Adaptive Gaussian Mixture Trajectory Model for Physical Model Control using Motion Capture Data

Erik Herrmann
German Research Center for Artificial Intelligence (DFKI) & Saarland University, Saarbrücken, Germany
erik.herrmann@dfki.de

Han Du
German Research Center for Artificial Intelligence (DFKI) & Saarland University, Saarbrücken, Germany

Noshaba Cheema
German Research Center for Artificial Intelligence (DFKI), Saarland University & Max-Planck Institute for Informatics, Saarbrücken, Germany

Janis Sprenger
German Research Center for Artificial Intelligence (DFKI) & Saarland University, Saarbrücken, Germany

Somayeh Hosseini
German Research Center for Artificial Intelligence (DFKI) & Saarland University, Saarbrücken, Germany

Klaus Fischer
German Research Center for Artificial Intelligence (DFKI) & Saarland University, Saarbrücken, Germany

Philipp Slusallek
German Research Center for Artificial Intelligence (DFKI) & Saarland University, Saarbrücken, Germany

ABSTRACT

To enable the physically correct simulation of the interaction of a 3D character with its environment the internal joint forces of a physical model of the character need to be estimated. Recently, derivative-free sampling-based optimization methods, which treat the objective function as a black box, have shown great results for finding control signals for articulated figures in physics simulations. We present a novel sampling-based approach for the reconstruction of control signals for a rigid body model based on motion capture data that combines ideas of previous approaches. The algorithm optimizes control trajectories along a sliding window using the Covariance Matrix Adaption Evolution Strategy. The sampling distribution is represented as a mixture model with a dynamically selected number of clusters based on the variation detected in the samples. During the optimization we keep track of multiple states which enables the exploration of multiple paths. We evaluate the algorithm for the task of motion capture following using figures that were automatically generated from 3D character models.

CCS CONCEPTS

• **Computing methodologies → Animation**; **Physical simulation**; *Motion capture*.

KEYWORDS

Simulation, Motion Capture, Sampling, Optimization

ACM Reference Format:
Erik Herrmann, Han Du, Noshaba Cheema, Janis Sprenger, Somayeh Hosseini, Klaus Fischer, and Philipp Slusallek. 2019. Adaptive Gaussian Mixture Trajectory Model for Physical Model Control using Motion Capture Data. In *Symposium on Interactive 3D Graphics and Games (I3D '19), May 21–23, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3306131.3317027

1 INTRODUCTION

Interactive applications involving human characters often require the believable interaction with the environment based on physics simulations. However, controlling the underactuated articulated human body model in a simulation is a difficult problem. Recently, derivative-free optimization methods based on the evaluation of trajectories sampled from a distribution have been successfully applied to this problem by Rajamäki and Hämäläinen [2017]. These methods iteratively use previous examples to guide the exploration in the sample space to find optimal parameters for the model.

The problem of controlling a figure can be simplified by using motion capture data of natural motion as a reference. The sampling algorithm introduced by Liu et al. [2015] using Covariance Matrix Adaption Evolution Strategy (CMA-ES) [Hansen 2006] in combination with a sliding window has shown good results for the problem of reconstructing the control signals for an articulated figure model in a physics simulation based on motion capture data. CMA-ES is a randomized black box optimization algorithm for non-linear functions. In each iteration, actions are sampled from a Gaussian distribution and scored based on a cost function. The best parameters are then used as the parents of the next generation by updating the mean and covariance of the distribution. However, solution spaces which have a complex and asymmetric shape cannot be approximated by a single Gaussian [Calinon et al. 2012]. Furthermore, keeping only a single state when moving the sliding window forward can more likely trap the distribution into a local minimum. Additionally, the method presented by Liu et al. [2015] works based on Proportional Derivative (PD) controllers which require additional tweaking of parameters. Furthermore, PD-controllers require high simulation update rates to run smoothly. On the other hand, the method introduced by Rajamäki and Hämäläinen [2017] based on Monte Carlo Tree Search uses angular velocity constraints of the physics engine to control arbitrary articulated figures. Angular velocity constraints that are solved as part of the other constraints of the physics engine enable lower update rates and need less parameters to be tweaked for different figures (cf. [Tsai et al. 2010]).

We simplify the method of Liu et al. [2015] for open loop control reconstruction of a physical model based on a sample representation as trajectories of angular velocities as used by [Rajamäki and Hämäläinen 2017]. This enables the control of different figure models also at a low simulation update rate without parameter tuning. Additionally, we model the sampling distribution using a Gaussian mixture trajectory model that we optimize along a sliding window using CMA-ES. The algorithm keeps a separate state for each cluster to prevent ending up in a local minimum. In each update we prune and split clusters of the distribution based on the number of elite samples they have produced. The resulting tree structure enables the backtracking of the best leaf to the root.

We give a brief review of related work on physics-based motion synthesis with focus on sampling-based methods in Section 2. Our method is described in Section 3. An evaluation of our method for following motion capture data by automatically generated articulated figures is described in Section 4. Section 5 gives a brief conclusion.

