Tracking Low-Level Cloud Systems with Topology

Mingzhe Li[*] Dwaipayan Chatterjee[†] Franziska Glassmeier[‡]
University of Utah Karlsruhe Institute of Technology Delft University of Technology
Fabian Senf[§] Bei Wang[¶]
Leibniz Institute for Tropospheric Research University of Utah

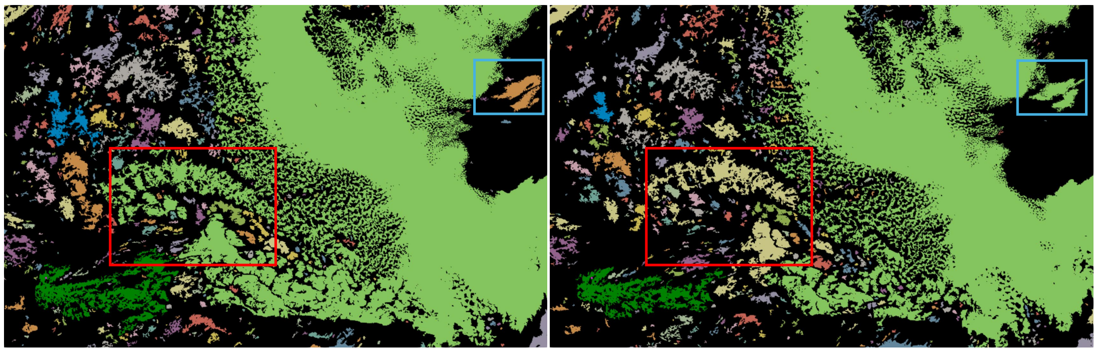

Figure 1: Topology-driven cloud tracking for a marine stratocumulus dataset over the ocean west of Africa. The two images show the detected cloud systems at 09:00 and 10:00 UTC on August 1, 2023. Color encodings indicate correspondences between cloud systems across the one-hour interval, allowing us to observe their evolution as they appear, disappear, merge, and split. Red and cyan boxes highlight examples of a splitting event and a merging event, respectively.

Abstract

Low-level clouds are ubiquitous in Earth's atmosphere. Their response to atmospheric conditions are essential to understanding the climate system and its sensitivity to anthropogenic influences. High-resolution geostationary satellites now resolve cloud systems with unprecedented detail, promoting cloud tracking as a vital research area for studying their spatiotemporal dynamics. It enables disentangling advective and convective components driving cloud evolution. This, in turn, provides deeper insights into the structure and lifecycle of low-level cloud systems and the atmospheric processes that govern them. In this paper, we propose a novel framework for tracking cloud systems using topology-driven techniques based on optimal transport. We first obtain a set of anchor points for the cloud systems based on the merge tree of the cloud optical depth field. We then apply topology-driven probabilistic feature tracking of these anchor points to guide the tracking of cloud systems. We demonstrate the utility of our framework by tracking clouds over the ocean and land to test for systematic differences in the two physically distinct settings. We further evaluate our framework through case studies and statistical analyses, comparing it against two leading cloud tracking tools and two topology-based general-purpose tracking tools. The results demonstrate that incorporating system-based tracking improves the ability to capture the evolution of low-level clouds. Our framework will inform low-level cloud characterization studies that fully profit from detailed satellite data.

Keywords: Feature tracking, merge tree, optimal transport, topology in data visualization, topological data analysis, applications

[*]e-mail: mingzhe.li@utah.edu
[†]e-mail: dwaipayan.chatterjee@kit.edu
[‡]e-mail: f.glassmeier@tudelft.nl
[§]e-mail: senf@tropos.de
[¶]e-mail: beiwang@sci.utah.edu
[*†] These authors contributed equally as co-first authors.

1 Introduction

Low-level clouds (i.e., clouds with a base below 6,500 ft and limited vertical extend) are an important part of the atmosphere by transporting heat and moisture, and by interacting with radiation [58]. Understanding the structure and evolution of low-level clouds is key to improving our knowledge of atmospheric processes. Low-level clouds such as *shallow cumulus* clouds (colloquially known as *fair weather clouds* with their cotton-ball-like appearance) and *stratocumulus* clouds (gray or whitish cloud decks patterned by dark, cloud-free lines or cells [62]) significantly affect the Earth's radiative budget because they form cloud fields that can stretch over hundreds of kilometers. Due to their fine-scale structure, however, their accurate representation in climate models remains a significant challenge and a dominant contribution to the uncertainty in climate projections [36, 42]. Accurately describing these clouds in space as well as time is thus a prerequisite for their reliable representation in climate models. This emphasizes the importance of cloud characterization and tracking [44].

Cloud evolution is shaped by *convection* and *advection*. Convective motion, driven by buoyancy and atmospheric instability, especially governs local-scale vertical motion. These dynamics influence cloud growth, dissipation, and structural changes. In contrast, advection refers to the large-scale horizontal transport of clouds by prevailing winds, guided by atmospheric circulation patterns. Cloud tracking, therefore, helps differentiate convective and advective processes [1, 51], enabling studies on cloud dynamics [51], lifecycles [49], microphysical processes [14], and the evolution of cloud patterns [24]. What makes it interesting and different from classical computer vision problems is that clouds are a representation of a continuous process (such as condensation, evaporation, split, and merge) and cannot be treated as a fixed object [6]. Clouds occur in moist turbulent flow, which results in fractal characteristics of cloud boundaries, which in turn makes individual clouds difficult to identify and track. Recent progress in tracking methods, driven by growing satellite data records, has made it possible to study cloud characteristics in greater detail and improve our understanding of their behavior.

We present a novel topology-driven framework for tracking low-level clouds. We model low-level clouds as cloud systems that may consist of multiple cloud objects that are geometrically close, and use probabilistic feature tracking based on optimal transport to track them in a time-varying setting.

Our contributions are as follows:

- We present a new framework to track cloud systems using time-varying satellite image data. We first obtain a set of anchor points for the cloud systems based on the merge tree of the cloud optical depth field. We then apply merge-tree-based feature tracking of the anchor points to guide the tracking of cloud systems.
- We demonstrate the utility of our framework by tracking cloud systems from satellite data over both ocean and land, recognizing that cloud-tracking challenges may differ across these two physically distinct regimes.
- We further compare our framework with two leading cloud tracking tools and two topology-based general-purpose tracking tools via visualizations and statistical evaluations.

The source code is publicly available at `https://github.com/tdavislab/cloud-tracking`.

2 Related Work: Cloud Physics and Tracking

Clouds form and evolve across various spatiotemporal scales within Earth's atmosphere [31]. Low-level clouds in the warm regions of the lower latitudes are composed of liquid droplets, which makes them very effective at reflecting incoming sunlight back to space. This cooling effect is not strongly compensated for by a reduced emissivity of heat from the surface as low-level clouds re-emit absorbed long-wave radiation at cloud-top-temperature that are not much colder than the surface. Low-level clouds over land or ocean exhibit distinct characteristics. Over land convection is affected by surface inhomogeneities and strong surface heating, which leads to pronounced diurnal cycles. In contrast, oceanic convection is more organized and sustained, shaped by the ocean's high heat capacity, abundant moisture, uniform temperatures, and large-scale atmospheric circulation.

Advanced Geostationary Satellites (GS), such as Meteosat Third Generation (MTG [19]) and GOES-16's Advanced Baseline Imager (ABI [57]), provide high spatial and temporal resolution. However, they still do not resolve the fine-scale structure of low-level cloud systems [16, 17], which often span scales of hundreds of meters. Despite this limitation, treating the aggregation of these individual cloud objects—spanning hundreds of kilometers—as spatial distributions enables adequate resolution to capture their variability [4, 21].

Cloud tracking begins with the identification of individual clouds in the dataset, a process that depends on the type of cloud and the specific scientific objectives of the study. For instance, studies focusing on deep convective cells often rely on physical threshold-based approaches, such as brightness temperature [7, 12, 51, 52] or radar reflectivity [26, 41]. In the case of mixed-phase clouds, methods typically involve a combination of cloud mask (cloudy or non-cloudy pixel) and cloud optical depth [5]. Cloud optical depth (COD) quantifies the extent to which the cloud attenuates light primarily due to the scattering and absorption by cloud droplets. It is governed by cloud geometric thickness, cloud water mass, droplet concentration, and particle size distribution [39]. For shallow low-level clouds, cloud identification methods vary: [49] employs a cloud mask, while [25] applies a reflectivity threshold. However, shallow cumulus clouds, being inherently broken and scattered in nature, pose additional challenges. The resolution of the cloud mask, such as that provided by the CLAAS-2 product [37], may be insufficient to fully resolve these fragmented cloud structures.

The present and upcoming generations of geostationary satellites provide continuous, high spatiotemporal resolution observations, significantly advancing our ability to study rapidly evolving dynamic systems in the Earth's atmosphere. Historically, tracking approaches using geostationary satellite data have focused on mesoscale convective systems. Techniques for tracking clouds between successive observations range from manual methods [23, 50, 51] to automated approaches such as spatial correlation [2, 8, 48] and area-overlapping methods [35, 59, 61]. In some cases, a combination of correlation and overlapping techniques has been applied to improve accuracy [47]. Furthermore, fully automated tracking methods have been developed, primarily focusing on deep convective clouds to analyze mesoscale clusters [12] or to establish a generalized framework for diverse Earth system datasets, enhancing both adaptability and computational efficiency [33].

Individual clouds in a shallow cumulus cloud fields are not randomly distributed but organized into clustered patterns [21, 55]. Their broken structure leads to distinct patterns of cloud shadows and illuminations [15, 30]. [49] uses the Spinning Enhanced Visible and InfraRed Imager [57] on board the European geostationary Meteosat Second Generation (MSG) satellites and employs particle image velocimetry to track these clouds over the ocean but does not account for low-level clouds' splitting and merging behaviors. This phenomenon becomes particularly important when a cloud grows and merges with neighboring clouds, or when a complex cloud splits into smaller ones. Similarly, [25] uses the Advanced Himawari Imager (AHI) rapid scan onboard HIMAWARI-8 [22] and applies the Kalman filter as a motion prediction model to estimate the locations of cloud objects across successive time frames. However, their approach assumes a constant velocity field throughout the tracking process. Complementing observational studies, extensive research has been conducted to track simulated shallow cumulus clouds in large-eddy simulations [18, 56, 65], providing valuable insights into cloud lifecycle and dynamics.

3 Related Work: Topological Feature Tracking

Topology-based feature tracking for time-varying scalar fields is a two-step process: first, topological features are extracted at each time step; and second, these features are matched between adjacent time steps by solving a correspondence problem (or assignment problem in cloud science). Various topological descriptors—such as merge trees and persistence diagrams—have been used for feature tracking; see [64, Section 7.1] for a review. Soler et al. [34, 53] used persistence diagrams to perform topology tracking. They extracted points in a persistence diagram that encode homological features at each time step, and relied on lifted Wasserstein [53] or Wasserstein [34] matching between persistence diagrams at adjacent time steps to establish correspondences.

The main idea of merge-tree-based tracking is to follow the nodes of merge trees that represent critical points of the underlying scalar fields, using tree matching to establish correspondences. Doraiswamy et al. [7] combined merge trees with optical flow to track deep convective clouds. Pont et al. [40] extended the work on edit distance [54] and introduced a new Wasserstein metric between merge trees to support feature tracking. Yan et al. [63] used the labeled interleaving distance between merge trees to support geometric-aware feature tracking.

Most recently, Li et al. [29] introduced a probabilistic framework for tracking topological features using merge trees and optimal transport. They represented a merge tree as a measure network—a network associated with a probability distribution—and introduced a distance metric for comparing merge trees using partial optimal transport. This distance offers flexibility in capturing both intrinsic and extrinsic information of merge trees.

A traditional method for establishing correspondence between features is to calculate the overlap between regions surrounding the features. Lukasczyk et al. [20, 32] matched superlevel set components by measuring the overlap between their corresponding regions. Similarly, Saikia et al. [45, 46] performed topological feature tracking using merge trees. Their approach assesses the similarity of subregions segmented by merge trees at adjacent time steps, based on the overlap size between two regions and the similarity between histograms of scalar values within each region.

4 Technical Background

4.1 Merge Tree

Let $f : \mathbb{M} \to \mathbb{R}$ be a scalar field defined on a 2D domain $\mathbb{M} \subset \mathbb{R}^2$. A merge tree captures the connectivity among sublevel sets of f. We consider two points $x, y \in \mathbb{M}$ to be equivalent, denoted $x \sim y$, if $f(x) = f(y) = a$ and they belong to the same connected component of the sublevel set $f^{-1}(-\infty, a]$. Mathematically, the merge tree is a quotient space $T(\mathbb{M}, f) = \mathbb{M}/\sim$. For topology-based cloud tracking, f corresponds to the cloud optical depth (COD) field in a satellite image with a particular timestamp; it quantifies how much a ray of light is attenuated as it travels through a cloud. A higher optical depth indicates greater extinction of light within the cloud. Since we are interested in high-value areas of f, we work with the merge tree of $-f$, as shown in Fig. 2.

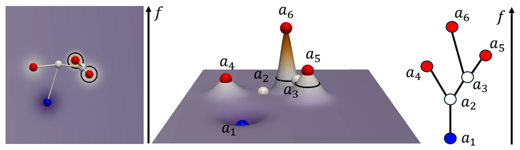

Figure 2: Left: a 2D visualization of a scalar field f with an embedded merge tree of $-f$. Middle: a 3D visualization of the graph of f. Right: an abstract visualization of the merge tree of $-f$. Local maxima are in red, saddles are in white, and the global minimum is in blue. The black contour passing through the saddle a_3 encloses two peak areas of the local maxima a_5 and a_6, respectively.

As illustrated in Fig. 2, we construct a merge tree as follows. We sweep the graph of f (middle) with a hyperplane at the function value a starting from the maximal value of f. As a decreases, we initiate a new branch of the merge tree each time we encounter a local maximum (e.g., at a_6, a_5, and a_4). Such a branch grows longer as a decreases, eventually merging with another branch at a saddle point. For instance, at the saddle a_3, the branch starting at a_6 merges with the branch starting at a_5. Given a simply connected domain, all branches eventually merge into a single connected component at the global minimum a_1. Leaves, internal nodes, and the root of the tree correspond to the local maxima, saddles, and the global minimum of f, respectively. With a slight abuse of notation, the merge tree $T = (V, E)$ is a rooted tree whose node set V is equipped with the scalar function f.

We define a mapping $\phi : \mathbb{M} \to T$ between points in the domain and the merge tree. The inverse image of an edge $e \in E$ under this mapping $\phi^{-1}(e)$ is called a *topological zone*. For example, the two peak areas enclosed by the black contour in Fig. 2 are the topological zones of the edges a_3a_6 and a_3a_5 in E, respectively. The area of a topological zone can serve as an *importance* measure for an edge in the merge tree [27], as it reflects the size of the peak in the domain. We may simplify branches with small topological zones during computation by employing this importance measure.

4.2 Feature Tracking with Optimal Transport

Li et al. [29] introduced topology-based feature tracking based on partial optimal transport. Our framework utilizes and significantly extends the work of Li et al. [29], making it suitable for cloud tracking. The key idea in [29] is the introduction of partial Fused Gromov-Wasserstein (pFGW) distance between merge trees. The pFGW distance generates a probabilistic matching between nodes in a pair of merge trees; such a matching serves as the starting point for deriving trajectories of the cloud systems in downstream analysis.

Optimal transport in a nutshell. To illustrate optimal transport, assume there are a number of factories with specific production capacities and a number of warehouses with prescribed storage capacities. Optimal transport aims to find the most efficient way to transport goods from the factories to the warehouses by minimizing the transportation cost while respecting the capacity constraints. For partial optimal transport, we allow losing a certain amount of goods during transportation.

Measure network. Following [29], we model merge trees as measure networks. That is, a merge tree can be represented as a triple $T = (V, p, W)$, where $p : V \to [0, 1]$ is a probability measure on the node set V (i.e., $\sum_{x \in V} p(x) = 1$ for all $x \in V$), and $W : V \times V \to \mathbb{R}$ denotes the pairwise intrinsic node relation.

A measure network $T = (V, p, W)$ may also be equipped with node attributes from an attribute space (A, d_A) that encodes extrinsic information. An example of a node attribute is the geometric location of its corresponding critical point in the domain. In this context, the attribute distance d_A is the Euclidean distance between critical points in the domain. We will discuss our choices for p, W, and (A, d_A) in the context of cloud tracking in Sec. 5.3.

Partial Fused Gromov-Wasserstein distance. The pFGW distance introduced by Li et al. [29] is based on the theory of partial optimal transport [3, 60]. Given two measure networks $T_1 = (V_1, p_1, W_1)$ and $T_2 = (V_2, p_2, W_2)$ equipped with node attributes, let $n_1 = |V_1|$ and $n_2 = |V_2|$ be the number of nodes. A coupling $C \in \mathbb{R}^{n_1 \times n_2}$ is a nonnegative matrix that encodes a joint probability measure between p_1 and p_2, with row and column marginals equal to p_1 and p_2, respectively. Formally, the set $\mathcal{C} = \mathcal{C}(p_1, p_2)$ of all couplings between T_1 and T_2 is

$$\mathcal{C}(p_1, p_2) = \{C \in \mathbb{R}_+^{n_1 \times n_2} \mid C\mathbf{1}_{n_2} = p_1, C^\top \mathbf{1}_{n_1} = p_2\}, \quad (1)$$

where $\mathbf{1}_n = (1, 1, ..., 1)^\top \in \mathbb{R}^n$. Following (1), optimal transport requires a coupling (matching) to preserve all measures p_1 and p_2.

