Self-Optimal Clustering Technique Using Optimized Threshold Function

Anurag Sharma 230174
Aditya Nitin Patil 230074
Nonit Gupta 230712
Sachin Kumhar 230891

***Abstract*—This paper presents a self-optimal clustering (SOC) technique which is an advanced version of improved mountain clustering (IMC) technique. This Technique is compared with some of the widely used clustering techniques such as K-means, fuzzy C-means, Expectation and Maximization, and K-medoid. It includes a comparison of proposed technique with IMC and its updated version. All these known techniques are qualitatively and quantitatively tested through various case studies and examples using different validation indices like global silhouette index, Dunn index, partition index and separation index. The optimizing factor in the threshold function is optimized using interpolation.**

I. Introduction

Clustering is a grouping of objects which have some similarity. Objects belonging to the same cluster should be similar. Also, the objects of one cluster should differ from objects of the other cluster. This clustering can be in a hard or fuzzy way. Hard clustering includes assigning an object to a particular cluster while in fuzzy type clustering; an object is assigned a membership depending upon its belongingness with other clusters. But none of the present techniques can determine the clusters based on the data because they contain some implicit information regarding the shape and configurations of the cluster. This leads to the development of various clustering techniques.

Fuzzy C-means (FCM) gives good results, probabilistic clustering gives non-overlapping clusters but due to its large number of steps increases its computational complexity. In MMC, once the potential cluster point is determined, potential of other points is reduced. In IMC-1 and IMC-2, once the potential point is determined it is removed from the dataset maintaining potential of other points. In IMC, the threshold function is heuristically determined, which remains constant.

In SOC, we used the opportunity to optimize this threshold function through Lagrange interpolation and this technique proved effective after comparing it with other clustering techniques.

II. Overview of Some Clustering Based Techniques

Over time, many advanced clustering techniques have been proposed to make clustering more accurate and efficient. Some of these include methods based on local data variations, normalized cuts, feature space analysis, saddle point detection, multiresolution techniques, and color-texture analysis. Other important methods include Karger's graph-based approach, unsupervised segmentation, and threshold-based classifiers.

Most traditional segmentation methods (like those in references [3], [20]–[22]) directly group data points based on their positions in regular space (like 2D or 3D). These work well when the data is uniform or homogeneous, meaning the features don't vary too much within regions. However, the method proposed in this paper is different. It doesn't depend on how data behaves in the real world (like physical properties or processes). Instead, it works in hyperspace, which means each data point is represented by multiple dimensions or features (like height, weight, color, etc.). By using these dimensions in hyperspace, the method makes it easier and more flexible to process complex data, even when the real-world structure is not simple or uniform.

Popular clustering techniques include Fuzzy C-Means (FCM), MMC, Expectation-Maximization (EM), Mountain Clustering, K-means, and K-medoid. Mountain Clustering is proposed for estimating the number and location of cluster centres, and it is easy to implement but becomes computationally expensive in high dimensions. To address this, the MMC technique was introduced. The SOC technique presented in this paper is an advanced version of the IMC technique, and it improves upon previous methods by using an optimized threshold function based on Lagrange interpolation, enhancing clustering performance significantly.

III. SOC Technique

Algorithm

1) To ensure that all data points lie within a unit hypercube, the values in each dimension of the dataset are normalized. For a dataset defined in a D-dimensional hyperspace, each data instance x_j is made up of values $x_{1j}, x_{2j}, ..., x_{Dj}$. This normalization process transforms each instance by scaling its values, so they fall between 0 and 1.

$$\bar{x}_j = \frac{x_j - (x)_{\min}}{(x)_{\max} - (x)_{\min}}; \quad \text{for all values of } j = 1, 2, 3, \ldots, n \tag{1}$$

