mathematics

MDPI

Article

Fast Search Using *k-d* Trees with Fine Search for Spectral Data Identification

YoungJae Son, Tiejun Chen and Sung-June Baek *

Department of Intelligent Electronics and Computer Engineering, Chonnam National University, Gwangju 61186, Republic of Korea; ya_ho@jnu.ac.kr (Y.S.); magicalbird@jnu.ac.kr (T.C.)
* Correspondence: tozero@jnu.ac.kr

Abstract: Spectral identification is an essential technology in various spectroscopic applications, often requiring large spectral databases. However, the reliance on large databases significantly increases computational complexity. To address this issue, we propose a novel fast search algorithm that substantially reduces computational demands compared to existing methods. The proposed method employs principal component transformation (*PCT*) as its foundational framework, similar to existing techniques. A running average filter is applied to reduce noise in the input data, which reduces the number of principal components (*PCs*) necessary to represent the data. Subsequently, a *k-d* tree is employed to identify a relatively similar spectrum, which efficiently constrains the search space. Additionally, fine search strategies leveraging precomputed distances enhance the existing pilot search method by dynamically updating candidate spectra, thereby improving search efficiency. Experimental results demonstrate that the proposed method achieves accuracy comparable to exhaustive search methods while significantly reducing computational complexity relative to existing approaches.

Keywords: fast search; spectroscopy identification; *k-d* tree search; fine search

MSC: 68P10

1. Introduction

The utilization of spectroscopic techniques, such as Raman spectroscopy and infrared spectroscopy, has steadily increased in recent years. Raman spectroscopy reveals the molecular fingerprint of a sample and provides quantitative information about its chemical composition [1]. Spectroscopy also reveals sensitive biochemical alterations in cells and tissues. These properties can be used in disease diagnosis and treatment, and they have applications in a variety of fields, including biology and medicine [2]. Consequently, significant research efforts have been devoted to advancing spectral signal processing techniques. Among these efforts, one of the most critical and challenging tasks is the fast and precise identification of measured spectra, which requires careful comparison and analysis against well-established and comprehensive spectral databases [3]. This task is essential not only for improving the efficiency and precision of spectral measurements but also for enabling reliable interpretation and utilization of spectral data in various scientific and industrial applications [4].

Various spectrum identification methods have been proposed, with deep learning-based approaches receiving considerable attention in recent years [5,6]. To effectively leverage these techniques, the availability of a large spectral database is essential [7]. However, existing spectral libraries typically provide only a single sample for each type of

check for updates

Received: 13 January 2025
Revised: 5 February 2025
Accepted: 8 February 2025
Published: 9 February 2025

Citation: Son, Y.; Chen, T.; Baek, S.-J. Fast Search Using *k-d* Trees with Fine Search for Spectral Data Identification. *Mathematics* **2025**, *13*, 574. https://doi.org/10.3390/math13040574

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

spectrum, which limits the ability to design robust classification models. As a result, search techniques, such as correlation search and Euclidean distance (*ED*) search, are commonly employed. These methods are intuitive, computationally efficient, and compatible with existing spectral libraries [8].

Fast search techniques are particularly suitable for embedded systems with limited computational resources [9,10]. For instance, portable spectrometer systems designed to detect hazardous substances, such as explosives and poisons, at accident sites exemplify the utility of such techniques [11,12]. Such applications demand the development of identification methods that are both fast and accurate. To meet this requirement, we focus on fast search methods that utilize similarity evaluation techniques. Unlike deep learning and machine learning approaches, which often incur significant computational overhead during the training process due to the size and complexity of datasets, similarity evaluation methods facilitate efficient searches. By relying on a single representative spectrum for comparison, these methods significantly reduce computational demands while maintaining accuracy [13].

The *ED* between spectra is a commonly used metric for similarity measurement [14]. The spectrum most similar to the input spectrum is identified as the corresponding material. A fast *ED* comparison method typically utilizes a clustering structure to optimize the search process. The primary advantage of such a structure lies in its ability to reduce the search space by grouping spectra in the database based on their similarity, thereby minimizing computational effort.

In previous studies, hierarchical clustering was employed to assess the similarity between spectra. This approach grouped closely related spectra into clusters, enabling an initial identification of the input spectrum. Subsequently, a pilot search algorithm was applied within the reduced subset to determine the spectrum most similar to the input, achieving both efficient and accurate identification [15].

