ISSN 2278-3091
Volume 9, No.4, July – August 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse23942020.pdf
https://doi.org/10.30534/ijatcse/2020/23942020

R-Tree based Density Clustering for Multispectral Images

Prasad Kaviti[1] ,Valli Kumari Vatsavayi[2]
[1]Dept. of Computer Science and Systems Engg.AUCE(A), Andhra University
prasadkaviti@gmail.com
[2]Dept. of Computer Science and Systems Engg.AUCE(A), Andhra University
vallikumari@gmail.com

ABSTRACT

Density based clustering is one of the most popular clustering techniques that groups data in dense regions. These dense regions are identified with the help of a few parameters which are sensitive to estimate. Clustering accuracy depends on effective selection of these parameters. The proposed method uses density-based clustering and R-Tree. R-Tree provides faster evaluation of nearest neighbours and also is capable to store and handle multi-dimensional data efficiently. The proposed algorithm has two steps: i)R-tree is constructed from multispectral image data. Data is partioned into dense clusters/regions using nearest neighbours on the R-tree data, ii) A threshold parameter `t' which is similar to `epsilon' in DBSCAN algorithm is used to regulate the dense regions formed in the initial step by merging neighbour dense regions which are within `t'-radius. This procedure ignored the sensitive parameters like `minpts' in DBSCAN by evaluating the nearest neighbours using R-Tree. Hence the number of parameters also reduced. Results show that generation of nearest neighbours is faster and better in the proposed method when compared to traditional density-based clustering methods DBSCAN, OPTICS and K-Means.

Key words : Density Clustering, multispectral images, R-Tree based clustering, DBSCAN Algorithm.

1. INTRODUCTION

Clustering is the process that includes identification and grouping of similar or dissimilar objects [22]. Clustering analysis is widely used by varied sectors such as business analytics, data manipulation, and image analysis. Clustering methods can be classified in different methods like partitional, hierarchical, density-based [13] and grid-based methods [9]. One of such popular clustering techniques is Density-based clustering [15].

The fundamental idea of Density-based clustering is to further grow the given cluster until the density in the neighbourhood falls short of the threshold, i.e., for each cluster within a given data points, the radius of the cluster has to contain at least a minimum number of points. There are a number of density-based clustering methods [8] [18] such as Mean Shift [20], DBSCAN [13], OPTICS [14], DENCLUE, VDBSCAN, DVBSCAN,DBCLASD and ST-DBSCAN [21].

DBSCAN discovers high-density regions in spatial databases with noise and creates clusters out of them [10]. The main advantage with DBSCAN is it detects clusters of arbitrary shape and noise points [19]. But determining the initial parameters eps, minpts is difficult and if there is variation in the density, noise points are not detected. Ordering points to identify the clustering structure (OPTICS) is an algorithm for clustering data whose characteristics are strikingly similar to that of DBSCAN. OPTICS doesn't consider the parameter epsilon. Instead of choosing `epsilon' value manually, OPTICS considers the value as greater than the maximum distance between pair of data points in the dataset. But it leads to quadratic complexity since every neighbourhood query returns the full dataset. The quality in contrast between OPTICS and DBSCAN is that the former can handle data of fluctuating densities. Clustering returned by OPTICS is very similar to that from that which is created by DBSCAN. In this paper, instead of minpts evaluation in DBSCAN, nearest neighbours can be evaluated using R-Tree and then density-based clustering is performed. The results are compared with standard algorithms like DBSCAN with kd-Tree, K-Means, OPTICS. Results shown higher efficiency towards the proposed method

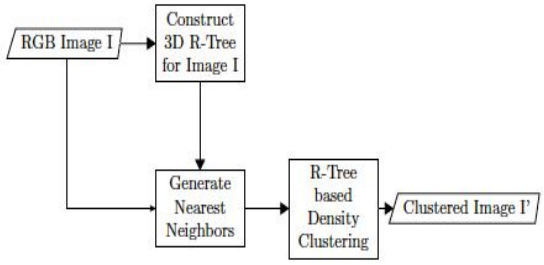

Figure 1: Data flow of Density-based Clustering with R-Tree

2. NEAREST NEIGHBOUR SEARCH ALGORITHMS

Different techniques are used for nearest neighbour search [3]. To reduce complexities, a variety of techniques are proposed [4] which are suitable for different applications such as multimedia data manipulation, information extraction, databases, data mining and pattern recognition to name but a

few. By paying attention to different applications and data, each of these techniques has to use a structure for maintaining, indexing points and searching [5]. Some of these structures are techniques for nearest neighbour search [17] such as B-Tree, X-Tree, Ball-Tree, kd-Tree, R-Tree, etc [16]. A brief overview of some of these data structures is presented as follows.

