Jurnal Politeknik Caltex Riau
http://jurnal.pcr.ac.id

Modifikasi DBSCAN (*Density-Based Spatial Clustering With Noise*) pada Objek 3 Dimensi

Ibnu Daqiqil Id[1], Astrid[1] dan Evfi Mahdiyah[1]

[1]Jurusan Ilmu Komputer Universitas Riau
email: ibnu.daqiqil@unri.ac.id, astrid@unri.ac.id, evfi.mahdiyah@unri.ac.id

Abstrak

Algoritma DBSCAN merupakan algoritma pengelompokan data spasial berdasarkan kerapatan objek 2 dimensi. Pada paper ini akan dibahas bagaimana melakukan clustering pada objek 3 dimensi menggunakan algoritma DBSCAN. Modifikasi yang dilakukan adalah dengan merubah mekanisme penghitungan jarak kerapatan antara objek. Berdasarkan hasil pengujian didapatkan rata-data Silhouette Coefficient adalah 0.550 dengan eps=0.2 dan minpts= 3. Dari data tersebut, cluster yang dihasilkan memiliki stuktur yang baik dan tidak sensitive terhadap noise.

Kata kunci: *Spatial Data, Clustering, DBSCAN*

Abstract

DBSCAN algoritm is a spatial data clustering based on object density applied to 2-dimensional object. In this paper will discuss how to perform clustering in object 3 dimensions based on density using DBSCAn. Modification is done by changing the mechanism of calculating the distance density between objects. Based on the test results, the average of Silhouette Coefficient is 0.550 with eps = 0.2 and minpts = 3. Based on that data, the cluster has a good structure and constistecy.

Keywords: *Spatial Data, Clustering, DBSCAN*

1. Pendahuluan

Data mining adalah penemuan tren yang menarik atau pola pada dataset yang besar, dengan tujuan untuk menuntun pada keputusan tentang kegiatan kedepan. *Data mining* dapat juga dikatakan sebuah langkah dalam proses *Knowledge Discovery in Database* (KDD) yang terdiri dari penerapan analisis data dan penemuan algoritma yang menghasilkan enumerasi tertentu terhadap pola pada data [1]. Tan juga mengartikan *data mining* sebagai sebuah proses ekstraksi informasi baru dari sejumlah besar data yang dapat berguna dalam proses pengambilan keputusan [2]. Proses *data mining* pada sejumlah besar data spasial dikenal sebagai *spatial data mining* [3].

Spatial Data Mining adalah bagian dari *data mining* yang merupakan proses menemukan pola yang menarik dan sebelumnya tidak dikenal tetapi secara potensial dapat berguna dari dataset spasial yang besar. *Spatial data mining* merujuk kepada *extraction of knowledge, spatial relationship*, atau pola yang menarik lainnya yang tidak secara eksplisit disimpan dalam basis data [4]. Basis data spasial menyimpan *space-related data*, misalnya peta,

Dokumen diterima pada 12 Maret, 2017
Dipublikasikan pada 16 Mei, 2017

space-related data yang dimaksud adalah data topologi atau data jarak. Basis data spasial terdiri dari obyek dengan spatial dan non-spatial data. Proses penemuan untuk spatial data lebih kompleks dari data relasional. Hal ini terkait dengan efisiensi algoritma dan kompleksitas pola yang ditemukan dalam basis data spasial. *Spatial data mining* harus mempertimbangkan *neighbour object*, hal ini diperlukan karena atribut *neighbour* dari beberapa obyek mempengaruhi obyek itu sendiri [5]

Sebagian besar penelitian terbaru pada data spasial menggunakan teknik *clustering* dikarenakan oleh sifat dari data tersebut. *Clustering* merupakan proses pengelompokan sejumlah data masing besar menjadi beberapa kelas sesuai dengan ciri khas masing-masing. *Density-Based Spatial Clustering Algorithm with noise* (DBSCAN) adalah salah satu algorima pengelompokan bedasarkan kepadatan (*density*) data [6]. Algoritma ini adalah algoritma yang memberikan hasil yang paling konsisten [7] .Konsep kepadatan dalam DBSCAN menghasilkan tiga macam status dari setiap data, yaitu inti (*core*), batas (*border*), dan derau (*noise*) [8].

