CEIG - Spanish Computer Graphics Conference (2016)
Alejandro García-Alonso and Belen Masia (Editors)

Phase Space Data-Driven Simulation of Elastic Materials

C. Monteagudo[1], M. Lozano[2], I. García-Fernández[2] and F. Martinez-Gil[2]

[1]Escola Tècnica Superior d'Enginyeria. Universitat de Valencia, Spain
[2]Departament d'Informàtica. Universitat de Valencia, Spain

Abstract

The use of data driven models in computer animation offers several benefits. We present an analysis of a regression model as a method to simulate cloth. In our approach, we generate data from a simple mass-spring system and we fit a regressor. Then, we assemble more complex mass-spring systems and use the learnt model to simulate them. To validate the approach we perform several tests. We analyze the elastic properties of a single learnt spring, measuring its stiffness coefficient, and compare it to the original, physics-based, model. We also build several test scenarios which include the simulation of a piece of cloth under gravity, comparing the regression model and the physics-based model. Finally we test the behaviour of the regression model for systems with high stiffness coefficient and compare its stability properties with a semi-implicit Euler integration method.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality

1. Introduction and related work

In computer animation, problems such as cloth simulation, facial animation or soft tissue behaviour (eg. for surgery training systems) require fast and stable simulation techniques for deformable materials. Finite Element Modeling (FEM) has been broadly used in the context of physically based animation. However Finite Element simulation is, in many cases, too expensive for real-time deformable simulations. On the contrary, other physically based models, such as the Mass-Spring Model, (MSM) have been used traditionally in this context since they are fast and handle easily large rotations or topology changes. The main drawback of MSM simulation is that it becomes unstable if the material is stiff.

Data driven methods can be useful in this context, if we are capable to provide a learning method that reproduces stable deformation behaviours. Data driven and machine learning approaches are not new in the context of elastic materials simulation. Many authors use data driven approaches to fit model parameters, using both real material data or reference models such as FEM [WOR11, MBT*12, SVAC12]. Other authors use Radial Basis Functions to approximate the forces for haptic rendering [Fon09] or train the relationship between low resolution deformation and high resolution deformation [FYK10, KSO10, KGBS11, SSH12, SSH14] Bickel et al. [BBO*09] use corrotational FEM with a parametrized linear model which changes with strain, thus giving nonlinear behaviour. They learn the relationship from data. Kim et al. [KKN*13] focus on more specific behaviors and train deformation of cloth that fit prerecorded movements of an actor.

An excellent review of data driven methods for deformable objects in computer graphics can be found in [OBBW12]. According with the classification proposed in that survey, our work can be viewed as a mechanical-data driven method as the fitting process involves an optimization procedure, that relates model parameters to deformations and external forces (eg. gravity) must be introduced. In this paper, we use a regression method to simulate cloth avoiding the numerical integration step. In our approach, the regressor learns the behaviour of a single spring and then, we scale this approach to simulate cloth at interactive rates without instabilities. In this work we use MSM as a reference model, since it offers a simple physically-based model that can be analytically validated.

Regession models have been previously used for complex animations. Ladicky et al. [LJS*15] use regression forests to learn fluid dynamics from the SPH model. They approximate the relationship between current state of the fluid in the neighbourhood of a particle and the acceleration of that particle, which has some similarities with our approach, in the sense that they use a phase space representation, as we shall do.

In this paper we explore the possibility to use machine learning techniques to build a model able to imitate elastic physically simulated objects formed by particle systems. Our model uses phase space representation (namely position and velocity in Cartesian coordinates). The rest of the paper has been organized as follows: the next section reviews the simulation framework designed to generate the training sets to be learnt, then the Section 3 shows the regression model used and the training methodology followed for the scenarios considered. The model is validated by means of an analysis of the stiffness of the simulated material, presented in Section 4. The model is tested in a more complex scenario, by simulating a piece

DOI: 10.2312/ceig.20161316

of cloth, as it is described in Section 5. From the results of the previous sections, in Section 6 we present a discussion of the limitations of our approach, that must be overcome to make it useful in complex simulations.

