Cumulus cloud modeling from images based on VAE-GAN

Zili ZHANG1,2, Yunchi CEN1, Fan ZHANG1, Xiaohui LIANG1*

1. State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China
2. Department of Computer Science and Engineering, Shijiazhuang University, Shijiazhuang 050035, China

*Corresponding author, liang_xiaohui@buaa.edu.cn
Received: 30 August 2020    Accepted: 15 December 2020

Supported by the National Key R&D Program of China (2017YFB1002702).

Citation: Zili ZHANG, Yunchi CEN, Fan ZHANG, Xiaohui LIANG. Cumulus cloud modeling from images based on VAE-GAN. Virtual Reality & Intelligent Hardware, 2021, 3(2): 171—181
DOI: 10.1016/j.vrih.2020.12.004

Abstract

Background Cumulus clouds are important elements in creating virtual outdoor scenes. Modeling cumulus clouds that have a specific shape is difficult owing to the fluid nature of the cloud. Image-based modeling is an efficient method to solve this problem. Because of the complexity of cloud shapes, the task of modeling the cloud from a single image remains in the development phase.

Methods In this study, a deep learning-based method was developed to address the problem of modeling 3D cumulus clouds from a single image. The method employs a three-dimensional autoencoder network that combines the variational autoencoder and the generative adversarial network. First, a 3D cloud shape is mapped into a unique hidden space using the proposed autoencoder. Then, the parameters of the decoder are fixed. A shape reconstruction network is proposed for use instead of the encoder part, and it is trained with rendered images. To train the presented models, we constructed a 3D cumulus dataset that included 200 3D cumulus models. These cumulus clouds were rendered under different lighting parameters.

Results The qualitative experiments showed that the proposed autoencoder method can learn more structural details of 3D cumulus shapes than existing approaches. Furthermore, some modeling experiments on rendering images demonstrated the effectiveness of the reconstruction model.

Conclusion The proposed autoencoder network learns the latent space of 3D cumulus cloud shapes. The presented reconstruction architecture models a cloud from a single image. Experiments demonstrated the effectiveness of the two models.

Keywords 3D cloud model; 3D autoencoder network; Generative adversarial network

1 Introduction

Cumulus clouds are important elements in creating virtual environments. However, modeling clouds remains a challenging task because clouds are a kind of participating media and are fluid in nature. Many methods have been proposed to model clouds. Generally, mainstream cumulus modeling methods include procedure-based approaches, physically based approaches, and image-based methods. Procedural modeling[1-3] can generate realistic cloud scenes; however, these methods usually require designers to manually tune many parameters to obtain realistic clouds. To address this problem, physically based
simulation methods\cite{8-12} have been proposed. These methods typically simulate the formation of clouds by solving Navier-Stokes equations and thermodynamic equations. Hence, these methods can generate realistic time-varying clouds. However, both the procedural modeling and physically based methods have limitations in modeling specific cloud scenes. Unlike the first two approaches, the goal of image-based methods\cite{8-12} is to model a cloud based on a given cloud image by utilizing geometric information in the image.

Dobashi et al. proposed a method for modeling clouds from a single photograph\cite{9}. The method uses different algorithms to model different cloud types. For cumulus clouds, the cloud surface is determined from the input image by using a geometry-based approach. The 3D modeling process is based on the assumption that the inside of a cloud is denser than the boundary. However, the method cannot model the details of the cloud surface because it only considers the overall shape. To overcome this problem, Yuan et al. proposed a method to model cumulus clouds from a single image\cite{10}. The method generates a 3D shape by inverse solving of a single scattering model\cite{10}. The cloud shape is assumed to be symmetric, and some lighting parameters, such as the extinction coefficient and albedo, are set to be constant. However, many clouds do not meet these assumptions. Meanwhile, the side surface of the cumulus cloud is still unnatural owing to the stitching of only the front and back surfaces. To solve this problem, we propose a deep learning-based approach inspired by modeling 3D objects from images using a convolutional neural network.

