Depth for Multi-Modal Contour Ensembles

N.F. Chaves-de-Plaza[†1,5] , M. Molenaar[1] , P. Mody[2,5] , M. Staring[2,3] ,
R. van Egmond[1] , E. Eisemann[1] , A. Vilanova[4] , K. Hildebrandt[1]

[1]TU Delft, Netherlands
[2]Department of Radiology, Leiden University Medical Center, Netherlands
[3]Department of Radiation Oncology, Leiden University Medical Center, Netherlands
[4]TU Eindhoven, Netherlands
[5]HollandPTC, Netherlands

Abstract

The contour depth methodology enables non-parametric summarization of contour ensembles by extracting their representatives, confidence bands, and outliers for visualization (via contour boxplots) and robust downstream procedures. We address two shortcomings of these methods. Firstly, we significantly expedite the computation and recomputation of Inclusion Depth (ID), introducing a linear-time algorithm for epsilon ID, a variant used for handling ensembles with contours with multiple intersections. We also present the inclusion matrix, which contains the pairwise inclusion relationships between contours, and leverage it to accelerate the recomputation of ID. Secondly, extending beyond the single distribution assumption, we present the Relative Depth (ReD), a generalization of contour depth for ensembles with multiple modes. Building upon the linear-time eID, we introduce CDclust, a clustering algorithm that untangles ensemble modes of variation by optimizing ReD. Synthetic and real datasets from medical image segmentation and meteorological forecasting showcase the speed advantages, illustrate the use case of progressive depth computation and enable non-parametric multimodal analysis. To promote research and adoption, we offer the contour-depth Python package.

CCS Concepts

*• **Human-centered computing** → **Scientific visualization**; • **Mathematics of computing** → **Nonparametric statistics**; Statistical graphics; **Cluster analysis;***

1. Introduction

The problem of analyzing the distributional properties of contour ensembles arises in a wide range of domains like meteorology, where analysts need to interpret multiple simulation runs [LP08]; medicine, where clinicians plan interventions using robust representations of the organs [KHS*19]; and biology [MM22], where changes in cells' morphology across a population of cells can be indicative of looming disease. The contour depth methodology has become established to visually analyze contour ensembles in terms of their representatives, confidence bands, and outliers. Examples include analyzing variations of meteorological forecasts [WMK13] and determining representative and outlying contours in medical image segmentations [MW18, CdPMS*24].

There are two contour depth notions available: Inclusion Depth (ID) [CdPMS*24] and Contour Band Depth (CBD) [WMK13]. ID assesses the number of times the contour contains and is contained by other contours. CBD determines the centrality of a contour by counting the number of times it falls in the band formed by tuples of other contours in the ensemble. When dealing with real data, contours tend to intersect multiple times. Non-nested pairs of contours do not contribute to the

depth score, resulting in less discriminative CBD and ID depths. To overcome this challenge, epsilon ID (eID) and eCBD consider partial containment. The depth scores that ID and CBD yield can be used to summarize contour ensembles in terms of their representatives, confidence bands, and outliers, which can be visualized using contour boxplots [WMK13].

The main practical limitation of contour depth methods is their scalability. Most practical implementations of CBD only consider bands formed by pairs of contours. Even then, given that there are N^2 bands formed by pairs of contours in a N-contour ensemble, CBD takes $\mathcal{O}(MN^3)$ operations to compute an ensemble's depth, where M is the resolution of the domain used to perform the contour comparisons. By only considering pairwise relationships, ID provided an order-of-magnitude speedup taking $O(MN^2)$ time, without sacrificing performance (i.e., ID and CBD yield comparable depth estimates). Nevertheless, this might not be sufficient in use cases that require multiple depth evaluations like interactive analysis of large contour ensembles and clustering [Jör04].

In this paper, we accelerate the computation of ID. In particular, we present a linear time algorithm for computing eID that leverages precomputed inclusion fields. Computing a contour's depth reduces to querying these fields in $\mathcal{O}(M)$ time. Moreover, we introduce the inclusion matrix, which encodes the inclusion relationship between pairs of contours, for accelerating the recomputation of an

---

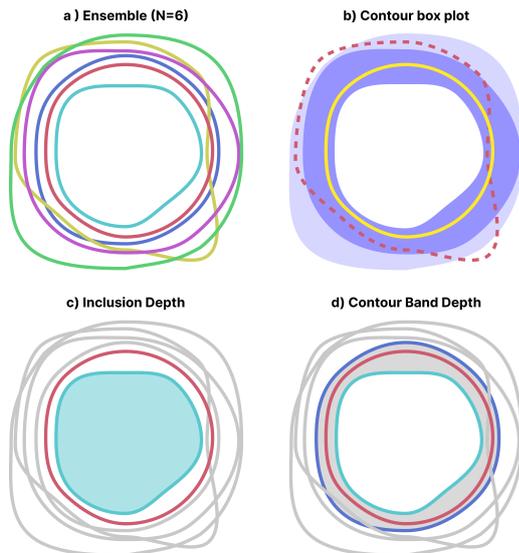† Corresponding author. E-mail: n.f.chavesdeplaza@tudelft.nl

Figure 1: *Computation of Inclusion Depth (ID) and Contour Band Depth (CBD) for the six-member ensemble in (a). In (c), ID involves evaluating containment relationships between contour pairs. CBD (d) counts the number of times contours fall within bands defined by a subset of contours, shown in purple and blue. Additionally, (b) presents depth scores through contour boxplots, providing a statistical summary of the ensemble with median (yellow), confidence bands (light and dark purple), and outliers (dashed red line).*

ensemble's depths when adding or removing groups of contours, without requiring to recompute the whole ensemble's depths. The ability to quickly recompute depths is useful when computing depths progressively [SGN12] or when updating an ensemble's configuration based on user interaction; and critical when using procedures that require multiple calls to the depth function like clustering.

A limiting assumption of existing contour depth methods is that contours in the ensemble were drawn from the same distribution. In practical scenarios with multiple modes of variation, global depth analysis may produce unexpected results, such as assigning high-depth scores to points that are outliers within one mode but centrally located in the overall ensemble [PD23].

We overcome the uni-modality assumption by introducing an extension of the contour depth framework for multi-modal ensembles. Central to this extension is the use of relative depth (ReD). By optimizing the ensemble's average ReD, the CDclust algorithm disentangles its modes of variation. Each iteration of CDclust entails calling a contour depth procedure several times on subsets of the data. Therefore, crucial to CDclust's practical application are the newly introduced fast depth computation schemes. Through experiments with synthetic datasets, we illustrate how ReD and CDclust facilitate non-parametric analysis of multi-modal ensembles. Additionally, we show two case studies in the fields of medical image segmentation and meteorological forecasting that further demonstrate the practical utility of the multi-modal depth toolkit.

In summary, our main contributions are:

- Schemes for accelerated computation and recomputation of contour

depths, in particular, a linear time algorithm for eID and the inclusion matrix, which removes the dependency on the contours' domain resolution when recomputing depths on subsets of the ensemble. These speedups are crucial to enable use cases like progressive depth computation and clustering.
- The first framework for multi-modal depth analysis of contour ensembles. The CDclust algorithm leverages the inclusion matrix to disentangle modes of variation in a contour ensemble by maximizing its average relative depth.

2. Related Work

Our research advances uncertainty visualization methods when using ensembles to characterize underlying distributions. Ensembles permit quantifying uncertainty related to initial conditions, training data, or model parameters [APH*21]. When visualizing ensembles, the data type, dimensionality, and analytical tasks must be considered [WHLS19]. We focus on ensembles of contours derived from spatial data, addressing scenarios like thresholding scalar fields.

Spaghetti plots are commonly used to display contour ensembles, but they become cluttered and less trustworthy for larger ensembles [SZD*10, PFCB23]. Our focus is on providing an overview of the statistical properties of the ensemble such as its representatives, confidence bands, and outliers. Existing methods are categorized into parametric and non-parametric approaches. Parametric methods assume a distribution, such as Gaussian models fitted to contours' PCA-reduced signed distance fields (SDF) [FKRW16, FBW16] or Gaussian models at each grid point [PH11]. Non-parametric methods, like Contour Probability Plots [KTB*18] and EnConVis [ZLC*23], avoid distributional assumptions and offer accurate point-wise descriptions. A hybrid approach uses pairwise contour comparisons to determine centrality [DJW16].

