Study on Parameter Correction of Spring Particle Model Based on Generative Adversarial Network

Mingjun Wang, Meiliang Wang

Abstract—**This paper is a proposal of a method meant to correct the parameters of the mass-spring model by means of using a Generative Adversarial Networks (GAN). Given that the traditional mass-spring parameters are set randomly during the simulation of the mass-spring model of the cloth, it becomes easy to cause the cloth to be excessively stretched during the simulation of the mass-spring model. Nonetheless, a significant number of researchers have tried to use various methods to effectively adjust the parameters of the model. However, most of the results have not been satisfactory. Therefore in this paper, GAN are used to automatically learn and correct the mass-spring parameters, in order to better solve the problem of over-stretching the cloth generated by the cloth simulation based on the spring particle model.**

Index Terms—**Mass-Spring Model, Generative Adversarial Network, Cloth Simulation, Least Squares Loss Function, Simulation and Analysis**

I. Introduction

FABRICS such as cloth and clothes are flexible deformable bodies, that show a very rich wrinkle effect under the action of various forces. This makes cloth simulation a very complex technology in the virtual reality. The cloth is a distinctive fabric. On the one hand, the textiles field focuses much on accurate modeling using fabric-related dynamics theory, while on the other hand the field of computer graphics puts much of its focus on using virtual simulation technology. This is done to generate the animation effect of realistic fabrics in computers. The past two decades reveal that, although researchers in the textile field are able to use various measuring instruments and methods to obtain various physical parameters of fabrics, they have also gone on to develop some cloth models. However, due to the complexities and the low performance of the computer hardware at that time, most of these models have not been developed and widely used in the field of virtual cloth simulation.

Researchers in the textile field are able to accurately reproduce the dynamic behavior of fabrics such as cloth by establishing the corresponding relationship between the geometric deformation of the cloth and mechanical parameters, such as bending stiffness and Young's model. In 1986, computer graphics began to explore cloth simulation models, and Weil [1] pioneered the cloth simulation based on geometric models. With the improvement of computer hardware performance and the continuous improvement of simulation technology, more and more researchers in the graphics field have begun to gradually introduce research results in the field of textiles, and continue to develop physical cloth-based dynamic modeling methods. In particular, some studies have introduced continuous domain modeling. Compared with discrete domain modeling, continuous domain modeling can better express the micro-characteristics of deformable fabrics objects, such as wrinkles.

Although the continuous model can obtain more realistic cloth simulation effects, the performance of the simulation is still not high based on the current hardware conditions, which restraints its application and promotion in cloth animation. Moreover, cloth is a kind of fabric that is woven from a large number of interlaced yarns, with gaps between the yarns. From a micro perspective, cloth is not a strict continuous medium, and it does not meet the premise assumptions in continuous models, which makes it difficult to accurately express deformation details such as wrinkles of fabrics even in the most complicated continuous models. Mass-Spring Model (MSM) is widely used for cloth simulation and clothing animation research. The model is simple and intuitive, with a small amount of calculation, and can be conveniently used to generate clothing animations [2]. It has been widely used by many researchers [3, 4].

Classic MSM for cloth simulation was proposed by Provot in 1995 [5]. In this model, cloth is not considered as a continuum, but as a grid of rows and columns connected by the spring, as shown in Fig. 1. The nodes of the spring are mass points, and the mass of the cloth is evenly distributed on these mass points. Each spring is assumed to meet the ideal linear elastic relationship, has no mass, and the initial length is not zero. According to the different deformation characteristics of the cloth, the spring is divided into three types: structural spring, shear spring and bending spring. The structural spring is used to maintain the basic shape of the fabric. When the fabric is squeezed or stretched, it will cause the structural spring to deform, and then generate spring stress to prevent further deformation; the shear spring is used to prevent excessive twisting in the diagonal direction when the fabric is subjected to shearing forces; the bending spring is used to prevent excessive bending deformation of the

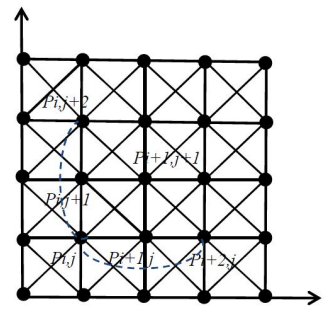
Manuscript received April 2, 2021; revised Aug 21, 2021. This work was supported in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LY17F020003.

Mingjun Wang is a Lecturer of College of Engineering, Lishui University, Lishui, Zhejiang, China (e-mail: mingjun_w@lsu.edu.cn).

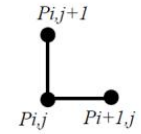
Meiliang Wang (Corresponding Author) is an Associate Professor of College of Engineering, Lishui University, Lishui, Zhejiang, China (phone: 86-578-2299331; fax: 86-578-2299331; e-mail: 229747969@qq. com).

fabric. A large number of experimental results show that in most cases, the force of the bending spring should be much smaller than the others. Otherwise, in the process of animation generation, they are easy to cause the numerical solution to be unstable, making the position of the cloth patch untidy.