DBSCAN hanya dapat digunakan pada data spatial 2 dimensi. Oleh karena itu, penelitian ini bertujuan untuk memodifikasi algoritma DBSCAN sehingga dapat digunakan pada objek-objek spatial yang berada pada ruang 3 dimensi. *Clustering* data 3 dimensi banyak digunakan pada bidang *space-time analysis*, data siesmic, biologi, bioinformatika dan lain-lain. Berdasarkan permasalahan tersebut, maka penulis akan melakukan penelitian dengan judul "Modifikasi DBSCAN (*Density-Based Spatial Clustering With Noise*) Pada Objek 3 Dimensi". Dengan adanya modifikasi ini diharapkan peneliti akan dapat melakukan clustering data 3 dimensi dengan baik.

2. Kajian Pustaka

2.1 *Density-Based Spatial Clustering of Application with Noise (DBSCAN)*

Pada dasarnya ada dua jenis algortima *clustering* [9] yaitu algoritma partisi dan algoritma hierarchical. Algoritma partisi akan membentuk sebuah partisi dari sebuah basis data *D* yang berisi *n* objek sehingga menjadi *k* cluster. *k* adalah sebuah parameter input pada algoritma ini. Algoritma partisi biasanya mulai dari satu atau beberapa partisi dari *D* dan kemudian menggunakan sebuah fungsi perulangan untuk mengoptimasi fungsi objektif partisi tersebut. Setiap kluster merepresentasikan pusat grafitasi dari tiap-tiap kluster (*k-means algorithms*) atau dengan salah satu objek dari kluster yang terletak di dekat pusat dari kluster (*k-medoid algorithms*). Akibatnya, algoritma partisi menggunakan dua prosedur. Pertama, menentukan wakil k untuk meminimalkan fungsi tujuan (fungsi objektif). Langkah kedua, menetapkan setiap objek kedalam cluster "terdekat" dari objek tersebut. Langkah kedua menyiratkan bahwa partisi setara dengan diagram Voronoi dan setiap cluster terkandung dalam salah satu sel voronoi. Dengan demikian, semua bentuk cluster ditemukan oleh algoritma partisi berbentuk convex.

Diantara berbagai jenis algoritma clustering, *density based clustering* lebih efisien untuk menentukan cluster pada data dengan kepadatan yang berbeda [8]. *Density-Based Spatial Clustering of Application with Noise (*DBSCAN*)* adalah salah satu contoh pelopor perkembangan teknik pengelompokan berdasarkan kepadatan atau yang biasa dikenal dengan sebutan *density based clustering* .

Density-Based Spatial Clustering of Application with Noise (DBSCAN) merupakan sebuah metode clustering yang membangun area berdasarkan kepadatan yang terkoneksi (*density connected*). DBSCAN merupakan algoritma yang didesain oleh ester et.al pada tahun 1996 dapat mengidentifikasi kelompok-kelompok dalam kumpulan data spasial yang besar dengan melihat kepadatan lokal dari elemen-elemen basis data, dengan hanya menggunakan satu parameter input. DBSCAN juga dapat menentukan apakah informasi diklasifikasikan

sebagai *noise* atau *outlier*. Disamping itu, proses kerja DBSCAN cepat dan sangat baik untuk berbagai macam ukuran database - hampir linear.

Setiap objek dari sebuah radius area (*cluster*) harus mengandung setidaknya sejumlah minimum data. Semua objek yang tidak termasuk di dalam *cluster* dianggap sebagai *noise*. Dengan menggunakan distribusi kepadatan dari titik-titik dalam database, DBSCAN dapat mengkategorikan titik-titik tersebut ke dalam kelompok-kelompok terpisah yang menandakan kelas-kelas yang berbeda

Proses komputasi DBSCAN berdasarkan enam definisi dan dua lemma sebagai berikut [8] :

Definisi 1. *Eps-neighborhood* dari suatu titik

$$N_{Eps(p)} = \{ q \in D \mid dist(p,q) < eps \} \quad (1)$$

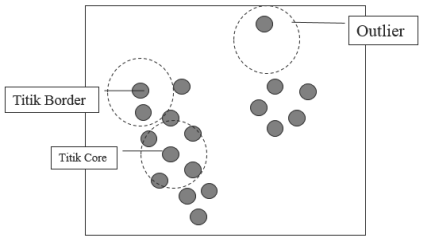
Sebuah titik yang masuk kedalam suatu cluster, titik tersebut minimal mempunyai satu titik lain yang letaknya lebih dekat ke titik tersebut dibandingkan dengan nilai *eps*.

