TU WIEN Informatics

Real-Time Volumetric Rendering of Meteorological Cloud Data

A Procedural Upscaling and Visualization Pipeline for weBIGeo

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Informatics

by

Wendelin Muth
Registration Number 12226614

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dr.in techn. MSc Manuela Waldner

Vienna, March 23, 2026

Wendelin Muth	Manuela Waldner

Declaration of Authorship

Wendelin Muth

I hereby declare that I have written this thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work – including tables, maps and figures – which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.
I further declare that I have used generative AI tools only as an aid, and that my own intellectual and creative efforts predominate in this work. In the appendix "Overview of Generative AI Tools Used" I have listed all generative AI tools that were used in the creation of this work, and indicated where in the work they were used. If whole passages of text were used without substantial changes, I have indicated the input (prompts) I formulated and the IT application used with its product name and version number/date.

Vienna, March 23, 2026

Wendelin Muth

Acknowledgements

Above all, I want to thank my parents for making it possible for me to focus on my studies, and for supporting me throughout. I would also like to thank my supervisor Manuela Waldner for her guidance during this project.

Abstract

Traditional meteorological visualizations collapse the vertical structure of the atmosphere into two-dimensional map overlays, losing precisely the information most relevant in complex terrain. Expert tools for three-dimensional atmospheric exploration exist, but target domain scientists on dedicated workstation infrastructure. This thesis presents a pipeline for the real-time volumetric rendering of meteorological cloud data within the weBIGeo web-based geographic visualization platform, with the goal of making cloud structure intuitively readable to general users in a standard web browser.

The pipeline transforms hourly ICON-D2 forecast output into a compressed, streamable tile hierarchy that a WebGPU ray-marcher samples at interactive frame rates. Preprocessing a single forecast timestamp completes in approximately 33 s of compute time, producing a tile hierarchy of roughly 86 MiB on average. Rendering cost is well within interactive bounds: even under high cloud coverage, the cloud pass consumes around 2.25 ms of GPU time, a small fraction of the 33 ms budget for 30 fps. Qualitative evaluation against EUMETSAT satellite imagery shows that large-scale cloud patterns are reproduced with reasonable fidelity. Comparison against webcam imagery reveals three concrete limitations: the coarse and non-uniform vertical resolution of the source data is insufficient to resolve sharp fog layer boundaries, the sub-grid density distribution does not preserve the character of small cloud elements such as wispy puffs, and the tile resolution is too coarse to encode the surface texture of individual cumulus cells. The system is best understood as a large-scale atmospheric context layer rather than a precise local forecast tool.

Contents

CHAPTER 1

Introduction

In alpine environments, the vertical structure of the atmosphere is often what determines conditions on the ground. A hiker planning a ridge traverse, for instance, is less concerned with aggregate cloud cover than with whether clouds sit above or below the summit, whether it extends into the descent valley, or whether it is likely to lift by midday. Standard meteorological visualizations provide no direct answer to these questions. Atmospheric data is routinely communicated through two-dimensional map overlays that collapse vertical structure in favor of readability, a convention that serves many purposes but loses precisely the information most relevant in complex terrain.

Volumetric representations offer a more faithful alternative. By rendering atmospheric data as three-dimensional structures situated within the landscape, they allow the spatial relationship between clouds and terrain to be perceived directly, with altitude, depth, and lateral extent immediately apparent from the view. Such visualizations exist in meteorological research, where tools such as Met.3D [1] support the interactive three-dimensional exploration of forecast and ensemble data. These systems are designed for expert users and operate on dedicated infrastructure. Bringing comparable capability to interactive, web-based geographic environments accessible to general users remains an open problem.

This thesis addresses that gap within the context of weBIGeo [2], a research project at TU Wien whose aim is to make large geographic datasets interactively explorable in a standard web browser. Built upon the AlpineMaps framework [3], which provides tile-based 3D terrain rendering for the Austrian alpine region, weBIGeo has demonstrated this approach with applications such as real-time avalanche simulation. Volumetric cloud rendering constitutes a natural extension of the platform, adding a meteorological dimension to an existing geographic visualization context. The hiking scenario described above represents a long-term motivation for the platform, but is not achievable at the resolution of current publicly available forecast data, as discussed in Chapter 5.

The input data used in this thesis comes from the ICON-D2 model of the German Weather Service (DWD) [4], which provides volumetric atmospheric forecast fields across 65 vertical layers at a 2.2 km horizontal resolution. Ray-marching through volumes spanning hundreds of kilometers of alpine terrain differs fundamentally from rendering the localized cloudscapes typical of game engine environments. Rather than just per-sample rendering cost, the binding constraints here are VRAM budget, tile streaming bandwidth, and texture compression fidelity. Decomposing the domain into a streamable tile hierarchy, encoding it within the memory budget of a web client, and rendering it at interactive frame rates each impose their own constraints.

The primary contributions of this thesis are:

1. A GPU-accelerated preprocessing pipeline that transforms coarse ICON-D2 forecast data into upsampled, detail-augmented volumetric tiles organized in a TMS-compatible hierarchy for efficient streaming.
2. The integration of a ray-marching renderer, inspired by real-time cloud rendering techniques developed in the games industry, into the weBIGeo rendering pipeline, adding volumetric cloud visualization as an atmospheric layer to the platform.
3. A discussion of the practical constraints and tradeoffs encountered when operating at geographic scale in a web-based environment, with respect to memory, encoding, and rendering performance.

These contributions are structured around three research questions:

1. How can low-resolution meteorological cloud data be upsampled and augmented to produce continuous volumetric representations suitable for real-time visualization?
2. What preprocessing and data transformation steps are required to convert numerical weather forecast data into a format usable by a GPU-based geographic visualization engine?
3. How should volumetric cloud rendering adapt to viewing conditions to avoid obstructing the underlying terrain while preserving atmospheric context?

The remainder of this thesis is structured as follows. Chapter 2 surveys the relevant prior work in volumetric cloud rendering, sub-grid detail synthesis, and the weBIGeo and AlpineMaps frameworks this system builds upon. Chapter 3 describes the pipeline and rendering system in detail. Chapter 4 presents results and evaluates the system against the research questions. Chapter 5 concludes and outlines directions for future work.

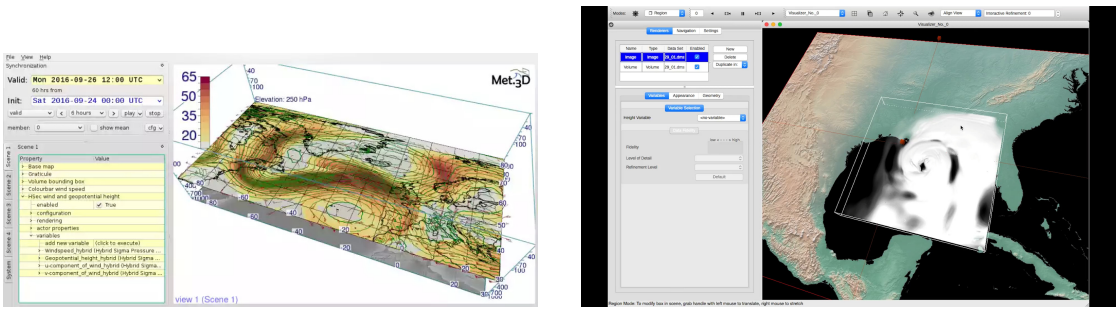
CHAPTER 2

Related Work

This chapter surveys the prior work this thesis builds upon, covering meteorological visualization tools, the real-time volumetric cloud rendering techniques that inform the rendering system, interpolation and procedural methods used in the preprocessing pipeline, and the geographic rendering context in which the system is deployed.

2.1 Meteorological Visualization Tools

Two prominent scientific tools for three-dimensional atmospheric visualization are Met.3D [1] and VAPOR [5]. Met.3D supports interactive volumetric exploration of ensemble weather prediction data, while VAPOR targets large volumetric datasets from atmospheric and oceanic simulations. Figure 2.1 shows example views from each tool.



(a) Met.3D. Screenshot from the Met.3D documentation. © 2015–2025 Met.3D contributors.

(b) VAPOR. Frame from a demonstration video by Scott Pearse [6].

Figure 2.1: Expert-oriented tools for volumetric atmospheric visualization. Both systems are designed for workstation use by domain scientists, in contrast to the web-based, general-audience context targeted by this thesis.

Both tools target domain scientists and require dedicated workstation infrastructure. The system presented in this thesis must instead run in a standard web browser, stream data over a network, and present meteorological information readably without domain expertise.

2.2 Real-Time Volumetric Cloud Rendering

The theoretical foundation for volumetric cloud rendering is the volume rendering equation, formalized by Kajiya and Von Herzen [7] and later expressed in the optical model framework by Max [8]. Early real-time approaches often relied on impostors—flat billboards textured with pre-rendered cloud images—which are computationally inexpensive but cannot represent internal volumetric structure. Harris and Lastra [9] demonstrated volumetric rendering as a more convincing alternative at interactive frame rates.

The most influential modern approach is the Nubis pipeline [10, 11, 12], developed for the game Horizon Zero Dawn and extended in subsequent productions. Nubis evaluates cloud density procedurally at each ray-march sample using composites of Perlin and Worley noise, shaped by weather parameters and vertical profile functions. A two-phase ray-march skips empty space via coarse MIP-level samples before committing to fine integration. Lighting combines Beer–Lambert attenuation, Henyey-Greenstein phase scattering [8], and an in-scattering approximation. The rendering approach in this thesis is directly inspired by the Nubis pipeline; the key difference is that cloud density is derived from ICON-D2 forecast fields rather than authored procedurally, with noise added only for sub-grid detail.

2.3 Sub-Grid Detail Synthesis

A central challenge of this work is bridging the resolution gap between the coarse ICON-D2 output and the sub-kilometer detail required for visually convincing rendering. For vertical interpolation, Piecewise Cubic Hermite Interpolating Polynomials (PCHIP) [13] are used, chosen because they preserve monotonicity and handle non-uniform spacing correctly by construction. To synthesize sub-grid cloud structure, turbulent kinetic energy from the model drives a coordinate displacement approximating sub-grid mixing, motivated by Rodean [14]. Procedural noise follows the Nubis approach: inverted Worley noise [15] for cumulus shapes and Fractional Brownian Motion (fBm) [16] for multi-scale stratus texture.

2.4 Web-Based Geographic Visualization

The AlpineMaps open-source framework [3] implements a tile-based 3D terrain renderer for the Austrian alpine region in C++ and Qt with WebGPU as its rendering backend; browser deployment is achieved through Emscripten compilation. weBIGeo [2] is a

Netidee-funded research project at TU Wien that extends AlpineMaps to make large geographic datasets interactively explorable in a standard web browser. This thesis is developed as part of weBIGeo. Data is organized according to Tile Map Service conventions, allowing hierarchical tile streaming on demand.