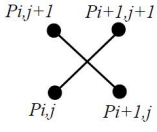
In cloth simulation, the spring is in a state of constant stretching and contraction under the interaction of various internal and external forces. The dynamic simulation of the cloth is essentially the change of the motion state of the particle under the force, that is, the dynamic process of the mass-spring system. However, during the calculation simulation of the mass-spring model, it is easy to have simulation situations of overstretching. This is mainly because it is difficult to accurately adjust the parameters of the mass-spring model during the calculation. As a result, the parameters are adjusted excessively, and the cloth is stretched too much. Many researchers have tried to use various methods to effectively adjust the parameters of the model. However, most of the results are not satisfactory. With the development of artificial intelligence neural networks, there is a deep learning network called Generative Adversarial Networks (GAN). It is an unsupervised learning network, which has been proven to have a very powerful automatic modeling capability. GAN provides a way to learn deep representations without requiring a large amount of labeled training data. This paper uses the powerful modeling capabilities of the generative adversarial network to establish a parameter model of the mass-spring model, and then uses this model to automatically correct the parameters of the mass-spring model.



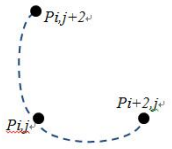
(a) Mass-Spring



(b) structural spring



(c) shear spring



(d) bending spring

Fig. 1. Schematic diagram of Mass-Spring Model

II. Problems with MSM

The mass-spring model proposed by Provot [5] separates the cloth into individual masses, and the motion of the masses conforms to the laws of classical mechanics. It has high calculation efficiency and good performance. In the cloth simulation process, by adjusting the simulation parameters, a variety of different shapes and physical properties can be simulated. The three forces of the spring are simulated inside the fabric. These forces play a role in fixing the fabric structure.

The topological structure of the mass-spring model is shown in Fig. 1. The position of the mass point represents the three-dimensional spatial position of a point in the fabric. The point has a certain mass, but it is negligible and uniformly distributed. The spring connected between the mass points follow Hooke's law. The mass-spring model has some advantages. For example, its mathematical formula is simple and suitable for multiple topological meshing; its structural calculation and graphics processing can be performed simultaneously.

MSM can use the spring connection between particles and the external force to analyze the force relationship of each particle. The internal force is mainly generated by the relationship of the spring following Hooke's law. The specific expression is as follows:

$$F_{inside} = \sum k_{ij}\left(\left|x_j - x_i\right| - l_{ij}^0\right)\frac{(x_j - x_i)}{\left|x_j - x_i\right|} \tag{1}$$

In the formula, k_{ij} represents the Hook's elastic coefficient of the spring between two particles i, j, x_i represents the position of particle i, x_j represents the position of particle j, and l_{ij}^0 represents the original length of the spring between i and j.

The mass point is not only affected by the above internal spring force, but also by external forces such as: tensile force, gravity, damping force, air resistance, etc. :

$$F_{outside} = F_{pull} + F_{\text{gravity}} + F_{\text{damping force}} + F_{\text{air resistance}} \tag{2}$$

At the same time, the sum of the internal and external force vectors on a particle becomes the resultant force on the particle. According to Newton's second law, the acceleration of the particle is calculated as follows:

$$F_{\text{sum}} = F_{inside} + F_{\text{outside}} = ma \tag{3}$$

$$m\frac{\partial^2 X}{\partial t^2} = F_{sum} \tag{4}$$

Where m is the mass of the particle and X is the coordinate vector of the particle.

The advantage of MSM for cloth simulation is that the model is simple, the calculation efficiency is high, the real-time performance of cloth simulation is met, and the nonlinear problems of large rotation and large deformation can be handled relatively easily. However, because the traditional MSM is difficult to correlate the constitutive relationship of the cloth itself, the selection of its spring parameters k_{ij} is relatively arbitrary; therefore, the calculation accuracy is not high, the simulation degree is not realistic enough, and the phenomenon of super elasticity is easy to occur, so this has also become a major limitation in the application of MSM.

MSM's cloth modeling method is to separate the cloth into multiple smaller mass points, each of which is connected to each other by a spring that conforms to Hooke's law. MSM only needs to use the deformation of the spring between the particles, and does not need to establish the stiffness matrix. It uses Newton's second law to directly solve the equation of motion of the particle. This can ingeniously avoid some of the limitations encountered in cloth simulation. It is widely used in deformation simulation of soft materials such as cloth.

The core of using MSM to solve related problems is to determine the spring stiffness in MSM. The value of the spring stiffness will directly affect the calculation accuracy of MSM. In the existing applications that use MSM, the randomness of the spring stiffness coefficient is the main reason for the low accuracy of MSM. At present, some scholars are studying the method of solving stiffness about the spring of MSM. Lloyd[6] divides many of the MSM spring stiffness selection methods into two broad categories. The first is a data-driven method, which uses the actual deformation of the measured object as a reference to estimate the spring stiffness of the MSM. The second type is the analytical derivation method, which relates the constitutive relationship of the materials, and maps the material relationship to the spring coefficient to derive the analytical expression of the related spring stiffness of the MSM. This article attempts to use GAN to establish a spring coefficient model.

III. Generative Adversarial Network

So far, deep learning models can be divided into three types. The first is a generative model structure. This model describes the deep relationship between the data itself and its category attributes. It is expressed as a joint probability in mathematics. This model is often used in unsupervised learning. The results of the research include deep belief networks, auto encoders, deep-level Boltzmann machines, and recurrent neural networks. The second is a discriminative model. This model describes the characteristics of the data and is expressed as conditional probability in mathematics. The most successful research result of this model is the convolutional neural network structure. The third is a hybrid model, which is a network model that contains both a generative model and a discriminant model. Currently, the most widely used hybrid model is to use the generative model to set the initial state of the discriminant model.

