International Journal of Information and Communication Sciences
2018; 3(2): 33-41
http://www.sciencepublishinggroup.com/j/ijics
doi: 10.11648/j.ijics.20180302.13
ISSN: 2575-1700 (Print); ISSN: 2575-1719 (Online)

3D Firework Reconstruction from a Given Videos

Zhihong Wang, Linyi Hu

State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing, China

Email address:
scse1082@126.com (Zhihong Wang), linyi.h@qq.com (Linyi Hu)

To cite this article:
Zhihong Wang, Linyi Hu. 3D Firework Reconstruction from Videos. *International Journal of Information and Communication Sciences*. Vol. 3, No. 2, 2018, pp. 33-41. doi: 10.11648/j.ijics.20180302.13

Received: August 14, 2018; **Accepted**: September 12, 2018; **Published**: September 18, 2018

Abstract: Reconstruction of a 3-dimension(3D) firework show from a given videos is a key technology in light source simulation in computer graphics, which can be more effective and real than traditional method. Although the firework model is already very mature, however, to our best knowledge, there is not any existing method that can reconstruct a firework show from a given video. And due to the lack of camera arguments and depth message, reconstruction is very challenging. In this paper, a method is proposed to solve the problem. A rendering model which requires some parameters which describe the color and position information of firework as input and generates a 3D firework show as output is constructed, and then the problem becomes getting the parameters needed for the rendering model from the given video. The parameters are divided into two groups according to the relevance, and then different neural networks including 3D Convolution Neural Network (3D-CNN) and Recurrent Neural Network(RNN) are designed respectively to extract these parameters needed by our rendering model from a given video. It is found to be practicable and effective to reconstruct a 3D firework from a given video by testing this work with some firework videos in various perspective.

Keywords: 3D-Reconstruction, Neural Networks, Firework

1. Introduction

Scenes in many computer graphics applications contains a large amount of light sources. In the modeling of virtual scenes, setting of light sources plays a crucial role in results of 3D scenes rendering. In addition to these high-precision models, to achieve a high-quality effect, there should be lights which are close to the real. In rendering complex objects and large-scale scenes, the setting of light sources or the determining the light sources properties has become one of the most important aspects in 3D scene modeling. Currently, in order to set multiple light sources which provide a more real sensation in complex scenes, professional artists will invest a lot of time in designing the light sources in the scene, which is not only cumbersome but time-consuming. To improve the modeling efficiency and simulate the position of the light sources more quickly, it is indispensable to automatically set the light source. In this article, it is focused on the fireworks that can be used as light sources. Current fireworks models are based on estimating the explosion equation of the fireworks based on prior knowledge, and then artificially set these parameters needed for the model to make the effect looks like a natural fireworks. It also cost a lot of time to pursue a more real effect, or be close to a real fireworks model. Therefore, it is worth to find a method that can automatically extract parameters from given videos and to reconstruct a three-dimensional fireworks model.

In this paper, it is focused on learning the explosion patterns, trajectory of the particles, and the color changing rules from given firework videos, and then construct a 3D fireworks which is similar to the given video. The firework model built on the basis of video accords with the real motion law of the fireworks and it can provide a more realistic lighting source for the scene. It not only saves a lot of time but enhances the user's immersion by analysis the audio automatically. To accomplish this task, this paper first build a model that generates fireworks from a series of parameters, referring to present models and slightly improved them. Then parameters are divided into two groups according to the relevance of them as well as the difficulty of extracting them. Different methods are designed for each group, including traditional image processing methods and neural networks, to analysis and extract these parameters from the

given video. Finally, the parameters are passed into the rendering model to build a three-dimensional fireworks effect, which is used to compare with the original video. Figure 1 shows the effect of our method. The three rows are different videos, the first column is the given video, and the other three columns is the prediction under three different view space at the same time.

Figure 1. *Effects of our method.*

To the best of our knowledge, we for the first time, proposed a method that automatically extract parameters from the fireworks video, and then reconstruct a three-dimensional fireworks, which can be used as light scenes in big scene, from these parameters. This paper improved the existing fireworks model to make it more suitable for our problem, and designed a neural network structure to analyze the video efficiently, and finally a simple but effective method was proposed for obtaining these parameters from the video. After verification, it is found that the 3D firework effect generated by this method is roughly similar to the fireworks of the original video.

2. Prior Work

2.1. Prior Model of Fireworks

Currently, almost every firework rendering algorithms are based on independent particle system [1, 2], which means that the forces acting on the particles are mutually independent and have no influence on each other.

