Smooth Conditional Transition Paths in Dynamical Gaussian Networks

Michał Matuszak[1], Jacek Miękisz[2], and Tomasz Schreiber*[1]

[1] Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Chopina 12/18, 87–100 Torun, Poland
{gruby, tomeks}@mat.umk.pl
[2] Institute of Applied Mathematics and Mechanics, University of Warsaw, Banacha 2, 02–097 Warsaw, Poland
miekisz@mimuw.edu.pl

Abstract. We propose an algorithm for determining optimal transition paths between given configurations of systems consisting of many objects. It is based on the Principle of Least Action and variational equations for Freidlin–Wentzell action functionals in Gaussian networks set-up. We use our method to construct a system controlling motion and redeployment between unit's formations. Another application of the algorithm allows a realistic transformation between two sequences of character animations in a virtual environment. The efficiency of the algorithm has been evaluated in a simple sandbox environment implemented with the use of the NVIDIA CUDA technology.

Keywords: Formation Redeployment, Animation Blending, Transition Path, Reconfiguration, CUDA

1 Introduction

Simulations of moving groups of agents that preserve their motoric characterizations play an important role across a broad spectrum of applications. Observation of biological systems initiated works on coordination among multiple agents. In a pioneering work [15], a computer model was constructed for synchronized animal motion observed for example in bird flocks or fish schools. It is important to emphasize that motion of individual units was calculated only on the basis of their local environment. In military applications [1], formations allow for a more effective use of limited resources, such as sensors, by division of the environment into portions so each formation's member can focus attention on an assigned segment while the rest is covered by the partners. This mechanism is used for example by groups of fighter pilots to optimize the usage of their radars and visual perception. Such an approach can also be applied to spacecrafts in a deep space or in the Earth orbit, see survey papers [20, 21] for a comprehensive description. We focus our attention on the problem of a reconfiguration of formations. Our method can be used to reduce casualties from a

* Deceased author (1975 – 2010).

hostile fire, to present less vulnerable targets or to evacuate from an exposed area [22]. Other applications involve parking systems that smoothly drive cars in and out from a parking lot. Finally, the entertainment industry may use our algorithm in real-time computer strategies.

The character animation is undoubtedly an element of the computer graphics, which is of a great importance for enjoyable computer games, credible CG movies or medical visualizations. One of the most popular techniques for generating realistic motion in virtual environments is a skeletal animation technique [4]. Skeletal systems are hierarchical in nature and provide the artist with a control of the character in an efficient manner. An animator can focus his attention on the motion of a simplified structure (the skeleton) rather than manually alter the geometry (character's meshes) itself. For smooth animations it is crucial to generate smooth transitions between system's configurations. Such transitions can be generated by mixing existing ones. One of the available methods is an interpolation of motion data, which has been shown to be a powerful technique if changes between interpolated classes meet predefined/given constraints [18]. The algorithm presented below attempts to address that drawback and provides methodology to produce optimal transition paths between arbitrary configurations.

One of the approaches to describe motion of systems of interacting particles is based on a variational principle. It is assumed in classical mechanics that the trajectory of a system between two points in the space minimizes the action functional. Then by a variational calculus one obtains Euler-Lagrange equations of motion. Reformulation of such an approach to the case of the space of curves in a set-up well suited for our needs was presented in [8].

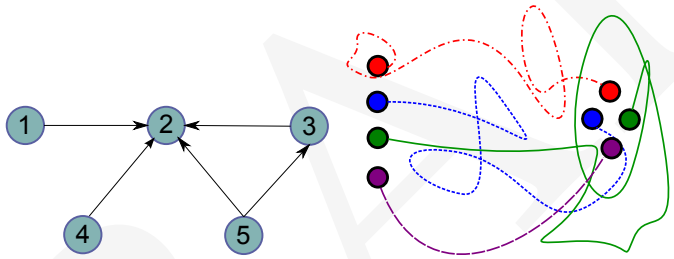
Here we use Gaussian networks. The term Gaussian network was introduced in [19] and describes an extension of a Bayesian network [12] to continuous variables. Gaussian networks are widely used for decision making and inference. In molecular biology they are used to describe protein dynamics [7]. Gaussian networks better characterize classes of behavior and provide better understanding than the standard representations [19]. To describe time evolution of Gaussian networks we use stochastic diffusion processes whose behavior is effectively characterized by the large deviation theorem due to Freidlin and Wentzell [6]. The theory is based on the property that very unlikely events, when they occur, do so with a high probability by following the pathway that is the most probable. Thus the rare events become in a sense predictable. The crucial role in the theory plays an action functional whose minimization produces an approximation of the probability of rare events and enables the computation of the maximum likelihood trajectory by which such an event occurs [8].

