GOMD 2019
N.N. and N.N.
(Guest Editors)

Volume 0 (2019), Number 0

Cumuliform Cloud Formation Control using Parameter-Predicting Convolutional Neural Network

Zili Zhang[1,2], Yue Ma[1], Yunfei Li[1], Frederick W. B. Li[3], Hubert P. H. Shum [4], Bailin Yang [5], Jing Guo [6] and Xiaohui Liang[1]

[1]State Key Laboratory of Virtual Reality Technology and Systems, Beihang University
[2] Department of Computer Science and Engineering, Shijiazhuang University
[3] Department of Computer Science, University of Durham
[4] Department of Computer and Information Sciences, Northumbria University
[5] Department of Computer and Electronic Engineering, Zhejiang Gongshang University
[6] Shanxi Institute of Technology

Abstract

Physically-based cloud simulation is an effective approach for synthesizing realistic cloud. However, generating clouds with desired shapes requires a time-consuming process for selecting the appropriate simulation parameters. This paper addresses such a problem by solving an inverse cloud forming problem. We propose a convolutional neural network, which has the ability of solving nonlinear optimization problems, to estimate the spatiotemporal simulation parameters for given cloud images. The cloud formation process is then simulated by using computational fluid dynamics with these control parameters as initial states. The proposed parameter-predicting model consists of three components, including the feature extraction network, the adversarial network and the parameter generation network. These subnetworks form two parallel branches for different functionality-feature extraction and parameter estimation. To solve the challenge of estimating high-dimensional spatiotemporal simulation parameters, we adapt an encoder and decoder network to compress these parameters into a low-dimensional latent space. We train the proposed deep learning model with pairwise data of time series parameters and the corresponding synthetic images, which are rendered by the density fields of the synthesized clouds under different illuminations. In the practice, our method can simulate physically plausible cloud evolution processes and generate clouds with desired shapes for the real-world and synthetic images.

CCS Concepts

• ***Computing methodologies*** → ***Modeling methodologies; Shape modeling;***

1. Introduction

In computer graphics, clouds are important weather phenomena for outdoor scenes synthesis, heavily contributing to scene realism. Physically-based cloud simulation has become an important research field as it can generate realistic clouds. Over the years, different methods [HBSL03, DKNY08, DG17] have been proposed to achieve visually plausible cloud scenes for games and movie production. However, it is difficult to generate clouds matching user-specified shapes by using these methods. This limit their applications for movies, theater scenes of battlefields and virtual experiment relating to recreating cumulus cloud observations, in which it is desirable to generate clouds with specific shapes through physically natural cloud evolution processes.

Traditionally, this problem can be tackled by the forward cloud formation methods [MDN02, HBSL03, FBDY15]. A user could find the appropriate parameters that produce a satisfactory result through a repetitive trial-and-error process according to the corresponding output image. To avoid the time-consuming parameter tuning process, Dobashi et.al. [DKNY08] propose a feedback-based method to automatically tune some physical parameters, including the coefficient of latent heat and water vapor supply, so that the simulated cloud can closely match the user-specified shape. However, the constraint of shape only includes a specified top contour. Moreover, as both the latent heat coefficient and water vapor supply are adjusted during run-time according to the height difference between the simulated cloud and target contour, the setting lacks physical rationality, resulting in unrealistic situations such as constant heat on ground. Also, this approach does not produce a physically natural cloud formation process due to an extra feedback control. While some fluid accelerating simulation methods [TSSP17, XWY19] can accelerate the parameter design interaction, the time-consuming trial-and-error process is still required.

Solving an inverse cloud formation problem could remove the repetitive trial-and-error process. As the cloud formation process is governed by a set of nonlinear dynamical equations which use some physical parameters as initial state, Arcia-Dorado et

al. [GDAB*17] propose a Markov Chain Monte Carlo optimization based method to explore the search space and find proper initial conditions that can exhibit the desired cloud state by using a procedurally-generated model. The method can obtain the desired cloud coverage, such as the percentage of sky covered by clouds throughout a day, without any artificial force during simulation. However, it is not easily extendable to high-dimensional constraints. Moreover, it is impossible to describe all cloud shapes using a finite state space.

In our method, in contrast to [DKNY08], which relied on feedback control, we focus on solving the inverse cloud formation problem to automatically generate appropriate parameters that control the cumuliform cloud formation to match the cloud shape from a given image without adding any artificial force during simulation. Inspired by recent advances in deep learning for solving both nonlinear optimization problems [LSD*18] and fluid simulation [MTP*18, XWY19, KAT*19], we propose a deep neural network to estimate parameters controlling the formation process of clouds. To efficiently solve the inverse cloud formation problem, we first propose a set of decoupled parameters that play an important role in the shape of cloud by analyzing the formation process. Based on that, we present a novel parameter-predicting convolutional neural network (PPCNN) consisting of two parallel streams, namely feature extraction and parameter generation. Instead of directly predicting control parameters according to the raw images, we first use the feature extraction branch based on a generative adversarial network to encode the image for eliminating noise and lighting influence. The extracted features are then used as the inputs to the parameter generation branch processing by a convolutional neural network.

As it is necessary to finely discretize these control parameters in both space and time for numerical simulation, the parameter space has a large degree of freedom. Optimizing such a large number of parameters makes it difficult for training the model. As a solution, we propose to compress the parameters into a latent space by using an encoder and decoder network. Consequently, the proposed PPCNN learns from the control parameter latent space representation instead of the raw higher dimensional parameters. To train the model, we use [DKNY08] to generate a simulated dataset under various control parameter conditions and render the simulated data under different illumination to get the synthetic images.

We evaluate our method on both synthetic images and real photos. We give both qualitative and quantitative analysis on the accuracy of the simulated results using structural similarity index (SSIM), mean, standard deviation of peak signal-to-noise ratio (PSNR), intersection over union (IOU) and perceptual loss. We analyze the parameter generation power by calculating the mean absolute percentage error (MAPE) between the generated parameters and the simulation-based results. We have also compared our method with previous approaches [DKNY08, YLH*14]. Results show that our approach achieves effective parameter estimation, which generates cloud matching user-specified shape and preserves physically natural dynamics.

Our main contributions include:

- A new method to reconstruct clouds from a given image by solving an inverse formation problem combining atmospheric fluid dynamics and data-driven methods.
- A set of decoupled influence components, which are useful for learning control parameter and analyzing the relationship between the cloud shape and the environment.
- A model to predict the learned latent space of control parameters for cloud images consisting of a discriminator branch and an parameter generation branch based on an autoencoder.

This paper is organized as follows. We describe existing approaches of fluid control, inverse cloud simulation and learning-based fluid simulation in Section 2. In Section 3, we formulate in detail the problem of solving inverse cloud formation. The structure of the parameter predicting model is described in Section 4. Both quantitative and qualitative experiments results are highlighted in Section 5, followed by discussion and conclusion in Section 6.

2. Related Works

We now review the most closely related work to our method, including fluid control, inverse cloud simulation and learning-based fluid simulation.

Fluid Control In terms of target guiding fluid simulation, many control methods have been proposed for guiding fluid animations with desired shape.

Treuille et al. [TMPS03] proposed a method for controlling smoke simulation to match user-specified keyframes by optimizing appropriate wind forces based on the quasi-Newton optimization technique. Mcnamara et al. [MTPS04] controlled physics-based smoke and water simulations by using the adjoint method. More recently, Pan and Manocha [PM16] used a space-time optimization solver to control smoke animation. These methods assume that the entire simulation process is a function whose gradient information can be computed, which can generate natural simulation. However, these methods are computationally expensive.

To avoid expensive optimization over the entire fluid sequence, other methods have been proposed. Hong and Kim [HK04] controlled smoke animation by adding a potential dimension to the simulation space. Shi and Yu [SY05] generated the control forces according to the error between the current fluid shape and the target keyframe shape. Raveendran et al. [RTWT12] controlled fluid motion with the guidance of a dense sequence of control meshes generated by a volume preserving morph. These methods can control smoke and water animation so that they form user specified shapes. However, they apply virtual artificial control forces that influence the natural physical governing law over the simulation domain.

All these methods do not consider the phase transition process, which is important for cloud formation. To obtain the desired cloud, Dobashi et.al. [DKNY08] proposed a feedback control based method that tuned the latent heat coefficient and water vapor supply in real-time. As adding artificial controls, including a feedback control and an external force due to the geometric potential field, over the simulation domain, the approach failed to produce a physically natural cloud formation process. In contrast, our method achieves shape guiding cloud formation with physically natural dynamics.