2. Simulation framework

Let us first consider a standard particle system simulation, based on second Newton's Law. We start a simulation step from the current state of the particles. First, external forces and gravity are computed. Next, the internal, elastic, forces of the material are computed using a physics based model. Finally, once all the forces have been computed, a numerical integration step is used to know the state of the particles (position and velocity) after a time interval Δt. We shall call this model the *reference model*. Our proposal is to substitute the computation of the internal forces by an approximate function, that is fitted from a set of sample data. Thus, our simulation framework will consider a function ϕ that takes as inputs the state of the particles, described by their position $\mathbf{x}$, and velocity $\mathbf{v}$, and provides a new state in the form of a change in position $\Delta\mathbf{x}$ and velocity $\Delta\mathbf{v}$. We shall call this model the *learnt model*.

In this paper we will use a mass-spring as our reference model. In a mass-spring system, every particle is connected to other particles by means of springs; the springs generate forces on the particles they connect as a result of their deformation. In the learning scenario, we use a single particle attached to a spring (reference model), and then we use the generated data to fit the regression model. Moreover, we use a one-dimensional formulation for the reference model. In one dimension, the equation for a particle of mass $m = 1$ attached to a spring, with stiffness coefficient k and damping coefficient c is

$$\ddot{e}(t) = -ke(t) - cv(t),$$

where $e(t)$ is the elongation of the spring which can be computed as $e = x - x_0$, with x_0 the location of the particle where the spring is in equilibrium.

To be able to build new deformable structures using a learned spring, we need to do some transformations to locate it in the learning scenario where we must assure that the orientation is the one required by the learning process (1D). To properly orientate any particle we simply rotate and scale the distance vector between two particles to the one dimensional space of the training set (see Figure 1). Thus, we can obtain the correct prediction for the oscillating particle p_i. Being a linear model, we are able to apply the same prediction with the opposite sign to the other particle p_j. Then we orient the displacements along the direction of the original distance vector in the simulation scenario.

Given $\Delta t > 0$, used for time discretization of the solution, we want to learn the relationship between the states of a particle at times t and $t + \Delta t$. In order to build the dataset to fit the regression model we shall use the semi-implicit Euler integration scheme. It is important to note that the results will depend on the integration scheme used to generate the data set. In our case, if we denote $v = \dot{e}$,

$$v(t+\Delta t) = v(t) - \Delta t(ke(t) + cv(t)), \quad (1)$$

$$e(t+\Delta t) = e(t) + \Delta t v(t+\Delta t), \quad (2)$$

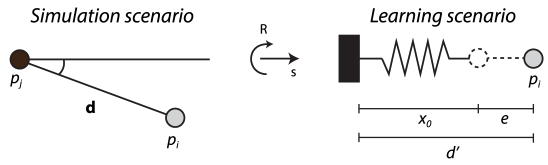

Figure 1: *Rotation and scale of the distance vector from the simulation scenario to the space of the learning scenario.*

and, by substituting (1) in (2), we get

$$e(t+\Delta t) = (1 - k\Delta t^2)e(t) + \Delta t(1-c)v(t).$$

We want to learn the relationship between (x, v) and $(\Delta e, \Delta v)$, where $\Delta e = e(t+\Delta t) - e(t)$ and $\Delta v = v(t+\Delta t) - v(t)$. Then the previous expressions can be written in matrix form as

$$\begin{pmatrix} \Delta e \\ \Delta v \end{pmatrix} = \Delta t \begin{pmatrix} -k\Delta t & 1-c \\ -k & c \end{pmatrix} \begin{pmatrix} e(t) \\ v(t) \end{pmatrix}. \quad (3)$$

Once we have fitted this relationship, we can evaluate it by simulation as follows. First, we apply external (gravity) forces to the particles, computing a semi-implicit Euler step. Then, we use the resulting positions and velocities of the particles as the input vector of the trained model. Since the model is applied for every spring attached to every particle, we shall also add to the sample the functions to learn, in our case, the elongation and velocity increments. Once we have obtained the elongation change, Δe, we can apply it directly to the particle location, since

$$\Delta x = x(t+\Delta t) - x(t) = x(t+\Delta t) - x_0 - (x(t) - x_0) = \Delta e.$$

3. The Regression Model

Regression models attempt to learn a mapping $y = f(x)$, where x represents some input vector, and y represents an output vector. According to expression (3) the dynamics we want to fit follows a linear relationship between the current state of the system and its evolution. For this reason, we are going to use linear regression, in which the mapping f is a linear function.

