The Visual Computer (2021) 37:1931–1948
https://doi.org/10.1007/s00371-020-01953-y

SURVEY

A survey of modeling, rendering and animation of clouds in computer graphics

Prashant Goswami[1]

Published online: 14 August 2020

Abstract
Clouds play an important role in enhancing the realism of outdoor scenes in computer graphics (CG). Realistic cloud generation is a challenging task, which entails processes such as modeling, photorealistic rendering and simulation of the clouds. To these ends, several techniques have been proposed within the CG community in the last 4 decades with one or more of the above stated focuses. The growth of modern hardware has also enabled development of techniques that can achieve cloud display and animation at interactive frame rates. In this survey, we review the prominent work in the domain and also summarize the evolution of the research over the time.

Keywords Clouds · Modeling · Rendering · Animation

1 Introduction

Clouds form an integral component of several outdoor scenes and are hence key to enhancing their realism. This applies both to static and dynamic scenes. The focus in static scenes is to obtain a photorealistic image where the clouds can convey a convincingly real static appearance. On the other hand, dynamic scenes could be more demanding with the additional requirement to have naturally evolving or animated clouds. For applications such as movies, the computational time to simulate or render the clouds might not pose any hard constraints, while some other applications such as the flight simulators and games might necessitate that the computational time spent on the clouds is only a small fraction of the total frame time. In order to cope with the different scenarios, the techniques could be offline, online or some combination thereof in terms of their computational aspects.

Over the years, work has been done on the different fronts to generate realistic clouds in CG. Cloud modeling approaches strive to create shapes that resemble the real clouds. In order to convey the impression of realness, clouds further need an effective illumination model to capture the light transport on the generated shapes. For dealing with dynamic scenes, the animation techniques are required to produce a viable evolution of the modeled clouds as a function of time. Furthermore, while some applications require the generated clouds and the rendering to be more accurate, others can trade off their quality for the efficiency.

Clouds are broadly classified into around 10 types (Fig. 1), differing significantly in their appearance and formation dynamics [57,67]. Furthermore, clouds can also be classified based on the various parameters, for instance, elevation (low, medium or high), content (water, ice) or structure (layered, scattered). In terms of their appearance, the cumulus clouds and its variants are particularly interesting owing to the interesting patterns they form. These clouds are formed as a result of the strong vertical ascending currents and often assume conspicuous shapes. Hence, a bulk of the research in CG is directly or indirectly targeted toward these types of clouds.

In this survey, we review the various techniques of cloud generation in CG in the light of the aforementioned three major aspects. Section 2 covers the prominent work related to cloud *modeling*. In Sect. 3, the fundamentals of light transport are introduced, followed by a review of the existing methods to *render* and illuminate the clouds. Cloud *animation* techniques are studied in Sect. 4 after a brief overview of the basics of the underlying physics. The various methods are discussed for their performance in Sect. 5, followed by the conclusion in Sect. 6.

✉ Prashant Goswami
prashant.goswami@bth.se

[1] BTH Karlskrona, Karlskrona, Sweden

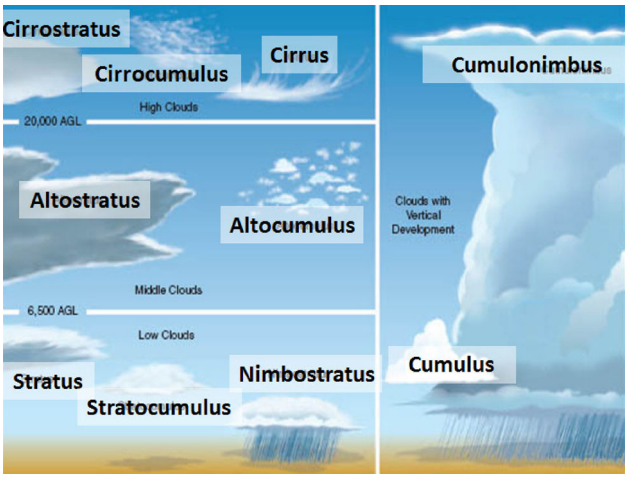


Fig. 1 Various classification of clouds based on their elevation and appearance [57]. Cloud elevation is often reported by weather stations in feet above ground level (AGL)

2 Modeling

Cloud modeling in CG refers to creating shapes that resemble real clouds in their appearance. Varied types of techniques have been proposed to the end of cloud modeling. This includes mesh-based and volumetric methods, generating shapes from texture or noise and even from images. In this section, we discuss the existing modeling methods by classifying each method to the most relevant category it belongs to.

Texture Some of the earliest work has employed 2D/3D textures for the cloud shape modeling. A simple method to model cloud structure is presented in [31] by Gardner. In their paper, the authors employ ellipsoids to model coarse three-dimensional cloud structure and apply a mathematical texturing function (simplified Fourier series) on it to model the cloud detail, shading intensity and translucence of the sky. To create the cloud layers, the same parameters are applied on a single textured plane instead of the ellipsoids. The method supports both the vertical and horizontal cloud development and could create impressive cirrus, stratus and cumulus clouds for the technology available at that time.

Based on the Gardner's work, Elinas and Stuerzlinger [29] model clouds as composed of ellipsoidal primitives. The ellipsoids are textured, and the texture properties like transparency are controlled to simulate irregular appearance of the clouds. The solid texturing of the polygonal mesh objects is used by Ebert and Parent for modeling gaseous phenomena in [26], where controlled transparency of objects or planes defines the space occupied by the gaseous substance. Hardware interpolated opacity on textures obtained with a fractal algorithm is applied in [96]. In [103], the idea of digital synthesis of Jupiter is explored by creating planetary atmospheric flow. In their method, software tools are employed to convert the texture images to particles. The particles are input to a fluid dynamics engine that updates their positions and converts them back to images.

Noise Texture mapping on 3D shapes could be a complex problem. To this end, it is often easier to generate shapes with the help of procedural noise. This observation has been exploited in a number of works that employ one or more kinds of noise to model the clouds. Spectral synthesis motivated by turbulence theory is employed for cloud modeling by Sakas in [82]. Texture is defined in the frequency domain and transformed to the Euclidean space. In [40], a simple noise-based approach to generate static procedural clouds on GPU is implemented. Xu et al. [102] simulate clouds based on a modified cellular automata method on the GPU. The stochastic probability fields created from Brownian motion functions control the cloud evolution.

Webank et al. [99] create several types of clouds procedurally through variations of density functions and cloud shape functions. A control function additionally influences the large-scale cover of the corresponding cloud type. The other relevant parameters that define a particular cloud type (elevation base, altitude range, etc.) are also passed as inputs to the cloud modeler. The volumetric effect for cumulus cloud rendering in Neyret's phenomenological shader [69] is created by adding Perlin noise on the surface representation of the cloud. Goswami and Neyret [34] model clouds through hypertexture generation [74] inside large spherical parcels based on the underlying physics parameters on the GPU. This is improved upon in Goswami [32] by introducing *cloud map*, a precomputed noise texture to govern cloud shapes and to improve the rendering efficiency.

Geometric Ebert in [25] uses volumetric modeling to generate cumulus clouds with the help of a procedural volumetric implicit function. The idea to generate cloud density with cellular automata and to evolve them with the time-varying transition rules is introduced in [15,16]. Other variations and optimizations [3,30] have also been proposed. Clouds are modeled as isosurfaces using marching cubes by Trembilski and Broßler in [95]. Given coarse geometric data, postprocessing steps like refining, sharp edge smoothing and vertex displacement are applied to achieve the cloud visualization. Schpok et al. [87] provide a cloud modeling framework that uses volumetric implicits to define the density, transparency and shadowing of the cloud. The user interactively outlines the cloud shape using ellipsoids and describes the low-level details from a collection of preset noise filters.

Hufnagel et al. [44] visualize clouds from weather volume data by extracting information and interpreting the cloud type and appearance information from it. The different types of clouds are first identified and then modeled by placing par-

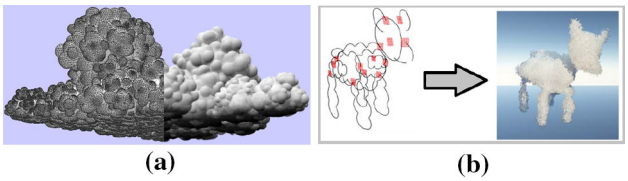


Fig. 2 Cloud modeling with **a** hierarchical blobs in Neyret [6] and **b** sketched 2D outlines by Wither et al. [101]

ticles (metaballs) of appropriate size, density and texture in the scene. They extended the use of original metaballs with flattened metaballs and metaballs textures to improve upon the efficiency and realism of the clouds.

Shapes Shapes from meshes or curves have also served as a useful inspiration to generate the clouds. Bouthors and Neyret [6] model cumulus cloud shapes on meshes with the help of a blob hierarchy superimposed on top of each other (Fig. 2a). While placing a number of child blobs on the parent, a combination of field functions controls several factors like cloud size, overlap, bottom flatness. Sethi [88] models clouds through B-Spline surface where the sampled height values constitute the control points.

Wither et al. [101] present a method to model cumulus clouds from sketched 2D outlines. The skeleton is inferred from the sketched silhouette, followed by placing the spheres along the skeleton which provides the final mesh (Fig. 2b). In Stiver et al. [92], an input sketch is converted into a cloud mesh and this mesh is filled with particles to model the various types of clouds. Cloud modeling is achieved in Yu and Wang [106] by sampling input 2D shapes to particles and 3D shapes to tetrahedrons. When using 2D shapes, the data are stretched to 3D, leading to the step of extraction of medial axis, which aids in the sampling process later. Correspondence is established between the source and the target cloud models, and the cloud is morphed along a linear or more sophisticated path.

The clouds created for "Puss In Boots" [62] are modeled as meshes representing shapes that could be used to dress the set and house the animated characters. The meshes together with the procedural noise are converted into a compact data structure for sparse volume data named *volumetric dynamic grid* (VDB) that enable fast and cache-coherent data access of large volumes.

Image-based Cloud images captured from terrestrial angles have been utilized to model the cloud shapes. Peng and Chen [73] propose to capture variable cloudiness in input images by formulating it as a labeling problem. Each sky pixel is assigned a label based on the Igawa sky model (uses solar zenith, azimuth, etc.) [45], while minimizing a formulated energy function with respect to the labels in the provided input image.

