

Physically-based Character Control in Low Dimensional Space

Hubert P. H. Shum¹, Taku Komura², Takaaki Shiratori³, and Shu Takagi¹

¹ RIKEN, 2-1 Hirosawa, Wako, Saitama, Japan

² Edinburgh University, 10 Crichton Street, EH8 9AB, United Kingdom

³ Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh PA USA

Abstract. *In this paper, we propose a new method to compose physically-based character controllers in low dimensional latent space. Source controllers are created by gradually updating the task parameter such as the external force applied to the body. During the optimization, instead of only saving the optimal controllers, we also keep a large number of non-optimal controllers. These controllers provide knowledge about the stable area in the controller space, and are then used as samples to construct a low dimensional manifold that represents stable controllers. During run-time, we interpolate controllers in the low dimensional space and create stable controllers to cope with the irregular external forces. Our method is best to be applied for real-time applications such as computer games.*

Keywords: Character animation, dynamics simulation, dimensionality reduction, controller interpolation

1 Introduction

Recently, physical-based character simulation is attracting attention of game producers mostly because the simulated characters can react to a changing environment. The wide variety of simulated motions makes the game characters lively and increases the attractiveness of the games. State-of-the-art techniques have shown that optimized dynamic controllers are robust against external perturbations in different environments [21], which makes them particularly suitable for interactive systems such as fighting games. However, in case the perturbations are unpredictable, adjusting or synthesizing controllers during run-time becomes very challenging.

In this paper, we propose a method to synthesize new robust controllers using pre-computed controllers. We choose a finite state machine based controller as a fundamental framework due to its robustness and easiness to design [23]. Although some researches linearly interpolate each parameter of controllers [22], the non-linear relationship among the controllers tends to cause instability in the resultant motion. Therefore, we prepare samples of stable controllers through optimization and construct a non-linear low dimensional manifold using landmark Isomap[3]. To effectively construct the control manifold, we adapt a large number of sub-optimal controllers found during the optimization process.

Since these controllers give supplementary knowledge about the distribution of stable controllers, the manifold faithfully represents the stable region in the high dimensional space. During run-time, we interpolate optimal controllers in the low dimensional latent space with reference to the observed environment. Linear interpolation in the reduced space becomes a non-linear interpolation in the full control space, and hence produces robust controller interpolations.

We show that our system can produce stable controllers for characters perturbed by different external forces. Since the simulated characters can cope with arbitrary perturbations during run-time, our system is particularly suitable for situations where the external force is unpredictable, such as when several characters collaboratively push one object together.

2 Related Works

Dynamics Character Control: Designing a physically-based controller for character animation is difficult and non-trivial, as joint torques, instead of the joint angles, need to be specified. To ease the controller designing process, researchers propose to track captured motions [10, 14] or manually designed keyframes [2, 23], or use abstract models such as the inverted pendulum [7, 12, 17] to reduce the number of control variables. We apply keyframe based tracking because of its easiness of designing and implementation. Our idea to collect different controllers and composing a manifold for non-linear interpolation is also applicable for pre-designed abstract models, which may also require parameter tuning for handling unpredictable external perturbation.

Controller Optimization: The controller space is highly non-linear and non-differentiable. Traditional optimization techniques like gradient descent method can easily get caught in local minima [22]. On the contrary, as covariance matrix adaptation (CMA) does not assume a smooth space [5], it is particularly suitable for optimizing physically-based controllers [19, 21]. Since it is difficult to directly construct a controller for a drastically different environment, continuous optimization is proposed for gradually adjusting the environment during the optimization process [18, 22]. In this paper, we also apply CMA under the continuous optimization framework.

Dimensionality Reduction: Non-linear dimensionality reduction techniques have been widely applied for character control [4, 13]. We apply a method called landmark Isomap [3], which can discover the intrinsic dimensions of a given data set, and is much more computationally efficient than other methods such as Gaussian Process Latent Variable Models [9] and Isomap [16].

