Data-driven cloth simulation

Lisheng Ren, Yihao Zhu

Rensselaer Polytechnic Institute

Abstract

This paper deals with cloth simulation using data-driven approach. Data-driven cloth simulation is based on collecting of experience from a large dataset rather than mimic the real cloth behavior using a physic simulation algorithm. Data-driven approach is advantageous as it is very cheap for calculation if pre-trained offline with sufficient data. However, without large data set, data-driven approach may not behave as expected. We tried cloth simulation using three major machine learning models, including Random forest [Breiman 2001], Linear Regression [Christian Igel 2011] and Neuron Network [Bishop 1995]. In addition, we also applied various ways to extract features from cloth for computer to learn. However, we failed to create a robust simulation eventually. It is shown in our best possible results that the cloth simulation works for one hundred frames*

Keywords: data-driven, cloth simulation, machine learning

* With a time step of 0.001

Data-driven cloth simulation

1. Relevant Works and Inspiration

The first relevant work is the simulation of fluid using regression forest. This paper uses Smoothed Particle Hydrodynamics [Lucy 1977] (SPH) model to generate training data. The input feature is shown as follows.

For single particle, we constructed boxes around it. The boxes have fixed size and have fixed relative position to the particle. The input feature vector is constructed as the average relative velocity of each box. (there are the same number of feature as that of boxes).

This paper did a lot of work to limit the dimension of training data, such as choosing relative velocity instead of using absolute velocity. The reason behind this is most of behaviors of a cloth is symmetry to translation speed. Feature vector using absolute velocity can not capture this symmetry and will require a huge amount of data and training time to ensure the model can acquire the symmetry between cloth behavior and translation velocity through learning.

Our project used the same method in the construction of input vector. However, our case is a little more complex. In fact, a cloth has two coordinate systems, with the first one specifying the position of a particle in space (such as a particle is at x-y-z coordinate of space), and the second one showing particles' topological relation to other particles (two-dimension coordinate such as a particle is of row three and column two). Our input vector should provide such information. Thus, we have two choices, one of which is that each box has fixed x-y-z space coordinate and the feature corresponding to the box contains

information about the topology two-dimension coordinate of particles in that box. The other choice is that each box is fixed in topology coordinate with each feature containing information about space coordinate of particles in a box. The second one is chosen as our input vector.

Also, we get a smaller number of sampling boxes than the paper, reducing the number of boxes and the number of entries in input vector. It might lower the accuracy since input vector contains less information. However, it also reduces the dimensionality while increases the robustness of result.

The second relevant work deals with general methods in emulation of physics-based models using neuron networks. The part is most helpful as it is concerned with neuron network for emulating a spring mass model based dolphins. The dolphin has 23 particles in all. There is no windows or boxes to choose particles for input vector. Since 23 particles are not a great number, they take all the particles as inputs for predicting velocity of any particle. However, we cannot afford to use all the particles as input in our cloth simulation, since cloth have way more particles than dolphin.

2. Validate the possibility

Before starting the final project, we validated the possibility of our project using k-nearst-neighbor [Christian Igel 2001] (abbreviate as k-nn algorithm). For k-nn regression algorithm, when it encounters a test sample, k-nn finds closest k samples in training set. Then it takes the average training output of these k samples as a result. With k equals one, the model will definitely gives zero training error, but it is more likely to overfit. Overfit

means that our learned model is too complex and will only behave correctly on some training datas. k larger than one is a technique called regularization, which constraints the model complexity of learning to increase its resistance to overfit.

The training data is generated from a single training scene during possibility validation. Then the data is used to build a k-nn regressor, which is then plugged into a cloth simulation program to visualize the learning results on training scene.

When the k value is set to one, the data-driven cloth simulation can perfectly reconstruct the training scene. However, when the k is set to higher value, the simulation of the training scene will explode. The reason is that when k larger than one, it will encounter a test sample, and k-nn will find closest k samples in training set. The average training output of these k samples is taken as a result. The error produced during this process could not be fixed by the model though the elapse of time step.

In terms of the testing scene, it does not show any result (it fast explodes even on test scene with almost everything the same as training scene except cloth size). However, we conclude this as insufficient amount of training time. We believe it can eventually works given reasonable amount of training. We think there is a possibility for our approach.