In Section 2, we outline the Principle of Least Action adapted to our needs and present our algorithm to simulate smooth optimal transition paths. Applications, implementation, and a discussion are contained in following sections.

2 Gaussian network

Gaussian networks [17] are systems which consist of a finite number of nodes whose states are described by continuous variables (these might be positions of certain objects as in our examples). A state of each node is subject to a stochastic dynamics which can be decomposed into a deterministic drift and a stochastic part of the diffusion type. Both the drift and the stochastic part depend on the states of other variables (perhaps just neighboring ones in the spatial networks). We may think about such a dynamics as a continuous limit of a collection of random walks biased by states of other walkers.

In Fig. 1(a), a simple Gaussian network is presented. Arrows describe influence of neighboring nodes on a stochastic dynamics of a given node. More formally, interactions between nodes are contained in the function μ and the matrix Σ in Eq. (1) below. In Fig.1(b) we can observe transitions between stable space configurations of a Gaussian network.



(a) Gaussian Network (b) Transition

Fig. 1. (a) A schematic representation of nodes in a Gaussian network. (b) Without noise the system would stay in a stable configuration. With noise the system may leave the domain of attraction and can experience rare transitions between stable configurations.

Let $\psi(t)$ is the configuration of the system at time t. For a Gaussian network with n nodes it is a column vector in R^n which evolves according to

$$\psi(t+\Delta t) = \psi(t) + \mu(\psi(t))\Delta t + \sqrt{\epsilon}\Sigma\Delta W(t), \tag{1}$$

where:

- μ is called an instantaneous drift. We will assume that $\mu(\psi) = B\psi$, for a given $n \times n$ matrix B.
- $\Delta W(t)$ are independent normal random variables with the zero mean and the variance equal to Δt. and Σ is a $n \times n$ matrix which can introduce correlations between stochastic parts of time evolution of different nodes in Eq. (1). We can think about $\Sigma\Delta W(t)$ as the source of a noise.
- ϵ is called the instantaneous variance.

In continuous time, Eq.(1) can be written as the Ito stochastic differential equation

$$d\psi(t) = B\psi(t)dt + \sqrt{\epsilon}\Sigma dW(t) \tag{2}$$

We introduce a local steering contribution $\dot{w}$ as

$$\dot{w} = \dot{\psi} - B\psi$$

where by a dot we denote a derivative with respect to time t.

In the discrete case, $\Delta w = \sqrt{\epsilon}\Sigma\Delta W$. Hence $w(t)$ is called the full "error" or the fluctuation (deviation) of our system. During the minimization process, the value of $\sqrt{\epsilon}$ can be omitted (set to 1).

We propose the following Lagrange function which defines our system:

$$L(\psi, \dot{\psi}) := \frac{1}{2}\left(\dot{\psi} - B\psi\right)' A^{-1}\left(\dot{\psi} - B\psi\right),$$

where by an apostrophe we denote a transpose of a vector or a matrix and $A := \Sigma\Sigma'$

The Principle of Least Action – of fundamental use for our applications – indicates that the system moves along the path which minimizes the action functional on the time interval $[0, T]$:

$$S(\psi) := \int_0^T L(\psi(t), \dot{\psi}(t))dt \tag{3}$$

Our construction is based on the Freidlin-Wentzell theorem [6] on large deviations in stochastic processes which roughly states that the probability of the trajectory $\bar{\psi}$ which deviates from the optimal one is proportional to $exp(-\frac{S(\bar{\psi})}{\epsilon})$.