On the other hand, partial optimal transport [3] allows partial coupling, thus partial matching between two measure networks. It relaxes the requirement for the coupling to sum to a number $m \leq 1$ (i.e., $m \in [0, 1]$). The set $\mathcal{C}_m = \mathcal{C}_m(p_1, p_2)$ of the *relaxed couplings* is

$$\begin{aligned}&\mathcal{C}_m(p_1, p_2)\\ &= \{C \in \mathbb{R}_+^{n_1 \times n_2} \mid C\mathbf{1}_{n_2} \leq p_1, C^\top \mathbf{1}_{n_1} \leq p_2, \mathbf{1}_{n_1}^T C\mathbf{1}_{n_2} = m\}.\end{aligned} \quad (2)$$

The pFGW distance is defined on the set of relaxed couplings $\mathcal{C}_m$.

Given a pair of merge trees modeled as measure networks T_1 and T_2, the pFGW distance is defined as

$$\begin{aligned}d_q(T_1, T_2) = \min_{C \in \mathcal{C}_m} \sum_{i,j,k,l} \Big(&(1-\alpha)\, d_A(a_i, b_j)^q \\ &+ \alpha\, |W_1(i,k) - W_2(j,l)|^q \Big) C_{i,j} C_{k,l}.\end{aligned} \quad (3)$$

Here, $d_A(a_i, b_j)$ is the node attribute distance between $a_i \in V_1$ and $b_j \in V_2$. $|W_1(i,k) - W_2(j,l))|$ describes the *structural distortion* when we match pairs of nodes $(a_i, a_k) \in T_1$ with $(b_j, b_l) \in T_2$. The pFGW distance incorporates a parameter α to balance the weights between these two components. In the context of matching a pair of merge trees, the pFGW distance provides flexibility in preserving both the node properties (such as critical point locations in the domain) and the merge tree structure in the optimal coupling. It also allows the appearance and disappearance of new features.

In the “factory-warehouse” scenario, we are in the setting of optimal transport when $m = 1$ in (3). p_1 and p_2 prescribe the capacities of factories and warehouses, respectively, and the coupling C describes a transportation plan that respects these capacity constraints. The transportation cost is described by the attribute distances (e.g., Euclidean distances) between factories and warehouses as well as the structural relations among them (e.g., factories owned by a given company should transport goods to warehouses owned by the same company). Solving an optimization problem of (3) means finding the transportation plan with the lowest cost. On the other hand, we are in the setting of partial optimal transport when $0 < m < 1$ in (3), meaning that we allow $1 - m$ percent of goods to be lost/ignored during transportation.

5 Method

Cloud tracking faces three major challenges. First, clouds observed in satellite images are complex, time-varying phenomena involving numerous events, as cloud systems appear, disappear, merge, and split. Second, there is no consensus among domain scientists on the definition of cloud objects and cloud systems. Third, there are no ground-truth cloud tracking results available for satellite images supporting any form of supervised learning.

In this section, we describe our novel framework of topology-driven cloud tracking. Working closely with domain scientists, we first introduce the definition and detection of cloud objects (Sec. 5.1). We then describe our strategy of using critical points as anchor points for cloud objects (Sec. 5.2). Subsequently, we compute a matching between the anchor points using partial optimal transport (Sec. 5.3). We then generate trajectories for cloud systems formed by (possibly) multiple cloud objects (Sec. 5.4).

Compared to the work of Li et al. [29], our novel framework uses customized node probability with existing knowledge of the cloud data. Additionally, we propose the concept of tracking cloud systems and introduce a matching algorithm for cloud systems based on optimal transport. Our choice of merge-tree-based tracking is justified by its effectiveness in capturing the locality structure among local maxima.

5.1 Detecting and Simplifying Cloud Objects

In a cloud optical depth (COD) field f from a satellite image, regions with high function values usually indicate thicker clouds.

Cloud object detection. We use a thresholding strategy for the detection of cloud objects. We define each connected component of a superlevel set of f at a chosen threshold a (i.e., $f^{-1}[a, \infty)$) as a *cloud object*. Currently, there is no consensus on the threshold value a to detect low-level clouds. This results in a lack of widely accepted ground-truth data for cloud detection and tracking. Following established practices [33], we test a range of thresholds from 0.5 to 5.0 at a gap of 0.5 to analyze the impact of the threshold on (a) the number of cloud objects, and (b) their size distributions. We select $a = 2.0$ to mitigate both under- and over-segmentation; further details on threshold selection are provided in the supplement.

Cloud object simplification. We want to separate features from noise in our real-world cloud data. Stratocumulus clouds often cover large, continuous areas that can stretch hundreds of kilometers; therefore, we may consider removing smaller cloud objects that are deemed insignificant from stratocumulus clouds. On the other hand, shallow cumulus clouds are characterized by their small size (covering hundreds of meters across), relatively flat bases, and puffy tops; they could also grow into deeper convective systems depending on the available convective mass flux [10], as seen in their COD values. To handle these differences, we use different approaches to simplify the identification of cloud objects for stratocumulus and shallow cumulus, respectively.

For stratocumulus (typically over the ocean), we simplify by discarding small cloud objects based on area; see Fig. 3. By ignoring cloud objects smaller than 10 pixels, we can get rid of more than 70% of cloud objects from the field and still cover more than 97% of the total cloud area; see supplement for statistical details. Fig. 3 gives an example of area-based simplification. The cloud area maps show cloud objects in gray and the background in black. In the simplified cloud area map, cloud objects smaller than 10 pixels are ignored after simplification (cf. the green boxes).

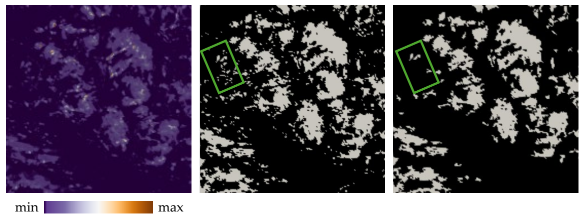

Figure 3: Apply area-based simplification to stratocumulus clouds. From left to right: the COD field, the cloud area map, and the simplified map by excluding cloud objects smaller than 10 pixels.

For Shallow cumulus (typically over land), instead of filtering by size, we apply a higher COD threshold to remove regions with low values, focusing only on the prominent clouds.

5.2 Attaching Anchor Points to Cloud Objects

We need to associate cloud objects with topological features to perform topology-driven tracking. To that end, we use nodes of merge trees that correspond to the critical points of the COD field f as anchor points for cloud objects.

To attach anchor points, we could associate a subtree of the merge tree to each cloud object. For example, Fig. 4(a) shows five cloud objects enclosed by the white contours. The tree structure within each cloud object is a subtree of the global merge tree. We use the local maxima in this subtree as the anchor points of the cloud object. These anchor points' trajectories are subsequently used to derive the trajectory of cloud objects. In practice, we may reduce the number of anchor points for computational efficiency; see Fig. 4(b) for an example and the supplement for technical details.

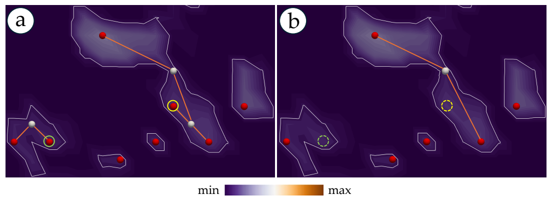

Figure 4: (a) A set of cloud objects enclosed by white contours; each contains a subtree of the global merge tree. Local maxima (a.k.a., anchor points) are in red, and saddles are in white. (b) Simplifying subtrees by removing the highlighted anchor points (inside green or yellow circles) and their parent saddles.

5.3 Anchor Point Tracking with Partial Optimal Transport

We adapt the work of Li et al. [29] to track anchor points with partial optimal transport. Building on prior knowledge of the characteristics of the COD field, we enhance this work by incorporating a tailored probability distribution for critical points.

Model merge trees as measure networks. We first model a merge tree T of the COD field f as an attributed measure network $T = (V, p, W)$ with node attributes (A, d_A). We modify the framework of [29] to focus on tracking local maxima of a merge tree, which act as anchor points for cloud systems.

For each local maximum $x \in V$, we set its probability as $p(x) = 1/n$, where n is the number of local maxima in T. For saddles and the global minimum, we assign a probability of 0 due to the high complexity and uncertainty of the COD field [43], which causes their locations to be highly unstable. This instability makes it challenging to find suitable matchings for these nodes. Furthermore, disregarding the probability of saddles and the global minimum enables us to preserve more anchor points without compromising computational efficiency, ultimately enhancing the robustness of tracking.

We use the pairwise node relation matrix W to encode the tree distance. Recall that each node $v \in V$ is equipped with a function value

$f(v)$. The tree distance between two adjacent nodes in T (i.e., the edge weight) is $W(a,b) = |f(a) - f(b)|$, whereas the tree distance between two nonadjacent nodes is the shortest path distance between them. Previous works [28,29] show that the tree distance can encode the scalar field topology via merge tree structures. Specifically, in the context of cloud tracking, the tree distance reflects the locality among the anchor points: anchor points attached to the same cloud object belong to the same subtree. Our framework captures anchor point locality inherently, regardless of whether saddles or the global minimum are preserved in the optimal transport.

We encode the locations of critical points as the node attributes. Given a pair of merge trees $T_1 = (V_1, p_1, W_1)$ and $T_2 = (V_2, p_2, W_2)$, the node attribute for $a_i \in V_1$ is (x_i, y_i), in which x_i and y_i denote the coordinates of the critical point. Similarly, the node attribute for $b_j \in V_2$ is (x_j, y_j). The attribute distance d_A between a_i and b_j is

$$d_A(a_i, b_j) = d_E\big((x_i, y_i), (x_j, y_j)\big) \tag{4}$$

d_E in Eq. (4) represents the Euclidean distance, and our framework prevents the matching of anchor points that are far apart.

Matching critical points with partial optimal transport. By computing the pFGW distance between a pair of merge trees T_1 and T_2 following (3), we obtain an optimal coupling C between their nodes. We interpret the coupling as a probabilistic matching between critical points from adjacent time steps. The sub-matrix of the coupling matrix, consisting only of rows and columns corresponding to local maxima, represents the matching between anchor points.

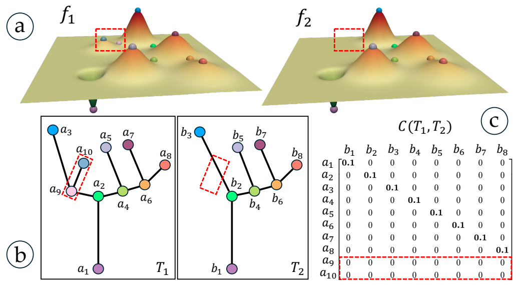

Figure 5: Partial optimal transport. We match the merge trees of $-f_1$ and $-f_2$ (b) using pFGW with $m = 0.8$, producing a coupling matrix C (c). Matched nodes in the two trees share the same color (a-b).

We provide a simple example in Fig. 5. Let f_1 and f_2 in (a) be the scalar fields of adjacent time steps. T_1 and T_2 in (b) are the merge trees of $-f_1$ and $-f_2$, respectively. Let p_1 and p_2 be uniform measures on all the nodes, including local maxima, saddles, and the global minimum. Based on structural similarity between T_1 and T_2, it is natural to match nodes $a_i \in T_1$ with $b_i \in T_2$ for $i \in [1, 8]$. On the other hand, nodes a_9 and a_{10} in T_1 are missing from T_2 (c.f., the red boxes). Ideally, we want to ignore a_9 and a_{10} during the matching process. Based on these intuitions, setting $m = 0.8$ in the pFGW distance produces the desired transportation plan captured by $C = C(T_1, T_2)$ in (c).

Each entry $C_{i,j}$ in C denotes the probability of matching $a_i \in T_1$ with $b_j \in T_2$. We see that $C_{i,i} = 0.1$, which aligns well with our expectations. Meanwhile, $C_{9,j} = C_{10,j} = 0$ for $j \in [1, 8]$, indicating that $a_9, a_{10} \in T_1$ are not matched to any nodes in T_2. A sub-matrix of C, with rows corresponding to a_3, a_5, a_7, a_8, a_{10} and columns corresponding to b_3, b_5, b_7, b_8, represents the probabilistic matching between anchor points. While this example shows a one-to-one node matching, our framework generally allows multiple nonzero entries in a row or column, which distinguishes our framework from other topology-based frameworks that produce one-to-one matchings.

5.4 Computing Trajectories for Cloud Systems

Constructing cloud systems. We first merge cloud objects into cloud systems. A cloud system may consist of multiple cloud objects that are geometrically close. Eytan et al. [9] suggested that most of the radiative effect of a cloud is confined within ~4km around the cloud. Therefore, we merge cloud objects within 4km away from each other as a cloud system and identify cloud objects farther than 4km as different cloud systems. The set of anchor points for each cloud system is the collection of anchor points for all cloud objects within the system.

Tracking cloud systems. As described in Sec. 5.2, each cloud system contains one or more local maxima as its anchor point. We can use the matching between the anchor points (see Sec. 5.3) to compute the trajectory for cloud systems.

We introduce a matching score between cloud systems at adjacent time steps. We denote the set of anchor points for a cloud system X as P_X. The matching probability from the optimal coupling between an anchor point v_1 at time step t and v_2 at time step $(t+1)$ is $C_t(v_1, v_2)$. Then, for the cloud systems X (at time step t) and Y (at time step $(t+1)$), the matching score between them is

$$S_t(X, Y) = \sum_{x \in P_X, y \in P_Y} C_t(x, y). \tag{5}$$

Informally, this score is the probability of mass transportation from X to Y. The higher this score is, the more likely the two are matched.

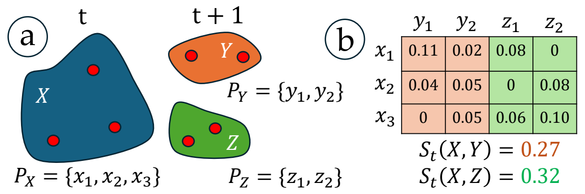

Figure 6: Cloud system matching score. (a) Cloud system X at time t and cloud systems Y and Z at time $(t+1)$ with their set of anchor points. (b) Selected rows and columns of the coupling matrix C.

Fig. 6 gives an example of matching scores involving cloud systems X, Y, and Z. In (a), there are three anchor points for X at time step t, and two for Y and Z at time step $(t+1)$, respectively. The coupling matrix C for the anchor point matching probability is in (b). For example, the matching probability between anchor point x_1 from X and y_1 from Y is 0.11. The matching score $S_t(X, Y)$ equals the sum of the first two (orange) columns for anchor points y_1 and y_2, which is 0.27. Similarly, $S_t(X, Z)$ equals the sum of the last two (green) columns for anchor points z_1 and z_2, which is 0.32.

With the matching scores, we can match the cloud systems via a bipartite graph matching algorithm. Let H_t denote the set of cloud systems at time step t. We use $S_t(X, \cdot) = \sum_{h \in H_{t+1}} S_t(X, h)$ to denote the total outgoing probability for a cloud system $X \in H_t$ and $S_t(\cdot, Y) = \sum_{h \in H_t} S_t(h, Y)$ the total incoming probability for $Y \in H_{t+1}$. A matching between the cloud system X and Y is *valid* if $S_t(X, Y)$ is nonzero and satisfies one of the two conditions:

1. $Y = \arg\max_{h \in H_{t+1}} S_t(X, h)$, and $X = \arg\max_{h \in H_t} S_t(h, Y)$;
2. $S_t(X, Y) \geq \max\{S_t(X, \cdot), S_t(\cdot, Y)\} \times r$.

Condition (1) means that X and Y are mutually the best match; otherwise, condition (2) implies that the matching score must exceed a threshold proportional to the maximum cumulative score of both X and Y. The proportionality factor is governed by the parameter r, which controls the strictness of the matching criteria. In practice, we set $r = 0.1$. We justify this parameter choice in the supplement.

Among valid matchings, we search for a one-to-one cloud system matching strategy that prioritizes the pairing of cloud systems with larger areas. We implement this process using a greedy algorithm.

1. **Sorting by area.** First, we sort all cloud systems $X \in H_t$ in descending order based on their areas, ensuring that larger cloud systems are processed first.
2. **Greedy matching.** For each cloud system X in the sorted list, we evaluate all candidate cloud systems $Y \in H_{t+1}$ that satisfy matching conditions (1) or (2) and choose the one with the largest area to be matched to X.
3. **Handling unmatched systems.** For all remaining cloud systems that are unmatched, we mark them as terminated (for $X \in H_t$) or newly formed (for $Y \in H_{t+1}$).

By combining all the selected matchings across adjacent time steps, we generate a set of trajectories for the cloud systems. We do not match clouds across non-adjacent time steps, so a cloud that disappears and later reappears is recorded as separate trajectories.