Where

$$(x)_{\min} = [\min(x_{1,1}, \ldots, x_{n,1}),\ \min(x_{1,2}, \ldots, x_{n,2}),\ \ldots,\ \min(x_{1,D}, \ldots, x_{n,D})]$$
$$(x)_{\max} = [\max(x_{1,1}, \ldots, x_{n,1}),\ \max(x_{1,2}, \ldots, x_{n,2}),\ \ldots,\ \max(x_{1,D}, \ldots, x_{n,D})]$$

n is the number of data points in the dataset and D is the total number of dimensions of hyperspace. $x_{\min}$ and $x_{\max}$ are the minimum and maximum values across all instances for each dimension, respectively. This ensures that all features are scaled proportionally and contributes to more accurate clustering results.

2) To define the neighborhood around a data point for the m^{th} cluster, a threshold value δ_m is used. This threshold is a positive value calculated using a heuristic formula, which is further adjusted by a factor β_m. This β_m acts as an optimizing parameter and is later refined using interpolation techniques explained in the algorithm. In the beginning, β_m is simply set to 1 while identifying the cluster for the first time.

$$\delta_m = \left(\frac{1}{2n} \sum_{j=1}^{n} \frac{\min(\mathbf{x}^j)}{\sum_{i=1}^{D} x_i^j} \right) \cdot (\beta_m).$$

3) To calculate the potential of a data point for the m^{th} cluster, a mountain function is used. This function depends on the distance between a chosen point r and every other point j in the dataset. The distance is calculated using the formula:

$$d^2(r, j) = (r - j)Q(r - j)^T$$

Here, Q is just a unity (identity) matrix.
Once the distances are found, the potential is calculated using:

$$P_m^r = \sum_{j=1}^{n} \exp\left(-\frac{d^2(r, j)}{\delta_m^2} \right)$$

This basically means that a point's potential is higher if many other points are close to it.

4) Once the potential values for all points have been calculated, we choose the point with the highest potential as the cluster center for the cluster. In simple terms, the point that is surrounded by the most other points becomes the cluster center.
Mathematically, this is written as:

$$c_m = x^* \quad \text{where} \quad P_m^* = \max_r P_m^r$$

5) After selecting the center for the m^{th} cluster, we check the distance of all other data points from this center. Any point whose Euclidean distance from the cluster center c_m is less than or equal to the threshold value δ_m is grouped into the m^{th} cluster.

6) Once we assign points to the m^{th} cluster, we remove those points from the dataset, so they're not used again in the next clustering steps.

7) We then repeat Steps 2 to 6 on the remaining data points. This process continues until we've formed the total number of clusters, say M, which is considered the optimal number based on earlier calculations.

8) Any leftover data points that weren't assigned during clustering are now assigned to the nearest cluster. This is done by checking which cluster center is closest using Euclidean distance.

9) Now that the clusters have been formed, we check how good they are using something called the Global Silhouette Index (GSI). This value tells us how well-separated and well-formed our clusters are. A GSI close to 1 means the clusters are very well formed. For each cluster m, we calculate a silhouette value S_m, and if most of them are near 1, then the clustering is considered high quality. Let's say for cluster t, the threshold used is δ_t and the silhouette value is S_t.

10) To improve the method further, we try to find a relationship between δ_t (the threshold) and S_t (the silhouette value). We do this by applying Lagrange interpolation over all the clusters.
We have M cluster-wise values like $(\delta_1, S_1), (\delta_2, S_2), ..., (\delta_M, S_M)$
Using these, we construct the interpolation polynomial:

$$S_t = \sum_{m=1}^{M} S_m \cdot l_m(\delta_t)$$

$$S_t = \sum_{m=1}^{M} S_m \cdot \prod_{\substack{k=1 \\ k \neq m}}^{M} \frac{\delta_t - \delta_k}{\delta_m - \delta_k}$$

Expanding this expression, we get:

$$S_t = S_1 \cdot \prod_{k=2}^{M} \frac{\delta_t - \delta_k}{\delta_1 - \delta_k} + S_2 \cdot \prod_{\substack{k=1 \\ k \neq 2}}^{M} \frac{\delta_t - \delta_k}{\delta_2 - \delta_k} + \ldots + S_M \cdot \prod_{k=1}^{M-1} \frac{\delta_t - \delta_k}{\delta_M - \delta_k}$$

This gives a polynomial formula where S_t depends on δ_t. The goal is to get S_t as close to 1 as possible, meaning the best cluster quality.