A closely related system is TrailView [17], a web-based tool for planning outdoor activities in alpine terrain (Figure 2.2). Its cloud pipeline ray-marches over Perlin and Worley noise textures parameterized by forecast values from the Open-Meteo API.

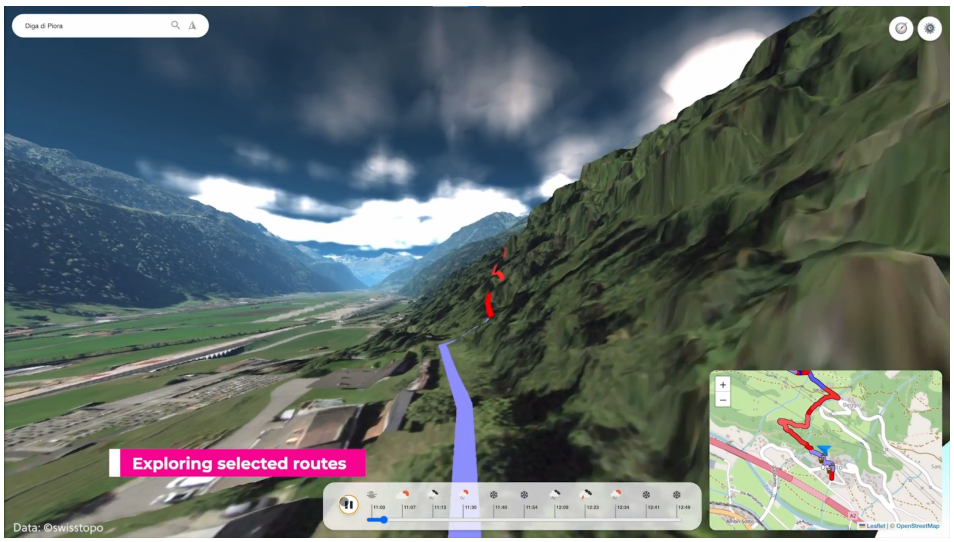


Figure 2.2: TrailView showing clouds rendered over alpine terrain with an active route and minimap. Frame from the supplemental video of [17].

The key distinction is that in TrailView, weather data serves as a scalar signal modulating a procedurally generated noise field, so the spatial structure of the clouds bears no direct relationship to the forecast. This thesis takes the opposite approach: the ICON-D2 volumetric field is itself the source of cloud structure, with procedural noise added only for sub-grid detail. The result is grounded in the spatial and vertical structure of the forecast, at the cost of greater preprocessing complexity and more demanding memory and streaming constraints.

CHAPTER 3

Implementation

3.1 System Overview

The system is divided into two phases: an ahead of time preprocessing pipeline and a runtime renderer. Figure 3.1 illustrates the end-to-end flow.

The split exists because the two phases differ fundamentally in their computational character. Transforming raw forecast data into a form suitable for real-time ray-marching is too expensive to perform per-frame and needs to run only once per forecast timestep. The renderer reads only the finished tile files and has no knowledge of the upstream pipeline that produced them.

Preprocessing. The preprocessing pipeline runs ahead of time once per ICON-D2 forecast timestep. It resamples raw model fields to a uniform metric grid, augments them with sub-grid cloud structure, and encodes the results into a streamable TMS tile hierarchy of GPU-compressed textures. A low-resolution shadow map is precomputed from the finished tiles and used at runtime to darken terrain beneath cloud cover.

Runtime renderer. The renderer is integrated into the weBIGeo rendering pipeline as a series of WebGPU compute passes. It is event-driven rather than running a fixed render loop, producing a new frame in response to camera movement or data updates. Preprocessed tiles are loaded on demand and radiance and transmittance are accumulated along rays cast through the cloud volume, producing a composited atmospheric layer over the terrain.

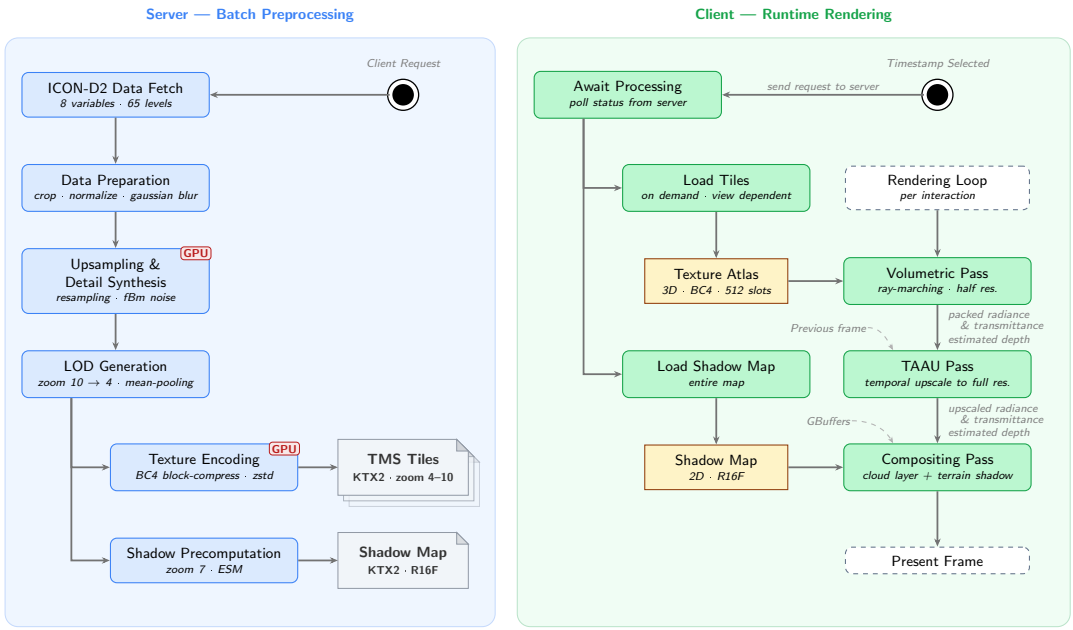


Figure 3.1: End-to-end pipeline. ICON-D2 forecast data is preprocessed into a TMS tile hierarchy, which is streamed at runtime to the WebGPU renderer integrated into weBIGeo.

3.2 Input Data

The preprocessing pipeline consumes forecast data from the ICON-D2 model of the German Weather Service (DWD) [4], a convection-permitting regional model covering Central Europe. The model runs every three hours and produces hourly forecasts up to a horizon of 48 hours. DWD publishes ICON-D2 output on both the native icosahedral grid and a regular latitude-longitude grid; this work uses the regular grid files. All variables share a horizontal resolution of approximately 2.2 km. After download, the data is cropped to the Austrian region (46.2°–49.2° N, 9.4°–17.4° E).

Vertical levels. The model atmosphere is discretized into 65 layers spanning from the surface to approximately 22.5 km altitude. Layers are thin and closely spaced near the surface, where atmospheric gradients are steepest, and become progressively thicker at higher altitudes. Layer boundaries are defined by 66 half-level heights (HHL). Most variables are defined at layer centers, which lie at the midpoints between adjacent HHL values. This distinction is relevant for interpolation: cell-center variables require separate stencil handling, as described in Section 3.3.1.

Variables. Table 3.1 lists the variables used in this work. Cloud cover fraction (CLC) acts as a threshold controlling cloud spatial extent: the procedural density field is accepted

where it exceeds $1 - \mathrm{CLC}$. The specific cloud water (QC) and ice (QI) contents, together with temperature (T) and pressure (P), determine the volumetric extinction coefficient (Section 3.3.1). Specific humidity (QV) enters through a virtual temperature correction. Turbulent kinetic energy (TKE) drives sub-grid detail synthesis (Section 3.3.1). The half-level height field (HHL) maps model levels to metric altitudes.

Variable	Unit	Levels	Description
HHL	m	66	Geometric height at half-level boundaries
CLC	%	65	Cloud cover fraction
QC	$\mathrm{kg\,kg^{-1}}$	65	Specific cloud water content
QI	$\mathrm{kg\,kg^{-1}}$	65	Specific cloud ice content
QV	$\mathrm{kg\,kg^{-1}}$	65	Specific humidity
T	K	65	Temperature
P	Pa	65	Pressure
TKE	$\mathrm{m^2\,s^{-2}}$	66	Turbulent kinetic energy

Table 3.1: ICON-D2 variables used in the preprocessing pipeline. HHL and TKE are defined at the 66 layer boundaries; all other variables are defined at the 65 layer centers.

3.3 Preprocessing Pipeline

The preprocessing pipeline transforms raw ICON-D2 forecast data into a TMS tile hierarchy of KTX2 textures ready for streaming. It runs ahead of time, once per forecast timestep, and consists of three stages: upsampling and detail synthesis, LOD generation, and shadow precomputation. The CPU-side code is written in Python; the computationally intensive synthesis step is GPU-accelerated via a WebGPU compute shader. The runtime renderer reads only the final KTX2 files and does not access any forecast data.

The scalar value stored per voxel is the volumetric extinction coefficient β ($\mathrm{km^{-1}}$), which quantifies how strongly a unit path length attenuates light. The input variables from Table 3.1 each play a distinct role in computing this final value.

- **CLC** (cloud cover fraction) determines *where* clouds exist spatially. It acts as a threshold: the procedural density field is accepted wherever it exceeds $1 - \mathrm{CLC}$, so grid cells with high cloud fraction let a greater portion of the procedural field contribute to the volume.
- **QC** and **QI** (specific cloud water and ice contents) determine how optically thick the cloud is. The ICON-D2 model stores these as *grid-mean* mixing ratios averaged over the entire grid cell, regardless of how the condensate is distributed within it. CLC provides a partial description of that distribution—it tells us what fraction of the cell is covered—but not where within the cloudy fraction the condensate

actually sits. The sub-grid placement is decided by the detail synthesis. To compute the extinction coefficient we therefore first recover the *in-cloud* concentration, i.e. how much condensate is present per unit volume inside the cloudy part of the cell, by dividing the grid-mean mixing ratios by $\mathrm{CLC}_{\mathrm{eff}}$. Together with temperature T and pressure P, these in-cloud values yield the volumetric water contents from which β is derived.

- **TKE** (turbulent kinetic energy) drives the sub-grid detail synthesis (Section 3.3.1), modulating coordinate displacement and the blend between stratus and convective noise paths.
- **T**, **P**, and **QV** enter through the ideal-gas air-density calculation used to convert mixing ratios to volumetric water contents.

The following subsections describe each pipeline stage in detail.

3.3.1 Upsampling and Detail Synthesis

This stage produces one volume tile for each TMS tile at the maximum zoom level (zoom 10). Each tile has a resolution of $256 \times 256 \times 64$ voxels and covers a geographic footprint of approximately 26.3 km × 26.3 km. The vertical extent is capped at 14.0 km altitude, above which visible clouds rarely occur; this limits the output volume to a meteorologically relevant range and allows the vertical voxel count to be halved from 128 to 64 relative to what the full model top would require. The value stored per voxel is the volumetric extinction coefficient β (km^{-1}), described in Section 3.3.1.