Merge and split events. Cloud systems often merge and split as they evolve, providing weather scientists with insights into their evolution. Our framework supports computing and visualizing these events. Following the tracking algorithm outlined above, we have computed all valid matchings, with one labeled as the main trajectory and the others identified as secondary trajectories. In the example in Fig. 6, we let the main trajectory of the cloud system X go to Z if the area of Z is larger than Y. However, the trajectory from X to Y can also be considered secondary due to the high matching score between X and Y. Including this secondary trajectory allows us to interpret the scenario as follows: at time step t, cloud system X splits into two systems, Y and Z, at time step $t+1$, with Z continuing along the main trajectory.

6 Experimental Results

In this section, we experiment with two datasets from geostationary satellites. The first **Marine Cloud** dataset focuses on marine stratocumulus clouds over the ocean west of Africa during August 2023, whereas the second **Land Cloud** dataset covers shallow cumulus cloud systems over central Europe from April to September between 2018 and 2019; see supplement for details on these datasets. We review and compare against two state-of-the-art cloud tracking tools (Sec. 6.1), with parameter justifications (Sec. 6.2). We then perform statistical evaluations (Sec. 6.3) and discuss our tracking results in Secs. 6.4 and 6.5. Additionally, we compare our approach with two topology-based general-purpose tracking tools in Sec. 6.6.

6.1 Two Leading Cloud Tracking Tools

We report the results from two state-of-the-art open-source cloud tracking tools for comparative analysis: tobac [13,33] and PyFLEXTRKR [11]. We refer to our tool as the pFGW framework.

We briefly compare the two cloud tracking frameworks with ours. All three tools involve superlevel thresholds for cloud detection. The tool tobac applies the Watershed algorithm [38] with the threshold, whereas PyFLEXTRKR and pFGW use superlevel set components to identify cloud objects. For cloud tracking, tobac tracks the centroids of cloud objects and computes the matching plan with the minimum sum of the Euclidean distances between matched cloud centroids. PyFLEXTRKR computes the region overlap for cloud tracking. In comparison, pFGW combines topological and geometric information of anchor points and summarizes the tracking results for all anchor points within a cloud system. Furthermore, instead of tracking cloud objects, pFGW considers multiple cloud objects as a cloud system and tracks the system as a whole. For simplicity, we use *cloud entity* to refer to either cloud object or cloud system for the rest of the paper. We report other algorithmic details of tobac and PyFLEXTRKR in the supplement.

6.2 Parameter Configurations

Computing the pFGW distance requires two parameters α and m (see Sec. 4.2). We intend to put a higher weight on preserving the geometric location of nodes while still considering the intrinsic merge tree structure, leading to $\alpha \leq 0.5$. We set $\alpha = 0.4$ for the **Marine Cloud** dataset and $\alpha = 0.2$ for the **Land Cloud** dataset. We adopt the strategy of [29] to tune m. Specifically, we impose a threshold on the maximum distance between matched nodes across adjacent time steps and select the largest m such that no matches exceed this threshold. This approach preserves high-probability couplings while avoiding clearly spurious matches. For a fair comparison, we use the same superlevel set threshold for cloud detection for all three methods. See the supplement for other parameters of the three methods and a discussion about parameter choices.

6.3 Evaluation Metrics

Tracking clouds over time using satellite observations is challenging due to their dynamic nature, including their appearances, disappearances, splitting, merging, and transformation. Based on earlier studies in cloud science [12,26], we utilize three evaluation metrics to assess the tracking results.

First, we study the distribution of **timespans** for cloud entities (cloud objects or cloud systems). The timespan is the duration of time (i.e., the number of time steps) a cloud entity travels along its trajectory. This metric indicates how consistently a tracking method monitors cloud entities. However, we do not postulate that every cloud entity should be long-lived.

Second, we investigate the distribution of the standard deviation of a cloud property (e.g., mean COD value of a cloud entity at a given time step) along trajectories, referred to as the **SD of mean COD**. The physical properties of a cloud entity are expected to remain fairly stable over time, with no significant deviations. A mismatch is likely to result in an increase in the standard deviation along the trajectory. We consider only trajectories with a timespan longer than the median timespan and at least three time steps, as the standard deviation is highly sensitive to small sample sizes.

Third, we examine the distribution of the **linearity loss** of trajectories. The linearity loss of a trajectory is the root mean square error (RMSE) of the centroids of cloud entities from the line of best fit. Given the momentum of clouds, it is reasonable to expect short-lived trajectories to be nearly linear without abrupt jumps. However, longer trajectories are more likely to follow the mean flow, which can vary across time and space and lead to curved paths not captured by this metric. For consistency, we evaluate this metric on the same set of trajectories used in the mean mean COD study.

6.4 Case Study: Marine Cloud Dataset

We first examine the **Marine Cloud** dataset, which focuses on marine stratocumulus clouds over the ocean west of Africa. We highlight our topology-driven tracking results in Fig. 1 using two time steps on Aug 1, 2023, at 09:00 and 10:00 UTC, respectively. We then perform a detailed analysis of the cloud tracking results using a subregion from the same dataset also on Aug 1, 2023 in Fig. 7. There are 28 time steps within the day from 09:00 to 15:45 UTC with 15-minute intervals.

Cloud detection and tracking. We first compare the cloud detection results across the three cloud tracking tools: pFGW, PyFLEXTRKR, and tobac. All three methods successfully identify cloud entities from the COD fields, with small discrepancies due to the minor differences between watershed-based (used by tobac) and superlevel-set-based (used by PyFLEXTRKR and to some extent pFGW) strategies.

Meanwhile, we highlight the differences between tracking *cloud objects* versus *cloud systems* in Fig. 7. For PyFLEXTRKR (3rd column), at 09:45 UTC, the central green object (magenta box) splits into two distinct objects (green and gray). At 10:00 UTC, these two

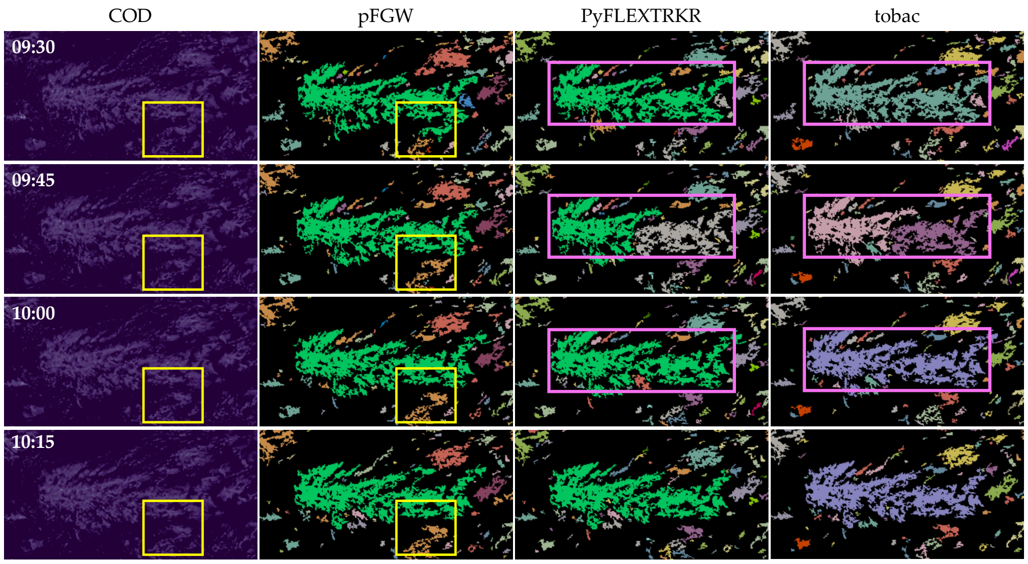

Figure 7: Tracking results of a region in the **Marine Cloud** dataset on Aug 1, 2023, at 9:30, 9:45, 10:00, and 10:15 UTC, respectively. Cloud entities are colored by feature correspondences. From left to right: visualizations of COD fields, tracking results for pFGW, PyFLEXTRKR, and tobac, respectively. For pFGW, yellow blocks from top to bottom showcase a cloud transition process from the green cloud system to the orange one. For PyFLEXTRKR and tobac, magenta boxes highlight suboptimal tracking results due to the transient splitting and merging of cloud objects.

objects merge back together, and the trajectory of the newborn gray object terminates. For tobac (4th column), we observe similar cloud splitting and merging events at 09:45 and 10:00 UTC, respectively; however, tobac considers the objects (magenta boxes) at 9:45 and 10:00 UTC to be new entities, giving rise to three new trajectories. However, the cloud-splitting event at 09:45 UTC is not obvious in the COD field. In contrast, at 09:45 UTC, pFGW does not split the same green cloud system in the center, as our tracking method aggregates nearby cloud objects into a single cloud system.

Previous studies report significant COD uncertainties outside the 5-50 range [43]. With a detection threshold of 2.0, such uncertainties can blur cloud boundaries, leading to transient splits and merges. Tracking cloud systems (instead of cloud objects) with pFGW mitigates this by avoiding numerous short-lived trajectories for these transient events.

As shown in Fig. 7, we gain additional insights by tracking cloud systems instead of cloud objects using pFGW. The yellow boxes in the 1st and 2nd columns highlight a cloud transition process, where a part of the central green system splits and merges into the bottom orange system. In contrast, PyFLEXTRKR and tobac track all cloud objects individually in this region, thus it is harder to infer the change in proximity between clouds.

Statistical evaluation. We statistically evaluate the **Marine Cloud** dataset from Aug 1 to Aug 8, 2023. The observed time period for each day is from 09:00 to 15:45 UTC with a 15-minute interval. We calculate the tracking results for each day separately and then aggregate the statistics from all eight days to evaluate the overall performance for each method. For pFGW, we include statistics involving tracking cloud systems (*pFGW-system*) and tracking cloud objects (*pFGW-object*).

Fig. 8 shows the distributions of trajectory timespan. These distributions are comparable across all three methods. The distributions of pFGW for tracking cloud objects and tracking cloud systems are also similar. Specifically, PyFLEXTRKR generates more short-lived trajectories (primarily those lasting less than 15 minutes); tobac generates fewer longer-lived trajectories compared to the other two methods. In comparison, pFGW preserves long-term trajectories while reducing the number of short-lived ones.

Fig. 9 presents the overall statistics for comparison and displays the *trajectory timespan* distribution in the form of a box plot (1st column). PyFLEXTRKR generates more short-lived trajectories

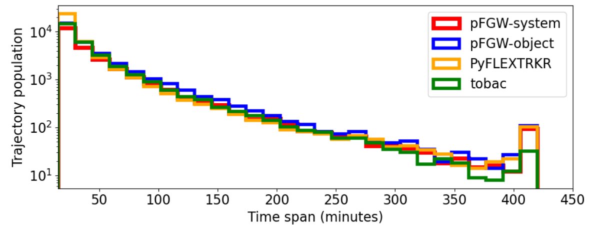

Figure 8: **Marine Cloud** dataset: distribution of trajectory timespans in log-scale for pFGW tracking cloud systems (red) and objects (blue), PyFLEXTRKR (orange), and tobac (green); data aggregated over eight days (Aug 1-8, 2023).

compared to the other two methods, with the median trajectory timespan being just 15 minutes (one timestep). In contrast, pFGW and tobac exhibit a higher median value of 30 minutes (two timesteps), whereas pFGW has a higher interquartile range and mean. It shows that pFGW performs the best at preserving the trajectory duration among the three approaches.

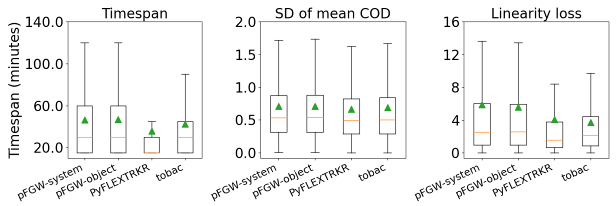

Figure 9: **Marine Cloud** dataset: box plots showing the median (orange line), mean (green triangle), and interquartile range (box boundary) of the distribution for three evaluation metrics.

We compute the standard deviation of mean COD and the linearity loss for trajectories that last for at least 45 minutes (three timesteps); see the 2nd and 3rd columns of Fig. 9. Three methods have similar performance in preserving the mean COD of cloud entities along the trajectory. On the other hand, pFGW has a higher linearity loss compared to the other two methods. This is anticipated as cloud merging and splitting events have been observed to introduce undesirable shifts in the centroid position of the cloud system along the main trajectory. In particular, such position shifts can be drastic for large cloud systems, which are often long-lived for stratocumulus clouds. In contrast, tobac is less effective at identifying cloud merging and splitting events for large cloud entities (see Fig. 7 4th column); PyFLEXTRKR performs worse in maintaining the trajectory timespan. As a result, both tools generate fewer trajectories with high linearity loss.

6.5 Case Study: Land Cloud Dataset

The **Land Cloud** dataset is collected above central Europe, where the complex land-driven convection strongly affects the low-level cloud systems. As the land warms up during the day, shallow cumulus often initiates in the morning, grows mature over time, and reaches its peak in the late afternoon. Hence, during its life cycle, the optical depth of shallow cumulus changes over time. Therefore, we divide the data into three periods for each day: 06:00 to 09:00 UTC for the *morning*, 09:05 to 15:00 for the *midday* period, and 15:05 to 17:55 for the *late afternoon*. In particular, we are interested in the morning and midday periods, as these are typically when the initiation and maturation of shallow cumulus occur, respectively.

For the morning period, we set the superlevel set threshold to 9.0, and for the midday period, we set it to 10.0. This decision is based on the observed quality of cloud segmentation using superlevel set components, as outlined in [12], and the parameter sensitivity analysis described in Sec. 5.1, with further details in the supplement. We do not simplify cloud entities by area because shallow cumulus clouds (particularly during the initiation stage) are smaller and have more gaps between individual cloud objects.

Cloud detection and tracking. We use the data from May 1, 2018, for our case study. We check the transition from 08:30 to 08:35 UTC for data in the morning. For data in the midday, we check the transition from 12:05 to 12:10 UTC. We color all the new cloud entities in magenta at 8:35 UTC and 12:10 UTC, respectively. These new entities may arise from the formation of shallow cumulus, the splitting of a cloud entity, or the loss of cloud tracking.

The morning period reveals a cluster of shallow cumulus clouds developing near the center of the COD field, as illustrated in the first two rows of Fig. 10. These shallow cumulus clouds are identified as a set of small cloud entities in the tracking results. When comparing the performance of pFGW and PyFLEXTRKR, it becomes evident that PyFLEXTRKR loses a significant number of trajectories for these tiny cloud entities; see the cyan box in the 3rd column. This limitation stems from the region-overlap-based approach used by PyFLEXTRKR to track clouds. In small clouds, even slight positional shifts can lead to insufficient overlaps, causing the tracker to lose these clouds. In comparison, pFGW is based on the clouds' geometric location and topological information, making it more robust when tracking small shallow cumulus clouds.

As the day progresses towards midday, the shallow cumulus system and many small cumulus cells evolve, growing thicker and merging into large cloud entities. In the bottom two rows, we observe large cloud entities in the top half of the image (red box), and clusters of small cloud entities in the bottom half (orange box). Both pFGW and PyFLEXTRKR exhibit similar performance in tracking the large cloud entities. However, for the small clouds, PyFLEXTRKR generates numerous new cloud entities (see magenta cloud entities in the orange box), showing its limitations in tracking smaller clouds consistently. In comparison, pFGW exhibits a better capability in consistently tracking small shallow cumulus clouds.

tobac shows better performance in preserving the trajectories for small cloud objects than PyFLEXTRKR (c.f. 3rd and 4th column). However, we observe that the large cloud objects at 8:30 UTC (1st row, pink boxes) are not tracked by tobac during the morning period. These two objects merge into one at 8:35 UTC, which is treated as a new object by tobac, similar to what we have observed in Sec. 6.4. Furthermore, tobac also fails to track the large objects during the midday transition (3rd and 4th row, yellow box).

Statistical evaluation. We perform the statistical evaluation similar to Sec. 6.4. We collect statistics for the datasets for three days on May 1, Jun 23, 2018, and May 12, 2019, respectively in Fig. 12. The 1st row shows the distribution of evaluation metrics for the morning period, and the 2nd row shows the distribution for the midday period.

We start with the trajectory timespan distribution. Among the three methods, PyFLEXTRKR performs the worst on tracking small shallow cumulus, which constitutes the majority of the cloud entity population. Therefore, for both morning and midday periods, most trajectories from the PyFLEXTRKR last for less than five minutes (one time step); see Fig. 11 and Fig. 12 1st column. Meanwhile, tobac does not generate trajectories with a lifetime above 300 minutes in the midday period as the other two methods do; see Fig. 11 2nd row. This reflects our previous observation in Fig. 10 4th column that tobac performs worse than the other two methods in tracking large cloud entities, many of which are persistent in the **Land Cloud** dataset. In contrast, pFGW performs the best on maintaining trajectories for small shallow cumulus in the morning and has a similar trajectory lifetime distribution to tobac in tracking cloud objects during the midday period; see Fig. 12 1st column. By merging nearby cloud objects into cloud systems, pFGW may get fewer trajectories with a long lifetime. However, pFGW still performs better than tobac in maintaining long-term trajectories in the midday period; see Fig. 11 2nd row.