The render of the three-dimension fireworks with particle system on computer graphics was first proposed by Loke [3]. They proposed some characteristics of some fireworks, including the properties such as color, shape, size, speed, position, and special effects such as groin, star, rotation and so on. Combined with the research of some other scholars [4], they are concluded the following conclusions:

Color: color is produced by the burning of metal. The colors will not be fixed during the process of fireworks display, and they gradually dim from the initially bright colors to disappear.

Transparency: In the actual fireworks display, as the particles themselves continue to burn, they slowly fade and disappear at the final of the display, which can be obtained by processing the transparency characteristics of the particles.

Size: As the particle is burned, its size gradually becomes smaller. This can be achieved by changing the particle size or set an attribute to describe the changing rate of particle size.

Trail: When watching a fireworks display, you can see an obvious trailing trail behind the bright spot of movement. Therefore, when simulating a firework, how to display the trail of fireworks is very important.

Movement: When a particle is born, it is given an initial position and an initial speed. After that, the particles continue to move to a new position and get a new speed. The particle motion can be described by the following equation: $ds=u(t)dt$, where dt is the time increment, $u(t)$ is the particle's velocity function, and ds is the displacement increment.

Fireworks generated according to these rules above is already very real. This paper have constructed a model for rendering three-dimensional fireworks based on these rules and assumptions, which will be discussed in detail in the section

2.2. Convolution Neural Network and Recurrent Neural Network

Convolution neural network (CNN, or ConvNet) is first used to solve document recognition problem [5]. Recently it is widely used in image classification on large dataset such as ImageNet [6]. A ConvNet contains a feature extractor

composed of convolution layers and sub sampling layers, which is the difference between a ConvNet and an ordinary neural network. There are usually several feature maps in a convolution layer of CNN, and each feature plane consists of a number of neurons that arranged as matrix. The neurons in the same feature plane will share the same weights, which in fact are called convolution kernel.

The convolution kernel is usually initialized in the form of a random decimal matrix [7] whose value are usually very small and the bias should set to zero. The convolution kernel will gradually learn reasonable weights during the network training process. One of the benefit of sharing weights is to reduce the connections between layers of the network and at the same time reducing the risk of over-fitting. Sub-sampling layer, also known as pooling, usually perform in two forms: mean pooling and max pooling [8]. Sub-sampling can be viewed as a special kind of convolution process. Convolution and sub-sampling greatly simplify model complexity and reduce model parameters.

Recurrent neural network(RNN) is widely used in information processing with timing characteristics such as voice and video. RNN is a neural network that models sequence data, which means that the current output of a sequence is also related to the previous output [9]. The specific manifestation is that the network will memorize the previous information and apply it to the calculation of the current output. In other words, the nodes between the hidden layers are connected, different from traditional networks, and the input of the hidden layer not only includes the output of the input layer but includes the output of hidden layers.

The Long Short-Term Memory Neural Network (LSTM), which is also a time-recursive neural network, is a special type of RNN. LSTM was proposed by Hochreiter & Schmidhuber in 1997 [10] and was improved and promoted by Alex Graves in recent years [11]. The main difference between LSTM and RNN is that it adds a processor used to judge whether information is useful or not, which is usually called cell. A cell consists of three gates, called the input gate, the forgot gate, and the output gate. A forgot gate is used to determine whether a message entered in the LSTM network is useful or not [12]. Only the information that meets the algorithm's certification will remain, and the inconsistent information will be forgotten through the forget gate. So long-term dependence information can be learnt with LSTM network, and it is suitable for processing and predicting important events with relatively long intervals and delays in time series.

3. Methodology

3.1. Problem Setup

Our goal is to learn the variation rule of particle attributes, such as color, position and size, from a given fireworks video, and then render a 3D fireworks with this rule which should be as close as possible to the original video. To achieve this goal, this work first built a computer graphic model with openGL for rendering 3D fireworks, which accepts 25 parameters and will be discussed later in section 3.2.2. So the question can be translated into how to learn these 25 parameters from a given fireworks video, and the difference between the learned parameter and its real value should be as small as possible, in other words, it should be guaranteed that the firework show rendered looks like the given video.

Suppose we have the labelled source training dataset $Ds = \{V, a\}$ with N training instances and M attributes. V denotes the training videos, while a is the arguments, and we have the knowledge that M is equal to 25 in this problem. The i-th vector whose size is M in a is the arguments of the i-th video which is also expressed as V^i in V. Given a new video V^*, the goal is then to learn a function $a^* = V^*$ with all available training information and predict the argument vector a^*.

3.2. 3D Firework Rendering Model

3.2.1. Overview

This paper design a new model called differential firework model(DFM) based on the principle of differential to render the target firework with some arguments. As is seen, after the explosion of fireworks, several particles were scattered from the center of the explosion. The system can be can analyzed with the ensemble-isolation method. On the whole, the system has an initial position, and it is also affected by gravity and other external forces such as wind. In isolation, the particles spreads from the explosion center with a certain centrifugal speed at a unique angle. The size and color of particles are attenuated in a certain proportion due to constant combustion during the movement of particles. The particles will leave ashes, which will continue to burn with their color changing during burning, in the movement process. They are the leaving ashes that look like trails. All particles in a frame shares the same color and size, and all ashes generated at the same frame shares the same color and size as well.