The basic framework for the generating adversarial networks (GAN) in 2014 was first proposed by Goodfellow et al. [7], and it belongs to the hybrid model. This model has been proven to have very powerful modeling capabilities. It only needs a small amount of labeled training data to learn deep representation patterns.

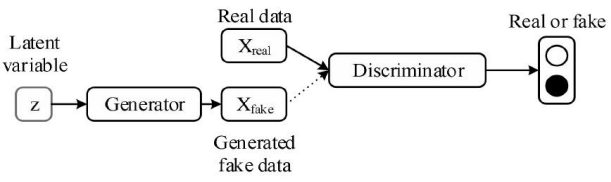


Fig. 2. Schematic diagram of standard generation adversarial network

The theoretical basis of the generative adversarial networks is game theory. The GAN model creates two networks to simulate the zero-sum game between two persons. They are called as Generator and Discriminator respectively, where Generator is used to generate samples and the discriminator is used to discriminate the samples probability. They are completely independent. As shown in Fig. 2, the generative adversarial networks usually use the hidden variable Z (usually set to obey normally distributed random noise) as the network input, generate samples X_{fake} through Generator (G). Discriminator (D) is used to estimate the probability of samples from real data X_{real}. They are updated by the back-propagation algorithm to perform competitive learning to achieve the purpose of training. In continuous iterations, G can obtain the data features of the real samples and generate samples with the same feature distribution as the real samples.

A. General Generative Adversarial Approach

The models of Generating and discriminant are usually composed of multi-layer networks containing convolutional or fully connected layers. P_{data} is the distribution of real data x. To learn to the distribution P_g of the generating models on the data x, to use $G(z)$ for the prior variable $P_z(z)$ of the input noise to represent the data space mapping, and then to output a single scalar by the discriminant model. The training of D and G is a game problem about the minimization of the value function $K(G,D)$.

$$\min_G \max_D K(D,G) = E_{x\sim P_{data}(x)}[\log(D(x))] + E_{z\sim P_z(z)}[\log(1-D(G(z)))] \quad (5)$$

Equation (5) is essentially two optimization problems, which are disassembled into the discriminant model of equation (6) and the generation model of equation (7).

$$\max_D K(D,G) = E_{x\sim P_{data}(x)}[\log(D(x))] + E_{z\sim P_z(z)}[\log(1-D(G(z)))] \quad (6)$$

Through training, D assigns the correct labels to the real data and the samples generated by G as much as possible. After the optimization of the stochastic gradient ascent algorithm, D(x) keeps increasing and D(G(z)) gradually decreases.

$$\min_G K(D,G) = E_{z\sim P_z(z)}[\log(1-D(G(z)))] \quad (7)$$

At the same time, G minimizes log(1-D(G(z))) through the stochastic gradient descent algorithm, so that the sample generated by G has the highest probability of being judged as real data.

In continuous adversarial training, when the equilibrium state of Pg=P_{data} shown in equation (8) is reached, G and D achieve the optimal solution:

$$D_G^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)} = 1/2 \quad (8)$$

The GAN network achieves the best training effect. Generator model captures the data manifold of P_{data} and maps Generator data distribution to the real data.

B. Least Squares Generative Adversarial Method

The least squares loss function [8] is used to replace the cross entropy loss function of the original GAN. The original GAN uses the KL contrast divergence [9] to measure the difference between the distribution P_g and the distribution P_{data}. Minimizing the loss function of the original GAN is equivalent to minimizing Jensen-Shannon contrast divergence [10].

$$S(G)=KL\left(P_{data}\frac{P_{data}+P_g}{2}\right)+KL\left(P_g\frac{P_{data}+P_g}{2}\right)-\log 4 \tag{9}$$

The loss function can be rewritten by the least squares loss function, as follows:

$$\min_D F(D)=\frac{1}{2}E_{x\sim P_{data}(x)}[D(x)-b]^2+\frac{1}{2}E_{z\sim P_z(z)}[D(G(z))-a]^2 \tag{10}$$

$$\min_G F(G)=\frac{1}{2}E_{x\sim P_{data}(x)}[D(x)-c]^2+\frac{1}{2}E_{z\sim P_z(z)}[D(G(z))-c]^2 \tag{11}$$

There is one more term $E_{x\sim P_{data}(x)}[D(x)-c]^2$ in equation (11) than the original loss function. Since it has nothing to do with the G network, it is regarded as a constant. The constants a and b represent the marks of the real image and the generated image respectively, and c is a value determined by the discriminant model to judge that the generated image is real data.

After the model G is fixedly generated, the optimization formula for the discriminating model D is:

$$D^*(x)=\frac{bP_{data}(x)+aP_g(x)}{P_{data}(x)+P_g(x)} \tag{12}$$

The Jensen-Shannon contrast divergence using the least squares loss function is:

$$2S(G)=E_{x\sim P_{data}(x)}[D^*(x)-c]^2+E_{x\sim P_g(x)}[D^*(x)-c]^2 \tag{13}$$

The above optimized loss function is equivalent to minimizing the chi-square contrast divergence [11] between $P_{data}+P_g$ and $2P_g$ under the constraints of $a-c=1$ and $a-b=2$.

$$2S(G)=\chi^2_{Peason}(P_{data}+P_g 2P_g) \tag{14}$$

According to the constraints of a, b and c, the confrontation loss under the least squares loss function is:

$$\min_D F(D)=\frac{1}{2}E_{x\sim P_{data}(x)}[D(x)-1]^2+\frac{1}{2}E_{z\sim P_z(z)}[D(G(z))]^2 \tag{15}$$

$$\min_G F(G)=\frac{1}{2}E_{z\sim P_z(z)}[D(G(z))-1]^2 \tag{16}$$

The losses in Eq(15) and Eq(16) are used to replace the cross-entropy loss function in the conventional GAN network, which has a better conversion effect and a more stable training process than the conventional GAN network.

