ARL-TR-9425 • Mar 2022

DEVCOM ARMY RESEARCH LABORATORY

Machine-Learning Groundwork and Cluster Analysis for Vortex Detection

by Prabhat Kumar, Melvin Felton, David Ligon, Sandra Collier, and Deryck James

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9425 ● MAR 2022

Machine-Learning Groundwork and Cluster Analysis for Vortex Detection

by Prabhat Kumar, Melvin Felton, David Ligon, Sandra Collier, and Deryck James
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
March 2022	Technical Report	13 October 2020– 31 May 2021

4. TITLE AND SUBTITLE
Machine-Learning Groundwork and Cluster Analysis for Vortex Detection

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Prabhat Kumar, Melvin Felton, David Ligon, Sandra Collier, and Deryck James

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLS-E
Adelphi, MD 20783

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-9425

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
Primary author's email: <prabhat.kumar.civ@army.mil>.

14. ABSTRACT
During flight, an aircraft produces trailing wingtip wake vortices as a result of the relative motion of the craft and surrounding air. To detect these vortices, Doppler LIDAR is used (both on the ground and on the craft) to measure (relative) radial wind speeds using various scan geometries. These measurements can be processed to produce information about strength, circulation, and overall dynamics of the wake vortices. In this vortex detection project, we lay a machine-learning groundwork for the goal of characterizing and tracking coherent atmospheric structures produced by the wingtips of a C-17 aircraft. This study will aid research that aims to provide essential information used to dictate flight pattern protocol for safe and efficient aircraft maneuvers and group orientations during airdrops. Two outcomes of this project are (1) the LIDAR Cluster Analysis Program (LCAP), a preliminary software using k-means and DBSCAN clustering algorithms to partition a LIDAR scan into regions of interest, and (2) a methodology to isolate the vortex regions (with little noise) from the remainder of the LIDAR scan.

15. SUBJECT TERMS
wake vortex, Doppler, LIDAR, machine learning, cluster analysis, DBSCAN, k-means, vortex detection, vortex isolation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Prabhat Kumar
Unclassified	Unclassified	Unclassified	UU	65	19b. TELEPHONE NUMBER (Include area code) 301-394-2667

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

Acknowledgments

This document serves as the final technical report for the author's apprenticeship rotation with the Propagation Team in the Battlefield Environments Division under the Computational and Information Sciences Directorate at the US Army Combat Capabilities Development Command Army Research Laboratory. I would like to thank Dr Sandra Collier, Mr Melvin Felton, and Dr David Ligon for their support and encouragement during my rotation.

1. Introduction

1.1 Background and Motivation

During flight, an aircraft produces trailing wingtip wake vortices as a result of the relative motion of the craft and surrounding air, namely when lift is produced. To summarize, from Green,[1] the lifting surface of the aircraft creates a pressure difference with higher pressure on the bottom of the wing and lower pressure on top. This pressure difference accelerates air from the bottom of the wing, around the wingtip, toward the top of the wing and the resulting circulation of air on either side of the craft are the wingtip wake vortices. These vortices present a physical safety concern for objects in their vicinity. A large amount of wake vortex research is dedicated to providing safety guidelines at airports, for example in aircraft spacing. During takeoff or landing, a leading aircraft leaves behind two counter-rotating wake vortices which can prove detrimental to a following craft. Giuni[2] mentions, "Imposed roll, loss of altitude and strong structural loads are some of the dangers that the following aeroplane will suffer." There has been much interest in determining proper flight safety protocol, for example, Giuni goes on to say, "To avoid such wake encounters, regulations require aircraft to maintain set distances behind each other and set time intervals between landings and take-offs... The goal of research on this aspect is to be able to create real-time automated systems that could measure and predict wake vortex conditions at airports so that planes may fly closer, airport congestion may be reduced and runway service rate increased." See Perry et al.,[3] Robins et al.,[4] Cariou and Thobois,[5] and Hallock and Holzapfel[6] for more on wake vortex research as it pertains to airport flight safety protocols. The latter study compiles wake vortex research for airport safety from a 27-year period between 1991 and 2018. Hallock describes what is known and still under investigation about the behavior of vortices in terms of its ground motion, lateral/vertical motion, decay processes, as well as modeling efforts via computational fluid dynamics (CFD) and fast-time predictive systems. Hallock spends time covering the progress of CFD, which he notes has made great headway, but variations in its results are still a bit too great for comfort. Proper research on aircraft spacing during take-off and landing is one way to improve general efficiency, but spacing during flight for multiple aircraft flying in formation may further improve performance, especially in the case of military aerial operations.

Lovell[7] considers potential benefits: “[r]ather than avoiding wake vortices it is actually possible for following air vehicles to make use of them to increase performance. Birds fly in V formation so that the upwash produced outboard of the wing tip vortices of the leading member can be used to reduce the lift that it is necessary for the following members in the V to generate." Ray et al.[8] describe the Autonomous Formation Flight (AFF) by the NASA Dryden Flight Research Center. The study involves two F/A-18 Hornets, one of which flies in the wake vortex of the leading craft, and finds an 18% reduction in fuel flow and 20% reduction in drag for the following craft. Following this, in a lecture given at MIT, Larson and Schkolnik[9] notes the potential benefits of AFF given by “commercial fuel savings of $0.5 to 1 million per year per trailing aircraft and application to UAV swarming and aerial refueling." Ning[10] echoes these findings while presenting a new, safer extended formation of aircraft involving expanding the spacing by five wingspans, which result in about 30% drag savings in a 2-aircraft formation and 40% drag reduction in a 3-aircraft formation.

Ning[10] mentions close spacing also involves a high risk of collision, (hence, the reason for his study on extended formation flight.) It is also easy to see how wake vortices could wreak havoc during military aerial operations such as airdrops. Lovell[7] mentions “[o]f more importance to military operations is the fact that the reduced pressure near the core of wing tip vortices may lead to the condensation of water particles to form visible contrails, with a consequent reduction in survivability in hostile airspace." Another negative consequence Lovell mentions is “the vortices generated by a C-130 [have a] potential effect on the air drop environment if the aircraft departs from unyawed flight." We argue that these effects should be mitigated during planning and execution. So, how do we begin addressing these large and invisible atmospheric structures?

To detect these vortices, ground and onboard Doppler LIDAR is used to measure (relative) radial wind speeds using various scan geometries. A Doppler LIDAR system works by firing laser light into the atmosphere and detecting light which is backscattered off the pervading aerosol particles at a certain distance away. Depending on the relative motion between the LIDAR system and the aerosol particles, the backscattered light will display a change in frequency which can be detected and processed to compute the velocity of the particle along the direction of the incident beam. Taking many measurements allows for the system to ascertain an average

radial wind speed at that point. We can adjust the distance the beam travels (time of flight) and the detection angles to sweep out an area or volume of the atmosphere that we want to know more about. The shape of the beam sweep determines the scan geometry. For example, a range-height indicator (RHI) scan is one where we hold the azimuthal (longitudinal) angle fixed, but vary the elevation (latitude) angle. In this project, we primarily address the data gathered using this type of scan geometry. For a more in-depth primer on using Doppler LIDAR for atmospheric research please see Clifton et al.[11]

The goal then becomes to characterize the vortex regions and/or the background atmosphere. By characterization, we mean describing the core positions, circulation strength, downwash, core separation and vortex decay, for example, as in the case of Köpp et al.[12] Some algorithms such as those used in determining information about circulation require high accuracy in determining vortex core location as noted in Wu et al.[13] An onboard LIDAR system is used in Michel et al.[14]; they use Monte Carlo simulations to determine vortex center locations to within ± 0.5 m, and use GPS data in tandem with LIDAR data to determine the vortex falling velocity to infer their circulation. Toward the goal of determining vortex center locations and tracking these structures through space and time, our study is focused on a methodology to isolate the vortex region in a Doppler LIDAR RHI scan.

1.2 Machine Learning with Atmospheric Doppler LIDAR Data

In recent years, there has been plenty of investment in research for machine learning as we uncover its ever-growing potential. The reason for this is exponentially more computational power allowing for the expanded efforts toward data curation and research into powerful algorithms for seemingly intractable computational problems. Machine learning may be broken into two major categories: supervised and unsupervised learning. In essence, a supervised learning approach includes ground-truth values with which the algorithm's results are compared, and an unsupervised approach does not include ground-truth values. In this report, we mainly focus on the latter. For more in-depth discussions on machine-learning approaches, please see Hastie et al.[15] and Goodfellow et al.[16]

Machine learning in the context of analyzing Doppler LIDAR data seems to be in its infancy. One of the most basic tasks would be classifying various scans as containing turbulent structures or not. In 2019, Pan et al.,[17] use a You Only Look Once

(YOLO) convolutional neural network (CNN) architecture (which is developed in Redmon et al.[18]) to classify Doppler LIDAR data as either containing a wake vortex or not; the resulting accuracy was 94.46% with a recall rate of 91.27%. A year later Pan et al.[19] uses a support vector machine (SVM) for the same classification task, likely on a different data set than their SVM study, and compare it using a k-nearest neighbors (kNN) algorithm. Using SVM results in an accuracy of 0.78 with recall rate 0.63, and kNN results in 0.77 accuracy and 0.63 recall rate. Wang et al.[20] use a combination of CNN and an extreme learning machine (ELM) for their vortex identification tasks on simulated 2-D flow-fields. Compared to others methods like Q-criterion, λ-criterion, AdaBoost, and CNN only, the proposed algorithm does not perform well in categories of accuracy and computational time individually, but taking both categories into account shows it to be the superior method. Cheliotis et al.[21] trains a quadratic discriminant analysis algorithm on Doppler LIDAR scans to classify three different types of turbulent structures, namely unaligned thermals, rolls, and streaks. Liu et al.[22] uses chaotic oscillatory neural networks (developed in Aihara et al.[23]) on Doppler LIDAR data in their attempt to forecast mesoscale wind shears and turbulence at the Hong Kong International Airport.

A subset of unsupervised learning research is dedicated to cluster analysis, where the goal is to uncover hidden structures in the data set by grouping similar data points into clusters. In a clustered data set, points in any cluster are assumed to be dissimilar enough from those in another that each would be labeled differently. One way to think about this is that we want to minimize intracluster distances and maximize intercluster distances. We note that “(dis)similarity" is dependent on the chosen metric/distance measure. The Euclidean metric may produce different results than the taxicab metric, and still other measures of similarity exist which may not necessarily have to do with spatial relationship. While there may be numerous concepts of clusters, we focus on those which are the basis for two different clustering algorithms, k-means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The cluster concept forming the basis for the former uses the idea of convex sets; the clusters are assumed to be more globular (see Bhargav and Pawar[24]). For the latter, as the name suggests, whether or not a grouping of points is considered a cluster is dependent on the density of those points, see Ester et al.[25] These clustering algorithms will be developed later. There has not been much found in the way of clustering being used on Doppler LIDAR data, except in research involving object detection for autonomous vehicles, and even less so

for atmospheric research. One interesting study, though, by Maulik et al.[26] uses a k-means algorithm on Doppler LIDAR data to recognize wake variability measurement in wind turbines for improving wind turbine layout and controls.

2. Machine Learning: A Proposed Framework for Vortex Detection

Machine learning has overtaken much of the automation processes of data science tasks and in this project it is used in much the same way. The main idea is to learn aspects of a data set which may not so readily reveal themselves through the stochastic approach that has been in place up to this point.

Over the course of the project, members have compiled and curated a large data set of RHI Doppler LIDAR scans of wingtip vortices. The scans are of the vortex cross sections; that is, they are taken in the plane perpendicular to the flight path. The data consists of at least 1500 raw data files, a subset of which have been pre-processed into well-organized MATLAB structures, which we refer to as **wtv structures**, consisting of substructures which house relevant information to determine the characteristics of the vortices (distance of range gates, radial wind speed [RWS], effective radial wind speed [RWSeff], signal-to-noise ratio [SNR], etc.)

One goal of machine learning in this project is to locate vortex centroids, which can help accelerate subsequent algorithms that produce information on the circulation, core radius, separation, downwash and eventually motion estimation/temporal evolution. Inspired by the WINDCUBE WV characterization algorithm in Cariou and Thobois,[5] we propose the following tasks described more in-depth in the next sections:

1. First, establish the existence of a vortex in a scan, which is a binary classification problem solved by the methods of supervised learning.
2. Determine and single out a region of interest for the vortex, which is a matter of clustering data points presenting behavior consistent with those we see in actual vortices. This would constitute an unsupervised approach which has been the focus of the team's efforts over the past few months.
3. Finally, we predict the vortex centroids through regression based on supervised learning.

Machine-Learning Tasks

As noted previously, we propose using both supervised and unsupervised approaches. For the first task of recognizing the existence of a vortex, we use a supervised approach. We can pre-label data points as "Vortex" (or "1") if they contain vortices,or "Non-vortex" (or "0") if they do not. A number of files have already been recognized as containing vortices, so this task becomes one of either organizing files into two folders representing each class and/or just appending the label in the wtv structure directly. Once this pre-labeling step is complete, we can train and test an algorithm such as a CNN. A large data set will be required to train, so we also attempt to make use of data augmentation techniques. Once a viable model* is produced, we can pass the scan(s) containing vortices to the next machine-learning phase.

The next step is to recognize a vortex region of interest (ROI) within each scan, which will primarily be an unsupervised learning approach consisting of clustering and feature selection. This is the step the team has focused on for past few months, and for which we present results in the sections to come. Two major subcomponents of this step include data smoothing and data reduction. We want to smooth the data over any discontinuities, but especially to help in replacing some missing (not a number[NaN]) values. This smoothing is suggested to be done using a Savitzky-Golay filter (SGF). To help in characterizing vortices, it would help to also characterize the background atmosphere. However, some of the background atmosphere in each scan (especially in the latter half of the scans) contains an exorbitant density of NaN values, locations at which the LIDAR system could not reasonably determine the RWS. Smoothing over these regions will likely not add (and may even detract) from the background atmospheric information we seek. So, we would like to reduce the scan down to the relevant vortex regions and statistically viable background (the latter which is to be defined.) Another technique for data reduction which we are exploring is principal component analysis (PCA).†

*Model viability will be measured by accuracy maximization and/or loss minimization, which typically go hand in hand and can be managed with regularization techniques (k-fold cross-validation, flip points, etc.)

†The features of a data set may have overlapping information, and the measure of overlap we are considering is orthogonality. If we consider the features of a data set to be the “axes" of a coordinate system, these axes may overlap, so there would be redundancy in the representation of the vectors (data points). Orthogonalizing these axes (using, for example, the Gram-Schmidt process), removes these redundancies. We can organize these axes (components) in order of how much variation of the data set each contains. Then we can select the components up to the k-th most variation (which we call the k-th principal component), and process the resultant transformed data set.

The main problem this machine-learning step presents is generalization. The difference between this step and the other two are in what the data set consists of. The other two steps view each scan as a data point. This step views the sampled points within each scan as the data points (i.e., one scan is the entire data set). A clustering algorithm can surely be optimized for each scan. The generalization problem, then, is to determine the possibility of “recycling" the optimizations (that is, learn from experience) for a more general and efficient application on new, incoming data.

Currently, the team has been testing uses for the k-means and DBSCAN clustering algorithms. The initial hypothesis is to use DBSCAN over the entire scan to highlight the vortex region of interest, and then use k-means over the vortex region itself. While k-means has shown promise in terms of optimization (highlighting the overall vortex regions well), run-time, and quality of clustering over each global scan, DBSCAN has shown the most promise in terms of detail (partitioning the vortex regions into substructures which we remain curious about exploring.) These differing behaviors are thought to lie in the assumptions underlying each algorithm; k-means produces convex clusters, but DBSCAN uses the idea of density-based clusters (see Section 3.1 for a more detailed description). One issue we face is developing metrics to compare the clustering qualities based on multiple cluster concepts. The team has also tested the effects of filling missing values and smoothing the data on clustering, which is pointing in a promising direction for our goal of vortex isolation.

The last step of this machine-learning process is to predict the vortex centers using supervised regression techniques. We consider each scan to be a data point labeled with the correct centroid locations. The procedure for training/testing is similar, if not equal, to that of the first step. For this task we will research the benefits and disadvantages of regression techniques like multilinear, lasso/ridge, and logistic as well as other algorithms such as support vector machines and neural networks.

3. Applied Cluster Analysis: Vortex Detection

Recent research efforts were dedicated to the machine-learning task of isolating the vortex region of interest. An initial hypothesis led to research involving two clustering algorithms, k-means and DBSCAN. DBSCAN is applied first to isolate the vortex region and then k-means is used to further break down the region for later analyses. We explore both algorithms acting individually on a scan to highlight their separate behaviors.

3.1 Cluster Concepts, Algorithms and Optimization/Parameter Selection

We provide a slightly more in-depth discussion of the k-means and DBSCAN clustering approaches followed by descriptions of optimization and parameter selection process, respectively, for the algorithms.

3.1.1 k-means

k-means is probably the simplest clustering formalism to understand and apply. Its simplicity in part lies in the types of clusters it is assumed to find, namely convex clusters. A set (which is a subset of real vector space) is convex if we take two different points, x and y, in the set, then every point lying on the line segment connecting x and y also lie in the set. Mathematically,

Definition 1 *Let $A \subset \mathbb{R}^n$. A is said to be **convex** if given $x, y \in A$, then $\lambda x + (1 - \lambda)y \in A$ for all $\lambda \in [0, 1]$.*

For example, any globular cluster of points is convex, whereas say data in the shape of a torus is not convex. Figure 1 shows an example of convex and non-convex polygons.

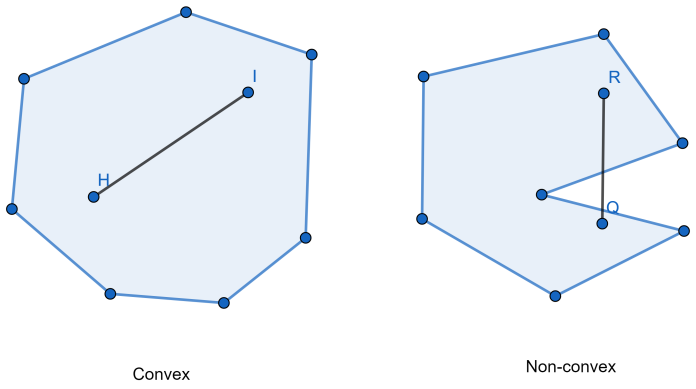


Fig. 1 Example of convex (left) and non-convex (right) polygons

The process for finding the resulting clusters using k-means is Lloyd's algorithm,[15] which is outlined here.