Definisi 2. *Directly Density-Reachable*

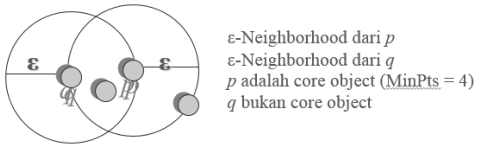
Terdapat dua macam titik yang masuk dalam suatu *cluster*, yaitu titik yang terletak di pinggir *cluster* (*border points*) dan titik-titik yang terletak di pusat *cluster* (*core points*), sebagaimana yang ditunjukkan pada Gambar 1.

$$p \in N_{Eps(Q)} \quad (2)$$

Eps-neighborhood dari suatu border point cenderung memiliki titik yang lebih sedikit dibandingkan dengan *Eps-neighborhood* dari suatu *core point* . *Border point* akan masih menjadi bagian dari suatu cluster apabila *border point* memiliki *Eps-neighborhood* dari suatu *core point q* sebagaimana yang ditunjukkan pada Gambar 2.



Gambar 1. Titik Border (*border point*) dan Titik Core (*core point*)



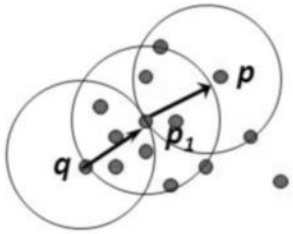
Gambar 2. Titik Border (*border point*) dan Titik Core (*core point*)

Agar titik *q* menjadi *core point*, titik tersebut perlu memiliki suatu jumlah minimum dari titik dalam *Eps-neighborhood*-nya. Kondisi *core point* ditunjukan oleh persamaan 3

$$\left\| N_{Eps(Q)} \right\| \geq MinPts \quad (3)$$

Definisi 3: *Density-Reachable*

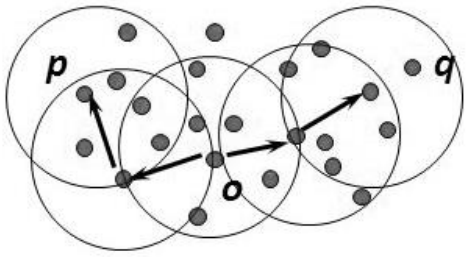
Suatu titik p merupakan *density-reachable* dari suatu titik *q* berdasarkan *Eps* dan *MinPts* jika terdapat suatu rantai titik-titik *p1…,pn, p1=q, pn=p* dimana *pi+1* merupakan *directly density-reachable* dari pi . Pada Gambar 3 ditunjukkan ilustrasi dari suatu *density-reachable.*



Gambar 3. Ilustrasi *Density-Reachable*

Definisi 4: *Density-Connected*

Terdapat kasus ketika dua *border point* akan masuk ke dalam suatu cluster yang sama tetapi dimana dua *border point* tersebut tidak membagi suatu *core point* spesifik. Dalam situasi ini titik-titik tersebut tidak akan menjadi *density-reachable* satu sama lainnya. Namun harus ada *core point* q yang mana menjadi *density-reachable* dari kedua *border point* tersebut. Pada Gambar 3 ditunjukkan bagaimana *density connectivity* bekerja. Suatu titik p merupakan *density-connected* ke titik *p* berdasarkan *Eps* dan *MinPts* jika terdapat suatu titik o dimana keduanya *p* dan *q* merupakan *density-reachable* dari o berdasarkan Eps dan *MinPts.*



Gambar 4. Ilustrasi *Density-Connected*

Definisi 5: *Cluster*

Jika suatu titik p merupakan bagian dari suatu cluster C dan titik q merupakan *density-reachable* dari titik p berdasarkan jarak tertentu dan jumlah minimum titik-titik dalam jarak tersebut, maka q juga merupakan bagian dari *cluster* C. Dua titik dimiliki oleh *cluster* yang sama *C*, dimana *p* merupakan *density-connected* ke *q* berdasarkan jarak tertentu dan jumlah minimum titik-titik dalam jarak tersebut.

Definisi 6: *Noise*

Noise adalah sekumpulan titik-titik, dalam basis data, yang tidak masuk dalam *cluster*.