IV. MSM Parameter Correction

A. Cloth Simulation Algorithm

The cloth changes depending on the movement of the particles. In the cloth simulation based on MSM, the motion of the particle can be expressed in the form of ordinary differential equation with an initial value of zero in the time domain. Generally, the differential equation in MSM can be obtained by numerical integration method to obtain the position and velocity of each particle. In order to more efficiently simulate the dynamic changes of cloth, this paper uses the explicit Euler method with high computational efficiency to solve the dynamic differential equations (as shown in equation (17)). Equations (2) and (3) have analyzed the gravitational, damping, and elastic forces on the cloth particles, and finally the resulting force on the cloth particles is F_{sum}. The dynamic simulation of cloth based on MSM conforms to the law of dynamics, and can calculate the physical quantities such as acceleration, velocity and displacement in discrete time according to the resultant force of the particle:

$$\begin{cases} a_{ij}=\dfrac{F_{sum}}{m} \\ v_{ij}(t+\Delta t)=v_{ij}(t)+a_{ij}(t)\Delta t \\ \mathrm{s}_{ij}(t+\Delta t)=s_{ij}(t)+v_{ij}(t+\Delta t)\Delta t \end{cases} \tag{17}$$

Where discrete simulation time, $t_i=t_{i-1}+\Delta t$, Δt are time steps. At time t, the acceleration of the particle in the i-th row and the j-th column in the cloth patch is represented by $a_{ij}(t)$, and the speed and displacement are $v_{ij}(t)$ and $s_{ij}(t)$ respectively. The corresponding particle velocity at the next time step (at time $t+\Delta t$) is $v_{ij}(t+\Delta t)$ and the displacement is $s_{ij}(t+\Delta t)$. In the cloth dynamic simulation, the time interval $T=\Delta t$ is used for time integration, and the mass points are iteratively updated to simulate the continuous dynamic effect of the cloth in real time.

Because the spring parameters k_{ij} of formula (1) are chosen randomly in the traditional method, it is easy to cause unpredictable sudden changes in the spring parameters k_{ij}, which will cause the cloth simulation to overstretch and seriously affect the simulation results. In this paper, the GAN method is used to learn the real cloth movement to establish a spring parameters model, and then the spring parameters generated by the model are used to numerically solve and simulate the cloth.

B. Parameter K Correction Based on GAN

It can be known from equations (1) and (17) that parameter k_{ij} is a key parameter for cloth simulation. Referring to Fig. 2, the overall design diagram of the parameter K correction of Fig. 3 is designed. Assume that the simulated cloth is represented by a m×n node matrix, the use of the word embedding model (Word2VEC[12]) to generate the cloth description vector v_{text} and the random number generator to generate a Gaussian-Distributed random vector v_{Gau} is described in Fig. 3 , then they are combined into Vector v. Input this vector into Generator to generate fake k_{ij} matrix m_K. Then, through the Cloth Simulator to generate fake simulation pictures, these fake simulation pictures and real pictures are sent to Discriminator at the same time, and finally Discriminator gives the judgment of Real or Fake. Through continuous learning of the GAN, when Discriminator gives real judgment, the generated m×n order k_{ij} matrix m_K can be used to simulate real cloth simulation.

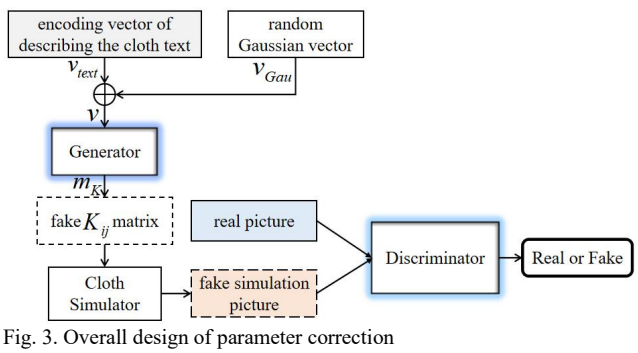


Fig. 3. Overall design of parameter correction

V. GAN Network Design

A. Generative Model

1) Network Design

In 2015, Radford et al. [13] proposed that Deep Convolutional GAN (DCGAN) is a better improvement after the GAN. The improvement is mainly on the network structure, and DCGAN successfully applied the convolutional neural network to the GAN, and greatly improved the stability of the GAN training and the quality of the generated results, so far, the network structure of DCGAN has been widely used.