Algorithm 1 ***(Lloyd's algorithm for k-means):*** *Suppose we have a data set $X = \{x_i : i = 1, 2, ..., N\}$ that we would like to partition into K clusters. We first initialize the K "cluster centers", $m_1, m_2, ..., m_K$. Then we repeat the following steps until the cluster center locations have converged.*

1. *Assign the data point x_j to the closest cluster center, for $j = 1, 2, ..., N$.*
2. *For each cluster, compute a new cluster center by taking the mean of the assigned points.*

Initialization of the K clusters may be done randomly, as in the case for the k-means++ implementation, see Arthur and Vassilvitskii.[27] However, using a different initialization may result in different clustering results. In particular, for Lloyd's algorithm, we can compute the intracluster sum of distances between each point and the final cluster center. We can repeat this for every new initialization of the algorithm, and then use the clustering for which the intracluster sum of distances is minimum. This process is the basis for MATLAB's implementation of k-means; please see MATLAB's documentation[28] for the "kmeans" function.

3.1.2 k-Means Optimization

All in all, we used a brute-force method for optimizing the k-means algorithms. That is, we applied k-means clustering using a set of k values, $\{1, 2, 3, ..., n\}$, and evaluating the clustering quality using the silhouette score metric. From MATLAB's documentation[28] on the "silhouette" function (which summarizes Kaufman and Rousseeuw[29]), the silhouette score for the i-th point, S_i, is defined as

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)},$$

where a_i is the average distance from the i-th point to the other points in the same cluster as i, and b_i is the minimum average distance from the i-th point to points in a different cluster, minimized over clusters. The silhouette value ranges from –1 to 1. A high silhouette value indicates that the i-th point is well matched to its own cluster, and poorly matched to other clusters. We use a mean silhouette score averaged over the points to determine the quality of the clustering for a particular k value. The k value resulting in the highest average silhouette score yields the optimal number of clusters for the data set. Figure 2 shows an example graph of the

(average) silhouette score varying over the number of clusters when evaluating k-means. Though difficult to tell, the maximum silhouette score of 0.743371 occurs at $k = 5$. In our preliminary experiments, we also tested the other local maxima for the quality of their clusters. While increasing k leads to improved detail in the clustering, it comes with a higher computational cost; more details are presented in the results.

3.1.3 DBSCAN

From Ester et al.,[25] the essential idea behind the density-based model of clustering is that we can require for each point in a data set there be a certain minimum number of points, m, within a certain distance, ϵ, of that point; however, this requirement may not be enough to fully characterize clusters due to **border points**, points on the border of a cluster. These have less points in their neighborhoods than **core points**, those inside of a cluster. So, we would have to set, m, to a low value to include all points of a cluster. The details and full development of this clustering concept are given in Ester et al.[25]; however, the Appendix provides some fundamental definitions and concepts.

3.1.4 DBSCAN Parameter Selection Automation

The overall parameter selection process we use is provided in MATLAB's documentation[28] for the "dbscan" function; the documentation summarizes the findings from Ester et al.[25]

Two parameters are of particular interest for DBSCAN for the proper clustering of a data set: the minimum number of neighboring points required to determine a core point, m, and the neighborhood size of a point, ϵ. From the MATLAB documentation[28] on the "dbscan" function, we can choose $m \geq n + 1$, where n is the dimension of the data set. Further, "One strategy for estimating a value for [ϵ] is to generate a k-distance graph for the input data X. For each point in X, find the distance to the kth nearest point, and plot sorted points against this distance. The graph contains a knee. The distance that corresponds to the knee is generally a good choice for epsilon, because it is the region where points start tailing off into outlier (noise) territory." The graph being referred to is called the k distance graph (kDG), or sometimes we refer to it as the k nearest points graph (kNPG). The k value referred to here is the same as m.

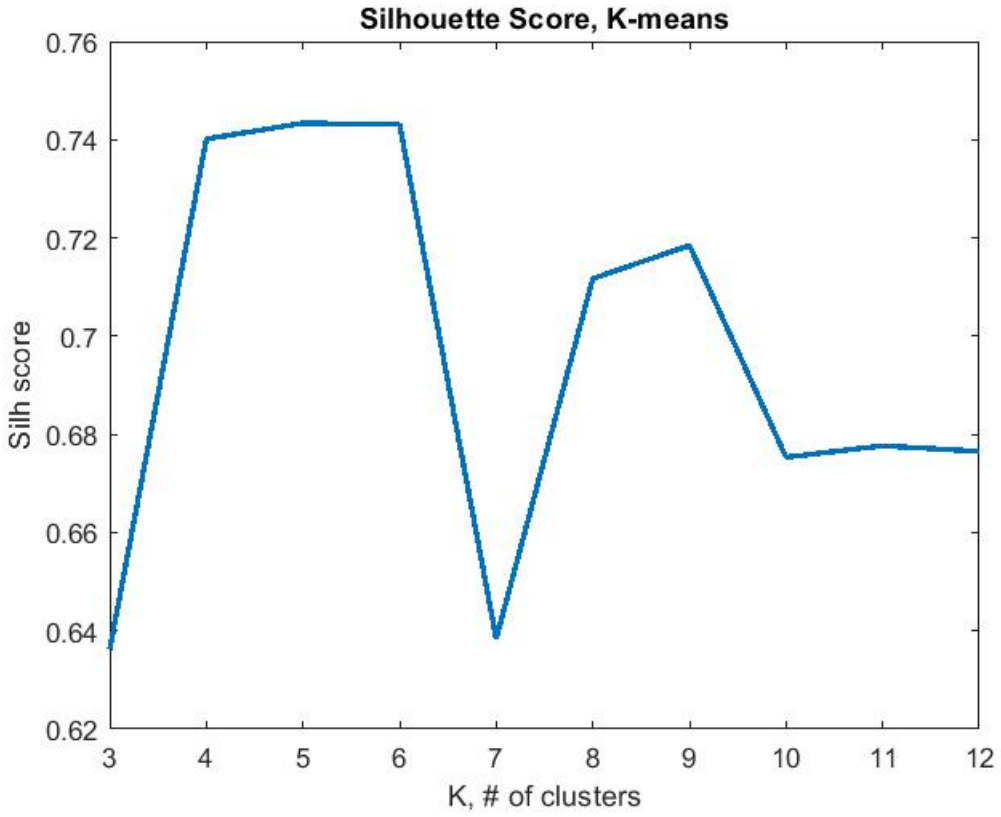


Fig. 2 Example plot of silhouette scores vs k. While subtle, the maximum silhouette score of about 0.743371 occurs for $k = 5$.

3.2 Data Set: Preparation and Use

We describe the data processing from raw to the wtv structures and finally the cluster matrices that are used in this project. The main processing is based on Boccippio's[30] volume velocity processing (VVP) algorithm. The basic idea is taking the volumetric pattern to create a vertical profile of the wind vectors, which the profiler does at the source LIDAR. The mean wind which is established by the profiler contains uniform velocities in each altitude layer. This mean wind is then removed from the raw wind.

The final wtv structure houses substructures, each containing a different aspect of the scanned area like horizontal and vertical spatial coordinates, RWS, RWSeff, SNR, and Beta (the backscatter coefficients). Each substructure is a matrix. For example, RWS may be represented by an $m \times n$ matrix given by $[r_{ij}]_{i,j=(1,1)}^{(m,n)}$, where r_{ij} gives the matrix entry of the i-th row and j-th column. Each row represents an elevation angle (a ray) of the scan, and each column represents a range at which the LIDAR took a sample measurement of the RWS. These matrices are then flattened into 1-D vectors, and appended to form the cluster matrix. That is, the cluster matrix primarily contains columns which are the flattened matrices of each of the features we wish to examine. For illustration, suppose we want to perform cluster analysis using RWS, SNR and Beta, which are represented by the matrices $R = [r_{ij}]_{i,j=(1,1)}^{(m,n)}$, $S = [s_{ij}]_{i,j=(1,1)}^{(m,n)}$, and $B = [b_{ij}]_{i,j=(1,1)}^{(m,n)}$, respectively. We flatten R into a vector, r_f, B into b_f, and S into s_f, as shown in Eq. 1. Finally, we append these columns into a matrix M also given in Eq. 1.

$$r_f = \begin{bmatrix} r_{11} \\ \vdots \\ r_{1n} \\ r_{21} \\ \vdots \\ r_{2n} \\ \vdots \\ r_{m1} \\ \vdots \\ r_{mn} \end{bmatrix}, \; b_f = \begin{bmatrix} b_{11} \\ \vdots \\ b_{1n} \\ b_{21} \\ \vdots \\ b_{2n} \\ \vdots \\ b_{m1} \\ \vdots \\ b_{mn} \end{bmatrix}, \; s_f = \begin{bmatrix} s_{11} \\ \vdots \\ s_{1n} \\ s_{21} \\ \vdots \\ s_{2n} \\ \vdots \\ s_{m1} \\ \vdots \\ s_{mn} \end{bmatrix} \quad \Longrightarrow \quad M = \begin{bmatrix} r_{11} & b_{11} & s_{11} \\ & \vdots & \\ r_{1n} & b_{1n} & s_{1n} \\ r_{21} & b_{21} & s_{21} \\ & \vdots & \\ r_{2n} & b_{2n} & s_{2n} \\ & \vdots & \\ r_{m1} & b_{m1} & s_{m1} \\ & \vdots & \\ r_{mn} & b_{mn} & s_{mn} \end{bmatrix} \tag{1}$$

M is called a **cluster matrix**, which is then passed into our algorithms for analysis.

The following data-cleaning concepts are used primarily during vortex isolation.

3.2.1 Filling in the Missing (NaN) Values

For the first part of data cleaning, we fill the missing (NaN) values in the scan to reduce any potential variation from lack of data. In particular, we make use of MATLAB's "fillmissing" function; please see MATLAB's documentation.[28] This function works on a vector; if the input is a matrix, then it will work independently along each column of the data set. While there are a number of methods for replacing missing values, we chose a simple moving average. That is, when a NaN is encountered in the vector, the function replaces the value with mean of the values within a window centered on the NaN value. The user can select the window size. For our data set, this function is applied along each ray of the scan.

3.2.2 Smoothing with a Savitzky-Golay Filter

Next, we apply a smoothing method known as a Savitzky-Golay filter (SGF), also along the rays of the scan. The basic idea behind the SGF is polynomial interpolation. We take a set of data points, fit a polynomial, and locally replace the data points by the corresponding values of the fitted polynomial. The SGF concept is developed more in-depth in Schafer.[31] For a primer on numerical methods on polynomial interpolation, see Trefethen and Bau.[32] MATLAB's "sgolayfilt" function[28] automates this process. Initially, it was proposed that the 2-D analog of SGF (developed in Suhling et al.[33]) would provide a better smoothing of the entire scan, but due to time limitations, we leave this avenue to be examined in the future.

3.3 Applying the Algorithms

3.3.1 Example of Automating DBSCAN Parameter Selection

For our example, we use Doppler LIDAR data consisting of an RHI scan of wingtip wake vortices. The scan is taken in the plane perpendicular to flight path of the aircraft. In this example, we make use of RWS only, plots of both raw RWS and RWSeff are presented in Fig. 3, with the vortices circled by a black border. The goal is to see whether the vortex region can be highlighted properly via clustering.

This data set is 3-D, so we must choose the minimum number of neighboring points $m \geq 4$; we choose $m = 10$. Now to choose ϵ, we first produce the k-distance graph

(kDG) for the data set. The full graph is presented in Fig. 4, along with a plot zoomed in on the "knee" of the graph. The goal is to automate the search for the ϵ values in the "knee" region.

The following algorithm gives our current implementation for selecting ϵ.

Algorithm 2 ***(DBSCAN Parameter Selection):*** *With the minimum number of neighboring points, m, held fixed.*

1. *Compute the average slope, m_{avg}, of the entire kDG, using the first and last points.*
2. *Compute the approximate derivative of each point in the kDG.*
   (The "derivative" at the i-th point is actually a finite approximation using a five-point stencil using the $(i-j)-th$ points for $j=-2,-1,0,1,2$.)
3. *Find the points (and corresponding ϵ values) whose approximate derivatives falls within a certain tolerance of m_{avg}. If no such points are found, then increase the tolerance, and search again.*
4. *Use any of these ϵ values as the ideal, but it is suggested to use the lowest.*

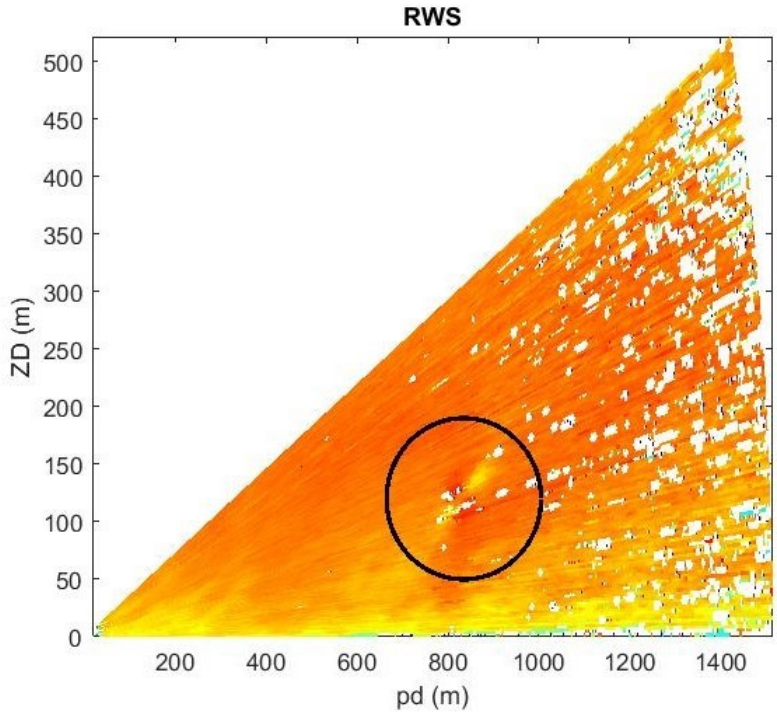
These steps are illustrated by the plots in Figs. 5–9.

Again, the steps in this algorithm just serve as a visual example. Using this method, we find the optimal parameter values to be $m = 10$ and $\epsilon = 0.016986$. We apply these in actual DBSCAN algorithm, resulting in the clusters depicted in Fig. 7.

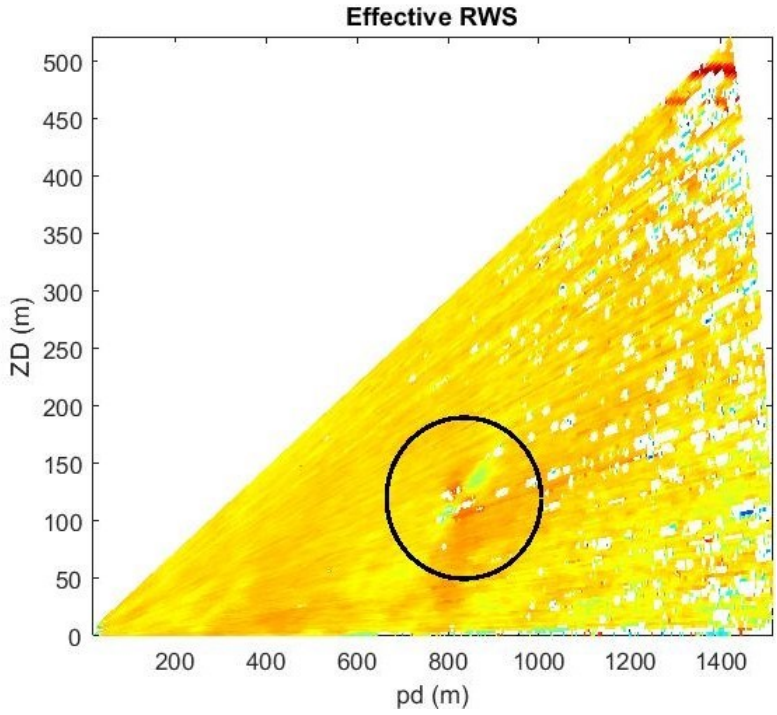
3.3.2 Problems to Address

The previous example of our algorithm's implementation gives us a few ideas to think about for the future. First, the clustering highlights the "larger structure" regions well; we have over 250 clusters. For reference, k-means produces around 5–8 clusters. We could explore which of the 250 clusters have the most relevant information pertaining to our goal of vortex characterization. It may be useful to process the data further before running this optimization. For example, it was suggested that we extract the gradient data of the effective RWS.