Data Preparation and Shader Dispatch

Before GPU work begins, the raw model fields are prepared on the CPU: CLC values are converted from percent to $[0, 1]$, values below 0.001 are zeroed, and all fields are smoothed with a separable Gaussian blur to prevent isolated cloud patches with abrupt transitions. The prepared fields are uploaded to the GPU as 3D textures.

The synthesis shader runs as a WebGPU [18] compute pass using the *wgpu* library [19]. Tiles are processed in batches of 4×4, padded by one eighth of the tile resolution on all horizontal sides for boundary consistency. A second compute pass applies a 3D Gaussian blur to reduce aliasing artifacts.

Resampling

Each output voxel corresponds to a fixed metric altitude on a uniform vertical grid. Horizontally, the surrounding 4×4 source columns are combined using cubic B-spline weights, giving smooth reconstruction without ringing. Vertically, model levels are non-uniformly spaced and vary spatially because HHL depends on local terrain height. For each source column, the shader locates the two levels bracketing the target altitude, then

applies PCHIP interpolation over a four-point stencil [13], preventing spurious overshoots in steep vertical gradients.

Cell-center variables (CLC, QC, QI, QV, T, P) use stencil positions at the midpoints $z_k^c = (h_k + h_{k+1})/2$; the half-level field TKE uses HHL values directly. Near domain boundaries, stencil columns that do not span the query altitude are excluded and the remaining B-spline weights renormalized.

Extinction Coefficient

The voxel value written to the output texture is the volumetric extinction coefficient β (km^{-1}), which quantifies how strongly a unit path length attenuates light. It is computed separately for the liquid and ice components, summed, and weighted by a noise-derived density factor ρ_d (described in Section 3.3.1), so that the stored quantity is $\beta \cdot \min(\rho_d, 2)$. The density factor modulates cloud boundaries so that they taper smoothly rather than transitioning as a hard step. The clamp at 2.0 prevents extreme amplification where the procedural texture exceeds unity.

Condensate consistency corrections. Two inconsistencies between the cloud fraction and condensate fields occur in ICON-D2 output [4]; both were apparent from comparison with visible-light satellite imagery and are illustrated in Figure 3.2.

The first is CLC $= 0$ where QC + QI > 0. Following Tompkins [20] and assuming an exponential total-water distribution, cloud cover and condensate are related by:

$$\mathrm{CLC}_{\mathrm{diag}} = \min\left(\frac{q_w}{Q_{\mathrm{crit}}}, 1\right), \quad Q_{\mathrm{crit}} = 3 \times 10^{-5}\,\mathrm{kg\,kg^{-1}} \tag{3.1}$$

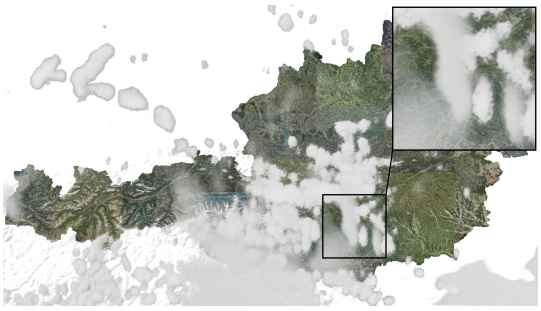
where $q_w = \mathrm{QC} + \mathrm{QI}$. The effective cloud fraction is $\mathrm{CLC}_{\mathrm{eff}} = \max(\mathrm{CLC}, \mathrm{CLC}_{\mathrm{diag}})$.

The second, more common inconsistency is CLC > 0 where QC + QI ≈ 0, occurring when the cloud fraction scheme retains non-zero values after microphysics has depleted condensate. A minimum condensate is reconstructed by inverting the same relation: $q_w = Q_{\mathrm{crit}} \cdot \mathrm{CLC}_{\mathrm{eff}}$, split into liquid and ice by a temperature-dependent ice fraction $f_{\mathrm{ice}} = \mathrm{clamp}(-T_c/38,\ 0,\ 1)$ where $T_c = T - 273.15$. The reconstructed values are applied as lower bounds on the NWP fields, gated by thermodynamic validity checks ($T > 180\,\mathrm{K}$, $P > 100\,\mathrm{Pa}$). This correction had a substantial effect on the visual output, filling cloud regions that would otherwise have been entirely transparent.

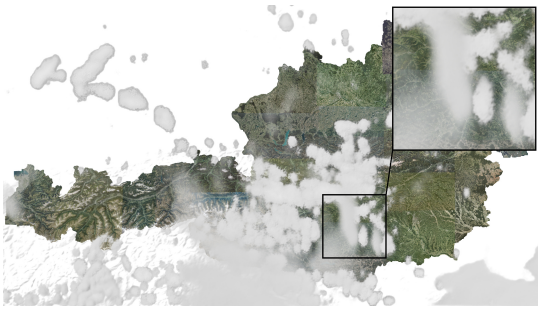
Air density and water content. Air density ρ_{air} at each voxel is derived from temperature T and pressure P via the ideal gas law with a virtual temperature correction for moisture:

$$\rho_{\mathrm{air}} = \frac{P}{R_d\, T_v}, \qquad T_v = T\,(1 + 0.608\, q_v) \tag{3.2}$$

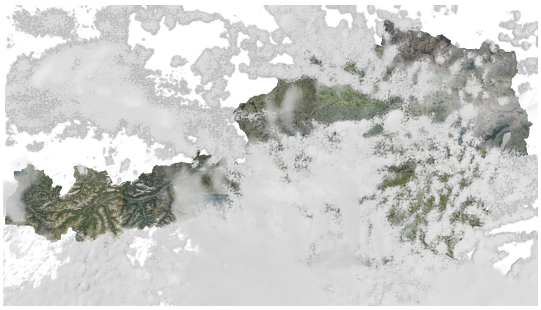
where $R_d = 287.05\,\mathrm{J\,kg^{-1}K^{-1}}$ is the gas constant for dry air, q_v is specific humidity, and the factor 0.608 follows from $(R_v/R_d - 1)$ with $R_v = 461.5\,\mathrm{J\,kg^{-1}K^{-1}}$. The grid-mean



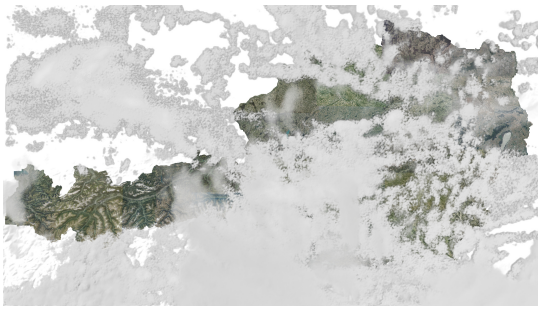
(a) No corrections applied. Two inconsistencies are present in the raw ICON-D2 fields: CLC may be zero where condensate exists, and condensate may be depleted where CLC remains non-zero. Both cause cloud regions to render as fully transparent.



(b) CLC correction only. Where condensate is present but CLC is zero, $\mathrm{CLC}_{\mathrm{diag}}$ raises $\mathrm{CLC}_{\mathrm{eff}}$ to a value consistent with the existing condensate. The visual effect is small in practice.



(c) Condensate correction only. Where CLC is non-zero but the microphysics scheme has depleted condensate, a minimum QC/QI is reconstructed from $\mathrm{CLC}_{\mathrm{eff}}$. This accounts for nearly all the visible change, though it introduces coverage overestimation.



(d) Both corrections applied. The result is visually close to (c); the CLC correction contributes additional cloud only where condensate was present but CLC was missing. Coverage overestimation remains.

Figure 3.2: Effect of the two-stage condensate consistency correction on the top-down rendered output, 2026-03-18 13:00 UTC. The condensate reconstruction (c) is responsible for nearly all the visible change; the CLC correction (b) has little independent effect in practice. The combined result (d) is used throughout this thesis. Coverage overestimation introduced by the condensate reconstruction is a known limitation, discussed further in Section 4.3.2.

mixing ratios QC and QI are divided by $\mathrm{CLC}_{\mathrm{eff}}$ to recover in-cloud values, since ICON-D2 stores grid-averaged quantities:

$$\mathrm{LWC} = \frac{\mathrm{QC}}{\mathrm{CLC}_{\mathrm{eff}}} \cdot \rho_{\mathrm{air}}, \qquad \mathrm{IWC} = \frac{\mathrm{QI}}{\mathrm{CLC}_{\mathrm{eff}}} \cdot \rho_{\mathrm{air}} \tag{3.3}$$

giving the volumetric liquid and ice water contents in $\mathrm{kg\,m^{-3}}$.

Extinction from water content. Extinction is computed from volumetric water content and an effective particle radius r_{eff}:

$$\beta = \frac{3}{2}\frac{\mathrm{LWC}}{\rho_w\, r_{\mathrm{eff,\,liq}}} + \frac{3}{2}\frac{\mathrm{IWC}}{\rho_i\, r_{\mathrm{eff,\,ice}}} \tag{3.4}$$

using $\rho_w = 1000\,\mathrm{kg\,m^{-3}}$ and $\rho_i = 917\,\mathrm{kg\,m^{-3}}$, valid in the geometric optics limit. The result is expressed in $\mathrm{km^{-1}}$.

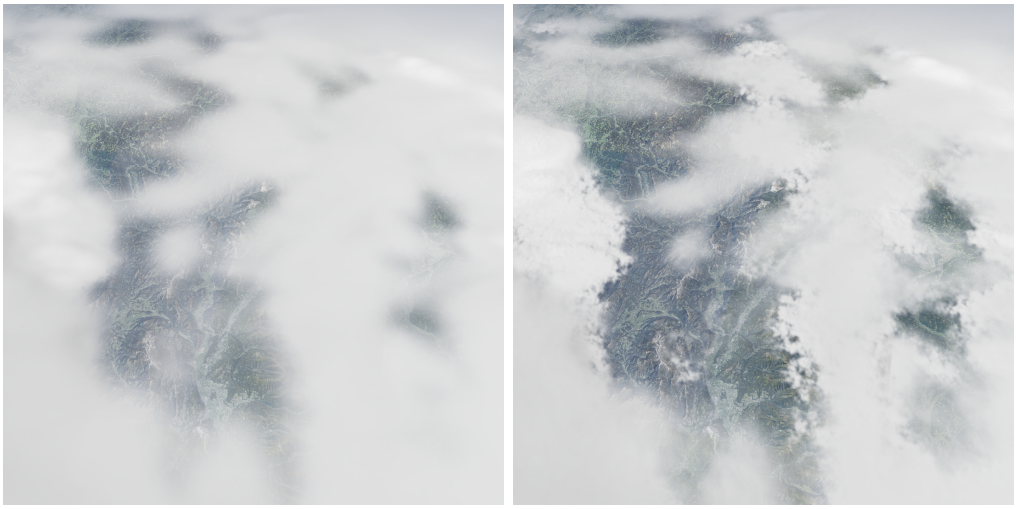
For liquid droplets, r_{eff} is derived from a fixed continental droplet concentration of $N = 200\,\mathrm{cm^{-3}}$ following Bower and Choularton [21], clamped to $[4, 15]\,\mu\mathrm{m}$. For ice crystals, the effective radius follows the Sun and Rikus [22] parameterization with the power-law revision of Sun [23], clamped to $[10, 150]\,\mu\mathrm{m}$.