We compute the standard deviation of mean COD and the linearity loss for trajectories that last for at least 15 minutes (three timesteps). When evaluating the ability to preserve mean COD along the same trajectory, all three methods have comparable performances during both morning and midday; see the 2nd column of Fig. 12.

Lastly, because there are fewer large cloud entities in the **Land Cloud** dataset, the linearity loss of pFGW trajectories is similar to that of tobac. On the other hand, it is anticipated that PyFLEXTRKR generates trajectories with the least linearity loss because PyFLEXTRKR often loses track of cloud entities on the **Land Cloud** dataset.

6.6 Comparison with Topology-based Tracking Tools

For completeness, we further compare pFGW against two topology-based general-purpose tracking tools: the Lifted Wasserstein Matcher (LWM) [53] and the Wasserstein distance between merge trees (MTW) [40].

We first compute the critical point trajectories using LWM and MTW, respectively. Next, we compute the cloud system trajectories for LWM and MTW using a postprocessing pipeline similar to that of pFGW. We use the parameter settings that generate the best critical point matching results for statistical evaluation; see the supplement for justifications. We compare the cloud system tracking performance using the statistics described in Sec. 6.3. Additional experimental details are in the supplement.

Statistical evaulation. Fig. 13 shows the distribution of cloud trajectory statistics (in Sec. 6.3) for the three topology-based methods.

We start with the trajectory timespan in Fig. 13 left. Among the three methods, MTW performs the worst in maintaining trajectory continuity. The mean and median trajectory timespan for MTW tracking results are the lowest. In comparison, pFGW and LWM have similar performance, while LWM has a slightly higher mean timespan for trajectories. For the standard deviation of mean COD, all three approaches demonstrate similar distributions for their results. This indicates that all three methods perform similarly in matching cloud systems with similar COD distributions, which are partly reflected by the anchor point COD values. Lastly, pFGW achieves the lowest linearity error on average, with LWM performing slightly worse, while MTW performs substantially worse; see Fig. 13 right (where the boxplot of MTW goes beyond the boundary). This is expected because MTW does not consider geometric locations when matching anchor points.

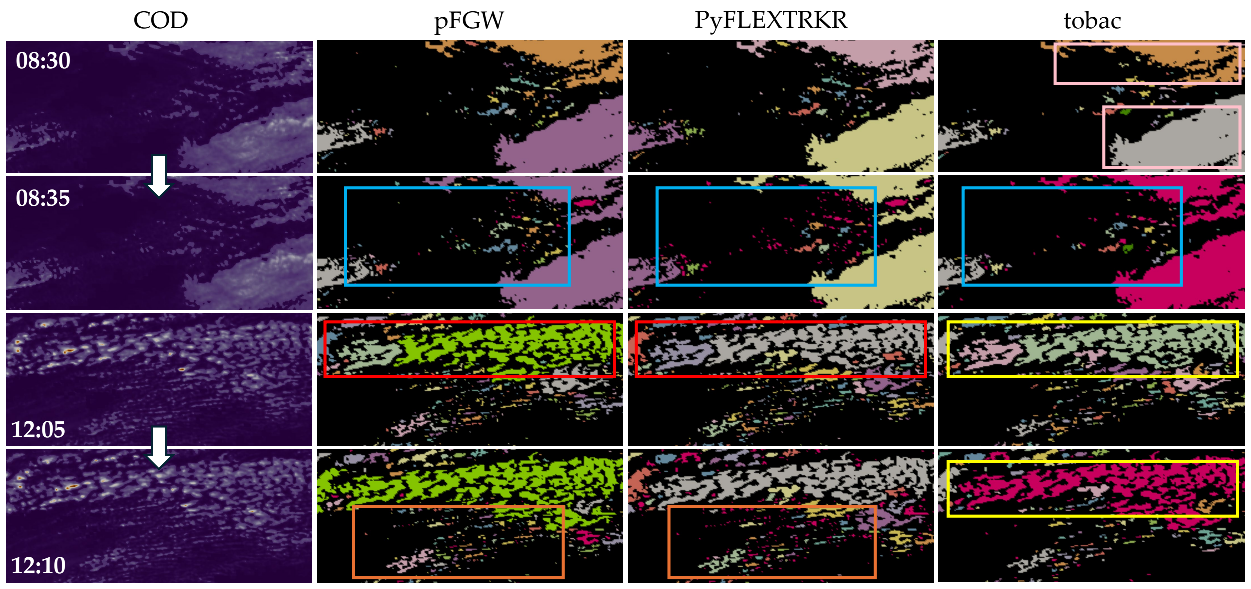

Figure 10: Tracking results of a region in the **Land Cloud** dataset on May 1, 2018. From left to right: visualizations of COD fields, tracking results for pFGW, PyFLEXTRKR, and tobac, respectively. The top two rows (resp. bottom two rows) are for the transition during the morning (resp. midday) period. All new cloud entities in the 2nd and 4th rows are colored magenta; others are colored by correspondences. Cyan and orange boxes (3rd column) highlight the areas where PyFLEXTRKR fails to track many small shallow cumulus clouds (shown as new entities in magenta). Yellow and pink boxes (4th column) emphasize the suboptimal tracking results from tobac when two large cloud objects merge.

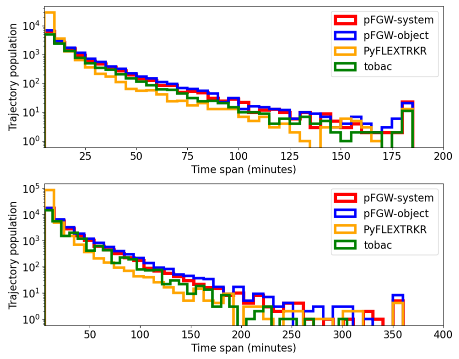

Figure 11: **Land Cloud** dataset: distributions of trajectory timespans in log-scale for pFGW tracking cloud systems (red) and objects (blue), PyFLEXTRKR (orange), and tobac (green); data aggregated over three days (May 1 and Jun 23 in 2018, and May 12 in 2019). The top row is for the morning period, and the bottom is for the midday.

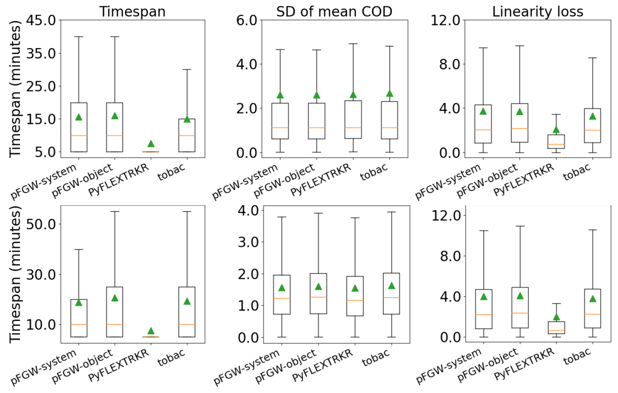

Figure 12: **Land Cloud** dataset: statistical evaluation for the morning data (1st row, 06:00–09:00 UTC) and midday data (2nd row, 09:05–15:00 UTC). Box plots show the median (orange line), mean (green triangle), and interquartile range (box boundary) for three evaluation metrics.

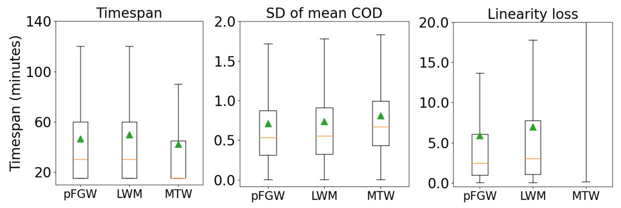

Figure 13: **Marine Cloud** dataset: box plots showing the median (orange line), mean (green triangle), and interquartile range (box boundary) of the distribution for three topology-based tracking methods. The box for the linearity loss for MTW exceeds the plot's upper bound.

7 Conclusion and Discussion

The case studies and statistical evaluations provide several important takeaways. First, our framework operates with cloud systems instead of cloud objects, reducing sensitivity to threshold selection and producing fewer short-lived cloud trajectories. Tracking low-level clouds as systems offers deeper insights into their proximity and evolution. Second, our framework demonstrates strong performance in tracking clouds compared to two state-of-the-art cloud tracking methods. Notably, it is the most consistent in tracking small shallow cumulus clouds over land as well as large stratocumulus over the ocean. In future work, we aim to enhance tracking quality by integrating additional cloud variables, such as cloud fraction, cloud liquid water path, and cloud top height [44].

Acknowledgments

ML and BW were partially supported by U.S. Department of Energy (DOE) DE-SC0021015 and U.S. National Science Foundation (NSF) IIS 2145499. DC was partly supported by the Helmholtz *Changing Earth* program and WarmWorld project (grant 01LK2202B) of the German Ministry of Education and Research (BMBF). FS was partly supported by the EU CleanCloud project (grant 101137639). We would like to thank Andi Walther for sharing the datasets, and the Dagstuhl Seminar 23342 (Computational Geometry of Earth System Analysis) for initializing the collaboration.

References

[1] S. Bley, H. Deneke, and F. Senf. Meteosat-based characterization of the spatiotemporal evolution of warm convective cloud fields over central europe. *Journal of Applied Meteorology and Climatology*, 55(10):2181 – 2195, 2016. doi: 10.1175/JAMC-D-15-0335.1 1

[2] L. M. V. Carvalho and C. Jones. A satellite method to identify structural properties of mesoscale convective systems based on the maximum spatial correlation tracking technique (MASCOTTE), 2001. doi: 10.1175/1520-0450(2001)040<1683:ASMTIS>2.0.CO;2 2

[3] L. Chapel, M. Z. Alaya, and G. Gasso. Partial optimal tranport with applications on positive-unlabeled learning. *Advances in Neural Information Processing Systems*, 33:2903–2913, 2020. 3

[4] D. Chatterjee, S. Schnitt, P. Bigalke, C. Acquistapace, and S. Crewell. Capturing the diversity of mesoscale trade wind cumuli using complementary approaches from self-supervised deep learning. *Geophysical Research Letters*, 51(12):e2024GL108889, 2024. doi: 10.1029/2024GL108889 2

[5] Q. Coopman, C. Hoose, and M. Stengel. Detection of mixed-phase convective clouds by a binary phase information from the passive geostationary instrument SEVIRI. *Journal of Geophysical Research: Atmospheres*, 124(9):5045–5057, 2019. doi: 10.1029/2018JD029772 2

[6] J. T. Dawe and P. H. Austin. Statistical analysis of an LES shallow cumulus cloud ensemble using a cloud tracking algorithm. *Atmospheric Chemistry and Physics*, 12(2):1101–1119, 2012. doi: 10.5194/acp-12-1101-2012 1

[7] H. Doraiswamy, V. Natarajan, and R. S. Nanjundiah. An exploration framework to identify and track movement of cloud systems. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2896–2905, 2013. doi: 10.1109/TVCG.2013.131 2

[8] R. M. Endlich and D. E. Wolf. Automatic cloud tracking applied to GOES and METEOSAT observations. *Journal of Applied Meteorology and Climatology*, 20(3):309 – 319, 1981. doi: 10.1175/1520-0450(1981)020<0309:ACTATG>2.0.CO;2 2

[9] E. Eytan, I. Koren, O. Altaratz, A. Kostinski, and A. Ronen. Longwave radiative effect of the cloud twilight zone. *Nature Geoscience*, 13:669–673, 10 2020. doi: 10.1038/s41561-020-0636-8 5

[10] F. Jansson et al. Cloud botany: Shallow cumulus clouds in an ensemble of idealized large-domain large-eddy simulations of the trades. *Journal of Advances in Modeling Earth Systems*, 15(11):e2023MS003796, 2023. doi: 10.1029/2023MS003796 4

[11] Z. Feng et al. PyFLEXTRKR: a flexible feature tracking python software for convective cloud analysis. *Geoscientific Model Development*, 16(10):2753–2776, 2023. doi: 10.5194/gmd-16-2753-2023 6

[12] T. Fiolleau and R. Roca. An algorithm for the detection and tracking of tropical mesoscale convective systems using infrared images from geostationary satellite. *IEEE Transactions on Geoscience and Remote Sensing*, 51(7):4302–4315, 2013. doi: 10.1109/TGRS.2012.2227762 2, 6, 8

[13] G. A. Sokolowsky et al. *tobac* v1.5: introducing fast 3d tracking, splits and mergers, and other enhancements for identifying and analysing meteorological phenomena. *Geoscientific Model Development*, 17(13):5309–5330, 2024. doi: 10.5194/gmd-17-5309-2024 6

[14] G. Köcher et al. Evaluation of convective cloud microphysics in numerical weather prediction models with dual-wavelength polarimetric radar observations: methods and examples. *Atmospheric Measurement Techniques*, 15(4):1033–1054, 2022. doi: 10.5194/amt-15-1033-2022 1

[15] J. J. Gristey, G. Feingold, I. B. Glenn, K. S. Schmidt, and H. Chen. On the relationship between shallow cumulus cloud field properties and surface solar irradiance. *Geophysical Research Letters*, 47(22):e2020GL090152, 2020. doi: 10.1029/2020GL090152 2

[16] Q. Han, W. B. Rossow, and A. A. Lacis. Near-global survey of effective droplet radii in liquid water clouds using ISCCP data. *Journal of Climate*, 7:465–497, 1994. doi: 10.1175/1520-0442(1994)007<0465:NGSOED>2.0.CO;2 2

[17] C. Henken, M. Schmeits, H. Deneke, and R. Roebeling. Using MSG-SEVIRI cloud physical properties and weather radar observations for the detection of Cb/TCu clouds. *Journal of Applied Meteorology and Climatology*, 50, 07 2011. doi: 10.1175/2011JAMC2601.1 2

[18] T. Heus and A. Seifert. Automated tracking of shallow cumulus clouds in large domain, long duration large eddy simulations. *Geoscientific Model Development*, 6(4):1261–1273, 2013. doi: 10.5194/gmd-6-1261-2013 2

[19] K. Holmlund et al. Meteosat third generation (MTG): Continuation and innovation of observations from geostationary orbit. *Bulletin of the American Meteorological Society*, 102(5):E990 – E1015, 2021. doi: 10.1175/BAMS-D-19-0304.1 2

[20] J. Lukasczyk et al. Dynamic nested tracking graphs. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):249–258, 2020. doi: 10.1109/TVCG.2019.2934368 2

[21] M. Janssens et al. Cloud patterns in the trades have four interpretable dimensions. *Geophysical Research Letters*, 48(5):e2020GL091001, 2021. doi: 10.1029/2020GL091001 2

[22] K. Bessho et al. An introduction to Himawari-8/9—Japan's new-generation geostationary meteorological satellites. *Journal of the Meteorological Society of Japan*, 94:151–183, 2016. doi: 10.2151/jmsj.2016-009 2

[23] A. Karagiannidis, K. Lagouvardos, V. Kotroni, and N. Mazarakis. Investigation of isolated thunderstorms lightning activity over eastern Mediterranean using meteosat rapid scan infrared imagery. *International Journal of Remote Sensing*, 37(20):5001–5020, 2016. doi: 10.1080/01431161.2016.1226000 2

[24] Koren, Ilan and Feingold, Graham. Adaptive behavior of marine cellular clouds. *Scientific Reports*, 2013. 1

[25] R. Kowch, C. R. Trepte, J. S. Reid, and R. Holz. Automated Tracking of Shallow Maritime Clouds on Geostationary Imagery to Extract Lifecycle Characteristics. In *AGU Fall Meeting Abstracts*, vol. 2022, pp. A12K–1244, Dec. 2022. 2

[26] V. Lakshmanan and T. Smith. An objective method of evaluating and devising storm-tracking algorithms. *Weather and Forecasting - WEATHER FORECAST*, 25:701–709, 04 2010. doi: 10.1175/2009WAF2222330.1 2, 6

[27] M. Li, H. Carr, O. Rübel, B. Wang, and G. H. Weber. Distributed augmentation, hypersweeps, and branch decomposition of contour trees for scientific exploration. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):152–162, 2025. doi: 10.1109/TVCG.2024.3456322 3

[28] M. Li, S. Palande, L. Yan, and B. Wang. Sketching merge trees for scientific visualization. In *2023 Topological Data Analysis and Visualization (TopoInVis)*, pp. 61–71, 2023. doi: 10.1109/TopoInVis60193.2023.00013 5

[29] M. Li, X. Yan, L. Yan, T. Needham, and B. Wang. Flexible and probabilistic topology tracking with partial optimal transport. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–18, 2025. doi: 10.1109/TVCG.2025.3561300 2, 3, 4, 5, 6

[30] A. H. Lohmann, Monahan and D. Heinemann. Local short-term variability in solar irradiance. *Atmospheric Chemistry and Physics*, 16(10):6365–6379, 2016. doi: 10.5194/acp-16-6365-2016 2

[31] U. Lohmann, F. Lüönd, and F. Mahrt. *An Introduction to Clouds From the Microscale to Climate*. Cambridge University Press, 2016. doi: 10.1017/CBO9781139087513 2