The general method of finding ϵ also seems inexact in its definition of the "knee" region of the kDG. From Fig. 8, it could be argued that we should look a bit farther

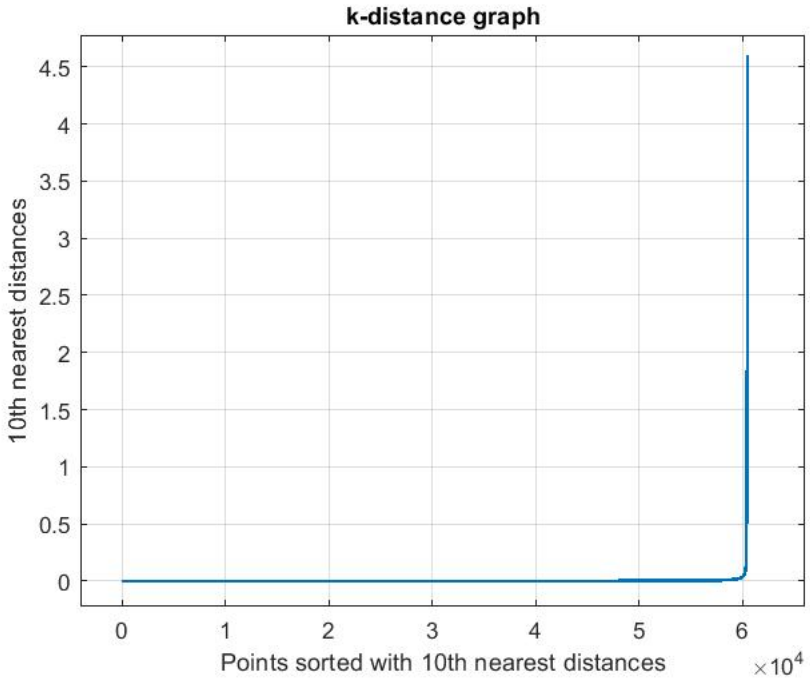


(a) Raw RWS data

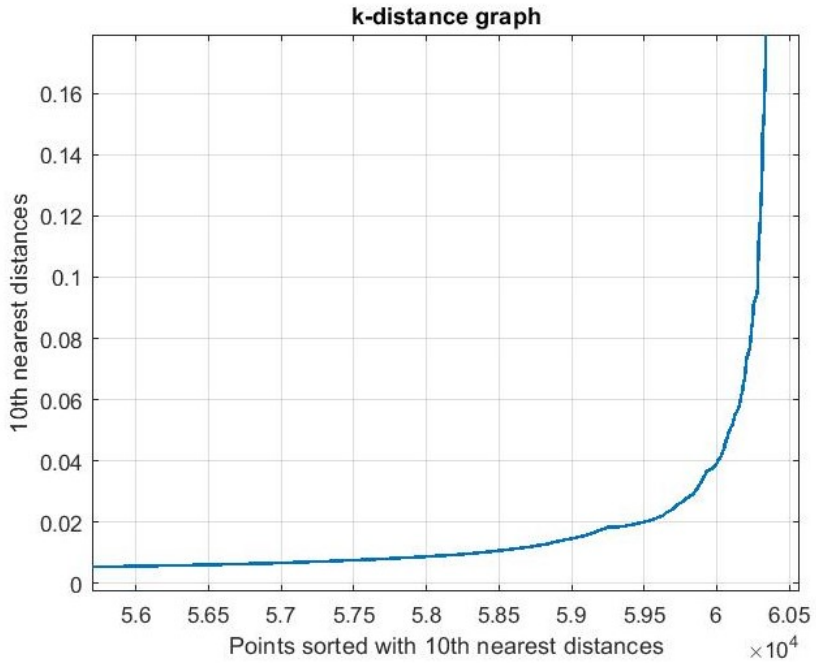


(b) Effective RWS data

Fig. 3 RWS and RWSeff plots for example parameter selection algorithm application. The vortices lie inside the black circular border.



(a) Full k-distance graph. Note, there are more than 60,000 data points.



(b) Zooming in on the "knee" region of the graph.

Fig. 4 k-Distance graphs to determine the best neighborhood size ϵ given a fixed minimum number of neighboring points, m

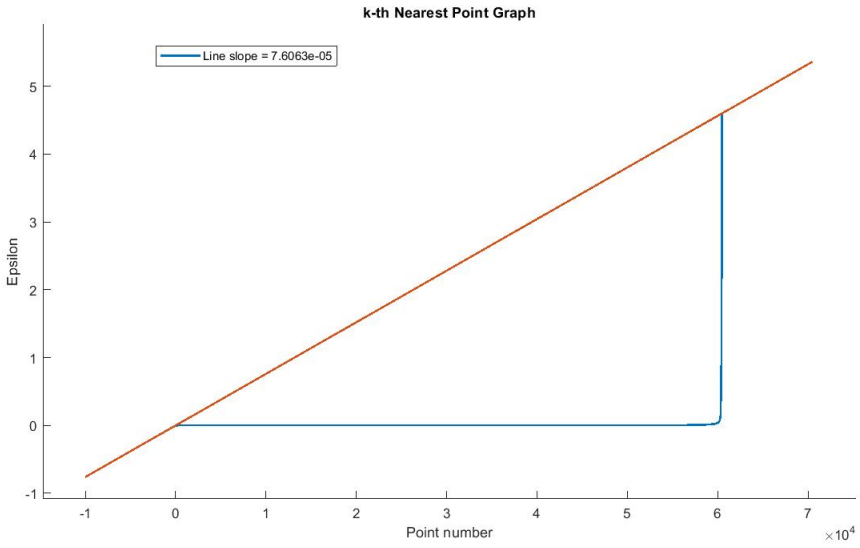


Fig. 5 We use the first and last points of the kDG to compute the average slope. The graph depicts the line with this computed average slope.

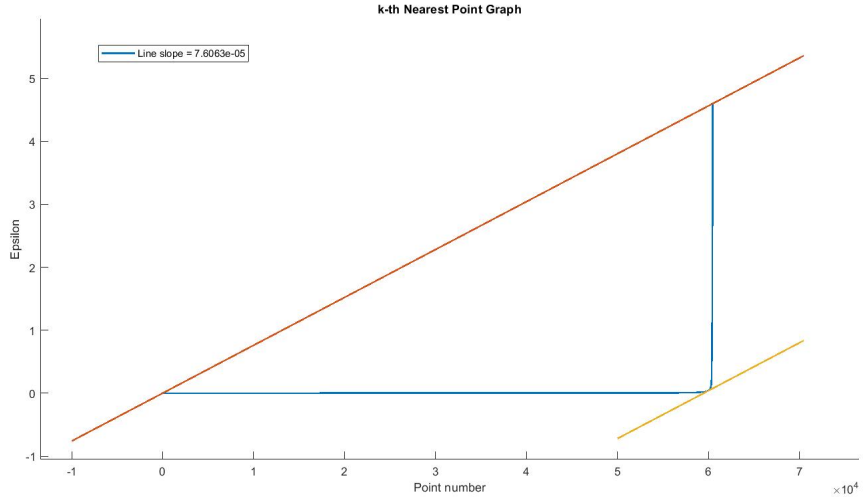


Fig. 6 We approximate the derivative at each point of the graph, and compare it to the average slope. Here the average slope is shifted toward the "knee" region of the graph.

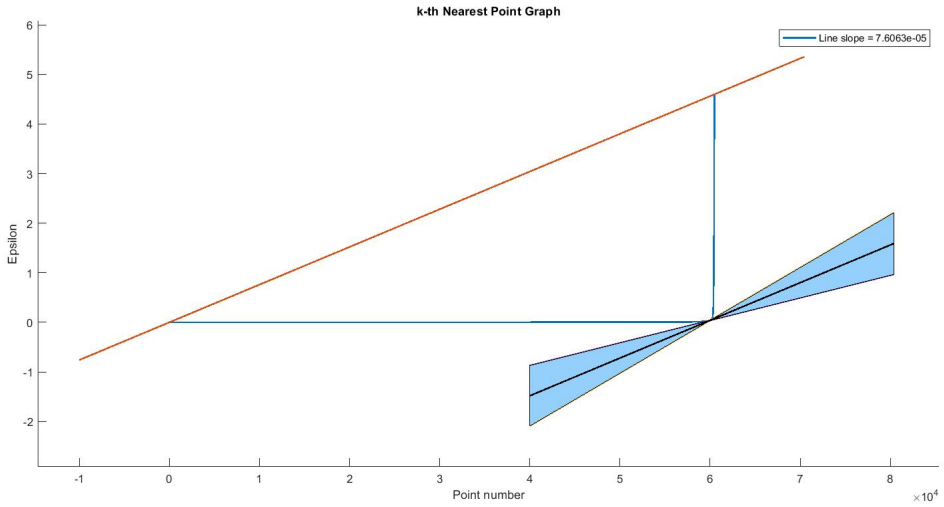


Fig. 7 We find the points whose derivative approximations lie within a tolerance region of the average slope (shown shaded in blue). For illustrative purpose we allowed the tolerance to be 40% of the average slope.

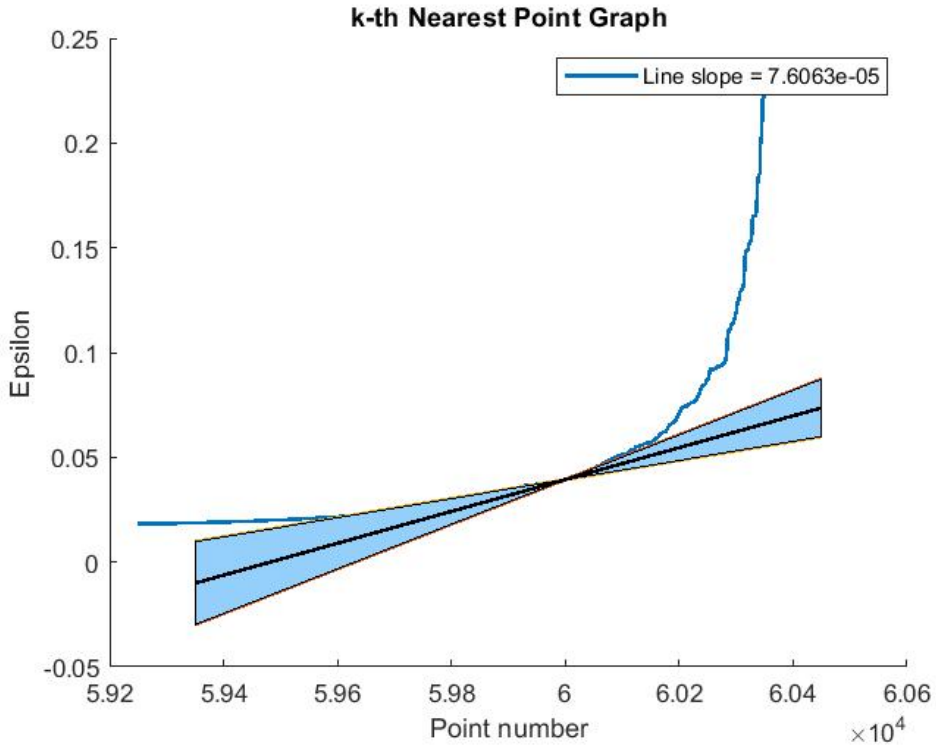


Fig. 8 A zoomed-in version of Fig. 5. We see that viable ϵ values lie approximately in the interval $[0.04, 0.05]$.

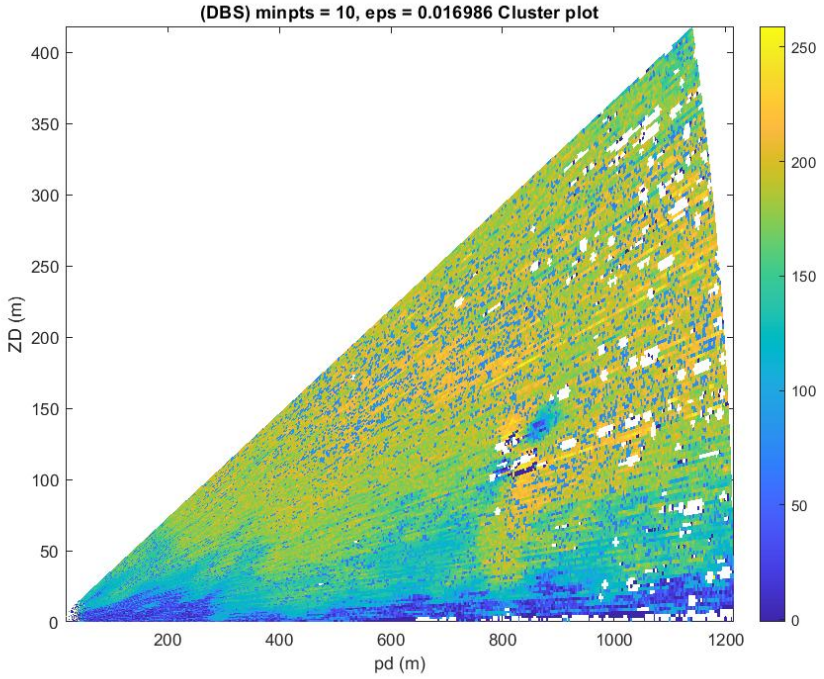


Fig. 9 Example of DBSCAN clustering results on the raw RWS data using the automated parameter selection process

up the graph. Indeed, it has been observed that increasing ϵ decreases the number of clusters. However, this process involves a fair amount of volatility; if we change ϵ slightly, while staying within the "knee" region, this could lead to large changes in the resulting number of clusters. It has been observed that cluster sizes usually lie in the interval $[4, 200]$, with a higher probability of lying toward the end points.

Another problem to address is whether the average slope of the kDG is an acceptable baseline in the search for ϵ. For example, what if either end of the graph is "longer", biasing the slope toward noise or an actual "cluster-able" point?

3.4 The Inception of the LIDAR Cluster Analysis Program (LCAP)

The LCAP grew out of desire to automatically run clustering algorithms on many scans sequentially. We hoped to mainly automate the process of uploading data to the MATLAB workspace, which then extended the capabilities to perform basic clustering optimization, as well as displaying and saving the results. The program maintains ability to run on manually selected data sets, as well.

The structure of LCAP is presented in Fig. 10. We call the main script "Driver"; it is responsible for running all other scripts needed for the analysis. We were able to automate directory list creation, directory accessing, and data file uploading processes. We build the cluster matrix from preselected features and finally perform our clustering analyses for k-means and DBSCAN, independently. This process is repeated for each folder in the directory list. Many of the subtasks of "Driver" are housed in separate scripts, which in Fig. 10 are represented by separate boxes. This structure diagram is not exhaustive, in that some of the subscripts may also have subscripts, but their main functions are provided. This structure for the LCAP eased testing of each component and easily allows the researcher to decide which subtasks they would like to include. For example, if only the k-means analysis is of interest, then the researcher can comment out the DBSCAN Analysis section.

4. Results

The results are divided into two sections: Section 4.1, Preliminary Runs, displaying our initial explorations involving the behavior of the two clustering algorithms, and Section 4.2, Vortex Isolation, illustrating the extended capability of the LCAP to isolate the vortex region.

4.1 Preliminary Runs

We present the results of the preliminary clustering experiments using three different types of scans. The first scan illustrates our most well-behaved vortex in terms of how nicely the vortex region is delineated by the algorithms. The next scan contains only one wingtip vortex whose RWS magnitudes are not as prominent as the well-behaved vortex. Lastly, we present a scan containing a vortex in the midst of NaN noise.

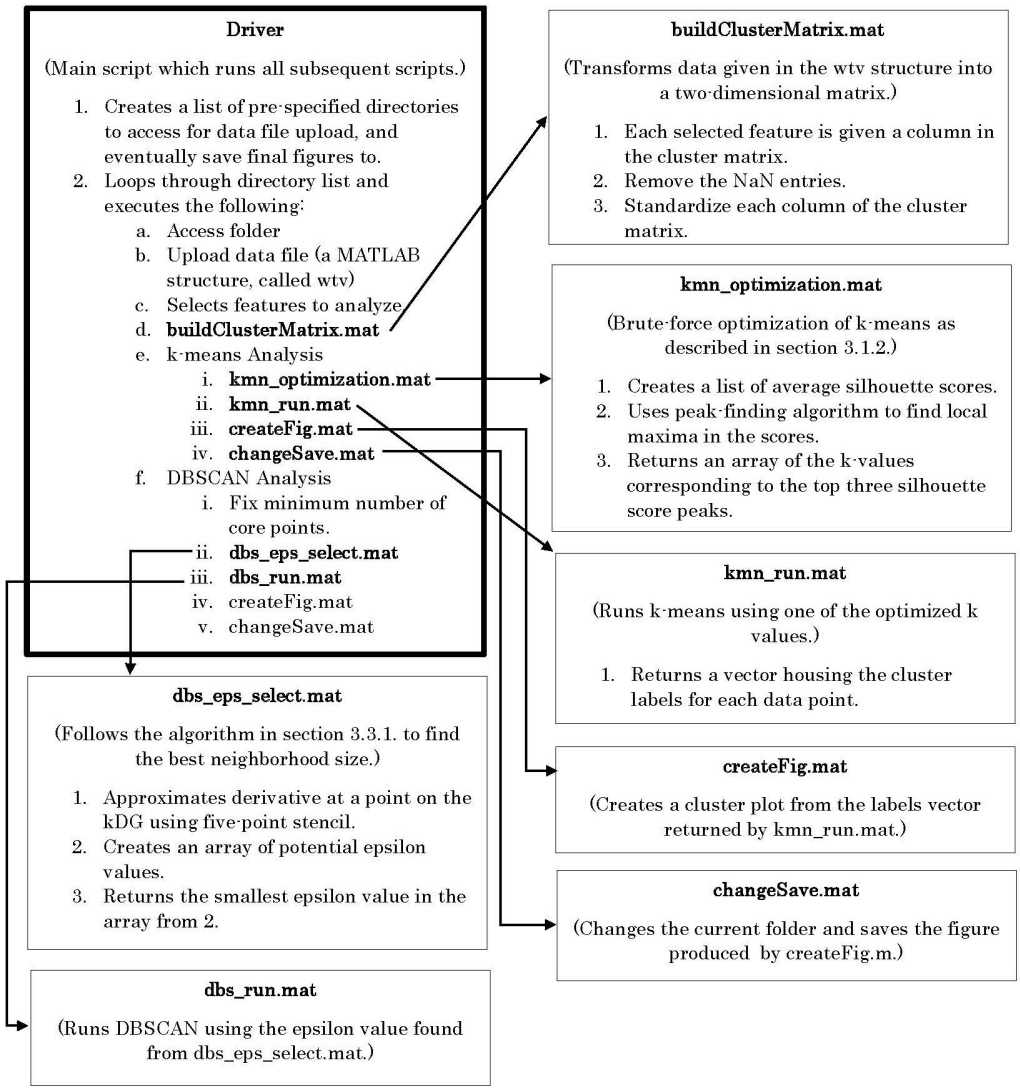


Fig. 10 Overall structure of LCAP. Each box represents a different script. The "Driver" is the main script which runs the entire analysis, from creating directories to running the actual clustering. Each of its subscripts may also have subscripts to help them function efficiently.