Sub-Grid Detail Synthesis

ICON-D2 has a nominal horizontal resolution of approximately 2.2 km, which is too coarse to produce visually convincing cloud structure at the scales visible in a 3D map view. The synthesis shader augments the upsampled model fields with sub-grid detail through turbulent coordinate displacement and procedural noise. A Kelvin-Helmholtz billow displacement was also explored but ultimately not adopted.

Figure 3.3 illustrates the overall effect of the synthesis.

Kelvin-Helmholtz billowing. Kelvin-Helmholtz (KH) instability produces characteristic wave-like billows at wind-shear interfaces [24], frequently at scales below the ICON-D2 grid spacing. A shader-based synthesis was attempted, computing a Richardson number proxy from TKE and vertical wind shear, estimating billow wavelength from Michalke's linear stability result [25], and applying sinusoidal coordinate displacements along the shear direction. Two problems prevented a satisfactory result. First, the true Richardson number requires the Brunt-Väisälä frequency, which was not reliably computable from the available shader inputs; the TKE-based proxy introduced compounding errors. Second, the billow phase $\vec{p} \cdot \hat{n}$ is evaluated in Web Mercator coordinates of order 10^6 m, causing small changes in the shear direction $\hat{n}$ to produce incoherent phase values across adjacent voxels. The KH synthesis code is retained for reference; a more robust implementation would compute the billow field as an offline step where Richardson numbers can be evaluated directly from the full model grid.



(a) Upsampled ICON-D2 field. (b) After detail synthesis.

Figure 3.3: Horizontal cloud density slice before and after sub-grid detail synthesis. The synthesized field adds sub-kilometer structure while preserving the large-scale forecast pattern.

Turbulent displacement. TKE drives a coordinate displacement approximating sub-grid turbulent mixing. The turbulent velocity scale is estimated as $\sigma_h \approx \sqrt{\frac{2}{3}\text{TKE}}$ assuming isotropic turbulence. Three decorrelated fBm noise samples produce an independent displacement along each axis. The displacement amplitude scales with σ_h, normalized against a reference intensity corresponding to a TKE of $5\,\text{m}^2\,\text{s}^{-2}$, up to a maximum displacement of 500 m, motivated by the Lagrangian mixing length concept discussed in Rodean [14].

Procedural noise. After coordinate warping, a procedural noise field is evaluated at the displaced position to produce the density factor ρ_d. The noise adapts to the local cloud regime through a convective weight that blends between two synthesis paths. A base cumulus term $w_{\text{cu}} = \text{smoothstep}(0.05, 0.5, \text{TKE})$ fades with altitude via $1 - \text{smoothstep}(5000, 8000, z)$. A cumulonimbus bypass term $w_{\text{cb}} = \text{smoothstep}(0.8, 1.5, \text{TKE})$ overrides the altitude fade when TKE exceeds approximately $1\,\text{m}^2\,\text{s}^{-2}$, preserving deep convective structure at high altitudes. The final weight is $w = \max(w_{\text{cu}} \cdot w_{\text{alt}},\ w_{\text{cb}})$.

In the *stratus path* (low convective weight), gradient fBm provides smooth interior brightness variation. Edge perturbation is concentrated near cloud boundaries through a mask that peaks at CLC = 0.5.

In the *convective path* (high convective weight), the cloud footprint is shaped by gradient fBm at two scales (2200 m and 800 m dominant wavelength). Surface texture is derived from a product of three noise fields: gradient fBm at 1500 m scale (producing a web-like

modulation), Worley fBm at 600 m, and high-frequency Worley fBm at 150 m, producing the rounded cauliflower structure characteristic of cumulus clouds. The texture is blended toward a neutral value in the cloud core to avoid interior voids, and high TKE attenuates texture amplitude to model the breakdown of coherent structure in strongly turbulent conditions.

The two paths are blended by the convective weight. The blended perturbation is added to CLC to form a base shape, which is then passed through a saturate-and-smoothstep sequence:

$$\rho_{\text{shape}} = \text{smoothstep}(0,\ e,\ \text{clamp}(\text{CLC} + p,\ 0,\ 1)) \tag{3.5}$$

where p is the blended perturbation and e interpolates from 1.0 (stratus) to 0.5 (convective) with the convective weight. The outer clamp discards positive perturbations in dense cells, asymmetrically favoring erosion over growth and producing the fraying observed at convective cloud boundaries. The final density factor is $\rho_d = \rho_{\text{shape}} \cdot t$, where t is the blended texture value.

3.3.2 LOD Generation

The intermediate tiles from the synthesis stage all sit at zoom level 10. Table 3.2 summarizes the tile counts and texel resolutions across the full hierarchy; the vertical resolution of 219 m per texel is constant across all levels.

Zoom	Tiles	Tile width	Hor. texel size
4	1	2505 km	9.8 km
5	4	1252 km	4.9 km
6	6	626 km	2.4 km
7	12	313 km	1.2 km
8	28	157 km	611 m
9	84	78 km	306 m
10	336	39 km	153 m

Table 3.2: Tile counts and Web Mercator equatorial tile widths per zoom level. Horizontal texel sizes are equatorial values; at Austria's latitude ($\approx 47.7°$ N) they are smaller by $\cos(47.7°) \approx 0.67$.

The generator traverses the tile tree depth-first: starting from a root tile, it recursively requests its four children before computing its own value, so data always flows from leaves toward the root. This ordering keeps memory usage low, avoiding the pressure that a breadth-first traversal would create. At the leaf level, the intermediate binary files are loaded for processing. Parent tiles are produced by 2×2 mean pooling on the horizontal axes of the four child tiles, halving horizontal resolution at each zoom step while keeping vertical resolution fixed. Vertical resolution is not downsampled because the altitude range represented by each tile is constant regardless of zoom level, and reducing it would discard thin cloud layers that remain perceptually significant at small map scales.

Texture Encoding

Each tile is stored in the KTX2 container format [26], which natively supports mipmap chains and GPU-compressed texture formats. Tiles are compressed using BC4, which encodes a single-channel texture in 4×4 blocks at 4 bits per texel, chosen for its wide hardware support and suitability for single-channel data. Before compression, extinction values are scaled by $1/32$, mapping the physical range into $[0, 1]$; the renderer applies the inverse factor after sampling. Non-linear encoding was considered but is incompatible with hardware texture filtering, which decompresses and interpolates in a single step.

To support the ray-marcher's coarse MIP-level empty-space detection (Section 3.4.5), the compressor applies conservative quantization: any block containing at least one non-zero input encodes to a non-zero output, so that no cloud region is silently rounded away. This invariant propagates through the mipmap chain.

After BC4 encoding, the KTX2 container applies zstd supercompression for further lossless size reduction. File sizes are reported in Section 4.1.1.

3.3.3 Shadow Precomputation

Cloud shadows on the terrain are precomputed as a separate low-resolution texture assuming a vertical light direction. The shadow map is built from zoom 7 tiles stitched into a single volume. For each column, extinction is summed from top to bottom, and the altitude at which cumulative optical depth first exceeds $\tau = 2.0$ is recorded as a normalized shadow entry height $h \in [0, 1]$. To ensure compatibility with bilinear filtering, h is stored as an Exponential Shadow Map value [27] $e(h) = e^{ch}$ with $c = 4$, producing soft shadow edges without multi-sample lookups. A Gaussian blur with $\sigma = 1$ is applied before saving, and the result is stored as a single KTX2 file with `R16_SFLOAT` encoding and zstd supercompression. The vertical light simplification means shadow shape does not vary with sun position; computing sun-angle shadows is a natural direction for future work.

3.4 Runtime Renderer

The renderer runs as a WebGPU compute shader at half the output resolution, with one thread per pixel. Each thread casts a ray through the cloud volume and accumulates radiance and transmittance along it. The result is stored in two half-resolution textures: an `rgba16float` target whose RGB channels carry accumulated radiance and whose alpha channel carries transmittance, and an `r32float` target whose single channel carries a transmittance-weighted mean depth. The depth value is used by the TAAU pass for reprojection. Both targets are then upsampled by TAAU [28] before compositing with the main scene. The workgroup size is $8 \times 8 = 64$ threads, which fills one wavefront on AMD hardware and two warps on NVIDIA hardware, maximizing occupancy on both architectures.

3.4.1 Atlas and Tile Lookup

At render time the shader must sample an arbitrary number of tiles without knowing in advance which ones will be needed. WebGPU provides no 3D texture array type, and the number of distinct texture bindings per shader is limited (16 in Chrome, 64 in Firefox on the test system). A texture atlas is therefore used: all tiles are packed into a single 3D texture.

The atlas is a $4 \times 4 \times 32$ grid of 512 slots, each holding one tile at $256 \times 256 \times 64$ texels in BC4 format, with a full MIP chain per slot. The total atlas size is approximately 1.14 GiB, which exceeds the 1 GiB maximum reported by Firefox 148 but fits within the 2 GiB limit of Chrome 145; the application therefore requires a Chromium-based browser.

The 512-slot capacity accommodates all 336 zoom-10 tiles covering Austria. At runtime, tiles are loaded at the zoom level appropriate for the camera distance. The shader locates tiles through a CPU-maintained flat lookup table at zoom-10 resolution: each entry stores the atlas slot index of the tile covering that cell, or a sentinel. A tile at zoom z updates a $(2^{10-z})^2$ block of entries. Once the tile is identified, the sample position is projected into normalized tile-local coordinates and then into atlas texture coordinates accounting for the tile's slot offset. Coordinates are clamped by one texel inset from each face to prevent bleeding between adjacent slots.

3.4.2 Ray Setup and Terrain Occlusion

The ray origin $\mathbf{o}$ is the camera position. The ray direction $\hat{d}$ is reconstructed from the pixel's screen coordinates and the camera projection matrices. The terrain depth at the corresponding full-resolution pixel serves a separate purpose: unprojecting the terrain fragment into camera-relative world space yields a vector whose magnitude equals the terrain distance, used directly as t_{far}. Terrain therefore naturally occludes the ray with no per-step depth test. t_{near} is set by the AABB intersection with the cloud volume bounding box.