11) Substitute $S_t = 1$ in the above polynomial equation. This helps us form an equation that only has one unknown: δ_t. All the other terms like $\delta_1, \delta_2, \ldots, \delta_M$ and $S_1, S_2, \ldots, S_M$ are already known from the earlier steps.

12) We now solve the equation from Step 11 to find possible values (roots) for δ_t. Once we have all the roots, we check which one gives a silhouette value S_t that is closest to 1 when plugged back into the original equation. That particular root is selected and named η.

This value of η becomes our updated threshold function δ_t corresponding to the maximum value of S_t.

13) We calculate a new set of β_m values by dividing this selected threshold η by the old thresholds for each cluster:

$$\beta_m = \frac{\eta}{\delta_m}, \quad m = 1, 2, \ldots, M$$

These new β_m values are used to update the thresholds δ_m in the clustering steps.

14) Now, using the updated β_m values, we repeat the entire clustering process again starting from Step 2 all the way to this step. We keep repeating until the threshold values δ_m become stable or don't change much anymore and the Global Silhouette Index (GSI) is maximized.

1) Calculation of β: We have used widely accepted indices like Silhouette Index, Dunn Index, Partition Index, and Separation Index. These indices are computed based on inter- and intra-clustered distances. For better quality clustering, we want maximum inter- and minimum intra-cluster distances. Changes in δ_m vary the cluster quality, inter-cluster, and intra-cluster distances, and alter values of different validation indices. By changing δ_m, we can optimize clustering.

For this, we have the best index, Global Silhouette Index (GSI), to improve δ_m. We used the Lagrange interpolation polynomial method to find the relationship between S_m and δ_m. We generalize them into δ_t and S_m. Through interpolation, we get a $(M+1)^{th}$ degree polynomial for a total of M number of clusters formed. The maximum of S_t is 1, thus the root η of the polynomial where the value of δ_t corresponds to maximum S_t. This η divided by δ_m gives β_m, which are multiplied to the threshold function for optimizing the cluster quality. Using these β_m values, we repeat the algorithm steps until δ_m converges. The repetition is usually done ten times so that we get a value of β_m for which the GSI is maximum and find the optimum number of clusters required.

To justify the algorithm mentioned earlier, a sample image comprising two different colors is analyzed here, and the variation in δ_m and S_m with each of the ten repetitions in the algorithm performed is tabulated in Table **??**. It shows the values of δ_m and S_m at different iterations. This variation shows the inherent tendency of shifting δ_m toward η and attaining possible values closest to it with every repetition of the algorithm until δ_m converges.

In the first iteration with the value of β_m assumed to be unity, the value of η corresponding to the maximum possible value of S_m, i.e., unity, is obtained as 0.0177. On dividing η with δ_m, the corresponding β_m values are obtained. Utilizing these computed β_m values in the calculation of δ_m for the next iteration, the δ_1 and δ_2 values are shifted toward η. After many iterations, we get a point at which GSI is maximum. For this point, we also calculated other validation indices like PI, SI, and DI, which are plotted against the number of iterations.

A. Measures of Cluster Quality

We employed 4 indices for determining cluster quality.

1) GSI: This assesses overall cluster quality for point i, where $a(i)$ is the average intra-cluster distance and $b(i)$ is the average distance to any other cluster.