2 RELATED WORK

The control of an articulated figure in a physical simulation based on motion capture data is a well studied problem with different solutions including analytical controllers, numerical optimization, random sampling and reinforcement learning. Gleicher et al. [1997] apply space-time constraints to retarget motion to a physical model which enables further editing. Popivić and Witkin [1999] present an improved method for space-time constraint-based motion retargeting, which also allows for complex editing by mapping to a simplified intermediate representation of the model. Muico et al. [2009] introduce an online controller that makes use of a reference motion that is pre-processed offline using space-time constraints. The motion is then adapted online to environment constraints by solving a linear complementary problem in a small window. De Lasa et al. [2010] present a framework for optimization-based motion synthesis where multiple objectives are combined based on a state machine and solved at runtime using a prioritized QP-solver.

Zordan et al. [2005] retarget motion capture data to a physical model using PD-controllers and apply an analytical balance controller based on virtual forces to modify the leg angles. Yin et al. [2007] present the SIMBICON framework that follows reference poses from a state machine using PD-controllers and also modifies the angles of the hips and legs to keep balance. Coros et al. [2010] improve upon the SIMBICON controller by integrating a balance controller based on an inverted pendulum model. Tsai et al. [2010] apply angular velocity constraints of the physics engine to control the model but also apply an inverted pendulum model to keep balance. Lee et al. [2010] present an online balance controller that modulates streams of reference motion data to keep balance using PD-controller tracking. Geijtenbeek et al. [2012] develop an online balance controller using virtual forces and apply CMA-ES offline to optimize the PD-controller gains and weights for different characters. For an overview of analytical methods and methods based on optimization we refer the reader to a survey by Geijtenbeek and Pronost [2012].

Liu et al. [2010] introduce a sampling framework to reconstruct open loop control sequences to follow motion capture data. The authors improve their original sampling algorithm for trajectory following by applying CMA-ES updates and a sliding window [Liu et al. 2015]. Liu et al. [2016] also trained a graph of linear feedback policies by reconstructing random walks through a motion graph via their sampling-based optimization algorithm. Hämäläinen et al. [2014] developed an algorithm based on random sampling that adapted a distribution tree from which new actions were sampled and combined it with an approximate nearest neighbor search model. Additionally, Hämäläinen et al. [2015] presented the Control Particle Belief Propagation (C-PBP) method, a probabilistic framework based on particle belief propagation along trajectories for articulated figure control. Recently, Rajamäki and Hämäläinen [2017; 2018] have presented an improved and simplified sampling framework based on Monte Carlo Tree Search that combines a particle filter with a neural network model and an approximate nearest neighbor search model to synthesize new motions based on constraints at runtime. Naderi et al. [2017] present a planning algorithm for climbing motions in a physics simulation that uses CMA-ES or alternatively C-PBP [Hämäläinen et al. 2015] to control the physical model.

The sampling based optimization algorithm CMA-ES [Hansen 2006] has also been applied for the training of policy models in a physics simulation by Ding et al. [2015]. Furthermore, reinforcement learning based on policy gradient methods [Schulman et al. 2017] that explore the parameter space via random sampling have been successfully applied to locomotion generation in physics simulations by Peng et al. [2017]. The authors further improved their approach to accurately follow motion capture data [Peng et al. 2018a] and reference video data [Peng et al. 2018b]. Similarly, Ho and Ermon [2016] apply generative adversarial imitation learning to follow motion capture data. There is a lot of recent advances regarding controller model learning using reinforcement and imitation learning in physics environments. However, this is not the focus of this work.

We apply a sampling-based approach which represents the problem as a black box and build upon the method introduced by Liu et al. [2015] for open loop control reconstruction. Liu et al. sample correction torques from a Gaussian distribution to modify an initial guess to successfully follow reference motion capture data using a physical model. They adapt the distribution using CMA-ES based on samples of the previous iteration. The optimization is applied along a sliding window in the temporal domain to make sure previous parts of the motion have a sufficient quality before optimizing later parts of the motion. The method makes use of PD-controllers which require tuning for specific tasks and adaption for the influence of gravity. We simplify the sampling according to Rajamäki and Hämäläinen [2017] by sampling trajectories of angular velocity constraints as control signals of the physical model instead of joint torques. Angular velocity constraints can be directly solved as part of all constraints of the physics engine in a single linear complementary problem (cf. [Tsai et al. 2010]). It leads to higher stability at lower simulation update rates and reduces the number of parameters that need to be tweaked for each articulated figure compared to PD-controllers. Additionally, we propose an improvement of the sampling distribution representation as a Gaussian mixture model to better capture the variation in the samples.

3 ADAPTIVE GAUSSIAN MIXTURE TRAJECTORY MODEL

We want to find a sequence of control signal frames $C(t) = (a_0, ..., a_T)$ for an articulated figure model to follow a reference motion capture sequence $Q(t) = (q_0, ..., q_T)$. Each control signal frame a_t consists of a list of angular velocity constraints for each degree of freedom of the model. The reference frames q_t are a list of quaternions for each joint of the model that can be used to calculate the pose of each body via forward kinematics.

We apply an iterative sampling algorithm over a sliding window to find the control sequence for an articulated figure similar to [Liu et al. 2015]. To better represent the sampling space we adapt a Gaussian mixture model using CMA-ES instead of a single Gaussian. Similar to previous work on sampling-based optimization, the method can be easily parallelized.