Run-time Synthesizing of Controllers: To deal with run-time changes in the environment and the conditions of the character, run-time controller optimization is proposed [8, 11]. Because optimization is computationally costly, it is suggested to precompute a set of controllers and apply run-time interpolation [22]. Since interpolating controllers in the irregular controller space often introduces instability, we propose to interpolate the controllers in a reduced space constructed from samples of stable controllers.

3 The Proposed Framework

The overview of our system is shown in Figure 1. Our objective is to train a series of controllers that can handle a task (such as walking against external perturbation) with different task values (such as the amount of the perturbation). Given an initial controller, our system applies continuous optimization to train a set of controllers that can cope with an incremental task parameter (the green area). The optimal controller for the current task value is used as the initial controller in the next iteration. CMA is used to optimize a controller with respect to the next task value (the red area). Each optimization step generates a covariance matrix and a number of controllers, which are applied in the next optimization step until the optimal controller is found. During each optimization step, reasonably good controllers that can keep walking without losing balance for 20 steps are stored in a database (the blue area). After the optimization is finished, landmark Isomap is applied to analyze the controllers and create a latent space of stable controllers, which is used for interpolation during run-time.

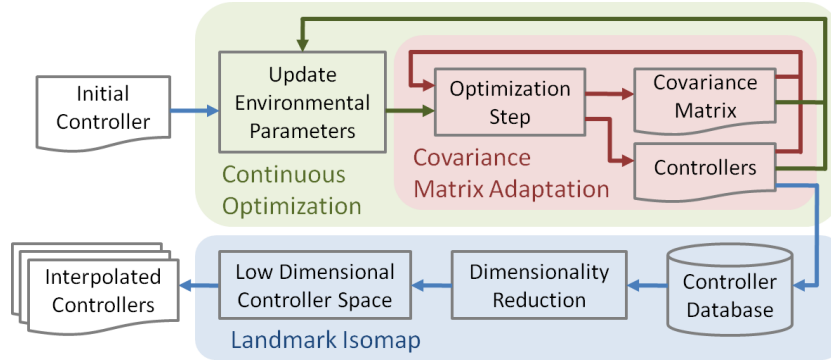


Fig. 1. We implement CMA under the continuous optimization framework. Based on the collected controller samples, we apply landmark Isomap to compute the latent space for run-time controller interpolation.

3.1 Model and Controller

Our character model has 14 joints. The hip and shoulder joints are modelled as two dimensional universal joints, while the rest are modelled as one dimensional hinge joints. In total, our model has 18 degrees of freedom. The total body size and weight are set $160cm$ and $70kg$, respectively, and the segment size and weight are set by referring to [1].

We implemented a physics-based gait controller similar to [23]. Four keyframes are used to simulate a walking motion. The target joint angles and proportional / derivative gains for each keyframe are provided by the controller. The system

transits from one keyframe to another with reference to a set of predefined durations. During optimization, we optimize the joint angles and the proportional gains for all keyframes. The derivative gains are set 10% of the proportional gains. The target joint angles and gain parameters are set constant for the arms in our experiments.

3.2 Continuous Optimization

In this section, we explain how we apply continuous optimization [22] to create a series of controllers with respect to different task parameters.

The idea of continuous optimization is to iteratively optimize the controller and increase the task parameter value. The system starts with a given initial controller. In each iteration, the task parameter is incremented, making the controller unstable. The system then optimizes the controller until an optimal one is found. Such a controller will be considered as the initial controller in the next iteration with an updated task value.

Figure 2 illustrates an example of continuous optimization. The task here is to walk in the presence of an external pushing force. Let us assume the controller contains two control parameters only, which are represented by axes. Each point in the graph thus represents a controller. The grey area indicates the space that the controller is unstable and the white area indicates where is stable. The optimization starts with a manually designed controller x_0 . We apply 5N of pushing force to the character and carry out the optimization until the optimal controller x_1 is found. Using x_1 as a starting point, we apply 10N of force and carry out another iteration of optimization, which returns a stable controller x_4 . Further increasing the force results in a stable controller x_6 .