To surpass the performance of existing methods, this study combines *k-d* trees with a fine search process. *k-d* trees are data structures that generalize binary search trees to k dimensions, offering low computational costs. While *k-d* trees are typically used for exact solutions [16], we adapt them to identify approximate neighbors [17,18]. A fine search is then applied as a postprocessing step to ensure that the results are consistent with those obtained through an exhaustive search.

This paper is organized as follows. Section 2 presents a review of previously developed search methods. Section 3 introduces the proposed fast algorithm, while Section 4 compares its simulation results with those of existing techniques. Finally, Section 5 provides a summary of the experimental results and conclusions.

2. Previous Methods

2.1. Full Search with Partial Distortion Search

The full search method involves identifying the vector that minimizes the squared Euclidean distance between the input vector $\mathbf{x} = (x_1, x_2, ..., x_K)$ and the database vector $\mathbf{Y} = \{\mathbf{y}_i | i = 1, 2, \cdots, N\}$, as shown in Equation (1). Here, $\mathbf{y}_i = (y_{i1}, y_{i2}, ..., y_{iK})$ represents the reference spectrum, and $d^2(\mathbf{x}, \mathbf{y}_i)$ denotes the squared Euclidean distance between the input vector $\mathbf{x}$ and the spectrum $\mathbf{y}_i$, as described in Equation (2).

$$d^2_{min} = min\ \{d^2(\mathbf{x}, \mathbf{y}_i),\ \ i = 1, 2, \cdots, N\}. \tag{1}$$

$$d^2(\mathbf{x}, \mathbf{y}_i) = \sum_{k=1}^{K} (x_k - y_{ik})^2. \tag{2}$$

The search method using the squared Euclidean distance sequentially searches the entire database to find the reference vector closest to the input vector. This method requires $N \times K$ multiplications, $N(2K-1)$ additions, and $N-1$ comparisons for a k-dimensional input vector where the number of spectra is N. This method has a limitation in that the computational load increases as the number of spectra N grows. A simple initial solution to this problem is a partial distortion search (PDS).

PDS is an algorithm that terminates early by setting a termination condition during the calculation of $d^2(\mathbf{x}, \mathbf{y}_i)$. Assuming that the current minimum distance is d^2_{min} along the minimum search path, the algorithm stops calculating the distance when there exists a value $q < K$ that satisfies the condition in Equation (3). This method reduces $(K-q)$ multiplications and $2(K-q)$ additions [19].

$$\sum_{k=1}^{q} (x_k - y_{ik})^2 \geq d^2_{min}. \tag{3}$$

2.2. Fast Search Based on Principal Component Analysis

Assuming that the dimension of the spectrum is 3300, the full search method calculates the Euclidean distance between all data points. Performing these calculations on the entire dataset demands substantial computational resources, rendering it impractical for fast searches. To address this challenge, Principal Component Analysis (PCA) is frequently employed to reduce computational costs. PCA is a multivariate analysis technique that reduces data dimensionality while preserving the essential information of the original dataset [20,21].

Let the reference spectra $\mathbf{Y}$ be an $N \times M$ matrix. A correlation matrix $\mathbf{R}$ of dimension $N \times N$ is constructed. The matrix $\mathbf{R}$ can be decomposed as shown in Equation (4), where the diagonal eigenvalue matrix Λ contains the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$. The transformed principal components (PCs) are then computed using Equation (5).

$$\mathbf{R} = \mathbf{Y}\mathbf{Y}^T = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T. \tag{4}$$

$$\mathbf{PC} = \mathbf{V}^\mathbf{T}\mathbf{Y}. \tag{5}$$

The degree of compression achieved through PCA can be quantified as a percentage of the cumulative sum of eigenvalues, as given in Equation (6).

$$CR(K) = \sum_{i=1}^{K} \lambda_i / \sum_{i=1}^{N} \lambda_i. \tag{6}$$

The cumulative ratio reaches approximately 1 when 250 principal components are used. This indicates that employing 250 PCs provides results nearly identical to those obtained via a full search, significantly reducing computational costs. However, using 250 PCs still involves a considerable amount of computation.

Previous studies have demonstrated that a search method utilizing a smaller number of PCs can achieve comparable results. The performance of such methods, along with the proposed approach, is discussed in detail in Section 4.