3.1 Sampling Algorithm

The motion is represented as a mixture model trajectory with T discrete time steps.

$$M(t) = (m_0, ..., m_T) \tag{1}$$

where each element m_t is a continuous distribution representing the possible actions at time step t by a Gaussian mixture model.

$$a_t \sim m_t$$

$$m_t = \sum_i^k w_i N(\mu_t^i, \Sigma_t^i)$$

Here μ_t^i represents the guess of the control at time t of cluster i, Σ_t^i is a covariance matrix and w_i is a weight with a value between 0 and 1. We construct the control signal trajectory $C(t)$ to follow the motion by iteratively sampling control signal frames for each time step from the distribution trajectory $M(t)$ along a sliding window.

The sampling algorithm is described in Algorithm 1. In each iteration, we use a Monte Carlo Tree Search (MCTS) according to [Rajamäki and Hämäläinen 2017] to sample control trajectories from the distribution $M(t)$ inside of a sliding window. The state s_t resulting from a control signal $a_t \sim m_t$ is evaluated based on the cost function described in Section 3.2. The sampling function returns as soon as the end of the current sampling window was reached or all samples lost control before the end was reached. When a sample loses control the state of an active sample is copied to use the full sampling budget in each step. Furthermore, we filter the samples based on their cost according to [Liu et al. 2015] and keep only a certain percentage p_{filter} of the best samples.

To check whether a trajectory has lost control we check three conditions. Firstly the angle between the body up axis and the global up axis must not exceed the maximum angle in the reference data. Secondly, only the feet may touch the ground, and lastly, there should not be any self-collision between bodies of the legs. Similar to [Rajamäki and Hämäläinen 2017] we multiply the Gaussians of the distribution of the current step m_t with the Gaussians of the previous step m_{t-1} before sampling an action a_t as shown in Equation 2.

Algorithm 1: Sampling algorithm for motion control reconstruction

```
Input  : $C_{ref}, n_{samples}, n_{save}, p_{filter}, n_{window}, k_{max}, n_{iter}, dt$
Output : $\hat{C}$ reconstructed control signal
$M \leftarrow init\_distribution(C_{ref}, k = 1)$
$start, best\_end \leftarrow 0, 0$
$iter\_counter \leftarrow init\_iter\_counter()$
$R \leftarrow init\_tree\_root()$
while $start < length(M)$ do
    $end \leftarrow best\_end + n_{window}$
    $T \leftarrow MCTS(R, M, n_{samples}, p_{filter}, start, end, dt)$
    $S \leftarrow extract\_latest\_trajectories(T, n_{save})$
    $S \leftarrow sort\_by\_avg\_cost(S)$
    if $length(first(S)) == 0$ then
        break
    end
    $best\_end \leftarrow start + length(first(S))$
    $med\_end \leftarrow start + length(median(S))$
    $iter\_counter \leftarrow update\_iter\_counter(med\_end)$
    $M \leftarrow update\_distribution(M, S, start, end, k_{max})$
    if $iter\_counter(med\_end) > n_{iter}$ then
        $k \leftarrow$ get_active_clusters(M(start))
        $R, start \leftarrow move\_window(R, S, start, med\_end, k)$
    end
end
$best_k \leftarrow argmin(avg\_cost(R))$
$\hat{C} \leftarrow extract\_trajectory(R, best_k)$
return $\hat{C}$
```

$$a_t \sim N(\mu_t^i, \Sigma_t^i) \times N(\mu_{t-1}^i, \Sigma_{t-1}^i) \tag{2}$$

This way the variation between steps is reduced and the resulting motion is smoother. To make up for the loss of variation we scale the resulting covariance matrix with a constant factor.

We use the n_{save} latest trajectories generated by the MCTS algorithm to update the means and covariance matrices of the distribution trajectory $M(t)$. During the update of the distribution we prune and split clusters based on the number of elite samples they produced. In the following we refer to the best samples of an iteration as elite samples. The distribution adaption is described in Section 3.3.

Similar to [Rajamäki and Hämäläinen 2017] we keep multiple states and construct a control tree R. Each node in the tree corresponds to one cluster of the distribution at the current start step of the window. The sliding window is moved and the control tree R is extended based on the number of times a step has been reached by the median sample without losing control. We use the best samples for the k active clusters to update the control tree R. The reconstruction stops when the start step of the sliding window has reached the end of the reference motion or the MCTS algorithm returns only trajectories of 0 length. After the optimization has finished the trajectory extracted from the best leaf of the control tree R is returned as final result.

The search window can also be reset multiple times with the previous result as initial guess to further improve the reconstruction until the cost converges or a maximum number of reconstruction attempts has been reached.