Contour depths, a nonparametric method, exhibit desirable properties such as sensitivity to shape and topology, making them suitable for downstream analyses like clustering [Jör04] and regression [PVB13]. Contour Band Depth [WMK13], while effective for ensemble characterization, scales poorly with the size of the ensemble. A recently proposed alternative with more favorable scaling behavior is the Inclusion Depth [CdPMS*24]. In this paper, we unify both depth notions using the inclusion matrix, capturing the topological relationships among ensemble members. The proposed approach achieves an order of magnitude speedup in Contour Band Depth (CBD) and accelerates the recomputation of depths, which is relevant in interactive scenarios and clustering. Furthermore, we present a linear algorithm for epsilon Inclusion Depth (eID), enabling using eID with large contour ensembles.

Depth methods, assuming a uni-modal distribution, may yield unexpected results in the presence of multiple modes. Previous research addresses mode variation in contour ensembles through clustering. We leverage depth to support this process. Notable approaches include detecting multi-scale symmetries using high-dimensional transform-invariant spaces and nearest neighbor search [TN14]; and using lower-dimensional representations like PCA-reduced contours SDFs [FKRW16] with existing clustering methods such as KMeans [KTB*18], density-based clustering [ME19] and agglomerative hierarchical clustering [FBW16], which favors compact elliptical clusters for Gaussian mixture model fitting [FKRW16]. Finally, the EnConVis framework for contour ensemble analysis emphasizes the importance of the distance function in clustering and classification tasks [ZLC*23].

Depth methods enhance clustering but are yet to be explored in contour contexts. Notable instances include a scheme for clustering multi-variate data using l1 depth [Jör04], recently adapted to use curve depth [LdMMV21]; the bisecting k-spatialMedian algorithm based on spatial or l1 depth [DDPW07]; depth-based clustering analysis (DBCA) for affine-invariant and noise-robust clustering [JCSW16]; CRAD, a density-based clustering algorithm employing robust data depth [HG17]; the depth difference (DeD) metric for determining optimal cluster count [PB19], and depth-based medoids clustering algorithm (DBMCA) for high-dimensional directional data [PD23].

3. Background: Contour Depth

The contour statistical depth methodology permits characterizing an ensemble of contours in terms of the centrality, or alternatively outlyingness, of their members. In the following, we discuss the two main notions of contour depth: Inclusion Depth and Contour Band Depth. Figure 1 illustrates the available contour depth notions and how to visualize an ensemble's summary statistics using contour boxplots.

Inclusion Depth Let C be an ensemble of N contours. The Inclusion Depth (ID) of $c_i \in C$ results from the number of other contours that c_i contains and in which c_i is contained [CdPMS*24]:

$$\text{ID}(c_i|C) = \frac{2}{N}\min\{IN_{in}(c_i), IN_{out}(c_i)\} \text{ with}$$

$$\text{IN}_{in}(c_i) = \sum_{j=1}^{N} in(c_i) \subset in(c_j), \text{ and} \tag{1}$$

$$\text{IN}_{out}(c_i) = \sum_{j=1}^{N} in(c_j) \subset in(c_i),$$

where $in(c_i)$ denotes the subset in the plane enclosed by the contour and $\subset$ yields 0 or 1, depending on the contours' inclusion relationship. The ID values range between [0,1]. When using bitmaps of M pixels to represent contours, ID has a computational complexity of $\mathcal{O}(MN^2)$.

Contour Band Depth The Contour Band Depth (CBD) of $c_i \in C$ is the average number of times that the contour falls inside the band formed by any other J-band with $J \in \{2,3,4,...,N-1\}$ [WMK13]. We say a contour c_i falls in the band formed by J other contours if it contains the contours' intersection and is contained by their union:

$$CB(c_i|c_1,...c_j) = \bigcap_{j=1}^{j} in(c_j) \subset in(c_i) \text{ and } in(c_i) \subset \bigcup_{j=1}^{j} in(c_j) \tag{2}$$

Contour Band Depth (CBD) can be written as

$$\text{CBD}(c_i|C) = \sum_{j=2}^{J} \frac{1}{\binom{N}{j}} \sum_{k=1}^{\binom{N}{j}} CB(c_i|B_k^j), \tag{3}$$

where B_k^j is the kth band of the set of j-contours bands. The CBD values range between [0, 1]. CBD is computationally expensive for $J > 2$, so, in practice, $J = 2$ is used. In Sec. 5, we illustrate how to obtain compute CBD in $\mathcal{O}(N^2)$ time using the inclusion matrix.

Epsilon Contour Depth When contours intersect, there tend to be ties (*i.e.*, pairs of contours for which neither contains the other) and

low depth scores. To mitigate this, variants of CBD and ID have been introduced that use the modified epsilon subset operator

$$A \subset_\varepsilon B = 1 - \begin{cases} 0 & |A| = 0, \\ |A-B|/|A| & \text{otherwise,} \end{cases} \tag{4}$$

where $|A|$ denotes the area of A, $A - B$ the relative set difference and $\subset_\varepsilon$ outputs a continuous value between [0,1].

The modified epsilon ID (eID) [CdPMS*24], replaces $\subset$ in Eq. 1 with $\subset_\varepsilon$. Similarly, the modified CBD, which we will refer to as epsilon CBD (eCBD), replaces $\subset$ in Eq. 2 with $\subset_\varepsilon$, yielding the epsilon band containment operator

$$CB_\varepsilon(c_i|c_1,...c_j) = \min\left(\bigcap_{j=1}^{j} in(c_j) \subset_\varepsilon in(c_i), in(c_i) \subset_\varepsilon \bigcup_{j=1}^{j} in(c_j)\right), \tag{5}$$

Computing eID takes $\mathcal{O}(MN^2)$ time.

Computing eCBD entails forming a $N \times \sum_J \binom{N}{J}$ matrix listing the outputs of Eq. 5. Individual depth values are then computed by thresholding and averaging matrix entries. Because eCBD requires assembling the complete matrix, it is not possible to apply the same acceleration strategy as for CBD. Therefore, eCBD has a complexity of $\mathcal{O}(MN\sum_J \binom{N}{J})$.

4. Linear Epsilon Inclusion Depth Computation

The Epsilon Inclusion Depth (eID) replaces the subset operator in Eq. 1 by the epsilon subset operator, defined in Eq. 4, to compute the proportion of area of one contour that is contained in another (IN_{in}^ε and IN_{out}^ε). By reorganizing the loops in these expressions, it is possible to obtain an algorithm to compute eID in $\mathcal{O}(NM)$. In the following, we simplify notation by using $c_i = in(c_i)$. $c_i(m)$ yields c_i's value at the mth domain point.

Eq. 6 provides the derivation for IN_{in}^ε. We start by plugging Eq. 4 into IN_{in} in Eq. 1. Note that the set difference can be written as a loop over the M bitmap pixels of a contour, where $c_i(m) = 1$ if pixel m is in contour i, and 0 otherwise. We compute $\sum_{j=1}^{N}(1 - c_j(m))$ ahead of time and store it in a lookup table $\text{pre}_{in}^\varepsilon(m) = \sum_{j=1}^{N}(1 - c_j(m))$. Computing these values takes $\mathcal{O}(MN)$ time but only needs to be done once for all contours.

$$\text{IN}_{in}^\varepsilon(c_i) = \sum_{j=1}^{N} 1 - \frac{|c_i - c_j|}{|c_i|}$$

$$= N - \frac{1}{|c_i|}\sum_{j=1}^{N}|c_i - c_j|$$

$$= N - \frac{1}{|c_i|}\sum_{j=1}^{N}\sum_{m}^{M}(1 - c_j(m))c_i(m) \tag{6}$$

$$= N - \frac{1}{|c_i|}\sum_{m}^{M}c_i(m)\sum_{j=1}^{N}(1 - c_j(m))$$

$$= N - \frac{1}{|c_i|}\sum_{m}^{M}c_i(m)\text{pre}_{in}^\varepsilon(m)$$

The same idea also applies to IN_{out}^ε. We again refactor the formula to obtain a precomputed lookup table $\text{pre}_{out}^\varepsilon(m) = \sum_{j=1}^{N}\frac{c_j(m)}{|c_j|}$ which is shared between all contours. Computing $IN_{out}^\varepsilon(c_i)$ and $IN_{in}^\varepsilon(c_i)$ now takes $\mathcal{O}(M)$ time with a precomputation of $\mathcal{O}(MN)$ to create the

lookup tables. This results in $\mathcal{O}(MN)$ time complexity to compute eID for all N contours in the ensemble.