Inverse Cloud Simulation The most closely-related simulation

methods to our work in computer graphics are various forms of inverse cloud simulation. These methods usually requires to find a set of parameters or the initial conditions to control the simulation behaving as desired. Dobashi et al. [DIO*12] used genetic algorithms to search for optimal illumination parameters, such as albedo and extinction coefficients, for rendering clouds such that the appearance of the synthetic cloud is similar to the cloud in a given photograph. Due to the high dimensionality of control parameters for cloud formation, it is difficult to extend the method to solve the inverse cloud formation problem. To estimate the cloud thickness from an image, Yuan et al. [YLH*14] determined the shape of a cumulus cloud by inversely solving a single scattering model. Closer to our area of application, Garcia-Dorado et al. [GDAB*17] proposed a Markov Chain Monte Carlo optimization based method to find proper initial conditions, including the gradients in landscape and the initial wind and humidity values that can exhibit the desired cloud coverage and the behaviors of the weather. They searched the parameters via state changes. However, it was difficult to extend it to achieve our aim due to the high-dimensional constraints and the infinite cloud shapes.

In contrast to the above methods, our approach based on deep learning is powerful to solve nonlinear optimization problems. We propose a convolutional neural network to estimate control parameters used as the initial conditions to drive the cloud formation and evolution. This allows us to generate clouds with desired shapes, while keeping the nature of physically based simulation.

Learning-based Fluid Simulation Ladický et al. [JSP*15] employed random regression forests to estimate the acceleration of every fluid particle for each frame. However, the method required handcrafted features that lack the generality and abstraction power of CNNs. Recently, deep learning with neural networks has been successfully applied in solving problems in fluid simulation. Jonathan et al. [TSSP17] used convolutional neural network to solve the incompressible Euler equations, which included a large sparse linear system with many parameters. These methods were only designed to accelerate the enforcement of fluid simulation. Deep learning have also been applied to synthesize smoke details [CT17, XFCT18], simulate liquid drops [MLC*18] and learn the evolution of fluid flow [WBT18]. Our work shares some similarities in term of the goals with the method [MTP*18], which proposed a model using deep reinforcement learning to control coupled fluid and rigid systems in 2D domains. The method only focused on the fluid interactions with rigid bodies and did not consider the shape of fluid. In contrast to our aim, Kim et al. [KAT*19] proposed a generative model using convolution neural network, which could synthesize a fluid velocity field for a given time from a set of parameters. The input parameters usually contained the position and width of the fluid source and the current time. Our approach is motivated by these developments and we present a novel deep neural network to estimate the proper control parameters for a given cloud image.

3. Problem Definition

We focus on controlling cumuliform cloud formation by dealing with an inverse problem, which is solved using a deep learning method. Fig. 1 shows the basic idea of our system. First, the designers only need to give a cloud image as the input for the system. Second, the proposed parameter-predicting Convolutional Neural Network (PPCNN) predicts parameters $\widehat{\mathbf{C}}$ for the input, which is then processed by the autoencoder CNN to form a latent space encoding the physical control parameters. Third, the autoencoder CNN decodes the input $\widehat{\mathbf{C}}$ to $\mathbf{C}$. Finally, we solve the Naiver-Stokes equations and thermodynamics equations to simulate cloud formation using these control parameters $\mathbf{C}$ as input, and the simulated result matches the given cloud image.

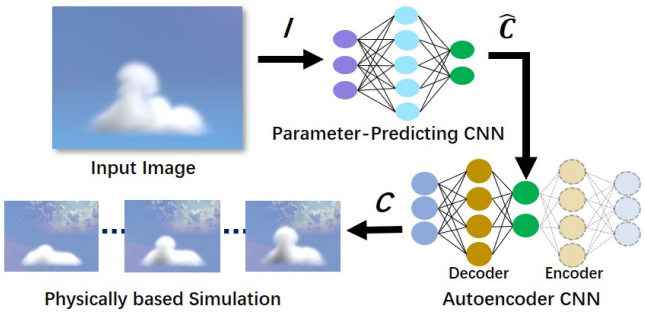

Figure 1: *Overview of our method.* $\mathbf{C}$ *denotes the decoupled control parameters, which is yielded by the autoencoder CNN through an encoded space of* $\widehat{\mathbf{C}}$*. The key of our pipeline is the parameter-predicting CNN which can estimate the optimal control parameters* $\widehat{\mathbf{C}}$ *for a given cloud image. Then, the autoencoder CNN decodes the latent space encoding of* $\widehat{\mathbf{C}}$ *into* $\mathbf{C}$*.*

In this section, we give an overview of both the cloud formation process and the decoupling of physical influence elements to obtain a set of proper control parameters for efficient learning. We then formulate the parameter estimation problem for designing the deep learning model described in the next section.

3.1. Cloud Formation and Parameter Decoupling

In this paper, we focus on cumulus clouds, which are usually resulted from the upwards motion of air parcels. Cumulus clouds usually have flat bases and round tower tops in appearance. The cloud formation has two consecutive stages: birth and lift.

Cloud Formation The birth of a cloud on the ground is caused by heating the air parcels [HJ14, DG17]. The particles of air parcels get steadily warmer on the ground by absorption of solar radiation. When each particle reaches its convective temperature T_c at some time, these particles begin to rise above the ground. Meanwhile, the birth of cloud is also related to the placement of each air parcels on the ground, which is called the area of emitting particles for clarity. The larger the area, the more coverage the cloud expands.

At the lifting stage, each air parcel born from the ground rises into the atmosphere due to buoyant and mechanical forces. Assuming the atmosphere is incompressible and inviscid, the motion of the air parcels can then be governed by the following Navier-Stokes

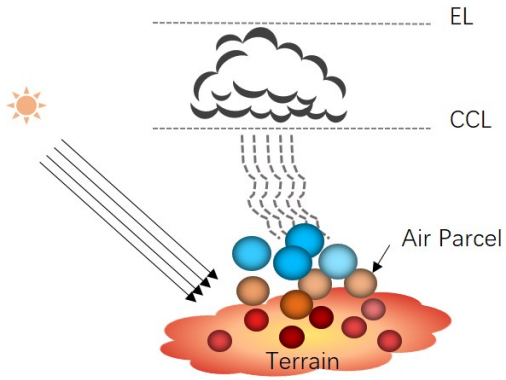

Figure 2: *Overview of cloud formation process. Colored circles denote air parcels with different temperatures from the terrain. CCL and EL are convective condensation level and equilibrium level, respectively.*

equations:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p - \rho g\mathbf{z} + \mathbf{B} + \mathbf{f} \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2}$$

where $\mathbf{u}$ denotes the velocity vector, ρ is the density, p is the pressure, g is the coefficient for the gravity, $\mathbf{z}$ is the unit vector $(0,0,1)$ pointing in the upward vertical direction, $\mathbf{f}$ accounts for the external forces affecting fluid flow, and $\mathbf{B}$ is the buoyance expressed as follows:

$$\mathbf{B} = k_b \frac{T_p - T_a}{T_a}\mathbf{z} \tag{3}$$

where T_p and T_a are the the air parcel temperature and the ambient temperature, respectively. k_b is the coefficient for the buoyance.

The temperature of the rising air parcels decreases due to adiabatic cooling during the rising process. When air parcels reach the convective condensation level (CCL), cloud originates due to phase transition, i.e., saturated air parcels condense into droplets to form a cloud. The cloud formation process described above is shown in Fig. 2. It should be noted that the evolution of cloud also contains other physical processes, such as evaporation and autoconversion. As we mainly focus on cumulus cloud formation, it is reasonable simplification to only consider the transition between water vapor and droplet.

Parameter Decoupling There are many physical elements to jointly influence the cloud formation process and the final cloud shapes. To enable efficient parameter learning and analyzing the relationship between the paramters and the cloud shape, we propose a stategy to decouple them.

Since each air parcel gets steady internal energy at the stage of birth and its temperature gradually decreases due to its expansion during the lift, the temperature of the ground directly influences the temperature of each particle. Depending on the temperature at ground level, particles inside clouds reach their equilibrium at a higher or lower altitude in the atmosphere. Meanwhile, combing the different temperature profiles of the ground and the distinct air parcel emission patterns result in clouds of various shapes [GDAB*17]. If the ground temperature decreases during an extended period and more particles are released at higher temperatures, it will generate a cloud with an upside-down turret shape and a larger cloud base, as shown in the left image of Fig. 3). If the amount of air parcels emission oscillates during the simulation, the cloud will then mainly spread horizontally, as shown in the right image of Fig. 3. Hence, the particle releasing pattern over a period of time is also an important factor.

Figure 3: *Examples of the influence of different ground temperature profiles on the cloud formation.*

During the lift process, the ambient temperature T_a also plays an important role on the buoyance. Meanwhile, the phase transition between water vapor (w_v) and liquid water (w_l) begins to occur. Several parameters, including the phase transition ratio α, phase transition parameters and dry adiabatic lapse rate, influence the amount of the phase transition. Meanwhile there are several parameters used to simplify more complex processes, e.g., k_b and g, which affect the buoyance, to be considered.

