# Spacetime Sweeping: An Interactive Dynamic Constraints Solver

Seyoon Tak Oh-young Song Hyeong-Seok Ko

Graphics & Media Lab.
Seoul National University
{tak,tsitfel,ko}@graphics.snu.ac.kr

## Abstract

*This paper presents a new method for editing an existing motion to satisfy a set of user-specified constraints, and in doing so guaranteeing the kinematic and dynamic soundness of the transformed motion. We cast the motion editing problem as a constrained state estimation problem based on the per-frame Kalman filter framework. To handle various kinds of kinematic and dynamic constraints in a scalable fashion, we develop a new algorithm called spacetime sweeping, which sweeps through the frames with two consecutive filters. The unscented Kalman (UK) filter estimates an optimal pose for the current frame that conforms to the given constraints, and feeds the result to the least-squares (LS) filter. Then, the LS filter resolves the inter-frame inconsistencies introduced by the UK filter due to the independent handling of the position, velocity, and acceleration. The per-frame approach of the spacetime sweeping provides a surprising performance gain. It is remarkable that now editing of motion that involves dynamic constraints such as dynamic balancing can be done interactively.*

## 1 Introduction

Since motion capture became a commonplace technique for generating realistic human-like animation, motion editing for its reuse has become an important problem in computer animation research. Almost always, an original captured motion needs to be modified even slightly to make the character coherently interact with other characters or the environment, or to meet various other needs of the animators. Researchers labelled their motion editing techniques with some different terms such as motion retargeting, motion adaptation, motion transformation, or motion filtering.

In the past few years, a variety of *constraint-based* motion editing techniques [7, 8, 14, 2, 18] have been proposed. Constraints are the mathematical formulation of the features to be preserved in a motion. Actually, all of our everyday motion are conducted under a number of constraints. To run a constraint-based motion editing algorithm, therefore, animators have to specify the kinematic and dynamic constraints involved in the desired motion. For example, a plausible limbo motion calls for a set of constraints illustrated in Figure 1. Under the given constraints, the above methods optimize an objective function parameterized in space and time.

In most of the above constraint-based methods, the constraints consist of pure kinematic constraints such as the desired position or trajectory of end-effectors. It is interesting to note that the methods based on kinematic constraints can quite effectively generate useful variations of the original motion. Several noticeable results are reported in [7, 8, 14, 2]. But in some cases, the motion generated from kinematic editing is not acceptable when the motion is viewed in a dynamics context. For example, when we retarget the motion of an adult to a child, especially when he is carrying or swinging a heavy object, the kinematic modification of the original motion can not exhibit the different handling of the situation of the child.

Notably, Popović and Witkin [18] took the first step to include dynamics for editing captured motion. Their formulation is a constrained optimization problem, which has dynamic constraints as well as kinematic constraints, to enforce physical correctness. To make their optimization problem tractable, they devised a sophisticated character simplification scheme. Nevertheless, the reduced problem still remains in the realm of so-called spacetime optimiza-

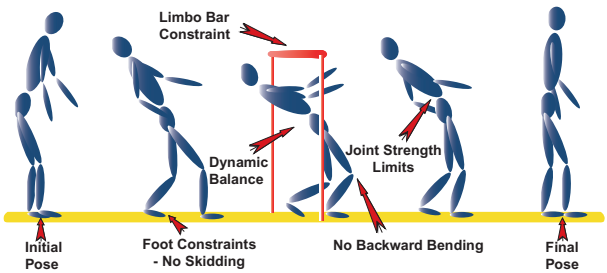



**Figure 1.** Constraints in a limbo motion

tion. Spacetime constraints [30, 3] is a well established principle to generate realistic, dynamic motion by solving an optimization problem. However, spacetime optimization is computationally expensive, which prohibits its use in practical animation production.

In this paper, we present a new scalable method to realize spacetime constraints. We call it *spacetime sweeping*, which is a contrary concept to spacetime optimization. Instead of optimizing a single scalar value over the entire duration of motion, spacetime sweeping tries to find the optimum value at every frame in per-frame basis so that it can run in real-time or at an interactive speed. Since spacetime optimization cannot guarantee the global optimum, if spacetime sweeping can find solutions of reasonable quality, its value for practical use can be quite considerable. However, there remains an important challenge in formulating the spacetime constraints into a spacetime sweeping problem: the dynamic constraints involve the velocity, acceleration, or other quantities which prohibit their per-frame basis handling. We present a double-filter technique to overcome such a hurdle in the subsequent sections.

Spacetime sweeping works as an enhancement operator. The first application of the sweeping may not produce completely correct results. For example, there may be a slight dynamic imbalance or skidding in the supporting foot. However, further sweeping can be applied to the above result, and every application produces improved results in the kinematic or dynamic accuracy. Such incremental behavior can be valuable in animation productions. Instead of obtaining the final motion after a few hours, animators prefer to see the rough outline of the motion interactively, and if they like it they can improve the quality afterwards. The above description is just to manifest the enhancing nature of the sweeping algorithm. In actual experiments, it generated a quite accurate result on the first application, and the result quickly converged by later applications.

To achieve the above features, we cast the motion editing problem as a constrained state estimation problem based on the per-frame Kalman filter framework. The Kalman filter optimally estimates the state of a system through two steps: prediction and correction. In our problem, we (1) predict motion parameters at each frame referring to the original motion data and, then (2) correct the predicted result to meet a set of user-specified constraints.

The rest of this paper is organized as follows: Section 2 reviews the related work, Section 3 introduces how we apply the Kalman filter to our problem, Section 4 explains the dynamic constraints (balance and torque constraints), Section 5 presents the double-filter method for dynamic constraint, Section 6 reports several experimental results, and finally, Section 7 concludes the paper.