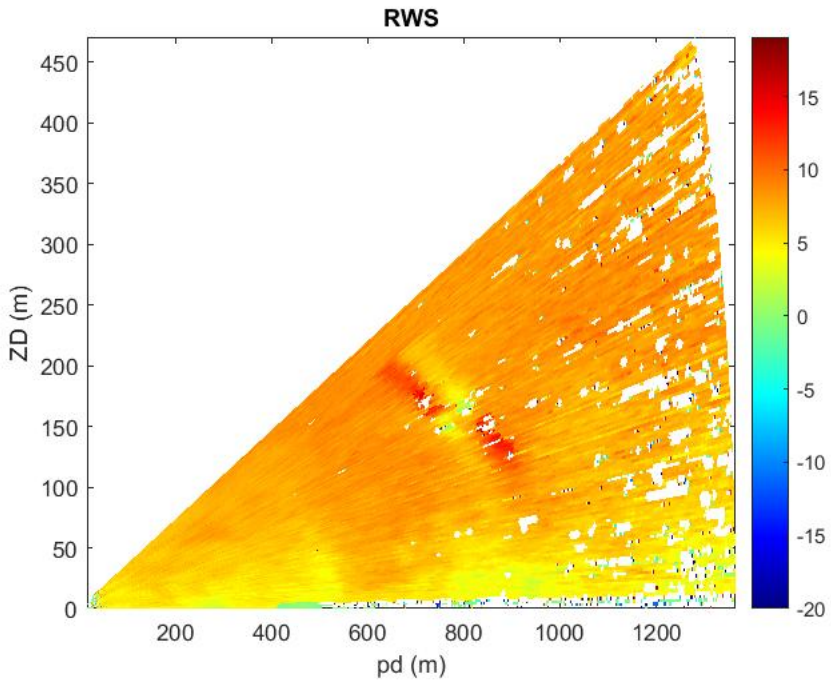


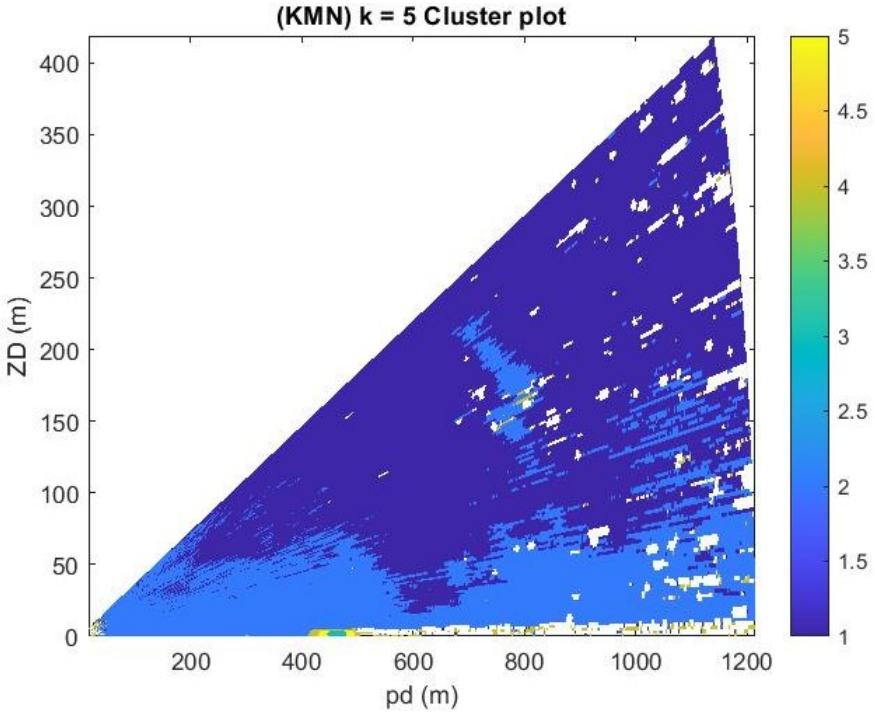
Fig. 11 Original RWS plot of well-behaved vortex. The vortex region is clearly displayed solely using the available color map.

4.1.1 Well-behaved/Ideal Vortex

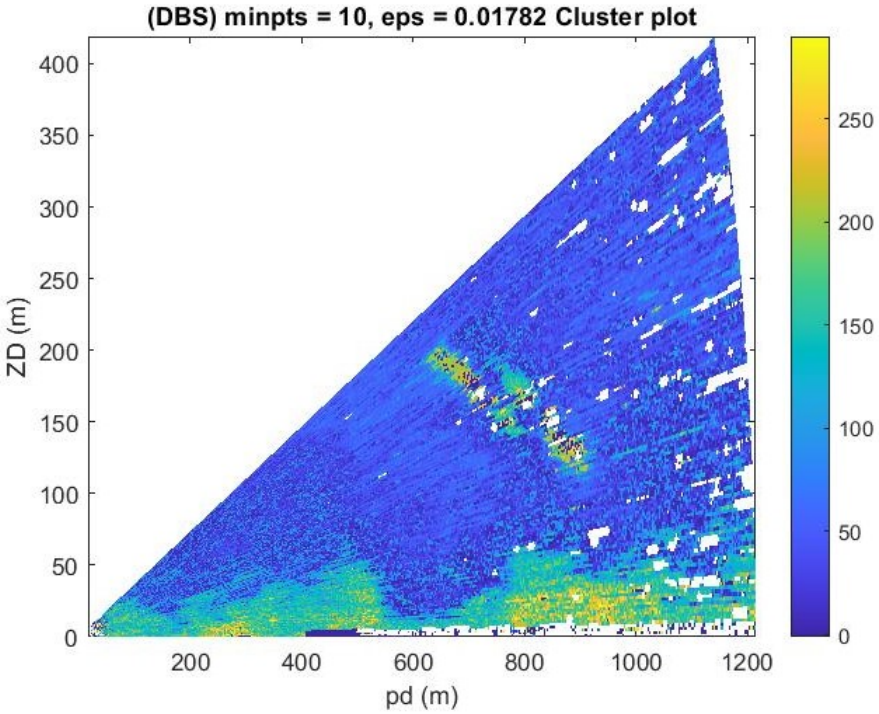
Our most well-behaved scan is displayed in Fig. 11. It is well behaved because of how the vortex region, showing two trailing wingtip vortices, can be clearly discerned in the raw RWS plot without any data manipulation, but primarily because of how well our clustering algorithms group the vortex region. In the figure, we see the positive RWS region highlighted in red (using color map values around 10–15) and the negative RWS region is highlighted in yellow (using color map values around 0–5.)

We illustrate the results of both clustering algorithms using RWS, SNR, and Beta on a scan. While most of the relevant information lies in the RWS, including SNR and Beta accounts for a nontrivial change in the clustering results. The clustering results are presented in Fig. 12.

We see that both k-means and DBSCAN perform well visually in highlighting the vortex regions, with few exceptions. While the vortex region is clearly displayed in the higher elevation angles, the clustering algorithms associate this region with the



(a) k-means with $k = 5$.



(b) DBSCAN with minpts = 10 and (automatically selected) eps = 0.01782.

Fig. 12 Applying k-means and DBSCAN algorithms to the well-behaved scan in Fig 11

data toward low elevation angles. This association is likely due to the high gradient nature of both the vortex and winds in the low-elevation region; the gradients for the latter are likely due to topographic influence. So any basic attempts to isolate the vortex clusters could also result in isolation of this low-elevation region. We also note k-means seems to only cluster the positive RWSs from Fig. 11, whereas DBSCAN mainly segments the negative RWSs with a bit of the positive RWSs. A major difference is in the number of clusters found: 5 clusters for k-means and almost 300 clusters for DBSCAN. For k-means, $k = 5$ was found as the optimal number of clusters using the brute-force algorithm described Section 3.1.2, and similarly the 300 clusters resulting from DBSCAN were determined by the parameter selection process in Section 3.1.4.

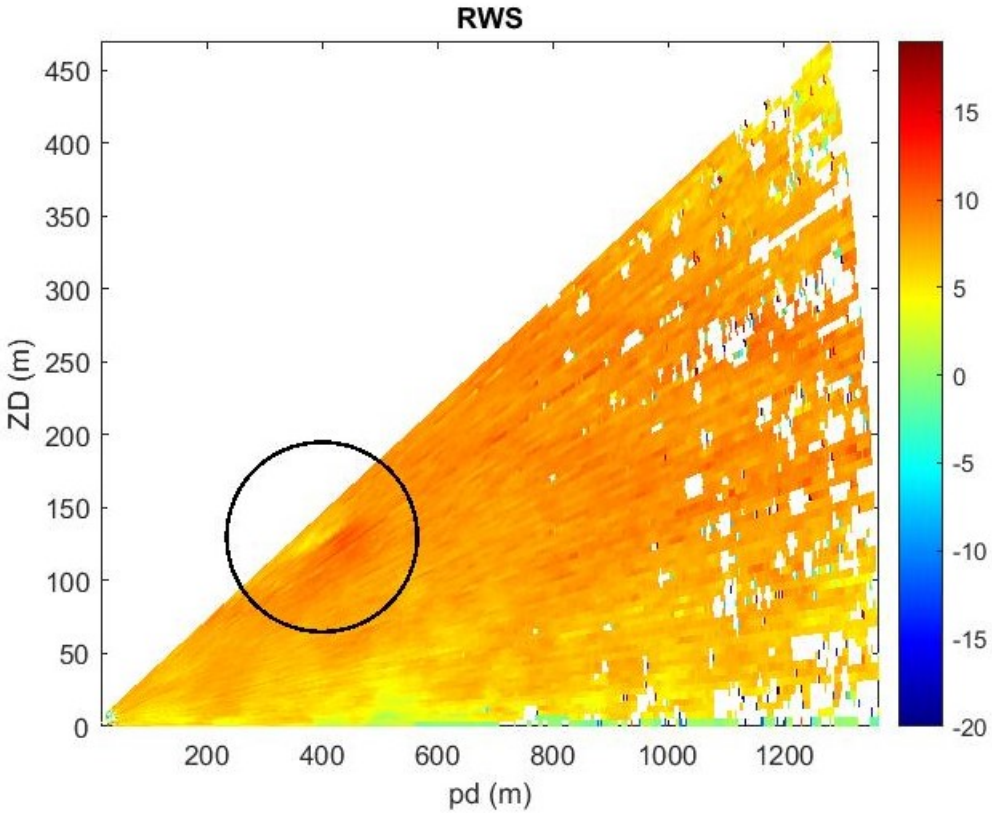


Fig. 13 Plot of the RWS where only one vortex exists, shown in the black circle. Note the RWS magnitudes for the vortex region are not as pronounced as they are for the previous scan.

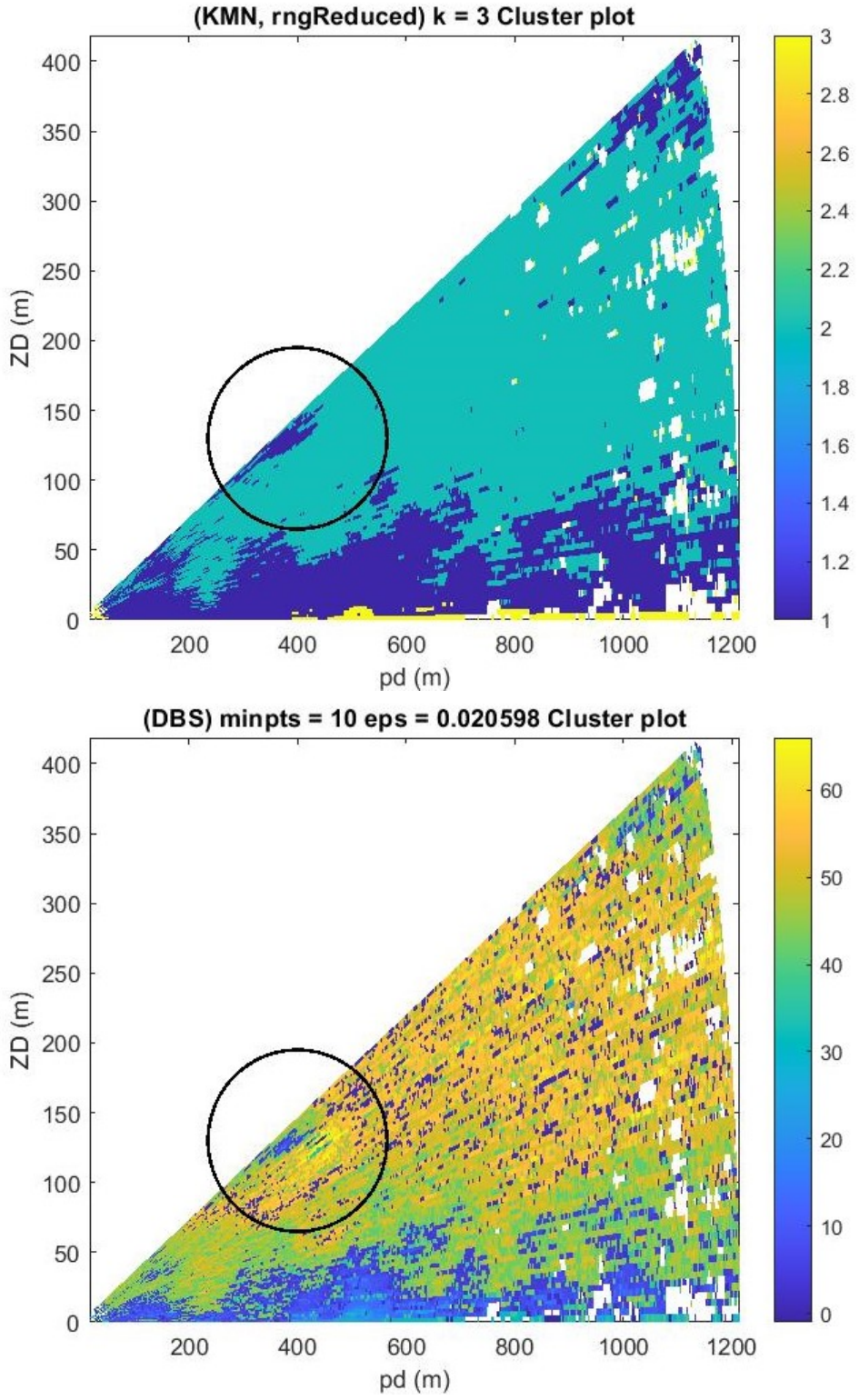


Fig. 14 Clustering results for the single vortex/low RWS magnitude case. (Top): k-means with $k = 3$ and (bottom): DBSCAN with minpts = 10 and (automatically selected) $\epsilon = 0.020598$.

4.1.2 Single Vortex Scan

We examine the case of only one vortex within the scan. Figure 13 depicts such a scan with the vortex inside a black circular border. The RWS magnitudes for the vortex region are not as pronounced as the previous scans.

Figure 14 shows the results of the clustering algorithms on only the RWS of this scan. We chose not to include SNR and Beta for this scan to speed up the algorithm and, more importantly, to illustrate the power of the algorithms given less information. We see similar results: the vortex regions are being clustered with the data points at low elevation angles, k-means seems to highlight the positive RWSs, while DBSCAN highlights a bit of both the positive and negative RWSs. We note the algorithms' ability in discerning such regions given the magnitudes of the RWS.

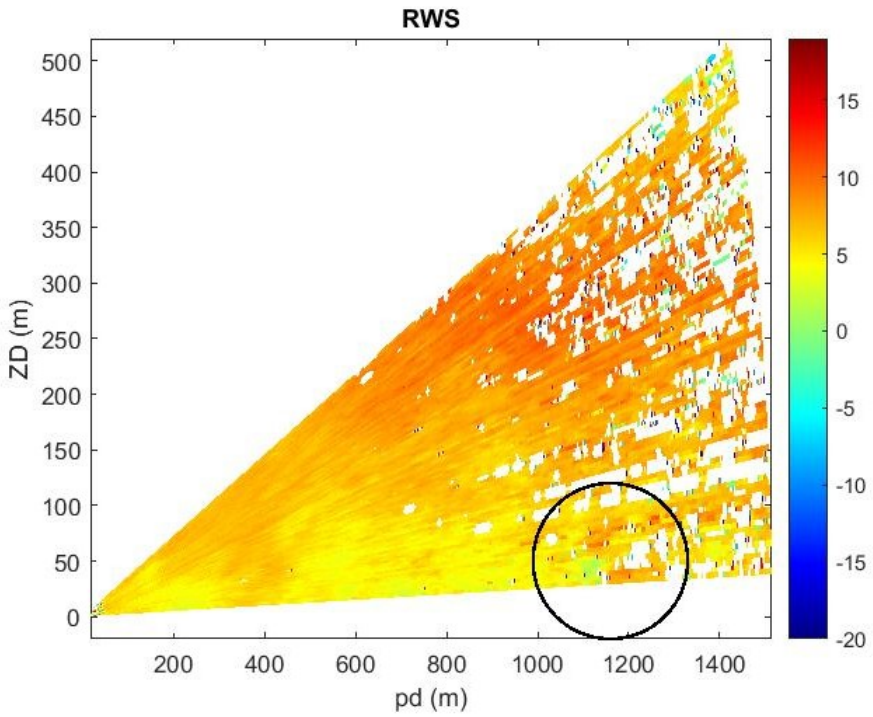


Fig. 15 A scan containing a vortex region in the midst of noise. The black circle encompasses the vortex region.