To achieve the aim of decoupling parameters, we first choose the ground temperature and the air parcel emission area, which are denoted as S_T and S_A, respectively, as control elements based on the foregoing analysis. We then decouple the two control elements from the rest of parameters, including other environment elements and the manual tuning parameters. To focus on analyzing the influence of S_T on the buoyance, the parameter T_p is set to be linear change from bottom to top similar to the method [DKNY08]. Meanwhile, the coefficient for buoyance force k_b is also set to be constant. The factor of CCL only determines the cloud-base height, hence, the parameter is set to be constant without a negative effect on controlling cloud shape. In short, we opt for the spatiotemporal distribution of the ground temperature, S_T and the parcel emission area, S_A to control cloud formation. For clarity, we name the two high-dimensional influence components as control parameters (CPs), i.e., $CPs = [S_T, S_A]$.

3.2. Parameter Learning Formulation

According to the described cloud formation, we now formulate the solution to the inverse cloud formation problem. We aim to form the desired cloud shape from a given cloud image by controlling the cloud formation process via a set of control parameters based on fluid dynamics. The input to our system is a cloud photograph, and the system predicts a set of proper initial generation conditions represented as control parameters that minimize the following ob-

jective function $\mathcal{O}$:

$$\arg\min_{\mathbf{C}} \mathcal{O}(S(R(NS(\mathbf{C},\alpha))),S(I)) \tag{4}$$

where $\mathbf{C}=[C_1,C_2,\cdots,C_t]$ is a series of control parameters used for controlling cloud formation, $C_i \in \mathbb{R}^n$ is the control vector at time i, and t is the total simulation time. α denotes a vector consisting of other initial parameter values, such as the ambient temperature T_a and the coefficient for buoyance k_b, which could be given according to the sounding data or the statistical law. I denotes a specified cloud photograph with height H_{img} and width W_{img}. NS is the evolution function solving the fluid dynamics equations with the parameters $\mathbf{C}$ and α as inputs to generate a cloud density field. Then the operator R renders the generated cloud to obtain a cloud image. The function S extracts the structure information for the input image. The objective function is to minimize the cloud structure difference between I and the reconstructed cloud, which is the output of R.

Due to the nonlinear nature of the cloud formation process and the high dimensionality of control parameters, $\mathbf{C}$, it is very hard to estimate a set of proper parameters for a user-specified cloud image. Existing works [XWY19, KAT*19] show that deep convolutional neural networks can tailor non-linear functions to input data. Motivated by this, we propose a novel model based on deep convolutional neural network, $G(I;\theta)$, to fit the control parameters for a given cloud image, where θ is the weight vector of the model and $G(I;\theta):\mathbb{R}^{H_{img}\times W_{img}} \mapsto \mathbb{R}^{n\times t}$ is the output of our network. As it is an ill-posed inverse problem, we finally formulate the inverse problem as an optimization problem:

$$\theta^* = \arg\min_{\theta} E_{I\sim \boldsymbol{I}} d(G(I;\theta),\mathbf{C}) \tag{5}$$

where $\boldsymbol{I}$ denotes the set of the cumuliform cloud images, $d(\cdot,\cdot)$ is a similarity metric, $\mathbf{C}$ is the simulation-based data of the generation conditions for the input image I. It is difficult to obtain the real-world generation conditions for a given photo by using physical equipment, such as radiosonde balloon and satellite. In lieu of real-world fluid data, we generate a synthetic dataset (Section 5.1) for training by using a Naiver-Stokes solver.

4. Parameter-Predicting Convolutional Neural Network

In this section, we present the details of the proposed parameter-predicting convolutional neural network (PPCNN) as shown in Fig. 4, which maps the input image I to the latent space of controlling parameters. As mentioned in Section 1, there are two main challenges: the high-dimensionality of spatiotemporal controlling parameters and the difference between synthesized and real cloud images due to the illumination and noise. We use an autoencoder to reduce the dimensionality of controlling parameters (Section 4.1). The second challenge is addressed by using generative adversarial network to formulate the feature extractor, F, learning joint features of the two image domains as the input for the parameter generator, P (Section 4.2). During training, as shown in Fig. 4, the synthetic images and the natural images are passed through the branch F-P and F-G-D, respectively. At test phase, the parameter prediciton is only achieved by the F-P pair (see Fig. 1).

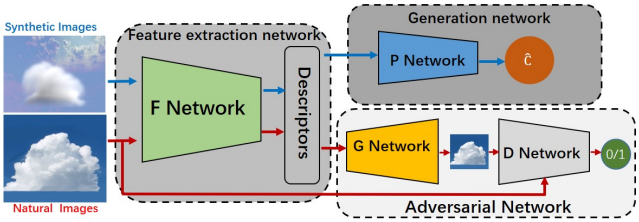

Figure 4: *Framework overview of the parameter-predicting network. In the training phase, our model consists of two parallel branch: feature extraction branch (F-G-D networks) and parameter estimation branch (F-P networks). The descriptors denote the features of the input images extracted by the network F. After training the model, we only use the second branch to estimate the optimal control parameters for a given image.*

4.1. Dimensionality Reduction

In this paper, the simulation of cloud formation is based on the Eulerian method [Bri15]. Hence, we use a two-dimensional uniform grid, Ω, with the resolution of $W_s \times H_s$ as the parcel emission area. Meanwhile, we set the number of frames to be M. If $W_s=64$, $H_s=64$ and $M=64$, the controlling parameters $\mathbf{C} \in R^{W_s\times H_s\times M}$, would have hundreds of thousands of spatial degrees of freedom. Due to the high-dimensionality of $\mathbf{C}$, directly predicting this formulation will result in too many parameters θ to be optimized, which in turn requires a large number of data to fit θ. To address this issue, we choose to extract a low-dimensional feature vector from the high-dimensional controlling parameters using a convolutional autoencoder.

As it is difficult to label the optimal low-dimensional features of the control parameters, the unsupervised learning is appropriate in our case. The convolutional autoencoders [MMCS11], which are unsupervised feature learning algorithms have been used in both image and video processing [PHC15, SZL*18, HNK*19]. Moreover the autoencoder can better preserve the nonlinear relationships among parameters than traditional linear methods. We adapt the model to achieve the aim of reducing the dimensionality of control parameters. Meanwhile, the extracted features cover more comprehensive information, which results in better parameter prediction performance.

The proposed autoencoder uses a mirrored structure. We use a series of convolutional layers activated by leaky rectified linear units (LeakyReLU) for encoder and decoder, with a bottleneck layer of dimensionality m_l. This layer yields the latent space encoding that the proposed PPCNN predicts. The encoder $\mathcal{H}_c(\mathbf{C};\Im_c)$ compresses the high-dimensional controlling parameters $\mathbf{C}$ to a low-dimensional latent space $\widehat{\mathbf{C}}$. The decoder $\mathcal{H}_d(\widehat{\mathbf{C}};\Im_d)$ maps from the latent space back to the original input. Given a dataset of $\mathbf{C}$, the parameters of the encoder and decoder will be found by minimizing the reconstruction error as follows:

$$\arg\min_{\Im_c,\Im_d} \|\mathcal{H}_d(\mathcal{H}_c(\mathbf{C};\Im_c);\Im_d)-\mathbf{C}\|_2 \tag{6}$$

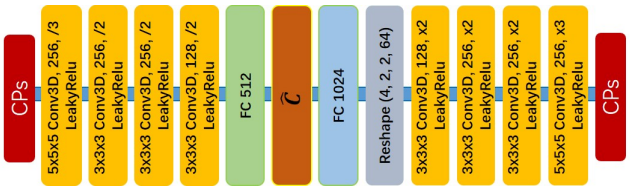

Figure 5: *The architecture of the autoencoder. The output of the encoder network is denoted as* $\widehat{\mathbf{C}}$.

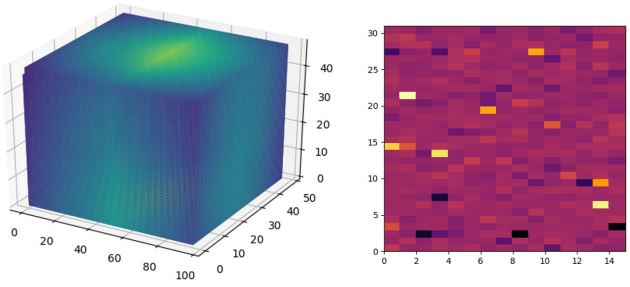

Figure 6: *The visualization for a example of* $\mathbf{C}$ *and* $\widehat{\mathbf{C}}$. *The left is the input for the autoencoder, and the right is the two-dimensional visualization for the spatial latent space vector generated by the encoder.*

The implementation of the autoencoder model is shown in Fig. 5. In our experiments, we set the m_l to be 512.