# 2 Related Work

## 2.1 Motion Editing and Retargeting

Bruderlin and Williams [1] regarded a motion as time-varying signals, and applied signal processing techniques to these signals. Witkin and Popović [31] introduced a motion warping technique. Gleicher [8] introduced a method for retargeting a captured motion into different characters. Lee and Shin [14] proposed multi-resolution approach in dealing with motion signals by using a hierarchical curve fitting technique. Choi and Ko [2] proposed an online motion retargeting technique using inverse rate control based on Jacobian inverses. Shin et al. [20] also presented an online technique based on the notion of dynamic importance of end-effectors. Recently, Gleicher wrote a survey article that compares a variety of motion editing methods [9].

## 2.2 Physically Based Spacetime Methods

Originally, the spacetime constraints approach was proposed by Witkin and Kass [30], and Cohen [3] developed a variation of the approach that uses spacetime windows. They cast motion synthesis as a constrained optimization problem for producing optimal motion that satisfies user-specified constraints. In order to derive physically valid motions, they enforced Newton's law as the constraint and optimized the energy function. Recently, Popović and Witkin [18] proposed a physically based spacetime method for transforming a capture motion by using a character simplification technique.

## 2.3 Kalman Filter

The Kalman filter has been one of the most widely used methods for the estimation of dynamic systems, due to its simplicity, optimality, tractability, and robustness [12, 16]. The original Kalman filter estimates the state of a linear system from the measurements available online in such a way that the error is minimized in the least-squares sense. However, virtually every nontrivial real world applications are nonlinear, so the *extended Kalman filter* (EKF) [27, 28] approximates nonlinear transformations with linear ones (Jacobians). The EKF has been very popular in a number of estimation and control applications for decades. Some researchers [11, 26, 24], however, very recently pointed out the fundamental flaws of the EKF which mainly result from the use of linear approximations, and proposed the *unscented Kalman filter* (UKF). Our spacetime sweeping algorithm is based on the UKF.

## 3 Motion Editing in the Kalman Filter Framework

In this section we explain how the constraint-based motion editing problem is formulated into the Kalman filter framework. We begin with introducing the unscented Kalman filter (UKF).

### 3.1 Unscented Kalman Filter

The basic framework of the EKF involves estimation of the state of a nonlinear dynamic system, which consists of the *process model* $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{v}_k)$ and the *measurement model* $\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k)$, where $\mathbf{x}_k$ is the state of the system, $\mathbf{z}_k$ is the measurement, and the random variables $\mathbf{v}_k$ and $\mathbf{n}_k$ represent the process and measurement noise. The EKF estimates recursively the mean and covariance of the state through the following *predictor-corrector* algorithm. Assuming $\mathbf{v}_k$ and $\mathbf{n}_k$ have zero-mean noise,

**Predict** (time update)

$$\hat{\mathbf{x}}_k^- = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, 0) \tag{1}$$

**Correct** (measurement update)

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_k^-, 0)) \tag{2}$$

The time update equation is responsible for predicting the *a priori* estimate $\hat{\mathbf{x}}_k^-$ of the next state. The measurement update equation is responsible for correcting $\hat{\mathbf{x}}_k^-$ by reflecting the new measurement $\mathbf{z}_k$ to obtain the *a posteriori* estimate $\hat{\mathbf{x}}_k$. $\mathbf{K}$ is the Kalman gain that is designed to minimize the measurement error.

The fundamental limitation of the EKF comes from its linearization approach (Jacobian) for calculating mean and covariance. It reduces accuracy and causes the convergence problem when the nonlinearity is severe. Also, the method requires calculation of the Jacobian matrix which is difficult to implement.

Julier and Uhlmann [11] proposed the UKF as a derivative-free alternative to the EKF. The basic mechanism of the UKF is same as the one described above by Equations (1) and (2). But the UKF performs state estimation by *approximating the probability distribution undergoing the nonlinear function*, rather than approximating the nonlinearity itself as in the EKF. For this, the UKF utilizes the *unscented transformation* (UT), a deterministic sampling approach to calculate the statistics of a random variable which undergoes a nonlinear transformation. For further discussion, we first elaborate on the UT.

Consider an $n_x$ dimensional random variable $\mathbf{x}$ having the mean $\hat{\mathbf{x}}$ and covariance $\mathbf{P}_x$, and suppose that it propagates through an arbitrary nonlinear function $\mathbf{g} : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}$ to generate $\mathbf{y} = \mathbf{g}(\mathbf{x})$ (Figure 2(a)). We choose a set

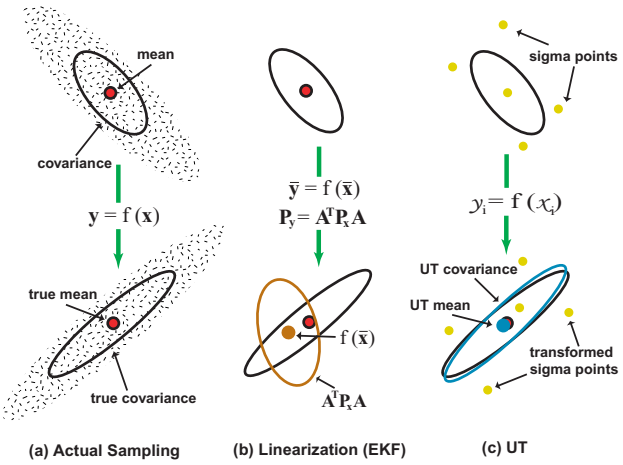



**Figure 2.** Mean and covariance propagation: (a)the true mean and covariance using real sampling, (b)EKF's linearization approach, (c)UT (5 sigma points used in 2D cases.)

of $2n_x + 1$ weighted samples $\mathcal{X}_i$ (*sigma points*) deterministically so that they completely represent the true mean and covariance of random variable $\mathbf{x}$ (Figure 2(c)).