In recent years, reconstructing 3D objects from a single image has received significant attention\cite{14-22} as a result of the development of deep learning-based methods. The ShapeNet\cite{23} dataset provides training data for this task. There are three types of storage forms for 3D models: point clouds, voxels, and meshes. In terms of point clouds, Fan et al. presented an end-to-end deep learning network that uses the encoder-decoder structure for reconstructing the 3D model based on a single image\cite{14}. The method employs a novel metric for the loss function using the Chamfer distance and Earth Mover distance. Navaneet et al. proposed a deep learning technique to learn both 3D point cloud reconstruction and pose in a self-supervised manner\cite{15}. The method employs images from the training set that belong to a similar 3D model to aid the training. Choy et al. presented a recurrent neural network architecture (3D-R2N2) model that can reconstruct 3D objects in the form of a 3D occupancy grid from a single- or multi-view image\cite{16}. The method requires no image annotations or object class labels for training or testing. However, this approach is not appropriate for reconstructing complex objects, such as motorcycles and bicycles. Kato et al. first proposed a rendering mesh using a neural renderer for reconstructing 3D objects from a single image and 2D-to-3D image style transfer\cite{17}. Liu et al. proposed a depth-preserving latent generative adversarial network (GAN) for 3D reconstruction from a monocular depth image of an object\cite{22}.

The above learning-based methods can effectively reconstruct a 3D shape from a single image; however, they are not adequate for modeling cumulus clouds because of the lack of a training dataset and the coupling relationship between the shape and the illumination. To overcome these problems, we propose a method based on the variational autoencoder (VAE) and GAN, and we construct a 3D cumulus dataset, including 3D models and corresponding images. The contributions of this study are as follows:

(1) We propose a model based on VAE-GAN for modeling cumulus clouds from images. The presented neural network learns the latent space of the 3D shapes of cumulus clouds.

(2) We adopt a two-step training strategy to decouple the structure and illumination. The proposed neural network uses voxels as input and consists of an encoder, a decoder, and a discriminator. After training the network, the output of the encoder, $E_{img}$, is the latent space of the 3D cumulus cloud shape.

(3) We designed a novel encoder, $E_{img}$, which takes an image as input based on ImageNet-18, to replace the previous encoder, $E_{id}$. To reconstruct a 3D cumulus cloud from the input image using the proposed model, the new encoder should have the same output as $E_{id}$.
2 Methods

An overview of the pipeline for reconstructing the 3D shape of the cloud from a given image is shown in Figure 1. This approach is very different from rigid body reconstruction. Moreover, the cumulus cloud is an active medium that exhibits light attenuation and scattering. The final rendering result is not only affected by the structure of the cloud, but also by the illumination. It is thus necessary to eliminate the influence of illumination for modeling cumulus clouds. For decoupling the relationship between the cumulus structure and the illumination, a two-step training strategy is proposed. We designed an autoencoder, CumAE$_{3D}$, based on VAEs [24] and GANs [25]. We first train the network CumAE$_{3D}$, which obtains the latent space representation of each 3D cumulus model, by means of self-supervised training. Next, the decoder is fixed. We designed a novel encoder that employs the cumulus cloud image as input. Therefore, the main concern is determining how to obtain the same latent space description using the new encoder to remove the influence of illumination on reconstructing the shape. Our cloud shape reconstruction network is illustrated in Figure 1.

2.1 Feature extraction of the 3D cumulus model

Inspired by the work of 3D-VAE-GAN [26], we designed a 3D cumulus autoencoder network, CumAE$_{3D}$, which has a network structure similar to that of [26]. In addition, two additional parts were introduced for training. First, we used a discriminator to distinguish the generated 3D cumulus models from the autoencoder, which was used as a generator. To achieve a better reconstruction result, we trained the generator and the discriminator in turn. Second, we designed the encoder based on the VAE. The 3D cumulus models were mapped to the hidden space, which had a normal distribution. Then, the final 3D cumulus model was obtained by sampling from the hidden space.

2.1.1 Generative model

Currently, VAE and GAN are the two most renowned generative models in deep learning. In this study, we designed an autoencoder structure based on both VAE and GAN. The autoencoder was used to model the distribution of the 3D cumulus models. A traditional autoencoder network structure is shown in Figure 2. This makes the difference between the input and output as small as possible. Hence, the decoder has the ability to model 3D cumulus models. However, it is unable to learn the continuous distribution of the hidden space of the 3D cumulus models, which results in a weak generalization ability. To solve this problem, we make the latent space of the 3D cumulus shapes obey a normal distribution. Therefore, we can resample from the hidden space to generate a new 3D cumulus models that adheres to the normal distribution. This shows that the decoder has good generalization ability.