Lemma 1:

Suatu *cluster* dapat dibentuk dari salah satu *core point* dan akan selalu memiliki bentuk yang sama.

Lemma 2:

Jika p menjadi suatu core point dalam suatu *cluster* C dengan jarak minimum (*Eps*) dan suatu jumlah minimum titik dalam jarak tersebut (*MinPts*). Jika sekumpulan O merupakan

density-reachable dari p berdasarkan *Eps* dan *MinPts* yang sama, maka C sama dengan sekumpulan O.

2.2 *Euclidean Space*

Ruang euclidean adalah termasuk konsep matematik dasar yang paling kuno yang dipercayai dijabarkan pertama kali oleh Euclid (323-283 SM), seorang pemikir asal yunani. Ruang Euclidean adalah ruang 2d dan 3D yang kita pakai sebagai acuan dalam menerangkan konsep matematik dasar tentang pengertian suatu ruang. Dari sini kita dapat menyatakan konsep-konsep tentang jarak, panjang dan sudut dari suatu koordinat dalam suatu dimensi. *Euclidean distance* adalah perhitungan jarak dari 2 buah titik dalam *Euclidean space*.

3. Metodelogi Penelitian

Metodologi penelitian yang digunakan untuk mendapatkan hasil sesuai dengan yang diharapkan adalah studi pustaka/literatur untuk mengetahui metode yang digunakan an dijadikan referensi dalam penelitian ini. Pengumpulan data dan informasi mengenai objek pada penelitian ini adalah hal kedua yang harus dilakukan untuk mengetahui data terkini dan dapat menetapkan metode yang digunakan sesuai dengan studi pustaka, yang dilanjutkan dengan pemodelan sistem. Langkah selanjutnya adalah mempelajari algoritma metode yang digunakan agar dapat membuat rancangan metode dengan lebih baik. Proses yang dilakukan selanjutnya adalah perancangan dan pembuatan program. Pengujian pada program yang sudah dibuat perlu dilakukan untuk mengetahui apakah modifikasi algoritma yang dilakukan dapat berfungsi dengan baik dan bagaimana kualitas *cluster* yang dihasilkan.

4. Pembahasan

4.1 Ruang Koordinat Nyata dan *Euclidean Distance* Objek 2D

Ruang koordinat nyata sering dinyatakan sebagai $\mathcal{R}^n, n$ adalah interger positif. Suatu elemen $\mathcal{R}^n$ dinyatakan sebagai **x** = (x_1, x_2, ..., x_n), x adalah bilangan nyata. Pada operasi vektor antara suatu x dan y dalam $\mathcal{R}^n$ dapat dinyatakan sebagai berikut,

$$\boldsymbol{x} + \boldsymbol{y} = (x_1 + y_1, x_2 + y_2, \dots, x_n + y_n) \tag{4}$$

$$\boldsymbol{ax} = \left(ax_q + ax_2, \dots, ax_n\right) \tag{5}$$

$$\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i = x_1 y_{q1} + x_2 y_2 + \dots + x_n y_n \tag{6}$$

Panjang suatu vektor dapat dinyatakan sebagai,

$$\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}} = \sqrt{\sum_{i=1}^{n} (x_i)^2} \tag{7}$$

Euclidean distance atau jarak Euclidean didefinisikan sebagai,

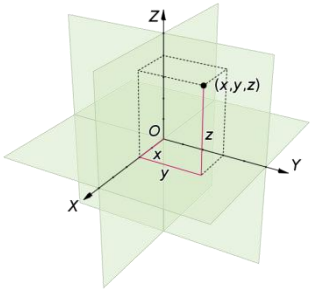
$$d(x,y) = \|x-y\| = \sum_{i-1}^{n}(x_i - y_i)^2 \quad (8)$$

Sebagai contoh, misalnya kita punya suatu bentuk trajektori objek referensi di ruang 2D yang dinyatakan sebagai $t_{ref}(t) = f\big(rx_i(t), ry_i(t)\big)$, dan trajektori aktual $t_{act}(t) = f\big(ax_i(t), ay_i(t)\big)$, maka jarak Euclidean antara dua fungsi ini adalah,

$$d(x,y) = \sqrt{(rx_1 - ax_1)^2 + (rx_2 - ax_2)^2 + \ldots + (rx_n - ax_n)^2} \quad (9)$$

4.2 *Euclidean Distance* Objek 3D

Objek 3D atau tiga dimensi adalah setiap objek tiga dimensi yang mempunyai lebar, tinggi, dan kedalaman (*width*, *height*, dan *depth*). Dengan kata lain objek 3D adalah objek yang dipaparkan dalam bentuk 3 dimensi pada paksi atau koordinat x, y, dan z (gambar 5).