$$\begin{array}{lll} \mathcal{X}_0 = \hat{\mathbf{x}} & W_0 = \kappa/(n_x + \kappa) & i = 0 \\ \mathcal{X}_i = \hat{\mathbf{x}} + (\sqrt{(n_x + \kappa)\mathbf{P}_x})_i & W_i = 1/\{2(n_x + \kappa)\} & i = 1, ..., n_x \\ \mathcal{X}_i = \hat{\mathbf{x}} - (\sqrt{(n_x + \kappa)\mathbf{P}_x})_i & W_i = 1/\{2(n_x + \kappa)\} & i = n_x + 1, ..., 2n_x \end{array} \tag{3}$$

where $\kappa$ is a scaling parameter, $(\sqrt{(n_x + k)\mathbf{P}_x})_i$ is the $i$th row or column of the matrix square root of $(n_x + k)\mathbf{P}_x$, and $W_i$ is the weight associated with the $i$-th sigma point so that $\sum_{i=0}^{2n_x} W_i = 1$. Now each sigma point is propagated through the nonlinear function to yield a set of transformed sigma points,

$$\mathcal{Y}_i = \mathbf{g}(\mathcal{X}_i) \qquad i = 0, \ldots, 2n_x \tag{4}$$

The mean and covariance of $\mathbf{y}$ are approximated by a weighted average mean and covariance of the transformed sigma points.

$$\begin{array}{rcl} \hat{\mathbf{y}} & = & \sum_{i=0}^{2n_x} W_i \mathcal{Y}_i \\ \mathbf{P}_y & = & \sum_{i=0}^{2n_x} W_i (\mathcal{Y}_i - \hat{\mathbf{y}})(\mathcal{Y}_i - \hat{\mathbf{y}})^T \end{array} \tag{5}$$

As illustrated in Figure 2, compared to the EKF's linear approximation, the UT is accurate to the second order for any nonlinearity function.

The UKF is an extension of the UT to the Kalman filter framework. The following box (Algorithm 1) compactly shows the standard UKF algorithm and its related equations. More comprehensive discussion about the UKF can be found in [11, 26].

Initialize the state mean and covariance
For $k = 1,\ldots,\infty$

1. Calculate sigma points $\mathcal{X}_{k-1}$ using Equation (3)

2. Predict (time update):

$$\begin{aligned}
\mathcal{X}_{k|k-1} &= \mathbf{F}\,[\mathcal{X}_{k-1}] \\
\hat{\mathbf{x}}_k^- &= \textstyle\sum_{i=0}^{2n_x} W_i \mathcal{X}_{i,k|k-1} \\
\mathbf{P}_k^- &= \textstyle\sum_{i=0}^{2n_x} W_i [\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-][\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-]^T + \mathbf{V}_k
\end{aligned}$$

3. Correct (measurement update):

$$\begin{aligned}
\mathcal{Y}_{k|k-1} &= \mathbf{H}\,[\mathcal{X}_{k|k-1}] \\
\hat{\mathbf{y}}_k^- &= \textstyle\sum_{i=0}^{2n_x} W_i \mathcal{Y}_{i,k|k-1} \\
\mathbf{P}_{\mathbf{y}_k\mathbf{y}_k} &= \textstyle\sum_{i=0}^{2n_x} W_i [\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^T + \mathbf{N}_k \\
\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} &= \textstyle\sum_{i=0}^{2n_x} W_i [\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^T \\
\mathbf{K}_k &= \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}\mathbf{P}_{\mathbf{y}_k\mathbf{y}_k}^{-1} \\
\hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\mathbf{P}_{\mathbf{y}_k\mathbf{y}_k}\mathbf{K}_k^T
\end{aligned}$$

$W_i$ = sigma point weights, $\mathbf{P}_k^-, \mathbf{P}_k$ = state error covariance, $\mathbf{V}_k$ = process noise covariance, and $\mathbf{N}_k$ = measurement noise covariance.

**Algorithm 1.** The standard UKF algorithm

It is reported that (1) UKF yields more accurate estimations than EKF in most applications, since it guarantees the second order accuracy for any nonlinearity; (2) UKF is fast since it scales same as the linear Kalman filter; (3) UKF is easier to implement than EKF since it does not bear any linearization (Jacobian) [11, 26, 24]. For these reasons, in this work we adopt the UKF as the numerical core for our motion editing engine.

### 3.2 Formulating Motion Editing with UKF

The algorithm we develop in this paper is a constraint-based motion editing, which transforms a given motion sequence to a new version so that the resulting motion satisfies the given constraints. In this section, we show how our motion editing problem is formulated in the Kalman filter framework. First, we focus only on the kinematic constraints.

We parameterize a motion by $\mathbf{x}(t) = [\mathbf{p}(t), \mathbf{q}^0(t), ..., \mathbf{q}^j(t)]$, where $\mathbf{p}(t)$ and $\mathbf{q}^0(t)$ are the position and orientation of the root, and $\mathbf{q}^1(t), ..., \mathbf{q}^j(t)$ are rotations at the joints. In dealing with orientations and rotations, we carry both quaternions and exponential maps together. It has been reported that unit quaternions and exponential maps are numerically well-conditioned, and do not create gimbal-lock singularities [21, 10].

Figure 3 shows the UKF's predictor-corrector framework for constraint-based motion editing. At each frame, we predict the pose $\mathbf{x}^-(t_k)$ for the new motion referring to the original motion data, then correct the pose to $\mathbf{x}(t_k)$ in order to meet the desired constraints. The same procedure is repeated for each frame while sweeping through the frames.