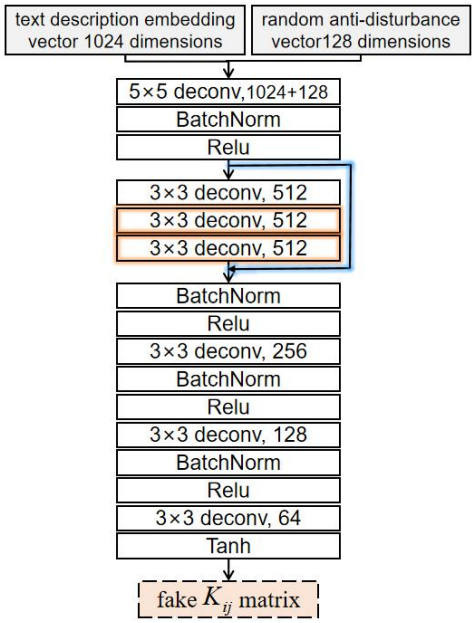


Fig. 4. Generator model structure diagram

This paper improves the original DCGAN network structure in Fig. 4 . The input of the first deconvolutional layer is text description embedding vector 1024 dimensions and random anti-disturbance vector 128 dimensions, a total of 1024 + 128 = 1152 dimensions, and the volume of the first deconvolution layer is modified. The convolution kernel is 5×5, and the convolution kernels of other deconvolution layers are 3×3; at the same time, two layers of deconvolution layers are added, and a residual structure is used to avoid the gradient disappearing; add a batch normalization layer[14] before the deconvolution layer to solve the gradient explosion problem[15]. A batch normalization module is included before the input data of each deconvolution layer, which ensures that the same data distribution is obtained as much as possible to reduce the difference between the input data, and also solves the problem caused by the deepening of the neural network layers. A Relu activation function is added between each deconvolution layer, finally, a Tanh function is used as the output of the matrix.

2) Choice of Loss Function

In order to obtain better image conversion quality, L2 loss is added to the loss function of the generated model on the basis of the least squares loss function equation (16). L2 regularization is to prevent overfitting during training. It is based on the L2 norm, adding an L2 norm term with parameters as a penalty term to the overall generation loss function.

$$\min_G L_2(G) = \frac{\varphi}{2n}\sum_w w^2 \tag{18}$$

In Equation (18), w is the weight in the neural network, n is the number of samples, and φ is the regular term coefficient.

The L2 regularization uses Equation (19) to update the model parameters:

$$w \leftarrow w + \eta\frac{\varphi}{n}w \tag{19}$$

Where η is the update speed of L2 regularization. When using the gradient descent algorithm, η <0, but the opposite is true in the gradient ascent algorithm.

Modifying equation (16), the loss function of the generative model in the basic confrontation loss can be expressed as:

$$\min_G L_s(G) = \frac{1}{2}\left\|D(G(z)) - 1\right\|_2^2 \tag{20}$$

So the total loss function can be expressed as:

$$L_{total} = \alpha L_s + \beta L_2 \tag{21}$$

Where α and β are used as hyper parameters for each part of the generation loss.

B. Discriminant Model

1) Network Design

The adversarial training of the discriminative model and the generative model constitute the generative adversarial neural network. The output of the generated model and the collected real images is trained as the training set of the discriminant model, and the probability of judging the input of the real image and generating a simulated image is used as the output of the discriminant model.

The discriminant model contains 7 convolutional modules in Fig. 5. The output of the convolution module is used as the input of the next convolution module after the batch normalization module. LeakyReLU is added as an activation function between each convolution module to ease the difficulty of network training. Fig. 5 also shows the change in the number of convolution kernels, and a residual module is introduced between three consecutive convolutional layers. Finally, the output is obtained through the fully connected layer and the sigmoid activation function.

Fig. 5 shows the changes in the number of network layers and the size of each layer. Two different sizes of convolution kernels are used in the discriminant model, which are 5×5 and 3×3, respectively. Among them, the step size of the first 6 convolution kernels is 2, and the step size of the last convolution kernel is 1. After continuous feature extraction and dimensionality reduction, the output of the discriminant model is finally obtained.

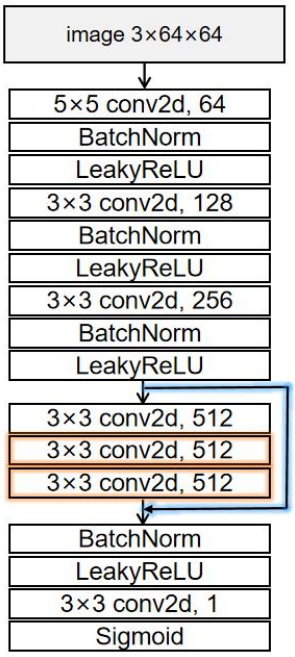


Fig. 5. Discriminator model structure diagram

2) Choice of Loss Function

Compared with the generative model, the task of discriminant model processing is relatively simple, so the complexity of the loss function of discriminant model is relatively small. The discriminant model is an important auxiliary training part of the generative model, and the output of the discriminant model is an important condition for generating network update weights. In this paper, the least squares loss of formula (15) is used as the basic loss function of the discriminant model.

VI. Simulation and Analysis

A. Data Preparation and Training Parameter Settings

ArcSim[16, 17] is an open source engine for simulating thin sheets made of deformable materials (such as paper, cloth, plastic, etc.). The engine fully implements the cloth simulation process in this article, including modules such as physical solution, strain limitation, dynamic mesh reconstruction, and collision detection, and supports deformation and simulation of plastic materials, which can achieve effects such as paper wrinkles and metal dents.

The training data and evaluation data come from the images generated by the simulation. By changing the simulation data, 4615 two-dimensional training images and 1680 two-dimensional evaluation images were generated. At the same time, 1023 two-dimensional image test data are generated, and 20 two-dimensional images are selected for comparison. The size of these two-dimensional images is 64×64.

