https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Atmospheric Measurement Techniques Discussions

ampycloud: an algorithm to characterize cloud layers above aerodromes using ceilometer measurements

Frédéric P.A. Vogt[1], Loris Foresti[2], Daniel Regenass[1], Néstor Tarin Burriel[3], Mervyn Bibby[3], Przemysław Juda[2], Simone Balmelli[2], Tobias Hanselmann[3], and Pieter du Preez[3]

[1]Federal Office of Meteorology and Climatology - MeteoSwiss, Chemin de l'Aérologie 1, 1530 Payerne, Switzerland.
[2]Federal Office of Meteorology and Climatology - MeteoSwiss, Via ai Monti 146a, 6605 Locarno-Monti, Switzerland.
[3]Federal Office of Meteorology and Climatology - MeteoSwiss, Operation Center 1, 8058 Zurich-Airport, Switzerland.

Correspondence: Frédéric P.A. Vogt (frederic.vogt@meteoswiss.ch)

Abstract. Ceilometers are used routinely at aerodromes world-wide to measure the height and sky coverage fraction of cloud layers. This information, possibly combined with direct observations by human observers, contributes to the production of Meteorological Aerodrome Reports (METARs). Here, we present ampycloud, a new algorithm and associated Python package for automatic processing of ceilometer measurements, with the aim to characterize cloud layers above aerodromes. The ampycloud algorithm has been developed at the Swiss Federal Office of Meteorology and Climatology (MeteoSwiss) as part of the AMAROC (AutoMETAR/AutoReport rOund the Clock) program, to fully automate the creation of METARs at Swiss civil aerodromes. ampycloud is designed to work with no (direct) human supervision. The algorithm consists of three distinct, sequential steps that rely on agglomerative clustering methods and Gaussian mixture models to identify distinct cloud layers. The robustness of the ampycloud algorithm stems from the first processing step, simple and reliable. It constrains the two subsequent processing steps that are more sensitive, but also better suited to handle complex cloud distributions. The software implementation of the ampycloud algorithm takes the form of an eponym, pip-installable Python package developed on Github. The code is made accessible to the general community as an open-source software under the terms of the 3-Clause BSD license.

1 Introduction

Ceilometers are being used at numerous aerodromes worldwide to measure autonomously the height and sky coverage fraction of cloud layers (i.e. the cloud amount; Wauben et al., 2006; ICAO, 2011; de Haij et al., 2016). These parameters form an essential component of METerological Aerodrome Reports (METARs; WMO, 2022). These compact telegrams provide a detailed description of the current meteorological conditions on and above the aerodrome ground to pilots, air traffic controllers, and aerodrome safety services. Regulations from the International Civil Aviation Organization (ICAO) and the corresponding European Commission of Implementing Regulation (CIR, EU373) dictate that METARs should be issued at 30 min intervals, or less in case of a rapid change of operational significance in the meteorological conditions. Depending on local agreements, the latter situation leads to a SPECI (with the same template as METARs) or a SPECIAL (with the same template as local routine reports, known as MET REPORTs). METARs/SPECIs are representative of the whole aerodrome and are thus mainly

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Atmospheric Measurement Techniques Discussions

used by pilots for flight planning, while MET REPORTs/SPECIALs are representative of specific runways, and are mostly used for managing landing and departure operations by air traffic controllers (ATCs).

The first METAR in Switzerland was issued in 1969 (Willemse and Furger, 2017). The Swiss Federal Office of Meteorology and Climatology (MeteoSwiss) is currently responsible for the production of METARs at the Swiss civil airports of Geneva (ICAO code: LSGG), Kloten (ICAO code: LSZH), and an additional 8 regional aerodromes. Since 2007, it has done so using the SM⍙RT (System for Meteorological Automated ReporTing) software that has been fully developed and maintained by MeteoSwiss. SM⍙RT is comprised of both a backend component and a frontend component. The backend is responsible for the data collection from meteorological sensors and includes algorithms for the generation of AUTO METARs / AUTO MET REPORTs and AUTO SPECIALs. The frontend includes the SM⍙RT editor which is used by the observers to compile and send the reports (supported by the AUTO messages that serve as proposals), and a series of "viewers" showing for example the real-time data from meteorological sensors at different runways, and a map of lightning hits.

Up to now, the automatic METAR proposals generated by SM⍙RT at LSGG and LSZH have been systematically reviewed and adjusted by certified human observers during the airport operational hours (05:30-23:30 local time). The 24/7 automation of METARs is a challenging objective that has been pursued by several meteorological services over the years (see e.g. Leroy, 2006; Wauben et al., 2006; Hartley and Quayle, 2014; JMA, 2022). With its AMAROC (AutoMETAR/AutoReport rOund the Clock) program, MeteoSwiss is no exception (MeteoSwiss, 2022). Since November 2016 at LSGG and April 2021 at LSZH, during non-operational hours only, SM⍙RT-generated METARs are being issued without any live assessment/validation by a human observer. These are referred to as AUTO METARs.

Deriving cloud base heights and sky-coverage fractions from ceilometer observations requires dedicated software. A pioneering algorithm assembled by Esbjörn Olsson at the Swedish Meteorological and Hydrological Institute (SMHI) in 1995 was subsequently shared with ceilometer manufacturers Vaisala and Eliasson (at least), who further developed it (B. Nordberg, SMHI, private communication). The original SMHI algorithm was also the base for the software deployed at aerodromes in the Netherlands by the Royal Netherlands Meteorological Institute (KNMI) (Wauben, 2002). In the Unites States, the "sky condition algorithm" developed for the Automated Surface Observing Systems (ASOS; Nadolski, 1998) is another example.

Despite their widespread use, very little detailed information exists on these different algorithms, beyond the fact that they rely on clustering methods. None of the associated software are open-source, and several are in fact considered trade secrets. As a result, National Weather Services are often forced to re-develop custom algorithms and/or associated software from scratch. The opacity surrounding the different algorithms responsible for generating AUTO METAR worldwide also prevents any external (neutral) assessment, validation, and/or intercomparison of their capabilities.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024


Atmospheric Measurement Techniques Discussions

In this article, we present ampycloud, a new open-source algorithm and associated Python package designed to derive the base height and sky coverage fraction of clouds layers from ceilometer measurements. ampycloud is part of the SM⍙RT back-end. It is one among several new and/or improved algorithms developed at MeteoSwiss as part of the AMAROC program to generate fully-automated METAR reports at Swiss civil airports. The algorithm itself is introduced in Sec. 2. The ampycloud Python package, its performances, as well as its deployment and operational use by MeteoSwiss are all described in Sec. 3. The known limitations of ampycloud are discussed in Sec. 4, together with different enhancement possibilities. Our conclusions are presented in Sec. 5.

The algorithm described in this article corresponds to version 1.0.0 of the ampycloud Python package, that was released in November 2023. As a key dependency of the operational AUTO METAR software of MeteoSwiss, ampycloud must remain robust and stable. As such, if an evolution of the code is almost inevitable, this process will be carefully controlled. Any future release of the code (and associated changes) will be fully documented in the online documentation of the ampycloud package (ampycloud, 2024a), to which we refer the interested readers for more details on the evolution of the algorithm after the publication of this article.

2 The ampycloud algorithm

2.1 Requirements

An algorithm that has to operate 24/7 without direct human supervision should meet specific requirements. In particular, it should be:

1. robust (any input, no matter its plausibility, is processable), and
2. reliable (any input results in a reasonable output).

For the specific case of generating information to be used in METARs, the algorithm should also be designed to be:

3. stable (a small change in the input results in a small change in the output),
4. conservative (if two possible outputs are equally probable, the worst one – from an aerodrome/flight safety perspective – is to be favored), and
5. fast (the processing time should be less than ∼1 min, to issue SPECIs or SPECIALs as soon as warranted).

Furthermore, any meteorological algorithm can also benefit from being:

6. physics-based (methods and parameters have a physical interpretation and justification).

This can ease, for example, the selection of parameter values and the identification of specific shortcomings of the algorithm, and thus elements with a potential for improvement. A supervised machine learning method trained to reproduce (subjective)

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024


Atmospheric Measurement Techniques Discussions

cloud observations would be an example of a non-physics-based algorithm, which is generally more difficult to interpret and to maintain (typically denoted to as black box). ampycloud was developed with these six requirements in mind.

2.2 Algorithm concept

The ampycloud algorithm is composed of three sequential steps, which we refer to as *slicing*, *grouping*, and *layering*. The slicing step is designed for robust detection of the main cloud layers. It is meant to constrain the ampycloud outputs to a reasonable range, no matter the input data. The high degree of reliability of this step is achieved by reducing its capacity to handle complex cloud distributions. Specifically, the slicing step is subject (by design) to two pitfalls:

1. the splitting of cloud layers bases that are thick and/or have a marked up-/downward trend, and
2. the aggregation of close-but-distinct cloud layers.

The grouping and layering steps of the ampycloud algorithm are designed specifically to refine the slicing step. The slicing, grouping, and layering steps of ampycloud thus form a coherent sequence, with each step operating on the outcome of the previous one. The entire process is illustrated in Fig 1, which presents the ampycloud diagnostic diagram for a mock, demonstration dataset. The details of this diagram will be explained, for each step, in Sec. 2.5.1 to 2.5.3.

It must be stressed that the scope of ampycloud is to characterize cloud layers with well-defined bases only. In its current state, the algorithm does not concern itself with the question of whether a vertical visibility (VV) code should be reported instead of a cloud base height in METARs. Obscuration of the sky by fog or snow must be handled by a separate algorithm. It also does not consider the question of convective cloud detection (CB/TCU), which is best done using a combination of lightning and weather radar data. Essentially, ampycloud will report compliant METAR codes[1] of up to 3 cloud layers at most, with the following selection rules (ICAO, 2018):

- the lowest layer is always reported
- the second layer up is reported if it has a sky coverage of SCT or more
- the third layer up is reported if it has a sky coverage of BKN or more.

2.3 Input data

ampycloud relies on ceilometer *hits* acquired over a time interval Δt to characterize the cloud layer base heights and amounts above a specific area, which is a well-established approach (Wauben et al., 2006; Nadolski, 1998; Wauben et al., 2006; ICAO, 2011; WMO, 2021). It relies on the wind *dragging* the cloud distribution over the ceilometer(s) line(s)-of-sight to enhance their otherwise-very-restricted spatial sampling capabilities, and thus obtain a more representative view of the entire sky. The

[1] By definition, the METAR codes FEW, SCT, BKN, and OVC correspond to sky coverage fractions of 1-2 oktas, 3-4 oktas, 5-7 oktas, and exactly 8 oktas (respectively). NCD indicates that no clouds are detected by the automatic system.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

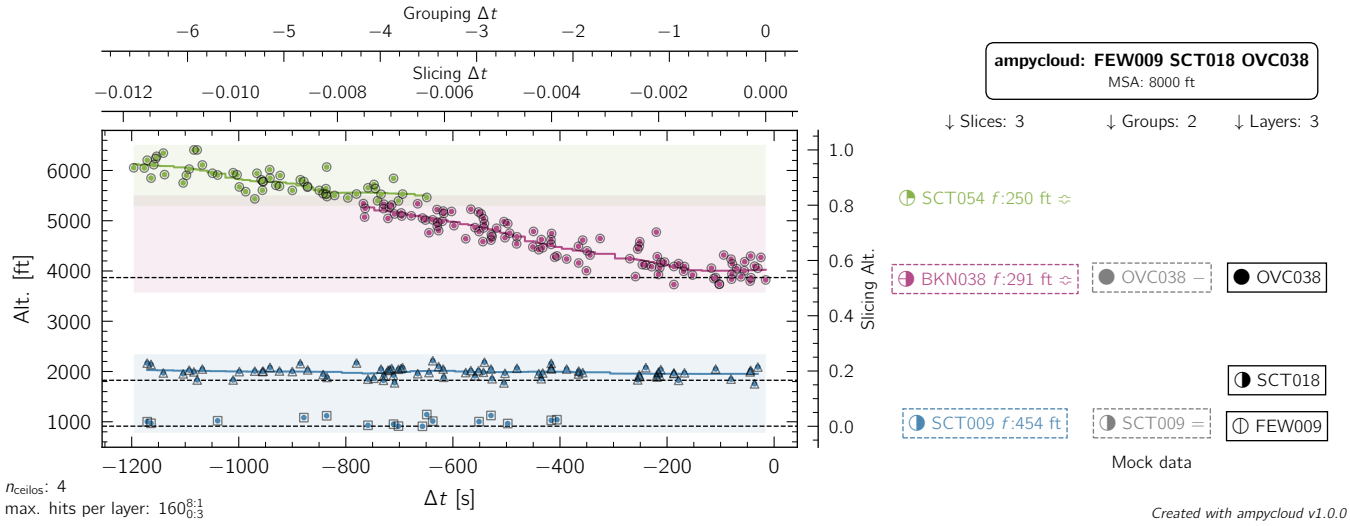


Figure 1. ampycloud diagnostic plot for a mock (demonstration) dataset, designed to illustrate the slicing, grouping, and layering steps of the algorithm. All the individual (pseudo-) ceilometer hits are visible as colored points in the diagram. Their color corresponds to the horizontal slice $\mathcal{S}_i$ (with $i \in [1,2,3]$) they are assigned to by the first step of the algorithm. The colored rectangles expand $\pm\epsilon/100 \cdot T(\mathcal{S}_i)$ above (below) the maximum (minimum) altitude of each slice $\mathcal{S}_i$, with $\epsilon \in [0,100]$ a constant and $T(\mathcal{S}_i)$ the thickness of the slice $\mathcal{S}_i$. This vertical range is used to identify overlapping slices that may require grouping. Colored lines correspond to LOWESS fits of the ceilometer hit distribution within each slice: they are used to compute the slice fluffiness f (in ft) used in the grouping step. The black squares/triangles/circles around each ceilometer hit denote the final layer $\mathcal{L}_i$ (with $i \in [1,2,3]$) they are assigned to, as a result of the third step of the algorithm. The intermediate groups $\mathcal{G}_i$ (with $i \in [1,2,3]$) identified by the second algorithm step are not shown for readability purposes. Secondary x/y axes on the right/top of the diagram show the rescaled hit times/altitudes axis used for the slicing and grouping steps. The characteristics of each slice $\mathcal{S}_i$/group $\mathcal{G}_i$/layer $\mathcal{L}_i$ with a sky coverage fraction of FEW or more are shown on the right-hand-side of the diagram, including the measured sky coverage in oktas. Overlapping slices are tagged with $\div$. Groups for which the layering step was executed are tagged with $-$, $=$, or $\equiv$ depending if 1, 2, or 3 distinct sub-layers were identified. Cloud slices/groups/layers found are reported according to METAR syntax, i.e. [FEW|SCT|BKN|OVC]nnn, where nnn is the cloud base height in hundreds of feet above ground. Not all cloud slices/groups/layers need to be reported according to the ICAO rules (ICAO, 2018). The ones that are of "operational significance" are boxed. The final selection of cloud layers is shown to the top right of the diagram. To the bottom left, the number of ceilometers contributing data to the diagram is indicated explicitly, alongside the maximum number of ceilometer hits possible per cloud layer (160 in this example) corresponding to the number of individual time steps among all the ceilometers. The sup-/subscripts indicate the values of the parameters Θ_8 and Θ_0 (with the format $8{:}\Theta_8$ and $0{:}\Theta_0$), which define the OVC/FEW thresholds in terms of number of holes/hits respectively. One should note that the number of slices, groups, and layers reported on the right-hand-side of the diagrams do include clusters that are comprised of less than Θ_0 hits.