Algorithm 2: Simplified Monte Carlo Tree Search (MCTS() in Algorithm 1)

```
Input  : $R, M, n_{samples}, p_{filter}, start, end, dt$
Output : $T$ trial control tree
$T = copy(R, start)$
for $t = start; t < end; t++$ do
    for $n = 0; n < n_{samples}; n++$ do
        leaf ← select_leaf(T, t)
        $p\_cluster\_idx \leftarrow leaf.cluster\_idx$
        $cluster\_idx \leftarrow sample\_child(M, t-1, p\_cluster\_idx)$
        action ← $sample(M, t, cluster\_idx)$
        state ← step_sim(leaf.state, action, dt)
        cost ← $E_{state}(state)$
        if not lost_control(state) then
            $leaf.extend(cost, action, state)$
        end
    end
    $T \leftarrow filter(T, p_{filter})$
end
return $T$
```

3.2 Cost Function

We evaluate the fitness of the state s_t resulting from a control sample a_t of a trajectory C using the cost function shown in Equation 3, which is based on the cost function described by [Liu et al. 2015].

$$E_{state}(s_t) = w_p E_p(s_t) + w_r E_r(s_t) + w_b E_b(s_t) \tag{3}$$

The weighted cost terms for the root E_r, pose E_p and balance E_b are described in the following equations.

Equation 4 shows the root term which calculates the error of the root transformation in the global coordinate system.

$$E_r = d_q(q_{root}, \hat{q}_{root}) + d_v(p_{root}, \hat{p}_{root}) \tag{4}$$

Here p_{root} and q_{root} are the position and orientation of the root body. The function $d_q()$ is the rotation distance measure and $d_v()$ is the L2-norm. The reference root position and orientation is given by $\hat{p}_{root}$ and $\hat{q}_{root}$, respectively.

Equation 5 shows the pose term, which represents the point cloud distance of the figure pose and the target pose in the coordinate system relative to the root body.

$$E_p = \sum_{i=0}^{N_{bodies}} d_v(p_i, \hat{p}_i) \tag{5}$$

Here p_i is the position of a body in the coordinate system of the root body and $\hat{p}_i$ is the reference body position in the root coordinate system of the reference pose. Equation 6 shows the balance term which was introduced by [Liu et al. 2010].

$$E_b = \sum_{i=0}^{N_{bodies}} d_v(r_i, \hat{r}_i) \tag{6}$$

Here $r_i = (p_i - p_{COM})|_{y=0}$ is the vector from the body to the center of mass projected on the ground. This term penalizes moving the body parts too far away from the center of mass.

3.3 Mixture Model Adaption

In order to better model the variation, we represent the distribution using multiple clusters similar to [Calinon et al. 2012]. The distribution is modified in each iteration by applying the CMA-ES update step [Hansen 2006] separately on individual segments of the trajectory based on the best samples. An overview of the steps of the distribution adaption is given in Algorithm 3.

Algorithm 3: Mixture Model Adaption (update_distribution() in Algorithm 1)

```
Input  : $M, S, start, end, k_{max}$
Output : $M$ updated distribution
for $t = start; t < end; t++$ do
    $S_t \leftarrow sort\_by\_cost(S(t))$
    $M(t) \leftarrow prune\_clusters(M(t), S_t)$
    $M(t), S_t` \leftarrow split\_clusters(M(t), S_t, k_{max})$
    $n\_clusters$ ← get_active_clusters(M(t))
    for $k = 0; k < n\_clusters; k++$ do
        $M(t)(k) \leftarrow cma\_update(M(t)(k), S_t`(k))$
    end
end
return $M$
```

In the first step, we split and prune existing clusters of the distribution based on the number of elite samples that were sampled from them. These samples are ordered separately for each step t based on the cost value. The order is used to assign a weight to each sample with a logarithmic scale according to [Hansen 2006]. If the sum of the weights of the samples generated by a cluster is below a threshold, the cluster will be pruned. The remaining clusters are then split by applying the kMeans algorithm on the elite samples that they produced. The number of clusters of a segment m_t is automatically found by applying the kMeans algorithm recursively and selecting the best clustering based on the minimum Bayesian Information Criterion (BIC) score [Pelleg et al. 2000]. Due to a limited sampling budget, only a limited number of clusters can be kept track of in each iteration. Therefore, the clusters are ordered based on their weight and clusters with more samples will be split first, so that more clusters are available during the BIC evaluation.

