Procedural Cloudscapes

Antoine Webanck, Yann Cortial, Eric Guérin, Eric Galin

EUROGRAPHICS 2018 / D. Gutierrez and A. Sheffer
(Guest Editors)

Volume 37 (2018), Number 2

Procedural Cloudscapes

A. Webanck[1], Y. Cortial[2], E. Guérin[2] and E. Galin[1]

[1] Univ Lyon, Université Lyon 1, CNRS, LIRIS [2] Univ Lyon, INSA-Lyon, CNRS, LIRIS

Abstract

We present a phenomenological approach for modeling and animating cloudscapes. We propose a compact procedural model for representing the different types of cloud over a range of altitudes. We define primitive-based field functions that allow the user to control and author the cloud cover over large distances easily. Our approach allows us to animate cloudscapes by morphing: instead of simulating the evolution of clouds using a physically-based simulation, we compute the movement of clouds using key-frame interpolation and tackle the morphing problem as an Optimal Transport problem. The trajectories of the cloud cover primitives are generated by solving an Anisotropic Shortest Path problem with a cost function that takes into account the elevation of the terrain and the parameters of the wind field.

Keywords: clouds, procedural modeling, morphing, implicit surfaces.

CCS Concepts

•**Computing methodologies** → **Procedural animation; Volumetric models;**

1. Introduction

Cloudy skies play an important part in the appearance of scenic landscapes. Clouds have complex shapes and play a major part in the lighting conditions of a synthetic scene. At sunset, clouds burst into fiery colors and generate crepuscular rays which contributes to the majesty of the sky and add dramatic visual effects.

Modeling and simulating synthetic clouds remains a complex problem. The challenge stems not only from the complexity of the underlying physics featuring unstable dynamical systems, but also from the size of the three dimensional domain that should be simulated. Moreover, simulation methods do not provide sufficient control to artists and designers who want to tune the distribution, dimension and location of clouds to produce special lighting effects. Finally, simulations need to compute an entire sequence of iterations to define the cloudscape at a given input time.

Another challenging problem is the control of the cloud shapes. While primitive-based cloud modeling systems have been proposed for creating complex volumetric clouds such as *Cumulus*, these approaches need hundreds of primitives to define a single cloud, which makes them inappropriate for large cloudscapes modeling.

We approach the problem differently. Instead of physically simulating clouds, we present a novel procedurally-based cloudscape model based on key-framing and morphing. It is efficient and allows for a high degree of control. Moreover, contrary to primitive-based existing methods, we do not focus on the shape of a single cloud but on the procedural definition of large cloudscapes.

In our method, the presence of clouds is described by implicit skeletal functions, making it compatible with many existing sketching and procedural methods. We rely on a new formalism for the morphing of clouds with an Optimal mass Transport formulation. Because each interpolated frame of the animation is described by the same model, the user can freely consider it as a new key-frame and modify it to deform the animation.

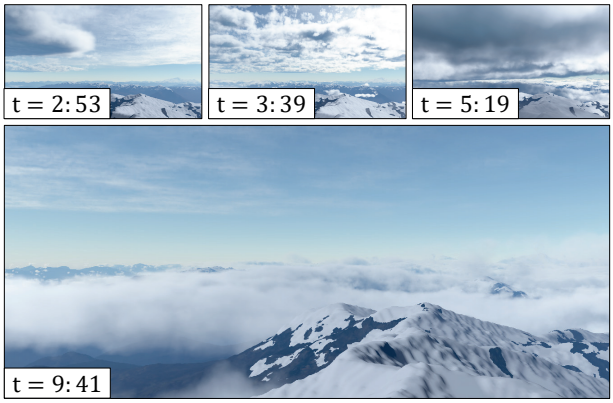


Figure 1: *Morphing of cloudscapes by combining an implicit-based density definition and an Optimal Transport algorithm.*

We claim the following contributions. We present an original and unified framework for efficient generation and control of large cloudscapes featuring different types of clouds. Clouds are modeled by combining functions defining the shape of the clouds, and

high level cloud cover control functions. We present a new method for the intuitive authoring of cloudscape animations: we introduce an efficient morphing algorithm that relies on an *Anisotropic Shortest Path* combined with an *Optimal Transport* method to solve the correspondence problem and compute a consistent interpolation between two input cloudscapes.

To our knowledge, our approach is the first to allow user-controlled authoring of animated cloudscapes by using a continuous density field morphing algorithm that takes into account the elevation of the terrain and the wind field, while providing a unified treatment of different types of clouds. The major motivation of our work is to define the entire cloudscape as a time varying analytical function, not requiring a simulation. This work has several applications, in particular in the entertainment industry, where there is a need for methods able to control the weather on large landscapes.

2. Related work

Existing techniques for modeling and animating three dimensional volumetric clouds can be classified into three categories: physically-based simulations, procedural generation and sketch-based modeling. Volumetric representations define a density function characterizing the water content of the clouds at every point in space. Simulating the interaction between light and clouds is beyond the scope of the paper; for a more in depth overview on cloud rendering techniques, we refer the reader to the recent work of [KMM*17].

Procedural modeling approaches typically define the density function as a combination of procedural noise [LLC*10] and ellipsoid primitives defining a compact support. The shape of stratiform clouds is generally modeled by horizontal layers of varying thickness [Gar85, BNL06] whereas cumuliform clouds are defined by blending ellipsoid primitives [Gar85, DNYO98, Ney97, SSEH03, BN04]. Details are added to those large-scale models by combining high-frequency procedural noise.

While previous approaches are effective for modeling a few clouds, they do not scale well and cannot be used to model large cloudscapes. Ebert *et al.* [EMP*94] proposed to use turbulence, a sum of noise functions at different scales, combined with some warping functions to define clouds with vortexes and Coriolis effects in the atmosphere of an entire planet. Clouds can be animated either by using four-dimensional noise or warping functions. Dobashi *et al.* [DNYO98] proposed to use multiple satellite images to capture the animation of clouds in the atmosphere. A hybrid technique combining a fluid simulation over a coarse three dimensional grid in the atmosphere and procedural generation for modeling clouds was proposed in [DYN06].

Primitive-based clouds can be animated using traditional key-framing and particle systems [SSEH03]. Specific morphing algorithms [JLCLW*05, LJW06, LJWcH07, CMCM11] were proposed to animate clouds by creating a correspondence graph between the initial and final cloud models and generating a generic interpolating model as described in [GA96]. A fundamental problem of these approaches is that the time varying model produces trajectories that linearly interpolate the positions of the initial and final primitives, which yields unrealistic animations. Moreover, they generate many primitives and do not scale to larger cloudscapes. In contrast, our morphing approach, based on the combination of Optimal Transport and Anisotropic Shortest Path, solves those problems by constructing consistent trajectories and limiting the number of generated primitives.

Sketching techniques aim at providing the user with a high level control over the cloud modeling. Complex models can be created by carefully editing all the parameters describing the cloud representation [SSEH03], but this method is time consuming and cumbersome for the user. Cloud-sketching systems [WBC08, DKNY08] allow the artist to outline the global shape of a cloud from which the system automatically generates a corresponding three-dimensional cloud.

Physically-based simulations aim at computing the evolution of clouds according to the laws of fluid dynamics and thermodynamics. Note that simulating and animating clouds is different from animating smoke [CT17, Thu17] because of the thermodynamics.

Methods for simulating and animating cloud formation using cellular automata combined with a coupled map lattice were proposed in [DKY*00, MYDN01]. A method for animating clouds surrounding the earth at planetary level based on high and low pressure regions was proposed in [DYN06]. Eulerian approaches [HL01, HBSL03, Ney97] take into account the thermodynamic equations but have several important limitations such as limited domain, coarse resolution, and lack of control. In contrast, Lagrangian approaches like *smoothed particle hydrodynamics* [MCG03] are considered more reliable on moving features, even at low resolutions. Barbosa *et al.* [BDY15] recently proposed a cloud simulation method based on *position-based* fluids [MM13]. The simulation, based on physics and thermodynamics, involves particle merging and splitting processes in dense and sparse cloud regions respectively, and requires a high number of particles.

The cloud shapes and motion depend on many simulation parameters and the initial status. The formation of clouds can be simulated using the complex fluid dynamics of the air [DNYO98, MDN02], but the process is costly and it is very difficult to adjust the parameters so that the clouds reproduce the desired shapes. [DKNY08] proposed a feedback method for better controlling the shape of cumuliform clouds.