longer the time interval or the larger the wind speed are, the better is the spatial representativity of the dataset, but the worse is its ability to serve as an accurate view of the "current" state of the sky (especially in case of rapidly changing conditions).

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Atmospheric Measurement Techniques Discussions

ampycloud requires every ceilometer hit h to be comprised of four elements:

$$h \equiv (h_{\rm alt}, h_{\rm tme}, h_{\rm cid}, h_{\rm tpe}), \tag{1}$$

with $h_{\rm alt}$ the measured cloud base altitude above ground (in ft), $h_{\rm tme}$ the (absolute or relative) observation time (in seconds), $h_{\rm cid}$ a ceilometer reference id, and $h_{\rm typ}$ the hit type, that will be explained below. The model and amount of ceilometers in operations at LSGG and LSZH is summarized in Table 1. It must be stressed that the use of ampycloud is not restricted to any specific model of ceilometer in particular, provided that individual hits can be specified following Eq. 1. One should note that for the moment, ampycloud cannot use full back-scatter profiles to derive cloud base hits independently from the ceilometers' proprietary software.

Table 1. Summary of the ceilometers deployed at the LSGG and LSZH civil aerodromes, as of June 2022.

Aerodrome	City	Ceilometer		
		count	model	manufacturer
LSGG	Geneva	4	CL31	Vaisala
LSZH	Kloten	7	CL31	Vaisala

ampycloud can process measurements from any number of ceilometers, and makes no distinction between them. Data are pooled together before the analysis, independently of the spatial distribution of the ceilometers. One can of course run ampycloud on a smaller number of ceilometers specific for a given runway, e.g. for generation of automatic local routine meteorological reports (AUTO MET REPORTs). Aside from operational redundancy, the main benefit of using multiple ceilometers to characterize cloud layers lies in the boost of the probability to detect clouds (resp. clear sky) for cloud layers with very low (resp. very high) sky coverage fractions. This fact, demonstrated experimentally by Wauben (2002), is illustrated in Fig. 2.

The specification of $h_{\rm cid}$ for every hit is necessary to compute a correct estimation of the sky covering fraction under the assumption that cloud layers have a unique base per time step per ceilometer line-of-sight.

The hit type $h_{\rm tpe}$ is also crucial to the correct estimation of the sky coverage fraction of cloud layers. Typically, ceilometers can report multiple cloud base altitudes and/or a vertical visibility (VV) with every observation (i.e. at every time step). For example, the ceilometers deployed at LSZH and LSGG report, for every measurement, either up to 3 distinct cloud base heights, or a vertical visibility and signal range. The latter occurs if the cloud base is obscured (for example due to fog or precipitation). The hit type $h_{\rm tpe}$ is used to keep track of these differences. It must be stressed, however, that ampycloud does not treat VV hits differently than regular hits. Every hit h fed to ampycloud is treated as a regular cloud base hit. It is up to the user to decide if and when VV hits should be provided to ampycloud, and whether they need to be pre-processed in any way before doing so. Not passing VV hits to ampycloud may lead to an underestimation of the cloud amount, but the fact that VV hits can also be

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Atmospheric Measurement Techniques Discussions

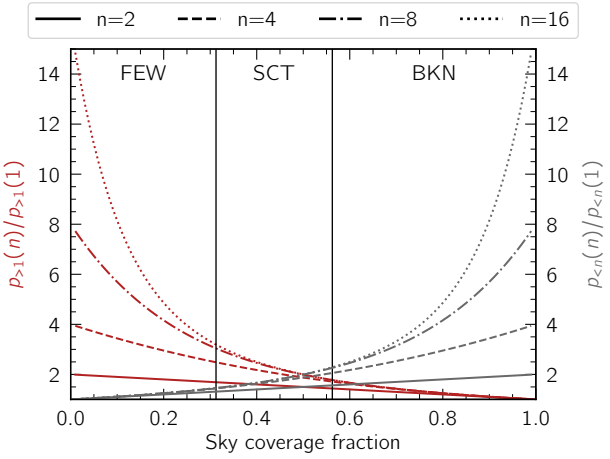


Figure 2. Improvement in the probability of getting at least one cloud hit with n ceilometers $p_{>1}(n)$ compared to only one ceilometer $p_{>1}(1)$ (at any one specific moment in time), ignoring any spatial scales considerations in the cloud and ceilometer distributions (red curves). The mirrored probabilities of detecting at least one clear sight-line $p_{<n}$ are shown in grey. The benefits of using $n > 1$ ceilometers to characterize cloud layers above a specific geographical location are strongest in situations with very low or very high sky coverage fractions. In other words, a larger number of ceilometers is mostly beneficial to the distinction between NCD and FEW conditions, respectively BKN and OVC.

caused by precipitation could also lead to the reporting of spurious cloud layers. As of ampycloud v1.0.0, MeteoSwiss feeds all observed VV hits to the code, without further modification/selection of their reported height.

ampycloud poses no restriction on the length Δt over which ceilometer hits can be pooled. For LSGG and LSZH, ampycloud is being fed ceilometer hits spanning $\Delta t = 15$ min for AUTO METARs (from all ceilometers), and $\Delta t = 6$ min for AUTO MET REPORTS (from specific pairs of ceilometers associated to a given runway). ampycloud works in a relative time space, and will automatically assign to each measurement a time difference with respect to the most recent one.

2.4 The parameters of ampycloud

The parameters of ampycloud are summarized in Table 2. Those specifically related to the slicing, grouping, and layering steps will be discussed in Sec. 2.5. The other ampycloud parameters are responsible for the generation of compliant METAR codes for each slice $\mathcal{S}_i$, group $\mathcal{G}_i$, or layer $\mathcal{L}_i$.

Θ_0 is the maximum (absolute) number of cloud hits for a given slice/group/layer to still be considered to be NCD by ampycloud; Θ_8, on the other hand, is the maximum number of "holes" (i.e. non-detection of clouds) for a given slice/group/layer to be considered to be OVC by ampycloud. We adopt $\Theta_0 = 3$ hits ($\Theta_8 = 1$ hole) to avoid assigning low (high) density cloud

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Atmospheric Measurement Techniques Discussions

layers to noise fluctuations and false cloud detections (e.g. returns by airplanes passing above a ceilometer).

ampycloud computes the base altitude of a given slice $\mathcal{S}_i$, group $\mathcal{G}_i$, or layer $\mathcal{L}_i$ as the β_h-percentile of the ordered hit altitudes, within the β_t-percentage of the most recent hits in the slice, group, or layer. We adopt $\beta_h = 5\%$, and $\beta_t = 100\%$ by default, such that all hits within a given slice, group, or layer are used to derive the altitude of its base (as commonly done by other cloud height algorithms). It must be stressed that ampycloud does not apply any weighting to the different hits prior to computing the base height or sky coverage fraction of a specific slice, group, or layer. This choice is driven by logic: one pools ceilometer measurements over a certain time interval Δt to obtain a (more) representative view of the sky. All cloud hits, irrespective of their age, are treated equally when identifying specific cloud layers. It is thus only logical to treat them all equally when computing the cloud base height of the identified layers. For ampycloud, old hits are not less valid: they merely represent the state of the sky further away than the ceilometer lines-of-sight. If a user were to prefer the cloud base heights to be more representative of the most recent hits within a slice/group/layer, the β_t parameter can be set to lower values, e.g. $\beta_t = 30\%$. Note that changing the value of β_t does not have any effect on the cloud amounts.

As will be discussed in Sec. 2.5.3, the layering step is able to separate cloud layers very close from one another if they are well defined. The grouping step can also lead to sub-groups in very close proximity to each others when they are well defined, which can be problematic from a user perspective. The parameters $\Delta h_{l,\text{vals}} = [250, 1000]$ and $\Delta h_{l,\text{lims}} = [0, 10000, \infty]$ are introduced to ensure that groups and layers identified in the second and third step of the algorithm are sufficiently far apart from one another. Essentially, the base altitude of cloud groups/layers must be at least $\Delta h_{l,\text{vals}}[k]$ ft apart if they have a base altitude in the range $[\Delta h_{l,\text{lims}}[k-1]; \Delta h_{l,\text{lims}}[k]]$.

In the ampycloud Python package, two additional parameters H_{MSA} and Δ_{MSA} are used to crop any hit with an altitude above ground significantly beyond the applicable Minimum Sector Altitude (MSA) for a given airport, namely beyond $H_{\text{MSA}} + \Delta_{\text{MSA}}$, where H_{MSA} is the MSA value (e.g. 10'000 ft and 8'000 ft for LSGG and LSZH, respectively) and Δ_{MSA} is a buffer to properly treat cloud layer bases whose thickness/fluctuations extend slightly beyond the MSA.

The sequential 3-steps structure of ampycloud will be discussed next. We do already note, however, that with a first robust *slicing* step being fine-tuned by the subsequent *grouping* and *layering* of cloud structures, the ampycloud algorithm is not overly sensitive to any of its parameter in particular.

2.5 The three ampycloud steps

2.5.1 Slicing

Given a set of hits gathered over a time interval Δt, the first ampycloud step consists in the identification of (horizontal) altitude "slices" $\mathcal{S}_i$. The algorithm does so using an agglomerative clustering approach with an average linkage criterion, and a

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Atmospheric Measurement Techniques Discussions

Table 2. parameters (and associated default values) controlling the behavior of ampycloud as of version 1.0.0 of the eponym Python package. The values of each parameter were chosen and verified using specific examples and a large-scale statistical assessment of the algorithm's performances (see Sec. 3.4).

Parameter	Value	Unit	Python variable name
		General	
H_{MSA}	LSGG: 1e4, LSZH: 8e3	ft	`MSA`
Δ_{MSA}	1500	ft	`MSA_HIT_BUFFER`
Θ_0	3	-	`MAX_HITS_OKTA0`
Θ_8	1	-	`MAX_HOLES_OKTA8`
β_h	5	%	`BASE_LVL_ALT_PERC`
β_t	100	%	`BASE_LVL_LOOKBACK_PERC`
L_{frac}	0.35	-	`LOWESS['frac']`
L_{it}	3	-	`LOWESS['it']`
$\Delta h_{l,\mathrm{vals}}$	[250, 1000]	ft	`MIN_SEP_VALS`
$\Delta h_{l,\mathrm{lims}}$	$[0, 10000, \infty]$	ft	`MIN_SEP_LIMS`
		Slicing step	
α_s	0.2	-	`SLICING_PRMS['distance_threshold']`
τ_s	10^5	s	`SLICING_PRMS['dt_scale']`
$\Delta h_{s,\mathrm{min}}$	1000	ft	`SLICING_PRMS['alt_scale_kwargs']['min_range']`
		Grouping step	
ϵ	+10	%	`GROUPING_PRMS['alt_pad_perc']`
τ_g	180	s	`GROUPING_PRMS['dt_scale']`
f_{b}	2	-	`GROUPING_PRMS['fluffiness_boost']`
$\Delta h_{g,\mathrm{min}}$	100	ft	`GROUPING_PRMS['alt_scale_range'][0]`
$\Delta h_{g,\mathrm{max}}$	500	ft	`GROUPING_PRMS['alt_scale_range'][1]`
		Layering step	
Θ_{min}	2	okta	`LAYERING_PRMS['min_okta_to_split']`
δ	0.95	-	`LAYERING_PRMS['gmm_kwargs']['delta_mul_gain']`
γ	100	-	`LAYERING_PRMS['gmm_kwargs']['rescale_0to']`

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Atmospheric Measurement Techniques Discussions

Manhattan metric[2]. The key to the robustness of this processing step lies in the rescaling of the cloud base hits, both along the time and altitude dimensions, applied prior to the clustering.