The linear regression model has the form $f(X) = \beta_0 + \sum_{i=1}^{i=n} X_i\beta_i$ where β_i are the unknown parameters, X_i the variables that describe the state of the particle and n the number of characteristics of the state space, the problem is to estimate the parameters using the least squares method (LSM). This criterion is valid if the data (x, y) are independent random draws as in this case happens [HTF01]. Thus, the regression process calculates the coefficients $\{\beta_0, \beta_1, \ldots, \beta_n\}$ that minimize the sum of squares:

$$S(\beta) = \sum_{i=1}^{i=N} \left(y_i - \beta_0 - \sum_{j=1}^{j=n} x_{ij}\beta_j \right)^2, \quad (4)$$

where N is the number of data and x_{ij} is the component j of data i and y_i is the output for data i.

k_{MSM} (reference model)	k_{REG} (learnt model)
100	100.51 $(\sigma < 10^{-4})$)
250	253.18 $(\sigma < 10^{-6})$)
500	512.89 $(\sigma < 10^{-5})$)
750	779.37 $(\sigma < 10^{-4})$)
1000	1052.91 $(\sigma < 10^{-4})$)

Table 1: *Differences between the spring constant of the reference model (k_{MSM}) and that of the learnt model (k_{REG}).*

3.1. The training set

The learning scenario contains a single spring, with a stifness k, which has been initially stretched or pushed certain distance d from the equilibrium distance l. This simple scenario lets us to evaluate the behaviour of the spring model while providing a complete training set for supervised machine leaning models. Specifically, we have generated 5 independent training sets for different values of the stiffness coefficient in the physical model ($k = [100, 250, 500, 750, 1000]$). This sets are used to learn 5 different regression-based springs and therefore let us the possibility to validate and/or calibrate the k obtained by the regression model, as the next section shows.

In our case, the regressor try to find a mapping $y = f(x)$, where x allocates the particle state at time t and y corresponds to the position and velocities deltas computed for this step. Specifically, the x vector only contains the particle elongation and velocity in 1D as the learning scenario is defined in one dimension, so we need only 2 floats ($p.x$, $v.x$). The vector y allocates the velocity and position changes for the step considered so it also needs 2 floats $(\Delta x, \Delta y)$. According to this learning schema, the generalization of the process to build new spring-based structures based on the learned model needs to perform some transformations to the particles involved to be useful (see Figure 1). Once the regression model has been fit, it can be used to predict the next cinematic state of any particle connected to a spring from its current state. It is easy to realize that the learnt model avoids numerical integration as it working as a next-state predictor.

4. Validation

In order to validate and test the learning method proposed we have use different simulation scenarios. As a first step, we evaluate the learnt spring model, and then we compare its behaviour with the physical one. In the experiment, we have applied different forces to the learnt model to infer from the empirical elongation the value of the spring constant k. This value is then compared with the value of the physical model. Table 1 displays the results for different k's. For each value of k of the learnt model, the displayed value is a mean acquired applying different forces (from 5 units of force to 50, in intervals of 5). The value of σ corresponds to the standard deviation of the sample. As the Table 1 shows, the relative error of the learnt model is less than 0.06% in the worst case.

In a second experiment we have assemble several learnt-springs to build an 'elastic rope' without extra learning, that is using the model of the previous experiment. It is known that the equivalent spring of several connected springs follows this expression: $\frac{1}{K} = \sum_{i=1}^{i=n} \frac{1}{k_i}$. In our experiments we have fixed an extreme of the 'rope' and then have applied a constant force to the other. The elongation obtained has been divided by the number of constitutive springs in order to get the resulting spring coefficient k. This is compared with the k of the phyisical model. In Table 2 the results for ropes with 2, 5 and 10 nodes are displayed.

	k_{MSM}				
nodes	100	250	500	750	1000
2	100.51	253.18	512.89	779.37	1052.91
5	101.61	254.51	510.65	768.45	1027.90
10	103.97	260.16	521.07	782.73	1045.17

Table 2: *Values of the learnt k for different reference values of k_{MSM}. Every row corresponds to a different number of involved nodes for the rope.*

The results show a good consistence of the data when scaling up the number of nodes. This robustness ans stability of the learned model are desirable properties that allow to consider more complex situations. In Figure 2 the absolute errors of the parameter k are displayed for different lengths of ropes (different number of nodes).

