UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DANIEL CARVALHO DE RAMOS

ANÁLISE COMPARATIVA DE ALGORITMOS DE CLUSTERIZAÇÃO PARA RECONHECIMENTO DE OBJETOS EM SISTEMAS DE RADAR AUTOMOTIVO

PONTA GROSSA

2022

DANIEL CARVALHO DE RAMOS

ANÁLISE COMPARATIVA DE ALGORITMOS DE CLUSTERIZAÇÃO PARA RECONHECIMENTO DE OBJETOS EM SISTEMAS DE RADAR AUTOMOTIVO

Comparative analysis of clustering algorithms for object recognition in automotive radar systems

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador(a): Max Mauro Dias Santos

PONTA GROSSA

2022

4.0 Internacional

Esta licença permite download e compartilhamento do trabalho desde que sejam atribuídos créditos ao(s) autor(es), sem a possibilidade de alterá-lo ou utilizá-lo para fins comerciais. Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

DANIEL CARVALHO DE RAMOS

ANÁLISE COMPARATIVA DE ALGORITMOS DE CLUSTERIZAÇÃO PARA RECONHECIMENTO DE OBJETOS EM SISTEMAS DE RADAR AUTOMOTIVO

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 29/abril/2022

_____

Max Mauro Dias Santos
Doutorado
Universidade Tecnológica Federal do Paraná

_____

Cristhiane Gonçalves
Doutorado
Universidade Tecnológica Federal do Paraná

_____

Mauren Louise Squario Coelho de Andrade
Doutorado
Universidade Tecnológica Federal do Paraná

PONTA GROSSA

2022

Dedico esse trabalho a minha família e a todos que amam tecnologia e a indústria automotiva.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado saúde, coragem, persistência, inteligência e sabedoria para conduzir meus planos durante toda a minha vida, por todo o discernimento e por ter colocado as pessoas certas no meu caminho. E com a permissão d'Ele vejo esse sonho se tornando realidade.

A minha mãe Maria de Lourdes, sem ela esse sonho não seria possível, agradeço por acreditar na minha capacidade e por ter me incentivado a sempre estudar, e ir em busca de um futuro melhor.

Aos meus colegas de curso, que de uma forma ou outra contribuíram para que eu chegasse ate aqui, pelos momentos bons e ruins que passamos juntos, pelas broncas que tomei quando precisei.

Aos grandes professores que tive o prazer de ser aluno durante o curso, quais contribuíram bastante para com a pessoa que sou hoje, e com o profissional que serei daqui a alguns dias. Pelo incentivo na continuação dos estudos e o apoio nas realizações acadêmica.

Um agradecimento mais que especial ao meu orientador Dr. Max Mauro Dias Santos pelo compartilhamento de conhecimento para comigo, por todo apoio, encorajamento do início ao fim na construção deste trabalho, por todo tempo disposto, por todos os momentos em que se preocupou e me fez acreditar que conseguiria, teve paciência e compreensão. Além de um grande professor, é também um ser humano excepcional.

As demais pessoas que torcem pelo meu sucesso, que mandaram energias positivas e que contribuíram de alguma forma, meu muito obrigado!

RESUMO

CARVALHO, Daniel. ANÁLISE COMPARATIVA DE ALGORITMOS DE CLUSTERIZAÇÃO PARA RECONHECIMENTO DE OBJETOS EM SISTEMA DE RADAR AUTOMOTIVO. XX f. Trabalho de conclusão de curso – Campus Ponta Grossa, Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2022.

A grande dificuldade do ramo automotivo é criar um sistema de navegação seguro e confiável para o veículo autônomo, é algo que envolve muitas etapas entre elas, a fusão de sensores e redes de comunicação veicular para o reconhecimento de objetos, uma alternativa para esse desafio é a utilização de algoritmos de clusterização em sistemas de radares automotivos. O algoritmo de clusterização pode ser definido como uma técnica de *Machine Learning* que envolve o agrupamento de pontos de dados, e funciona da seguinte maneira, dado um conjunto de pontos de dados, podemos usar um algoritmo de agrupamento para classificar cada ponto de dados em um grupo específico. Em teoria, os pontos de dados que estão no mesmo grupo devem ter propriedades e ou recursos semelhantes, enquanto os pontos de dados em grupos diferentes devem ter propriedades e ou recursos altamente diferentes. O agrupamento é um método de aprendizado não supervisionado e é uma técnica comum para análise de dados estatísticos usada em muitos campos. Existem diversos métodos que foram desenvolvidos para a aplicação da clusterização, dentre eles temos dez métodos principais, o *Affinity Propagation, Agglomerative Clustering, BIRCH, DBSCAN, K-Means, Mini-Batch K-Means, Mean Shift, OPTICS, Spectral Clustering, Mixture of Gaussians*. Nesse trabalho, vamos apresentar uma análise comparativa dos algoritmos de clusterização, para verificarmos, qual tem a maior eficiência para ser utilizado em um sistema de radar automotivo para o reconhecimento de objetos, ponto de extrema importância, para a construção de veículos autônomos.

Palavras-chave: Radar automotivo, *clustering,* algoritmos, reconhecimento de objetos e ADAS.

ABSTRACT

CARVALHO, Daniel. COMPARATIVE ANALYSIS OF CLUSTERING ALGORITHMS FOR OBJECT RECOGNITION IN AUTOMOTIVE RADAR SYSTEM. XXF. Completion of course work – Campus Ponta Grossa, Federal Technological University of Paraná. Ponta Grossa, 2022.

The great difficulty of the automotive industry is to create a safe and reliable navigation system for the autonomous vehicle, it is something that involves many steps, among them, the fusion of sensors and vehicular communication networks for the recognition of objects, an alternative to this challenge is the use of clustering algorithms in automotive radar systems. Clustering algorithm can be defined as a Machine Learning technique that involves grouping data points, and it works as follows, given a set of data points, we can use a clustering algorithm to classify each data point into a specific group. In theory, data points that are in the same group should have similar properties and/or features, while data points in different groups should have highly different properties and/or features. Clustering is an unsupervised learning method and is a common technique for analyzing statistical data used in many fields. There are several methods that have been developed for the application of clustering, among them we have ten main methods, *Affinity Propagation, Agglomerative Clustering, BIRCH,* DBSCAN*, K-Means, Mini-Batch K-Means, Mean Shift, OPTICS, Spectral Clustering, Mixture of Gaussians.* In this work, we will present a comparative analysis of the clustering algorithms, to verify which one has the highest efficiency to be used in an automotive radar system for object recognition, a point of extreme importance, for the construction of autonomous vehicles.

Keywords: Automotive radar, Clustering, Algorithms, Object recognition and ADAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ADAS	*Advanced Driver Assistance System*
AD	*Autonomous Driving*
AV	*Autonomous Vehicle*
AEB	*Autonomous Emergency Braking*
ACC	*Adaptive Cruise Control*
DBSCAN	Clusterização Espacial Baseada em Desnidade de Aplicações com Ruído
GMMs	Modelos de mistura gaussiana
GSA	Grupo de sistemas automotivos
UTFPR-PG	Universidade Tecnológica Federal do Paraná – Ponta Grossa

SUMÁRIO

1 INTRODUÇÃO

O sistema de navegação de um veículo autônomo depende de uma fusão robusta de sensores, o sistema é composto por um conjunto de câmeras, radares, lidares e ultrassônicos, o grande desafio desses sensores é fazer o reconhecimento de objetos durante o percurso do veículo, pois em algum momento esse sistema pode falhar.

Criar um sistema de navegação seguro e confiável para o veículo autônomo não é uma tarefa simples. Com o avanço da tecnologia de radares, é possível implementar algoritmos de clusterização para que os radares reconheçam os objetos e seja capaz de classifica-los, essa é uma forma de aprimorar a utilização de sistemas de radares, pois diferente de outros sensores, ele tem a vantagem de funcionar muito bem em condições climáticas adversas como neblinas e dias chuvosos, e antes os radares que só conseguiam medir velocidade, hoje conseguem detectar com precisão qual o objeto que está na sua frente.

O radar é uma tecnologia bastante conhecida e utilizada, que se baseia no envio e recebimento de ondas eletromagnéticas para medir, detectar e localizar obstáculos no ambiente. O radar é especialmente indicado para aplicações automotivas, pois os veículos são bons refletores para as ondas eletromagnéticas e, assim, sua distância, posição e velocidade podem ser determinadas com precisão, o radar é de extrema importância para aplicação em veículos autônomos, pois sua confiabilidade e versatilidade os tornam peças insubstituíveis em sistemas ADAS/AD modernos, a mudança para frequências mais altas de 76 para 81 GHz trouxe novos desafios tecnológicos, isso fez com que antes o radar que só media a velocidade de automovel, agora, junto com a utilização das técnicas de clusterização seja capaz de detectar e localizar obstáculos no ambiente, como pedestres, ciclistas, carros e caminhões.

Entre as tecnologias de visão, os sistemas de radar são a tecnologia mais bem estabelecida e mais segura, introduzida em 2000 com sistemas de radar de curto alcance (SRR), dedicados à detecção de ponto cego ou cruzamento de linha. Os radares incluem SRR e Radar de Longo Alcance (LRR), fornecendo travagem de emergência ou Controle de Cruzeiro Adaptativo, grandes empresas que fabricam radar, tem mostrado uma evolução significativa na tecnologia de seus produtos, como veremos nesse trabalho, o radar possui alguns parâmetros e cada um deles deve ser considerado, dependendo da aplicação do sensor, como por exemplo o short range, e o Long range, que é o grau de alcance do radar, a maioria dos veículos utilizara 2 ou mais radares, por isso é muito importante a etapa de fusão de sensores para que eles tenham o máximo de aproveitamento na captura de objetos.

Os algoritmos de clusterização, podem ser utilizados em sistema automotivos de veículos autônomos. Os veículos autônomos são carros que utilizam um sistema de navegação independente do controle de um motorista, é utilizado para o transporte de pessoas ou bens, os veículos autônomos utilizam sensores para detectar os objetos ao seu redor, entre eles, radar, lidar e câmeras, para realizar a comunicação e a fusão entre esses sensores, é necessário técnicas que façam esse processo e crie um sistema de navegação confiável e seguro para o reconhecimento de objetos, e desta forma eliminar a possibilidade de que acidentes ocorram com o veículo autonômo, sendo assim uma alternativa para realizar esse processo é utilizar os algoristmos de clusterização.

Atualmente temos dez principais algoritmos de clusterização, são eles, o *Affinity Propagation, Agglomerative Clustering, BIRCH, DBSCAN, K-Means, Mini-Batch K-Means, Mean Shift, OPTICS, Spectral Clustering, Mixture of Gaussians,* vamos explorar o funcionamento e a eficiência de cada um desses algoritmos e verificar qual tem o melhor reconhecimento de objetos, para que assim seja aplicado em um sistema de radar automotivo, que terá os seguintes processos, a nuvem de pontos geradas pelo radar recebe a aplicação do algoritmo de clusterização, para fazer o agrupamento dos clusters, logo após, é feito um reconhecimento de objetos em cima dos clusters, aplicando alguma tecnica de rede neural, como por exemplo a YOLO e como resultado final a classificação dos objetos captados pelo radar a partir da simulação computacional.

1.1 OBJETIVOS

Este trabalho é proposto com a finalidade de analisar o funcionamento dos algoritmos de clusterização e verificar quais tem os melhores desempenhos para a implementação no sistema de radar automotivo para o reconhecimento e rastreamento de objetos.

1.2 Objetivos específicos

Os objetivos específicos são:

- Identificar e analisar as características e propriedades de radar automotivo.
- Descrever, analisar e comparar os algoritmos de clusterização para aplicações em radar automotivo.
- Definir métricas para avaliar o desempenho de três algoritmos de clusterização.

- Identificar indices de desempenho para avaliar o uso de recursos computacionais como processamento, memória e comunicação.
- Simular a detecção, aplicação dos algoritmos de clusterização para a identificação e rastreamento de objetos em cenários de testes.
- Analisar comparativamente os resultados obtidos.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido em 6 capítulos, onde o capítulo 2 trata sobre o radar automotivo, o funcionamento, bem como suas características e propriedades.

No capítulo 3, apresenta-se as técnicas de clusterização, seu funcionamento, vantagens e desvantagens de cada algoritmo.

No capítulo 4, serão mostrados as aplicações e simulações dos 3 algoritmos de clusterização com o melhor desempenho no requisito da distribuição dos dados de uma nuvem de pontos geradas pelo radar automotivo.

No capítulo 5, tem-se as simulações e os resultados da aplicação de algoritmo de clusterização para o reconhecimento de objetos, onde são apresentados alguns pontos de discussões e recomendações a respeito de como aplicar esses conceitos formais de testes no domínio de aplicações.

Por fim, no capítulo 6, tem-se as Conclusões, contendo uma síntese dos conhecimentos adquiridos, e futuros temas para discussão.

2 RADAR AUTOMOTIVO

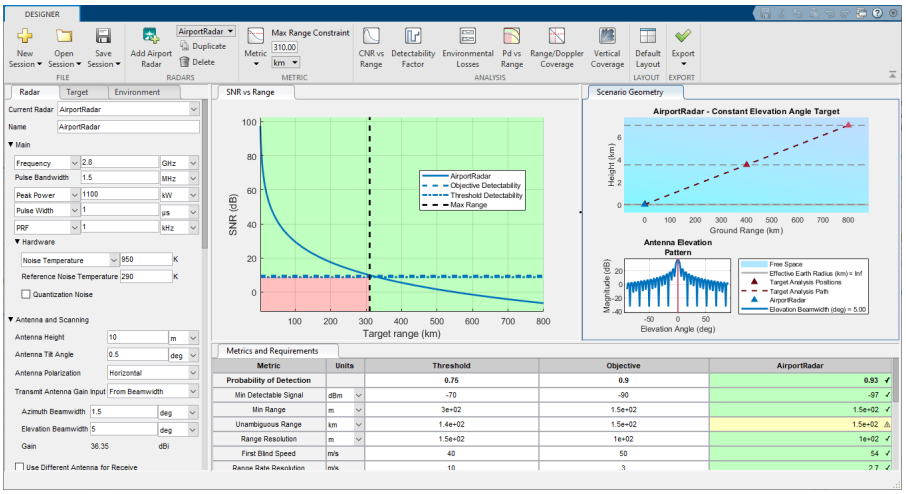
O radar é um sensor eletromagnético que em veículos autonômos são utilizados para medir a distância e a velocidade do veículo da frente, porém uma nova aplicação vem sendo desenvolvida, a partir de uma nuvem de pontos geradas no campo de visão do radar, e com a ajuda de um algoritmo de clusterização, o radar é capaz de definir aglomerados e identificar objetos. (RENESAS ELECTRONICS, 2022)

O funcionamento do radar clássico consiste em enviar um sinal de transmissor com certa frequência que atinge o objeto e retorna ao receptor, desta forma é possível medir a frequência e o tempo de batimento. Isso permite medir a posição e velocidade do objeto.

O radar tem seus prós e contras, porém um ponto importante é que ele é menos sensivel as condições climáticas, e também possui um longo alcance de detecção e um boa resolução e estimativa de posição. No MATLAB temos algumas opções para implementar sistemas de radar multifuncionais.

O Radar Toolbox inclui algoritmos e ferramentas para projetar, simular, analisar e testar sistemas de radar multifuncionais. Exemplos de referência fornecem um ponto de partida para a implementação de sistemas de radar aerotransportados, terrestres, embarcados e automotivos, basta configurar o radar designer que permite que usuário possa projetar um novo sistema de radar a partir de um dos cinco tipos de radar predefinidos, defina os requisitos de desempenho, como ilustrado na figura 1.

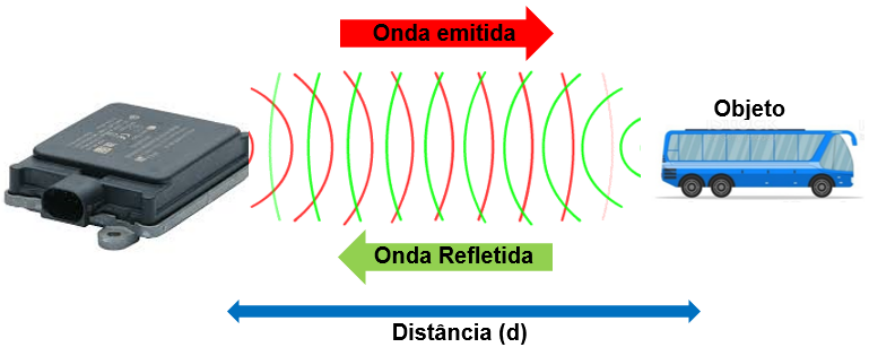
Figura 1: Exemplo de sistema de radar MATLAB.



Fonte: MATLAB (2022)

Agora será mostrado o funcionamento e a evolução dos radares. A figura 2 mostra o comportamento das ondas que são emitidas e recebidas pelo radar.

Figura 2: Funcionamento do radar com objeto parado.



Fonte: Autória própria (2022)

O cálculo da velocidade e da distância do veículo que está na frente de um radar é mostrado na figura 3. O radar envia a onda eletromagnética, essa reflete no veículo que está a uma velocidade v_r, e retorna para o radar como a soma $f + f_D$.

Figura 3: Radar frequência e velocidade com objeto em movimento.

Fonte: Autoria própria (2022)

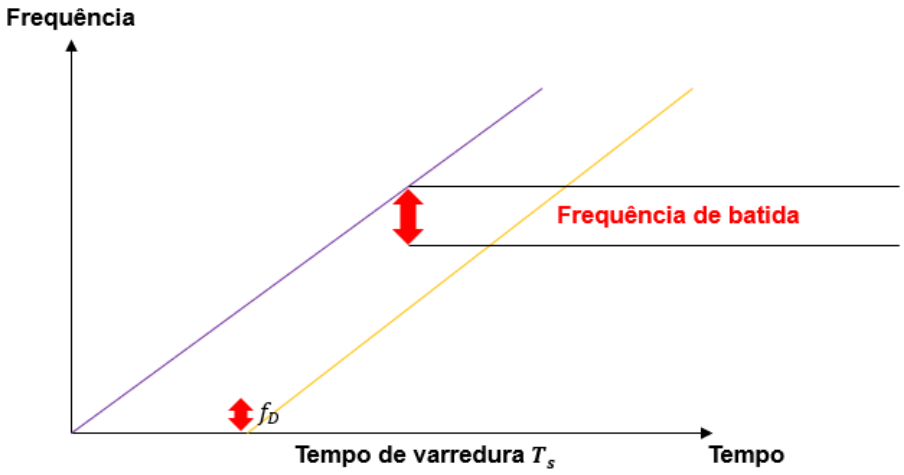
Sabemos que um alvo em movimento induz a uma frequência Doppler:

$$f_D = \frac{2v_r}{\lambda} \tag{1}$$

sendo λ o comprimento de onda eletromagnética do radar medida em m, a frequência Doppler f_D em Hz e a unidade de v_r em m/s.

A figura 4 apresenta um gráfico com o comportamento da frequencia da ilustração representada na figura 3, é possível observar que a frequencia de batida não é relacionada apenas com o range do alvo, mas também à sua velocidade relativa em relação ao radar, ela aconetce quando 2 ondas de diferentes frequencias se sobrepõe.