To minimize the action functional we use the Euler-Lagrange differential equation (for the Lagrange function of a system of interacting particles we obtain in this way the Newton equations of motion),

$$\frac{\delta L}{\delta\psi} - \frac{d}{dt}\left(\frac{\delta L}{\delta\dot{\psi}}\right) = 0. \tag{4}$$

For a symmetric matrix B one obtains

$$\frac{\delta L}{\delta\psi} = -\left(BA^{-1}\dot{w}(t)\right)$$
$$\frac{d}{dt}\left(\frac{\delta L}{\delta\dot{\psi}}\right) = \left(A^{-1}\ddot{w}(t)\right)$$

and hence we get

$$\ddot{w}(t) = -ABA^{-1}\dot{w}(t) \tag{5}$$

We solve Eq. (2) as a system of linear ordinary differential equations along a fixed stochastic trajectory and get

$$\psi(T) = exp(TB)\psi(0) + \left[\int_0^T exp((T-s)B)\dot{w}(s)ds\right]$$

We know that $\dot{w}(s) = exp((-ABA^{-1})s)\dot{w}(0)$, so

$$\psi(T) = exp(TB)\psi(0) + \left[\int_0^T exp((T-s)B)exp(-sABA^{-1})ds\right]\dot{w}(0)$$

For the uncorrelated noise ($\Sigma = \mathbf{1}$) we can write:

$$\begin{aligned}\psi(T) &= exp(TB)\psi(0) + \left[\int_0^T exp((T-2s)B))ds\right]\dot{w}(0) \\ &= exp(TB)\psi(0) + exp(TB)\left[\int_0^T exp(-2sB)ds\right]\dot{w}(0) \\ &= exp(TB)\psi(0) + \frac{1}{2}exp(TB)B^{-1}\left[\mathbf{1} - exp(-2TB)\right]\dot{w}(0)\end{aligned}$$

From the above we get the initial steering configuration

$$\dot{w}(0) = 2B\left[\mathbf{1} - exp(-2TB)\right]^{-1}\left[exp(-TB)\psi(T) - \psi(0)\right] \tag{6}$$

The following procedure lies at the heart of the algorithm. The starting configuration $\psi(0)$ and matrix B must be given.

1. Set the timer $t := 0$.
2. If we do not initialize the 'force' transition to a new configuration, then
   (a) Use the rules given by Eq. 1 with $\epsilon = 1$ and $\Sigma = \mathbf{1}$
   (b) Set $t := t + \Delta t$.
3. If we initialize the 'force' transition to a new configuration and provide matrix B_1, then
   (a) Compute initial steering configuration for a given transition time T, given by Eq. 6 and denote it as $\dot{w}(t)$.
   (b) Set local timer $t1 := 0$.
   (c) Compute the new configuration

$$\psi(t + \Delta t) = \psi(t) + (B\psi(t) + \dot{w}(t))\Delta t$$

   (d) Update the local steering contribution

$$\dot{w}(t + \Delta t) = \dot{w}(t) - (B\dot{w}(t))\Delta t$$

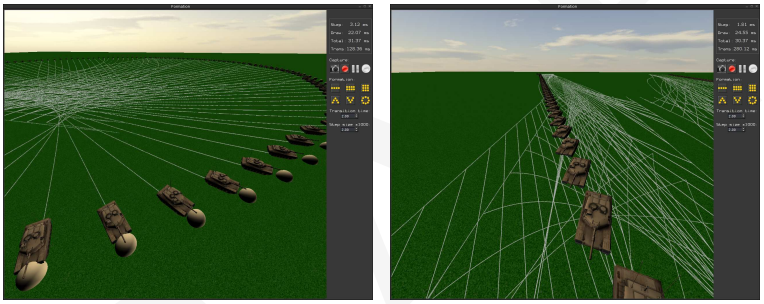
   (e) Set $t1 := t1 + \Delta t$
   (f) Update global timer $t := t + \Delta t$ and, if $t1 < T$, return to 3c else set $B := B_1$ and terminate the transition stage.
4. Return to 2

3 Formation redeployment

Similarly to [20] we define a *formation* as a set of more than one unit, whose dynamic states are coupled through a common control law. That is, the members of the set must

- Track a desired state relative to a non–empty subset of other members
- The tracking control law must depend at least upon the state of this subset at the minimum.

The second point ensures that the motion of a unit is controlled not only with use of its individual state (position, velocity, etc.), but also is affected by the state of other units. Orbit correction algorithm of the GPS satellites only require position and velocity of an individual satellite thus they do not satisfy the second requirement. Several formations for a set of units are considered:



(a) square $\mapsto$ circle (b) wedge $\mapsto$ line

Fig. 2. Screenshots from the sample application. In (a) 225 units transit from the square formation to the circle formation. The current position of an agent is represented by a tank and the distance covered is denoted with a white trace. Similarly in (b) a transition occurs from wedge to line configuration.