Gambar 5. Ilustrasi Object 3D

Berdasarkan persamaan 9 kita telah mengetahui bagaimana cara untuk mengitung jarak pada titik 2D. Pada objek 3D, jika kita melihat pergerakan titik terjadi 2 kali pergerakan yaitu pada ruang X dan Y lalu bergerak ke ruang Z. Dengan menggunakan teorema pitagoras 2 kali dapat ditunjukkan pada persamaan 10. Persamaan inilah yang akan digunakan untuk mengitung jarak di ruang 3D.

$$d\big(x_1, y_{1,} z_1\big), d\big(x_2, y_{2,} z_2\big) = \sqrt{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + (z_1 - z_2)^2} \quad (10)$$

4.3 Modifikasi DBSCAN pada Objek 3 Dimensi

Untuk menemukan *cluster*, DBSCAN memulai dengan sembarang titik p dan mengambil semua titik yang *density-reachable* dari p dengan memperhatikan *Eps* dan *MinPts*. Jika p merupakan core point, prosedur ini menghasilkan klaster berdasarkan Eps dan *MinPts*. Jika p merupakan border point, tidak ada titik yang *density-reachable* dari p dan DBSCAN mengunjungi titik berikutnya dalam database.

Penggunaan nilai global untuk *Eps* dan *Minpts*, DBSCAN dapat menggabungkan dua *cluster* menjadi satu *cluster*, jika dua *cluster* dengan densitas (kepadatan) yang berbeda "dekat" satu sama lain. Definisikan jarak antara dua titik S_1 dan S_2 sebagai $dist\,(S_1, S_2) = \min\{dist(p,q) |\, p \in S_1, q \in S_2\}$. Kemudian dua set titik setidaknya memiliki densitas dari *cluster* tertipis akan dipisahkan satu sama lain hanya jika jarak antara dua set lebih besar dari Eps. Sebagai akibat, *recursive call* dari DBSCAN mungkin dibutuhkan untuk mendeteksi

cluster dengan nilai yang lebih tinggi untuk *MinPts*. Hanya hal ini tidak bermanfaat karena aplikasi DBSCAN menghasilkan algoritma dasar yang sangat efisien. Selanjutnya, klastering rekursif dari setiap titik hanya diperlukan dalam kondisi yang dapat dengan mudah dideteksi. Untuk menghitung jarak antara titik dapat menggunakan persamaan. Adapun modifikasi implementasi algoritma DBSCAN pada objek 3D adalah pada fungsi expandCluster.

Algorithm DBSCAN
Input: object set D, region radius ε, density threshold MinPts
output: function γ : D → N, which assigns a cluster label to each point

```
1: Label: =1
2: ∀p ∈ D do γ(p) :=' UNCLASSIFIED' enddo
3: ∀p ∈ D do
4:     if γ(p) =' UNCLASSIFIED' then
5:         if ExpandCluster(D, p, Label, ε, MinPts) = true then
6:             label := label+1
7:     endif
8: enddo
```

Fungsi ExpandCluster adalah fungsi yang bertanggung jawab untuk menentukan sebuah titik akan masuk kedalam *cluster* atau hanya berupa *noise*. Apabila suatu titik tidak mempunyai jumlah tetangga yang lebih besar dari nilai *minPts* , maka titik tersebut akan dikelompokkan ke dalam *noise*. Fungsi ini akan mencari pada semua titik yang belum mendapat label lalu mengitung jarak titik tersebut dengan regionnya melalui fungsi regionQuery.