$$
\begin{aligned}
\text{IN}_{out}^{\varepsilon}(c_i) &= \sum_{j=1}^{N} 1 - \frac{|c_j - c_i|}{|c_j|} \\
&= N - \sum_{j=1}^{N} \frac{|c_i - c_j|}{|c_j|} \\
&= N - \sum_{j=1}^{N} \sum_{m}^{M} \frac{(1-c_i(m))c_j(m)}{|c_j|} \\
&= N - \sum_{m}^{M} (1-c_i(m)) \sum_{j=1}^{N} \frac{c_j(m)}{|c_j|} \\
&= N - \sum_{m}^{M} (1-c_i(m)) \text{pre}_{out}^{\varepsilon}(m)
\end{aligned}
\tag{7}
$$

5. Fast Depth Recomputation

a) Inclusion matrix   **b) Epsilon inclusion matrix**

$\square \, in(c_i) \subset in(c_j)$ $\square \, in(c_i) \not\subset in(c_j)$ $in(c_i) \subset_\epsilon in(c_j)$ $0 \,\square\!\square\!\square\!\square\!\square\, 1$
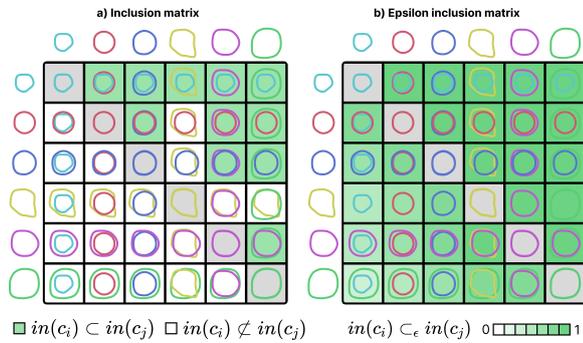
Figure 2: *Inclusion (a) and epsilon inclusion (b) matrices of the contour ensemble in Fig. 1. In the strict inclusion matrix, cells are colored if a row contour is a subset of the column contour. The epsilon inclusion matrix values range between 0 and 1, discretized into seven bins for visualization simplicity.*

In the following, we introduce the inclusion matrix, which permits decoupling the depth computation from the assessment of the pairwise inclusion relationship between contours. We show how, in practice, this translates to a significant speedup in the computation of ID and CBD ($J = 2$) on an ensemble's subsets, a feature critical for use cases that require depth evaluations within the ensemble, like clustering.

At the hearts of ID and CBD are the subset and epsilon subset operators, which permits establishing the containment relationship between all pairs of contours in the ensemble. We term the matrix that collects all the pairwise comparisons inclusion matrix $\mathfrak{C}$ and epsilon inclusion matrix $e\mathfrak{C}$, respectively. Starting with the latter, a cell $e\mathfrak{C}_{ij}$ with $i,j \in N$ is computed as:

$$
e\mathfrak{C}_{ij} = in(c_i) \subset_\varepsilon in(c_j)
\tag{8}
$$

where $\subset_\varepsilon$ is the operator defined in Eq. 4. To obtain $\mathfrak{C}$, it suffices to threshold $e\mathfrak{C}$ as

$$
\mathfrak{C}_{ij} = \mathbb{1}_{\geq 1}[e\mathfrak{C}_{ij}],
\tag{9}
$$

where $\mathbb{1}[\cdot]$ is the indicator function.

Fig. 2 depicts $\mathfrak{C}$ and $e\mathfrak{C}$ for an ensemble of six contours. The epsilon inclusion matrix (b) has values that range between 0 and 1, with one denoting full containment. In practice, entries are only zero if the two contours are disconnected components. If this is not the case and $A \not\subset_\varepsilon B$ in Eq. 4, then the entry will be lower than one but not zero, systematically increasing the depth scores, but preventing ties due to the non-perfect nestedness of contours. In general, the inclusion matrices are not symmetric. For example, $\mathfrak{C}$ is not symmetric as for $i \neq j$, if $in(c_i) \subset in(c_j)$ then $in(c_j) \not\subset in(c_i)$. It is also not antisymmetric because $in(c_i), in(c_j)$ might not share a containment relationship like in the case where they are disconnected components.

The inclusion matrix provides the information needed to compute CBD when only bands formed by two contours are considered. Therefore, in the particular case of CBD with $J = 2$, it is possible to obtain a quadratic runtime. It is possible to determine the number of bands a function falls in by calculating the number of functions above (N_a) and below (N_b) that function, and using the formula $N_{bands} = N_a N_b + N - 1$ [SGN12]. This simplification works because of the assumption that a function cannot fall in a band formed by functions that cross over [LP08]. In the contour case, by setting $N_a = \text{IN}_{out}$ and $N_b = \text{IN}_{in}$, both of which can be obtained from the inclusion matrix, it is possible to obtain CBD in $\mathcal{O}(MN^2)$, the time it takes to compute the inclusion matrix. It must be noted that this strategy does not apply to eCBD because eCBD requires operating on the full contours-vs-bands matrix.

The inclusion matrix decouples the initial computation of the pairwise inclusion relationships from the depth calculations. Therefore, adding or removing small subsets of contours is fast. Adding N' new contours to the ensemble grows the inclusion matrix from N^2 to $(N + N')^2$ entries. Adding these $2NN' + N'^2$ new entries takes $\mathcal{O}(MNN' + MN'^2)$ time, significantly faster than recomputing the matrix from scratch. In the next section, we will show how this feature enables CDclust. Additionally, in the experiments section, we show how it can be used to progressively compute depth.

6. Multi-Modal Analysis

6.1. Relative Depth

Relative Depth (ReD) is an extension of the concept of depth to multiple clusters or modes of variation. Intuitively, a contour belongs to the correct partition if the contour's depth in the partition it belongs to is higher than what it could attain if it belonged to any other partition. In the following, we refer to the former as depth-within and to the latter as depth-between.

Let I_K be a partitioning of the N contours into K clusters. $I_K(k)$ yields the ids of the contours belonging to partition k. Given a contour $c_i \in C$ with $i \in I_K(k)$, we compute its relative depth ReD_i as

$$
\text{ReD}_i = \text{ReD}(c_i|C,I_K) = D_i^w - D_i^b
\tag{10}
$$

with the depth-within defined as

$$
D_i^w = D(c_i|\{c_j|j \in I_K(k)\}),
\tag{11}
$$

and depth-between as

$$
D_i^b = \max_{l \neq k; l \in \{1,\dots,K\}} D(c_i|I_K(l)),
\tag{12}
$$

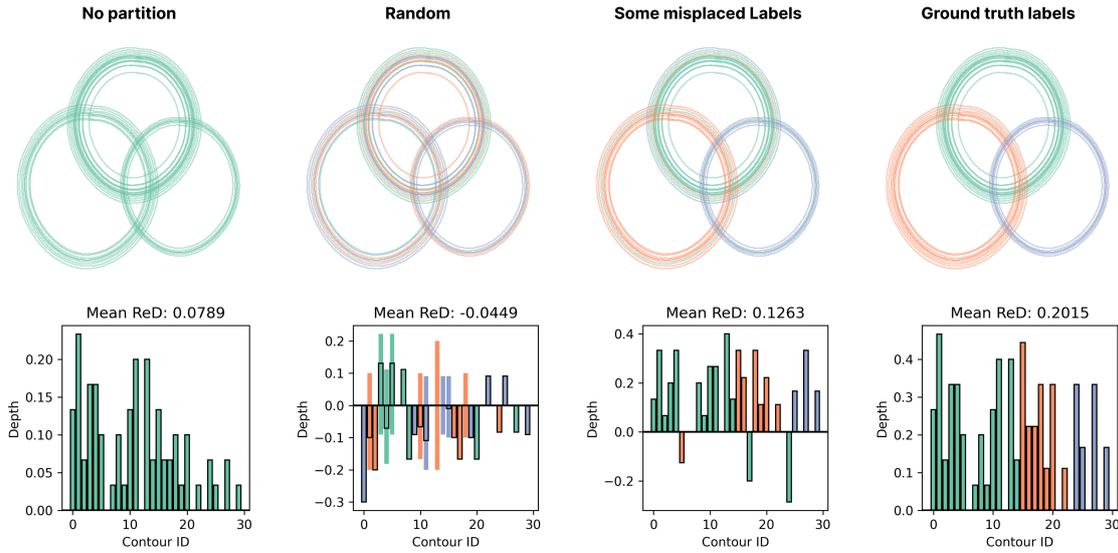where D is any suitable contour depth notion like Inclusion Depth or

Figure 3: *Relative depth scores as a function of the clustering labels for an ensemble of $N = 30$ contours in a three-ring configuration. Each ring has a different proportion of contours. The top row illustrates the different label assignments. The bottom row depicts the depth-within cluster (bar above 0 line), depth-between cluster (bar below 0 line), and relative depth (bar with black stroke and no fill) for each ensemble member.*

Contour Band Depth. ReD values range between [-1, 1]. A contour that attains the minimum value in this range is likely assigned to the wrong partition or corresponds to an outlier because its D^w is zero and its D^b is the maximum value. In contrast, a contour with the maximum value of the range is considered the median of the partition it belongs to.

