Privacy-Preserving Shape Matching with Leveled Homomorphic Encryption

Agha Aghayev[1] and Yadigar Imamverdiyev[2]

[1] Azerbaijan Technical University, Azerbaijan agayevagha@gmail.com
[2] Azerbaijan Technical University, Azerbaijan yadigar.imamverdiyev@aztu.edu.az

Abstract. Homomorphic Encryption (HE) allows parties to securely outsource data while enabling computation on encrypted data, protecting against malicious parties and data leakages. More recent HE schemes enable approximate arithmetic on complex vectors and approximation of non-linear functions, specifically useful for image processing algorithms. The Fourier Shape Descriptor (FSD) is a classical method for shape matching via frequency-domain representation, and we show that FSD can be computed entirely in the encrypted domain. To the best of our knowledge, this is the first work to implement secure shape descriptors and matching via HE. We also present two experiments for similar and different shapes, and measure the performance of the encrypted algorithm.

Keywords: Cryptography, Homomorphic Encryption, Fourier Descriptor, Privacy-Preserving, Encrypted Computation

1 Introduction

A variety of computer vision tasks, such as object classification, retrieval, pattern recognition and matching, etc., include clever ways of defining and finding objects in digitized images. The final goal is to distinguish and recognize different items in a collection of images. Given the vast number of objects, their possible orientations, and the presence of noise or distortions, computer vision tasks become even more challenging. The shape of the objects has been considered a foundational feature for detection and classification tasks. In the literature, numerous computer vision algorithms exploit various shape representations to capture dominant features of objects while ignoring minor details and noise. While some algorithms use the full object content, the shape boundary information can be as effective for some applications. The effectiveness of these methods lies in their simplicity and efficiency, compared to today's trending large machine learning models [19]. For shape matching, we aim to find the features to select similar shapes from a database with the same feature representation. One of the effective and fast shape descriptor methods is the Fourier Shape Descriptor (FSD), and it has been widely used for analysis of shapes [16]. It is a *contour-based* descriptor and works by converting contour locations to complex numbers, and then to the frequency domain by using the Discrete Fourier

Transform (DFT). The earlier descriptors capture the important features and are effective for global matching, while the later descriptors capture finer details and noise. Moreover, in cases of large amounts of data, one may want to use other third-party services, such as cloud services, and may want to store the features in a database. Cloud services may provide extensive computational power, large memory capacity, and a variety of information services. However, the well-known problem of cloud services is their safety and data security. Cloud services may face data privacy leaks during the process, or malicious providers can keep the identifying elements of clients or sensitive data long after the relationship with the client [1, 13]. Even big tech companies face data leakages with today's cutting-edge security precautions. In the context of cloud services, the use of shape descriptors with raw access to private and sensitive images raises concerns in areas such as medical imaging and biometric verification. Taking advantage of the power of external services while ensuring security, we see the importance of privacy-preserving techniques. Hence, we adopt homomorphic encryption to enable secure computation on encrypted data, without decrypting it in the first place.

While many traditional cryptographic schemes overcome high security issues, they do not allow any computation on encrypted data. In a mathematical context, the term homomorphism is a structure-preserving map between two algebraic structures of the same type. In cryptography, homomorphic encryption is a novel type of encryption paradigm that allows certain types of operations on encrypted data without decryption. The first Fully Homomorphic Encryption (FHE) scheme was proposed by Craig Gentry [7] in his thesis work, allowing for homomorphic addition and multiplication unlimited times. Since then, there has been a great development in private computation, bringing it to a practical level for real-world applications. Following his work, leveled homomorphic encryption was introduced [4], allowing a predefined number of operations on ciphertexts. Generally speaking, such cryptographic schemes differ in the domains they operate on—complex numbers, real numbers, integers, or binary data. The last generation of FHE schemes, also known as the CKKS scheme [5], allows for approximate computation on complex numbers $\mathbb{C}$, meaning that it can approximate smooth functions via Chebyshev interpolation. It can also be used as a leveled HE scheme, e.g., allowing up to L number of operations on a ciphertext. Furthermore, the Fourier Shape Descriptors (FSD) require computation on complex numbers and include non-linear operations, making the CKKS scheme a suitable solution. Because its mathematical operations translate naturally into the homomorphic domain over complex numbers, the FSD algorithm is an ideal candidate for privacy-preserving applications. Hence, our work focuses on employing homomorphic encryption for secure FSDs, and to the best of our knowledge, it is the first of its kind for shape comparison.