Physically-based simulations are both computationally and memory demanding. Moreover, the complexity and instability of the thermodynamic and fluid dynamic systems make them unsuitable for authoring special effects. Finally, simulations require the computation of many steps to obtain the density field at a given time.

Our method proceeds differently: instead of simulating complex chaotic dynamic systems, our method focuses on user-control. Our key-frame morphing algorithm automatically generates a generic model representing the time varying density function. The correspondence between the initial and the final density distributions is obtained by using an Optimal Transport algorithm. It minimizes an anisotropic cost which takes into account the mass of the cloud, the wind and the elevation of the terrain with a view to generating plausible and consistent animations.

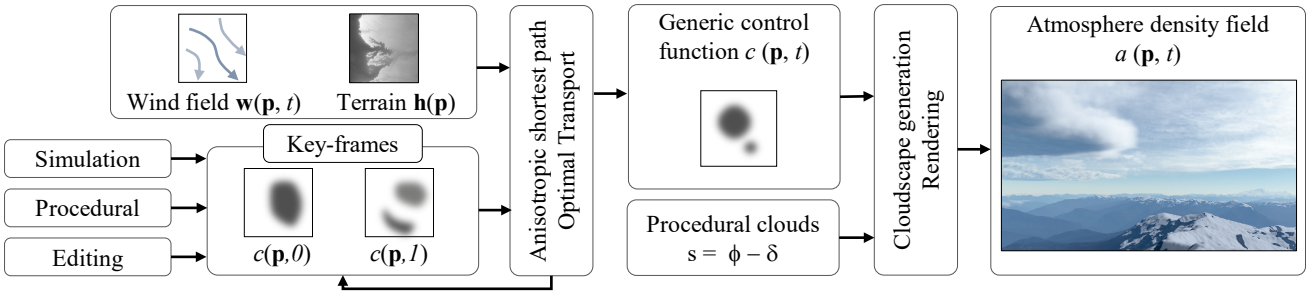


Figure 2: *Overview: given a set of input cloud cover key-frames, we generate a generic animated control function $c_i(\mathbf{p},t)$ which defines the time varying cloud cover. It is combined with the cloud layer models to compute the density functions $a_i(\mathbf{p},t)$ that are finally rendered.*

Type	Symbol	Definition
Per cloud type	ϕ_i	Cloud shape
	δ_i	Details
	s_i	Final shape
	c_i	Control for cloud type i
	a_i	Atmosphere for cloud type i
Global	a	Global atmosphere
	c	Global control
	σ	Normalizing function

Table 1: *Notations for the functions used in our cloudscape model.*

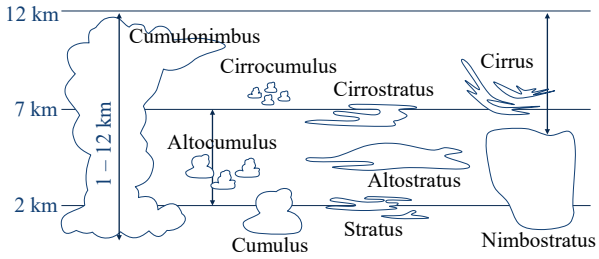


Figure 3: *Our procedural cloudscape model represents a variety of types of cloud layers featuring* Cumulus humilis*,* Altocumulus*, and* Altostratus*.*

3. Workflow

In this section, we present an overview of our cloudscape model and of the workflow for modeling and animating cloudscapes, and introduce notations.

Given an input terrain defined as an elevation function $h(\mathbf{p})$, wind conditions $w(\mathbf{p},t)$ and a set of images depicting the cloud layers at different key-frames, our method automatically generates a generic parametrized representation of the cloudscape interpolating the input key-frames (Figure 2). The key-frames may come from different applications and sources: they may be the result of a simulation with a coarse time step, real cloud cover data or obtained by interactive editing.

Our method proceeds as follows. First, the user sets the environment *i.e.* the terrain defined as an input height-field, and the wind conditions defined as a vector-field. For every key-frame, the user sketches a set of input images representing the cover map of the different cloud layers.

We then convert the input discrete cloud cover images into a continuous primitive-based representation that defines the control functions at the key-frames. Key-frames are then interpolated using an Anisotropic Shortest Path, and an Optimal Transport resulting in a complete animated atmosphere model. Finally, the animated cloudscape is rendered using a single scattering model.

The resulting animation can then be precisely tuned to match a desired state between two key-frames by simply inserting a new key-frame to guide the morphing process. Note that new key-frames may be captured from the morphing itself. An interesting property of our matching process based on Optimal Transport is that inserting such intermediate key-frames into the control sequence preserves the morphing without changing the animation. We detail our procedural cloudscape model in Section 4 and show how to author procedural animated cloudscapes by using an Anisotropic Shortest Path and an Optimal Transport method in Section 5. Table 1 provides an overview of the notations used throughout the paper.

4. Cloudscape model

As observed and described in meteorology, there exists a vast variety of types of cloud that span at various altitudes and with different shapes and movements. For example, fuzzy *Cirrus* form at high altitudes whereas convective cumuliform such as *Cumulus*, *Cirrocumulus* or *Altocumulus* extend over a wide range of altitudes.

Our model defines different types of clouds (Figure 3) denoted as $\mathcal{T}$ in a generic procedural framework. Every cloud type, denoted as $\mathcal{T}_i$, is defined by its characteristic density function a_i that implicitly defines the shape of clouds, its base elevation e_i and altitude range r_i (Table 2).

The base elevation and range limit the support of the density function a_i. We define the compact support of the function a_i as the set of points where the density is not null:

$$\Omega_i = \{\mathbf{p} \in \mathbb{R}^3 \,|\, a_i(\mathbf{p}) \neq 0\}$$

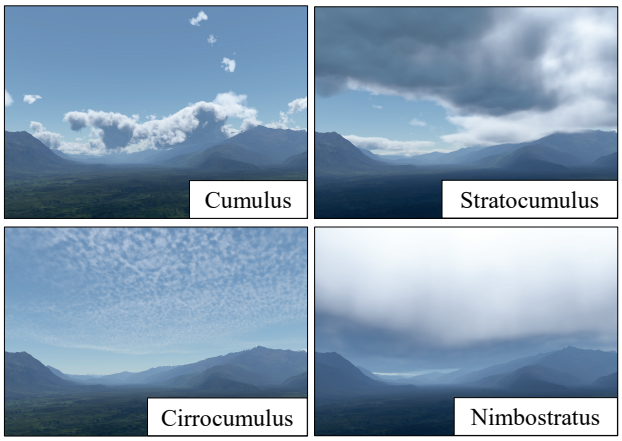


Figure 4: *Our procedural cloudscape model allows to represent different types of clouds.*

	Type	Elevation	Range
High	*Cirrus*	8.6 - 13.0	1.0 - 2.0
	Cirrostratus	6.0 - 8.0	0.1 - 3.0
	Cirrocumulus	6.0 - 8.0	0.2 - 0.4
Medium	*Altostratus*	3.0 - 5.0	1.0
	Altocumulus	2.0 - 6.0	0.2 - 0.7
	Stratus	0.1 - 0.7	0.2 - 0.8
	Stratocumulus	0.5 - 1.5	0.2 - 1.2
Low	*Nimbostratus*	0.1 - 1	2.0 - 5.0
	Cumulus	0.8 - 1.5	0.1 - 5.0
	Cumulonimbus	0.8 - 1.0	7.0 - 12.0

Table 2: *Type, base elevation e_i (*km*) and range r_i (*km*) parameters for several major tropospheric clouds for temperate latitudes as implemented in our system.*

Visually, the cloud shape of a given type $\mathcal{T}_i$ is similar to the compact support of the cloudscape function a_i. Although clouds are organized into a layered structure, the different layers may overlap. Figure 4 shows renderings of several types of clouds.

Complex and realistic models with a high level of detail can be created by blending many implicit primitives as demonstrated in [BN04]. In this approach the primitives not only define the density distribution but also the shape of the cloud. While this approach is effective for modeling a few clouds, it does not scale well and cannot be used to model large cloudscapes.

Therefore, for every cloud type $\mathcal{T}_i$, we define the density function a_i as a combination of two procedural functions: the shape function s_i representing the global characteristic density function of a given cloud type (Section 4.1) including its fine details, and c_i defining the presence of clouds (Section 4.2).

