In Proceedings of the Third International Conference on Motion In Games (MIG), Zeist, the Netherlands, 2010

# Motion Parameterization with Inverse Blending

Yazhou Huang and Marcelo Kallmann

University of California, Merced

**Abstract.** Motion blending is a popular motion synthesis technique which interpolates similar motion examples according to blending weighs parameterizing high-level characteristics of interest. We present in this paper an optimization framework for determining blending weights able to produce motions precisely satisfying multiple given spatial constraints. Our proposed method is simpler than previous approaches, and yet it can quickly achieve locally optimal solutions without pre–processing of basis functions. The effectiveness of our method is demonstrated in solving two classes of problems: 1) we show the precise control of end-effectors during the execution of diverse upper–body actions, and 2) we also address the problem of synthesizing walking animations with precise feet placements, demonstrating the ability to simultaneously meet multiple constraints and at different frames. Our several experimental results demonstrate that the proposed optimization approach is simple to implement and effectively achieves realistic results with precise motion control.

**Keywords:** motion parameterization, character animation, walk synthesis, spatial constraints.

## 1 Introduction

Keyframe animation and motion capture represent popular approaches for achieving high–quality character animation in interactive applications such as in 3D computer games and virtual reality systems. In particular, motion blending techniques [17, 18, 15] provide powerful interpolation approaches for parameterizing pre–defined example animations according to high–level characteristics. While direct blending of motions is able to produce fast and realistic motions, it remains difficult to achieve blendings and parameterizations able to precisely satisfy generic spatial constraints. We show that with an optimization approach we are able to always solve spatial constraints when possible, and usually less example motions are required to cover the spatial variations of interest.

Our method models each spatial constraint as an objective function whose error is to be minimized. The overall multi–objective inverse blending problem is solved by optimizing the blending weights until a locally optimal solution is reached. Solutions can be found in few milliseconds and no pre–computation of basis functions is needed. The method is therefore suitable for interactive applications and several results running in real–time are presented.

While previous work has addressed the maintenance of spatial properties in a single motion interpolation step [18, 15, 11], we focus on optimizing blending weights until best meeting multiple spatial constraints. Our approach has the additional flexibility of modeling spatial constraints with objective functions which are independent of the abstract space used by the motion blending. Generic spatial constraints can be handled and Inverse Kinematics problems can also be solved based on motion blending. We have in particular recently applied the presented framework to an interactive virtual reality training application [5], and the obtained results were very effective.

This paper demonstrates our methods in three scenarios: pointing to objects, pouring water and character locomotion. The spatial constraints of inverse blending are modeled differently for each scenario. As a result, our interactive motion modeling framework allows animators to easily build a repertoire of realistic parameterized human–like motions (gestures, actions, locomotion, etc) from examples which can be designed by hand or collected with motion capture devices.

## 2 Related Work

Several approaches to motion interpolation have been proposed involving different techniques such as: parameterization using Fourier coefficients [22], hierarchical filtering [4], stochastic sampling [23], and interpolation based on radial basis functions (RBFs) [17]. Our motion blending framework is closely related to an extension of the verbs and adverbs system which performs RBF interpolation to solve the Inverse Kinematics (IK) problem [18]. RBFs can smoothly interpolate given motion examples and the types and shapes of the basis functions are optimized in order to better satisfy the constraint of reaching a given position with the hand.

More generically, spatial properties such as feet sliding or hand placements are well addressed by the geostatistical interpolation method [15], which computes optimal interpolation kernels in accordance with statistical observations correlating the control parameters and the motion samples. Another approach for improving the maintenance of spatial constraints is to adaptively add pseudo–examples [18, 11] in order to better cover the continuous space of the constraint. This random sampling approach however requires significant computation and storage in order to meet constraints accurately and is not suited for handling several constraints.

In all these previous methods spatial constraints are only handled as part of the motion blending technique employed, i.e., by choosing sample motions which are close to the desired constraints and then using the abstract interpolation space to obtain motion variations which should then satisfy the constraints. Another possible technique sometimes employed is to apply Inverse Kinematics solvers in addition to blending [18, 6], however risking to penalize the obtained realism.

The work on mesh–based IK [20] does address the optimization of blending weights for the problem of blending example meshes. However, although our approach is simple and intuitive, no specific previous work has specifically analyzed and reported results applied to skeletal motion, and in particular also simultaneously solving multiple constraints and at different frames.

The Scaled Gaussian Process Latent Variable Model (SGPLVM)[9] provides a more specific framework targeting the IK problem which optimizes interpolation kernels specifically for generating plausible poses from constrained curves such as positional trajectories of end–effectors. The approach however focuses on maintaining constraints described by the optimized latent spaces. Although good results are obtained, constraints cannot be guaranteed to be precisely met.