Our contributions. In this work, we demonstrate how to perform shape matching directly over encrypted shape vectors, enabling secure computation in the encrypted domain. We present two real-world experiments involving both similar and different shapes, and we verify the correctness of the underlying computations. Furthermore, we establish the security requirements of our approach, analyze its efficiency with packed images, and provide an open-source implementation based on the OpenFHE library [2]. Our experiments show that shape comparison with the optimal parameters—1024 point samples and 32 descriptors—can be executed in (approximately) 0.89s. The implementation is available at `https://github.com/aghayevagha/FSDHE`.

2 Literature Review

Shape descriptors have been widely adopted for similarity matching in the image domain. FSDs have diverse applications across multiple domains, including detection of military objects like UAV [15], traffic sign recognition [10], biomedical applications [16], topographic classification [9], zero-watermarking [17], plant leaf recognition [18], biometric authentication [3], sign language recognition [14], etc. Despite its various applications, the core algorithm is almost the same for most use cases.

On the other hand, most of the work on Homomorphic Encryption (HE) has centered around either theoretical advancements to improve scheme efficiency or the development of software and hardware specifically tailored for HE. Finally, HE has been applied to a variety of real-world use cases, such as machine learning, private database search, and secure data analysis. For a comprehensive overview of the current state-of-the-art in HE, we refer interested readers to the recent survey by Marcolla et al. [11]. As our algorithm uses the discrete Fourier transform, there are improved and efficient implementations of it [8]. Some non-linear functions like the inverse of the square root function have been investigated with the CKKS scheme in [12]. The Chebyshev interpolation has been widely used for approximating various non-linear functions, and OpenFHE supports the adoption of this method for many non-linear lambda functions that are employed in our work.

Despite many recent applications of homomorphic encryption (HE) in variety of areas, there has been little work on privacy-preserving shape descriptors. This study presents, to the best of our knowledge, the *first* practical approach to privacy-preserving shape matching using FSDs.

3 Notations

We denote the set of complex numbers by $\mathbb{C}$, and represent a complex number as $c = a+bi$, where $a, b \in \mathbb{R}$ (real numbers) and $i^2 = -1$. The complex conjugate of c is computed by $\bar{c} = a-bi$, and its magnitude is defined as $|c| = \sqrt{c \cdot \bar{c}} = \sqrt{a^2 + b^2}$. Throughout the paper, we implicitly use *Euler's formula*, which plays a central role in the computation of the Discrete Fourier Transform (DFT):

$$e^{i\theta} = \cos\theta + i \sin\theta$$

We denote vectors by bold letters, for example, $\mathbf{v}$. The standard basis vector $\mathbf{e}_k^T$ (transpose) is defined as $(0, 0, .., 1, .., 0)$, where the k-th element equals 1.

4 Methodology

In this section, we first provide the details of the FSD algorithm: how to select a subset from boundary points, converting them to the frequency domain, how to use the Euclidean distance between descriptor vectors to verify shape similarity, etc. Then we follow by briefly defining the CKKS scheme [5] and its functions, how to build the algorithm on the encrypted domain, and finally design experiments for the correctness of the algorithm.

4.1 Fourier Shape Descriptors

As mentioned earlier, FSDs are *contour-based* shape descriptors. The algorithm begins by extracting the boundary from a binary image (i.e., an image consisting of 0s and 1s). Once the boundary is obtained, we can employ two strategies for the selection of a set of points. Let N represent the total number of points that will be selected for representing the boundary of the shape. To benefit from advanced techniques such as the Fast Fourier Transform (FFT), the number of points is typically chosen as a power of two, although this is not mandatory. The algorithm operates under the assumption that the points are sampled either at equal angular intervals ($\theta = 2\pi/N$) or uniformly with respect to arc length. In our work, we adopt the equal arc-length representation for the rest of the paper. Once the set of points $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ are extracted from the object boundary, they are mapped into the complex domain as $(z_1, z_2, \ldots, z_N) = (x_1 + y_1 i,\ x_2 + y_2 i,\ \ldots,\ x_N + y_N i)$. There are several challenges to be considered when comparing various shapes using descriptors to ensure meaningful results. These include the placement of the object in the image, differences in the scale of shapes that can increase the magnitude of the descriptor values, and rotational variations that may increase the difference of plain descriptors. Therefore, effective shape descriptors should be invariant to translation, scale, and rotation.