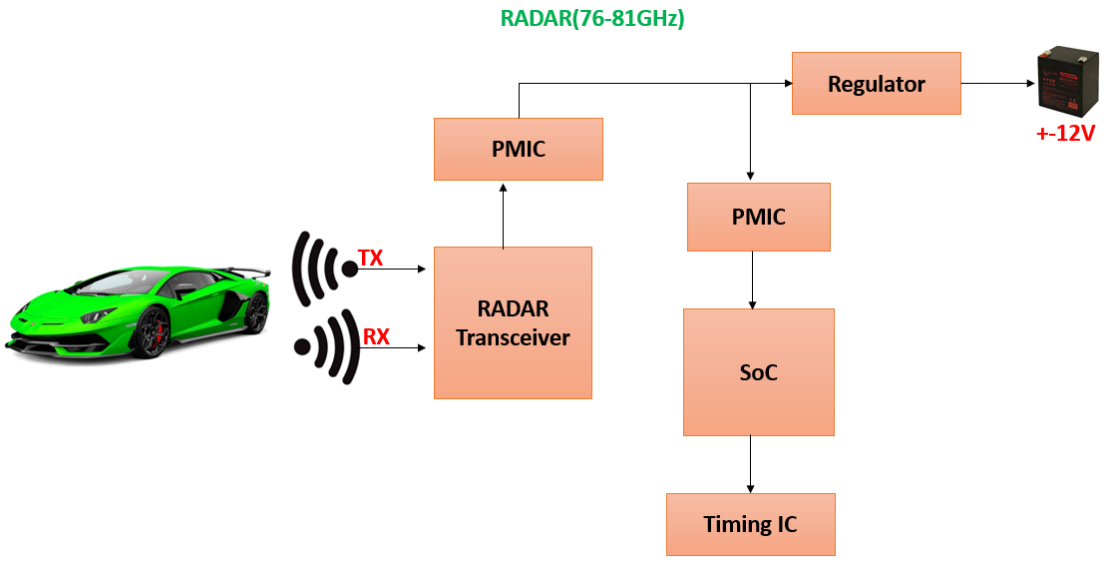
Figura 4: Gráfico das frequências



Fonte: Autoria própria (2022)

A topologia básica de um radar mostrada na figura 5, inclui um ou mais transceptores de radar MMIC, conectados a uma unidade de processamento de alto desempenho (MCU ou SoC). (RENESAS ELECTRONICS,2022).

Figura 5: Arquitetura básica de um radar.



Fonte: Autoria própria (2022)

2.1 Geração de Radar Automotivo

As gerações dos radares, é muito importante para tecnologia, pois mostra que com a evolução da tecnologia dos semicondutores, assim, os radares que antes, só conseguiam calcular a velocidade de um objeto, hoje conseguem com precisão identificar qual é o objeto que está a sua frente, isso faz com que o radar apresente uma alta confiabilidade em aplicações como ADAS/AD, e também no AEB que são sistemas de assistência ao condutor, desta forma o radar tem sua aplicação em veículos autônomos, pois diferente de outros sensores, o radar funciona em diversas condições climáticas. (RENESAS ELECTRONICS,2022).

1° Geração (1970-1980) – Primeiras aplicações para radar automotivo em sistemas totalmente analógicos. Identificação de objetos;

2° Geração (1980-1990) – Sistemas de radio frequência com placas digitais separadas. Identificação de objetos, posição e velocidade;

3° Geração (1990-2005) – Melhor desempenho do sistema de radio frequência com placa digital ainda separados, tendo um RF menor e com variado alcance. Sistemas SoC e IC;

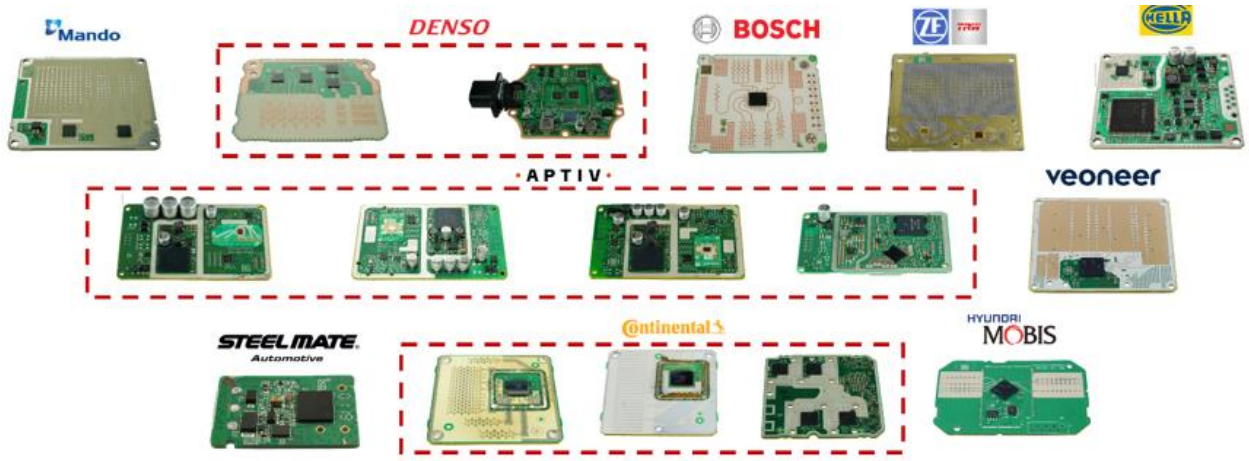
4° Geração (2005-2025) - Diversas antenas para identificação de formato de objetos e extração de características. Maior poder de processamento e implementação de algoritmos de clusterização. Maior, exatidão, precisão e resolução sob diferentes condições;

5° Geração (2025-Futuro) – MIMO radar, algoritmos mais avançados para reconhecimento de objetos e eventos. Identificação da distância, velocidade, orientação e altura de objetos. Redução de tamanho e capacidade de identificar formato de objetos com aplicação de técnicas de inteligência artificial.

Quanto maior a geração do radar, mais apurado é seu nível de detecção e classificação de objetos, os radares de ultima geração são equipados com câmeras, classificados com imaging, são capazes que reconhecer objetos e classifica-lós.(RENESAS ELECTRONICS,2022)

A figura 6 apresenta as placas utilizadas na construção de alguns radares dos principais fabricantes do mercado, o radar da Mando, DENSO, BOSCH, VEONEER.

Figura 6: Placas de radares.



Fonte: SYSTEMPLUS (2020)

As gerações de radares de diferentes fabricantes são mostrados na figura 7.

Figura 7: Geração de Radares.

Fonte: SYSTEMPLUS (2022)

O radar tem algumas classificações quanto a sua aplicação, é o caso do short range e o Long range, os radares são classificados de acordo com União Internacional de Telecomunicações (ITU), como short range, middle range, long range e imaging.

Nos veículos autonomos é possível colocar mais de um radar, a influência de um radar sobre o outro, é o alcance das zonas de detteção, quanto mais radares for colocado no veículo, mas apurado vai ser sua detecção de objetos que estejam mais próximos, devido a sua confiabilidade, os radares são aplicados em funções ADAS/AD e tornam o veículo e a estrada

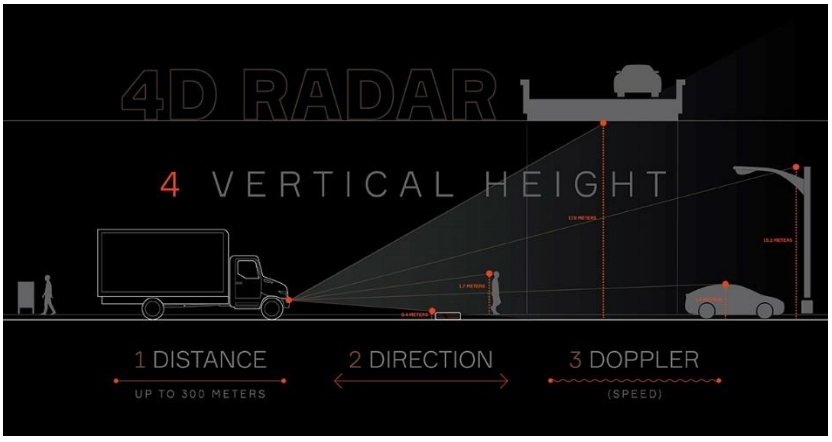
mais seguros. (RENESAS ELECTRONICS,2022).

O sistema de radar MIMO, é um nova tecnologia que está sendo desenvolvida, é um sistema de radar de múltiplas antenas. Cada antena de transmissão irradia uma forma de onda arbitrária independentemente das outras antenas de transmissão. Cada antena receptora pode receber esses sinais. Devido às diferentes formas de onda, os sinais de eco podem ser reatribuídos ao único transmissor. De um campo de antena de N transmissores e um campo de K receptores resulta matematicamente em um campo virtual de elementos K·N com um tamanho ampliado de uma abertura virtual (JIAN LI, 2009).

Outra tecnologia em evolução são os radares Imaging 4D, é um sensor de alta resolução e longo alcance que oferece vantagens, principalmente quando se trata de identificar a altura de um objeto, esses sensores adicionam informações verticais, e desta forma conseguem receber o rótulo de radar de imagens, devido à riqueza dos dados que retornam; isto é, com dados horizontais e verticais, o radar pode detectar muitos pontos de reflexão diferentes, que, quando mapeados, começam a se assemelhar a uma imagem.. Essa tecnologia é importante no desenvolvimento de sistemas avançados de assistência ao motorista ADAS. (RENESAS ELECTRONICS)

A figura 8 apresenta a visualização que o radar imaging 4D recebe do ambiente, nesse radar é adicionado a terceira dimensão x, y e z retornando as informações do ambiente como imagem.

Figura 8: Funcionamento Radar 4D Imaging.



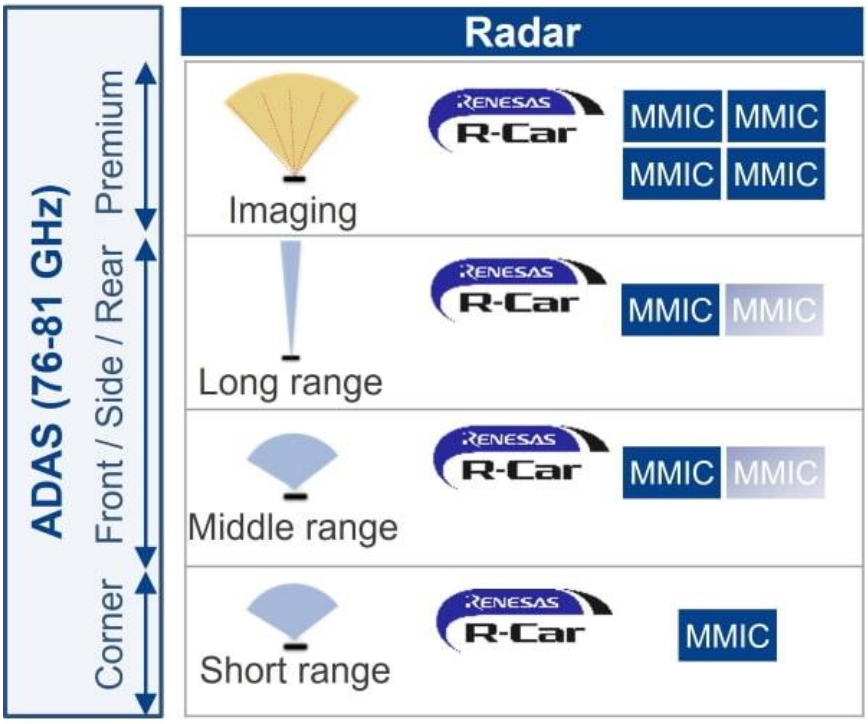
Fonte: APTIV (2022)

As informações aqui apresentadas sobre classificação e demais características dos

radares estão disponíveis no relatório de comparação das fichas técnicas dos fabricantes de radar e estão disponíveis em (RENESAS ELECTRONICS,2022).

As classificações dos radares é mostrada na figura 9, de acordo com seu campo de visão, alcance e nível de processamento.

Figura 9: Alcance dos diferentes tipos de radar.



FONTE: RENESAS (2022)

A União Internacional de Telecomunicações define duas categorias de sistemas de radar automotivo, dependendo de sua função:

- Categoria 1: Inclui sistemas de radar que proporcionam funções de conforto ao condutor, permitindo uma condução mais tranquila. Esta categoria inclui o controle de cruzeiro adaptativo (ACC) e o radar de prevenção de colisões (CA), para alcances de medição de até 250 metros (ITU).

- Categoria 2: Define sensores para aplicações de alta resolução, que contribuem para a segurança passiva e ativa de um veículo, como por exemplo detecção de ponto cego, assistente de mudança de faixa e alerta de tráfego traseiro, detecção de pedestres e bicicletas próximo a um veículo. O alcance é inferior à Categoria 1, com um máximo de 50m a 100m, dependendo da aplicação. O objetivo desses sistemas

é melhorar a segurança no trânsito, aumentando a segurança passiva e ativa de um veículo. (ITU).

Os tipos de radar também podem ser classificados de acordo com a faixa de medição:

- Radar de curto alcance (SRR), com grande campo de visão e alta resolução e alcance de até 50 m.
- Radar de médio alcance (MRR), com campo de visão médio e alcance de até 100 m,
- Radar de longo alcance (LRR), que não requer alta resolução ou amplo campo de visão, mas visa o maior alcance possível, até 250m.

O quadro 1 mostra os tipos de radar e as suas aplicações, 3 tipos diferentes de alcance são mostrados, podendo ser com alta resolução ou não, o SRR tem um alcance curto, porém alta resolução, é excelente para evitar colisões utilizando o sistema de frenagem de emergência AEB e manter a segurança no trânsito, já o MRR tem um alcance médio, visa segurança no trânsito e assistência ao condutor, já o LRR não tem tanta qualidade na resolução, mas tem um longo alcance desta forma eles proporcionam funções de conforto ao condutor, permitindo uma condução mais tranquila.

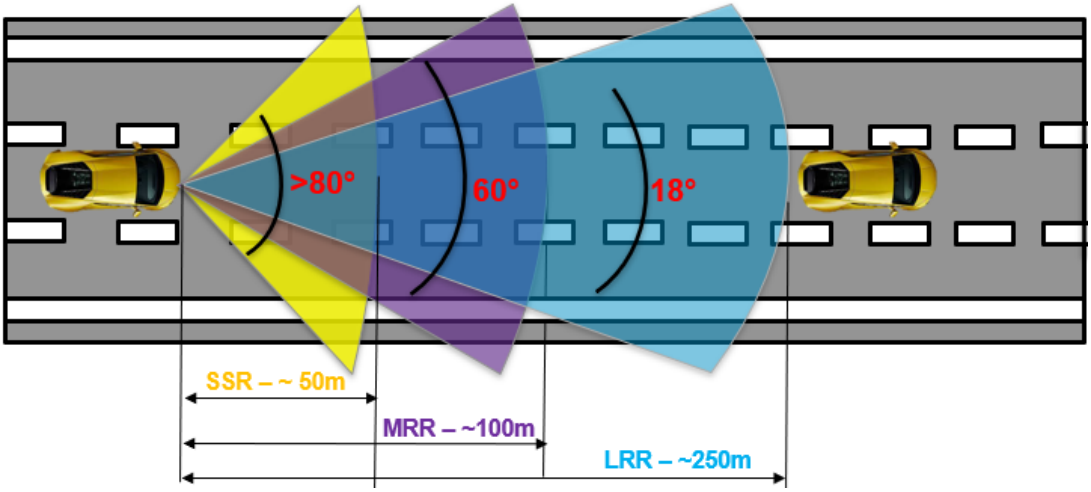
Quadro 1: Tipos de radar e suas aplicações.

Tipo de Radar	Aplicação	Motivo
SRR	AEB Detecção de Ponto cego	Grande campo de visão e alta resolução e alcance de até 50 m, ideal para frenagem de emergência e segurança no trânsito.
MRR	Lane change Assistance Detecção de pedrestes e ciclistas próximos a outros veículos,	Possui um campo de visão médio e alcance de até 100 m, ideal para o auxilio de mudança de faixa e detecção de pedestres e ciclistas na estrada.
LRR	ACC Forward/Rear Collision Warning	Não requer alta resolução ou amplo campo de visão, mas visa o maior alcance possível, até 250m.

FONTE: Autoria própria (2022)

A figura 10 apresenta as faixas de medição, o alcance em metros e a classificação do radar de acordo com união internacional de telecomunicações.

Figura 10: Faixa de medição dos radareS

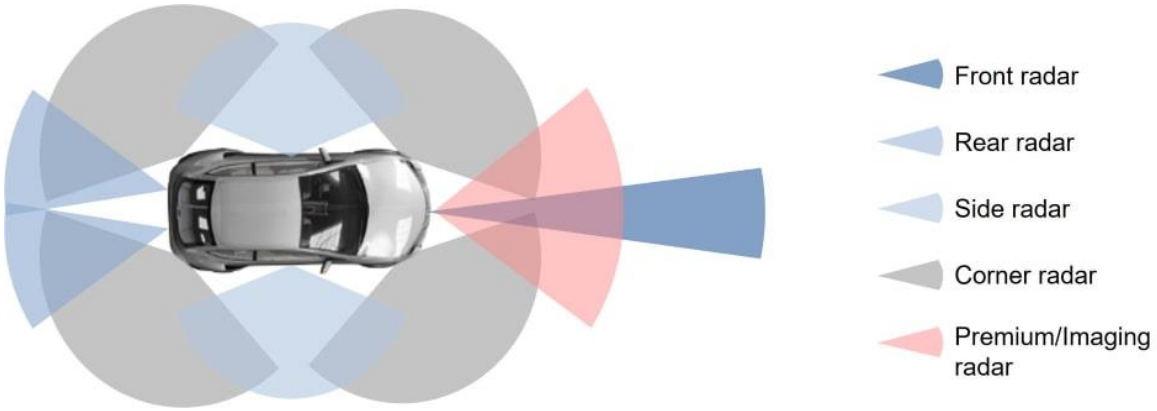


FONTE: Autoria própria (2022)

É possivel fornecer uma cobertura de 360°, diferentes sensores de radar com funções diferentes precisam ser colocados no carro. Os dados obtidos devem ser combinados para obter informações precisas em tempo real sobre o entorno.

A figura 11 mostra a distribuição dos radares em um veículo para cobertura de 360°.

Figura 11: Distribuição dos radares para cobertura de 360°.



FONTE: RENESAS (2022)

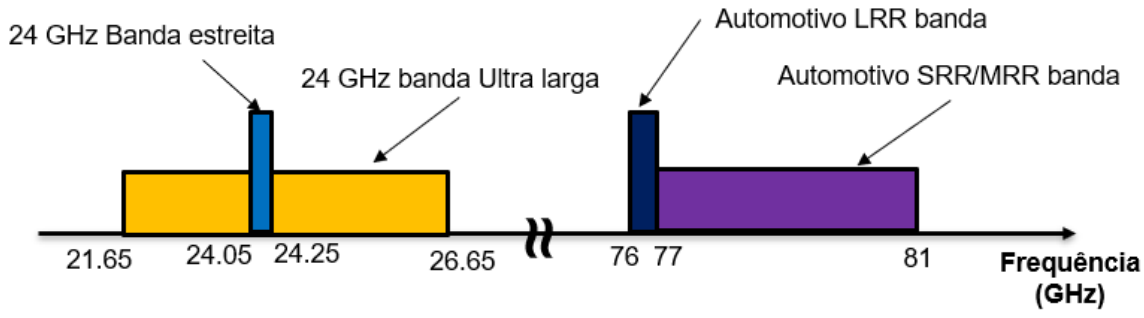
As possíveis aplicações da figura 11, são vistas em auxiliar de estacionamento de veículos, conhecido como Parking Assistance, outra aplicação possivel é o AEB frenagem de emergência, *Adaptive Cruise Control* (Sistema de controle de cruzeiro) que ajusta automaticamente a velocidade do veículo para se manter a uma distância segura do veículo a frente, *Forward/Rear Collision Warning*, que serve para prevenir colisões com um veículo mais

lento ou que está parado na estrada, e também podemos aplicar no *Lane change Assist,* que é o assistente de mudança de faixa, e por fim o alerta de tráfego traseiro. (RENESAS ELECTRONICS,2022).

Os sensores de radar podem ser facilmente instalados atrás de elementos comuns do carro, como pára-choques ou emblemas da empresa, para que fiquem invisíveis e não afetem a estética. Essa integração fica mais fácil com frequências de operação mais altas, pois o tamanho das antenas, que determina o tamanho do módulo, é linearmente proporcional ao comprimento de onda e, portanto, inversamente proporcional à frequência de operação. Havia quatro bandas de frequência principais usadas em sistemas de radar automotivo, duas na banda K (cerca de 24 GHz) e duas na banda E (entre 76 e 81 GHz) (RENESAS ELECTRONICS,2022).