After training the autoencoder, the controlling parameters $\mathbf{C} = [C_1, C_2, \cdots C_M]$ will be encoded as $\widehat{\mathbf{C}} \in R^{m_l}$. Fig. 6 gives a visualization for an example of $\mathbf{C}$ and $\widehat{\mathbf{C}}$. To visualize the latent space vector generated by the encoder better, we change it into a two-dimensional form. As a consequence, the network PPCNN will predict the low-dimensional parameters $\widehat{\mathbf{C}}$ instead of the raw control parameters $\mathbf{C}$. At the stage of cloud formation simulation, the output for the PPCNN is decoded to yield the raw spatiotemporal control parameters, namely $\mathbf{C} = \mathcal{H}_d(\widehat{\mathbf{C}}; \Im_d)$.

4.2. Network Architecture

We train the proposed parameter-predicting network using our simulated dataset. The input are synthesized cloud images generated by rendering three-dimensional cloud density fields, which are obtained by performing the numerical simulation at the training stage. However, inputs to the testing stage may be real cloud images, comprising different characteristics from synthesized cloud images. Specifically, when light rays are injected into a cloud absorebed by cloud particles, there are still long chains of light-matter interactions forming an integral part to determine the cloud appearance. Hence, photos of real clouds usually display more perfect silverlining effect than synthesized cloud images. As multiple scattering of light is in the order of thousands of photon-matter interactions, it is impossible to simulate all such interactions for each cloud particles. Hence, the traditional model trained by the simulated dataset may have sub-optimal performance for natural cloud images.

4.2.1. The Double-Branched Deep Learning Model

To solve the domain shift problem, we propose a double-branched deep learning model. As shown in Fig. 4, the proposed parameter-predicting model contains four components: F network, G network, D network and P network. The four networks form two parallel streams, namely the adversarial branch and the generation branch. The former branch consists of F-G-D networks, trained by natrual images $\widetilde{I}$, and the latter branch consists of F-P networks, trained by synthetic images I.

As shown in Fig. 4, we learn the non-linear relationship between the control parameters and the features extracted by the network F in order to generate effective control parameters for both synthetic and natural images. To learn a shared encoding representation for both the source data with labels and the target data without labels, Ghifary et. al. proposed a model called DRCN [GKZ*16]. The DRCN did not only preserve the ability of modeling the label distribution in source domain but also encoded useful information from the target domain. In our paper, synthetic images, natural images, control parameters correspond to the source domain, the target domain and the label, respectively. During our experiments, we found that the traditional reconstruction losses computing pixel by pixel (eg. MSE used in DRCN) are not useful for our object as the input cloud images usually have a similar appearance and background. Inspired by the success of Generative Adversarial Networks (GANs) [GPAM*14], we choose the adversarial loss to substitute MSE, namely using GAN as the reconstruction network, which transfers rich gradient information to the learned embedding.

The input of generator network G is $x_g = [F(\widetilde{I}), z]$ which is a concatenated version of the encoder feature vector $F(\widetilde{I})$ and a random noise vector $z \in \mathbb{R}^d$ sampled form $\mathcal{N}(0,1)$. The network G generates a cloud image as output, the dimensionality of which is same as the input image. The discriminator D takes the real image $\widetilde{I}$ or the generated image $G(x_g)$ as input and is modeled as a binary classifier, which detects whether the input is a synthesized cloud image or a real one. We use 0 and 1 as class labels for a synthesized image and a real one, respectively. F and G are updated based on the real images. Hence, the gradients of network G and D are generated using the loss functions L_g and L_d respectively:

$$L_G = \min_G \mathbf{E}_{x \sim \tilde{\mathbf{I}}}(log(1 - D(G(F(x), z)))) \tag{7}$$

$$L_D = \min_D \mathbf{E}_{x \sim \tilde{\mathbf{I}}}(log(D(x)) + log(1 - D(G(F(x), z)))) \tag{8}$$

where $\tilde{\mathbf{I}}$ denotes the set of real images.

The network P takes the embedding $F(I)$ as input and $\widehat{C}$ as output. The inverse problem solved during training aims at minimizing the error between the predicted parameters and the simulation-based parameters:

$$L_P = \min_P \mathbf{E}_{x \sim \mathbf{I}}(L_2(R(F(x)), \widehat{C}) \tag{9}$$

where $L_2(\cdot)$ is L_2 loss function and $\mathbf{I}$ denotes the set of the synthetic images.

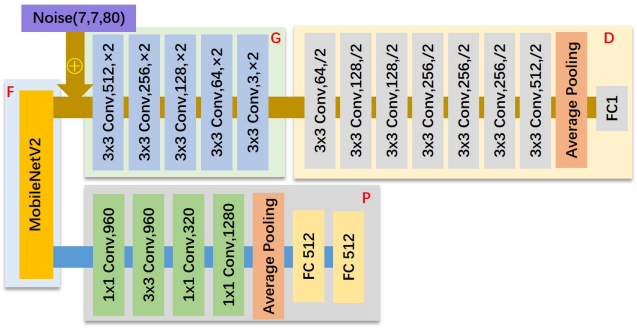

Figure 7: *The architecture of the parameter-predicting CNN. The output of the MobileNetV2 contains 160 feature maps with the size of* 7×7*. Each convolutional layer is specified by its kernel width, kernel height, no. of filters and stride, while each fully connected layer is specified by its no. of neurons.*

The feature extraction network, F, has to effectively minimize the reconstructor loss and parameter generation constraints. Thus, the loss for the network F is summarized as:

$$L_F = \min_F(\mathbf{E}_{x\sim\mathbf{I}}(L_2(R(F(x)),\widehat{C}_i)+\lambda\mathbf{E}_{x\sim\tilde{\mathbf{I}}}(log(1-D(G(F(x),z))))) \tag{10}$$

where λ controls the strength of the second terms.

4.2.2. Implementation

The implementation of our parameter-predicting model is shown in Fig. 7. We have tried to apply four different models pre-trained using ImageNet as our feature extraction network: VGG19, ResNet18, ResNet50 and MobileNetV2. During our experiments, we found that MobileNetV2 is as well as ResNet50 and is better than others. Considering that MobileNetV2 has less parameters (only 1/8 of ResNet50), we finally decided to choose MobileNetV2 [SH] as the feature extraction network, F, to extract deep feature maps from the input cloud images.

The input image is a 3-channel RGB image with the resolution of 224×224. The F network extracts the features of dimensionality m_f, denoted as Z. In the experiments, we set m_f to be 7840. Both the generator and the discriminator networks, G and D, are modeled with convolutional neural networks (CNNs). The generator G contains five convolutional leayers. The last layer is followed a tanh activation and the others are followed by a LeakyReLU activation function. The dimensionality of input for G is the same as the features extracted by F, i.e., 7840. The subnetwork D is composed of seven convolutional layers, one pooling layer and a full connected layer. Each convolutional layer is followed by a LeakyReLU activation function. The input of D is a generated image with the dimensionality of 224^2. We use four convolutional layers, activated by a ReLU activation function except for the third one, one pooling layer and two full connected layers for the network P. The last fully connection layer predicts the latent space of the controlling parameters $\mathbf{C}$.

At the training stage, both branches are trained in turns and optimally reach an equilibrium state. The outline of our pipeline is summarized in Algorithm 1.

Algorithm 1 Iterative training procedure.

training iterations=N
for t in $i : N$ **do**
  Sample k synthesized images denoted as $S : \{s_i, c_i\}_{i=1}^k$
  Let $f_i = F(s_i)$ be the embeddings for synthesized images
  Sample k real images denoted as $T : \{t_i\}_{i=1}^k$
  Let $h_i = F(t_i)$ be the embeddings for real images
  Sample k random noise samples $\{z_i\}_{i=1}^k \sim N(0,1)$
  Let h_i concatenate with z_i and denote it as h_{z_i}
  Update D using the following objective:
    $L_D = \max_F \frac{1}{k}\sum_{i=1}^k log(D(t_i)) + log(1-D(G(h_{z_i})))$
  Update G using the following objective:
    $L_G = \min_G \frac{1}{k}\sum_{i=1}^k log(1-D(G(h_{z_i})))$
  Update F using the following objective:
    $L_F = \min_F \lambda\frac{1}{k}\sum_{i=1}^k log(1-D(G(h_{z_i}))) + \frac{1}{k}\sum_{i=1}^k ||P(f_i) - c_i||^2$
  Update P using the following objective:
    $L_P = \min_P \frac{1}{k}\sum_{i=1}^k ||P(f_i) - c_i||^2$
end for

5. Experiments

This section reports the experimental validation of our approach. We refer readers to our supplemental video for the cloud evolution results. We train and evaluate all our networks on a 12GB NVIDIA® Titan X GPU with Intel® I7-8700 CPU and 16.0G RAM unless ortherwise noted.