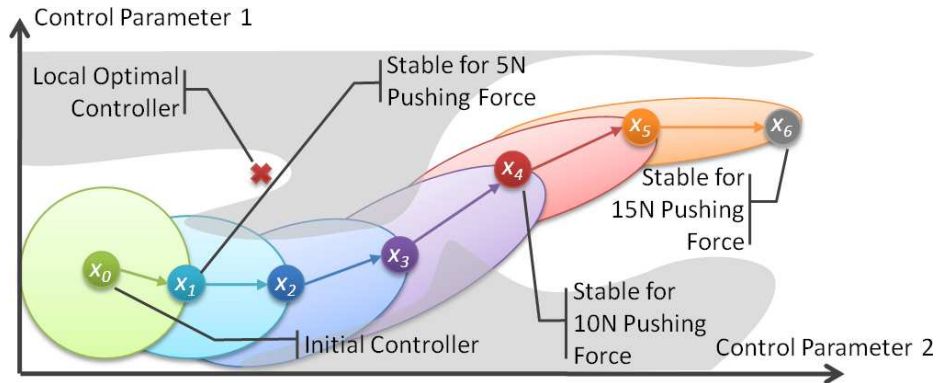


Fig. 2. Using continuous optimization, the optimization engine takes intermediate steps to create the final controller. The colored regions visualize the sampling areas calculated by covariance matrix adaptation.

Continuous optimization has two major advantages. First, since the optimal controller is used in the next iteration as the initial controller, the optimization engine explores area around the previously stable controller. This helps to maintain the similarity among the set of resultant controllers. Second, since the task parameter is updated incrementally, the optimization system is less likely to get trapped in local optimal solution. For example, in Figure 2, if we optimize a $15N$ controller directly from the initial controller x_0 , there will be a high chance that the optimization gets caught at the local optimal solution indicated by the red cross. However, because the task parameter is updated incrementally, we indirectly guide the optimization process step by step and avoid the local optimal.

3.3 Covariance Matrix Adaptation

In this section, we explain the covariance matrix adaptation (CMA) method [5] that we used to optimize the controllers with respect to a given task parameter.

The main concept of CMA is to repeatedly sample and evaluate a set of controllers based on a covariance matrix and a reference controller. In contrast to stochastic search, the covariance matrix is used to reflect the intrinsic correlation between control parameters during sampling. In each optimization step, the reference controller is replaced by the best controller found in the previous step, while the covariance matrix is updated based on a set of good controllers.

The details of CMA are explained below. In each optimization step, Ω controllers are sampled with normal distribution:

$$x = x_{ref} + \sigma N(0, C) \quad (1)$$

where x_{ref} is the reference controller, σ is the standard derivation, and $N(0, C)$ is a zero mean multivariate normal distribution with the covariance matrix C . σ controls the step size of each optimization step and is set to 0.003 in our system. The samples are then evaluated using the objective function described in Section 3.4.

Based on the evaluation of the sample controllers, the system updates the covariance matrix C and the reference controller x_{ref} . Considering the best λ of the samples, the new covariance matrix is calculated by

$$C' = \frac{1}{\lambda} \sum_{i=1}^{\lambda} (x_i - x_{ref})(x_i - x_{ref})^T \quad (2)$$

where x_i represents a set containing the best λ samples, and we set $\lambda = \Omega \times 25\%$. The new reference controller is replaced by the sample with the highest score, x_{best} :

$$x'_{ref} = x_{best} \quad (3)$$

Note that in the original CMA implementation, x'_{ref} is updated as a weighted sum of the best λ samples. However, this implementation performs sub-optimally in our system because it implies interpolation in the irregular controller space.

The ellipses around the controller from x_0 to x_5 in Figure 2 represent their respective sampling areas. The initial covariance matrix at x_0 is set as uniform sampling in all dimensions. Then, the covariance matrix evolves based on the best samples. As a result, the sampling area deforms to fit in the stable area of the controller space, and hence optimization becomes more effective in the later steps.

Rank μ Update Here, we explain how to speed up the CMA using one of its variant called rank μ update [6].

