Mean shift, mode seeking, and clustering implementation- Circular Linear Data

Osama Al Sheikh Ali

I. Introduction

Clustering is a central idea in unsupervised learning because it helps us discover natural groups in data without using any labels. Most standard clustering methods work on data that live in ordinary Euclidean space, where distances behave normally. However, many real problems include circular or angular measurements, and angles do not behave like linear values. For example, 0 and 2π represent the same direction, but a linear method would treat them as far apart. Because of this, classic clustering algorithms often fail when a circular component is present.

To address this, the work in this project is inspired by the original mean shift algorithm introduced by Cheng [1]. Mean shift is an iterative procedure that moves points toward regions where the data density is higher. At each step, a point is replaced by a weighted average of its neighbours, defined through a kernel function. This process can be seen as a mode-seeking behaviour: every point "climbs" the underlying density surface until it reaches a local maximum. In Cheng's formulation, the update rule is

$$m(x) = \frac{\sum_{s\in S} K(s-x)\, w(s)\, s}{\sum_{s\in S} K(s-x)\, w(s)}. \tag{1}$$

and the difference $m(x)-x$ is the actual "mean shift". Cheng also showed that, for Gaussian kernels, this update is equivalent to taking a gradient ascent step on a smooth density estimate, which helps explain why the method converges reliably.

Cheng additionally introduced the idea of a "blurring" process, where the entire dataset is repeatedly transformed by applying mean shift to every data point. Points that end up at the same location (a mode) are treated as belonging to the same cluster. This makes mean shift a deterministic and non-parametric approach, without the need for preset cluster counts.

In this project, I adapt these ideas to circular–linear data, where each point has an angular component θ and a linear component r. The aim is to detect the unknown modes in the four provided datasets and to assign cluster labels, along with estimating basic statistics such as means, variances, and a circular–linear correlation measure. The next section describes the methodology and how the mean shift update is modified to respect circular geometry, followed by results and visualisations for each dataset.

II. METHODOLOGY

The algorithm is designed to identify clusters in data that combines angular and linear variables. It is built on four main components. The first is the iterative mode-finding procedure, where each data point is shifted toward regions of higher density until convergence. This produces a set of candidate cluster centers, or modes. The second component is mode merging, which identifies unique clusters by combining nearby convergence points based on thresholds in both the angular and radial dimensions. Once the modes are established, the third component assigns each data point to the nearest mode using a distance measure that accounts for the circular nature of angles and the linear scale of the radial values. Finally, the fourth component computes statistics for each cluster, summarizing the circular and linear distributions, as well as their correlation.

A. Kernel Choice

To handle data that mixes angles and linear values, I had to adjust the original mean shift formulation. Angles behave differently from ordinary numbers, since 0 and 2π represent the same direction, so a simple Euclidean kernel would not make sense. Because of this, I used a product kernel where each dimension has its own distance measure.

For the angular part, a von Mises kernel works well because it is essentially the circular analogue of a Gaussian, as explained in [2], [3], [4]. It is defined as

$$K_\theta(\theta_1, \theta_2) = \exp(\kappa \cos(\theta_1 - \theta_2)), \tag{2}$$

with $\kappa = 1/h_\theta^2$ as defined in [4] and wikepedia. Since it uses the cosine of the angle difference, it automatically handles wrap-around at 2π, meaning that angles near the boundary are still treated as close. This matches the behaviour expected for circular data and fits well with mean shift's density-seeking interpretation.

The radius r, on the other hand, is a linear variable, so I used the standard Gaussian kernel,

$$K_r(r_1, r_2) = \exp\left(-\frac{(r_1 - r_2)^2}{2h_r^2}\right), \tag{3}$$

which follows the same type of kernel used in Cheng's original mean shift work, but with a bandwidth parameter included for tuning.

Finally, the two parts are combined by multiplying them:

$$K\big((\theta_1, r_1), (\theta_2, r_2)\big) = K_\theta(\theta_1, \theta_2)\, K_r(r_1, r_2). \tag{4}$$

This product kernel keeps the method simple and interpretable, while letting each component be handled in a way that respects its underlying geometry.