Given the minimum and maximum hit altitudes $h_{\min}$ and $h_{\max}$, the interval $[h_{\min}; h_{\max}]$ is mapped linearly onto the interval $[0; 1]$. If, however, $h_{\max} - h_{\min} < \Delta h_{s,\min}$, it is the interval:

$$[\frac{h_{\max} + h_{\min}}{2} - 0.5\Delta h_{s,\min}; \frac{h_{\max} + h_{\min}}{2} + 0.5\Delta h_{s,\min}]$$

with length $\Delta h_{s,\min}$ that gets mapped onto $[0; 1]$ instead. Essentially, $\Delta h_{s,\min}$ (which is a parameter of ampycloud, see Table 2 for the complete list) ensures that cases with a small altitude dispersion do not get over-stretched, and thus "over-sliced".

Along the time axis, hits are rescaled (i.e. normalized) by a very large number τ_s. If τ_s is large enough, the distance between hits along the time direction becomes essentially negligible in the rescaled space. As a result, the measure of the linkage distance –used by the agglomerative clustering method to decide whether to merge hits or not– is dominated by the altitude component of the hits. In this rescaled space, a distance threshold α_s will identify horizontal slices of hits whose rescaled mean altitude are separated by no more than α_s[3]. With $\tau_s \gg 1$, the value of α_s essentially represents the maximum thickness of individual horizontal slices, expressed as a fraction of the total altitude range in the sample, such that ampycloud will not produce slices thinner than $\alpha_s \cdot \Delta h_{s,\min} = 200$ ft (with its default parameters).

This rescaling scheme is well suited to identify vertically-stratified cloud layers with a stable base as a function of time. However, it is ill-suited for cloud layers with a thick and/or time varying base height, where the variation is larger than $\alpha_s \cdot (h_{\max} - h_{\min})$. This shortcoming is illustrated in Fig. 1 with the top layer being *sliced* in two. The slicing step is also ill-suited to distinguish isolated cloud layers separated by less than $\alpha_s \cdot (h_{\max} - h_{\min})$. This is illustrated with the bottom two cloud layers in Fig. 1, that get merged into a single slice.

The grouping and layering steps of ampycloud are designed specifically to handle these limitations of the slicing step. As a result, the output of ampycloud is not very sensitive to the values of $\Delta h_{s,\min}$ and α_s, in that any over-/under-slicing of hits is compensated by the subsequent processing steps.

2.5.2 Grouping

This processing step is designed to handle the *over-slicing* of coherent cloud hit structures as a result of a broad altitude span or marked up-/downward trends of a cloud layer. It begins by identifying so-called *overlapping* slices. These are flagged by the symbol "≍" in the ampycloud diagnostic diagrams (see Fig. 1). By (our) definition, slices i and j are overlapping if any one

[2]implemented in the Python package via the sklearn package (Pedregosa et al., 2011).

[3]The attentive reader may wonder why the slicing step relies on a 2-D clustering algorithm, given that the use of $\tau_s \gg 1$ essentially implies that 1-D clustering approach would be equivalent. The reason for this is essentially historical, and related to the initial assembly of the ampycloud algorithm.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024


Atmospheric Measurement Techniques Discussions

of the following two conditions are satisfied:

$$\bar{h}(\mathcal{S}_i) + \frac{50+\epsilon}{100} \cdot T(\mathcal{S}_i) > \bar{h}(\mathcal{S}_j) - \frac{50+\epsilon}{100} \cdot T(\mathcal{S}_j) \tag{2}$$

$$\bar{h}(\mathcal{S}_i) - \frac{50+\epsilon}{100} \cdot T(\mathcal{S}_i) < \bar{h}(\mathcal{S}_j) + \frac{50+\epsilon}{100} \cdot T(\mathcal{S}_j) \tag{3}$$

with $\bar{h}(\mathcal{S}_i)$ and $T(\mathcal{S}_i)$ the mean height and thickness of the slice $\mathcal{S}_i$, respectively, and ϵ a parameter of the ampycloud algorithm. For example, in Fig. 1, the top two slices (green and red) are overlapping, according to this criterion with our adopted value of $\epsilon = +10\,\%$.

Having identified a bundle of n overlapping slices, ampycloud uses an agglomerative clustering method with a single linkage criterion and an Euclidean metric to identify coherent sub-groups g_k, with $k \in [1;2;\ldots]$. Each sub-group g_k is subsequently assigned to a master group $\mathcal{G}_i$ based on the slice $\mathcal{S}_i$ to which the majority of the hits in group g_k belong. Hence, the ampycloud grouping of n overlapping slices will result in no less than 1 and no more than n master groups $\mathcal{G}$. In other words, there can be no more groups $\mathcal{G}$ than slices $\mathcal{S}$. Essentially, the slicing step of ampycloud is constraining the grouping step, which would otherwise be overly sensitive to noisy datasets. It also ensures that sparse cloud layers do not get broken into sub-groups along the time axis.

The hit altitudes and (relative) times are being rescaled prior to feeding them to the (single linkage) agglomerative clustering method of the grouping step. The rescaling is performed separately for each bundle of overlapping slices. In all cases, the time axis scaling factor is set to τ_g. The altitude scaling factor is taken to be the minimum slice fluffiness f_{min} in the overlapping bundle, provided that $f_{\text{min}} \in [\Delta h_{g,\text{min}},\ \Delta h_{g,\text{max}}]$. If f_{min} is larger (smaller) than $\Delta h_{g,\text{max}}$ ($\Delta h_{g,\text{min}}$), this value is used as the altitude rescaling factor instead.

We formally define the *fluffiness* f of a slice $\mathcal{S}_i$ as:

$$f(\mathcal{S}_i) = f_{\text{b}} \cdot \frac{1}{N} \sum_{h_k \in \mathcal{S}_i} |h_{\text{alt},k} - L(h_{\text{tme},k})| \tag{4}$$

with f_{b} a constant, N the number of ceilometer hits in the slice, $h_{\text{alt},k}$ the individual ceilometer hit altitudes, and $L(h_{\text{tme},k})$ the LOWESS-derived (Cleveland, 1979) altitude corresponding to the hit time $h_{\text{tme},k}$. The LOWESS[4] algorithm[5] is used to robustly measure the mean cloud hit height as a function of time using a series of localized weighted linear regressions. The LOWESS fit of each slice in Fig. 1 is shown using colored lines. It relies on two parameters: the fraction of slice hits (along the time axis) L_{frac} to use for each linear regression, and the number of iterations L_{it} for each fit. We note that although the ampycloud algorithm uses the fluffiness of (overlapping) slices $\mathcal{S}$ only for the grouping step, the ampycloud Python package computes the fluffiness of every slice $\mathcal{S}_i$, group $\mathcal{G}_i$, and layer $\mathcal{L}_i$ by default.

[4]Locally Weighted Scatterplot Smoothing
[5]implemented in the Python package via the statsmodels package (Seabold and Perktold, 2010).

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Atmospheric Measurement Techniques Discussions

The agglomerative clustering method with single linkage criterion uses the closest elements of sub-clusters to decide whether to merge them or not. With a linkage distance of 1 fixed inside ampycloud, τ_g and the slice fluffiness determine the maximum distance (in time and altitude, respectively) below which two ceilometer hits will be assumed to belong to the same cloud structure. Unlike the slicing step that ignores the time information (with $\tau_s \ggg 1$), the grouping step is much better suited to track coherent altitude variations as a function of time, as demonstrated in Fig 1. In that example, the hits in the top two slices (red and green at 3'800 ft and 5'400 ft, respectively) are grouped into a single master group with a base at 3'800 ft.

2.5.3 Layering

By design and as illustrated in Fig. 1, the slicing step will inevitably bundle distinct cloud structures into common slices if isolated cloud layers are separated by less than $\alpha_s \cdot (h_{\text{max}} - h_{\text{min}})$. The layering step is designed to correct this shortcoming. It relies on the computation of Gaussian mixture models with 1, 2, and 3 components for every master group $\mathcal{G}_i$ with a sky coverage fraction of at least Θ_{min} oktas. For each model, the value of the associated Bayesian Information Criterion BIC_i is recorded.

Statistically speaking, the best fitting model out of the three will have the lowest BIC value. However, ceilometer hits belonging to a given cloud layer typically will not follow a Gaussian distribution. Using the relative likelihood (and associated probabilities) of different Gaussian Mixture models to identify the most likely number of components in a group would thus typically lead to an overestimate: not because more components represent a better fit, but rather because they represent a somewhat-less-but-still-terrible one (see Appendix A for details). Hence, ampycloud does not simply follow this criterion alone to decide whether a given group requires to be broken up. The following selection approach is adopted instead.

The starting assumption for each group $\mathcal{G}_i$ is that it does not contain sub-layers. Moving sequentially through the Gaussian Mixture models with 1, 2, and 3 components, ampycloud will identify n sub-layers if $BIC_n < \delta \cdot BIC_{\text{current best}}$. In other words, ampycloud will deem n sub-layers to be present only if the decrease in the model's BIC value over the current best amount of sub-layers is greater than $(1 - \delta)$.

BIC scores are sensitive to the data values in the underlying dataset. To alleviate the consequences of this fact, ampycloud rescales the altitude range of every individual master group $\mathcal{G}_i$ between 0 and γ prior to searching for sub-layers. A given δ value will thus split groups based upon the relative separation of their sub-layers in terms of their altitude dispersion, irrespective of whether the groups are located at 2000 ft or 20'000 ft.

This approach still remains unstable for small datasets. This is why ampycloud will not attempt to break-up groups with a sky coverage fraction of less than $\Theta_{\text{min}} = 2$ oktas. The adopted values of $\delta = 0.95$ and $\gamma = 100$ imply that two perfectly Gaussian sub-layers would be separated by ampycloud if their means are offset by at least $\sim$5 standard deviations from one

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024


Atmospheric Measurement Techniques Discussions

another (see Appendix A for details).

3 The ampycloud Python package

3.1 Implementation

The ampycloud Python package (Vogt and Regenass, 2023) is developed on Github (ampycloud, 2024b) as a regular, non-MeteoSwiss specific, pip-installable Python package. The source code is made available to the general community as open-source software under the terms of the 3-Clause BSD License. As a project, ampycloud follows a Continuous Integration/Continuous Delivery approach. It relies on Github Actions to run automated quality/stability/validity checks, publish the documentation, and upload new versions to the Python Package Index (PyPI).

We voluntarily do not discuss here the technical implementation of the ampycloud Python package itself in extensive detail, given that software is prone to evolution. It suffices to say that 1) ampycloud requires to be fed ceilometer hits stored in a suitably-formatted pandas `DataFrame`, 2) the slicing and grouping steps rely on the `sklearn.cluster.AgglomerativeCluste` class from the scikit-learn Python package, 3) the layering step relies on the `sklearn.mixture.GaussianMixture()` class from the same package, and 4) the fluffiness of slices/groups/layers is computed using the LOWESS algorithm implemented in the statsmodels package. We refer the interested readers to the official ampycloud online documentation (ampycloud, 2024b) for further information on its different classes and functions.

3.2 Deployment at MeteoSwiss

The ampycloud Python package is imported by MeteoSwiss as an external dependency within a Python service known as the *cloud calculator* (closed-source software). The cloud calculator is one of several calculators that comprise the autometpy software, developed as part of the AMAROC project. The cloud calculator runs on the MeteoSwiss OpenShift container platform, where it stands ready to receive calculation requests. The requests are issued and orchestrated by SM⍙RT, that decides how often and with what data the cloud calculator is called. At the LSGG airport, SM⍙RT makes 3 requests to the cloud calculator every minute in:

- AUTO METAR mode, using data from all 4 ceilometers over the last 15 min.
- AUTO MET REPORT mode using data from the 2 ceilometers representative of RWY22 over the last 6 min.
- AUTO MET REPORT mode using data from the 2 ceilometers representative of RWY04 over the last 6 min.

Monitoring the status and computational performances of the cloud calculator is done using a flexible dashboard providing access to various metrics (CPU load, memory use, request time, number of requests, errors). Dedicated log files are also collected and can be queried whenever necessary.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024


Atmospheric Measurement Techniques Discussions

3.3 Computational speed performance

We use the mock dataset presented in Fig. 1 to keep track of the speed performance of the ampycloud package over the course of its development, by means of a dedicated Github Action (ampycloud, 2024c). With ampycloud v1.0.0, the processing of this dataset takes 0.20 s on average on a 16-inch MacBook Pro 2019 with an 2.3 GHz 8-Core Intel Core i9. On a server running Ubuntu with Linux kernel 6.2 with 4 cores, the processing lasts 0.30 s (ampycloud, 2024d). These values are representative of ampycloud processing times in general. Among the real examples presented in Appendix B, the case in Fig. B2 takes the most time to process (0.35 s), whereas the case in Fig. B12 takes the least time (0.12 s). These processing times do not include the creation of the ampycloud diagnostic diagrams, which are optional and require an additional ~ 2 s per diagram.

3.4 Comparison with reference METARs

We present in Appendix B a series of representative, real examples from the Swiss civil aerodromes. These are meant to illustrate the behavior of the ampycloud algorithm under varying circumstances and conditions. For each case, the actual METAR cloud codes (produced by a human observer at the time) are provided for comparison purposes. For clarity, each case is discussed directly within the caption of the Figure.

The overall performance of the ampycloud algorithm was evaluated in a statistical sense, using a reference dataset of 2128 cases extracted over a 5 year period (2018-2022) from LSGG METARs. These cases were selected to provide a comprehensive sub-sample of all the METARs over this period, suitable for validation purposes within MeteoSwiss and for the Swiss civil aviation authorities, with a selection focus set on situations with high-impact consequences for aerodrome operations.