The density field of the atmosphere a is computed by summing all the cloud types contributions together. Summing the terms allows the overlapping cloud layers to smoothly blend together:

$$a(\mathbf{p},t) = \sum_{i \in \mathcal{T}} a_i(\mathbf{p},t)$$

As the sum may get larger than one, we normalize the result by using a sigmoid-like function $\sigma(x) = x/\sqrt{1+x^2}$.

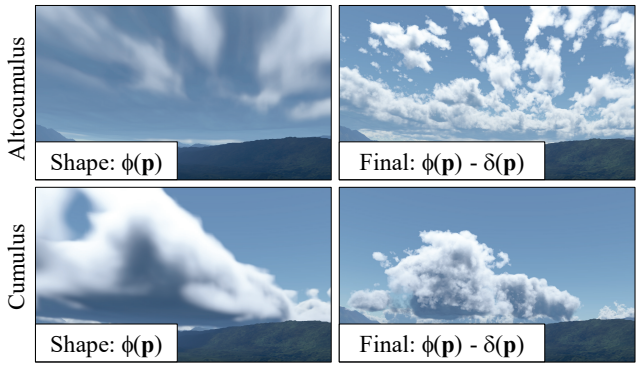


Figure 5: *Influence of the cloud shape functions: ϕ_i defines the overall density distribution (left); by subtracting details δ_i, we generate detailed shapes characterized by their complex support (right).*

4.1. Cloud shape functions

Every cloud type $\mathcal{T}_i$ is characterized by a specific cloud shape function denoted as $s_i = \max(\phi_i - \delta_i, 0)$. The function ϕ_i defines the overall aspect of the cloud and generates a low frequency distribution of density in space. The function δ_i defines high frequency details which are subtracted and produce finer details (Figure 5). The maximum function guarantees that the density field is always positive and plays an important part in the visual appearance of the clouds as they are defined as the compact support of the density field. Each type of clouds comes with its own definition of these two fields ϕ_i and δ_i, which produces the desired cloud effect. Appendix B gives an example of the functions ϕ_i and δ_i used.

Cloud growth and death The final density a_i of the cloud type i is obtained by combining s_i and c_i. The way to account for the field c_i is specific for every type of clouds. Stratiform clouds tend to smoothly fade in and out when they appear or disappear, therefore we multiply the density by the control field to obtain the corresponding fading effect. In contrast, cumuliform clouds have a more complex growth and vanishing process and do not vanish like high altitude clouds. Therefore, we subtract a varying value from the field to obtain a shrinking effect.

Recall that $s_i = \max(\phi_i - \delta_i, 0)$ defines the shape of the cloud. Let $\bar{c}_i = 1 - c_i$ denote the complementary of the control cloud cover function, we define the density function for every type of cloud $\mathcal{T}_i$ as:

$$a_i = (1 - \bar{c}_i \alpha_i)\, s_i - (1 - \alpha_i)\bar{c}_i$$

Since the density function should not be negative and less than 1, we clamp every cloud field a_i to the unit interval using the sigmoid-like function σ.

We control the behavior of the different types of clouds $\mathcal{T}_i$ by using a single coefficient, denoted as $\alpha_i \in [0,1]$, which weights clouds fading or shrinking. Shrinking is adapted to cumuliform clouds whereas fading is more appropriated to fog or high altitude stratiform clouds. Setting $\alpha_i = 1$ produces a fading effect whereas $\alpha_i = 0$ generates a shrinking effect; intermediary values allows us to weight effects (Figure 6). α_i is a constant defined for each cloud type, and is not interpolated between key-frames; in our implementation we used $\alpha_i = 0.1$ for *Cumulus* and $\alpha_i = 1$ for vanishing *Cirrostratus*. Recall that the compact support Ω_i of the function a_i represents the set of points where the density is not null. In the case of a fading effect, *i.e.* $\alpha_i = 1$, Ω_i remains unchanged when c_i diminishes. In contrast, prescribing $\alpha_i = 0$, will result in a shrinking effect of the domain as c_i decreases.

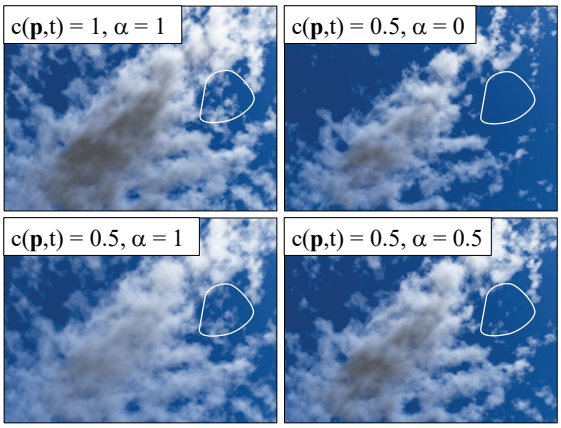


Figure 6: *Influence of the parameter* α*. The reference cloud (top left) can be modified to obtain fading (bottom-left), shrinking (top-right) or intermediary effects (bottom-right).*

Animation Every layer of our atmosphere model is animated independently. In our method, we rely on four-dimensional simplex noise and cellular [Wor96] noise functions to define the time varying density field. The control field $c_i(\mathbf{p},t)$ defines the global evolution of the cloud cover. $\phi_i(\mathbf{p},t)$ defines the evolution of the global shape of the clouds, whereas $\delta_i(\mathbf{p},t)$ is used to produce the small animation details.

4.2. Control functions

For every layer $\mathcal{T}_i$, we define a corresponding control function $c_i(\mathbf{p},t)$ that describes the large scale cover of the corresponding cloud type. This control function $c_i(\mathbf{p},t)$ can be produced in several ways, for instance procedurally or by a physical simulation. Figure 7 shows examples of control fields drawn by the user to obtain special effects such as a heart-shaped cloud cover, the condensation trails, or the hole in the dense cumuliform cloud cover matching a high mountain peak.

In order to animate these fields, we define them as implicit functions, by blending large scale spherical shaped primitives together.

During editing, the user provides the cloud covers as input images. This discrete representation is converted into a continuous

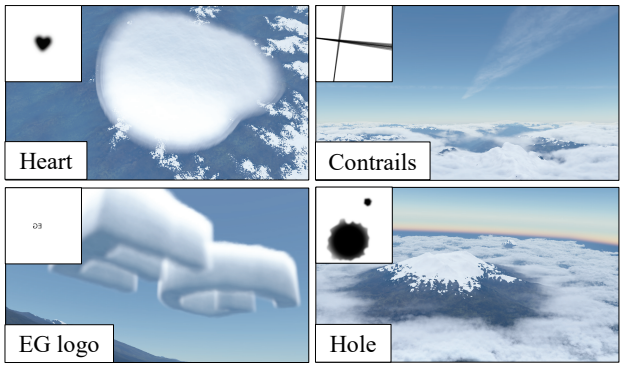
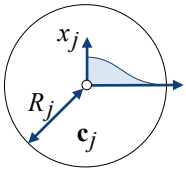


Figure 7: *Influence of the control field functions c. A heart-shaped cloud was carved in a layer of* Stratocumulus *(top left); condensation trails* Cirrus *produced by an aircraft cruising at high altitude (top right); letters carved in* Stratus *(bottom left) and cloud hole in* Cumulus *(bottom right).*

primitive-based representation. We sample the image over a coarse regular grid and generate a set of overlapping compactly supported kernels (control primitives). The density of each primitive is computed according to the greyscale value of the sampled image. Other parameters such as the radius or the altitude of the primitives are defined according to the cloud type.