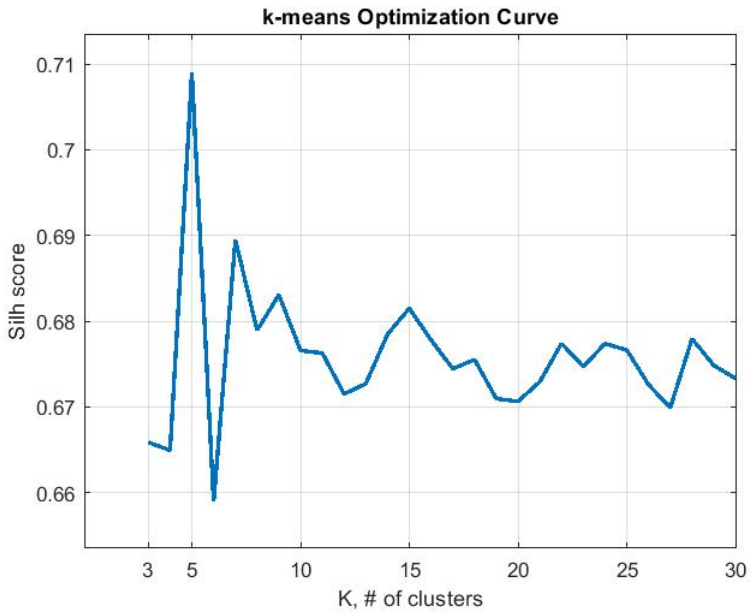


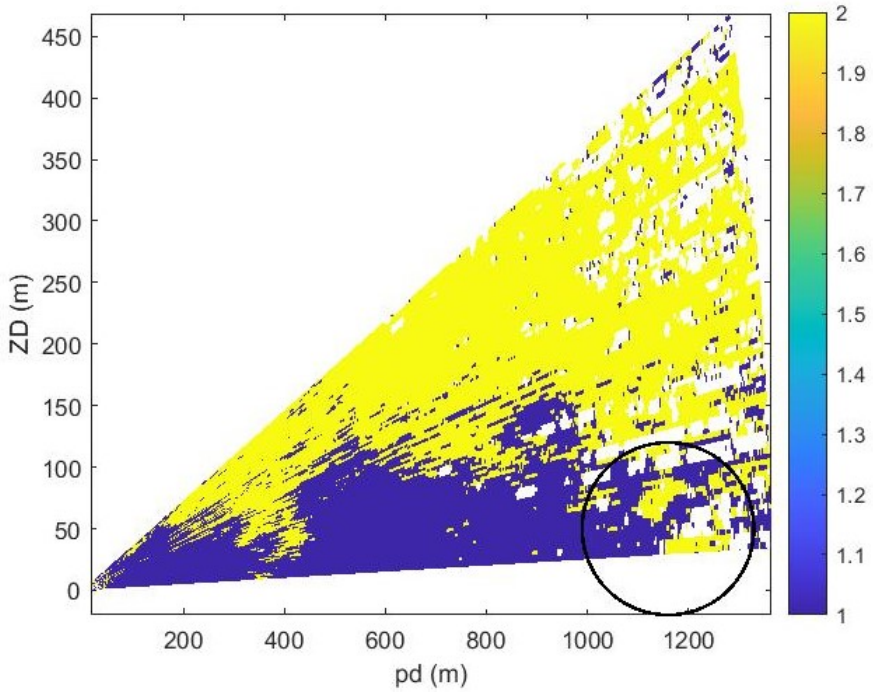
Fig. 16 Optimization curve of the LIDAR presented in Fig. 15

4.1.3 NaN-Noise Vortex

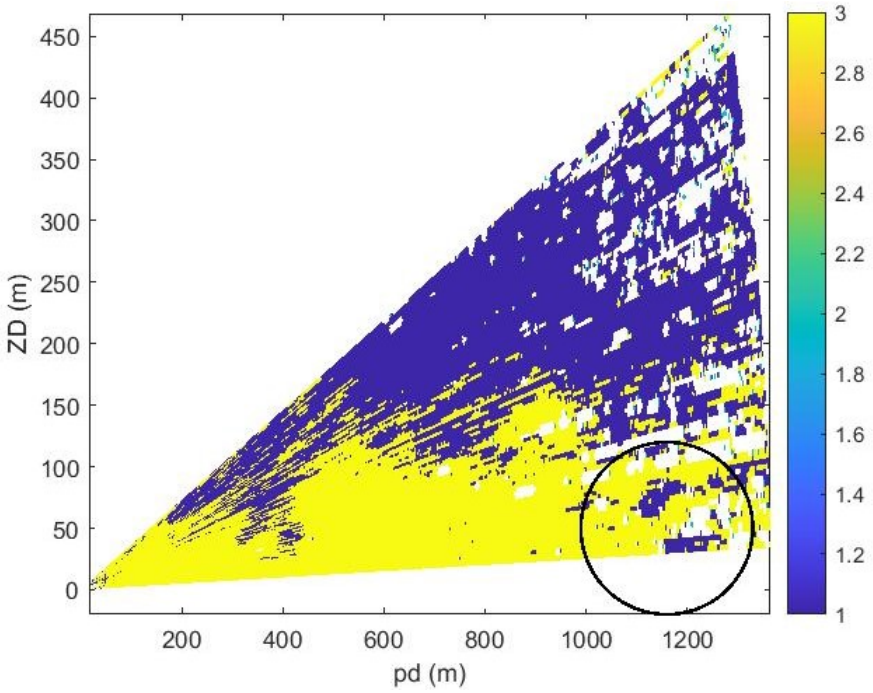
We look at the case of a scan containing a vortex region in the midst of noise (NaNs), to see how these algorithms perform without filling in data and smoothing. We also illustrate the effects of changing parameter settings of the algorithms, namely the number of clusters for k-means and the minimum number of neighboring points for DBSCAN. Figure 15 illustrates the RWS data of such a scan; the vortex region is given by the shades of green and darker orange-red inside a black circular border.

For k-means, we illustrate the clustering results for $k \in \{2, 3, 5, 7, 9, 15\}$. The main reason for choosing these values stems from the optimization curve of silhouette score versus k-value, which is shown in Fig. 16. The clustering results using these k values are presented in Fig. 17.

First, in Fig. 16, we note the entire range of k values, $[3, 30]$ is shown. When automating the analysis, we only chose to run the experiments on $k \in [3, 12]$. During our initial (manual) experimental runs, we found that $k = 2$ would usually produce the result with the highest silhouette score, but the vortex region is visually indiscernible from a majority of the scan. As a result, we start with $k = 3$. After about $k = 12$, the question becomes one of whether improvement in detail is worth the

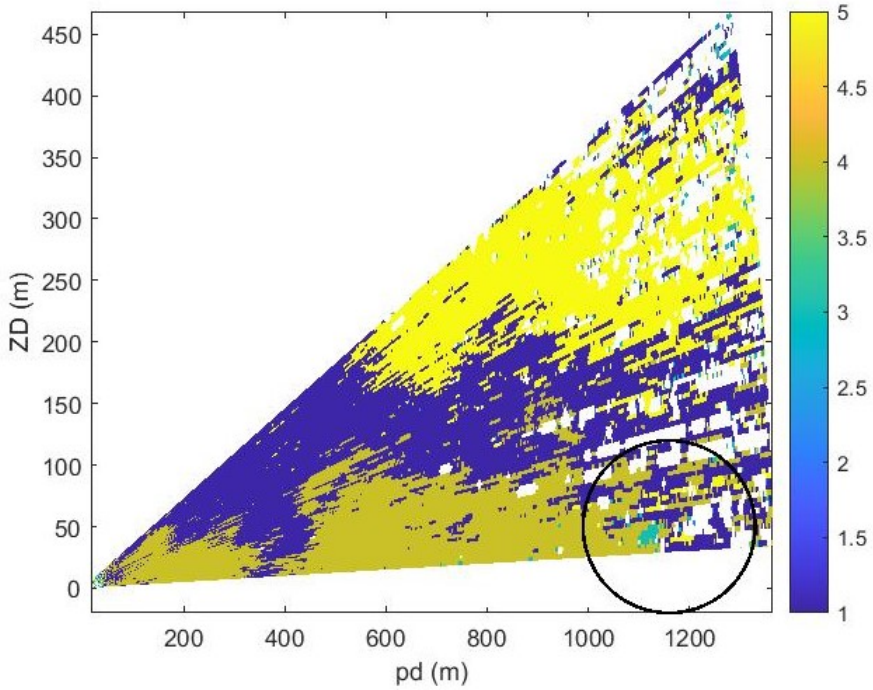


(a) $k = 2$

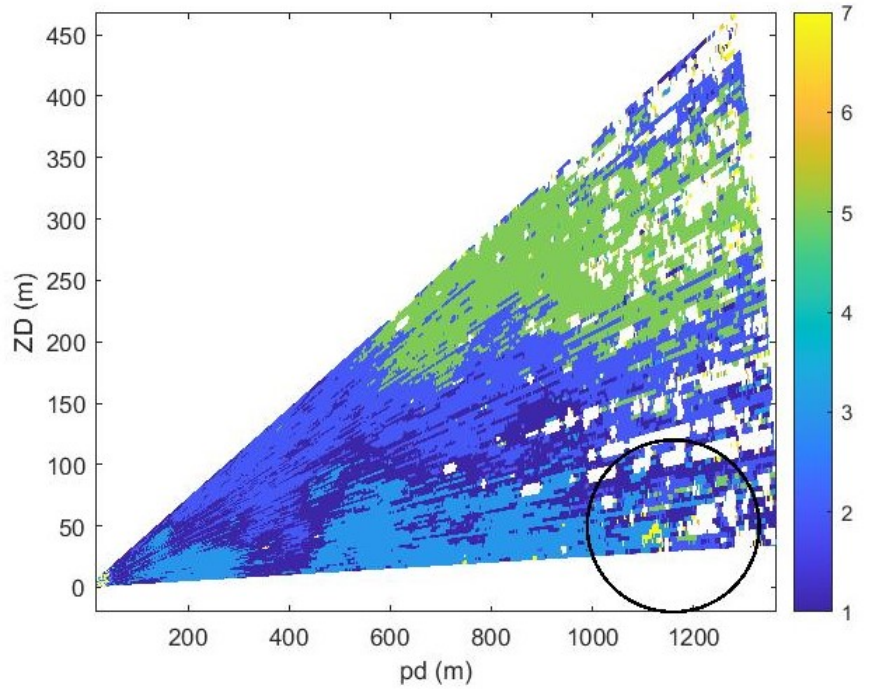


(b) $k = 3$**. For** $k = 2, 3$**, each part of the vortex region (inside the black circle) is indiscernible from portions of the atmosphere they were respectively clustered with.**

Fig. 17 k**-means clustering results using** $k \in \{2, 3, 5, 7, 9, 15\}$

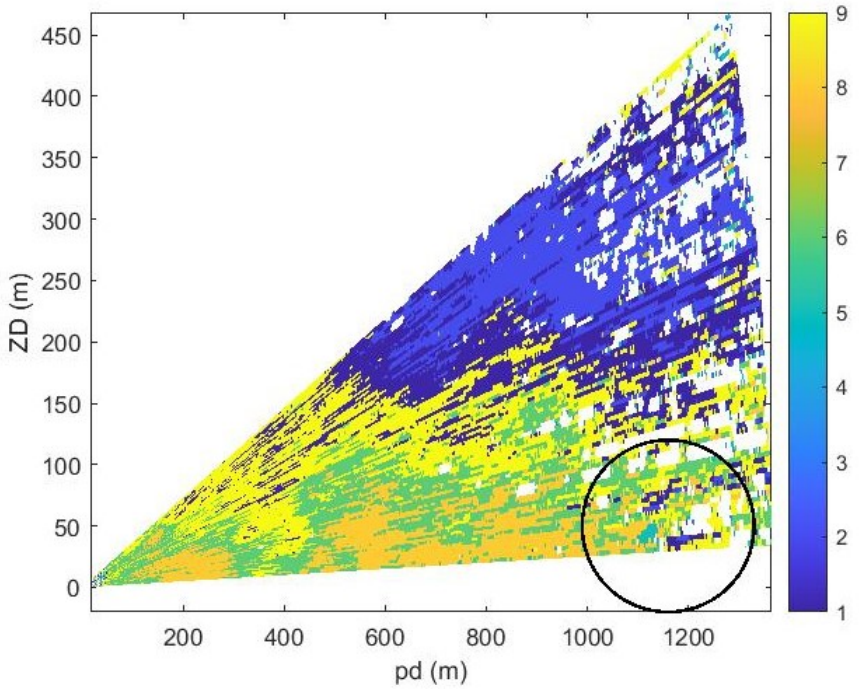


(c) $k = 5$**. The optimal** $k = 5$ **case shows a part of the vortex clearly differentiated from the remainder of the scan.**

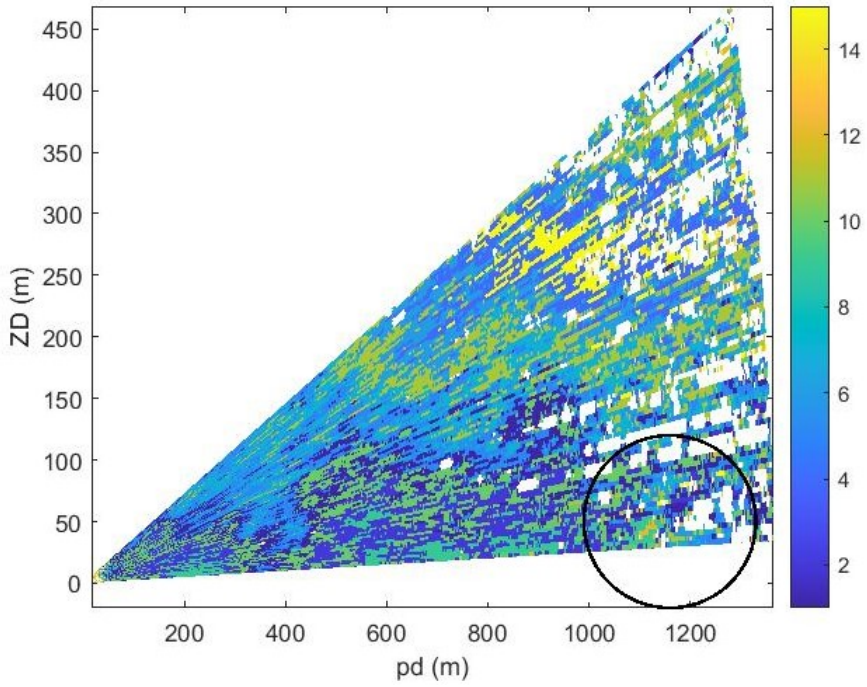


(d) $k = 7$

Fig. 17 k**-means clustering results using** $k \in \{2, 3, 5, 7, 9, 15\}$ **(continued)**



(e) $k = 9$**. For** $k = 7, 9$**, there is more segmentation of the scan, especially of the vortex region, but separating the regions becomes more difficult.**



(f) $k = 15$**. For** $k = 15$ **the cost of computation time becomes more than the benefit of added detail.**

Fig. 17 k**-means clustering results using** $k \in \{2, 3, 5, 7, 9, 15\}$ **(continued)**

computational time it takes to test the extra k values. In addition, we find the silhouette score does not improve for these extra k-values. From the optimization curve, $k = 5$ is clearly the optimal parameter. For $k = 7$ and 9, we see local maxima in the silhouette score; these are also the second and third highest silhouette values, respectively, so these are presented in Fig. 17 to compare with the optimal result.

Evaluating the clustering results, we see in Fig. 17 that for $k = 2$, part of the vortex region indiscernible from the lower altitude (<125 m) portion of the scan, while the other is indiscernible from the higher altitude (>125 m) atmosphere. For $k = 3$, the result is fairly similar, and the extra cluster does not work toward improving the isolation. $k = 5$ begins clearly highlighting the 0-RWS region from the scan in Fig. 15, while the remainder of the vortex still remains undifferentiated from the region between about 100 and 150 m, where the rest of Cluster 1 lies in the figure. For the $k = 7$ clustering, the scan continues to be segmented by altitude. The same 0-RWS vortex region is now more finely clustered. The clusters that compose the vortex region also seem speckled throughout the scan, which is surprising given the non-convexity of the result. An initial hypothesis is that this occurs because we are using the usual Euclidean metric to determine data point (dis)similarity, which is ideal for spatial relation, but we have not included the spatial coordinates of the data points for this analysis. (Different metrics for this analysis will be explored in the future.) The $k = 9$ and $k = 15$ cases produce greater detail, but we begin losing the general vortex structure, which is why during automated analysis we cut off our k value optimization at 12 clusters. While the vortex region is segmented further as we increase the number of clusters, providing more detail to its internal structure, the goal of isolating the overall vortex region seems to warrant the $k = 5$ result.

One item to address with using this brute-force optimization for k-means is that the optimization curve may not be the same for the given data set. An example of a different silhouette versus k graph is presented in Fig. 18. The reason for this change in the optimization curve is most likely due to the randomness involved in initial cluster centroid assignment. The biggest difference we can see in this graph from the previous is the optimal k has changed from 5 to 7. Many of the other silhouette scores for each k value have changed as well. Visually, the clusters display very little change. We still see that after $k = 10$, that the silhouette score does not improve.

Now, we discuss the DBSCAN algorithm applied to this scan. We present results for

$m \in \{10, 20, 50, 100, 600, 1000\}$, with $\epsilon = 0.017455$ (except for the $m = 10$ case, where $\epsilon = 0$) in Fig. 19. Note: This $\epsilon = 0$ result is also strange; we discuss a bit more below. While ϵ was allowed to change in the automated parameter selection phase, the algorithm always found this same ϵ for each m after $m = 10$. The $m = 10$ case was also a special case in that the tolerance reached 100% before the algorithm found a viable matching slope using the kDG process described in Section 3.3.1.

The main difference between the results is that the number of resulting clusters decreases with increasing m. Visually, in each of the results, the vortex region (in particular the 0-RWS area) is given by its own cluster. However, in the "perspective" of the DBSCAN algorithm these data points are actually noise. As m increases, so does the number of "un-clusterable" points. When $m = 600$, we start losing the defined edges of the vortex region that were displayed for the lower m values. When $m = 1000$, we have lost the vortex region.

We display the silhouette scores associated with each of the DBSCAN clustering results in Table 1. A near perfect silhouette score belongs to $m = 10$, but we take this result with a grain of salt until the physical interpretation of $\epsilon = 0$ is properly accounted in the context of the DBSCAN algorithm. $\epsilon = 0$ means that the neighborhood radius for some point p is 0: meaning a point q cannot be considered a neighbor of p unless $p = q$, and the condition for the minimum number of points within such a neighborhood of p (to determine whether p is a core point) can never be satisfied; so all points must be noise. However, there are clearly 200 clusters for a data set containing about 60,000 points. (It is more likely this $\epsilon = 0$ is just an estimate for a low value.) Regardless, we note for higher m values, there are large drops in the silhouette score, representing degrading cluster quality, which is exaggerated further when considering the $m = 600$ and $m = 1000$ case. It is noted from the SciKit-Learn documentation on clustering algorithms[34] that typically the silhouette scores for k-means will be higher than those for DBSCAN because of the convexity condition; hence, it does not make sense to compare the quality of the algorithms based on this silhouette metric.

At this point, we may easily extract the noisy data points identified by the DBSCAN algorithm, and isolate the vortex region plus a few specks of similarly-categorized atmosphere. However, can we improve the quality of the extracted vortex region?