We present in Fig. 3 a comparison of the densest cloud layer identified by the ampycloud algorithm compared to the actual METAR. In 57.8% of the cases, the ampycloud output is consistent with the METAR, and at most 1 density-category off in 95.7% of the cases. The cases where ampycloud differs more significantly from the METAR land in two categories. The cases when ampycloud underestimates the density of the cloud layer (aka "misses"; 3.2% of the cases) are related to situations where cloud layers are not seen by the ceilometers - either because they are not above the line of sight, or because the ceilometer signal is blocked by a lower layer. The "false alarms" (1.2% of all cases), where ampycloud overestimates the density of the cloud layer, are nearly all related to cases of low-level fog seen as a low-level cloud layer by the ceilometers.

This evaluation can be adjusted to focus on high-impact cases assessed using operationally-relevant categories. In Fig. 4, we present the ability of ampycloud to report a cloud "ceiling", i.e. a BKN or OVC cloud layer. The ampycloud identification of a ceiling (or absence thereof) is in agreement with the corresponding METAR 87.7% of the time. From Fig. 3, one can see that false alarms (4.6%) and misses (7.8%) are dominated (respectively) by cases where ampycloud characterizes a SCT layer as BKN (3.3% of the cases), and a BKN layer as SCT (5.5% of the cases). These two types of situations are a direct consequence

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Atmospheric Measurement Techniques Discussions

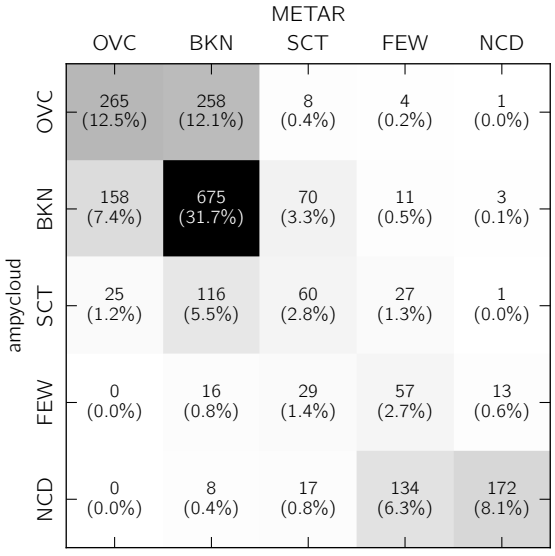


Figure 3. Matrix comparing the categorization of the densest cloud layer identified by the ampycloud algorithm (vertical axis) with the actual METAR validated by a human observer (horizontal axis), for a reference dataset of 2128 cases assembled from LSGG observations. Boxes located above the diagonal can be associated to "false alarms" where ampycloud finds a cloud layer denser than the METAR. The cases below the diagonal can be understood as "misses", where the opposite happens.

of the limited spatial sampling capabilities of ceilometers.

The matrix in Fig. 3 is useful to identify cases where the density of cloud layers is strongly mis-estimated. But it cannot guarantee that the "correct" cloud layers are being identified when they have the correct density. We thus provide in Fig. 5 a comparison of the altitude of the densest cloud layer measured by ampycloud with respect to that reported in the corresponding METAR. In 64.4% of the cases, ampycloud identifies a cloud layer height in good agreement with the METAR. The remaining cases reveal a tendency of ampycloud to provide somewhat lower cloud altitudes that the METARs. This is a direct consequence of the conservative approach of the ampycloud algorithm, that considers the entire set of ceilometer hits (in its default configuration) to derive a given cloud base height. The example in Fig. B11 illustrates this behavior. Cases in the bottom row (6.6% of the total) correspond to situations where the cloud layer is being missed entirely by the ceilometers.

4 Limitations and future prospects

In its current implementation, ampycloud uses a fixed time interval Δt to characterize cloud layers. In doing so, ampycloud is essentially trading-off its responsiveness to rapidly evolving conditions with its ability to obtain a representative view of the entire sky. We note that the default time interval $\Delta t = 15$ min adopted by ampycloud is shorter than in other algorithms

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Atmospheric Measurement Techniques Discussions

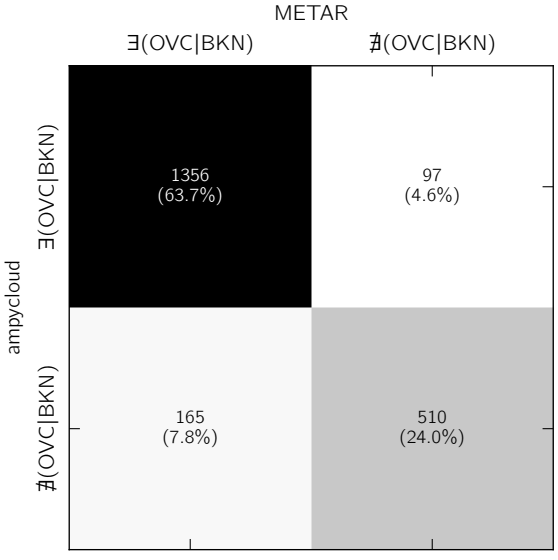


Figure 4. Same as Fig. 3, but with simplified categories related to the presence/absence of a cloud ceiling.

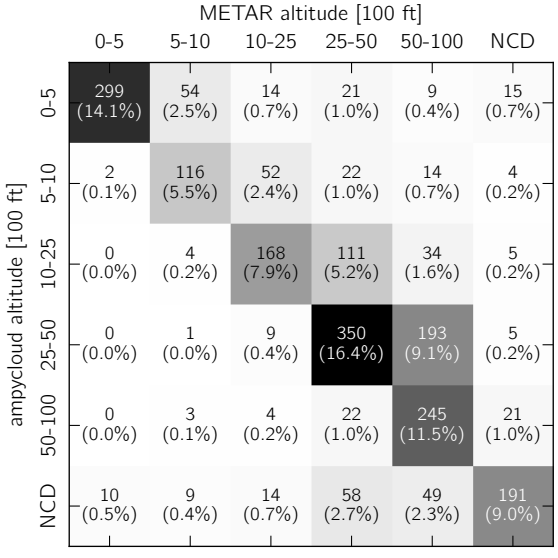


Figure 5. Same as Fig. 3, but for the altitude of the densest cloud layer. The differences between ampycloud and the METARs are dominated by cases where ampycloud returns a cloud layer altitude somewhat lower than the METAR. This is a direct consequence of the conservative design of the algorithm, that (in its default configuration) considers the entire look-back time to compute the altitude of a given set of ceilometer hits.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024


Atmospheric Measurement Techniques Discussions

with similar goals (see e.g. Wauben, 2002; ICAO, 2011). But unlike those algorithms that give additional weight to the most recent measurements, ampycloud treats every hit equally. This implies that ampycloud may possibly respond somewhat less rapidly to changes, but should be more stable when analyzing sparse cloud layers. Understanding the exact behavioral differences between ampycloud and other algorithms with similar purpose would require a dedicated comparison of their respective performances against a reference dataset, which is outside the scope of this article.

It must be stressed that ampycloud does not challenge the quality/nature of the ceilometer hits that it is being provided: it trusts them all fully and equally. The capacity of the algorithm to provide an accurate assessment of cloud layers above an aerodrome is thus directly limited by the ability of ceilometers to see the clouds up to the aerodrome's MSA in the first place, and measure their heights accurately (Costa-Surós et al., 2013; Wagner and Kleiss, 2016). Any spurious cloud hit (be it caused by measurement noise, dense smog, rain, or airplanes flying overhead) will inevitably impact the outcome of ampycloud. So is a "lack of cloud hits", for example when the view towards upper layers is blocked by lower ones in complex situations.

One enhancement possibility for the ampycloud algorithm would be to use the wind speed to identify suitable look-back times as a function of altitude, in order to obtain a (more) representative sampling of the sky. Doing so in real-time could be achieved, at least up to the aerodrome minimum sector altitudes, by direct measurements (for example using wind profilers), by data extraction from numerical models, or by a combination of both.

The layering step of ampycloud is the part of the algorithm that shows the most potential for improvement, for the following reasons. Unlike the slicing and grouping steps, the parameters γ and δ associated to this step are less-directly relatable to physical quantities. For specific distributions of cloud hits, this step can be sensitive to the random seed used by the system[6]. Most importantly, the use of Gaussian Mixture Models clearly is not ideal from a physics perspective, given the fact that cloud base hits are not expected (nor seen) to systematically follow a Gaussian distribution, although it remains a sufficiently valid approximation.

The main limitation of ampycloud, however, currently resides in the small amount of information that it relies on in order to characterize cloud layers. In particular, ampycloud processes ceilometers hits without questions, and without differentiating them based on the weather process that generated them. The use of complete backscatter profiles from the ceilometers could help improve this state-of-affair: for example, by enabling the identification of spurious hits triggered by heavy precipitation or aircrafts, and/or by supplementing cloud base height measurements with cloud layer thickness which could significantly ease the identification of coherent structures. The use of complete backscatter profiles could also enable the identification of the maximum observed height (in case of thick cloud layers), and better account for the sky coverage density of partially-obscured cloud layers.

[6]This seed is fixed by ampycloud to ensure that results are strictly repeatable on a given system.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024


Atmospheric Measurement Techniques Discussions

The use of a finite number of ceilometers alone cannot always be sufficient to differentiate very sparse cloud layers from a "true" clear sky. If a series of ceilometers see no clouds, the use of pyrgeometer (see e.g. Aviolat et al., 1998; Marty and Philipona, 2000; Dürr and Philipona, 2004), visible all-sky cameras (Wacker et al., 2015), infrared all-sky imagers (Aebi et al., 2018) or even satellite images could help ascertain the validity of a sky-clear condition – albeit without the same ability to infer the altitude of the sparse cloud layers that may have been missed by the ceilometers. Similarly to Doppler lidars, the use of rotating beam ceilometers (WMO, 2021) with the ability to probe more than one sightline could clearly improve the reliability of AUTO METAR for sparse cloud layers (i.e. for intermediate okta values), albeit at an increased financial cost and maintenance challenge (a rotating component is more prone to technical issues).

5 Conclusions

The ampycloud algorithm was developed at MeteoSwiss for fully automatic METAR production (AUTO METAR) at Swiss civil aerodromes. The eponym Python package is released online as open-source software under the terms of the 3-Clause BSD license. The code and its dedicated online (technical) documentation is completed by this article describing the underlying algorithm and its scientific motivation.

The public release of ampycloud takes place within a large paradigm change towards Open Government Data in Switzerland. For the project itself, an open-source software can facilitate tests of the algorithm at various locations beyond the Swiss civil aerodromes. It should also enable detailed comparisons with other algorithms with a similar purpose. Up to now, such comparisons were essentially impossible due to the private nature of source codes and lack of detailed documentation. With this article and associated supplementary material (Vogt, 2023b), we purposely seek to make the ampycloud processing of the examples shown in Figs. B1 to B12 reproducible by motivated users and researchers.

The ampycloud Python package forms an integral part of the software infrastructure of MeteoSwiss. Yet, it does not contain any MeteoSwiss-specific code nor does it rely on any specific MeteoSwiss hardware/software infrastructure (which are all contained within autometpy, see Sec. 3.2). The package focuses tightly on the implementation of the ampycloud algorithm itself, with only a handful of dependencies, all well-maintained and under active development within the Python community. The ampycloud package is not designed to interact with the outside World (and other programming languages, e.g. via JSON format). The implementation of the necessary Application Programming Interfaces (APIs) are left to the interested users.

The ampycloud algorithm and associated Python package version 0.5.0 was first deployed at the LSGG airport in Geneva on 31.10.2022 outside the operational hours of the airport, with its outcome supervised by human observers during operational hours and used as METAR proposal. The version 1.0.0 described in this article was deployed on 28.11.2023. The automatic production and distribution of AUTO METAR without human supervision at Geneva airport is foreseen during the year 2024. The accuracy of the ampycloud algorithm has been tested in detail on a series of specific examples (as illustrated in Figs. B1 to

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Atmospheric Measurement Techniques Discussions

B12), and statistically over a reference set of cases extracted over a period of 5 years. With a correct identification of a ceiling (or absence thereof) 87.7% of the time, the results of the ampycloud algorithm are found to be in good agreement with the corresponding METARs, such that the use of this algorithm at LSGG was approved by the Swiss civil aviation authorities. Several elements of the ampycloud algorithm do have a clear potential for improvements: the necessity and exact benefits of these improvements will be continuously investigated and evaluated by MeteoSwiss over the coming years, throughout and beyond the transition from METAR to AUTO METAR at Swiss civil aerodromes.

Code and data availability. The ampycloud Python package is freely available on Github (https://github.com/MeteoSwiss/ampycloud), with each release archived on Zenodo (DOI:10.5281/zenodo.8399683). The script and ceilometer hits used to generate the Figures in Appendix B have also been stored on Zenodo (DOI:10.5281/zenodo.10171151), from where they can be downloaded freely.

Appendix A: The ampycloud Gaussian mixture model selection rule

In ampycloud, Gaussian mixture models with 1, 2 and 3 components are used to determine whether a given group $\mathcal{G}$ is comprised of multiple distinct sub-layers. To illustrate how the Bayesian Information Criterion (BIC) scores allow us to reliably select the optimal number of sub-layers in a given group, we create a series of artificial distributions of cloud hits. Each distribution is comprised of either 2 or 3 Gaussian layers, each with a number of randomly-generated hits $N_{\text{hits}} \in [10, \ldots, 350]$, and with the layers separated in altitude by up to $\Delta = 8$ times their standard deviation σ. Four examples of these distributions are presented in Fig. A1.