Silhouette width:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \quad -1 < s(i) < 1$$

$$S_m = \frac{1}{N_m} \sum_{i=1}^{N_m} s(i).$$

$$GSI = \frac{1}{M} \sum_{m=1}^{M} S_m.$$

The silhouette value S_m for the m^{th} cluster is defined accordingly, and the GSI is computed as follows.

2) PI: It measures the ratio of intra-cluster spread to inter-cluster separation. Lower PI is desired.

It is computed as:

$$PI = \sum_{m=1}^{M} \frac{\sum_{j=1}^{n} (\mu_{jm})^2 \left\| \bar{x}^j - \bar{c}_m \right\|^2}{N_m \sum_{k=1}^{M} \left\| \bar{c}_k - \bar{c}_m \right\|^2}$$

where:

- k is the number of clusters,
- n_i is the number of points in cluster C_i,
- c_i is the centroid of cluster C_i,
- $\|x - c_i\|$ is the Euclidean distance of a point x from the centroid.

where x_i are data points and c_i are cluster centroids.

3) SI: It uses a minimum distance separation for partition validity.

4) DI: It identifies clusters that are compact and well-separated. Higher DI values indicate better clustering.

It is computed as:

$$\text{DI} = \frac{\min\limits_{1 \le i < j \le k} d(C_i, C_j)}{\max\limits_{1 \le l \le k} \text{diam}(C_l)}$$

where:

- $d(C_i, C_j)$ is the distance between clusters C_i and C_j,
- $\text{diam}(C_l)$ is the diameter of cluster C_l defined as $\max_{x,y \in C_l} \|x - y\|$.

B. Need for Optimum Threshold Function

IMC-1 uses a fixed threshold δ_m for each cluster. IMC-2 improves IMC-1 by multiplying δ_m with a heuristic constant. However, β_m is not data-driven—it is just a guess.

In SOC, we use Lagrange interpolation to optimize β_m using the data given at every iteration. We multiply this β_m with δ_m to improve cluster quality.

C. Determining Optimum Number of Clusters

We start with $M = 2$, run the SOC algorithm for this M, and compute GSI. We repeat this for about 10 iterations and choose the M for which GSI is the maximum.

D. Simulation, Mathematical Proof, and Modeling of the SOC Algorithm and Its Results

This part includes mathematical proofs on:

a) Does SOC reliably converge to a good solution?
b) Can we mathematically justify that the optimized threshold δ_m improves over iterations?

Mathematically, a series converges when its sum of absolute values is finite. A similar case is for integration. A sequence $\{a_n\}$ converges if $|a_n| \to L$. In SOC, it is stressed that a series converges even if the sum of its absolute value is not finite. These series are called conditionally convergent.

In SOC, through Lagrange interpolation, β_m is optimized at every point and as the threshold value gets updated, it converges to an optimal value η, which is determined to maximize the global silhouette index (GSI).

Further simulations demonstrate that the GSI steadily increases and peaks at a specific iteration (e.g., the 9th), indicating convergence. The convergence is validated using interpolation node plots and MATLAB simulations, showing alignment between simulated η values and those computed through the SOC algorithm. Despite minor deviations due to cluster composition changes across iterations, the method reliably converges to an optimal threshold.

IV. Results and Discussion

A. Results of Comparison

Synthetic and natural images are used as case studies and examples here.

B. Evaluation of Clustering Performance

To evaluate the effectiveness of the proposed Self-Optimal Clustering (SOC) method, a comprehensive comparison is conducted with several widely used clustering algorithms, including IMC-1, IMC-2, K-Means, K-Medoids, Fuzzy C-Means (FCM), and Expectation-Maximization (EM). These algorithms are applied across a variety of synthetic and natural images selected as case studies.

The clustering performance is quantitatively assessed using standard cluster validity indices—namely, the Global Silhouette Index (GSI), Partition Index (PI), and Separation Index (SI). In all test scenarios, the SOC algorithm consistently demonstrates superior performance, achieving higher GSI values along with significantly optimized PI and SI scores compared to the other methods.