3.4.3 Step Size

The step size at distance t along the ray is:

$$\begin{aligned} \Delta t_{\text{distance}} &= \max(t/c_d,\, \Delta t_{\min}) \\ \Delta t_{\text{horizon}} &= (1 - |\hat{d}_z|)^2 \cdot c_h \\ \Delta t_{\text{length}} &= (t_{\text{far}} - t_{\text{near}})/32 \\ \Delta t &= \min(\Delta t_{\text{distance}} + \Delta t_{\text{horizon}},\, \Delta t_{\text{length}}) \end{aligned} \tag{3.6}$$

The base term grows linearly with distance, sampling nearby clouds finely and distant ones coarsely. The horizon bonus enlarges steps for near-horizontal rays, extending view distance at the cost of fine-grained sampling. The length-based cap ensures at least approximately 32 samples over the full ray. Default values are $\Delta t_{\min} = 125\,\text{m}$, $c_d = 50$, and $c_h = 1000\,\text{m}$.

3.4.4 Adaptive LOD

At zoom 10, each tile covers roughly 39 km horizontally with 256 texels (about 150 m per texel), while 64 texels span 14.0 km vertically (about 220 m per texel). The two axes have different resolution, so a single MIP level optimized for one would either oversample or undersample the other. This anisotropy worsens at lower zoom levels, since tile width doubles with each zoom step down while the vertical extent remains fixed.

The MIP level is therefore determined independently for each axis. For a given step, the number of texels traversed along the horizontal and vertical components of the ray is computed separately from the step size, the ray direction, and the tile's texel dimensions. The finer of the two requirements is used, with a bias of -0.5 to favor slightly higher detail.

Algorithm 3.1: Per-axis adaptive LOD selection.

Input: Step size Δt; ray direction $\hat{d}$; tile zoom z
Output: MIP level ℓ

$s_{xy} \leftarrow \Delta t \cdot \|\hat{d}_{xy}\| \,/\, \text{texelSize}_{xy}(z)$;
$s_z \leftarrow \Delta t \cdot |\hat{d}_z| \,/\, \text{texelSize}_z$;
$\ell_{xy} \leftarrow \log_2(\max(s_{xy}/2,\ 1))$;
$\ell_z \leftarrow \log_2(\max(s_z/2,\ 1))$;
return $\text{clamp}(\min(\ell_{xy},\ \ell_z) - 0.5,\ 0,\ 5)$

3.4.5 Ray-Marching

Sampling the full volume at a fine step size wastes the majority of samples on empty air. The renderer avoids this with a two-state march that alternates between a *search* phase and an *integration* phase.

In the *search phase* the ray takes coarse steps of $4\Delta t$ and samples the volume at $\ell + 3$. When a non-zero sample is found, the ray transitions to the integration phase *without advancing* t, so that integration always begins from inside the detected cloud.

In the *integration phase* the ray steps at Δt with the MIP level ℓ from Algorithm 3.1, recomputed each iteration from the current step size and ray direction. Where density is positive, radiance and transmittance are accumulated (see Section 3.4.6). Where density is zero, n_{empty} is incremented. The ray returns to the search phase only when two conditions are both met: $n_{\text{empty}} \geq 8$, and d_{fine} has been exhausted. d_{fine} is initialized to $4\Delta t$ on each entry into the integration phase, preventing a premature return to coarse search. The density is also multiplied by a camera fade weight $\text{fade}(t) \in [0, 1]$ before the density test; this is described in Section 3.4.7.

Algorithm 3.2 summarizes the complete march. Here $V(\mathbf{p},\ \ell)$ denotes trilinear sampling into the atlas at world position $\mathbf{p}$ and MIP level ℓ; the value returned is the extinction

coefficient in m^{-1}, obtained by multiplying the stored km^{-1} representation by 10^{-3}. All distances (t, Δt) are in metres throughout the march.

Algorithm 3.2: Two-state ray-marcher.

Input: Ray $(\mathbf{o}, \hat{d})$; interval $[t_{\text{near}}, t_{\text{far}}]$; jitter $\xi \in [0, 1)$
Output: Radiance L, transmittance T

$t \leftarrow t_{\text{near}}$; $L \leftarrow 0$; $T \leftarrow 1$; $n \leftarrow 0$;
search $\leftarrow$ true; $n_{\text{empty}} \leftarrow 0$; $d_{\text{fine}} \leftarrow 0$;
while $T > 0.01$ ***and*** $t < t_{\text{far}}$ ***and*** $n < 128$ **do**
  $\Delta t,\ \ell \leftarrow \text{stepSize}(t, \hat{d}),\ \text{lod}(\Delta t, \hat{d})$;
  if *search* **then**
    $\beta \leftarrow V\Big(\mathbf{o} + \hat{d}(t + 4\Delta t\,\xi),\ \ell + 3\Big)$;
    if $\beta > 0$ **then**
      search $\leftarrow$ false; $d_{\text{fine}} \leftarrow 4\Delta t$; $n_{\text{empty}} \leftarrow 0$;
    else
      $t \mathrel{+}= 4\Delta t$; $n \mathrel{+}= 1$;
    end
  else
    $\beta \leftarrow V\Big(\mathbf{o} + \hat{d}(t + \Delta t\,\xi),\ \ell\Big) \cdot \text{fade}(t)$;
    if $\beta > 0$ **then**
      $L \mathrel{+}= T \cdot \text{shade}(\beta, \ell, \Delta t)$; $T \mathrel{*}= \exp(-\beta\, s_e\, \Delta t)$; $n_{\text{empty}} \leftarrow 0$;
    else
      $n_{\text{empty}} \mathrel{+}= 1$;
    end
    $d_{\text{fine}} \mathrel{-}= \Delta t$;
    if $d_{\text{fine}} \le 0$ ***and*** $n_{\text{empty}} \ge 8$ **then**
      search $\leftarrow$ true;
    end
    $t \mathrel{+}= \Delta t$; $n \mathrel{+}= 1$;
  end
end
return $(L,\ T)$

A coarse hit does not increment n, since the transition re-enters the loop at the same t. Only steps that advance the ray consume the budget of 128.

3.4.6 Lighting

Since existing parts of weBIGeo do not use physically based rendering, scene radiance values are in arbitrary units. The sun radiance is therefore scaled by an empirically determined factor of 800, chosen so that cloud brightness matches the surrounding scene under typical midday illumination.

Since both the view direction and the sun direction are constant for a given pixel, the phase function is evaluated once before the march begins and reused at every sample.

Phase function. The fraction of light scattered toward the camera depends on the angle θ between the view and sun directions. It is modeled with a three-term function:

$$p(\theta) = 0.5 \cdot p_{\mathrm{HG}}(\theta,\, g) + 0.2 \cdot p_{\mathrm{HG}}(\theta,\, -g/2) + \frac{0.3}{4\pi} \tag{3.7}$$

where p_{HG} is the Henyey-Greenstein phase function [8] and $g = 0.7$ is the anisotropy coefficient. The forward lobe produces the silver-lining effect when the view direction is close to the sun. The weak backward lobe adds a slight brightening on the opposite side. The isotropic term prevents cloud regions facing away from the sun from going completely dark. The weights follow the Nubis pipeline [29].

Solar transmittance. At each sample point with positive density, a secondary ray is cast toward the sun and accumulates optical depth τ over eight steps. The step size grows geometrically by a factor of 1.5 per iteration, with the initial size derived from the geometric series sum so that the full sequence spans exactly the distance to the volume boundary. The MIP level increases along the ray to compensate for increasingly sparse sampling, following the cone-sampling approach introduced in the Nubis pipeline [10]. To prevent clouds from appearing excessively dark, the extinction coefficient is reduced by a configurable factor during the sun ray march.

From the accumulated τ, two transmittance estimates are combined. The primary estimate blends Beer–Lambert attenuation with a powder sugar term [10] that approximates the increased extinction visible at cloud edges when backlit:

$$\begin{aligned} T_{\mathrm{beer}} &= e^{-\tau} \\ T_{\mathrm{powder}} &= 1 - e^{-2\tau} \\ E_{\mathrm{prim}} &= \mathrm{mix}(T_{\mathrm{beer}},\; 2\,T_{\mathrm{beer}} \cdot T_{\mathrm{powder}},\; w(\theta)) \end{aligned} \tag{3.8}$$

where $w(\theta) = s_p \cdot \mathrm{smoothstep}(0.5, -0.5, \cos\theta)$ strengthens the powder effect when looking away from the sun, scaled by a user-configurable intensity parameter s_p. The secondary estimate approximates multiple scattering [29] with a softer attenuation curve, gated by a directional factor that activates it when looking toward the sun:

$$E_{\mathrm{sec}} = 0.7\, e^{-0.25\,\tau} \tag{3.9}$$

This ensures deep cloud interiors remain partially lit. The final solar energy is $E_{\mathrm{sun}} = \max(E_{\mathrm{prim}},\; E_{\mathrm{sec}})$.

Ambient and atmospheric light. Ambient sky light E_{amb} is a constant term representing uniform illumination from the sky hemisphere. It is modulated by a height-dependent factor that darkens low-altitude samples and brightens cloud tops. When the

platform's atmosphere model is active, the local Rayleigh in-scattering at the sample altitude is evaluated as an atmospheric tint E_{atm}. The ambient light is interpolated toward this tint to control saturation. Unlike sunlight, this contribution is diffuse and not multiplied by the phase function.

Accumulation. The radiance contribution from one integration step is:

$$\Delta L = T \cdot \beta\, s_e\, \alpha \cdot \left(E_{\text{sun}}\, p(\theta) + E_{\text{amb}}\right) \cdot \Delta t \tag{3.10}$$

where β is the extinction coefficient returned by V (in m^{-1}), s_e is a user-configurable extinction scale factor and α is the single-scattering albedo. Transmittance is then updated as $T \leftarrow T \cdot \exp(-\beta\, s_e\, \Delta t)$.

3.4.7 Camera Fade

Clouds close to the camera obstruct the terrain beneath the viewpoint. To prevent this, cloud density is attenuated within a fade zone around the camera. The fade weight is based on an effective distance d_{fade} from the camera to the sample point $\mathbf{p}$:

$$d_{\text{fade}} = \max(\|\mathbf{p}_{xy} - \mathbf{o}_{xy}\|,\ 0.5\,(\mathbf{o}_z - \mathbf{p}_z)) \tag{3.11}$$

The horizontal term targets clouds that are laterally close to the camera. The vertical term targets clouds that lie well below the camera, which is common when viewing alpine terrain from above. The weight increases cubically from zero at $r_{\text{near}} = 1\,\text{km}$ to one at $r_{\text{far}} = 50\,\text{km}$, both scalable by a tunable parameter. Because r_{near} is also used as the ray's minimum t, no samples are taken inside the fade zone at all.

3.4.8 Temporal Jitter and TAAU Compatibility

Because the renderer runs at half resolution, poor jitter distribution causes ghosting or under-convergence in the TAAU pass. The per-pixel jitter ξ combines two components. Spatially, Interleaved Gradient Noise (IGN) [30] provides a smooth, non-repeating distribution within each frame without requiring a texture fetch. Temporally, the R1 low-discrepancy sequence [31], based on the golden ratio, shifts the IGN pattern by a different amount each frame: $\xi = \text{fract}(\text{IGN}(x, y) + r_f)$ where r_f is the R1 value for frame f. This ensures TAAU accumulates non-redundant samples across frames.