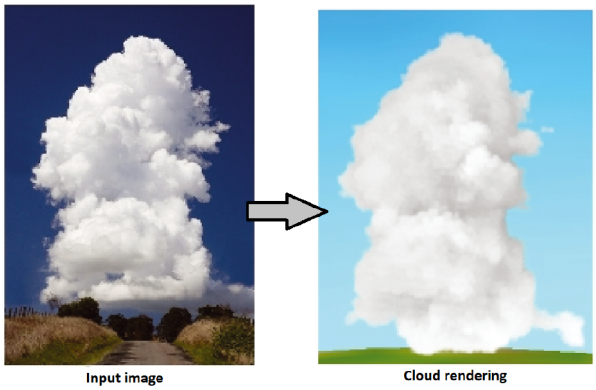


Fig. 3 Cloud modeled and rendered from a single image by Yuan et al. [107]

Alldieck et al. [1] generate cloud models from a hemispherical image input covering an entire sky. To this end, firstly the sun illumination is filtered, followed by classification of the cloud and sky pixels in the image and then reconstruction of the sky without clouds. The cloud intensity and opacity values are used to create vertices on the hemisphere, which generates the cloud mesh. Clouds are also modeled from a single image in [107] by Yuan et al. by assuming a symmetric cloud structure. The image is segmented into various subregions, and the height field is computed using the pixels labeled as clouds. A propagation procedure is devised that constructs cloud geometry from the cloud pixels progressively. In order to improve the generated appearance, the front part of the cloud is refined by adding shape constraints and the back portion geometry is simplified. Finally, the cloud surface obtained is filled with the particles via an adaptive sampling process (Fig. 3).

Satellite-images The cloud distribution present in the satellite images is employed as a tool to generate similar weather visualization effects. Dobashi et al. [17,18] and Tomoyuki and Dobashi [94] proposed cloud modeling in 3D using metaballs by making use of a single satellite image. To this end, cues are taken from the satellite image by classifying each pixel as either cloud or background. Metaballs are generated in pixels with the maximum intensity identified as cloud regions of the satellite image. After each such addition, the center and radius of the metaball are approximated. The termination criterion is based on the difference of the synthetic image with that of the satellite image, given an error threshold. Wang et al. [100] generate satellite view clouds based on the weather forecast data and group all the particles into water, ice or snow to determine their extinction and scattering coefficients. Kowsuwan and Kanongchaiyos [53] generate

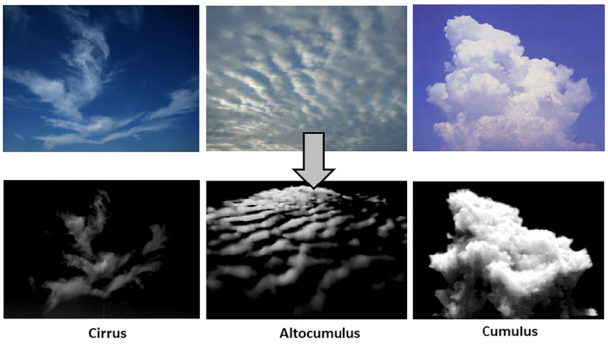


Fig. 4 Cirrus, altocumulus and cumulus cloud texture and the corresponding density distribution, generated from single images by Dobashi et al. [19]

3D clouds from satellite images combined with the creation rules of cellular automata.

Dobashi et al. [22,23] build a volumetric hierarchical data structure for the earth-scale cloud visualization which categorizes the visible data blocks as surface, point or volume to enforce the different levels of detail. Their system precomputes and stores the integrated intensities and opacities of the cloud data for various viewing and lighting directions for efficient rendering at runtime. Griffith [35] visualize clouds generated by time-varying *large eddy simulations* (LES) for virtual reality applications.

Dobashi et al. [19] synthesize density distribution of different types of clouds from single images. The initial and common step while modeling various cloud types is the computation of *cloud image* from a given image that extracts the opacity and intensity of clouds in the given image. Thereafter, cirrus clouds are created with 2D texture since they are thin and seldom contain self-shadows. For the altocumulus clouds, a three-dimensional density distribution is defined using metaballs. In order to model the cumulus clouds, surface shape is generated by calculating the thickness at each pixel (Fig. 4).

Yuan et al. [108] extract geometric structures of clouds from multi-spectral satellite images. This is achieved by classifying pixels as clouds or non-clouds in the visual image, estimating the cloud top height from the combined infrared and water vapor channels. Parameters such as cloud thickness and extinction properties are inferred by using the mid-wave infrared channel in combination with the visual one.

Table 1 compares some of the above stated modeling methods based on their various characteristics.

3 Rendering

For the sake of simplicity, cloud rendering can be assumed to consist of two essential components. The lighting captures the illumination or interaction of cloud particles with the surrounding impinging light. The rendering technique employs ray casting, rasterization or a variant to convert the existing shape representation, light settings, etc., to display the final cloud on the screen. In the following, we look into each of these two aspects separately.

3.1 Illumination

Clouds receive their illumination from the light coming from the sun, sky, ground and other cloud particles. An important phenomenon that illuminates both the atmosphere and the clouds is scattering. In a broad sense, scattering can be defined as the redirection of the incident light due to interactions with the molecules of the medium. The albedo for the clouds is close to 1, which implies that there is very little absorption of light. This leads to a high degree of re-emitting of received light by the cloud particles in the forward direction, commonly known as anisotropic scattering [69,81]. For a wide range of clouds, the optical effects occurring at the cloud–atmosphere interface are also key to obtaining the right effects.

Light transport equation The nature of scattering effect on a particle is dependent on the particle size, the refractive index and the wavelength of the visible light. For example, Rayleigh scattering is responsible for giving the sky its blue color due to scattering by its tiny particles. On the other hand, for larger particles such as those in the clouds, forward (Mie) scattering is predominant. The scattering phase function gives the angular distribution of light intensity scattered by a given particle for a given wavelength. The scattering equation combines these terms to capture the scattering behavior.

Considering the case of single scattering as shown in Fig. 5, the intensity of light reaching at P_a is given by

$$I_a(\lambda) = I_s(\lambda)\exp(-\tau(P_aP_b,\lambda)) + \int_{P_a}^{P_b} I_p(\lambda)\beta_\rho(l)F(\theta)\exp(-\tau(PP_b,\lambda))dl \quad (1)$$

Here, λ is the wavelength of the light, I_s is the intensity of sky light in the viewing direction, P_b is the other end of cloud falling in the view direction when connected to P_a, θ is the scattering angle, $F(\theta)$ is the scattering phase function, and τ is the optical depth obtained by integrating the attenuation coefficient (β_ρ) along the path, where the gathered density ρ is a function of the path length. The first term in the equation accounts for the attenuated light on the path of P_aP_b, whereas the second accounts for the scattered component [71]. In order to solve the multiple scattering phenomena, the incident intensity at point p (P in Fig. 5), I_p, has to be gathered for all major contributing directions. I_p is obtained as $I_p(\lambda) =$

Table 1 Summary of the characteristics of various modeling methods: if the method is shown to support physics on the modeled clouds and produces two- or three-dimensional cloud output and the prominent feature(s) representing the method

Method	Animation	2D/3D	Feature
[15]-2000	Micro-level	3D	Cellular automata
[87]-2003	Procedural	3D	Two-level modeling
[6]-2004	–	3D	Hierarchical blobs
[101]-2008	–	3D	Sketch-based interface
[22]-2009	–	2D	Satellite image clouds
[19]-2010	–	2D and 3D	Single-photo based
[107]-2014	–	3D	Cloud shape from images
[34]-2017	Macro-level	3D	Hypertextures
[99]-2018	Procedural	3D	Field functions, morphing

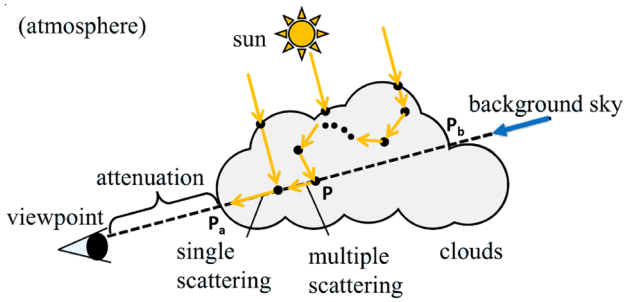


Fig. 5 Single and multiple scattering of the light, as it passes through the cloud before reaching the viewpoint [13]

$I_c(\lambda)\exp(-\tau(PP_c,\lambda))$, where I_c represents the attenuated light of the sun at the top of the atmosphere (point P_c).

The interaction of cloud with the sun light and its environment and hence its color is calculated using single scattering model into account in several works such as [17,18,94,102]. In such a model, the light is assumed to be scattered only once before it reaches the viewpoint (Fig. 5). While single scattering assumption simplifies the computation, in reality the light inside a cloud is scattered multiple times. Hence, a significant amount of research work has been dedicated to designing efficient rendering models and approximations to capture multiple scattering.

Other effects Apart from scattering induced within the cloud, work has also been done to reproduce other effects that become prominent in certain clouds or lighting situations. Max et al. [61] compute photorealistic rendering of clouds by using a diffusion approximation at the dense cloud core but by additionally accounting for multiple anisotropic scattering at the cloud borders where drop density is nearly zero. Two important lighting factors accounted for by [69] are as follows. Firstly, the rays crossing cloud corona are treated differently by applying Lambertian illumination as against the scattered illumination at the core. Secondly, concave regions of the clouds facing each other act as light traps and a major source of re-emission in these regions. The reader is also referred to [10] for more detailed exposition on various facets of radiative transfer and [80] for calculating light intensities in the presence of a participating media.

3.2 Rendering techniques

A majority of the methods in CG use volumetric representation of clouds for the purpose of rendering and illumination. A few of these techniques assume the volume to be composed of particles for the sake of simplification of the light computation. Apart from the representation, several variations have also been suggested for rendering. This includes variants of the traditional ray casting as well as rasterization that can also exploit the inbuilt hardware capability for a more efficient rendering. In the following, we review the existing methods based on their prominent rendering features.

Particle-based Blinn [4] is one of the earliest works in CG that proposes light interaction with matter such as clouds and dust. The medium is assumed to be uniform and composed of particles on which a brightness function of the form $B = w\mu\varphi(a)S$ is computed for each particle, which determines the amount of light escaping in the view direction. Here, w is the albedo controlling the levels of scattering from each particle (chosen to be low), μ is the cosine of the angle between the normal of the surface and observer vector, φ is the phase function, a is the angle between the observer and the light direction and S is the scattering probability. They have applied this model to illuminate the rings of Saturn and the cloud layer on a hypothetical planet.

Kajiya and Herzen [48] focus on a more generic problem to ray trace volume densities. They present an alternative to the Blinn scattering model by modeling multiple radiative scattering against particles with high albedo. The equation is solved differently for low and high albedo approximations and even expressed as spherical harmonics for the latter case. (The scattering model followed is the same as with Blinn.) Klassen [50] considers general sky illumination caused by

scattering of the light by suspended air particles without using cloud particles.

The generated clouds in [38] by Harris et al. are illuminated by adapting the multiple, forward scattering lighting model in [15]. However, instead of directly illuminating particles, a smooth 3D texture is constructed in every frame with the particles' attributes. Riley et al. [78] render single and multiple scattering atmospheric effects with aerial perspective. They assume the atmosphere to consist of two broad particle types, air and moisture and apply different phase functions to each to account for vast difference in their scattering properties.