Translation invariance Before computing the Discrete Fourier Transform (DFT), subtracting the centroid (mean) of the points eliminates coordinate bias, effectively centering the object at the origin. Let z_{avg} denote the centroid of the shape, then

$$z_{avg} = \frac{1}{N}\sum_{k=1}^{N} z_k$$

and subtract it from all of the points: $z_i \leftarrow z_i - z_{avg}, \quad$ for $i = 1, 2, \ldots, N$, while preserving *the centroid distance.*

DFT on shape signatures Given the set of zero-mean complex points, we compute the discrete Fourier transform of $\{z_n\}_{n=1}^{N}$ by

$$Z_k = \frac{1}{N} \sum_{n=0}^{N-1} z_n \, e^{-i\frac{2\pi}{N}kn}, \quad \text{for } k = 0, 1, \ldots, N-1$$

We refer to the coefficients Z_k, for $k = 0, 1, \ldots, N-1$ as the *Fourier descriptors* (FD) of the shape. The first FD component Z_0 reflects the absolute position of the shape and is therefore discarded in shape matching.

4.2 Scale and Rotation Invariance

The Fourier descriptor can be improved to achieve invariance with respect to both shape scale and arbitrary rotation. To ensure rotation invariance, we consider only the magnitudes of the shape descriptors, discarding the first descriptor Z_0 . Scale invariance is achieved by normalizing the magnitudes of all descriptors with respect to the magnitude of the second descriptor, $|Z_1|$. The invariant descriptor will have the final form:

$$FD = \left[\frac{|Z_2|}{|Z_1|}, \frac{|Z_3|}{|Z_1|}, \ldots, \frac{|Z_{N'}|}{|Z_1|}\right] \tag{1}$$

On the latter, N' denotes the *truncation parameter*, referring to the number of retained descriptors used for shape matching. The low-frequency descriptors (early descriptors) contain overall geometry and structure of shapes, useful for general shape matching (see figure 4). However, the high-frequency descriptors capture small details, local changes and variations, and noise. Hence, they increase algorithm sensitivity and computational cost. In practice, the selection of $2 \leq N' \ll N$ balances efficiency and generalization; a small N' is enough for general shape matching. Finally, we compute the Euclidean distance between normalized shape descriptors as a similarity metric:

$$\text{Dist}(FD^A, FD^B) = \sqrt{\sum_{k=2}^{N'} \left(\frac{|Z_k^A|}{|Z_1^A|} - \frac{|Z_k^B|}{|Z_1^B|}\right)^2} \tag{2}$$

4.3 The CKKS scheme

As discussed earlier, Homomorphic Encryption is a cutting-edge cryptographic technique for secure computation on encrypted data. In this section, we will provide some basic functions of HE, especially in the context of the CKKS scheme proposed by Cheon et al. [5], which supports approximate arithmetic on complex numbers. The vector message is a complex vector $\mathbf{z} = (z_0, z_1, \ldots, z_{N-1}) \in \mathbb{C}^{N/2}$ and they are mapped into cyclotomic rings $\mathbb{Z}[X]/(X^N + 1)$ as plaintext domain and encrypted to $\mathbb{Z}_Q[X]/(X^N+1)$ domain, where Q is the modulus element. The CKKS homomorphic encryption scheme consists of the following components:

- $\texttt{KeyGen}(\lambda) \rightarrow \texttt{s}, \texttt{pk}$: generates the secret key $\texttt{s}$ and public key $\texttt{pk}$ based on security parameter λ.