3.4.9 Temporal Accumulation and Upscaling

The accumulation pass reads the current half-resolution cloud render and reprojects a full-resolution history buffer from the previous frame, blending the two after variance clipping. The output is a full-resolution `rgba16float` texture with blended pre-multiplied radiance (RGB) and transmittance (alpha), passed directly to compositing. Because the current frame is rendered with a sub-pixel jitter and the history buffer is maintained at full resolution, successive frames progressively fill in sub-pixel information, converging to a full-resolution image without an explicit upsampling step. Algorithm 3.3 summarizes the per-pixel operations.

Algorithm 3.3: Per-pixel temporal accumulation and upscaling.

Input: Current sample $(\hat{L}_c, T_c)$; depth d; history buffer; camera matrices
Output: Blended output $(\hat{L}_b, T_b)$
`// Samples are stored pre-multiplied:` $\hat{L} = L \cdot (1 - T)$

if $d \leq 0$ ***or*** *reprojected UV* $\notin [0,1]^2$ **then**
  return $(\hat{L}_c, T_c)$;
end
$L_c \leftarrow \hat{L}_c \,/\, \max(1 - T_c,\ \epsilon)$;
Reconstruct world position from d; reproject to previous frame $\rightarrow$ history UV;
Sample history $(\hat{L}_h, T_h)$ at history UV; $L_h \leftarrow \hat{L}_h \,/\, \max(1 - T_h,\ \epsilon)$;
Gather 3×3 neighborhood $\rightarrow$ compute (μ_L, σ_L), (μ_T, σ_T);
Compute clip weight w_{clip} from history–current divergence (floor 0.5);
$T_h^{\text{raw}} \leftarrow T_h$;
Clip L_h to $[\mu_L - \sigma_L,\ \mu_L + \sigma_L]$; Clip T_h to $[\mu_T - \sigma_T,\ \mu_T + \sigma_T]$;
$\alpha \leftarrow 0.95 \cdot w_{\text{clip}}$;
if $T_h^{\text{raw}} > 0.999$ ***and*** $T_c < 0.95$ **then**
  $\alpha \leftarrow 0$; `// Disocclusion: discard history`
end
$L_b \leftarrow (1 - \alpha)\, L_c + \alpha\, L_h$;
$T_b \leftarrow \exp\big(-\big((1 - \alpha)\,(-\log T_c) + \alpha\,(-\log T_h)\big)\big)$;
$\hat{L}_b \leftarrow L_b \cdot (1 - T_b)$;
return $(\hat{L}_b,\ T_b)$

Reprojection and variance clipping. Each output pixel reconstructs its world-space position from the current depth and camera matrices, then projects it into the previous frame's clip space to obtain a history UV. Ghost artifacts from disocclusion or fast camera motion are suppressed by constraining the history sample to a $[\mu - \sigma,\ \mu + \sigma]$ AABB derived from a 3×3 neighborhood of the current frame [28]. Transmittance is blended in optical depth space ($\tau = -\log T$) rather than linearly, which respects the exponential nature of extinction and avoids biasing toward more transmissive samples.

Pixel jitter. The low-resolution render is shifted by a sub-pixel offset cycling through a 2×2 half-pixel grid pattern, independent of the per-sample ray jitter ξ described in Section 3.4.8.

Limitations. The half-resolution input creates tension between ghosting prevention and convergence: aggressive clipping suppresses ghosting but slows convergence in noisy regions, while permissive clipping allows ghosting. The current parameters produce acceptable results under typical navigation but do not fully resolve this tradeoff. A more sophisticated approach, such as adaptive clipping or cloud-aware reprojection, would improve visual stability and is a direction for future work.

3.4.10 Compositing

The cloud layer is composited onto the terrain inside weBIGeo's deferred shading pass. The TAAU output is converted from pre-multiplied to straight RGB, then tonemapped with a per-channel Reinhard operator [32] ($c' = c/(c+1)$) to match the existing scene's range. If the atmosphere model is active, cloud pixels beyond 300 km are progressively faded toward the atmospheric color (full blend at 600 km) as an artistic distance fog. The result is composited over the terrain with $\alpha = 1 - T$.

CHAPTER 4

Results

All measurements reported in this chapter were performed on the system described in Table 4.1, which was used for both the preprocessing compute stages and the runtime rendering measurements.

Component	Specification
CPU	AMD Ryzen 7 9700X
RAM	16 GB
GPU	NVIDIA GeForce RTX 5070 Ti
OS	Windows 10
WebGPU backend	Dawn (native)
Output resolution	2560×1440
Volumetric pass resolution	1280×720

Table 4.1: Test system configuration.

This chapter evaluates the pipeline from three perspectives. Section 4.1 characterizes the preprocessing stage in terms of output storage requirements and execution time. Section 4.2 reports the rendering performance of the cloud layer at runtime. Section 4.3 presents a visual comparison of the rendered output against webcam and satellite imagery.

4.1 Preprocessing Performance

4.1.1 Output File Sizes

Table 4.2 reports the tile counts and storage footprint of the TMS hierarchy produced for a single forecast timestamp. Uncompressed sizes assume `float16` storage at full voxel resolution. Compressed sizes are after BC4 encoding, which reduces the data to 0.5

bytes per texel, and subsequent zstd supercompression. The total compressed hierarchy amounts to approximately 86 ± 54 MiB on average, which is small enough to be served and streamed without imposing a significant storage or bandwidth burden.

Zoom	Count	Raw [MiB]	Compressed [MiB]
4	1	9.1	0.1 ± 0.0
5	4	36.6	0.2 ± 0.1
6	6	54.9	0.6 ± 0.2
7	12	109.7	1.9 ± 0.9
8	28	256.0	6.2 ± 3.1
9	84	768.0	19.3 ± 11.1
10	336	3 072.0	58.1 ± 38.8
Total	471	4 306.3	86.3 ± 53.8

Table 4.2: Tile counts and compressed storage per zoom level for one forecast timestamp.

4.1.2 Execution Time

Download. ICON-D2 forecast files are fetched from the DWD open-data server over HTTPS: 522 GRIB2 files totaling approximately 310 ± 180 MiB per model run. Table 4.3 gives per-variable size statistics.

Category	Variable(s)	$\mu \pm \sigma$ [KiB]	Empty
Dense	qv, t, p	917.1 ± 91.2	—
Sparse	tke	528.8 ± 146.4	—
	clc	274.9 ± 162.6	15.5 %
	qc	67.4 ± 41.7	54.8 %
	qi	220.0 ± 244.4	4.6 %
Static	hhl	629.0 ± 163.4	—
	All (used)	608.5 ± 353.6	—

Table 4.3: ICON-D2 GRIB2 file-size statistics per variable group. Empty files (< 10 KiB) are excluded from the fit but their fraction is reported.

Compute stages. Detail synthesis and LOD generation each take around 12 s and together dominate the compute time, at 11.6 ± 1.3 s and 12.4 ± 0.7 s respectively. Data loading contributes 7.9 ± 0.6 s, and shadow map generation is negligible at 0.8 ± 0.2 s. The total compute time, excluding the network-dependent download, is 32.5 ± 2.7 s ($n = 9$ runs on the system described in Table 4.1).

4.2 Runtime Performance

4.2.1 Test Setup

The application was run in native mode using the Dawn WebGPU backend rather than inside a browser, which avoids the overhead of the JavaScript runtime and removes browser-imposed memory limits. The renderer and all WebGPU constraints described in Chapter 3 apply equally in native mode, since the same WGSL shaders and WebGPU API are used in both configurations. The native backend was chosen solely to simplify profiling and remove confounding variables.

The tile server was hosted locally on the same machine, eliminating network latency as a variable. Tile loading performance is therefore not representative of a real deployment over a network, but this does not affect the rendering measurements, which assume tiles are already resident in the atlas.

Four test scenarios were defined to cover a range of cloud coverage conditions: Scenario 1 features medium coverage, Scenario 2 features high coverage, Scenario 3 features full coverage, and Scenario 4 features almost no coverage. For each scenario, rendering time was measured over a large number of frames both with and without the cloud layer active, allowing the cost attributable to the cloud system to be isolated.

4.2.2 GPU Frame Time

The application produces frames in response to user input rather than at a fixed rate, so the relevant target is interactivity: a frame time below 33 ms (30 fps). Figure 4.1 shows the GPU frame time distribution across scenarios. All comparisons between cloud and no-cloud conditions are highly significant (Mann–Whitney U test, $p < 0.001$ in every scenario). Even in the worst case (Scenario 2, high coverage), mean GPU time with clouds is 2.25 ms, well within the 33 ms budget.

Frame time varies with scene content as expected. Notably, Scenario 3 (full coverage) is faster than Scenario 2 (high coverage): in a fully overcast scene, rays reach the minimum transmittance threshold and terminate early, whereas patchy cover requires traversing longer stretches of empty space. Scenario 4 (almost no coverage) isolates the fixed cost of the cloud system at approximately 0.29 ms, incurred by the coarse-step search loop and the TAAU pass running unconditionally.

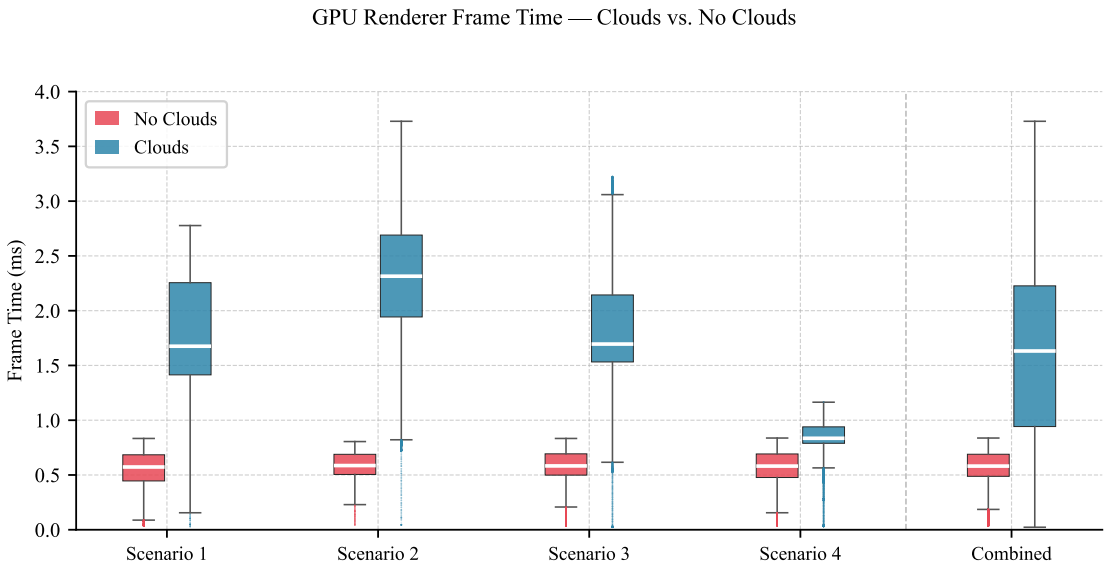


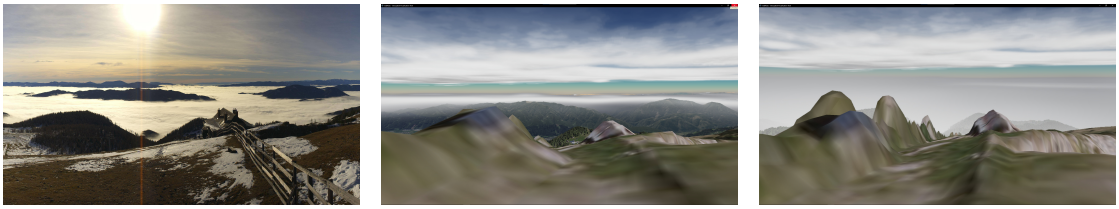
Figure 4.1: Frame time distribution of the GPU renderer timer across scenarios, comparing cloud vs. no-cloud rendering. Boxes show the interquartile range, whiskers extend to 1.5 IQR, and the white line marks the median.