kd-Tree : A kd-Tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space [6]. At every level of a kd-Tree there is splitting in all children nodes concerning a specific dimension, using a plane which is perpendicular to the corresponding axis, also known as the hyperplane. Beginning with the root node, the algorithm moves through the tree iteratively, in the same way in the case of the search point were being inserted (i.e. depending on whether the point is lesser than or greater than the current node in the split dimension, it goes left or right).

kd-Trees are not suitable, however, for efficiently finding the nearest neighbour in high dimensional spaces [7]. As a general rule, if the dimensionality is k, the number of points in the data, N, should be N≫2k. Else, when using kd-Trees for high-dimensional data, too many points in the tree will be examined which leads to complex evaluation, hence, is not much efficient than an exhaustive search, and other methods such as approximate nearest neighbour are used instead.

Ball-Tree :A ball tree is also a binary tree with a hierarchical (binary) structure[1]. To begin with, two clusters (each resembling a ball) are created and, as it is a multi-dimensional space, each ball may be roughly called a hypersphere. Any point in n-dimensional space must belong to any one of the clusters alone but not to both [2]. It will belong to the cluster whose centroid is closest to it. Any ball can be randomly picked if the distance of a point from the centroids of both the balls is same. Even if both (virtual) hyperspheres intersect, the points must belong to only one of the two balls. Next, each of the balls is again divided into two sub-clusters so that each one is now considered a ball; meaning that in these sub-clusters, two more centroids exist and similarly, the point belongs to that ball whose distance from the sub-centroid is closest. Again, the clusters are sub-divided, and each has new sub-sub balls and so on iteratively until certain depth. The main disadvantage is slower than kd-Trees in low dimensions.

R-Tree : R-Trees are hierarchical data structures based on B+-Trees [11]. R-Trees are used for dynamic organization of a set of multi-dimensional objects indexing them by the minimum bounding d-dimensional rectangles (for simplicity, MBRs in the sequel).Each node of the R-Tree corresponds to the MBR that bounds its children [12]. Instead of pointing to children nodes, the leaves of the tree contain pointers to the database objects directly. When data is organized in an R-Tree, the neighbours within a given distance rand the k nearest neighbours of all points can efficiently be computed using a spatial join. This is beneficial for many algorithms based on such queries.

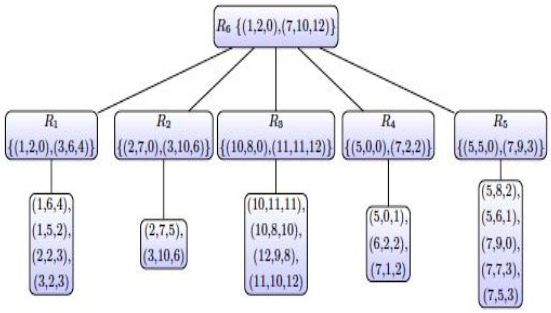

Figure 2: R-Tree with random points

When performing a range search on R-Tree, we can start from the top tree level and drill down, ignoring all the boxes that don't intersect our query box. For a small query box, this means discarding all but a few boxes at each level of the tree. A range search in an R-Tree takes O(K log(N)) time on average (where K is the number of results), compared to O(N) of a linear search. In other words, it's extremely fast.

kd-Tree is similar to R-Tree, but instead of sorting the points into several boxes at each tree level, we sort them into two halves (around a median point) either top and bottom or left and right oscillating between x and y split on each level. Compared to R-Tree, kd-Tree can usually contain points alone (not rectangles) and cannot handle insertion or deletion of points, but it's easier to construct and quite quick.

kd-Tree and Ball-Tree are the best with Euclidean distance, but they can be only used for Euclidean distance-based nearest neighbour search due to its inherent nature. Further, on high-dimensional datasets, kd-tree and Ball-Tree tend to perform poorly. Their performance can be inefficient compared to a brute force approach in such scenarios.

3.R-TREE CONSTRUCTION

Geographical information systems and spatial databases are widely used in the current world. Along with this, spatio-temporal database, processing of points and trajectories of moving objects are widely studied. New kinds of data such as audio, video, images, maps are a part of the final product, be it images or multimedia _les, are being developed. All the above applications must rely on the R-Tree data structure for storing and retrieval. The application of this data structure is extensive right from spatial-temporal to multimedia databases.