The presented approach can be seen as a post–processing step for optimizing a given set of blending weights, which can be initially computed by any motion blending technique. We demonstrate in this paper that our optimization framework is able to address any kind of constraints without even the need of specifying an abstract parameterization space explicitly. Only error functions for the spatial constraints are necessary in order to optimize the blending weights using a given motion interpolation scheme.

We also apply our method for parameterization of locomotion, which is a key problem in character animation. Many methods have been previously proposed for finding optimal solutions for path following [14, 12], for reaching specified locomotion targets [19, 10], or also for allowing interactive user control [21, 14, 1]. Most of these works combine motion blending techniques with motion graphs [12, 2, 3, 8, 19] and can then generate different styles of locomotion and actions. Different than these methods, we give specific attention to precisely meet specified feet placements. The geostatistical interpolation method [15] reduces feet sliding problems but still cannot guarantee to eliminate them. Other precise feet placement techniques [13, 7] are available, however not based on a generic motion blending approach.

In conclusion, diverse techniques based on motion blending are available and several of these methods are already extensively used in commercial animation pipelines for different purposes. Our work presents valuable experimental results demonstrating the flexibility and efficiency of a simple optimization framework for solving inverse blending problems involving multiple spatial constraints. Our approach is effective and easy to implement, and was first described in [5], where it was applied to an interactive virtual reality training application. The formulation is again described here for completeness purposes, and then several extensions and new results are presented to effectively model diverse parameterized upper–body actions, and also to model parameterized walking animations with precise footstep placements.

## 3 Inverse Blending

Given a set of similar and time–aligned motion examples, we first employ a traditional RBF motion interpolation scheme to compute an initial set of blending weights. These weights are then optimized to meet a given constraint $C$.

Each motion $M$ being interpolated is represented as a sequence of poses with a discrete time (or frame) parameterization $t$. A pose is a vector which encodes the root joint position and the rotations of all the joints of the character. Rotations are encoded with exponential maps but other representations for rotations (as quaternions) can also be used. Each constraint $C$ is modeled with a function $e = f(\boldsymbol{M})$, which returns the error evaluation $e$ quantifying how far away the given motion is from satisfying constraint $C$. We first select the $k$ motions which are the ones best satisfying the constraints being solved. For example, in a typical reaching task, the $k$ motion examples having the hand joint closest to the target will be selected. The $k$ initial motions are therefore the ones with minimal error evaluations. The initial set of blending weights $w_j$, $j = \{1, \ldots, k\}$, are then initialized with a RBF kernel output of the input $e_j = f(M_j)$. We constrain the weights to be in interval $[0, 1]$ in order to stay in a meaningful interpolation range and we also normalize them to sum to 1. Any kernel function can be used, as for example kernels in the form of $exp^{-\|e\|^2/\sigma^2}$. In this work we do not attempt to optimize kernel functions in respect to the constraints [17, 15] and instead we will optimize the weights independently of the interpolation method. Our interpolation scheme computes a blended motion with:

$$\boldsymbol{M}(\boldsymbol{w}) = \sum_{j=1}^{k} w_j M_j, \quad \boldsymbol{w} = \{w_1, \dots, w_k\}. \tag{1}$$

In order to enforce the given constraint $C$, our goal is to find the optimal set of blending weights $\boldsymbol{w}$, which will produce the minimum error $e^*$ as measured by the constraint error function $f$:

$$e^* = min_{w_j \in [0,1]} \; f\left(\sum_{j=1}^{k} w_j M_j\right). \tag{2}$$

The presented formulation can easily account for multiple constraints by combining the error metric of the given constraints in a single weighted summation. In doing so we introduce two more coefficients for each constraint $C_i$, $i = \{1, \dots, n\}$: a normalization coefficient $n_i$ and a prioritization coefficient $c_i$. The goal of $n_i$ is to balance the magnitude of the different constraint errors. For example, positional constraints depend on the used units and in general have much larger values than angular values in radians, which are typically used by orientation errors. Once the normalization coefficients are set, coefficients $c_i \in [0, 1]$ can be interactively controlled by the user in order to vary the influence (or priority) of one constraint over the other.

The result is essentially a multi–objective optimization problem, with the goal being to minimize a new error metric composed of the weighted sum of all constraints' errors:

$$f(\boldsymbol{M}(\boldsymbol{w})) = \sum_{i=1}^{n} \left(c_i \, n_i \, f_i \left(\boldsymbol{M}(\boldsymbol{w})\right)\right). \tag{3}$$

Independent of the number of constraints being addressed, when constraints are fully satisfied, $e^* \to 0$.