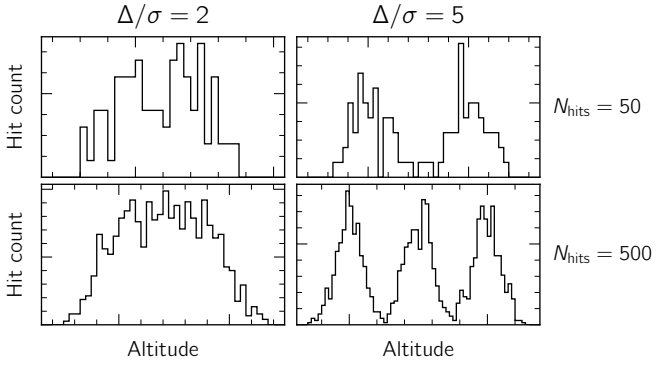


Figure A1. Simulations of 2 (top row) and 3 (bottom row) cloud layers using random normal distributions with N_{hits} artificial cloud base hits per layer. Individual layers are separated by $\Delta = 2$ (left column) and $\Delta = 5$ (right column) standard deviations σ.

The Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) scores provide means to assess the goodness of fit of different Gaussian mixture models. The smaller the AIC or BIC scores, the better the model fit. In ampy-

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024


Atmospheric Measurement Techniques Discussions

cloud, we use BIC scores to decide whether a given group $\mathcal{G}$ is composed of sub-layers. The AIC score tends to favor models with larger numbers of components, which is at odd with the ampycloud approach of favoring –for near-equal scores– the least number of Gaussian components to prevent unjustified sub-layering. Therefore, we rely on the BIC score only.

The relative likelihood of different Gaussian mixture models can be used to assign probabilities to each of them. The Bayesian probability of model i is:

$$p_{BIC,i} = \frac{e^{-0.5(BIC_i - BIC_{\min})}}{\sum_i e^{-0.5(BIC_i - BIC_{\min})}} \tag{A1}$$

with $BIC_{\min}$ the minimum BIC score amongst the Gaussian mixture models under consideration.

We show in Fig. A2 the values of $p_{BIC,i}$ for the simulated datasets with 2 Gaussian sub-layers. The Gaussian mixture model approach appears incapable of identifying 2 distinct components when $\Delta \lesssim 2\sigma$, i.e. when the layers are separated by less than twice their standard deviation. For cases with $\Delta \gtrsim 2\sigma$, the probability of the distributions being comprised of a single Gaussian distribution drops extremely rapidly, in favor of the 2 components model. The transition is slightly less sharp for cases with few number of cloud base hits (i.e. cases where $N_{\text{hits}} \lesssim 50$). The case of 3 simulated cloud layers is shown in Fig. A3. A single component is favored by the Gaussian Mixture Model approach for cases where $\Delta \lesssim 2\sigma$. A narrow, intermediate zone favoring 2 components is present for cases where $2\sigma \lesssim \Delta \lesssim 3\sigma$, whereas 3 components are correctly identified for cases where $\Delta \gtrsim 3\sigma$.

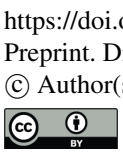
Unlike the simulated cases presented here, real cloud base hits are not following a Gaussian distribution, in particular due to temporal trends in the cloud base heights (a fact which is clearly visible in Figs. B1 to B8). For real cases, using of the probabilities p_{BIC} to select the optimal number of sub-layers present inside a given group $\mathcal{G}$ works *too efficiently*, in the sense that a single Gaussian component is ruled out too rapidly in the case of broad, flat layers. To circumvent this limitation, ampycloud uses a slightly-adjusted selection criteria based on the BIC values, which varies more smoothly as a function of the (normalized) layer separation Δ/σ. The general idea is to assume 1 component is present, and only favor a solution with 2 (or 3) components if the decrease of the associated BIC scores is sufficiently *significant*. In other words, a model with j components is favored over a model with i components only if:

$$BIC_j < \delta \cdot BIC_i \tag{A2}$$

where δ is a multiplicative factor (a parameter of the ampycloud algorithm).

The consequences of these selection criteria are shown in the bottom rows of Fig. A2 and A3. Varying the value of δ allows to more easily decide the level Δ/σ at which 2 (resp. 3) components are favored over a single one. With the default ampycloud value of $\delta = 0.95$, this transition occurs at $\Delta/\sigma \approx 5$. Sub-layers are therefore identified in a given group $\mathcal{G}$ only if they are well separated, as illustrated in Fig. A1. For the case of 3 simulated sub-layers, the selection criteria defined in Eq. A2 also has the advantage that it never favors 2 components (unlike the probabilities p_{BIC}, as illustrated in Fig. A2). If 3 components cannot be identified unambiguously, no sub-layering occurs.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024



Atmospheric Measurement Techniques Discussions

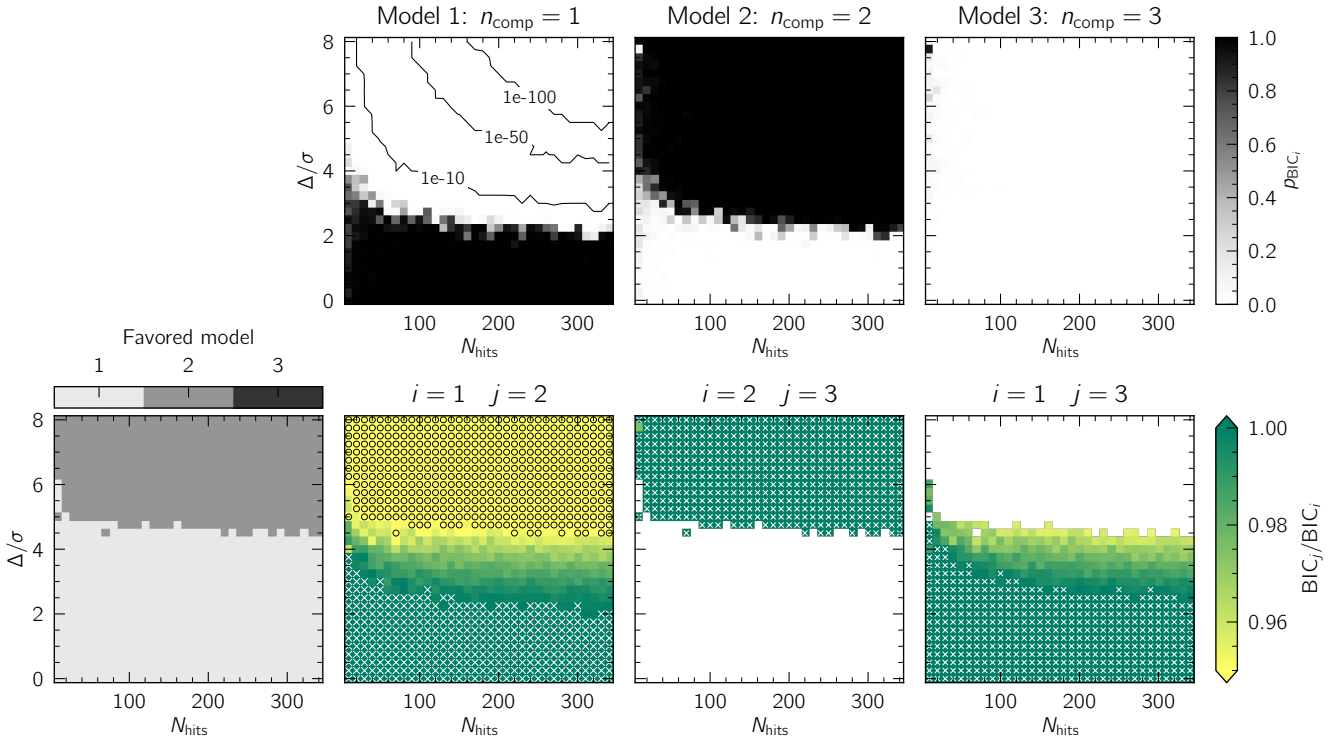


Figure A2. Illustration of the ability of the ampycloud algorithm to distinguish the presence of 2 simulated Gaussian sub-layers inside a group $\mathcal{G}$, as a function of the relative separation of the sub-layers Δ (expressed in terms of the layers standard deviation σ) and the number of cloud hits per layer N_{hits}. Top row: BIC probabilities p_{BIC} computed following Eq. A1, built from the relative likelihood of Gaussian mixture models with 1 (left), 2 (middle), and 3 (right) components. Each pixel in the image corresponds to the median of 10 independent realizations. For very low numbers of hits per layers ($N_{\text{hits}} \lesssim 50$), the selection of the best Gaussian mixture model is less sharp. When the sub-layers are close from one another (with $\Delta \lesssim 2\sigma$), the Gaussian mixture model approach is unable to reliably identify two distinct components. Bottom (right): distributions of BIC_j/BIC_i, for the cases of 1 component vs 2, 2 components versus 3, and 1 component versus 3. Cases where $BIC_j/BIC_i > 1$ are tagged with a white cross: these are regions where the model with j components is unambiguously rejected against the model with i components because of a larger BIC score. On the other hand, cases where $BIC_j/BIC_i < 0.95$ are tagged using black circles. For these and for these only, ampycloud will favor the model with j components following Eq. A2. The resulting map of the model favored by ampycloud as a function of N_{hits} and Δ/σ is visible in the bottom-left diagram.

Appendix B: Detailed examples

We present in Figs. B1 to B12 the ampycloud diagnostic diagrams from a series of representative, real situations taken from the LSGG and LSZH aerodromes. These examples serve to illustrate the behavior of ampycloud in different conditions, and compare the algorithm's output with the official METAR cloud codes that were validated and issued by a human observer at the time. For the ease of readability, each example is discussed in detail in the associated Figure caption. These examples are all

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

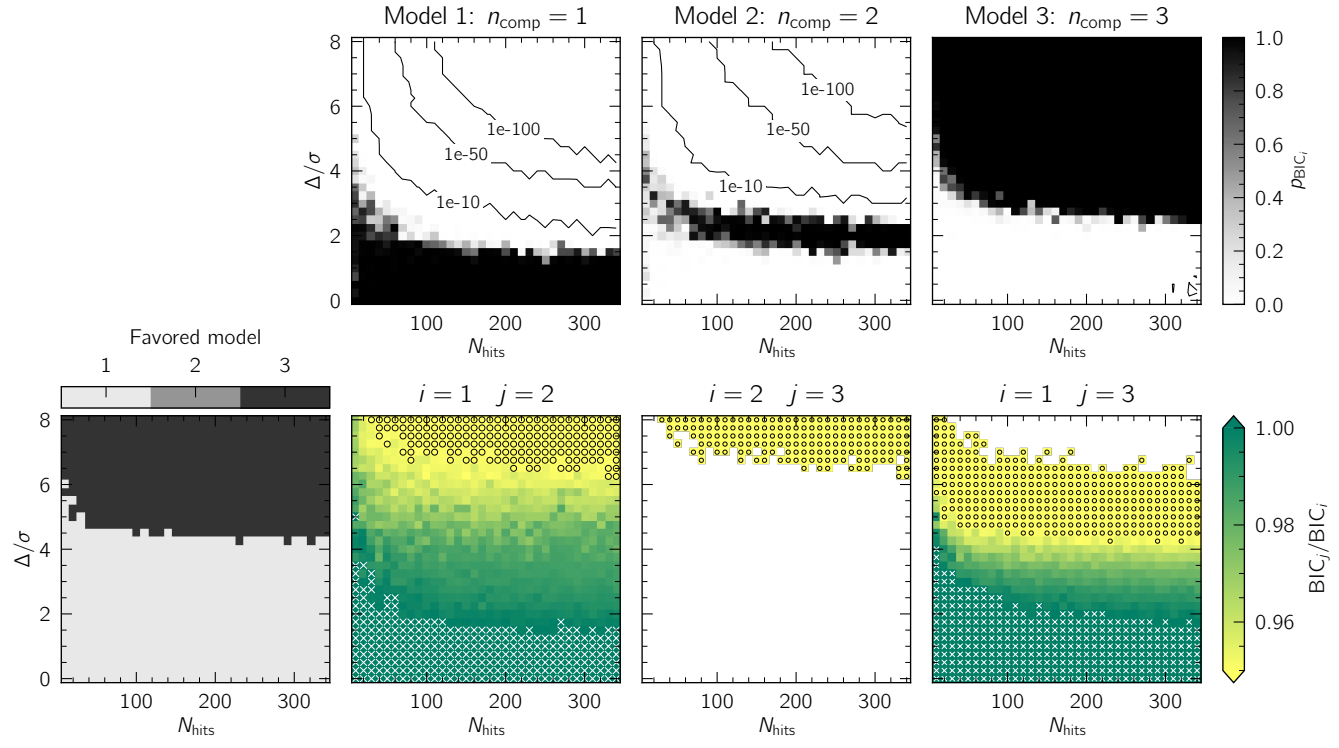


Figure A3. Same as Fig. A2, but for 3 simulated layers. As for the case of 2 simulated layers, the ampycloud algorithm is able to correctly identify 3 distinct layers when they are separated by ∼5 standard deviations ($\Delta \gtrsim 5\sigma$). A single layer is favored otherwise.

part of the set of cases used to assess the "scientific stability" of the ampycloud algorithm over the course of its development, by means of dedicated tests (ampycloud, 2024e) run using the pytest module.

The underlying sets of ceilometer hits associated to each example are made available to the interested reader, alongside a small Python script designed to process them using ampycloud and generate the associated ampycloud diagnostic diagrams. This material is archived on Zenodo and publicly available (under a Creative Commons Attribution 4.0 International license; Vogt, 2023b).