2.3. Cluster Search with Pilot Search

Cluster search (CS) is an algorithm based on hierarchical clustering (HC) that utilizes a similarity measure between data clusters. In CS, all distances between clusters in a spectral dataset are precomputed [15]. Assuming that the dataset consists of M clusters, there are $M \times (M-1)/2$ pairs of distances. The CS method calculates the center of each cluster and designates the data point closest to the center as the "center data". Distances between the

center data and all other points within the same cluster are then calculated and stored, with the farthest distance defining the cluster size.

The *CS* algorithm begins by computing the distance between the input spectrum and each cluster center, sorting the clusters in ascending order of distance, and identifying the closest cluster. The distance between the input spectrum and the center of the closest cluster is denoted as $d(\mathbf{x}, \mathbf{c})$, which is initialized as the minimum distance d_{min}.

For a given cluster, if the difference between $d(\mathbf{x}, \mathbf{c})$ and $d(\mathbf{y}, \mathbf{c})$ exceeds the current d_{min}, all members of that cluster are excluded from consideration as the closest candidates. Conversely, if the condition in Equation (7) is not satisfied, the spectra within the cluster are examined sequentially. The search order progresses from the cluster center outward to the farthest member. If a specific member meets the condition, the remaining members of the cluster are excluded from further inspection. This process is repeated across all M clusters, ultimately identifying the spectrum closest to the input.

$$d(\mathbf{x}, \mathbf{c}) - d(\mathbf{y}, \mathbf{c}) > d_{min}. \tag{7}$$

Pilot search begins by identifying a relatively close spectrum and then refining the search to locate the closest spectrum from this initial point. Previous studies have demonstrated that pilot search can effectively identify a relatively close spectrum using only a small number of *PCs*. Once a sufficiently close spectrum is initially identified, the subsequent determination of the closest spectrum can be achieved with substantially reduced computational effort.

3. Proposed *k-d* Tree with Fine Search

PCT is commonly employed to reduce the data size required for identifying the initial proximity spectrum. By mitigating noise in the data, the computational cost of determining the initial proximity spectrum is further minimized. Among the various types of noise reduction filters available, this study employs a running average filter to avoid additional computational overhead. Let $\mathbf{x}$ represent the input and M denote the filter order. The running average filter can be expressed as follows:

$$\mathbf{s}_i = \mathbf{s}_{i-1} + (x_i - x_{i-M+1})/M, \ \ i = 1, 2, \cdots, N-1. \tag{8}$$

$$\mathbf{s}_0 = \sum_{i=-M/2}^{M/2} x_i/M, \ \ x_i = 0 \ \ if \ \ i < 0. \tag{9}$$

The multiplications and additions required to compute the average filtering are each $N + M/2$, which is significantly less computationally intensive than a typical smoothing filter.

This filter not only reduces the noise present in the data but also significantly decreases the dimensionality of the data. As shown in Figure 1, the raw data to be identified contain substantial noise, whereas the filtered data exhibit reduced noise, leading to lower data complexity. Consequently, a smaller number of *PCs* can be used compared to previous studies. In this study, a running average filter with a length of M = 11 is employed.

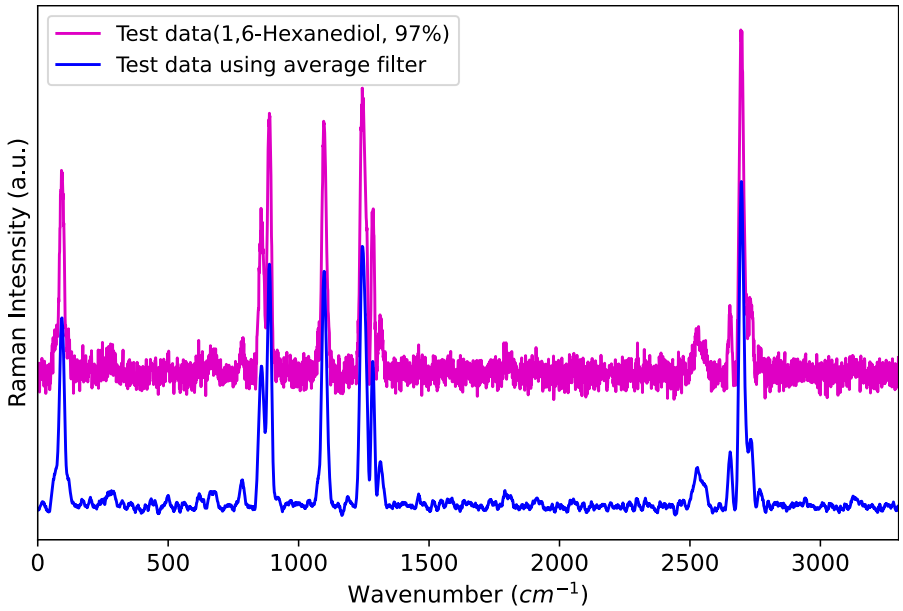


Figure 1. Raman spectrum with and without smoothing.