Fig. 3 depicts the ReD (using ID) per contour for different partitionings of an ($N = 30$) ensemble of contours made of overlapping rings spawned in different locations with perturbed radius. The first row shows the ensemble and its partitioning with each partition colored differently. The second row depicts the D_i^w (colored bar above zero line), D_i^b (mirrored colored bar below zero line), and ReD_i (non-colored bar with black stroke) per contour (horizontal axis). The first column represents the unimodal case in which calculating the ReD reduces to computing the depth-within of each ensemble member. The other three columns show a random partitioning, a partitioning in which only some labels were exchanged, and the generative ground truth labels. It can be observed how the average ReD is maximized by the partitioning with generative labels because there are no contours with non-zero D_i^b.

Interestingly, the average ReD in the case with ground truth labels is also larger than in the uni-modal case, despite the latter not having contours with positive depth-between. This shows how the incorrect uni-modal assumption of the traditional depth notion negatively affects overall depth scores. In the experiments, we leverage this observation to show how ReD can be used as a cluster validation tool to determine the optimal number of clusters K.

6.2. CDclust

The average ReD score of a partitioning I_K provides an indication of its quality. Specifically, we say that I_K is satisfactory if the average ReD is

maximized, which entails maximizing the depth-within and minimizing the depth-between of every contour. The problem of obtaining the I_K that maximizes ReD can be formulated as

$$I_K = \underset{I_K}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} \operatorname{ReD}(c_i | C, I_K) = \frac{1}{N} \sum_{i=1}^{N} D_i^w - D_i^b, \quad (13)$$

where C is fixed.

The optimization problem in Eq. 13 has a large discrete search space. We adopt a heuristic inspired by KMeans [Scu10] to obtain a reasonable solution. Algorithm 1 presents the pseudocode of CDclust. CDclust takes as input the contour ensemble C, the desired number of components K, random trials T, and iterations it_{max}. In practice, there are potentially many local optima. Additionally, in some cases, a cluster might become empty. To ensure a better exploration of the solution space, we permit the user to define a number of random trials to perform.

Starting from a random partitioning, CDclust proceeds to iteratively increase the partitioning depth by reassigning contours to the cluster that represents them best. Specifically, at each iteration, the algorithm computes the contours' depth with respect to the other clusters and collects these depth values in the matrix $D_K \in \mathbb{R}^{N \times K}$. We define the competing cluster of a contour as the cluster that maximizes its depth

$$I_{comp} = \underset{l \in \{1,...,K\}}{\operatorname{argmax}} D(c_i | I_K(l)). \quad (14)$$

If the current assignment $I_K(c_i)$ maximizes the contours' depth, then it is not relocated. Otherwise, the algorithm reassigns to its competing cluster.

Algorithm 1 Depth-Based Contour Clustering (CDclust)

Require: C, K, T, it_{max} ▷ N-contour ensemble,
    number of components, number of random trials and of iterations

1: $I_K^* \leftarrow \{\}$ ▷ Best partition
2: $\mu ReD^* \leftarrow -\infty$ ▷ Best average ReD
3: **for** $t \in \{1,...,T\}$ **do**
4:    $I_K \leftarrow$ random partitioning of C into K clusters
5:    **for** $i \in \{1,...,it_{max}\}$ **do**
6:      $D^K \in \mathbb{R}^{N \times K}$ ▷ Between-cluster depth matrix
7:      **for** $k \in 1,...,K$ **do**
8:        $D_{\cdot,k}^K \leftarrow \{D(c_i|C_k)|c_i \in C\}$ ▷ Via inclusion matrix
9:      **end for**
10:     $D^w \leftarrow \{D_{i,k}^K | k = I_K(c_i) \text{ and } i = \{1,...,N\}\}$
11:     $D^b \leftarrow \{D_{i,l_i}^K | l_i = \text{argmax}_{l_i \neq I_K(c_i)} D_{i,l_i}^K \text{ and } i = \{1,...,N\}\}$
12:     $I_K' \leftarrow I_K$
13:     $I_K \leftarrow \{\text{argmax}_k D_{i,k}^K | i \in \{1,...,N\}\}$
14:     $\mu ReD \leftarrow \frac{1}{N} \sum_i D_i^w - D_i^b$
15:     **if** $\mu ReD > \mu ReD^*$ **then**
16:       $I^{K_*} \leftarrow I_K$
17:       $ReD^* \leftarrow ReD$
18:     **end if**
19:     **if** $I_K = I_K'$ **then**
20:       **return** I_K
21:     **end if**
22:    **end for**
23: **end for**
24: **return** I_K

6.3. CDclust Complexity

CDclust's runtime depends on the number of trials T and a maximum number of iterations it_{max}. Within each iteration, CDclust requires computing the depth of each contour with respect to each cluster. If the inclusion matrix is used, then its precomputation is the bottleneck of the algorithm taking $\mathcal{O}(MN^2)$ time. Within the loop, it takes $\mathcal{O}(N)$ time per contour to compute its depth with respect to all clusters, yielding a complexity of $\mathcal{O}(N^2)$. Therefore, in this case, CDclusts complexity is $\mathcal{O}(MN^2 + it_{max}TN^2)$.

When using the linear time eID, one needs to compute the inclusion fields at each iteration, which takes $\mathcal{O}(MN)$ time. The most expensive part of the algorithm is the computation of the between-cluster depth matrix, which takes $\mathcal{O}(KN)$ time. In total, CDclust with linear eID runs in $\mathcal{O}(it_{max}TMN + it_{max}TKN)$. Note that when a high resolution grid in the plane is used to resolve the contours, MN may be larger than N^2. In this case, CDclust with the linear time eID has slower iterations than CDclust with the inclusion matrix.

7. Experiments on Synthetic Data

This section presents the results of experiments with synthetic datasets, demonstrating the performance of the proposed methods. The experimental code (https://graphics.tudelft.nl/paper-multimodal-contour-depth) and contour-depth Python package (https://graphics.tudelft.nl/contour-depth) are available as GitHub repositories. Further speedups can be achieved by using a more performant programming language and implementing parallelism in the code. We ran all the experiments on a Mac Book Pro (2022) with an M1 Pro processor (without GPU acceleration) and 32 GB RAM.

7.1. Fast Computation of Contour Depth

Setup We use the shape outside outliers detailed in the Inclusion Depth paper [CdPMS*24]. We define a stochastic model from which we can sample inlier shapes and outlier shapes with higher amplitude and frequency, endowing them with distinct shapes. We use an outlier contamination proportion of 0.1. The second column of Fig. 7 shows an example of the shape outlier dataset.

To generate datasets of varying sizes, we start with the full ensemble ($N = 300$) and sample increasingly smaller -nested- subsets in increments of 10 until 10 elements remain, yielding sampling sizes $Ns = \{10, 20, 30, ..., 300\}$. For the unoptimized CBD in the scaling behavior experiment, we only consider until $N = 150$ due to its steep increase of computational cost. For each combination of method/sample size, we run five random trials to derive confidence intervals of the results. Finally, for the progressive depth calculation experiment, we use $N = 150$. To increase difficulty, we shuffle the shapes in the ensemble, interleaving inliers and outliers.