## 4 Action Parameterization with Inverse Blending

Figure 1 presents several examples obtained for parameterizing upper–body actions. Different types of spatial constraints were used. Constraint $C_{pos}$ is a 3-DOF positional constraint which requires the end–effector (hand, finger tip, etc) to precisely reach a given target location. Constraint $C_{line}$ is a 2-DOF positional constraint for aligning the hand joint with a given straight line in 3D space. Constraint $C_{ori}$ is a 1 to 3 DOFs rotational constraint which requires the hand to comply to a certain given orientation. Note that all these constraints are only enforced at one given frame of the motion. Constraints

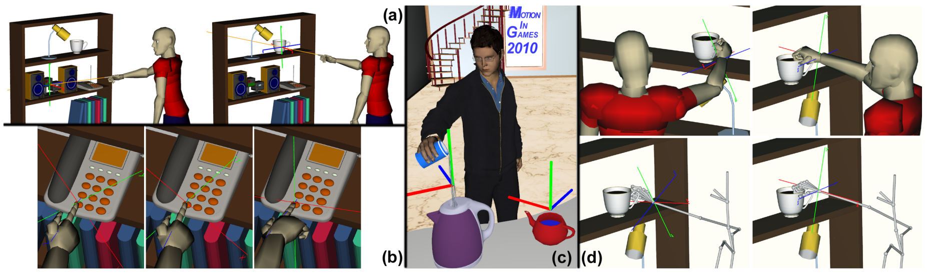

**Fig. 1.** This figure presents several results obtained by inverse blending. A $C_{line}$ constraint is used for precisely pointing to distant objects (a) and for pouring exactly above the teapot (c). Positional $C_{pos}$ and rotational $C_{ori}$ constraints are used for pin–pointing a button on the telephone (b). Note that the finger tip precisely reaches the button, and the hand orientation matches that of the shown x-axis. Constraints $C_{pos}$ and $C_{ori}$ are also used for achieving precise grasping motions (d).

can also be combined in order to allow additional ways of parameterizing motions. For example by combining $C_{line}$ and $C_{ori}$, precise grasping targets can be achieved (Figure 1-d), and different hand orientations can be obtained when pin–pointing buttons (Figure 1-b).

## 5 Parameterized Character Locomotion with Inverse Blending

This section demonstrates our inverse blending framework for generating parameterized walking motions with precise control of feet placements for character navigation.

First, two sets of motion examples are prepared with clips obtained from motion capture. The first set $\boldsymbol{R}_m$ consists of right foot stepping motions. Each example motion $M_r \in \boldsymbol{R}_m$ represents one full step forward with the right foot while the left foot remains in contact with floor as the support foot, see Figure 2-a. The second set of example motions $\boldsymbol{L}_m$ is prepared in the same way but containing stepping examples for left foot.

The motions in both sets contain many variations, e.g. step length, step direction, body orientation, velocity, root joint position, etc. These should well span the variations of interest, which are to be precisely parameterized by inverse blending. Figure 2-b illustrates how some of the motions in our database look like. No alignment of the motions is needed and the variations will actually be explored by the inverse blending optimization in order to reach any needed alignments on–line. We also mirror the example motions in both sets in order to guarantee the same number of examples (and variations) are available in each set.

As we are interested in precisely controlling each footstep location during walking, the length and direction of each step is parameterized while the support foot contact on the floor is maintained. Let $\theta^e$ be the orientation of the support foot at the starting frame of one stepping motion example (rotational constraint). Let $v^s$ and $v^e$ be vectors encoding the position of the stepping foot in respect to the support foot at the start and at the end frames respectively (positional constraints). Figure 2-c illustrates these parameters for one left step. Each stepping motion of interest is then specified as a