3.1. k-d Tree Search

k-d tree search is an extension of the binary search algorithm applied to multidimensional data. This search achieves maximum efficiency when the data in each dimension are independent of each other [22]. Therefore, *PCT* is applied to reduce the correlation between each dimension, and then a *k-d* tree is constructed from it in this study.

k-d tree search starts at the root node and computes the distance to the input data in the *k*-dimensional space, going all the way to the last child node, to identify the closest data point. The algorithm backtracks through the nodes it has traversed, updating the closest distance and checking the opposite branch when necessary to ensure that the closest data are found. This process continues until the root node is reached, at which point the closest data point is determined.

In the proposed method, the *k-d* tree is constructed using a specified number of *PCs* rather than all of them. Consequently, *k-d* tree search identifies a potentially nearest neighbor rather than the exact nearest neighbor. Therefore, a refined search step remains necessary to accurately identify the nearest spectrum.

3.2. Fine Search

Fine search is the process of finding the closest spectrum from the relatively close spectrum obtained through a *k-d* tree search. This process leverages a pre-stored distance table to reduce computational overhead. Let the input spectrum be $\mathbf{x}$, and let the current closest spectrum be $\mathbf{y}_{min}$, with $d_{min} = d(\mathbf{x}, \mathbf{y}_{min})$. Assume that $\mathbf{y}_{min,n}$ represents the spectra sorted by their proximity to $\mathbf{y}_{min}$. To determine the closest spectrum, check the following inequality for $\mathbf{y}_{min,n}$, which is the *n-th* closest spectrum to $\mathbf{y}_{min}$:

$$d(\mathbf{y}_{min}, \mathbf{y}_{min,n}) > 2d_{min}, \quad n = 0, 1, \cdots. \tag{10}$$

If this inequality is not satisfied, $d(\mathbf{x}, \mathbf{y}_{min,n})$ is computed. If the calculated distance is smaller than the current closest distance d_{min}, the value of d_{min} is updated. Otherwise, the current spectrum $\mathbf{y}_{min}$ is identified as the closest spectrum based on the following rationale:

$$\begin{aligned} ||\mathbf{y}_{min} - \mathbf{y}_{min,n}|| &> 2||\mathbf{y}_{min} - \mathbf{x}|| \\ ||\mathbf{y}_{min} - \mathbf{y}_{min,n}|| - ||\mathbf{y}_{min} - \mathbf{x}|| &> ||\mathbf{y}_{min} - \mathbf{x}|| \\ ||(\mathbf{y}_{min} - \mathbf{y}_{min,n}) - (\mathbf{y}_{min} - \mathbf{x})|| &> ||\mathbf{y}_{min} - \mathbf{x}|| \\ ||\mathbf{x} - \mathbf{y}_{min,n}|| &> ||\mathbf{y}_{min} - \mathbf{x}||. \end{aligned} \tag{11}$$

When transitioning from the second to the third inequality, the triangle inequality $||\mathbf{u} - \mathbf{v}|| >= |||\mathbf{u}|| - ||\mathbf{v}|||$ is applied. The final inequality, $d(\mathbf{x}, \mathbf{y}_{min,n}) > d_{min}$, confirms that further inspection for the closest spectrum is unnecessary.

The existing pilot search algorithm initially identifies a spectrum that is relatively close and subsequently searches for the closest match based on this initial candidate. In contrast, the proposed method improves upon this approach by dynamically updating the candidate spectrum. Specifically, it begins with a relatively close spectrum and iteratively refines the candidate whenever a closer spectrum is identified.

This method requires a table to store spectrum indices in order of proximity to a given spectrum, along with a distance table between the spectra. By continuously updating the closest spectrum when a closer match is found, the proposed approach significantly reduces the search space and improves overall performance. Figure 2 shows a simplified flow diagram of the proposed method, while Figure 3 provides a detailed flow diagram of the fine search algorithm. The corresponding pseudocode is presented in Algorithm 1.