In the model, given the number of particles in the fireworks section, the number of particles and the direction of each particle will be easily calculated with algorithm 1, which is calculated under spherical coordinate system. After that, for each frame indexed by i, this work will calculate the color, size and distance form explosion centers of the ashes generated by all other frames before the i-th frame. It is too sparse because each particle can generate only one ash in one frame, so I have to insert several extra ashes uniformly between original ashes and calculated the color and size of each ash on the particle motion trajectory with the interpolation method. As shown in figure 2, if the ashes is dense enough, it can be used to simulate various shapes after smoothing the edges.

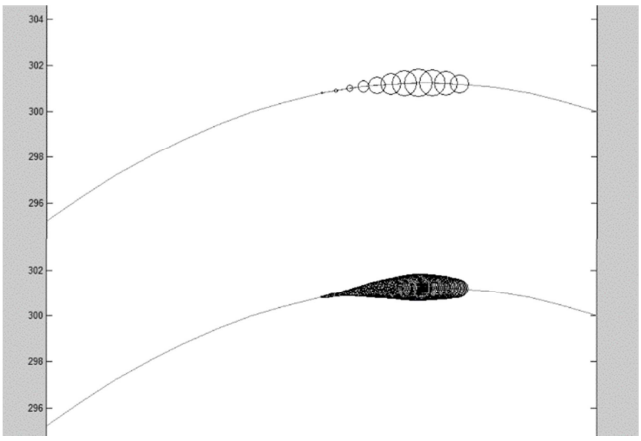

Figure 2. *Insert ashes to get trail.*

Algorithm 1: Initialize directions
Require:N>0 and N is integer
Output:dirs.
1:n←(N/2)+1
2:α←2∗π/N
3:for i=0 to n do
4: θ←i∗α
5: m←max(N∗sin(θ), 1)
6: β←2∗π/m
7: for j=0 to m do
8: γ←j∗β
9: dir←(sin(θ)∗sin(γ), cos(θ), sin(θ)∗cos(γ))
10: dirs. push(dir)
11: end for
12:end for

3.2.2. Parameters

This work have constructed a model that requires several parameters. The first parameter is the number of particles in the cross section of fireworks, through which this work can generate particles in all directions. Then it is needed to describe the distance between the particle and the explosion center without considering the external force, and the distance between the fireworks and the origin in the horizontal and vertical direction under the external force. In order to achieve a better accuracy, a cubic formula is used to fit each distance. For each distance, which can be described as:

$$\mathrm{d} = w_0 * t^3 + w_1 * t^2 + w_2 * t + w_3 \tag{1}$$

Where d is the distance, w_0 to w_3 are the parameters describing this distance, w_0 is correction term to make the model more real and it is usually relatively small. This paper tried more correction term with higher power, and find that cubic is the best choice to fit the model. w_1 is regard as the acceleration, w_2 is called the initial speed, and w_3 is the initial position. For these three distances, each description is represented by four parameters, so there are 12 parameters to describe the state of the particles. For each particle in direction dir, the position can be calculated as Eq2:

$$\begin{cases} d_c = w_0 * t^3 + w_1 * t^2 + w_2 * t + w_3 \\ d_x = w_4 * t^3 + w_5 * t^2 + w_6 * t + w_7 \\ d_y = w_8 * t^3 + w_9 * t^2 + w_{10} * t + w_{11} \\ \mathrm{p} = d_c * dir + (d_x,\ d_y, 0) \end{cases} \tag{2}$$

where d_c denotes the centrifugal distance, d_x and d_y denotes the distance between the fireworks and the origin in the horizontal and vertical direction respective. t denotes the time of present frame, p denotes the final position arranged as a 3-dimension vector and *dir* denotes the direction which is a 3-dimension orthogonal vector.

In addition to the motion, some additional parameters are also needed to describe the status of particles, including the initial color of the particle, the initial size of the particle, the changing rate of color and size of particles over time, and the changing rate of color and size of ashes over time. Because the color has three channels of RGB, a total of 12 parameters are needed to describe the state change of the particles. All parameters and its division will be described in section 3.3.2.