In theory, we can speed up the optimization by using fewer samples (i.e. smaller Ω). In practice, a small number of samples usually results in an unreliable covariance matrix. This is because the samples drawn are less likely to cover a large area in the controller space.

Rank μ update is an algorithm to estimate the covariance matrix under a system with small number of samples. Apart from considering the samples in the previous step (Equation 2), we calculate the covariance matrix with respect to the matrices used in all the past steps:

$$C' = \frac{1}{S} \left(\sum_{s=1}^{S-1} C_s + \frac{1}{\lambda} \sum_{i=1}^{\lambda} (x_i - x_{ref})(x_i - x_{ref})^T \right) \quad (4)$$

where C_s denotes the covariance matrix used at step s , S represent the total number of steps, and the rest of the parameters are defined as Equation 2. By this way, the matrix is updated with respect to all previous steps, thus eliminating the risk of an unreliable covariance matrix.

Noise Induction Physical systems are usually noisy due to the randomness in contact points and control torque error, and hence the controller measurement fluctuates across simulations. For example, when the foot of the character touches the floor, a small error at the ankle orientation will result in a change in the center of pressure, and hence affect the overall balance of the character hugely.

To evaluate the controllers accurately, we simulate multiple trials and evaluate the average fitness for each of them. In each trial, we introduce a small adjustment on the initial posture and a small amount of control torque error to simulate the noise [21]. The final evaluation considering all the trials is more representative. Empirically, we found that 100 trials give a good balance between evaluation accuracy and computational cost.

3.4 Objective Function

In this section, we explain the objective function we used to evaluate the fitness of the controllers. Apart from traditional terms that consider stability, motion style and energy [20], we add an extra term to minimize the change in controlling parameters across optimization steps. The objective function is defined as a weighted sum of the following terms.

Controller Difference We minimize the difference between the initial controller and the current controller described in Section 3.2. This ensures that the two controllers are close in the controller space, and hence results in stable control when we synthesize new controllers by interpolating them.

$$E_{difference} = - \sum_i |x_{initial}^i - x^i| \quad (5)$$

$$i \in \{\text{angles, gains}\}$$

where $x_{initial}^i$ and x^i represent the i^{th} controller parameter of the initial and the current controller, respectively. The control parameter consists of target angles and gains for all keyframes.

Stability Stability is defined by the duration a character can walk without falling. We maximize the mean duration of walking from a number of simulations, and minimize their variance:

$$E_{stability_mean} = \text{mean}(T) \quad (6)$$

$$E_{stability_variance} = -\text{variance}(T)$$

where T is the time the character can walk. The maximum cut-off duration is set 20 seconds, which means that we terminate the simulation there assuming the gait is sufficient stable.

Motion Style To construct a locomotion controller that produces walking motion similar to humans, we compare the simulated cycle with the captured human data:

$$E_{style} = -\text{mean}(P - P_c) - \text{mean}(d - d_c) - \text{mean}(\Theta - \Theta_c) \quad (7)$$

where P is the position of the joints, d is the distance the body have travelled, and Θ is the orientation of the body around the vertical axis in the simulated motion, and P_c , d_c , Θ_c are the corresponding values in the captured motion.

Gain To avoid high gains that results in stiff movements, we also minimize the gain during optimization:

$$E_{gain} = - \sum_i x^i \quad (8)$$

$$i \in \{\text{gains}\}$$

where x^i represent the i^{th} gain value.

3.5 Landmark Isomap

In this section, we first give a brief explanation about the classical Isomap and the enhanced landmark Isomap. Then, we explain how we implement landmark Isomap in our system, and discuss different parameters that affect the result.

Isomap [16] is a well-known non-linear dimensionality reduction method. It uses local metric information to learn the underlying global geometry of a data set. More specifically, it creates a connectivity map for the dataset using K nearest neighbors. Each sample is represented by a node in the map, and each pair of neighbors is connected with an edge. Based on the local distance, Isomap computes a distance matrix containing the shortest distance between any two nodes using Dijkstra’s algorithm. Finally, multidimensional scaling (MDS) [24] is applied to calculate the eigenvectors of the distance matrix, hence reducing its dimensions.