Billboards Billboards are texture-mapped polygons which always face the viewer. In order to render the clouds from the cellular automata simulation in Dobashi et al. [15], a continuous density field expressed as metaballs is constructed at each cell by taking a weighted interpolation with all its neighboring cells. Billboards are placed at the centers of metaballs, sorted based on their distance to the sun or viewpoint and projected onto the image plane (Fig. 13a). The projection from the point of view of the light source provides light intensity reaching the clouds. On the other hand, the projection from the camera achieves blending cloud densities with the remaining scene. In their work, Dobashi et al. account for single scattering of light and considered sunlight, transmitted color from the sky and attenuation from the cloud particles to compute illumination. The light shafts are also rendered by scattering the sunlight passing through the cloud gaps. Yu and Wang [106] enhance the splatting of the modeled clouds by employing line integral convolution to form various types of convoluted and asymmetric textures, which are mapped on the billboards.

Radiosity-based The warped blobs in [91] also carry an intensity, a continuous field over which is constructed using the modified smoothing kernels. The intensity to and from the environment on the blobs is gathered as a result of a shooting process, similar to radiosity in principle. In order to compute the intensity field at any point in the environment, a ray is shot and the scattering equation is integrated along the path by determining the blobs that intersect with the ray. (Both single scattering and multiple scattering are handled.)

Stratiform clouds (with mostly horizontal development) are rendered by Bouthors et al. [7] (Fig. 6), by account for single, double, triple and above levels of scattering in addition to transparency. The effects reproduced by their method are labeled in Fig. 6, (a) diffuse reflectance, (b) glory, (c) fogbow, (d) pseudo-specular reflectance, (e) diffuse transmittance, (f) ground-clouds inter-reflection and (g) forward scattering. An altered Mie model is proposed for scattering with modified phase function and extinction parameters. Radiosity accounts for the light exchange between ground and the cloud.

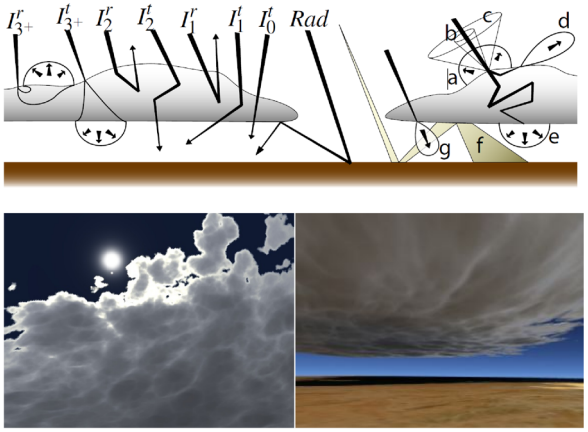


Fig. 6 Real-time illumination of the stratiform clouds by Bouthors et al. [7]

Specialized data structures Specific data structures have also been developed to capture one or more rendering effects efficiently. Miyazaki et al. [63] consider both the attenuation in the sunlight and viewing direction by dividing the volume data into *shadow-view slice (SVS)* to render the clouds (Fig. 7). The atmospheric density and the attenuation ratio are stored as a single texture, which is mapped to the SVS. In order to obtain the final intensity, the SVSs in the path of the viewpoint are blended.

Liao et al. [55] propose a simplified method for rendering dynamic clouds with the help of two new data structures. The shadow relation table (SRT) is a 2D grid structure that aids in determining the input illumination for every voxel. The *metaball lighting texture database* (MLTDB) aids in fast computation of the scattering parameters by maintaining a database of 32×32 projected metaball images with various densities and view angles. Like the previous methods, the first rendering pass entails illuminating each voxel in the octree, whereas in the second step, a back-to-front traversal of the octree is performed. While rendering in the second step, a billboard with the most suited texture from the MLTDB is selected for each node.

Bouthors et al. [8] propose an algorithm for interactive, multiple anisotropic light scattering computation in clouds. It approximates the light transport between an initial, receiving point on the flat surface and reaching at any point inside the volume using the concept of *collector area* (Fig. 10a). The collector area is the piece of the surface that receives a very large percentage of light from the source, which reaches the current point rendered. It is determined iteratively as a function of the rendered point and is inspired from the idea of the most probable paths [75]. The method is implemented on the GPU using shaders with several orders of Mie scattering (Fig. 10b).

Umenhoffer and Szirmay-Kalos [97] present a real-time method based on the particle hierarchies to render dynamic

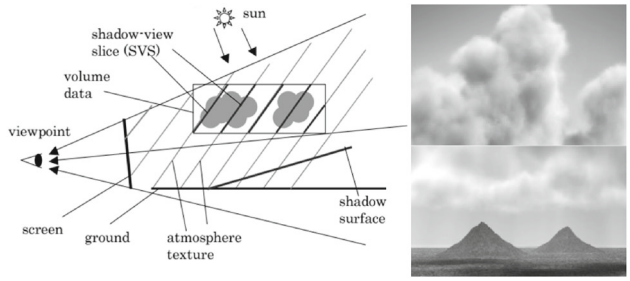


Fig. 7 Shadow-view slices and the associated rendering by Miyazaki et al. in [63]

Fig. 8 Cloud modeling and rendering generated by [71] with their multiple anisotropic scattering model

participating media under changing light conditions. A collection of particles determined from a given direction are grouped together as a *block*, the image of which replaces the particles. In the next step, a depth impostor is generated for each block, following which the particle blocks are rendered one by one. During this step, separate volume–light interactions with the particles are stored in the textures. Finally, the blocks are sorted for the viewing direction and rendered in back to front order in the rendering pass.

Max [58] used a variation of the shadow volume algorithm on segment blocks to calculate a simplistic shadowing and sunlight scattering effect on the clouds. In [26], Ebert and Parent combine scanline A-buffer rendering with volume rendering. Since their work dates to the pre-shader time, they create the A-buffer by determining all fragments for each pixel which are then used for volume rendering.

Precomputations In order to cope with the high runtime computational demand while rendering, several approaches store precomputed illumination components. In Nishita et al. [71], intensity of the first-order scattering of the sky light at each voxel is precomputed and stored based on the optical depth of that voxel from the cloud surface (Fig. 8). High-order forward, anisotropic scattering of a voxel within the cloud is captured by considering its form factor to its neighboring voxels and also stored. This step takes advantage of the fact that the scattering direction is narrow, and hence, only a few neighboring voxels influence a given voxel to this end.

Harris and Lastra [39] proposed a cloud shading algorithm for flight simulators wherein the rendering is split into two stages. The more expensive multiple forward scattering

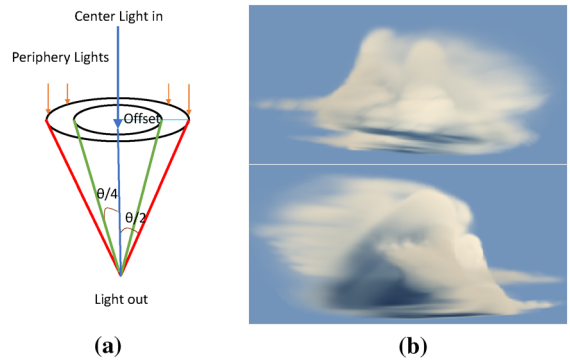


Fig. 9 **a** Cone-based regions identified for different illuminations for per-pixel transparency and **b** results produced from the storm data, by Riley et al. in [77]

is approximated in a preprocessing step, and the anisotropic scattering is done at runtime. The space is assumed to be filled with particles, and view-oriented textured polygons (*impostors*) are dynamically generated for efficient rendering. Wang [98] extends this texture splatting on particles to model about a dozen types of clouds with the help of artistic control. To this end, the particles are textured differently depending on the target cloud type.

The multiple scattering computation in [11] to render outdoor scenes with clouds and lightning is done in two steps. Firstly, both the direct and indirect intensities at basic light source points are precomputed and stored in a grid. Then, at runtime, the intensities are computed in real time by using the weighted sum of the basic intensities. The atmospheric scattering effects are precomputed and stored in [21], while an octree-based LOD provides cloud density to produce lightning scattering due to clouds.

The endless cloud animation in [46] is illuminated by sunlight and skylight with multiple scattering by precomputing, projecting onto spherical harmonics and storing the optical depth and transmittance for the basic volume data. The transmittance for each lighted voxel is clustered through sample points using a binary tree structure. The tree contains the sample points in the leaf nodes, together with the error and mean values in the interior nodes.

Rendering of cumulus clouds is accelerated through precomputations in [109] by storing quantities like the optical depth integral, single and multiple scattering in the reference particle for multiple camera positions and light directions. This information is utilized at runtime to compute the cloud lighting without volume ray casting (particle blending is used). A grid structure is employed to generate the particles, and for distant particles, a level-of-detail approximation obtains a coarser representation.

Volumetric Riley et al. [77] provide a method for weather-field data based on the particle scattering properties of the

constituent fields. The input data are translated to particle concentration, and volumetric transfer functions assign colors and opacities to these primitives generating a continuous field. The intensity attenuation or the optical depth inside the clouds is calculated as a function of the particle extinction and concentration. In their illumination model, light is broken into three categories assuming a scattering angle θ (Fig. 9). The shown central blue line represents the unscattered light. The other regions contributing to the forward scattering are the central (0 to $\theta/4$) and the peripheral ($\theta/4$ to $\theta/2$) regions, with their respective transparencies. Further, the cloud is assumed to be composed of water and ice particles and different phase functions are applied to each.

In their approach, Schpok et al. [87] achieve interactive rendering rates through slice-based volume rendering done using both CPU and GPU (shaders). The noise is divided into several octaves, which are stored as GPU textures. Transformation matrices corresponding to each octave are generated on the CPU. The user can tune parameters such as opacity and sharpness through a UI. Volumetric photon mapping is employed by Elek et al. [27] for the simulation of light transport in clouds at interactive rates on the GPU. Instead of rebuilding the photon map every time, temporal coherence present in the cloud illumination is exploited by storing a low resolution grid and upsampling it each frame by reshooting a much smaller fraction of photons in each frame. To account for the angular illumination information, the Henyey–Greenstein function is used.

Duarte and Gomes [24] first create a 3D mesh from the particles and then apply rasterization-based voxelization to the mesh, leading to the computation of a distance field and some noise parameters. These values are illuminated by ray-marching through the volume density and accounting for multiple scattering of light in the clouds. The clouds in "Puss in Boots" [62] are illuminated by dividing them into zones of first-order scattering (facing the light) and multiple-order scattering (facing away from the light) zones. An ambient occlusion pass is also added to account for visibility with the help of spherical harmonics. Qiu et al. [76] simulate light scattering in clouds via spherical harmonics and frequency domain volume rendering (Fig. 10).