R-Tree can quickly evaluate the nearest neighbours, which has extended its usage in major fields such as GIS (urban structures, water bodies, vegetation etc.), multimedia, spatial data and other higher-dimensional data structures along with time series and many others.

The R-Tree [11] in the proposed method takes a 3-band RGB Image as input. Each pixel in the image is a data point to the R-Tree. To insert a pixel into the tree, an MBR(Minimum Bounding Rectangle) has to be defined for each pixel point. Since a pixel point has 3 values, (R, G and B) the MBR will be a cube with each value of R, G, Bon their corresponding axes (R = x, G = y, B = z). The MBR for a point is defined as $(lower_R, lower_G, lower_B)$, $(upper_R, upper_G, upper_B)$. In this proposed method, the MBRfor a point is initialized as $lower_R = upper_R$, $lower_G = upper_G$ and $lower_B = upper_B$ so as to maintain symmetry in the dimensions across all axes. In this way, the length of the MBR cuboid formed will be the same on its corresponding axis.

To construct the R-Tree for the input data, after defining the MBR for each point, start from an empty root node `N'. If `N' is a leaf and number of entries in `N' < `M'(Maximum number of entries allowed to a node), insert the data point `P' into `N'. If `N' is not a leaf, then find minimum expansion required in the MBR of each of the existing entries in `N' such that `P' could be inserted and add `P' to the entry that requires least MBR expansion. Least MBR expansion is evaluated as the minimum difference over all the set of differences of MBR(E), MBR(P) for all entries 'E' in root node 'N'. Finally, if `N' is a leaf and number of entries in `N' $\geq$`M', split `N' into two, say Split1 and Split2. Splitting is performed by taking the maximum of all the lower bounds and minimum of all the upper bounds of the MBR for each of the entries along each of the available axes. The result will be two lists say List1 and List2, each containing new entries obtained by the above explained criteria. Select two entries (Split1 and Split2), one from each list having greatest normalized separation. Now, for each entry `E' in `N', add `E' to `Split1' if minimum expansion required for `Split1' to add `E' is less than minimum expansion required for `Split2', else add `E' to`Split2'. All the pixels of the image are inserted into the R-Tree by following this procedure.

Algorithm 1 R-Tree construction for 3-dimensional data.

Input: An RGB Image I
Output: Nearest neighbours with R-Tree

1: Defining MBR (Minimum Bounding Region) for each point ‘P’ with
$[(lower_R, lower_G, lower_B), (upper_R, upper_G, upper_B)]$
where:
(R, G, B) = Pixel values of Point P
$(lower_R, lower_G, lower_B)$ = Lower bounds of region enclosing P
$(upper_R, upper_G, upper_B)$ = Upper bounds of region enclosing P
2: Let N - root node, P - data point, L - leaf node, M - maximum entries of a node, E - entries $\leq M$.
3: procedure Node Insertion
4: if $N == L$ and $len(N) < M$ then
5: Insert P in N
6: else if $N! = L$ then
7: for entry E in N do
8: mbr difference = MBR(E) - MBR(P)
9: dictionary[E] = mbr difference
10: end for
11: Q = entry E in 'dictionary' with min(mbr difference)
12: MBR(Q) = MBR(Q)+min(mbr difference)
13: Insert P in Q
14: else if N == L and $len(N) \geq M$ then
15: $M_{dim} = [], M'_{dim} = []$
16: for each MBR(E) do
17: for each dim in (R,G,B) do
18: M_{dim}.append(E with max(lower(MBR_{dim})))
19: M'_{dim}.append(E with min(upper(MBR_{dim})))
20: end for
21: end for
22: Divide M_{dim}, $M'_{dim}$with M
23: spilt1, split2 = Select values in M_{dim}, $M'_{dim}$with greatest normalized separation
24: for entry E in N do
25: for entry E' in split1 do
26: mbr difference = MBR(E') - MBR(E)
27: list1.append(mbr difference)
28: end for
29: d1 = min(list1)
30: for entry E' in split2 do
31: mbr difference = MBR(E') - MBR(E)
32: list2.append(mbr difference)
33: end for
34: d2 = min(list2)
35: if $d_1 \leq d_2$then
36: Add E to split1
37: else
38: Add E to split2
39: end if
40: d2 = minimum increase in area of MBR (split2) required to add E to split2
41: Add E to group with min(d1, d2)
42: end for
43: end if
44: end procedure
45: Repeated the procedure till all the nodes are inserted.