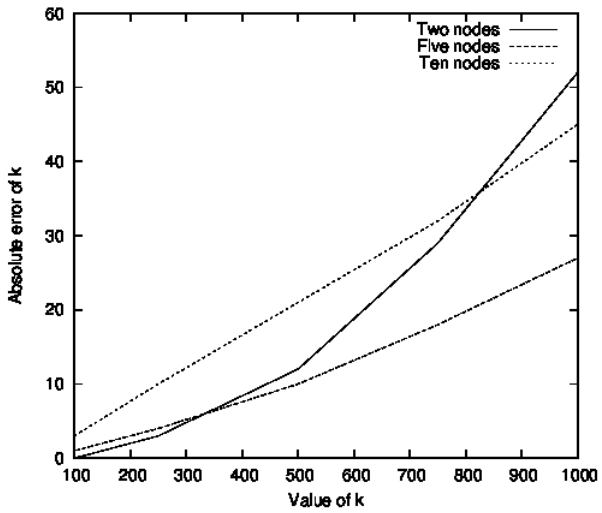

Figure 2: *Absolute error of k for ropes with different number of nodes.*

The curve with less slope corresponds to the rope with five nodes, that is followed for the curve with ten nodes. The continuous line corresponds to the rope with two nodes. Surprisingly, the ropes with more units present less absolute errors in the implicit k than the shortest rope.

5. Results on cloth simulation

Table 3 shows the stiffness values obtained from a experiment using a cloth with 5×5 particles. Every particle is attached to every

neighbor by a spring, taking as neighbor the 8 adjacent particles. In this case, an external force is applied at the lower edge of the cloth and the elongation is measured in the center of the lowest row, getting the values for k. The values associated to the stiffness k_{REG} learned by the regressor reveal higher differences compared with the rope scenario (4 times softer). To afford this, we use another linear regression to know the k_{MSM} values we must use during the learning phase to finally obtain similar spring behaviours. Considering data of Table 3, a linear regression has been fit with equation:

$$k_{\mathrm{MSM}} = 0.273\, k_{\mathrm{REG}} + 4.75.$$

The capacity of the learning model to generalize, that is, to be able to easily assemble new deformable structures without additional learning is a very interesting feature provided by the learning model presented. However, the resulting models with several springs attached to a particle result in an averaging of their effect which leads to the stiffness differences obtained in this scenario. This issue will be discussed in Section 6 in more detail. In the rope scenario this problem is less noticeable, since a particle has less springs attached to it than in the cloth simulation.

Physic spring k	k_{MSM}	k_{REG}
100	91.546	25.262
250	215.352	56.505
500	420.084	105.330
750	624.473	153.102
1000	828.755	200.499

Table 3: *Spring constants for the reference model and the learnt model in the experiment of a 5 X 5 nodes cloth. Note that the k_{REG} is about four times less than the k_{MSM}.*