B. Mean Computation

Computing an average of angles needs special care because angles wrap around at 2π. A simple arithmetic mean can give misleading results. For example, the average of 10° and 350° should be close to 0°, but a naive mean would incorrectly give 180°.

To handle this properly, I use the circular mean as explained in [3], [5]. Each angle is first mapped to a point on the unit circle, these points are averaged, and the result is converted back into an angle. The expression is

$$\bar{\theta} = \operatorname{atan2}\left(\sum_i w_i \sin(\theta_i), \sum_i w_i \cos(\theta_i)\right) \bmod 2\pi. \quad (5)$$

The reason I apply modulo 2π operation to shift the result into $[0,\, 2\pi]$, which keeps all angles in a consistent and positive range. This is because $\operatorname{atan2}$ returns values in the range $[-\pi,\, \pi]$.

For the radial component, the standard weighted mean is used

$$\bar{r} = \sum_i w_i r_i, \quad (6)$$

where the weights w_i are normalized kernel values.

C. Mode Merging

After running mean shift from every data point, I obtain many convergence points (modes). Often these points lie very close to one another, so I need a way to merge them into a final set of unique modes. Following the idea in Cheng's discussion of kernel bandwidth selection specially in Section IV.C, I treat two modes as the same if they are close in both angle and radius.

I merge two modes (θ_1, r_1) and (θ_2, r_2) when

$$|\theta_1 - \theta_2|_{\text{circular}} < h_\theta \quad \text{and} \quad |r_1 - r_2| < h_r,$$

where $|\theta_1 - \theta_2|_{\text{circular}}$ is the shortest angular distance, taking wrap-around into account. The thresholds h_θ and h_r are set equal to the bandwidth parameters. This is a simple heuristic that makes modes count as identical whenever they are separated by less than one bandwidth unit.

The current merging procedure is order dependent. The first mode I encounter is kept, and any nearby modes are discarded instead of being combined. This means that I do not average the points that fall into the same merged mode. A possible improvement would be to replace this "keep the first one" strategy with an averaging step so that the final merged mode better reflects all the convergence points that belong to it.

D. Cluster Assignment

After the unique modes have been finalized, every data point still needs to be linked to one of them. In this project, I decided to assign each point to the mode that is closest in a normalized (θ, r) distance. This is a design choice on my part, mainly because the two dimensions live on very different scales, and I wanted a simple rule that treats them fairly.

For a point $(\theta_{\text{point}}, r_{\text{point}})$ and a mode $(\theta_{\text{mode}}, r_{\text{mode}})$, the distance I use is

$$d = \sqrt{\left(\frac{|\theta_{\text{mode}} - \theta_{\text{point}}|_{\text{circular}}}{h_\theta}\right)^2 + \left(\frac{|r_{\text{mode}} - r_{\text{point}}|}{h_r}\right)^2}. \quad (7)$$

Normalizing by the bandwidths h_θ and h_r is important because θ is measured in radians, while r is measured in whatever distance units the data uses. Without normalization, these two numbers would not be directly comparable. After dividing by their bandwidths, both terms become dimensionless, which makes it reasonable to combine them with a Euclidean distance. This approach reflects my own judgment about what a "balanced" distance should look like in this setting.

Each point is then compared to every merged mode, and it is assigned to the one with the smallest normalized distance.

E. Evaluation Metrics

To assess the results of the clustering and the performance of the algorithm, several evaluation metrics were computed. These can be divided into two main categories: statistics describing each cluster and convergence metrics that evaluate how efficiently the algorithm works.

For each identified cluster, I calculate descriptive statistics to summarize both the angular and radial components. Since θ is circular and r is linear, each requires a different approach.

a) Circular statistics for θ: The mean direction of each cluster is computed using the circular mean formula introduced earlier in Mean Computation subsection as in equations 5. The spread of the angles is captured by the circular variance:

$$\text{Var}(\theta) = 1 - R, \quad (8)$$

where R is the mean resultant length, defined as

$$R = \frac{1}{n}\sqrt{\left(\sum \cos(\theta_i)\right)^2 + \left(\sum \sin(\theta_i)\right)^2}. \quad (9)$$

Here, R ranges from 0, indicating a uniform spread of angles, to 1, meaning all angles are identical. This follows standard circular statistics, as illustrated by [6], [7], [8].

