible by using our proposed B-spline-based equations, which, on the whole, are totally different from those for industrial robots.

Considering the use of the constraints offline, the proposed iterative soft constraint is one key to success in the comprehensive constraints. First, it makes torque limiting, which was not available before, possible. Second, it prevents conflict between the velocity and torque constraints when all constraints are used simultaneously. These are the main improvements over many previously mentioned works.

Then, for the offline filtering approach, a novel decomposition from raw data is shown to generate a B-spline curve that resembles the original trajectory with less error than existing methods. After that, all the proposed constraints can be used in filtering.

For offline optimization, the proposed density scheme and error in the trajectory are used as criteria before decomposing a B-spline to a larger number of control points that can be re-adapted.

For online consideration, the best way to decompose the B-spline is shown. However, it produces a delay equal to one sampling rate lag compared to existing methods. This is a minor drawback of our online scheme. The delay should be negligible if the drive rate of the robot is high enough. For example, many robots are set to drive at every few milliseconds.

The main factors that could deter progress toward a generalization of using B-spline-based constraints are the problems of:

1. Online collision constraint: It is not comprehensive enough to deal with multiple conflicting collision points at the same time.
2. Online torque constraint: This could violate other limits.

Our online method can ensure a smooth trajectory with angle and velocity limits. Hence, in online environments, it may be better to put torque in a cost function and to optimize the torque of all joints as a whole. Then, other methods can be used to deal with collision avoidance.

Acknowledgements

I am grateful to my senior, Dr. Shin-ichiro Nakaoka, and other staffs in National Institute of Advanced Industrial Science and Technology (AIST) who guided and helped me a lot in this project. I am also grateful to Prof. Katsushi Ikeuchi of the University of Tokyo for granting me such an interesting project like this.

References

1. N. S. Pollard, J. K. Hodgins, M. J. Riley and C. G. Atkeson, "Adapting Human Motion for the Control of a Humanoid Robot," *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, DC (May 11–15, 2002).
2. A. Safonova, N. S. Pollard and J. K. Hodgins, "Optimizing Human Motion for the Control of a Humanoid Robot," *Proceedings of the 2nd International Symposium on Adaptive Motion of Animals and Machines*, Kyoto, Japan (Mar. 4–8, 2003).
3. G. B. Hammam and D. E. Orin, "Whole-body humanoid control from upper-body task specifications," Patent Application Number: 20110066283, Publication date: Mar. 17, 2011.
4. J. Lee and S. Y. Shin, "A Hierarchical Approach to Interactive Motion Editing for Human-Like Figures," *Proceedings of ACM SIGGRAPH*, Los Angeles, CA (Aug. 8–9, 1999) pp. 39–48.
5. A. Ude, C. G. Atkeson and M. Riley, "Programming full body movements for humanoid robot by observation," *Robot. Auton. Syst.* **47**, 93–108 (2004).
6. J. Luo and K. Hauser, "Interactive Generation of Dynamically Feasible Robot Trajectories From Sketches using Temporal Mimicking," *Proceedings of the IEEE International Conference on Robotics and Automation*, St. Paul, Minnesota (May 14–18, 2012).
7. T. Moulard, E. Yoshida and S. Nakaoka, "Optimization-Based Motion Retargeting Integrating Spatial and Dynamic Constraints," *Proceedings of the 44th Symposium on Robotics*, Seoul, Korea (Oct. 24–26, 2013).
8. A. E. Khoury, F. Lamiraux and M. Taix, "Optimal Motion Planning for Humanoid Robots," *Proceedings of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany (May 6–10, 2013).