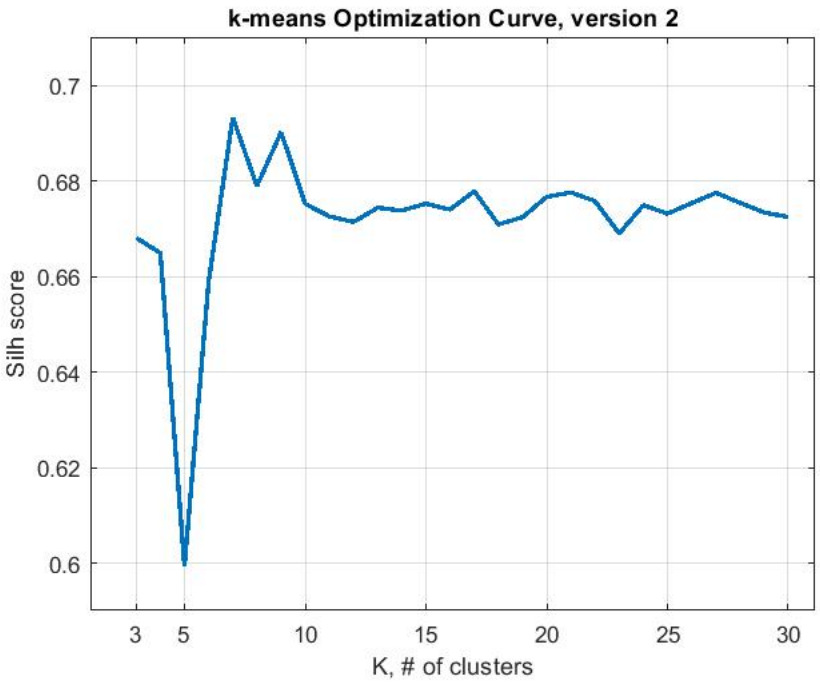
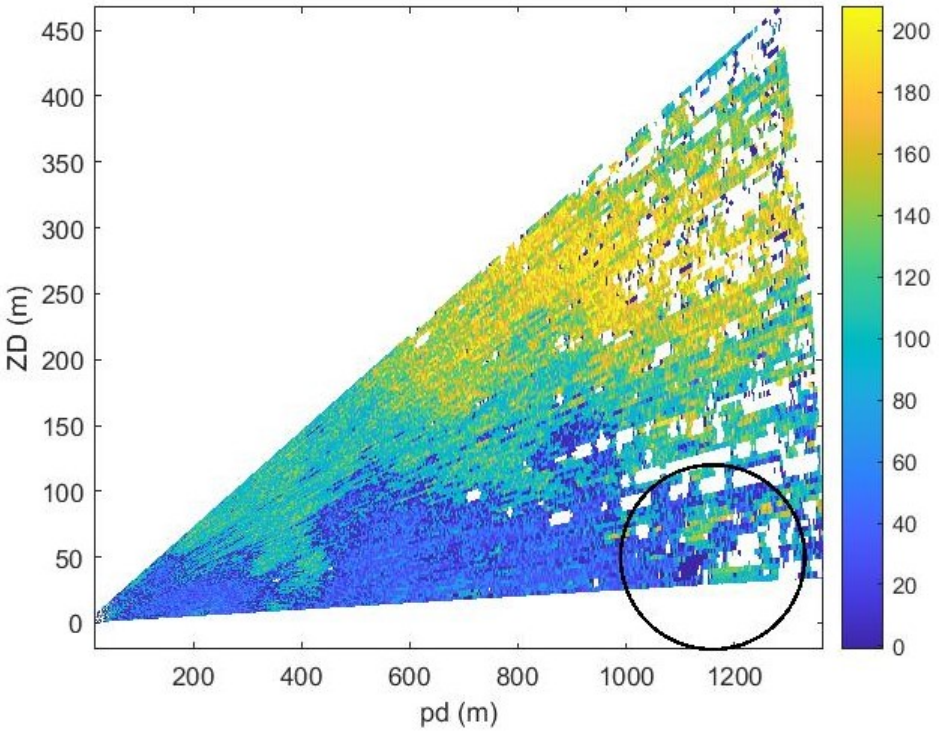


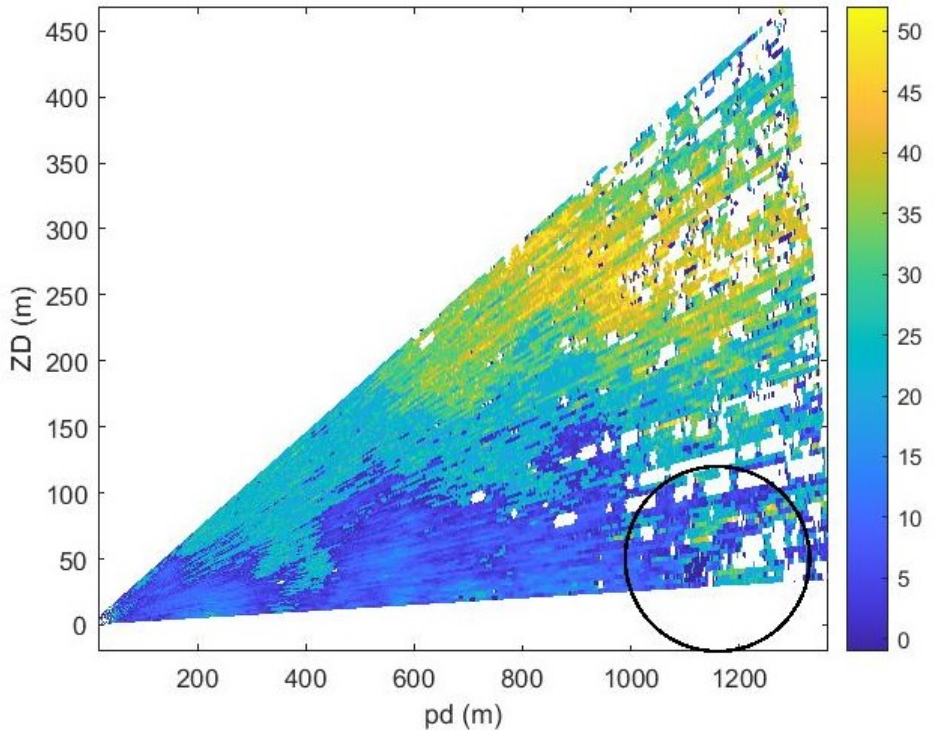
Fig. 18 Another optimization curve for the noisy vortex scan. Notice the optimal value has shifted from $k = 5$ to $k = 7$. The behavior of the graph after about $k = 10$ matches that of the previous optimization curve, in that the silhouette score does not improve as k increases.

Table 1 Example of silhouette scores for the DBSCAN result for the NaN-noise scan

m	ϵ	**silh. score**
10	0	0.9597
20	0.017455	0.6191
50	0.017455	0.6553
100	0.017455	0.6455
600	0.017455	0.2195
1000	0.017455	-0.2996

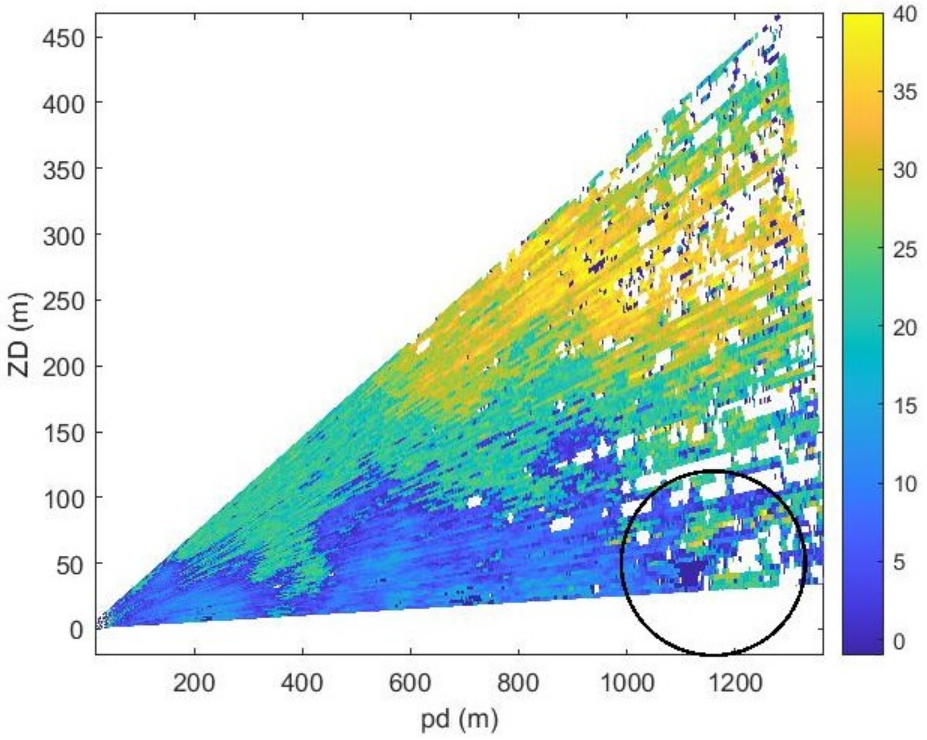


(a) $m = 10$**,** $\epsilon = 0$**. 200 clusters. The vortex region is categorized as noise.**

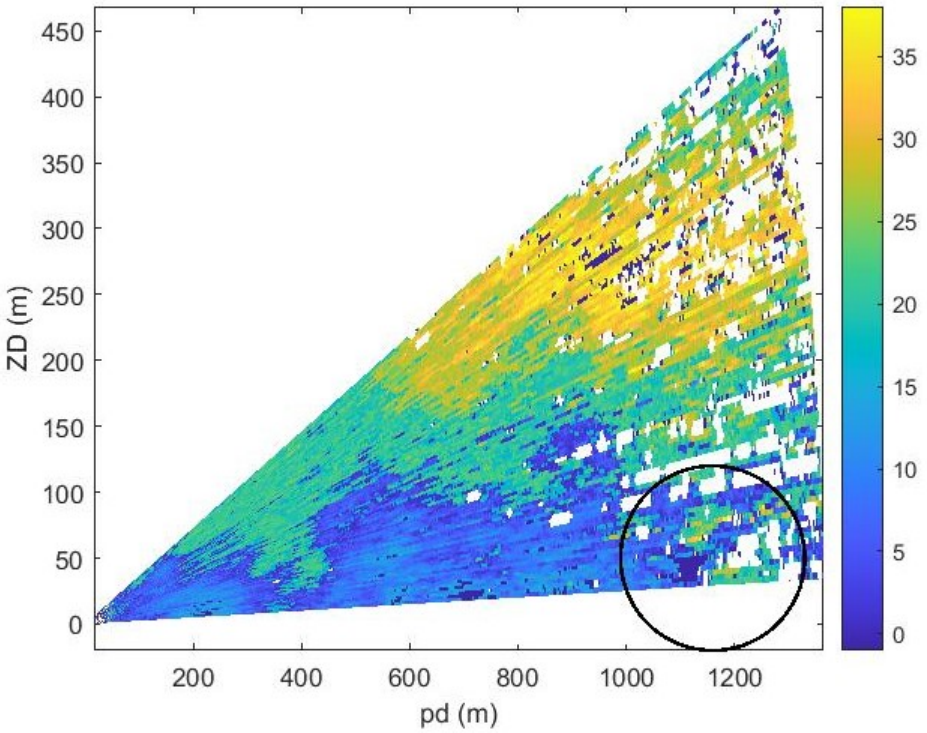


(b) $m = 20$**,** $\epsilon = 0.017455$**. 50 clusters. Small change in vortex region from the** $m = 10$ **case.**

Fig. 19 DBSCAN clustering results for $m \in \{10, 20, 50, 100, 600, 1000\}$**. As** m **increases, the number of data points categorized as noise also increases.**

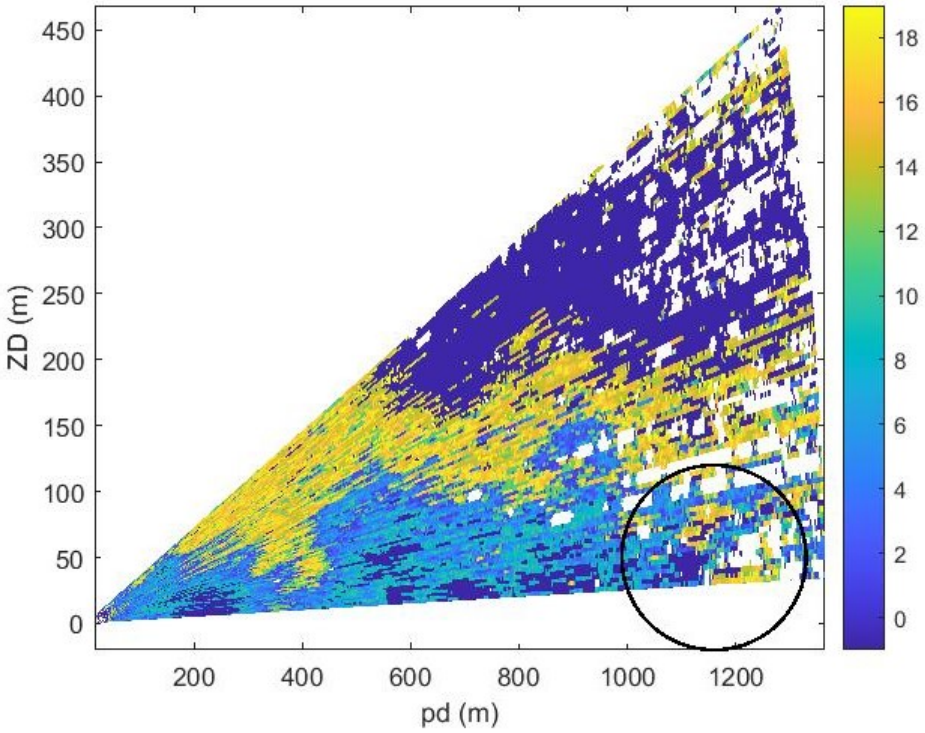


(c) $m = 50$**,** $\epsilon = 0.017455$**. 40 clusters.**

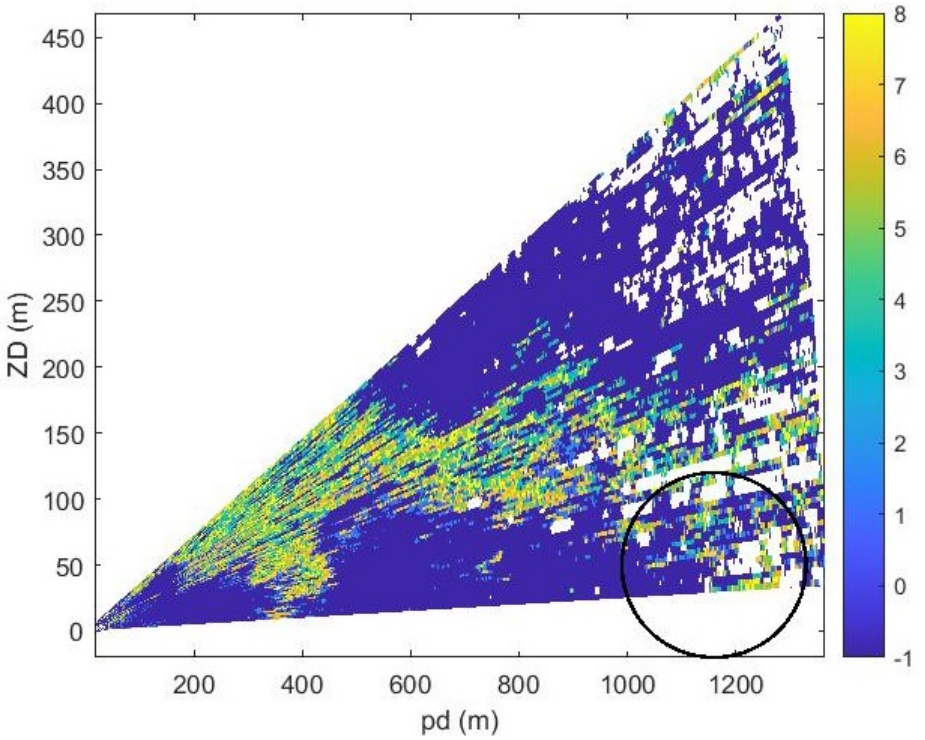


(d) $m = 100$**,** $\epsilon = 0.017455$**. 38 clusters.**

Fig. 19 DBSCAN clustering results for $m \in \{10, 20, 50, 100, 600, 1000\}$**. As** m **increases, the number of data points categorized as noise also increases. (continued)**



(e) $m = 600$**,** $\epsilon = 0.017455$**. 19 clusters.**



(f) $m = 1000$**,** $\epsilon = 0.017455$**. 8 clusters.**

Fig. 19 DBSCAN clustering results for $m \in \{10, 20, 50, 100, 600, 1000\}$**. As** m **increases, the number of data points categorized as noise also increases. (continued)**

4.2 Vortex Isolation

The LCAP was extended by the minute observation that the DBSCAN algorithm characterizes the vortex region as noise for smaller ϵ values. We can further perform data cleaning (filling in missing data and smoothing) to gain better detail. Because DBSCAN separates noise into its own category, we are able to extract the corresponding data points from the scan and isolate much of the vortex region. The reason we choose DBSCAN over k-means for the vortex isolation is because of its ability to work on non-convex clusters.

One of the initial issues we faced with using DBSCAN on a LIDAR scan is that the vortex region is characterized by the entire span of clusters produced by the algorithm. See Figs. 9, 12, and 14 for examples. We want the vortex region to be clustered separately from the rest of the scan, ideally in its own cluster. In this manner, we could simply isolate the vortex region using that particular cluster's label. Initially, to combat this overclustering of the vortex region, one could isolate a few clusters selected uniformly from the entire span; however, this method is likely to produce just as much noise as actual signal so as to render the vortex indiscernible. For some scans, it was noted that while the entire span of clusters composes the entire vortex region, a decent majority of the area is categorized as noise. Lowering ϵ allowed us to further increase the number of vortex points categorized as noise, thereby allowing us to isolate more of the vortex region.

4.2.1 NaN-Noise Vortex (Isolated)

We begin with the NaN-noise vortex scan from Section 4.1.3. First, to help better discern the region, we use RWSeff instead of RWS. We can see that by removing the mean wind, the vortex region visually stands out, and cleaning the scan further highlights the region. The original scan using RWS is presented in Fig. 15. We display the RWSeff and its cleaned version in Fig. 20. In the cleaned scan, there are still gaps which are due to the small size of the averaging window, but for experimentation we wanted to make sure to at least clean the vortex region.

We apply DBSCAN to both the unclean and cleaned scans, select the "noise cluster" from this result, and present the isolated regions in Fig. 21.

