Concurrent learning of a Probabilistic Graphical Model on the GPU

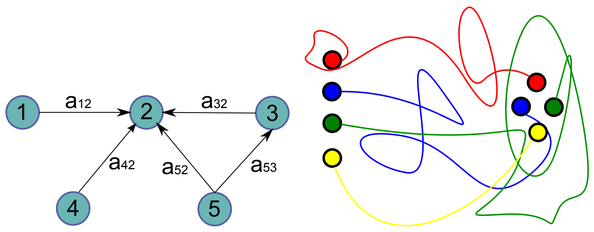
Michał Matuszak
Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland

Abstract

We introduce an algorithm for determining optimal transition paths between given configurations. The solution is obtained by solving variational equations for Freidlin–Wentzell action functionals. One of the applications of the method presented is a system controlling motion and redeployment between unit's formations. The efficiency of the algorithm has been evaluated in a simple sandbox environment implemented with the use of the NVIDIA CUDA technology.

Introduction

One of the approaches to describe the motion of the system of interacting particles is based on a variational principle. It is assumed in a 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 motions. Reformulation of such approach to the case of the space of curves in a set-up well suited for our needs was presented in [2].
Here we use Gaussian networks. The term Gaussian network describes an extension of a Bayesian networks to continuous variables. Gaussian networks are widely used for decision making and inference. Given a Gaussian belief network, we can generate a multivariate normal density, and vice versa.



(a) Gaussian Network

(b) Transition

Figure : (a) a schematic representation of nodes in a Gaussian network, (b) noise allows for a rare transitions between stable configurations.

In Fig. 1(a) we can see a simple Gaussian network. Each node is characterized by a mean and a variance of a normally distributed random variable and the relative importance of its parents.
More specifically, we use stochastic diffusion processes whose behavior is effectively characterized by the large deviation theorems due to Freidlin and Wentzell [1].

Dynamical system

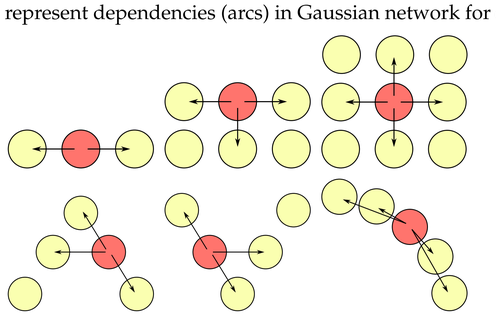
Our system is represented by a dynamical Gaussian network. $\psi(t)$ is the configuration of the system at time t. For a Gaussian network with n nodes it is a vector in R^n which evolves according to

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

where:

- μ is called an instantaneous drift. We will assume that $\mu(\psi) = B\psi$, for a given matrix B.
- $W(t)$ is an n dimensional Wiener process and $\Delta W(t) \sim \mathcal{N}(0, \Delta t)$, a normal random variable.
- $A = \Sigma\Sigma^T$.

Six pictures below represent dependencies (arcs) in Gaussian network for our example.



Simple formations such as *line* or *double line* require only a connection between nearest nodes. In more complex structures we have to extend number of arcs. In *circle* formation each node is connected to adjacent nodes and to their neighbours. Formations *wee* and *wedge* require additional connection to the nodes on the opposite branch.

Formation

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:

- $\vec{v}_{x\to y}$
- $\vec{v}_{y\to x} = -\vec{v}_{x\to y}$

For movement of the group of agents let $N(x)$ denote the number of neighbors of node x, $\vec{v}$ 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

$$\psi^{(x)}(t+\Delta t) = \psi^{(x)}(t)\,(1-\alpha\Delta t) + \overrightarrow{v}\,\Delta t + \alpha\Delta t\left[\frac{1}{N(x)}\sum_{y\sim x}\left(\psi^{(y)}(t) + \overrightarrow{v}_{y\to x}\right)\right] \tag{2}$$

Minimisation

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

$$d\psi(t) = dB\psi(t)dt + \sqrt{\epsilon}\Sigma dW(t) \tag{3}$$

Denote a local steering contribution as $\dot{w}_t\Delta t = \Sigma\mathcal{N}(0,\Delta t)$. Then w_t is called the full "error" or the fluctuation (deviation) of our system.
For ψ, which represents the trajectory, we have

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

We propose the following Lagrange function which defines our system:

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

where by a prime we denote a transpose of a given matrix.
The Least Action Principle, of fundamental use for our applications, indicates that the system moves along the path which minimizes the action functional:

$$\int_a^b L(t,\psi_t,\dot{\psi}_t)dt = S(\psi) \tag{4}$$

To minimize the action functional we use the Euler-Lagrange differential equation

$$\frac{\delta L}{\delta\psi} - \frac{d}{dt}\left(\frac{\delta L}{\delta\dot{\psi}}\right) = 0. \tag{5}$$