Landmark Isomap [3] is a speed-up version of Isomap. The two bottlenecks of the classical Isomap are the computation of the distance matrix by Dijkstra’s algorithm ($O(KN^2 \log N)$) and apply MDS on it ($O(N^3)$), where N is the number of samples and K is the neighbor size. Landmark Isomap speeds up the system by designating n of the data points to be landmark points, in which $n < N$. Pairwise distances are only computed between landmark and other samples ($O(KnN \log N)$). Due to the use of a smaller distance matrix, MDS is a lot faster ($O(n^2N)$).

In our system, we collect all the stable controller samples which were computed during the CMA optimization to compute the low dimensional latent space. Sampling only the optimal controllers will fail to create a representative low dimensional space, since the controllers are relatively sparse in the high dimensional space. Instead, we sample the controllers that have a good score in every optimization step. With the large number of good controllers, the computed low dimensional manifold will cover a large area of stable region.

For landmark Isomap, we need to designate samples to be landmarks. Ideally, the landmark should be distributed evenly among the samples, such that the error of the distance matrix is minimized. In our system, we randomly assigned 20% of the samples in each optimization step as landmarks. Since samples within each optimization step are similar to each other, this approach ensures that each sample is close to at least one landmark.

One important issue is the selection of the value K of the K nearest neighbor algorithm used for constructing the connectivity map. This heavily affects the quality of the final result. Large K can result in “short-circuit errors”, which means non-similar samples getting classified as neighbors. Small K can result in too many disconnected maps. We set K to be the average number of samples selected in each optimization step, as the samples within one step are usually similar.

Run-time Controller Interpolation During run-time, we interpolate the controllers in the low dimensional space. We first observe the task parameter during run-time, and select the two optimal controllers for the task value immediately below and above the observed value. Then, we linearly interpolate the two controllers in the low dimensional space, and inversely project the interpolated controller into the high dimensional space for character control.

Linear interpolation along the low dimensional manifold theoretically results in a more stable controller than that produced by interpolation in the full space, because the former is constructed from samples of stable controllers.

4 Experimental Results

In this section, we describe our experiments using a pushing controller. We use Open Dynamics Engine [15] for the simulator.

We apply continuous optimization to create a set of controllers that can counteract force continuously applied at the torso from the front and the back. The optimization starts with no pushing force. Whenever an optimal controller is found, we increase the force and apply the optimization again. We generate a set of controllers that can cope with the force from $0N$ up to $130N$ with the step size of $2N$. We computed 66 optimal controllers and sampled 2822 sub-optimal controllers. The whole process takes around 20 hours using an 8 core system (Duo Intel Xeon W5590).

We computed a low dimensional manifold of stable controllers by applying landmark Isomap to the collected samples. We set K of the K nearest neighbor operation in Isomap to 10, which is the average number of samples in each CMA optimization step. We determine the dimensionality of the latent space using leave-one-out cross validation. In our pushing motion experiment, we find that the reconstruction error does not increase significantly if the number of dimension is 5 or more. Therefore, we set the reduced dimension to 5.

During run-time, we observe the force applied to the character, and interpolate the optimal controllers in the low dimensional manifold. Since the interpolation and the forward dynamics simulation are fast, our system runs in real-time even with a slow system (Intel Core 2 Duo P8600).

We experiment on the stability of the character by changing the amount of external force applied during run-time (Figure 3). The system obtains the controller to deal with the force by interpolation. We observe that when the amount of pushing force increases, the character becomes stiffer and bend the upper body against the direction of the force for counteracting it. We repeated this experiment for 200 times with different variations of external forces, and examined that the character could walk without falling for more than 98% of the simulations.

We conduct experiments to simulate different number of characters pushing a physical object (Figure 4). The dynamic friction of the object is set to 5% of its velocity. We evaluate the force transferred from the object to the character using force sensors provided by the Open Dynamics Engine, and update the controller according to the value. When two or more characters are pushing the object cooperatively, the external force turns irregular, because the force exerted by one character indirectly affects those of the others. Still, our characters can stably walk by our controller.