5.1. Data Generation and Training

Since it is unrealistic to obtain a large amount of corresponding meteorological data about the environment states against the cloud shapes, we construct the training dataset through numerical simulation for the proposed model PPCNN. The input of our proposed model is a pair $[I, \widetilde{\mathbf{C}}]$, where I is a synthesized cloud image rendered from a cloud density field, which is obtained by simulating the cloud formation process based on the parameters $\widetilde{\mathbf{C}}$ containing multi-frame information of the control parameters (CPs). Specifically, $\widetilde{\mathbf{C}}$ is defined as $\widetilde{\mathbf{C}} = \{[S_T^1, S_A^1], [S_T^2, S_A^2], \cdots, [S_T^M, S_A^M]\}$, where S_T^i and S_A^i denote the distribution of the ground temperature and the emission area at frame i, respectively.

As we use a two-dimensional uniform grid, Ω, with the resolution of $W_s \times H_s$ as the parcel emission area, we have $S_A^i \in R^{W_s \times H_s}$ and $S_T^i \in R^{W_s \times H_s}$. If an air parcel was emitted from a cell positioned at (x,y) in frame i, the value of $S_A^i(x,y)$ is set to 1, or otherwise it is set to 0.

In the area of environmental engineering, the Gaussian distribution has been used to analyze the land surface temperature in some researches [Str03, EvdKVB13, ABP14]. Hence, we assume that the ground temperature obeys a two-dimensional Gaussian distribution. We denote the location of the cell with the highest temperature as (μ_x, μ_y). Then, the air-parcel temperature at the location

(x,y), $S_T^i(x,y)$, is calculated as follows:

$$S_T^i(x,y) = \frac{\widetilde{T}_p^i}{2 * \sigma_x \sigma_y \sqrt{1-\Lambda^2}} e^{\frac{(x-\mu_x)^2}{2\sigma_x^2} - 2*\Lambda \frac{(x-\mu_x)(x-\mu_y)}{\sigma_x \sigma_y} + \frac{(y-\mu_y)^2}{2\sigma_y^2}} \quad (11)$$

where $\widetilde{T}_p^i$ is the max temperature of air parcel emitted at frame i. We could control the variances, σ_x and σ_y, the correlation coefficient, Λ, and the mean, μ_x and μ_y, to achieve different distributions of air-parcel temperature. As the air parcels get warmer by absorbing the solar radiation on the ground, we also assume that the distribution of the air parcel temperature is independent each other.

When the temperature of the air parcel is higher than the ambient temperature, the air parcel begins to rise, namely the emission of an air parcel from the ground. Meanwhile, the ground temperature is constant. Then, we calculate the parameter S_A^i according to the S_T^i as follows:

$$S_A^i(x,y) = \begin{cases} 1 & S_T^i(x,y) > T_a \\ 0 & S_T^i(x,y) <= T_a \end{cases} \quad (12)$$

Hence, we use the distribution of the ground temperature to simulate the parameter of the air parcel emission.

We apply [DKNY08] to simulate cumulus cloud formation on a grid of size 64^3. To avoid obtaining low cloud density field, the duration of the CPs ranges from 40 to 100 frames. Meanwhile, we set the ambient temperature to be linearly changing from $22.4^\circ C$ (bottom) to $17.6^\circ C$ (top), which is similar to the settings in [DKNY08]. We also follow [DKNY08] to use the dry adiabat instead of the moist adiabat over the simulation domain. Hence, the temperature of air parcel is the only factor with an impact on the EL. The dry adiabat lapse rate is set to $9.8^\circ C/km$. This assumption is helpful for solving the inverse problem. However, it does not influence the controlling results.

To study the impact of emission area, we fix the shape of area as a rectangle and choose three different sizes, including 16×16, 40×40 and 54×54, which represent the small, medium and large scales, respectively. We change the variances, σ_x and σ_y, of the function $S_T^i(x,y)$ to study the distribution for the temperature of air parcels at different times. According to our experiments, when the variances, σ_x and σ_y are larger than 100, the distribution of temperature is near to the uniform distribution. To generate different distributions, we set both variances to be within the interval $[5,100]$. A large value indicates that all air parcels have similar temperatures. A small value generate the condition under which some air parcels have higher internal energy than other parcels. In the experiments, we found that if the distance between the means, (μ_x,μ_y), of the distributions of two ground temperatures is too small, the shapes of the generated clouds will not be distinctive. Hence, the sampling intervals of the two means, μ_x and μ_y, are set to 4 grid cells in our experiment. Meanwhile, we design three temperature profiles including increasing profile, decreasing profile and mixed profile. To ensure a sufficient amount of variance with respect to the controlling parameters, we set the value for the four parameters, i.e., the size of area, the variances, the means and the temperature profile, by randomly sampling from their range. Finally, our data set contains 15,000 scenes of different control parameters, and has a total size of 42.8GB.

For training the PPCNN, we initialize the parameters of the network using the method of He et al. [HZRS15]. Meanwhile, we employ the ADAM optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $decay = 5 \times 1e-5$ and an initial learning rate of $2 \times 1e-4$. The minibatches size is set to 16. The synthetic dataset is split into 80% and 20% between non-overlapping training, T and validation subsets, D. The first is used for training the parameters of the network, while the second is to monitor convergence and test the model.

5.2. Qualitative and Quantitative Analysis

In the following, we demonstrate that our proposed method can generate cumulus clouds with desired shapes using the control parameters learned via the PPCNN. To evaluate the effectiveness and robustness of our method, we conduct several experiments on both synthesized and real cloud images.

Synthesized Cloud Images Firstly, we verify the ability of our method using synthetic cloud images as the source images. Some examples of these source images are shown in the insets of the right column in Fig. 8. These clouds have different control parameter distributions. *Result 1* is generated by a smaller area of air parcel emission than the clouds in other rows. *Result 2* absorbs more heat from the ground during the formation process. *Result 3* absorbs heat from the generated area more evenly at each frame. We first use the proposed PPCNN to generate control parameters from these source images, and then simulate the cloud formation processes with these parameters using the physically based method [DKNY08]. The three images on each row display cloud evolution at different stages, which are at 40, 70 and 100 frames from left to right. Particularly, the right image in each row corresponds to the rendered cloud of the finally generated density distribution. The generated clouds highly resemble the real images by justifing from the cloud shape and appearance based on visual inspection. It indicates that the PPCNN can find the optimal control parameters for clouds of different shapes. This result is satisfactory for the purpose of finding the parameters to the cloud formation.

We implement quantitative analysis on the structure similarity between the given source image and the rendered image using the generated density field of the cloud. As we render the simulation-based result and the generated image under the same illumination conditions, similar to the method [EUT19], we calculate the structural similarity index (SSIM) and the mean and standard deviation of peak signal-to-noise ratio (PSNR) to evaluate the structure similarity. Table 1 summarizes the similarity statistics of all presented synthetic image examples. The averaged PSNR value is 30.68 for image differences, across all test dataset. These tests indicate that our method is capable of effectively estimating control parameters to drive cloud formation to match given images.

Real Cloud Images Secondly, we investigate the capability of our proposed method to handle real cloud images as the source images. Two examples of these source images are shown in the insets of the right column in Fig. 9. These two cumulus clouds exhibit significantly different shape characters as they are influenced by different initial environment conditions, such as the profile of the soil temperature. Cloud4 shows two convex upward parts at the cloud top, while cloud5 has one bulge at the cloud top. In addition, the difference between the two cloud shapes may be caused

Figure 8: *Cloud simulated from synthesized cloud images with different shape features. From top to bottom: Result 1, Result 2 and Result 3, respectively. Source cloud images are shown in the small insets of the right column.*

Table 1: *Shape Similarity Analysis*

Image	Cloud Area Ratio		SSIM	PSNR
	Simulation-based	Ours		
Result 1	10.77	10.80	0.89	20.26
Result 2	22.83	23.93	0.93	23.66
Result 3	21.29	22.21	0.95	28.29

by the distribution of the ground temperature. With our method, we generate the control parameters for these two source cloud images, respectively. We then simulate cloud formation using the learned parameters based on the fluid dynamics. The three images on each row display cloud evolution at different stages, which are at 40, 70 and 100 frames from left to right. Particularly, the right image in each row corresponds to the final reconstructed cloud. The results show that our approach can control the cloud formation process to generate cumulus clouds highly resembling the clouds from user specified real photograph.

As our work does not to directly reconstruct natural images, it is unsuitable to measure the natural image against the generated one pixel by pixel. We choose the intersection over union (IOU) to measure the shape and the perceptual loss [JAFF16] to analyze the structure similarity between the given natural image and the generated one. To calculate the IOU, we first resize the natural and the generated images to the same size. Then, we binarize them to eliminate the influence of color. In our experiments, the cloud and background pixels are set to be 1 and 0, respectively. To calculate the perceptual loss, we calculate the cosine distance of the deep features of images which are extracted by using the subnetwork F as shown in Fig.4. The results are shown in the column of $\lambda = 0.001$ in Table 3. These tests indicate that the natural image and the generated one have similar shape and structure.