- *line* - where the units move in a row
- *double line* - where the units move in a two parallel rows
- *square* - where the units are regularly distributed inside a square
- *circle* - where the units move on the edge of a circle
- *V* - where the units move in a ”V” shape
- *wedge* - where the units move in a reverse ”V” shape

Fig. 3 shows a schematic description of the formations i.e. every unit is represented by a dot and is connected to its spatial neighbours by edges which illustrate neighbour influence on the node state. The configuration is described as a position of each unit on the plane.

More formally the formation is represented by a Gaussian network with units as nodes and presented connections as arcs. For each pair of connected nodes x and y a pair of vectors is given:

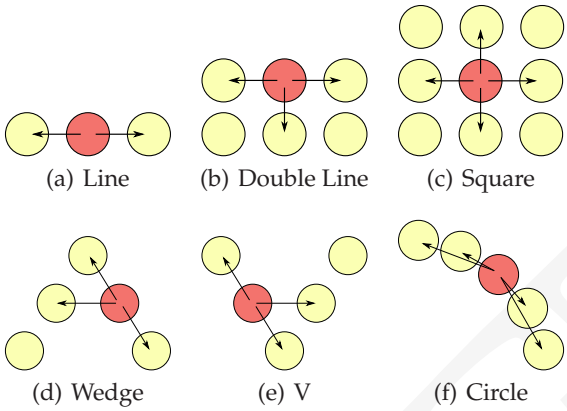


Fig. 3. Relationship between nodes in defined formations. Simple formations, such as *line* or *double line*, require only a connection between the nearest nodes. In more complex structures we have to extend the number of arcs. In *circle* formation each node is connected to the adjacent nodes and to their neighbors. Formations *V* and *wedge* require an additional connection to the nodes on the opposite branch.

- $\boldsymbol{\nu}_{x\to y}$
- $\boldsymbol{\nu}_{y\to x} = -\boldsymbol{\nu}_{x\to y}$

For movement of the group of agents let $N(x)$ denote the number of neighbors of node x, $\boldsymbol{\nu}$ the velocity of the entire formation and $\alpha \in [0,1]$ represents the impact of the neighbors on node position then state of x in time $t + \Delta t$ is defined as follow

$$\psi^{(x)}(t+\Delta t) = \psi^{(x)}(t)\,(1-\alpha\Delta t) + \overrightarrow{v}\Delta t + \alpha\Delta t\left[\frac{1}{N(x)}\sum_{y\sim x}\left(\psi^{(y)}(t) + \overrightarrow{v}_{y\to x}\right)\right] \tag{7}$$

More precisely, the matrix B can be constructed in the following way:

1. Add an artificial node $e \equiv 1$ to the Gaussian network.
2. Set the coefficients as follow
   - $B_{xx} = -\alpha$
   - $B_{xy} = \begin{cases} \frac{\alpha}{N(x)} & \text{if } y \sim x \\ 0 & \text{if } y \nsim x \end{cases}$
   - $B_{xe} = \boldsymbol{\nu} + \frac{\alpha}{N(x)}\sum_{y\sim x}\boldsymbol{\nu}_{x\to y}$,

The construction of matrix B was a crucial point in this paragraph and allows for straightforward use of Eq. (1) and Eq. (6) for the simulation. In Fig. 4 the traveled paths are presented by agents during transitions between given formations. As we can see, optimal paths are not the shortest ones. It is expected behavior because the algorithm does not minimize the length of the paths, but rather looks for most probable ones (see Fig. 2). Using shortest paths is highly dissuaded from

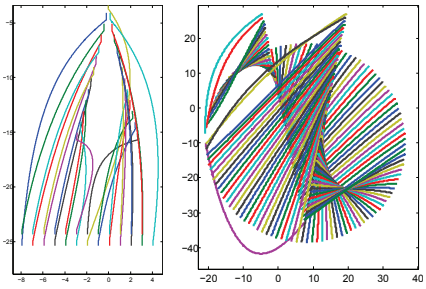


Fig. 4. On the left we see a transition of 25 units from *V* formation to *double line*. On the right 121 tanks redeploy from *circle* to *wedge* configuration.

use due to observed phenomenon in military, where all movements incidents during changes of formation were mainly caused by obtaining the shortest practical route [22].