3.2.3. Randomization

To get a more real visual effect, some random element are added to our DFM. This paper set random coefficients for the centrifugal direction and centrifugal speed of particles, and they are implemented in the initialization of the centrifugal direction in order to pursue higher computational efficiency. The coefficients of speed is added to the unit direction vector. The calculation of the direction is changed from line 9 in algorithm 1 to algorithm 2:

Algorithm 2: Get Randomized Direction
Require:0<c<1 denotes the randomization range
θ, γ is the same in algorithm 1
Output:dir
1: $r_v \leftarrow \mathrm{rand}(-1,1) * \mathrm{c} + 1$
2: $r_x \leftarrow \mathrm{rand}(-1,1) * \mathrm{c} + 1$
3: $r_y \leftarrow \mathrm{rand}(-1,1) * \mathrm{c} + 1$
4: $r_z \leftarrow \mathrm{rand}(-1,1) * \mathrm{c} + 1$
5:dir $\leftarrow (\sin(\theta) * \sin(\gamma) * r_x, \cos(\theta) * r_y, \sin(\theta) * \cos(\gamma) * r_z)$
6: dir $\leftarrow r_v * \mathrm{norm}(\mathrm{dir})$

In addition, the parameter of camera are also randomized, such as initial position and perspective. It is ensured that the camera's position was below the fireworks but not directly below it to simulate the actual photographic process, and the focus of the camera was near the center of the explosion and remained stationary.

3.3. Dataset and Labels

3.3.1. Dataset

Because of the novelty of this work, there is not any existing dataset, so a script was written to generate the dataset with our DFM described in section 3.2.

This work first set up a parameter generator, which ensures that the generated parameters obey the following two rules: 1. In the initial case, the fireworks will not be too dark, that is,

the highest value of RGB is no less than 0.5, otherwise the brightness will be adjusted at the same random rate to ensure the highest value is more than 0.5; 2. The fireworks will not completely disappear in at least 40 cycles, that is, the brightness and size of the fireworks will not be too small. In addition, the initial position of the explosion and the viewing angle of the camera are randomly selected. The viewing angle is fixed during the fireworks explosion.

12,000 videos was generated as our dataset with this script. All the arguments the model need is also what this paper want to extract from the input videos, so the labels of this regression problem is the arguments which are these videos generated with. For each video 25 parameters needed to regress as labels was saved.

3.3.2. Label Group

According to the relevance of the parameters, all the 25 parameters are divided into two groups.

As table 1 shows, The first group has 13 parameters which describe the particle's motion characteristics, including the number of particles, the initial velocity and exchange of velocity, the initial position of the whole particle system, and its velocity in the horizontal and vertical directions. Each descriptions of velocity consists of three parameters. The second group has 12 parameters, which describes the particle status, including the initial size and color of the particles, the decay rate of size and color of particles, as well as the decay rate of size and color of the ash.

3.3.3. Auxiliary Labels

The particles are dispersed from all directions at the same initial velocity from the explosion center, and the air resistance is basically the same when the particles are dispersed. The exchange of velocity is basically the same regardless of the wind force and gravity. Since the wind and gravity of the entire particle system are basically the same as well, consider the particle system as a whole. Under the action of external forces, this whole system will move in the same way. Therefore, at any time, all particles will be approximately on the same sphere, which reflected in the video is that the outer particles are approximately on the same circle. So a circle can be fit to represent the motion of the particles for each frame of the video. The change in the radius of the circle is just the change of the position of the particle from the center of the explosion regardless of the external force, and the change of the center reflects the change of the position of the entire particle system under the influence of the external force. And further it can obtained the exchange of speed of the whole system.

For each video in our data set, this work obtained a circle for each frame by calculating the frame rate of the tag and the video, which will be used as an auxiliary tag for the video. Fireworks in some videos will completely disappear after a certain frame, so all the auxiliary labels of the following frames is needed to set to 0, otherwise it is very difficult for the models to learn the auxiliary parameters, for the dark frame can reflect to various different values.

Table 1. *Parameters and division.*

Group	Function	Number	Description
1	number	1	the number of particles in the cross section of firework
	centrifugal distance	4	the distance between particle and explosion center
	horizontal deviation	4	horizontal deviation under external forces
	vertical deviation	4	vertical deviation under external forces
2	start color	3	the start color of the particles
	particle color decay	3	decay rate of the colors of particles
	ash color decay	3	decay rate of the colors of ashes
	start size	1	the start size of the particles
	particle size decay	1	decay rate of the size of particles
	ash size decay	1	decay rate of the size of ashes

3.4. Criteria

If training an independent regression model for each individual argument or label [13], then this can be modeled as minimizing the expected loss over all the training instances for the j-th attribute a_j; and it leads to the following formulation as:

$$\Theta_j = \text{argmin}_{\Theta_j} \sum_{i=1}^{N} \Phi(\Psi_j\left(V^i; \Theta_j\right) - a_j^i) \qquad (3)$$

where Θ_j in Eq(3) indicates the optimized parameter set of the *j*-th argument prediction network;a_j^i is the *j*-th argument of the *i*-th video; and $\Phi()$ is the loss function penalized the value differences of predicted attributes and ground-truth arguments. The $\Phi()$ can be mean square error (MSE) loss or any other loss functions [14]. In our experiments, there is no significant difference of these loss functions; and MSE will be used to evaluate our result.