Figure 9: *Cloud simulated from real cloud photographs. From top to bottom: Result 4 and Result 5, respectively. Source cloud images are shown in the small insets of the right column.*

5.3. Comparison to Previous Methods

In order to demonstrate the importance of estimating the control parameters on the cloud simulation, we compared our method with the feedback control approach [DKNY08] and the method proposed by Yuan et al. [YLH*14].

Fig. 10 compares our method against the simulation approach of Dobashi et al. [DKNY08]. We first generate the density field of the cumulus cloud as shown in the leftmost image to match the user-specified top contour line (the red curve) using the method [DKNY08]. Then, we estimate the optimal control parameters using the rendered image of the density field as input. Finally, we simulate the cloud formation process using the generated physical control parameters as initial conditions. The second to the rightmost images show the snapshots of the cloud formation process at frames 20, 40, 60, 80 and 100. The rightmost image shows that the generated cloud can match with the cloud shape specified by the contour line drawn by the user. Moreover, there is no extra manual force, e.g., the external force due to the geometric potential field, required to control the cloud formation during the simulation process. Meanwhile, our method estimates the controlling parameters from the standpoint of the overall simulation. It avoids manually adjusting the physical parameters according the difference between the simulated cloud and the target at each frame. The property of solving inverse formation explains why our approach can generate controlled cloud animation with better visual plausibility than methods that use non-zero ghost forces throughout the simulation domain.

As we simulate the cloud formation process to match a user-specified cumulus cloud image, the reconstructed density distribution of cloud is more visually natural. To demonstrate this, we compare our method with the method [YLH*14] modeling cumulus cloud shape by inversely solving the single scatter equation. The method assumes that the shape of the cloud is symmetric. After obtaining the front and back surfaces, the side surface is formed by connecting the vertices on the two boundaries of the front and back surfaces. As the method [YLH*14] directly reconstructs a 3D model from input images, the method could generate more similar model with the input image than our method. However, the recon-

Figure 10: *A comparison of our method against [DKNY08]. The leftmost image is generated by using [DKNY08] and the red curve is given by users. The second to the rightmost images display the cloud formation process using our method.*

Figure 11: *Compare with Yuan et al. [YLH*14]. From left to right: input image, reconstructed image with [YLH*14], and generated image using our approach, respectively.*

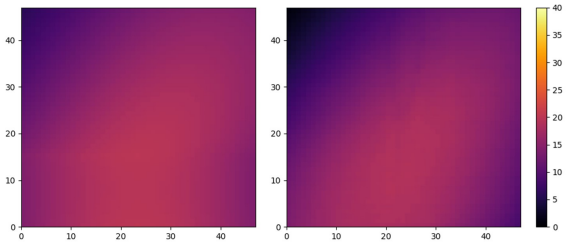

Figure 12: *Examples of the control parameters for Result 3 at frame 30. The left is the simulation-based one, and the right is the generated parameters using our method.*

structed side surface using the method [YLH*14] is less natural than ours (see Fig. 11). There is an obvious center line on the surface of the cloud due to manual stitching. Meanwhile, the symmetric assumption of [YLH*14] also leads to unnatural cloud shape.

5.4. Performance Analysis

Generated Parameters Analysis To analyze the parameter generation power of the PPCNN, we compare generated parameters for the images, which have a direct correspondence to the original dataset, i.e. the simulation-based samples. Fig. 12 is the visualizaiton of the control parameters at the frame 30 for *Result 3*. The simulation-based and generated results are shown on the left and the right, respectively. It is obvious that the parameter distribution of the simulation-based data and the generated one are similar by visual inspection. Meanwhile, we do more detailed analysis for the generated parameters as shown in Fig. 13. Both the left and middle images of the Fig. 13 show the parameter mean plots comparing the simulation-based result and our model output for the *Result 1* image and all test dataset, respectively. The right image shows the plots of parameter mean variance for each frame across all the test dataset. The mean of temperature $mean_i$ for the ith frame is calculated as follows: $mean_i = \frac{1}{W_s \times H_s \times |D|} \sum_D \sum_{y=1}^{H_s} \sum_{x=1}^{W_s} S_T^i(x,y)$, where $|D|$ denotes the amount of the test dataset. The variance var_i is defined as: $var_i = \frac{1}{W_s \times H_s \times |D|} \sum_D \sum_{y=1}^{H_s} \sum_{x=1}^{W_s} ||S_T^i(x,y) - mean_i||^2$. The generated and simulation-based parameters are shown on the top row. We calculate the mean absolute percentage error (MAPE) or all cells on the bottom level in the simulation domain at each time (see the bottom row of Fig. 13) to measure the difference between the generated and simulation-based data. The MAPE accounts for that the max value for the mean of control parameters is less than 3%. As shown in the top-right figure, the distribution of the generated parameters is similar with the simulation-based one. We further calculate the mean absolute percent error between the generated vs simulaion-based per sample, and Fig. 14 shows the statistical results. It is noted that 82.6% of test data have error lower than 0.2. Table 2 gives the statistics of mean and variance of the simulation-based and generated control parameters for the three synthetic images as shown in Fig. 8. These tests indicate that the CNN shows a high control parameter estimation quality.

For the real images, we calculate the mean of each control parameter cell across all simulation frames. The results for Cloud4 and Cloud5 are visualized in the left and right of Fig. 15, respectively. There are three areas with more high temperature in the left figure, which correspond to the tree humps of the input image. As the real cloud5 has one top, the right figure contains only one area with more high mean temperature. The experimental results demonstrate that our method effectively estimates the distribution of control parameters which can describe the intricate shapes of the complex real cloud as shown visually in Fig. 9.

Cost of Simulation According to our method, the cost of simulation mainly includes two items, i.e., the cost of training and the time of cloud formation simulation. The training cost depends on the number of iterations and the resolution of the simulation grid. In our experiments, the resolution of the simulation grid is 64^3. Under these setting, the training time for the parameter-predicting CNN and the autoencoder CNN are 4h and 24h, respectively. After training the PPCNN, the time of parameter prediction is 0.016s. The formation simulation is carried out on the CPU and the GPU is used for rendering clouds. We used Blender for rendering. The average computation time for each time step of the simulation is 0.31

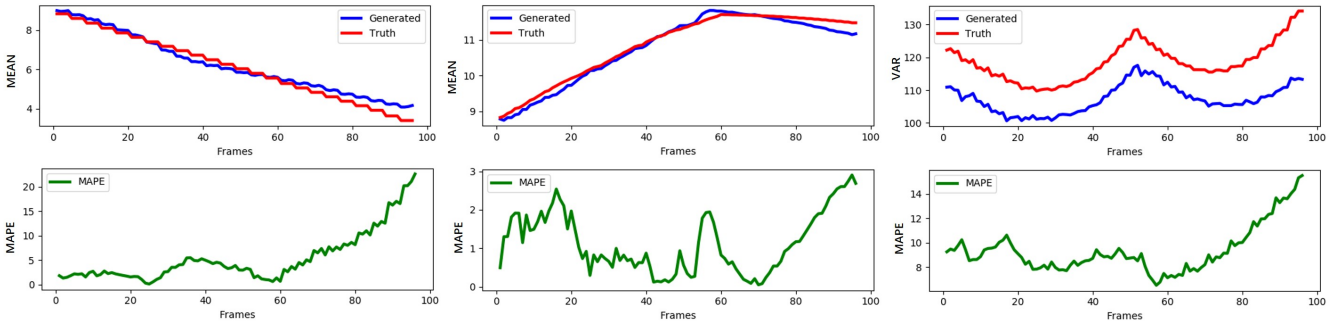

Figure 13: *Examples of the generated parameters. From left to right: the mean for Result 1, the mean and the variance across all test dataset, respectively. The top row shows the distribution of generated parameters and the simulation-based data. The bottom row shows the MAPE for each frame of control parameters.*

Table 2: *Generated Parameter Analysis*

Image	Frame30				Frame60				All Frames			
	Simulation-based		Generated		Simulation-based		Generated		Simulation-based		Generated	
	Mean$^\circ C$	Variance	Mean$^\circ C$	Variance	Mean$^\circ C$	Variance	Mean$^\circ C$	Variance	Mean$^\circ C$	Variance	Mean$^\circ C$	Variance
Result 1	7.17	112.85	6.99	71.97	5.56	70.69	5.59	45.31	6.12	92.01	6.24	64.41
Result 2	36.43	23.98	27.16	22.35	20.82	14.89	21.59	15.21	22.61	50.7	23.76	42.84
Result 3	22.19	9.02	24.44	18.32	16.21	4.82	18.92	12.18	22.29	47.03	20.67	34.47

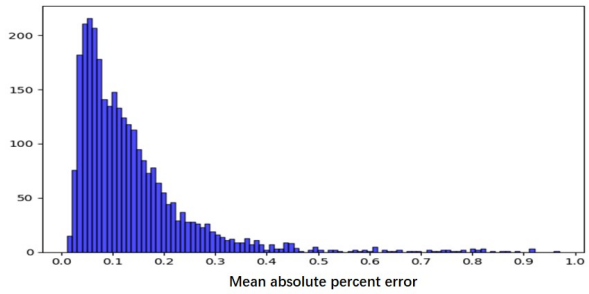

Figure 14: *Histogram of mean absolute percent error between the generated and simulation-based data per sample. The vertical axis denotes the sample size.*

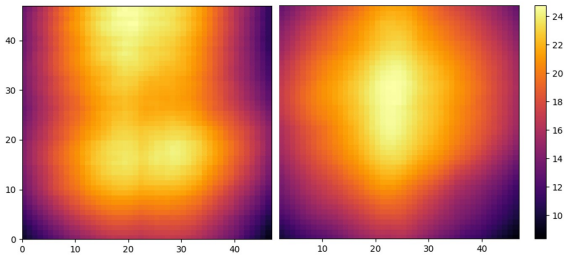

Figure 15: *Mean temperature plot of each cell of generating control parameters across all simulation frames for real images as shown in Fig. 9. From left to right show the results for Result 4 and Result 5, respectively.*

seconds, while that of Dobashi et al. [DKNY08] is 0.37 seconds. However, some care must be taken when interpreting this number because our method takes a cloud image as constraints, while the method [DKNY08] only uses a top contour as constraints.