Image-inspired Dobashi et al. [12] proposed to solve the *inverse rendering problem* which determines the rendering parameters to display realistic clouds in the synthetic images. This is done by estimating the parameters from real images of clouds taking into account the difference between the color histograms with that of the synthetic image. The set of parameters involved (intensity of sky, sun, etc.) are determined with the help of genetic algorithms. In order to obtain the best values, several iterations are performed for convergence. While performing the match between the synthetic and target image, a set of images with different rendering parameter settings

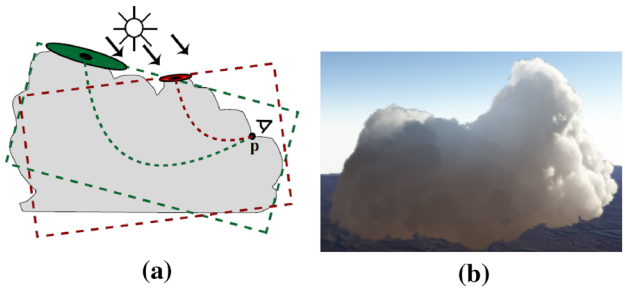


Fig. 10 Interactive GPU-based multiple anisotropic scattering in the clouds by Bouthors et al. [8] **a** collector area determined as a function of the point rendered **p** and **b** rendered cloud

Fig. 11 Multi-scattered cloud rendering from the radiance-predicting neural networks [49] method, combining Monte Carlo integration with the data-driven radiance predictions

are precomputed to reduce the computational time. Both single scattering and multiple scattering of light are taken into account for rendering the clouds.

Kallweit et al. [49] render clouds by using *radiance-predicting neural networks* (RPNN) together with Monte Carlo integration on the GPU (Fig. 11). The learning data consist of several cloud exemplars that help train on spatial and directional light distribution. In order to render a new scene, the descriptors are extracted from the visible points of the clouds and supplied as input to the trained neural network. The RPNN predicts the radiance function for the desired shading configurations. The descriptor is hierarchical and fed progressively to the neural network to enhance its ability to learn faster and predict better. The proposed method can achieve effects such as the cloud whiteness in the inner part, edge-darkening effects and silver lining in agreement with the reference image, with a few minutes of computation.

Table 2 compares the above discussed methods based on their various characteristics. Most of the modern approaches leverage GPU computational power in the form of shaders or CUDA, to perform one or more of the illumination subroutines. Some of these methods capture self-shadow of the cloud or the shadow cast by it on the ground. Furthermore, a number of these algorithms also account for multiple scat-

Table 2 Summary of the characteristics of various rendering methods: if the rendering method is CPU or GPU-based, handles single or multiple scattering and supports shadowing, modeling method used (T stands for texture, G for geometric and I for image, based on classification in Sect. 2)

Method	CPU/GPU	Scattering	Shadow	Modeling	Feature
[71]-1996	CPU	Multiple	–	Blobs (G)	Lookup table
[15]-2000	CPU	Single	Ground	Noise (T)	Shafts
[39]-2001	CPU	Multiple	Self	Particles (G)	Impostors
[77]-2003	GPU	Multiple	–	Particles (G)	Cone light bands
[38]-2003	GPU	Multiple	Self	Noise (T)	Flat 3D texture
[87]-2003	GPU	Single	Self	Noise (T)	Procedural
[63]-2004	GPU	Single	Ground	Volume (T)	Shadow-View Slice
[21]-2004	GPU	Multiple	–	Particles (G)	Diffusion
[55]-2004	GPU	Multiple	Self	Noise (T)	SRT, MLTDB
[97]-2005	GPU	Multiple	Self	Particles (G)	Hierarchical impostors
[7]-2006	GPU	Multiple	Ground	Noise (T)	Radiosity
[8]-2008	GPU	Multiple	Self	Noise (T)	Collector area
[27]-2012	GPU	Single	Self	Texture (T)	Photon mapping
[12]-2012	GPU	Multiple	–	Noise (T)	Inverse rendering parameter
[109]-2014	GPU	Multiple	Self	Particles (G)	High performance, precomputed
[49]-2017	GPU	Multiple	Self	Image (I)	RDNN
[99]-2018	GPU	Single	–	Noise (T)	Procedural

tering (anisotropic in most cases) of light within the cloud volume to compute the lighting.

A survey on optical models for direct volume rendering is laid out in Max [59], in which methods are classified based on phenomena like absorption, emission, scattering and shadows. Cerezo et al. have surveyed participating media techniques in [9]. Hufnagel and Held [43] have presented a survey on cloud lighting and rendering techniques. Ye has discussed cloud rendering methods in [104]. A brief overview of the discussed rendering methods together with some available tools and applications is also a part of the course [86]. [42] implements dynamic cloud rendering in the game engine Frostbite.

4 Animation

In many dynamic scenes, the requirement is not merely to model the clouds, but also to animate them. Cloud animation is a complex task, which entails dealing with the parcel and environment physics and setting the right initial conditions that leading to cloud formation. Whereas in atmospheric science, the goal is to produce a simulation as close to the ground reality as possible, in CG, we can benefit from the fact that the animation has to only appear reasonably realistic. This relaxation helps us design specific solutions that are both simplified and accelerated. There are two broad types of methods employed in CG to generate cloud animations: procedural and physics-based.

4.1 Procedural animation

Procedural methods give the illusion of animating clouds without the direct use of underlying physics. This is made possible by modeling the basic behavior of the physics field through noise, texture or obtaining cues from images, videos or other sources.

Noise-based Gardner [31] achieved animation of the modeled clouds by varying the modeled mathematical texturing parameters in textured plane/ellipsoids with time. Ebert and Parent [26] propose two different methods, surface- and volume-based for animating the modeled texture gases. The surface-based animation entails positioning planes at the scene boundaries with evolving parameters, which provide transparency and motion to the containing fluid. In order to animate three-dimensional volume, each point in the fluid is moved along a set path to generate swirling gases. Similarly, Max et al. [60] employ 3D textures to render clouds which are advected by the wind flow. Sakas and Westermann [83] present a functional approach for the visual simulation of cloud and fire where turbulence is generated using time-varying fractals.

Turbulent theory is applied in conjunction with the spectral analysis Sakas [82] to achieve animation. According to Reynolds, a turbulent flow can be formulated as a superposition of two motions, $u = \bar{U} + u'$, where $\bar{U}$ is the general translative velocity component and u' is the random fluctuative movement. In this work, Sakas obtained the large-scale wind movements to the clouds through $\bar{U}$ and turbulence is captured through u'. These components are evolved by ani-

mating spectral functions in the frequency domain, and the presented results demonstrate the technique on clouds as well as water vapor in the indoor settings.

Much like with cloud modeling, its evolution is handled by controlled variation of the density as a function of the implicit function and time in [25]. In [70], Neyret animates the procedural textures by advecting them as a fluid. This method relies on a low-resolution grid fluid simulation where density and texture are advected using the simulation. Similar to mip-mapping, the texture is decomposed into several layers to pick up the layers most suited to the local deformation. Schpok et al. [87] animate the obtained clouds with a two-level approach. While macro-level animation (keyframe, etc.) governs the general direction of the cloud movement, the micro-level handles effects like wisps straying away, cloud appearing or disappearing at the edges through pre-computed volumetric texture noise (Fig. 14b).

Grudziński and Dębowski [36] generate particle-based clouds with a probabilistic cloud generation function and a few global variables. The particles are assigned parameters like velocity, lifespan, maximum height, etc., as they are created, and this helps create desired trajectories that resemble a particular cloud type (stratus, cumulus, etc.). Kusumoto et al. [54] animate cumulus clouds on multiple prespecified target fields for keyframe animation. The simulation contains controlled heat sources to adjust the amount of heat generated on the ground, thereby influencing the buoyancy forces. The effect of wind is incorporated by shifting all the grid velocities by a user-specified wind velocity.

The procedurally modeled cloudscapes by Webank et al. [99] are also animated procedurally through the process of morphing between two selected models or density fields (Fig. 12). An optimal transport function helps to establish correspondences between the best pair matches in the source and target cloudscapes. The animated primitives are created from the interpolation of the identified pairs such that they follow the shortest trajectory.

Image-based The cues for procedural animation can also be extracted from images or videos. The motivation to convert texture to particles in [103] is to update the fluid mechanics properties of the regions through these particles, while animating Jupiter's atmosphere. The initial vorticity field on particles is seeded using a large black and white image of the planet with hand-marked vortex features, approximating the contribution of each vortex in the surrounding elliptical region. Thereafter, the particles are advected in two dimensions based on the barotropic model [41].

Dobashi et al. [17,18] animate satellite clouds based on user input flows of Bezier curves. The user input is estimated by observing a sequence of satellite images, and the modeled metaballs are advected based on this flow vector. The cloud motion accounts for moving, appearing and disappearing clouds and can produce 3D animated visualization of the clouds as seen from the space. In [105], the growth direction of the turrets in convective clouds is formulated with the help of the captured clouds videos.

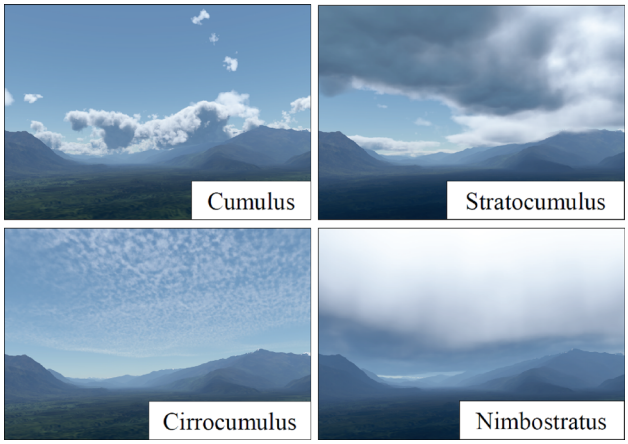


Fig. 12 A wide variety of clouds are modeled and animated procedurally by Webank et al. [99]

Jhou and Cheng [47] animate clouds present in a given image through automatic motion creation. Two parameters, cloudiness and cloud structure, are extracted as cues from the image and a content-aware wind field incorporating mean flow, and turbulence is generated to synthesize the cloud flow. In the animation obtained, however, the maximum variation between any two frames is limited.

Rule-based Dobashi et al. [15,16] for the first time employed cellular automata for the cloud growth and extinction. The simulation space is divided into a three-dimensional grid and each grid cell stores physical variables like humidity *(hum)*, cloud *(cld)* and phase transition *(act)* (allowed state 0 or 1 for each variable). Given the simulation state at time t, these variables are evolved for the next time step $t+1$ by employing simple Boolean transition rules on each grid cell. Their simulation and rendering method is combined in [5] to produce real-time results. The method is capable of creating clouds advected by the wind, and empty regions with extinguished clouds are reinitialized with new values. [15] additionally employs ellipsoids to control these physical properties and the cloud shapes and sizes (Fig. 13b).