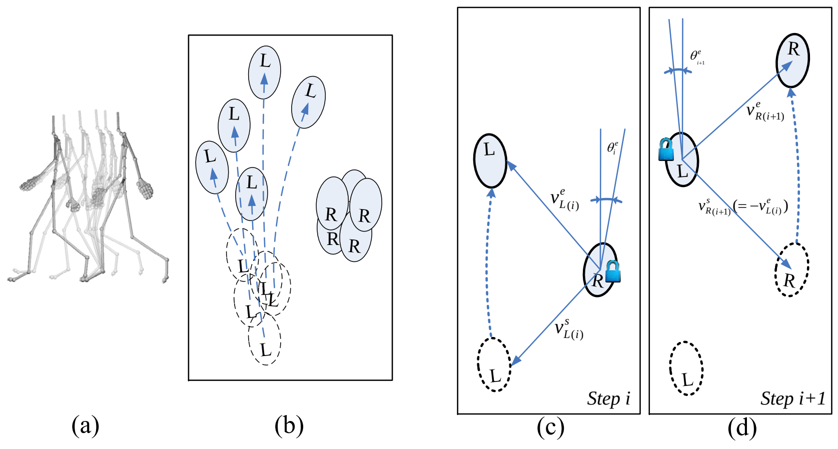

**Fig. 2.** (a) shows snapshots of one right foot stepping motion from $\boldsymbol{R}_m$ example set. (b) is a top–down footprint view of several left foot stepping motions used in $\boldsymbol{L}_m$ example set. The footprints are from diverse walk segments and do not need to be aligned. (c) shows the constraints for computing step $i$: $\theta^e$ is the rotational constraint for the support foot (with lock icon), $v^s$ and $v^e$ are positional constraints for the stepping foot at the start and end of the motion respectively. (d) shows the constraints for step $i+1$, which immediately follows step $i$.

function of these parameters with $M(v^s, v^e, \theta^e)$, which will be obtained by inverse blending procedures based only on the stepping examples available in the $\boldsymbol{R}_m$ and $\boldsymbol{L}_m$ motion sets.

With the given constraints $v^s, v^e, \theta^e$ described above, the process for obtaining $M(v^s, v^e, \theta^e)$ is composed of 4 phases, as described in the next paragraphs.

**Phase 1**: the set of blending weights $\boldsymbol{w}^s$ is computed by inverse blending such that the stepping foot respects the positional constraint $v^s$ at the start frame $t^s$. As these weights are computed to meet constraints $v^s$ we use the notation $\boldsymbol{w}^s(v^s)$ for the obtained blending weights.

**Phase 2**: we solve a new inverse blending problem for determining the blending weights $\boldsymbol{w}^e$ at the end frame $t^e$ in order to meet two constraints: the positional constraint $v^e$ for the stepping foot and the rotational constraint $\theta^e$ for the support foot. Therefore the obtained weights $\boldsymbol{w}^e(v^e, \theta^e)$ produce an end posture with the stepping foot reaching location $v^e$, while the support foot respects the orientation specified by $\theta^e$.

**Phase 3**: the average lengths $l_{avg}$ of the example motions in phase 1 and 2 is used to time–align the blended motions. The blending weights used to produce the required stepping motion is finally obtained as a function of the frame time $t$, such that $\boldsymbol{w}(t)$ $=interp(\boldsymbol{w}^s(v^s), \boldsymbol{w}^e(v^e, \theta^e), t)$, $t \in [0, l_{avg}]$. The interpolation function $interp$ employs a smooth in and out sine curve and each of the motions are time warped to $l_{avg}$ in order to cope with variations of step lengths and speeds in the used motion set. The final parameterized stepping motion is then obtained with $M(v^s, v^e, \theta^e) = \boldsymbol{w}(t)\Sigma_{i=1}^{k} M_i$ . This process is illustrated in Figure 3.

**Phase 4**: this phase consists of a velocity profile correction [24] in order to maximally preserve the overall quality of the original motions since several blending operations have been performed at this point. For that we select the root joint velocity profile of the motion example which gave most contribution in the inverse blending procedures. The time parameterization of $\boldsymbol{w}(t)$ is then adapted on the fly in order to obtain a motion with the root joint velocity profile matching the selected reference profile. Figure 3 bottom–right exemplifies the root increment against frames during a two–step walking sequence showing how root velocity changes over time. This has been proven to well preserve the quality of the obtained results.

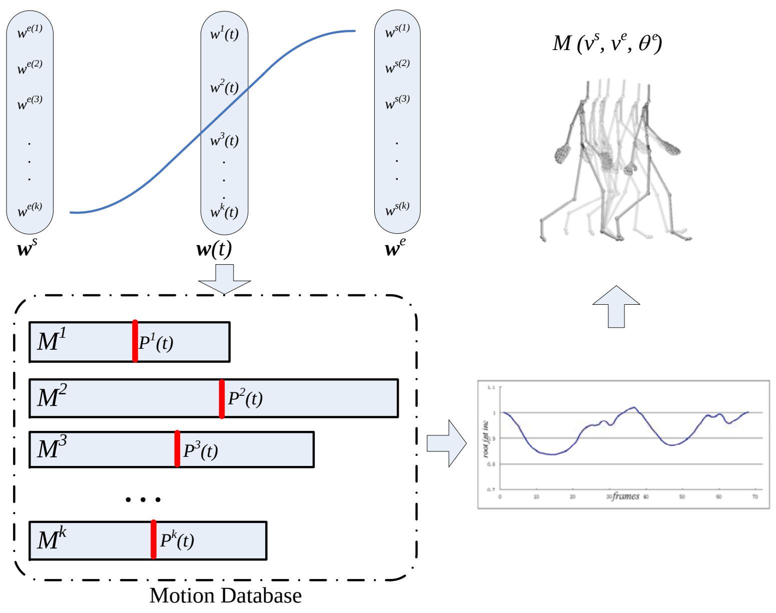

**Fig. 3.** The motion examples selected by the inverse blending $M^i (i = 1 \sim k)$ are blended with interpolated weights $\boldsymbol{w}(t)$ which ensure spatial constraints both at the start and end frame of the motions. Velocity profiles are adapted on–line to preserve the original quality and realism.