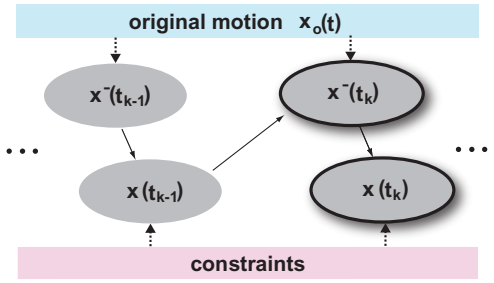



**Figure 3.** Prediction and correction scheme for constraint-based motion editing

**Motion Prediction:** We design the process model for motion prediction so that the prediction is based on the original motion data. We can use a simple first-order position-velocity model or a more complex model using forward dynamics of the body. But we already have the original motion $\mathbf{x}_{\mathrm{o}}(t)$, which is a good goal the new motion has to resemble. So, in this work, we use the original motion directly for the process model $\mathbf{f}(\cdot)$, that is

$$\mathbf{x}^-(t_k) = \mathbf{x}_{\mathrm{o}}(t_k). \tag{6}$$

Unlike the standard UKF algorithm in Algorithm 1, we do not propagate the state error covariances through frames, instead we provide $\mathbf{P}_k^-$ as a user-controllable parameter, from which we calculate the sigma points for the next correction step.

**Motion Correction:** To satisfy the given constraints, we treat the constraints as perfect measurements in the correction step. In the kinematic motion editing, inverse kinematics (IK) is a very powerful and essential technique. IK constraints can be represented as $\mathbf{h}_{\mathrm{IK}}(\mathbf{x}) = \mathbf{z}$, where $\mathbf{x}(t)$ is the motion parameter, $\mathbf{z}(t)$ is the desired paths of designated body points, and $\mathbf{h}_{\mathrm{IK}}(\cdot)$ is formulated by recursive forward kinematics.

To reflect a set of given constraints, we correct the prediction $\mathbf{x}^-(t_k)$,

$$\mathbf{x}(t_k) = \mathbf{x}^-(t_k) + \mathbf{K}_k[\mathbf{z}(t_k) - \mathbf{h}_{\mathrm{IK}}(\mathbf{x}^-(t_k))]. \tag{7}$$

The Kalman gain $\mathbf{K}_k$ transforms the error in the measurement space to the motion update that needs to be added to the result of the prediction. It is computed by the UKF equations described in Algorithm 1.

There exist two user-controllable parameters, which significantly affect the details of the resulting motion.

State error covariance matrix $\mathbf{P}_k^-$: The state error covariance means a degree of uncertainty, and so it affects the

amount of the displacement of each DOF in the correction step. A large state error covariance results in a large displacement of the corresponding joint.

Measurement noise covariance matrix $\mathbf{N}_k$: The measurement noise covariance is interpreted as rigidity of constraints. Typically, this parameter is set to zero treating the constraints as perfect measurements. It is especially useful when two constraints conflict to each other. The one with a larger covariance (soft constraint) yields to the other with a smaller covariance (hard constraint).

According to our experiments, the above UKF based method produces surprisingly accurate results compared to the EKF based methods. When the retargeting task involves only kinematic constraints, the above method could produce kinematically retargeted result in realtime without potential discontinuities due to the per-frame approach. This is demonstrated in Section 6.

## 4 Dynamic Constraints

For a motion to be plausible in a dynamic sense, the motion should be (1) dynamically balanced and (2) comfortable. In this section, we introduce two dynamic constraints: balance constraints and torque constraints.

### 4.1 Balance Constraints

Human is a two-legged creature, and thus balance is an important factor for judging the realism of motion. In a static posture, balance is easily accomplished by confining the center of mass of the character to the supporting area. In a moving character, however, balancing involves more complicated process.

The *zero moment point* (ZMP) has been widely used in robotics to deal with the dynamic balance problem in biped robots [25, 6, 17, 5]. Tak et al. [23] introduced an optimization-based balancing algorithm, which modifies a given motion into a dynamically balanced one based on the ZMP concept. According to [23], the dynamic balance constraint can be stated in terms of ZMP as: *a motion is balanced when and only when the ZMP trajectory is kept within the supporting area.* Note that this ZMP-based balance purely depends on physical validness. In a real world, the ZMP always exists within the supporting area, even when the body is falling down at the moment. In a graphical world, however, the ZMP of a motion can go out of the supporting area, which means the motion is physically incorrect. In this paper, balancing means correcting physically incorrect motion in a balance context into the correct one while preserving the original motion as much as possible. More comprehensive discussions on the ZMP concept and its relation to dynamic balancing can be found in [23].

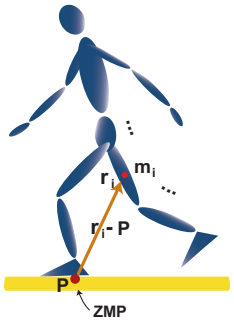



**Figure 4.** The ZMP during a motion

The ZMP can be derived from the body dynamics of the character. Since the sum of the moment contributions made from each body segments should be zero at ZMP (Figure 4), we can obtain the ZMP by solving the following equation for $\mathbf{P}$

$$\sum_i (\mathbf{r}_i - \mathbf{P}) \times \{m_i(-\ddot{\mathbf{r}}_i + \mathbf{g}) + \mathbf{f}_i\} = 0, \qquad (8)$$

where $m_i$ is the mass of link $i$, $\mathbf{r}_i$ is the center of mass of link $i$, $\mathbf{f}_i$ is the external forces applied on link $i$, and $\mathbf{g}$ is the acceleration of gravity.

The balancing algorithm involves three steps: (1) We calculate the ZMP trajectory of the original motion using the above ZMP equation, and find out the portions in which the motion is not balanced, then (2) if there are some portions of ZMP trajectory that go outside the supporting area, we project them into the area. This is done by an off-line job. (3) We finally correct the motion so that it satisfies the new ZMP trajectory constraint. Since $\mathbf{r}_i$ and $\ddot{\mathbf{r}}_i$ of Equation (8) is parameterized with the character's DOFs and their derivatives, we can formulate the balance constraint as $\mathbf{h}_{\text{balance}}(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}) = \mathbf{z}$, where $\mathbf{z}$ is the desired ZMP trajectory.