A figura 12 mostra as frequências mais utilizadas pelos radares segundo a união internacional de telecomunições. As bandas de 24 GHz devem ser descontinuadas, devido à interferência em aplicações de radioastronomia e exploração da terra. Como alternativa, a faixa de frequência de 76 GHz a 81 GHz tem sido aceita pela maioria dos países como a faixa de frequência para radares automotivos. Lá, a largura de banda de 1 GHz é reservada para LRR (76 a 77 GHz), enquanto a largura de banda de 4 GHz está disponível para aplicativos que exigem melhor resolução.

Figura 12: Frequencia de banda dos radares.



FONTE: Autoria própria.

O quadro 2, apresenta as características típicas de radar automotivo na faixa de frequência 76-81 GHz, de acordo com a Recomendação da ITU-R M.2057-0.

Quadro 2: Caracteristiscas do radar.

Tipos de Radar (Classificação ITU)	UMA	B	C	D	E
	Radar automotivo para aplicações frontais	Radar automotivo de alta resolução para aplicações frontais	Radar automotivo de alta resolução de canto	Radar automotivo de alta resolução	Radar automotivo de alta resolução Aplicações de alcance muito curto
Exemplos de aplicações	ACC, prevenção de colisões	Monitoramento de pista	Detecção de ponto cego, assistente de mudança de faixa	Alerta de cruzamento de trafego traseiro	Auxiliar de estacionamento, detecção de pedestres, frenagem de emergência em baixa velocidade
Faixa típica	<= 250m	<= 100m	<= 100m	<= 100m	<= 50m
Resolução de alcance típica	75 cm	7,5 cm	7,5 cm	7,5 cm	7,5 cm
Faixa de frequência	76-77 GHz	77-81 GHz	77-81 GHz	77-81 GHz	77-81 GHz
Largura de banda máxima	1 GHz	4 GHz	4 GHz	4 GHz	4 GHz
Pire máximo	55dBm	33dBm	33dBm	45dBm	33dBm

FONTE: Autoria própria (2022)

As equações que regem o comportamento das ondas eletromagnéticas são as 4 equações de Maxwell, que podem ser consultadas no livro Fundamentos de Física – Eletromagnetismo, dos autores HALLIDAY & RESNICK, WALKER J, e estão escritas a seguir:

Lei de Gauss: $$\emptyset_e = \oint_s \vec{E}.\hat{n}dA = \frac{Q}{\varepsilon_0} \quad (2)$$

Lei de Gauss para o magnetismo: $$\oint_s \vec{B}.\hat{n}dA = \frac{Q}{\varepsilon_0} \quad (2.1)$$

Lei de Faraday: $$\oint_c \vec{E}.dl = -\frac{d}{dt}\int_S \vec{B}.\hat{n}dA \quad (2.2)$$

Lei de Ampere: $$\oint_c \vec{B}.dl = \mu_0 I + \mu_0 \varepsilon_0 \frac{d}{dt}\int_S \vec{E}.\hat{n}dA \quad (2.3)$$

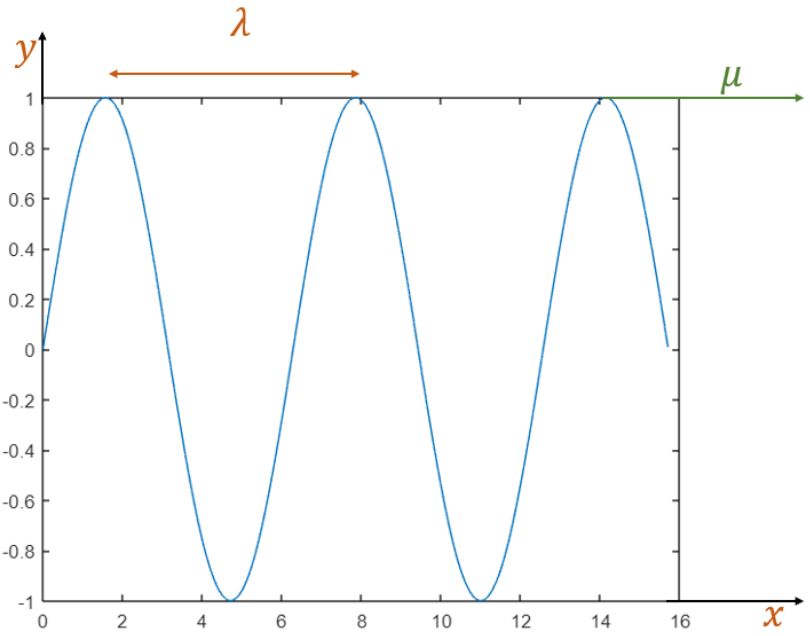
As equações de Maxwell tem relação com as ondas eletromagnéticas, sabemos que a equação de uma onda em uma corda é:

Equação da onda $$\frac{\partial^2 y(x,t)}{\partial x^2} = \frac{1}{\partial} \frac{\partial^2 y(x,t)}{\partial t^2} \quad (3)$$

Onde $y(x,t)$, é a posição de pontos da corda no instante t.

A figura 13 apresenta uma onda eletrogmanética com seus parâmetros.

Figura 13: Parâmetros de uma onda eletromagnética.



Fonte: Autoria própria (2022)

Onde:

$\lambda -$ Comprimento de onda.

$\mu -$ densidade linear de massa.

$v = \sqrt{\frac{T}{\mu}} =$ velocidade da onda

$T =$ Tensão da corda

$K = \frac{2\pi}{\lambda} =$ número de onda.

As ondas eletromagnéticas se propagam na velocidade da luz (c), $c = 3 \times 10^8 m/s$, tanto em meios materiais quanto no vácuo, podem sofrer reflexão, refração, absorção, difração, interferência, espalhamento e polarização. (HALLIDAY, RESNICK, WALKER, 2016).

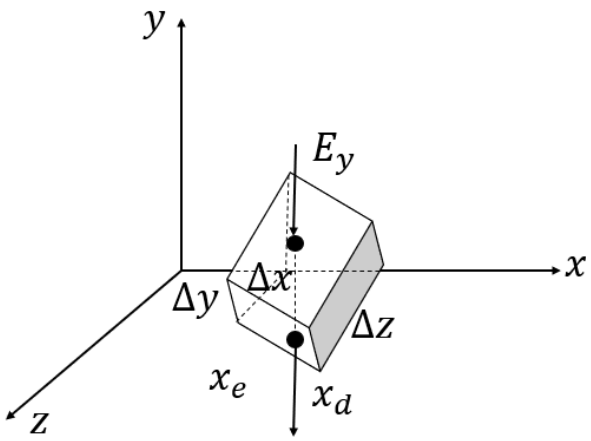
A equação (4) a seguir mostra a solução da equação da onda (3):

Solução da equação da onda: $y = y_0 \sin(kx - \omega t)\,,\ com\ \omega t = 2\pi f$ (4)

Agora será mostrado como as equações de Maxwell acarretam uma equação de onda, desta forma vamos considerar os seguintes cenários, análise é no vácuo (sem correntes e sem cargas elétrica). Sendo E e B são função do tempo e uma coordenada: X (Ondas planas),

Desta forma vamos considerar um elemento de volume no vácuo representado na figura 14.

Figura 14: Objeto de volume no vácuo.



Fonte: Autoria própria (2022)

O cálculo do fluxo elétrico através dos elementos de área:

- $Área$ ΔxΔy: Campo E_z não depende de Z, logo **:** $\emptyset_e = 0$
- $Área$ ΔxΔz: Campo EY não depende de Y, logo **:** $\emptyset_e = 0$
- $Área$ ΔyΔz: Campo EX não depende de X.

$$\emptyset_e = (E_{xd} - E_{xe})\Delta y.\Delta z = 0 \quad (5)$$

Pois não temos cargas internas.

$E_{xd} = E_{xe}$, Logo EX não depende de x. (6)

Campo elétrico que varia no espaço deve ser perpendicular a direção de propagação. Vamos assumir que o campo elétrico em $y(E_y)$ varia com x.

Partindo das análises acima para as outras 3 equações de Maxwell, obtemos as seguintes conclusões:

Equação da onda para EY: $$\frac{\partial^2 E_y}{\partial x^2} = \mu_0 \varepsilon_0 \frac{\partial^2 E_y}{\partial t^2} \quad (7)$$

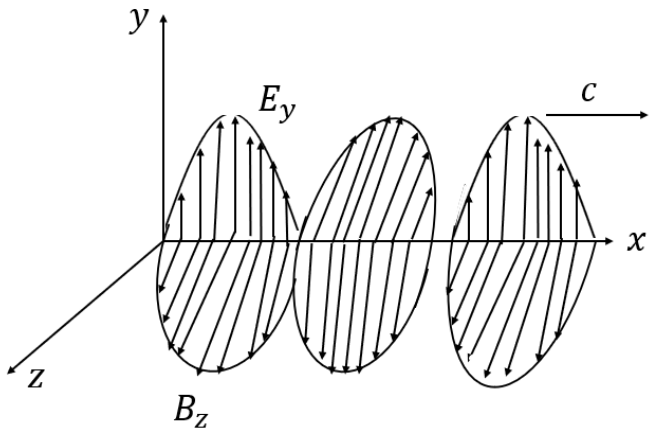
Equação da onda para BZ: $$\frac{\partial^2 B_z}{\partial x^2} = \mu_0 \varepsilon_0 \frac{\partial^2 B_z}{\partial t^2} \quad (8)$$

Com a velocidade de propagação sendo igual a velocidade da luz. Desta forma temos que as equações de Maxwell geraram as equações para as ondas eletromagnéticas no vácuo.

- $\vec{E}$ *e* $\vec{B}$ são perpendiculares entre si, e a direção de propagação estão em fase.
- Propagam-se na velocidade da luz (c), e a direção de propagação é $\vec{E} \times \vec{B}$

A figura 15 mostra o gráfico e as considerações que foram feitas a partir da nossa análise acima.

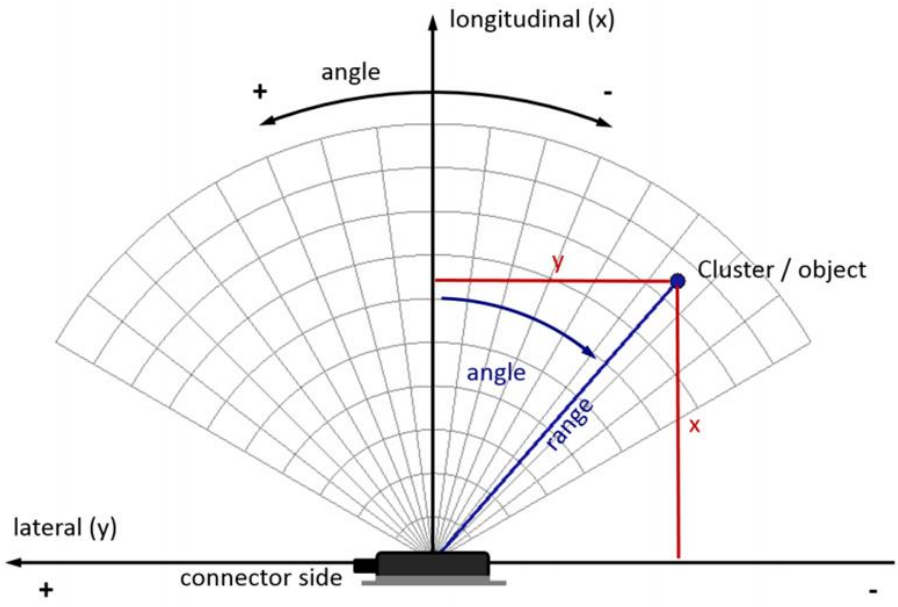
Figura 15: Gráfico das ondas eletromagneticas.



Fonte: Autoria própria (2022)

A figura 16 mostra sistema de coordenadas para o cluster e objetos.

Figura 16: Sistema de coordenadas cluster/objetos.

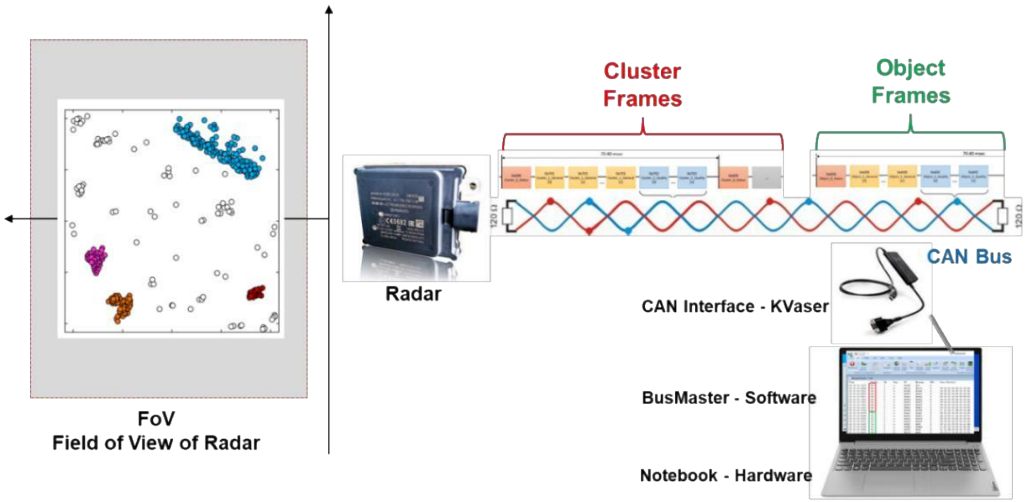


Fonte: Technical Documentation - ARS 404-21 (Entry) ARS 408-21 (Premium) (2020)

O radar operacional é capaz de identificar os pontos de nuvens em seu campo de visão, onde existe um algoritmo IP que é capaz de definir clusters e identificar objetos. Essas informações estão disponíveis no barramento CAN.

A figura 17 mostra como acontece esse processo.

Figura 17: Processo da captura de nuvens de pontos e algoritmo IP.



Fonte: GSA-UTFPR-PG (2022)

Os radares comerciais nos dão apenas informações do Cluster e Objetos. Não conseguimos ter acesso ao algoritmo de Clusterização porque é um IP das empresas. IP (*Intellectual Property*), algo confidencial, sendo assim precisaríamos de ter um radar de Engenharia para fazer implementação de algoritmos de cluster.

O grupo GSA da UTFPR-PG, possui apenas um radar da Continental/VALEO da DAF que já é de produção, não é instrumentado e não conseguimos fazer muita coisa.

Porém foi comprado um radar da Continental na mesma condição. Que pelo menos possuímos o DBC. Desta forma só nos sobra o MATLAB que conseguimos implementar e fazer os testes algoritmos de clusterização.

2.2 POINT CLOUDS

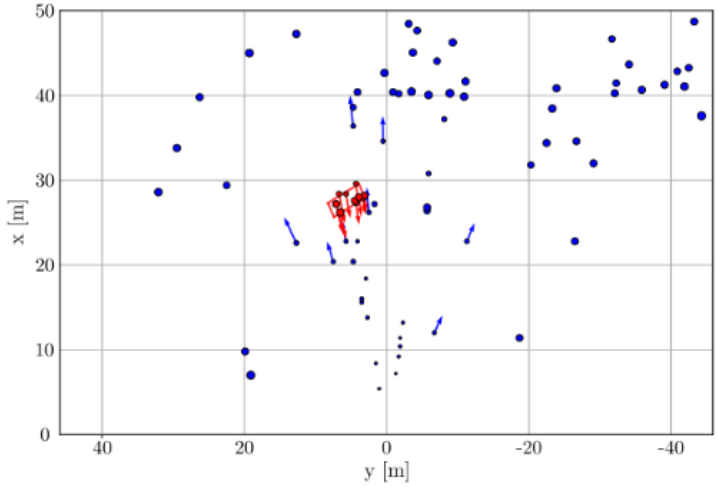
Os points clouds, nada mais são que a nuvem de pontos geradas pelo radar, encontramos uma definição matemática no artigo $2D$ *Car Detection in Radar Data with PointNets*, dos autores Andreas Danzer, Thomas Griebel, Martin Bach, and Klaus Dietmayer, onde os Point clouds são representadas como um conjunto de quatro dimensões.

$$P = \{p_i | \; i = 1, 2, \dots, n\}, \qquad n \in \mathbb{N}$$

Sendo n o número de alvos do radar, e cada ponto $p_i = (x, y, vr, \sigma)$, sendo x, y coordenadas, vr a velocidade doppler compensada por movimento, e σ radar cruzado por valores de secção (DANZER A, et. Al, 2019).

A figura 18 mostra a captura da nuvem de pontos realizada por um radar.

Figura 18: Nuvem de pontos captadas pelo radar

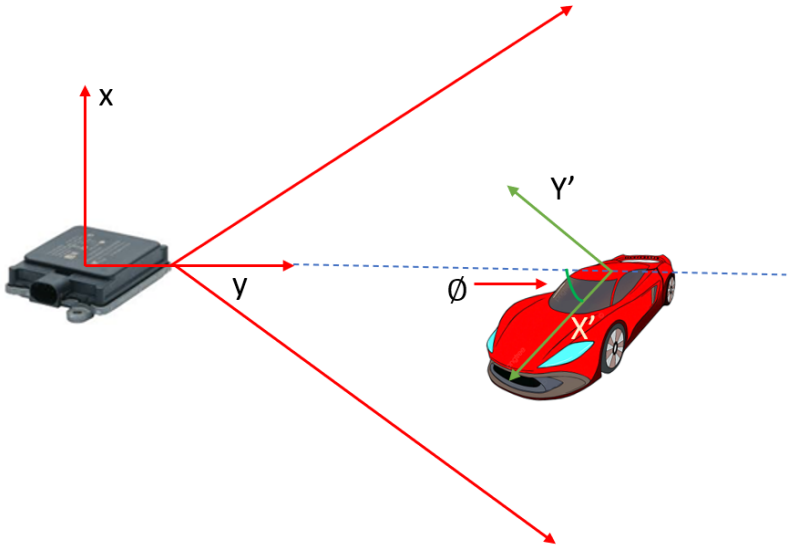


Fonte: DANZER, et al (2019)

Para o reconhecimento dos objetos é importante que o radar esteja posicionado de forma que ele consiga abranger a maior parte da nuvem de pontos que o objeto está formando, assim é levado em conta o ângulo de visão. (DANZER, et al. 2019).

A figura 19 mostra como o ângulo de visão que é definido como o angulo entre o eixo y do sistema de coordenadas do radar e o eixo X' do sistema de coordenada do objeto.

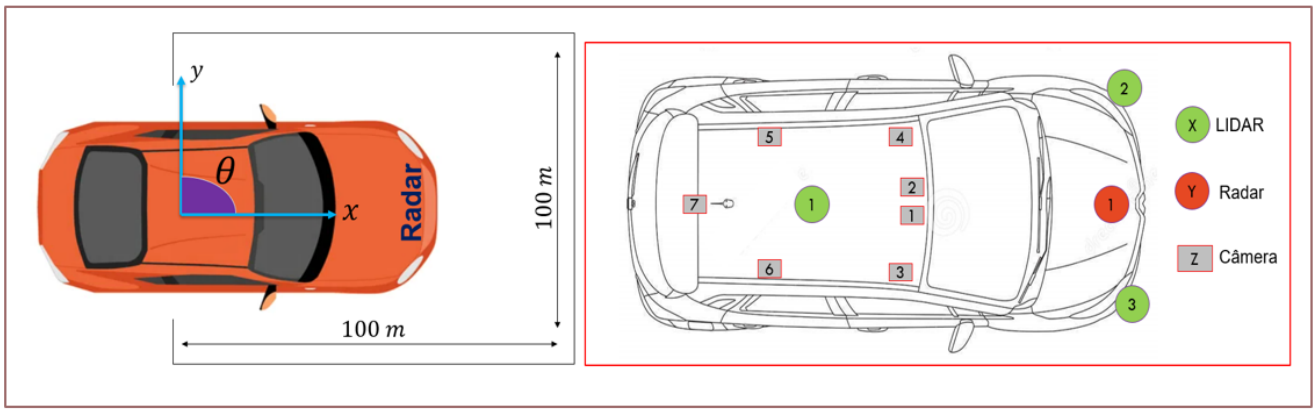
Figura 19: Ângulo de visão



Fonte: Autoria própria (2022)

A figura 20 mostra o alcance do radar em coordenadas x e y, e a configuração da distribuição dos outros sensores no veículo autônomo, o radar geralmente é instalado na parte frontal do veículo para ter um maior alcance e para gerar a nuvem de pontos.