4 Transitions between animations

The main purpose of this section is to apply the technique developed in Paragraph 2 to solving the problem of finding optimal transition path between animations. First we have to translate the motion capture date into terms of a Gaussian network. The resulted network should reproduce smoother version of the given motion. Coefficients of the matrix B will be learnt with the use of a regression method. It turn out that the task is non–trivial and first we have to define hierarchical dependencies in the motion data.

The motion capture data[3] grants us with a structure of bones S and set of motions M. S is the skeleton of an animated character that is typically defined as a hierarchy of segments. The skeleton consists of a *root* segment, that is on the top in the hierarchy and does not have any ancestors. Each other segment poses a parent segment and may have one or more children. Segments represent bones and include their properties such as their length, direction, degrees of freedom, local coordinate system, etc. Character motion M_i, with respect to the skeleton S, is defined as a doublet $M_i := (R_i, A_i)$, where R_i is the position of the skeleton's *root* segment for each frame, A_i is a sequence of orientation of each segment that is $A_i = \{a^t_{i,k}; t = 0, \ldots, T; k = 0, \ldots, \|S\|\}$ so particularly $a^t_{i,k}$ represents an orientation of *k*th bone during frame t for an *i*th animation. Current character configuration is defined as the orientation of each bone in the local coordinate system. Using the technique of skeletal animation [4], we can simulate defined motion.

Coefficients of the Gaussian network can be learnt with the use of a linear regression method [5]. For each segment k the regression model can be described

[3] The data used in this project was obtained from `http://www.mocap.cs.cmu.edu`. The database was created with funding from NSF EIA-0196217.

by the dependent vector $a_{i,k}^{t+1}$ and a set of regressors $\{a_{i,1}^{t}, \ldots, a_{i,\|S\|}^{t}\}$. Results of that straightforward method produced very unstable animations i.e. motion of a character becomes unreliable. Another approach of using *multiple linear regression* [5] with regressors defined by hierarchical structure of the skeleton suffered similar drawbacks. We were also unable to manually define correct relations in given skeletons. It appears that intuitive dependencies between human bones are misleading. For example, the position of *head* was the best described by configuration of: *right femur*, *right tibia*, *right foot* and *left wrist*. Resulting motion did not reproduce correctly given motion and was highly unstable.

To find proper relations in a given motion we decided to use one of the optimization techniques. The number of possible DAGs (Direct Acyclic Graphs) as a function of the number of nodes, G(n), is given by the following recursive function [16]

$$G(1) = 1, G(n) = \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k(n-k)} G(n-k) \tag{8}$$

Let us enumerate some values of function G $G(2) = 3, G(4) = 543, G(5) = 29281$, $G(10) = 4.2 \times 10^{18}$, $G(100) = 1.1 \times 10^{1631}$. Testing all possible DAG patterns is computationally unfeasible because the number of DAGs grows super-exponentially [12]. In our implementation we have used simulated annealing method [11] to search in possible DAG space. For a given skeleton S and animation that consists of F frames the energy function has the following form:

$$\mathcal{E}(M_i) = \frac{1}{F\|S\|} \sum_{f=1}^{F} \sum_{b=1}^{\|S\|} \|c_{i,b}^{f} - s_{i,b}^{f}\|^2, \tag{9}$$

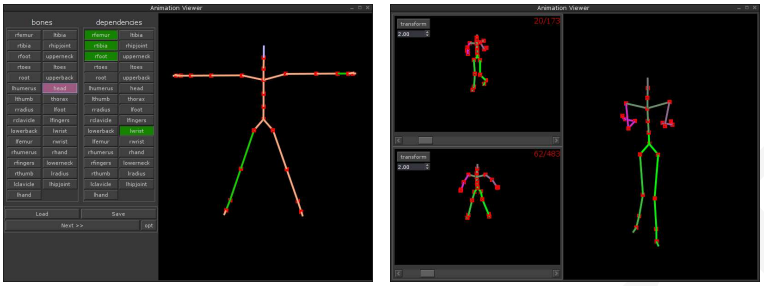
where $c_{i,b}^{f}$ is a configuration of bone b on the ith animation at frame t obtained from motion capture data, $s_{i,b}^{f}$ has similar meaning, but is computed by simulation of the Gaussian network. In other words we compute *mean square error* between positions of bones given by motion capture data in each frame and their configuration computed with presented method.