b) Linear statistics for r.: For the radial component, which behaves like a normal linear variable, I compute the mean

$$\bar{r} = \frac{1}{n}\sum r_i \quad (10)$$

and the variance

$$\text{Var}(r) = \frac{1}{n}\sum (r_i - \bar{r})^2. \quad (11)$$

c) Circular–linear correlation.: To understand the relationship between angle and radius, I compute the circular–linear correlation coefficient following the methods of **Mardia (1976)** and **Johnson & Wehrly (1977)** [9], [10].

$$\rho_{cl} = \sqrt{\frac{r_{xs}^2 + r_{xc}^2 - 2 r_{xs} r_{xc} r_{cs}}{1 - r_{cs}^2}}. \quad (12)$$

In this formula:

$$r_{xs} = \text{correlation between } r \text{ and } \sin(\theta),$$
$$r_{xc} = \text{correlation between } r \text{ and } \cos(\theta),$$
$$r_{cs} = \text{correlation between } \cos(\theta) \text{ and } \sin(\theta).$$

F. Convergence Metrics

To evaluate the algorithm itself, I track how quickly each data point converges. For every point, I record the number of mean shift iterations required until the shift magnitude falls below a chosen tolerance. Additionally, the total runtime of the clustering process is measured using Python's `time.time()` for high-resolution timing.

From these measurements, I compute summary statistics such as the average, median, and maximum number of iterations. These metrics give insight into the computational efficiency and help identify regions where convergence is slower or more difficult.

By combining cluster statistics and convergence metrics, I can evaluate both the geometric quality of the clustering results and the performance of the algorithm in a comprehensive way.

III. Results

The proposed clustering pipeline was applied to four datasets to evaluate its performance on circular-linear data. Figures 1–4 show the resulting clusters in polar and Cartesian coordinates. The polar view highlights how the algorithm captures the angular and radial structure, while the Cartesian view confirms that clusters form coherent groups with clearly identifiable modes.

Table I summarizes the statistical properties of the detected clusters. Across all datasets, clusters show stable variance in both angle and radius, and the estimated modes align closely with the visual cluster centers.

Convergence statistics are reported in Table II. The algorithm is generally stable, with average iterations typically below 25. For data1.csv, the maximum average iteration reached 82.7, and the overall maximum iteration was 182. Despite these higher values, runtime remained manageable and convergence was reliable.

IV. Discussion and Conclusion

A. Performance Analysis

The results show that mean shift clustering works well for detecting meaningful groups in circular-linear data, although the performance is not uniform across all datasets. The convergence statistics in Table II make this very clear. File 1 stands out with much slower convergence, with an average of 82.7 iterations and a maximum of 182, while the other files settle around 21 to 22 iterations. What makes File 1 interesting is that its clusters are very well separated, so at first it feels surprising that the algorithm needs so many steps. Normally, strong separation gives steep density gradients that help points move quickly toward their modes.

A closer look at the data helps explain this behaviour. File 1 has a very different internal structure compared to the others. Cluster 1 in particular has a high radial variance (0.8479) and a low angular variance (0.0406), which makes the cluster narrow in θ but stretched out in r. This kind of elongated shape creates a density surface that is more like a long ridge instead of a sharp peak. Points that start far out along the radius do not see a steep gradient toward the center, so their movement becomes very small from one step to the next. Even though the final mode is clear, the path to get there is slow because the local slope of the density is gentle. This matches the high iteration counts we observe.

In contrast, File 2 converges much faster at around 21 iterations even though its clusters are closer to each other. The reason seems to be that the clusters in File 2 are more compact in both dimensions. Their density peaks are sharper, so points move more directly toward the modes. The separation between clusters does not seem to matter as much as how concentrated each cluster is internally. File 3 supports this idea as well. Even though it contains three clusters and one of them is much larger than the others (300, 543, and 157 points), the average convergence is 22.4 iterations, very similar to File 2. This suggests that as long as clusters have concentrated cores, the algorithm can guide points efficiently, regardless of how many clusters there are.

This highlights the main takeaway which is the speed of convergence depends more on the internal density shape of each cluster than the distance between different clusters. File 1 is separated but internally diffuse, while File 2 is closer together but sharply peaked. The algorithm moves faster in the second case because the gradients are steeper in the regions where points actually travel.