Deep-level CNNs have a large number of learning parameters and can easily implement various applications, but the extensive expression ability always relies on massive and multiple types of data for iterative training, otherwise overfitting will easily occur. Therefore, in practical scenario applications, data augmentation effectively helps neural networks reduce overfitting and improve model accuracy. In this paper, six methods of flipping left and right, local distortion, rotation, displacement, scaling, and brightness adjustment are used to increase the diversity of images, which can effectively reduce the problem of GAN overfitting and improve model performance.

In order to reduce the difference in data distribution and improve the training speed of the model, the original data pixel value range was moved from [0, 255] to [-1, 1], and the binary file was saved as the pickle format as training data. In addition to increasing the number of samples in the process of expanding the data, the categories of the samples should also be expanded. Therefore, 1% to 6% white Gaussian noise was added to the training data to increase the diversity of the data and improve the anti-interference ability of the model.

Various training parameters in the experiment have an impact on the training of the model, and have a direct impact on the training speed and model quality. Different types of data have different biases on the setting of training parameters. The generated model loss function uses the hyper parameter settings of $\alpha=1.5$ and $\beta=0.1$. The initial learning rate of the model is 0.0001, the initial learning rate decay rate is 0.5, and the learning rate is updated after every 3 training rounds. Batch size is 32. In the model optimization, the standard Adam optimization algorithm is used, and the value of the momentum beta1 is set to 0.9.

B. Adversarial Training of GAN

The great advantage of generative adversarial network in image generation is due to its adversarial training. The generative model and discriminant model affect and restrict each other. In the end, when the discriminant model cannot distinguish between real data and generated data, the generated model has achieved the best training effect.

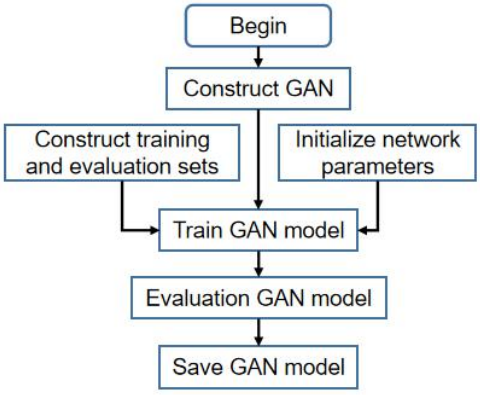


Fig. 6. Model Training Flowchart

The overall model training process is shown in Fig. 6. The implementation of the algorithm is further described below.

1) Construct a generative network. As shown in Fig. 4, the outputs of the activation layer of the first deconvolution layer and the second to the fourth deconvolution layers are stitched together as the inputs of the batch processing layer.

2) Construct a discriminant network. As shown in Fig. 5, the discrimination network is used to output the probability that the image is judged as a real simulation image.

3) Determine the loss function of the generated network. The loss function of the generated network uses formula (21).

4) Determine the loss function of the discriminative network, and use the formula (15).

5) Construct training and evaluation datasets, and limit the pixel values of each image to between -1 and 1.

6) The word embedding model is used to generate the cloth description vector and the random number generator is used to generate a Gaussian-distributed random vector. The two are combined into an input vector.

7) Train the deep generation adversarial neural network, and optimize the loss of the discriminant network and update the network weights through the back-propagation algorithm and adaptive moment estimation algorithm.

8) The network training batch processing set Ψ is used as the training set for generating the network, and the loss of the generated network is optimized and the network weight is updated through backpropagation algorithm, adaptive moment estimation algorithm and L2 regularization.

9) The evaluation data set is used to evaluate the generated network, and the normalized mean square error of the generated image and the evaluation set image is calculated, and the optimal weight is obtained. So far, the training of the deep generation adversarial neural network is finished.

10) The cloth description vector and the vector generated by the random number generator are used as inputs, and the trained neural network model is used to output the MSM model parameters, and the fabric simulation program is used to generate the simulation image.

C. Evaluation

The quality of the generated image needs to be measured by evaluation indicators, and commonly used evaluation indicators include Normalized Mean Square Error (NMSE), Structural Similarity Index (SSIM), and Peak Signal-to-Noise Ratio (PSNR), etc. In addition, the comparison of some models will also reflect the pros and cons of the models.

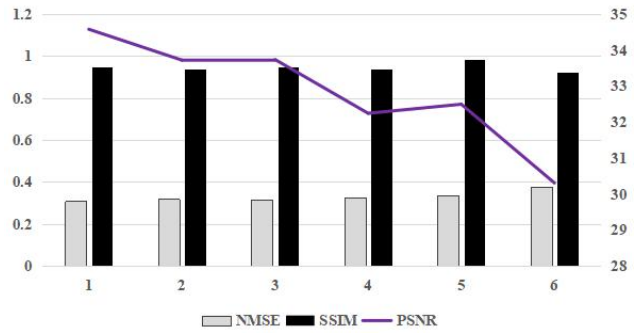


Fig. 7. Evaluation of Conversion Quality under Different Noises

Fig. 7 is the average of the evaluation results of 32 randomly selected images under different Gaussian noise. It can be found that the average PSNR of the converted image is greater than 33.5dB when the Gaussian noise is less than 4%; and when the Gaussian noise ratio is higher than 4%, the PSNR of the converted image is less than 31.4dB. As noise continues to increase, the quality of the conversion decreases. Although the 6% Gaussian noise ratio is not included in the training data, the average SSIM of the converted image is 0.922, and a good conversion effect is still obtained.