### 4.2 Torque Constraints

The torque a human body can exert at a joint is limited. Therefore a motion violating these torque limits may look unrealistic or uncomfortable. [15, 13] have introduced methods to generate more comfortable version of the original motion by directly or interactively modifying the motion parameters so that the joint torque remains within the strength limits. We achieve the same goal by imposing the joint torque constraints in our dynamic constraints solver.

The desired torque limits can be obtained from biomechanics literature [29], or specified by animators for their own purposes. The torque constraints are formulated as $\mathbf{h}_{\text{torque}}(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}) = \mathbf{z}$, where $\mathbf{z}$ is the desired torque limits. The function $\mathbf{h}_{\text{torque}}(\cdot)$ relates the motion parameters

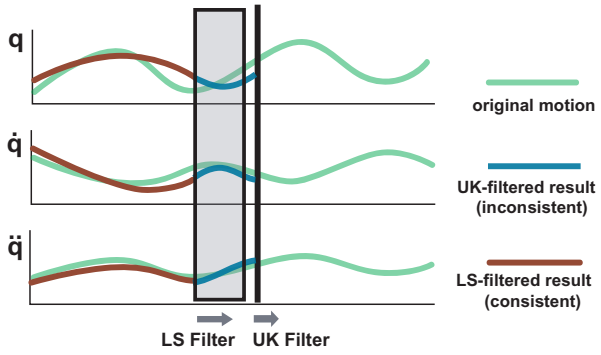



**Figure 5.** Double filter sweeping for dynamic constraints

to the joint torques, which is generally called inverse dynamics. In our work, to achieve maximum performance, we use the linear-time recursive Newton-Euler method [4, 19] in implementing the inverse dynamics.

# 5 Spacetime Sweeping for Dynamic Constraints

Given the motion sequence $\mathbf{x}(t)$, the motion parameters involved in the dynamic constraints (e.g. $\dot{\mathbf{x}}$,$\ddot{\mathbf{x}}$) have different nature from the ones in the kinematic constraints. To handle the dynamic constraints in the per-frame Kalman filter framework is much more challenging because they involve velocity and acceleration properties.

To realize this, we develop a *double-filter sweeping* method, which is the concatenation of two filters: (1) the UK filter and (2) the least-squares (LS) filter. The idea is that, first, the UK filter resolves the dynamic constraints manipulating $\mathbf{x}$, $\dot{\mathbf{x}}$, $\ddot{\mathbf{x}}$ as if they are independent DOFs, then the LS filter is applied to the inconsistent UK-filtered result to get the consistent version. The operation of the double-filter sweeping is illustrated in Figure 5.

## 5.1 UK Filter

We form the augmented motion $[\mathbf{x}_o\ \dot{\mathbf{x}}_o\ \ddot{\mathbf{x}}_o]$ from the original motion sequence $\mathbf{x}_o$. Then, the UK filter is applied to each of $\mathbf{x}_o$, $\dot{\mathbf{x}}_o$, and $\ddot{\mathbf{x}}_o$ independently, to produce the intermediate version of the motion. However, both $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ are direct functions of $\mathbf{x}$ in the time domain, which means that the stream of ($\mathbf{x}$,$\dot{\mathbf{x}}$,$\ddot{\mathbf{x}}$) from the UK filer may not constitute the valid tuples of position, velocity, and acceleration.

The UK filter used here is basically the same mechanism as described in Section 3.2 except that the state vector is now $[\mathbf{x}\ \dot{\mathbf{x}}\ \ddot{\mathbf{x}}]$ and the dynamic constraints appearing in the correction step (Equation (7)) are formulated in the form of $\mathbf{h}_{\text{balance}}(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}) = \mathbf{z}$ or $\mathbf{h}_{\text{torque}}(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}) = \mathbf{z}$.

## 5.2 LS Filter

In the above intermediate UK-filtered result, the position-velocity-acceleration relationship may have been corrupted. The mission of the LS filter is to rectify (in an optimal sense) the corrupted relationship. The LS filter is applied to the intermediate result to produce the double filtered motion ($\mathbf{x}$,$\dot{\mathbf{x}}$, $\ddot{\mathbf{x}}$). Figure 5 shows the application of the double filter to a single DOF ($\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$).

We accomplish the job of rectifying the corrupted relationship using the B-spline curve fitting technique. For each DOF ($\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$) of the UK-filtered result, we find the B-spline curve $\mathbf{C}$ that, together with $\dot{\mathbf{C}}$ and $\ddot{\mathbf{C}}$, best approximates ($\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$). It corresponds to finding the control points $\mathbf{c}$ that give the least square solution to the set of equations $\mathrm{B}\mathbf{c}=\mathbf{q}$, $\dot{\mathrm{B}}\mathbf{c}=\dot{\mathbf{q}}$, $\ddot{\mathrm{B}}\mathbf{c}=\ddot{\mathbf{q}}$, where B, $\dot{\mathrm{B}}$, $\ddot{\mathrm{B}}$ are the matrix of the B-spline curve basis function and its derivatives evaluated at corresponding time steps. We can combine them into the single equation $\mathbf{B}\mathbf{c} = \mathbf{d}$, where $\mathbf{B} = [\mathrm{B}\ \dot{\mathrm{B}}\ \ddot{\mathrm{B}}]^T$ and $\mathbf{d} = [\mathbf{q}\ \dot{\mathbf{q}}\ \ddot{\mathbf{q}}]^T$. This is an over-determined least squares problem. With the covariance matrix $\mathbf{V}$ included, we find $\mathbf{c}$ that minimizes

$$(\mathbf{B}\ \mathbf{c} - \mathbf{d})^T \mathbf{V}^{-1} (\mathbf{B}\ \mathbf{c} - \mathbf{d}), \tag{9}$$

where $\mathbf{V}$ is a diagonal matrix for scaling the sensitivity among the fitting data $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$. The classical linear algebra gives the solution to this problem [22]. Now, the discrete form of the final motion is obtained by evaluating the B-spline curves at all time steps.