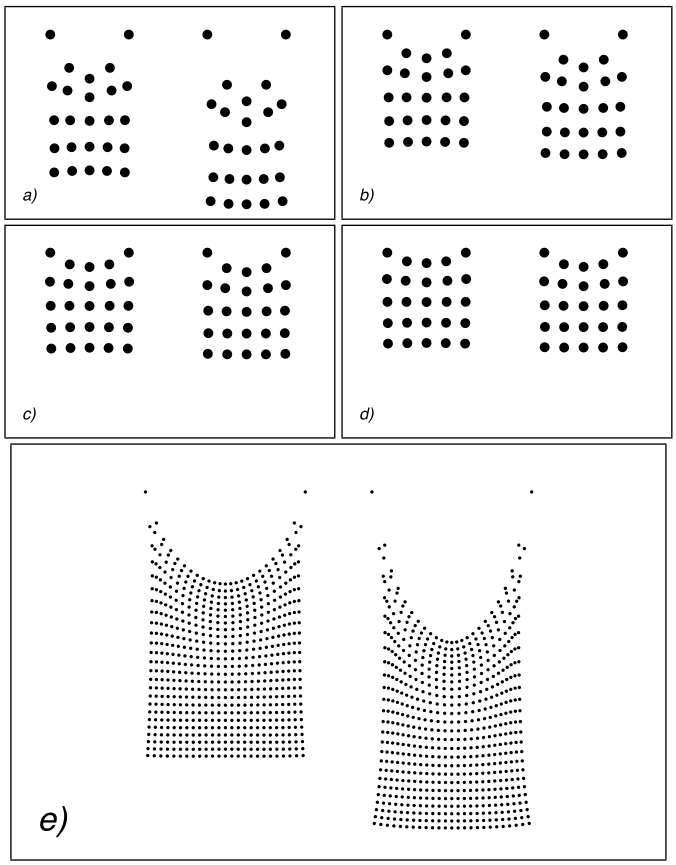

Figure 3: *Physical model (left) vs Regression model (right) for a) $k = 100$, b) $k = 250$, c) $k = 500$, d) $k = 750$ e) Stiffness differences found for a 25x25 mesh size.*

Figure 3 shows the last frame of both, physical and regression models for different stiffness values and gravity computed for all particles. In this way we can easily recognize the differences at a final or convergence state. On the other hand, the last figure (e) shows the stiffness differences mentioned in a 25x25 deformable mesh.

6. Limitations of the model

The experimental results show a deviation of the regression-based model respect to the physical model; in general, we have observed that the regression-based model is less stiff than the original mass-spring model when several springs are attached to a particle. The cause is the average step carried out to get the new position of the particle as the result of the contributions of several springs. Given n springs connected to a particle, at time t, the algorithm will call upon n times the regressor with $e(t)$ and $v(t)$ as inputs to get $\Delta\mathbf{v}$ and $\Delta\mathbf{x}$ as outputs. The term $\Delta\mathbf{x}$ has been learned from the Euler's integration expression:

$$\Delta x = \Delta t v(t) + \Delta t^2(-ke(t)).$$

Therefore, if the particle i is connected with n springs the sum of the contributions of the springs will be:

$$\Delta\mathbf{x}_i = \sum_j^n \Delta\mathbf{x}_j = \sum_j^n (\Delta t\mathbf{v}_i(t) + \Delta t^2(-k_j e_j(t))) =$$
$$= n\Delta t\mathbf{v}_i(t) + \sum_j^n \Delta t^2(-k_j e_j).$$

If this expression is applied directly, the contribution of the velocity $\mathbf{v}_i$ will be applied n times, causing an overshooting situation, and leading to physical inaccuracy and numerical instability. In order to overcome this problem, in our implementation we have averaged the final correction, as

$$\Delta\mathbf{x}_i = \frac{1}{n}\sum_j^n (\Delta t\mathbf{v}_i(t) + \Delta t^2(-k_j e_j)) =$$
$$= \Delta t\mathbf{v}_i(t) + \frac{1}{n}\sum_j^n \Delta t^2(-k_j e_j) =$$
$$= \Delta t\mathbf{v}_i(t) + \sum_j^n \frac{\Delta t^2(-k_j e_j)}{n}.$$

This solution solves the instability issue, but introduces an averaging of the effect of the springs affecting particle i. Since the sum in $\mathbf{x}_j$ is not computed explicitly, but obtained as a numerical value from the regressor, it is not possible, in our current design, to apply an averaging only to the term involving $\mathbf{v}_i$ to avoid damping.

As a possible solution to this situation, in future developments we intent to learn different regression models for the dynamics of the particles, depending on the number of springs attached to them. This approach will solve the averaging problem, and will prevent the need of adding the contribution of the springs, thus saving computations.

7. Conclusion

We have presented a data driven method to simulate elastic materials. The regression method designed replaces the numerical integration step by a prediction of the next particle state. To do this, the regressor has previously learnt the relationship between the elongation of a spring and the particle changes in phase space. As expected, the results indicate that the regression provides a very good approximation of the learnt model, and provides good generalization for one dimensional systems (ropes). When extended to cloth simulation, the assembly process provides softer materials, with a young modulus lower than the original reference material, due to the averaging discussed in Section 6. However, the relationship between the original stiffness and the learnt stiffness has been found to be linear, which provides an easy fitting procedure.