Because the sigmoid function in the cross entropy loss can easily reach saturation, GAN training is unstable. The experiment trains the cross entropy loss and the least squares loss under the same conditions, and compares them under the same evaluation criteria. In addition, each loss function affects the characteristics of the transformed image, and will have a significant impact on the resolution and details of the transformed image during the training process. The experiment uses progressive addition under the same conditions and evaluation criteria. The loss function is trained and tested separately, reflecting the effect of the loss function on image conversion more intuitively.

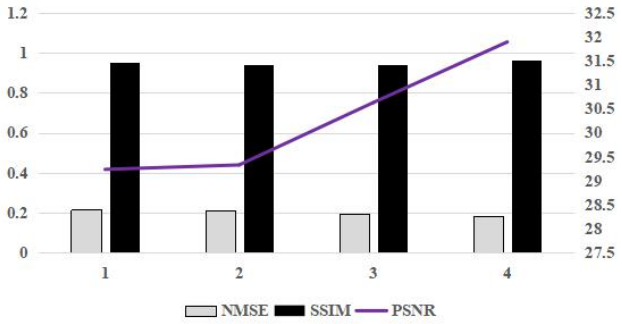


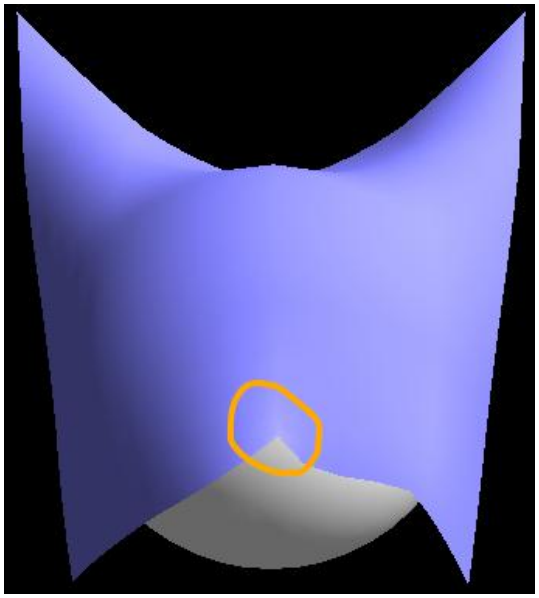
Fig. 8. Evaluation of Conversion Quality under Different Losses

The abscissa of Fig. 8 from left to right indicates that the number of loss functions used is gradually increased. It can be seen that with the increase of the loss function, the NMSE continues to decrease and the conversion quality is improved to varying degrees. At the same time, as the standard deviation of the NMSE continues to decrease, the range of data fluctuations also decreases. As the loss function increases, the GAN also tends to stabilize. When only one loss is used, the average SSIM of the transformed image reaches an observable level of 0.952. As the number of loss functions increases, the conversion accuracy of the adversarial network also improves, the PSNR of the converted image increases by 2.66dB, and the NMSE decreases by 0.032.

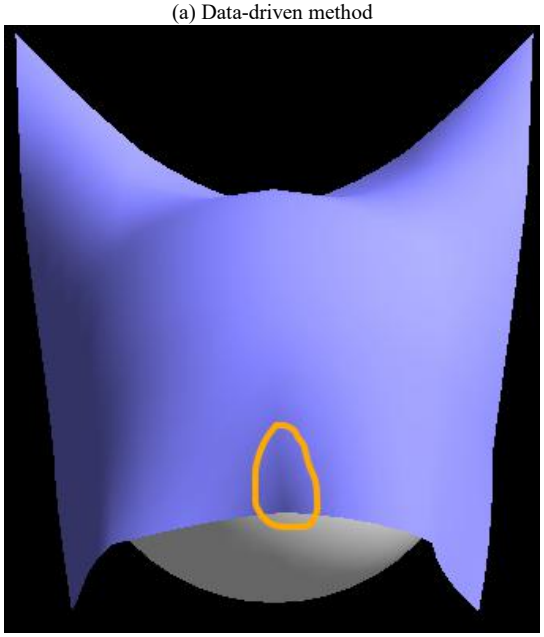
Fig. 9 is a cloth simulation image generated by a simulation program. Fig. 9 (a) is the use of data-driven method, Fig. 9 (b) is the use of analytical derivation method, they are cloth simulation images generated using the traditional mass-spring model parameter generation method. It can be seen that the stretching phenomenon of different programs has been generated in both figures. Fig. 9 (c) is a cloth image generated by using the GAN method to generate the mass-spring model parameters. It can be seen that the image eliminates the cloth stretching phenomenon.