4.NEAREST NEIGHBOUR SEARCH WITH R-TREE

We employ another data structure known as priority queue to obtain nearest neighbours from the spatial tree. On keen observation, we find that the boxes that are nearer to the query point are prone to have the user searched points when we look for a specific set of boxes for K closest points. To leverage the advantage of this procedure, we begin our search at the top level by organizing the largest boxes in a queue to encompass everything: closest to farthest. Now, we open the closest box, which essentially removes it from the queue and inserting all its children into the queue by the side of the larger ones.

This procedure is iteratively processed wherein the nearest box is opened and the children are inserted back into the queue. Any point is confirmed to be the nearest point if when it is deleted from a queue and is an actual point. Progressively, the next nearest point is the second point from the top and so on. This algorithm works fast because of its processing wherein it must deal with only a few boxes as the overall tree is approximately the same size with respect to its tree branches (well balanced) and therefore ignoring the remaining branches.

An R-Tree is best known to find nearest neighbours [12] efficiently due to its spatial indexing structure. In this particular algorithm, the nearest neighbours for a given query point are searched in the following manner. Let `N' be the root of the tree and `P' be the query point and `L' be the leaf. Starting from the root, for each entry say `E' in `N', the distance between `P' and `E' is calculated. These distances are sorted in ascending order. The entry `E' with minimal distance is picked out. If `E' is not a leaf, then `N' is replaced with `E' (i.e, N = E) and the process is restarted. But if `E' is a leaf (i.e, E = L) then, `L' is the first nearest neighbour (1-NN) of `P' and hence `L' is added to the list of nearest neighbours. If there are `k' deferent `L' with same distance, then all of them are added to the list of nearest neighbours and they are the k - nearest neighbours of `P'. Finally, the list of nearest neighbours (NN) is returned.

Algorithm 2 aims to find the nearest neighbours of a query point `P' by searching only those entries `E' that have the minimum distance from `P' and ignoring all other entries, thereby reducing the average search time complexity. In mathematical terms, let the set of data points in the tree be (P0,P1, P2,…,Pn). Let `Pi' be the query point. Then a point `Pj' is a nearest neighbour of `Pi' if P0 <Pj<Pn and distance(Pi,Pj) < distance(Pi,X)where X belongs to set of all {P0, P1, P2,…..,Pn} and X! = Pj.

Algorithm 2 Nearest neighbour Search with R-Tree

Input: Data point P, R-Tree of Image
Output: Nearest neighbours to 'P'

```
1: Let N be the root node and P be a data point
2: for each entry E in N do
3:    dist[E] = distance(P,E)
4:    Sorted (dist) in ascending order
5: end for
6: for entry E with dist[0] do
7:    if E is not leaf then
8:    N=E
9:    repetition from step 2
10:else
11:   for each data point in E do
12:      added data point to List(NN)
13:   end for
14: end if
15: end for
16: returned NN
```

5.DENSITY-BASED CLUSTERING WITH R-TREE

The proposed method is a variant of density-based clustering. This method can be used to process data points in multi-dimensional space. To achieve this, an N-dimensional R-Tree is designed. An R-Tree is a data structure that uses spatial access methods for indexing multi-dimensional data. With the help of R-Tree, performance improvement can be observed in the nearest neighbour search for large data due to it's spatial indexing nature.

The dataset used in the input contains colour images of 3 bands. The R-Tree is first initialised with 3 dimensions, one for each colour band. Insertion is then performed by specifying the index of the data point and the bounds of the region enclosing the data point, in this case a cube. Each of the bands is given as bounds following the format $(lower_R, lower_C, lower_B)$, $(upper_R, upper_C, upper_B)$ where $(lower_R, lower_C, lower_B)$,are lower length bounds, $(upper_R, upper_C, upper_B)$ are the higher length bounds of the cube along the X,Y,Z axes.

On each of the data points a nearest neighbour query is run. In this search, for an individual query point, all the neighbouring points with bounds overlapping this query point are returned as neighbours. As well as the children of the neighbouring points if they overlap with the query point are also returned as its neighbours. The role of bounds provided earlier proves to be effective in determining the overlaps. Starting from the point with the highest neighbours, a label is assigned to the point as well as it's neighbours and they are marked as classified. The process is repeated until all the points are marked as classified.