Without loss of generality, let us consider a given layer $\mathcal{T}_i$. Every sphere primitive C_j is characterized by several parameters: its center $\mathbf{b}_j$, radius R_j and maximum density x_j respectively. The corresponding compactly supported field function is defined by combining a smooth decreasing falloff function with the Euclidean distance to the center:

$$c_j(\mathbf{p},t) = g_j \circ d_j(\mathbf{p},t) \qquad d_j(\mathbf{p},t) = \|\mathbf{b}_j(t) - \mathbf{p}\|/R_j(t)$$

We use the compactly supported cubic falloff function:

$$g_j(r) = x_j \left(1 - r^2\right)^3 \qquad \text{if } 0 \le r \le 1 \text{ and 0 otherwise}$$

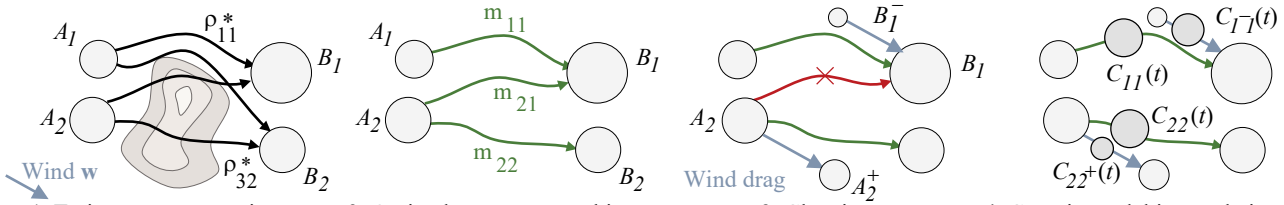
The field function $c(\mathbf{p},t)$ is defined by summing the contributions:

$$c(\mathbf{p},t) = \sum_j c_j(\mathbf{p},t)$$

As for cloud shape functions, we normalize the result by using the sigmoid-like function σ. The control functions enables us to define the cloud cover over large regions and allows to create special effects as depicted in Figure 7. Moreover, the primitive-based model allows us to propose a generic and consistent morphing algorithm that interpolates the distributions of two key-frames (Section 5).

5. Cloudscape morphing

Our cloudscape model defines the animation of the atmospheric density field $a(\mathbf{p},t)$ by combining the animation of the cloud models $s(\mathbf{p},t)$, and by animating the primitives of the control functions $c(\mathbf{p},t)$. The cloud shape functions s define the intrinsic animation



1. Trajectory computation **2.** Optimal transport matching **3.** Ghosting **4.** Generic model interpolation

Figure 8: *Overview of the morphing process. 1. All the possible trajectories between source and destination primitives A_i and B_j are computed. 2. Then, an Optimal Transport is performed to obtain the best match between pairs. 3. Trajectories that have a cost $\kappa(\rho^*_{ij})$ higher than a user-prescribed threshold are discarded and ghosts are inserted. 4. Finally, generic animated primitives C_{ij} are created from paired primitives, they follow the shortest path trajectory ρ^*_{ij} and their control parameters are interpolated.*

of the clouds such as smooth fluffy details for *Cirrus* or the convective animation of *Cumulus*. In contrast, the animation of the control function c defines the global evolution of the cloud cover.

Our approach for solving the animation problem consists in morphing two cloudscape models. A key-frame is defined as a set of control primitives at a given time step and belonging to the same cloud type. Without loss of generality, we address the morphing of two control fields, denoted as c_A and c_B. Out of clarity, let $A = \{A_i\}$ and $B = \{B_j\}$ denote the two models defining the control functions $c_A(\mathbf{p})$ and $c_B(\mathbf{p})$ at time steps $t_A = 0$ and $t_B = 1$ respectively. Our method proceeds in four steps as outlined in Figure 8.

Trajectory computation We first create the complete match graph $\mathcal{G}$ linking all the primitives A_i and B_j. For every edge (A_i, B_j) we compute the trajectory ρ_{ij} of the candidate interpolating primitive $C_{ij}(t)$ as the Anisotropic Shortest Path between the centers of the primitives (Section 5.1). The cost function of the Anisotropic Shortest Path algorithm takes into account the elevation of the terrain and the wind field to generate plausible trajectories.

Matching The complete graph has $n_A \times n_B$ edges. In order to reduce the number of the time varying primitives $C_{ij}(t)$, we compute an optimal correspondence graph $\widetilde{\mathcal{G}}$ matching only a few primitives A_i and B_j of the initial and final models. We solve the Optimal Transport of the mass of the primitives according to the previously computed trajectories ρ_{ij} (Section 5.2) to generate $\widetilde{\mathcal{G}}$ with at most $n_A + n_B - 1$ primitives.

Trajectory analysis and ghosting At the end of the matching step, some primitives may still have very high mass transportation costs. Such cases may occur when the prescribed key-frames are incompatible with the wind direction or the terrain. Therefore, we analyze the generated trajectories and the mass transport to invalidate some candidate primitives C_{ij}. For instance low altitude primitives with trajectories that intersect the terrain or move against the wind direction are discarded (Section 5.3). For every canceled edge (A_i, B_j) in the optimal graph $\widetilde{\mathcal{G}}$, we create two ghost primitives A_i^+ and B_j^- matching A_i and B_j respectively and perform a specific trajectory computation to guarantee that the overall mass transport is preserved.

Interpolation The cloudscape control function $c(\mathbf{p},t)$ is finally obtained by instantiating the generic primitive-based model $C(t) = \{C_{ij}(t)\}$ at the given time step t (Section 5.4).

5.1. Trajectory computation

We address the computation of a complex weighted Anisotropic Shortest Path problem on a continuous domain by taking into account the influence of the wind field, denoted as $w(\mathbf{p},t)$, and, depending on the cloud type, the elevation of the terrain denoted as $z(\mathbf{p})$ (Figure 8 step 1).

Let $T = [t_A, t_B]$ the time interval. Our goal is to compute a continuous path ρ from an initial point $\mathbf{a}(t_A)$ to a final point $\mathbf{b}(t_B)$ that minimizes the line integral over the path of a cost weighting function $\kappa(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$ that depends on the position $\mathbf{p}$ and the first successive derivatives denoted as $\dot{\mathbf{p}}$ and $\ddot{\mathbf{p}}$ respectively.

Let $\mathcal{P}$ denote the set of all continuous paths in a compact domain $\Omega \times T$ from $\mathbf{a}$ to $\mathbf{b}$ that are piecewise twice continuously differentiable, *i.e.* $\mathcal{P}$ denotes the set of continuous functions $\rho : T \to \Omega$, for which $\rho(t_A) = \mathbf{a}$ and $\rho(t_B) = \mathbf{b}$. Let $\kappa : \mathcal{P} \to [0, \infty($ denote the functional characterizing the cost of a path $\rho \in \mathcal{P}$:

$$\kappa(\rho) = \int_{t_A}^{t_B} \kappa(\mathbf{p}(t), \dot{\mathbf{p}}(t), \ddot{\mathbf{p}}(t))\, \mathrm{d}t$$

The continuous Anisotropic Shortest Path problem consists in finding a path ρ^* that minimizes the functional $\kappa(\rho)$:

$$\rho^*(\mathbf{a}, \mathbf{b}) = \underset{\rho \in \mathcal{P}}{\operatorname{argmin}}\ \kappa(\rho)$$

We approximate the continuous Anisotropic Shortest Path problem by embedding a spatio-temporal graph in the considered domain $\Omega \times T$. The shortest path is computed in three dimensions: two dimensions in space plus one dimension in time. Note that we do not sample the elevation, which would yield a fourth dimension; instead we take into account the elevation and the slope of the terrain as in [GPMG10] when computing the cost and warp the trajectories according to the elevation of the terrain (Figure 10).

The nodes of the graph are defined as spatio temporal points characterized by their position in Ω and time step in T, and the edges of the graph connect the spatio temporal point as depicted in Figure 9. This converts the continuous shortest-path problem into a shortest-path problem on a finite graph.

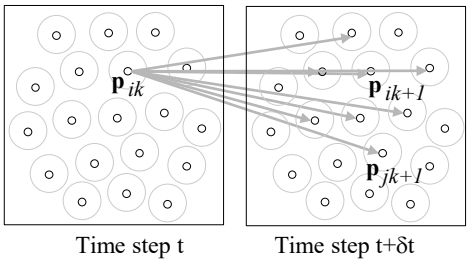


Figure 9: *Overview of the structure of the graph: the nodes $\mathbf{p}_{ik}$ of a given time-step t are connected only to the nodes $\mathbf{p}_{jk+1}$ of the next time-step $t+\delta t$; only a few edges have been drawn out of clarity.*

The fundamental problem consists in finding a good sampling of the three-dimensional domain $\Omega \times T$ to capture the influence of the wind field and compute plausible trajectories. We subdivide the time interval T into N time-steps, we will denote $\delta T = T/N$. The geometric domain Ω is sampled by generating a Poisson distribution of discs or radius r to generate nodes. The choice of the Poisson disc radius has an important impact on the results. The sampling should be sufficiently small to capture both the influence of the elevation of the terrain and the variations of speed. Let ε_Ω denote the desired terrain sampling accuracy, and ε_w the wind speed accuracy. The Poisson disc size can be defined as a function of the wind drag distance during one time step δT, and the terrain sampling accuracy:

$$r = 1/2 \min(\varepsilon_\Omega, \varepsilon_w \delta T)$$

In our experimentation, we used an average distance between samples of $2r = 1\,\text{km}$ with a time step $\delta T = 5$ min, resulting in a speed accuracy of $\approx 10\,\text{km}\,\text{h}^{-1}$.