Figura 20: Alcance do radar e distribuição dos sensores.

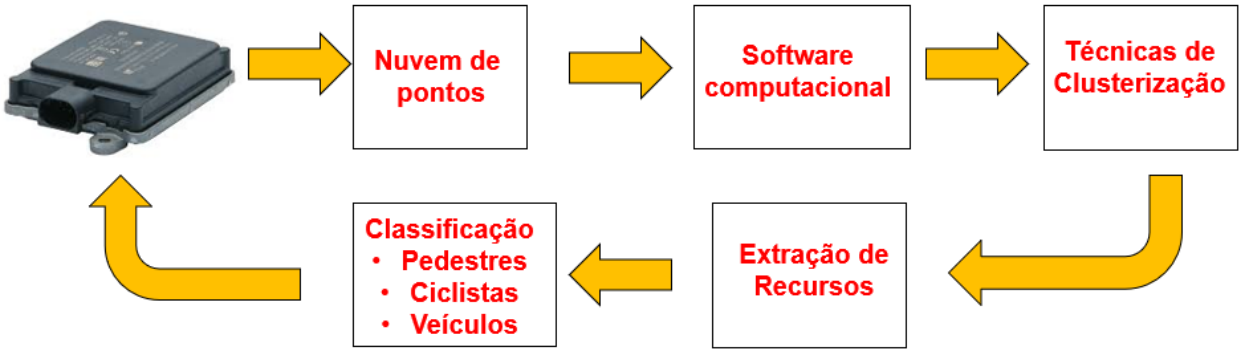


Fonte: Autoria própria (2022)

2.2.1 Aplicação em Radar Automotivo para Reconhecimento de Objetos

Como mencionado, o objetivo é utilizar a nuvem de pontos geradas pelo sistema de radar, para aplicar os algortimos de clusterização e fazer o reconhecimento de objetos. Na figura 21 mostra o diagrama de como é feito a utilização da teoria de clusterizaçãopara aplicação em um sistema de radar automotivo, na aplicação em um veículo autônomo, quando o veículo estiver em movimento o ciclo abaixo vai se repetir e a todo o momento o radar fará a leitura do ambiente, para que o algoritmo faça o reconhecimento dos objetos ao seu redor.

Figura 21:Diagrama da aplicação da clusterização em radar.

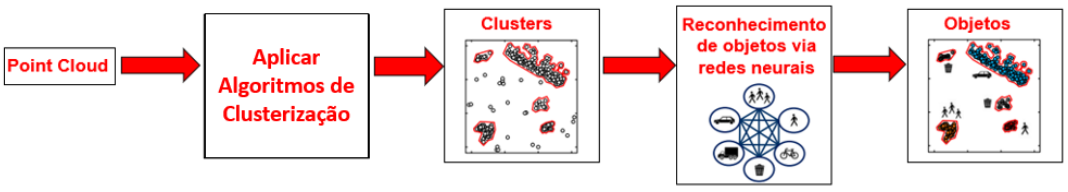


Fonte: Autoria própria (2022)

O bloco que se refere a extração de recursos é a utilização de vários métodos de reconhecimento de objetos, como por exemplo, YOLOv3, YOLOv4 e YOLOv5 são ótimas redes de treinamento quando o objeto está em movimento.

A figura 22 descreve um diagrama com uma situação que poderia ser aplicado em uma situação real, com os *Point Cloud* gerados pelo radar, poderiamos aplicar algum algoritmo de clusterização, e posteriormente aplicar uma outra tecnica de reconhecimento de objetos, via redes neurais, desta forma, o resultado final seria os objetos identificados, podendo ser eles, pedestre, veiculos, ciclistas, caminhões.

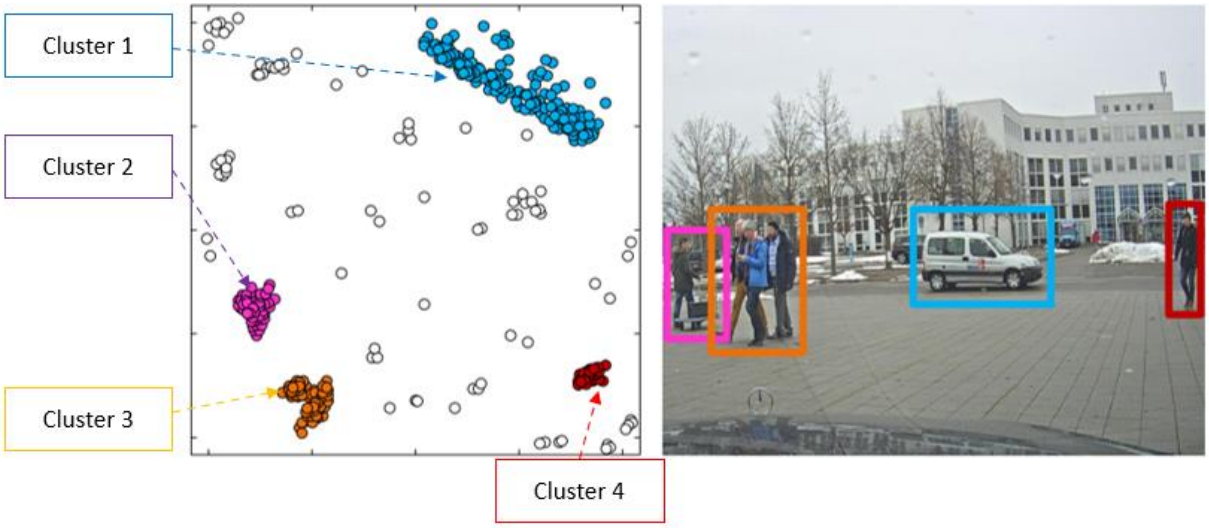
Figura 22: Processo para aplicação da clusterização em sistema de radar.



Fonte: Autoria própria (2022)

Na figura 23 mostra como fica o resultado final da aplicação, é uma imagem retirada do artigo A *Multi-Stage* Clustering Framework for Automotive Radar Data dos autores Nicolas Scheiner, Nils Appenrodt1, Jurgen Dickmann, e Bernhard Sick.

Figura 23: Resultado da aplicação da clusterização em sistema de radar.



Fonte: SCHEINER, et al. (2021)

Com a clusterização é possível identificar e reconhecer objetos e junto com a câmera obtem-se uma fusão de sensores mais robusta com um melhor desempenho, com exatidão e precisão para atender aos funções ADAS – Advanced Driver Assistance System e AD – Autonomous Driving.

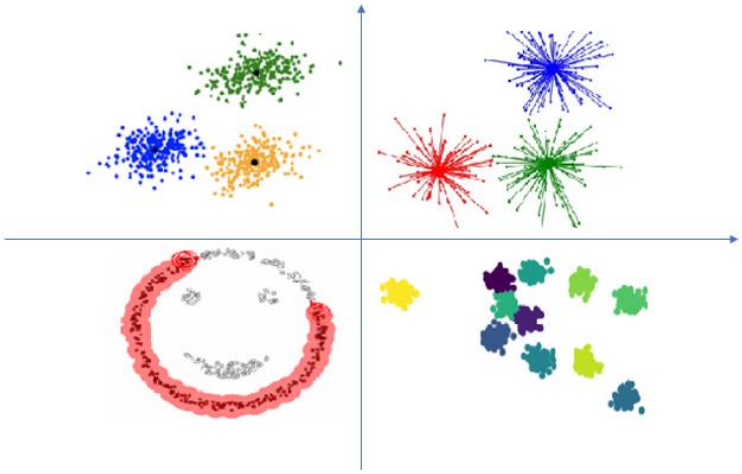
3 FUNDAMENTOS DE CLUSTERIZAÇÃO

A teoria dos conjuntos é uma entidade matemática de agrupamento de elementos com uma característica específica, e a clusterização tem os mesmos princípios da teoria dos conjuntos, a palavra clusterização do inglês “cluster”, nada mais é do que agrupar, esse agrupamento pode ser de um banco de dados, de clientes de uma loja, de produtos de uma fábrica ou do que for necessário agrupar, o termo tem relação ao Big Data, que é o trabalho com um grande conjunto de dados, na clusterização são tomados como base princípios estatísticos de distribuição de amostras e de resultados, dentre eles vale destacar quatro princípios, o modelo de distribuição, modelos de conectividade, modelos centroides e modelos de densidade.(BOUVEYRON, et. al, 2019)

No modelo de distribuição, os agrupamentos são formados considerando a probabilidade de que todos os pontos de dados em um cluster sejam da mesma distribuição, seja ela gaussiana ou normal, o modelo de conectividade forma os agrupamentos dos objetos do cluster tendo em vista a regra da proximidade, no modelo centroides, a similaridade entre dados é considerada a partir da proximidade de um ponto de dados ao chamado centroide dos clusters, são modelos executados iterativamente, ou seja, em uma espécie de *“looping”* a fim de encontrar o melhor local para agrupar dados, o modelo de densidade, o que vale é a densidade dos dados agrupados em um gráfico hipotético.

A figura 24 apresenta os diferentes tipos de distribuição que podemos encontrar durante a aplicação do método de clusterização, os métodos são *K-means, Affinity, BIRCH* e *DBSCAN*, cada método usa um tipo diferente de distribuição estatística, o método *DBSCAN* por exemplo usa o princípio da densidade.

Figura 24: Tipos de distribuição/Clusterização



Fonte: Autoria própria (2022)

A Clusterização é uma técnica de *Machine Learning* que envolve o agrupamento de pontos de dados, e funciona da seguinte forma, dado um conjunto de pontos de dados, pode se usar um algoritmo de agrupamento para classificar cada ponto de dado em um grupo específico. Os pontos de dados que estão no mesmo grupo devem ter propriedades e recursos semelhantes, enquanto os pontos de dados em grupos diferentes devem ter propriedades e ou recursos altamente diferentes.

Uma definição de Clusterização é dada em HRUSCHKA & EBECKEN (2001). Considerando um conjunto de n objetos $X = \{X_1, X_2, \dots, X_n\}$ onde cada $X_1 \in \mathbb{R}^p$ é um vetor de p medidas reais que dimensionam as características do objeto, estes devem ser clusterizados em *k cluster* disjuntos $C = \{C_1, C_2, \dots, C_k\}$ de forma que tenhamos as seguintes restrições:

1. $C_1 \cup C_2 \cup \dots \cup C_k = X$;
2. $C_i \neq \emptyset, \forall_i, 1 \leq i \leq k$;
3. $C_i \cap C_j = \emptyset, \forall_i \neq j, 1 \leq i \leq k, 1 \leq j \leq k$.

É importante ressaltar por essas condições que um objeto não pode pertencer a mais de um cluster, e que cada cluster tem que ter pelo menos um objeto. COLE (1998) ainda acrescenta que o valor de *k* geralmente é desconhecido. Se *k* é conhecido, o problema é referido como um problema de k-Clusterização.

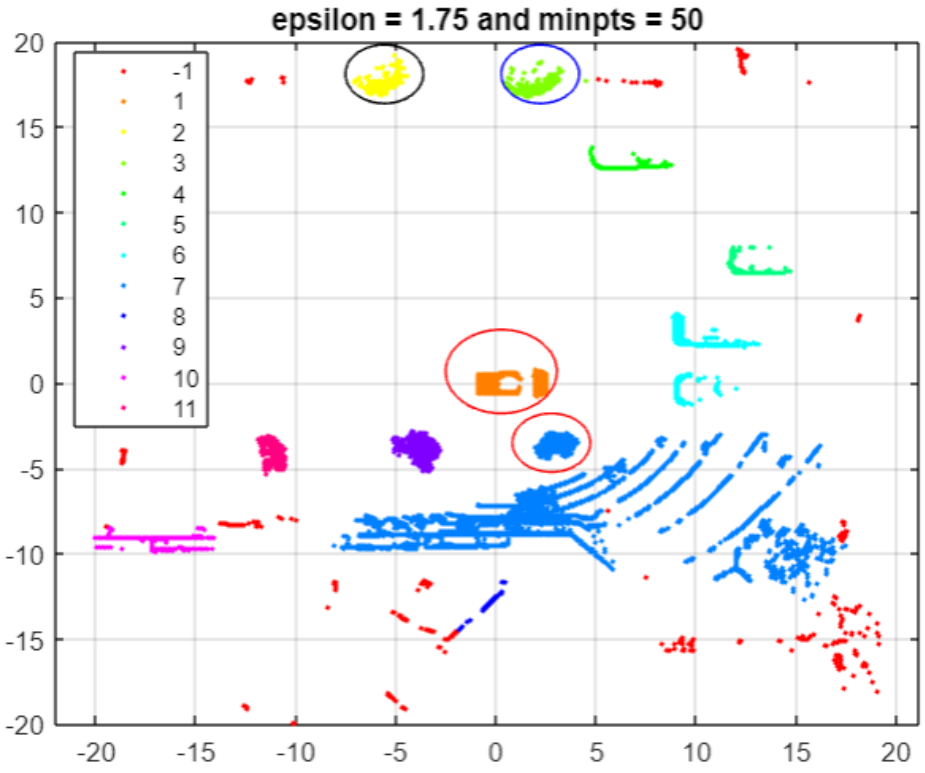
Encontrar o melhor agrupamento para um conjunto de objetos nem sempre é tão trivial. HRUSCHKA & EBECKEN (2001) destacam que o problema o melhor agrupamento é NP-completo e não é computacionalmente possível encontrá-lo, a não ser que *n* (número de objetos) e k *(*números de cluster) sejam extremamente pequenos, visto que o número de partições distintas em que podemos dividir *n* objetos em *k* clusters aumenta aproximadamente como $\frac{k^n}{n!}$.

ANKERST et al. (1999) descrevem que existem três razões interconectadas para explicar por que a efetividade dos algoritmos de Clusterização não é tão simples. Primeiro, quase todos os algoritmos de Clusterização requerem valores para os parâmetros de entrada que são difíceis de determinar, especialmente para conjuntos de dados do mundo real contendo objetos com muitos atributos. Segundo os algoritmos são muito sensíveis a estes valores de parâmetros, frequentemente produzindo partições muito diferentes do conjunto de dados mesmo para ajustes de parâmetros significativamente pouco diferentes. Terceiro, os conjuntos

de dados reais de alta dimensão têm uma distribuição muito ampla que não pode ser revelada por um algoritmo de Clusterização usando somente um ajuste de parâmetro global.

A figura 25 apresenta um exemplo de clusterização, aplicada sobre uma nuvem de pontos gerada por um lidar, no canto superior esquerdo podemos ver a legenda indicando os clusters que foram gerados.

Figura 25: Exemplo de Clusterização



Fonte: Autoria própria.

3.1 ALGORITMOS DE CLUSTERIZAÇÃO

Dez algoritmos são bastante utilizados para diferentes tipos de aplicações, por cientistas de dados, são eles, o *Affinity Propagation*, *Agglomerative Clustering*, *BIRCH*, *DBSCAN*, *K-Means*, *Mini-Batch K-Means*, *Mean Shift*, *OPTICS*, *Spectral Clustering*, *Mixture of Gaussians*, cada algoritmo tem características especificas e devem ser utilizados de acordo com a necessidade do agrupamento a ser realizado.

3.1.1 *Affinity Propagation*

O artigo publicado por J. Frey* and Delbert Dueckem 2007, *Clustering by Passing Messages Between Data Points Brendan*. Explica que o algoritmo *Affinity Propagation,* propagação de afinidade em português, é um algoritmo de baixo erro, alta velocidade, flexível, uma das vantagens desse algoritmo é que ele toma como entrada medidas de similaridade entre pares de pontos de dados e vai simultaneamente considerando todos os pontos de dados como exemplos potenciais. As mensagens de valor real são trocadas entre os pontos de dados até que um conjunto de exemplares de alta qualidade e os clusters correspondentes aparecem gradualmente (FREY J, DUECHEM D, 2007).

- **Semelhanças entre pontos de dados:** representa o quão adequado um ponto é para ser o exemplar de outro. Se não houver semelhança entre dois pontos, eles não podem pertencer ao mesmo cluster, essa semelhança pode ser omitida ou definida como *Infinity* dependendo da implementação.

- **Preferências**: Representa a adequação de cada ponto de dado para ser um exemplar. Pode se ter algumas informações a priori de quais pontos poderiam ser favorecidos para essa função, e assim podemos representá-los por meio de preferências.

Tanto as semelhanças quanto as preferências são frequentemente representadas por meio de uma única matriz, onde os valores na diagonal principal representam as preferências. A representação matricial é boa para conjuntos de dados densos. Onde as conexões entre os pontos são esparsas, é mais prático não armazenar toda a matriz $n\ x\ n$ na memória, mas manter uma lista de semelhanças com os pontos conectados.

O algoritmo então passa por várias iterações, até convergir. Cada iteração tem duas etapas de passagem de mensagens:

Cálculo de responsabilidades: A responsabilidade $r(i,k)$ reflete a evidência acumulada de como o ponto k é adequado para servir como exemplo para o ponto i, levando em consideração outros exemplos potenciais para o ponto i. A responsabilidade é enviada do ponto de dados i para o ponto de exemplo candidato k.

Cálculo de disponibilidades: A disponibilidade $a(i,k)$ reflete a evidência acumulada de quão apropriado seria para o ponto i escolher o ponto k como seu exemplar, levando em consideração o suporte de outros pontos de que o ponto k deveria ser um exemplar. A disponibilidade é enviada do ponto de exemplar candidato k para o ponto i.

Para calcular as responsabilidades, o algoritmo usa semelhanças e disponibilidades originais calculadas na iteração anterior, na primeira iteração eles são considerados iguais a zero.

As responsabilidades são definidas para a similaridade de entrada entre o ponto i e o ponto k como seu exemplar, menos o maior da soma de similaridade e disponibilidade entre o ponto i e outros exemplares candidatos. A lógica por trás do cálculo de quão adequado é um ponto para um exemplar é que ele é mais favorecido se a preferência inicial a priori for maior, mas a responsabilidade diminui quando há um ponto semelhante que se considera um bom candidato, então há uma 'competição' entre os dois até que um seja decidido em alguma iteração. (FREY J, DUECHEM D, 2007).

O cálculo de disponibilidades, então, usa responsabilidades calculadas como evidência de que cada candidato seria um bom exemplo. A disponibilidade $a(i,k)$ é definida como a autorresponsabilidade $r(k,k)$ mais a soma das responsabilidades positivas que o candidato exemplar k recebe de outros pontos.

Finalmente, podemos ter diferentes critérios de parada para encerrar o procedimento, como quando as mudanças nos valores caem abaixo de algum limite ou o número máximo de iterações é atingido. Em qualquer ponto através do procedimento de propagação de afinidade, a soma das matrizes Responsabilidade (r) e Disponibilidade (a) nos dá a informação de agrupamento que precisamos: para o ponto i, o k com máximo $r(i,k) + a(i,k)$ representa o ponto exemplar. Ou, se precisarmos apenas do conjunto de exemplares, podemos escanear a

diagonal principal. Se $r(i,i) + a(i,i) > 0$, o ponto i são um exemplar. (FREY J, DUECHEM D, 2007).

Com o *Affinity Propagation*, não é preciso especificá-lo explicitamente, mas ainda pode precisar de alguns ajustes se obtivermos mais ou menos clusters do que podemos encontrar. Desta forma, apenas ajustando as preferências podemos diminuir ou aumentar o número de clusters.

Definir preferências para um valor mais alto levará a mais clusters, pois cada ponto é "mais certo" de sua adequação para ser um exemplo e, portanto, é mais difícil de "bater" e incluí-lo sob a "dominação" de algum outro ponto. Como regra geral, podemos definir todas as preferências para a similaridade mediana para um número médio a grande de clusters, ou para a similaridade mais baixa para um número moderado de clusters. No entanto, algumas execuções com preferências de ajuste podem ser necessárias para obter o resultado que atenda exatamente às nossas necessidades. (FREY J, DUECHEM D, 2007).