Algorithm 3 R-Tree based density clustering

Input: Nearest neighbours from algorithm 2, Threshold t
Output: Clustered image I'

1: Let O - List(P, Length(NN), NN), C - List of clustered points, CL - Cluster label to
be assigned 0 to k, t - Threshold
2: sort O in descending by Length(NN)
3: for each item i in O do
4: if point P in O[i] is not in C then
5: Assign Label CL to P in O[i]
6: for each neighbor neigh in NN of O[i] do
7: if neigh not in CL then
8: Assign Label CL to neigh
9: add neigh to C
10:end if
11: end for
12: end if
13: Add P of O[i] to C
14: CL = CL + 1
15: end for
16: NC ←0New list of clusters
17: for each initial cluster, centroid is calculated
18: for each centroid c not in NC do
19: flag = 0
20: for each centroid CC other than c and not in NC do
21: if Distance (c, CC) ≤ t then
22: Add CC to cluster c
23: flag = 1
24: end if
25: end for
26: if (flag == 1) then
27: Add c to NC
28: end if
29: end for
30: for each c not in NC do
31: for each CC in NC do
32: add c to cluster having min(Distance(c, CC))
33: end for
34: end for

The result of the above process generates large number of clusters as R-Tree and is highly sensitive towards similar data. To reduce the number of clusters initially formed, a manual parameter "t" is introduced. The parameter "t" indicates the radius or distance within which all the clusters can be grouped as one. This reduces the number of clusters formed to an adequate number based on the value of "t". This similarity check using "t"is performed only on the cluster centres rather than each individual point to reduce time complexity.

Starting from a random cluster centre `A', a label `0' is assigned to it and it is marked as classified. Now the distance between `A' and another centre `B' that has not yet been marked as classified is calculated. If this distance is within the value of parameter "t", the same label `0' is assigned to all points within the cluster `B' and `B' is marked as classified. This process is repeated for all other cluster centres that are within the "t" radius of `A'. If there are no more clusters that can be grouped with `A', the label is incremented and the whole process is restarted with a new random cluster centre that has not been marked as classified. The process terminates when all the cluster centres have been marked as classified or when no more clusters can be grouped. The final number of clusters formed is significantly less than that of the initial process and varies with the parameter "t".

6.EXPERIMENTAL RESULTS

Tests were conducted on two RGB images (Image A, Image B) from the 'BSDS' (Berkeley Segmentation dataset) [25] of resolution 481 x 321 and one USGS [26] Sentinel multispectral satellite image (Image C) of resolution 640 x 480. To determine the accuracy of the proposed method, two of the most popular clustering metrics namely `Silhouette' [23] and `Davies-Bouldin index' (DBI) [24] are considered. The Silhouette and DB Index scores are calculated for each of these images using proposed Density-based clustering with R-Tree algorithm and compared with the scores obtained from K-Means, DBSCAN and OPTICSrun on the same images. Silhouette scores near to 1 and DB Indexes near to 0 are resulting the best.

Image	K-Means	DBSCAN	OPTICS	**DBC with R-Tree**
Image A	0.73	0.60	0.76	**0.86**
Image B	0.77	0.67	0.77	**0.88**
Image C	0.78	0.66	0.71	**0.83**

Table 1: Comparison of Silhouette values for BSDS Images(A,B) and Sentinel Image(C)using K-Means, DBSCAN, OPTICS, proposed Density-based Clustering with R-Tree algorithm

Image	K-Means	DBSCAN	OPTICS	**DBC with R-Tree**
ImageA	0.42	0.91	0.76	**0.31**
Image B	0.16	0.98	0.98	**0.11**
ImageC	0.21	0.66	0.34	**0.07**

Table 2: Comparison of DB Index values for BSDS Images(A,B) and Sentinel Image(C)using K-Means, DBSCAN, OPTICS, proposed Density-based Clustering with R-Tree algorithms

Figure 3(a), Figure 4(a) are the two original BSDS images and Figure 3(b, c, d, e) and Figure 4(b, c, d, e) are the resulting clustered images using K-Means, DBSCAN, OPTICS

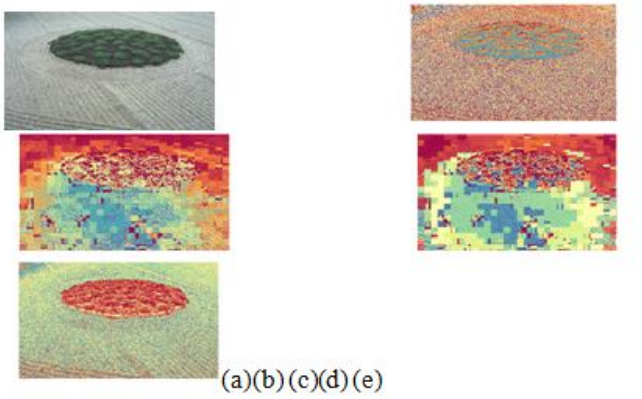

Figure 3: Clustering results for BSDS Image A: (a) Original Image (b) Clustering result using K-Means Algorithm (c) Clustering result using DBSCAN algorithm (d) Clustering result using Optics Algorithm (e) Clustering result using proposed Density-based clustering with R-Tree