The cost function κ is defined as a weighted sum of three terms that combines the influence of the terrain and the wind. The cost function involves an *elevation term* $\kappa_e(\mathbf{p})$ evaluating the intersection between the terrain and the cloud layer range (Figure 10), a terrain *slope* term $\kappa_s(\mathbf{p}, \dot{\mathbf{p}})$ that evaluates the slope of the terrain in the direction of the trajectory, and a *wind* $\kappa_w(\mathbf{p}, \dot{\mathbf{p}})$ term evaluating how clouds drift from the wind field. The last two terms strongly depend on the direction of the trajectory $\dot{\mathbf{p}}$, resulting in a global anisotropic cost function.

Let $\nabla \widetilde{z}(\mathbf{p})$ denote the gradient of the smoothed terrain, the slope term can be written as:

$$\kappa_s(\mathbf{p}, \dot{\mathbf{p}}) = \nabla \widetilde{z}(\mathbf{p}) \cdot \dot{\mathbf{p}} / ||\dot{\mathbf{p}}||$$

Note that other slope metrics could be used, for instance we could use an asymmetric terrain elevation metric so that clouds would require more energy to travel uphill than downhill.

The elevation term $\kappa_e(\mathbf{p})$ computes the intersection of the smoothed terrain at $\mathbf{p}$ with the cloud range $[a, b]$:

$$\kappa_e(\mathbf{p}) = (\widetilde{z}(\mathbf{p}) - a)/(b - a)$$

We clamp the value of $\kappa_e(\mathbf{p})$ to $[0, 1]$ so that if the terrain rises above the cloud range, the elevation term equals to 1.

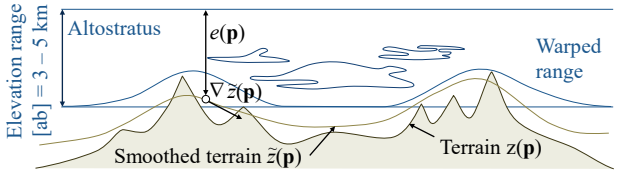
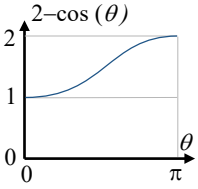


Figure 10: *The elevation range of the cloud layer base is warped by a smoothed terrain elevation function $\widetilde{z}$. The terrain influences the cost only when $\widetilde{z}(\mathbf{p}) > a$ where a denotes the base elevation of the cloud layer.*



Recall that $\mathbf{w}$ denotes the wind vector field, the function κ_w measures the cost for diverging from the trajectory formed by being advected by the wind. It is a function of the difference between the speed of the trajectory and the speed of the wind, and also takes into account the angle $\theta \in [0, \pi]$ between the two directions. Thus we define:

$$\kappa_w(\mathbf{p}, \dot{\mathbf{p}}) = ||\dot{\mathbf{p}} - \mathbf{w}||\,(2 - \cos(\theta)) \qquad \cos(\theta) = \dot{\mathbf{p}}/||\dot{\mathbf{p}}|| \cdot \mathbf{w}/||\mathbf{w}||$$

The angle-dependent penalty term $2 - \cos(\theta)$ is depicted hereby and prevents movement in the opposite direction of the wind.

5.2. Matching

Automatic and consistent matching is a fundamental problem in morphing. Although several heuristics such as cellular matching or user-prescribed matching have been proposed in the context of implicit surface modeling [GA96], existing approaches generate too many correspondence links or require considerable user-tuning. Our approach is inspired from the Optimal Transport algorithm of Bonneel *et al.* [BvdPPH11]. It generates at most $n_A + n_B - 1$ edges between the set of initial and final primitives A_i and B_j.

First, we compute the masses m_{A_i} and m_{B_j} of the initial and final primitives A_i and B_j. We integrate the control functions over their compact support, thus for a given control primitive, its mass m_{A_i} can be written as:

$$m_{A_i} = \int_{\Omega_{A_i}} c_{A_i}(\mathbf{p})\, \mathrm{d}\omega$$

The total mass m_A is defined as the sum of all masses m_{A_i}. The masses m_{B_j} and m_B are defined in the same way. In our implementation, we use spherical primitives with a cubic falloff function that allows us to derive a closed form expression of the mass with respect to the radius and maximum density of the primitives; details can be found in Appendix A. Finally, Optimal Transport requires that the initial and final total masses should be equal; thus we perform a normalization step so that the masses of the initial and final models A and B should be equal to 1: $\widetilde{m}_{A_i} = m_{A_i}/m_A$ and $\widetilde{m}_{B_j} = m_{B_j}/m_B$.

We use the previously computed minimal cost $\kappa(\rho^*(A_i, B_j))$ to define the cost associated to the transport of a unit of mass between a source primitive A_i and a target primitive B_j. Given the normalized masses and the trajectory cost for every pair (A_i, B_j), the

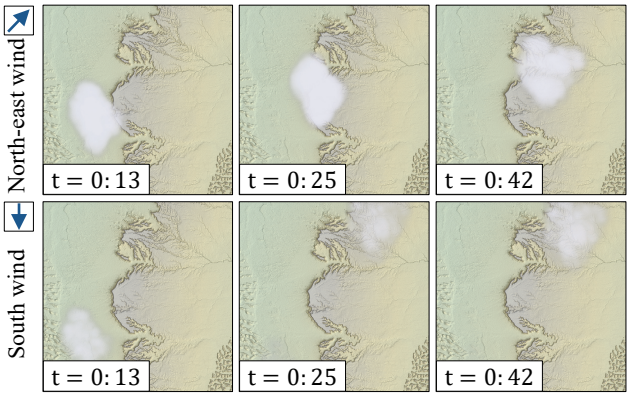


Figure 11: *Under favorable north-east wind conditions (top), no ghosting effects occur as clouds can be transported by admissible trajectories. Under opposite wind conditions (down), trajectories are discarded and ghosts are created for every primitive.*

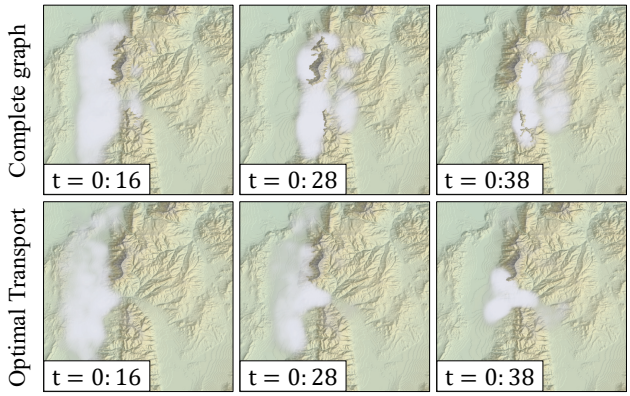


Figure 13: *Complete matching generates many cover primitives and their trajectories do not take into account the wind field and the terrain elevation. In contrast, our approach generates fewer primitives with consistent trajectories.*

Optimal Transport algorithm minimizes the cost of the morphing transformation between the initial and final models (Figure 8 step 2). Moreover, it computes the ratios of mass transported, denoted as m_{ij}, from each source primitive $A_i \in A$ to each target primitive $B_j \in B$. Non null factors m_{ij} represent a valid transportation edge (A_i, B_j) and therefore corresponds to an interpolating primitive C_{ij}. Recall that a notable property of the resulting transport plan is that it is composed of less than $n_A + n_B - 1$ edges, and generates the same number of interpolating primitives C_{ij}.

5.3. Trajectory analysis and ghosting

The Optimal Transport generates at most $n_A + n_B - 1$ paired primitives along with their trajectories $\rho^*(A_i, B_j)$. Some trajectories however may still have large costs: this may be the case for primitives moving in opposite direction to the wind (Figure 11), which are part of the solution of Optimal Transport problem but generate unrealistic animations (Figure 8 step 3).

