Y-Graph: A Max-Ascent-Angle Graph for Detecting Clusters

Junyi Guan, *Member, IEEE*, Sheng Li, *Senior Member, IEEE*, Xiongxiong He, Jiajia Chen, Yangyang Zhao, and Yuxuan Zhang

***Abstract*—Graph clustering technique is highly effective in detecting complex-shaped clusters, in which graph building is a crucial step. Nevertheless, building a reasonable graph that can exhibit high connectivity within clusters and low connectivity across clusters is challenging. Herein, we design a max-ascent-angle graph called the "Y-graph", a high-sparse graph that automatically allocates dense edges within clusters and sparse edges across clusters, regardless of their shapes or dimensionality. In the graph, every point x is allowed to connect its nearest higher-density neighbor δ, and another higher-density neighbor γ, satisfying that the angle $\angle\delta x\gamma$ is the largest, called "max-ascent-angle". By seeking the max-ascent-angle, points are automatically connected as the Y-graph, which is a reasonable graph that can effectively balance inter-cluster connectivity and intra-cluster non-connectivity. Besides, an edge weight function is designed to capture the similarity of the neighbor probability distribution, which effectively represents the density connectivity between points. By employing the Normalized-Cut (Ncut) technique, a Ncut-Y algorithm is proposed. Benefiting from the excellent performance of Y-graph, Ncut-Y can fast seek and cut the edges located in the low-density boundaries between clusters, thereby, capturing clusters effectively. Experimental results on both synthetic and real datasets demonstrate the effectiveness of Y-graph and Ncut-Y.**

***Index Terms*—Complex-shaped clusters, normalized-cut, graph clustering.**

I. Introduction

THE intrinsic heterogeneity of nonuniform data implies the existence of latent structure [1], and uncovering the latent structure is what clustering aims at. Clustering methods group data points into clusters for data analysis based on similarity [2]. So researchers can delve into the intrinsic properties of the data and discover new knowledge in fields like computer science, biology, and social science. Different clustering algorithms have been proposed based on specific assumptions regarding the nature of "cluster" [3], such as partitional, hierarchical, density-based, and graph-based. Density-based methods are effective in reconstructing arbitrary shapes, owing to the cluster definition as high-density areas with low-density gaps separating themselves from other clusters [4].

DBSCAN [5], a typical density-based method, captures maximum density-connected point sets according to a specific density-connectivity criterion to achieve arbitrary-shaped cluster reconstruction. However, obtaining an effective density-connectivity criterion often requires tedious parameter tuning. Subsequent works, e.g., [6], [7], developed adaptive parameter tuning techniques, still, these methods would possibly merge high-overlapping clusters [8].

Mean-Shift [9] also sees a cluster as a dense area and can well overcome the abovementioned limitations. It detects clusters differently: it first constructs a density surface of the data, and then lets each point perform a "mean-shift" procedure (i.e., a gradient ascent) on the local density surface until convergence. So the shifted data points are grouped into clusters and the local density peaks (i.e., local density maximum area) are identified as cluster centers. Note that the "mean-shift" procedure can easily locate boundaries between clusters that are always in valley structures (i.e., local density minimum area) and detect all local density peaks as cluster centers [10]. However, this can also cause Mean-Shift to over-divide multi-peak clusters [11].

Density Peak Clustering (DPC) [12] can address the over-division issue by selecting high-representativeness density peaks as cluster centers based on its center assumption—-*cluster centers are density peaks that have high densities and are far away from points with higher densities*. Besides, in terms of allocation strategy, DPC applies a bottom-up hierarchical clustering technique according to a specific linkage metric (i.e., "*minimum center-boosting distance*" named by [13]). However, this linkage metric does not take into account the density connectivity between clusters, which makes the reconstruction performance of DPC on arbitrary shape unrobust [14]. Later works, e.g., [15], [16], improved DPC's allocation strategy, but still followed the main assumption of DPC, i.e., finding high-representativeness density peaks as final centers. Besides, since whether a density area is merged with others is depended on its density peak's representativeness rather than its density connectivity to other areas, false mergings of low density-connected density areas may possibly occur, resulting in a loss of information about the underlying structure of the data.

Received 17 July 2023; revised 30 August 2024; accepted 17 October 2024. Date of publication 24 October 2024; date of current version 26 November 2024. This work was supported in part by the National Science Foundation of China under Grant 62306282 and Grant 62233016 and in part by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang under Grant 2024C04023. Recommended for acceptance by A. Bonifati. *(Corresponding author: Sheng Li.)*

Junyi Guan is with the School of Information Science and Technology, Hangzhou Normal University, Hangzhou 311121, China (e-mail: jonnyguan73@163.com).

Sheng Li, Xiongxiong He, Jiajia Chen, Yangyang Zhao, and Yuxuan Zhang are with the College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China (e-mail: shengli@zjut.edu.cn; hxx@zjut.edu.cn; fl_katrina@163.com; zyyzhao09@163.com; xb905429695@163.com).

Digital Object Identifier 10.1109/TKDE.2024.3486221

Graph-based methods obtain clusters by cutting the graph of data into high-connected regions [18]. Classically, the spectral clustering technique describes clustering as a minimum-cost graph-cutting problem and provides an effective standard linear algebra solution [19]. More like a graph-cutting tool for data clustering, the construction of a graph is crucial for spectral clustering. Intuitively, a reasonable graph should have high data connectivity within clusters while low across clusters.

Spectral clustering often builds a dataset into a kNN (or a ϵ-neighborhood) graph with a Gaussian kernel weight function to capture the data connectivity information. Nevertheless, picking a reasonable k (or ϵ) for constructing a reasonable graph—a graph with dense edges within clusters and sparse across clusters—is challenging. Because a trade-off exists between inter-cluster connectivity and intra-cluster non-connectivity. Besides, to ensure good data connectivity, the edges of the graph may be much more than the points within the dataset when building a kNN-graph. Thus, when processing large data, a huge number of edges may make for a more difficult time-consuming clustering task. On the other hand, storing such a graph requires a lot of memory. Therefore, a graph with a few edges that can effectively reflect the differences in data connections within and across clusters is of great significance.

Inspired by the character that a cluster has a unique property on the data density surface, i.e., a high-density connected region, we designed a max-ascent-angle graph (Y-graph[1]) that can efficiently reflect the underlying structure of data with a few edges, where the edge weight is to reflect the density connectivity of points. The proposed clustering problem—-*cutting the graph into high-density connected clusters with the minimum weight cost (density-connected cost)* can be solved by using traditional spectral clustering tools, i.e., Ncut solver [20]. The main contributions of this paper are as follows:

1) By maximizing the ascent angles of all points, the Y-graph achieves high pairwise connectivity within clusters and strong disconnectivity between clusters.
2) The edge weight is designed as the similarity of neighbor probability distribution that can reasonably represent the density connectivity between points. Then, by applying the Ncut technique on the Y-graph, the proposed Ncut-Y algorithm can robustly cut the edges of low-density connectivity areas between clusters. This enables it to effectively capture complex-shaped clusters.
3) The construction of Y-graph with high sparsity only requires kNN distances of data as input and is suitable for large-scale data clustering.

The rest paper is composed: Section II introduces the related works; Section III mainly focuses on the proposed algorithm; Section IV demonstrates the experiments and discussions; Section V gives the final conclusion.

II. Related Works

By integrating the geometric characteristic of a cluster's density surface, we propose a sparse graph for the efficient capturing of underlying data structure.

[1]The code is available at https://github.com/Guanjunyi/Y-graph

A. Graphs for Clustering

Graph-theoretical methods are often used for clustering problems in the form of graph-cutting [21], where a crucial step is graph building [22].

For example, the single-linkage method [23], a classic hierarchical clustering algorithm, initially constructs the dataset into a Minimum Spanning Tree (MST) based on a linkage metric known as the *minimum member distance*. Subsequently, it divides the MST into a specified number of clusters with the minimum weight, essentially solving a graph-cutting problem of MST. In recent research, Yan Ma et al. [25] calculated inter-cluster distances using the centroids of MST and incorporated the cut edge length as a merging criterion, reducing complexity while maintaining clustering performance. Gaurav Mishra et al. [27] constructed MST based on subcluster centroids to identify adjacent pairs, ensuring the accuracy of the merging process. Moreover, in [24], a merging technique based on the circumference proximity of the graph is devised to guide the merging process of subclusters, facilitating the efficient and accurate amalgamation of adjacent subclusters into the final result within hybrid clustering algorithms. To better capture the topological structure of data, Rashmi Maheshwari et al. [26] integrated entropy with local neighborhood information of the graph for a more precise computation of data point density distribution.

Notably, DPC also deals with a graph-cutting problem. Because its allocation strategy applies a linkage metric, the *minimum center-boosting distance* that first named in [13], to build the dataset into a Density-Boosting Minimum Spanning Tree (DBMST) to solve the DBMST cutting problem [11]. However, these MST structures cannot fully cover the structural information of clusters. Because a tree structure only focuses on describing the parent-child relationship and will lose the relationship between nodes at the same level, thereby, missing out on some important structural information. Consequently, the algorithm may fail to interpret the cluster structure completely or cut the edges (between clusters) that should be cut, thereby, leading to a poor clustering result.

Therefore, a graph that can reasonably reflect the data structure, i.e., a graph with high data connectivity within clusters while low data connectivity between clusters, is necessary. Compared to a tree structure, a well-built graph structure can provide higher information connectivity for data, such as Fully Connected Graph (FCG), kNN-graph, ϵ-neighborhood graph (hereinafter, ϵ-graph), and so on [30]. FCG provides the highest information connectivity for data since each point is directly connected to all other points. But as the densest graph, FCG does not well reflect the local structure of the data. In contrast, kNN-graph and ϵ-graph can do better, which is credited to their popularity in practical graph-cutting tasks.

In a kNN-graph, each point is only directly connected to its k nearest neighbors, while ϵ-graph only connects points within its ϵ-neighborhood, in other words, the distance between the two connected points is less than ϵ value. However, the kNN-graph and ϵ-graph do not deliberately reflect the differences between intra-cluster data connectivity and inter-cluster data connectivity.

In this work, we designed a highly sparse graph that can efficiently reflect the difference between intra-cluster connectivity and inter-cluster non-connectivity. To be noted, for solving general graph-cutting problems, we can use spectral clustering that is simple to implement and can be efficiently solved by standard linear algebra methods [19], [31].

B. Spectral Clustering

Let $X = \{x_1, x_2, \ldots, x_n \mid x_i \in \mathbb{R}^d\}$ be a $n \times d$ data matrix with n data points and d features, where x_i represents the ith point. With given X, one can obtain its symmetric similarity matrix $S \in \mathbb{R}^{n \times n}$ according to a specific similarity measure criterion. The (i, j)th element S_{ij} represents the similarity between x_i and x_j.

In general, an adjacency graph A (typically an ϵ-graph) and a Gaussian kernel-based edge weight function w are used to define S, as:

$$S_{ij} = w_{ij} \times A_{ij}, w_{ij} = \exp\left(-\frac{||x_i - x_j||_2^2}{\phi}\right) \tag{1}$$

where ϕ is a kernel parameter, and A_{ij} represents the adjacency between x_i and x_j, as:

$$A_{ij} = \begin{cases} 1 & \text{if } ||x_i - x_j||_2 < \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Spectral clustering views similarity matrix S as a graph and aims to divide the graph into subgraphs (clusters) by cutting edges between clusters with the minimum weight cost. In other words, spectral clustering treats the clustering problem as a minimum graph-cutting problem which can be described as the following optimization problem:

$$\min_H \Psi = Tr(H^T L H), \quad \text{s.t. } H^T H = I \tag{3}$$

where $H \in \mathbb{R}^{n \times c}$ is a label feature matrix, where c means the number of label features [31]. $L = I - \hat{S}$ is a normalized Laplacian matrix, where $\hat{S} = D^{-1/2} S D^{-1/2}$ (or $D^{-1} S$) is the normalized similarity matrix. $D = diag(d_1, d_2, \ldots, d_n) \in \mathbb{R}^{n \times n}$ is a degree matrix that is a diagonal matrix whose entries are row sums of S, as:

$$d_i = \sum_{j=1}^{n} S_{ij} \tag{4}$$

Fortunately, by expressing the spectral clustering problem as the standard trace minimization problem (3), we can solve it by using the matrix H with the first c eigenvectors of L as its rows. Since the eigenvectors in H can take on continuous values, we need to choose a classical clustering algorithm (e.g., K-means [33]) to partition them into clusters after matrix H is obtained.