[32] J. Lukasczyk, G. Weber, R. Maciejewski, C. Garth, and H. Leitte. Nested tracking graphs. *Computer Graphics Forum*, 36(3):12–22, 2017. doi: 10.1111/cgf.13164 2

[33] M. Heikenfeld et al. tobac 1.2: towards a flexible framework for tracking and analysis of clouds in diverse datasets. *Geoscientific Model Development*, 12(11):4551–4570, 2019. doi: 10.5194/gmd-12-4551-2019 2, 4, 6

[34] M. Soler et al. Ranking viscous finger simulations to an acquired ground truth with topology-aware matchings. In *IEEE 9th Symposium on Large Data Analysis and Visualization*, pp. 62–72, 2019. doi: 10.1109/LDAV48142.2019.8944365 2

[35] V. Mathon and H. Laurent. Life cycle of Sahelian mesoscale convective cloud systems, 2001. doi: 10.1002/qj.49712757208 2

[36] N. Bellouin et al. Bounding global aerosol radiative forcing of climate change. *Reviews of Geophysics*, 58(1):e2019RG000660, 2020. doi: 10.1029/2019RG000660 1

[37] N. Benas et al. The MSG-SEVIRI-based cloud property data record CLAAS-2. *Earth System Science Data*, 9(2):415–434, 2017. doi: 10.5194/essd-9-415-2017 2

[38] L. Najman and M. Schmitt. Watershed of a continuous function. *Signal Processing*, 38(1):99–112, 1994. Mathematical Morphology and its Applications to Signal Processing. doi: 10.1016/0165-1684(94)90059-0 6

[39] T. Nakajima and M. D. King. Determination of the optical thickness and effective particle radius of clouds from reflected solar radiation measurements. part i: Theory. *Journal of the Atmospheric Sciences*, 47(15):1878–1893, Aug. 1990. doi: 10.1175/1520-0469(1990)047<1878:dotota>2.0.co;2 2

[40] M. Pont, J. Vidal, J. Delon, and J. Tierny. Wasserstein distances, geodesics and barycenters of merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 2021. doi: 10.1109/TVCG.2021.3114839 2, 8

[41] D. Rosenfeld. Objective method for analysis and tracking of convective cells as seen by radar. *Journal of Atmospheric and Oceanic Technology*, 4(3):422 – 434, 1987. doi: 10.1175/1520-0426(1987)004<0422:OMFAAT>2.0.CO;2 2

[42] S. C. Sherwood et al. An assessment of earth's climate sensitivity using multiple lines of evidence. *Reviews of Geophysics*, 58(4):e2019RG000678, 2020. doi: 10.1029/2019RG000678 1

[43] S. Platnick et al. The MODIS cloud optical and microphysical products: Collection 6 updates and examples from terra and aqua. *IEEE Transactions on Geoscience and Remote Sensing*, 55(1):502–525, 2017. doi: 10.1109/TGRS.2016.2610522 4, 7

[44] S. W. Freeman et al. Advancing our understanding of cloud processes and their role in the earth system through cloud object tracking. *Bulletin of the American Meteorological Society*, 105(1):E297 – E299, 2024. doi: 10.1175/BAMS-D-23-0204.1 1, 9

[45] H. Saikia and T. Weinkauf. Fast topology-based feature tracking using a directed acyclic graph. In *Topological Methods in Data Analysis and Visualization*, pp. 155–169. Springer, 2017. doi: 10.1007/978-3-030-43036-8_10 2

[46] H. Saikia and T. Weinkauf. Global feature tracking and similarity estimation in time-dependent scalar fields. In *Computer Graphics Forum*, vol. 36, pp. 1–11, 2017. doi: 10.1111/cgf.13163 2

[47] M. Schröder, M. König, and J. Schmetz. Deep convection observed by the spinning enhanced visible and infrared imager on board meteosat 8: Spatial distribution and temporal evolution over Africa in summer and winter 2006. *Journal of Geophysical Research: Atmospheres*, 114(D5), 2009. doi: 10.1029/2008JD010653 2

[48] R. Scofield, R. Kuligowski, and C. Davenport. The use of the hydro-nowcaster for mesoscale convective systems and the tropical rainfall nowcaster (TRaN) for landfalling tropical systems. In *Preprints, Symp. on Planning, Nowcasting, and Forecasting in the Urban Zone*, p. 1.4. Amer. Meteor. Soc., Seattle, WA, 2004. 2

[49] T. Seelig, H. Deneke, J. Quaas, and M. Tesche. Life cycle of shallow marine cumulus clouds from geostationary satellite observations. *Journal of Geophysical Research: Atmospheres*, 126(22):e2021JD035577, 2021. doi: 10.1029/2021JD035577 1, 2

[50] F. Senf and H. Deneke. Satellite-based characterization of convective growth and glaciation and its relationship to precipitation formation over Central Europe. *Journal of Applied Meteorology and Climatology*, 56(7):1827 – 1845, 2017. doi: 10.1175/JAMC-D-16-0293.1 2

[51] F. Senf, F. Dietzsch, A. Hünerbein, and H. Deneke. Characterization of initiation and growth of selected severe convective storms over central Europe with MSG-SEVIRI. *Journal of Applied Meteorology and Climatology*, 54(1):207 – 224, 2015. doi: 10.1175/JAMC-D-14-0144.1 1, 2

[52] F. Senf, D. Klocke, and M. Brueck. Size-resolved evaluation of simulated deep tropical convection. *Monthly Weather Review*, 146(7):2161–2182, 2018. doi: 10.1175/MWR-D-17-0378.1 2

[53] M. Soler, M. Plainchault, B. Conche, and J. Tierny. Lifted Wasserstein matcher for fast and robust topology tracking. In *IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 23–33. Berlin, Germany, 2018. doi: 10.1109/LDAV.2018.8739196 2, 8

[54] R. Sridharamurthy, T. B. Masood, A. Kamakshidasan, and V. Natarajan. Edit distance between merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 26(3):1518–1531, 2020. doi: 10.1109/TVCG.2018.2873612 2

[55] B. Stevens, S. Bony, H. Brogniez, L. Hentgen, C. Hohenegger, C. Kiemle, T. S. L'Ecuyer, A. K. Naumann, H. Schulz, P. A. Siebesma, and et al. Sugar, gravel, fish, and flowers: Mesoscale cloud patterns in the tradewinds. *Q. J. R. Meteorol. Soc.*, Sep 2019. doi: 10.1002/qj.3662 2

[56] T. Heus et al. A statistical approach to the life cycle analysis of cumulus clouds selected in a virtual reality environment. *Journal of Geophysical Research: Atmospheres*, 114(D6), 2009. doi: 10.1029/2008JD010917 2

[57] T. J. Schmit et al. Introducing the next-generation advanced baseline imager on GOES-R. *Bulletin of the American Meteorological Society*, 86(8):1079 – 1096, 2005. doi: 10.1175/BAMS-86-8-1079 2

[58] K. E. Trenberth, J. T. Fasullo, and J. Kiehl. Earth's global energy budget. *Bulletin of the American Meteorological Society*, 90(3):311 – 324, 2009. doi: 10.1175/2008BAMS2634.1 1

[59] D. A. Vila, L. A. T. Machado, H. Laurent, and I. Velasco. Forecast and tracking the evolution of cloud clusters (ForTraCC) using satellite infrared imagery: Methodology and validation. *Weather and Forecasting*, 23(2):233 – 245, 2008. doi: 10.1175/2007WAF2006121.1 2

[60] C. Villani. *Topics in Optimal Transportation*, vol. 58. American Mathematical Society, 2003. 3

[61] M. Williams and R. A. Houze. Satellite-observed characteristics of winter monsoon cloud clusters, 1987. doi: 10.1175/1520-0493(1987)115<0505:SOCOWM>2.0.CO;2 2

[62] World Meteorological Organization. Manual on the observation of clouds and other meteors – international cloud atlas, 2017. 1

[63] L. Yan, T. B. Masood, F. Rasheed, I. Hotz, and B. Wang. Geometry-aware merge tree comparisons for time-varying data with interleaving distances. *IEEE Transactions on Visualization and Computer Graphics*, 29(8):3489–3506, 2023. doi: 10.1109/TVCG.2022.3163349 2

[64] L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *Computer Graphics Forum*, 40(3):599–633, 2021. doi: 10.1111/cgf.14331 2

[65] M. Zhao and P. H. Austin. Life cycle of numerically simulated shallow cumulus clouds. part i: Transport. *Journal of the Atmospheric Sciences*, 62:1269–1290, 2005. doi: 10.1175/JAS3414.1 2

Supplement

In this supplement, we describe the cloud data in Appendix A, followed by implementation details in Appendix B. We provide descriptions for the two other leading cloud tracking tools in Appendix C. Next, we discuss parameter configurations and limitations of our comparative analysis in Appendix D. We provide details on qualitatively and quantitatively comparing three topology-based tracking tools in Appendix E. Finally, we provide a runtime analysis in Appendix F.

A Cloud Data

Geostationary satellites offer continuous measurements of cloud systems as they evolve over time, a capability utilized in cloud remote sensing since the launch of the first Applications Technology Satellite (ATS-1) in 1966 [11]. This study employs two distinct resolutions of cloud optical depth (COD) retrievals [2,16,20] derived from the visible and near-infrared channels of the SEVIRI instrument onboard Meteosat's second-generation satellites.

The first dataset, referred to as **Marine Cloud**, utilizes the CLAAS-3.0 product [4], focusing on marine stratocumulus clouds over the ocean west of Africa (26.74°S to 4.52°S, 10.52°E to 27.99°W) during August 2023. This dataset retains SEVIRI's native resolution, with a 15-minute temporal repeat cycle and a spatial resolution of 3 km at nadir. Stratocumulus cloud systems are prevalent in this region during the austral winter months (July–September) [5,6]. Fig. 1 shows an example of the COD field from the **Marine Cloud** dataset.

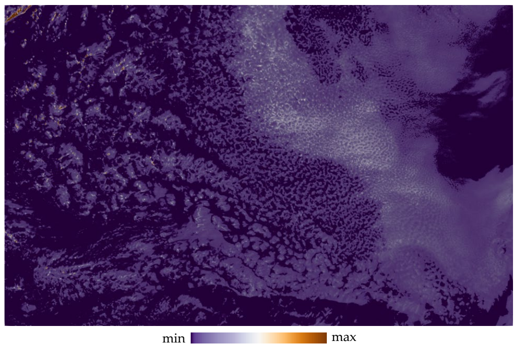

Figure 1: A snapshot of the COD field at 09:00 UTC on Aug 1, 2023, from a marine stratocumulus cloud dataset over the ocean west of Africa.

The second dataset, labeled as **Land Cloud**, covers shallow cumulus cloud systems over central Europe (47.3°N to 55.4°N, 1.9°E to 9.3°E) across 13 selected days of low-level cloud occurrences from April to September between 2018 and 2019. This dataset simulates the capabilities of the Meteosat Third Generation (MTG) mission, offering an enhanced spatial resolution of 2 km $\times$ 1 km and a 5-minute temporal repeat cycle. **Land Cloud** significantly improves the standard MSG cloud dataset, enabling more detailed tracking of convective systems in central Europe [2,20].

In the visible range, satellites receive part of the solar radiation reflected by the clouds or the earth's surface. The near-infrared channels going beyond the visible range help examine how objects reflect, transmit, and absorb the sun's infrared emission. The retrieval algorithm operates on the principle that cloud reflectance is predominantly governed by COD with minimal sensitivity to particle size at visible wavelengths. In contrast, at near-infrared wavelengths, cloud reflectance is primarily influenced by particle size [3,9,13].

Modern retrieval methods, such as the Cloud Physical Properties (CPP) algorithm used with SEVIRI by the Royal Meteorological Institute of the Netherlands [16], rely on a combination of non-absorbing visible wavelengths (0.6 or 0.8 µm) and near-infrared wavelengths (1.6 or 3.8 µm). While the 1.6 µm wavelength is well-suited for thicker clouds, the 3.8 µm wavelength is better for thinner clouds. However, retrievals at 3.8 µm involve greater uncertainty due to their proximity to thermally emitted radiance and the lower solar irradiance at this wavelength compared to 1.6 µm. Consequently, the CPP algorithm primarily utilizes 0.6 and 1.6 µm reflectances for retrieving COD, particle size, and cloud liquid water path (CLWP).

We chose COD as the input for our cloud system tracking framework due to the following reasons: (i) COD eliminates the need to account for solar zenith angle and surface characteristics, both of which significantly impact reflectance values, and (ii) COD is the most accurate retrieval product available at the enhanced resolution [2]. However, one must note that the COD uncertainties become large outside a value range of 5 - 50 but remain below 8-10% within this range [17].

B Implementation

All experiments are done on a laptop with a 12th Gen Intel(R) Core(TM) i9-12900H 2.50 GHz CPU with 32 GB memory. We use the python library *scipy* [14] to identify superlevel set components for cloud object detection. We use the *ParaView 5.11.1* [1] and *TTK 1.1.0* [18] to compute merge trees and topological zones. We follow the work of Li et al. [8] to compute the pFGW distance for merge tree matching, which has open-source code on GitHub [7]. We will provide our implementation on GitHub upon publication.

C Description for Leading Cloud Tracking Tools

In this section, we describe the two leading cloud tracking tools reported in the experiments: tobac and PyFLEXTRKR.

The tool tobac takes a sequence of thresholds to identify connected components of the superlevel set of the COD field. Specifically, for a fixed threshold, it calculates the bounding box of each superlevel set component and selects a feature point from the bounding box using one of four strategies: the center, the maxima, or the barycenter weighted by either the absolute COD value or its difference from the threshold. Then, tobac uses the watershed algorithm [12] to detect cloud objects. The watershed algorithm first identifies the local peak area for each feature point. A cloud object is then created at the local maxima of the identified peak area and expanded by iteratively adding the surrounding pixels with the highest COD values. The expansion terminates when the cloud object touches another one or reaches the boundary of the superlevel set component. Subsequently, the trajectory of the cloud object is defined by the trajectory of the feature point. To match a feature point to one in the next time step, tobac searches for possible candidates within a user-defined neighborhood in the domain. The matching strategy that minimizes the sum of the squared Euclidean distance between the feature point and its matched point produces the tracking result.

PyFLEXTRKR identifies cloud objects using either superlevel set components or the watershed algorithm. Then, it computes the region overlap between cloud objects at adjacent time steps and determines the trajectory of the cloud object based on the overlapped region size.

For cloud detection, tobac employs the watershed algorithm to expand each cloud from its respective feature point. This expansion is essential as tobac relies on feature points to facilitate tracking in subsequent stages. In particular, if one can guarantee that each superlevel set component has exactly one feature point inside, the cloud object detection result is identical to the superlevel set component. However, since tobac selects the feature point from the component's bounding box, none of the four feature detection strategies can guarantee this outcome. In contrast, pFGW identifies cloud objects using superlevel set components, whereas PyFLEXTRKR offers flexibility by supporting both superlevel sets and the watershed algorithm for cloud detection.

During cloud tracking, pFGW combines topological and geometric information of anchor points and summarizes the tracking results for all anchor points within a cloud system. In contrast, both tobac and PyFLEXTRKR rely exclusively on geometric information. tobac tracks clouds using a single feature point for each cloud object; it may produce an unstable trajectory due to the instability in a feature point's location across time steps. On the other hand, PyFLEXTRKR tracks features based on region overlap; however, it could be challenging to handle small or fast-moving features with insufficient overlaps between adjacent time steps [8].

D Experimental Parameters

We discuss experimental settings and parameters in addition to those mentioned in Sec. 5 and Sec. 6.

D.1 Cloud Object Detection and Simplification

Object detection parameter sensitivity analysis. Currently, there is no consensus on the threshold value a to detect low-level clouds. Following established practices [10], we test a range of thresholds from 0.5 to 5.0 at a gap of 0.5 to analyze the impact of the threshold on the cloud area size and numbers. This approach is similar to using brightness temperature thresholds for tracking deep convective cloud systems [10] and references therein. We use the statistics computed on the **Marine Cloud** dataset as an example. Fig. 2(a) shows the cumulative distribution of cloud object number density as cloud size increases. The distribution curves appear consistent across different superlevel set thresholds, indicating that the cloud area size distribution is not highly dependent on the superlevel set threshold within the range of $[0.5, 5.0]$. In particular, more than 70% of cloud objects are below 10 pixels regardless of the threshold; see the zoom-in view at Fig. 2(c). In Fig. 2(b), the differences among curves are more noticeable. For example, when the threshold is 0.5, cloud objects with more than 1000 pixels contribute to 92.15% of the total cloud coverage. In comparison, 80.36% of the cloud area coverage comes from clouds with more than 1000 pixels when the threshold is 5.0. This result indicates that when choosing a threshold that is too small, large clouds will dominate the cloud area coverage, and we may mistakenly interpret multiple cloud systems as a single one altogether. On the other hand, if we use a very high threshold, we may obtain too many small cloud objects over-segmented from a large one. Therefore, we choose a threshold of 2.0 to avoid potential issues with extreme values.