- $\texttt{Enc}(\texttt{pk}, m) \to c$: encrypts the plaintext vector m under the public key $\texttt{pk}$.
- $\texttt{Dec}(\texttt{s}, c) \to m' \approx m$: decrypts the ciphertext and returns an approximate version of the original plaintext.
- $\texttt{Add}(c_1, c_2) \to c$: performs addition of two encrypted vectors; also works for subtraction.
- $\texttt{PMult}(c, m) \to c$: performs element-wise multiplication between an encrypted vector and a plaintext vector.
- $\texttt{Mult}(c_1, c_2) \to c$: performs element-wise multiplication between two encrypted vectors.
- $\texttt{RotateKeyGen}(\texttt{s}, \{k_i\}) \to \texttt{rtks}$ and $\texttt{Rot}(c, k) \to c'$: generates rotation keys and performs cyclic rotation by k steps on the encrypted vector using the corresponding rotation key $\texttt{rtk}_k$.
- $\texttt{Conj}(c) \to c'$: computes the complex conjugate of the encrypted vector via an automorphism of index $2N - 1$.

It is worth noting that rotation keys may occupy a large amount of memory and should be considered in practical applications. Another important consideration when deploying homomorphic encryption schemes is the *multiplicative depth* of the instance." Each multiplication operation increases the multiplicative depth of the ciphertext, and once the threshold is exceeded, decryption becomes impossible because the message is lost in the accumulated noise of the ciphertext.
Approximation of Non-linear functions Some operations on the FSD algorithm require non-linear functions that are not natively supported by the CKKS scheme. The difference between approximate and plain computation depends on the parameter choices used for approximating non-linear functions, and can be adjusted with respect to the sensitivity of the algorithm. For instance, the magnitude calculation of a complex number requires a square root operation, and scale normalization is achieved by division over the magnitude of the second descriptor, which works over small numbers, as we will see in experiments. In our work, we approximate such functions with *Chebyshev interpolation* method. Chebyshev interpolation is an efficient method for approximating smooth functions with bounded error. It approximates a given function $f(x)$ defined on predefined closed interval $[a, b]$ using Chebyshev polynomials $T_k(x)$

$$P_n(x) = \sum_{k=0}^{n} c_k T_k(x),$$

and the coefficients c_k are computed from the function values at Chebyshev nodes:

$$x_j = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{2j+1}{2n+2}\pi\right), \quad j = 0, \ldots, n.$$

The latter interpolation method is nicely abstracted by some libraries and can be used to approximate any smooth lambda function in OpenFHE library. Such approximations can be considered accurate enough for many practical computations with high-degree polynomials, which may also require higher multiplicative depths. In our case, we go for a trade-off between approximation error and multiplicative depth to provide a solution within the practical limits of leveled homomorphic encryption.

4.4 Homomorphic evaluation

For *translation invariance*, we first compute the sum of the encrypted vector's components across all slots, obtain the mean, and subtract it from each element. The sum of slots can be performed in $O(\log N)$ rotations and summation of ciphertexts. This method also improves space complexity by decreasing the number of rotation keys. After receiving the shape ciphertexts c^A and c^B from the client, we can apply

Algorithm 1 `Sumslot(·)` Homomorphic Summation of slots

```
1: Input: Ciphertext $c$ encrypting $[m_0, m_1, \dots, m_{N-1}]$
2: Output: Ciphertext $c_{\text{sum}}$ encrypting a vector with each slot being $\sum_{i=0}^{N-1} m_i$
3: $c_{\text{sum}} \leftarrow c$
4: for $i \leftarrow 1$ to $N-1$ by $i = i \cdot 2$ do
5:     $c_{\text{rot}} \leftarrow \texttt{Rot}(c_{\text{sum}}, i)$
6:     $c_{\text{sum}} \leftarrow \texttt{Add}(c_{\text{sum}}, c_{\text{rot}})$
7: end for
8: return $c_{\text{sum}}$
```

To compute the average, we multiply the ciphertext by the scalar $1/N$, i.e., $c_{\text{avg}} = \texttt{PMult}(c_{\text{sum}}, 1/N)$, and then subtract the resulting mean from the original ciphertext vectors, respectively:

$$c^A \leftarrow \texttt{Sub}(c^A, c^A_{\text{avg}}), \quad c^B \leftarrow \texttt{Sub}(c^B, c^B_{\text{avg}}).$$