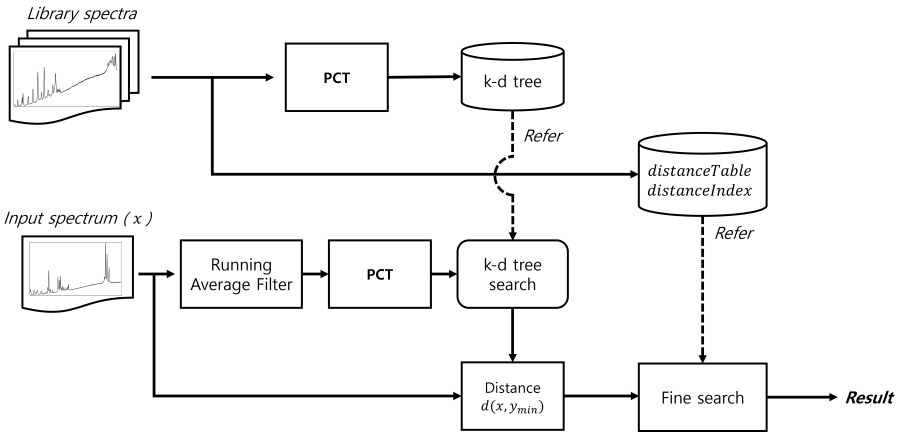


Figure 2. Flow diagram of the proposed method.

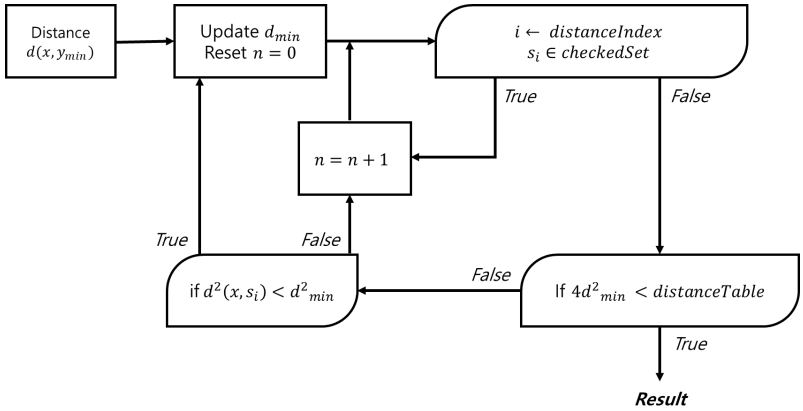


Figure 3. Flow diagram of the fine search.

Algorithm 1: Fine search algorithm

$\mathbf{x}$: input spectrum
$\mathbf{s}_i$: *i-th* spectrum in the spectral library $\mathbf{S}$ of size N
$distanceIndex[i, j]$: index of the *j-th* closest spectrum to $\mathbf{s}_i$
$distanceTable[i, j]$: squared distance between $\mathbf{s}_i$ and $\mathbf{s}_j$
$minIndex$: index of the spectrum closest to $\mathbf{x}$ using *k-d* search

$checkedSet = \{minIndex\}$ ▷ Create a set of indices of the examined spectra
$d^2_{min} = (\mathbf{x} - \mathbf{s}_{minIndex})^2$ ▷ Calculate a distance between $\mathbf{x}$ and $\mathbf{s}_{minIndex}$

$n = 0$
while $n < N$ **do**
  $i = distanceIndex[minIndex, n]$ ▷ Get the next index from $distanceIndex$
  if $i \in checkedSet$ **then**
    $n = n + 1$ ▷ Skip *i-th* spectrum if it is already examined
    continue
  end
  if $4d^2_{min} < distanceTable[minIndex, i]$ **then**
    break ▷ Stop searching if exit condition meets
  end
  $checkedSet = checkedSet \cup \{i\}$ ▷ Put i into the $checkedSet$
  $d^2 = (\mathbf{x} - \mathbf{s}_i)^2$ ▷ Calculate the current distance
  if $d^2 < d^2_{min}$ **then**
    $d^2_{min} = d^2$ ▷ Update the minimum distance
    $minIndex = i$ ▷ Update the minimum index
    $n = 0$ ▷ Reset search depth for new exploration
  else
    $n = n + 1$
  end
end
return $minIndex$ ▷ Return the index of the spectrum closest to $\mathbf{x}$

4. Experimental Results

The Raman spectrum library used in the experiment comprised 14,085 chemical substances, with each spectrum represented as a one-dimensional vector of size 3300. Background noise and additive noise in the spectrum library were removed using noise removal techniques [23,24]. Four Raman spectroscopy systems were used to measure the spectra: the Renishaw 2000, an FT-Raman system, and two inVia instruments. Figure 4 shows examples of preprocessed Raman spectra from the experiments, including 3-aminophenylboronic acid monohydrate, 2-amino-5,6-dimethylbenzimidazole, heneicosane, and triphenylantimony(V) dichloride.