Different $\hat{S}$ describes different clustering goals. The Ratio-cut clustering [32] uses $\hat{S} = D^{-1} S$, which tends to partition the graph into clusters of similar size, while the Ncut clustering uses $\hat{S} = D^{-1/2} S D^{-1/2}$, which tends to partition the graph into clusters that are of similar size and relatively sparse compared to each other.

Spectral clustering demonstrates excellent clustering performance; however, scalability remains a challenge. In [28], a sparse similarity graph of size $O(n)$ was constructed by leveraging the intrinsic structural information of data points, significantly speeding up the subsequent spectral clustering process. Geping Yang et al. [29] analyzed the four stages of large-scale spectral clustering and proposed corresponding acceleration methods, effectively enhancing the computational efficiency of spectral clustering. In this work, we constructed a highly sparse Y-graph and utilized Ncut for clustering, demonstrating good scalability and computational efficiency.

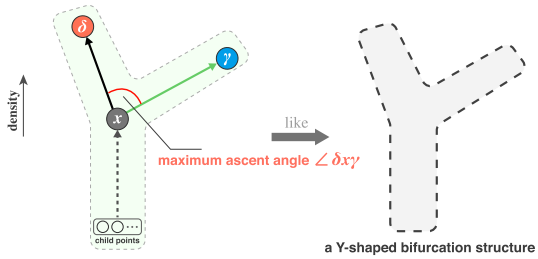


Fig. 1. An explanation for the naming of the Y-graph.

III. The Ncut-Y Algorithm

According to the geometric characteristics of clusters on the data density surface, we designed a graph structure that is well-suited for capturing data associations within the dataset, called the "Y-graph". Fig. 1 illustrates the reason we call our graph the "Y-graph". As shown, in the Y-graph, each point x is allowed to find two associated neighbors: 1) the closest higher-density neighbor (δ) and 2) the second higher-density neighbor (γ), to determine its max-ascent-angle $\angle\delta x\gamma$. This density-boosting two-outdegree association structure resembles the shape of the letter "Y".

Fig. 2 illustrates the main idea of our Y-graph building and its role in the clustering process. First, kNN-based density estimation is employed to estimate the density distribution of the data points, which helps to capture the local density information. Next, connections between each point and two high-density points within its local neighborhood are established, forming "the maximum ascent angle" (as described in Section III-A), which helps to construct our Y-graph structure. Subsequently, we apply our Cross KL-based weight function (as discussed in Section III-B) to assign weights to the edges in the graph structure. This weighting step enhances the importance of informative edges in the graph. Finally, based on our cluster assumption (see Section III-C), the Ncut solver is utilized to find and cut the low-density boundaries among strong-associated clusters to obtain the clustering result.

A. Graph Building

Given a dataset $X = \{x_1, x_2, \ldots, x_n \mid x_i \in \mathbb{R}^d\}$, for each point x_i, we define its neighbors as $N_k(x_i) = \{x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(k)}\}$, where $x_i^{(k)}$ means x_i's kth nearest neighbor. Then, we estimate the kNN-based density of points as in (5), where $k_\rho = 2k$ for obtaining a relatively smooth density distribution of data, and $d_{ij} = dist(x_i, x_j)$ represents

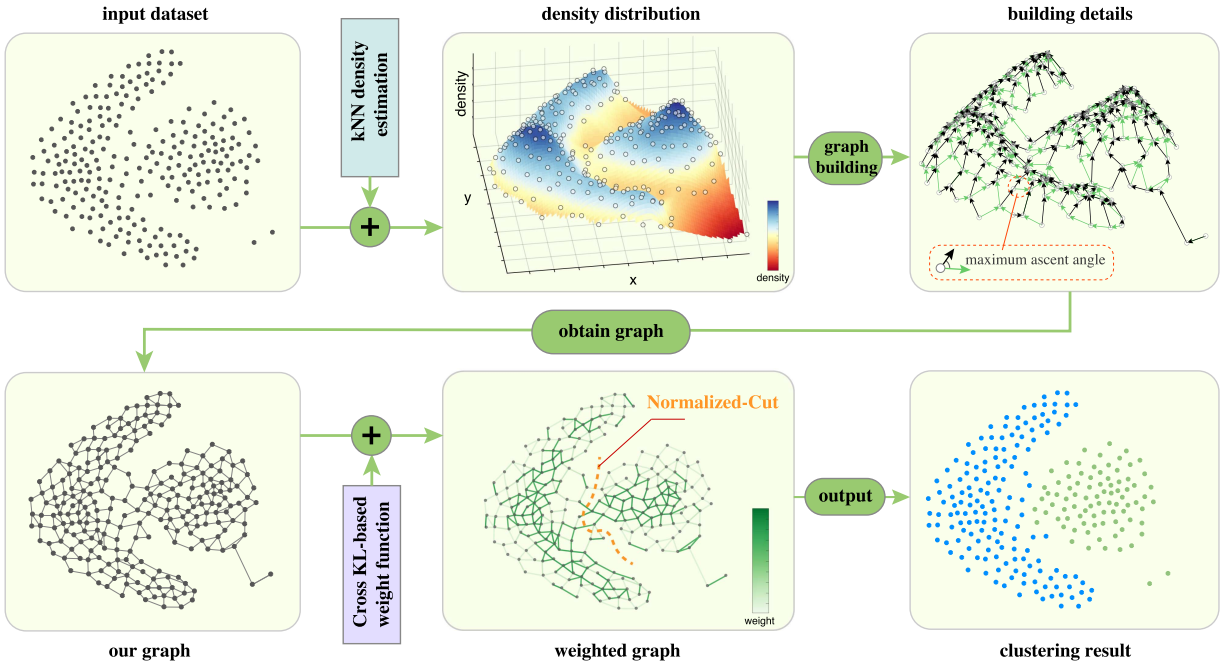


Fig. 2. The clustering framework of the proposed Ncut-Y algorithm.

the euclidean distance between x_i and x_j.

$$\rho_i = \frac{1}{\frac{1}{k_\rho}\sum_{x_j \in N_{k_\rho}(x_i)} d_{ij}} \tag{5}$$

Each point x_i with density value is assumed to fall on the density surface of data X, and then, let point x_i find and connect its two higher-density neighbors δ_{x_i} and γ_{x_i}, i.e., $\delta_{x_i}, \gamma_{x_i} \in N_k(x_i), , \min(\rho_{\delta_{x_i}}, \rho_{\gamma_{x_i}}) > \rho_{x_i}$. In what follows, we provide the corresponding neighbor-finding strategy:

First, δ_{x_i} is defined as x_i's *closest higher-density neighbor*, which may along the steepest path of x_i to the higher density area, as:

$$\delta_{x_i} = \arg\min_{x_z}(d_{iz})$$
$$\text{s.t. } \rho_{x_z} > \rho_{x_i}, x_z \in N_k(x_i), d_{iz} < d_{nei} \tag{6}$$

where a small-value radius paramter d_{nei} is added to ensure that δ_{x_i} is indeed a neighbor nearby x_i, set as:

$$d_{nei} = \frac{2}{n}\sum_{x_i \in X} dist(x_i, x_i^{(k_b)}) \tag{7}$$

$k_b = \lceil \ln(n^2) \rceil$ is a small value to ensure the proximity within radius d_{nei}. Then, for $\delta_{x_i} = \varnothing$, we reset $\delta_{x_i} = x_i$.

On this basis, we introduce a concept of *"ascent angle"-the angle formed between point x_i and its two higher-density neighbors (one is δ_{x_i}), with point x_i as the vertex*, denoted as θ_{x_i}, as:

$$\theta_{x_i} = \left(\cos^{-1}\left(\frac{\overrightarrow{x_i\delta_{x_i}} \cdot \overrightarrow{x_i x_z}}{||\overrightarrow{x_i\delta_{x_i}}||_2 \cdot ||\overrightarrow{x_i x_z}||_2}\right)\right)$$
$$\text{s.t. } \rho_{x_z} > \rho_{x_i}, x_z \in N_k(x_i), x_i \in N_k(x_z), d_{iz} < d_{nei} \tag{8}$$

The ascent angle θ reflects the size of the association field of a point. To obtain the maximum ascent association field for each point x_i, we connect an edge point x_i to γ_{x_i} (the second higher-density neighbor) to maximize the ascent angle θ_{x_i}. So, the neighbor γ_{x_i} should meet the condition:

$$\gamma_{x_i} = \arg\max_{x_z}(\theta_{x_i})$$
$$\text{s.t. } \rho_{x_z} > \rho_{x_i}, x_z \in N_k(x_i), x_i \in N_k(x_z), d_{iz} < d_{nei} \tag{9}$$

For point x_i with $\gamma_{x_i} = \varnothing$, we reset $\gamma_{x_i} = x_i$.

After obtaining δ and γ neighbors, we connect points to their neighbors to build a graph $G(X, E)$, where X is the node set, and the edge set $E = \{e_{ij} | x_j = \delta_{x_i} \vee x_j = \gamma_{x_i} \vee x_i = \delta_{x_j} \vee x_i = \gamma_{x_j}\}$. Herein, $G(X, E)$ is called "*the max-ascent-angle graph*", and is denoted as "Y-graph".

Fig. 3 presents the feasibility of our Y-graph structure in a continuous two-peak density surface (without a stationary point). In this method, we divide the points into four catalogies, including the non-peak point within a single-peak density area (A-type), the saddle point on the valley (B-type), the peak point (C-type), and the non-saddle valley point (D-type). As shown, B-type and D-type points at the valley position connect two single-peak density areas, which ensures that the association between the two single-peak density areas is recorded; while the edges of A-type points enhance the data connectivity in the single-peak density areas.

To assume that an approximate continuous density surface is composed of n data points: n_a A-type points, n_b B-type points, n_c C-type points, and n_d D-type points, generally, $n_a \gg (n_b + n_d) \gg n_c, n = n_a + n_b + n_c + n_d$. Every point x (except peak point) connects two edges to its δ_x and γ_x points, so, our Y-graph has $2(n_a + n_b + n_d)$ edges.

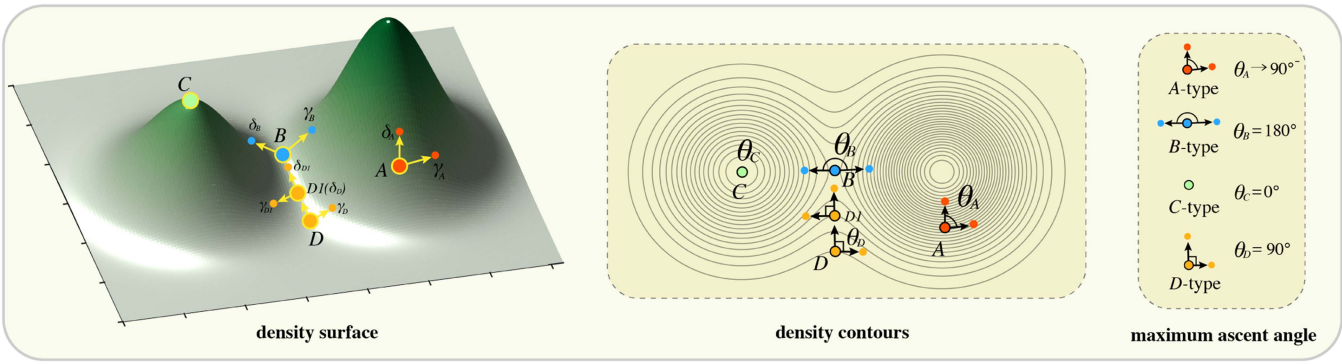


Fig. 3. The maximum ascent angles of four types of data points on a continuous density surface.