Achieving translation invariance, we compute the FSD components by first generating the k-th Fourier weights ($k \leq N'$) as an N-dimensional complex vector $\mathbf{w}_k \in \mathbb{C}^N$, scaled by a factor of $1/N$ for normalization to avoid additional ciphertext–plaintext multiplications

$$\mathbf{w}_k = \frac{1}{N}\left[e^{-2\pi i \frac{k\cdot 0}{N}}, e^{-2\pi i \frac{k\cdot 1}{N}}, \dots, e^{-2\pi i \frac{k\cdot (N-1)}{N}}\right].$$

Finally, the k-th FSD component is computed by multiplying the centered ciphertext with the weight vector element-wise, followed by slot-wise summation (basically computing the dot product of two vectors):

$$S^A_k = \texttt{Sumslot}(\texttt{PMult}(c^A, \mathbf{w}_k)).$$

for $k = 1, ..N'$, where S^A_k is an encrypted vector whose all components are Z^A_k.
Scale and Rotation Invariance. To build an instance of the FSD vector that is invariant to all conditions, we need to build vector (1) in the encrypted

domain. This is first achieved by plugging the corresponding ciphertexts into a single descriptor ciphertext, multiplying each C_k by a mask of vector $\mathbf{e}_{k-1}^T$ (since numbering starts from 1), and summing together:

$$\sum_{k=2}^{N'} \texttt{PMult}(S_k, e_{k-1}^T) = (C_2, C_3, \ldots, C_{N'}, c_0, c_0, ..)$$

resulting in $C^A = (C_2^A, C_3^A, .., C_{N'}^A, c_0, c_0, ..)$ and $C^B = (C_2^B, C_3^B, .., C_{N'}^B, c_0, c_0, ..)$ where $C_i^X = Z_i^X$ for $i = 2, .., N'$ (c_0 denotes encryption of a number very close to zero). The latter result can also be achieved with 1 lower multiplicative depth, but it would require N rotations. To achieve rotation invariance, we compute the magnitudes of the encrypted descriptors:

$$|C| = \sqrt{\texttt{Mult}(C, \texttt{Conj}(C))} = [|C_2|, |C_3|, .., |C_{N'}|, c_0, c_0, ..]$$

as for a complex number $c = a + bi$, its magnitude is $|c| = \sqrt{c \cdot \bar{c}}$. After computing the multiplications, we apply the square root approximation of $f(x) = \sqrt{x}$ via Chebyshev interpolation with approximately 120 degree polynomial on $[0 : 5000]$ interval, which requires 8 multiplicative depth (for pairings of various degree of polynomials and required multiplicative depth, see [6]). For scale invariance, the descriptors are normalized by the magnitude of the first non-zero component (typically $|C_1|$) as a reference. Since homomorphic division is not directly supported, we can multiply them by the inverse $1/|C_1|$. It is equivalent to directly approximating of square root function, with the components by magnitude of the second FD component S_1^A and S_1^B, accordingly:

$$1/|C_1^A| = 1/\sqrt{\texttt{Mult}(S_1^A, \texttt{Conj}(S_1^A))}$$

$$1/|C_1^B| = 1/\sqrt{\texttt{Mult}(S_1^B, \texttt{Conj}(S_1^B))}$$

After computing square of the gradient by element-wise multiplication of ciphertext with its conjugate, we approximate the function $f(x) = 1/\sqrt{x}$ on the interval $[0 : 10000]$ with the same degree polynomial (requiring 8 multiplicative depth) for each shape, resulting in $1/|C_1^A|$ and $1/|C_1^B|$. Coming together, we apply normalization by the latter, finally achieving the desired descriptor vector:

$$FD = \texttt{Mult}(|C|, 1/[C_1]) = \left[\frac{|C_2|}{[C_1]}, \frac{|C_3|}{[C_1]}, .., \frac{|C_{N'}|}{[C_1]}, c_0, c_0..\right]$$

As a result, the set FD^A and FD^B form a translation-, scale-, and rotation-invariant representation of the encrypted shape. Finally, after obtaining the encrypted descriptors FD^A and FD^B, we compute the Euclidean distance (2) between them by first subtracting one from the other

$$FD_{diff} = \texttt{Sub}(FD^A, FD^B)$$

We then compute element-wise multiplication, sum the slots of the resulting ciphertext, and take square root:

$$\text{Dist}(FD^A, FD^B) = \sqrt{\texttt{Sumslot}(\texttt{Mult}(FD_{diff}, FD_{diff}))}$$

Consequently, this yields the Euclidean distance entirely within the encrypted domain, without any decryption.