The procedure described above is applied each time a stepping motion has to be generated. For producing stepping motions for the right foot, $M_R(v^s, v^e, \theta^e)$ is obtained by using the example motions from $\boldsymbol{R}_m$. Left foot stepping motions $M_L(v^s, v^e, \theta^e)$ are similarly obtained using examples from $\boldsymbol{L}_m$. As a result a walking sequence achieving precise feet placements at each step can be obtained with the following concatenation of alternating steps: $M_L(v_1^s, v_1^e, \theta_1^e)$, $M_R(v_2^s, v_2^e, \theta_2^e)$, $M_L(v_3^s, v_3^e, \theta_3^e)$, $\cdots$ .

During each stepping motion in the sequence above, the character is translated at every frame to make sure the support foot does not slide on the floor (i.e. its location and orientation are maintained), this will essentially make the character walk forward. At the end of each stepping, the support foot changes, and its location and orientation are updated, ready for the following step. With this, the common problem of feet–sliding is here eliminated.

When computing each stepping motion, we make constraint $v^s_{i+1}$ equal to $-v^e_i$ from the previous step (see Figure 2-c and 2-d), for smooth transition between step $i$ and step $i+1$. The negation appears because the support foot and stepping foot are swapped from step $i$ to $i+1$.

Figure 4 shows the end posture $P^e_L$ (thick red line) of the left step $M_L(v^s_i, v^e_i, \theta^e_i)$ and the start posture $P^s_R$ (thick green line) of the right step $M_R(v^s_{i+1}, v^e_{i+1}, \theta^e_{i+1})$. With $v^s_{i+1} = -v^e_i$, inverse blending generates postures $P^e_L$ and $P^s_R$ matching the constraints and with body postures which are very close to each other. The small difference in the body posture is handled by smoothly concatenating the stepping motions with a brief ease–in blending period from $M_L$ going into $M_R$, achieving a smooth overall transition.

In the examples presented in this paper we have used only six stepping motion examples in each motion set, and yet the described inverse blending procedure can precisely control each foot placement within a reasonable range. If larger databases are used, a wider range for the constraints can be specified. Figure 5 shows several results obtained by our real–time walking control application. Several animation sequences are also available in the video accompanying this paper.

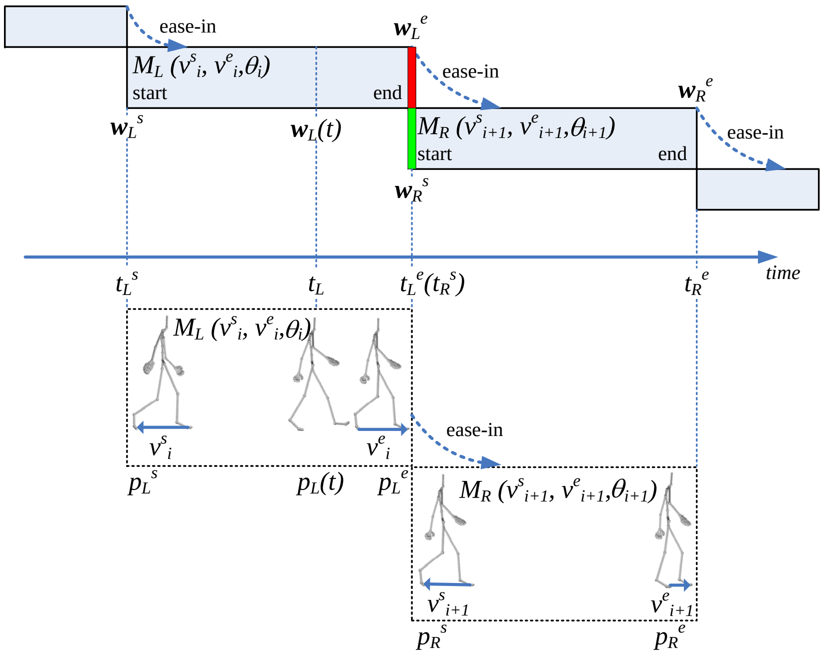

**Fig. 4.** This figure illustrates the generation of $M_L(v_n)$, and the concatenation of $M_L(v_n)$ and $M_R(v_{n+1})$ for achieving the final walking synthesis.