Scaling Behavior Fig. 4 compares the runtimes of the linear eID computation with other contour depth methods. In particular, the figure includes strict CBD ($J = 2$) and ID. We differentiate whether the method was optimized or not. Optimized CBD refers to computing strict CBD using the expression presented in Sec. 5, ID has no optimized version and unoptimized eID refers to using the inclusion matrix to compute the depths. The performance gains are evident. ID and unoptimized eID are at least an order of magnitude faster than CBD when more than two contours are used to form the band. Linear eID is, in turn, an order of magnitude faster than methods based on the inclusion matrix, computing depths of 300-contour ensembles in under ten seconds. These results confirm the speed-ups that ID and linear eID achieve. It is important to note that speed is only one factor to consider when selecting a depth notion. In practice, the properties of strict depth notions might be desired. In this case, the best-performing methods, optimized CBD ($J = 2$) and ID, have a time complexity of $\mathcal{O}(MN^2)$. In the case of CBD, if more bands are desired, the performance of the methods will rapidly degrade as it depends on the number of possible bands that can be formed out of J contours.

Progressive Depth Computation We now demonstrate the usage of fast depth computation for progressively calculating and rendering depths, which can enhance analytical processes [SPG14]. Fig. 5 compares the runtimes of the batched and progressive depth computation of a $N = 100$ ensemble. We assume that the ensemble's contours become available one at a time. For the batched method, we recompute the ensemble's ID every time a new contour arrives. For the progressive method, we only compute missing entries of the inclusion matrix and then perform a depth update of the ensemble. As can be observed in the line plot, the cost of adding a contour to the ensemble is significantly higher for the batched version. The $N = 100$ ensemble takes an average of 57 seconds per contour with the first one taking a fraction of a second and the last one more than four minutes. In contrast, the progressive version takes advantage of the information contained in the inclusion matrix to avoid unnecessary recomputations. It takes 1.15 seconds on
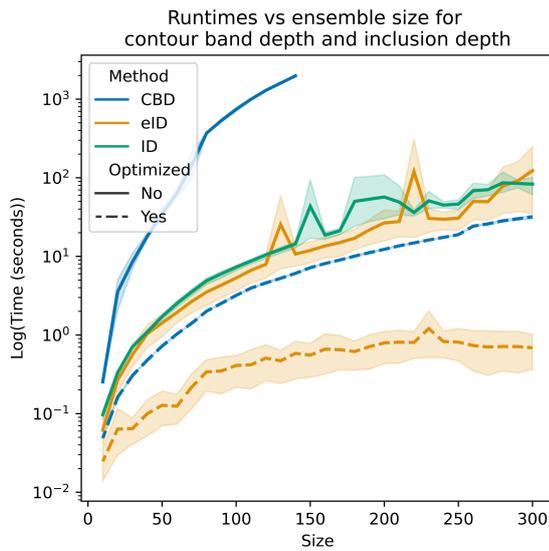
Figure 4: *Comparison of mean runtimes for different sample sizes of optimized and unoptimized versions of CBD, ID, and eID. The y-axis uses a logarithmic scale and the shaded area denotes the 95 percent confidence interval across replications.*

average to recompute the ensemble's depths, which means that the whole ensemble can be progressively rendered in less than two minutes, allowing for interactive rates. The vertical stripe on the right side of the figure illustrates how the incremental calculation of depth works.
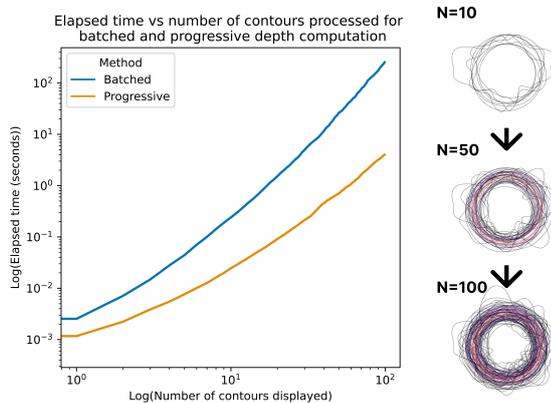

Figure 5: *Comparison of the time it takes to compute depths of a growing ensemble ($N = 100$) using batched and progressive depth calculation. The x and y-axes have log scales. The x-axis indicates how many contours have been processed at the time given by the corresponding point in the y-axis. The strip to the right depicts the updating of the depth scores as the ensemble grows.*

7.2. Multi-Modal Contour Analysis

Setup We use three datasets ($N = 100$) that contain multiple modes of variation. First, the three rings dataset has three overlapping groups of circles each with perturbed radii and centers. Each circle group has a different number of circles and spread (different radii distribution). Second, the non-nested cluster dataset contains three groups of circles C1, C2, and C3 arranged such that C1 and C2, and C1 and C3 are nested but C2 and C3 are not. Circles in each group have perturbed radii and centers and different spreads. Finally, we reuse the shape outlier dataset from the last subsection, which can be thought of as an ensemble with two modes of variation: inliers and outliers. Figs. 6 and 7 illustrate these datasets.

In preliminary experiments, we observed that the performance of CDclust decreases when using ID due to the method's tendency to yield ties if the contours intersect. Therefore, unless mentioned otherwise, we use eID as a depth notion for both ReD and CDclust. For CDclust, we use $T = 5$ and $it_{max} = 10$. The number of clusters K changes depending on the experiment's purpose.

We compare CDclust against two relevant existing methods that leverage a PCA-reduced SDF representation of the contours. To obtain a contour's SDF representation, we compute the signed distance of each pixel to the closest point on the contour and use principal components analysis to keep the dimensions in the resulting field that explain 0.999 of the variance [FKRW16]. First, we consider KMeans [Scu10], which iteratively improves the clustering by assigning points to the closest center. Similarly to CDclust, we set the number of attempts to 5 and the maximum number of iterations to 10. Second, we consider agglomerative hierarchical clustering (AHC) with average linking, which is part of the CVP pipeline proposed in [FBW16, FKRW16]. We choose the number of clusters to match the one used in CDclust and KMeans. For both clustering algorithms, we use Sklearn's implementation with Euclidean distance as the distance metric.

Cluster Validation Using ReD Average ReD (μReD) can be used to determine the optimal number of clusters. Fig. 6 depicts this cluster validation strategy for the three rings dataset ($N = 100$). For different values of K, we run CDclust and compute the clustering μReD. To reduce the sensitivity to a specific clustering result, we perform this process ten times, varying CDclust's random seed. The graph (top row of Fig. 6) shows the mean μReD per K surrounded by a 95% confidence interval. It can be observed how $K = 3$, the desired clustering, consistently maximizes μReD. As K increases, the mean μReD decreases and the uncertainty in the clustering results increases. The figure's bottom section shows the resulting clusterings for one of the random seeds. As can be observed, in some cases higher K clusterings preserve the inlier structure of $K = 3$, assigning magnitude outliers to the extra clusters. The depth-within of the swapped contours does not change because of their outlier status, but their depth-between increases, which results only in a slight decrease in μReD. In other cases, a ring group is split into two or more components, reminiscing clusterings obtained with hierarchical methods.

Comparative Evaluation of CDclust In the synthetic datasets we considered, we observed that ReD, KMeans, and AHC exhibited a similar clustering behavior when using the ground truth K. The methods' behavior changed when exploring alternative Ks. The first column of Fig 7 presents an example of the non-nested cluster dataset. We clustered the dataset with $K = 2$. Both CDclust and AHC put the small group of contours (in orange for CDclust and AHC) in a separate cluster.
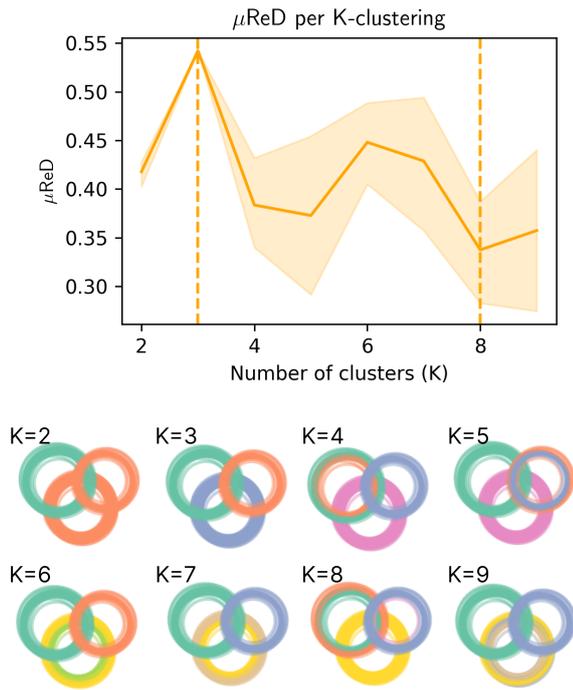
Figure 6: *Selection of the optimal number of clusters using μReD. The line plot depicts the mean μReD per K-clustering across ten samples. The shaded area corresponds to a 95% confidence interval. K = 3 and K = 8 (vertical dashed lines) attain the highest and lowest mean μReD, respectively. The bottom section presents examples of the resulting clusterings for one of the samples.*

Figure 7: *Comparison of clustering results of CDclust, KMeans, AHC and the reference labels for the non-nested cluster and shape outlier datasets with K = 2.*

In contrast, KMeans classified these contours as belonging to the same cluster as the inner ones, which are partially disconnected/unnested. This example shows how CDclust has increased sensitivity to the nestedness relationship between contours, which could be useful in cases where one wants to flag groups of contours with a different nestedness relationship.