Both VAE and GAN can generate new data with the same distribution as target data $X$ by using a latent variable, $Z$, which is a normal distribution. Hence, the encoder is used to learn latent variable $Z$ in our task. Then, a 3D cumulus model can be

![Figure 1 Diagrammatic representation of reconstruction of cloud shape from a given image.](image1)

![Figure 2 Generative model structure.](image2)
obtained by using a decoder, $g$, with $Z$ as the input; that is, $X = g(Z)$.

### 2.1.2 3D cumulus autoencoder network

We propose a 3D cumulus autoencoder network, which is based on Section 2.1.1, the description above, to extract the structural features of 3D cumulus models in latent space. The architecture of the model named CumAE$_{3D}$ consists of the encoder, decoder, and discriminator, as shown in Figure 3. In the model, both the encoder and decoder constitute the generator.

The encoder includes five convolution layers, two fully connected layers, and a random sampling (RD) layer. The output of each convolution layer is normalized by a BatchNorm3D layer, which makes the input of each layer have the same distribution. Meanwhile, the structure can accelerate the convergence rate and avoid gradient dissipation during the process of back propagation. The two fully connected layers are named FC1 and FC2, respectively. FC1 learns the mean value of the latent space variable obeying a normal distribution, and FC2 learns the standard deviation. The RD layer uses the outputs of FC1 and FC2 as input. Then, the final latent space representation of the 3D model is obtained by random sampling.

In the process of sampling, we must take sample $Z^k$ from the normal distribution $P(Z|X_k)$. As the mean and standard deviation are calculated in the FC1 and FC2 layers, back propagation of the gradient in the model is eliminated. To solve this problem, we first take a sample from a standard normal distribution. Then, the sample is multiplied by the variance and added to the mean. The process is shown in Figure 4.

Inspired by ResNet\cite{27}, the decoder consists of five substructures that have the same structure as the residual block, as shown in Figure 5. The structure enables the generated model to preserve more details. The substructure consists of a backbone network and a side branch. The backbone network first passes through a deconvolution layer. It then adjusts the output through a 3D convolution layer with the size of $3 \times 3 \times 3$. The side branch contains only one trilinear network. The final result is the sum of the two outputs of the backbone network and the branch part.

The discriminator is regarded as a binary classification network. The input is the cumulus 3D voxel model with the size of $64 \times 64 \times 64$, and the output is the probability that the distribution of the generated model and the cumulus 3D dataset will be the same.

### 2.1.3 Autoencoder loss function

According to Section 2.1.2, the 3D cumulus autoencoder CumAE$_{3D}$ consists of two parts: generator $G$ and discriminator $D$. The generator consists of an encoder and decoder. For the generator, the loss function can be composed of four parts: $L_{3D-GAN}$, $L_{KL}$, $L_{rec}$, and $L_{rul}$. The total loss function of the generator is the following:

$$L_{G} = L_{3D-GAN} + L_{KL} + L_{rec} + L_{rul}$$
\[ L_1 = L_{3D-GAN} + \alpha_1 L_{KL} + \alpha_2 L_{rec} + \alpha_3 L_{mask} \]  
where \( L_{3D-GAN} \) denotes the recognition of the reconstructed model by the discriminator, \( L_{KL} \) is the divergence loss introduced by VAE, \( L_{rec} \) represents the difference between the reconstructed model and the input model, and \( L_{mask} \) is the difference between the reconstructed mask and the original one. In addition, \( \alpha_1, \alpha_2, \) and \( \alpha_3 \) are scaling parameters. Each element in the loss function is calculated as follows:

\[ L_{3D-GAN} = \log D(x) + \log (1 - D(G(x))) \]  
\[ L_{KL} = D_{KL}(q(z|x)||p(z)) \]  
\[ L_{rec} = \| G(E(y)) - x \|_2 \]  
\[ L_{mask} = \| IP(G(x)) - M(x) \|_2 \]

where \( x \) is the 3D cumulus model, and \( M(x) \) is the mask of model \( x \).