As the previous section described, this work divide the arguments into two groups, so it will jointly optimize all the prediction tasks in a group at once, and the model of Eq (3) will be extended as:

$$\Theta = \text{argmin}_{\Theta} \sum_{i=1}^{N} \sum_{j=1}^{M} \Phi(\Psi_j\left(V^i; \Theta_j\right) - a_j^i \qquad (4)$$

where Θ denotes the parameter shared across all the tasks in one group.

3.5. Algorithm Flow

To get the parameters needed by our rendering model from the video, one of the simplest algorithms is to use a 3D ConvNet, as is shown in section 4.1, to extract parameters from the video. The results are analyzed to found that some

of the parameters, including all parameters in group2, were better learned with the deviation within 5%. While there are many parameters that are not so well satisfying. Therefore, it is needed to design other algorithms to solve these parameters.

As is shown in figure 3, this work extract the given video into frames as well as get the differences frames between two adjacent frames in the video except for the first frame. For the parameters in group 1, the last softmax layer is replaced in a conv2d model called inceptionV3[15] with a fully connected layer, whose size is the same as the size our auxiliary labels. This work retrain its parameters to fit our secondary tags got in section 3.3.3. Then the last layer from our trained model is removed to get features [16]. This paper predict each frame of every video in the data set and save the output as the feature sequences of its frames, and these frame feature sequences will then be regard as the input to the LSTM model, which will be discussed in detail in section 4.2.

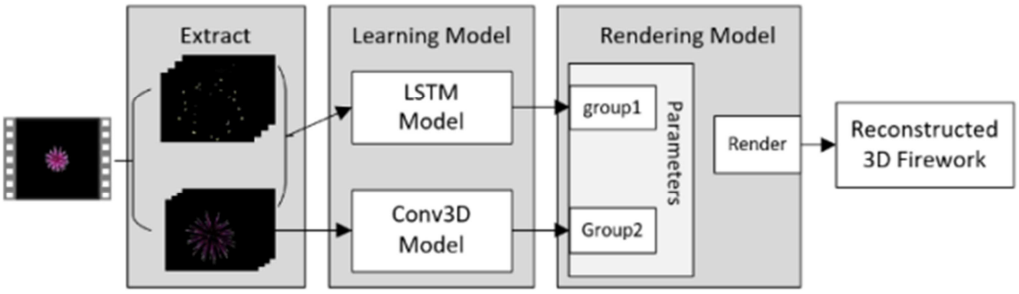

Figure 3. *Overview of our algorithm.*

For some of the parameters in group 2, it can use 3D-CNN, or it can extract features with any 2D ConvNet architecture and then use LSTM to learn the temporal features, which is good enough as well.

4. Network Structure

This section explain in detail how our network is designed, and elaborate how to train them on our dataset to extract parameters from given video.

4.1. Modeling with 3D ConvNets

Compared to 2D ConvNet, 3D ConvNet has the ability to model temporal information better due to 3D convolution and 3D pooling layers [17], this paper establish a 3D convolution neural network with frames of each video as input and arguments in group 2 as output.

Suppose the size of our videos is clipped to l*h*w*c, where l is the length in number of frames, h and w are the height and width respectively, c is the number of channels in each frame. The size of 3D convolution kernel and pooling kernel are also referred by d*k*k or (d, k, k), where d is the depth while k is the spatial the same as it of 2D convolution kernel.

4.1.1. Network Architectures

This section describe the network architecture in detail. The network is consisted of 10 convolution layers, 5 pooling layers, 1 flatten layers, and 3 fully connect layers after the flatten layers. The structure of convolution and pooling layers are shown in table 2, which is consisted of six layer groups. The number of filter for each group are 3, 6, 12, 24, 48, 96, which is limited to the memory of our gpu. The first Conv3D layer has a kernel size of (1, 3, 3) and a stride of (1, 2, 2), and it is designed to down-sample each frame of videos adaptively. All the other Conv3D layers have a kernel size of (3, 3, 3), for it is learned [18] that a kernel size of (3, 3, 3) is the best choice for C3D model to extract features. All pooling layers are max pooling with the kernel size and stride equals to (2, 2, 2) except for the first and third pooling layers, because I do not want temporal information merge too early.