The bandwidth parameters $h_\theta = 0.15$ and $h_r = 0.65$ were tuned manually and then kept the same for all datasets. This worked fairly well, although not perfectly for every case. The slow behaviour in File 1 suggests that a slightly larger bandwidth might have helped smooth the density surface and reduce the number of iterations. The trade-off is that too large a bandwidth risks merging distinct clusters that should remain separate. Still, the fact that File 3 shows clear detection of its three modes even with very different cluster sizes is a good indication that the method handles uneven distributions well, which is one of the main strengths of density-based approaches like mean shift.

B. Circular-Linear Correlation

The correlation coefficients in Table I are generally low (0.03 to 0.19), suggesting weak relationships between angle and radius within clusters. This makes sense if the data generation process treats θ and r as independent. However, Cluster 1 in File 2 shows the highest correlation (0.1938), which might indicate a hidden pattern where certain angles tend to have larger or smaller r. looking at Figure 2 would reveal whether this is a real effect or just sampling noise. The low correlations also validate the product kernel design. If θ and r were strongly correlated, a product kernel (which assumes independence) might not be ideal. A more complex approach is to use a joint kernel that captures dependencies,

but this would complicate the method significantly and does not seem necessary given these results.

C. Mean Shift Use Case

Mean shift works best when the clusters really come from clear density modes in the data. The method is non-parametric and deterministic, so you do not need to guess the number of clusters before running it, and the same dataset will always give the same output. This makes it useful when the structure of the data is not known in advance. In this project the circular-linear version also fits naturally because each part of the data gets a kernel that matches its geometry.

There are still some limits. The computation cost is high, around $O(n^2)$ per iteration, since every point compares itself with all the others. For File3 with 1000 points, the total work is approximtley $1000 \times (1000 \times 22.4)$, which is about 22 million kernel evaluations. This is fine for the 1 to 1.5 second runtimes we see here, but a dataset with 10,000 points would be around 100 times slower, which becomes impractical very fast.

The algorithm also has trouble when the data does not form clear peaks. If the points are more uniform or stretched out in low density shapes, mean shift can create many tiny clusters or fail to separate them in a useful way. The slow convergence in File 1 is a sign of this issue. Cluster 1 has a large radial variance (0.8479), so it is more spread out than peaked. A different method might decide this is a diffuse region instead of a single neat cluster. In this sense mean shift is very sensitive to the actual shape of the density.

Another weakness is the bandwidth choice. I picked h_θ and h_r by trying different values and seeing what works. Bad choices can easily break the results. If the bandwidths are too small, the algorithm splits clusters too much, and if they are too big, it merges things that should stay apart. Mean shift does not have an automatic way to estimate good bandwidths from the data. The mode merging step helps a bit by grouping nearby modes, but the merging thresholds also depend on the bandwidths, so it creates a bit of a circular problem. This is probbably the biggest practical drawback of using mean shift in real applications.

D. Complexity

The main bottleneck of the method is the $O(n^2)$ cost in each iteration. Every point has to compare itself with all other points, and this happens again for many iterations (between 15 and 180 in our datasets). So the full cost is $O(n^2 \times \text{max\_iters})$. For File1 this is roughly $300^2 \times 82.7 \approx 7.4$ million operations, which is no problem on normal hardware. But if we had $n = 10{,}000$, the work would jump to around 10^8–10^9 operations, and that could take minutes or even hours.

There are some known tricks, like using k-d trees or ball trees, that reduce the work to $O(n \log n)$ per iteration by skipping faraway points. The issue is that these structures are made for Euclidean space. Since our data is circular-linear, the angle wraps around, so building such trees is not straightforward. A simpler idea could be to put points into angle–radius bins using a grid or hash table. Then each point only checks nearby bins. This would be faster but also a bit approximate.

The number of iterations depends mostly on the tolerance value (tol $= 10^{-4}$) and the shape of the density. File1 needs many iterations because parts of the density seem flat, so points move slowly. If we use a stricter tolerance, we get even more iterations. If we loosen it, points might stop too early and miss the true mode. There is no perfect tolerance value that works everywhere. It depends on the scale of the data and how accurate we want the result to be.