Using either the clean or unclean scans results in great isolation of the vortex region. A clean scan allows more of the vortex region to stand out after isolation, but it

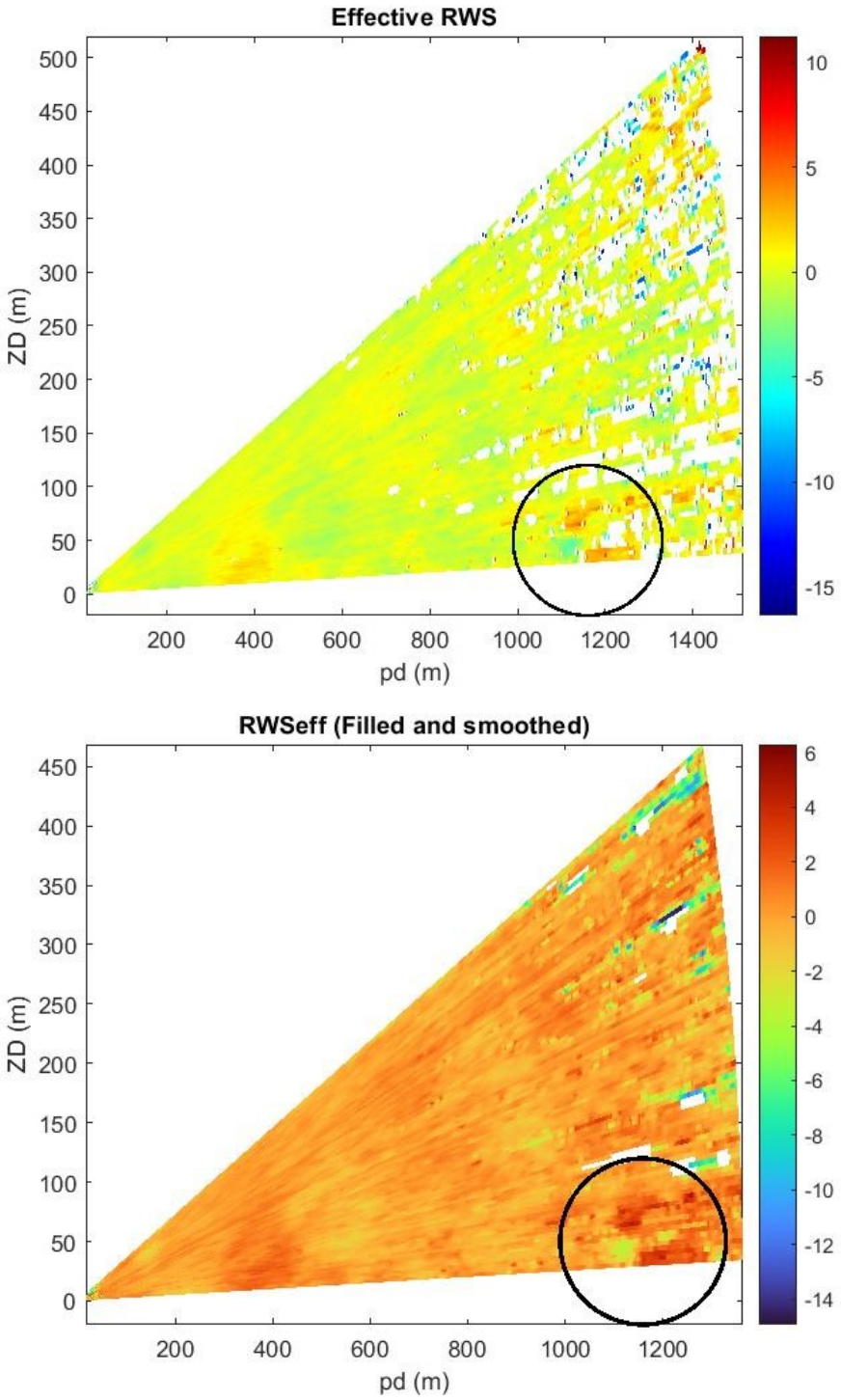


Fig. 20 (Top): Scan with RWSeff. (Bottom): Same scan with cleaning. There are still gaps in the scan because the window size for the ray-wise moving average was too small; however, we made sure to at least fill in the gaps for the vortex region.

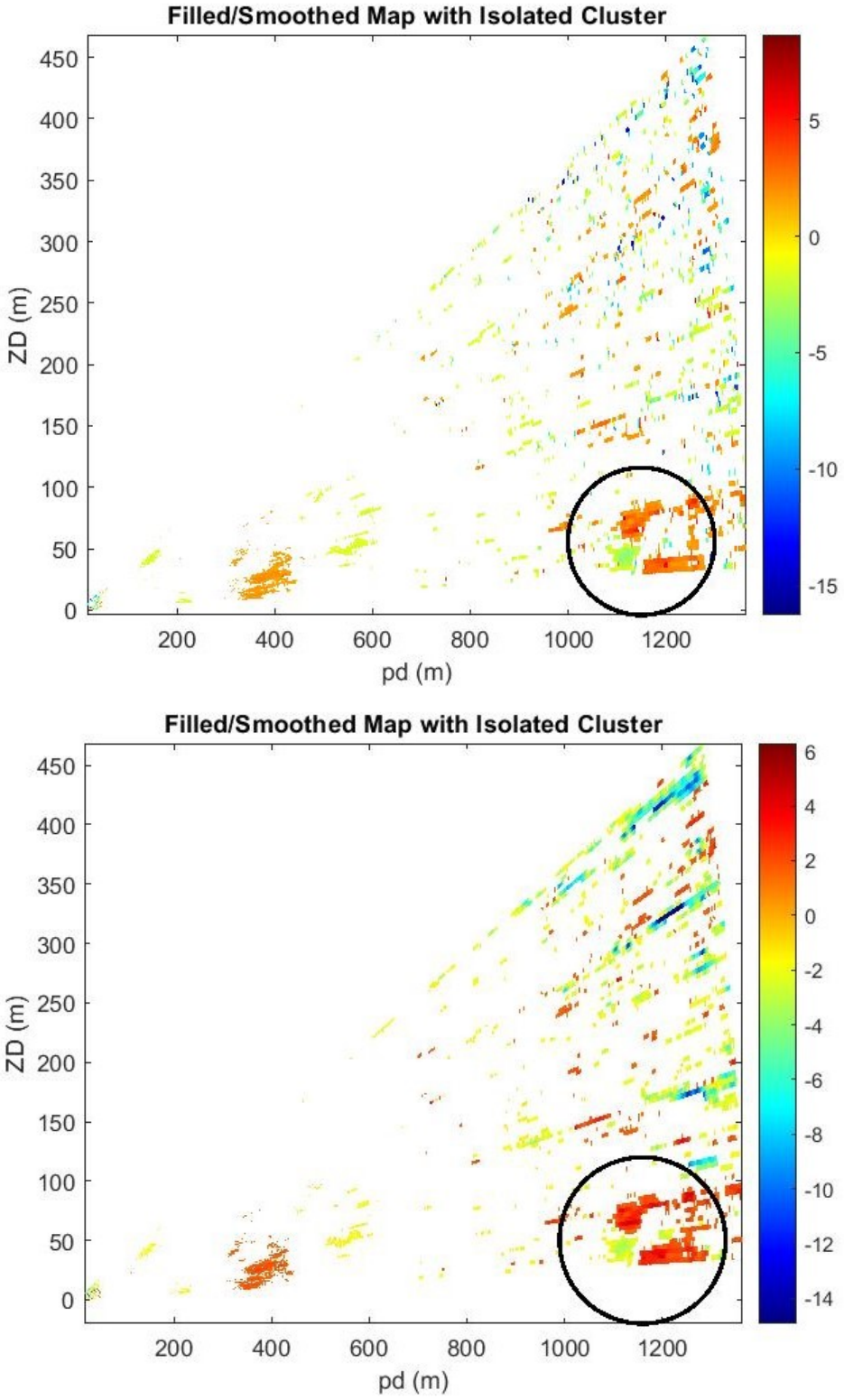


Fig. 21 Vortex isolation result with noisy (uncleaned) vortex scan (top) and cleaned scan (bottom)

Table 2 Example of silhouette scores for the DBSCAN result for the cleaned NaN-noise scan

m	ϵ	**silh. score**
10	0.0059941	0.4274
20	0.0081834	0.3820
50	0.017543	0.5792
100	0.031368	0.8905
600	0.06164	0.7858
1000	0.093722	0.7899

also allows for more extraneous points than the unclean scan. Compared to the previous result of using only RWS, we are able to isolate a larger region via the "noise cluster" from DBSCAN.

For comparison with Table 1, Table 2 shows the silhouette scores for the DBSCAN algorithm applied to the cleaned scans with $m \in \{10, 20, 50, 100, 600, 1000\}$. Again, ϵ was allowed to vary with the parameter selection algorithm. We see that the silhouette score displays almost opposite behavior to those for the uncleaned scans; that is, the silhouette score increases with increasing m, up to a certain point. This seems to warrant an attempt to optimize the parameter selection by using the silhouette score for which we then face a mixed-integer optimization problem; this may be a future avenue for more robust clustering. An example clustering result, using the $m = 100$ case, is presented in Fig. 22. All cases from Table 2 yield similar clustering, namely one cluster and the remaining data points categorized as noise. This clustering represents what we desire in terms of isolating the vortex, but also computation time; more clusters mean longer processing times.

4.2.2 Series of Scans: Initial Spatiotemporal Tracking of Vortices

We now apply the full methodology of data filling, smoothing, clustering, and isolating to a series of scans which show a vortex region changing over time. The results are presented in Fig. 23. Similar to the previous results, the uncleaned isolation results displays slightly less extraneous data points than the cleaned version, but we also lose portions of the vortex region.

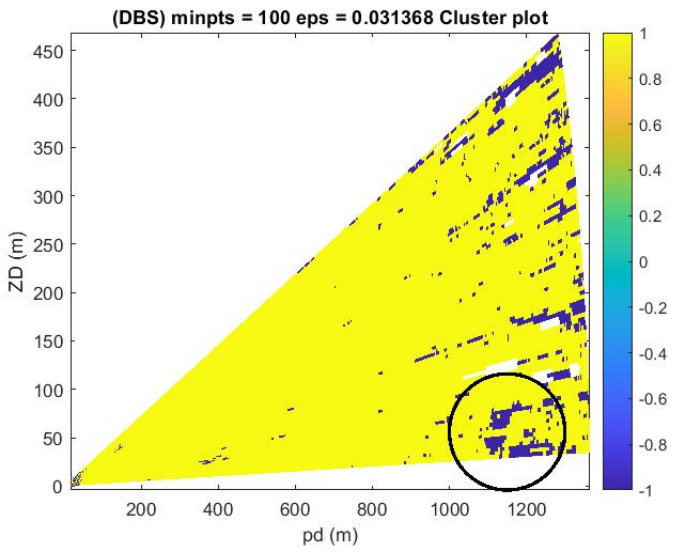
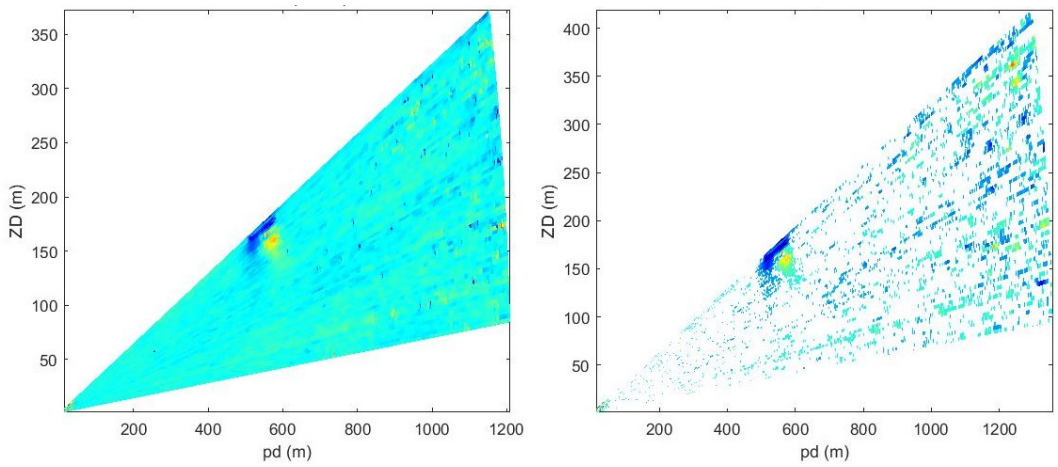
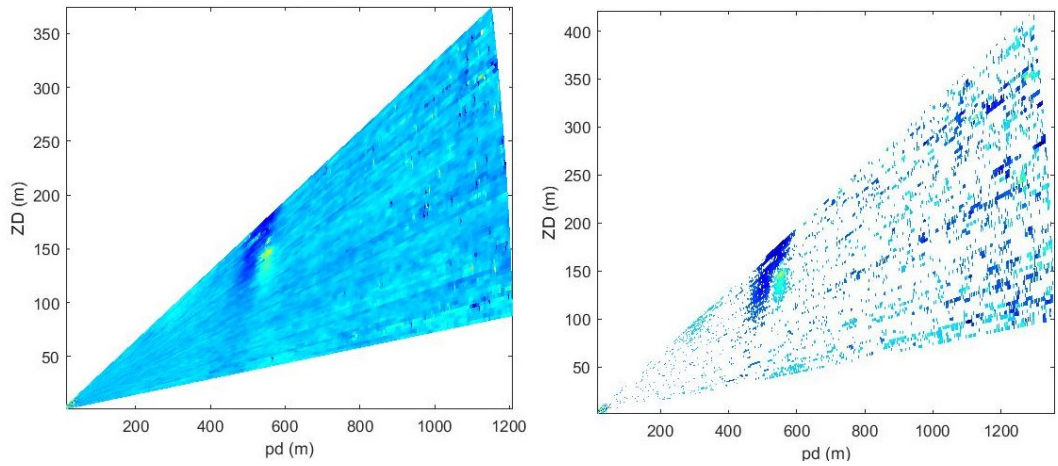


Fig. 22 Example DBSCAN clustering for the cleaned NaN-noise scan using $m = 100$ and (automatically selected) $\epsilon = 0.031368$. All cases from Table 2 yield similar clustering, namely one cluster and the remaining data points categorized as noise.

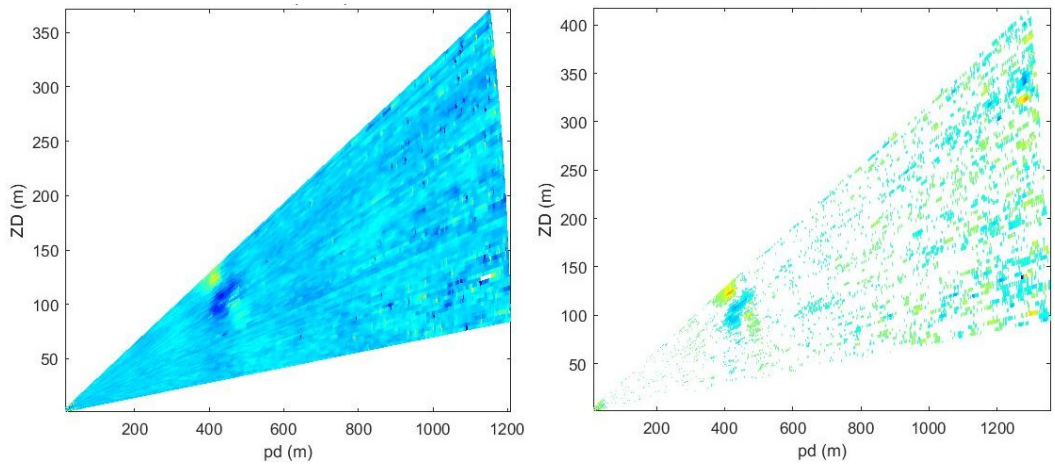


(a) Vortex tracking series scan 1

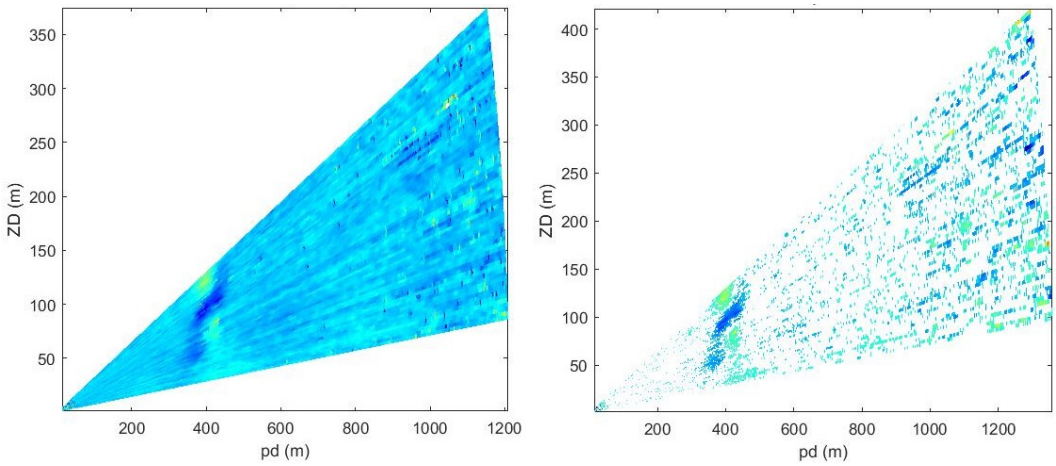
Fig. 23 Isolating the vortex region from a time series of scans for basic spatiotemporal tracking