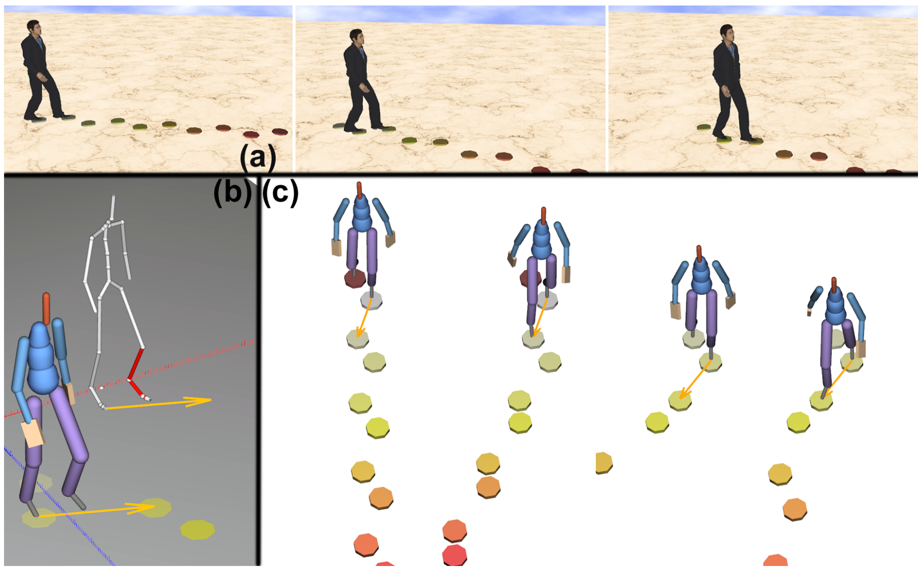

**Fig. 5.** The figure presents several snapshots of obtained walking motions where each step precisely meets the given feet targets (a and c). The targets for each step can be adjusted on the fly achieving a controllable locomotion generator with precise feet placements. The generation of the stepping motions is illustrated in the lower-left image (b), where the gray skeleton shows the inverse blending solution for the left foot, prior to concatenation.

## 6 Results and Discussion

With suitable example motions in a given cluster, inverse blending can produce motions exactly satisfying given spatial constraints and fast enough for real–time applications. Several of the figures in this paper illustrate the many experiments successfully conducted in different scenarios. To evaluate the performance of our method, a reaching task was designed to measure the errors produced by our method against a single RBF interpolation, with the 16 reaching motion database from Mukai et.al [15]. A total of 144 reaching targets (shown as yellow dots in Figure 6, each specifying a 3-DOF $C_{pos}$ constraint) were placed evenly on a spherical surface within reach of the character. The end locations of the hand trajectory in 16 example motions are shown as gray dots.

For each reaching target we first apply standard RBF interpolation alone to generate a reaching motion and record the final hand position where the character actually reaches. These 144 final positions were used to construct a mesh grid, which is shown in Figure 6-a. Each triangle on the mesh is colored in respect to the average errors from its vertices, representing the distance error between the final hand positions and their corresponding reaching targets. We then use our inverse blending optimization to perform the same tasks, and the constructed mesh is shown in Figure 6-b. The reaching motions generated by inverse blending can precisely reach most of the targets. Errors measured are practically zero across most of the mesh, and increase only at the boundary of the surface. In this specific task, the radius of the spherical surface was set to $80cm$, and both methods used eight example motions from the database ($k = 8$) for computing each reaching task.