Ablation Analysis In this experiment, we study the effect of the adversarial network (as shown in Fig.4) to the overall performance. Fig.4 shows that the network F is updated using a combination of losses from the supervised parameter prediction network P and the adversarial network G-D. To analyze the effects of GAN, firstly, we only use the parameter estimation branch, i.e., the F-P pair, and only use the synthetic data to train. The setting, i.e., $\lambda = 0$, makes the network F to be updated only by using the gradients from the network P. After training the network, F-P, we test the performance on the real images. Fig. 16 shows the generated results for Cloud4 and Cloud5. Comparing with Fig. 9 by visual inspection, the branch of GAN improves the performance of parameter prediction.

Secondly, we further change the parameter λ to analyze the effects of GAN. As the GAN branch mainly influences the preformance for real cloud images, we quantitatively measure the results for *Result 4* and *Result 5* by calculating the IOU and the perceptual loss. We set the λ to be 0.001, 0.01 and 0.1 in the experiments. Table 3 presents the results for these settings. It can be observed that the GAN does improve the performance of predicting control parameters for real cloud images. As λ is bigger, the network F learns more information about real images without the corresponding control parameters. It causes that the feature descriptor extracted by F contains less information about the control parameters. Hence, it is noted that the model PPCNN obtains the best performance under the setting $\lambda = 0.001$.

Figure 16: *Examples for real images generated only using the F-P branch. From letf to ringt: Result 4 and Result 5.*

Table 3: *Performance Analysis for different λ value*

Image	$\lambda = 0$		$\lambda = 0.001$		$\lambda = 0.01$		$\lambda = 0.1$	
	IOU	Cosine	IOU	Cosine	IOU	Cosine	IOU	Cosine
Result 4	0.74	0.89	**0.76**	**0.95**	0.51	0.82	0.39	0.58
Result 5	0.68	0.76	**0.82**	**0.85**	0.66	0.54	0.58	0.57

Convergence of The Algorithm In order to evaluate the convergence of the presented architecture, we plot the training error after each iteration. Training time highly depends on the prediction quality and the dimensionality of the control parameters. Generally, the higher the dimensionality of the control parameter latent space, the lower the model convergence rate. Meanwhile, the parameter λ in Eq. (10) also influences the convergence of the learning. In our experiments, we find that the mean error is 0.45, 0.46, 0.52 and 0.53 with $\lambda = 0$, $\lambda = 0.001$, $\lambda = 0.01$ and $\lambda = 0.1$, respectively. Fig. 17 shows the convergence plot of our model with $\lambda = 0.001$, with training iterations on the x-axis and error on the y-axis. The curve shows that the control parameters is already estimated with good accuracy when the model is trained about 40 iterations. Although the L1 loss is lower when $\lambda = 0$, the IOU measures is smaller than that of $\lambda = 0.001$ (see the Table 3). This is because that the network F does not learn the features of natural images due to only training on the simulation-based data when $\lambda = 0$. When the λ value is larger than 0, it is noted that the L1 loss is smaller, the IOU is bigger.

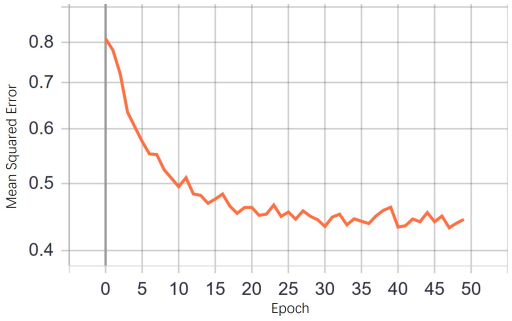

Figure 17: *Convergence plot of the L_1 loss for the parameter latent space with 512 dimensionality.*

Figure 18: *Example of high resoltution. From left to right: Frame 40, Frame 70 and Frame 100, respectively.*

5.5. Discussion and Limitations

In this experiment, we demonstrate how our model performs on the additional dataset with higher spatial resolutions, i.e., 128^3. Due to the additional complexity of this dataset, we set the latent space size of the control parameters, m_l, to be 2048, correspondingly. The results (see Fig.18) show that our method successfully predicts the control parameters for the input.

To validate physical rationality of predicting parameters, we compare our method to state-of-the-art researches on the relationship between cumulus shape and latent heat supplying. For example, the profile of the ground temperature generating *Result 1* is monotonically decreasing (see the top-left image in Fig. 13) similar to the cloud shown in Fig. 6(b) of [DG17], and the two clouds have the similar upside-down turret shape. We also compare the variation of control parameters for the cloud with almost constant width to the well-known case [DPS*14], which studied the topic in a laboratory environment. Fig. 19 has a visualization of the ground temperature predicted by our method for the clouds having similar shapes as the cloud highlighted in the green box, which is generated from [DPS*14]. The variations of predicting parameters are similar to the profile of latent heat supply for the red cloud which is simulated by supplying nearly constant heat in the method [DPS*14]. As this kind of cloud is generated due to overall rising motion of the air parcels driven mainly by buoyancy forces [DPS*14], it is noted that the top of the cloud generated with higher temperature is higher than the one with lower temperature.

While we have shown that our approach can generate realistic cloud with desired shape, it also has some limitations. First, our system is designed to learn the space-time distribution of the ground temperature which controls the cumulus cloud formation process. This implies that the proposed method is not suitable for estimating full environment conditions to control cloud formation. Meanwhile, since we train the PPCNN on the dataset of cumulus clouds, our approach cannot generate clouds without the main features of cumulus cloud such as cirrus, clouds without a wide base (as shown in Fig. 20) due to lack of this kind of shape in the dataset. Another limitation of our method is its dependence on the simulated dataset, which does not fully express the real physical evolution. Hence, it is not suitable for our method to some application scenarios in which users need higher accuracy for coming up solutions. In addition, our method inherits the limitations of the deep learning on different domains. The method does not guarantee successful prediction of proper control parameters for arbitrary input images, e.g., freehand contour sketching of cloud.

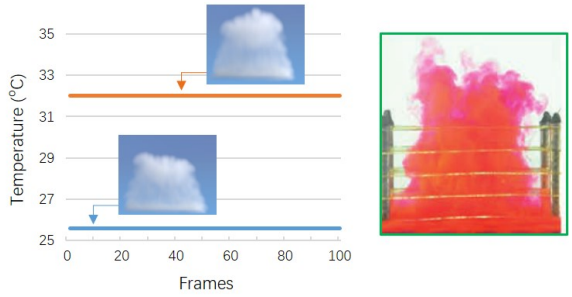

Figure 19: *Examples of validating physical rationality. Both the blue and orange lines are the visualization of the ground temperatures predicted by our method for the clouds showing above the two lines, respectively. The two clouds and the cloud generated from the method [DPS*14], which is highlighted with a green box, have similar shapes with almost constant width. The variations of predicting parameters are similar to the profile of latent heat supply for the red cloud, which is simulated by adding nearly constant moderate heat in the laboratory environment of [DPS*14].*

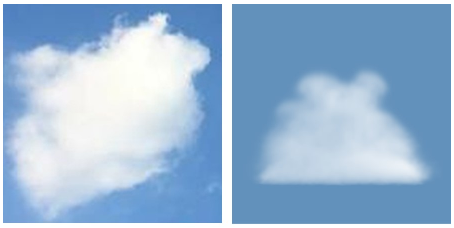

Figure 20: *Example of failing to generate clouds without a wide base. The left is the input image, the right is the generated one.*

6. Conclusions and Future Work

In this paper, we presented a learning-based approach for solving the inverse cumulus cloud formation. In our approach, we proposed a parameter predicting neural network to estimate the proper control parameters for a given image. By designing an auto-encoder neural network, the higher dimensionality of control parameters is reduced to a lower dimensionality, which make our approach achieve a high convergence rate. The optimal control parameters through the PPCNN after training can be estimated in real-time. The results show that our method could simulate natural cloud formation and finally obtain desired cumulus cloud shape by using the learned parameters as initial conditions. Another important aspect of the proposed system is its representation capability about the relationship between the cloud shape and the distribution of the ground temperature. We demonstrate that our method is helpful for users to control cloud formation for tuning weather conditions via human intervention.