4.3 Qualitative Results

The rendered output was evaluated against ground-level webcam imagery and EUMETSAT true-color satellite composites [33], using matched UTC timestamps. Webcam views were reproduced by positioning the renderer camera to approximately match each webcam's location and direction.

4.3.1 Webcam Comparison

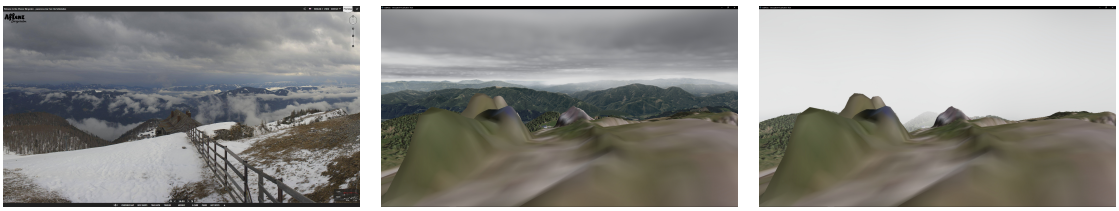
Figures 4.2–4.4 show three webcam timestamps alongside the corresponding renderer output, each representing a distinct cloud situation: dense valley fog, a general overcast with low-level puffs, and well-defined convective cumulus. Each timestamp is shown in two renderer views. The default-fade variant matches normal application behavior, where clouds near the camera are gradually suppressed to keep the surrounding terrain visible. Because this suppression is not present in reality, the minimum-fade variant is also included: it shows the unattenuated cloud volume and therefore provides a more honest basis for visual comparison with the reference photograph.



(a) Webcam (b) weBIGeo, default fade (c) weBIGeo, minimum fade

Figure 4.2: Webcam comparison, Aflenz, 2025-12-09 09:00 UTC. Webcam image courtesy of Aflenzer Bürgeralm, retrieved via bergfex.at.

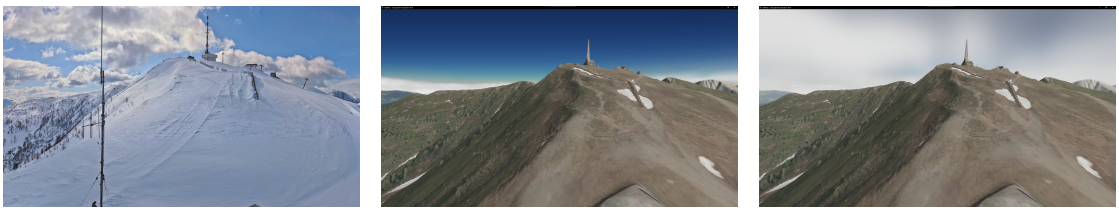
The webcam shows dense fog filling the valley below the surrounding peaks. In the rendered output the cloud mass extends past the peak, demonstrating that the coarse vertical resolution of ICON-D2 cannot resolve sharp fog layer boundaries.



(a) Webcam (b) weBIGeo, default fade (c) weBIGeo, minimum fade

Figure 4.3: Webcam comparison, Aflenz, 2026-03-16 17:00 UTC. Webcam image courtesy of Aflenzer Bürgeralm, retrieved via bergfex.at.

The webcam shows a general overcast layer with smaller, wispy puffs at lower altitude. The default-fade view captures the broad overcast reasonably well, but the lower puffs are not reproduced as distinct structures—the sub-grid density distribution within each voxel is not accurate enough to preserve the character of small cloud elements.



(a) Webcam (b) weBIGeo, default fade (c) weBIGeo, minimum fade

Figure 4.4: Webcam comparison, Goldegg, 2026-03-17 09:00 UTC. Webcam image courtesy of WMS WebMediaSolutions, retrieved via bergfex.at.

The webcam shows well-defined cumulus clouds with sharp edges and visible internal structure. The renderer captures clouds at roughly the correct location, but they appear

as smooth blobs: at $256 \times 256 \times 64$ voxels per tile, the resolution is too coarse to encode fine surface texture.

Across all three examples, general cloud presence is broadly consistent with the reference, but the spatial accuracy is insufficient at the scale of a single viewpoint. These limitations follow directly from the resolution of the ICON-D2 source data and the tile encoding.

4.3.2 Satellite Comparison

Figure 4.5 shows four timestamps comparing EUMETSAT true-color composites against a top-down renderer view of the same domain and time.

The top-down satellite view is more directly comparable to the spatial structure of the model data, and the agreement is considerably closer than in the webcam case. Large-scale cloud systems and the broad pattern of clear and cloudy regions are reproduced with reasonable fidelity.

One consistent discrepancy is coverage overestimation in some regions, likely a consequence of the $\mathrm{QC} + \mathrm{QI}$ reconstruction scheme producing non-negligible extinction where CLC remains non-zero but microphysics has depleted condensate.

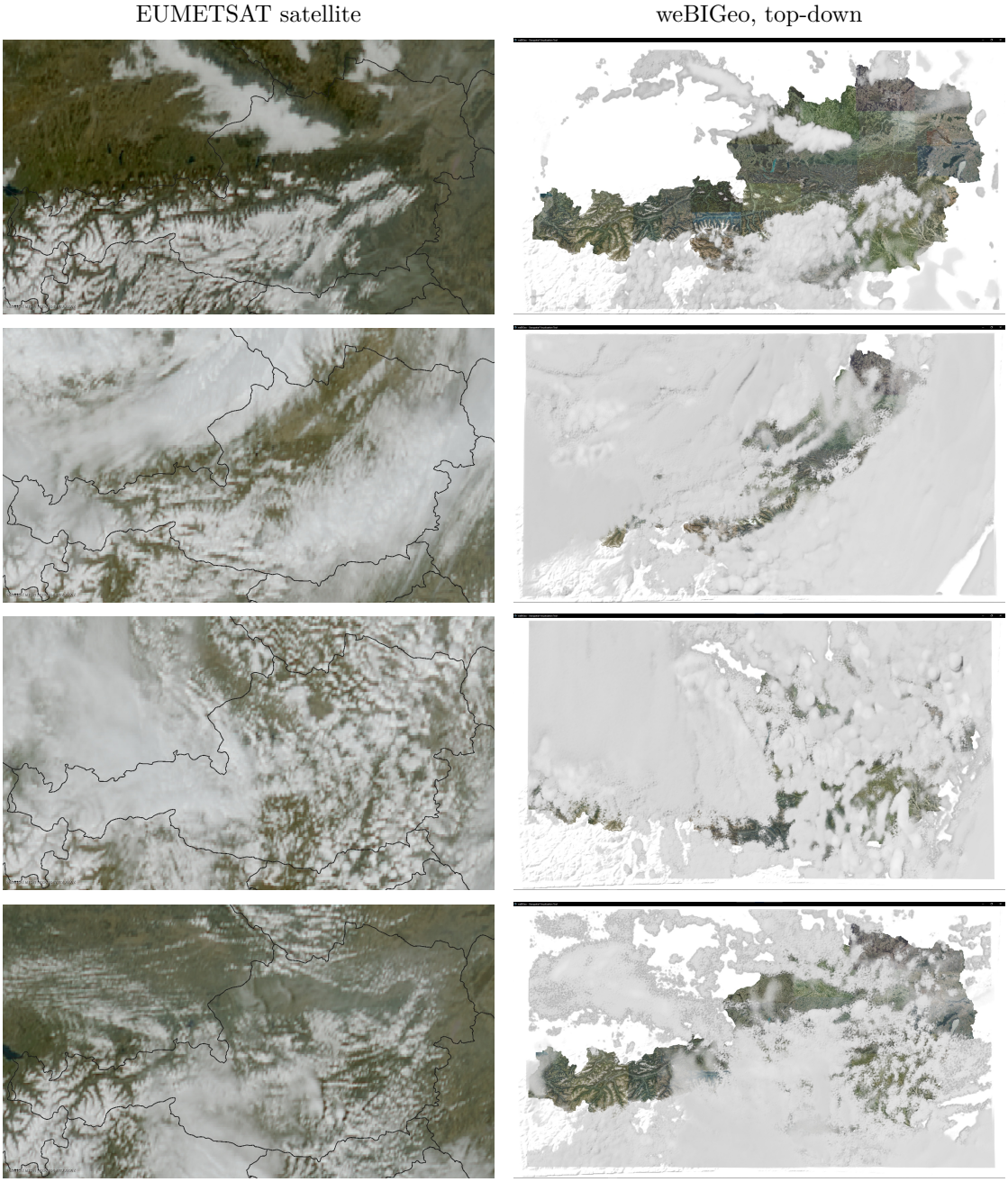


Figure 4.5: Satellite comparison at four timestamps (top to bottom): 2026-03-13 09:00, 2026-03-16 12:00, 2026-03-17 12:00, and 2026-03-18 13:00 UTC. Satellite images © EUMETSAT.

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

This thesis presented a pipeline for the real-time volumetric rendering of meteorological cloud data within the weBIGeo geographic visualization platform. The pipeline transforms hourly ICON-D2 forecast data into a streamable TMS tile hierarchy of KTX2 textures, which a WebGPU ray-marching renderer samples at interactive frame rates. To the author's knowledge at the time of writing, no comparable system renders the full volumetric structure of operational numerical weather prediction output in a web-based 3D terrain environment.

A motivating use case was to let a hiker judge whether mountain peaks sit above or below the cloud layer. The 2.2 km horizontal resolution of ICON-D2 proved too coarse for this purpose, and the procedural detail synthesis, while visually plausible, is not observationally constrained. The system is better understood as a large-scale atmospheric context layer than as a precise local forecast tool.