Solving the Euler-Lagrange equation leads us to:

$$\psi_T = exp(TB)\psi_0 + \left[\int_0^T exp((T-s)B)\dot{w}_s ds\right]$$

To utilise the previous result we should also calculate $\dot{w}_0$. For simplicity we assume that $A = 1$.

$$\dot{w}_0' = 2B\left[\mathbf{1} - exp(-2TB)\right]^{-1}\left[exp(-TB)\psi_T - \psi_0\right] \tag{6}$$

Algorithm

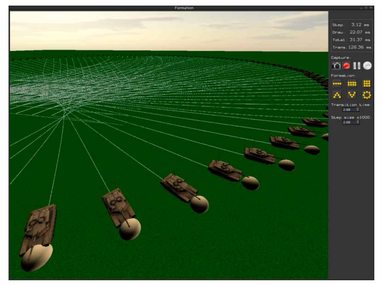
The following procedure lies at the heart of the algorithm. The starting configuration ψ_0 and matrix B must be given.

1. Set the timer $t := 0$.
2. If the transition does not occur, then
   2.1 Use the rules given by Eq. 1
   $$\psi(t+\Delta t) = \psi(t) + B\psi(t)\Delta t + \sqrt{\epsilon}\mathcal{N}(0,\Delta t)$$
   2.2 Set $t := t + \Delta t$.
3. If the transition occurs, then
   3.1 Compute initial steering configuration for a given transition time T, given be Eq. 6 and denote it as $\dot{w}(t)$.
   3.2 Set local timer $t1 := 0$.
   3.3 Compute the new configuration
   $$\psi(t+\Delta t) = \psi(t) + (B\psi(t) + \dot{w}(t))\Delta t$$
   3.4 Update the local steering contribution
   $$\dot{w}(t+\Delta t) = \dot{w}(t) - (B\dot{w}(t))\Delta t$$
   3.5 Set $t1 := t1 + \Delta t$
   3.6 Update global timer $t := t + \Delta t$ and, if $t1 < T$, return to 3 else terminate the transition stage.
4. Return to 2

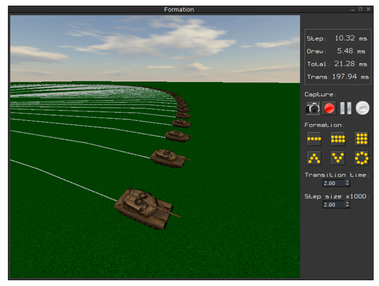
This research has been supported by the National Science Centre grant 2011/01/ N/ST6/00573 (2011-2014).

Results

The programme has been implemented in language D. Some matrix operations incorporate LAPACK, BLAS and CUBLAS 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.



(a)

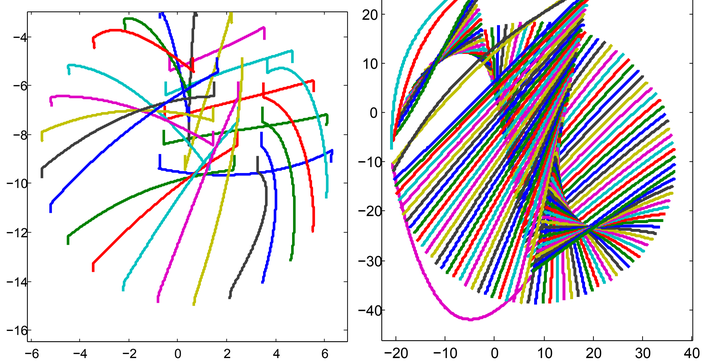


(b)

The mean square error (MSE) between the target configuration a simulated one can by controlled by adjusting step size. For *transition time* set on 2.0 and *step size* on 0.002 the MSE was always lower than 0.0001.
In the following table we can see dependencies between the number of simulated objects (tanks) and the time required for a single step and a transition.

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**

As we can see, a single step can be computed very fast. The most consuming part during w_0 calculation is matrix exponential. We should emphasise that computations are performed for x, y and z coordinates independently.



On the left we can see transitions of 25 units from *square* formation to *circle*. On the right 121 tanks redeploy from *circle* to *wedge* configuration.

Bibliography

[1] Freidlin, M.I., Wentzell, A.D. Random perturbations of dynamical systems, Vol. 260, *Grundlehren der Mathematischen Wissenschaften (2nd ed.)* New York: Springer-Verlag (1998).

[2] 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).