For performance comparison, a total of 2817 types, corresponding to 20% of the entire database, were randomly chosen and had noise levels of 15, 20, and 24 dB added to generate test data. Since actual spectra contain non-linear noise, their presence can negatively impact performance. Therefore, it is crucial to identify the optimal parameters that allow for accurate analysis, even when harsh noise is added to the test data.

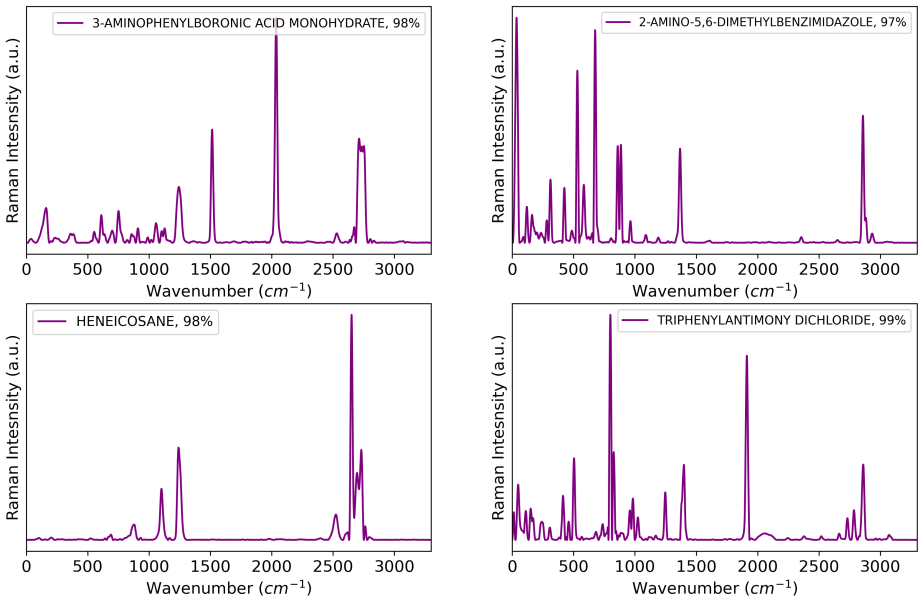


Figure 4. Examples of Raman spectra.

The performance of the algorithm was objectively evaluated using two criteria: the number of multiplication operations and the number of addition operations. The computational cost of each operation was calculated as the average number of operations across 2817 test samples.

The performance of the proposed method is significantly influenced by the number of *PCs* used. Using too few *PCs* reduces the computational load during the *k-d* tree search process but increases the burden on the fine search process, potentially resulting in a higher total computational cost. Conversely, using too many *PCs* minimizes the need for a fine search, as a *k-d* tree search alone becomes sufficient. Therefore, determining the optimal number of *PCs* is crucial for balancing efficiency and accuracy.

Table 1 illustrates this trade-off, highlighting the balance between the two processes. Specifically, when more than 40 PCs are used, the fine search step becomes redundant. In our method, 16 PCs were selected as the optimal number, minimizing the total computational cost.

Table 1. Computational complexity of the proposed method according to the number of *PCs*.

Number of *PCs*	*k-d* Tree Search		Fine Search		*k-d* Tree Search + Fine Search	
	Multiplication	Addition	Multiplication	Addition	Multiplication	Addition
4	13,413	13,513	973,523	1,946,728	986,936	1,960,238
8	28,772	30,489	228,099	456,122	256,872	486,604
16	67,760	60,752	11,872	23,739	79,633	104,475
20	88,421	108,499	8,586	17,168	97,008	125,648
40	190,272	245,500	7,601	15,198	197,874	260,659
80	400,167	532,750	7,601	15,198	407,768	547,868

To compare the performance of each algorithm, all test Raman spectra were evaluated, and the results are presented in Table 2. The number of *PCs* used in the algorithm was consistent with the optimal number of *PCs* determined in the previous study [15].

Table 2. Comparison of the multiplication and addition operations of each algorithm.