In the future, we would like to try more challenging problems. In some applications, such as flight simulation, it is useful to estimate the full environment conditions to reconstruct a similar scene for a user-given real cloud image. We plan to extend our framework to handle the problem of full environment condition estimation. To achieve the objective, it is helpful to generate real cloud evolution dataset by combining the cloud formation simulation in the laboratory and meteorological satellite data. In our current method, the user specifies the shape of clouds via giving a real or synthesized image. However, some users may want to specify cloud shape by freehand counter lines or semantic descriptions, when there is no proper image meeting the users' expectations. Extending our method to incorporate user constraints is also an important future direction.

7. Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This paper is supported by the National Key R&D Program of China (No.2017YFB1002702) and the National Natural Science Foundation of China (No.61572058).

References

[ABP14] ANNIBALLE R., BONAFONI S., PICHIERRI M.: Spatial and temporal trends of the surface and air heat island over milan using modis data. *Remote Sensing of Environment 150* (2014), 163–171. 7

[Bri15] BRIDSON R.: *Fluid simulation for computer graphics*. CRC press, 2015. 5

[CT17] CHU M., THUEREY N.: Data-driven synthesis of smoke flows with cnn-based feature descriptors. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 69. 3

[DG17] DUARTE R. P., GOMES A. J.: Real-time simulation of cumulus clouds through skewt/logp diagrams. *Computers & Graphics 67* (2017), 103–114. 1, 3, 12

[DIO*12] DOBASHI Y., IWASAKI W., ONO A., YAMAMOTO T., YUE Y., NISHITA T.: An inverse problem approach for automatically adjusting the parameters for rendering clouds using photographs. *ACM Trans. Graph. 31*, 6 (2012), 145–1. 3

[DKNY08] DOBASHI Y., KUSUMOTO K., NISHITA T., YAMAMOTO T.: Feedback control of cumuliform cloud formation based on computational fluid dynamics. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 94. 1, 2, 4, 8, 9, 10, 11

[DPS*14] DIWAN S. S., PRASANTH P., SREENIVAS K., DESHPANDE S., NARASIMHA R.: Cumulus-type flows in the laboratory and on the computer: Simulating cloud form, evolution, and large-scale structure. *Bulletin of the American Meteorological Society 95*, 10 (2014), 1541–1548. 12, 13

[EUT19] ECKERT M.-L., UM K., THUEREY N.: Scalarflow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 239. 8

[EvdKVB13] ESSA W., VAN DER KWAST J., VERBEIREN B., BATELAAN O.: Downscaling of thermal images over urban areas using the land surface temperature–impervious percentage relationship. *International Journal of Applied Earth Observation and Geoinformation 23* (2013), 95–108. 7

[FBDY15] FERREIRA BARBOSA C. W., DOBASHI Y., YAMAMOTO T.: Adaptive cloud simulation using position based fluids. *Computer Animation and Virtual Worlds 26*, 3-4 (2015), 367–375. 1

[GDAB*17] GARCIA-DORADO I., ALIAGA D. G., BHALACHANDRAN S., SCHMID P., NIYOGI D.: Fast weather simulation for inverse procedural design of 3d urban models. *ACM Trans. Graph. 36*, 2 (Apr. 2017). URL: http://doi.acm.org/10.1145/2999534, doi:10.1145/2999534. 2, 3, 4

[GKZ*16] GHIFARY M., KLEIJN W. B., ZHANG M., BALDUZZI D., LI W.: Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision* (2016), Springer, pp. 597–613. 6

[GPAM*14] GOODFELLOW I., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680. 6

[HBSL03] HARRIS M. J., BAXTER W. V., SCHEUERMANN T., LASTRA A.: Simulation of cloud dynamics on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (2003), Eurographics Association, pp. 92–101. 1

[HJ14] HOUZE JR R. A.: *Cloud dynamics*, vol. 104. Academic press, 2014. 3

[HK04] HONG J.-M., KIM C.-H.: Controlling fluid animation with geometric potential. *Computer Animation and Virtual Worlds 15*, 3-4 (2004), 147–157. 2

[HNK*19] HOU L., NGUYEN V., KANEVSKY A. B., SAMARAS D., KURC T. M., ZHAO T., GUPTA R. R., GAO Y., CHEN W., FORAN D., ET AL.: Sparse autoencoder for unsupervised nucleus detection and representation in histopathology images. *Pattern recognition 86* (2019), 188–200. 5

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1026–1034. 8

[JAFF16] JOHNSON J., ALAHI A., FEI-FEI L.: Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision* (2016), Springer, pp. 694–711. 9

[JSP*15] JEONG S., SOLENTHALER B., POLLEFEYS M., GROSS M., ET AL.: Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 199. 3

[KAT*19] KIM B., AZEVEDO V. C., THUEREY N., KIM T., GROSS M., SOLENTHALER B.: Deep fluids: A generative network for parameterized fluid simulations. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 59–70. 2, 3, 5

[LSD*18] LORE K. G., STOECKLEIN D., DAVIES M., GANAPATHYSUBRAMANIAN B., SARKAR S.: A deep learning framework for causal shape transformation. *Neural Networks 98* (2018), 305–317. 2

[MDN02] MIYAZAKI R., DOBASHI Y., NISHITA T.: Simulation of cumuliform clouds based on computational fluid dynamics. *Proc. Eurographics 2002 Short Presentation* (2002), 405–410. 1

[MLC*18] MUKHERJEE R., LI Q., CHEN Z., CHU S., WANG H.: Neuraldrop: Dnn-based simulation of small-scale liquid flows on solids. *arXiv preprint arXiv:1811.02517* (2018). 3

[MMCS11] MASCI J., MEIER U., CIREŞAN D., SCHMIDHUBER J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks* (2011), Springer, pp. 52–59. 5

[MTP*18] MA P., TIAN Y., PAN Z., REN B., MANOCHA D.: Fluid directed rigid body control using deep reinforcement learning. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 96. 2, 3

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. In *ACM Transactions On Graphics (TOG)* (2004), vol. 23, ACM, pp. 449–456. 2

[PHC15] PATRAUCEAN V., HANDA A., CIPOLLA R.: Spatio-temporal video autoencoder with differentiable memory. *arXiv preprint arXiv:1511.06309* (2015). 5

[PM16] PAN Z., MANOCHA D.: Efficient optimal control of smoke using spacetime multigrid. *arXiv preprint arXiv:1608.01102* (2016). 2

[RTWT12] RAVEENDRAN K., THUEREY N., WOJTAN C., TURK G.: Controlling liquids using meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012), Eurographics Association, pp. 255–264. 2

[SH] SANDLER M., HOWARD A.: Mobilenetv2: The next generation of on-device computer vision networks. 7

[Str03] STREUTKER D. R.: Satellite-measured growth of the urban heat island of houston, texas. *Remote Sensing of Environment 85*, 3 (2003), 282–289. 7

[SY05] SHI L., YU Y.: Taming liquids for rapidly changing targets. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM, pp. 229–236. 2

[SZL*18] SONG J., ZHANG H., LI X., GAO L., WANG M., HONG R.: Self-supervised video hashing with hierarchical binary auto-encoder. *IEEE Transactions on Image Processing 27*, 7 (2018), 3210–3221. 5

[TMPS03] TREUILLE A., MCNAMARA A., POPOVIĆ Z., STAM J.: Keyframe control of smoke simulations. In *ACM Transactions on Graphics (TOG)* (2003), vol. 22, ACM, pp. 716–723. 2

[TSSP17] TOMPSON J., SCHLACHTER K., SPRECHMANN P., PERLIN K.: Accelerating eulerian fluid simulation with convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 3424–3433. 1, 3

[WBT18] WIEWEL S., BECHER M., THUEREY N.: Latent-space physics: Towards learning the temporal evolution of fluid flow. *arXiv preprint arXiv:1802.10123* (2018). 3

[XFCT18] XIE Y., FRANZ E., CHU M., THUEREY N.: tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 95. 3

[XWY19] XIAO X., WANG H., YANG X.: A cnn-based flow correction method for fast preview. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 431–440. 1, 2, 5

[YLH*14] YUAN C., LIANG X., HAO S., QI Y., ZHAO Q.: Modelling cumulus cloud shape from a single image. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 288–297. 2, 3, 9, 10