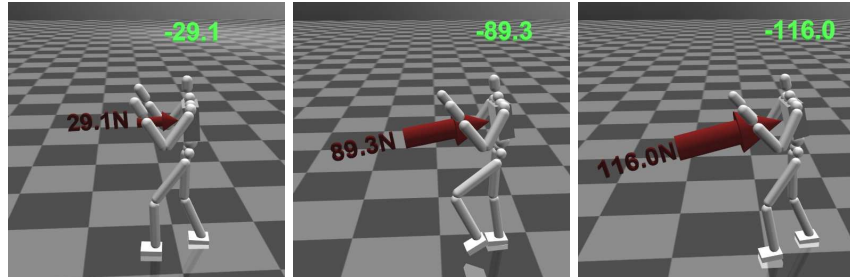


Fig. 3. The character applies the appropriate interpolated controller to counteract the pushing force. The red arrow represents the pushing force, while the green text represents the force that the controller can cope with.

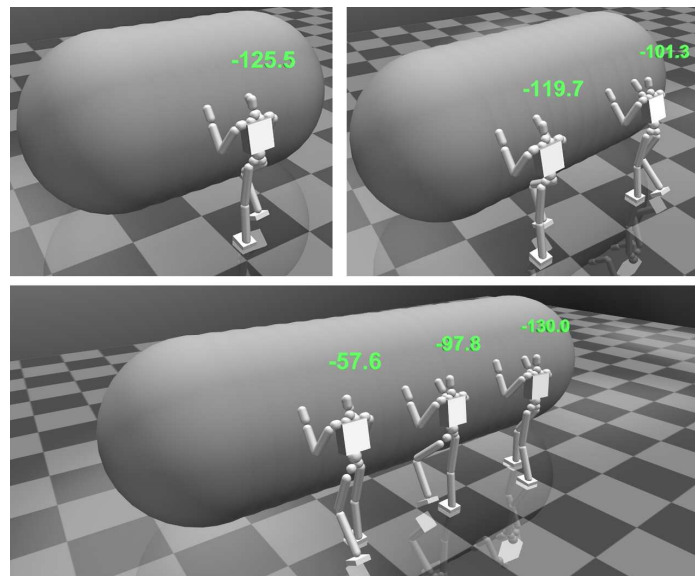


Fig. 4. Characters cooperatively push a large object. The green text represents the force that the controller can cope with.

5 Discussions

In this paper, we explained our method to automatically construct a low dimensional manifold for stable gait control from a rich set of samples obtained during continuous CMA optimization. The low dimensional manifold represents the stable area in the full controller space, and hence is capable of synthesizing stable controllers in the high dimensional space.

Constructing a reliable low dimensional manifold provides an important tool in dynamics control. Apart from controller interpolation shown in this paper, it is possible to apply run-time optimization in the reduced space [8, 11]. Since the dimensionality of the controller is greatly reduced, optimization can be done much more effectively.

Recent research suggests incorporating inverse pendulum model as an abstracted model of the body [17]. Our method provides a potentially better alternative for controller abstraction, since inverse pendulum model is only valid during single supporting stages, and its reliability depends on the posture of the character model during gait motion. Our low dimensional representation does not have these constraints.

Interaction among physically-controlled characters is a tough problem due to the instability of the controllers. One of our future directions is to simulate physical interactions among multiple characters.