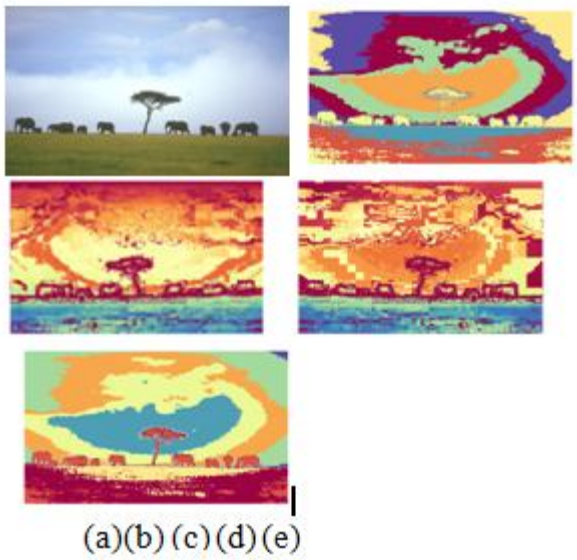

Figure 4: Clustering results for BSDS Image B: (a) Original Image (b) Clustering result using K-Means Algorithm (c) Clustering result using DBSCAN algorithm (d) Clustering result using Optics Algorithm (e) Clustering result using proposed **Density-based clustering with R-Tree**

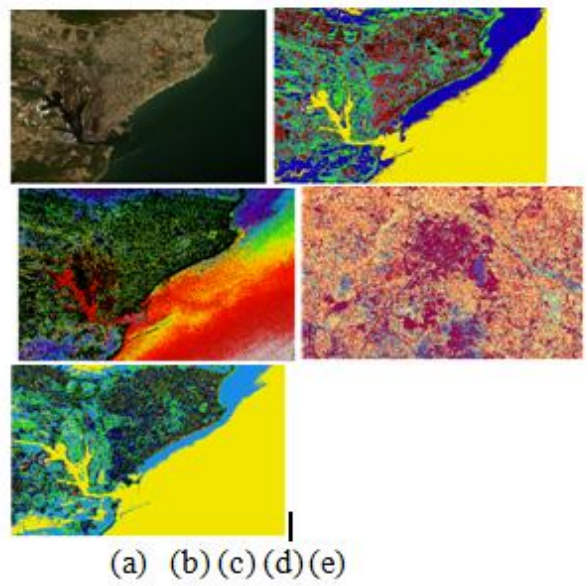

Figure 5: Clustering results for Sentinel Image C: (a) Original Image (b) Clustering result using K-Means Algorithm (c) Clustering result using DBSCAN algorithm (d) Clustering result using Optics Algorithm (e) Clustering result using proposed Density-based clustering with R-Tree and proposed Density-based clustering with R-Tree on those two BSDS images. Similarly, Figure 5(a) is the original Sentinel image and Figure 5(b, c, d, e) are the resulting clustered images using K-Means, DBSCAN, OPTICS and proposed Density-based clustering with R-Tree on that Sentinel image.

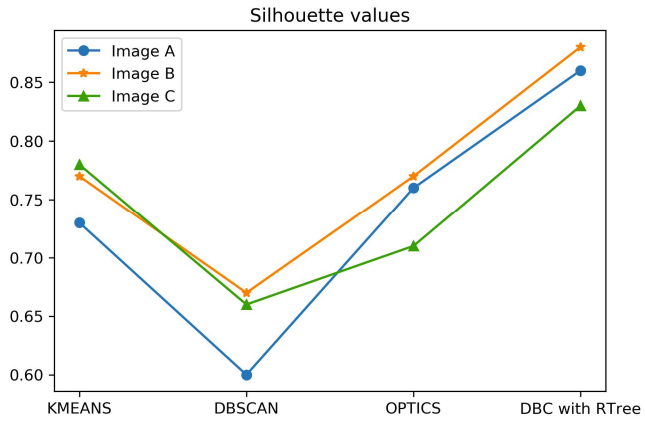

Figure 6: Comparison of Silhouette values for three images (A, B, C) using algorithms K-Means, DBSCAN, OPTICS, Density-based Clustering with R-Tree