In the graph, A-type points will always connect points within its single-peak density area, which means that A-type points have $2n_a$ edges within single-peak density areas. While B-type and D-type points will connect single-peak density areas, which means there are $2(n_b + n_d)$ edges that describe valley areas.

Because $2n_a \gg 2(n_b + n_d)$, our Y-graph can use most edges ($2n_a$ edges) to effectively describe the high data connectivity within local high-density areas, and few edges ($2(n_b + n_d)$ edges) to well describe the low data connectivity of valley areas. In other words, our Y-graph can automatically and reasonably allocate edges: dense edges in local high-density areas, and sparse edges in local low-density areas. Such graph structure is well-suited for data clustering tasks because it can describe data associations with a fixed number of edges (i.e., about $2n$ edges but no more than $2n$).

Additionally, there are some interesting features of the maximum ascent angle of $A \sim D$ type points on a continuous density surface, as in Discussion 1.

Discussion 1: The B-type point owns largest ascent view-angle $\theta_B = 180°$; A-type point owns $\theta_A \to 90°^{-}$; C-type point C has no ascent view-angle $\theta_C \to 0°$; and D-type point D has its ascent view-angle $\theta_D \to 90°$.

For a non-peak point x, its direction to nearest higher-density point δ_x is perpendicular to its density contour line, so: 1) if x is a B-type point, its direction to γ_x point (within another single-peak density area) will also perpendicular to its density contour line, so $\theta_B = 180°$; 2) if x is a A-type point on a convex density surface, its γ_x point (within the same single-peak density area) will infinitely close to the density contour line but higher than the density contour line (since $\rho_{\gamma_x} > \rho_x$), so $\theta_A \to 90°^{-}$; 3) if x is a D-type point on a concave density surface, its direction to γ_x point can be perpendicular to its direction to δ_x, so $\theta_D = 90°$.

In practice, clusters often consist of multiple single-peak density areas [34], also known as multi-peak clusters. Therefore, a multi-peak cluster contains some valley areas, called inter-cluster valleys. Different from valley areas across clusters (called across-cluster valleys), inter-cluster valleys usually have a stronger association. To better reflect the difference between inter-cluster valleys and across-cluster valleys, in what follows, an edge weight function is designed.

B. Edge Weight Function Design

Fig. 4 shows a probability density distribution of two clusters. As shown, clear differences exist between inter-cluster valleys and across-cluster valleys. Given two edges $edge1$ and $edge2$ fall in the across-cluster valley and the inter-cluster valley, respectively. By observing the neighbor probability distribution of points a, b, c, and d, we find that points a and b (of $edge1$) have a high similar distribution, while points c and d (of $edge2$) have a low similar distribution. Inspired by this, we designed an edge weight function $w(\cdot)$ based on the cross KL divergence [35], which reflects edge e_{ij}'s weight w_{ij} as the similarity of neighbor probability distribution (hereinafter, the distribution similarity) between points x_i and x_j connected by e_{ij}.

For a point $x_x \in X$, its neighboring area is set as $N(x_i) = \{x_i, N_k(x_i)\} = \{x_i^{(0)}, x_i^{(1)}, \ldots, x_i^{(k)}\}$ (where $x_i^{(0)} = x_i$). Think of the neighboring area $N(x_i)$ as a sample space $\{x_i^{(0)}, x_i^{(1)}, \ldots, x_i^{(k)}\}$, and let all point within the space have the probability $\frac{1}{\sum_{x \in N(x_i)} \rho_x}\{\rho_{x_i^{(0)}}, \rho_{x_i^{(1)}}, \ldots, \rho_{x_i^{(k)}}\}$. Then, we have x_i's discrete neighboring probability distribution as $\mathcal{P}_{x_i} = \{\mathcal{P}_{x_i}(x_i^{(0)}), \mathcal{P}_{x_i}(x_i^{(1)}), \ldots, \mathcal{P}_{x_i}(x_i^{(z)})\}$, defined as:

$$\mathcal{P}_{x_i}(x_i^{(z)}) = \frac{\rho_{x_i^{(z)}}}{\sum_{x \in N(x_i)} \rho_x} \tag{10}$$

For each pair of adjacent points x_i and x_j in graph $G(X, E)$ (i.e., $\exists e_{ij}$ or $e_{ji} \in E$), we have the cross KL divergence between discrete neighboring probability distribution $\mathcal{P}_{x_i}$ and $\mathcal{P}_{x_j}$ as:

$$D(\mathcal{P}_{x_i}||\mathcal{P}_{x_j}) = \frac{1}{2}\left(D_{KL}(\mathcal{P}_{x_i}||\mathcal{P}_{x_j}) + D_{KL}(\mathcal{P}_{x_j}||\mathcal{P}_{x_i})\right) \tag{11}$$

where $D(\mathcal{P}_{x_i}||\mathcal{P}_{x_j})$ is the KL divergence between points x_i and x_j as:

$$D_{KL}(\mathcal{P}_{x_i}||\mathcal{P}_{x_j}) = \sum_{z=0}^{k} \mathcal{P}_{x_i}(x_i^{(z)}) \log\left(\frac{\mathcal{P}_{x_i}(x_i^{(z)})}{\mathcal{P}_{x_j}(x_j^{(z)})}\right) \tag{12}$$

Then, we view the cross KL divergence $D(\mathcal{P}_{x_i}||\mathcal{P}_{x_j}) \in [0, +\infty)$ between $\mathcal{P}_{x_i}$ and $\mathcal{P}_{x_j}$ as the dissimilarity between points x_i and x_j. On this basis, we define weight function w

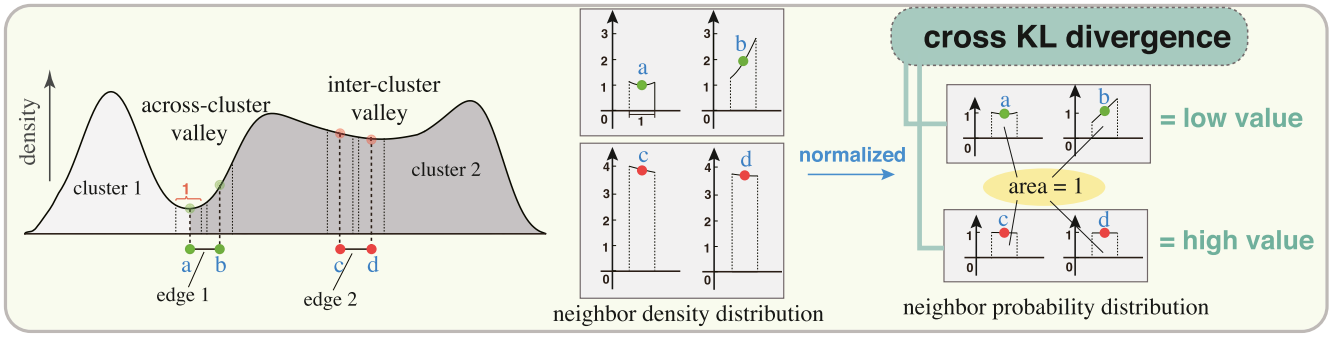


Fig. 4. The motivation of the design of our cross KL edge weight function.

of edges in graph $G(X, E)$ as:

$$w_{ij} = \begin{cases} e^{-\lambda \cdot (D(\mathcal{P}_{x_i}||\mathcal{P}_{x_j}))} & e_{ij} \in E \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where parameter $\lambda > 0$ is user-preset (default is 1). Since $D(\mathcal{P}_{x_i}||\mathcal{P}_{x_j}) \in [0, +\infty)$, we have $w_{ij} \in [0, 1]$ that indicates the distribution similarity between points x_i and x_j.

Based on $G(X, E, w)$, for dataset X, we can easily obtain its adjacency matrix $A \in \mathbb{R}^{n \times n}$ as:

$$A_{ij} = \begin{cases} 1 & \exists e_{ij} \in E \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

and then, define its similarity matrix $S \in \mathbb{R}^{n \times n}$ as:

$$S_{ij} = w_{ij} \times A_{ij} \tag{15}$$

Discussion 2 is a detailed analysis of the relationship between Cross KL edge weight and data density distribution.

Discussion 2: Given a point $x_i \in X$, we denote $\mathcal{P}^{(\min)}_{N(x_i)} = \min_{x_z \in N(x_i)}(\mathcal{P}_{x_i}(x_z))$ and $\mathcal{P}^{(\max)}_{N(x_i)} = \max_{x_z \in N(x_i)}(\mathcal{P}_{x_i}(x_z))$. If point x_j is x_i's adjacent point in G (i.e., $e_{ij} \in E$), $D_{KL}(\mathcal{P}_{x_i}||\mathcal{P}_{x_j}) \leqslant (k+1)(\mathcal{P}^{(\max)}_{N(x_i)} \log \frac{\mathcal{P}^{(\max)}_{N(x_i)}}{\mathcal{P}^{(\min)}_{N(x_j)}})$. Let $\frac{\mathcal{P}^{(\max)}_{N(x_i)}}{\mathcal{P}^{(\min)}_{N(x_j)}} = 1 + \Delta\mathcal{P}_1$. Then, $D_{KL}(\mathcal{P}_{x_i}||\mathcal{P}_{x_j}) \leqslant (k+1)(\mathcal{P}^{(\max)}_{N(x_i)} \log(1 + \Delta\mathcal{P}_1))$. Clearly, when $\Delta\mathcal{P}_1 \to 0$, $D_{KL}(\mathcal{P}_{x_i}||\mathcal{P}_{x_j}) \to 0$. Similarly, $\Delta\mathcal{P}_2 \to 0$, $D_{KL}(\mathcal{P}_{x_j}||\mathcal{P}_{x_i}) \to 0$, where $1 + \Delta\mathcal{P}_2 = \frac{\mathcal{P}^{(\max)}_{N(x_j)}}{\mathcal{P}^{(\min)}_{N(x_i)}}$. Therefore, $\Delta\mathcal{P}_1$ and $\Delta\mathcal{P}_2$ (i.e., the maximum probability difference between the neighboring areas of points x_i and x_j) determines the upper limit of $D(\mathcal{P}_{x_i}||\mathcal{P}_{x_j})$.

Let average density $\rho^{(ave)}_{N(x_i)} = \frac{1}{1+k} \cdot \sum_{x_z \in N(x_i)}(\rho_{x_z})$, max density $\rho^{(\max)}_{N(x_i)} = \max_{x_z \in N(x_i)}(\rho_{x_z})$, and corresponding density difference $\Delta\rho_1 = \rho^{(\max)}_{N(x_i)} - \rho^{(ave)}_{N(x_i)}$. Then according to (10), $\mathcal{P}^{(\max)}_{N(x_i)} = \frac{1}{1+k} \cdot (1 + \frac{\Delta\rho_1}{\rho^{(ave)}_{N(x_i)}})$. Similarly, we can have $\mathcal{P}^{(\min)}_{N(x_j)} = \frac{1}{1+k} \cdot (1 - \frac{\Delta\rho_2}{\rho^{(ave)}_{N(x_j)}})$, where $\Delta\rho_2 = \rho^{(ave)}_{N(x_j)} - \rho^{(\min)}_{N(x_j)}$.