4.2 Physics-based simulation

Stam [90] came up with the first unconditionally stable, grid-based model to produce complex fluid-like flows while still taking large time steps. This work has been the basis of a large number of grid-based fluid simulations in the last 2 decades, including for the clouds. Additionally, the scientific literature in meteorology and atmospheric science [84,85,89] has been

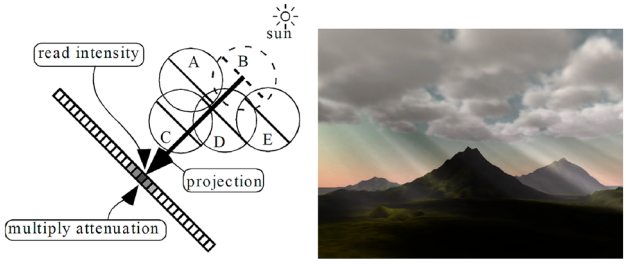


Fig. 13 Cellular automata by Dobashi et al. [15] to produce cloud animation **a** billboard projection, and **b** final rendering together with the light shafts

a key reference to the researchers working with clouds in the field of CG. In the following, we first touch upon the basics of the fluid and cloud physics and then review the relevant methods.

4.2.1 Fluid physics

Grid-based fluid solver The grid solver by Stam is based on the Navier–Stokes equation and evolution of the involved quantities in time (mass and momentum conservation, respectively, given by the following equations).

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

$$\frac{\partial(\mathbf{u})}{\partial(t)} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu\nabla^2 u + \mathbf{f} \tag{3}$$

Here, $\mathbf{u}$ represents the velocity field of the fluid, ν kinematic viscosity, ρ density, p pressure and f any external force. Given the state of simulation at time t, the state at $t+1$ is calculated by executing four steps. In the first step, external forces $\mathbf{f}$ like gravity are added to the field. The second step accounts for the advection of the fluid on velocity $(-\mathbf{u} \cdot \nabla)\mathbf{u}$. In the third step, the effect of the viscosity is accounted using the diffusion term $\nu\nabla^2 u$. The fourth step (projection) makes the field divergence free. Finally, the advection step is carried out, and to this end, each point of the field is traced backward in the time using a virtual particle. This step ensures unconditional stability with large time steps. Though the said model demonstrates simulation of smoke and was not tailored to work for clouds in its original form, it formed the basis of several semi-Lagrangian cloud simulations later. Furthermore, with some modifications these equations can also be applied for particle-based fluid simulations.

Cloud physics Cloud dynamics necessitates adding additional physics to a simple fluid solver that plays a key role in generating clouds. Air is a mixture of variety of gases, predominantly nitrogen (78%) and oxygen (21%) and a very small percentage of water vapor among other components. Hence, it obeys laws applicable to all the gases as given by the ideal gas equation. In context of the cloud physics, a commonly used term is *parcel* which refers to an imaginary body of air, to which the relevant thermodynamic quantities can be imparted. Some of the relevant concepts involved in the parcel physics are introduced below.

Humidity Absolute humidity is the concentration of water vapor in the air and is measured by the mass of water vapor divided by the mass of dry air in a volume of air at a given temperature. Relative humidity is the ratio of the current absolute humidity to the highest possible maximum humidity. At 100% relative humidity, the air is saturated and can no longer hold the water vapor, thus creating the possibility of rain. It is pertinent to observe that the amount of water vapor air can hold at any time dependent on the temperature of air. The saturated mixing ratio (ω_s) is the threshold beyond which the excess vapor present in the air condenses into liquid water $\omega_s = 621.97\frac{e_s}{P-e_s}$, where e_s is the saturated vapor pressure and P is the atmospheric pressure.

Temperature Temperature profile of the atmosphere and the ground where the air parcels originate are important factors governing the cloud formation. The atmospheric scientists introduced additional notions of temperature for the sake of convenience. The potential temperature $\Theta = T(\frac{P_0}{P})^{0.286}$ accounts for the pressure variation and the virtual temperature $T_v = T(1 + 0.61\omega)$ for the moisture variation of temperature, respectively. The virtual potential temperature combines both these effects $\Theta_v = \Theta(1 + 0.61\omega)$. It is interesting to note that the buoyant force which leads to vertical movement of the parcels is expressed as $\mathbf{f_{bouy}} \approx \frac{(\Theta_v^{\text{parcel}} - \Theta_v^{\text{air}})}{\Theta_v^{\text{air}}}\mathbf{g}$ per unit mass. The temperature lapse rate gives the rate of temperature change with the altitude. It has been observed that Θ_v first drops with height and then increases, due to which the parcels (water vapor) never leave the atmosphere.

Thermodynamics The water vapor releases heat upon condensation to liquid water (L_c, latent heat of condensation per unit mass) and absorbs heat during evaporation (L_e, latent heat of evaporation per unit mass). Specific heat of the air (C_p) is the amount of energy required to raise its temperature by one degree celsius. These physical quantities are required to compute the evaporation/condensation of water content inside the parcel and the change in its temperature thereof.

We refer the reader to [34,37] for more detailed exposition to the cloud physics.

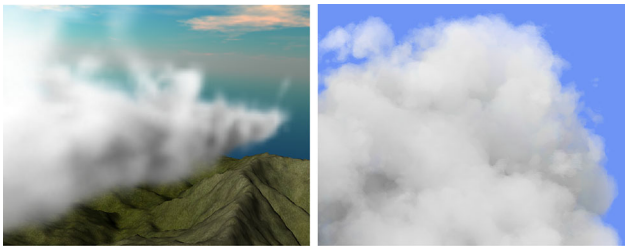


Fig. 14 **a** A frame of the GPU-based cloud simulation and rendering achieved by Harris et al. [38] and **b** developing cumulus cloud modeled with two-level modeling and rendering in Schpok et al. [87]

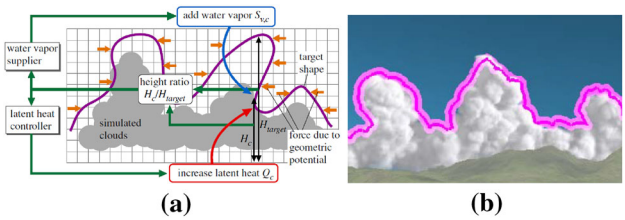


Fig. 15 Feedback control method based on the user specified contour and computational fluid dynamics, to simulate desired shapes of the cumuliform clouds by Dobashi et al. [14]

4.3 Methods

Based on the underlying model, most of the physical methods simulating clouds can be classified as either grid-based or particle-based.

Grid-based Grid-based methods and their variations were for long the only choice for simulating the micro-level cloud physics. A grid-based solver is proposed by Overby et al. in [72] for the interactive cloud simulation. Harris et al. [38] (Fig. 14a) implemented an interactive cloud simulation on the NVIDIA programmable graphics hardware. A 3D grid-based simulation is structured as layers of 2D textures for easy computation on the GPU, allowing gradual evolution of clouds in calm skies. Their semi-Lagrangian solver incorporates the aforementioned cloud physics and also accounts for the vorticity confinement.

Miyazaki et al. [67] used *coupled map lattice* (CML) to model and simulate various types of clouds based on the atmospheric fluid dynamics. CML is an extension of the cellular automata and uses a 3D grid and the traditional Navier–Stokes equations for simulation. The user specifies the type of cloud desired together with the initial and boundary conditions, grid resolution and other variables. All the variables are stored in the grid cells and are implemented through grid-based operations. Further, for generating certain types of clouds Bénard convection is additionally employed. Qiu et al. [76] also model clouds using CML. Miyazaki et al. in [64] improve their existing technique by a more direct inclusion of the heat equations (heat advection) and the phase transition in the grid solver. Their model also accounts for numerical dissipation by adding vorticity confinement, which is generated as a function of the velocity field.

Mizuno et al. [65] model volcanic clouds using modified physical laws that assume these clouds to be composed of two materials, magma and air. In this formulation, the evolution of the velocity field is given by Navier–Stokes equation, leading to the evolution of the mass of magma and entrained air separately. Mizuno et al. [66] simulate volcanic clouds by obtaining the pressure formulation in Navier–Stokes equations from coupled map lattice (CML) method. As against defining the mass evolution of matter over time, [65,66] formulate volcanic cloud density evolution as an equation.

In [20], Dobashi and Yamamoto develop a framework for animating clouds surrounding the earth as seen from space. The motion of clouds is controlled by physics based not just on the Navier–Stokes equations, but also on the cloud thermodynamics and the Coriolis force. The input to their physics is a pressure map supplied by the user, specifying regions of high and low atmospheric pressure on the planet's surface.

Dobashi et al. [14] present a method for guided cumuliform cloud formation, allowing the user to specify the desired cloud shape which is achieved by the underlying simulation (Fig. 15). Their physics is based on the existing approach of parcels rising, cooling and phase transition from vapor to liquid water together the aforementioned grid-based equations. The cloud shape is guided by projecting the user-specified contour in the simulation domain and determining its difference from the current cloud formation. This difference in addition to an artificial geometric potential force is fed back into the system to influence the parameters that control cloud growth.

In [46], Iwasaki et al. produce endless cloud animation from a limited volume data, inspired by the concept of creating an endless video from a limited input video. At runtime, the animation is created by stochastically selecting two similar volumes (one being the current frame) and cross-dissolving them. Stam and Fiume [91] have presented a model for animating smoke, fire and other gaseous phenomena with the help of the advection–diffusion equation on *blobs*. Each blob is characterized by its temperature, density and velocity in the simulation domain wherein the blobs change in size and shape (termed as "blob warping" by the authors) and is advected as a result of the diffusion process.

Particle-based Neyret [68] is one of the earliest works to obtain a simulation of convective clouds based on high-level physics variables. The parcels are modeled in 2D as particles (bubbles), which rise up due to the buoyancy force and become visible upon condensation. A bubble undergoes

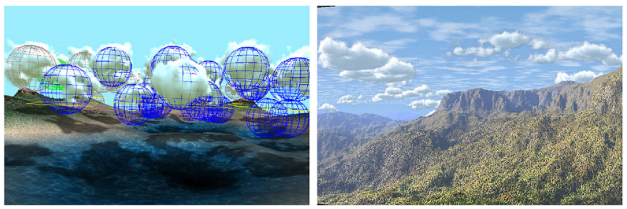
mixing dynamics with the surrounding atmosphere and its surrounding bubbles and also exerts attractive force toward these neighbors. Bubbles can be dynamically created or destroyed with the cloud evolution. Furthermore, vortices attached to the bubbles are responsible for enhancing small-scale shape and movement features on the clouds.

A particle-based cloud simulation on the GPU is proposed by Barbosa et al. in [2]. The particle physics is handled using position-based dynamics [56] and the cloud dynamics leverages the principles introduced in Sect. 4.2.1. The cloud detail is achieved by using adaptive particles, which can be merged or split in regions where clouds disappear or more details emerge, respectively. Additionally, the scalar fields such as temperature are smoothened out to reduce the value difference between the neighboring particles. Elhaddad et al. [28] also simulate clouds using particles but by approximating the force between these particles based on the Lennard–James potential.