Author contributions. The ampycloud algorithm was designed by Vogt, with inputs from Foresti and Regenass. The ampycloud Python package was assembled by Vogt with important contributions from Regenass, Foresti and Tarin Buriel, and feedback from Bibby, Juda, Balmelli, Hanselmann, and du Preez. The large scale statistical assessment of the code was performed by Regenass, with contributions from Foresti, Bibby and Tarin Buriel. All authors contributed to the writing of this article.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

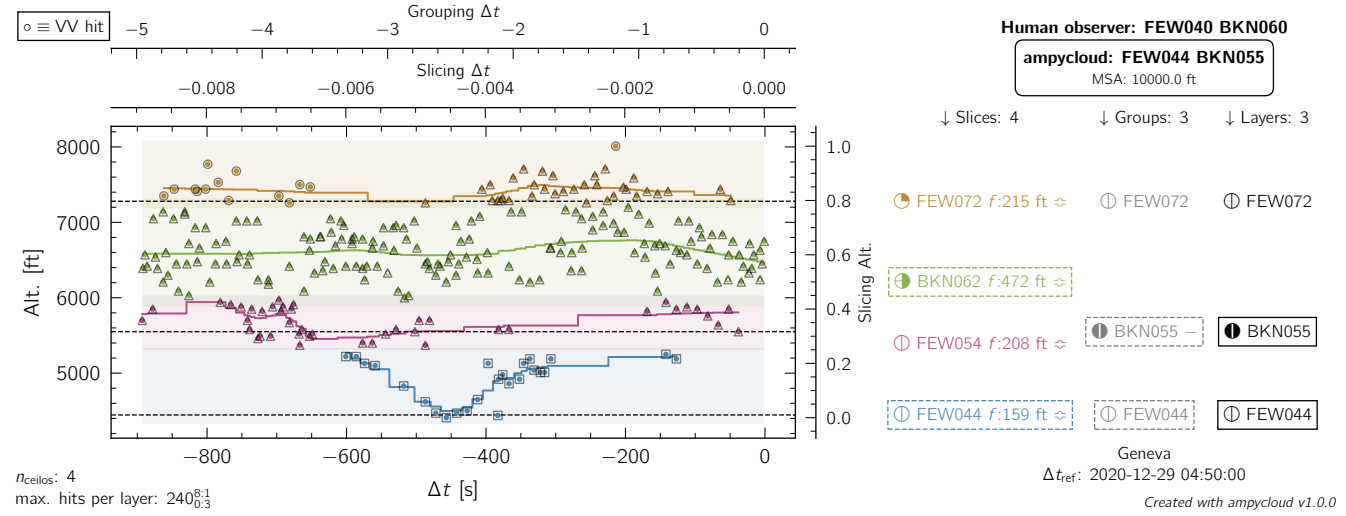


Figure B1. ampycloud diagnostic diagram for LSGG on 29 December 2020 at 04:50 UTC. The hits from 4 ceilometers over a period of 15 minutes are being considered. The four dominant slices are found to be overlapping, but the separations between individual hits is too large for the second processing step to bundle them all into a common master group. The symbol – in the Groups column on the right-hand-side indicates that sub-layers are being searched (but not found) by the layering step for the second group (from bottom; BKN055), which is the only structure with sufficient hits to do so (i.e. with a sky coverage fraction larger or equal to 2 oktas).

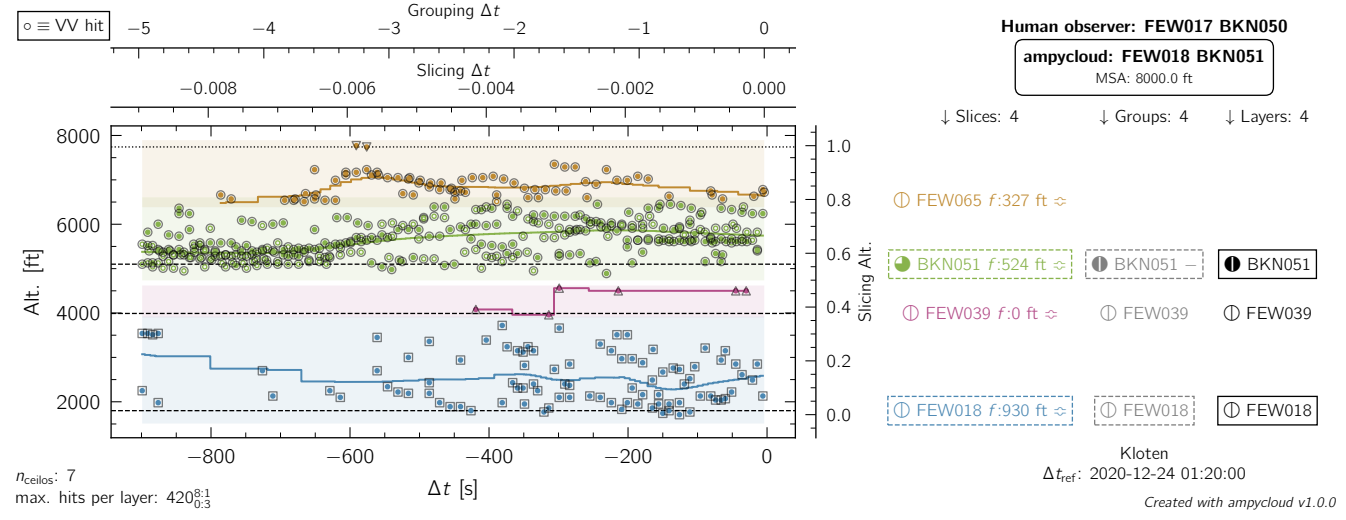
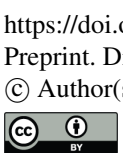


Figure B2. Same as Fig. B1, but for LSZH on 24 December 2020 at 01:20 UTC. VV hits are indicated using colored points with no filling: they are treated like regular cloud base hits by ampycloud. In this example, the grouping step has been used to merge the majority of hits in the top two slices: only the highest two cloud base hits remain as the upper-most layer, but they are discarded as a minimum of 4 hits are required for a layer be considered having a sky coverage fraction above 0 oktas (i.e. $\Theta_0 = 3$).

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024



Atmospheric Measurement Techniques Discussions

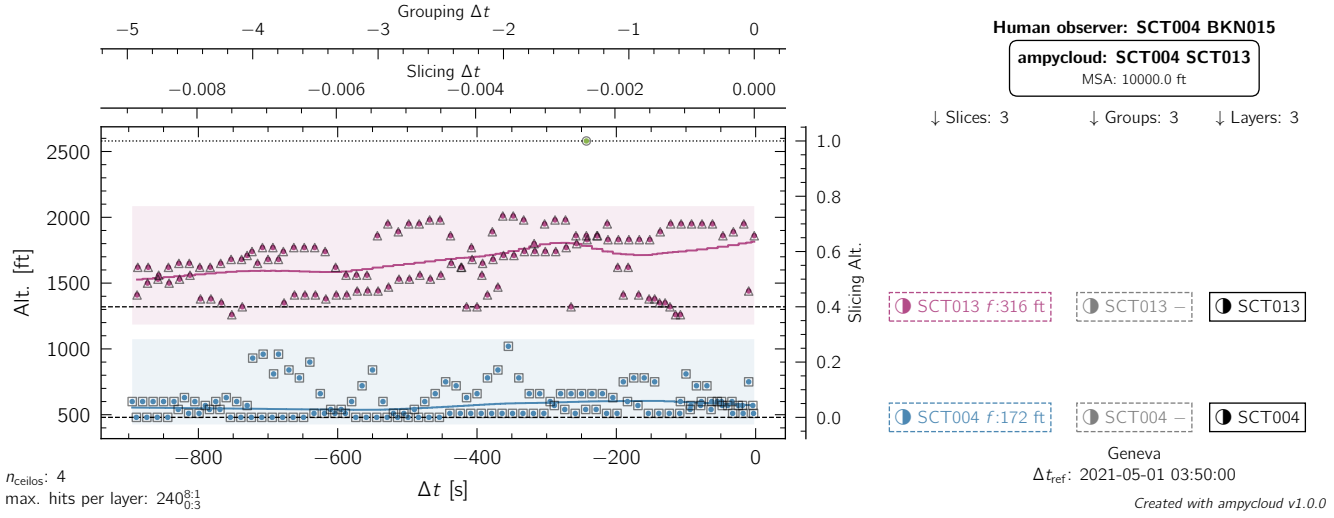


Figure B3. Same as Fig. B1, but for LSGG on 01 May 2021 at 03:50 UTC. In this case, the slicing step correctly identifies the two dominant cloud layers present. The grouping and layering step do not modify the initial slices. The top SCT013 layer slightly underestimates the sky coverage fraction that was reported to be BKN by the human observer, plausibly because of its obscuration by the lower layer.

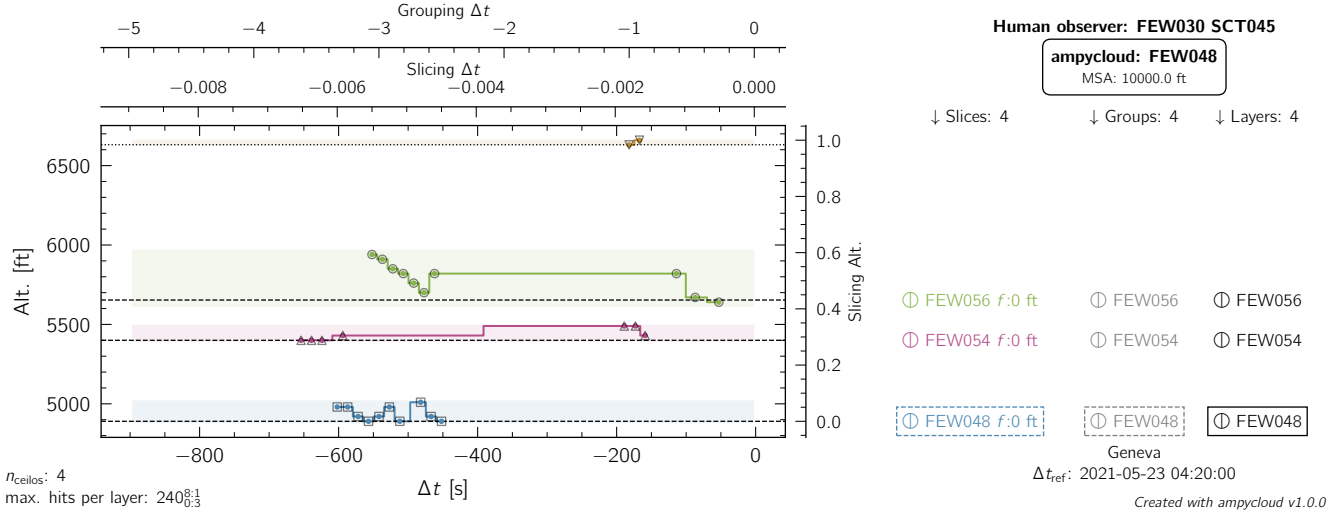


Figure B4. Same as Fig. B1, but for LSGG on 23 May 2021 at 04:20 UTC. In this case with very sparse layers, the slicing step correctly identifies the dominant cloud layers present. The layer at 3'000 ft reported by the observer is being missed entirely by the ceilometers, while the sky coverage fraction of the layer at 4'500-4'800 ft is being slightly underestimated by the ceilometers.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

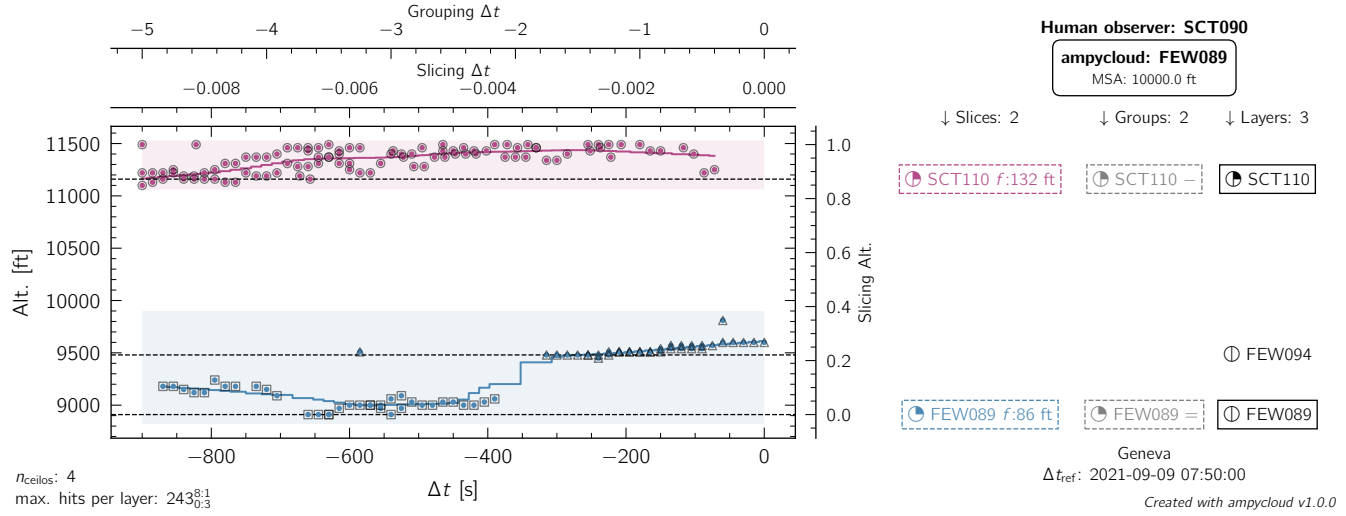


Figure B5. Same as Fig. B1, but for LSGG on 09 September 2021 at 07:50 UTC. The ampycloud layering step identifies two sub-layers separated by ∼500 ft in the bottom group, with a sky coverage fraction of FEW smaller than the SCT090 reported by the human observer, who likely merged the two sub-layers together. It is however worth noting that even the group FEW089 identified by ampycloud does not reach a sky coverage fraction of SCT, indicating that the ceilometers were globally looking at clear sightlines. The top layer SCT110 detected by ampycloud is not reported in view of the MSA applicable at the LSGG aerodrome.