In addition, we impose the following requirements:

- Coefficients in matrix B are bound by constant b_r.
- The mean square error computed for the entire skeleton should not exceed some (large) constant s_b.

The purpose of these conditions is to ensure the numerical stability of the algorithm. During simulations, we set $b_r = 10^3$ and $s_b = 10^{12}$. The applied simulated annealing algorithm would eventually converge to the target bones hierarchy (see Image 5(a)).

After applying described scheme to the motion we obtain matrix B. The animation obtained from the simulation of the Gaussian network looks very similar to the original motion. The main difference is that the new animation



(a) bones hierarchy (b) transitions between animations

Fig. 5. Screenshots from the sandbox. In (a) we can see that orientation of the head is best described by orientation of: right femur, right tibia, right foot and left wrist, (b) presents transition from walking to jumping animation.

is more smooth. To maintain the motion in large time horizon we add small noise to the simulation [2], without that, the motion gradually fades away (it converges to the steady state). At any moment, the animation of a motion can be interrupted and by use of the method from Par. 2, even for a nonsymmetric matrix B, smoothly transformed to a selected frame of another motion (Fig. 5(b)).

5 Implementation

The programme has been implemented in language `D` [3]. Some matrix operations incorporate LAPACK, BLAS and CUBLAS [14] subroutines. All test runs were executed on a machine with Intel Core 2 Q9300 2.50 GHz CPU, 4GB RAM and NVIDIA GTX 480. The application is single threaded, so it is applicable to the core of only one processor. All computations were performed with double precision arithmetic.

The mean square error (MSE) between the target configuration and a simulated one can be controlled by adjusting the step size. For the *transition time* set on 2.0 and *step size* on 0.002 the observed value of MSE was lower than 0.0001.

An efficient computation of a matrix exponential plays a crucial role in the presented method. Numerous methods for computing e^A were developed. The straightforward one, which uses Euler series, is inefficient even in the scalar case. The survey [13] presented a wide variety of methods and pointed the most powerful ones. Today, due to evolution in computer hardware and highly optimized BLAS subroutines the most cogent technique is the scaling and squaring method combined with Padé approximants [9]. The sequential version of that algorithm has been implemented. To parallelize the computation of the matrix exponential we use NVIDIA CUBLAS library [10, 14] and substitute serial matrix operations with parallel ones.

In Table 1 we can see dependencies between the number of simulated objects (tanks) and the time required for a single step and a transition. As we can see,

Table 1. Step and transition times.

quantity	step (ms)	CPU (ms)	GPU (ms)	speedup
25	0.04	1.7	40	0.07
100	0.6	24	109	0.22
169	1.6	75	193	0.38
225	3.1	172	275	0.63
400	9	680	637	1.07
625	22	2200	1470	1.50
900	49	12000	3480	3.45
1225	89	54000	6890	7.84
1600	155	253000	12400	17.57
2025	240	785000	26200	**29.96**

a single step can be computed very fast. The most consuming part during w_0 calculation is matrix exponential. GPU accelerated version is up to 30 times faster than the sequential one. An analysis of parallel profiler output shows that time is mainly (up to 70%) spent on matrix multiplications (*dgemm*). Transfers of the data from host to device (CPU $\rightarrow$ GPU) and vice versa take up to 30% of the time. The rest of the time is consumed on memory allocations and other matrix operations. We should emphasize that computations are performed for x, y and z coordinates independently.

6 Conclusions

The algorithm for determining optimal transition paths was presented. For the evaluation purpose two applications have been successfully implemented. The problem of the formation redeployment was solved by our algorithm and produced enjoyable results. The resulting group dynamics is highly complex and a great care must be taken when such an environment is simulated. The other tested application was devoted to transitions between animations. The problem was more complicated due to the need of representing the motion in terms of Gaussian networks. The obtained animations were satisfying for the viewer. Simulations on the GPU were also performed and provided a significant gain in the runtime, but only in the large enough networks. Future enhancements may include hierarchical grouping of Gaussian nodes before transitions which would significantly improve a computation time. Another option is to optimize the transition time rather than to use the given one.