The previous result hints at the strength of contour depth in identifying shape outliers. For ID, the user must select a depth threshold for the outliers. CBD uses an automatic mechanism to determine it. We explored the utilization of clustering methods to identify the shape outliers, using the shape outliers dataset and K = 2. The second column of Fig. 7 shows an example of the results. As can be observed, CDclust assigned all the 16 shape outliers to the same group. It also assigned contours with extreme magnitudes to the group of outliers, highlighting as inliers the highly central core of circular contours. KMeans and AHC, relying on distances rather than on the inclusion relationships, do not achieve a clear separation. KMeans splits the shape outliers, assigning 11 to the green cluster and 5 to the orange one. Furthermore, in the orange cluster KMeans mixes representative contours with shape outliers. For K=2, AHC only separated one magnitude outlier from the rest, combining shape outliers and inliers. This result is sensitive to AHC's linking method and the K used. When we tried larger values for K, AHC assigned shape outliers to low cardinality clusters, producing a satisfactory separation of the inliers similar to CDclust's. Nonetheless, needing to tinker with both *K* and the linking method hinders the
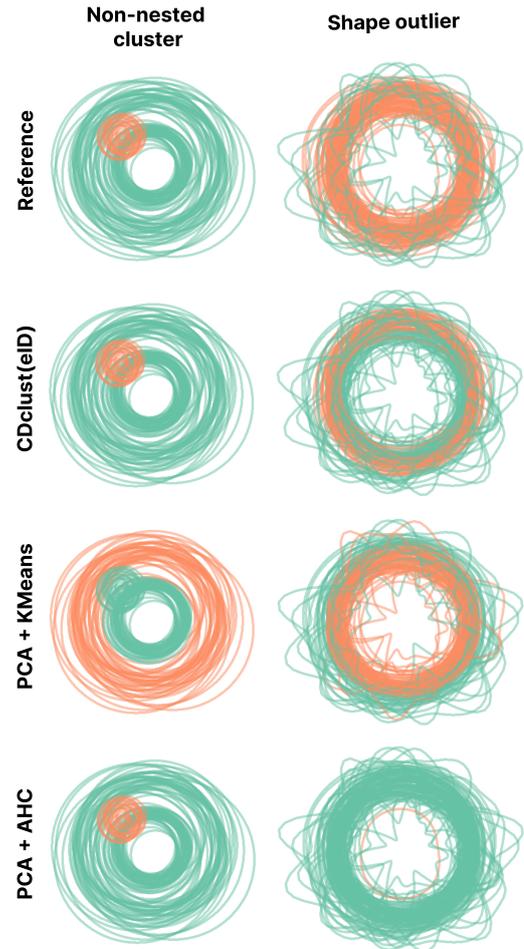
method's practicality. The results indicate that CDclust can also be used to separate outliers from inliers, permitting us to automatically obtain a robust outlier-free cluster, which can be used in downstream procedures.

8. Case Studies

In this section, we demonstrate how CDclust+ReD can be used to perform non-parametric multi-modal visual analysis of real datasets. We use eID because it is the fastest depth notion available and because contours in real data tend to intersect. For CDclust, we use the same configuration as in the previous section. For CVP, we implement the pipeline as described in [FKRW16]. In summary, CVP uses agglomerative hierarchical clustering of the PCA-reduced SDF representations of the contours to find the modes of variation. The cluster representatives are the geometric medians of each cluster in PCA space. The bands are computed from the SDFs by adding and subtracting from the mean SDF a user-selected number of standard deviations (we use one standard deviation for the results in this section).

For visualization of the results, we use spaghetti plots and contour boxplots [WMK13]. We render contour boxplots using a single hue to accommodate multiple modes of variation and consider the median (thick solid line) and the confidence bands (semi-transparent polygon with the same hue as the median line). We do not render the outliers for clarity of exposition. Nevertheless, for CDclust+ReD, we filter out the bottom ten percent of contours with the lowest depth per cluster and then compute the band with the remaining contours. We use the same computer as in the experiments with synthetic data.

8.1. Segmentation Ensembles

Data With the advent of deep learning-based auto-contouring technologies, segmentation of organs-at-risk (OARs) in radiotherapy has been largely automated [MCCC*20]. Nevertheless, clinicians still need to perform a quality assessment of the segmentations, which requires understanding the uncertainty in the predictions. We consider the computerized tomography (CT) of a patient with head and neck cancer treated at HollandPTC between 2018 and 2020. The IRB approved the research protocol for the use of patient data in research, all patients signed an informed consent form. We trained 30 segmentation models based on the popular UNet architecture [RFB15] on different subsets of the training split of the dataset of the Head and Neck Auto Segmentation MICCAI Challenge [RZS*17]. The MICCAI dataset contains CT scans of patients with head and neck cancer with ground truth segmentations of nine OARs. To further augment the ensemble size and the variability of the predictions, we trained each model using different learnable weight initializations. Using the resulting models to segment the parotid gland yields an ensemble of 120 scalar maps of per-voxel softmax probabilities. For the analysis, we focused on 540×540 pixels 2D slices of the OARs. We obtain the contours by thresholding the probabilities with an iso-value of 0.8.

Analysis Fig. 8 illustrates a depth-based multimodal analysis of a slice of the ensemble of segmentations of the right parotid gland using depths. We focused on the parotid gland because it is not always clearly visible in CT, which can increase inter-clinician variability. In these cases, a visual statistical summary can help clinicians understand the range of predictions. The spaghetti plot (a) provides an overview of 120 segmentations, revealing trends that are challenging to disentangle visually due to occlusion. Using contour boxplots based on the eIDs of the ensemble (b) simplifies data display and showcases variability in wide confidence bands. Notably, the median contour differs significantly from the outer band boundary, suggesting multiple modes of variation. To validate this hypothesis, we used CDclust with $K=2$ (max average ReD). The resulting clustering reveals a split into inner and outer sections (c). The orange cluster has more members, which explains its representative shape being selected as the median in (b). While contour boxplots improve on spaghetti plots, occlusion persists. In (d), clinicians can drill down by clicking on the cluster of interest in the vertical proportions bar, revealing that most of the teal cluster's variation is concentrated in the bottom right, where confidence bands are wider.

8.2. Weather Forecast Ensembles

Data A common use case for contour statistical models is to analyze meteorological forecast data. We consider data from the European Centre for Medium-Range Weather Forecasts (ECMWF). Specifically,
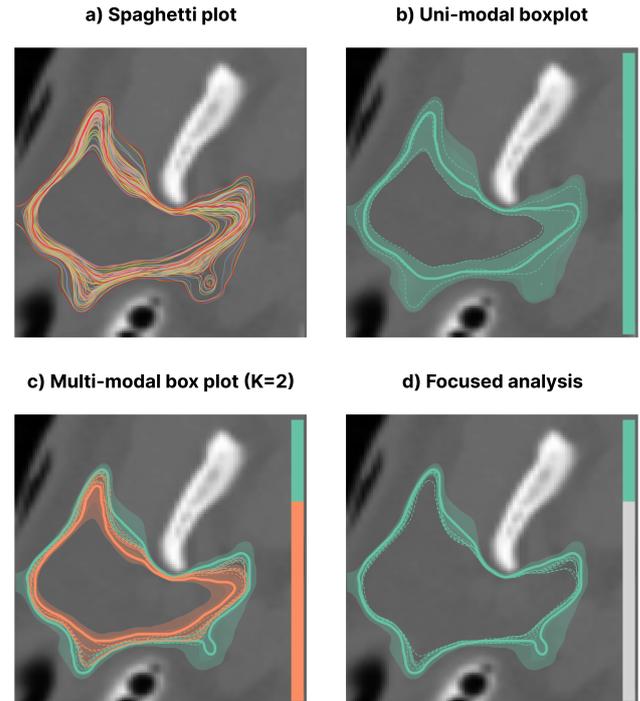