(b) Vortex tracking series scan 2

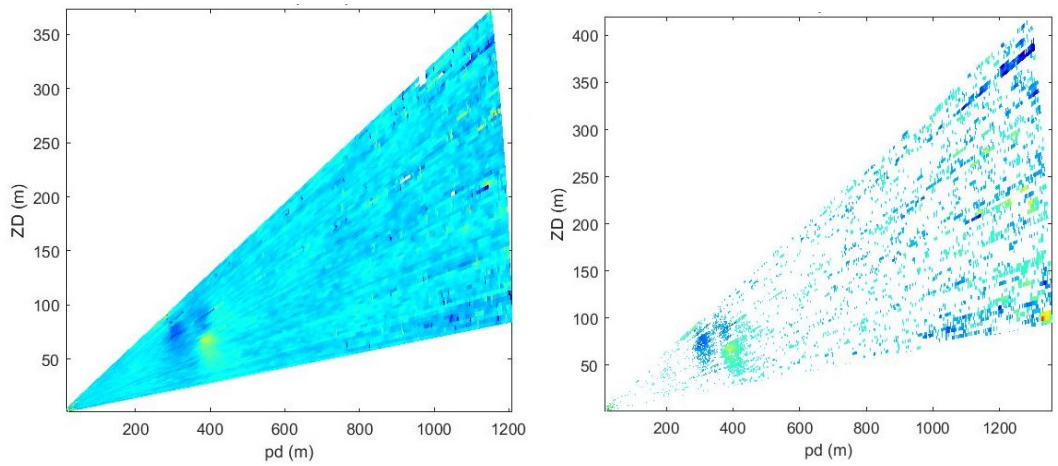


(c) Vortex tracking series scan 3

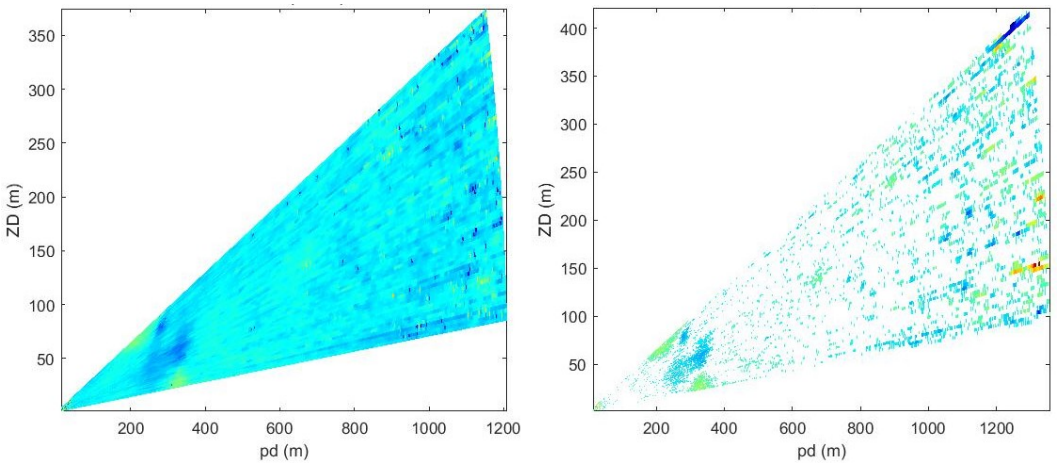


(d) Vortex tracking series scan 4

Fig. 23 Isolating the vortex region from a time series of scans for basic spatiotemporal tracking (continued)



(e) Vortex tracking series scan 5



(f) Vortex tracking series scan 6

Fig. 23 Isolating the vortex region from a time series of scans for basic spatiotemporal tracking (continued)

We note that for these results and those in Section 4.2.1 we did not use the automatically selected value for ϵ. After running a number of experiments not presented here, we found that a low value ($\epsilon < 0.005$) generally performed well in categorizing the vortex region, so to speed up the algorithm, we fixed the value of $\epsilon = 0.001$. As a caution, reducing ϵ further results in a higher number of clusters, at times on the order of a few thousand. While this high number of clusters may be of use when examining the scan in depth, after a certain point the clustering becomes useless as individual points are categorized as their own clusters, which in turn causes more points to be classified as noise. The minimum number of neighboring points, m,

was held fixed at 10. The lowest value we could use was $m = 3$, but a difference of seven minimum neighboring points does not affect the algorithm much in terms of its runtime or the number of resulting clusters. Increasing m only results in more points being classified as noise, therefore increasing the number of extraneous points in the final isolation.

The results are promising in their ability to track a vortex through time and space. Further processing is required to eliminate the extraneous portions of the scan farther away from the LIDAR system. The current run time of these algorithms to upload a scan, process the relevant features into a cluster matrix, run DBSCAN, display, and save the results has been on the order of less than 10 s on a laptop. We hope that as this project grows, that this time complexity is further reduced through improved algorithms, design, and potentially hardware acceleration, so that the system will be able to handle real-time processing in the field.

5. Conclusion

In this research we present our initial study for applying machine learning to study atmospheric wake vortices. We review previous work on the vortex characterization and recent studies of machine learning used for atmospheric research, provide a self-contained description of the algorithms and methods used, introduce the LCAP, and display our results from simple applications of the k-means and DBSCAN algorithms. We hope this serves as a foundation for further software development for atmospheric research.

6. Future Work

This study provides numerous avenues to explore in the future. The explorations mentioned previously include (1) determining the cost/benefits and viability of using a 2-D SGF for data smoothing, (2) a more robust method for selecting the ϵ parameter for the DBSCAN algorithm, (3) understanding how different metrics/measures of similarity affect the clustering (finding the most ideal), particularly for k-means, and (4) performing mixed-integer optimization in find both m and ϵ for the DBSCAN algorithm. The development of the LCAP allows the researcher to adjust and test individual pieces of code, which in turn allows for a more directed exploration into improved algorithms and methods. There are parameters and adjustments that must be made within the code, for which we aim to make the

program more user-friendly (e.g., graphical user interface), but we will also weigh these benefits with the cost of computation.

The LCAP is currently designed to operate on single Doppler LIDAR RHI scans, but we also plan to expand to other scan geometries and multi-LIDAR data sets. The next steps in testing the strength of these algorithms is to see whether they pick up on rolls off of buildings and mountains, (or any turbulent features in general.) One of the main issues affecting the spatiotemporal tracking of coherent turbulence is the ever-evolving topology of the structures. There has been great headway in tracking rigid objects, however, not so much in the way of fluid flow. The LCAP currently operates on single scans, but if we can efficiently run the algorithms on a “stream" of evolving data (e.g., video), then we can take a further step toward tracking atmospheric turbulence through time and space. We also avoid the use of CFD to inform our machine-learning algorithms, but as noted in the introduction, there is still work to be done with the CFD itself.

As mentioned in the results section, we are still researching more refined techniques to isolate the entire vortex regions from any scan; we do not want the methodology to only work on certain scans. Even though we are able to isolate the vortex, there is still cleaning to be done; perhaps via extracting the vortex signal through a method like spatial Fourier analysis, or convolutional filters. We do not attempt to compare the performance qualities between the k-means and DBSCAN algorithms using the silhouette score metric. A separate data science-oriented research could look at developing methods for comparing clustering quality of algorithms using different clustering concepts. We also intend to further explore the clustering algorithms in tandem to see what we can learn about the structure of wingtip wake vortices.

The current algorithms were tested on a laptop, so there is work to be done in examining the potentials of hardware acceleration. The larger goal is to assimilate the LCAP with other vortex characterization programs and ultimately use this full program in real time, which in itself will involve a great amount of software development. After the machine-learning models become viable for field testing, we aim for the software architecture to allow for “simple" re-training and re-evaluation.

This study provides reasonable groundwork for implementing machine learning in atmospheric research. The idea is in its infancy, but progress is expected as we expand our knowledge and improve our capabilities in data science.

7. References

1. Green SI. Wing tip vortices. In: Green SI, editor. Fluid vortices, fluid mechanics and its applications; Springer, Dordrecht; 1995. p. 427–469.

2. Giuni M. Formation and early development of wingtip vortices [PhD thesis]. [Glassgow (Scotland)]: University of Glasgow; 2013.

3. Perry RB, Hinton DA, Stuever RA. NASA wake vortex research for aircraft spacing. NASA Langley Research Center; 1996.

4. Robins R, Delisi D, Hinton D. NWRA AVOSS wake vortex prediction algorithm version 3.1.1. 2002. NASA/CR-2002-211746.

5. Cariou J-P, Thobois T. Collecting large wake vortex data with operational Doppler LIDARs for operational implementing of new wake turbulence regulations at major airports. 19th Coherent Laser Radar Conference; 2018.

6. Hallock JN, Holzäpfel F. A review of recent wake vortex research for increasing airport capacity. Progress in Aerospace Sciences. 2018;98:27–36.

7. Lovell D, editor. Military vortices. NATO; 2003 Mar. RTO-MP-069-I.

8. Ray R, Cobleigh B, Vachon M, John CS. Flight test techniques used to evaluate performance benefits during formation flight. In: AIAA Atmospheric Flight Mechanics Conference and Exhibit; 2002 Aug 5–8. https://doi.org/10.2514/6.2002-4492.

9. Larson G, Schkolnik G. Autonomous Formation Flight. NAS4-00041 TO-104. MIT Course 16.886: Air Transportation Systems Architecting. 2004 Spring.

10. Ning SA. Aircraft drag reduction through extended formation flight [PhD thesis]. [Stanford (CA)]: Stanford University; 2011.

11. Clifton A, Boquet M, Burin Des Roziers E, Westerhellweg A, Hofsass M, Klaas T, Vogstad K, Clive P, Harris M, Wylie S, Osler E, Banta B, Choukulkar A, Lundquist J, Aitken M. Remote sensing of complex flows by Doppler wind lidar: Issues and preliminary recommendations. National Renewable Energy Laboratory, US Department of Energy; 2015. Report No.: NREL/TP-5000-64634.

12. Kopp F, Rahm S, Smalikho I. Characterization of aircraft wake vortices by 2 micrometer pulsed Doppler LIDAR. Journal of Atmospheric and Oceanic Technology. 2004;21(2):194–206.

13. Wu S, Zhai X, Liu B. Aircraft wake vortex and turbulence measurement under near-ground effect using coherent Doppler LIDAR. Optics Express. 2019;27(2):1142–1163.

14. Michel DT, Dolfi-Bouteyre A, Goular D, Augère B, Planchat C, Fleury D, Lombard L, Valla M, Besson C. Onboard wake vortex localization with a coherent 1.5 μm Doppler LIDAR for aircraft in formation flight configuration. Optics Express. 2020;28(10):14374–14385.

15. Hastie T, Tibshirani R, Friedman JF. The elements of statistical learning. 2nd ed. Springer; 2009. (Springer Series in Statistics).

16. Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016. (Adaptive Computation and Machine Learning Series).

17. Pan W, Yingjie D, Qiang Z, Jiahao T, Jun Z. Deep learning for aircraft wake vortex identification. IOP Conference Series: Materials Science and Engineering. 2019;685:012015.

18. Redmon J, Divvala SK, Girshick RB, Farhadi A. You only look once: unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016;779–788.

19. Pan W, Wu Z, Zhang X. Identification of aircraft wake vortex based on SVM. Mathematical Problems in Engineering. 2020; Article ID 9314164. p 8. https://doi.org/10.1155/2020/9314164.

20. Wang J, Guo L, Wang Y, Deng L, Wang F, Li T. A vortex identification method based on extreme learning machine. International Journal of Aerospace Engineering. 2020; Article ID 8865001. p 10.

21. Cheliotis I, Dieudonné E, Delbarre H, Sokolov A, Dmitriev E, Augustin P, Fourmentin M. Detecting turbulent structures on single Doppler LIDAR large datasets: an automated classification method for horizontal scans. Atmospheric Measurement Techniques. 2020;13(12):6579–6592.

22. Liu JNK, Kwong KM, Chan PW. Chaotic oscillatory-based neural network for wind shear and turbulence forecast with lidar data. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 2012;42(6):1412–1423.

23. Aihara K, Takabe T, Toyoda M. Chaotic neural networks. Physics Letters A. 1990;144(6):333–340.

24. Bhargav S, Pawar M. A review of clustering methods forming non-convex clusters with missing and noisy data. International Journal of Computer Sciences and Engineering. 2016;4:39–44.

25. Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining; (KDD'96) AAAI Press; 1996. p. 226–231.

26. Maulik R, Rao V, Renganathan SA, Letizia S, Iungo GV. Cluster analysis of wind turbine wakes measured through a scanning Doppler wind LIDAR. In: AIAA Scitech 2021 Forum. AIAA 2021-1181. https://doi.org/10.2514/6.2021-1181.

27. Arthur D, Vassilvitskii S. K-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms; (SODA '07) Society for Industrial and Applied Mathematics; 2007. p. 1027–1035.

28. MathWorks. Matlab r2020b documentation, version 9.9.0.1467703. The MathWorks Inc.; 2020.

29. Kaufman L, Rousseeuw PJ. Finding groups in data: an introduction to cluster analysis. Wiley; 2005. (Wiley Series in Probability and Statistics).

30. Boccippio DJ. A diagnostic analysis of the VVP single-Doppler retrieval technique. Journal of Atmospheric and Oceanic Technology. 1994;12:230–248.

31. Schafer RW. What is a Savitzky-Golay filter? [lecture notes]. IEEE Signal Processing Magazine. 2011;28(4):111–117.

32. Trefethen LN, Bau D, III. Numerical linear algebra. Society for Industrial and Applied Mathematics; 1997.

33. Suhling M, Arigovindan M, Hunziker P, Unser M. Multiresolution moment filters: theory and applications. IEEE Transactions on Image Processing. 2004;13(4):484–495.

34. Scikit-Learn. Scikit-learn: machine learning in Python. User Guide: 2.3 Clustering. [accessed 2022 Jan 19]. https://scikit-learn.org/stable/modules/clustering.html.

Appendix. Basic Results for k-means and DBSCAN

A.1 k-Means Clusters Are Convex

The following proof is based on Schmidt.[1] We show that clusters resulting from the k-means algorithm are convex.

Recall, the definition of a convex set given in Definition 1 in Section 3.1.1.

Suppose we have a data set $X = \{x_1, x_2, ..., x_n\}$ that is partitioned via the k-means clustering algorithm into clusters $C_1, ..., C_k$, which have cluster centers $m_1, ..., m_k$, respectively. We want to show C_i is convex. Consider two point $x_a, x_b \in C_i$. By the k-means algorithm, these points are closer to their cluster center, m_i, than to any other cluster center. Mathematically,

$$\|x_a - m_i\| \leq \|x_a - m_j\| \ \forall \ j = 1, 2, ..., k$$

$$\|x_b - m_i\| \leq \|x_b - m_j\| \ \forall \ j = 1, 2, ..., k$$

We now consider a point $x_c = \lambda x_a + (1 - \lambda)x_b$ for some $\lambda \in [0, 1]$. That is, we have selected a point on the line segment connecting points x_a and x_b. We show x_c also lies within the cluster set C_i. Note: x_c need not be any data point in X; if it were and we showed that it lies as far to the cluster center m_i as x_a or x_b in C_i, then surely this shows C_i is convex.

$$\begin{aligned}
\|x_c - m_i\| &= \|(\lambda x_a + (1 - \lambda)x_b) - m_i\| && \text{(A-1)} \\
&= \|(\lambda x_a + (1 - \lambda)x_b) - (\lambda m_i + (1 - \lambda)m_i)\| && \text{(A-2)} \\
&= \|\lambda(x_a - m_i) + (1 - \lambda)(x_b - m_i)\| && \text{(A-3)} \\
&\leq \lambda \|x_a - m_i\| + (1 - \lambda) \|x_b - m_i\| && \text{(A-4)} \\
&\leq \lambda \|x_a - m_i\| + (1 - \lambda) \|x_a - m_i\| && \text{(A-5)} \\
&= \|x_a - m_i\| && \text{(A-6)}
\end{aligned}$$

where from line (A-3) to (A-4) we have the triangle inequality and from line (A-4) to (A-5) we assume without loss of generality that $\|x_b - m_i\| \leq \|x_a - m_i\|$. Since, $\lambda \in [0, 1]$, $(1 - \lambda) \in [0, 1]$, and it follows that $(1 - \lambda) \|x_b - m_i\| \leq (1 - \lambda) \|x_a - m_i\|$. It follows that, x_c is no farther away from m_i as x_a is, and therefore lies within C_i.

[1] Schmidt M. Density-based clustering. CPSC 340: Machine learning and data mining. University of British Columbia; 2017 Fall. https://www.cs.ubc.ca/ schmidtm/Courses/340-F17/L9.pdf.

A.2 Fundamental Concepts for Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

The following is a condensed version of the presentation of the original paper on DBSCAN by Ester et al.[2] For each of these definitions, we are speaking in the context of a subset X of a real vector space $\mathbb{R}^n$, with a defined metric function, d. That is, we are speaking in the context of a metric space. The subset X in question is our data set.

Definition 2 *Let $\epsilon > 0$. We define the ϵ-**neighborhood** (ϵ-n-hood) of a point $p \in X$, $N_\epsilon(p)$ as the set of all points whose distance from p is no greater than ϵ. That is,*

$$N_\epsilon(p) = \{q \in X : d(p, q) \leq \epsilon\}.$$

Further, we may require a minimum number of points to be within a ϵ-neighborhood of a point. We will call this minimum number, M.

Definition 3 *A point $q \in X$ is **directly-density reachable** from another point $p \in X$ with respect to ϵ and M if:*

1. *$q \in N_\epsilon(p)$ and*
2. *$|N_\epsilon(p)| \geq M$, where $|.|$ yields the cardinality of a set.*

Definition 4 *A point $q \in X$ is **density-reachable** from $p \in X$ (with respect to ϵ and M) if there is a chain of points $p_1, ..., p_n$ with $p_1 = p$ and $p_n = q$ such that p_{i+1} is directly-reachable from p_i.*

Definition 5 *A point $q \in X$ is **density-connected** to a point $p \in X$ (with respect to ϵ and M) if there is a point r such that both p and q are density-reachable from r.*

Definition 6 *Allowing X to be a database of points, now, we define a non-empty $C \subset X$ to be a **cluster** when it satisfies the following:*

[2]Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clustering in large spatial databases with noise. In: Proceedings of the Section International Conference on Knowledge Discovery and Data Mining (KDD'96). AAAI Press; 1996. p. 226–231.

1. *For all $p, q \in X$, if $p \in C$ and q is density-reachable from p, then $q \in C$. (Maximality)*

2. *For all $p, q \in C$, p is density-connected to q. (Connectivity)*

Definition 7 *Letting $C_1, ..., C_k$ to be clusters of the database X, we define the **noise**, S, as the set of points in X not belonging to C_i for any $i = 1, ..., k$. That is,*

$$S = \{p \in X : p \notin C_i \ \forall i\}.$$

List of Symbols, Abbreviations, and Acronyms

1-D	one-dimensional
2-D	two-dimensional
3-D	three-dimensional
AFF	Autonomous Formation Flight
ARL	Army Research Laboratory
CFD	computational fluid dynamics
CNN	convolutional neural network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
ELM	extreme learning machine
GPS	global positioning system
kDG	k distance graph
kNN	k-nearest neighbor
kNPG	k nearest points graph
LCAP	LIDAR Cluster Analysis Program
MIT	Massachusetts Institute of Technology
NaN	not a number
NASA	National Aeronautics and Space Administration
PCA	principal component analysis
RHI	range-height indicator
ROI	region of interest
RWS	radial wind speed

RWSeff	effective radial wind speed
SGF	Savitzky-Golay filter
SNR	signal-to-noise ratio
SVM	support vector machine
UAV	unmanned aerial vehicle
VVP	volume velocity processing
YOLO	You Only Look Once

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLD DCI TECH LIB
2 (PDF)	DEVCOM ARL FCDD RLC E P KUMAR FCDD RLC ES C WILLIAMSON