Goswami and Neyret [33,34] proposed a hybrid, physics-driven procedural model for real-time cumulus cloud animation and rendering on the landscape scale (Fig. 16a). The atmosphere is represented implicitly through curves that provide temperature and humidity values as a function of the altitude. The cost of physics is significantly reduced by carrying out it at the macro-level, on a few large parcels on the CPU itself. The parcel–parcel interaction is handled through smoothed particle hydrodynamics (SPH) forces, and a drag force accounts for the friction with the surrounding atmosphere. The parcels also exchange mass with the implicit environment owing to the processes of entrainment and detrainment. The method also allows the user to experiment on the atmospheric profile and select dewpoint altitude, saturation ratio, etc.

The original idea of hybrid macro-physics and micro-amplification is refined in [32] by incorporating a *cloud map* for governing cloud shapes and coverage over the landscape (Fig. 16b). The physics computation is limited to a few parcels within a unit octree. However, instead of generating hypertexture directly inside each of the parcels, the parcel physics attributes are projected onto a texture. In addition, a cloud map containing a precomputed texture of noise is employed. Both these textures are combined at runtime to generate the animated cloud cover through volume ray casting in the shader. The reported benefits of this approach are higher frame rates, more realistic cloud shapes and better span, especially toward the horizon.

Duarte and Gomes [24] simulate real-time cumulus clouds based on the dataSkewT/LogP diagrams (Fig. 17b). Instead of determining the required physical parameters or setting initial conditions for the simulation, this information is obtained from sounding data. The data provide the buoyant force to solve the equation of motion of the air parcels without solving differential equations. Authors [24,32,34] follow the evolution of the parcels from their birth to death. Whereas in [32,34], most of this process happens as a natural part of the simulation, Duarte and Gomes [24] ensure it through explicit steps.



(a) (b)

Fig. 16 **a** Hybrid physics inspired procedural animation and rendering in [34] and **b** cloud map replacing direct volume amplification in parcels in [32] (overall application running at 200 fps)

(a) (b)

Fig. 17 **a** High-performance rendering of cumulus clouds by Yusov [109] achieved with the use of precomputed reference particle and **b** cloud systems generated from sounding data by [24]

Dobashi et al. have reviewed their various simulation and rendering approaches in [13]. One or more of the cloud modeling, rendering and simulations aspects is laid out in [51,52,79]. Table 3 summarizes the above discussed animation methods on their various features.

5 Performance

In this section, we review some of the above stated methods on their performance. Whereas the hardware capability in the early days did not allow for high-performance computations, in the recent years, more dedicated techniques targeting interactive simulation and rendering have been designed.

1980–2004 With the cellular automata method, Dobashi et al. [16] reported a computation time of 0.19 s per time step for simulation on SGI Indigo2 R10000 195 MHz. The CML system in [67] allowed the user to model the clouds interactively but is intended to produce the images and not for real-time performance. For a grid size of 250 × 250 × 5, 2 s simulation and 30 s rendering time are reported on Pentium III. The rendering speedup in Wang [98] is given to be 100× for their improved model over [39] by adjusting the transparency of the sprites. For a dense cloud coverage, their method takes 10 ms on a 733 MHz machine.

Table 3 Summary of the characteristics of various animation methods: nature of the animation technique, if the animation is executed on CPU/GPU and the prominent feature of the method

Method	Animation	CPU/GPU	Feature
[82]-1993	Procedural	CPU	Spectral synthesis
[90]-1999	Grid-based	CPU	Unconditionally stable
[15]-2000	Procedural	CPU	Cellular automata
[70]-2003	Procedural	CPU	Procedural textures
[38]-2003	Grid-based	GPU	GPU textures, vorticity confinement
[65]-2003	Grid-based	CPU	Coupled map lattice, volcanic clouds
[14]-2008	Grid-based	CPU	Cumuliform clouds, feedback control
[54]-2012	Grid-based	CPU	Keyframe control, target field
[2]-2015	Particle-based	GPU	Position-based fluid
[28]-2016	Particle-based	CPU	Lennard–James potential
[34]-2017	Particle-based	CPU/GPU	Macro-physics, micro-rendering
[24]-2017	Particle-based	CPU	SkewT/LogP diagrams
[99]-2018	Procedural	CPU	Key-frame interpolation, morphing
[32]-2019	Particle-based	CPU/GPU	Cloud map, macro-physics

Table 4 Various presented methods (2012–2019) compared for their modeling, rendering and animation times (T_m, T_r and T_a, respectively)

Method	Model size (g/p)	T_m	T_r	T_a	Feature
[12]-2012	200 × 200 × 200/–	–	30 s	–	Image-based (GPU)
[107]-2014	100 × 100 × 200/80 K	5 min	–	–	Image-based (CPU)
[109]-2014	–/10 K	–	36 ms	–	Precomputed lighting (GPU)
[2]-2015	–/840 K	–	–	1219 ms	Position-based fluids (GPU)
[28]-2016	–/49K	–	1173 ms	–	Lennard–James potential (CPU)
[34]-2017	–/82	–	42 ms/fr	4.3 ms/fr	Macro-physics (CPU), micro-rendering (GPU)
[24]-2017	–/250K	–	–	44 ms/fr	SkewT/LogP diagram (CPU)
[49]-2017	1200 × 1200 × 1200/–	–	9 min	–	Deep scattering, RPNN (GPU)
[99]-2018	–	–	10 s/fr	30 min	Key-frame interpolation, morphing (GPU)
[32]-2019	–/12	–	3.3 ms/fr	1.6 ms/fr	Cloud map, macro-physics

Some of the times are listed per frame (/fr). Model size is given in the sampling grid resolution and/or the particle count (g/p) used

Several of the approaches developed from 2003 onward utilized the shaders on GPU for the rendering purpose. Riley et al. [77] implement their visualization system on an Nvidia GeForce FX 5800 Ultra with Cg and achieve a frame rate of about 1–5 fps on the selected dataset. Harris et al. [38] on an Nvidia GeForce FX Ultra card for the grid size 64^3 update physics 1–5 times each frame and with the forward scattering can achieve a frame rate of around 30 fps for some cases. The application without physics by Schpok et al. [87] can run at a frame rates between 5 and 30 fps on a Pentium IV processor on an Nvidia GeForce4 Ti4600. The dynamic cloud rendering with the specialized data structures (scattering, self-shadowing and transparency) in [55] is capable of running at more than 30 fps for selected parameters setting on a 2.2 GHz AMD Athlon machine with an Nvidia GeForce 5600 graphics card.

2005–2011 The implementation of the stratiform clouds in [7] with shaders is done on a PC with an Nvidia Quadro FX 1400. Together with the multiple anisotropic scattering and other captured effects, they achieve about 18 fps. The computation time to animate satellite clouds in [20] took about 0.1 s for each time step on a desktop PC with a Pentium IV 3.6 GHz and a grid size 160 × 80 × 4. The collector area algorithm by [8] is tested on a Pentium 4 at 1.86 GHz with an Nvidia 8800 GTS graphics board for multiple anisotropic scattering. For a cumulus cloud with 5K triangles and 512^3 hypertexture, a frame rate of 2 per second is reported allowing interactive rendering.

The cellular automata simulation in [102] is implemented on an Nvidia GeForce 8600 GTX using Cg, and for a simulation volume of 64 × 64 × 64, a maximum frame rate of 21.3(19.6) is achieved without(with) illumination. Dobashi et al. [22] use a PC with an Intel Core 2 Quad Extreme Q6800 (2.93 GHz) and an Nvidia GeForce 8800 Ultra for visualization of the earth-scale clouds. The original cloud data dimensions 16,384 × 8192 × 32 occupied 4 GB memory and needed out-of-core processing. The rendering rates achieved

are 5–30 fps. The endless cloud animation and associated rendering [46] are tested on a PC with an Intel Core i7 X-980 CPU and an Nvidia GeForce GTX (implemented on CUDA). For a volume size of $128^2 \times 64$ voxels, the frame rates of up to 26 fps are reported. During this period, some techniques also leveraged CUDA apart from the shaders and benefitted from the rapidly growing parallel computation capabilities of the GPU.

2012–2019 In the problem of estimating rendering parameters from the image, Dobashi [12] reports that after a time of 10–20 min of preprocessing, the optimal parameter estimation takes less than a minute. Yuan et al. [107] generate the cloud structure from an input image on an Intel(R) Core(TM) i5-2300 CPU 2.80 GHz, 4 GB of RAM in about 5 min (80,000 sampling particles produced). On an Intel Core i7 CPU and an Nvidia GeForce GTX 680 GPU, precomputed lighting of cumulus clouds in [109] took about 15 ms to render a high quality image (Fig. 17a).

Most of the physics-based, real-time cloud simulation methods were developed during this period. Cloud animation based on position based fluids [2] on an Intel Core i7-4770k and an Nvidia GeForce GTX 780 Ti GPU took around 736 ms for a single iteration in simulation for 540 K particles, with the adaptive particle merging and splitting feature turned on. On an Intel i7-4790K CPU, Lennard–Jonnes force approximation [28] took 450 ms to compute a single frame consisting of 500 particles per cell. Goswami and Neyret [34] report achieving a frame rate of 21 per seconds for up to 85 physics parcels (including simulation and rendering) on an Intel Core 3.4 GHz machine and an Nvidia GeForce GTX 970 GPU. On a similar machine, [32] reaches a peak performance of 200 fps supporting both physics and visualization using cloud maps. Duarte and Gomes [24] report consuming about 52 ms to process single cloud dynamic step of the sounding data with the wind when using 250 K particles on the CPU.

The procedural cloudscapes [99] on an Intel Core i7 with an Nvidia GTX 970 took 10 s to render a single frame after the additional computation time of cloud shapes and morphing process. Another interesting development during this period is the use of artificial intelligence for cloud generation [49]. On Nvidia Titan, a time range of few seconds to few minutes is reported to synthesize images incorporating high-order scattering inspired by the original cloud images. In Table 4, modeling, rendering and animation times for the various techniques published during 2012–2019 are listed.

6 Conclusion

A variety of cloud shapes based on various requirements can be obtained through various cloud modeling approaches in CG. Apart from the shapes and volumes, images have served as a useful inspiration to generate rich cloud structures. Furthermore, this step can also be delegated to technical artists, for example, as done in the movie production. Cloud rendering on the other hand is a more complex task that involves interaction of the light from the sun, sky and the ground with the cloud particles to reproduce the right effect. Clouds display a great deal of forward scattering of the incident light, which makes capturing light transport a volumetric and view-dependent phenomena. It is impossible and impractical to track all the light paths. Hence, all the methods optimize the light transport by limiting the number of light bounces through the cloud and also by designing the right phase functions and making other suitable approximations. The rendering quality and efficiency have been the primary focus of a bulk of the developed research methods on this front. To this end, approaches have been designed that leverage GPU computational power, preprocessing and specialized data structures.