Table 2. *ConvNet Architecture.*

	Layer	filters	Kernel	stride	padding
1	Conv3D	3	(1, 3, 3)	(1,2,2)	valid
2	Conv3D	6	(3, 3, 3)		same
	MaxPooling3D		(1, 2, 2)	(1,2,2)	valid
3	Conv3D	12	(3, 3, 3)		same
	Conv3D	12	(3, 3, 3)		same
	MaxPooling3D		(2, 2, 2)	(2,2,2)	valid
4	Conv3D	24	(3, 3, 3)		same
	Conv3D	24	(3, 3, 3)		same
	MaxPooling3D		(1, 2, 2)	(1,2,2)	valid
5	Conv3D	48	(3, 3, 3)		same
	Conv3D	48	(3, 3, 3)		same
	MaxPooling3D		(2, 2, 2)	(2,2,2)	valid
6	Conv3D	96	(3, 3, 3)		same
	Conv3D	96	(3, 3, 3)		same
	MaxPooling3D		(2, 2, 2)	(2,2,2)	valid

This work add the flatten layers after the last pooling layer in order to reshape the output from a 4D matrix to a tensor. The first two fully connect layer has 2048 outputs with a dropout rate at 0.5, while the final layer has 12 outputs, which is equal to the number of parameters in group 2.

4.1.2. Network Setting

The size of videos in dataset is (96, 600, 800, 3), and reshape all the videos into (48, 300, 400, 3). It can not clip them into smaller size because of the lost of details, which is important for calculating parameters. For each convolution layer and fully connect layer, this work add a l2 regularization with the weight of 0.00001 in order to avoid over-fitting. The loss function is set to be mean square error which is the same as equation (4). The batch size is set to 16

and train 70% samples per epoch. It will train the model for 1000 epochs and stop training if the loss has not decreased in recent 20 epochs. This work choose Adam [19], which can adaptively change the learning rate and avoid vanishing learning rate, to optimize our loss function. Because of the mean square error loss function, the initial learning rate should be very small, and set it to be 1e-5. What is more, the decay is set to 1e-6, β_1 is set to 0.9 and β_2 is set to 0.999.

4.1.3. Training and Result

To make all parameters have the similar weight while training, first normalize the labels with a min-max-scaler to rearrange the labels between -1 and 1. `tanh' is picked up as the activation functions for fully connect layers because `tanh' can generate negative numbers as result which is different from `relu'. This work randomly pick 10000 videos as our train set and 2000 videos as validation set to train our models.

After training, the mean square error of all parameters in group 2 is decreased to 0.0731.

This work add different coefficient to our loss function in order to focus more on the parameters that are learned not so well as others, and then the model of Eq (4) can be extended as:

$$\Theta = \text{argmin}_{\Theta} \sum_{i=1}^{N} \sum_{j=1}^{M} \lambda_j * \Phi(\Psi_j(V^i; \Theta_j) - a_j^i \quad (5)$$

Where λ_j denotes the coefficient of the j-th parameter in loss function. Now it can pay more attention to the parameters that are not so good as others, and at the same time, it should be ensured that the average of all the coefficients is 1. The other settings is kept fixed and train the model again, the loss is decreased to 0.0832 and the mean square error of all parameters is decreased to 0.0456.

4.2. Modeling with Inception V3 and LSTM

This paper use a model with Inception V3 and LSTM to solve the parameters in group 1, which is shown in Figure 4, which will be discussed in detail in the next several sub-sections.