E. Comparison with HDBSCAN

If I were to compare mean shift with HDBSCAN, the two methods actually follow very different ideas about what a cluster is. Mean shift tries to move every point uphill on the density surface until it reaches a mode, while HDBSCAN builds a hierarchy of density regions and looks for the ones that stay stable across scales.

One important thing is how they treat noise. Mean shift always assigns every point to a cluster, even points that look like they do not really belong anywhere. In some of my figures, a few border points seem like outliers but the algorithm still forces them into the nearest mode. HDBSCAN would likely mark these as noise, which might give a cleaner clustering.

HDBSCAN also handles different density levels better. It can detect clusters with very different shapes or densities because it works across many scales. Mean shift only uses a single bandwidth, so all clusters need to have kind of similar density structure. For example, in File 3 the cluster sizes are very different (157 to 543 points), and a density-hierarchy method might capture this variation in a more natural way.

There is also a difference in runtime. HDBSCAN usually runs in $O(n \log n)$, while mean shift takes $O(n^2)$ per iteration. For the datasets here (300–1000 points) this is fine, but for large data HDBSCAN would scale much better. This becomes a real issue if we had something like $n = 10000$, since mean shift would be much slower.

But using HDBSCAN on circular-linear data is not straightforward. The normal version works with Euclidean distances, which do not respect the angle wrap-around. It would need a custom distance function, and the tree structures inside HDBSCAN would have to handle the circular part correctly. Mean shift ended up being easier to adapt because the product kernel treats the angle and radius separately in a natural way.

TABLE I: Cluster Statistics for All Files

File	Cluster	Size	Mode (θ, r)	Mean (θ, r)	Var(θ)	Var(r)	Corr(θ, r)
File 1	Cluster 1	100	0.004, 5.153	6.265, 4.835	0.0406	0.8479	0.1263
	Cluster 2	200	3.354, 0.988	3.261, 0.955	0.1981	0.0379	0.0595
File 2	Cluster 1	255	0.006, 4.784	6.198, 5.003	0.0238	0.9893	0.1938
	Cluster 2	545	0.785, 4.022	0.760, 4.055	0.0271	0.3670	0.1196
File 3	Cluster 1	300	0.007, 5.219	0.001, 5.009	0.0402	1.0688	0.0921
	Cluster 2	543	2.370, 3.985	2.378, 3.877	0.0227	0.4696	0.1817
	Cluster 3	157	3.092, 2.100	3.264, 1.906	0.0440	0.4916	0.1730
File 4	Cluster 1	300	6.252, 4.948	6.275, 5.014	0.0427	1.0357	0.0482
	Cluster 2	500	3.124, 2.088	3.146, 1.986	0.0285	0.2466	0.0368

TABLE II: Convergence Statistics Per File

File	Avg Iters	Median	Max	Time (s)
File 1	82.7	93	182	0.98463
File 2	21.0	20	37	0.95672
File 3	22.4	18	60	1.52920
File 4	21.5	17	39	1.00468