Figure 8: *Different stages of the ensemble analysis process. a) and b) present an overview of the ensemble using a spaghetti plot and a contour boxplot based on the depths of the complete ensemble. c) and d) present a multi-modal analysis of the ensemble. c) depicts an overview of the different modes of variation and d) focuses on the less representative variation mode.*

the ECMWF Ensemble Prediction System (EPS) provides ensembles of predictions for different variables like precipitation, temperature, and pressure. The forecasts include $N = 50$ perturbed members and a control run. We analyze the same data as in [FKRW16], which is the forecast from 00:00 UTC 15 October 2012. More details about this type of data can be found in [LP08]. The region under consideration encompasses $101 \times 41 \times 62$ grid points, which corresponds to latitude, longitude, and geopotential height dimensions. For the analysis, we consider 2D fields, corresponding slices of the region where the geopotential height is $500 hPa$. To obtain contours from this field, we threshold them using an iso-value of $5600m$. The spaghetti plot in Fig. 9 depicts the extracted contours laid over the geographical region they span.

Analysis Fig 9 (b) shows the results of utilizing CVP to analyze the forecast ensemble [FKRW16]. The majority of the ensemble's members belong to the purple (25) and orange (23) clusters. and the geometric medians (solid lines) are similar in shape, with the orange one exhibiting more pronounced curves towards the middle of the map. The green cluster contains the fewest members (3), and its shape differs from the other two, especially at the left of the map. When performing non-parametric analysis with CDclust (c), one can observe trends similar to CVP's. In particular, the proportions (24, 17, and 10 members) remain similar, and the shapes of the representatives too. This shows that both clustering procedures identified similar trends in the data. The two methods mainly
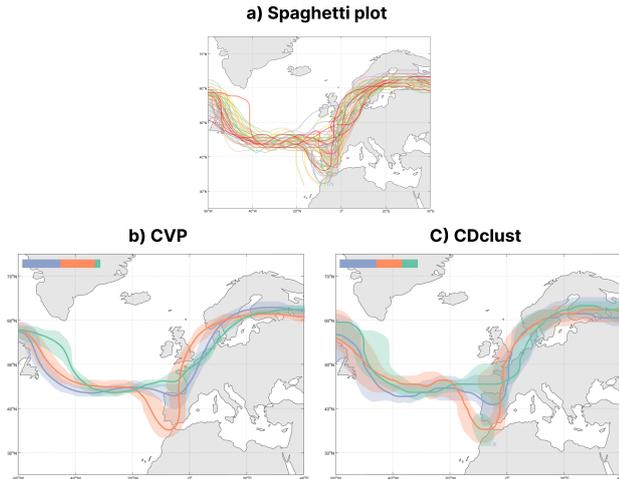
Chaves-de-Plaza et al. / Depth for Multi-Modal Contour Ensembles

a) Spaghetti plot

b) CVP   **C) CDclust**

Figure 9: *Comparison of parametric (b) and non-parametric (c) analysis of the ensemble of 500 hPa geopotential contour lines (ECMWF ENS forecast from 00:00 UTC, 15 October 2012 valid at 00:00 UTC, 20 October 2020). (a) presents an overview of the ensemble using a spaghetti plot. The horizontal colored bar in (b) and (c) encodes the cluster's proportions in decreasing order.*

differ in the bands' shapes and the representatives' smoothness. The depth-based bands are generally thicker, and the trajectories of the representatives are more distinct because they are made from inlier contours in the ensemble. In contrast, CVP synthesizes bands and representatives, producing smoother graphical elements. A clear visual difference that arose in this case study is the blob in CDClust's green cluster. Both methods use a threshold to define the bands' extents: unit standard deviation for CVP and keeping the top 90% contours depth-wise for CDclust. The blob arises because two members of the CDclust's green cluster (which agree with CVP) contain such a feature, but only one was flagged as an outlier and removed. This difference highlights the importance of trying different values for the threshold parameters of both methods. Finally, our results reinforce that, in practice, analysts can benefit from considering parametric and non-parametric analysis [ZLC*23].

9. Discussion and Conclusions

Contour depth has gained prominence in non-parametric analysis across domains such as meteorological forecasting and medical image segmentation [WMK13, MW18, CdPMS*24]. The efficacy of contour depth methods hinges on their scalability with increasing ensemble size. Our contributions significantly enhance existing methods by introducing a linear time algorithm for Epsilon Inclusion Depth (eID) computation. Furthermore, we introduce an inclusion matrix, facilitating depth computation on ensemble subsets without reevaluating the inclusion relationship, a process dependent on domain resolution. These accelerated depth computation methods find applications in progressive depth computation [SPG14] and interactive depth updating.

We also generalize contour depth using relative depth and introduce CDclust to address the assumption of contours drawn from the same distribution. To our knowledge, CDclust is the first depth-based contour clustering algorithm. Experiments on synthetic data demonstrate that

CDclust largely agrees with KMeans and Agglomerative Hierarchical Clustering, but exhibits sensitivity to clusters violating the nestedness relationship. The desirability of this property depends on the application. We further demonstrated CDclust's practical utility by analyzing ensembles arising from two domains. In medical image segmentation, we showcase how clinicians can disentangle trends through multi-modal analysis. This positions contour depth methodology for interactive refinement of segmentations [WLC*22, TJL*20] based on representative selection [MW18]. Our meteorological forecasting example compares non-parametric and parametric multi-modal analyses, revealing the visualization-altering assumptions of CVP's method. Adopting both parametric and non-parametric lenses is crucial in practice [ZLC*23]. The proposed methods and the contour-depth Python library contribute to this approach.

There are several future work avenues. First, eID's formulation facilitates obtaining a linear algorithm based on precomputed maps. It is unclear whether other depth notions like contour band depth [WMK13] can profit from similar strategies. Second, the runtime of linear eID depends on the grid resolution, reducing its effectiveness in cases that require multiple evaluations. Addressing this dependency and implementing parallelism, for instance, via a GPU implementation, would increase the contour depth methodology's reach. Third, using ReD to select the optimal K showed suboptimal clusters can obtain high ReD. While alternative schemes are possible, we found running CDclust multiple times helps avoiding local optima. Finally, CDclust uses a global depth notion. Future investigations could adapt CDclust to enable local analysis [MW18] for multi-scale insights. Additionally, working directly with the scalar field from which contours arise and integrating speedups into functional depth cases are intriguing future research avenues [ME19].

References

[APH*21] ABDAR M., POURPANAH F., HUSSAIN S., REZAZADEGAN D., LIU L., GHAVAMZADEH M., FIEGUTH P., CAO X., KHOSRAVI A., ACHARYA U. R., MAKARENKOV V., NAHAVANDI S.: A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion 76* (2021), 243–297. doi:10.1016/j.inffus.2021.05.008. 2

[CdPMS*24] CHAVES-DE PLAZA N. F., MODY P., STARING M., VAN EGMOND R., VILANOVA A., HILDEBRANDT K.: Inclusion depth for contour ensembles. *IEEE Transactions on Visualization and Computer Graphics* (2024), 1–12. doi:10.1109/TVCG.2024.3350076. 1, 2, 3, 6, 10

[DDPW07] DING Y., DANG X., PENG H., WILKINS D.: Robust clustering in high dimensional data using statistical depths. *BMC Bioinformatics 8*, 7 (2007), S8. doi:10.1186/1471-2105-8-S7-S8. 3

[DJW16] DEMIR I., JAREMA M., WESTERMANN R.: Visualizing the central tendency of ensembles of shapes. SIGGRAPH ASIA 2016 Symposium on Visualization, ACM. URL: https://doi.org/10.1145/3002151.3002165, doi:10.1145/3002151.3002165. 2

[FBW16] FERSTL F., BÜRGER K., WESTERMANN R.: Streamline variability plots for characterizing the uncertainty in vector field ensembles. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 767–776. doi:10.1109/TVCG.2015.2467204. 2, 7

[FKRW16] FERSTL F., KANZLER M., RAUTENHAUS M., WESTERMANN R.: Visual analysis of spatial variability and global correlations in ensembles of iso-contours. *Computer Graphics Forum 35*, 3 (2016), 221–230. doi:10.1111/cgf.12898. 2, 7, 8, 9, 10