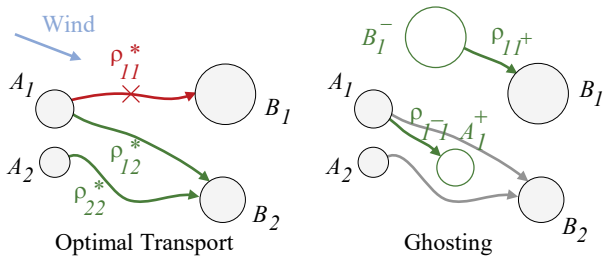


Figure 12: *After applying the Optimal Transport, trajectories that have a cost higher than a threshold are discarded (left). Ghosts are inserted according to the wind direction so as to obtain consistent trajectories (right).*

In those cases, we invalidate the path $\rho^*(A_i, B_j)$ and create corresponding *ghost primitives*, denoted as A_i^+ and B_j^- (Figure 12), that have the same radii as A_i and B_j, but a null maximum density $x_{A_i^+} = 0$ and $x_{B_j^-} = 0$. Their locations are derived form the centers $\mathbf{b}_{A_i}$ (and $\mathbf{b}_{B_j}$ respectively) by integrating the wind field forward and backward in time respectively, which also defines the two trajectories $\rho^*(A_i, A_i^+)$ and $\rho^*(B_j^-, B_j)$. We then add two edges in the graph (A_i, A_i^+) and (B_j^-, B_j).

When all the paths of a primitive are invalid, no mass transport exists and the primitive is excluded from the Optimal Transport algorithm and transformed into a ghost. This special case occurs in particular when a primitive is isolated from other primitives.

In addition to the previous ghosting algorithm, the user may create ghost primitives in the description of any key-frame. Ghosts have a mass equal to 0 however, as their maximum density is 0. Therefore, we allow the user to assign a virtual mass so that they should be taken into account automatically by the Optimal Transport algorithm.

Alternatively, the user may explicitly create edges between ghosts and other primitives after the Anisotropic Shortest Path and Optimal Transport computations as a post processing step, which allows to create special effects. Figure 13 shows an example of the way our method can account for the terrain geometry to avoid peaks. A complete-matching strategy produces unnatural trajectories that are not consistent with the wind field.

5.4. Interpolation

The previous steps provide us with a generic particle-based model $C(t)$ that represents the interpolation between A and B. New animated primitives are created from paired primitives. The centers $\mathbf{b}_{ij}(t)$ of the time varying primitives $C_{ij}(t)$ are computed according to their corresponding trajectory $\rho_{ij}(t)$ and the radii $R_{ij}(t)$ linearly interpolate R_i and R_j (Figure 8 step 4). Note that the interpolation of the maximum density x_{ij} has to be weighted by the mass ratio m_{ij} computed by the Optimal Transport, therefore we have: $x_{ij}(t) = (1-t)\, m_{ij}\, x_i + t\, m_{ij}\, x_j$.

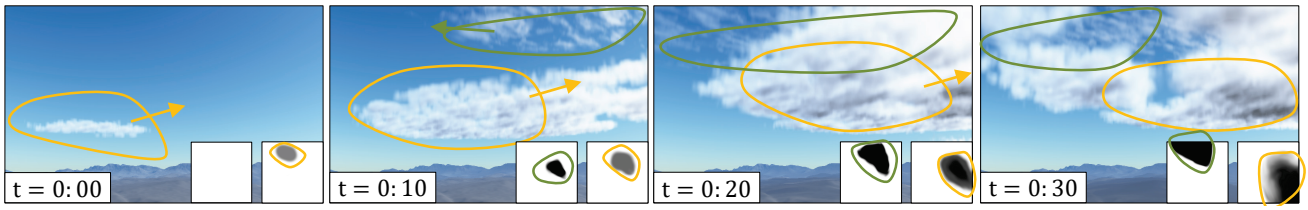


Figure 14: *Example of crosswinds with two different cloud types (*111 *primitives). The* Stratus *form at low altitude and follow the low altitude wind blowing from West-to-East.* Cirrus *start to appear at high altitudes and drift North-East to South-West an independent wind direction. The clouds start covering the sky according to their own wind field and user-defined key-frames.*

Figure	Layers	Size	Duration
Cross-winds (14)	2	$56 \times 56\,\text{km}^2$	0.5 h
Cold front (15)	8	$588 \times 120\,\text{km}^2$	10 h
Fog-rise (17)	2	$126 \times 126\,\text{km}^2$	2 h

Table 3: *Statistics for several animations: number of cloud type, size of the cloudscape, and duration of the scenario.*

6. Results

Our method was implemented and tested on an Intel Core i7 with 16 GB of RAM and a Nvidia GTX 970. The cloudscapes shown throughout the paper were generated using our algorithm. The cloudscape model and the control functions were implemented as shaders, and the different cloudscapes were rendered using a GPU-based single-scattering ray-marching algorithm.

6.1. Control

Our method allows to author various cloudscapes animations. Table 3 reports statistics for different scenarios. Note that the Optimal Transport algorithm guarantees consistency with respect to the key-frame insertion.

Our method allows to control wind constraints for different cloud layers. Figure 14 shows *Stratus* at medium altitude and *Cirrus* at high altitudes transported by winds blowing in different directions and at different speeds. The animation was obtained easily by prescribing different wind directions for every altitude range.

Figure 15 shows a meteorological cold front animation over the French Alps (see the accompanying video). A cold front is the leading edge of a cooler mass of air, replacing a warmer mass of air at ground level, and which lies within a fairly sharp surface of low pressure. The 10-hours scenario was easily created using our system by morphing cloud cover key-frames for every cloud types in the same wind direction from West to the East. It involves eight different types of cloud such as *Cumulus*, *Cumulus humilis*, or *Stratocumulus*.

Figure 17 shows a mountain range with fog dissipating in valleys and *Cumulus* forming at high altitudes. The wind inside the fog layer was defined as a small upward wind, whereas the wind in the *Cumulus* layer was set horizontal and stronger. Two key-frames were used to define the morphing of the fog. In the first key-frame, primitives were placed inside the valley beds, whereas the second key-frame was set to empty, causing an automatic fade-out thanks to the automatic ghost generation in the matching process.

6.2. Performance

The overall morphing process, encompassing the shortest paths computation and the Optimal Transport, runs in only a few seconds. This step needs to be performed only once and for all for every pair of key-frames. The primitives count for the *cross-winds* (Figure 14) and *fog-rise* (Figure 17) cloudscapes is 110 and 460 respectively.

We implemented the cloudscape model and the control functions as shaders on the GPU. The different cloudscapes shown throughout the paper were rendered using a GPU-based single-scattering ray-marching algorithm. Our procedural cloud functions with closed-form expression are computationally demanding as they involve many evaluations of noise functions for every type of cloud. The rendering of a single frame at 1280×720 resolution took about 10 s.

Although the performance of visualization is too slow to use it in real-time applications, to our knowledge, our method is the first that allows for generating analytical closed-form models for various cloud types and controlling cloudscapes in a coherent way. Simplifying and accelerating rendering for real time applications by optimizing the hierarchical representation of the different cloud primitives inside their corresponding layers is an important ongoing research, out of the scope of this paper.

6.3. Validation

We presented the results to meteorological experts (two researchers and a forecast teacher), a pilot and an virtual environment software expert. All appreciated the quality and realism of the animation and the rendering.

We conducted a small user study to compare the plausibility of our cloudscape animations to reality. The participants were asked to watch a real cloud animation of a given cloud type, then had to watch a synthesized animation produced by our method. Figure 16 shows some frames extracted from the *Cirrus* and *Altostratus* videos. Users were asked to rank the plausibility of our method with respect to the real example on a five-level Likert scale. We had 15 participants between the ages of 21 and 54. Some of them had significant computer graphics exposure, or were accustomed to