Within that scope the system performs well. Preprocessing completes in approximately 33 s per timestamp, producing roughly 86 MiB of compressed tiles. The cloud pass consumes around 2.25 ms of GPU time even under high coverage. Satellite comparison shows reasonable fidelity at large scales; webcam comparison reveals three concrete limitations: insufficient vertical resolution for sharp fog boundaries, loss of small cloud element character, and too-coarse tile resolution for cumulus surface texture.

The three research questions are answered as follows. First, PCHIP vertical interpolation onto a uniform metric grid produces continuous volumetric fields from the non-uniformly spaced model levels, while TKE-driven coordinate displacement and regime-adaptive procedural noise add sub-grid structure — visually plausible, though the result is not physically accurate at fine scales, and coverage overestimation from the condensate reconstruction remains a limitation at larger scales too. Second, the coarse, physics-unit NWP

output must be transformed into the regular, compressed, GPU-native format required by a real-time renderer: this demands regridding, condensate consistency correction, extinction computation, LOD generation by mean pooling, shadow precomputation, and BC4 + zstd encoding into a TMS hierarchy. Third, distance- and altitude-dependent camera fades attenuate cloud density near the viewpoint, preventing obstruction of the terrain while preserving the atmospheric context layer at range.

A practical limitation is that the 1.14 GiB texture atlas exceeds Firefox's WebGPU maximum, currently restricting the system to Chromium-based browsers.

Overall, the system provides a solid foundation for an approximate large-scale overview of the atmospheric cloud state, and opens a concrete path toward richer meteorological context in web-based geographic visualization.

5.2 Future Work

Several directions for improvement follow directly from the limitations identified in this work.

Higher-resolution input data. The most significant limitation is source data coarseness. Higher-resolution NWP output for Austria does not appear to be publicly available; a dedicated simulation at, e.g., 500 m grid spacing would improve accuracy but requires significant infrastructure. Alternatively, constraining detail synthesis with real observations (radar or satellite) rather than TKE alone could help.

Machine learning detail synthesis. A data-driven super-resolution approach trained on pairs of high- and low-resolution cloud fields from convection-permitting simulations could replace the hand-crafted procedural noise with more physically plausible sub-grid detail.

Improved ray-march shading. Applying seamless high-resolution noise textures during ray marching, as in the Nubis pipeline, would allow surface detail that scales continuously with distance, removing the current dependence on tile resolution.

Coverage overestimation. The QC + QI reconstruction scheme introduces extinction where microphysics has depleted condensate. Tuning Q_{crit} or adopting a more physically grounded scheme could reduce this bias.

Kelvin-Helmholtz billowing. A more robust implementation would compute the billow field offline, where Richardson numbers can be evaluated accurately and globally coherent phases defined without floating-point precision issues.

Sun-angle shadows and temporal animation. Computing the shadow map along the actual sun vector would produce more plausible terrain shadowing. Blending between consecutive forecast timestamps would allow smooth temporal evolution of the cloud layer.

Temporal accumulation. The current TAAU implementation does not fully resolve the tension between ghosting prevention and convergence speed, particularly during fast camera motion. A more principled treatment of the semi-transparent medium in the reprojection step is a natural direction for future work.

Pre-baked atmospheric shading. The current shadow map captures only vertical sunlight attenuation through the cloud column. A natural extension would be to precompute richer shading information—such as warm reddish tints and elongated shadows at low sun angles—parameterized by sun elevation and broad cloud state, and composited at render time without additional per-frame cost.

Adaptive zoom cap for browser compatibility. The 3D texture atlas ($\approx 1.14\,\mathrm{GiB}$) exceeds Firefox's WebGPU memory limit, requiring a Chromium-based browser. Querying available GPU memory at startup and capping the maximum loaded zoom level accordingly would restore Firefox compatibility by substituting coarser tiles on memory-constrained devices.

Overview of Generative AI Tools Used

Various AI tools were used throughout this thesis project. Claude (Opus 4.6, Sonnet 4.6, Sonnet 4.5) and Gemini (3.1 Pro, 3.0 Pro, 2.5 Pro, 3 Flash) were the primary tools used across all phases of work. This included exploring the research problem, evaluating potential solution approaches and algorithmic trade-offs, understanding technical concepts, debugging, and implementation. GPT-5.2, Grok 4.1, and GitHub Copilot 0.37.8 were additionally used for research support and code completion.

For writing, Claude and Gemini were used to proofread text and improve sentence structure and clarity.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] M. Rautenhaus, M. Kern, A. Schäfler, and R. Westermann, "Three-dimensional visualization of ensemble weather forecasts – part 1: The visualization tool met.3d (version 1.0)," *Geoscientific Model Development*, vol. 8, no. 7, pp. 2329–2353, 2015.

[2] P. Komon, G. Kimmersdorfer, A. Celarek, and M. Waldner, "Data-driven compute overlays for interactive geographic simulation and visualization," in *2025 IEEE Visualization and Visual Analytics (VIS)*, pp. 186–190, IEEE, 2025.

[3] A. Celarek, L. Dworschak, G. Kimmersdorfer, J. Lindner, P. Komon, J. Maier, and M. Rampp, "Alpinemaps.org." `https://github.com/AlpineMapsOrg/renderer`, 2022. Accessed: 2026-02-21.

[4] G. Zängl, D. Reinert, P. Rípodas, and M. Baldauf, "The ICON (icosahedral non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core," *Quarterly Journal of the Royal Meteorological Society*, vol. 141, no. 687, pp. 563–579, 2015.

[5] S. Li, S. Jaroszynski, S. Pearse, L. Orf, and J. Clyne, "VAPOR: A visualization package tailored to analyze simulation data in earth system science," *Atmosphere*, vol. 10, no. 9, p. 488, 2019.

[6] S. Pearse, "Visualizing WRF data in vapor 3.1." YouTube, `https://www.youtube.com/watch?v=CtXHdI4WUDE`, 2019. Accessed: 2026-03-17.

[7] J. Kajiya and B. von Herzen, "Ray tracing volume densities," *ACM SIGGRAPH Computer Graphics*, vol. 18, pp. 165–174, 07 1984.

[8] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, 1995.

[9] M. J. Harris and A. Lastra, "Real-time cloud rendering," *Computer Graphics Forum*, vol. 20, no. 3, pp. 76–85, 2001.

[10] A. Schneider and N. Vos, "The real-time volumetric cloudscapes of Horizon: Zero Dawn," in *ACM SIGGRAPH Courses: Advances in Real-Time Rendering in Games*, ACM, 2015.

[11] A. Schneider, "Nubis: Real-time volumetric cloudscapes in a nutshell." Eurographics 2018 Talk, https://www.guerrilla-games.com/read/nubis-realtime-volumetric-cloudscapes-in-a-nutshell, 2018.

[12] A. Schneider, "Nubis, evolved: Real-time volumetric clouds for skies, environments, and VFX," in *ACM SIGGRAPH Courses: Advances in Real-Time Rendering in Games*, ACM, 2022.

[13] F. N. Fritsch and R. E. Carlson, "Monotone piecewise cubic interpolation," *SIAM Journal on Numerical Analysis*, vol. 17, no. 2, pp. 238–246, 1980.

[14] H. C. Rodean, *Stochastic Lagrangian models of turbulent diffusion*, vol. 45. Springer, 1996.

[15] S. Worley, "A cellular texture basis function," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pp. 291–294, Association for Computing Machinery, 1996.

[16] K. Perlin, "An image synthesizer," *SIGGRAPH Comput. Graph.*, vol. 19, pp. 287–296, July 1985.

[17] J. A. Croci, C. Skorski, J. Rauscher, R. Pajarola, and A. Diehl, "Trailview: An interactive 3d hike planning tool," in *Proceedings of the 30th International Conference on 3D Web Technology*, pp. 1–10, ACM, 2025.

[18] W3C GPU for the Web Working Group, "WebGPU." https://www.w3.org/TR/webgpu/, 2024. Accessed: 2026-02-23.

[19] A. Klein and contributors, "wgpu-py: Next generation GPU API for Python." https://github.com/pygfx/wgpu-py, 2024. Version v0.28.1. Published via PyPI.

[20] A. M. Tompkins, "Cloud parametrization," in *Proc. ECMWF Seminar on Parametrization of Subgrid Physical Processes*, pp. 27–62, 2008.

[21] K. N. Bower and T. W. Choularton, "A parameterisation of the effective radius of ice free clouds for use in global climate models," *Atmospheric research*, vol. 27, no. 4, pp. 305–339, 1992.

[22] Z. Sun and L. Rikus, "Parametrization of effective sizes of cirrus-cloud particles and its verification against observations," *Quarterly Journal of the Royal Meteorological Society*, vol. 125, no. 560, pp. 3037–3055, 1999.

[23] Z. Sun, "Reply to comments by Greg M. McFarquhar on 'parametrization of effective sizes of cirrus-cloud particles and its verification against observations'. (October B, 1999, 125, 3037–3055)," *Quarterly Journal of the Royal Meteorological Society*, vol. 127, no. 571, pp. 267–271, 2001.

[24] S. A. Thorpe, "Experiments on the instability of stratified shear flows: miscible fluids," *Journal of Fluid Mechanics*, vol. 46, no. 2, pp. 299–319, 1971.

[25] A. Michalke, "On the inviscid instability of the hyperbolic-tangent velocity profile," *Journal of Fluid Mechanics*, vol. 19, no. 4, pp. 543–556, 1964.

[26] The Khronos Group, "Ktx file format specification version 2.0." `https://github.com/KhronosGroup/KTX-Specification`, 2021. Accessed: 2026-02-22.

[27] T. Annen, T. Mertens, H.-P. Seidel, E. Flerackers, and J. Kautz, "Exponential shadow maps.," in *Graphics Interface*, pp. 155–161, ACM press, 2008.

[28] L. Yang, S. Liu, and M. Salvi, "A survey of temporal antialiasing techniques," in *Computer graphics forum*, vol. 39, pp. 607–621, Wiley Online Library, 2020.

[29] A. Schneider and N. Vos, "Nubis: Authoring real-time volumetric cloudscapes with the Decima engine," in *ACM SIGGRAPH Courses: Advances in Real-Time Rendering in Games*, ACM, 2017.

[30] J. Jimenez, "Interleaved gradient noise." `https://www.iryoku.com/next-generation-post-processing-in-call-of-duty-advanced-warfare/`, 2014. Accessed: 2026-02-22.

[31] M. Roberts, "The unreasonable effectiveness of quasirandom sequences." `https://extremelearning.com.au/unreasonable-effectiveness-of-quasirandom-sequences/`, 2018. Accessed: 2026-02-22.

[32] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic tone reproduction for digital images," *ACM Trans. Graph.*, vol. 21, pp. 267–276, July 2002.

[33] EUMETSAT, "Meteosat SEVIRI true color RGB imagery." `https://www.eumetsat.int`.