3. Random forest

3.1 Feature selection

As shown in Figure 1, we draw boxes around every vertex of the cloth in two-dimension plain, before collecting vertices in boxes as our samples. As for samples in a single box, we calculate the average three-dimension position and velocity of them relative to the red particle. Then six features are collected for each box, including relative position and velocity, with three dimensions for each. Thus, our total feature number is six multiply box count. In terms of boxes outside cloth edge, we still assume they are existent. However, we artificially put a minus one thousand average velocity and position in three dimensions to indicate they are actually different from other boxes in the training. (If it is for linear models, minus one thousand will cause very serious problem. However, since it is a random forest, it would make a cut in the input space to discriminate those special particles) We want our model can spontaneously learn those boxes with minus one thousand are significant.

1

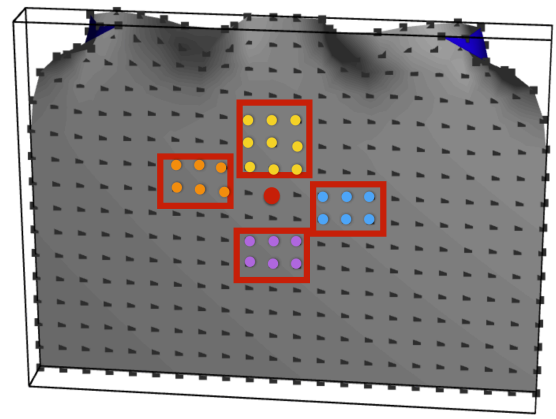

Figure 1: Draw boxed to extract features from the cloth

The learning output is selected as the acceleration of particle.

3.2 Test run one

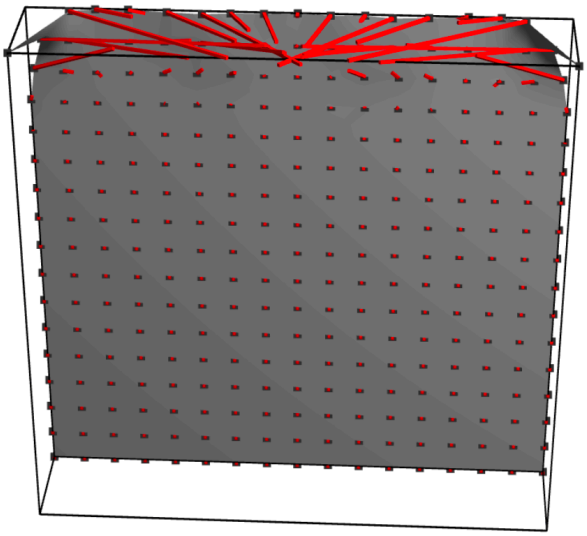

Figure 2: Cloth corner behaves abnormally in test scene

In our first test run, the approach is as shown above. We find that cloth corner behaves reasonably correct in our training scene. But it immediately behaves abnormally after our model is applied to test scene (Figure 2).

We later figure out that it is because of giving large minus feature to boxes outside the cloth, which could mislead our model. In that case, we can conclude the model only learns that fact that a vertex is at the corner, and the model actually ignores the features extracted from vertices around the given vertex. This will make all vertices at the corner behave similarly in all scenes, rather than based on the vertices that actually decide its movement.

We fix this by creating virtual vertices outside the cloth. All virtual vertices move closely following the vertex we draw boxes around. This approach will minimize the impact of empty boxes.

3.3 Test run two

In our test run two, the corner vertices seem fixed. However, after advance some frames as shown in figure 3, we observe that the vertices tend to go down abnormally.

The reason for this bug is that our output in training sample is the acceleration after damping factor is applied. Nevertheless, as described above, the feature we select is relative velocity of vertices in the box. Relative velocity cannot suggest any information about the damping effect, meaning our model could never learn correct damping, as it does not have sufficient information relevant to damping. We fix this by changing model learning output to the acceleration before applying damping, and then apply damping factor to particles according to their velocity.