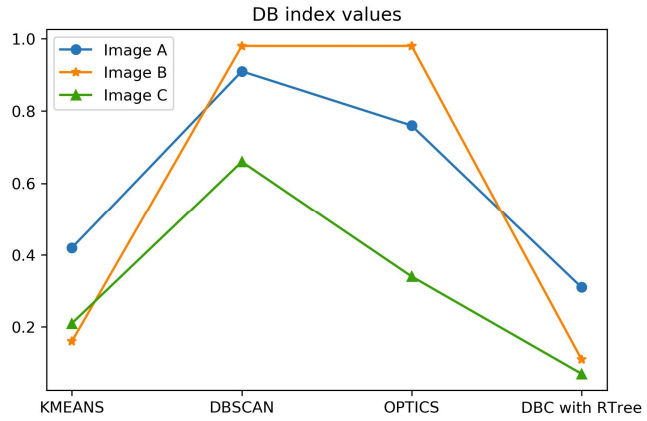

Figure 7: Comparison of DB Index values for three images (A, B, C) using algorithms K-Means, DBSCAN, OPTICS, Density-based Clustering with R-Tree

Comparison of Silhouette scores for BSDS images(A,B) and Sentinel image(C) using K-Means, DBSCAN, OPTICS, Density-based Clustering with R-Tree Comparison of Silhouette scores for BSDS images(A, B) and Sentinel image(C) using K-Means, DBSCAN, OPTICS and proposed Density-based Clustering with R-Tree algorithms is presented in table 1. As per the result analysis, the proposed density-based clustering algorithm is scoring more accuracy than other algorithms with 0.86, 0.88, 0.83Silhouette scores for Image A, Image B and Image C respectively.

Comparison of DB Index values for BSDS images(A, B) and Sentinel image(C) using K-Means, DBSCAN, OPTICS and proposed Density-based Clustering with R-Tree algorithms is presented in table 2. As per the observation, the proposed density-based clustering algorithm is scoring DB Index values near to 0 i.e, highly accurate than other algorithms with 0.31, 0.11, 0.07 DB Index scores for Image A, Image B and Image C respectively. Figure 6 is the graphical representation for Silhouette values comparison and similarly Figure 7 is the graphical representation for comparing DB Index values for three images.

7.CONCLUSION

Density based clustering using R-Tree reduces the need of specifying two parameters to just one unlike traditional DBSCAN. The `minPts' are automated through Nearest Neighbours Search and have minor to zero-effect on overall clustering efficiency. The R-Tree also proved to be efficient in handling multi-dimensional data and faster generation of Nearest Neighbours and accurate clustering of data points than other high-dimensional tree structures like `Kd-Tree' and `Ball Tree'. Experiments were performed on RGB-images from the `Berkeley segmentation dataset' and satellite image and the results compared
with three different clustering algorithms. The metrics used were `Silhouette' and `DBI'. Both metrics were observed to achieve a higher value in their respective terms than other Density based clustering methods. The visual representation shows that the clusters are well formed thus exposing the details of the image.

REFERENCES

1. Mohamad Dolatshah, Ali Hadian, Behrouz Minaei-Bidgoli, “Ball*-tree: Efficient spatial indexing for constrained nearest-neighbour search in metric spaces",arXiv:1511.00628 [cs.DB]
2. Stephen M. Omohundro, “Five Balltree Construction Algorithms (1989)", International Computer Science Institute
3. Kumar N., Zhang L., Nayar S. (2008) What Is a Good Nearest Neighbours Algorithm for Finding Similar Patches in Images?. In: Forsyth D., Torr P., Zisserman A. (eds)Computer Vision - ECCV 2008. ECCV 2008. Lecture Notes in Computer Science, vol 5303. Springer, Berlin, Heidelberg https://doi.org/10.1007/978-3-540-88688-4_27
4. Kibriya A.M., Frank E. (2007) An Empirical Comparison of Exact Nearest Neighbour Algorithms. In: Kok J.N., Koronacki J., Lopez de Mantaras R., Matwin S., Mladeni_cD., Skowron A. (eds) Knowledge Discovery in Databases: PKDD 2007. PKDD 2007.Lecture Notes in Computer Science, vol 4702. Springer, Berlin, Heidelberg
5. Mohammad Reza Abbasifard, Bijan Ghahremani, Hassan Naderi,“A Survey on Nearest Neighbor Search Methods", International Journal of Computer Applications (0975 {8887) Volume 95{ No.25, June 2014 https://doi.org/10.5120/16754-7073
6. Russell A. Brown, Building a Balanced k-d Tree in O(kn log n) Time, Journalof Computer Graphics Techniques (JCGT), vol. 4, no. 1, 50-68, 2015,http://jcgt.org/published/0004/01/03/
7. Rina Panigrahy, “An Improved Algorithm Finding Nearest Neighbor Using Kd-trees",Microsoft Research, Mountain View CA, USA
8. Fatma G unseliYa_sar ;G ozdeUlutagay, Challenges and possible solutions to density based clustering", 2016 IEEE 8th International Conference on Intelligent Systems (IS)
9. GözdeUlutagay, Efendi Nasibov, “Fuzzy and crisp clustering methods based on the neighbourhood concept: A comprehensive review", Journal of Intelligent and Fuzzy Systems: Applications in Engineering and Technology - FUZZYSS'2011: 2nd International Fuzzy Systems Symposium, Volume 23 Issue 6, November 2012 Pages 271-281.
10. Bhuyan, Rupanka, Borah, Samarjeet, “A Survey of Some Density Based Clustering Techniques", DO - 10.13140/2.1.4554.6887, 2013.
11. A. Guttman, R-trees: A dynamic index structure for spatial searching, In: Proc. Of13th Int. Conf. on