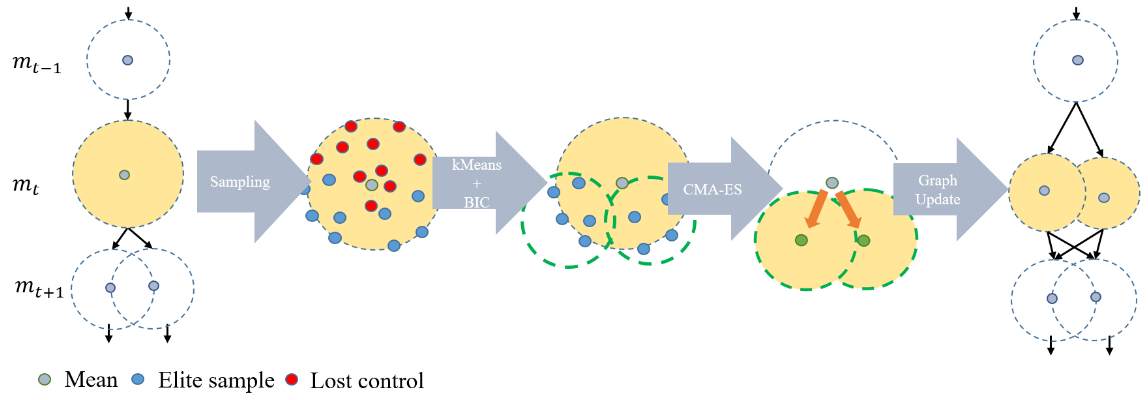

Figure 1: The steps of the Gaussian mixture model adaption for one segment m_t. The dashed circles represent a cluster in the distribution. Red points represent samples that lost control and blue points represent elite samples that successfully reached the end of the window. Clusters are split using kMeans into k clusters based on the minimum BIC score. The samples are then used to update the distribution. After the update of the distributions for each step, the graph structure is modified to allow transitions from the previous step and to the next step. During the sampling, a transition is randomly chosen based on the current cluster.

To better capture the variation of successful actions for each step, we include all samples in the update of each cluster of m_t. For this purpose we order the remaining samples, that are not associated with a cluster, after the associated samples. The cluster assignment therefore affects the order and thus the weight of each sample in the update of each cluster. The mean and the covariance matrix of the clusters are then updated based on the reweighted elite samples according to the standard CMA-ES update [Hansen 2006]. To apply the CMA-ES update we also keep track of the evolution path, step size and number of updates for each cluster. After the splitting of the clusters of the distribution for each step, we modify the graph structure to include transitions to and from the new clusters. In the next iteration of the sampling, one transition from step m_t to m_{t+1} will be randomly selected with equal probabiliy. An overview of the steps of the cluster splitting is given in Figure 1.

4 EXPERIMENTAL RESULTS

4.1 Experiment Setup

The algorithm was implemented based on the Open Dynamics Engine version 0.12 [Smith 2004] and makes use of modifications provided by [Rajamäki and Hämäläinen 2017] to enable reproducable simulations when states are copied between workers. The friction coefficient was set to 0.8. The simulation update rate was set to 30 FPS, which is possible due to the use of angular velocity constraints. The articulated figure body models used in the experiments were generated from reference character models. We used MakeHuman[1] version 0.1.0 to generate the characters and exported the models with CMU skeletons. The unit scale during the export was set to decimeter. Our figure generator uses a T-pose and a manually defined body radius as input to create axis aligned capsule bodies in the physics engine. The generated figures have 8 ball joints and 4 hinge joints resulting in 28 degrees of freedom. We apply a standard skeleton retargeting method [Monzani et al. 2000] for the retargeting between the kinematic skeleton and the figure joints. However, for the feet, we need to calculate an extra offset to make sure the bodies are aligned with the flat ground during the T-pose. The character models and corresponding figures used in the experiments are shown in Figure 2.

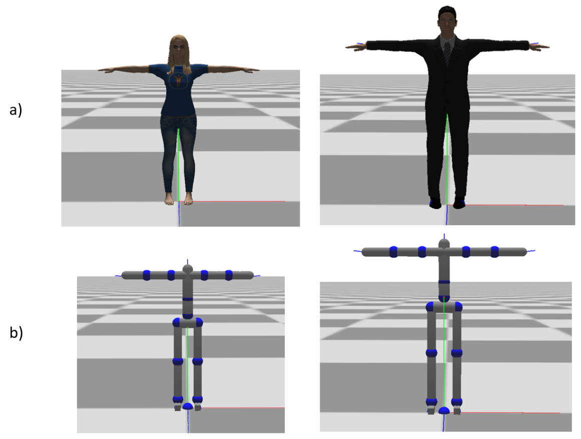

Figure 2: a) Female and male character models generated by MakeHuman. b) The corresponding articulated figures with 12 joints and 28 degrees of freedom used in the experiments.

For the sampling algorithm, we set an initial noise value of 0.8 radians for each joint, except for the shoulders, which were set to 0.2 radians, and the spine joints, which were set to 0.4 radians. The initial guess of the mean for the distribution is generated via finite difference. We have fixed the number of kMeans iterations to 1000. The sliding window size was set to 10 steps for the experiments. The weights for the root position, root orientation, pose and balance

[1]MakeHuman: http://www.makehumancommunity.org/

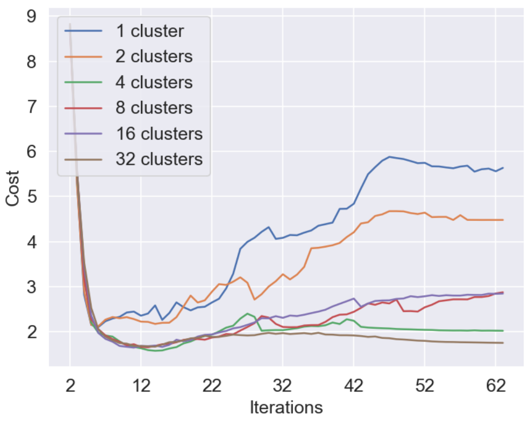

Figure 3: Comparison of the average cost of 5 reconstruction attempts using a different number of clusters.

term were set to 1, 1, 1 and 0.1, respectively. We set the threshold for the cluster pruning based on the weighted sum of elite samples to 0.01 and the constant for the scaling of the smoothed covariance matrix to 2. Smoothing was activated for all experiments. The learning rate of each cluster was adjusted based on the number of successfully reconstructed samples per frame according to [Hansen 2006]. The number of minimum iterations per frame was set to 10 and the sample filtering per frame to 60%. The experiments were run on a workstation with an Intel i7 4770k CPU and 32 GB RAM.