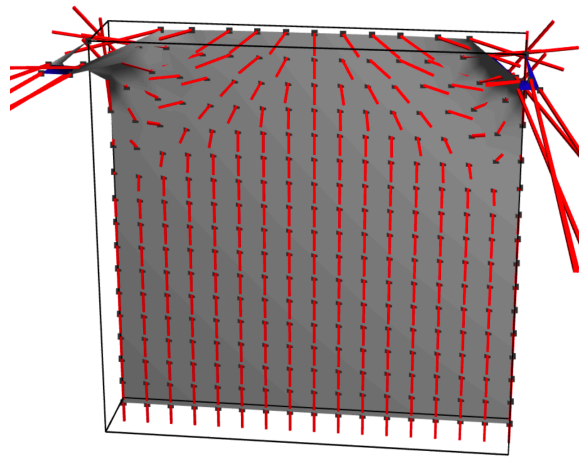

Figure 3: Corners are fixed, but vertices tend to go down

3.4 Test run three

As shown in figure 4a, the cloth now behaves more correctly, but is still away from our ground truth data (figure 4b). It is due to the reason that the particles at corner and edge have different characteristics for behaviors compared with those in the middle of cloth. It means the ground truth of particle in the corner can only learn from training data

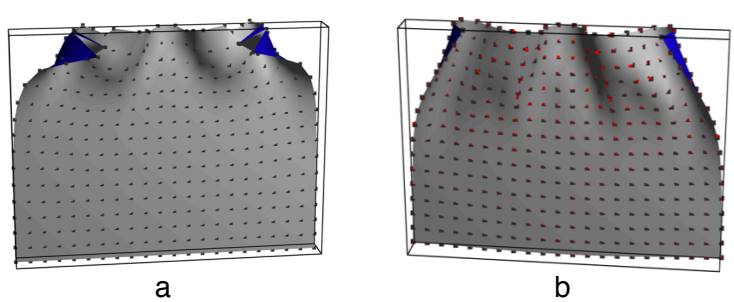

Figure 4: Cloth do not explode on first serval frames. But still behaves significantly different from spring model.

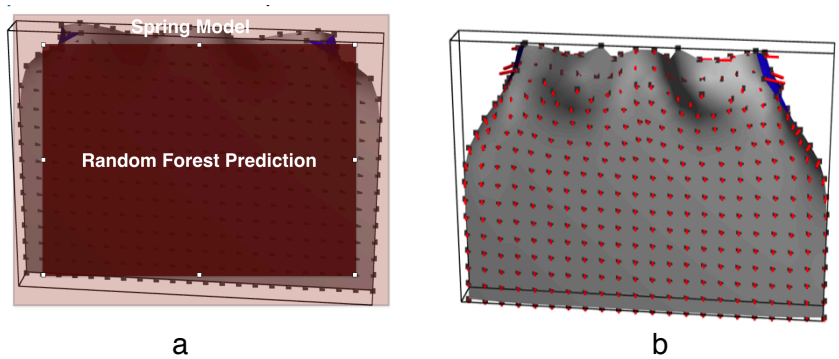

Figure 5: Result of applying two layers of tradition vertices

collected from particles in the corner. Particles in corner have the same dimensionality as those in the middle, which require the same amount of training data. However, there are significantly less training data for corner particles. So we tried to apply traditional spring mass model for two surrounding layers of vertices on the cloth (Figure 5a). It is expected that those two layers can guide our machine learning vertices to behave correctly. However, the result is still disappointing (figure 5b).

We think it is due to the reason that we do not have time to train enough scenes. Since that the less the training samples and the more complex the learning model, the higher the risk for overfitting.

4. Switching to Neuron Networks

We cannot alter the due date of the final project, so we hope our algorithm requires less training to behave correctly. It means that we want a model with a flexible complexity, so that we can adjust parameters to alter training efficiency. We choose to switch to neuron network [Bishop 1995].

4.1 Feature selection

We selected the same features for both input and output for the random forest.