Experimental results. The homomorphic equivalent of the algorithm was benchmarked using the OpenFHE library version 1.5.4 on a system with 64GB RAM and an AMD Ryzen 9 9950X processor. The algorithm requires a multiplicative depth of 23, corresponding to a ring dimension of 65,536 for 128-bit security, with the ciphertext modulus limited to 1,448 bits (automatically determined by the library). We computed FSDs for two shapes using **multi-threading** to accelerate the computation. We design two experiments: one between two similar aircraft, and another between a tree and an aircraft. We use arc-length sampling, where points are sampled at equal distances along the contour. The figure below shows an example of uniform arc-length sampling with 128 points. In contrast, real point sampling results in almost a curved line along the boundary due to the small distance between points, as the algorithm benefits from a high number of points.

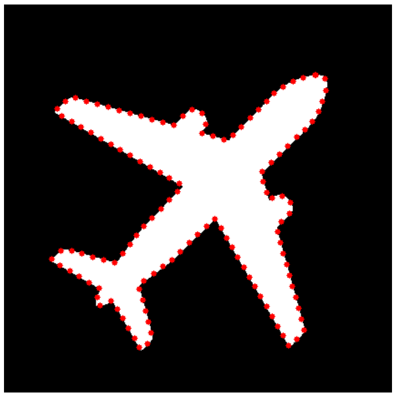

Fig. 1. Example of point sampling with 128 points

The first experiment includes two similar aircraft, one of them being rotated. We sampled 1,024 points from each and computed the Euclidean distance between 32, 64, and 128-dimensional Fourier descriptors.

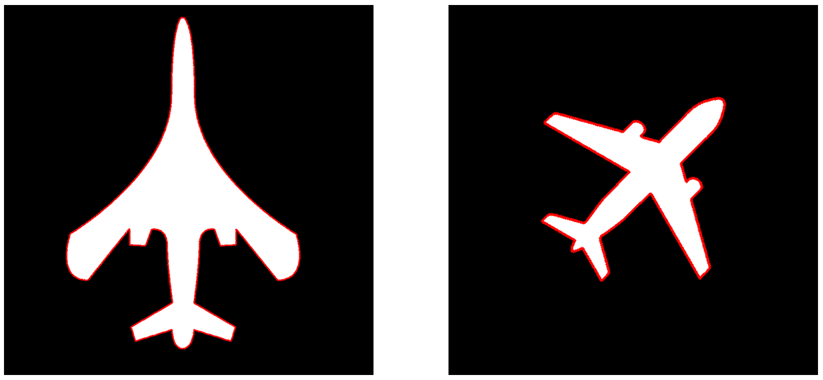

Fig. 2. First experiment: similar shapes

In practice, except for the first Fourier component, which carries the most information, other descriptors tend to have small magnitudes, especially the higher-frequency descriptors, minimally affecting the distance metric of general shape matching. However, we carefully need to consider the input boundaries for the functions we approximate (e.g. square root). When computing the magnitudes, some descriptors had squared magnitudes reaching values up to 500. Hence, considering unknown samples and potential squared values, we set the boundary to $[0, 1000]$, ensuring that all descriptor magnitudes are properly captured within this range. For similar objects like the latter example, we expect the result of the Euclidean distance to be closer to zero. Chebyshev approximations tend to blow up near zero when applied over large ranges; therefore, we use a different square root approximation with a polynomial of approximately 120 degrees over the range [0, 10] to achieve higher precision. Although the approximated functions introduce small differences (particularly near zero), the result of encrypted evaluation can be considered sufficiently accurate for practical applications. The following table shows Euclidean distances over 3 different dimensions of Fourier descriptors:

Table 1. Evaluation results for similar shapes

Domain / Dimension	$N' = 32$	$N' = 64$	$N' = 128$
Plain	0.07991	0.08139	0.08163
Encrypted	0.08606	0.08958	0.09679