Fungsi ExpandCluster
Input: object set D, region radius ε, density threshold MinPts
output: true atau false

```
1: seeds := regionQuery(D, CurrentPoin, ε)
2: if (|seeds| < MinPts ) then
3:     γ(CurrrenPoint) := 'noise'
4:     return false
5: else
6:     ∀p ∈ D do γ(p) := Label enddo
7:     seeds := seeds \ {CurrentPoint}
8:     while seeds ≠ ∅ do
9:         p := seeds.first()
10:        result := regionQuery(D, p, ε)
11:            if |result| ≥ MinPts then
12:            ∀ResPoin ∈ P do
13:                if γ(ResPoin) ∈ {'UNCLASSIFIED','NOISE'} then
14:                    if γ(ResPoin) =' UNCLASSIFIED'then seeds := seeds ∪ {ResPoin}
15:                γ(ResPoin) := Label
16:                endif
17:            enddo
18:        endif
19:        seeds := seeds \ {p}
20:    enddo
21:    return true
22: endif
```

Fungsi ***RegionQuery***
Input: object set D, region radius ε, density threshold MinPts
output: Object Set

```
1: poin_tetangga = []
2: ∀p ∈ D do
3:     if distance3D(p, D) ≤ eps then
4:         poin_tetangga.append(p)
5:     endif
6: enddo
8: return poin_tetangga
```

Pemanggilan fungsi regionQuery(Point, Eps) mengembalikan *Eps-Neighborhood* dari Point dalam *SetOfPoints* sebagai daftar titik. Karena *Eps-Neighborhood* diharapkan kecil jika dibandingkan dengan ukuran keseluruhan *data space*.

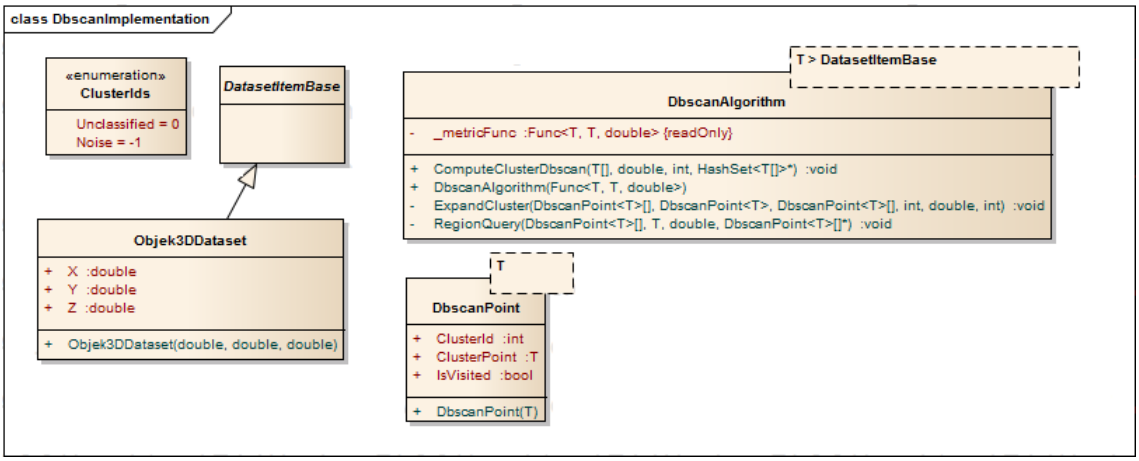
Fungsi ***distance3D***
Input: P1, P2
output: Object Distance

```
1: distance = 0
2: for d=1 to 3
3:     distance = distance + (P1[d] – P2[d])^2
4: next
5: return sqrt(distance)
```

Rata-rata kompleksitas waktu eksekusi dari 1 region query adalah O(log n). Untuk setiap *n points database*, paling tidak ada satu *region query*. Demikian, rata-rata kompleksitas waktu eksekusi DBSCAN adalah O(n * log n)

4.4 Perancangan

Berdasarkan algortima DBSCAN maka dapat dirancang *class diagram* seperti pada gambar 6. Kelas-kelas tersebut akan diimplementasikan menggunakan bahasa python. *Class* DbscanAlgorithm akan memiliki tiga method yaitu *computeClusterDbscan*, *ExpandCluster* dan *RegionQuery*. *Class* inilah yang akan mengimplementasikan algoritma DBSCAN yang telah dimodifikasi.