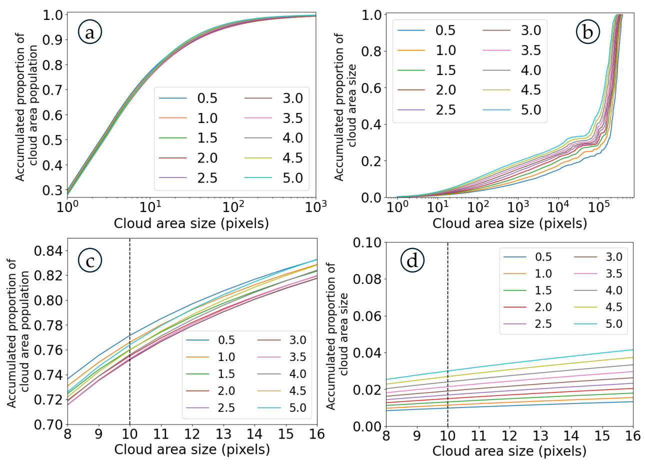

Figure 2: (a) the accumulative proportion of cloud object number density. (b) the accumulative proportion of cloud area size. Each curve represents a superlevel set threshold value between 0.5 and 5.0. (c) and (d) are the zoom-in views for (a) and (b), respectively. Notice that the x-axes in (a) and (b) are based on log-scale, whereas those in (c) and (d) are not. Statistics are computed on the **Marine Cloud** dataset from Aug 1 to Aug 14, 2023.

The **Land Cloud** dataset exhibits more complex characteristics and patterns of low-level clouds. Specifically, we can observe the initiation and maturation process for shallow cumulus clouds at different times of the day due to the overland convection. The variation of COD for low-level clouds has to be considered when choosing the superlevel set thresholds. We perform parameter sensitivity analysis for two time periods: 06:00 to 09:00 UTC for the morning and 09:05 to 15:00 UTC for the midday.

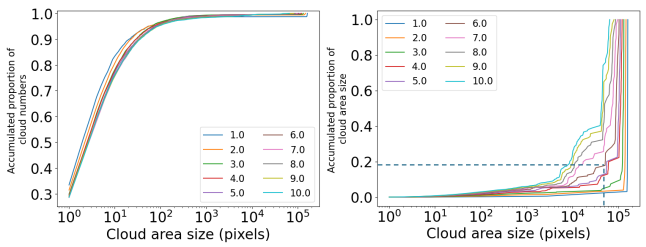

Figure 3: (a) the accumulative proportion of cloud object number density. (b) the accumulative proportion of cloud area size. Each curve represents a superlevel set threshold value between 1.0 and 10.0. Statistics are computed on the **Land Cloud** dataset, including the snapshots collected between 06:00 and 09:00 UTC for all 13 selected days between 2018 and 2019. Notice that the x-axes are based on log-scale.

We test the threshold between 1.0 and 10.0 to perform a parameter sensitivity analysis similar to the **Marine Cloud** dataset for the morning data. Fig. 3 shows the accumulative proportion of cloud object number density and cloud area coverage. The dotted lines in Fig. 3(b) highlight a point on the curve for the threshold at 6.0, indicating that less than 80% of the low cloudiness is due to cloud clusters smaller than $50,000$ pixels. When the threshold is below 6.0, this percentage becomes lower. This observation indicates that it may be difficult to identify small shallow cumulus clouds from the large cloud cluster when the threshold is low. We further examine the cloud detection results using different superlevel set thresholds. In these results, we select the lowest threshold that reasonably separates cloud objects. Fig. 4 shows an example of cloud detection results with the threshold between 6.0 and 10.0. The COD field in the blue box has lower COD values compared to the high-value area in the yellow box. The area in the blue and yellow boxes is identified as a single cloud object until the threshold reaches 9.0. Therefore, we choose 9.0 as the threshold for this snapshot.

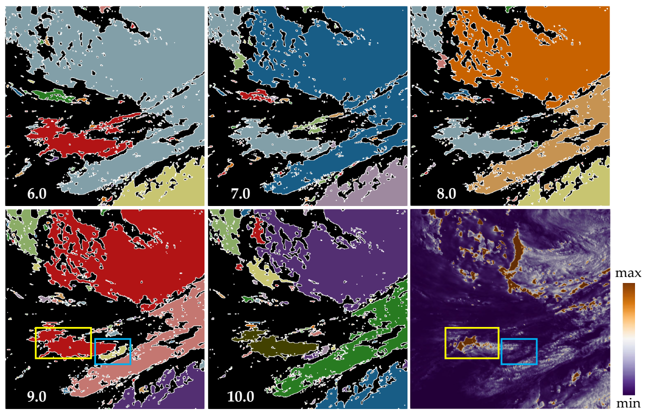

Figure 4: Individual cloud objects detected using superlevel set thresholds between 6.0 and 10.0 for the COD field data (2nd row, 3rd column) collected on May 1, 2018, at 07:00 UTC. The white numbers represent the superlevel set thresholds. Regions within the blue boxes and yellow boxes are identified as a single cloud object until the superlevel set threshold reaches 9.0.

Fig. 5 shows the cumulative distribution of cloud number density and cloud area coverage for the midday data in **Land Cloud**. We change the range of thresholds for testing to $[3.0, 12.0]$ because we expect to see shallow cumulus clouds in higher COD values during midday. Specifically, the cloud area size distribution (see Fig. 5 right) is sensitive to the superlevel set threshold below 6.0. We examine the snapshots using the threshold from 6.0 to 12.0. Fig. 6 shows an example COD field during the midday period, with a subset of cloud detection results using the threshold between 8.0 and 12.0. The COD field within the blue box in Fig. 6 has higher values on the bottom left and lower values on the top right. 10.0 is the lowest threshold value that separates these two regions as individual objects. Therefore, we use 10.0 as the threshold for this snapshot.

We examine snapshots at 07:00, 08:50, 11:00, and 13:00 UTC for each day in the **Land Cloud** dataset and determine the best threshold for each snapshot. We summarize the decisions for snapshots at 07:00 and 08:50 UTC for the morning data and those for snapshots at 11:00 and 13:00 UTC for the midday data. The final threshold decision is made by choosing the most popular threshold for the time period.

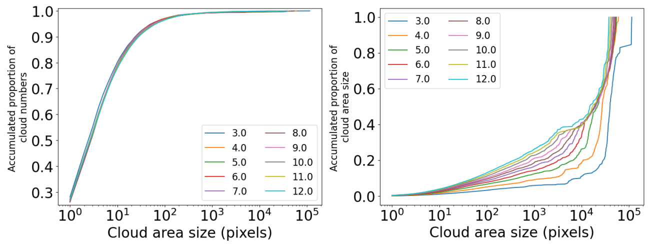

Figure 5: (a) the accumulative proportion of cloud object number density. (b) the accumulative proportion of cloud area size. Each curve represents a superlevel set threshold value between 1.0 and 10.0. Statistics are computed on the **Land Cloud** dataset, including the snapshots collected between 09:05 and 15:00 for all 13 selected days between 2018 and 2019.

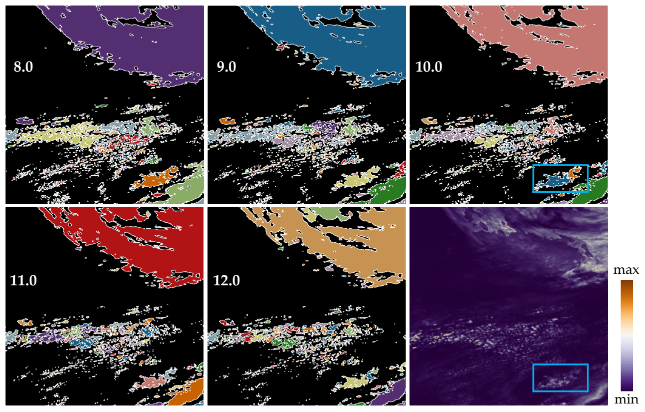

Figure 6: Individual cloud objects using superlevel set thresholds between 8.0 and 12.0 for the bottom right COD field data (2nd row, 3rd column) collected on May 1, 2018, at 11:00 UTC. The white numbers represent the superlevel set thresholds.

Object simplification parameter tuning. We refer to the statistics of cloud area size in Fig. 2 to determine the area-based simplification level for the dataset of stratocumulus: by ignoring cloud objects smaller than 10 pixels, we can get rid of more than 70% of cloud objects from the field (see Fig. 2(c)), and still cover more than 97% of the total cloud area (see Fig. 2(d)), regardless of the superlevel set threshold. Therefore, we choose the area-based simplification level to be 10 pixels.

D.2 Cloud Tracking

Parameter tuning for pFGW distance. Following [8], to compute the pFGW distance, we need to tune the two parameters: α and m. Recall that α is the parameter balancing the weight between the intrinsic information of merge trees and the extrinsic information of nodes. $\alpha \to 0$ leads to a high weight on keeping the geometric locations of nodes in tracking, whereas $\alpha \to 1$ leads to a high weight on keeping the merge tree structure. $\alpha = 0.5$ puts the same weight on both parts. We attempt $\alpha \in \{0.2, 0.4\}$, putting a higher weight on preserving the geometric location of nodes while still considering the intrinsic merge tree structure. We choose the better results between the two choices of α for demonstration.

The parameter m determines the amount of probability mass to be preserved in the partial optimal transport, allowing feature appearance and disappearance. We follow a similar strategy to [8] to adjust the parameter m. In particular, we determine the threshold for the maximum possible distance between matched nodes from adjacent time steps (we discuss this option in the next paragraph). Then, we search for the highest m value such that no matching between nodes farther than the above threshold exists. We want to keep the highest probability in the coupling without introducing obviously incorrect matching.

We noticed that the better choice of α is 0.4 for **Marine Cloud**. This is because for stratocumulus, there are often a few anchor points inside the cloud objects due to their relatively large area. Using a higher α value to emphasize the locality of anchor points within each cloud object is beneficial. On the other hand, the better α for **Land Cloud** is 0.2. This is because many tiny cloud objects in **Land Cloud** only have one anchor point, which means that there is less anchor point locality information than **Marine Cloud**. It is necessary to add the weight of geometric location information in the loss function to obtain accurate tracking.

Maximum matched distance. Both tobac and pFGW have a parameter of the maximum distance between two matched clouds at adjacent time steps. For tobac, the parameter helps restrain the size of the neighborhood from searching within at the next time step. For pFGW, this parameter helps determine the parameter m, the probability mass (i.e., the amount of "goods") to be preserved in the partial optimal transport (see Sec. 5.3). Because the time interval between snapshots and the domain area is consistent for a dataset, we transform this threshold into a limit of the average speed of clouds. For the **Marine Cloud** dataset, we use the distance threshold corresponding to an average moving speed at $20m/s$ as the highest translation speed of feature points from tobac, following the analysis in [10]. For anchor points from pFGW, this average speed threshold is $30m/s$ because the anchor point position is more sensitive to COD values. For the **Land Cloud** dataset, we increase the average speed threshold $40m/s$ to adapt to more complex low-level cloud changes for both approaches.

Intra-cloud anchor point simplification. When there are too many anchor points for a given cloud object, we reduce the number of anchor points for computational efficiency. In particular, we remove anchor points based on the area of their topological zones. For example, two of the five cloud objects in Fig. 7(a) contain multiple local maxima as anchor points (in red). We highlight the local maxima with green and yellow circles, respectively, where the associated topological zones of their adjacent edges fall below a specified threshold. As we remove these local maxima and their pairing saddles, we obtain the simplified subtrees inside each cloud object; see Fig. 7(b). Subsequently, both cloud objects now contain one less anchor point.

Anchor point simplification can be prone to instabilities. Although persistence diagrams remain stable under Wasserstein and Bottleneck distances, the critical points involved in the pairings themselves may not exhibit such stability. For instance, in Fig. 7 (a), the lower-left component features a persistence pairing between a saddle point (shown in white) and a local maximum (in red, marked with a green circle). A slight perturbation in the underlying scalar field could cause the saddle point to change its pairing partner—potentially switching to the other local maximum (anchor point) within the same component. In our framework, the user defines a distance threshold to constrain anchor point movement. Anchor points from consecutive time steps are not matched if their separation exceeds this threshold. This approach helps reduce, though not entirely eliminate, the instability problem.

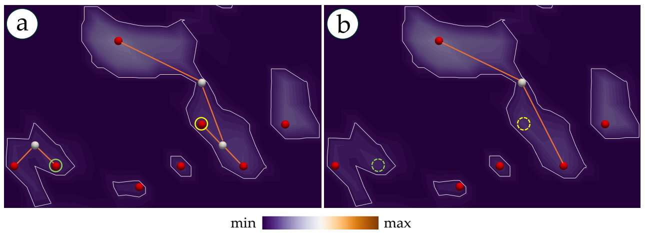

Figure 7: Left: a set of cloud objects enclosed by white contours; each contains a subtree of the global merge tree. Local maxima (a.k.a., anchor points) are in red, and saddles are in white. Right: simplifying subtrees by removing the highlighted anchor points (inside green or yellow circles) and their pairing saddles.

The threshold for intra-cloud simplification is determined based on the merge tree size and the available computational resources. Our framework computes the pFGW distance, which requires $O(n^2)$ space and $O(n^3)$ time, where n denotes the (maximum) size of the merge trees. However, the actual runtime is typically lower due to the use of sparse matrix multiplication. To control merge tree size, we progressively increase the threshold, eliminating anchor points located on merge tree edges whose associated topological zones fall below the threshold. This process continues until the tree size is reduced to a manageable level. In practice, we increment the topological zone area threshold by 5, starting from 0, until all merge trees contain fewer than 5000 nodes. For the **Marine Cloud** dataset, this threshold is set at 30 pixels; for the **Land Cloud** dataset, this threshold is set at 5 for both the morning data and the midday data.

Strictness of matchings for cloud systems. In Sec. 5.4, we discussed the strategy to compute the matching between the cloud system $X \in H_t$ and the cloud system $Y \in H_{t+1}$ using the optimal coupling matrix. A matching between X and Y is *valid* if either of the two conditions is satisfied: (1) the two cloud systems X and Y are the mutual best match, or (2) their matching score $S_t(X, Y)$ exceeds a threshold controlled by a parameter r; see Sec. 5.4 for details. The parameter r controls the strictness of the matching criteria. When r is large (e.g., ≥ 0.5), condition (2) is only met if condition (1) is also fulfilled, rendering condition (2) effectively redundant. On the other hand, when r is small (e.g., < 0.1), the matching criteria may become overly permissive, increasing the risk of mismatches despite low matching scores. To determine an appropriate r value, we conduct experiments using the pFGW framework with $r \in \{0.1, 0.2, 0.3\}$ and evaluate the resulting trajectory statistics shown in Fig. 8.

We first examine the trajectory timespan distributions with different choices of r using the histograms in Fig. 8 and the box plots in Fig. 9 1st column, respectively. For the **Marine Cloud** dataset, the histograms of timespan distributions in the 1st row of Fig. 8 are similar. We only observe minor differences for trajectories with a timespan of more than 300 minutes. However, in the box plots, the trajectories with $r = 0.1$ have a higher interquartile range of the timespan than the ones with $r = 0.2$ and $r = 0.3$; see Fig. 9 1st row, 1st column.

For the **Land Cloud** dataset, the histograms reveal more noticeable differences in the distributions of trajectory timespans; see Fig. 8 2nd and 3rd rows. For example, when $r = 0.3$, the

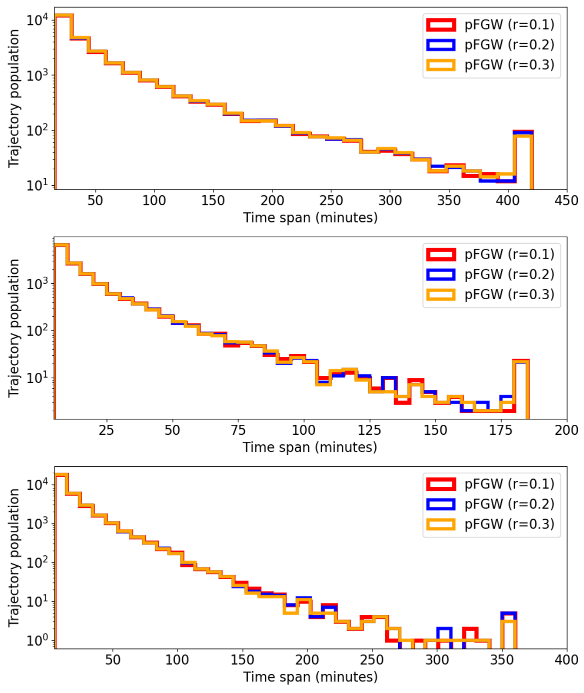

Figure 8: Distributions of trajectory timespans in log-scale for pFGW tracking cloud systems with $r \in \{0.1, 0.2, 0.3\}$. From top to bottom: the **Marine Cloud** dataset (1st row), the **Land Cloud** dataset during the morning (2nd row) and the midday (3rd row) period. Other experimental settings are the same as in Sec. 6.4 and Sec. 6.5, respectively.