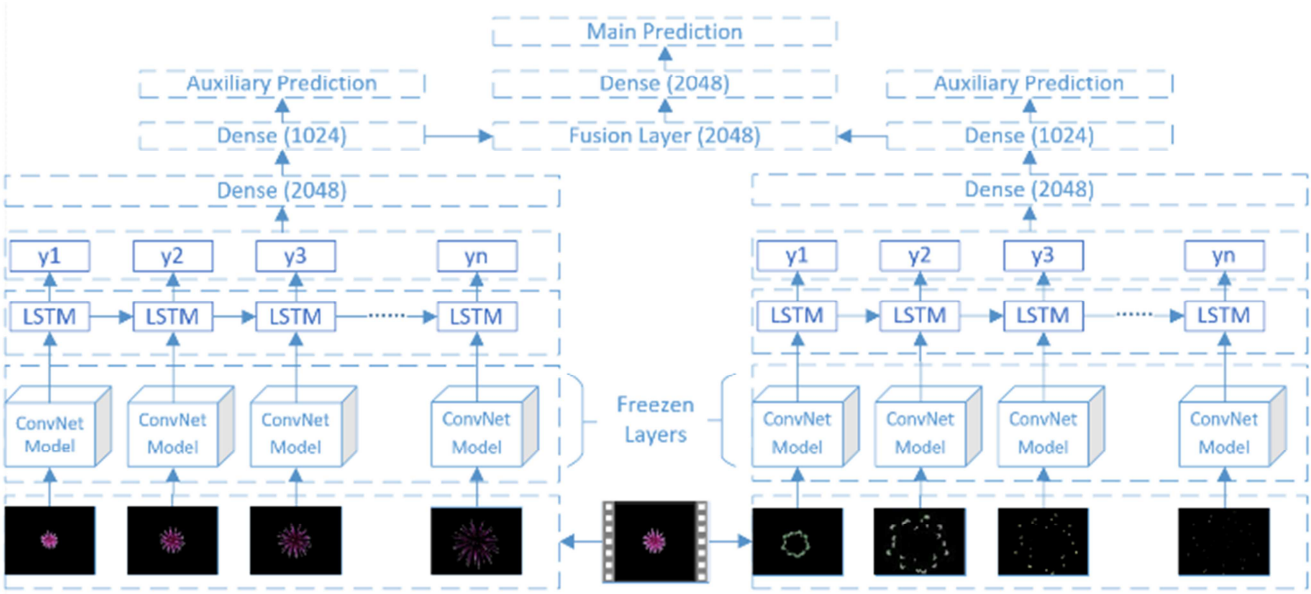

Figure 4. *Architectures of model with Inception V3 and LSTM.*

4.2.1. Feature Extraction

The LSTM layer requires a sequence of features arranged in tensor instead of frames arranged in 3-dimension matrix, I have to extract the features, which the LSTM layers can make use of, for each frame in videos with a 2D-ConvNet model as which inception V3 is chosen in this work. The model is used to exchange pictures into features, whose input is a picture in 3 dimensions and output is a tensor. The final softmax layers was replaced with a fully connect layer at a size of number of auxiliary labels described by section 3.3.3, because I want to regress the labels instead of classify them. Information in difference frames describe the color and location of particles except for ashes in the frame, which is closely related to the circle fitting. Mean square error is used as the loss function and pick Adam with default setting to optimize the loss function. The model is trained to fit the auxiliary labels with the original frames and difference frames and save the weights separately.

Directly extract videos into auxiliary labels may lose precision due to the deviation of our trained 2D-ConvNet model. So the final fully connect layers will be removed and features are extracted at the final pool layer with the size of 2048, which can remain more information than getting auxiliary labels. This work pre-process each video, exchange the original frames and difference frames into sequences separately with the two trained models correspondingly.

4.2.2. Network Architectures

As figure 4 shown, after exchanging spatial frames into

two different types of features for each video respectively, the features are sent into two sets of LSTM networks for temporal modeling as shown in left and right. The outputs of two LSTM model are combined to generate the final prediction, and each side has its own prediction as the auxiliary of the main prediction. All the predictions are used to fit the same parameters in group 1, and this will make the network on each side of the figure pay more attention to predict the parameters, while the combined fully connect layers will pay more attention to find the intersection between original frames and difference frames and combines the results together. The LSTM layer and all fully connect layers except for prediction layers have a dropout rate at 0.5.

4.2.3. Network Setting

The size of videos in dataset is (96, 600, 800, 3), and all the spatial frames in each video are reshaped into (300, 400, 3) to extract features from modified inception V3 models. The length of features for each frame is 2048, which is the size of the `avg_pool' layers in inception V3. The weight of l2 regularization of LSTM layer and fully connect layer is also set to 0.00001. Mean square error shown in Eq 5 is continue to be used as the loss function. As the whole model has three outputs, I combine the three loss functions by adding them with different coefficients as the final loss function. The loss function of main prediction has the coefficient of 0.6 while other two predictions have the coefficient of 0.2. The batch size is set to 32 and train 70% samples per epoch. This work will train the model for 1000 epochs and stop training if the loss has not decreased in recent 25 epochs. Adam is chosen to optimize our loss function and the initial learning is set to be 1e-5. The decay is set to 1e-6, β_1 is set to 0.9 and β_2 is set to 0.999, just like what we set for 3D-ConvNet before.

4.2.4. Training and Result

Just like what I do when training 3D-ConvNet model, I normalize the labels with a min-max-scaler to rearrange the labels between -1 and 1 and pick up `tanh' as the activation functions for fully connect layers. 10000 videos are randomly picked as our train set and 2000 videos as validation set to train our models.

After training, the combined loss is decreased to 0.0422, and the mean square error of parameters in group 1, which are want to solve with this model, is decreased to 0.0396.

This paper also want to focus more on the parameters which are not solved as well as others. So I pick up weighted mean square error shown in Eq (6) as the loss function of each predictions respectively. For each parameter, the better the prediction is, the lower the coefficient will be, and vice versa. All the three sub loss functions share the same coefficients. After training, the loss end with 0.0391, and the mean squared error end with 0.0373.