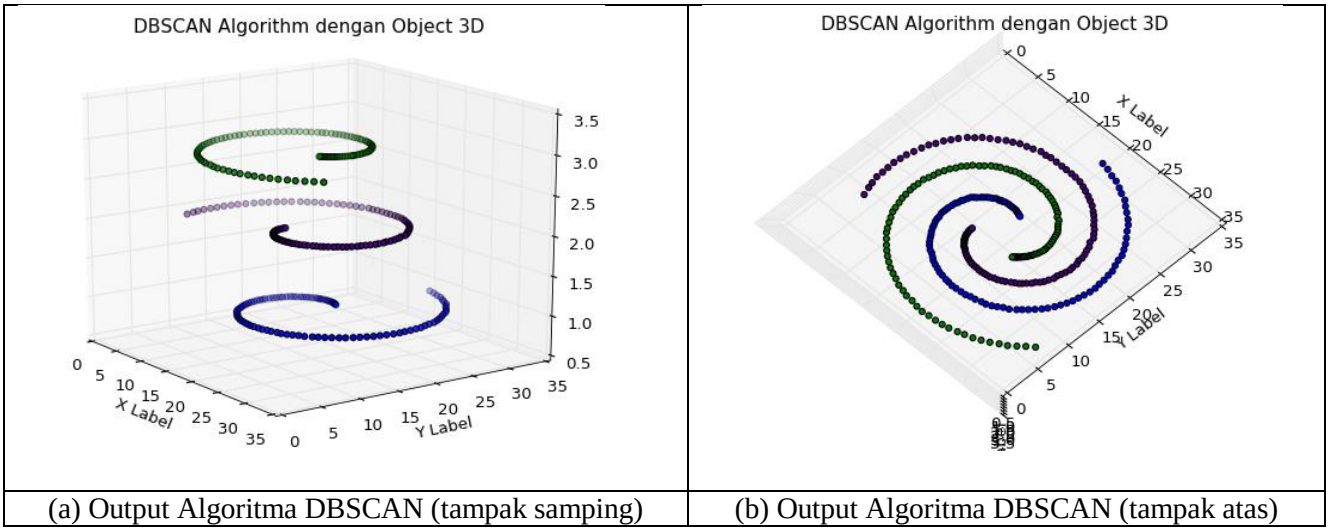


Gambar 6. Class Diagram

4.5 Hasil

Berdasarkan class diagram yang telah dibuat dilakukan implementasi menggunkan bahasa phyton. Modifikasi DBSCAN diujikan dengan dua jenis data yaitu data yang tidak memiliki *noise* dan data *multishape* memiliki *noise*. Visualisasi hasil *clustering* digambarkan dalam grafik 3D. Setiap *cluster* akan memiliki warna yang berbeda.

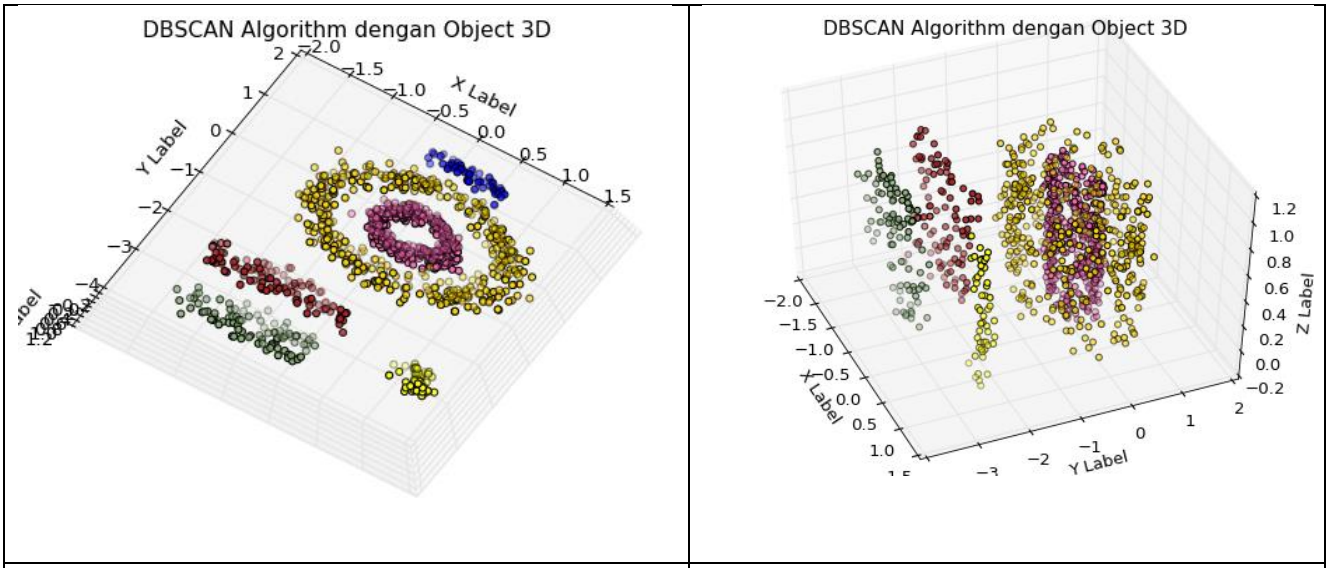
Pada data yang tidak memiliki *noise*, proses klasifikasi dapat dilakukan dengan sempurna (gambar 7). Semua titik dapat dimasukan kedalam cluster yang benar dan tidak ada noise yang ditemukan. Adapun nilai parameter *minPts*=2 dan *eps*=3 menghasilkan 3 *cluster*.