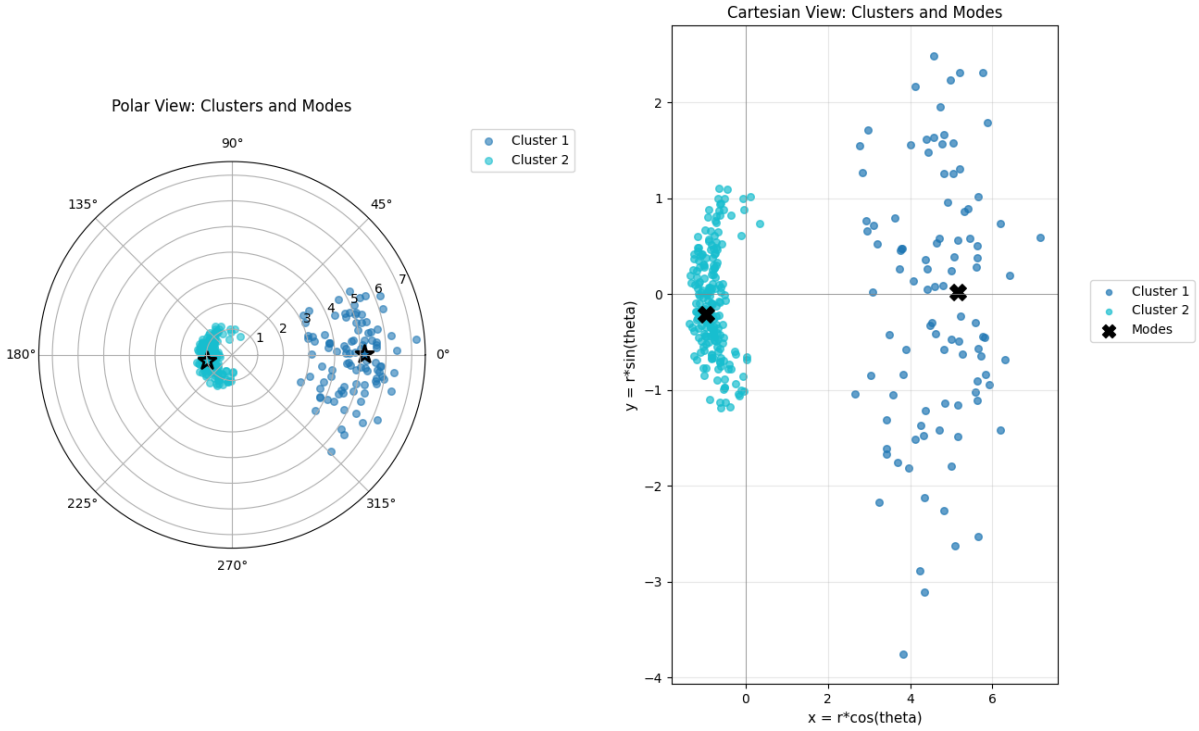

Fig. 1: Polar (left) and Cartesian (right) visualizations of the extracted clusters. The polar plot shows the angular–radial distribution of cluster points and their modes, while the Cartesian plot shows the spatial configuration of the same clusters.