So, $\Delta\mathcal{P}_1 = \frac{1}{k+1}(\frac{\Delta\rho_1}{\rho^{(ave)}_{N(x_i)}} + \frac{\Delta\rho_2}{\rho^{(ave)}_{N(x_i)}})$. In the same way, we have $\Delta\mathcal{P}_2 = \frac{1}{k+1}(\frac{\Delta\rho_3}{\rho^{(ave)}_{N(x_i)}} + \frac{\Delta\rho_4}{\rho^{(ave)}_{N(x_i)}})$, where $\Delta\rho_3 = \rho^{(ave)}_{N(x_i)} - \rho^{(\min)}_{N(x_i)}$, and $\Delta\rho_4 = \rho^{(\max)}_{N(x_j)} - \rho^{(ave)}_{N(x_j)}$. Therefore, once $\Delta\mathcal{P}_1 + \Delta\mathcal{P}_2 \to 0$, $D(\mathcal{P}_{x_i}||\mathcal{P}_{x_j}) \to 0$, i.e., their distribution similarity $w_{ij} \to 1$.

Let $\Delta\mathcal{P}^* = \Delta\mathcal{P}_1 + \Delta\mathcal{P}_2 = \frac{1}{k+1}(\frac{\Delta\rho_1+\Delta\rho_3}{\rho^{(ave)}_{N(x_i)}} + \frac{\Delta\rho_2+\Delta\rho_4}{\rho^{(ave)}_{N(x_j)}}) = \frac{1}{k+1}(\frac{\rho^{(\max)}_{N(x_i)}-\rho^{(\min)}_{N(x_i)}}{\rho^{(ave)}_{N(x_i)}} + \frac{\rho^{(\max)}_{N(x_j)}-\rho^{(\min)}_{N(x_j)}}{\rho^{(ave)}_{N(x_j)}})$. By introducing the concept of *density fluctuation range* of x_i's neighboring area as $R_{\rho_{N(x_i)}} = \rho^{(\max)}_{N(x_i)} - \rho^{(\min)}_{N(x_i)}$, we have $\Delta\mathcal{P}^* = \frac{1}{k+1}(\frac{R_{\rho_{N(x_i)}}}{\rho^{(ave)}_{N(x_i)}} + \frac{R_{\rho_{N(x_j)}}}{\rho^{(ave)}_{N(x_j)}})$. Obviously, a local density fluctuation range is no larger than the global density fluctuation range, i.e., $R_{\rho_{N(x_i)}} \leqslant R_{\rho_X} = [\min_{x_z \in X}(\rho_{x_z}), \max_{x_z \in X}(\rho_{x_z})]$. Thus, $\Delta\mathcal{P}^* \leqslant \frac{1}{k+1}(\frac{R_{\rho_X}}{\rho^{(ave)}_{N(x_i)}} + \frac{R_{\rho_X}}{\rho^{(ave)}_{N(x_j)}})$. For a dataset X, its R_{ρ_X} is fixed, so if $\rho^{(ave)}_{N(x_i)}$ and $\rho^{(ave)}_{N(x_j)}$ are large values, $\Delta\mathcal{P}^*$ tends to have a small value, thereby, obtaining a high distribution similarity w_{ij} value. In other words, if point x_i and its adjacent point x_j (connected by e_{ij}) in a relatively high-density area, they tend to have a high distribution similarity w_{ij} value.

Therefore, points x_i and x_j within a high-density area tend to have a high distribution similarity w_{ij}.

Fig. 5 shows the proposed graph and the visualization of edge weights on the *Agg* dataset [36] (a synthetic dataset). As shown, our Y-graph efficiently captures the structure of the whole dataset (left panel), and edges in high-density areas generally have higher weights (darker green) than those in low-density areas (right panel).

In the following part, a density-based cluster assumption is introduced to guide the clustering (i.e., graph cutting) task on the proposed Y-graph.

C. Our Cluster Assumption

To group density-connected points of similar density into clusters, we propose our own cluster assumption:

Assumption 1: A cluster is composed of a maximum set of adjacent points with similar density distributions.

According to Assumption 1, the Y-graph is to cut graph $G(X, E, w)$ into clusters of maximum distribution similarity (i.e., maximum weight value) with the minimum cutting weight cost, which is exactly what can be achieved by Ncut clustering [20]. According to Discussion 2, our Y-graph-cutting-based

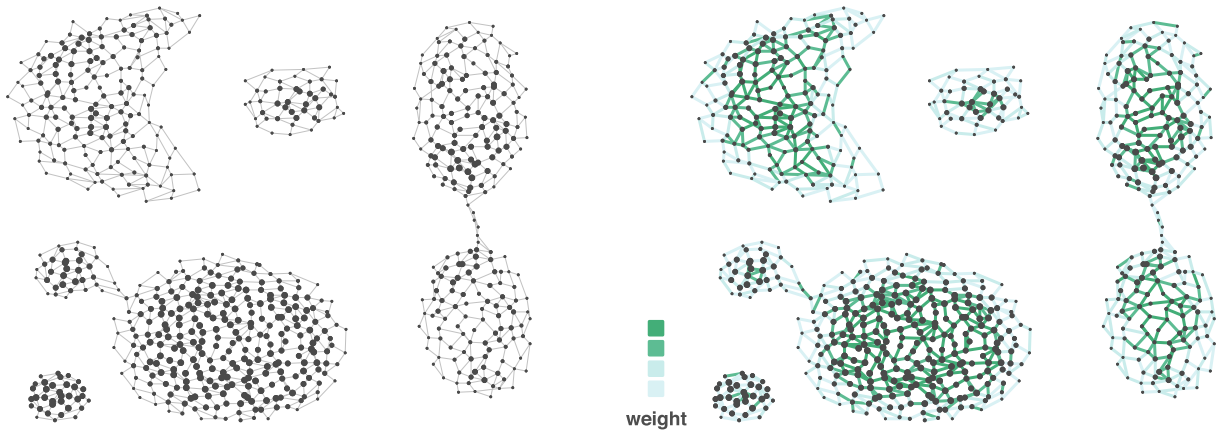


Fig. 5. The edge weight visualization of our Y-graph on the *Agg* dataset, where the size of the black point reflects its density size, i.e., a big point will have a large density.

clustering idea can well capture arbitrary-shaped clusters within high-density areas by cutting edges within across-cluster valley areas [38].

By applying Ncut solver with the input of our Y-graph-based similarity matrix S and cluster number C, i.e., $Ncut(S, C)$, the clustering result $Cl = \{Cl_1, Cl_2, \ldots, Cl_C\}$ can be obtained.

D. Pseudocode and Complexity

Algorithm 1 shows the pseudocode of the proposed clustering algorithm with three steps: 1) graph building; 2) weight evaluation; 3) Ncut solver.

The Y-graph building (step 1) needs a computational complexity of $O(n\log(n) + nk)$. To construct a graph, the first step is to perform a fast tree-based (e.g. kd-tree [48], cover trees [49]) kNN search with a complexity of $O(n\log(n))$. Then, within the neighborhood of each point, we identify two higher-density points that satisfy certain conditions and establish connections with them, with a complexity of $O(nk)$. The weight evaluation for each edge (step 2) requires a complexity of $O(nk)$, where k represents the number of neighbors for a given point. The Ncut solver (step3) requires $O(C|E| + CTn)$, where the Lanczos algorithm [39] with $O(C|E|)$ is used to solve the eigenvectors of Y-graph, and K-means with $O(CTn)$ is applied to partition these eigenvectors into clusters. Parameter C represents the given number of clusters, $|E|$ represents the number of edges in the Y-graph, and T means the iterations of K-means. Since $n < |E| < 2n$ as discussed in Section III-A, the overall complexity of our Ncut solver step is $O(Cn + CTn) = O(CTn)$.

Therefore, the overall time complexity of Ncut-Y clustering is $O(n\log(n) + nk + CTn)$,where k, T and C are far less than n.

IV. Expreiments

A. Experimental Set up

Datasets: The proposed algorithm is evaluated by a set of eighteen synthetic datasets and ten widely-used real-world datasets, as listed in Table I.

Comparison algorithms and settings: a state-of-the-art spectral clustering technique RESKM [29], four classic clustering algorithms (K-means [33], DBSCAN [5], Self-tuning Spectral Clustering (SSC) [41], and DPC [12]), three traditional Ncut [20] versions based on kNN-graph, ϵ-graph, and fully connected graph, respectively (denoted as Ncut-k, Ncut-ϵ, and Ncut-f, respectively), and the proposed Ncut-Y are presented for comparison.

Parameters are usually determined by evaluating the performance of algorithms across a wide range of configurations. Herein, K-means, SSC, and four versions of Ncut are evaluated ten times in different parameter settings, and the configuration that yields the best is chosen.

Data preprocessing: To preprocess datasets and mitigate the impact of dimensional metrics differences, the min-max normalization technique is employed [42], which rescales the values of each feature within a specific range (typically 0 to 1).

Machine configuration: The specified machine configuration consists of a Mac-Book Pro with a 2.9 GHz Intel Core i5 processor and 8 GB of RAM. The operating environment is Matlab (r2017b).

Evaluation metric: The popular Adjusted Rand Index (ARI) [51], Adjusted Mutual Information (AMI) [51], Normalized Mutual Infromation (NMI) [52], and Silhouette Coefficient (SC) [53] are used for the performance evaluation.

B. Experiments on Synthetic Datasets

1) Comparisons With kNN-Graph and ϵ-Graph: The Y-graph is compared with the kNN-graph and the ϵ-graph in terms of their capability and efficiency in capturing structural patterns in the *Eyes* [41] and *Impossible* (without noise) [2] datasets, both of which exhibit complex-shaped and imbalanced clusters. *Eyes* consists of three imbalanced clusters: two dense square-shaped clusters surrounded by one sparse ring-shaped cluster (see Fig. 6). Meanwhile, *Impossible* contains seven imbalanced clusters with various shapes, including spherical, ellipsoidal, nested spirals, and nested rings (see Fig. 7). These characteristics indicate that clustering these datasets is extremely challenging [2].

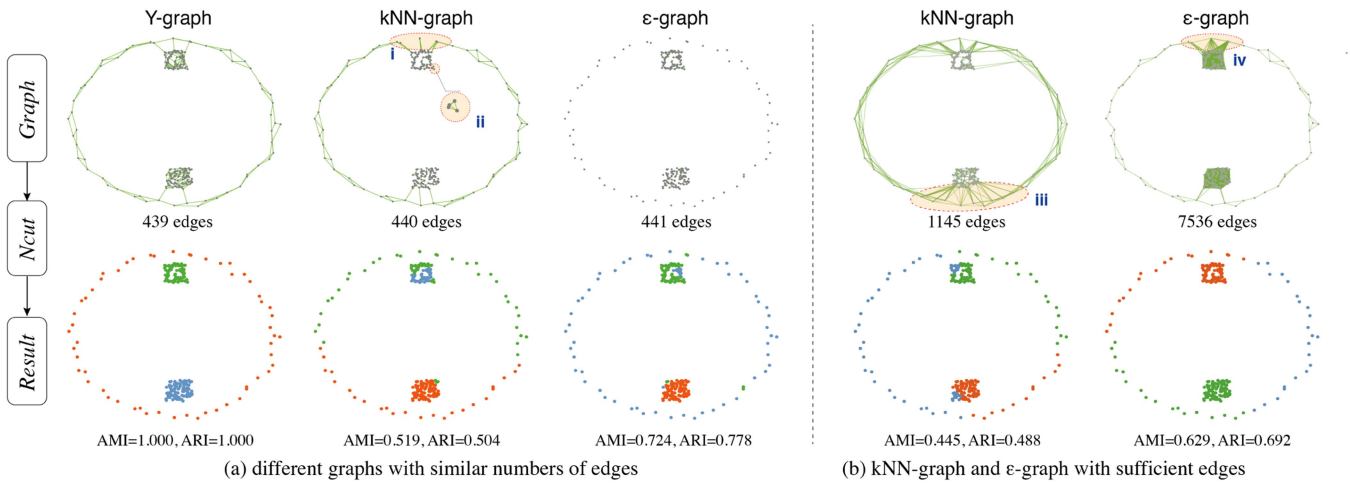


Fig. 6. The capability of the Y-graph, kNN-graph, and ϵ-graph to capture the structure of the *Eyes* dataset using Ncut.