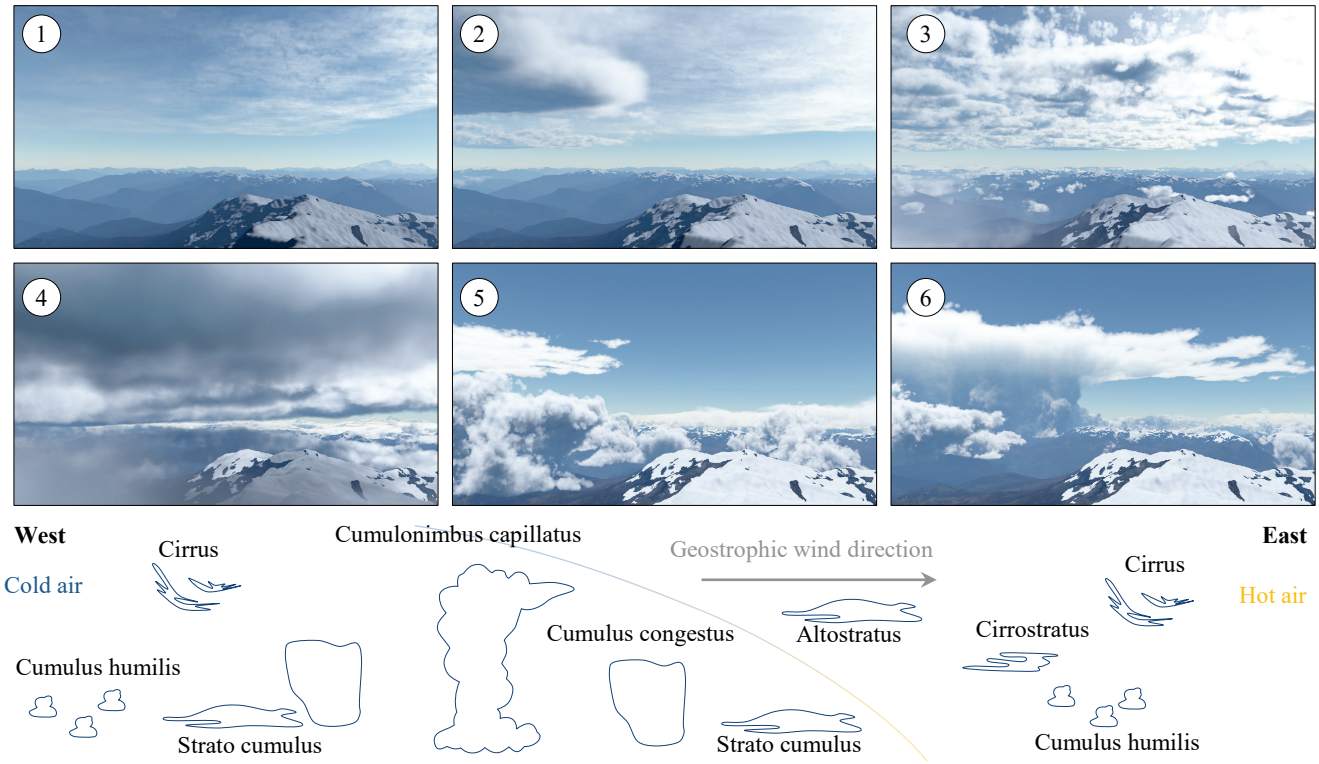


Figure 15: *This meteorological cold front scenario was created by morphing a vast variety of cloud types in the same West-to-East direction. The control field of each layer was defined by* 8 *primitives following the same wind direction. 1) The cold front starts with* Cirrus *followed by* Cirrostratus*. 2)* Altostratus *arrival. 3)* Altocumulus *and* Cumulus humilis *arrival. 4) Mid-altitude clouds are getting down and* Stratocumulus *are covering the landscape 5) At the heart of the front, stratiform clouds are replaced by convective clouds,* Cumulus congestus *6)* Cumulonimbus capilatus *finally appear, emphasizing the front instability.*

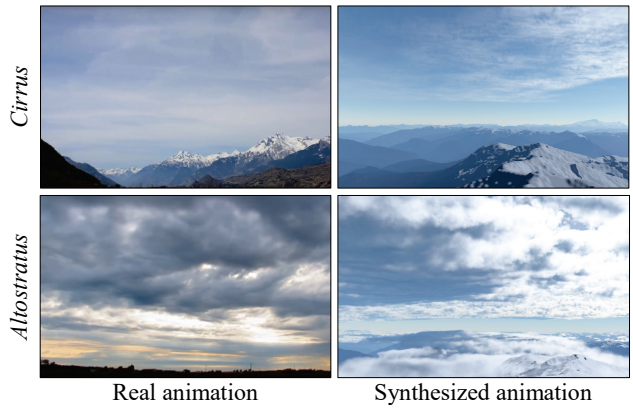


Figure 16: *Images from the videos used in the user study (extracted from* `https://youtu.be/vk-s9-qZ0bU` *and* `https://youtu.be/MtcxfjvwV1s`*).*

observing clouds. We performed the test on three types of clouds: *Cirrus*, *Altostratus* and *Cumulus*. Based on the null-hypothesis of the users answering equally negative and positive ranks, the *p-value* obtained by a *chi-squared* test was respectively less than 0.16, 0.011 and 0.034 for each type of cloud (χ^2 values of 2, 6.5 and 4.5 respectively).

The null-hypothesis can be thus discarded with a relatively strong certainty for *Altostratus* and *Cumulus*, demonstrating the effectiveness of our procedural model. In contrast, the *Cirrus* were not perceived as sufficiently convincing. One possible reason for this is that the wind directions were opposite in the real and synthesized animations, which was indeed reported by some participants at the end of the experiment.

6.4. Comparison

Our method compares favorably to primitive-based cloud animation systems in several ways. Traditional key-framing and particle systems [SSEH03] require tedious editing of the ellipsoid primitives constituting the clouds and do not scale for authoring large scenes. In contrast, our approach allows to manipulate large cloudscapes easily.

Moreover, our strategy based on Optimal Transport used for matching large cloud cover is central in our method and allows us to synthesize cloudscape animations consistent with the wind field and the terrain. Earlier morphing algorithms [JLCLW*05, LJW06, LJWcH07, CMCM11] that were proposed to animate primitive-

Figure 17: *Example of ghosting effects: the fog that covering the valleys (left) progressively rises and vanishes as the temperature increases and is swept away by small breezes (center); when the fog has almost completely disappeared,* Cumulus *start to form at higher altitudes. (*465 *primitives)*

based clouds generate trajectories that linearly interpolate the positions of the initial and final primitives, which yields unrealistic animations. Moreover, the matching process based on heuristics often generates many primitives, in general $O(n_A \times n_B)$, which makes them ill-suited for morphing large cloudscapes.

In contrast, our morphing approach, based on a combination of Optimal Transport and Anisotropic Shortest Path, solves those problems by generating consistent trajectories and limiting the number of generated primitives to $O(n_A + n_B)$. Furthermore, the cost function allows the user to influence the trajectory of the clouds. One important contribution of our method is the trajectory computation that accounts for the environment parameters such as the wind direction and the terrain shape. In addition, we propose for the first time a general model that embeds all the different types of clouds in a procedural and controllable way.

Limitations Our approach does not lend itself for extreme weather phenomena such as cyclones or hurricanes. In those particular cases, the wind direction should be defined at high resolutions, which is not currently possible in our modeling pipeline. Another limitation of our method is that morphing is limited to a single cloud type, *i.e.* we cannot transform a cloud type in another type with our morphing process. Because we control the cloud cover with large primitives, some of the generated clouds do not properly interact with the mountains over which they move. A solution would consist in using a larger amount of smaller primitives, at the price of more computationally demanding computations.

Finally, because we blend the densities of the cloud layers into a single density field, we are not able to take into account the different light-scattering properties of the cloud types.

7. Conclusion

We have presented a novel cloudscape editing framework which, for the first time, allows to generate and control complex evolutions of cloudscapes given input key-frames. It is achieved with a layered atmosphere model that procedurally defines the characteristics of the various types of clouds. Cloudscapes can be easily controlled by using geometric primitives that define the cover for every cloud layer. We avoid complex physically-based simulations over large volumetric domains by using a morphing approach based on an Anisotropic Shortest Path algorithm combined with Optimal Transport which, combined together, generate a continuous procedural density function model which can be evaluated directly at any time step and allows the user to author weather scenarios.

The user can choose to interact only through the control functions of the different layers, or edit specific control primitives to achieve special effects. Our framework enables the user to paint any of the key-frame layers during the morphing process, and insert or remove key-frames easily.

Future work could include adding other types of clouds and other procedural animations to broaden the range of meteorological phenomena, such as storms, or even gales. It would also be interesting to improve the influence of the elevation, the vegetation and water bodies over the generated clouds to take into account evapotranspiration. Finally, one important open problem would be to increase the scale of the model one level further to achieve a hierarchical, procedurally-defined cloudy atmosphere evolution over an entire planet, at different levels of detail.

Acknowledgments This work was supported by the project PAPAYA P110720-2659260, funded by the Fonds National pour la Société Numérique and the project HDW ANR-16-CE33-0001.