The above curve fitting is done within the *sweeping window*, the local duration just swept by the the UK filter as shown in Figure 5. The user can specify the duration of a sweeping window. The wider window costs more computation time, but gives a more accurate fitted result.

## 5.3 Discussion

The above double-filter sweeping method does not guarantee the dynamic correctness at the first trial. It is understandable considering the nature of per-frame-processing-then-fitting approximation. However, a very important feature of the sweeping method is its enhancing property: when we apply the double filter to the filtered result again, we get more accurate result. Another important property is that the result converges quickly. Therefore we repeat the sweeping until we get the acceptable result. In our experiments, it took on average 4 or 5 sweepings to converge. The process was quite acceptable since it allowed the animator to check the results interactively. But the convergence property was not always guaranteed. We found that, in some

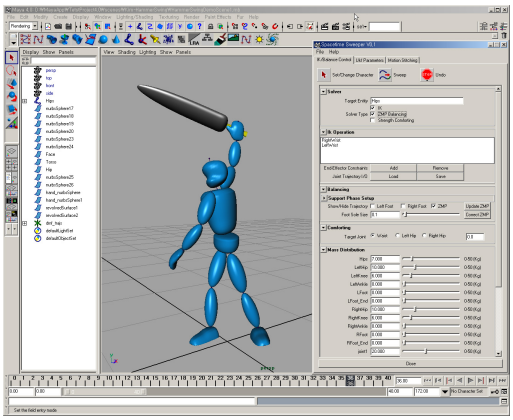


**Figure 6.** Motion editing system implemented as a Maya plug-in

cases, especially when the motion was highly dynamic, the method suffered from the convergence problem.

Our spacetime sweeping approach is near-sighted to the original motion. Therefore when a global optimum is required, our per-frame approach may not give the desired solution. For example, to generate a punch motion with the dynamic constraint that the final speed of a fist should be considerably larger than that of the original motion, the utmost pull-back action is required long before the arm makes the hit. But the sweeping algorithm tries to achieve such a constraint by modifying only the frames in the neighborhood of the final frame. In such a case, it is helpful to put a kinematic editing step before the double-filter sweeping, to generate a reasonable initial guess (we call it a *kinematic hint*).

# 6 Results

We have implemented the spacetime sweeping technique as a Maya plug-in using the Maya's plug-in API and MEL scripts on a PC with a Pentium-3 933MHz processor and a 512-Mbytes main memory. Figure 6 is a screen shot of our implementation. With this plug-in, animators could interactively edit a motion to their purpose by changing parameters or manipulating desired paths.

Our human model has total 54 DOFs with all the joints represented by quaternion ball joints. As input sequences, we used the optical motion capture data of walking, sword swing, ballet, etc. In order to show the retargeting quality and the range of application of our algorithm, we retargeted the captured data to new characters (with different link lengths, body mass distribution, joint strength limits), and to new environments (with obstacles, slopes, wind, or other terrain conditions). All the motion clips mentioned in this section can be found in `http://graphics.snu.ac.kr/research/ss/`.

## 6.1 Retargeting to Different Characters

We retargeted several captured motions (walking motion, sword swing, ballet dancing) of the original character to other characters with different link lengths and weights, joint strengths, etc. In the figures and animation clips, the thickness of a link reflects its weight.

**Realtime Kinematic Motion Retargeting.** In order to experiment our kinematic motion retargeting, we captured a walking sequence of a character having normal anthropometry, and retarget it to two other characters having quite different anthropometry: one with shorter limbs and the other with longer limbs. The animation clip #1 shows the result, and Figure 7 (a) is a snapshot during this motion. Although the animation was generated with off-line rendering, the sweeping process ran in real-time. We first adjusted the height of the pelvis of the two target characters to account for the differences in height. Then, we transmitted in on-line the joint angle data (used as the motion data) and the foot trajectories (used as the kinematic constraints) of the original character to the target characters.

**Sword Swing.** We captured a sword-swing motion (the sword weighs about 1.5 Kg), in which the subject made a big swing. We retargeted the motion to a very lean character (weighs $\frac{1}{4}$ of the original character) to see how the momentum of the heavy sword affects the motion of the target character. We performed the sweeping with the balance constraint. The motion clips #2 and #3 are the original and retargeted motion, respectively. We show a few snapshots during the motions in Figure 7 (b). The upper-body of the target character makes a big back-and-forth movement compared to the original character.

**Ballet Dancing.** We retargeted a ballet motion of a ballerina to a child character who has quite heavy lower limbs. Since the ballerina had strong joints (due to the training), she could effortlessly rotate the joints (especially the hip joint) over a wide range of angles. On the other hand, the joints of a child are usually weak, and such weakness will be more apparent in the hip joint when the lower limb is heavy. To account for the above distinctions, we imposed the torque constraints. Also with the balance constraint, we performed the sweeping. The motion clips #4 and #5 show the original and retargeted ballet motions, and Figure 7 (c) is showing some snapshots during the motions. As expected, the leg of the child could not be lifted as high as in the ballerina, and his upper body had to make sways to compensate the momentum of the heavy swing leg.

**Limbo Walking.** In this experiment, we used a normal walking motion as an input sequence to generate a limbo walking. To avoid unnecessary wandering in dealing with dynamic constraints, first we outlined a plausible limbo motion kinematically. Then, we performed the sweeping with the balance and torque (at the waist) constraints along with the limbo-bar constraint. We designated the torque constraints as a hard constraints, since injury is the first thing to avoid in the actual limbo motion. The resulting motion is shown in the animation clip #6. The top row of Figure 7 (d) shows a several snapshots taken during the motion.

We re-fed the above retargeted result to the sweeping algorithm in order to produce the limbo motion of a third character, who has a very heavy torso. In this setup, the character could not bend the torso as in clip #6 due to the (hard) torque constraints. The resulting animation is shown in the motion clip #7, and the snapshots during the motion is shown in the bottom row of Figure 7 (d).