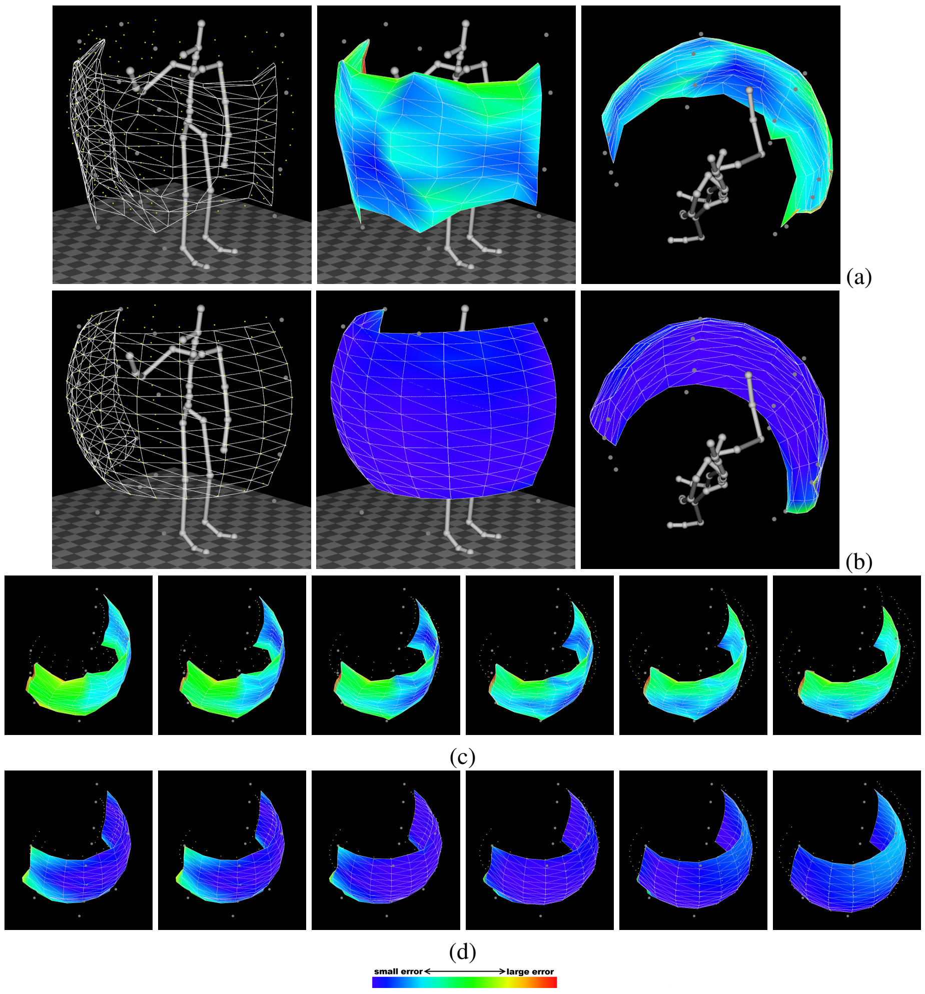

**Fig. 6.** Error evaluations. The meshes constructed by a standard RBF interpolation (a and c) result in much larger errors than by our inverse blending optimization (b and d), which most often produces no error.

Additional experiments were performed by varying the radius of the spherical surface to be $65, 70, 75, 80, 85$ $and$ $90cm$. Again a total of 144 reaching targets were generated on the spherical surfaces, covering a large volume of the workspace. These spherical surfaces are shown in Figures 6-c and 6-d. The constructed meshes by inverse blending are shown in Figure 6-d, and the results obtained with the RBF interpolation are shown in Figure 6-c. It is possible to note that the inverse blending optimization produces a smooth mesh very well approximating the yellow dots, and the errors produced by our method are clearly much lower, with most areas in pure blue.

Using standard optimization techniques [16] our inverse blending problems could be solved under 1.16 ms of computation time on average, with worse cases taking 2.29 ms (with a non–optimized single core code on a Core 2 Duo 2.13 GHz). Three scenarios (character performing pointing, pouring water and walking with precise feet placements) were used for this evaluation, with each scenario solving 5000 inverse blending problems towards random placements.The approach is therefore suitable for real–time applications, and in addition, it does not require pre–processing of the motion database, making it suitable for systems interactively updating the motion database (as in [5]).

In terms of limitations, two main aspects have to be observed. First, the ability of enforcing constraints greatly depends on the existing variations among the motion examples being blended. The number of needed example motions also depend on the size of the target volume space. For example, our walk generator can produce good results with only 6 stepping example motions (6 for left foot stepping, mirrored to become 12 for both feet) due to great variations available in the motions. However more example motions are typically needed to well cover a large reaching or pointing volume, and we have used 35 example motions in some cases. The second limitation, which is related to the first, is that the computational time required for finding solutions will also depend on the quality and number of motion examples (the $k$ value). However, as shown in our several examples, these limitations are easy to address by appropriately modeling example motions, and balancing the coverage vs. efficiency trade–off specifically for each action being modeled.

## 7 Conclusions

We have presented an optimization framework for satisfying spatial constraints with motion blending. Our approach is simple and can handle any type of spatial constraints. Several different actions (pointing, grasping, pouring, and walking) were successfully modeled and parameterized with precise placement of end–effectors. Our inverse blending framework has therefore shown to be a simple and powerful tool for achieving several useful motion parameterizations. We believe that our overall framework can significantly improve the process of modeling full–body motions for interactive characters.

**Acknowledgments:** This work was partially supported by NSF Awards IIS-0915665 and CNS-0723281, and by a CITRIS seed fund.