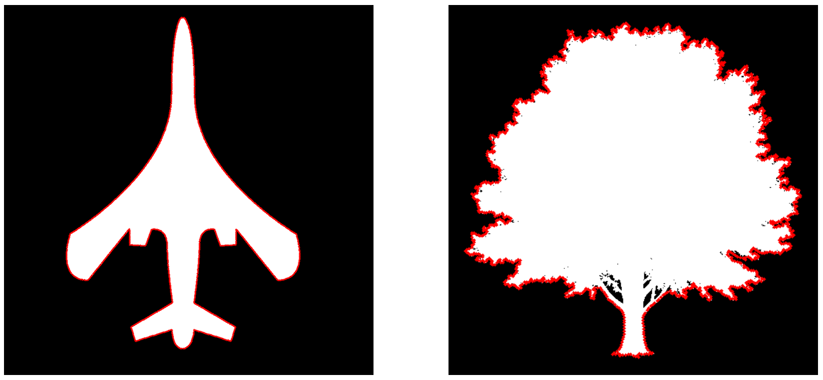

Fig. 3. Second experiment: different shapes.

The second experiment includes the same aircraft with tree, illustrating an example for non-similar objects. For different shapes, we expect the difference to be greater than at least one; we compute the difference metric on previous settings with 32, 64, 128-dimensional Fourier descriptor vectors, reflected in the table below.

Table 2. Evaluation results for different shapes

Domain / Dimension	$N' = 32$	$N' = 64$	$N' = 128$
Plain	1.68790	1.71584	1.72405
Encrypted	1.70075	1.72698	1.74338

Results from both experiments indicate that a Fourier descriptor with a truncation parameter as small as $N' = 32$ dimensions suffices for shape comparison, and higher-dimensional descriptors are generally unnecessary for effective shape matching. It is worth noting that there is no known general threshold for selection of N' or the Euclidean distance, as these may depend on the specific application. A visual reconstruction of shapes from the descriptors shows the generalized shape we use for comparison.

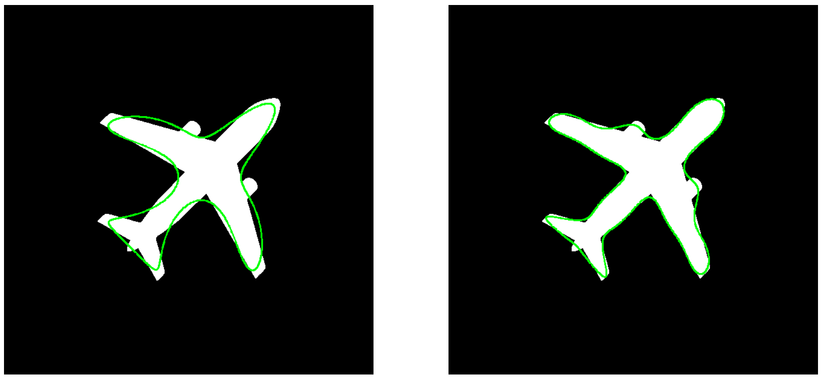

Fig. 4. Reconstruction of the shape with 16 and 32 dimensional FSDs, respectively

The construction can also be achieved via the latter algorithms, except that there is no division. The above illustration shows that even 16-dimensional descriptor can capture the shape very well, enabling us to reduce the dimension of FSDs and the execution time of the overall algorithm.

Execution performance. The following table illustrates the efficiency of the algorithm over the above experiments. The most time consuming part of the algorithm is descriptor computation, due to rotation operations.

Table 3. Execution performance over different dimensions

Dimension N'	$N' = 32$	$N' = 64$	$N' = 128$
Execution time	28.7s	47.8s	92.1s

The 32-dimensional FSD provides a good illustration of the performance–precision trade-off, requiring 28.7 seconds for execution. However, the runtime can be significantly reduced by packing 32 images into a single ciphertext (since 65,536/2 = 32,768 slots are available, and $32{,}768 = 32 \cdot 1024$). With this packing strategy, the execution time decreases by a factor of 32, yielding $\frac{28.7}{32} \approx 0.89$ seconds, for the case $N' = 32$ per shape on a high-performance machine—thus making shape comparison feasible in under one second per shape.