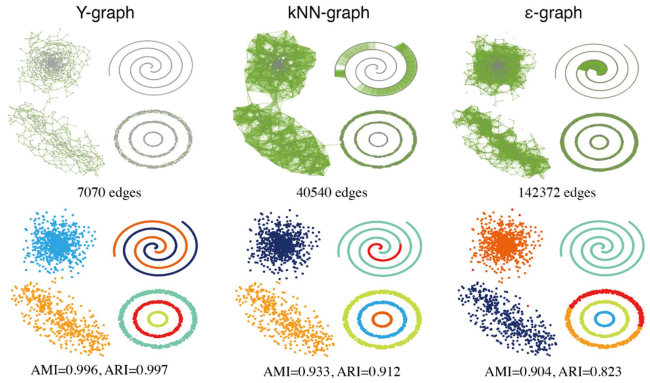


Fig. 7. The capability of the Y-graph, kNN-graph, and ϵ-graph to capture the structure of the *Impossible* dataset using Ncut.

For the *Eyes* dataset, we make two comparisons regarding the Y-graph, kNN-graph, and ϵ-graph: first, evaluating structural representation performance under similar numbers of edges; second, assessing performance when each graph has a sufficient number of edges to connect points within each cluster well.

Fig. 6(a) shows the structural representation performance comparison of different graphs with similar numbers of edges: the Y-graph (439 edges), kNN-graph (440 edges), and ϵ-graph (441 edges). As shown in Fig. 6(a) (upper panel), the ϵ-graph allocated all 441 edges to the dense square-shaped clusters, neglecting the ring-shaped cluster. This occurred because a small ϵ value was required to limit the edges to 441, resulting in the failure to capture the sparse ring shape. Despite this, it did not effectively capture the high interconnectivity of the square-shaped clusters. Unlike the ϵ-graph, the kNN-graph allocated an appropriate number of edges to the ring-shaped cluster but couldn't fully link it (see "area **i**"). Moreover, it failed to connect the two dense clusters because we set $k = 3$ to limit the edges to 441, resulting in insufficient connectivity (such as "area **ii**"). Compared to the kNN-graph and ϵ-graph, the Y-graph effectively represented the structure of the *Eyes* dataset with 439 edges. It fully connected the sparse ring-shaped cluster with a modest number of edges, while ensuring high interconnectivity within the dense square-shaped clusters with a larger number of edges. Additionally, edges between clusters were sparse. This is because the Y-graph, like the kNN-graph, can adjust the connectivity range by tuning the value of k. However, unlike the kNN-graph, the Y-graph limits each point to at most two high-density connections, maintaining sparsity consistently.

Then, by using the Ncut solver to the above Y-graph, kNN-graph, and ϵ-graph, yielding clustering results on the *Eyes* dataset. As shown in Fig. 6(a) (bottom panel) Ncut-Y effectively segmented the dataset, whereas Ncut-k and Ncut-ϵ faltered due to their inherent graph limitations.

Fig. 6(b) compares the structural representation performance of different graphs with sufficient edges. The kNN-graph requires 1,145 edges to effectively connect the ring-shaped cluster, while the ϵ-graph needs 7,536 edges. However, neither graph is well-suited for *Eyes*, as they lack dense edges within clusters and have sparse connections across clusters. This deficiency stems from the severe imbalance in cluster distribution. Specifically, the maximum gap within the sparse ring-shaped cluster exceeds the minimum gap between the ring-shaped cluster and the square-shaped clusters. Consequently, in ensuring effective connectivity among data points in the ring-shaped cluster, both the kNN-graph and ϵ-graph connect denser edges between clusters (see "area **iii** and **iv**"), leading directly to Ncut's failure in effectively partitioning *Eyes* based on these graphs. Similar issues also occur in the clustering on the *Impossible* dataset.

Fig. 7 displays graphs generated from the *Impossible* dataset comprising 3,795 points. The kNN-graph utilizes 40,540 edges, the ϵ-graph uses 142,372 edges, while our Y-graph employs only 7,070 edges. Due to imbalance cluster distribution and shape inconsistencies, both the kNN-graph and the ϵ-graph require larger parameters to effectively connect points within clusters, leading to many cross-cluster connections between two spiral-shaped clusters (the two clusters in the upper left corner

Algorithm 1: Ncut-Y Clustering.

Input: dataset $X = \{x_1, x_2, \ldots, x_n\}$ parameter k, and cluster number C.
Output: the clustering result $Cl = \{Cl_1, Cl_2, \ldots, Cl_C\}$
obtain kNN distances, by using fast kNN technique [40].
estimate density $\rho = \{\rho_1, \rho_2, \ldots, \rho_n\}$, according to (5).
// **Y-graph building**
for each point $x_i \in X$ **do**
  for each neighbor $x_i^{(K)} \in N_k(x_i)$, by K from 1 to k **do**
    if $\rho_{x_i^{(K)}} > \rho_{x_i}$ **then**
      $\delta_{x_i} \leftarrow x_i^{(K)}$ // obtain first outdegree point.
      break
    end if
  end for
  if $\delta_{x_i} = \varnothing$ **then**
    $\delta_{x_i} = x_i$
  end if
end for
for each point $x_i \in X$ **do**
  for each neighbor $x_i^{(K)} \in N_k(x_i)$, by K from 1 to k **do**
    find the second point γ_{x_i}, according to (9)
  end for
  if $\gamma_{x_i} = \varnothing$ **then**
    $\gamma_{x_i} = x_i$
  end if
end for
obtain edges E between associated points (i.e., x_i between its δ_{x_i} and γ_{x_i}.)
building a Y-graph $G(X, E)$
// **cross KL edge weight evaluation**
for each edge $e_{ij} \in E$ **do**
  obtain w_{ij}, according (13).
end for
obtain a weight Y-graph $G(X, E, w)$
obtain similarity matrix S based on graph $G(X, E, w)$, according (15)
// **Ncut solver**
execute $Cl = Ncut(S, C)$
returnClustering result $Cl = \{Cl_1, Cl_2, \ldots, Cl_C\}$.

of *Impossible*), resulting in the failure of Ncut-k and Ncut-ϵ to separate clusters accurately. In contrast, the Y-graph aptly captures cluster structures with fewer edges, distributing them based on data density (each point corresponds to two edges pointing to high-density regions), ensuring high connectivity within clusters and low connectivity between clusters. Consequently, Ncut-Y successfully partitions the seven clusters of the *Impossible* dataset.

The above experiments effectively demonstrate the efficacy and capability of the Y-graph in structural capturing.

TABLE I
DATASETS

Dataset	Instances	Attributes	Clusters	Source
Eyes	238	2	3	[41]
Impossible (no noise)	3795	2	7	[2]
Agg	788	2	7	[36]
Flame	240	2	3	[36]
Pathbased	300	2	3	[36]
Spiral	312	2	3	[36]
Threecircles	299	2	3	[41]
Jain	373	2	2	[36]
R15	600	2	15	[36]
S1	600	2	15	[36]
D31	3100	2	31	[36]
A3	7500	2	50	[36]
Birchrg1	100000	2	100	[37]
Atom	800	3	2	[37]
Cuboids	1002	3	4	[37]
Chainlink	1000	3	2	[37]
Tetra	400	3	4	[37]
Hypercube	800	3	8	[37]
Iris	150	4	3	[43]
Wine	178	13	3	[43]
Movementlibras	360	90	15	[43]
Waveform	5000	21	3	[43]
Drivedata	606	6400	4	[43]
Breastcancer	569	30	2	[43]
OlivettiFaces	400	92×112	40	[50]
YTF	10000	10	41	[44]
USPS	11000	10	10	[45]
MNIST	10000	500	10	[46]

2) Comparisons Among the State-of-the-Art Algorithms: The performance of the proposed Ncut-Y algorithm in reconstructing complex-shaped clusters was evaluated on different synthetic datasets with varied shapes. Fig. 8 shows the clustering results, where each color represents a different cluster. As shown, the Ncut-Y algorithm achieves near-perfect reconstruction of the complex-shaped clusters across all tested datasets.

Table II provides the AMI, ARI, NMI, and $\mathrm{S_{SC}}$ scores of all algorithms with the best results being highlighted, where metric $\mathrm{S_{SC}}$ represents the Silhouette Coefficient (SC) similarity between the clustering result and true label, as:

$$\mathrm{S_{SC}} = 1 - \frac{|\mathrm{SC}(\text{true label}) - \mathrm{SC}(\text{clustering result})|}{2} \quad (16)$$

Since $\mathrm{SC} \in [-1, 1]$, so we have $\mathrm{S_{SC}} \in [0, 1]$. The higher the similarity between the clustering result and the true label, the greater the $\mathrm{S_{SC}}$ value. It's crucial to emphasize that $\mathrm{S_{SC}}$ offers a broad approximation of the silhouette coefficient and doesn't entirely substitute the evaluation of clustering accuracy. Instead, it complements the assessment process as a supplementary scoring metric.

From Table II we can see that Ncut-Y outperforms other algorithms on most of the synthetic datasets, showcasing its high performance in cluster reconstruction. As verified, the proposed Ncut-Y algorithm exhibits excellent capabilities in effectively reconstructing complex-shaped clusters.

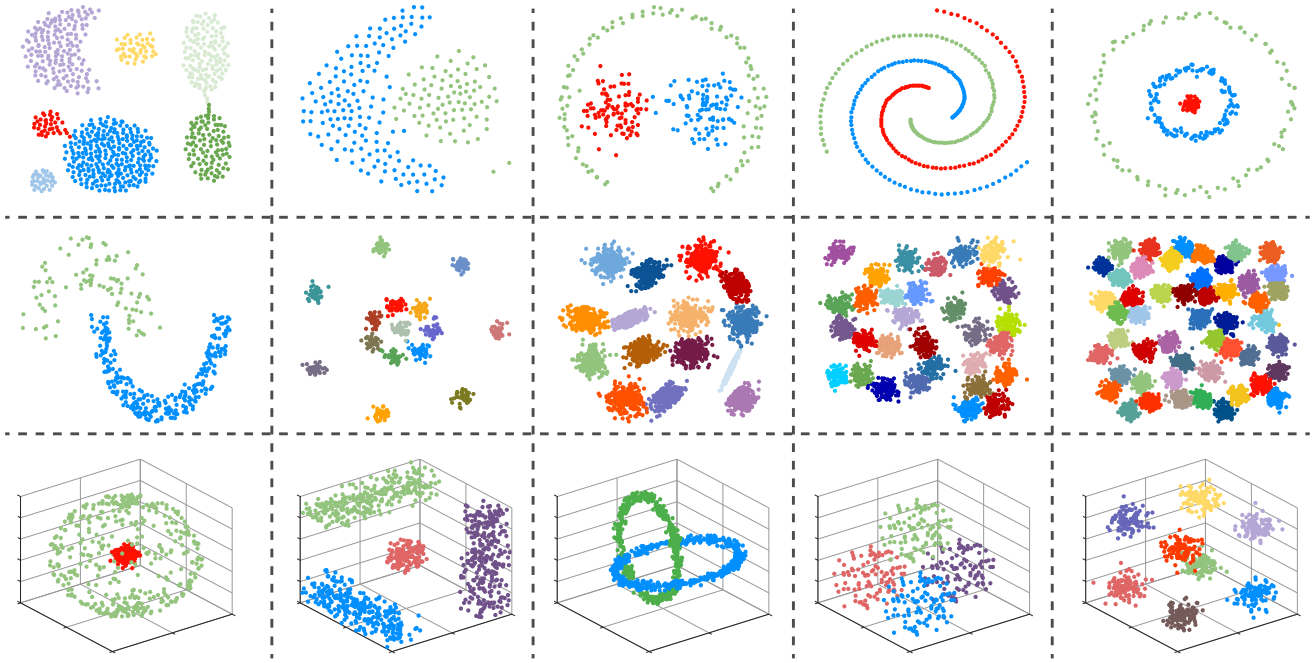


Fig. 8. The clustering results of the proposed Ncut-Y on 15 tested synthetic datasets (in Table I) of different shapes.