4.2 Result and Analyzation

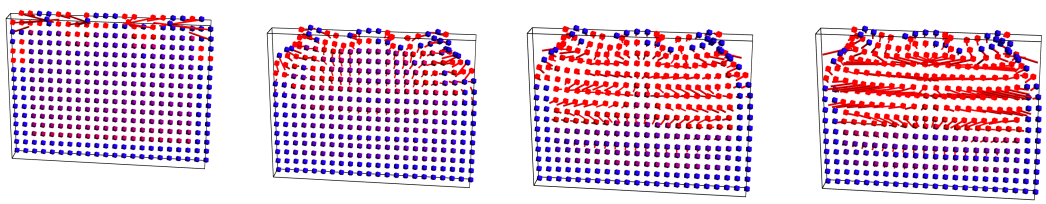

Figure 6: the accumulation of errors. More red means more error

Through a lot of experiments, we obtained our best result for test scene using the neuron network approach mentioned above. In the best test scene, we managed to run the simulation correctly for one hundred frames.

It is shown that the neuron network works significantly worse than random forest on training scenes. It is not surprising since random forest typically overfit the training data.

The errors are shown in figure 6, since the cloth model cannot self-correct the errors. The cloth cannot reach a converge state. The old errors will result in new odd situation and cause further errors. Finally, there will be too large errors and too strange situation, which is

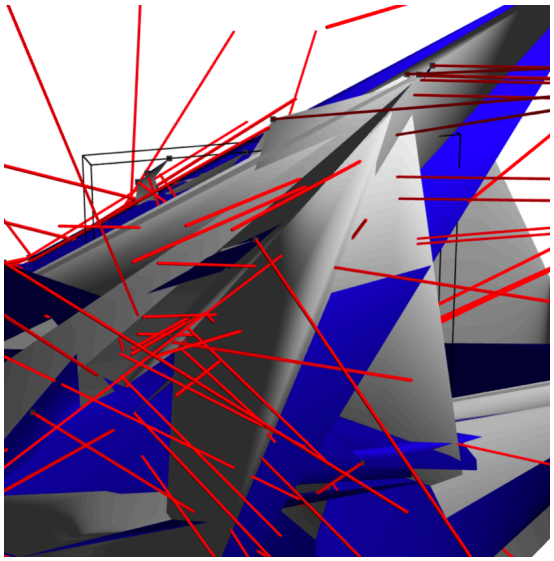

Figure 7: The exploded cloth after 200 frames of simulation

not similar to any samples from any of the training scenes and vertices state of the whole cloth will be totally unreasonable. In this case, the behavior of our model is undefined, as it cannot predict correctly according to surrounding vertices. The cloth will eventually explode as shown in figure 7.

5. Conclusion and future work

In this paper, we tested the data-driven cloth simulation approach. It can be concluded that it is very challenging to apply machine learning in a dynamic continuous model like cloth simulation. The reason behind our conclusion is that it also requires much more training samples to become robust as larger dimension of features gives machine learning model more prediction accuracy. Training a cloth simulation is quite different from predicting a number. When predicting a wrong number, we will not receive any penalty for the next prediction. However, if we predict a vertex wrong, it will have a direct impact on the prediction of all the following frames. If the model cannot effectively correct the errors soon enough (which means before some number of timesteps), it will finally lead to a disastrous result.

Is the data-driven cloth simulation achievable? We believe it is possible as it is a very interesting direction, and we would like to continue to work for it in the future. We need to do more training to see if we can reduce prediction errors to a trivial number to support the cloth to run plausibly within a reasonable amount of time. (For example, ten hours is good for a cloth in a game). We can also figure it out if we can generate training data dynamically during training process according to the performance status of model (such as in which test scene the model works bad). It will increase the efficiency of building up robustness through training.

We also want to make clear if there is an efficient way to enable the cloth to self-correct errors so that it can run infinity long with an exceptional efficiency.

References

Christian Igel. Data Mining: Lecture Notes, chapter 3, 2011

C. M. Bishop. Neural Networks for Pattern Recognition. Clarendon Press, 1995

C. Igel. Data Mining: Lecture Notes, chapter 2, 2011

BREIMAN, L. Random forests. In Machine Learning, 2011

Magdon-Ismail, Malik, et al. Learning From Data, 2012

C. M. Bishop. Neural Networks for Pattern Recognition. Clarendon Press, 1995