## References

1. Abe, Y., Liu, C.K., Popović, Z.: Momentum-based parameterization of dynamic character motion. In: SCA ’04: 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation. pp. 173–182. Eurographics Association, Aire-la-Ville, Switzerland (2004)
2. Arikan, O., Forsyth, D.A.: Synthesizing constrained motions from examples. Proceedings of SIGGRAPH 21(3), 483–490 (2002)

3. Arikan, O., Forsyth, D.A., O'Brien, J.F.: Motion synthesis from annotations. ACM Transaction on Graphics (Proceedings of SIGGRAPH) 22(3), 402–408 (2003)
4. Bruderlin, A., Williams, L.: Motion signal processing. In: SIGGRAPH '95. pp. 97–104. ACM, New York, NY, USA (1995)
5. Camporesi, C., Huang, Y., Kallmann, M.: Interactive motion modeling and parameterization by direct demonstration. In: Proceedings of the 10th International Conference on Intelligent Virtual Agents (IVA) (2010)
6. Cooper, S., Hertzmann, A., Popović, Z.: Active learning for real-time motion controllers. ACM Transactions on Graphics (SIGGRAPH 2007) 26(3) (Aug 2007)
7. Coros, S., Beaudoin, P., Yin, K.K., van de Pann, M.: Synthesis of constrained walking skills. ACM Trans. Graph. 27(5), 1–9 (2008)
8. Gleicher, M., Shin, H.J., Kovar, L., Jepsen, A.: Snap-together motion: assembling run-time animations. In: Proceedings of the symposium on Interactive 3D graphics (I3D). pp. 181–188. ACM Press, New York, NY, USA (2003)
9. Grochow, K., Martin, S., Hertzmann, A., Popović, Z.: Style-based inverse kinematics. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 23(3), 522–531 (2004)
10. Heck, R., Gleicher, M.: Parametric motion graphs. In: I3D '07: Proc. of the 2007 symposium on Interactive 3D graphics and games. pp. 129–136. ACM, New York, NY, USA (2007)
11. Kovar, L., Gleicher, M.: Automated extraction and parameterization of motions in large data sets. ACM Transaction on Graphics (Proceedings of SIGGRAPH) 23(3), 559–568 (2004)
12. Kovar, L., Gleicher, M., Pighin, F.H.: Motion graphs. Proceedings of SIGGRAPH 21(3), 473–482 (2002)
13. Kovar, L., Schreiner, J., Gleicher, M.: Footskate cleanup for motion capture editing. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA). pp. 97–104. ACM Press, New York, NY, USA (2002)
14. Kwon, T., Shin, S.Y.: Motion modeling for on-line locomotion synthesis. In: SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation. pp. 29–38. ACM, New York, NY, USA (2005)
15. Mukai, T., Kuriyama, S.: Geostatistical motion interpolation. In: ACM SIGGRAPH. pp. 1062–1070. ACM, New York, NY, USA (2005)
16. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes 3rd Edition: The Art of Scientific Computing. Cambridge Univ. Press, New York, NY, USA (2007)
17. Rose, C., Bodenheimer, B., Cohen, M.F.: Verbs and adverbs: Multidimensional motion interpolation. IEEE Computer Graphics and Applications 18, 32–40 (1998)
18. RoseIII, C.F., Sloan, P.P.J., Cohen, M.F.: Artist-directed inverse-kinematics using radial basis function interpolation. Computer Graphics Forum (Proceedings of Eurographics) 20(3), 239–250 (September 2001)
19. Safonova, A., Hodgins, J.K.: Construction and optimal search of interpolated motion graphs. In: ACM SIGGRAPH '07. p. 106. ACM, New York, NY, USA (2007)
20. Sumner, R.W., Zwicker, M., Gotsman, C., Popović, J.: Mesh-based inverse kinematics. ACM Trans. Graph. 24(3), 488–495 (2005)
21. Treuille, A., Lee, Y., Popović, Z.: Near-optimal character animation with continuous control. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers. p. 7. ACM, New York, NY, USA (2007)
22. Unuma, M., Anjyo, K., Takeuchi, R.: Fourier principles for emotion-based human figure animation. In: SIGGRAPH '95. pp. 91–96. ACM, New York, NY, USA (1995)
23. Wiley, D.J., Hahn, J.K.: Interpolation synthesis of articulated figure motion. IEEE Computer Graphics and Applications 17(6), 39–45 (1997)
24. Yamane, K., Kuffner, J.J., Hodgins, J.K.: Synthesizing animations of human manipulation tasks. In: ACM SIGGRAPH '04. pp. 532–539. ACM, New York, NY, USA (2004)