O algoritmo tem como base as seguintes equações:

$$a(k,k) \leftarrow \sum_{i' tal\ que\ i' \neq k} \max\{0, \mathrm{r(i', k)}\} \tag{9}$$

Onde a autodisponibilidade $a(k,k)$ é definida a soma das responsabilidades positivas que o candidato exemplar k recebe de outros pontos.

$$r(i,k) \leftarrow s(i,k) - \max_{k' tal\ que\ k' \neq k} \{a(i,k') + s(i,k')\} \tag{9.1}$$

Onde a responsabilidade $r(i,k)$ reflete a evidência acumulada $s(i,k)$, de como o ponto k é adequado para servir como exemplo para o ponto i.

$$a(i,k) \leftarrow \min\{0, \mathrm{r(k,k)} + \sum_{i' tal\ que\ i' \notin \{i,k\}} \max\{0, \mathrm{r(i', k)}\} \tag{9.2}$$

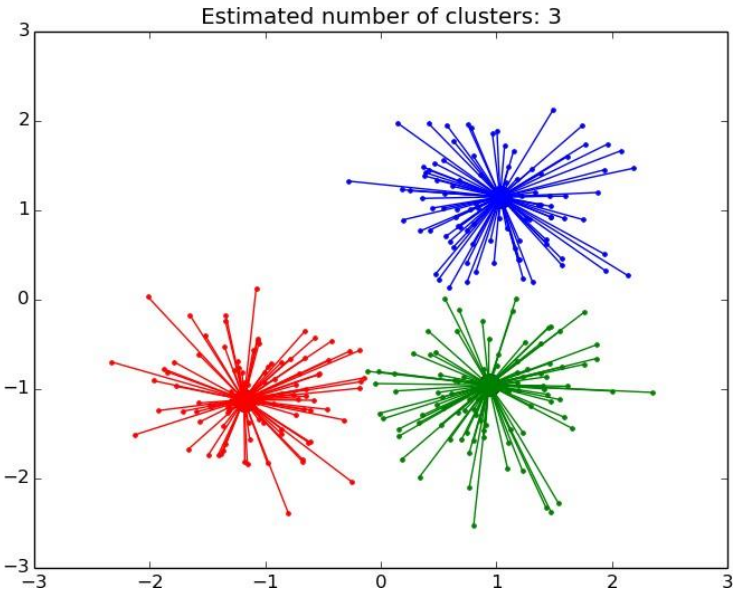
Onde a disponibilidade $a(i,k)$ é definida como a autorresponsabilidade $r(k,k)$ mais a soma das responsabilidades positivas que o candidato exemplar k recebe de outros pontos.

$$c(i,k) \leftarrow r(i,k) + a(i,k). \tag{9.3}$$

Onde $c(i,k)$ é definido como a soma das matrizes Responsabilidade (r) e Disponibilidade (a), e é o ponto exemplar.

Veja na figura 26 um exemplo da aplicação do algoritmo *Affinity Propagation,* podemos ver três grupos que foram separados após a aplicação dos cálculos utilizando as equações citadas acima.

Figura 26: Exemplo do algoritmo *Affinity Propagation*



Fonte: LEARN (2015)

3.1.2 *Hierarchical clustering*

De acordo com Friedman, Jerome (2009). *"14.3.12 Hierarchical clustering". The Elements of Statistical Learning.*

Os algoritmos de agrupamento hierárquico se dividem em 2 categorias: de cima para baixo e de baixo para cima. Os algoritmos ascendentes tratam cada ponto de dados como um único cluster no início e, em seguida, aglomeram pares de clusters sucessivamente até que todos os clusters tenham sido mesclados em um único cluster que contém todos os pontos de dados (FRIEDMAN, JEROME, 2009).

Para entender o algoritmo vamos ver as seguintes definições de Métricas de distância:

- Distância euclidiana é a distância mais curta entre dois pontos em qualquer

dimensão.

$$\|a - b\|_2 = \sqrt{\sum_i (a_i - b_i)^2} \quad (10)$$

Onde a_i representa a característica do indivíduo a, b_i representa a característica do indivíduo b, e i é o número de parcelas da amostra.

- Distância euclidiana quadrada

$$\|a - b\|_2^2 = \sqrt{\sum_i (a_i - b_i)^2} \quad (11)$$

Onde a_i representa a característica do indivíduo a, b_i representa a característica do indivíduo b, e i é o número de parcelas da amostra.

- Distância manhattan, define que a distância entre dois pontos é a soma das diferenças absolutas de suas coordenadas.

$$\|a - b\|_1 = \sum_i |a_i - b_i| \quad (12)$$

Onde a_i representa a característica do indivíduo a, b_i representa a característica do indivíduo b, e i é o número de parcelas da amostra.

- Distância máxima é a distância máxima de um ponto ao outro.

$$\|a - b\|_\infty = max_i |a_i - b_i| \quad (13)$$

Onde a_i representa a característica do indivíduo a, b_i representa a característica do indivíduo b, e i é o número de parcelas da amostra.

A escolha para o cálculo da distância depende da linguagem de programação que será utilizada. Depois da escolha do método a ser utilizado para o cálculo da distância, é importante ver os critérios de ligação, temos três tipos, a ligação simples, completa e média.

A ligação simples é calculada pela distância mínima absoluta entre os pontos, a completa é dada pela distância máxima absoluta, e pôr fim a ligação média é calculada em cima da média da distância de todos os pontos.

Veja abaixo as equações dos critérios de ligação simples, completa e média respectivamente.

$$L(r,s) = \min(D(x_{ri}, x_{si})) \tag{14}$$

A equação (14) representa o critério de ligação simples.

$$L(r,s) = \max(D(x_{ri}, x_{si})) \tag{14.1}$$

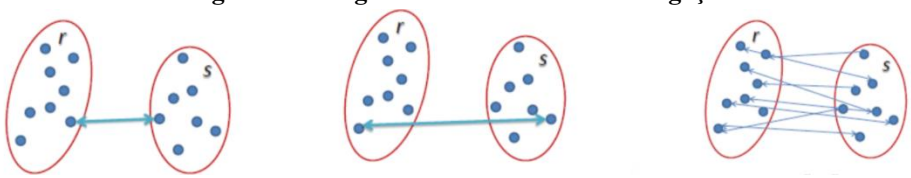
A equação (14.1) representa o critério de ligação completa.

$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{si}) \tag{14.2}$$

A equação (14.2) representa o critério de ligação média.

A figura 27, mostra o diagrama com os critérios de ligação, simples, completa e média respectivamente.

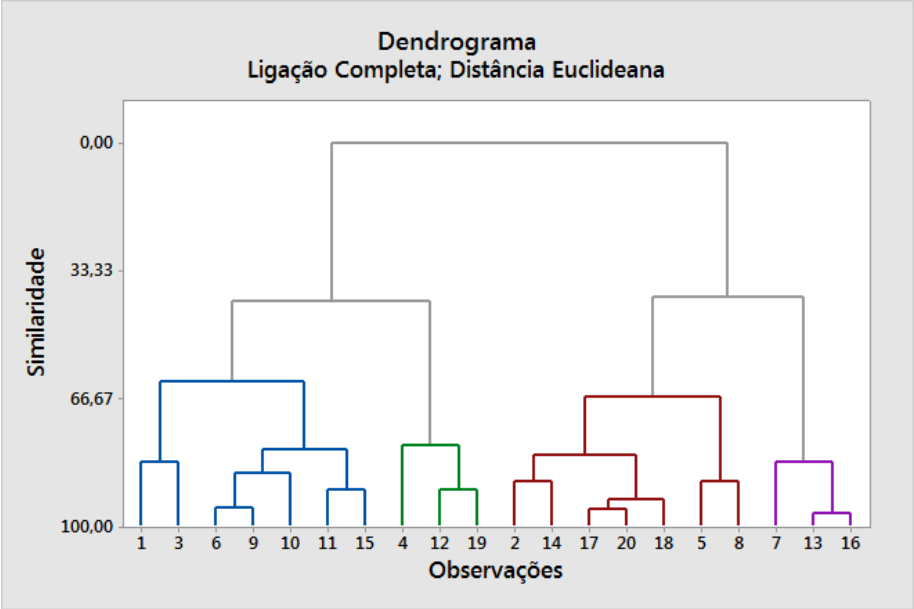
Figura 27: Diagrama com os critérios de ligação



Fonte: Autoria própria (2022)

Depois que tudo isso realizado, temos que criar um dendrograma, que é um gráfico que nos ajuda a identificar qual o número de agrupamentos que podemos criar, apresentado na figura 28.

Figura 28 – Dendograma



Fonte: MINITAB (2022)

Analisando o dendrograma, podemos ter uma noção mais precisa sobre a quantidade de *clusters* que podemos formar.

O agrupamento top-down interpreta todos os dados como um único cluster e vai os dividindo de forma recursiva sempre tendo em vista a métrica de distância e o critério de ligação escolhidos. O agrupamento aglomerativo é diferente do top-down pois ele forma clusters com o mínimo de distância interna possível, ele começa com a construção de clusters individuais e vai os agrupando de forma a ter os pontos mais próximos, tendo em vista a métrica de distância e o critério de ligação escolhidos (FRIEDMAN, JEROME, 2009)

O agrupamento hierárquico de baixo para cima é, portanto, chamado agrupamento aglomerativo hierárquico ou HAC. Essa hierarquia de clusters é representada como uma árvore (ou dendrograma). A raiz da árvore é o único cluster que reúne todas as amostras, sendo as folhas os clusters com apenas uma amostra.

No início cada ponto dos dados é tratado como um único cluster, ou seja, se houver X pontos de dados em nosso conjunto de dados, teremos X clusters. Em seguida, selecionamos uma métrica de distância que mede a distância entre dois clusters. Como exemplo, usaremos a ligação média que define a distância entre dois clusters como a distância média entre os pontos de dados no primeiro cluster e os pontos de dados no segundo cluster. (FRIEDMAN, JEROME, 2009)

Em cada iteração, combinamos dois clusters em um. Os dois clusters a serem combinados são selecionados como aqueles com a menor ligação média. Ou seja, de acordo com nossa métrica de distância selecionada, esses dois clusters têm a menor distância entre si e, portanto, são os mais semelhantes e devem ser combinados.

O passo 2 é repetido até chegarmos à raiz da árvore, ou seja, temos apenas um cluster que contém todos os pontos de dados. Desta forma, podemos selecionar quantos clusters queremos no final, simplesmente escolhendo quando parar de combinar os clusters, ou seja, quando paramos de construir a árvore.

O agrupamento hierárquico não exige que especifiquemos o número de clusters e podemos até selecionar qual número de clusters parece melhor, pois estamos construindo uma árvore. Além disso, o algoritmo não é sensível à escolha da métrica de distância; todos eles tendem a funcionar igualmente bem, enquanto com outros algoritmos de agrupamento, a escolha da métrica de distância é crítica. Um caso de uso particularmente bom de métodos de agrupamento hierárquico é quando os dados subjacentes têm uma estrutura hierárquica e você deseja recuperar a hierarquia; outros algoritmos de agrupamento não podem fazer isso. Essas vantagens do agrupamento hierárquico vêm ao custo de menor eficiência, pois possui uma complexidade de tempo de $O(n^3)$, diferentemente da complexidade linear de *K-Means e GMM*. (FRIEDMAN, JEROME, 2009).

3.1.3 BIRCH

De acordo com o artigo *An Efficient Data Clustering Method for Very Large Databases*, dos autores Tian Zhang, Radgu Ramakrishnan, Miron Livny. SIGMOD, disponível na Conferência do ACM Special Interest Group on Management of Data, de 1996, BIRCH é um algoritmo de clusterização com uma metodologia de agrupamento hierárquico, é projetado para trabalhar com grandes conjuntos de dados númericos, ele utiliza a ideia de *clustering features* (CF).

O CF é um vetor de três dimensões que possui informações sobre os objetos de um determinado grupo (RAMAHRISHNAN et. Al, 1996), sendo definido por:

$$CF =< n, LS, SS \tag{15}$$

Onde *n* é o número de pontos de um *cluster*, *LS* é a soma dos *n* pontos, e *SS* é a soma

quadrada dos *n* pontos.

Os *CF* possui as informações necessárias para a tomada de decisão do algoritmo BIRCH e são tiradas de uma leitura do conjunto de dados. Através das CFs é montado uma CF tree inicial, uma árvore que guarda os CFs para um agrupamento hierárquico, que pode ser entendido como uma compressão dos dados tentando preservar suas características.

As características do CF são aditivas, ou seja, se temos dois *clusters* disjuntos C_1e C_2, que possuem os CF_1e CF_2, respectivamente, um novo *cluster* composto pela junção de C_1e C_2,será simplesmente $CF_1 + CF_2$.

O algoritmo BIRCH aplica uma técnica multiface com uma leitura única do conjunto de dados, e se por acaso for necessário o algoritmo faz mais uma ou duas leituras adicionais, para melhorar a qualidade do resultado.

Fase 1: o algoritmo faz a leitura do conjunto de dados e constrói uma CF tree inicial, a qual pode ser vista como uma compressão multinível dos dados que tenta preservar características da estrutura de cluster existente nos dados.

Fase 2: o algoritmo aplica um segundo algoritmo para clusterizar os nós folhas da CF tree, o qual remove clusters esparsos como outliers e agrupa clusters densos dentro de outros ainda maiores.

O procedimento de execução do algoritmo é o seguinte, a partir da raiz é encontrada a folha apropriada para inserção: siga o caminho CF mais próximo usando uma métrica de similaridade.

Logo após é necessário modificar o nó folha encontrado, se o nó folha mais próximo não pode receber o dado, ou seja, o cluster deste nó ja alcançou o diâmetro máximo, crie um novo nó folha. Se não houver espaço para o novo nó, quebre o nó pai, atualize os CFs como resultado apenas da inserção do dado ou como resultado da inserção e da quebra de um nó.

Dado *n* pontos x_i, em um espaço d-dimensional, podemos definir este grupo de pontos como sendo um *cluster* em que os seguintes parâmetros o identificam

$$x_0 = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{16}$$

Onde x_0 é o centroide que é dado pela Eq. 16, R é o raio que é dado pela Eq. 16.1 e D é o diâmetro que é dado pela Eq. 16.2.

$$R = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x_0)^2} \tag{16.1}$$

$$D = \sqrt{\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_0)^2} \tag{16.2}$$

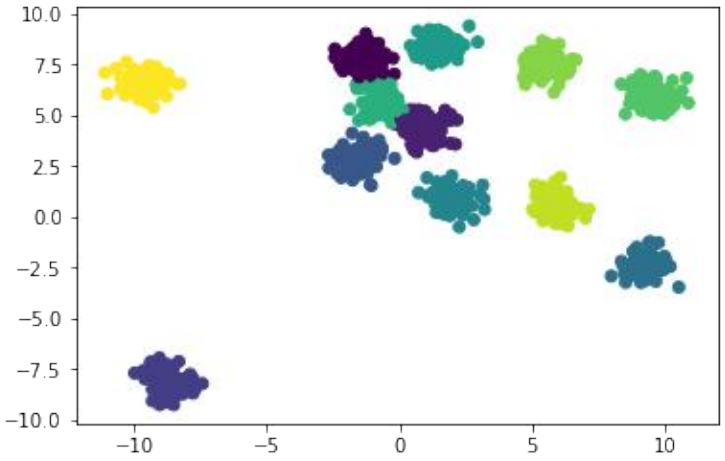
Onde R é a distância média dos objetos ao centroide do cluster e D é a distância par a par média dentro de um cluster. Ambos medem o espalhamento (ou a concentração) dos dados ao redor do centroide do cluster.

O algoritmo BIRCH possui 3 parametros, são eles:

- **Limite:** É o número máximo de pontos de dados que um subcluster no nó folha da árvore CF.
- **Fator de ramificação**: Este especifica o número máximo de subclusters CF em cada nó.
- **Número de clusters**: É o número de clusters que serão retornados após a conclusão de todo o algoritmo BIRCH.

Uma aplicação do algoritmo BIRCH em linguagem python é ilustrada na figura 29.

Figura 29: Algoritmo BIRCH com python



Fonte: YUGESH VERMA (2021)

3.1.4 DBSCAN

DBSCAN abreviação de “Density Based Spatial Clustering of Application with Noise” que em portugues significa (Clusterização Espacial Baseada em Desnidade de Aplicações com Ruído) é um algoritmo de clusterização não paramétrico baseado em densidade, semelhante ao desvio médio, foi proposto por (ESTER et al., 1996), é um método muito efetivo para identificar *clusters* de formato arbitrário e de tamanhos diferentes, ele identifica e separa os ruídos dos dados sem qualquer informação preliminar do grupo. O método so precisa de um parâmetro de entrada, mas oferece um suporte para determinar um valor apropriado para ele.

(ESTER et al.,1996) explica que a noção de *clusters* e o algoritmo DBSCAN podem ser aplicados para espaços Euclidianos de duas e três dimensões, como para qualquer espaço característico de alta dimensão. O método DBSCAN pode ser aplicavel a qualquer base de dados contendo dados de um espaço métrico, isto é, bases de dados com uma função de distância para pares de objetos (ESTER et al., 1998).

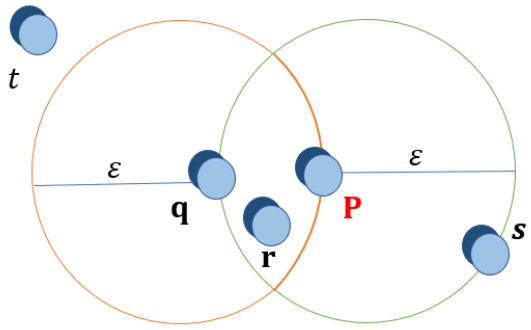
A ideia central do DBSCAN é que para cada ponto de um *cluster,* a vizinhança para um dado raio contém, pelo menos, um certo número de pontos, sendo assim, a densidade na vizinhança deve exceder um limiar. Para comprender o método, vamos ver algumas definições:

ε-vizinhança de um ponto: A vizinhança de um objeto p com raio ε é definida como

$\in$-vizinhança de p sendo definida como: $N_{\in}(p) = \{q\ em\ D |\ dist(p,q) < \varepsilon\}$.

A definição de ε-vizinhança de um ponto é mostrada na figura 30.

Figura 30: Vizinhança de um ponto



Fonte: Autoria própria (2022)

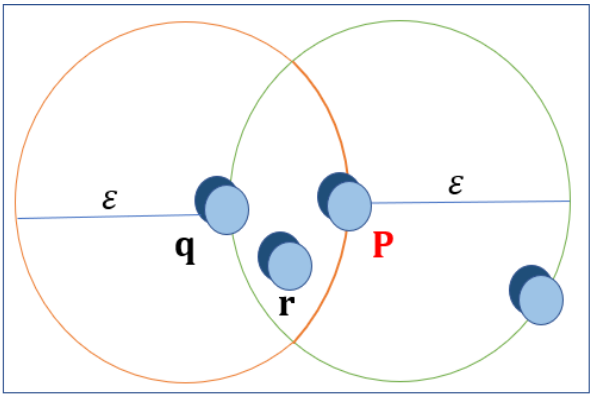
Ponto Central: Se ε-vizinhança de um objeto p contém pelo menos um número mínimo, *MinPts,* de objetos, então o objeto p é chamado de ponto central.

Pontos de Borda: Quando a ε-vizinhança de um objeto o contém menos que *MinPts*, porém tem um ponto central, então o objeto p é definido como ponto de borda.

Alcance Direto por Densidade: Um objeto p é alcançavel por densidade diretamente do objeto q, quando p está na ε-vizinhança de q, e q é um ponto central.

A figura 31 ilustra o alcance por densidade utilizado pelo método DBSCAN.