To produce the above animation clips, we swept 5 times for the sword-swing and ballet-dancing, and 4 times for the limbo-walking. The sweeping window of the LS filter was 32-frames wide. Although the lengths of the above motion sequences are different, they ran at a uniform interactive rate (5 frames per second).

### 6.2 Retargeting a Normal Walk to New Environments

The experiments in this section show that a walking sequence can be retargeted to accommodate different terrains, winds, etc. The retargeting was done by 3 times of the sweepings with a 36-frame wide sweeping window. The result is shown in the motion clip #8, and the snapshots during the animation are shown in Figure 8.

**Ascending Uphill / Descending Downhill.** We modelled a 20-degree slope, and retargeted a walking motion on a level ground to the ascending and descending versions. The spacetime sweeping with the balance constraints and the new adjusted foot trajectory constraints produced dynamically plausible walks.

**Walking on Widened Footprints.** We widened the footprints in the lateral direction, and ran the spacetime sweeping with the balance constraints. We can observe that the character rhythmically sways his body to meet the balance constraint, which could not be generated by kinematic constraints.

**Walking against a Wind.** We put a strong windstorm in front of the walking character. The magnitude of the wind force was set to one third of the gravity force. The retargeted character walks leaning forward in order to maintain the dynamic balance.

## 7 Conclusion

We have presented a new approach to the problem of editing an existing motion to satisfy kinematic and dynamic constraints. We cast the motion editing problem as a constrained state estimation problem based on the per-frame Kalman filter framework.

Spacetime sweeping incorporates the kinematic and dynamic constraints in a scalable framework. When only kinematic constraints are involved, application of the UK filter could guarantee the kinematically constrained motion with real-time performance. When dynamic constraints are involved, application of the double-filter (UK and LS filters) could produce a dynamically sound motion in an interactive rate.

Spacetime sweeping works as an enhancement operator, so that every application produces improved results in the kinematic or dynamic accuracy. Such incremental behavior can be valuable in animation productions, since the intermediate results can be interactively checked.

## Acknowledgment

This work was supported by Korea Ministry of Information and Communication. This work was also partially supported by Automation and Systems Research Institute at Seoul National University, and the Brain Korea 21 Project.

## References

[1] A. Bruderlin and L. Williams. Motion signal processing. In *Computer Graphics (SIGGRAPH 95 Proceedings)*, pages 97–104, August 1995.

[2] K. Choi and H. Ko. On-line motion retargetting. In *Pacific Graphics 99 Proceedings*, pages 32–42, October 1999.

[3] M. F. Cohen. Interactive spacetime constraint for animation. In *Computer Graphics (SIGGRAPH 92 Proceedings)*, pages 293–302, July 1992.

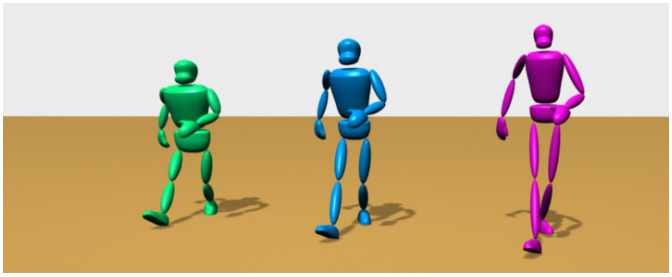
[4] J. J. Craig. *Introduction to Robotics*. Addison-Wesley, 1989.

[5] A. Dasgupta and Y. Nakamura. Making feasible walking motion of humanoid robots from human motion capture data. In *Robotics and Automation '99 Proceedings*, volume 2, pages 1044–1049, 1999.

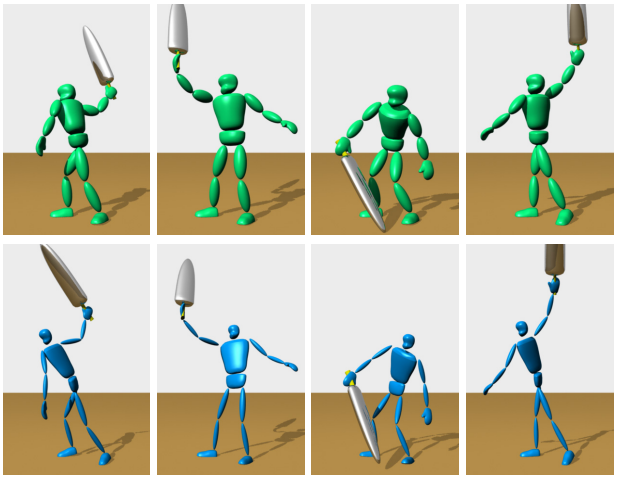
[6] T. Fukuda, Y. Komata, and T. Arakawa. Stabilization control of biped locomotion robot based learning with gas having self-adaptive mutation and recurrent neural networks. In *Robotics and Automation '97 Proceedings*, volume 1, pages 217–222, 1997.

[7] M. Gleicher. Motion editing with spacetime constraints. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 139–148, 1997.

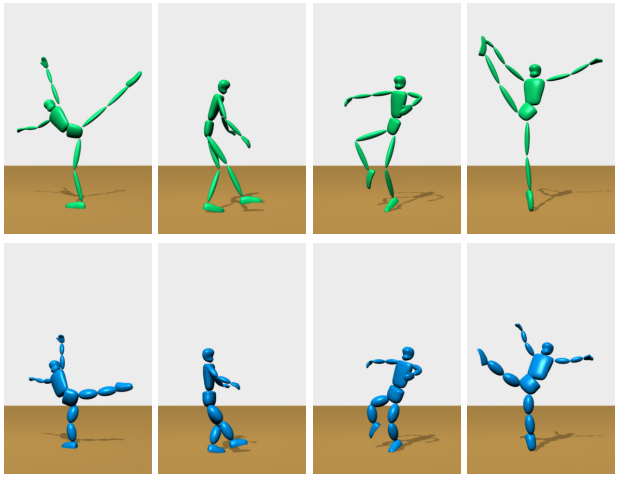
[8] M. Gleicher. Retargetting motion to new characters. In *Computer Graphics (SIGGRAPH 98 Proceedings)*, pages 33–42, July 1998.