Various clouds differ in their appearance and have distinct features (silver lining, lit/unlit side, fogbow) that need special treatment. Some techniques have worked toward reproducing these special effects. More recently, another emerging trend is the use of the artificial intelligence to aid in rendering computations. More methods are taking advantage of the available information in images and videos to model and render clouds. Machine learning methods can obtain clues from images that can help identify the desired parameters for realistic cloud rendering and to some extent are capable of automating this task.

The simulation methods in the computational fluid dynamics are computationally intensive and carry minute details, which are often unnecessary in CG. In the earlier work, due to the limitation of limited hardware memory, many methods could simulate only limited volumes of clouds. This is further exacerbated by the fact that even though most of the sky is filled with the invisible air and nearly no clouds, it needs to be modeled for the simulation. This created a bottleneck both in terms of memory and time efficiency. The use of adaptive grids can partly solve this problem [93]. Eulerian methods also have some other issues like numerical dissipation, resolution, etc. Efforts have also been made to design purely procedural approaches that can convey the look of animated clouds without physics computations; the application of such approaches, however, has been limited.

To cope with this, researchers have looked into alternate techniques to simulate the cloud physics. This includes particle-based methods, hybrid physics-driven procedural and sounding data-based approaches. While the grid-based approaches have traditionally been more common and a natural choice to simulate the clouds, recently, several methods have successfully demonstrated the strength of particle-based approaches too. However, most simulations still need to set the right initial conditions, state of the atmosphere, bound-

ary regions, etc. Recently, methods have been designed that can achieve interactive frame rates on the GPU while simulating and rendering clouds over the landscape. The existing methods can further explore the possibility of using artificial intelligence to animate clouds, which could be a promising direction for future research.

Acknowledgements Open access funding provided by Blekinge Institute of Technology.

References

1. Alldieck, T., Lundtoft, D.H., Montanari, N., Nikolov, I., Vlaykov, I.G., Madsen, C.B.: Modelling of Clouds from a Hemispherical Image, Computer Graphics and Visual Computing (CGVC). The Eurographics Association (2014)
2. Barbosa, C.W.F., Dobashi, Y., Yamamoto, T.: Adaptive cloud simulation using position based fluids. Comput. Anim. Virtual Worlds **26**(3–4), 367–375 (2015)
3. Bi, S., Bi, S., Zeng, X., Lu, Y., Zhou, H.: 3-Dimensional modeling and simulation of the cloud based on cellular automata and particle system. ISPRS Int. J. Geo-Inf. **5**, 86 (2016)
4. Blinn, J.F.: Light reflection functions for simulation of clouds and dusty surfaces. ACM SIGGRAPH Comput. Graph. **16**(3), 21–29 (1982)
5. Bo, Q., Tao, L.V.: Real-time dynamic cloud modeling and rendering. In: International Conference on Computer Graphics, Imaging and Visualization CGIV, pp. 285–290 (2005)
6. Bouthors, A., Neyret, F.: Modeling Clouds Shape, Eurographics (short papers) (2004)
7. Bouthors, A., Neyret, F., Lefebvre, S.: Real-time realistic illumination and shading of stratiform clouds. In: Eurographics Workshop on Natural Phenomena, pp. 41–50 (2006)
8. Bouthors, A., Neyret, F., Max, N., Bruneton, E., Crassin, C.: Interactive multiple anisotropic scattering in clouds. In: Proceedings of the ACM Symposium on Interactive 3D Graphics and Games, pp. 173–182 (2008)
9. Cerezo, E., Cazorla, F.P., Pueyo, X., Seron, F., Sillion, F.X.: A survey on participating media rendering techniques. Vis. Comput. **21**(5), 303–328 (2005)
10. Chandrasekhar, S.: Radiative Transfer. Dover Books on Intermediate and Advanced Mathematics, New York (1960)
11. Dobashi, Y., Enjyo, Y., Yamamoto, T., Nishita, T.: A fast rendering method for clouds illuminated by lightning taking into account multiple scattering. Vis. Comput. **23**(9), 697–705 (2007)
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)
13. Dobashi, Y., Iwasaki, K., Yue, Y., Nishita, T.: Visual simulation of clouds. Vis. Inf. **1**, 1–8 (2017)
14. Dobashi, Y., Kusumoto, K., Nishita, T., Yamamoto, T.: Feedback control of cumuliform cloud formation based on computational fluid dynamics. ACM Trans. Graph. **27**(3), 1–8 (2008)
15. Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T., Nishita, T.: A simple, efficient method for realistic animation of clouds. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pp. 19–28 (2000)
16. Dobashi, Y., Nishita, T., Okita, T.: Animation of clouds using cellular automaton. Proc. Comput. Graph. Imaging **98**, 251–256 (1998)
17. Dobashi, Y., Nishita, T., Yamashita, H., Okita, T.: Modeling of clouds from satellite images using metaballs. In: Proceedings Pacific Graphics, Sixth Pacific Conference on Computer Graphics and Applications, pp. 53–60 (1998)
18. Dobashi, Y., Nishita, T., Yamashita, H., Okita, T.: Using metaballs to modeling and animate clouds from satellite images. Vis. Comput. **15**(9), 471–482 (1999)
19. Dobashi, Y., Shinzo, Y., Yamamoto, T.: Modeling of clouds from a single photograph. Comput. Graph. Forum **29**(7), 2083–2090 (2010)
20. Dobashi, Y., Yamamoto, T.: A controllable method for animation of earth-scale clouds. In: Proceedings of CASA (2006)
21. Dobashi, Y., Yamamoto, T., Nishita, T.: Efficient rendering of lightning taking into account scattering effects due to clouds and atmospheric particles. In: Proceedings Ninth Pacific Conference on Computer Graphics and Applications, Pacific Graphics, pp. 390–399 (2001)
22. Dobashi, Y., Yamamoto, T., Nishita, T.: Interactive and realistic visualization system for earth-scale clouds. Poster paper, Pacific Graphics (2009)
23. Dobashi, Y., Yamamoto, T., Nishita, T.: An interactive rendering system using hierarchical data structure for earth-scale clouds. Sci. China Inf. Sci. **53**(5), 920–931 (2010)
24. Duarte, R.P.M., Gomes, A.J.P.: Real-time simulation of cumulus clouds through skewT/LogP diagrams. Comput. Graph. **67**(C), 103–114 (2017)
25. Ebert, D.S.: Volumetric modeling with implicit functions: a cloud is born. In: ACM SIGGRAPH Visual Proceedings: The Art and Interdisciplinary Programs of SIGGRAPH, vol. **147** (1997)
26. Ebert, D.S., Parent, R.E.: Rendering and animation of gaseous phenomena by combining fast volume and scanline A—buffer techniques. ACM SIGGRAPH Comput. Graph. **24**(4), 357–366 (1990)
27. Elek, O., Ritschel, T., Wilkie, A., Seidel, H.P.: Interactive cloud rendering using temporally-coherent photon mapping. In: Proceedings of Graphics Interface, pp. 141–148. Canadian Information Processing Society (2012)
28. Elhaddad, A., Elhaddad, F., Sheng, B., Zhang, S., Sun, H., Wu, E.: Real-time cloud simulation using Lennard–Jones approximation. In: Proceedings of the 29th International Conference on Computer Animation and Social Agents CASA, pp. 131–137 (2016)
29. Elinas, P., Stuerzlinger, W.: Real-time rendering of 3D clouds. J. Graph. Tools **5**(4), 33–45 (2000)
30. Fan, X.L., Zhang, L.M., Zhang, B.Q., Zhang, Y.: Real-time rendering of dynamic clouds. J. Comput. (Taiwan) **25**, 11–21 (2014)
31. Gardner, G.Y.: Visual simulation of clouds. ACM SIGGRAPH Comput. Graph. **19**(3), 297–304 (1985)
32. Goswami, P.: Interactive animation of single-layer cumulus clouds using cloud map. In: Smart Tools and Apps for Graphics—Eurographics Italian Chapter Conference (2019)