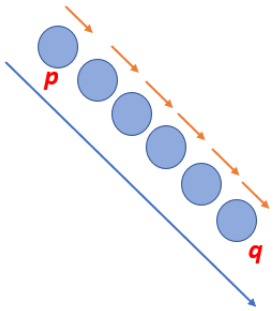
Figura 31: Alcance direto por densidade



Fonte: Autoria própria (2022)

Alcance por Densidade: Um objeto p é alcançavel por densidade do objeto q, em um conjunto D, se existe uma cadeia de objetos $\{p_1, p_2, \ldots, p_n\}$, tal que $p_1 = q$ e $p_n = q$ e p_{i+1} é alcançavel por densidade diretamente de p_i com respeito a ε e a *MinPts,* para $1 \leq i \leq n$, p_i em D, representada na figura 32.

Figura 32: Alcance por densidade método DBSCAN.



Fonte: Autoria própria (2022)

Conexão por densidade: Um objeto p é conectado por densidade do objeto q, em um conjunto D, se existe um objeto em tal os dois p e q são alcançavéis por densidade do objeto.

Cluster DBSCAN: Consideremos D como uma base de dados de pontos. Um cluster C com respeito à ε e *MinPts* é um subconjunto não vazio de D satisfazendo as condições a seguir:

1. $\forall p, q$: se p ∈ C e q é alcançavel por densidade a partir de p com respeito a ε e *MinPts,* então q ∈ C.

2. $\forall p, q$: se p ∈ C: p é conectado por densidade a q com respeito a ε e *MinPts.*

Ruído: Consideremos $C_1, C_2, \dots, C_k$ os *clusters* da base de dados D com respeito aos parâmetros $\boldsymbol{\varepsilon}$ e *MinPts,* i= 1, 2,..., K. Então, o ruído é definido como o conjunto de pontos na base de dados D que nao pertença a qualquer grupo Ci, sendo assim, o ruído = $\{p \in D \mid \forall i: p \notin Ci\}$.

Etapa 1: O DBSCAN começa com um ponto de dados inicial arbitrário que não foi visitado. A vizinhança deste ponto é extraída usando uma distância ε (todos os pontos que estão dentro da distância ε são pontos de vizinhança).

Etapa 2: Se houver um número suficiente de pontos (de acordo com *MinPts*) dentro dessa vizinhança, o processo de agrupamento será iniciado e o ponto de dados atual se tornará o primeiro ponto no novo agrupamento. Caso contrário, o ponto será rotulado como ruído (mais tarde esse ponto barulhento pode se tornar parte do cluster). Em ambos os casos esse ponto é marcado como “visitado”.

Etapa 3: Para este primeiro ponto no novo *cluster*, os pontos dentro de sua vizinhança de distância ε também passam a fazer parte dele. Este procedimento de fazer com que todos os pontos da vizinhança ε pertençam ao mesmo cluster é então repetido para todos os novos pontos que acabaram de ser adicionados ao grupo de *clusters*.

Etapa 4: Este processo das etapas 2 e 3 é repetido até que todos os pontos do cluster sejam determinados, ou seja, todos os pontos dentro da vizinhança ε do cluster foram visitados e rotulados.

Uma vez que terminamos com o cluster atual, um novo ponto não visitado é recuperado e processado, levando à descoberta de mais um cluster ou ruído. Esse processo se repete até que

todos os pontos sejam marcados como visitados. Uma vez que ao final disso todos os pontos foram visitados, cada ponto será marcado como pertencente a um cluster ou sendo ruído.

O DBSCAN apresenta algumas grandes vantagens sobre outros algoritmos de agrupamento. Em primeiro lugar, não requer um número pré definido de clusters. Ele também identifica outliers como ruídos, ao contrário do desvio médio, que simplesmente os lança em um cluster, mesmo que o ponto de dados seja muito diferente. Além disso, ele pode encontrar clusters de tamanho e formato arbitrários muito bem.

A principal desvantagem do DBSCAN é que ele não funciona tão bem quanto os outros quando os *clusters* são de densidade variável. Isso ocorre porque a configuração do limiar de distância ε e *MinPts,* para identificar os pontos de vizinhança varia de cluster para cluster quando a densidade varia. Essa desvantagem também ocorre com dados de dimensões muito altas, pois novamente o limite de distância ε torna-se difícil de estimar. (ESTER et al.,1996)

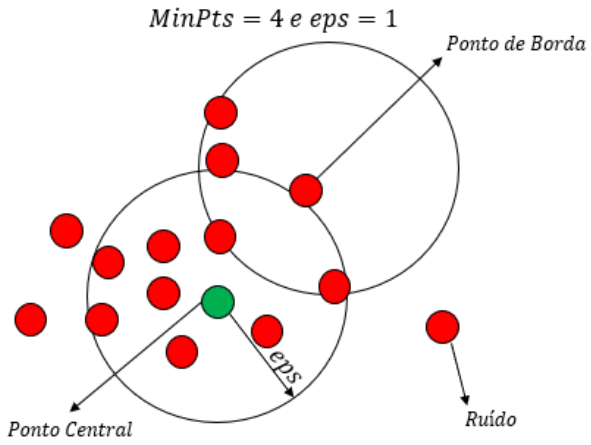
O diagrama da figura 33, descreve o funcionamento do algoritmo DBSCAN.

Figura 33: Diagrama de blocos DBSCAN.

Fonte: Autoria própria (2022)

A figura 34 representa os parâmetros do algoritmo DBSCAN, com *MinPts*=4, *eps*=1

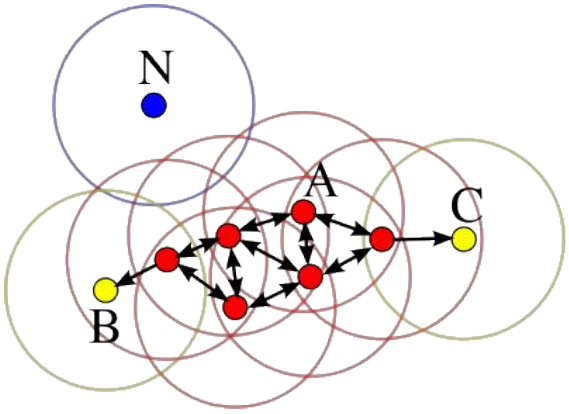
Figura 34: Parâmetros do algoritmo DBSCAN



Fonte: Autoria própria (2022)

Na figura 35 é apresentado como é feito o reconhecimento baseado em densidade usando o método DBSCAN.

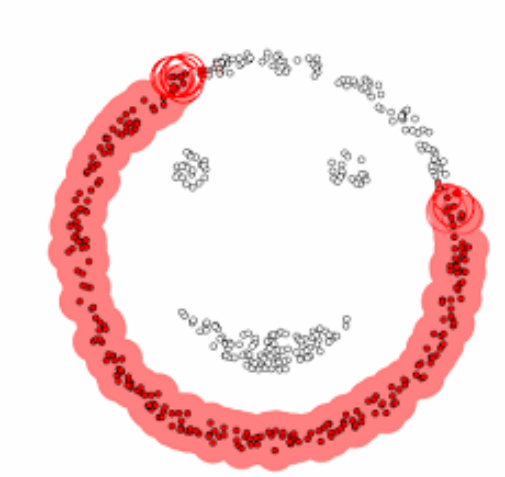
Figura 35: Reconhecimento DBSCAN



Fonte: SCHUBERT *et al.* (2017)

Na figura 36 podemos ver uma aplicação do algoritmo DBSCAN em um grupo de dados.

Figura 36: Algoritmo DBSCAN



Fonte: GEORGE SEIF (2018)

3.1.5 *K-Means*

As informações apresentadas estão disponíveis em "*The (black) art of runtime evaluation: Are we comparing algorithms or implementations?",* dos autores Hans-Peter; Schubert, Erich; Zimek, Arthur; publicado em 2016.

O *K-Means* é um algoritmo de aprendizado não supervisionado usado para encontrar grupos similares em um dado conjunto de dados (SCULLEY, 2010).

É o algoritmo de clusterização mais utilizado, e mais simples, foi descoberto primeiramente por STEINHAUS (1956) e LLOYD (1957) foi proposto em 1957, mas só foi publicado no ano de 1982, é um dos algoritmos mais usados para clusterização, pois é de fácil implementação, simplicidade, eficiência e sucesso empírico e possui várias extensões desenvolvidas em várias formas (JAIN,2009).

O algoritmo *K-Means* faz parte do grupo de algoritmos chamados de métodos particionais, e a função objetiva mais utilizada para espaços métricos nos métodos particionais é o erro quadratico, dado por:

$$E = \sum_{j=1}^{k} \sum_{x \in C_i} \|p - m_i\|^2 , \qquad para\ k\ \in (1, n) \tag{17}$$

Vamos apresentar agora alguns conceitos e notações. Consideremos um cojunto de dados com *n* pontos, definido:

$$P = \{x_i\}_{i=1}^{n}, \tag{18}$$

um agrupamento, particiona esse conjunto P em *k* subconjuntos.

$$C = \{C_1, C_2, \dots, C_k\} , \tag{19}$$

Cada C_i é denominado um *cluster* do conjunto. E para cada C_i assosciaremos um y_i que será

chamado de centroíde, este representa o *cluster* e será comparado com todos os elementos do mesmo. Para determinarmos será feita a média de todos os pontos pertecentes ao *cluster*,

$$y_i = \frac{1}{n_i}\sum_{x_j \in C_i} x_j, \tag{20}$$

Onde $n_i$será o número de elementos de C_i.

Para medir a dispersao dos elementos do cluster, e verificar o quão proximo eles estão, compararando a distância dos pontos ao centroide correspondente, usaremos a função de semelhanca,

$$F_s(P) = \sum_{i=1}^{k}\sum_{x_j \in C_i} D(x_j, y_i), \tag{21}$$

Onde D é a distância euclidiana entre $x_j \; e \; y_i$, sendo assim F_s, é a soma de todas as distancia entre cada elemento e o centroide do seu cluster, medindo assim quão bem o centroíde representa seu grupo. O algoritmo consiste nos seguintes passos quando se encontra com um conjunto de dados.

- Distribui todos os pontos do conjunto P de forma aleatória em *k clusters.*
- Calcula usando a equação (4), o centroíde de cada *cluster* C_i.
- Atribui cada ponto $x_j \in P$, a um *cluster* C_{i*}, do centroíde y_{i*}, mais próximo ao ponto, ou seja,

$$i^* = \underset{i=1,2,\ldots,k}{\operatorname{argmin}} \left\| x_j - y_i \right\|_2^2, \tag{22}$$

Isso quer dizer que um ponto x_j qualquer, será agrupado a C_i, quando este pponto possuir a menor distância ao centroíde desse *cluster,* comparado com a distância dos outros centroides dos restos dos outros *clusters.*

- Com a realização do passo acima muitos pontos irão mudar de grupo, por isso é necessário a atualização do centroíde de cada *cluster.* Desta forma é só repetir o 2°

passo, onde será encontrado um novo centroíde y_i para o *cluster* C_i.

- Os dois ultimos passos serão repetidos de forma iterativa, até que os respectivos centroides nao mudem mais de local ou satisfaça a precisão estabelecida, então esta iteracão será o ´ mínimo local do nosso problema.

Chegamos assim no teste de parada, que é analisado através da soma das diferenças dos centroides da iteração atual pela inferior.

$$\sum_{i=1}^{k} \|{y_i}^t - {y_i}^{t-1}\|_2^2 \leq \in, \tag{23}$$

Onde ${y_i}^t$ representa o centroide da iteração atual, $\in > 0$ é a precisão determinada e o limite da convergência. Abaixo podemos ver o pseudo-código do algoritmo *K-means.*

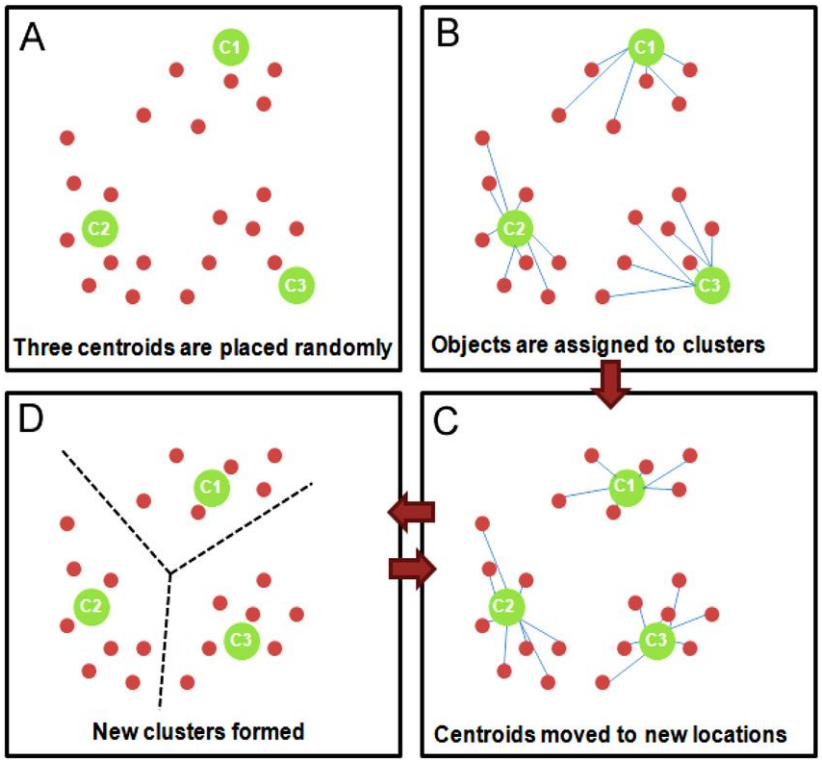
Considere: *P* conjunto dos pontos, k número de *clusters,* ε erro.

Algoritmo de Clusterização *k-means*

1: **Início**
2: $Iteração \leftarrow 0$
3: Inicializar $y_i, i = \{1,2,3,\dots,k\}$ Com pontos aleatórios de P
4: **Repita**
5: $C_i \leftarrow \emptyset, \quad \forall_j = 1,2,\dots,k$
6: **Para** $x_j \in P$ **faça**
7: $i^* \leftarrow \text{argmin}\left\|x_j - {y_i}^{iteração}\right\|^2$
8: $C_{i*} = C_{i*} \cup \{x_j\}$
9: **Fim**
10: **Para** $i = 1$ $até$ k **Faça**
11: ${y_i}^{iteração} = \frac{1}{|C_i|}\sum_{x \in C_i} x_j$
12: **Fim**
13: **Até** $\sum_{i=1}^{k}\left\|{y_j}^{iteração} - {y_i}^{iteração-1}\right\| < \epsilon$
14: **Fim**

A ilustração de como seria a aplicação do *K-Means* em um agrupamento é mostrada na figura 37, para achar os centroides e posteriormente o mínimo local do problema.

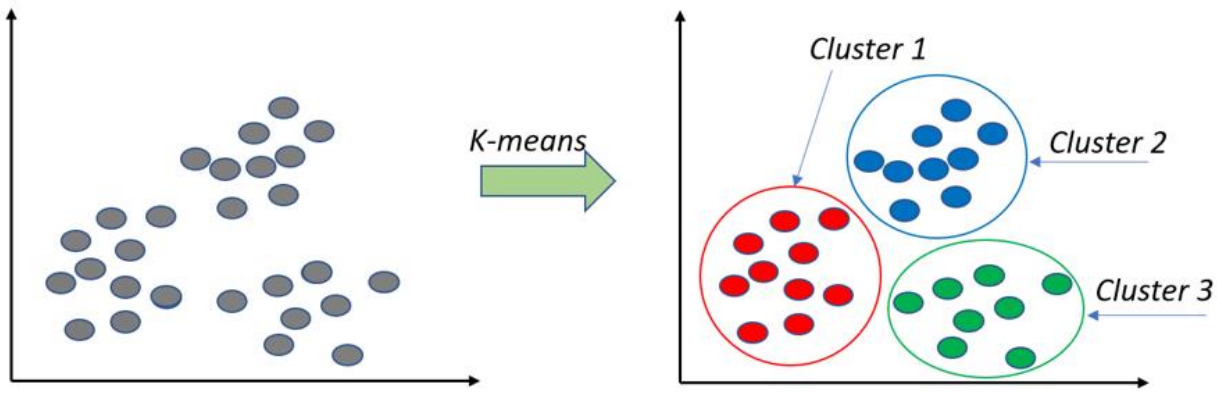
Figura 37: Funcionamento *K-Means*



Fonte: RUMDEEP (2013)

Veja na figura 38 a aplicação do algoritmo *K-Means* em um agrupamento, o agrupamento antes da aplicação *K-Means* é o lado da esquerdo, e a direita é após a aplicação do algoritmo.

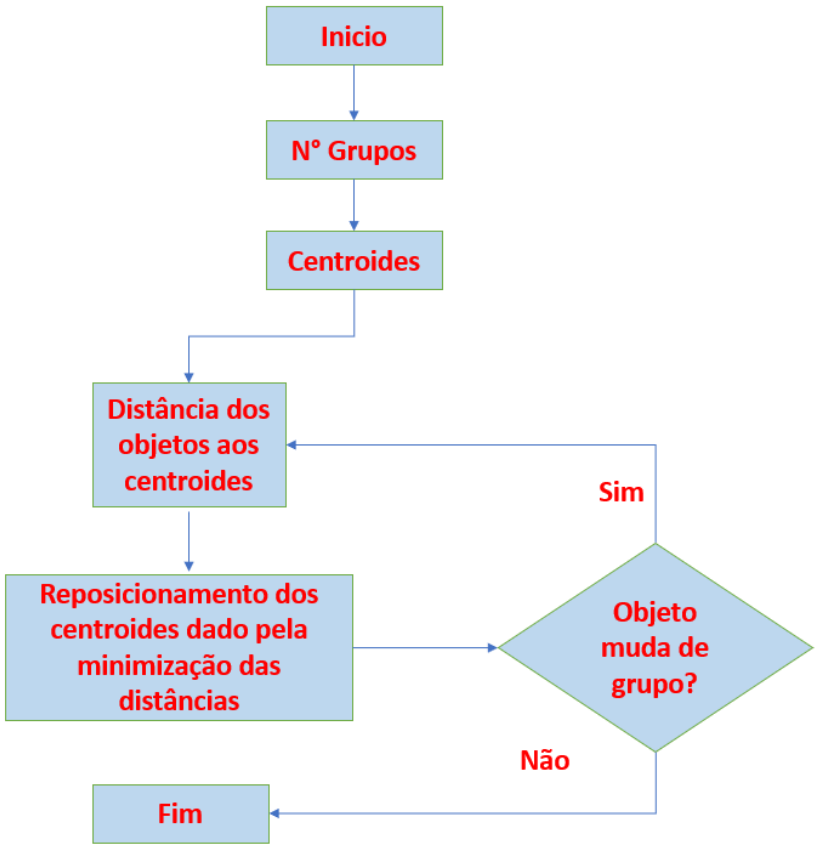
Figura 38: Aplicação do K-Means



Fonte: Autoria própria (2022)

Na figura 39 podemos ver um diagrama de blocos de como é feito o processo de clusterização do *k-means.*

Figura 39: Diagrama de blocos *k-means*



Fonte: Autoria própria (2022)

3.1.6 *Mini Batch K-Means*

As informações apresentadas estão disponíveis em *K-means vs Mini Batch K-means: A comparison* do autor Javier Béjar, do Departament de Llenguatges i Sistemes Informàtics Universitat Politècnica de Catalunya.

Segundo o artigo a ideia principal do algoritmo *Mini Batch K-Means* é usar pequenos lotes aleatórios de dados de tamanho fixo, para que possam ser armazenados na memória. A cada iteração uma nova amostra aleatória do conjunto de dados é obtida e utilizada para atualizar os clusters e isso é repetido até a convergência. Cada minilote atualiza os clusters usando uma combinação convexa dos valores dos protótipos e dos dados, aplicando uma taxa de aprendizado que diminui com o número de iterações. Essa taxa de aprendizado é o inverso do número de dados atribuídos a um cluster durante o processo. À medida que o número de

iterações aumenta, o efeito de novos dados é reduzido, de modo que a convergência pode ser detectada quando não ocorrem alterações nos clusters em várias iterações consecutivas (BÉJAR J, 2013).