References

- [1] Armstrong, H.G.: Anthropometry and mass distribution for human analogues. volume 1. military male aviators (1988)
- [2] Coros, S., Beaudoin, P., van de Panne, M.: Generalized biped walking control. In: SIGGRAPH '10: ACM SIGGRAPH 2010 papers. pp. 1–9. ACM, New York, NY, USA (2010)
- [3] De Silva, V., Tenenbaum, J.B.: Global versus local methods in nonlinear dimensionality reduction. In: Advances in Neural Information Processing Systems 15. vol. 15, pp. 705–712 (2003)
- [4] Grochow, K., Martin, S.L., Hertzmann, A., Popović, Z.: Style-based inverse kinematics. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers. pp. 522–531. ACM, New York, NY, USA (2004)
- [5] Hansen, N.: The cma evolution strategy: A comparing review. In: Towards a New Evolutionary Computation, Studies in Fuzziness and Soft Computing, vol. 192, pp. 75–102. Springer Berlin / Heidelberg (2006)
- [6] Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evol. Comput.* 11(1), 1–18 (2003)
- [7] Kwon, T., Hodgins, J.: Control systems for human running using an inverted pendulum model and a reference motion capture sequence. In: SCA '10: Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on Computer animation. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2010)
- [8] de Lasa, M., Mordatch, I., Hertzmann, A.: Feature-based locomotion controllers. *ACM Trans. Graph.* 29(4), 1–10 (2010)

- [9] Lawrence, N.D.: Gaussian process latent variable models for visualisation of high dimensional data. In: In NIPS. p. 2004 (2004)
- [10] Lee, Y., Kim, S., Lee, J.: Data-driven biped control. *ACM Trans. Graph.* 29(4), 1–8 (2010)
- [11] Mordatch, I., de Lasa, M., Hertzmann, A.: Robust physics-based locomotion using low-dimensional planning. *ACM Trans. Graph.* 29(4), 1–8 (2010)
- [12] Pratt, J.E., Tedrake, R.: Velocity-based stability margins for fast bipedal walking. vol. 340/2006, pp. 299–324 (2006)
- [13] Shin, H.J., Lee, J.: Motion synthesis and editing in low-dimensional spaces: Research articles. *Comput. Animat. Virtual Worlds* 17(3-4), 219–227 (2006)
- [14] da Silva, M., Abe, Y., Popović, J.: Interactive simulation of stylized human locomotion. In: SIGGRAPH '08: ACM SIGGRAPH 2008 papers. pp. 1–10. ACM, New York, NY, USA (2008)
- [15] Smith, R.: Open dynamics engine, <http://www.ode.org/>
- [16] Tenenbaum, J.B., Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. *Science* (5500), 2319–2323 (December)
- [17] Tsai, Y.Y., Lin, W.C., Cheng, K.B., Lee, J., Lee, T.Y.: Real-time physics-based 3d biped character animation using an inverted pendulum model. *IEEE Transactions on Visualization and Computer Graphics* (2009)
- [18] Van De Panne, M., Lamouret, A.: Guided optimization for balanced locomotion. In: Terzopoulos, D., Thalmann, D. (eds.) 6th Eurographics Workshop on Animation and Simulation, Computer Animation and Simulation, September, 1995. pp. 165–177. Eurographics, Springer, Maastricht, Pays-Bas (1995)
- [19] Wampler, K., Popović, Z.: Optimal gait and form for animal locomotion. *ACM Trans. Graph.* 28(3), 1–8 (2009)
- [20] Wang, J.M., Fleet, D.J., Hertzmann, A.: Optimizing walking controllers. In: SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers. pp. 1–8. ACM, New York, NY, USA (2009)
- [21] Wang, J.M., Fleet, D.J., Hertzmann, A.: Optimizing walking controllers for uncertain inputs and environments. *ACM Trans. Graph.* 29(4), 1–8 (2010)
- [22] Yin, K., Coros, S., Beaudoin, P., van de Panne, M.: Continuation methods for adapting simulated skills. In: SIGGRAPH '08: ACM SIGGRAPH 2008 papers. pp. 1–7. ACM, New York, NY, USA (2008)
- [23] Yin, K., Loken, K., van de Panne, M.: Simbicon: Simple biped locomotion control. *ACM Trans. Graph.* 26(3), Article 105 (2007)
- [24] Young, F.W.: Multidimensional scaling. In: Kotz-Johnson Encyclopedia of Statistical Sciences, vol. 5. John Wiley & Sons, Inc. (1985)