(a) Output Algoritma DBSCAN (tampak samping) (b) Output Algoritma DBSCAN (tampak atas)

Gambar 7. Visualisasi DBSCAN pada data tanpa noise

Pada data *multishape* dengan *noise*, DBSCAN dapat mendeteksi dengan baik. Dengan *minPts*=2 dan *eps*=0.25 maka objek-objek tersebut dapat dipisahkan dengan baik (gambar 8).



(a) Output Algoritma DBSCAN (tampak Atas) (b) Output Algoritma DBSCAN (tampak Samping)

Gambar 8. Visualisasi DBSCAN pada data multishape dengan noise

Selanjutnya untuk mengevaluasi kualitas *cluster* algortima DBSCAN menggunakan *Silhouette Coefficient* berdasarkan kombinasi dua parameter yaitu *minPt* dan *eps*. Adapun hasil pengujian ditunjukkan oleh tabel 1 berikut

Tabel 1. Evaluasi kualitas *cluster*

Eps	*MinPts*	Silhouette Coefficient	*Jumlah Cluster*	*Jumlah Noise Object*	*Jumlah Objek ter cluster*
0.2	2	0.572	5	91	1009
0.2	3	0.581	5	105	995
0.2	4	0.581	3	153	947
0.25	2	0.499	7	190	910
0.25	3	0.499	6	253	847
0.25	4	0.571	4	378	722

Berdasarkan hasil pengujian didapatkan nilai *Silhouette Coefficient* terbesar adalah 0.581 dengan eps=0.2 dan minpts= 3. Hasil tersebut mengidetifasikan bahwa objek tersebut tercluster dengan baik.

5. Kesimpulan

Adapun kesimpulan dari modifikasi algoritma ini adalah

1. Algoritma DBSCAN dapat dimodifikasi untuk mengelompokkan objek 3 dimensi dengan cara merubah fungsi region query dan pengitungan jarak antara objek menggunakan *euclidian distance* pada persamaan 10.
2. Data hasil modifikasi menunjukkan bahwa cluster yang dihasilkan memiliki struktur yang baik dan tidak sensitif terhadap *noise* yang ada pada data

Daftar Pustaka

[1] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. "Knowledge Discovery and Data Mining: Towards a Unifying Framework," in *Proc. of the Second International Conference on Knowledge Discovery*, 1986, Portland.

[2] P.-N. Tan. *Introduction to Data Mining*. Boston: Pearson Education, 2006.

[3] C. J. Matheus, P. K. Chan, and G. Piatetsky-Shapiro. "Systems for Knowledge Discovery in Databases," *IEEE Transactions on Knowledge and Data Engineering*, pp. 903-913, 1993.

[4] H. Jiawei and K. Micheline, *Data Mining: Concept and Techniques*. USA: Morgan Kaufmann, 2001.

[5] D. Verma and R. Nashine. "Trends in Spatial Data Mining. Data Mining: Next Generation Challenges and Future Directions," *International Journal of Modelling and Optimalization*, pp. 603-608, 2012.

[6] M. Ester *et al*. "A Density-based Algorithm for Discovering Clusters in Large Spatial Database with Noise," *KDD96*, Munchen, 1996, p. 226.

[7] A. Gunawan. "A Faster Algorithm for DBSCAN," Master Thesis, Technische Universiteit Eindhoven, Eindhoven, 2013.

[8] H. Bäcklund, A. Hedblom, and N. Neijman. "A Density-Based Spatial Clustering of Application with Noise," Linköpings Universitet, Nov. 30, 2011.

[9] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.