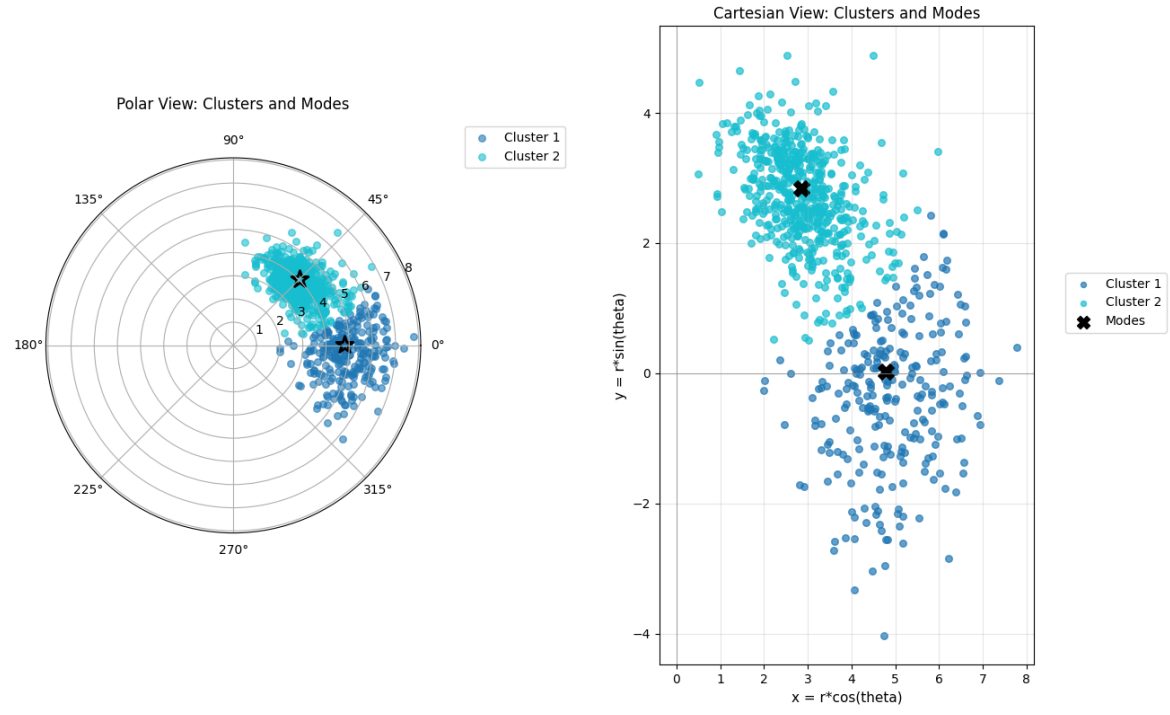

Fig. 2: Polar (left) and Cartesian (right) visualizations of the extracted clusters. The polar plot shows the angular–radial distribution of cluster points and their modes, while the Cartesian plot shows the spatial configuration of the same clusters.

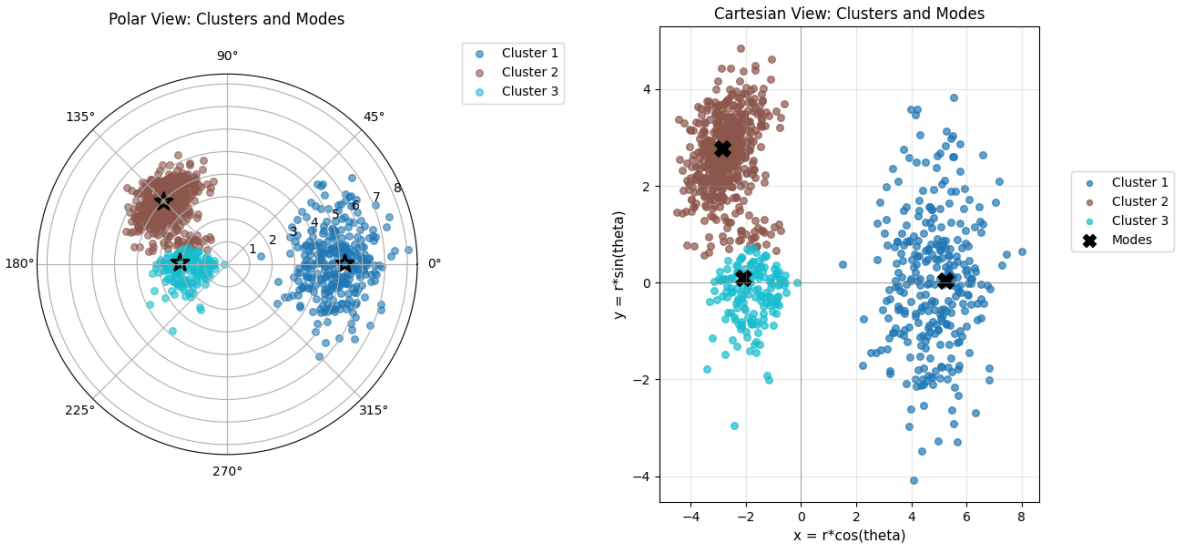

Fig. 3: Polar (left) and Cartesian (right) visualizations of the extracted clusters. The polar plot shows the angular–radial distribution of cluster points and their modes, while the Cartesian plot shows the spatial configuration of the same clusters.