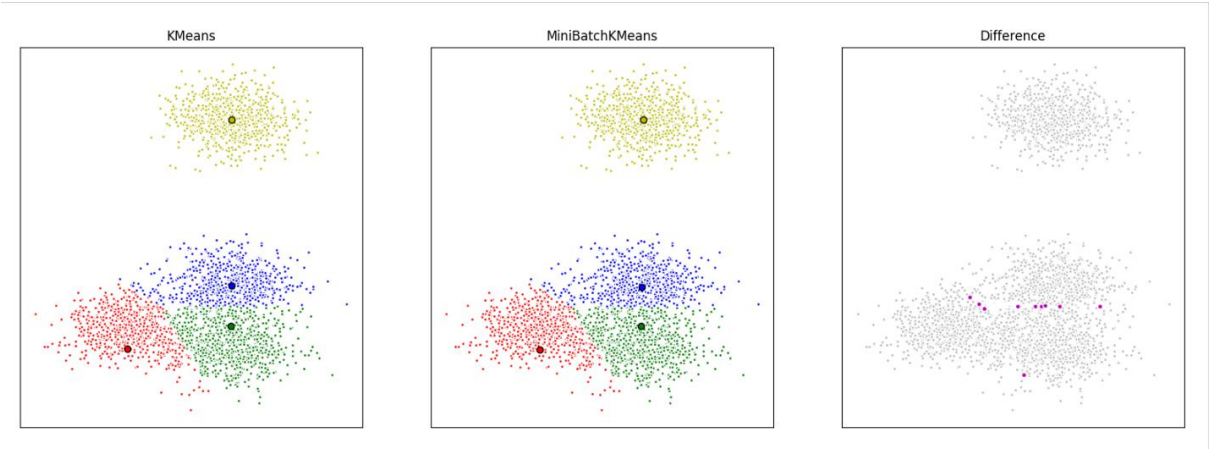
Os resultados empíricos sugerem que ele pode obter uma economia substancial de tempo computacional às custas de alguma perda de qualidade do cluster, mas não foi feito um estudo extensivo do algoritmo para medir como as características dos conjuntos de dados, como o número de clusters ou seu tamanho, afetam a qualidade da partição (BÉJAR J, 2013).

O algoritmo leva pequenos lotes escolhidos aleatoriamente do conjunto de dados para cada iteração. Cada dado no lote é atribuído aos clusters, dependendo das localizações anteriores dos centroides do cluster. Em seguida, ele atualiza as localizações dos centroides do cluster com base nos novos pontos do lote. A atualização é uma atualização de gradiente descendente, que é significativamente mais rápida do que uma atualização normal do *Batch K-Means*.

O *mini Batch K-Means* é mais rápido, mas fornece resultados ligeiramente diferentes do que o *K-Means* normal do lote. À medida que o número de clusters e o número de dados aumentam, a economia relativa em tempo computacional também aumenta. A economia de tempo computacional é mais perceptível apenas quando o número de clusters é muito grande. O efeito do tamanho do lote no tempo computacional também é mais evidente quando o número de clusters é maior. Pode-se concluir que, aumentando o número de clusters, diminui-se à similaridade da solução *Mini-Batch K-Means* com a solução *K-means*. Apesar de a concordância entre as partições diminuir à medida que o número de clusters aumenta, a função objetivo não se degrada na mesma taxa. Isso significa que as partições finais são diferentes, mas mais próximas em qualidade. (BÉJAR J, 2013).

A figura 40 apresenta a diferença da aplicação do algoritmo, *K-Means* e *Mini-Batch K-Means.*

Figura 40: Comparação entre *K-Means* e *Mini-Batch K-Means*.



Fonte: GeeksforGeeks (2019)

3.1.7 *Mean Shift*

Segundo Dorin Comaniciu e Peter Meer, "*Mean Shift: A robust approach toward feature space analysis"*.

O agrupamento de deslocamento médio é um algoritmo baseado em janela deslizante que tenta encontrar áreas densas de pontos de dados. É um algoritmo baseado em centroides, o que significa que o objetivo é localizar os pontos centrais de cada grupo/classe, que funciona atualizando candidatos para pontos centrais para serem a média dos pontos dentro da janela deslizante. Essas janelas candidatas são então filtradas em um estágio de pós-processamento para eliminar quase duplicatas, formando o conjunto final de pontos centrais e seus grupos correspondentes (COMANICIU D, MEER P, 2002)

Começamos com uma janela deslizante circular centrada em um ponto C (selecionado aleatoriamente) e tendo o raio r como kernel. O deslocamento médio é um algoritmo de escalada que envolve o deslocamento desse kernel iterativamente para uma região de densidade mais alta em cada etapa até a convergência.

A cada iteração, a janela deslizante é deslocada para regiões de maior densidade, deslocando o ponto central para a média dos pontos dentro da janela. A densidade dentro da janela deslizante é proporcional ao número de pontos dentro dela. Naturalmente, ao mudar para a média dos pontos na janela, ela se moverá gradualmente para áreas de maior densidade de pontos.

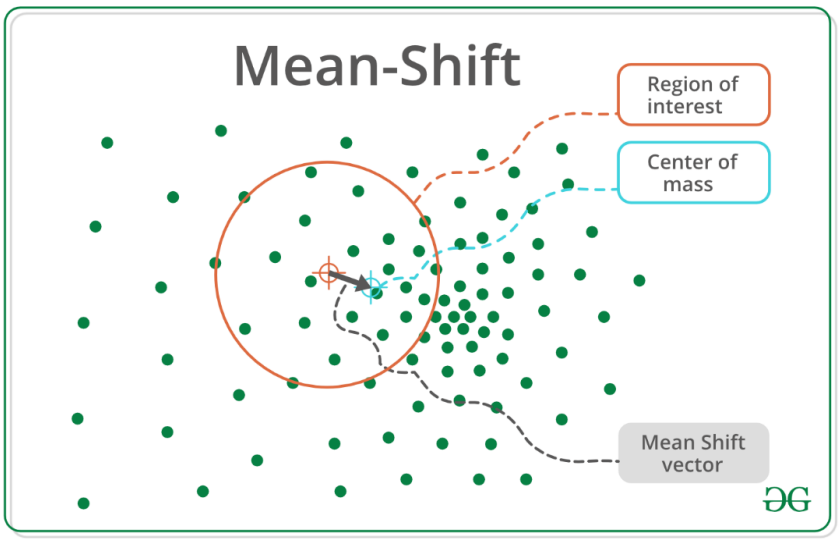
Continuamos deslocando a janela deslizante de acordo com a média até que não haja direção na qual um deslocamento possa acomodar mais pontos dentro do kernel.

Este processo das etapas 1 a 3 é feito com muitas janelas deslizantes até que todos os pontos estejam dentro de uma janela. Quando várias janelas deslizantes se sobrepõem, a janela que contém o maior número de pontos é preservada. Os pontos de dados são então agrupados de acordo com a janela deslizante em que residem.

Diferente do agrupamento K-Means, não há necessidade de selecionar o número de agrupamentos, pois o deslocamento médio descobre isso automaticamente. Isso é uma vantagem enorme. O fato de os centros dos clusters convergirem para os pontos de densidade máxima também é bastante desejável, pois é bastante intuitivo de entender e se encaixa bem em um sentido naturalmente orientado a dados. A desvantagem é que a seleção do tamanho da janela/raio "r" pode não ser trivial (COMANICIU D, MEER P, 2002)

A figura 41 apresenta um exemplo do *Mean shift,* com as devidas separações, a Região de interesse, o centro de massa, e os vetores *Mean shift*

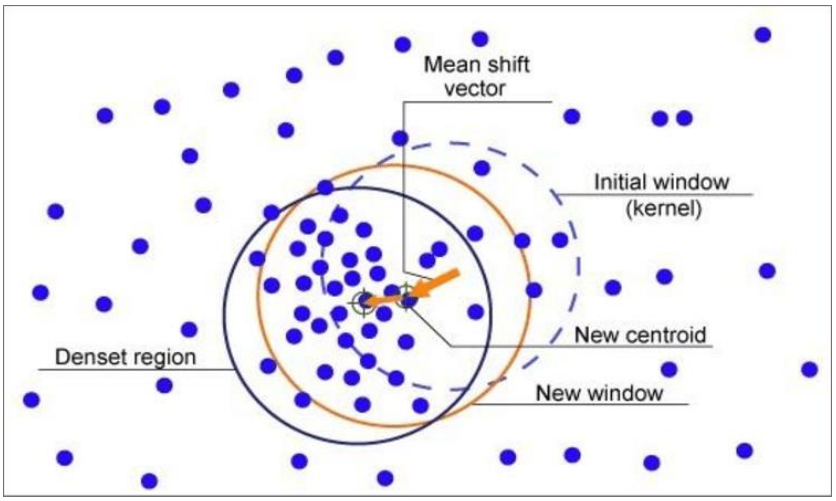
Figura 41: Exemplo de *Mean Shift*



Fonte: GeeksforGeeks (2019)

A figura 42 apresenta os parâmetros do método *Mean shift*, detalhadamente.

Figura 42: Parâmetros *Mean Shift*



Fonte: BEPERY (2021)

3.1.8 OPTICS

De acordo com Kriegel, Hans-Peter; Kröger, Peer; Sander, Jörg; Zimek, Arthur; (May 2011). *"Density-based clustering"*. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.* (KRIEGEL et.al, 2011)

O Algoritmo OPTICS é um parente do DBSCAN, que invoca um processo diferente. Ele criará um gráfico de acessibilidade que é usado para extrair clusters e, embora ainda haja uma entrada, épsilon máximo, os outros parâmetros não têm um efeito tão grande quanto seus equivalentes em outros algoritmos de agrupamento e são muito mais fáceis de usar os padrões. (KRIEGEL et.al, 2011)

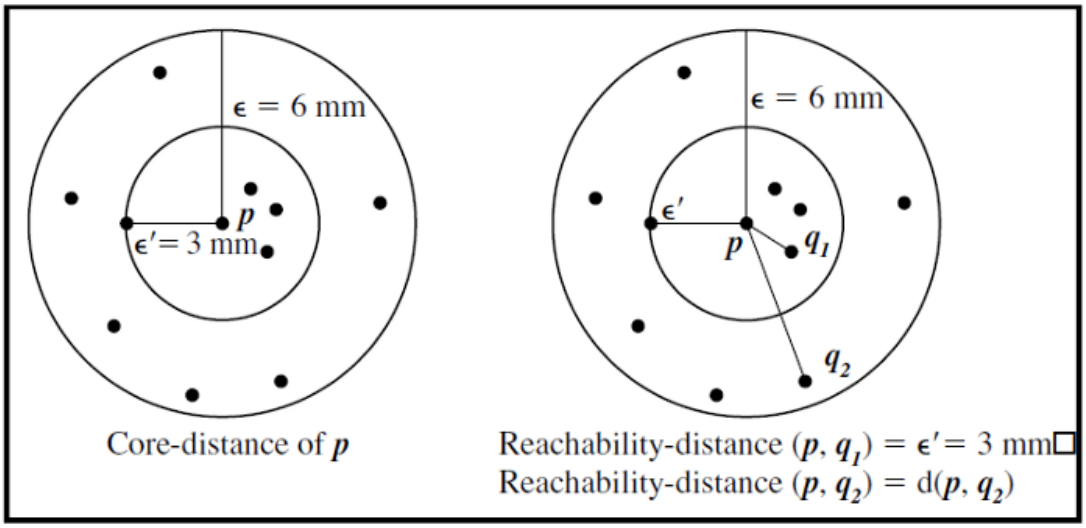
Para entender o funcionamento do OPTICS é necessário saber o funcionamento do DBSCAN, os parâmetros que ele usa e a diferença entre pontos centrais e limites.

Distância do núcleo: O épsilon mínimo para tornar-se um ponto distinto um ponto central, dado um parâmetro *MinPts* finito.

Distância de Acessibilidade: A distância de acessibilidade de um objeto p em relação a outro objeto o é a menor distância de o se o for um objeto central. Também não pode ser menor que a distância do núcleo.

A figura 43 mostra os parâmetros utilizados no algoritmo OPTICS.

Figura 43: Parâmetros do algoritmo OPTICS



Fonte: SINCLAIR (2019)

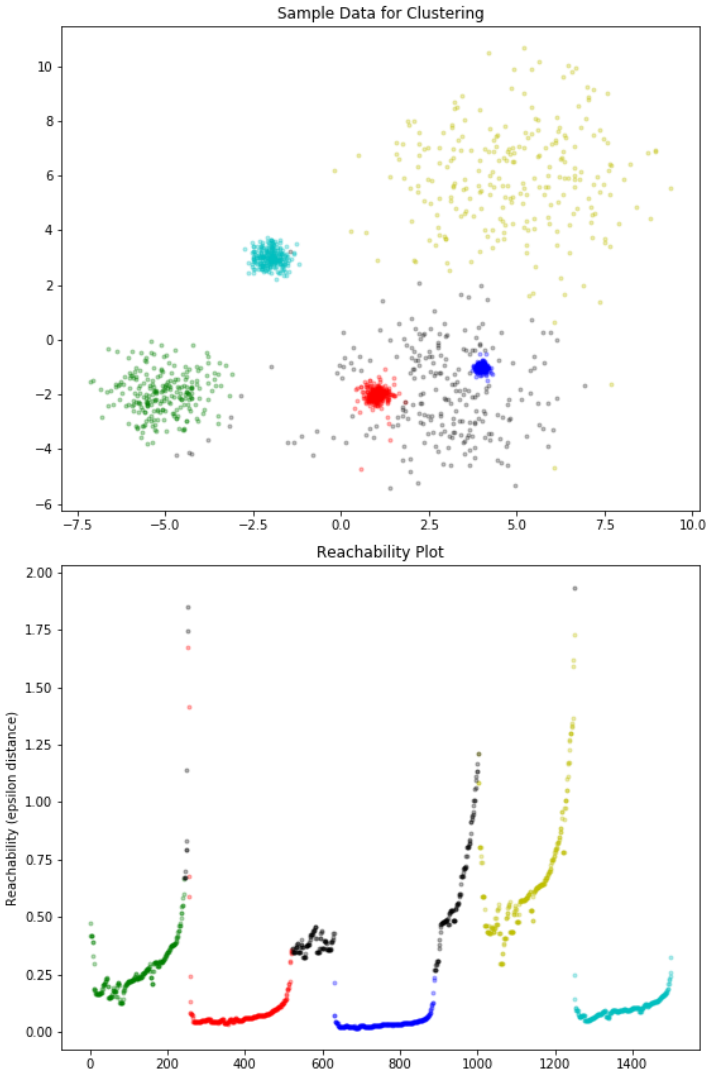
Embora o parâmetro *MinPts* seja usado nesses cálculos, a ideia é que não teria muito impacto porque todas as distâncias seriam dimensionadas aproximadamente na mesma taxa.

Essas definições são utilizadas para criar nosso gráfico de acessibilidade, que será usado para extrair os clusters. Primeiro, começamos calculando as distâncias do núcleo em todos os pontos de dados no conjunto. Em seguida, percorreremos todo o conjunto de dados e atualizaremos as distâncias de acessibilidade, processando cada ponto apenas uma vez. Atualizaremos apenas as distâncias de acessibilidade de pontos que precisam ser melhorados e ainda não foram processados. Isso ocorre porque quando processamos um ponto, definimos em pedra sua ordenação, bem como sua distância de alcance. O próximo ponto de dados escolhido para processar será aquele que tiver a distância de alcance mais próxima. É assim que o algoritmo mantém os clusters próximos uns dos outros na ordenação de saída (KRIEGEL et.al, 2011).

O próximo passo será extrair os rótulos de cluster reais do gráfico. A maneira mais comum de fazer isso é procurando por “vales” na parcela, usando mínimos e máximos locais. Alguns parâmetros mais podem entrar em jogo aqui, dependendo do método usado.

A figura 44 mostra uma comparação de alguns dados de amostra gerados e os rótulos ópticos resultantes e o gráfico de acessibilidade. Os pontos coloridos são aqueles identificados como clusters, enquanto os cinzas representam ruídos.

Figura 44: Dados de amostra gerados e os rótulos ópticos resultantes



Fonte: LEARN (2021)

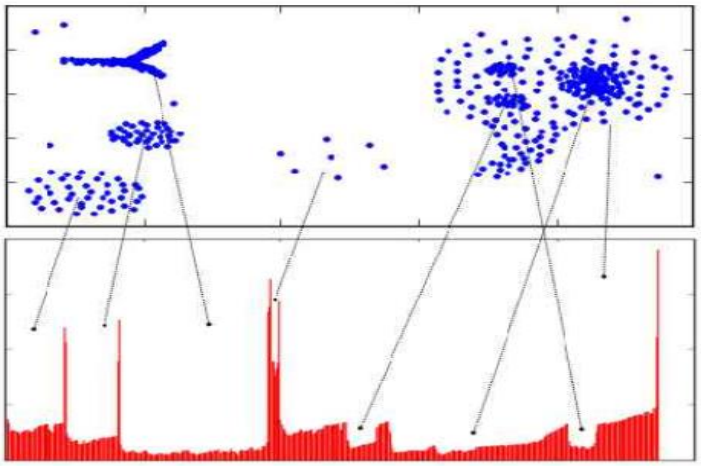
Observe que há uma boa quantidade de pontos identificados como pontos de ruído neste exemplo gerado. Eles têm densidades semelhantes à do aglomerado amarelo, mas não são reconhecidos nesta extração porque se concentra em separar as regiões mais densas. É aqu i que o ajuste fino dos parâmetros de extração se beneficiaria. Outros métodos de extração de clusters também podem ser usados neste caso.

Outro aspecto interessante do algoritmo OPTICS é uma extensão dele usada para detecção de outliers, chamada OPTICS-OF, isso dará uma pontuação discrepante para cada ponto, que é uma comparação com seus vizinhos mais próximos, em vez de todo o conjunto. Este é um tipo único de detecção de valores discrepantes devido a este princípio “local”. (KRIEGEL et.al, 2011)

Primeiro, precisamos criar uma medida “densidade de acessibilidade local”, que é o inverso da acessibilidade média (em relação ao ponto que você está calculando) dos vizinhos *MinPts*.

Na figura 45 podemos ver uma aplicação do algoritmo OPTICS

Figura 45: Aplicação do algoritmo OPTICS



Fonte: YING-SI, SHEN, (2013)

3.1.9 *Spectral Clustering*

As informações aqui apresentadas podem ser encontradas em *“Learning spectral clustering”*. dos autores BACH, F. e JORDAN, M.

Spectral Clustering é definido como um algoritmo de agrupamento crescente e que tem um desempenho melhor do que muitos algoritmos de agrupamento tradicionais em muitos casos. Ele trata cada ponto de dados como um nó de grafo e, assim, transforma o problema de agrupamento em um problema de particionamento de grafos (BACH, JORDAN,2006). Uma implementação típica consiste em três etapas fundamentais:

Construindo o Gráfico de Semelhança: Esta etapa constrói o Gráfico de Semelhança na forma de uma matriz de adjacência que é representada por A. A matriz de adjacência pode ser construída das seguintes maneiras:

Gráfico Épsilon-vizinhança: Um parâmetro épsilon é fixado de antemão. Então, cada ponto é conectado a todos os pontos que estão em seu raio épsilon. Se todas as distâncias entre quaisquer dois pontos forem semelhantes em escala, normalmente os pesos das arestas, ou seja, a distância entre os dois pontos não será armazenada, pois não fornecem nenhuma informação adicional. Assim, neste caso, o grafo construído é um grafo não direcionado e não ponderado.

K-Vizinhos Mais Próximos um parâmetro k é fixado de antemão. Então, para dois vértices u e v, uma aresta é direcionada de u para v somente se v estiver entre os k vizinhos mais próximos de u. Observe que isso leva à formação de um grafo ponderado e direcionado porque nem sempre é o caso de que para cada u tendo v como um dos k-vizinhos mais próximos, será o mesmo caso para v tendo u entre seus *k*-vizinhos mais próximos. vizinhos. Para tornar este gráfico não direcionado uma das seguintes abordagens é seguida. (BACH, JORDAN,2006).