4.2 Evaluation

To determine the optimal number of clusters we first evaluate the effect of the number of clusters by retargeting a walk motion to the female model. The samples per frame were set to 4096 and the number of elite samples to 2048. Table 1 shows the comparison of the mean and the standard deviation of the average cost value of the best motion samples of 5 reconstruction attempts without window reset for a different number of clusters. A plot of the average cost during the reconstruction is shown in Figure 3.

Table 1: The effect of different numbers of clusters on 5 reconstruction attempts of a walk motion with 477 frames.

clusters	average cost	median steps	$\overline{time}$	$\overline{iter}$
1	5.634 ± 0.626	167	550.4s	62.0
2	4.479 ± 1.667	202	582.8s	62.0
4	2.021 ± 0.402	324	1256.8s	62.0
8	2.874 ± 1.231	356	1536.8s	62.0
16	2.846 ± 1.226	477	1862.4s	62.0
32	1.752 ± 0.094	477	2507.2s	62.0

The evaluation shows that the median number of successfully reconstructed steps is increased while the mean and standard deviation of the cost is reduced when more than one cluster is used. This means that the reliability of the reconstruction can be increased by applying CMA-ES on the mixture distribution compared to applying CMA-ES on a single Gaussian as used in the algorithm of [Liu et al. 2015]. Note that the larger average cost using 8 clusters compared to 4 can be explained by the higher number of reconstructed frames. The minimum mean and standard deviation of the cost was achieved using 32 clusters. However, the processing time increases as well with the number of clusters, therefore, we select 16 clusters for the other experiments. Note that the quality of the motion can be further increased by increasing the number of samples, by increasing the minimum iterations per frame or by resetting the sliding window multiple times.

The results of the reconstruction of a walk motion with 477 frames for the male and female figure is shown in Figure 6. The motions were generated using 16 clusters, 4096 samples and 1024 elite samples. The female motion was generated in 26 minutes and 24 seconds with an average frame cost of 1.850 and the male motion was generated in 27 minutes and 12 seconds with an average frame cost of 2.498. Figure 4 plots the average error of the samples during the optimization and Figure 5 shows the change of the cost of individual frames. The average error increases in the beginning as more frames are reconstructed but reduces over time, as soon as all frames are reconstructed after iteration 50. The costs can also be reduced by disabling the smoothing due to higher variation of the samples. However, the resulting motion contains noticeable

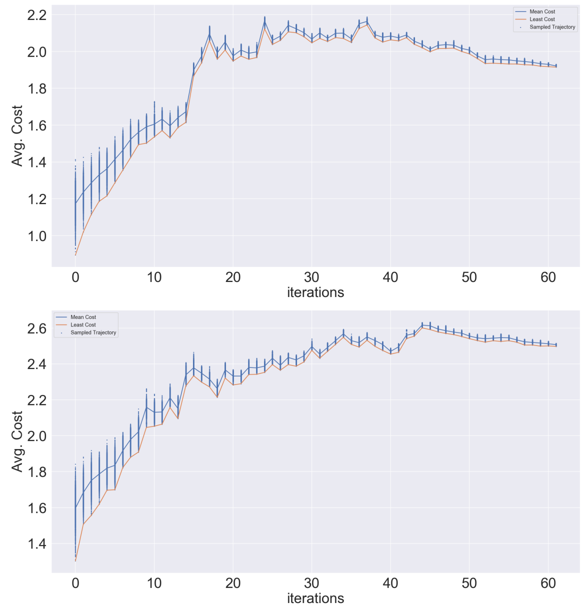

Figure 4: Change of the mean cost of samples during the reconstruction of a walk motion for the female model (top) and the male model (bottom) using 16 clusters. Each blue dot represents the mean cost of all frames of a sample including an error offset from the sliding window. The orange line represents the value of the best samples of each iteration and the light blue line represents the mean value of each iteration.