33. Goswami, P., Neyret, F.: Real-time landscape-size convective clouds simulation. In: ACM Proceedings of the 19th Symposium on Interactive 3D Graphics and Games, p. 135 (2015)
34. Goswami, P., Neyret, F.: Real-time landscape-size convective clouds simulation and rendering. In: Workshop on Virtual Reality Interaction and Physical Simulation. The Eurographics Association (2017)
35. Griffith, E.J.: Visualizing cumulus clouds in virtual reality. Ph.D. Thesis, Technical University of Delft (2010)
36. Grudziński, J., Dębowski, A.: Clouds and atmospheric phenomena simulation in real-time 3D graphics. In: Computer Vision/Computer Graphics Collaboration Techniques: 4th International Conference, MIRAGE. Rocquencourt, pp. 117–127 (2009)
37. Harris, M.J.: Real-Time Cloud Simulation and Rendering. Ph.D. Thesis, UNC (2003)
38. 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, pp. 92–101 (2003)
39. Harris, M.J., Lastra, A.: Real-time cloud rendering. Comput. Graph. Forum **20**(3), 76–85 (2001)
40. Hasan, M.M., Karim, M.S., Ahmed, E.: Generating and rendering procedural clouds in real time on programmable 3D graphics hardware. Pakistan Section Multitopic Conference, pp. 1–6 (2005)
41. Holton, J.R., Hakim, G.J.: An Introduction to Dynamic Meteorology. Academic Press, London (1972)
42. Högfeldt, R.: Convincing cloud rendering: an implementation of real-time dynamic volumetric clouds in frostbite. M.Sc. Thesis, Chalmers, University of Gothenberg (2016)
43. Hufnagel, R., Held, M.: A survey of cloud lighting and rendering techniques. J. WSCG **20**, 205–212 (2012)
44. Hufnagel, R., Held, M., Schröder, F.: Large-scale, realistic cloud visualization based on weather forecast data. In: Proceedings of the Ninth IASTED International Conference on Computer Graphics and Imaging CGIM, pp. 54–59 (2007)
45. Igawa, N., Koga, Y., Matsuzawa, T., Nakamura, H.: Models of sky radiance distribution and sky luminance distribution. Sol. Energy **77**(2), 137–157 (2004)
46. Iwasaki, K., Nishino, T., Dobashi, Y.: Real-time rendering of endless cloud animation. Short paper, Pacific Graphics (2011)
47. Jhou, W.C., Cheng, W.H.: Animating still landscape photographs through cloud motion creation. IEEE Trans. Multimed. **18**(1), 4–13 (2016)
48. Kajiya, J.T., Herzen, B.P.V.: Ray tracing volume densities. ACM SIGGRAPH Comput. Graph. **18**(3), 165–174 (1984)
49. Kallweit, S., Müller, T., Mcwilliams, B., Gross, M., Novák, J.: Deep scattering: rendering atmospheric clouds with radiance-predicting neural networks. ACM Trans. Graph. **36**(6), Article No. 231 (2017)
50. Klassen, V.R.: Modeling the effect of the atmosphere on light. ACM Trans. Graph. **6**(3), 215–237 (1987)
51. Koehler, O.: Interactive simulation of clouds based on fluid dynamics. M.Sc. Thesis (2009)
52. Kol, T.R.: Real-time cloud rendering on the GPU. M.Sc. Thesis, Utrecht University (2013)
53. Kowsuwan, N., Kanongchaiyos, P.: 3D cloud animation using CA based method. In: International Symposium on Intelligent Signal Processing and Communication Systems ISPACS, pp. 387–392 (2009)
54. Kusumoto, K., Dobashi, Y., Yamamoto, T.: Keyframe control of cumulus cloud simulation, Art. 4, ACM SIGGRAPH Asia Posters (2012)
55. Liao, H.S., Ho, T.C., Chuang, J.H.: Efficient rendering of dynamic clouds. In: Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry VRCAI, pp. 19–25 (2004)
56. Macklin, M., Müller, M.: Position based fluids. ACM Trans. Graph. **32**(4), Article No. 104 (2013)
57. Matsuoka, D.: Extraction, classification and visualization of 3-dimensional clouds simulated by cloud-resolving atmospheric model. Int. J. Model. Simul. Sci. Comput. **8**, 1750071 (2017)
58. Max, N.L.: Atmospheric illumination and shadows. SIGGRAPH Comput. Graph. **20**(4), 117–124 (1986)
59. Max, N.L.: Optical models for direct volume rendering. IEEE Trans. Vis. Comput. Graph. **1**(2), 99–108 (1995)
60. Max, N.L., Crawfis, R., Williams, D.: Visualizing wind velocities by advecting cloud textures. In: Proceedings Visualization, pp. 179–184 (1992)
61. Max, N.L., Schussman, G., Miyazaki, R., Iwasaki, K.: Diffusion and multiple anisotropic scattering for global illumination in clouds. J. WSCG **12**(2), 277–284 (2004)
62. Miller, B., Museth, K., Penney, D., Zafar, N.B: Cloud modeling and rendering for "Puss In Boots". ACM SIGGRAPH Talks (2012)
63. Miyazaki, R., Dobashi, Y., Nishita, T.: A fast rendering method of clouds using shadow-view slices (2004)
64. Miyazaki, R., Dobashi, Y., Nishita, T.: Simulation of cumuliform clouds based on computational fluid dynamics. Eurographics 2002 Short Paper. Eurographics Association (2002)
65. Mizuno, R., Dobashi, Y., Chen, B.Y., Nishita, T.: Physics motivated modeling of volcanic clouds as a two fluids model. In: Proceedings 11th Pacific Conference on Computer Graphics and Applications, pp. 440–444 (2003)
66. Mizuno, R., Dobashi, Y., Nishita, T.: Modeling of volcanic clouds using CML. J. Inf. Sci. Eng. **20**, 219–232 (2004)
67. Miyazaki, R., Yoshida, S., Dobashi, Y., Nishita, T.: A method for modeling clouds based on atmospheric fluid dynamics. In: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications, pp. 363–372 (2001)
68. Neyret, F.: Qualitative simulation of cloud formation and evolution. In: Eurographics Workshop on Computer Animation and Simulation, pp. 113–124 (1997)
69. Neyret, F.: A phenomenological shader for the rendering of cumulus clouds. Research Report RR-3947, INRIA (2000)
70. Neyret, F.: Advected textures. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 147–153 (2003)
71. Nishita, T., Dobashi, Y., Nakamae, E.: Display of clouds taking into account multiple anisotropic scattering and sky light. In: ACM Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 379–386 (1996)
72. Overby, D., Melek, Z., Keyser, J.: Interactive physically-based cloud simulation. In: Proceedings of 10th Pacific Conference on Computer Graphics and Applications, pp. 469–470 (2002)
73. Peng, K.C., Chen, T.: Incorporating cloud distribution in sky representation. In: IEEE International Conference on Computer Vision, pp. 2152–2159 (2013)
74. Perlin, K., Hoffert, E.M.: Hypertexture. ACM SIGGRAPH Comput. Graph. **23**(3), 253–262 (1989)
75. Premože, S., Ashikhmin, M., Shirley, P.: Path integration for light transport in volumes. In: Proceedings of the 14th Eurographics Workshop on Rendering, Eurographics Association, pp. 52–63 (2003)
76. Qiu, H., Chen, L.T., Qiu, G.P., Yang, H.: Realistic simulation of 3D cloud. WSEAS Trans. Comput. **12**, 331–340 (2013)
77. Riley, K., Ebert, D.S., Hansen, C., Levit, J.: Visually accurate multi-field weather visualization. In: Proceedings of the 14th IEEE Visualization VIS'03, pp. 37–44 (2003)
78. Riley, K., Ebery, D.S., Kraus, M., Tessendorf, J., Hansen, C.: Efficient rendering of atmospheric phenomena. In: Proceedings of

the Fifteenth Eurographics Conference on Rendering Techniques, pp. 375–386 (2004)
79. Rosa, M.: Large-scale cloudscapes using noise. M.Sc. Thesis, Rochester Institute of Technology (2013)
80. Rushmeier, H.E., Torrance, K.E.: The zonal method for calculating light intensities in the presence of a participating medium. ACM SIGGRAPH Comput. Graph. **21**(4), 293–302 (1987)
81. Sakas, G., Müller, S., Shirley, P.: Photorealistic Rendering Techniques. Springer, Berlin (2001)
82. Sakas, G.: Modeling and animating turbulent gaseous phenomena using spectral synthesis. Vis. Comput. **9**(4), 200–212 (1993)
83. Sakas, G., Westermann, R.: A functional approach to the visual simulation of gaseous turbulence. Comput. Graph. Forum **11**(3), 107–117 (1992)
84. Schlesinger, R.E.: A three-dimensional numerical model of an isolated deep convective cloud: preliminary results. J. Atmos. Sci. **32**(5), 934–957 (1975)
85. Schlesinger, R.E.: A three-dimensional numerical model of an isolated thunderstorm: part I. Comparative experiments for variable ambient wind shear. J. Atmos. Sci. **35**(4), 690–713 (1978)
86. Schneider, A., Vos, N.: The real-time volumetric cloudscapes of horizon zero dawn. In: Advances in Real-time Rendering, SIGGRAPH (2015)
87. Schpok, J., Simons, J., Ebert, D.S., Hansen, C.: A real-time cloud modeling, rendering, and animation system. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 160–166 (2003)
88. Sethi, M.: Generating clouds in computer graphics. M.Sc. Thesis, McGill University (1997)
89. Soares, P.M.M., Miranda, P.M.A., Siebesma, A.P., Teixeira, J.: An eddy-diffusivity/mass-flux parametrization for dry and shallow cumulus convection. Q. J. R. Meteorol. Soc. **130**(604), 3365–3383 (2004)
90. Stam, J.: Stable fluids. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH, pp. 121–128 (1999)
91. Stam, J., Fiume, E.: Depicting fire and other gaseous phenomena using diffusion processes. In: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH, pp. 129–136 (1995)
92. Stiver, M., Baker, A., Runions, A., Samavati, F.: Sketch based volumetric clouds. In: Proceedings of the 10th International Conference on Smart Graphics, pp. 1–12 (2010)
93. Suzuki, K., Dobashi, Y., Yamamoto, T., Dobashi, Y., Kaji, S., Iwasaki, K.: An efficient cloud simulation with adaptive grid structure. In: Mathematical Insights into Advanced Computer Graphics Techniques MEIS, vol. 32, pp. 109–118. Springer, Singapore (2019)
94. Tomoyuki, N., Dobashi, Y.: Modeling and rendering methods of clouds. In: Proceedings Seventh Pacific Conference on Computer Graphics and Applications, pp. 218–219 (1999)
95. Trembilski, A., Broßler, A.: Surface-based efficient cloud visualisation for animation applications. In: J. WSCG, pp. 453–460 (2002)
96. Trembilski, A., Broßler, A.: Transparency for polygon based cloud rendering. In: Proceedings of the ACM Symposium on Applied Computing, pp. 785–790 (2002)
97. Ummenhoffer, T., Szirmay-Kalos, L.: Real-time rendering of cloudy natural phenomena with hierarchical depth impostors. EG Short Presentations, The Eurographics Association (2005)
98. Wang, N.: Realistic and fast cloud rendering in computer games. In: ACM SIGGRAPH 2003 Sketches and Applications (2003)
99. Webanck, A., Cortial, Y., Guérin, E., Galin, E.: Procedural cloudscapes. Comput. Graph. Forum **37**(2), 431–442 (2018)
100. Wenke, W., Yumeng, G., Min, X., Sikun, L.: Automatic generation of large scale 3D cloud based on weather forecast data. In: International Conference on Virtual Reality and Visualization, pp. 69–73 (2012)
101. Wither, J., Bouthors, A., Cani, M.P.: Rapid sketch modeling of clouds. In: Proceedings of the Fifth Eurographics Conference on Sketch-Based Interfaces and Modeling, pp. 113–118. Eurographics Association (2008)
102. Xu, J., Yang, C., Zhao, J., Wu, L.: Fast modeling of realistic clouds. In: International Symposium on Computer Network and Multimedia Technology, pp. 1–4 (2009)
103. Yaegar, L., Upson, C., Myers, R.: Combining physical and visual simulation-creation of the planet Jupiter for the film "2010". ACM SIGGRAPH Comput. Graph. **20**(4), 85–93 (1986)
104. Ye, Z.: Volumetric cloud rendering: an animation of clouds. Master Thesis, Clemson University (2014)
105. Yoshimasa, T.: The motion of uneven structure of convective clouds. J. Atmos. Sci. **50**(4), 574–587 (1993)
106. Yu, C.M., Wang, C.M.: An effective framework for cloud modeling, rendering, and morphing. J. Inf. Sci. Eng. **2**, 891–913 (2011)
107. Yuan, C., Liang, X., Hao, S., Qi, Y., Zhao, Q.: Modelling cumulus cloud shape from a single image. Comput. Graph. Forum **33**(6), 288–297 (2014)
108. Yuan, C., Liang, X., Hao, S., Yang, G.: Modeling large scale clouds from satellite images. Short paper, Pacific Graphics (2013)
109. Yusov,E.:High-performance rendering of realistic cumulus clouds using pre-computed lighting. In: Proceedings of High Performance Graphics (HPG), pp. 127–136. Eurographics Association (2014)

Prashant Goswami is Assistant Professor at BTH Sweden since November 2014. He completed his B.Tech and M.Tech from IIT Delhi in 2005 in Computer Science and engineering and then later did his PhD at University of Zürich (2007–2011) in computer graphics. He spent around one year as a research fellow at NTU Singapore (2012–2013) and then later worked as a postdoc at INRIA, Grenoble (2013–2014) before joining his current position. His research interests are in the field of computer graphics including animation, dealing with large data sets, GPU-based programming, geometric modeling and physics-based computation.