5 Conclusion

In this work, we examined, for the first time, a use case of Homomorphic Encryption for privacy-preserving shape matching via FSDs. We have demonstrated an effective method of employing leveled homomorphic encryption with the CKKS scheme to execute the algorithm on encrypted data, thereby protecting users' information on external services while allowing secure computation. We used the OpenFHE library [6] to provide a proof-of-concept implementation of the algorithm and conducted two experiments on similar and different shapes. Our results show that approximate homomorphic encryption can be successfully applied to a shape matching algorithm in less than a second. Future work includes exploring more advanced similarity algorithms in the encrypted domain.

6 Acknowledgements

The authors thank Jules Dumezy for reviewing an earlier draft of this paper, providing useful discussions, and contributing to the improvement of the implementation's efficiency.

References

1. Abdulsalam, Y.S., Hedabou, M.: Security and privacy in cloud computing: technical review. Future Internet **14**(1), 11 (2021)
2. Al Badawi, A., Bates, J., Bergamaschi, F., Cousins, D.B., Erabelli, S., Genise, N., Halevi, S., Hunt, H., Kim, A., Lee, Y., et al.: Openfhe: Open-source fully homomorphic encryption library. In: proceedings of the 10th workshop on encrypted computing & applied homomorphic cryptography. pp. 53–63 (2022)
3. Belgacem, N., Fournier, R., Nait-Ali, A., Bereksi-Reguig, F.: A novel biometric authentication approach using ecg and emg signals. Journal of medical engineering & technology **39**(4), 226–238 (2015)
4. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. SIAM Journal on computing **43**(2), 831–871 (2014)
5. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: International conference on the theory and application of cryptology and information security. pp. 409–437. Springer (2017)
6. Developers, O.: Function evaluation example - openfhe. `https://github.com/openfheorg/openfhe-development/blob/main/src/pke/examples/FUNCTION_EVALUATION.md` (2025), accessed: Aug. 30, 2025
7. Gentry, C.: A fully homomorphic encryption scheme. Stanford university (2009)
8. Han, K., Hhan, M., Cheon, J.H.: Improved homomorphic discrete fourier transforms and fhe bootstrapping. IEEE Access **7**, 57361–57370 (2019)
9. Keyes, L., Winstanley, A.C.: Fourier descriptors as a general classification tool for topographic shapes. In: Proceedings Irish Machine Vision and Image Processing Conference. pp. 193–203. IPRCS (1999)
10. Larsson, F., Felsberg, M.: Using fourier descriptors and spatial models for traffic sign recognition. In: Scandinavian conference on image analysis. pp. 238–249. Springer (2011)
11. Marcolla, C., Sucasas, V., Manzano, M., Bassoli, R., Fitzek, F.H., Aaraj, N.: Survey on fully homomorphic encryption, theory, and applications. Proceedings of the IEEE **110**(10), 1572–1609 (2022)
12. Qu, H., Xu, G.: Improvements of homomorphic secure evaluation of inverse square root. In: International Conference on Information and Communications Security. pp. 110–127. Springer (2023)

13. Salih, B.M., Mohammad, O.K.J.: Cloud data leakage, security, privacy issues and challenges. Procedia Computer Science **242**, 592–601 (2024)
14. Shukla, P., Garg, A., Sharma, K., Mittal, A.: A dtw and fourier descriptor based approach for indian sign language recognition. In: 2015 Third International Conference on Image Information Processing (ICIIP). pp. 113–118. IEEE (2015)
15. Unlu, E., Zenou, E., Riviere, N.: Using shape descriptors for uav detection. In: Electronic Imaging 2017. pp. pp–1 (2018)
16. Valizadeh, G., Babapour Mofrad, F.: A comprehensive survey on two and three-dimensional fourier shape descriptors: biomedical applications. Archives of Computational Methods in Engineering **29**(7), 4643–4681 (2022)
17. Wang, B., Wang, W., Zhao, P., Xiong, N.: A zero-watermark scheme based on quaternion generalized fourier descriptor for multiple images. Computers, Materials & Continua **71**(2) (2022)
18. Yang, C.: Plant leaf recognition by integrating shape and texture features. Pattern Recognition **112**, 107809 (2021)
19. Zhang, D., Lu, G.: Review of shape representation and description techniques. Pattern recognition **37**(1), 1–19 (2004)