VII. Conclusion

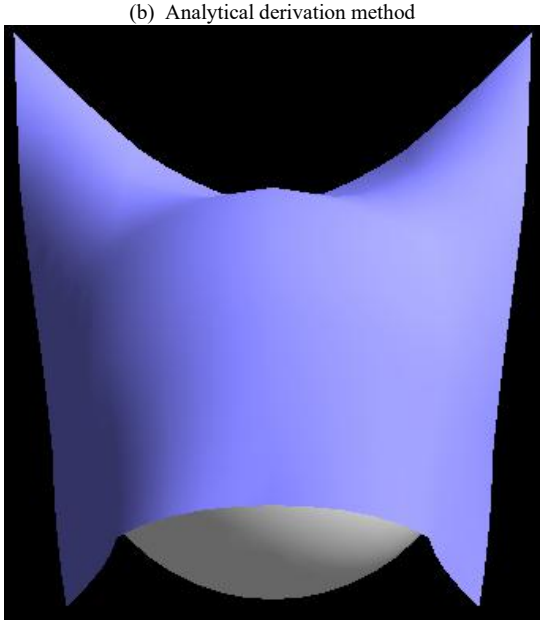
Since the traditional spring particle parameters are set randomly during the simulation of the mass-spring model of the cloth, it is easy to cause the cloth to be excessively stretched during the simulation of the mass-spring model. Numerous, research work has attempted to use various methods to effectively adjust the parameters of the model. However, most of the results are not satisfactory. In this paper, the GAN is used to automatically learn and correct the mass-spring parameters, so as to better solve the problem of over-stretching the cloth generated by the cloth simulation based on the mass-spring model, which provides a solution for such problems [18]. In the future, a hybrid methodology can be combined to modify the GAN in order to obtain better results [19].



(a) Data-driven method



(b) Analytical derivation method



(c) GAN method

Fig. 9. Simulation Comparison Chart

REFERENCES

[1] J. Weil. "The Synthesis of Cloth Objects," *Computer Graphics (SIGGRAPH '86)*, *Proceedings of Computer Graphics Forum 6*, 18-22, August, 1986, Dallas, Texas, vol. 20, pp. 49-54.

[2] T. Vassilev. "Dressing Virtual People," *Proceedings of the 4th World Multiconference on Systemics, Cybernetics, and Informatics (SCI' 2000)*, 23-26, July, 2000, Orlando, Florida, USA, pp. 124-132.

[3] F.S.R. Salazar, B.B. Machado, A. Ocsa, M.C.F.de Oliveira. "Cloth Simulation Using AABB Hierachies and GPU Parallelism," *Brazilian Symposium on Computer Games and Digital Entertainment*, pp. 97-107. 2010.

[4] H.M. Wang, F. Hecht, R. Ramamoorthi, J. O'Brien. "Example-base Wrinkle Synthesis for Clothing Animation," *ACM Transactions on Graphics*, vol. 29, no. 4, pp. 107. 2010.

[5] X. Provot. "Deformation Constraint in a Mass_Spring Model to Describe Rigid Cloth Behavior," *Proceedings of the Graphics Interface 1996 Conference*, 22-24, May, 1996, Toronto, Ontario, Canada, pp. 147- 154.

[6] B. Lloyd, G. Szekely, and M. Harders, "Identification of Spring Parameters for Deformable Object Simulation," *IEEE Transaction On Visualization and Computer Graphics*, vol. 13, no. 5, pp. 1081-1094. 2007.

[7] Goodfellow I, Pouget-Abadie J, Mirza M, et al. "Generative Adversarial Nets," *Advances in neural information processing systems*, vol. 3, no. 11, pp. 2672-2680. 2014.

[8] Mao X, Li Q, Xie H, et al. "Least Squares Generative Adversarial Networks," *Proceedings of the IEEE International Conference on Computer Vision*, 22-29, October, 2017, Venice, Italy, pp. 2794-2802.

[9] Liu H, Wu Z, Li X, et al. "Constrained Nonnegative Matrix Factorization for Image Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp.1299-1311. 2012.

[10] Bai L, Hancock E R. "Graph Kernels from the Jensen-Shannon Divergence," *Journal of mathematical imaging and vision*, vol.47, no. 1-2, pp. 60-69. 2013.

[11] Rathie P N, Coutinho M, Sousa T R, et al. "Stable and Generalized-t Distributions and Applications," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 12, pp. 5088-5096. 2012.

[12] Zhang X, Zhao J, LeCun Y. "Character-Level Convolutional Networks for Textclassification," *Advances in neural information processing systems*, pp. 649-657. 2015.

[13] Radford, A., L. Metz , and S. Chintala . "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." *Proceedings of International Conference on Learning Representations, ICLR 2016*, 2-4, May, 2016, San Juan, Puerto Rico, pp. 1-16.

[14] Ioffe S, Szegedy C. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Proceedings of the 32nd International Conference on Machine Learning*, ICML 2015, 6-11, July, 2015, Lille, France, pp. 1-11.

[15] Pascanu R, Mikolov T, "Bengio Y. on the Difficulty of Training Recurrent Neural Networks," *Proceedings of the 32nd International Conference on Machine Learning*, ICML 2013, 16-21, June, 2013, Atlanta, USA, pp. 1310-1318.

[16] Huamin, Wang, James, et al. "Data-Driven Elastic Models for Cloth: Modeling and Measurement," *ACM Transactions on Graphics*, Vol. 30, No. 4, Article 71, pp. 1-11. 2011.

[17] Liu T, Bargteil A W, O"Brien J F, et al. "Fast Simulation of Mass-Spring Systems," *ACM Transactions on Graphics*, vol. 32, no. 6, pp.1-7. 2013.

[18] Yang G, Yu S, Dong H, et al. "Dagan: Deep De-aliasing Generative Adversarial Networks for Fast Compressed Sensing MRI Reconstruction," IEEE transactions on medical imaging, vol. 37, no. 6, pp. 1310-1321. 2018.

[19] Remal Shaher Al-Gounmeein, and Mohd Tahir Ismail, "Comparing the Performances of Artificial Neural Networks Models Based on Autoregressive Fractionally Integrated Moving Average Models," IAENG International Journal of Computer Science, vol. 48, no.2, pp. 266-276. 2021.