All simulations are executed on a MacBook equipped with 16 GB of RAM and an Apple M3 processor. Notably, no preprocessing is applied to the input images prior to clustering, ensuring a fair and unbiased evaluation of algorithm performance.

To validate the performance of the proposed SOC method, three representative examples are considered: a synthetic four-colour image, a back-lit bottle image, and a natural landscape. These images vary in structure and complexity. Fig. 2 shows the original and SOC-segmented outputs.

Example 1: Here, we consider a natural scene image (shown in Fig. 2(c)), featuring diverse textures and color gradients. The image is clustered using K-means, FCM, EM, K-medoids, IMC-1, IMC-2, and SOC. As observed from the results in Table 1, the optimal number of clusters determined for this image is two. The performance of each clustering technique is evaluated using standard cluster validity indices: GSI, PI, SI, and DI. These metrics provide a comprehensive view of the compactness, separation, and distribution of clusters formed. Example 2: In this example, we consider an image containing a bottle with a back-lit background (as shown in Fig. 2(b)). Based on the evaluation and visualization, the optimum number of clusters is identified as four for this image. The performance comparison is presented in Table 2, highlighting cluster quality metrics such as GSI, PI, SI, and DI. Example 3: In this case, a synthetic image consisting of four distinct color regions is considered for clustering (Fig. 2(a)). From the GSI trend, it is observed that the optimal number of clusters for this image is four.

The results, summarized in Table 3, present the cluster quality measures including GSI, PI, SI, and DI for all the clustering methods. Although IMC-1 and IMC-2 show nearly identical performance in this specific case, SOC continues to demonstrate robust and consistent clustering. It yields comparable GSI and DI values while maintaining strong compactness (PI) and good separation (SI) similar to the best-performing methods.