Mang. of Data ACM SIGMOD, vol. 2, 1984, pp. 47-57.

12. Norbert Beckmann, Hans-Peter Kriegel , Ralf Schneider , Bernhard Seeger , “The R*-tree: an efficient and robust access method for points and rectangles", SIGMOD'90 Proceedings of the 1990 ACM SIGMOD international conference on Management of data, Pages 322-331 https://doi.org/10.1145/93605.98741
13. Hans-Peter Kriegel, Peer Kröger , Jörg Sander, Arthur Zime, \Density-based clustering",05 April 2011, https://doi.org/10.1002/widm.30
14. M. Ankerst, M. M. Breunig,, H. P. Kriegel, and J. Sander, " OPTICS: Ordering Points To Identify the Clustering Structure," Proceedings of the 1999 ACM SIGMOD International conference on Management of data, pp. 49-60, 1999.
15. M. T. H. Elbatta, and W. M. Ashour, "A dynamic method for discovering density varied clusters," International Journal of Signal Processing and Pattern Recognition,vol: 6(1), pp. 123-134, February 2013.
16. J.L. Bently, “Multidimensional search trees in database applications", IEEE Trans.Software Eng. 5 (4) (1979) 333-340.
17. M.G. Barrios, A.J. Quiroz, “A clustering procedure based on the comparison between the k nearest neighbours graph and the minimal spanning tree, Statistics and Probability", Letters 62 (2003) 23-34.
18. P. Viswanath, V.S. Babu, “Rough-DBSCAN: A fast hybrid density based clustering method for large data sets", Pattern Recognition Letters 30(16) (2009) 1477{1488. https://doi.org/10.1016/j.patrec.2009.08.008
19. M. Ester, H.P. Kriegel, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise", In: Proc. 2nd ACM SIGKDD, Portland, Oregon,1996, pp. 226{231.
20. Fukunaga, Keinosuke; Larry D. Hostetler (January 1975). "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition". IEEE Transactions on Information Theory. 21 (1): 32{40. doi:10.1109/TIT.1975.1055330.
21. DeryaBirant, AlpKut, “ST-DBSCAN: An algorithm for clustering spatial-temporal data", Data and Knowledge Engineering, Volume 60, Issue 1, January 2007, Pages208-221
22. Anil K. Jain and Martin H. C. Law. “Data clustering: A user's dilemma". In Sankar K.Pal, Sanghamitra Bandyopadhyay, and Samb-hunath Biswas, editors, PReMI, volume3776 of Lecture Notes in Computer Science, pages 1-10. Springer, 2005.
23. Rousseeuw, Peter, \Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis", Elsevier Science Publishers B. V., J. Comput. Appl. Math., November 1987, Volume 20, doi:10.1016/0377-0427(87)90125-7.
24. Davies, David L. and Bouldin, Donald W., “A Cluster Separation Measure", IEEE Computer Society, February 1979, Volume 1, Pages 224{227,10.1109/TPAMI.1979.4766909.
25. D. Martin and C. Fowlkes and D. Tal and J. Malik, “A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics", Proc. 8th Int'l Conf. Computer Vision, July,2001, Volume 2, Pages 416{423.
26. Earth Resources Observation and Science (EROS) Center, “USGS EROS Archive -Sentinel-2", doi:/10.5066/F76W992G.