TABLE II
THE COMPARISON OF AMI, ARI, FMI, AND S_{SC} ON SYNTHETIC DATASETS

Dataset	RESKM	K-means	SSC	DBSCAN	DPC	Ncut-ϵ	Ncut-k	Ncut-f	Ncut-Y
Eyes	0.857 0.902	0.573 0.643	0.793 0.766	**1.000 1.000**	0.489 0.456	0.629 0.692	0.445 0.488	0.615 0.679	**1.000 1.000**
	0.938 0.965	0.789 0.863	0.848 0.891	**1.000 1.000**	0.644 0.968	0.813 0.886	0.665 0.958	0.807 0.879	**1.000 1.000**
Impossible	0.940 0.922	0.698 0.629	0.935 0.915	0.907 0.891	0.732 0.659	0.904 0.823	0.933 0.912	0.577 0.488	**0.996 0.997**
	0.935 0.984	0.693 0.770	0.929 0.980	0.914 0.913	0.726 0.849	0.854 0.878	0.927 0.978	0.571 0.857	**0.997 0.999**
Agg	0.903 0.909	0.821 0.745	0.943 0.945	0.969 0.983	**0.992 0.996**	0.980 0.986	0.971 0.978	0.751 0.661	0.983 0.989
	0.929 0.987	0.802 0.998	0.957 0.992	0.987 0.989	**0.997 0.999**	0.989 0.997	0.983 0.995	0.732 0.998	0.992 0.997
Flame	0.907 0.950	0.433 0.476	0.539 0.612	0.839 0.944	**1.000 1.000**	0.927 0.967	0.336 0.253	0.897 0.950	0.932 0.967
	0.977 0.995	0.747 0.946	0.814 0.954	0.974 0.957	**1.000 1.000**	0.985 0.998	0.647 0.964	0.977 0.996	0.985 0.997
Pathbased	0.743 0.711	0.510 0.461	0.584 0.538	0.801 0.854	0.500 0.453	0.567 0.519	0.725 0.684	0.538 0.488	**0.923 0.951**
	0.809 0.927	0.662 0.778	0.701 0.814	0.902 0.981	0.659 0.779	0.691 0.803	0.792 0.908	0.675 0.786	**0.967 0.999**
Spiral	0.111 0.096	-0.005 -0.006	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	0.634 0.585	**1.000 1.000**
	0.404 0.763	0.328 0.691	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	0.729 0.971	**1.000 1.000**
Threecircles	**1.000 1.000**	0.160 0.057	**1.000 1.000**	**1.000 1.000**	0.150 0.014	**1.000 1.000**	**1.000 1.000**	0.604 0.509	**1.000 1.000**
	1.000 1.000	0.404 0.668	**1.000 1.000**	**1.000 1.000**	0.425 0.937	**1.000 1.000**	**1.000 1.000**	0.726 0.657	**1.000 1.000**
Jain	0.588 0.679	0.492 0.577	0.635 0.733	0.865 0.976	0.618 0.715	0.540 0.618	**1.000 1.000**	0.529 0.626	**1.000 1.000**
	0.866 0.964	0.820 0.960	0.890 0.965	0.991 0.711	0.882 0.966	0.839 0.961	**1.000 1.000**	0.843 0.961	**1.000 1.000**
R15	0.958 0.931	**0.994 0.993**	**0.994 0.993**	0.983 0.982	**0.994 0.993**	0.991 0.989	**0.994 0.993**	0.988 0.986	0.991 0.989
	0.936 0.921	**0.993** 0.998	**0.993** 0.998	0.983 0.998	**0.993** 0.998	0.990 0.998	**0.993** 0.998	0.987 **0.999**	0.990 **0.999**
S1	0.996 0.997	0.922 0.829	0.996 0.995	0.958 0.967	**0.997** 0.997	**0.997 0.998**	0.994 0.992	0.995 0.996	0.996 0.996
	0.997 **0.999**	0.845 0.936	0.996 0.998	0.969 0.985	0.997 **0.999**	**0.998 0.999**	0.993 0.997	0.996 **0.999**	0.996 **0.999**
D31	0.920 0.842	0.913 0.821	0.960 0.943	0.865 0.710	0.954 0.933	**0.965 0.951**	0.964 0.950	0.964 0.949	0.957 0.939
	0.849 0.902	0.827 0.967	0.945 0.991	0.722 0.937	0.935 0.994	**0.953 0.995**	0.951 0.991	0.951 0.991	0.941 0.994
A3	0.950 0.885	0.967 0.911	0.993 0.990	0.902 0.736	0.987 0.980	0.985 0.977	**0.994 0.992**	0.955 0.912	0.985 0.979
	0.888 0.901	0.914 0.963	0.990 **0.999**	0.746 0.938	0.980 0.998	0.977 0.998	**0.992 0.999**	0.915 0.978	0.979 0.997
Atom	**1.000 1.000**	0.261 0.189	**1.000 1.000**	0.432 0.636	0.164 0.085	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**
	1.000 1.000	0.656 0.857	**1.000 1.000**	0.797 0.981	0.647 0.884	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**
Cuboids	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	0.995 0.999	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**
	1.000 1.000	**1.000 1.000**	**1.000 1.000**	0.999 0.949	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**
Chainlink	**1.000 1.000**	0.064 0.088	**1.000 1.000**	**1.000 1.000**	0.352 0.302	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**
	1.000 1.000	0.544 0.866	**1.000 1.000**	**1.000 1.000**	0.686 0.931	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**
Tetra	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	0.834 0.899	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**
	1.000 1.000	**1.000 1.000**	**1.000 1.000**	0.924 0.953	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**
Hypercube	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	0.987 0.994	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**
	1.000 1.000	**1.000 1.000**	**1.000 1.000**	0.995 0.996	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**	**1.000 1.000**

Note: the four metric scores is recorded as $\begin{pmatrix} \text{AMI} & \text{ARI} \\ \text{FMI} & S_{SC} \end{pmatrix}$.

TABLE III
THE COMPARISON OF AMI, ARI, FMI, AND S_{SC} ON REAL-WORLD DATASETS. NOTE: THE FOUR METRIC SCORES IS RECORDED AS $\left(\begin{smallmatrix}\text{AMI} & \text{ARI}\\ \text{FMI} & S_{SC}\end{smallmatrix}\right)$

Dataset	RESKM	K-means	SSC	DBSCAN	DPC	Ncut-ϵ	Ncut-k	Ncut-f	Ncut-Y
Iris	0.847 0.868	0.733 0.716	0.854 0.868	0.577 0.568	0.861 0.886	0.848 0.884	0.883 0.904	0.862 0.886	**0.911 0.920**
	0.911 0.982	0.811 0.960	0.912 0.979	0.772 0.904	0.923 **0.999**	0.922 0.989	0.936 0.981	0.923 0.981	**0.946** 0.983
Wine	0.824 0.828	0.874 0.899	0.889 0.915	0.528 0.450	0.707 0.672	**0.908 0.933**	0.874 0.899	0.893 0.917	0.857 0.882
	0.885 **0.998**	0.933 0.991	0.944 0.991	0.680 0.967	0.784 0.987	**0.955** 0.992	0.933 0.991	0.945 0.992	0.922 0.996
Movementlibras	0.522 0.313	0.530 0.312	0.553 0.339	0.266 0.090	0.485 0.262	0.595 0.391	0.565 0.358	0.551 0.346	**0.620 0.426**
	0.360 0.861	0.362 0.797	0.386 0.831	0.262 0.997	0.346 **0.993**	0.445 0.815	0.417 0.820	0.407 0.817	**0.466** 0.851
Wavefrom	**0.394 0.291**	0.364 0.254	0.369 0.251	0.006 0.000	0.221 0.190	0.361 0.250	0.370 0.251	0.368 0.252	0.366 0.251
	0.530 0.917	**0**.504 0.893	0.501 0.895	0.489 0.650	0.477 **0.957**	0.500 0.893	0.501 0.894	0.502 0.893	0.501 0.899
Drivedata	0.631 0.665	0.524 0.505	0.467 0.439	0.520 0.561	0.602 0.605	0.254 0.136	0.257 0.135	0.254 0.136	**0.639 0.673**
	0.753 0.924	0.640 0.884	0.593 0.914	0.667 **0.962**	0.710 0.924	0.393 0.854	0.395 0.853	0.393 0.854	**0.759** 0.918
Breastcancer	**0.756 0.850**	0.611 0.730	0.691 0.799	0.376 0.503	0.415 0.471	0.524 0.654	0.695 0.805	0.668 0.773	0.735 0.837
	0.931 0.983	0.877 0.969	0.908 0.977	0.772 **0.999**	0.786 0.991	0.841 0.984	0.910 0.978	0.897 0.968	0.924 0.986
OlivettiFace	0.764 0.631	0.735 0.584	0.861 0.779	0.737 0.590	0.731 0.543	0.759 0.640	0.839 0.747	0.760 0.632	**0.867 0.784**
	0.643 0.967	0.599 0.985	0.785 0.986	0.599 0.968	0.563 0.940	0.651 0.990	0.754 0.990	0.642 0.978	**0.790 0.994**
YTF	0.770 0.547	0.755 0.574	0.735 0.480	0.672 0.385	0.733 0.498	0.729 0.518	0.782 0.569	0.750 0.536	**0.808 0.663**
	0.563 0.920	0.590 0.808	0.521 **0.999**	0.407 0.823	0.524 0.973	0.535 0.855	0.587 0.914	0.554 0.826	**0.676** 0.853
USPS	**0.798 0.737**	0.607 0.509	0.695 0.540	0.265 0.029	0.393 0.293	0.694 0.586	0.721 0.627	0.662 0.569	0.764 0.656
	0.764 0.989	0.560 0.948	0.598 **0.991**	0.296 0.581	0.404 0.823	0.635 0.963	0.669 **0.991**	0.616 0.958	0.695 0.986
MNIST	**0.940 0.949**	0.839 0.793	0.888 0.826	0.561 0.238	0.393 0.293	0.816 0.756	0.882 0.826	0.812 0.753	0.937 0.940
	0.955 0.998	0.817 0.989	0.846 **0.999**	0.395 0.746	0.404 0.823	0.782 **0.999**	0.845 0.988	0.779 0.998	0.946 **0.999**

Note: the four metric scores is recorded as $\left(\begin{smallmatrix}\text{AMI} & \text{ARI}\\ \text{FMI} & S_{SC}\end{smallmatrix}\right)$.

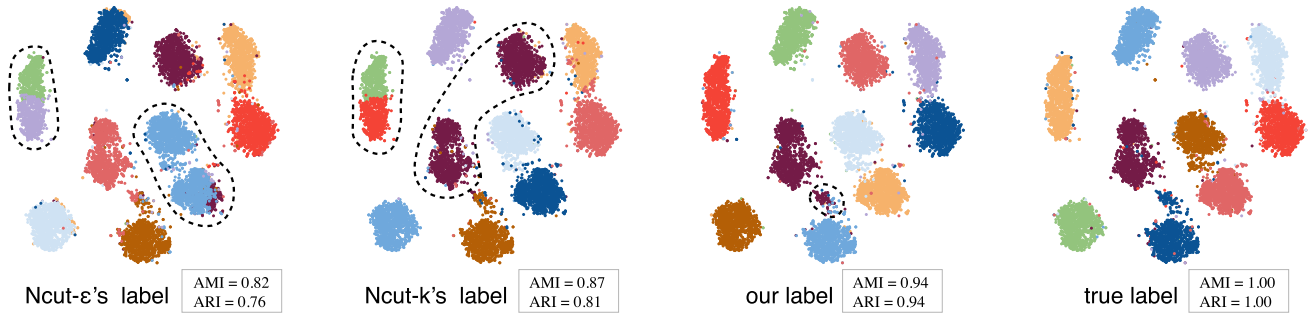


Fig. 9. The t-SNE-based visualization comparison between the true labels and clustering labels of different clustering algorithms on *MNIST*.