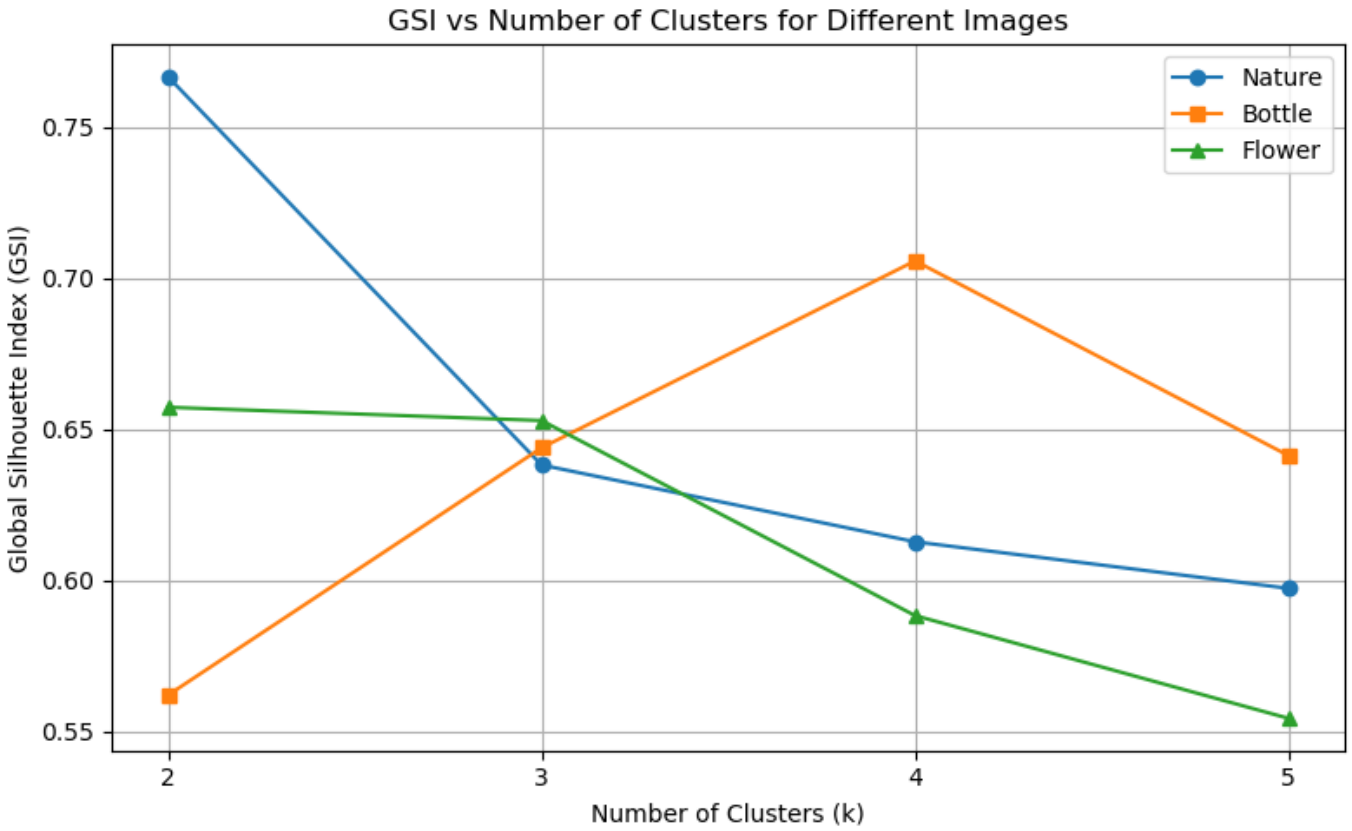


Fig. 1: GSI vs Number of Clusters for Different Images

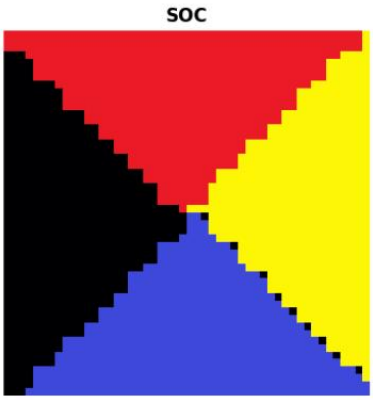
TABLE I: List of Clustering Algorithms (Existing/Conceptualized/Proposed) Having Initial Threshold Function With/Without Modification

Actual /Assumed name	**Description of changes made**	**Modified threshold function**
IMC-1	IMC technique with no changes in the threshold function	$\delta_m = \left(\frac{1}{2n}\sum_{j=1}^{n}\min(x^j)\right) / \left(\frac{1}{n}\sum_{j=1}^{n}x^j\right)$
IMC-max	IMC technique with 'min' replaced by 'max' in the threshold function	$\delta_m = \left(\frac{1}{2n}\sum_{j=1}^{n}\max(x^j)\right) / \left(\frac{1}{n}\sum_{j=1}^{n}x^j\right)$
IMC-half	IMC technique with the factor of '1/2' removed from the threshold function	$\delta_m = \left(\frac{1}{n}\sum_{j=1}^{n}\min(x^j)\right) / \left(\frac{1}{n}\sum_{j=1}^{n}x^j\right)$
IMC-2	IMC technique with a factor 'no. of cluster / (no. of cluster + 1)' multiplied to the threshold function	$\delta_m = \left(\frac{1}{2n}\sum_{j=1}^{n}\min(x^j)\right) / \left(\frac{1}{n}\sum_{j=1}^{n}x^j\right)\left(\frac{nk}{nk+1}\right)$
SOC	Proposed algorithm using IMC technique with no inherent changes in the INITIAL threshold function and a factor is multiplied in later threshold functions EXTERNALLY only	$\delta_m = \left(\frac{1}{2n}\sum_{j=1}^{n}\min(x^j)\right) / \left(\frac{1}{n}\sum_{j=1}^{n}x^j\right)(\beta_m)$
where nk = total number of clusters where β_m = optimizing factor multiplied externally		