Thus, the results suggest that the data driven simulation model presented is an adequate strategy to simulate MSM based deformable objects. The results obtained so far in simple scenarios are promising, and we expect to extend them to other situations such as three dimensional materials or the use of nonlinear reference models. In this later case, regression trees could be necessary to deal with the different states of the system.

Acknowledgements

This research is partially funded by grant TIN2014-59932-JIN.

References

[BBO*09] BICKEL B., BÄCHER M., OTADUY M. A., MATUSIK W., PFISTER H., GROSS M.: Capture and modeling of non-linear heterogeneous soft tissue. *ACM Trans. Graph. 28*, 3 (2009), 89:1–89:9. doi:10.1145/1531326.1531395. 1

[Fon09] FONG P.: Sensing, acquisition, and interactive playback of data-based models for elastic deformable objects. *The International Journal of Robotics Research 28*, 5 (2009), 630–655. doi:10.1177/0278364908100326. 1

[FYK10] FENG W.-W., YU Y., KIM B.-U.: A deformation transformer for real-time cloth animation. *ACM Trans. Graph. 29*, 4 (July 2010), 108:1–108:9. doi:10.1145/1778765.1778845. 1

[HTF01] HASTIE T., TIBSHIRANI R., FRIEDMAN J.: *The Elements of Statistical Learning*. Series in Statistics. Springer, 2001. 2

[KGBS11] KAVAN L., GERSZEWSKI D., BARGTEIL A. W., SLOAN P.-P.: Physics-inspired upsampling for cloth simulation in games. *ACM Trans. Graph. 30*, 4 (2011), 93:1–93:10. doi:10.1145/2010324.1964988. 1

[KKN*13] KIM D., KOH W., NARAIN R., FATAHALIAN K., TREUILLE A., O'BRIEN J. F.: Near-exhaustive precomputation of secondary cloth effects. *ACM Trans. Graph. 32*, 4 (July 2013), 87:1–87:8. doi:10.1145/2461912.2462020. 1

[KSO10] KAVAN L., SLOAN P.-P., O'SULLIVAN C.: Fast and efficient skinning of animated meshes. *Computer Graphics Forum 29*, 2 (2010), 327–336. doi:10.1111/j.1467-8659.2009.01602.x. 1

[LJS*15] LADICKÝ L., JEONG S., SOLENTHALER B., POLLEFEYS M., GROSS M.: Data-driven fluid simulations using regression forests. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 199:1–199:9. doi:10.1145/2816795.2818129. 1

[MBT*12] MIGUEL E., BRADLEY D., THOMASZEWSKI B., BICKEL B., MATUSIK W., OTADUY M. A., MARSCHNER S.: Data-driven estimation of cloth simulation models. *Computer Graphics Forum 31*, 2pt2 (2012), 519–528. doi:10.1111/j.1467-8659.2012.03031.x. 1

[OBBW12] OTADUY M. A., BICKEL B., BRADLEY D., WANG H.: Data-driven simulation methods in computer graphics: Cloth, tissue and faces. In *ACM SIGGRAPH 2012 Courses* (2012), SIGGRAPH '12, pp. 12:1–12:96. doi:10.1145/2343483.2343495. 1

[SSH12] SEILER M., SPILLMANN J., HARDERS M.: Enriching coarse interactive elastic objects with high-resolution data-driven deformations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012), pp. 9–17. 1

[SSH14] SEILER M., SPILLMANN J., HARDERS M.: Data-driven simulation of detailed surface deformations for surgery training simulators. *Visualization and Computer Graphics, IEEE Transactions on 20*, 10 (Oct 2014), 1379–1391. doi:10.1109/TVCG.2014.2317192. 1

[SVAC12] SAN-VICENTE G., AGUINAGA I., CELIGUETA J. T.: Cubical mass-spring model design based on a tensile deformation test and nonlinear material model. *IEEE Transactions on Visualization and Computer Graphics 18*, 2 (2012), 228–241. doi:10.1109/TVCG.2011.32. 1

[WOR11] WANG H., O'BRIEN J. F., RAMAMOORTHI R.: Data-driven elastic models for cloth: Modeling and measurement. *ACM Trans. Graph. 30*, 4 (2011), 71:1–71:12. doi:10.1145/2010324.1964966. 1