9. M. Riley, A. Ude, K. Wade and C. G. Atkeson, "Enabling Real-Time Full Body Imitation: A Natural Way of Transferring Human Movements to Humanoids," *Proceedings of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan (Sep. 14–19, 2003).
10. S. Y. Shin and C. H. Kim, "On-Line Human Motion Transition and Control for Humanoid Upper Body Manipulation," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan (Oct. 18–22, 2010).
11. A. D'Souza, S. Vijayakumar and S. Schaal, "Learning Inverse Kinematics," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, Hawaii (Oct. 29–Nov. 3, 2001).
12. T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev and E. Todorov, "An Integrated System for Real-Time Model-Predictive Control of Humanoid Robots," *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Atlanta, Georgia (Oct. 15–17, 2013).
13. M. Ruchanurucks, S. Nakaoka, S. Kudoh and K. Ikeuchi, "Generation of Humanoid Robot Motions with Physical Constraints using Hierarchical B-Spline," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Alberta, Canada (Aug. 2–6, 2005).
14. S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa and K. Ikeuchi, "Task Model of Lower Body Motion for a Biped Humanoid Robot to Imitate Human Dances," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Alberta, Canada (Aug. 2–6, 2005).
15. S. Nakaoka and T. Komura, "Interaction Mesh Based Motion Adaptation for Biped Humanoid Robots," *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Osaka, Japan (Nov. 29–Dec. 1, 2012).
16. S. Dalibard, A. Nakhaei, F. Lamiraux and J.-P. Laumond, "Whole-Body Task Planning for a Humanoid Robot: A Way to Integrate Collision Avoidance," *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Paris, France (Dec. 7–10, 2009).
17. F. Seto, K. Kosuge and Y. Hirata, "Self-Collision Avoidance Motion Control for Human Robot Cooperation System Using RoBE," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Alberta, Canada (Aug. 2–6, 2005).
18. L. Zlajpah and B. Nemec, "Kinematic Control Algorithms for On-Line Obstacle Avoidance for Redundant Manipulators," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland (Sep. 30–Oct. 4, 2002).
19. M. Ruchanurucks, S. Nakaoka, S. Kudoh and K. Ikeuchi, "Humanoid Robot Motion Generation with Sequential Physical Constraints," *Proceedings of the IEEE International Conference on Robotics and Automation*, Orlando, Florida (May 15–19, 2006).
20. W. Korb and I. Troch, "Data reduction for manipulator path planning," *Robotica* **21**, 605–614 (2003).
21. M. Ferch, J. Zhang and A. Knoll, "Robot Skill Transfer Based on B-Spline Fuzzy Controllers for Force-Control Tasks," *Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, Michigan (May 10–15, 1999).
22. E. Cohen, I. Daubechies and J. C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Commun. Pure Appl. Math.* **45**, 485–560 (1992).
23. S. J. Gotler, "Hierarchical and Variational Geometric Modeling with Wavelets," *Proceedings of the ACM Symposium on Interactive 3D Graphics*, Monterey, USA (Apr. 9–12, 1995) pp. 35–42.
24. A. Ude, C. G. Atkeson and M. Riley, "Planning of Joint Trajectories for Humanoid Robots using B-Spline Wavelets," *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA (Apr. 24–28, 2000).
25. K. G. Shin and N. D. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Trans. Autom. Control* **30**(6), 531–541 (1985).
26. J.-J. E. Slotine and H. S. Yang, "Improving the efficiency of time-optimal path-following algorithms," *IEEE Trans. Robot. Autom.* **5**(1), 118–124 (1989).
27. C. K. Chui and E. Quak, "An Introduction to Wavelets," **In**: *Wavelet Analysis and its Application*, vol. 1 (C. K. Chui, ed.) (Academic Press, Waltham, MA, 1992).
28. D. Forsey and R. Bartels, "Hierarchical B-Spline Refinement," *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques* (1988).
29. C. Rose, B. Guenter, B. Bodenheimer and M. F. Cohen, "Efficient Generation of Motion Transitions Using Space-Time Constraints," *Proceedings of ACM SIGGRAPH*, New York (1996).
30. W. Huyer and A. Neumaier, "Global optimization by multilevel coordinate search," *J. Glo. Optim.* **14**, 331–355 (1999).
31. D. Lee and M. W. Spong, "Passive Bilateral Teleoperation with Constant Time Delays," *Proceedings of the IEEE International Conference on Robotics and Automation*, New Orleans, LA (Aug. 4–9, 1996).
32. T.-Y. Li and J.-C. Latombe, "On-Line Manipulation Planning for Two Robot Arms in a Dynamic Environment," *Proceedings of the IEEE International Conference on Robotics and Automation*, Nagoya, Japan (May 21–27, 1995).
33. M. Mahvash and A. M. Okumura, "Friction Compensation for a Force-Feedback Telerobotic System," *Proceedings of the IEEE International Conference on Robotics and Automation*, Orlando, Florida (May 15–19, 2006).