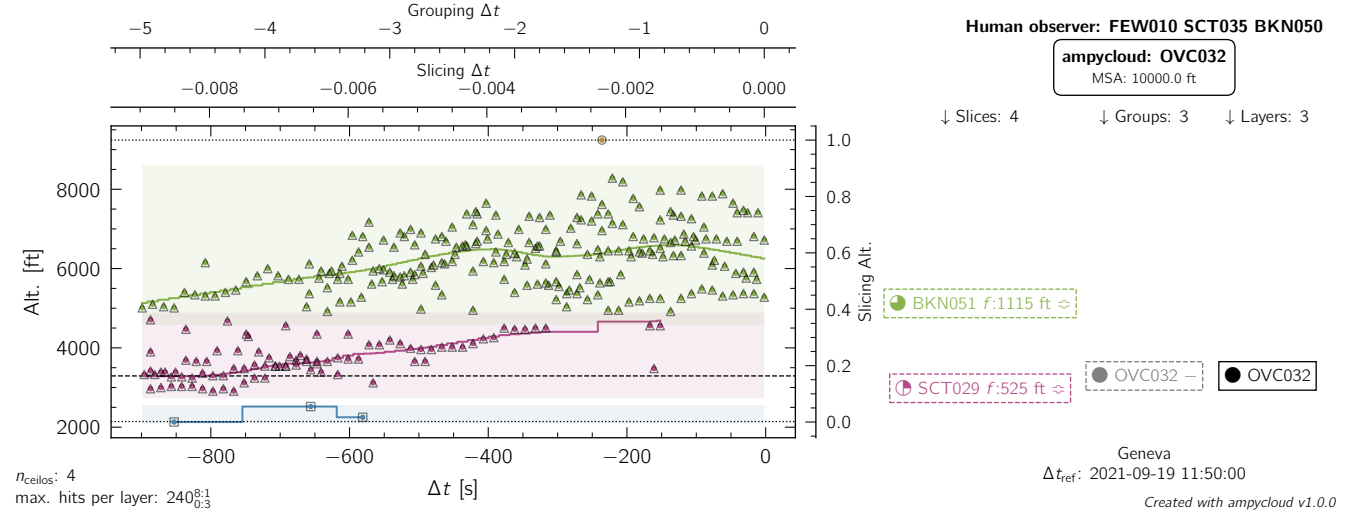


Figure B6. Same as Fig. B1, but for LSGG on 19 September 2021 at 11:50 UTC. The human observer reported two distinct layers at 3'500 ft and 5'000 ft, consistent with the initial ampycloud slices. The overall cloud hit distribution is however suggestive of a single coherent structure increasing from 3000 ft to 5'000 ft over the 15 min interval, and is identified as such by the grouping step of the ampycloud algorithm. The bottom layer at 1'000 ft is missed entirely by the ceilometers.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

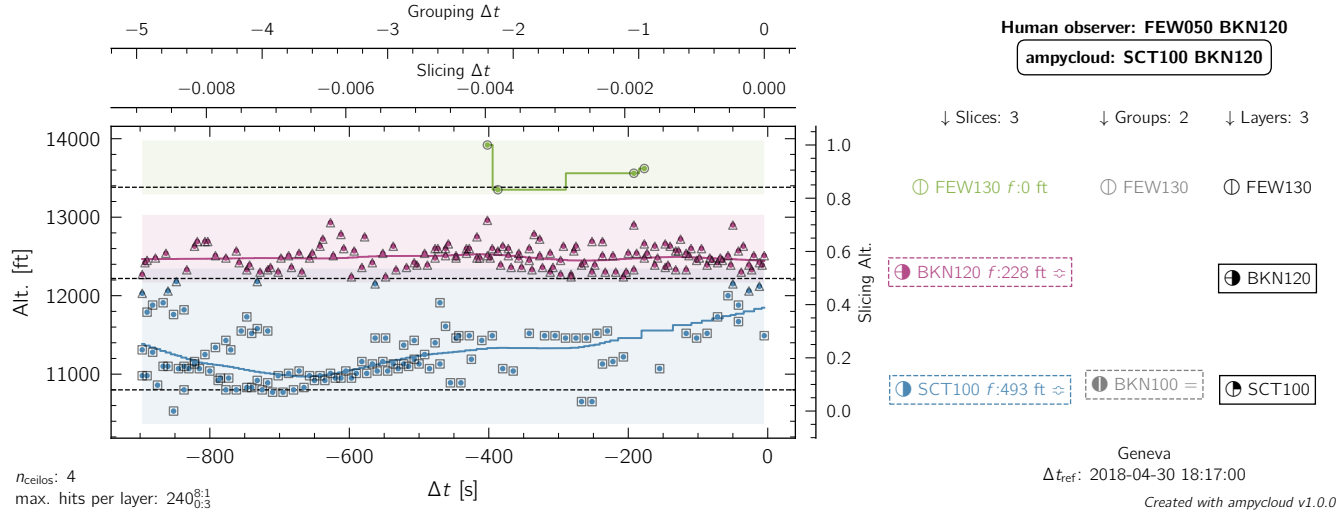


Figure B7. Same as Fig. B1, but for LSGG on 30 April 2018 at 18:17 UTC. The bottom two slices are merged into a single structure by the grouping step, then re-separated into two layers very similar to the original slices. This example was run without specifying any MSA, on the basis that the human observer reported the layer at 12'000 ft despite LSGG's MSA of 10'000 ft. Clouds at 5'000 ft were missed entirely by the ceilometers.

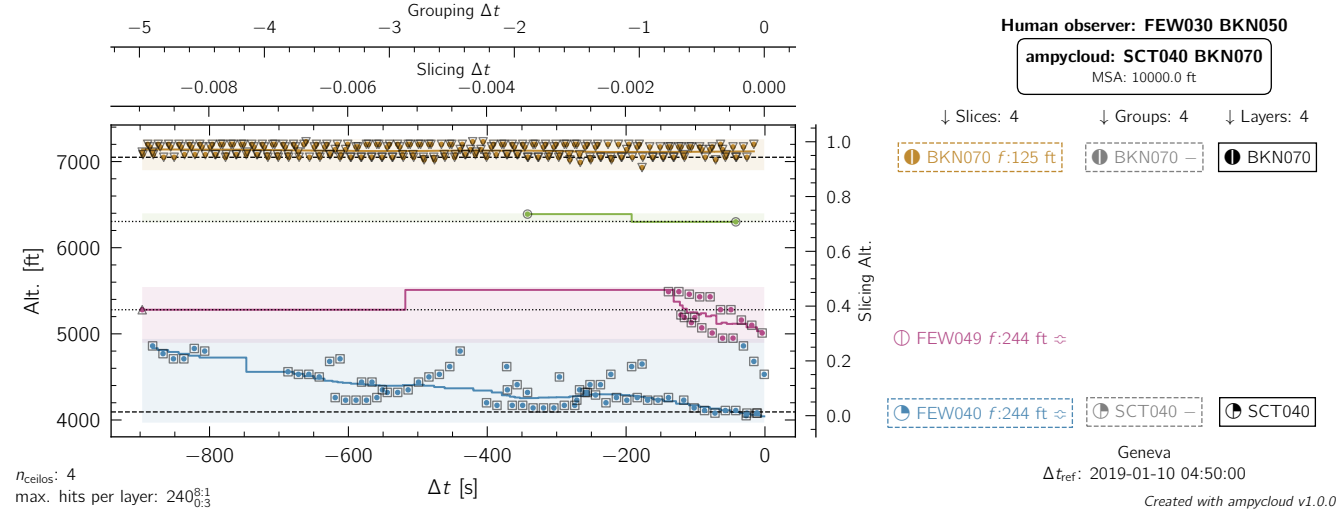


Figure B8. Same as Fig. B1, but for LSGG on 10 January 2019 at 04:50 UTC. It is because ampycloud accounts for the fluffiness of the bottom two slices that the grouping step decides that these form a single structure, despite the somewhat offset cluster of hits appearing within the last 150 s. It is not clear why the layer at 7'000 ft was not reported in the METAR, while clouds at 3'000 ft were missed entirely by the ceilometers.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

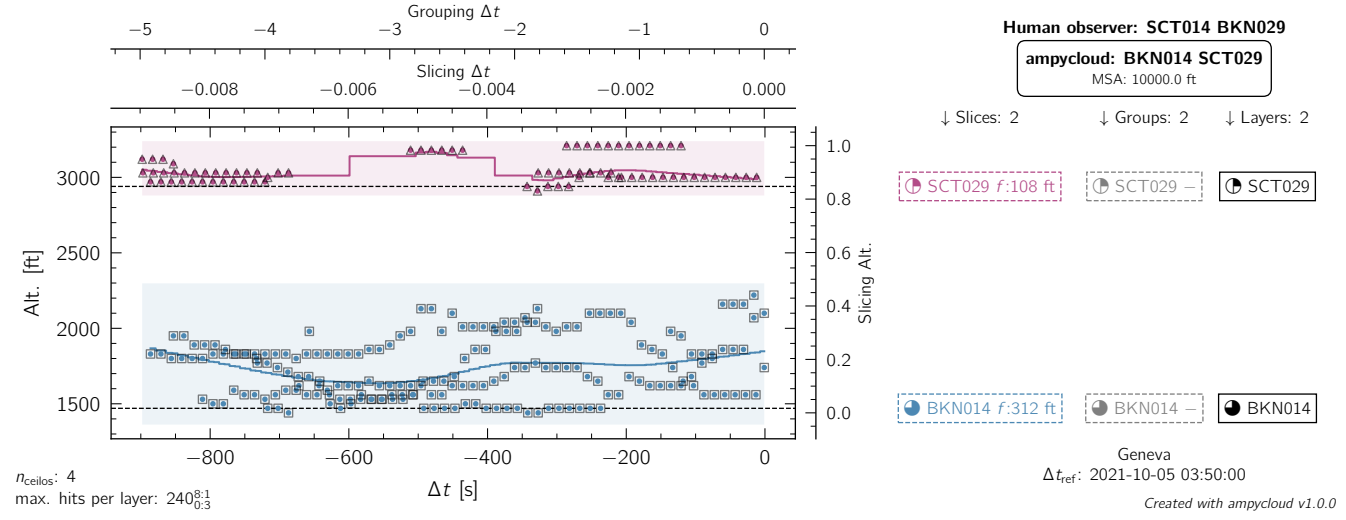


Figure B9. Same as Fig. B1, but for LSGG on 05 October 2021 at 03:50 UTC. The slicing step correctly identifies the two cloud layers present, albeit with a slightly different sky coverage fraction than those reported by the human observer. The top group is not being separated in 2 distinct sub-layers by the layering step because their separation of ∼180 ft would be smaller than the minimum value of $\Delta h_{l,\mathrm{vals}} = 250$ ft.

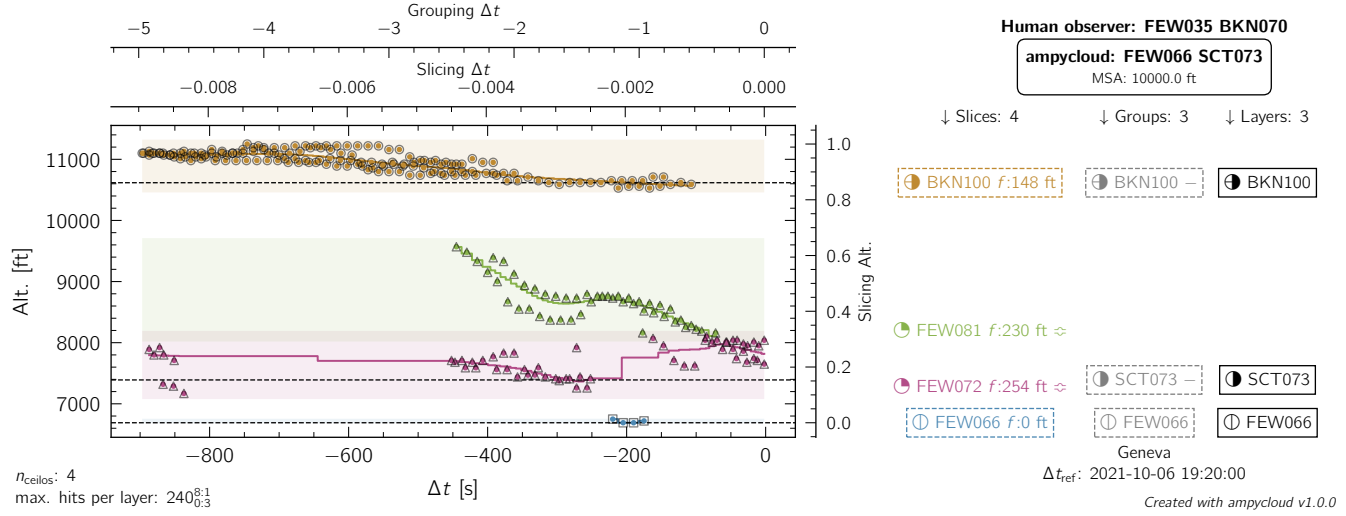


Figure B10. Same as Fig. B1, but for LSGG on 06 October 2021 at 19:20 UTC. The grouping step is key to connect the 2 central slices given the rapidly decreasing nature of the cloud base between −400 s and −100 s. Once again, the layer FEW035 reported by the human observer is being missed entirely by the ceilometers. One should note, however, that FEW (and SCT) layers are not operationally relevant as Low Visibility Procedure (LVP) rules applied by Air Traffic Control (ATC) require the presence of a ceiling (BKN or OVC).