### 2.2 Reconstruction network

In Section 2.1, we obtained the latent space expression of each 3D model, including the mean vector of 200 dimensions and the variance vector of 200 dimensions. Next, we trained a 3D reconstruction network using a rendered image as input. The structure of the reconstruction model based on the images is shown in Figure 6. For convenience, the model is named CumRN\(_{img}\). It is noted that network CumRN\(_{img}\) only uses a new encoder structure. It takes an image as input for substitution for the encoder, \( E_{3D} \). The decoder part is the same as in the previous network. As ResNet-18 has fewer parameters than in ResNet-50, and the size of the proposed dataset is smaller than that of ImageNet, we used ResNet-18 as encoder \( E_{img} \).

**Figure 6 3D reconstruction network structure taking images as input.**

#### 2.2.1 Implementation details

As the input nature images contain information about the structure and luminance, encoder \( E_{img} \) should only learn the structural features from the inputs to eliminate the influence of the luminance on the modeling cumulus cloud.

According to the description above, the model outputs a corresponding 3D voxel model for rendering cumulus cloud images with a resolution of 224 × 224 as input. We can obtain the latent space expression of the 3D voxel model using encoder \( E_{3D} \). Therefore, we must only train encoder \( E_{img} \) to output the same latent space for the input image, which is the rendering result of the 3D voxel. Hence, the encoder parameters are gradually adjusted, and the decoder parameters are fixed during training. We update the proposed model using the following objectives:

\[ L_2 = L_{latent} + \beta_1 L_{rec} + \beta_2 L_{mask} \]  
where \( \beta_1 \) and \( \beta_2 \) are scale parameters, \( L_{latent} \) denotes the consistency of the hidden space vector, and \( L_{rec} \) is the consistency of the final reconstructed 3D model, \( L_{mask} \) which measures the consistency between the forward projection of the 3D model and the input image. The three items measuring consistency—\( L_{latent}, L_{rec}, \) and \( L_{mask} \)—use the L2 loss function as follows:
where $x$ is the input image, $x_{\text{mask}}$ denotes the mask of the cloud, and $y$ is the 3D voxel model of the input. The function $P(\cdot)$ is the process of orthographic projection. We set $\beta_1 = \beta_2 = 0.5$ and fix the parameters of the decoder during training. Meanwhile, we use the Adam optimizer with an initial learning rate of encoder $E_{\text{img}}$ as 0.0025 with a batch size of eight. The presented architecture is stable and can reliably converge. The network converges at approximately 20 epochs. Figure 7 shows the convergence plot with training iterators on the $x$-axis and the errors on the $y$-axis. It is noted that, after approximately 15 epochs, the loss function error is less than 0.004 and remains at a constant level.

### 3D cumulus dataset

3D cumulus data are always difficult to obtain through existing detection methods, which makes it challenging to construct a 3D cumulus dataset. A large collection of natural cloud images collected from the Internet is available. Thus, we used Blender to manually model these 3D cumulus models according to the collected cumulus images. The dataset constructed in the previous step was stored in the form of a mesh. In addition, these 3D cumulus models were fed into the cumulus encoder network as training data. Consequently, it was necessary to employ data preprocessing to convert these cumulus models in the form of a triangular mesh into a voxel. We then rendered these cumulus models under different illumination parameters and camera positions. Finally, a cumulus dataset was constructed using the 3D models, illumination parameters, and rendered images corresponding to each other. Some examples are shown in Figure 8.

Because the dataset included fewer 3D cumulus models than needs training the proposed models, the input data were augmented to enhance the network generalization performance. Therefore, we adopted three augmentation methods. First, the process of random rotation was used for the select model. The model was rotated in the horizontal direction, and the degree of rotation was sampled from $0^\circ$ and $360^\circ$.
Second, the model was randomly scaled. For each dimension—length, width, and height—the scaling coefficient ranged from 0.8 to 1.2. Third, we translated the model. The translation range was set from −5 to 5. Negative numbers indicated that the dimension was translated in the opposite direction.

4 Results

We trained and evaluated the proposed models on a 12 GB Nvidia GeForce TITAN GPU. The experiments were divided into two parts. First, we evaluated the performance of CumAE_{3D}. Second, we conducted detailed experiments to validate the network, CumRN_{img}.

Figure 9 plots examples of the reconstruction results of the autoencoder CumAE_{3D}. The left column shows the mesh model as input; the second column is the corresponding voxel models. To verify the effectiveness of the decoder with residual structures, our method was compared with the decoder without residual structures, which uses a deconvolution structure for upsampling. Our results are shown in the last column. Compared with the third column, which shows the results of the decoder without residual structures, our method preserves more details. The results show that our method can better characterize the characteristics of the input model and reconstruct the input shape more completely.