Acknowledgements

M. Matuszak and T. Schreiber gratefully acknowledge the support from the Polish Ministry of Scientific Research and Higher Education grant N N201 385234 (2008-2010). The work of M. Matuszak has also been supported by the European Social Fund, Government of Poland and Kuyavian-Pomeranian Voivodeship as a part of Integrated Operational Program for Regional Development, Activity 8.2.2.

References

1. Balch, T., Arkin, R.C. Behavior–based formation control for multirobot teams, *IEEE Transactions on Robotics and Automation*, 14(6), pp. 926–939 (1998).
2. Balch, T., Hybinette, M. Behavior–Based Coordination of Large-Scale Robot Formations, *Multi-Agent Systems*, International Conference on Multiagent Systems – ICMAS, pp. 363–364, (2000)
3. Bell, K., Igesund, L.I., Kelly, S. Parker, M. Learn to Tango with D, Apress (2008).
4. Burtnyk, N. and Wein, M. Interactive skeleton techniques for enhancing motion dynamics in key frame animation, *Commun. ACM 19*, pp. 564–569 (Oct. 1976).
5. Draper, N.R., Smith, H. Applied regression analysis, 3rd edn. Wiley, New York (1998).
6. Freidlin, M.I., Wentzell, A.D. Random perturbations of dynamical systems, Vol. 260, *Grundlehren der Mathematischen Wissenschaften (2nd ed.)* New York: Springer-Verlag (1998).
7. Haliloglu, T., Bahar, I., Erman, B. Gaussian dynamics of folded proteins, *Physical Review Letters 79*, pp. 3090–3093, (1997).
8. Heymann, M., Vanden-Eijnden, E. The Geometric Minimum Action Method: A Least Action Principle on the Space of Curves, *Comm. Pure Appl. Math.***61.8**, 1052–1117, (2008).
9. Higham, N. J. The Scaling and Squaring Method for the Matrix Exponential Revisited, *SIAM J. Matrix Anal. Appl.,* **26(4)**, pp. 1179–1193 (2005).
10. Kirk, D.B., Hwu, W.-M.W. Programming Massively Parallel Processors: A Hands–on Approach, *Morgan Kaufmann*, 1 edition (2010).
11. Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. Optimization by Simulated Annealing, *Science 220*, pp. 671-680, (1983).
12. Koski, T., Noble, J. Bayesian Networks: An Introduction, John Wiley & Sons, Ltd (2009).
13. Moler, C., Van Loan, C. Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty–Five Years Later, *SIAM Rev.* **45, 3** (2003).
14. NVIDIA CUBLAS Library, *NVIDIA Corporation,* (2009).
15. Reynolds, C. W. Flocks, Herds, and Schools: A Distributed Behavioral Model, Computer Graphics, 21(4) (SIGGRAPH '87), pp. 25–34 (1987).
16. Robinson, R.W. Counting Unlabelled Acyclic Digraphs, Springer Lecture Notes in Mathematics, Combinatorial Mathematics V, pp. 28 – 43 (1977).
17. Rue, H., Held, L. Gaussian Markov Random Fields: Theory and Applications, *Monographs on Statistics & Applied Probability 104*, (2005).
18. Safonova, A., Hodgins, J.K. Analyzing the physical correctness of interpolated human motion, ACM Siggraph/Eurographics Symposium on Computer Animation (SCA '05), 171–180 (2005).
19. Shachter, R.D., Kenley, C.R. Gaussian influence diagrams *Management Science*, 35(5), pp. 527–550, (1989).
20. Scharf, D.P., Hadaegh, F.Y. and Ploen, S.R. A Survey of Spacecraft Formation Flying Guidance and Control (Part I): Guidance, *American Control Conference, Vol. 2*, pp. 1733 – 1739, (June 2003).
21. Scharf, D.P., Hadaegh, F.Y. and Ploen, S.R. A Survey of Spacecraft Formation Flying Guidance and Control (Part II): Control, *American Control Conference, Vol. 4*, pp. 2976–2985, (30 June – 2 July 2004).
22. U.S. Marine Corps Marine Rifle Squad, *Marine Corps Warfighting Publication (MCWP)* 3–11.2, (1997).