C. Experiments on Real-World Datasets

To assess the clustering performance of the Ncut-Y algorithm on high-dimensional and large-sized datasets, experiments were conducted on ten real-world datasets, including six UCI [43] datasets (*Iris*, *Wine*, *Movementlibras*, *Waveform*, *Drivedata*, and *Breastcancer*), three widely used machine learning datasets of large size (*YTF* [44], *USPS* [45], and *MNIST* [46] preprocessed by [10]), and the well-known *OlivettiFaces* preprocessed by [15]. Detailed information can be found in Table I.

Table III lists the clustering results of the experiments, with the best results being highlighted. As shown, Ncut-Y delivers outstanding performance overall, surpassing other algorithms in most tested datasets.

Fig. 9 illustrates t-SNE [47] data visualizations of the *MNIST* dataset, where the data points are labeled with the clustering labels obtained from three versions of Ncut (Ncut-ϵ, Ncut-k, and Ncut-Y), as well as the true labels. Regions, where main division errors occur, are outlined with black dashed lines. As shown, Ncut-Y achieves a highly accurate partitioning result that is close to the true labels, while Ncut-k and Ncut-ϵversions yield unsatisfactory results with noticeable division errors. In other words, Ncut-Ncut-Y can almost accurately partition the *MNIST* dataset, while Ncut-k and Ncut-ϵ versions exhibit limitations and produce suboptimal clustering outcomes.

Therefore, the proposed Ncut-Y algorithm is a promising alternative method for data clustering.

D. The Efficiency of Ncut-Y

1) The Effectiveness of Y-Graph: The effectiveness of capturing cluster structures is crucial for the success of graph-based clustering algorithms, particularly in the context of large-scale data sets. By accurately reflecting the internal and inter-cluster relationships based on data density distribution, the Y-graph can ensure a robust representation of clusters, making it an essential tool for Ncut to achieve high-quality clustering results.

To show the effectiveness and robustness of both the Y-graph and our edge weight function, we introduced a series of ablation experiments (about Ncut-Y, Ncut-k, the Gaussian kernel-based edge weight function f_{w1}, and our Cross KL edge weight function f_{w2}) on several two-dimensional synthetic datasets and high-dimensional real-world datasets, and the corresponding results are displayed in Table IV.

As shown in Table IV, Ncut-Y exhibits superior performance compared to Ncut-k, with function f_{w2} showing better results

TABLE IV
THE ABLATION EXPERIMENTS FOR NCUT-Y AND NCUT-k WITH DIFFERENT EDGE WEIGHT FUNCTIONS

Dataset	Agg ($d=2$)			Eyes ($d=2$)			Pathbased ($d=2$)			Jain ($d=2$)			Drivedata ($d=6400$)			MNIST ($d=500$)		
Metric	AMI	ARI	FMI	AMI	ARI	FMI	AMI	ARI	FMI	AMI	ARI	FMI	AMI	ARI	FMI	AMI	ARI	FMI
Ncut-k (with f_{w1})	0.968	0.975	0.981	0.487	0.448	0.640	0.587	0.541	0.703	0.467	0.515	0.790	0.169	0.131	0.430	0.814	0.719	0.756
Ncut-k (with f_{w2})	0.971	0.978	0.983	0.487	0.448	0.640	0.724	0.683	0.791	0.566	0.652	0.854	0.479	0.452	0.608	0.894	0.830	0.850
Ncut-Y (with f_{w1})	**0.983**	**0.989**	**0.992**	**1.000**	**1.000**	**1.000**	**0.923**	**0.951**	**0.967**	**1.000**	**1.000**	**1.000**	0.221	0.101	0.456	0.407	0.129	0.348
Ncut-Y (with f_{w2})	**0.983**	**0.989**	**0.992**	**1.000**	**1.000**	**1.000**	**0.923**	**0.951**	**0.967**	**1.000**	**1.000**	**1.000**	**0.639**	**0.673**	**0.759**	**0.937**	**0.940**	**0.946**

TABLE V
THE Y-GRAPH BUILDING SPEED ON TEN DIFFERENT SIZE SAMPLING DATASETS OF THE *BIRCHRG1* DATASET

Number of samples	10000	20000	30000	40000	50000	60000	70000	80000	90000	100000
Parameter k	10	20	30	40	50	60	70	80	90	100
Dist_time	0.069	0.132	0.228	0.356	0.564	0.941	0.921	1.185	2.279	2.447
Building_time	0.347	1.064	2.391	4.553	7.201	11.074	16.041	21.966	33.178	44.387
Total time	0.416	1.196	2.619	4.908	7.765	12.016	16.963	23.151	35.457	46.833

TABLE VI
THE GRAPH ATTRIBUTE COMPARISON OF THE KNN-GRAPH AND OUR Y-GRAPH WITH DIFFERENT k VALUES ON THE *YTF* DATASET

Graph attribute	k=100	k=300	k=500	k=1000	k=2000	k=3000	k=5000
Edges	638722/17526	1939884/17537	3239026/17539	6251335/17539	12062952/17539	17690617/17539	28503721/17539
Byte size (MB)	19.5844/0.6032	59.2927/0.6036	98.9394/0.6036	190.8678/0.6036	368.2242/0.6036	539.9669/0.6036	869.9567/0.6036
Solving time (s)	1.1177/1.5454	2.0349/1.5521	2.8982/1.6257	3.9097/1.5600	6.5642/1.7353	10.6712/1.5731	18.9178/1.5371

Note: the format displayed in the table is "kNN-graph/ours".

than function f_{w1}. This suggests that the Y-graph structure outperforms the kNN graph structure in representing cluster structures, and function f_{w2} demonstrates enhanced edge-related performance over function f_{w1}. Notably, when handling high-dimensional datasets: *Drivedata* and *MNIST*, our function f_{w2} demonstrates more clear advantages.

2) The Speed of Ncut-Y: Execution speed and memory consumption are two important criteria for algorithms in large-scale data clustering tasks, while the efficiency of the Ncut clustering framework mainly depends on the adjacency graph (i.e., the adjacency matrix).

The Y-graph construction process involves two primary stages: fast kNN distance computation and graph construction. Fast kNN distance computation takes $O(n\log(n))$, while graph construction takes $O(nk)$. Table V displays the building time of the Y-graph on ten different size sampling datasets of the *Birchrg1* dataset containing 100,000 points. As shown, when building a Y-graph with 10,000 data points, takes approximately 0.416 seconds. While, with 100,000 data points, the construction time increases to around 46.833 seconds. Despite this increase in time for larger datasets, the efficiency of Y-graph construction remains within an acceptable range.

Unlike the traditional kNN-graph, where the number of edges is directly related to the k value, our Y-graph is approximately a 2-outdegree graph, indicating its number of edges will never exceed $2n$. Therefore, our Y-graph structure can maintain a sparse structure while enlarging the correlation radius of points (i.e. increasing the k-value). More importantly, the sparse Y-graph not only consumes less memory but also enables the Ncut solver to have higher execution efficiency.

Table VI shows the graph attribute comparison of the kNN-graph and Y-graph with different k values on the *YTF* dataset (a dataset that contains 10,000 samples of 41 persons' faces) [44]. As shown, when the k value changes from 100 to 5000, the number of edges increases significantly, which directly leads to an increase in its byte size and solving time by the Ncut solver. In contrast, the number of edges in Y-graphs is much smaller and is stable at about 17,500 (less than $2n$, $n = 10,000$), which makes its byte size much smaller than the kNN-graph and its solving time less than kNN-graph's (when k become larger).

It is worth noting that our Y-graph structure consumes extremely low memory (for example, about 0.60MB for the *YTF* dataset with 10,000 points). So, Ncut-Y algorithm can better handle large-scale datasets. Additionally, smaller memory consumption can reduce the hardware cost and resource consumption required to run the algorithm. Therefore, Ncut-Y with a lower space complexity is generally considered a more efficient and feasible choice for large-scale datasets.

TABLE VII
THE RUNTIME COMPARISON OF ALGORITHMS ON DIFFERENT TESTED DATASETS (UNIT: SECOND)

Algorithm	Agg	S1	A3	YFT	USPS	Total
K-means	0.0219	0.8885	0.0538	0.1098	0.1550	1.2290
SSC	0.6155	6.1619	8.8015	27.4584	19.1594	62.1967
DBSCAN	0.0265	0.4443	0.7597	1.9487	2.0919	5.2711
DPC	0.1595	3.7238	5.5106	13.4838	14.6497	37.5274
FastDPeak	0.0473	0.1375	0.1666	0.8990	1.2638	2.5142
Ncut-f	0.4323	7.6918	32.1848	79.6266	77.4136	197.3491
Ncut-Y	0.0570	0.6573	1.8586	5.040	2.1048	9.7177

Table VII presents the runtime of different algorithms on the tested datasets, where FastDPeak [17] is a speed-up version of DPC, and has a time complexity of $O(n\log n)$. It can be observed that Ncut-Y, the fourth-fastest algorithm, completes execution on the *USPS* dataset in under two seconds. This efficiency is partly due to Ncut-Y's use of kNN distances as input, which can be fast computed using fast kNN techniques [40] such as the kd-tree [48] and the cover trees [49]. Additionally, the Y-graph can be efficiently solved using the Ncut solver. While Ncut-Y is slightly slower than FastDPeak, despite both algorithms using kNN as input. This is because Ncut-Y involves

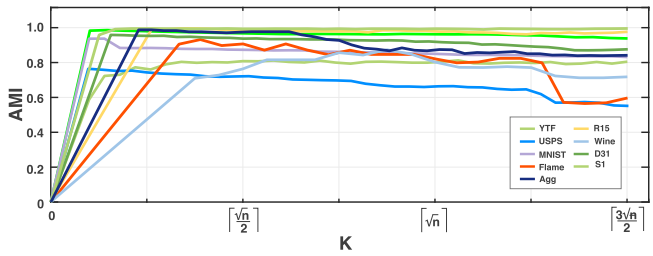


Fig. 10. The k-AMI plot of our algorithm on different datasets with $k \in [0, \lceil \frac{3\sqrt{n}}{2} \rceil]$.

additional time for constructing the Y-graph and performing the final Ncut partitioning.

As verified, the proposed Ncut-Y algorithm with fast speed and low memory consumption is promising for large-scale data clustering.

E. Parameter Insensitivity

The main parameter that requires adjustment in Ncut-Y is k, representing the number of neighbors. By default, k is set within $[\lceil \frac{\sqrt{n}}{2} \rceil, \lceil \sqrt{n} \rceil]$. Since k is crucial in our kNN-based density estimation and graph construction, examining the insensitivity of Ncut-Y to k is necessary.

Fig. 10 illustrates the k-AMI plot of several tested datasets, where k varies within $[0, \lceil \frac{3\sqrt{n}}{2} \rceil]$. As depicted in Fig. 10, Ncut-Y consistently achieves optimal performance within a wide range of k values, particularly around the default setting of $k \in [\lceil \frac{\sqrt{n}}{2} \rceil, \lceil \sqrt{n} \rceil]$. This validates the effectiveness of the chosen range $k \in [\lceil \frac{\sqrt{n}}{2} \rceil, \lceil \sqrt{n} \rceil]$ and demonstrates the insensitivity of Ncut-Y to k.

V. Conclusion

In this work, a max-ascent-angle graph (Y-graph) is proposed for detecting clusters. Besides, by applying the Ncut solver on the Y-graph, the Ncut-Y algorithm is proposed.