We calculated the reconstruction error and projection error on the test samples to evaluate the proposed model. The reconstruction error is defined as follows:

$$\text{Diff}_{\text{recon}}(x^m, \hat{y}^m_{\text{final}}) = \frac{\sum \sum \sum x^m(i,j,k)! = \hat{y}^m_{\text{final}}(i,j,k)}{N^3}$$

Figure 9 Comparison of reconstruction with different decoders. From left to right: input meshes, the corresponding voxels, the results of the decoder without residual structures, and the results of the decoder with residual structures.
where \( x^m \) is the input, \( \tilde{y}^m \) is the reconstruction voxel using the proposed model, and \( N \) is the dimension of the voxel data.

The projection error is defined as follows:

\[
Diff_{proj}(x^m, \tilde{y}^m) = \sum \sum x^m(i,j) \neq \tilde{y}^m(i,j)
\]

The results are provided in Table 1. It is noted that the proposed model with residual structures achieves the best performance.

We also compared our method with state-of-the-art deep learning-based 3D shape reconstruction\(^{[26]}\). Some reconstruction results for different clouds are shown in Figure 10. The left column displays the input synthetic cloud images; the right four columns show the reconstruction results. Note that the shapes of the four clouds are different. Both the reconstructed shapes and the projection results demonstrate that our method can reconstruct the 3D shape that matches the input images. Moreover, our method provides more details than the method proposed by Wu et al.\(^{[26]}\).

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Results for reconstruction and projection errors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reconstruction Error</td>
</tr>
<tr>
<td>Model without residual structures</td>
<td>0.0762</td>
</tr>
<tr>
<td>Model with residual structures</td>
<td>0.0710</td>
</tr>
</tbody>
</table>

Figure 10  Comparison of reconstruction results of network CumRN\(_{arch}\). Our method preserves more details than the method in [26].

In contrast to traditional methods\(^{[9]}\), the reconstruction results obtained by the deep learning method could better restore information about the side and back of the model. Meanwhile, it did not produce a severe appearance of fragmentation. We then evaluated this finding by selecting several rendered images viewed from the side. The reconstruction results are shown in Figure 11. The experiments showed that the proposed method restored the shapes of the side and back with no serious symmetry effect.
Conclusion

In this study, we proposed an autoencoder network, including an encoder, a decoder, and a discriminator, to solve the problem of modeling a cumulus cloud from a single image. The model combines VAE and GAN to learn the embedding of 3D cloud shapes. After training the autoencoder, the parameters of the decoder part are fixed. Then, another encoder structure that we designed is used instead of the encoder part; it employs a single image as input. Meanwhile, the GAN part is deleted. Finally, the new encoder using the image as the input and the decoder form the reconstruction network of the cumulus cloud from a single image. To train the two models, we constructed a 3D cumulus dataset, including 200 3D cumulus models. These cumulus clouds were rendered under different lighting parameters. The qualitative experiments showed that the proposed method can model 3D cumulus cloud shapes that better match the input image and have more structural details.

Some problems with the proposed approach remain to be solved. For example, we use a 3D cloud point as input instead of a voxel to improve the resolution of the reconstruction shape. Although reconstruction methods produce clear models, the 3D shapes usually have some residual small isolated areas due to a lack of structure information. The problem could be solved by making the autoencoder to explicitly learn the structure features.

Declaration of competing interest
We declare that we have no conflict of interest.

References

   DOI:10.1145/325334.325248
   DOI:10.1145/357306.357310
   DOI:10.1145/2699276.2721396
   DOI:10.1145/800031.808594
   DOI:10.1109/PCCGA.2001.962893


21 Yang B, Rosa S, Markham A. Dense 3D object reconstruction from a single depth view. IEEE transactions on pattern analysis and machine intelligence, 2018, 41(12): 2820–2834 DOI:10.1109/TPAMI.2018.2868195

reconstruction. IEEE Transactions on Multimedia, 2020, 1
DOI:10.1109/tmm.2020.3017924

DOI:10.1145/2980179.2980238

24 Kingma D P, Welling M. Auto-encoding variational bayes. 2013


DOI:10.1109/CVPR.2016.90