5. Result and Discussion

This paper have tested the approach with several videos. All the performance figures and tables reported in this paper were measured on a workstation with one NVIDIA GTX 1060 GPU.

5.1. Prediction

This work tried two different types of videos to measure the performance of our method, and then test a video generated by our rendering model called DFM without randomization and a video with randomization respectively.

5.1.1. Effect of Generated Video Without Randomization

This work first test our model with a video without randomization. I randomly set the arguments needed, generate the original video, extract features, and learn the parameters with our learning model. The effect is shown in the first row of Figure 1. The original video and the reconstructed effect in three different view spaces is shown, and the first view space is the same as the original video. Compared with section 5.1.2, it is proved that because of the randomized videos in dataset, the effect will not rise if using a video without randomization, and even decrease.

5.1.2. Effect of Generated Video with Randomization

This paper then test our model with a video with randomization. The result is shown in figure 1, and also show the original video and the reconstructed effect in three different view spaces, and the first view space is also the same as the original video. As is seen, the shape is similar to the original video, while the color has a little difference. The reason is the use of decay rate to decrease the number of parameters to be learn, and the decay rate of color is very sensitive to small changes.

5.2. Limitations

Our method can only handle simple firework whose particles explosion from its center in a sphere and the particle of the firework must be approximately symmetrical. What is more, because of the lack of information such as parameter of camera and depth message, it is impossible to tell where the positive side of the firework is. The similarity between our results and real video can only be told by observing without any mathematical formula.

6. Conclusion

This work have presented a method for reconstruction a 3d firework from a video. It is showed that auxiliary labels and auxiliary predictions work well when reducing the mean square error of extracting parameters from given video, and demonstrated that different weight in loss function for different task is necessary to improve the generalization performance. By testing our method with different types of video, it is found to be effective to extract features from given video and reconstruct a 3D firework which is similar to simple firework videos.

References

[1] Reeves W T, Blau R. Approximate and probabilistic algorithms for shading and rendering stuctured particle systems [J]. Acm Siggraph Computer Graphics, 1985, 19(3):313-322.

[2] Reeves W T. Particle systems—a technique for modeling a class of fuzzy objects [M]// Seminal graphics. ACM, 1998:91-108.

[3] Loke T S, Tan D, Seah H S, et al. Rendering Fireworks Displays [J]. IEEE Computer Graphics & Applications, 1992, 12(3):33-43.

[4] Zhang S. Fireworks Simulation Based on Particle System [C]// International Conference on Information and Computing Science. IEEE, 2009:187-190.

[5] Lecun Y L, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proc IEEE [J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.

[6] Deng J, Dong W, Socher R, et al. ImageNet: A large-scale hierarchical image database [C]// Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009:248-255.

[7] He K, Zhang X, Ren S, et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification [J]. 2015:1026-1034.

[8] Boureau Y, Bach F, Lecun Y, et al. Learning mid-level features for recognition [J]. 2010, 26(2):2559-2566.

[9] Lipton Z C, Berkowitz J, Elkan C. A Critical Review of Recurrent Neural Networks for Sequence Learning [J]. Computer Science, 2015.

[10] Graves A. Long Short-Term Memory [M]// Supervised Sequence Labelling with Recurrent Neural Networks. Springer Berlin Heidelberg, 2012:1735-1780.

[11] Graves A, Mohamed A R, Hinton G. Speech recognition with deep recurrent neural networks [C]// IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2013:6645-6649.

[12] Gers F A, Schmidhuber J A, Cummins F A. Learning to Forget: Continual Prediction with LSTM [J]. Neural Computation, 2014, 12(10):2451-2471.

[13] Liu Z, Luo P, Wang X, et al. Deep Learning Face Attributes in the Wild [C]// IEEE International Conference on Computer Vision. IEEE Computer Society, 2015:3730-3738.

[14] He K, Wang Z, Fu Y, et al. Adaptively Weighted Multi-task Deep Network for Person Attribute Classification [C]// ACM on Multimedia Conference. ACM, 2017:1636-1644.

[15] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception Architecture for Computer Vision [J]. 2015:2818-2826.

[16] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks [C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2012:1097-1105.

[17] Du T, Bourdev L, Fergus R, et al. Learning Spatiotemporal Features with 3D Convolutional Networks [C]// IEEE International Conference on Computer Vision. IEEE Computer Society, 2015:4489-4497.

[18] Du T, Bourdev L, Fergus R, et al. Learning Spatiotemporal Features with 3D Convolutional Networks [C]// IEEE International Conference on Computer Vision. IEEE, 2016:4489-4497.

[19] Kingma D, Ba J. Adam: A Method for Stochastic Optimization [J]. Computer Science, 2014.