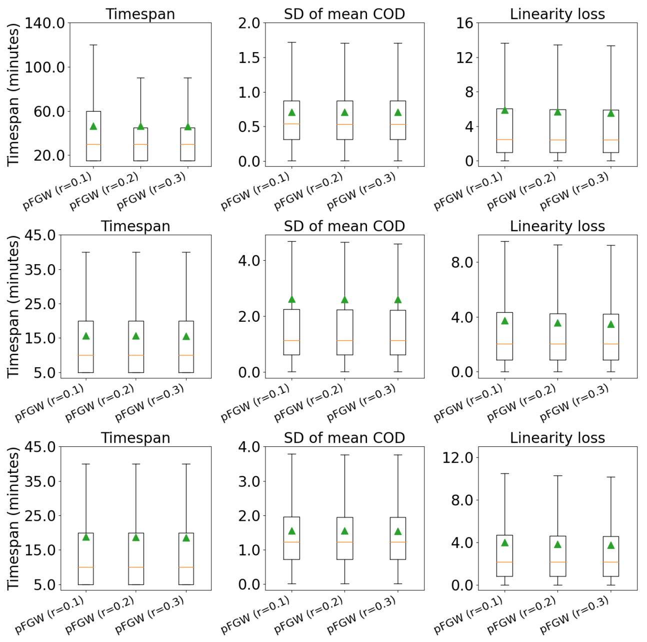

Figure 9: Box plots showing the median (orange line), mean (green triangle), and interquartile range (box boundary) of the distribution for three topology-based tracking methods. From top to bottom: the **Marine Cloud** dataset (1st row), the **Land Cloud** dataset during the morning period (2nd row) and during the midday (3rd row).

tracking results contain fewer long-lived trajectories during the midday, indicating that $r = 0.3$ might be too high for the **Land Cloud** dataset. Nonetheless, the overall statistics shown in the box plots of the timespan distributions for the **Land Cloud** dataset (Fig. 9, second and third rows, first column) remain largely similar. Changing the parameter r within the $[0.1, 0.3]$ range does not change the timespan distribution significantly.

We analyze the standard deviation of mean COD and the linearity loss for trajectories that last for at least 45 minutes (three timesteps) in the **Marine Cloud** dataset and 15 minutes (three timesteps) in the **Land Cloud** dataset, respectively. For both datasets, both metrics show similar distributions across all three experiments with different r values; see Fig. 9 2nd and 3rd columns. For example, in the **Marine Cloud** dataset, the average standard deviation of mean COD (represented by green triangles in the second column of box plots in Fig. 9) with $r = 0.1$ is only 0.51% higher than that with $r = 0.3$. Similarly, the mean trajectory linearity loss with $r = 0.1$ is only 6.05% higher than that with $r = 0.3$.

Based on the above statistics, we argue that the pFGW tracking result is not very sensitive to the threshold parameter r within the $[0.1, 0.3]$ range. To highlight the advantage of pFGW in preserving the trajectory timespan, we select $r = 0.1$ for the experiments in Sec. 6.

D.3 Parameter Configuration for Comparative Analysis

For our comparative analysis, we report parameter configurations for pFGW, tobac, and PyFLEXTRKR respectively.

We start with parameters that are shared by all three methods. For all datasets, we use 8-way connectivity to search for neighboring pixels when computing superlevel set components. For the **Marine Cloud** dataset, we choose 2.0 as the single COD value threshold to obtain superlevel set components; we use area-based simplification and remove the cloud objects below 10 pixels. For the **Land Cloud** dataset, we choose 9.0 as the COD value threshold for the morning period (06:00-09:00 UTC) and 10.0 as the threshold for the midday period (09:05-15:00 UTC); we do not simplify any cloud objects by area size.

pFGW uses the following parameters. We use $\alpha = 0.2$ for **Land Cloud** and $\alpha = 0.4$ for **Marine Cloud** to balance the weight between intrinsic and extrinsic information. We adopt $30m/s$ for the **Marine Cloud** dataset and $40m/s$ for the **Land Cloud** dataset as the maximum average speed parameter. We search the parameter m within the range of $[0.6, 0.9]$ and choose the highest one where the maximum distance between matched anchor points is below the distance threshold derived from the maximum average speed.

We use the following additional parameters when running the PyFLEXTRKR framework: We use superlevel set components as cloud objects. For both datasets, the maximum number of objects for a cloud object to correspond to in the next time step is 10. The overlap percentage threshold is 30%. The maximum number of clouds in the domain is 3000 for every time step. When a cloud object splits, the minimum region overlap size for the main trajectory to be assigned to is 3 pixels for **Marine Cloud** and 1 pixel for **Land Cloud**.

Our experiments with tobac use the following additional parameters. During cloud detection, tobac uses the barycenter of the bounding box of detected features as feature points of cloud objects. The barycenter is weighted by the COD value minus the superlevel set threshold. This barycenter strategy is labeled "weighted_diff" in the tobac tool. When tracking feature points, tobac uses the "predict" method, which predicts the translation direction of the feature based on its previous trajectory. Besides, tobac uses adaptive search for the maximum size of a feature point set to locate the matching feature in the next time step. The initial value of this parameter is 20, and decays at a speed of 0.9 until it reaches 5. For the **Marine Cloud** dataset, the maximum average speed of clouds between adjacent snapshots is $20m/s$. For the **Land Cloud** dataset, it is $40m/s$.

D.4 Limitations of Comparative Analysis

There is a limitation in our comparative analysis: tobac employs a multi-threshold cloud detection process while both pFGW and PyFLEXTRKR uses a single threshold for cloud detection. This multi-threshold detection process can find superlevel sets at different thresholds and summarize all the detected cloud objects. A typical use of this process is to break down a large cloud object into several objects at a higher threshold. For a fair comparison, we use the same threshold for cloud detection for all three methods.

E Comparison with Topology-Based Tracking Tools

In this section, we provide additional details for comparing the three topology-based tracking tools: pFGW, LWM, and MTW; see Sec. 6.6 for an overview. We start with the experimental settings for LWM and MTW in Appendix E.2. Next, we provide a qualitative evaluation using the **Marine Cloud** dataset in Appendix E.3, followed by a comparison using the **Land Cloud** dataset in Appendix E.4.

E.1 Description

LWM extends the Wasserstein distance between persistence diagrams by incorporating critical point locations. It aims to solve a minimum-cost matching problem between persistence diagrams, where the cost of matching is a linear combination of the Wasserstein distance and the Euclidean distance between the matched critical points. Additionally, the cost of adding or removing a persistence pair is a linear combination of its persistence and the Euclidean distance between its two critical points. In contrast, the MTW framework extends the edit distance between merge trees by introducing constraints. It computes the edit distance between branch decomposition trees generated from merge trees, with the matching cost based on the persistence of branches. However, this framework does not consider geometric information when tracking features.

E.2 Experimental Settings

Preprocessing. We follow the same process in Sec. 5.1 and Sec. 5.2 to obtain the simplified merge trees of the COD fields. MTW directly uses the simplified merge trees as the input. For LWM, we compute the persistence-based branch decomposition to obtain the persistence diagrams as the input.

Implementation. Both MTW and LWM are implemented as built-in modules of the Topology Toolkit (TTK) [18]. We use the TTK module for MTW computation. However, the TTK module for LWM only accepts simplified scalar fields as the input, while we can only obtain simplified merge trees (or persistence diagrams). Therefore, we implement the LWM approach in-house by modifying the code from the TTK module.

Parameter settings. For pFGW, we use an Euclidean distance threshold at $28km$ for the **Marine Cloud** dataset, which corresponds to an average speed of $\approx 30m/s$; see Appendix D.2 for a discussion on parameter configurations. The parameter settings of LWM focus on the weight balancing the Wasserstein distance between persistence pairs and the Euclidean distance between critical points. We normalize the range of the two distances and set the weight of the Euclidean distance to 1.0. We then gradually increase the weight of the Wasserstein distance, denoted as β, from 0 to assess the impact of persistence information. For MTW, the parameter ε_1 is a threshold to determine whether the saddles for two branches on the tree can be swapped, whereas ε_2 and ε_3 are used to control the persistence scaling during the transformation between branch decomposition trees and merge trees. Due to the high complexity and uncertainty [17] of the COD field, we have no prior knowledge about persistence to tune these parameters. Instead, we follow the recommended parameter setting from its original work [15].

Postprocessing. We compute the cloud system trajectories for LWM and MTW using a postprocessing pipeline similar to that of pFGW.

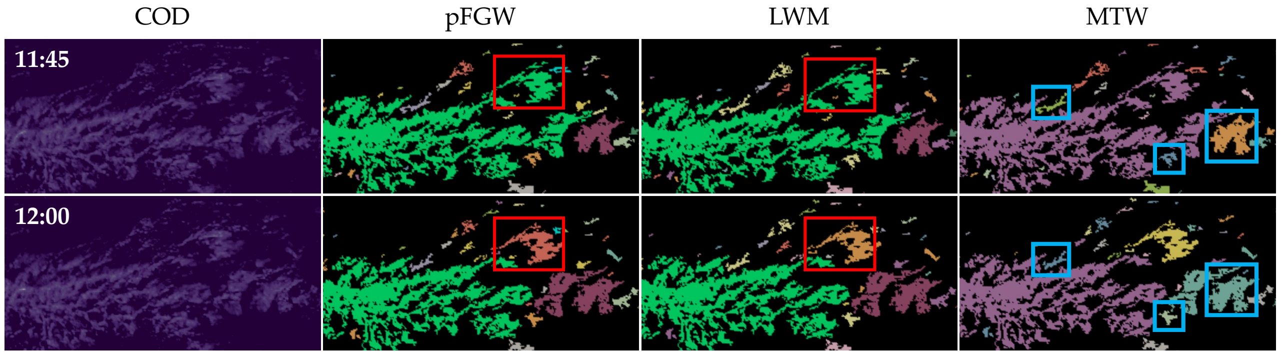

Figure 10: Tracking results of a region in the **Marine Cloud** dataset on Aug 1, 2023. From left to right: visualizations of COD fields, tracking results for pFGW, LWM, and MTW, respectively. Red boxes highlight the area where pFGW and LWM have different tracking results for a cloud splitting event (see Fig. 12 for zoom-in views). Cyan boxes highlight multiple areas where MTW generate incorrect correspondences for cloud systems.

First, we identify anchor points within the cloud system and determine matching scores based on the number of anchor point matchings. Then, we apply the algorithm described in Sec. 5.4 to extract the main trajectories of the cloud systems.

Evaluation. We first compare the anchor point matching results from all three topology-based approaches. Specifically, for LWM, we report the results using $\beta \in \{0.0, 0.1, 0.2\}$ to examine the impact of persistence diagram information on tracking. $\beta = 0$ means that the cost function of LWM uses all geometric information. Increasing β adds the weight of the Wasserstein distance in the cost function. When $\beta = 1.0$, we balance the weight between the Wasserstein distance and the Euclidean distance in the cost function. In addition, we evaluate the cloud tracking performance of the three approaches by comparing the statistics of the metrics for cloud trajectories described in Sec. 6.3.

E.3 Case Study: Marine Cloud Dataset

Anchor point matching. We examine the distribution of the Euclidean distance between matched anchor points in Fig. 11. The left histograms show that the distributions of Euclidean distances are similar between pFGW and LWM, with some differences on the right tail. The Euclidean distances between all matched points for pFGW are below 28km. In comparison, there are matched nodes in the results of LWM with $\beta = 0$ with the Euclidean distance beyond 28km, which is less likely to happen due to physical constraints [10]. This is because the cost of matching points to the diagonal is fixed by the saddle-maxima relation of the persistence pair, losing the flexibility to adapt to what the application needs. As the β value of LWM increases to 0.1 and then to 0.2, the number of matched nodes with short Euclidean distances decreases, and the number of faraway matched nodes increases; see Fig. 11 left. In addition, in Fig. 11 right, the mean and median Euclidean distance between matched nodes for LWM increases as β increases from 0 to 0.2. These observations indicate that adding the weight of the Wasserstein distance does not benefit the anchor point tracking. In contrast to LWM, pFGW uses the tree distance between anchor points to encode the topological information, reflecting the locality of anchor points within cloud objects. Such information is more robust than the persistence diagram information when performing feature-tracking tasks in such complex datasets. Among the three approaches, MTW performs the worst. Without geometric location information, MTW does not find many matchings between nearby anchor points. The mean and median Euclidean distance between matched anchor points is much higher than the other two methods; see Fig. 11 right.

Cloud tracking. In Sec. 6.6, we have evaluated the anchor point the statistics of cloud system trajectories using the three metrics in Sec. 6.3. In this section, we examine the visualizations of tracking results using the data from Aug 1, 2023. We check the transition

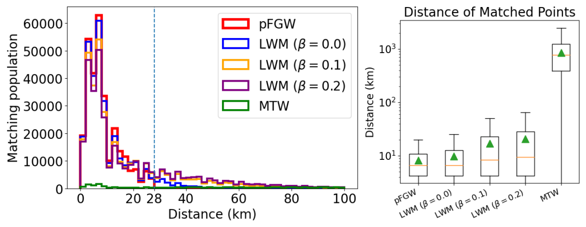

Figure 11: **Marine Cloud** dataset: histograms (left) and box plots in log-scale (right) for distributions of the Euclidean distances between matched anchor points. The Euclidean distance threshold for pFGW is 28km, as highlighted by the vertical dotted line in the histogram.

from 11:45 to 12:00 UTC.

During the transition, a splitting event occurs, as highlighted by the red boxes in Fig. 10. All three methods have detected the splitting event. However, pFGW and LWM demonstrate different cloud system trajectories for this event.

A closer examination in the zoomed-in view in Fig. 12 reveals that in the pFGW results, cloud system A at 11:45 is matched to cloud system C at 12:00, while cloud system B is identified as newly formed after splitting from a larger green cloud system. This matching is reasonable given the geometric proximity of A and C.

In contrast, LWM matches cloud system A' at 11:45 to B' at 12:00 and considers cloud system C' as newly formed. This results in a significant jump in the centroid location from A' to B', increasing the linearity loss of the trajectory. Moreover, we do not observe substantial changes in the COD fields in the corresponding areas in Fig. 10. The matching between cloud system A' and B' is likely a mismatch. Such errors in LWM can occur when the cost of matching the anchor points in the cloud system B' to the diagonal (representing their "birth") is higher than the cost of matching them to other faraway anchor points.

On the other hand, MTW fails to track many cloud systems accurately. The cyan boxes in Fig. 10 highlight multiple areas in which MTW does not generate matchings between the two time steps. This result shows that MTW performs the worst among the three topology-based approaches for this cloud tracking task.

E.4 Case Study: Land Cloud Dataset

We provide the experimental results and statistical evaluations for the three topology-based methods on the **Land Cloud** dataset.

Anchor point matching. Following the approach in Appendix E.3, we evaluate the effect of persistence diagram information on tracking for LWM using $\beta \in \{0.0, 0.1, 0.2\}$. The Euclidean distance distributions between matched anchor points are presented in Fig. 14. For pFGW, we include all matchings with nonzero probability in the

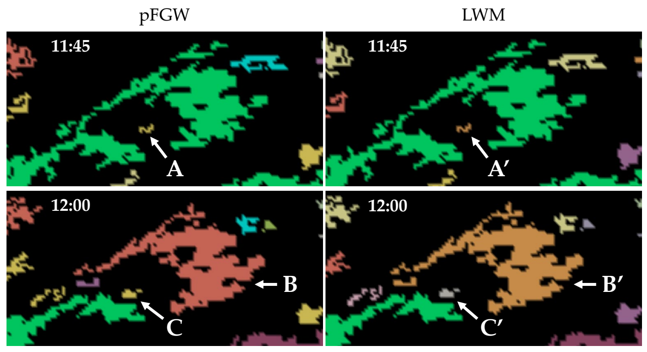

Figure 12: **Marine Cloud** dataset: Zoom-in views for Fig. 10 red boxes to demonstrate the trajectory differences between pFGW and LWM. pFGW matches cloud system A to C, whereas LWM matches A' to B'.

distributions, whereas LWM and MTW provide one-to-one anchor point matchings.

For both the morning and midday data, the Euclidean distance distributions of pFGW and LWM results are similar. In particular, for LWM ($\beta = 0$), only a small fraction of matched anchor points have distances greater than 12km; see Fig. 14 1st column. In contrast, LWM with $\beta = 0.1$ and $\beta = 0.2$ generates fewer short-distance matchings and more long-distance matchings between anchor points compared to LWM with $\beta = 0$. This observation indicates that $\beta = 0$ is the optimal choice for LWM. Increasing the weight of the Wasserstein distance in the cost function does not improve the accuracy of cloud tracking.

We notice that the mean and median Euclidean distances between matched anchor points for LWM ($\beta = 0$) are lower than those of pFGW; see Fig. 14 2nd column. This is because pFGW results include more pairs of matched anchor points with Euclidean distances close to 12km; see Fig. 14 1st column. However, pFGW results may include low-probability anchor point matchings, whose influence on the cloud system tracking results is minimal. Further evaluation of cloud trajectory statistics is necessary for a comprehensive comparison.

As with the **Marine Cloud** dataset, MTW performs the worst in matching nearby anchor points for the **Land Cloud** dataset across the morning and midday. The mean and median Euclidean distances for matched anchor points using MTW are significantly higher than the other two methods; see Fig. 14 2nd column.

Statistical evaluation. Fig. 15 presents the distribution of evaluated metrics for the three topology-based approaches. For LWM, we fix $\beta = 0$ because it performs the best for anchor point matching among all choices of β.