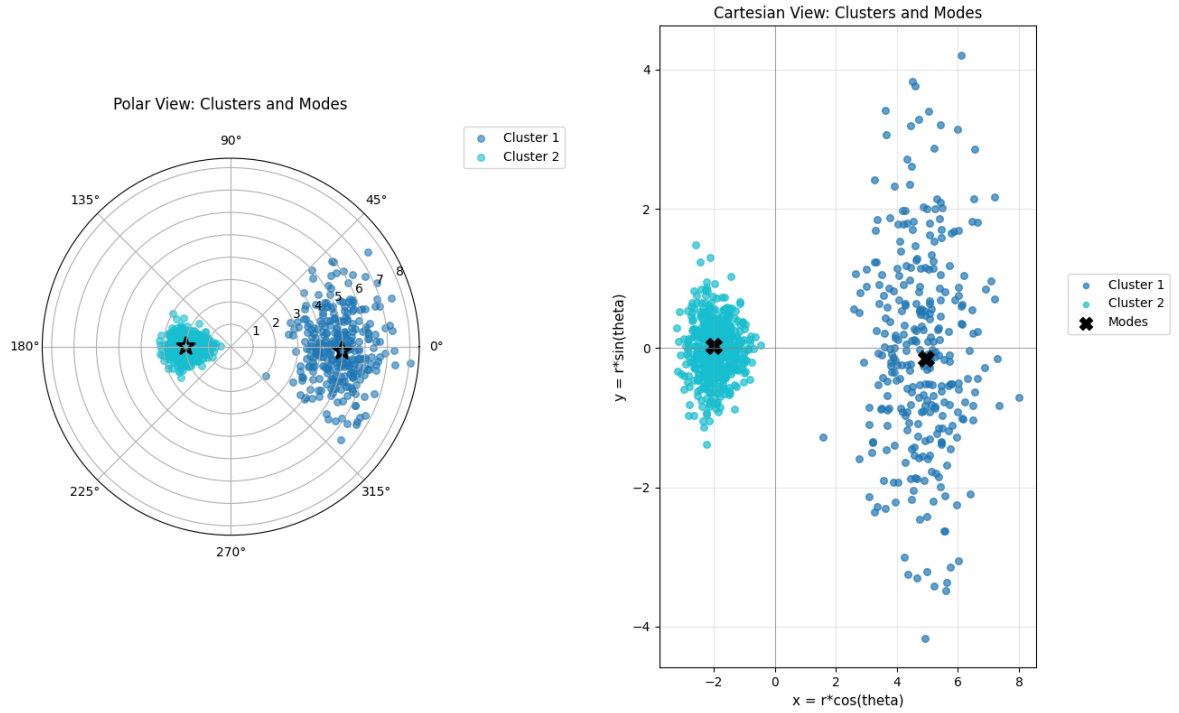

Fig. 4: Polar (left) and Cartesian (right) visualizations of the extracted clusters. The polar plot shows the angular–radial distribution of cluster points and their modes, while the Cartesian plot shows the spatial configuration of the same clusters.

REFERENCES

[1] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995.

[2] S.-J. Chang-Chien, W.-L. Hung, and M.-S. Yang, "On mean shift-based clustering for circular data," *Soft Computing*, vol. 16, p. 1043–1060, 2012.

[3] S. Tootoonian, "2.3.8 periodic variables – the von mises distribution – pattern recognition and machine learning," YouTube video, 2024, published 19 Aug 2024; Accessed 2025-11-22. [Online]. Available: https://www.youtube.com/watch?v=2kIGEEzie1M

[4] S. Developers, "Circular distributions — von mises distribution," Online documentation, 2025, accessed 2025-11-23. [Online]. Available: https://mc-stan.org/docs/functions-reference/circular_distributions.html

[5] K. V. Mardia and P. E. Jupp, *Directional Statistics*. Chichester, UK: John Wiley Sons, 2000.

[6] F. H. Allen and O. Johnson, "Automated conformational analysis from crystallographic data. 4. statistical descriptors for a distribution of torsion angles," *Acta Crystallographica Section B*, vol. 47, pp. 62–67, 1991, accessed 2025-11-20. [Online]. Available: https://journals.iucr.org/paper?mu0188

[7] M. W. MacArthur and J. M. Thornton, "Conformational analysis of protein structures derived from nmr data," *Proteins*, vol. 17, pp. 232–251, 1993. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/8272423/

[8] R. E. Watson, "Two educational comparisons of linear and circular statistics," Ph.D. dissertation, Loyola University Chicago, Chicago, Illinois, 1987, accessed 2025-11-21. [Online]. Available: https://ecommons.luc.edu/luc_diss/2501

[9] K. V. Mardia, "Linear-circular correlation coefficients and rhythmometry," *Biometrika*, vol. 63, no. 2, pp. 403–405, 1976, accessed 2025-11-23 09:37 UTC. [Online]. Available: https://www.jstor.org/stable/2335637

[10] R. A. Johnson and T. Wehrly, "Measures and models for angular correlation and angular-linear correlation," *Journal of the Royal Statistical Society Series B*, vol. 39, no. 2, pp. 222–229, 1977. [Online]. Available: https://www.jstor.org/stable/2984799