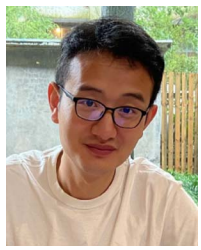
According to the density-based idea that clusters are high-density areas on the data density surface, the proposed "max-ascent-angle" concept effectively helps to distinguish intra-cluster and across-cluster points. By maximizing the ascent angles of all data points, the Y-graph achieves the automatic and reasonable allocation of dense edges in cluster areas, and sparse edges in valley areas. By using the cross KL divergence-based weight function to record the similarity of neighbor probability distribution, the edge weight effectively represents the density connectivity between points. On this basis, the Ncut-Y algorithm effectively seeks and cuts the edges in the valley areas between clusters on the Y-graph, thereby, achieving accurate complex shapes clustering. The Y-graph and the Ncut-Y algorithm excel in domains where data density can be effectively measured, especially in scenarios with low density between clusters. As analyzed, the Y-graph is fast built for only requiring kNN distances, and due to its high sparsity, Ncut-Y is suitable for large data clustering. The efficiency and robustness of Ncut-Y are well verified in the conducted comparison experiments on synthetic datasets and real-world datasets.

Notably, the performance of Y-graph heavily relies on the accuracy of density estimation. Herein, to achieve a fast speed, a relatively simple kNN-based density estimation method is utilized. In terms of the improvement of density estimation performance, it will be addressed as part of our future work. Besides, noise points may also be connected in the Y-graph, potentially impacting the clustering performance of the Ncut algorithm. In future work, we aim to improve the Y-graph and explore robust clustering algorithms to address the potential impacts of noise points on Ncut performance, enhancing its applicability to noisy data. Additionally, similar to the K-means algorithm, Ncut-Y requires the number of clusters as input. Nonetheless, in many real-world applications, automatic cluster detection is necessary. Therefore, we will continue our work in designing an automatic cluster detection technology to further improve the overall performance and applicability of the clustering algorithm.

References

[1] K. A. Jain, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999, doi: 10.1145/331499.331504.
[2] K. A. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010, doi: 10.1016/j.patrec.2009.09.011.
[3] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015, doi: 10.1126/science.aaa8415.
[4] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdiscipl. Rev.: Data Mining Knowl. Discov.*, vol. 1, no. 3, pp. 231–240, 2011, doi: 10.1002/widm.30.
[5] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Kdd*, vol. 96, no. 34, pp. 266–231, 1996.
[6] A. Karami and R. Johansson, "Choosing DBSCAN parameters automatically using differential evolution," *Int. J. Comput. Appl.*, vol. 91, no. 7, pp. 1–11, 2014.
[7] B. K. Sawant, "Adaptive methods for determining DBSCAN parameters," *Int. J. Innov. Sci., Eng. Technol.*, vol. 1, no. 4, pp. 329–334, 2014.
[8] A. Bechini, F. Marcelloni, and A. Renda, "TSF-DBSCAN: A novel fuzzy density-based approach for clustering unbounded data streams.," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 3, pp. 623–637, Mar. 2022, doi: 10.1109/tfuzz.2020.3042645.
[9] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 281–297, Aug. 1995.
[10] H. Averbuch-Elor, N. Bar, and D. Cohen-Or, "Border-peeling clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 7, pp. 1791–1797, Jul. 2020, doi: 10.1109/TPAMI.2019.2924953.
[11] J. Guan, S. Li, X. He, and J. Chen, "Clustering by fast detection of main density peaks within a peak digraph," *Inf. Sci.*, vol. 628, pp. 504–521, 2023, doi: 10.1016/j.ins.2023.01.144.
[12] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science.*, vol. 344, no. 6191, pp. 1492–1496, 2014, doi: 10.1126/science.1242072.
[13] J. Guan, S. Li, X. Chen, X. He, and J. Chen, "DEMOS: Clustering by pruning a density-boosting cluster tree of density mounts," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 10, pp. 10814–10830, Oct. 2023, doi: 10.1109/TKDE.2023.3266451.
[14] L. Sun, X. Qin, W. Ding, and J. Xu, "Nearest neighbors-based adaptive density peaks clustering with optimized allocation strategy," *Neurocomputing*, vol. 473, pp. 159–181, 2022, doi: 10.1016/j.neucom.2021.12.019.
[15] R. Liu, H. Wang, and X. Yu, "Shared-nearest-neighbor-based clustering by fast search and find of density peaks," *Inf. Sci.*, vol. 450, pp. 200–226, 2018, doi: 10.1016/j.ins.2018.03.031.
[16] J. Guan, S. Li, X. He, J. Zhu, and J. Chen, "Fast hierarchical clustering of local density peaks via an association degree transfer method," *Neurocomputing*, vol. 455, pp. 401–418, 2021, doi: 10.1016/j.neucom.2021.05.071.

[17] Y. Chen et al., "Fast density peak clustering for large scale data based on kNN," *Knowl.-Based Syst.*, vol. 187, 2020, Art. no. 104824, doi: 10.1016/j.knosys.2019.06.032.
[18] E. S. Schaeffer, "Graph clustering," *Comput. Sci. Rev.*, vol. 1, no. 1, pp. 27–64, 2007, doi: 10.1016/j.cosrev.2007.05.001.
[19] U. vonL, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, pp. 395–416, 2007, doi: 10.1007/s11222-007-9033-z.
[20] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000, doi: 10.1109/34.868688.
[21] R. Urquhart, "Graph theoretical clustering based on limited neighbourhood sets," *Pattern Recognit.*, vol. 15, no. 3, pp. 173–187, 1982, doi: 10.1016/0031-3203(82)90069-3.
[22] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, no. 2, pp. 248–264, 1972.
[23] R. Sibson, "SLINK: An optimally efficient algorithm for the single-link cluster method," *Comput. J.*, vol. 16, no. 1, pp. 30–34, 1973, doi: 10.1093/comjnl/16.1.30.
[24] M. M. Akhter and S.K. Mohanty, "A fast O (NlgN) time hybrid clustering algorithm using the circumference proximity based merging technique for diversified datasets," *Eng. Appl. Artif. Intell.*, vol. 125, 2023, Art. no. 106737, doi: 10.1016/j.engappai.2023.106737.
[25] Y. Ma, H. Lin, Y. Wang, H. Huang, and X. He, "A multi-stage hierarchical clustering algorithm based on centroid of tree and cut edge constraint," *Inf. Sci.*, vol. 557, pp. 194–219, 2021, doi: 10.1016/j.ins.2020.12.016.
[26] R. Maheshwari, S. K. Mohanty, and A. C. Mishra, "DCSNE: Density-based clustering using graph shared neighbors and entropy," *Pattern Recognit.*, vol. 137, 2023, Art. no. 109341, doi: 10.1016/j.patcog.2023.109341.
[27] G. Mishra and S. K. Mohanty, "A fast hybrid clustering technique based on local nearest neighbor using minimum spanning tree," *Expert Syst. Appl.*, vol. 132, pp. 28–43, 2019, doi: 10.1016/j.eswa.2019.04.048.
[28] A. Khan and S. K. Mohanty, "A fast spectral clustering technique using MST based proximity graph for diversified datasets," *Inf. Sci.*, vol. 609, pp. 1113–1131, 2022, doi: 10.1016/j.ins.2022.07.101.
[29] G. Yang et al., "Reskm: A general framework to accelerate large-scale spectral clustering," *Pattern Recognit.*, vol. 137, 2023, Art. no. 109275, doi: 10.1016/j.patcog.2022.109275.
[30] John Adrian Bondy, "Uppaluri siva ramachandra murty," *Graph Theory Appl.*, vol. 290, pp. 1–37, 1976, doi: 10.1137/1021086.
[31] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. 14th Int. Conf. Neural Inf. Process. Syst.: Natural Synthetic*, 2001, pp. 849–856.
[32] L. W. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 11, no. 9, pp. 1074–1085, Sep. 1992, doi: 10.1109/43.159993.
[33] J. MacQuee, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.
[34] J. Guan, S. Li, X. He, J. Zhu, J. Chen, and P. Si, "SMMP: A stable-membership-based auto-tuning multi-peak clustering algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 6307–6319, May 2023, doi: 10.1109/TPAMI.2022.3213574.
[35] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951, doi: 10.1214/aoms/1177729694.
[36] P. Fränti and S. Sieranoja, "K-means properties on six clustering benchmark datasets," *Appl. Intell.*, vol. 48, no. 12, pp. 4743–4759, 2018. [Online] available: http://cs.uef.fi/sipu/datasets/
[37] C. Michael and A. T. Ultsch, "Clustering benchmark datasets exploiting the fundamental clustering problems," *Data Brief*, vol. 30, 2020, Art. no. 105501.
[38] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Trans. Inf. Theory*, vol. 21, no. 1, pp. 32–40, Jan. 1975, doi: 10.1109/TIT.1975.1055330.
[39] J. K. Cullum and R. A. Willoughby, "Lanczos algorithms for large symmetric eigenvalue computations: Vol. I: Theory," *Soc. Ind. Appl. Math.*, pp. 32–75, 2002, doi: 10.1137/1.9780898719192.
[40] N. Bhatia, "Survey of nearest neighbor techniques," 2010, *arXiv:1007.0085*.
[41] L. Zelnik-manor and P. Perona, "Self-tuning spectral clustering," in *Proc. Neural Inf. Process. Syst.*, 2004, pp. 1601–1608.
[42] P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognit.*, vol. 39, pp. 761–775, 2006.
[43] K. Bache and M. Lichma, "UCI machine learning repository," 2013. [Online] available: http://archive.ics.uci.edu/ml
[44] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *Proc. Comput. Vis. Pattern Recognit.*, 2011, pp. 529–534.
[45] T. Hastie, J. Friedman, and R. Tibshirani, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Berlin, Germany: Springer, 2009.
[46] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," 2010. [Online] available: http://yann.lecun.com/exdb/mnist/
[47] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.
[48] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 2, no. 3, pp. 209–226, 1977.
[49] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proc. 23 rd Int. Conf. Mach. Learn.*, 2006, pp. 97–104.
[50] F. S. Samaria and A. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. 1994 IEEE Workshop Appl. Comput. Vis.*, IEEE, 1994, pp. 138–142.
[51] N. Xuan, V.S. Julien, and J. W. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, 2010.
[52] P. A. Estevez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 189–201, Feb. 2009.
[53] J. P. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, 1987.



Junyi Guan (Member, IEEE) received Ph.D. degree from the Zhejiang University of Technology (ZJUT), Hangzhou, China. He previously held a postdoctoral position at ZJUT. Since September 2024, he has been with Hangzhou Normal University as an associate professor. His current research interests include data mining, pattern recognition, unsupervised learning, and machine learning.

Sheng Li (Senior Member, IEEE) received the bachelor's degree from the Zhejiang University of Technology (ZJUT), Hangzhou, China, in 2006, the MSc degree in communications engineering, and the PhD degree in electronic engineering from the University of York, York, U.K., in 2007 and 2010, respectively. From November 2010 to October 2011, he was a postdoctoral researcher with the Ilmenau University of Technology, Ilmenau, Germany. Since April 2012, he has been with ZJUT, where he is currently an associate professor. Dr. Li received the K. M. Stott Prize for Excellence in Scientific Research, in 2010. He received the Best Paper Award from the VTC 2011 spring for the track signal processing for wireless communication. His research interests include signal processing, machine learning, and pattern recognition.

Xiongxiong He received the MS degree from Qufu Normal University, Qufu, China, in 1994, and the PhD degree from Zhejiang University, Hangzhou, China, in 1997. He held a post-doctoral position with the Harbin Institute of Technology from 1998 to 2000. He joined the Zhejiang University of Technology Hangzhou, China, in 2001, where he has been a professor with the College of Information Engineering. His research areas include nonlinear control, signal processing, and pattern recognition.

Jiajia Chen received the MA degree from East China Normal University, Shanghai, China. Her current research interests include data mining and pattern recognition.

Yangyang Zhao is currently working toward the PhD degree with the Zhejiang University of Technology. Her current research area include data mining and pattern recognition.

Yuxuan Zhang is currently working toward the master's degree with the Zhejiang University of Technology. His research interests include pattern recognition and medical image processing.