Method	Multiplication	Addition	Total
Full Search	46,480,500	92,946,915	139,427,415
Full Search + *PDS*	8,846,894	17,679,704	26,526,598
PCT + *PDS* (150 PCs)	822,617	1,135,948	1,958,565
CS (80 PCs)	433,801	593,138	1,026,939
PS (40 PCs) + *CS* (80 PCs)	319,385	350,336	669,721
Proposed method (16 PCs)	79,633	104,475	184,108

The full search + *PDS* method achieved a performance improvement of 80.97% compared to the existing full search method. When *PCT* was applied, a performance improvement of 92.62% was observed. Furthermore, the *CS* method, which constructs a cluster tree using 80 PCs, achieved a performance improvement of 47.57%, and the *PS* + *CS* algorithm, which introduces a pilot search to find a better initial spectrum, achieved a performance improvement of 34.78%. The overall average results of the main algorithms are shown in Figure 5.

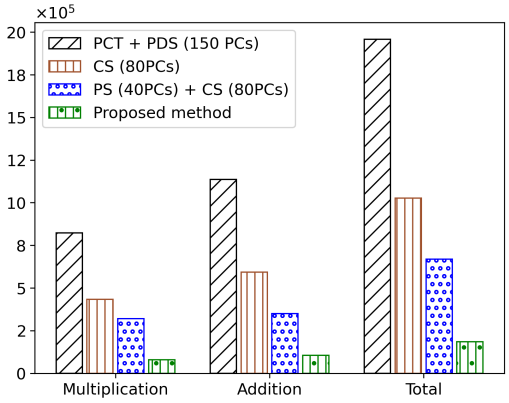


Figure 5. Comparison of multiplication and addition operations in major algorithms.

5. Conclusions

In this paper, we propose a novel search method to enhance search speed for spectrum identification. A running average filter is applied to suppress noise, improving the efficiency of *PCT*. By reducing data dimensionality through *PCT*, the computational complexity of distance calculations is significantly decreased. The *k-d* tree is constructed using these reduced-dimensional *PCs*. Since this approach does not inherently guarantee identical results to a full search, a fine search step is incorporated to ensure accuracy equivalent to that of a full search.

The optimal parameters for the proposed method were investigated and experimentally determined. The experimental results demonstrate that the proposed method significantly outperforms existing methods in terms of the number of required additions and multiplications, offering substantial improvements over traditional approaches. However, the fine search stage of the proposed algorithm necessitates additional storage to store the distances between spectra and their corresponding indices.

Consequently, the proposed method is particularly well suited for environments with limited computational power but sufficient memory resources. Deep learning techniques used for spectrum identification typically require extensive datasets, making them challenging to apply to library data containing only a single sample per type. In contrast, the proposed method is expected to be effective for retrieval problems in library datasets with only one spectrum per type.

Author Contributions: Conceptualization, Y.S. and S.-J.B.; methodology, Y.S. and S.-J.B.; validation, Y.S., T.C. and S.-J.B.; formal analysis, Y.S. and S.-J.B.; investigation, T.C.; resources, Y.S. and S.-J.B.; data curation, T.C.; writing—review and editing, Y.S. and S.-J.B.; visualization, Y.S.; supervision, S.-J.B.; project administration, S.-J.B.; funding acquisition, S.-J.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP)-Innovative Human Resource Development for Local Intellectualization program grant funded by the Korea government (MSIT) (IITP-2025-RS-2022-00156287, 50%) and ICAN (ICT Challenge and Advanced Network of HRD) support program (RS-2022-00156385, 50%).