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

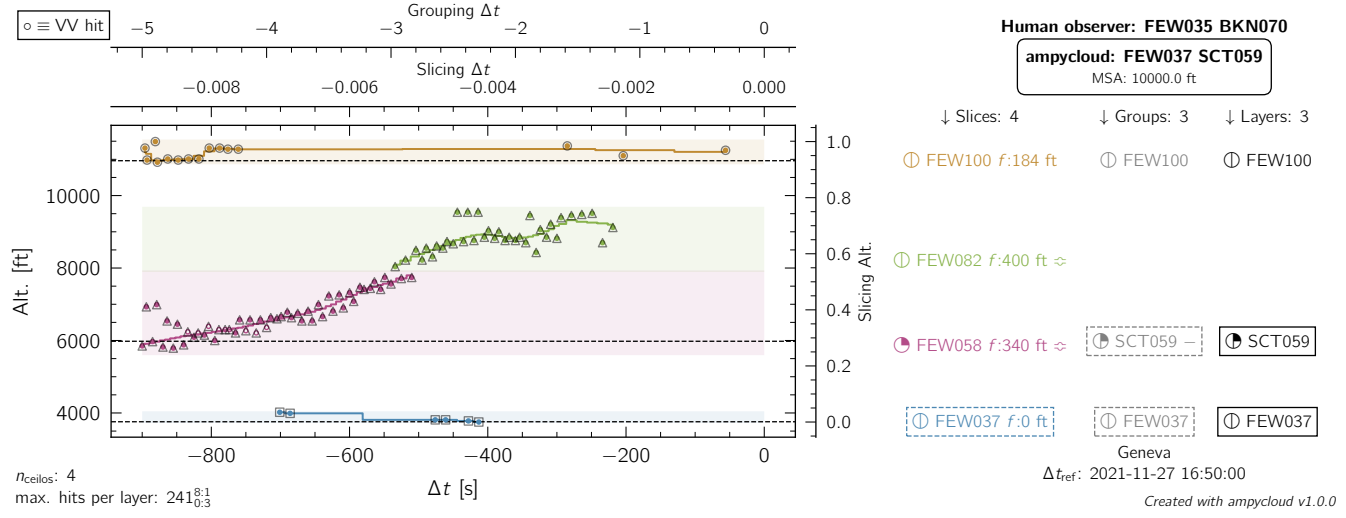


Figure B11. Same as Fig. B1, but for LSGG on 27 November 2021 at 16:50 UTC. The grouping correctly joins the middle two slices in this case with a rapidly rising cloud base. The cloud base of this layer reported by ampycloud is lower than that reported by the human observer, given that no weighting is being applied by ampycloud to derive it (see Sec. 2.4).

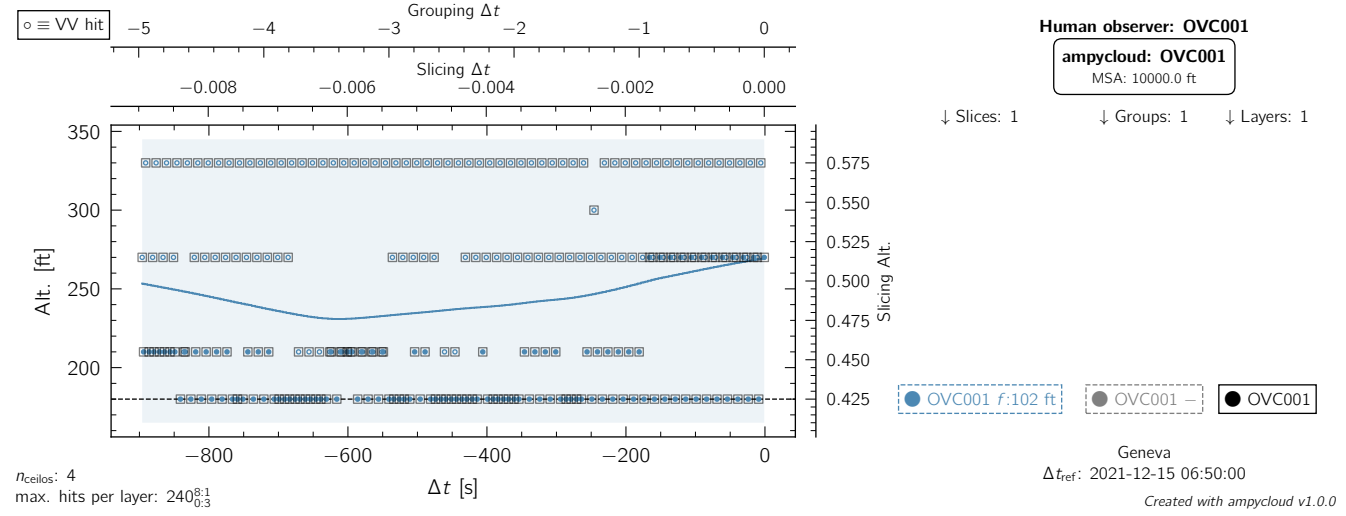


Figure B12. Same as Fig. B1, but for LSGG on 15 December 2021 at 06:50 UTC. A single OVC001 slice is (correctly) identified by the ampycloud algorithm in this case with a large number of VV hits reported by the ceilometers. We stress however once again that ampycloud is not intended nor designed to formally decide whether a VV code must be issued in the AUTO METAR (instead of OVC001 in this example). This task should be performed by a separate algorithm focusing on vertical visibility detection and reporting.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024


Atmospheric Measurement Techniques Discussions

Competing interests. The authors declare no competing interests.

500 *Acknowledgements.* We thank Kenneth Boutin and Chet Schmitt for sharing with us the detailed description of the latest sky-condition algorithm used by NOAA's ASOS Program. Diagrams in this article have been generated using the ampycloud Python module, which relies on the following Python packages: matplotlib (Hunter, 2007), numpy (Harris et al., 2020), pandas (McKinney, 2010; The pandas development team, 2021), scikit-learn (Pedregosa et al., 2011), scipy (Virtanen et al., 2020), and statsmodels (Seabold and Perktold, 2010). The ampycloud diagrams were enhanced using the metsymb LaTeXpackage (Vogt, 2023a).

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024


Atmospheric Measurement Techniques Discussions

References

Aebi, C., Gröbner, J., and Kämpfer, N.: Cloud Fraction Determined by Thermal Infrared and Visible All-Sky Cameras, Atmospheric Measurement Techniques, 11, 5549–5563, https://doi.org/10.5194/amt-11-5549-2018, 2018.

ampycloud: Ampycloud Online Documentation, https://meteoswiss.github.io/ampycloud, 2024a.

ampycloud: Ampycloud Github Repository, https://github.com/MeteoSwiss/ampycloud, 2024b.

ampycloud: Ampycloud Speed Test Action, https://github.com/MeteoSwiss/ampycloud/actions/workflows/CI_speed_check.yml, 2024c.

ampycloud: Ampycloud Speed Test Result, https://meteoswiss.github.io/ampycloud/installation.html\#testing-the-installation-speed-benchmark, 2024d.

ampycloud: Ampycloud Scientific Stability Tests, https://github.com/MeteoSwiss/ampycloud/blob/develop/test/ampycloud/test_scientific_stability.py, 2024e.

Aviolat, F., Cornu, T., and Cattani, D.: Automatic Clouds Observation Improved by an Artificial Neural Network, Journal of Atmospheric and Oceanic Technology, 15, 114–126, https://doi.org/10.1175/1520-0426(1998)015<0114:ACOIBA>2.0.CO;2, 1998.

Cleveland, W. S.: Robust Locally Weighted Regression and Smoothing Scatterplots, Journal of the American Statistical Association, 74, 829–836, https://doi.org/10.1080/01621459.1979.10481038, 1979.

Costa-Surós, M., Calbó, J., González, J. A., and Martin-Vide, J.: Behavior of Cloud Base Height from Ceilometer Measurements, Atmospheric Research, 127, 64–76, https://doi.org/10.1016/j.atmosres.2013.02.005, 2013.

de Haij, M., Apituley, A., Koestse, W., and Bloemink, H.: Transition towards a New Ceilometer Network in the Netherlands: Challenges and Experiences, in: TECO-2016 - WMO Technical Conference on Meteorological and Environmental Instruments and Methods of Observations, vol. Instruments and Observing Methods Report No. 125, World Meteorological Organization (WMO), Madrid, Spain, 2016.

Dürr, B. and Philipona, R.: Automatic Cloud Amount Detection by Surface Longwave Downward Radiation Measurements, Journal of Geophysical Research (Atmospheres), 109, D05 201, https://doi.org/10.1029/2003JD004182, 2004.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E.: Array Programming with NumPy, Nature, 585, 357–362, https://doi.org/10.1038/s41586-020-2649-2, 2020.

Hartley, B. and Quayle, T.: METAR AUTO Implementation at International Airports in New Zealand, in: TECO-2014 - WMO Technical Conference on Meteorological and Environmental Instruments and Methods of Observation, vol. IOM Report- No. 116, World Meteorological Organization (WMO), Saint Petersburg, Russia, 2014.

Hunter, J. D.: Matplotlib: A 2D Graphics Environment, Computing in Science and Engineering, 9, 90–95, https://doi.org/10.1109/MCSE.2007.55, 2007.

ICAO: Manual on Automatic Meteorological Observing Systems at Aerodromes, Tech. Rep. Doc 9837 AN/454, International Civil Aviation Organization, 2011.

ICAO: Meteorological Service for International Air Navigation, Annex 3 to the Convention on International Civil Aviation, Tech. Rep. AN 3, International Civil Aviation Organization, 2018.

JMA: Full Automation of Aeronautical Meteorological Observations and Reports at Aerodromes, Tech. rep., Japan Meteorological Agency, 2022.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Leroy, M.: Status of the Automatic Observation on Aerodrome and Ongoing Improvements in France, in: TECO-2006 - WMO Technical Conference on Meteorological and Environmental Instruments and Methods of Observation, vol. WMO/TD-No. 1354, Instruments and Observing Methods Report No. 94, World Meteorological Organization (WMO), Geneva, 2006.

Marty, C. and Philipona, R.: The Clear-Sky Index to Separate Clear-Sky from Cloudy-Sky Situations in Climate Research, Geophysical Research Letters, 27, 2649–2652, https://doi.org/10.1029/2000GL011743, 2000.

McKinney, W.: Data Structures for Statistical Computing in Python, in: {P}roceedings of the 9th {P}ython in {S}cience {C}onference, pp. 56–61, https://doi.org/10.25080/Majora-92bf1922-00a, 2010.

MeteoSwiss: Flugwetter Jahresbericht, Tech. rep., MeteoSwiss, 2022.

Nadolski, V. L.: Automated Surface Observing System (ASOS) User's Guide, Tech. rep., National Oceanic and Atmospheric Administration, Department of Defense, Federal Aviation Administration, and United States Navy, 1998.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E.: Scikit-Learn: Machine Learning in {P}ython, Journal of Machine Learning Research, 12, 2825–2830, 2011.

Seabold, S. and Perktold, J.: Statsmodels: Econometric and Statistical Modeling with Python, in: Proc. of the 9th Python in Science Conference, pp. 57–61, 2010.

The pandas development team: Pandas-Dev/Pandas: Pandas, Zenodo, https://doi.org/10.5281/zenodo.4452601, 2021.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., and van Mulbregt, P.: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods, 17, 261–272, https://doi.org/10.1038/s41592-019-0686-2, 2020.

Vogt, F. P. A.: Metsymb, https://doi.org/10.5281/zenodo.8302082, 2023a.

Vogt, F. P. A.: Ampycloud: Example Datasets, https://doi.org/10.5281/zenodo.10171152, 2023b.

Vogt, F. P. A. and Regenass, D.: Ampycloud, MeteoSwiss, https://doi.org/10.5281/zenodo.10125514, 2023.

Wacker, S., Gröbner, J., Zysset, C., Diener, L., Tzoumanikas, P., Kazantzidis, A., Vuilleumier, L., Stöckli, R., Nyeki, S., and Kämpfer, N.: Cloud Observations in Switzerland Using Hemispherical Sky Cameras, Journal of Geophysical Research: Atmospheres, 120, 695–707, https://doi.org/10.1002/2014JD022643, 2015.

Wagner, T. J. and Kleiss, J. M.: Error Characteristics of Ceilometer-Based Observations of Cloud Amount, Journal of Atmospheric and Oceanic Technology, 33, 1557–1567, https://doi.org/10.1175/JTECH-D-15-0258.1, 2016.

Wauben, W.: Automation of Visual Observations at KNMI: (II) Comparison of Automated Cloud Reports with Routine Visual Observations, in: Symposium on Observations, Data Assimilation and Probabilistic Prediction, AMS Annual Meeting, Orlando (FL), United States, 2002.

Wauben, W., Klein Baltink, H., de Haij, M., Maat, N., and The, H.: Status, Evaluation and New Developments of the Automated Cloud Observations in the Netherlands, in: TECO-2006 - WMO Technical Conference on Meteorological and Environmental Instruments and Methods of Observation, vol. WMO/TD-No. 1354, Instruments and Observing Methods Report No. 94, World Meteorological Organization (WMO), Geneva, 2006.

Willemse, S. and Furger, M., eds.: From Weather Observations to Atmospheric and Climate Sciences in Switzerland, vdf Hochschulverlag an der ETH Zürich, Zürich, aktualisierte version märz 2017 edn., 2017.

https://doi.org/10.5194/amt-2023-254
Preprint. Discussion started: 22 January 2024

Atmospheric Measurement Techniques Discussions

580 WMO, ed.: Guide to Instruments and Methods of Observation, Volume I - Measurement of Meteorological Variables, Chapter 15 - Observation and Measurement of Clouds, WMO-No. 8, WMO, Geneva, 2021 edn., ISBN 978-92-63-10008-5, 2021.

WMO, ed.: Aerodrome Reports and Forecasts: A Users' Handbook to the Codes, WMO-No.782, WMO, Geneva, 2022 edn., ISBN 978-92-63-10782-4, 2022.