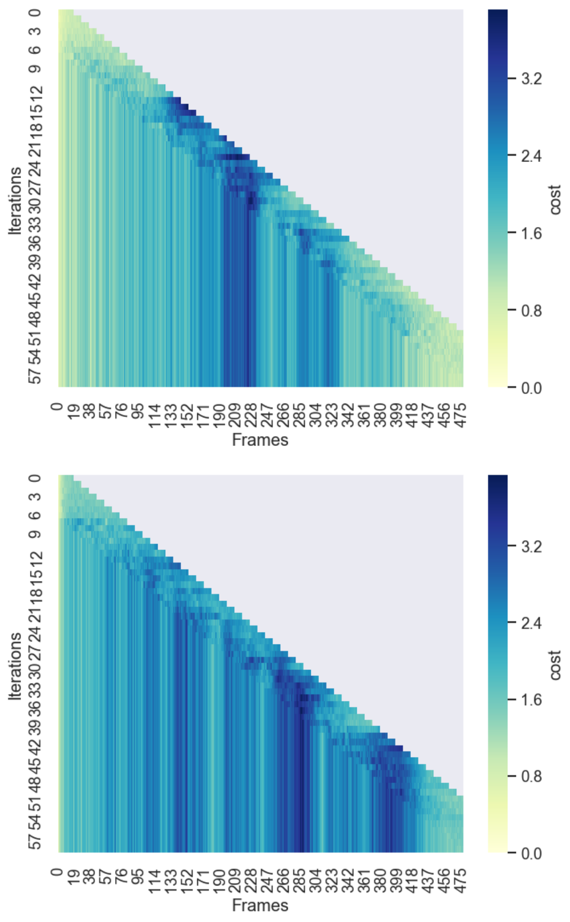

Figure 5: Change of the cost of frames during the reconstruction of a walk motion for the female model (top) and the male model (bottom) using 16 clusters. The color of each pixel represents the cost of a frame during a certain iteration. The optimized frames are restricted by a sliding window. The sliding window can be reset after it has reached the end to further improve the reconstruction.

jittering when smoothing is disabled. A faster generation with fewer samples is also possible, however it is less reliable.

Reference motions can also be adapted to changes in the underlying terrain without changing the cost function. An example of the reconstruction on randomly generated terrain is shown in Figure 7. The motion was reconstructed in 30 minutes and 9 seconds with an average cost of 3.002 using 16 clusters.

One limitation of our approach is the extra time needed for the splitting of the distribution by applying kMeans which we want to address in future work by parallelizing the splitting of different clusters. In general it is possible to find control signals that follow the reference motion capture data with a natural pose while keeping balance. However, the method can only be applied to reconstruct motion examples starting from an idle pose without external forces as it only optimizes the angular velocity constraints to produce

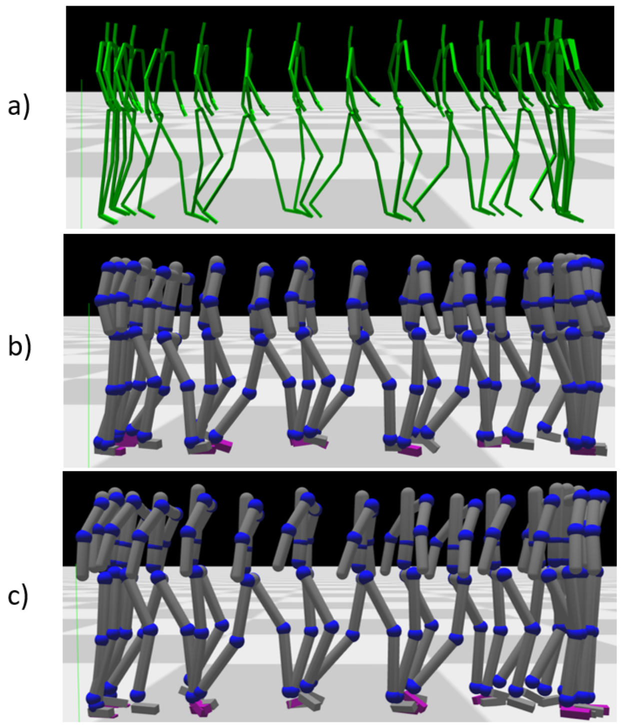

Figure 6: Walk motion a) reconstructed by the female b) and the male figure c) using 16 clusters. Bodies in contact with the ground are highlighted in purple.

joint torques. Different starting states, for example starting during a walk cycle, could be supported by also optimizing an external force applied on the first frame. The smoothing via the Gaussian multiplication is effective at reducing variation between frames, however it also thins out the distribution after multiple iterations, which can cause the algorithm to get stuck before reaching the end of the motion. We noticed in our experiments that the use of the mixture model counteracted this problem by increasing the variation and thus the reliability of the algorithm.

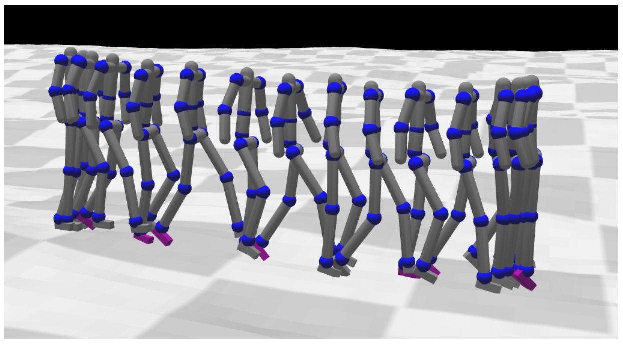

Figure 7: Walk motion reconstructed by the female figure on randomly generated terrain. Bodies in contact with the ground are highlighted in purple.

5 CONCLUSION

We have presented a sampling-based optimization method with application to open loop control reconstruction for articulated figures in a physics engine based on reference motion capture data. The method combines ideas of previous work to provide a simple motion reconstruction framework with little parameter tuning. Akin to Liu et al. [2015] we optimize a distribution of samples using CMA-ES over a sliding window. However, similar to Rajamäki and Hämäläinen [2017] we keep multiple states using a trajectory of mixture models and generate smooth control trajectories by multiplying Gaussians of consecutive steps before sampling. We apply the kMeans algorithm to split the distribution of samples and determine the number of Gaussians of the model automatically using the BIC score. The use of angular velocity constraints of the physics engine enables the retargeting of motions to figure models that were automatically generated from 3D models without additional parameter tweaking.