We first examine the trajectory timespan distribution. MTW results exhibit the highest mean trajectory timespan for both the morning and the midday data. Between the other two methods, pFGW achieves a slightly higher mean trajectory timespan than LWM for both the morning and midday periods; see Fig. 15 1st column.

Next, we check the standard deviation of the mean COD for trajectories that last for at least 15 minutes (three time steps). All three approaches have comparable distributions for this metric across the morning and midday; see Fig. 15 2nd column.

For the trajectory linearity loss, MTW trajectories exhibit significantly higher loss than those from pFGW and LWM, indicating frequent mismatches between distant cloud systems (Fig. 15 3rd column). Comparing pFGW against LWM, pFGW trajectories have a lower mean linearity loss and interquartile range than LWM (Fig. 15 3rd column), indicating better preservation of linearity. Overall, pFGW demonstrates the most reliable performance in maintaining the linearity of cloud system trajectories among the three topology-based approaches.

Cloud tracking. Lastly, we compare the cloud tracking results on the **Land Cloud** dataset using visualizations for our case study. We use the same set of data in Sec. 6.5 for evaluation. That is, we examine the time transition from 08:30 to 08:35 UTC for the morning data and from 12:05 to 12:10 UTC for the midday. New cloud entities at 8:35 UTC and 12:10 UTC are colored in magenta, which we use to evaluate the ability of each method to maintain consistent cloud system tracking.

The tracking results are demonstrated in Fig. 13. During the morning transition (1st and 2nd rows), small cloud systems emerge near the center of the COD field. For this transition, pFGW and LWM generate similar results in tracking these cloud systems; see the 2nd and 3rd columns of the first two rows. MTW, on the other hand, identifies many small cloud systems as tracked from the previous time step in the morning period. For example, in the 2nd row 4th column of Fig. 13, many small cloud systems in the center of the image are not colored in magenta, indicating that they are tracked from other systems at the previous time step. This explains why the mean and median timespans for MTW trajectories are high. However, the tracking quality of the MTW results is undesirable. We cannot find many color correspondences for these small cloud systems, see the 1st and 2nd rows of the 4th column in Fig. 13. The cloud system matching between the two time steps looks random. In other words, MTW trajectories are unlikely to reflect the actual cloud system movement.

In the **Land Cloud** dataset, the situation of random matching for MTW occurs because the anchor points of the small cloud systems tend to have their COD values (as well as the persistence of their pairs) similar to the cloud detection threshold. MTW, relying on the persistence information for critical point matching, tends to match anchor points with similar persistence information. Consequently, MTW tends to match these small cloud systems without considering the geometric proximity.

The tracking results for the midday transition (cf. Fig. 13 3rd and 4th rows) yield similar findings. Both pFGW and LWM produce visually comparable results, whereas MTW fails to track cloud systems correctly. For example, MTW loses the trajectory for two prominent cloud systems in Fig. 13 cyan boxes during the morning transition (see 1st and 2nd rows, 4th column) and in Fig. 13 red boxes during the midday transition (see 3rd and 4th rows, 4th column), respectively. This further demonstrates the limitations of MTW in tracking cloud systems.

F Runtime Analysis of Cloud Tracking Tools

We compare the runtime performance of the cloud-tracking methods using the hardware configuration described in Appendix B. Runtime for data transfer between modules, including loading and saving, is omitted. For each method, we first review the computational steps and then report runtimes for a set of instances from both the **Marine Cloud** and **Land Cloud** datasets. All three methods are evaluated on the same set of instances for comparison.

pFGW runtime. We begin by discussing the runtime of our pFGW implementation. The process starts with computing the merge tree of the scalar field using TTK [19], followed by intra-cloud critical point simplification for each merge tree. We then construct the measure network from the merge tree and compute the partial optimal transport for anchor-point probabilistic matching. Finally, we aggregate the anchor-point matches to obtain the cloud system trajectories.

For the **Marine Cloud** instances used in the runtime analysis, the original merge tree contains on average 61,234 nodes, which is reduced to 3,664 nodes after simplification. For the **Land Cloud** dataset, the original merge tree has on average 15,334 nodes, reduced to 3,900 after simplification. The breakdown of the average runtime for our pFGW implementation is shown in Tab. 1.

Following Tab. 1, by approximating the number of comparisons as

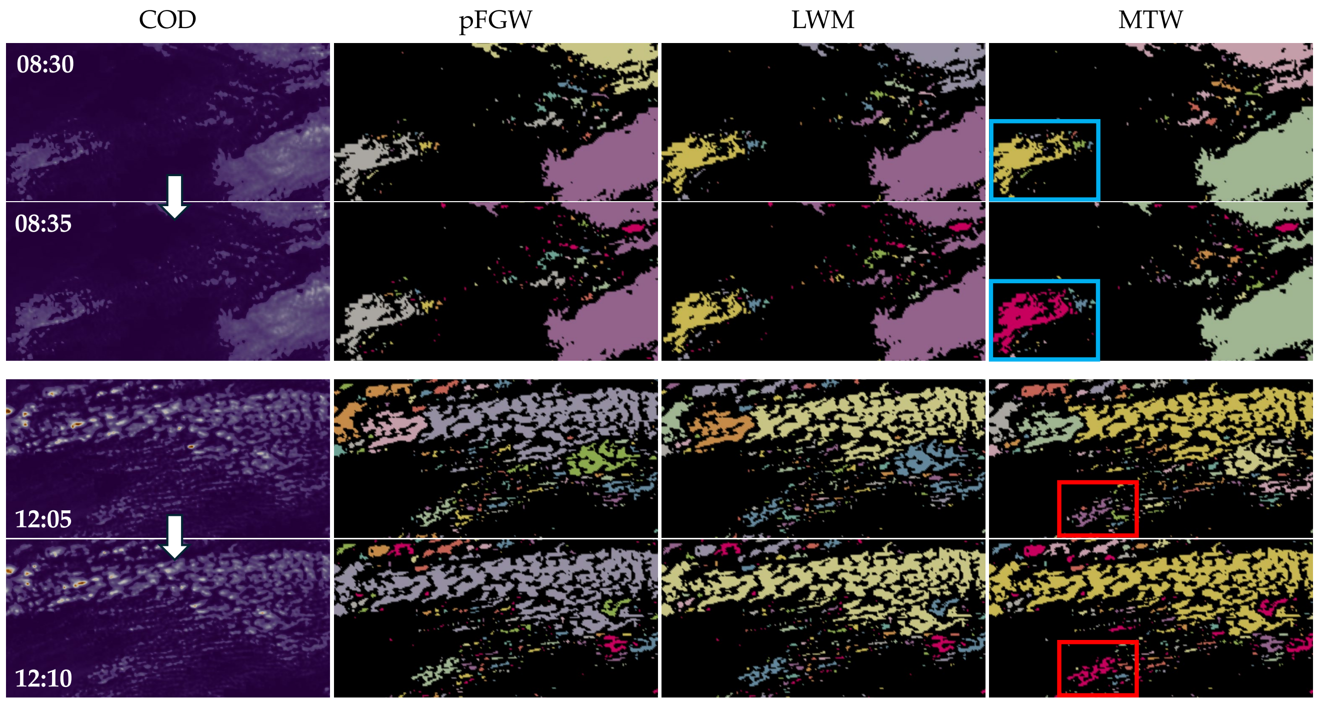

Figure 13: Tracking results of a region in the **Land Cloud** dataset on May 1, 2018. From left to right: visualizations of COD fields, tracking results for pFGW, LWM, and MTW, respectively. The top two rows (resp. bottom two rows) are for the transition during the morning (resp. midday) period. All new cloud entities in the 2nd and 4th rows are colored magenta; others are colored by correspondences. Cyan boxes and red boxes highlight the areas where MTW fails to track cloud systems with noticeable areas in the morning and the midday, respectively.

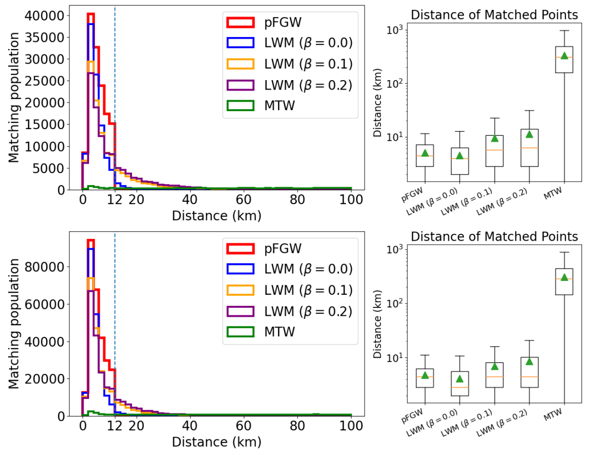

Figure 14: **Land Cloud** dataset: histograms (left) and box plots (right) for distributions of the Euclidean distances between matched anchor points. The Euclidean distance threshold for pFGW is 12km, as highlighted by the vertical dotted line in the histograms. The top row is for the morning data, and the bottom is for the midday data.

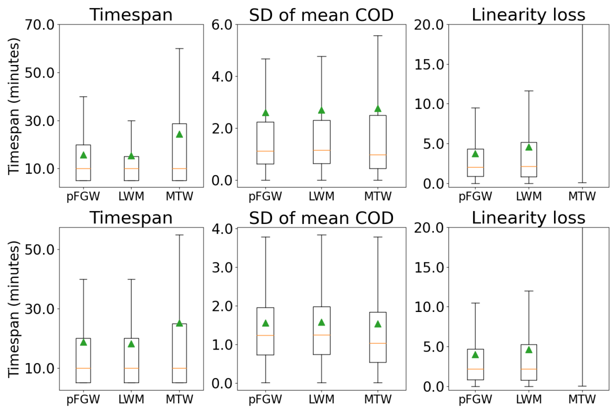

Figure 15: **Land Cloud** dataset: box plots showing the median (orange line), mean (green triangle), and interquartile range (box boundary) of the distribution for three topology-based tracking methods. The top row is for the morning data, whereas the bottom is for the midday data. The boxes for the linearity loss for MTW exceed the plots' upper bound.

the number of instances (differing by at most one), the total runtime of pFGW is 417.80 seconds per instance for **Marine Cloud** and 430.62 seconds per instance for **Land Cloud**.

tobac runtime. Second, we examine the runtime of tobac. The first step identifies feature points in the scalar field, followed by generating cloud object segmentations from these points. Finally, tobac computes matches between cloud objects and links them into trajectories. For the runtime analysis, the **Marine Cloud** instances contain on average 515 cloud objects per instance, while the **Land Cloud** instances contain 259 on average. The average runtime for each phase of tobac is shown in Tab. 2. Similarly, we estimate the number of comparisons by the number of instances, and the total runtime for tobac is 3.85 sec per instance for **Marine Cloud** and 1.96 sec per instance for **Land Cloud**.

PyFLEXTRKR runtime. Finally, we evaluate the runtime of PyFLEXTRKR. The first step segments the scalar field into superlevel set components (i.e., cloud objects), followed by computing region overlaps. The main trajectories are then constructed based on the overlap areas. For the runtime analysis, **Marine Cloud** instances contain on average 569 cloud objects, while **Land Cloud** instances contain 356. The average runtimes for each step are summarized in Tab. 3.

Therefore, by estimating the number of comparisons from the number of instances, the average total runtime per instance is 22.97 seconds for **Marine Cloud** and 3.70 seconds for **Land Cloud**.

Step	Marine Cloud	Land Cloud
Computing merge trees	0.84	0.11
Intra-cloud simplification	2.96	0.88
Measure network initialization	92.25	97.52
Optimal transport	309.57*	330.13*
Generate cloud systems and trajectories	12.18	1.98

Table 1: Average runtime breakdown of the pFGW implementation for **Marine Cloud** and **Land Cloud**. Times are reported in seconds per instance, except for optimal transport*, which is measured in seconds per comparison.

Step	Marine Cloud	Land Cloud
Feature point detection	2.25	0.85
Cloud object segmentation	1.39	1.02
Cloud object tracking	0.21*	0.09*

Table 2: Average runtime breakdown of the tobac implementation for **Marine Cloud** and **Land Cloud**. Times are reported in seconds per instance, except for cloud object tracking*, which is measured in seconds per comparison.

Step	Marine Cloud	Land Cloud
Cloud object segmentation	2.89	0.47
Region overlap computation	8.25*	1.18*
Generating cloud trajectories	11.83*	2.05*

Table 3: Average runtime breakdown of the method for **Marine Cloud** and **Land Cloud**. Times are in seconds per instance, except steps marked with *, which are measured per comparison.

Summary and discussion. In summary, pFGW exhibits a considerably longer runtime than tobac and PyFLEXTRKR. This is expected, as pFGW solves the optimal matching problem over a much larger set of nodes (anchor points), whereas tobac represents each cloud object by a single centroid and PyFLEXTRKR restricts matching to local neighborhoods based on regional overlap. In addition, both tobac and PyFLEXTRKR rely heavily on optimized algorithms from Python libraries, while most of pFGW's computation—apart from merge tree generation and matrix multiplication in optimal transport—is implemented directly in Python with limited use of external libraries. Significant performance improvements could be achieved by reimplementing these components in C++ and interfacing them with Python.

On the other hand, because the tracking problem naturally lends itself to parallelization, all matching and per-instance computations can be executed independently, with only the final trajectory connection requiring sequential processing. As a result, all three approaches remain computationally viable for practical applications. Moving forward, we aim to further optimize the optimal transport implementation to enhance runtime performance.

References

[1] J. Ahrens, B. Geveci, and C. Law. Paraview: An end-user tool for large-data visualization. *The Visualization Handbook*, page 717, 2005. 1

[2] H. Deneke et al. Increasing the spatial resolution of cloud property retrievals from meteosat seviri by use of its high-resolution visible channel: implementation and examples. *Atmospheric Measurement Techniques*, 14(7):5107–5126, 2021. 1

[3] Q. Han, W. B. Rossow, and A. A. Lacis. Near-global survey of effective droplet radii in liquid water clouds using ISCCP data. *Journal of Climate*, 7:465–497, 1994. 1

[4] J. F. Meirink et al. CLAAS-3: CM SAF CLoud property dAtAset using SEVIRI - Edition 3, 2022. 1

[5] J. H. Schween et al. Life cycle of stratocumulus clouds over 1 year at the coast of the atacama desert. *Atmospheric Chemistry and Physics*, 22(18):12241–12267, 2022. 1

[6] S. A. Klein and D. L. Hartmann. The seasonal cycle of low stratiform clouds. *Journal of Climate*, 6(8):1587 – 1606, 1993. 1

[7] M. Li, X. Yan, L. Yan, T. Needham, and B. Wang. Partial Fused Gromov-Wasserstein distance for topology tracking. https://github.com/tdavislab/GWMT, 2023. 1

[8] M. Li, X. Yan, L. Yan, T. Needham, and B. Wang. Flexible and probabilistic topology tracking with partial optimal transport. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–18, 2025. 1, 2, 3

[9] M. D. King et al. Remote sensing of liquid water and ice cloud optical thickness and effective radius in the arctic: Application of airborne multispectral MAS data. *Journal of Atmospheric and Oceanic Technology*, 21(6):857 – 875, 2004. 1

[10] M. Heikenfeld et al. tobac 1.2: towards a flexible framework for tracking and analysis of clouds in diverse datasets. *Geoscientific Model Development*, 12(11):4551–4570, 2019. 2, 3, 6

[11] W. P. Menzel. Cloud tracking with satellite imagery: From the pioneering work of Ted Fujita to the present. *Bulletin of the American Meteorological Society*, 82(1):33 – 48, 2001. 1

[12] L. Najman and M. Schmitt. Watershed of a continuous function. *Signal Processing*, 38(1):99–112, 1994. Mathematical Morphology and its Applications to Signal Processing. 1

[13] T. Nakajima and M. D. King. Determination of the optical thickness and effective particle radius of clouds from reflected solar radiation measurements. part i: Theory. *Journal of the Atmospheric Sciences*, 47(15):1878–1893, Aug. 1990. 1

[14] P. Virtanen et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 1

[15] M. Pont, J. Vidal, J. Delon, and J. Tierny. Wasserstein distances, geodesics and barycenters of merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 2021. 5

[16] R. A. Roebeling, A. J. Feijt, and P. Stammes. Cloud property retrievals for climate monitoring: Implications of differences between spinning enhanced visible and infrared imager (SEVIRI) on METEOSAT-8 and advanced very high resolution radiometer (AVHRR) on NOAA-17. *Journal of Geophysical Research: Atmospheres*, 111(D20), 2006. 1

[17] S. Platnick et al. The MODIS cloud optical and microphysical products: Collection 6 updates and examples from terra and aqua. *IEEE Transactions on Geoscience and Remote Sensing*, 55(1):502–525, 2017. 1, 5

[18] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1), 2018. 1, 5

[19] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The Topology ToolKit. https://topology-tool-kit.github.io/, 2021. 7

[20] F. Werner and H. Deneke. Increasing the spatial resolution of cloud property retrievals from Meteosat SEVIRI by use of its high-resolution visible channel: evaluation of candidate approaches with MODIS observations. *Atmospheric Measurement Techniques*, 13(3):1089–1111, Mar. 2020. 1