References

[BDY15] BARBOSA C. W. F., DOBASHI Y., YAMAMOTO T.: Adaptive cloud simulation using position based fluids. *Computer Animation and Virtual Worlds 26*, 3–4 (2015), 367–375. 2

[BN04] BOUTHORS A., NEYRET F.: Modeling clouds shape. In *Eurographics Short Papers* (2004). 2, 4

[BNL06] BOUTHORS A., NEYRET F., LEFEBVRE S.: Real-time realistic illumination and shading of stratiform clouds. In *Eurographics Workshop on Natural Phenomena* (2006), pp. 41–50. 2

[BvdPPH11] BONNEEL N., VAN DE PANNE M., PARIS S., HEIDRICH W.: Displacement interpolation using lagrangian mass transport. *ACM Transactions on Graphics 30*, 6 (2011), 158:1–158:12. 7

[CMCM11] CHUNG-MIN Y., CHUNG-MING W.: An effective framework for cloud modeling, rendering, and morphing. *Journal of Information Science and Engineering 27*, 3 (2011), 891–913. 2, 10

[CT17] CHU M., THUEREY N.: Data-driven synthesis of smoke flows with cnn-based feature descriptors. *ACM Transactions on Graphics 36*, 4 (2017), 69:1–69:14. 2

[DKNY08] DOBASHI Y., KUSUMOTO K., NISHITA T., YAMAMOTO T.: Feedback control of cumuliform cloud formation based on computational fluid dynamics. *ACM Transactions on Graphics 27*, 3 (2008), 94:1–94:8. 2

[DKY*00] DOBASHI Y., KANEDA K., YAMASHITA H., OKITA T., NISHITA T.: A simple, efficient method for realistic animation of clouds. In *SIGGRAPH* (2000), pp. 19–28. 2

[DNYO98] DOBASHI Y., NISHITA T., YAMASHITA H., OKITA T.: Modeling of clouds from satellite images using metaballs. In *Pacific Conference on Computer Graphics and Applications* (1998), pp. 53–60. 2

[DYN06] DOBASHI Y., YAMAMOTO T., NISHITA T.: A controllable method for animation of earth-scale clouds. In *Computer Animation and Social Agents* (2006), pp. 43–52. 2

[EMP*94] EBERT D. S., MUSGRAVE F. K., PEACHEY D., PERLIN K., WORLEY S.: *Texturing and Modeling: A Procedural Approach*. 1994. 2

[GA96] GALIN E., AKKOUCHE S.: Blob metamorphosis based on minkowski sums. *Computer Graphics Forum 15*, 3 (1996), 143–153. 2, 7

[Gar85] GARDNER G. Y.: Visual simulation of clouds. *SIGGRAPH Computer Graphics 19*, 3 (1985), 297–304. 2

[GPMG10] GALIN E., PEYTAVIE A., MARÉCHAL N., GUÉRIN E.: Procedural Generation of Roads. *Computer Graphics Forum (Proceedings of Eurographics) 29*, 2 (2010), 429–438. 6

[HBSL03] HARRIS M. J., BAXTER W. V., SCHEUERMANN T., LASTRA A.: Simulation of cloud dynamics on graphics hardware. In *SIGGRAPH/Eurographics Conference on Graphics Hardware* (2003), pp. 92–101. 2

[HL01] HARRIS M. J., LASTRA A.: Real-time cloud rendering. *Computer Graphics Forum 20*, 3 (2001), 76–85. 2

[JLCLW*05] JIN X., LIU S., C. L. WANG C., FENG J., SUN H.: Blob-based liquid morphing. *Computer Animation and Virtual Worlds 16*, 3–4 (2005), 391–403. 2, 10

[KMM*17] KALLWEIT S., MÜLLER T., MACWILLIAMS B., GROSS M., NOVÁK J.: Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks. *ACM Transactions on Graphics 36*, 6 (2017), 231:1–231:11. 2

[LJW06] LIU S., JIN X., WANG C. C. L.: Target shape controlled cloud animation. In *Computer Graphics International* (2006), pp. 578–585. 2, 10

[LJWcH07] LIU S., JIN X., WANG C. C. L., CHUEN HUI K.: Ellipsoidal-blob approximation of 3d models and its applications. *Computers & Graphics 31*, 2 (2007), 243–251. 2, 10

[LLC*10] LAGAE A., LEFEBVRE S., COOK R., DEROSE T., DRETTAKIS G., EBERT D. S., LEWIS J., PERLIN K., ZWICKER M.: A survey of procedural noise functions. *Computer Graphics Forum 29*, 8 (2010), 2579–2600. 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 154–159. 2

[MDN02] MIYAZAKI R., DOBASHI Y., NISHITA T.: Simulation of cumuliform clouds based on computational fluid dynamics. In *Eurographics Short Presentations* (2002), pp. 405–410. 2

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Transactions on Graphics 32*, 4 (2013), 104:1–104:12. 2

[MYDN01] MIYAZAKI R., YOSHIDA S., DOBASHI Y., NISHITA T.: A method for modeling clouds based on atmospheric fluid dynamics. In *Pacific Conference on Computer Graphics and Applications* (2001), pp. 363–372. 2

[Ney97] NEYRET F.: Qualitative simulation of convective cloud formation and evolution. In *Computer Animation and Simulation* (1997), pp. 113–124. 2

[SSEH03] SCHPOK J., SIMONS J., EBERT D. S., HANSEN C.: A real-time cloud modeling, rendering, and animation system. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 160–166. 2, 10

[Thu17] THUEREY N.: Interpolations of smoke and liquid simulations. *ACM Transactions on Graphics 36*, 1 (2017), 3:1–3:16. 2

[WBC08] WITHER J., BOUTHORS A., CANI M.-P.: Rapid sketch modeling of clouds. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2008), pp. 113–118. 2

[Wor96] WORLEY S.: A cellular texture basis function. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (1996), SIGGRAPH '96, pp. 291–294. 5

Appendix A: Mass of control primitives

We define the mass of a primitive $\mathcal{C}$ as the integration of its density $c(\mathbf{p})$ on its support Ω:

$$m_{\mathcal{C}} = \int_{\Omega} c(\mathbf{p})\,\mathrm{d}\omega = \int_{\Omega} g \circ d(\mathbf{p})\,\mathrm{d}\omega$$

By integrating over the spherical domain of radius R,we have:

$$m_{\mathcal{C}}(t) = \int_0^R \int_0^{2\pi} \int_0^{\pi} g(r)\, r^2 \sin\phi\,\mathrm{d}\phi\,\mathrm{d}\theta\,\mathrm{d}r = 4\pi \int_0^R g(r)\, r^2\,\mathrm{d}r$$

The integral can be computed for different falloff functions. The corresponding masses are defined by:

$$m_{\mathcal{C}} = \begin{cases} \dfrac{4}{3\times 5}\pi R^3 & \text{for } g(r) = -2\left(1-\dfrac{r}{R}\right)^3 + 3\left(1-\dfrac{r}{R}\right)^2 \\ \dfrac{4\times 16}{105\times 5}\pi R^3 & \text{for } g(r) = \left(1-\dfrac{r^2}{R^2}\right)^3 \end{cases}$$

Appendix B: Cloud functions

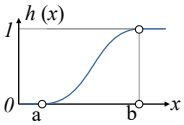
Here we present the functions ϕ and δ for *Altostratus* and *Altocumulus*. Further information about the other types of clouds is provided in the supplementary material. We use two turbulence functions, denoted as w and n respectively, defined as a sum of scaled cellular noise and simplex noise (Table 4).

$$\phi(\mathbf{p}) = k + (1-k)\,h(0.43, 1, n_{\phi}(\,(\mathbf{p}_x \cdot 0.35, \mathbf{p}_y, \mathbf{p}_z)/1000))$$
$$\delta(\mathbf{p}) = (1-k)(0.75\, n_{\delta}(\mathbf{p}/1000) + 0.25\,(1 - w_{\delta}(\mathbf{p}/1000)))$$

The turbulence functions are parameterized as follows:

Type	Octaves	Persistence	Lacunarity	Frequency
n_{ϕ}	8	0.52	2.02	36
n_{δ}	8	0.52	2.02	96
w_{δ}	4	0.33	3.00	400

Table 4: *Parameters of the turbulence functions.*



We denote $h(a,b,x)$ the smooth Hermite interpolating function between two values a and b. $k \in [0,1]$ is a morphing parameter between the closely related *Altostratus* ($k=1$) and *Altocumulus* ($k=0$) as those two types co-occur under the same meteorological conditions, any value in between gives a continuous blend between the two types. One modeling unit corresponds to a hundred meters.