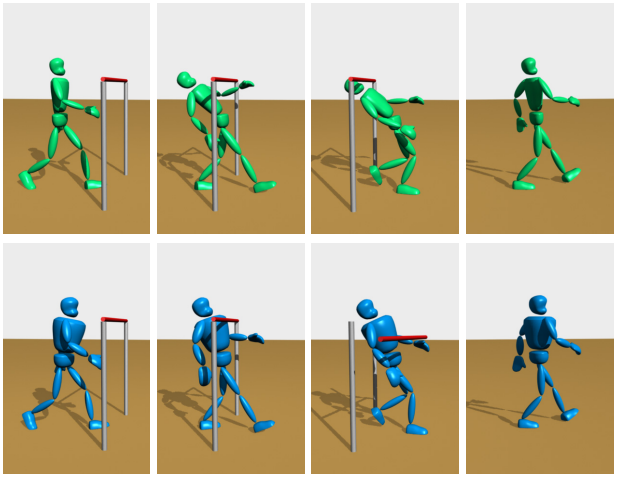
(a) Real-time kinematic motion retargeting



(b) Retargeting of sword-swing motion



(c) Retargeting of ballet motion



(d) Retargeting from a normal walk to two limbo walks

**Figure 7.** Motion retargeting to different characters

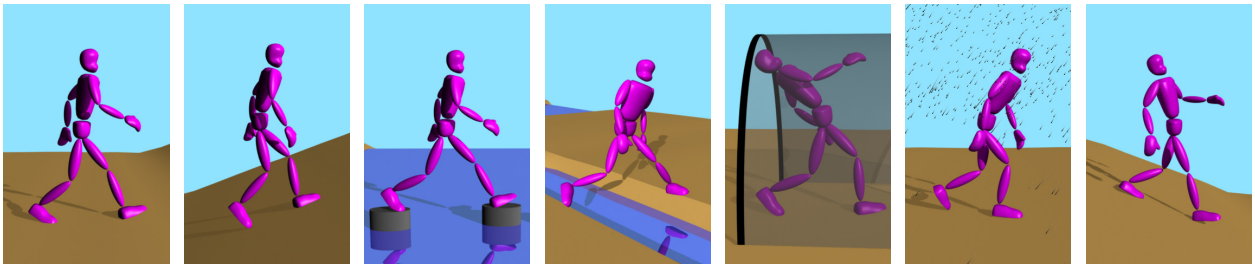


**Figure 8.** Retargeting a normal walk to some environmental conditions: From left to right, the original normal walk, ascending uphill, walking on irregular stepping-stones, walking with wide footsteps, limbo walking, walking against a strong wind, and descending downhill.

[9] M. Gleicher. Comparing constraint-based motion editing methods. *Graphical Models*, 63(2):107–134, August 2001.

[10] F. S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48, 1998.

[11] S. J. Julier and J. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, 1997.

[12] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME Journal of Basic Engineering*, pages 35–45, 1960.

[13] H. Ko. *Kinematic and Dynamic Techniques for Analyzing, Predicting, and Animating Human Locomotion*. PhD thesis, University of Pennsylvania, Department of Computer and Information Science, Philadelphia, PA 19104-6389, May 1994. MS-CIS-94-31.

[14] J. Lee and S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Computer Graphics (SIGGRAPH 99 Proceedings)*, August 1999.

[15] P. Lee, S. Wei, J. Zhao, and N. I. Badler. Stregth guided motion. In *Computer Graphics (SIGGRAPH 90 Proceedings)*, August 1990.

[16] P. S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, Inc., 1979.

[17] J. H. Park and Y. K. Rhee. Zmp trajectory generation for reduced trunk motions of biped robots. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)*, pages 90–95. IEEE/RSJ IROS, October 1998.

[18] Z. Popović and A. Witkin. Physically based motion transformation. In *Computer Graphics (SIGGRAPH 99 Proceedings)*, August 1999.

[19] A. A. Shabana. *Computational Dynamics*. John Wiley & Sons, Inc., 1994.

[20] H. J. Shin, J. Lee, S. Y. Shin, and M. Gleicher. Computer puppetry: An importance-based approach. *ACM Transactions on Graphics*, 20(2):67–94, 2001.

[21] K. Shoemake. Animating rotation with quaternion curves. In *Computer Graphics (SIGGRAPH 85 Proceedings)*, 1985.

[22] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, 1986.

[23] S. Tak, O. Song, and H. Ko. Motion balance filtering. *Computer Graphics Forum (Eurographics 2000)*, 19(3):437–446, 2000.

[24] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan. The unsceneted particle filter. Technical Report CUED/F-INFENG/TR 380, Dept. of Engineering, University of Cambridge, August 2000.

[25] M. Vukobratović. *Biped Locomotion*. Scientific Fundamentals of Robotics 7, Communications and Control Engineering Series. Springer-Verlag, Berlin, New York, 1990.

[26] E. A. Wan and R. van der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control*, October 2000.

[27] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report 95-041, Department of Computer Science, University of North Carolina, June 2000.

[28] G. Welch and G. Bishop. An introduction to the kalman filter. *SIGGRAPH 2001 Course Notes*, Augst 2001.

[29] D. A. Winter. *Biomechanics and Motor Control of Human Movement*. Wiley, New York, second edition, 1990.

[30] A. Witkin and M. Kass. Spacetime constraints. In *Computer Graphics (SIGGRAPH 88 Proceedings)*, pages 159–168, August 1988.

[31] A. Witkin and Z. Popović. Motion warping. In *Computer Graphics (SIGGRAPH 95 Proceedings)*, pages 105–107, August 1995.