Data Availability Statement: The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chen, H.; Xu, W.; Broderick, N.; Han, J. An adaptive denoising method for Raman spectroscopy based on lifting wavelet transform. *J. Raman Spectrosc.* **2018**, *49*, 1529–1539. [CrossRef]
2. Kong, K.; Kendall, C.; Stone, N.; Notingher, I. Raman spectroscopy for medical diagnostics—From in-vitro biofluid assays to in-vivo cancer detection. *Adv. Drug Deliv. Rev.* **2015**, *89*, 121–134. [CrossRef] [PubMed]
3. Li, J.; Chu, X.; Tian, S.; Lu, W. The identification of highly similar crude oils by infrared spectroscopy combined with pattern recognition method. *Spectrochim. Acta Part* **2013**, *112*, 457–462. [CrossRef]
4. Loethen, Y.L.; Kauffman, J.F.; Buhse, L.F.; Rodriguez, J.D. Rapid screening of anti-infective drug products for counterfeits using Raman spectral library-based correlation methods. *Analyst* **2015**, *140*, 7225–7233. [CrossRef] [PubMed]
5. Chen, T.; Son, Y.; Park, A.; Baek, S.-J. Baseline correction using a deep-learning model combining ResNet and UNet. *Analyst* **2022**, *147*, 4285–4292. [CrossRef]
6. Lussier, F.; Thibault, V.; Charron, B.; Wallace, G.Q.; Masson, J.F. Deep learning and artificial intelligence methods for Raman and surface-enhanced Raman scattering. *Trends Anal. Chem.* **2020**, *124*, 115796. [CrossRef]
7. Liu, Y. Adversarial nets for baseline correction in spectra processing. *Chemom. Intell. Lab. Syst.* **2021**, *213*, 104317. [CrossRef]
8. Park, J.K.; Park, A.; Yang, S.K.; Baek, S.-J.; Hwang, J.; Choo, J. Raman spectrum identification based on the correlation score using the weighted segmental hit quality index. *Analyst* **2017**, *142*, 380–388. [CrossRef] [PubMed]
9. Hajjou, M.; Qin, Y.; Bradby, S.; Bempong, D.; Lukulay, P. Assessment of the performance of a handheld Raman device for potential use as a screening tool in evaluating medicines quality. *J. Pharm. Biomed. Anal.* **2013**, *74*, 47–55. [CrossRef]
10. Sanchez, L.; Farber, C.; Lei, J.; Zhu-Salzman, K.; Kurouski, D. Noninvasive and nondestructive detection of cowpea bruchid within cowpea seeds with a hand-held Raman spectrometer. *Anal. Chem.* **2019**, *91*, 1733–1737. [CrossRef] [PubMed]
11. Moore, D.S.; Scharff, R.J. Portable Raman explosives detection. *Anal. Bioanal. Chem.* **2009**, *393*, 1571–1578. [CrossRef] [PubMed]
12. Izake, E.L. Forensic and homeland security applications of modern portable Raman spectroscopy. *Forensic Sci. Int.* **2010**, *202*, 1–8. [CrossRef]
13. Gray, R.M.; Neuhoff, D.L. Quantization. *IEEE Trans. Inf. Theory* **1998**, *44*, 2325–2383. [CrossRef]
14. Orchard, M.T. A fast nearest-neighbor search algorithm. *IEEE ICASSP* **1991**, *91*, 2297–2300. [CrossRef]
15. Park, J.K.; Lee, S.; Park, A.; Baek, S.-J. Fast Search Method Based on Vector Quantization for Raman Spectroscopy Identification. *Mathematics* **2020**, *8*, 1970. [CrossRef]
16. Moayeri, N.; Neuhoff, D.L.; Stark, W.E. Fine-coarse vector quantization. *IEEE Trans. Signal Process.* **1991**, *39*, 1503–1515. [CrossRef]
17. Murphy, O.J.; Selkow, S.M. The efficiency of using *k-d* trees for finding nearest neighbors in discrete space. *Inf. Process. Lett.* **1986**, *23*, 215–218. [CrossRef]
18. Tiwari, V.R. Developments in KD Tree and KNN Searches. *Int. J. Comput. Appl.* **2023**, *185*, 17–23. [CrossRef]
19. Bei, C.; Gary, R.M. An improvement of the minimum distortion encoding algorithms for vector quantization and pattern matching. *IEEE Trans. Commun.* **1985**, *33*, 1132–1133. [CrossRef]
20. Pearson, K. LIII. On lines and planes of closest fit to systems of points in space. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **1901**, *2*, 559–572. [CrossRef]
21. Jolliffe, I.T.; Cadima, J. Principal component analysis: A review and recent developments. *Philos. Trans. R. Soc. Math. Phys. Eng. Sci.* **2016**, *374*, 20150202. [CrossRef] [PubMed]
22. Bentley, J.L. Multidimensional binary search trees used for associative searching. *Commun. Acm* **1975**, *18*, 509–517. [CrossRef]

23. Baek, S.-J.; Park, A.; Ahn, Y.-J.; Choo, J. Baseline correction using asymmetrically reweighted penalized least squares smoothing. *Analyst* **2015**, *140*, 250–257. [CrossRef]
24. Press, W.H.; Flannery, B.P.; Teukolsky, S.A.; Vetterling, W.T. *Numerical Recipes in C*; Cambridge University Press: New York, NY, USA, 1988.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.