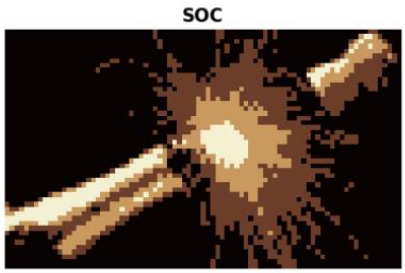
(a) Original: Four-Colour

(b) Original: Bottle

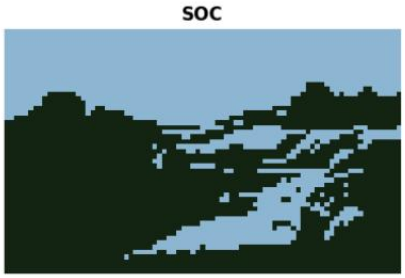
(c) Original: Nature



(d) SOC Result



(e) SOC Result



(f) SOC Result

Fig. 2: Visual comparison of original images (top row) with their corresponding SOC-segmented outputs (bottom row).

TABLE II: Results for Example 1 (Nature)

Method	GSI	PI	SI	DI
IMC-1	0.6083	0.0309	0.7393	0.7474
IMC-2	0.6065	0.0324	0.7136	0.7247
IMC-max	0.6115	0.0266	0.7863	0.7978
IMC-half	0.6101	0.0271	0.8119	0.7729
SOC	0.6072	0.0318	0.7244	0.7270
K-means	0.6104	0.0310	0.9716	0.7808
FCM	0.6101	0.0292	0.9283	0.7804
EM	0.5568	0.0333	1.0099	0.5595
K-medoid	0.6093	0.0283	0.8882	0.7645

TABLE III: Results for Example 2 (Bottle)

Method	GSI	PI	SI	DI
IMC-1	0.5938	0.0013	17.0855	0.5639
IMC-2	0.5896	0.0014	18.8696	0.4932
IMC-max	0.6485	0.0039	3.6483	0.8537
IMC-half	0.6332	0.0039	4.6574	0.6754
SOC	0.6043	0.0012	10.1565	0.7604
K-means	0.5926	0.0013	16.1668	0.4923
FCM	0.5913	0.0012	16.7719	0.5015
EM	0.2442	0.0018	111.3922	0.0921
K-medoid	0.5709	0.0015	19.9569	0.3486

TABLE IV: Results for Example 3 (Four Colors)

Method	GSI	PI	SI	DI
IMC-1	0.8988	0.0006	4.3591	0.8324
IMC-2	0.8988	0.0006	4.3591	0.8324
IMC-max	0.8977	0.0006	4.4522	0.8587
IMC-half	0.8988	0.0006	4.3674	0.8324
SOC	0.8988	0.0006	4.4123	0.8324
K-means	0.8988	0.0007	4.5685	0.8324
FCM	0.8988	0.0007	4.5636	0.8324
EM	0.8972	0.0007	4.5639	0.8302
K-medoid	0.8988	0.0006	4.4720	0.8324

CONCLUSION

This paper presents the Self-Optimal Clustering (SOC) technique as an optimized extension of the IMC method. While SOC does not achieve the best results in all cases, it consistently delivers competitive and reliable segmentation performance, especially in real-world image scenarios.

In some benchmarks, IMC-max outperforms SOC in terms of cluster quality indices, and IMC-2 also shows results very close to SOC. However, both rely on heuristic modifications, while SOC is based on a mathematically optimized threshold function.

Despite not always being the best, SOC demonstrates good clustering accuracy and can be effectively applied in practical image segmentation tasks, proving its real-world usability.

REFERENCES

[1] N. K. Verma and A. Roy, "Self Optimal Clustering Technique Using Optimized Threshold Function," *IEEE Systems Journal*, vol. 99, pp. 1–14, Jul. 2013.