The experiments show that applying CMA-ES on a mixture model with multiple clusters and keeping multiple states when moving the sliding window forward can lead to more reliable results compared to using a single Gaussian.

We plan to use the algorithm in combination with reinforcement learning to optimize trajectories in the replay memory during the training of a policy model. Furthermore, we plan to evaluate the method on other problems such as collision free path finding. For this purpose we want to evaluate alternative methods for the distribution splitting.

ACKNOWLEDGMENTS

This work is funded by the German Federal Ministry of Education and Research (BMBF) through the projects Hybr-iT (grant no. 01IS16026A) and REACT (grant no. 01IW17003) as well as by the ITEA 3 project MOSIM (grant no. 01IS18060C).

REFERENCES

Sylvain Calinon, Affan Pervez, and Darwin G Caldwell. 2012. Multi-optima exploration with adaptive Gaussian mixture model. In *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*. IEEE, 1–6.

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2010. Generalized Biped Walking Control. *ACM Transctions on Graphics* 29, 4 (2010), Article 130.

Martin De Lasa, Igor Mordatch, and Aaron Hertzmann. 2010. Feature-based locomotion controllers. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 131.

Kai Ding, Libin Liu, Michiel van de Panne, and KangKang Yin. 2015. Learning Reduced-Order Feedback Policies for Motion Skills. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

Thomas Geijtenbeek and Nicolas Pronost. 2012. Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 2492–2515.

Michael Gleicher. 1997. Motion editing with spacetime constraints. In *Proceedings of the 1997 symposium on Interactive 3D graphics*. ACM, 139–ff.

Perttu Hämäläinen, Sebastian Eriksson, Esa Tanskanen, Ville Kyrki, and Jaakko Lehtinen. 2014. Online Motion Synthesis Using Sequential Monte Carlo. *ACM Trans. Graph.* 33, 4, Article 51 (July 2014), 12 pages. https://doi.org/10.1145/2601097.2601218

Perttu Hämäläinen, Joose Rajamäki, and C. Karen Liu. 2015. Online Control of Simulated Humanoids Using Particle Belief Propagation. *ACM Trans. Graph.* 34, 4, Article 81 (July 2015), 13 pages. https://doi.org/10.1145/2767002

Nikolaus Hansen. 2006. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation*. Springer, 75–102.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*. 4565–4573.

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-driven biped control. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 129.

Libin Liu, Michiel Van De Panne, and KangKang Yin. 2016. Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics (TOG)* 35, 3 (2016), 29.

Libin Liu, KangKang Yin, and Baining Guo. 2015. Improving Sampling-based Motion Control. *Computer Graphics Forum* 34, 2 (2015).

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based Contact-rich Motion Control. *ACM Transctions on Graphics* 29, 4 (2010), Article 128.

Jean-Sébastien Monzani, Paolo Baerlocher, Ronan Boulic, and Daniel Thalmann. 2000. Using an intermediate skeleton and inverse kinematics for motion retargeting. In *Computer Graphics Forum*, Vol. 19. Wiley Online Library, 11–19.

Uldarico Muico, Yongjoon Lee, Jovan Popović, and Zoran Popović. 2009. Contact-aware Nonlinear Control of Dynamic Characters. *ACM Transactions on Graphics* 28, 3 (2009).

Kourosh Naderi, Joose Rajamäki, and Perttu Hämäläinen. 2017. Discovering and synthesizing humanoid climbing movements. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 43.

Dan Pelleg, Andrew W Moore, et al. 2000. X-means: Extending k-means with efficient estimation of the number of clusters.. In *Icml*, Vol. 1. 727–734.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018a. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Transactions on Graphics (Proc. SIGGRAPH 2018 - to appear)* 37, 4 (2018).

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 41.

Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. 2018b. SFV: reinforcement learning of physical skills from videos. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 178.

Zoran Popović and Andrew Witkin. 1999. Physically based motion transformation. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 11–20.

Joose Rajamäki and Perttu Hämäläinen. 2017. Augmenting sampling based controllers with machine learning. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 11.

Joose Julius Rajamäki and Perttu Hämäläinen. 2018. Continuous control monte carlo tree search informed by multiple experts. *IEEE transactions on visualization and computer graphics* (2018).

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Russ Smith. 2004. The Open Dynamics Engine. http://www.ode.org/.

Yao-Yang Tsai, Wen-Chieh Lin, Kuangyou B Cheng, Jehee Lee, and Tong-Yee Lee. 2010. Real-time physics-based 3d biped character animation using an inverted pendulum model. *IEEE transactions on visualization and computer graphics* 16, 2 (2010), 325–337.

KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: Simple Biped Locomotion Control. *ACM Trans. Graph.* 26, 3 (2007), Article 105.

Victor Brian Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. 2005. Dynamic response for motion capture animation. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 697–701.