[HG17] HUANG X., GEL Y. R.: Crad: Clustering with robust autocuts and depth. In *2017 IEEE International Conference on Data Mining (ICDM)* (2017), pp. 925–930. doi:10.1109/ICDM.2017.116. 3

[JCSW16] JEONG M.-H., CAI Y., SULLIVAN C. J., WANG S.: Data depth based clustering analysis. In *ACM SIGSPATIAL* (2016). doi:10.1145/2996913.2996984. 3

[Jör04] JÖRNSTEN R.: Clustering and classification based on the l1 data depth. *Journal of Multivariate Analysis 90*, 1 (2004), 67–89. doi:10.1016/j.jmva.2004.02.013. 1, 2, 3

[KHS*19] KOREVAAR E. W., HABRAKEN S. J. M., SCANDURRA D., KIERKELS R. G. J., UNIPAN M., EENINK M. G. C., STEENBAKKERS R. J. H. M., PEETERS S. G., ZINDLER J. D., HOOGEMAN M., LANGENDIJK J. A.: Practical robustness evaluation in radiotherapy - a photon and proton-proof alternative to ptv-based plan evaluation. *Radiotherapy and Oncology 141* (2023/11/15 2019), 267–274. doi:10.1016/j.radonc.2019.08.005. 1

[KTB*18] KUMPF A., TOST B., BAUMGART M., RIEMER M., WESTERMANN R., RAUTENHAUS M.: Visualizing confidence in cluster-based ensemble weather forecast analyses. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (2018), 109–119. doi:10.1109/TVCG.2017.2745178. 2

[LdMMV21] LAFAYE DE MICHEAUX P., MOZHAROVSKYI P., VIMOND M.: Depth for curve data and applications. *Journal of the American Statistical Association 116*, 536 (2021), 1881–1897. doi:10.1080/01621459.2020.1745815. 3

[LP08] LEUTBECHER M., PALMER T.: Ensemble forecasting. *Journal of Computational Physics 227*, 7 (2008), 3515–3539. doi:10.1016/j.jcp.2007.02.014. 1, 4, 9

[MCCC*20] MONTAGNON E., CERNY M., CADRIN-CHÊNEVERT A., HAMILTON V., DERENNES T., ILINCA A., VANDENBROUCKE-MENU F., TURCOTTE S., KADOURY S., TANG A.: Deep learning workflow in radiology: a primer. *Insights into Imaging 11*, 1 (2020), 22. doi:10.1186/s13244-019-0832-5. 9

[ME19] MA B., ENTEZARI A.: An interactive framework for visualization of weather forecast ensembles. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (2019), 1091–1101. doi:10.1109/TVCG.2018.2864815. 2, 10

[MM22] MYERS A., MIOLANE N.: Regression-based elastic metric learning on shape spaces of cell curves. In *NeurIPS 2022 Workshop on Learning Meaningful Representations of Life* (2022). URL: https://openreview.net/forum?id=8YKd0rwc4mu. 1

[MW18] MIRZARGAR M., WHITAKER R. T.: Representative consensus from limited-size ensembles. *Computer Graphics Forum 37*, 3 (2018), 13–22. doi:10.1111/cgf.13397. 1, 10

[PB19] PATIL C., BAIDARI I.: Estimating the optimal number of clusters k in a dataset using data depth. *Data Science and Engineering 4*, 2 (2019), 132–140. doi:10.1007/s41019-019-0091-y. 3

[PD23] PANDOLFO G., D'AMBROSIO A.: Clustering directional data through depth functions. *Computational Statistics 38*, 3 (2023), 1487–1506. doi:10.1007/s00180-022-01281-w. 2, 3

[PFCB23] PADILLA L., FYGENSON R., CASTRO S. C., BERTINI E.: Multiple forecast visualizations (mfvs): Trade-offs in trust and performance in multiple covid-19 forecast visualizations. *IEEE Transactions on Visualization and Computer Graphics 29*, 1 (2023), 12–22. doi:10.1109/TVCG.2022.3209457. 2

[PH11] POTHKOW K., HEGE H.-C.: Positional uncertainty of isocontours: Condition analysis and probabilistic measures. *IEEE Transactions on Visualization and Computer Graphics 17*, 10 (2011), 1393–1406. doi:10.1109/TVCG.2010.247. 2

[PVB13] PAINDAVEINE D., VAN BEVER G.: From depth to local depth: A focus on centrality. *Journal of the American Statistical Association 108*, 503 (2013), 1105–1119. doi:10.1080/01621459.2013.813390. 2

[RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (Cham, 2015), Navab N., Hornegger J., Wells W. M., Frangi A. F., (Eds.), Springer International Publishing, pp. 234–241. doi:10.1007/978-3-319-24574-4_28. 9

[RZS*17] RAUDASCHL P. F., ZAFFINO P., SHARP G. C., SPADEA M. F., CHEN A., DAWANT B. M., ALBRECHT T., GASS T., LANGGUTH C., LÜTHI M., JUNG F., KNAPP O., WESARG S., MANNION-HAWORTH R., BOWES M., ASHMAN A., GUILLARD G., BRETT A., VINCENT G., ORBES-ARTEAGA M., CÁRDENAS-PEÑA D., CASTELLANOS-DOMINGUEZ G., AGHDASI N., LI Y., BERENS A., MOE K., HANNAFORD B., SCHUBERT R., FRITSCHER K. D.: Evaluation of segmentation methods on head and neck ct: Auto-segmentation challenge 2015. *Medical Physics 44*, 5 (2017), 2020–2036. URL: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1002/mp.12197, arXiv:https://aapm.onlinelibrary.wiley.com/doi/pdf/10.1002/mp.12197, doi:10.1002/mp.12197. 9

[Scu10] SCULLEY D.: Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web* (New York, NY, USA, 2010), WWW '10, Association for Computing Machinery, p. 1177–1178. URL: https://doi.org/10.1145/1772690.1772862, doi:10.1145/1772690.1772862. 5, 7

[SGN12] SUN Y., GENTON M. G., NYCHKA D. W.: Exact fast computation of band depth for large functional datasets: How quickly can one million curves be ranked? *Stat 1*, 1 (2012), 68–74. doi:10.1002/sta4.8. 2, 4

[SPG14] STOLPER C. D., PERER A., GOTZ D.: Progressive visual analytics: User-driven visual exploration of in-progress analytics. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1653–1662. doi:10.1109/TVCG.2014.2346574. 6, 10

[SZD*10] SANYAL J., ZHANG S., DYER J., MERCER A., AMBURN P., MOORHEAD R.: Noodles: A tool for visualization of numerical weather model ensemble uncertainty. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1421–1430. doi:10.1109/TVCG.2010.181. 2

[TJL*20] TAJBAKHSH N., JEYASEELAN L., LI Q., CHIANG J. N., WU Z., DING X.: Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation. *Medical Image Analysis 63* (2020), 101693. doi:10.1016/j.media.2020.101693. 10

[TN14] THOMAS D. M., NATARAJAN V.: Multiscale symmetry detection in scalar fields by clustering contours. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2427–2436. doi:10.1109/TVCG.2014.2346332. 2

[WHLS19] WANG J., HAZARIKA S., LI C., SHEN H.-W.: Visualization and visual analysis of ensemble data: A survey. *IEEE Transactions on Visualization and Computer Graphics 25*, 9 (2019), 2853–2872. doi:10.1109/TVCG.2018.2853721. 2

[WLC*22] WANG R., LEI T., CUI R., ZHANG B., MENG H., NANDI A. K.: Medical image segmentation using deep learning: A survey. *IET Image Processing 16*, 5 (2022), 1243–1267. doi:10.1049/ipr2.12419. 10

[WMK13] WHITAKER R. T., MIRZARGAR M., KIRBY R. M.: Contour boxplots: A method for characterizing uncertainty in feature sets from simulation ensembles. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2713–2722. doi:10.1109/TVCG.2013.143. 1, 2, 3, 9, 10

[ZLC*23] ZHANG M., LI Q., CHEN L., YUAN X., YONG J.: Enconvis: A unified framework for ensemble contour visualization. *IEEE Transactions on Visualization and Computer Graphics 29*, 4 (2023), 2067–2079. doi:10.1109/TVCG.2021.3140153. 2, 10