Direcione uma aresta de u para v e de *v* para *u* se *v* estiver entre os *k* vizinhos mais próximos de *u* ou onde estiver entre os *k* vizinhos mais próximos de *v*.

Direcione uma aresta de *u* para *v* e de *v* para *u* se *v* estiver entre os k vizinhos mais próximos de *u* E u estiver entre os *k* vizinhos mais próximos de *v*.

Gráfico totalmente conectado: Para construir este gráfico, cada ponto é conectado com uma aresta não direcionada ponderada pela distância entre os dois pontos a todos os outros pontos. Como essa abordagem é usada para modelar as relações de vizinhança local, normalmente a métrica de similaridade gaussiana é usada para calcular a distância.

Projetando os dados em um espaço dimensional inferior: Este passo é feito para levar em conta a possibilidade de que membros do cluster possam estar distantes no espaço dimensional dado. Assim, o espaço dimensional é reduzido para que esses pontos estejam mais próximos no espaço dimensional reduzido e, portanto, possam ser agrupados por um algoritmo de agrupamento tradicional. Isso é feito pelo cálculo da Matriz Laplaciana do Grafo. Para calculá-lo, primeiro, o grau de um nó precisa ser definido. (BACH, JORDAN,2006).

O Grau do i-ésimo nó é dado por:

$$d_i = \sum_{j=1|(i,j)\in E}^{n} w_{ij} \tag{24}$$

Observe que w_{ij} é a aresta entre os nós i e j conforme definido na matriz de adjacência acima.

A matriz de graus é definida da seguinte forma:

$$D_{ij} = \begin{cases} d_i, i = j \\ 0, i \neq j \end{cases} \quad (25)$$

Assim, a Matriz Laplaciana do Grafo é definida como:

$$L = D - A \quad (26)$$

Esta Matriz é então normalizada para eficiência matemática. Para reduzir as dimensões, primeiro são calculados os autovalores e os respectivos autovetores. Se o número de clusters for k, então os primeiros autovalores e seus autovetores são tomados e empilhados em uma matriz de modo que os autovetores sejam as colunas.

Agrupando os dados: Esse processo envolve principalmente o agrupamento dos dados reduzidos usando qualquer técnica tradicional de agrupamento – normalmente o agrupamento K-Means. Primeiro, a cada nó é atribuída uma linha da normalizada da Matriz Laplaciana do Grafo. Em seguida, esses dados são agrupados usando qualquer técnica tradicional. Para transformar o resultado do *clustering*, o identificador do nó é retido. Abaixo podemos ver algumas propriedades: (BACH, JORDAN,2006).

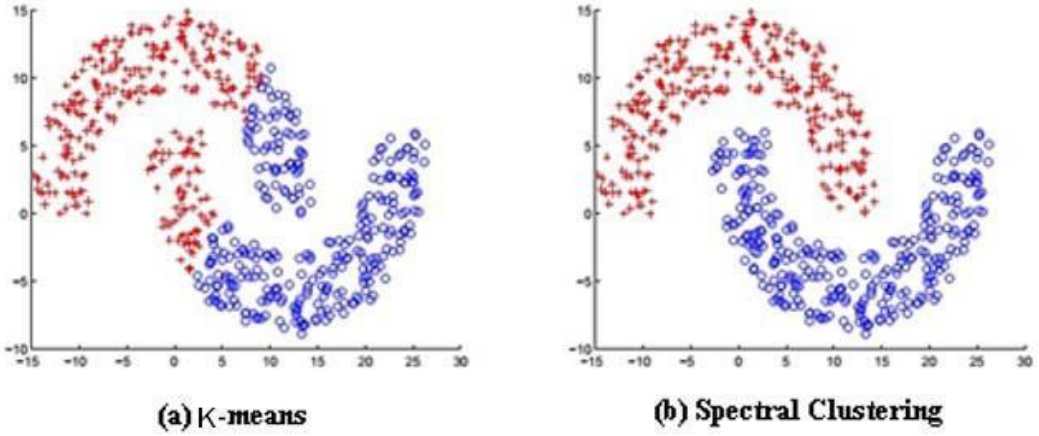
Assunção-Menos: Esta técnica de agrupamento, ao contrário de outras técnicas tradicionais, não assume que os dados seguem alguma propriedade. Assim, isso faz com que essa técnica responda a uma classe mais genérica de problemas de agrupamento.

Facilidade de implementação e velocidade: Este algoritmo é mais fácil de implementar do que outros algoritmos de agrupamento e é muito rápido, pois consiste principalmente em cálculos matemáticos.

Não escalável: Uma vez que envolve a construção de matrizes e cálculo de autovalores e autovetores, é demorado para conjuntos de dados densos.

Na figura abaixo 46 podemos ver a diferença do algoritmo *k-Means* e o *Spectral Clustering*.

Figura 46: Comparação do algoritmo *K-Means* e *Spectral Clustering*



Fonte: ABDELKARIM (2017)

3.2 Mixture of Gaussians

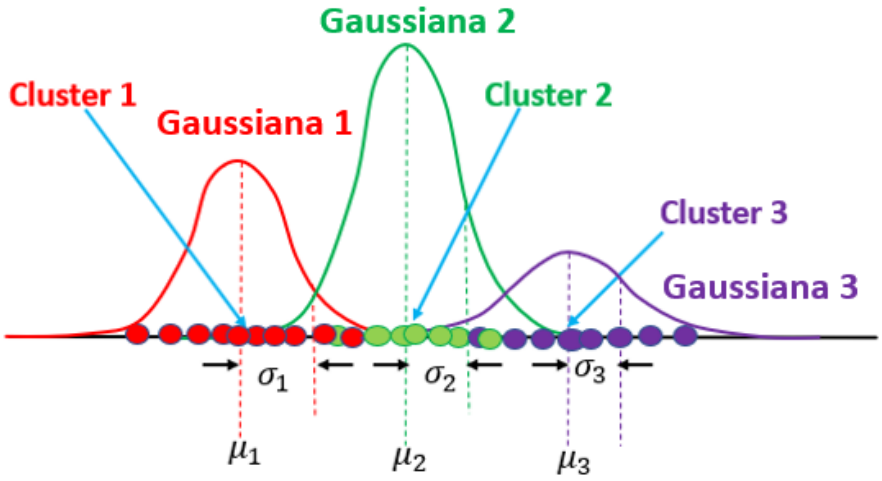
As informações sobre *Mixture of Gaussians*, podem ser encontradas no livro *Pattern Recognition and Machine Learning* (2006) Springer-Verlag Berlim, Heidelberg. do autor Bishop Christopher M.

Os modelos de mistura gaussiana (GMMs) é uma função composta por várias Gaussianas, cada uma identificada por $k \in \{1, \ldots, K\}$, onde K é o número de clusters do nosso conjunto de dados. Cada Gaussiana k na mistura é composto pelos seguintes parâmetros:

- Uma média μ que define seu centro.
- Uma covariância $\sum$ que define sua largura. Isso seria equivalente às dimensões de um elipsoide em um cenário multivariado.
- Uma probabilidade de mistura π que define quão grande ou pequena será a função gaussiana.

Na figura 47 é ilustrado três funções gaussianas, logo $K = 3$. Cada gaussiana explica os dados contidos em cada um dos três clusters disponíveis.

Figura 47: Mistura Gaussiana



Fonte: Autoria própria (2022)

Os coeficientes de mistura são probabilidades e devem atender a esta condição:

$$\sum_{k=1}^{K} \pi_k = 1 \tag{27}$$

Os parâmetros devem ser escolhidos de forma que eles fiquem mais perto do valor ideal possível, isso é feito garantindo que cada gaussiana se ajuste aos pontos de dados pertencentes a cada cluster, é utilizado a função máxima de probabilidade (BISHOP M CHRISTOPHER, 2006), assim a função densidade gaussiana é dada por:

$$\mathcal{N}(x|\mu, \sum) = \frac{1}{(2\pi)^{\frac{D}{2}}|\sum|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \sum^{-1}(x-\mu)\right) \tag{28}$$

Onde x representa nossos pontos de dados, D é o número de dimensões de cada ponto de dados. μ e $\sum$ são a média e a covariância, respectivamente.

Para fins posteriores, também é útil tomar o logaritmo dessa equação, que é dado por:

$$\ln(x|\mu,\sum) = -\frac{D}{2}\ln 2\pi - \frac{1}{2}\ln\sum - \frac{1}{2}(x-\mu)^T\sum^{-1}(x-\mu) \quad (28.1)$$

Diferenciando esta equação em relação à média e covariância e depois igualando a zero, poderemos encontrar os valores ótimos para esses parâmetros. Porém, como estamos lidando com muitas gaussianas, as coisas ficarão um pouco complicadas quando chegar a hora de encontrarmos os parâmetros para toda a mistura (BISHOP M CHRISTOPHER, 2006).

Nesse sentido, precisaremos introduzir alguns aspectos adicionais que será mostrado adiante. Vamos agora introduzir algumas notações adicionais. Primeiro, vamos supor que queremos saber qual é a probabilidade de que um ponto de dados xn venha do gaussiano k.

Podemos expressar isso como:

$$\mathrm{p}(z_{nk} = 1|x_n) \quad (28.2)$$

Que é lido como "dado um ponto de dados $\boldsymbol{x}$, qual é a probabilidade de ter vindo de Gaussiano k?" Nesse caso, z é uma variável latente que recebe apenas dois valores possíveis. É um quando x veio do gaussiano k, e zero caso contrário. Na verdade, não conseguimos ver essa variável z na realidade, mas conhecer sua probabilidade de ocorrência será útil para nos ajudar a determinar os parâmetros da mistura gaussiana, como discutiremos mais adiante (BISHOP M CHRISTOPHER, 2006).

Da mesma forma, podemos afirmar o seguinte:

$$\pi_k = p(z_{nk} = 1) \quad (28.3)$$

O que significa que a probabilidade geral de observar um ponto que vem da Gaussiana k é na verdade equivalente ao coeficiente de mistura para aquela Gaussiana. Isso faz sentido, porque quanto maior for a Gaussiana, maior será essa probabilidade. Agora seja z o conjunto de todas as possíveis variáveis latentes z, (BISHOP M CHRISTOPHER, 2006), portanto:

$$z = \{z_1, \dots, z_k\} \tag{28.4}$$

Sabemos de antemão que cada z ocorre independentemente dos outros e que eles só podem assumir o valor de um quando k é igual ao cluster de onde vem o ponto. Portanto:

$$p(z) = p(z_1 = 1)^{z_1} p(z_2 = 1)^{z_2} \dots p(z_K = 1)^{z_K} = \prod_{k=1}^{K} \pi_k^{z_k} \tag{28.5}$$

Podemos encontrar a probabilidade de observar nossos dados, acontece que na verdade é a própria função gaussiana seguindo a mesma lógica que usamos para definir $p(z)$, podemos afirmar:

$$\mathrm{p}(x_n|z) = \prod_{k=1}^{K} \mathcal{N}(x_n|\mu_k, \sum_k)^{z_k} \tag{28.6}$$

O objetivo é determinar qual a probabilidade de z dada nossa observação x, as equações que foram derivadas acima, juntamente com a regra de Bayes, irão determinar essa probabilidade.

Pela regra do produto de probabilidades, sabemos que:

$$\mathrm{p}(x_n, z) = \mathrm{p}(x_n|z)p(z) \tag{28.7}$$

Os operandos à direita são o que acabamos de encontrar. No entanto, primeiro precisaremos de $\mathrm{p}(xn)$ não de $\mathrm{p}(xn, z)$. Só precisamos somar os termos em z, logo:

$$p(x_n) = \sum_{k=1}^{K} p(x_n|z)p(z) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \sum_k) \tag{28.8}$$

Esta é a equação que define uma Mistura Gaussiana, e você pode ver claramente que ela depende de todos os parâmetros que mencionamos anteriormente. Para determinar os valores ótimos para estes, precisamos determinar a máxima verossimilhança do modelo. (BISHOP M CHRISTOPHER, 2006),

A verossimilhança como a probabilidade conjunta de todas as observações xn,

definidas por:

$$p(X) = \prod_{n=1}^{N} p(x_n) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \, \mathcal{N}(x_n|\mu_k, \sum_k) \tag{28.9}$$

Da mesma forma que foi feito com a função densidade gaussiana original, vamos aplicar o logaritmo a cada lado da equação:

$$\ln p(X) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \, \mathcal{N}(x_n|\mu_k, \sum_k) \tag{29}$$

Agora, para encontrar os parâmetros ótimos para a mistura gaussiana, tudo o que precisamos fazer é diferenciar essa equação em relação aos parâmetros. Porém aqui temos um problema. Podemos ver que há um logaritmo que está afetando a segunda soma. Calcular a derivada dessa expressão e depois resolver os parâmetros será muito difícil.

Precisamos usar um método iterativo para estimar os parâmetros, neste momento já temos tudo para definir como será essa probabilidade. Pela regra de Bayes, sabemos que:

$$\mathrm{p}(z_k = 1|x_n) = \frac{\mathrm{p}(x_n|z_k = 1)\,\mathrm{p}(z_k = 1)}{\sum_{j=1}^{K} \mathrm{p}\big(x_n|z_j = 1\big)\,\mathrm{p}(z_j = 1)} \tag{30}$$

De nossas derivações anteriores, temos que:

$$\mathrm{p}(z_k = 1) = \pi_k, \;\; \mathrm{p}(x_n|z_k = 1) = \mathcal{N}(x_n|\mu_k, \sum_k) \tag{31}$$

Então, vamos agora substituí-los na equação anterior:

$$\mathrm{p}(z_k = 1|x_n) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \sum_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\big(x_n|\mu_j, \sum_j\big)} = \gamma(z_{nk}) \tag{32}$$

É isso que procurávamos, daqui para frente vamos ver muito essa expressão. Em seguida, continuaremos nossa discussão com um método que nos ajudará a determinar facilmente os parâmetros para a mistura gaussiana. (BISHOP M CHRISTOPHER, 2006),

Expectativa algoritmo de maximização: É amplamente utilizado para problemas de otimização onde a função objetivo possui complexidades como a que acabamos de encontrar para o caso GMM. (BISHOP M CHRISTOPHER, 2006).

Sejam os parâmetros do nosso modelo:

$$\theta = \{\pi, \mu, \sum\} \tag{33}$$

Passo 1: Inicialize θ de acordo. Por exemplo, podemos usar os resultados obtidos por uma execução anterior de *K-Means* como um bom ponto de partida para nosso algoritmo.

Passo 2: Avaliar

$$\mathcal{Q}(\theta^*, \theta) = \mathbb{E}[\ln p(X, Z|\theta^*)] = \sum_{Z} p(Z|X, \theta) \ln p(X, Z|\theta^*) \tag{34}$$

Bem, na verdade já encontramos $\mathrm{p}(\mathrm{Z}|\mathrm{X}, \theta)$. Vamos trazer nossa equação anterior aqui:

$$\mathrm{p}(z_k = 1|x_n) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \sum_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(x_n|\mu_j, \sum_j\right)} = \gamma(z_{nk}) \tag{29}$$

Para modelos de mistura gaussiana, o passo de expectativa se resume a calcular o valor de γ em (29) usando os valores dos parâmetros antigos. Agora se substituirmos (29) em (34), teremos:

$$\mathcal{Q}(\theta^*, \theta) = \sum_{Z} \gamma(z_{nk}) \ln p(X, Z|\theta^*) \tag{37}$$

Ainda falta $\mathrm{p}(\mathrm{X}, \mathrm{Z}|\ \theta^*)$. É apenas a probabilidade completa do modelo, incluindo X e Z, e podemos encontrá-la usando a seguinte expressão:

$$p(X, Z|\theta^*) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi^{z_{nk}} \mathcal{N}(x_n|\mu_k, \sum_k)^{z_{nk}} \tag{38}$$

Que é o resultado do cálculo da probabilidade conjunta de todas as observações e variáveis latentes e é uma extensão de nossas derivações iniciais para p(x). O log desta expressão é dado por:

$$\ln p(X, Z|\theta^*) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}[\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \sum_k)] \tag{39}$$

E finalmente nos livramos desse logaritmo problemático que afetou a soma em (29). Com tudo isso no lugar, será muito mais fácil estimar os parâmetros apenas maximizando Q em relação aos parâmetros, mas lidaremos com isso na etapa de maximização. Além disso, devemos levar em consideração que a variável latente z só será 1 uma vez toda vez que a soma for avaliada (BISHOP M CHRISTOPHER, 2006). Com esse conhecimento, podemos facilmente nos livrar dele conforme necessário para nossas derivações.

Finalmente, podemos substituir (39) em (37) para obter:

$$\mathcal{Q}(\theta^*, \theta) = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma(z_{nk})[\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \sum_k)] \tag{40}$$

Na etapa de maximização, encontraremos os parâmetros revisados da mistura. Para isso, precisaremos fazer de Q um problema de maximização restrita e, assim, adicionaremos um multiplicador de Lagrange. Vamos agora revisar a etapa de maximização.

$$\theta^* = \text{argmax}_\theta \, \mathcal{Q}(\theta^*, \theta) \tag{41}$$

Onde:

$$\mathcal{Q}(\theta^*, \theta) = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma(z_{nk})[\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \sum_k)] \tag{42}$$

Que é o que acabamos com na etapa anterior. No entanto, Q também deve levar em conta a restrição de que todos os valores de π devem somar um. Para fazer isso, precisaremos adicionar um multiplicador de Lagrange adequado. Portanto, devemos reescrever desta forma:

$$Q(\theta^*, \theta) = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})[\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \sum_k)] - \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right) \quad (43)$$

E agora podemos determinar facilmente os parâmetros usando a máxima verossimilhança. Vamos agora tomar a derivada de Q em relação a π e igualar a zero:

$$\frac{\partial Q(\theta^*, \theta)}{\partial \pi_k} = \sum_{n=1}^{N}\frac{\gamma(z_{nk})}{\pi_k} - \lambda = 0 \quad (44)$$

Então, reorganizando os termos e aplicando uma soma sobre *k* em ambos os lados da equação, obtemos:

$$\sum_{n=1}^{N}\gamma(z_{nk}) = \pi_k\lambda \implies \sum_{k=1}^{K}\sum_{n=1}^{N}\gamma(z_{nk}) = \sum_{k=1}^{K}\gamma(z_{nk}) = \pi_k\lambda \quad (45)$$

Sabemos que a soma de todos os coeficientes de mistura π é igual a um. Além disso, sabemos que somar as probabilidades γ sobre k também nos dará 1. Assim, obtemos $\lambda = N$. Usando este resultado, podemos resolver para π:

$$\pi_k = \frac{\sum_{n=1}^{N}\gamma(z_{nk})}{N} \quad (46)$$

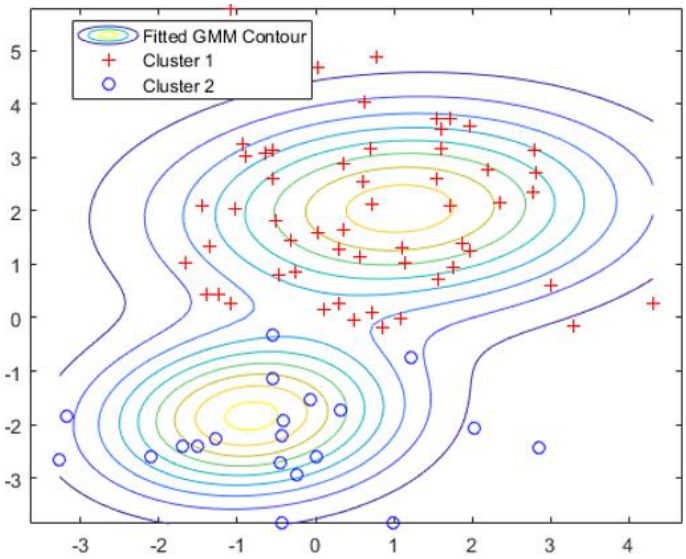
Da mesma forma, se diferenciarmos Q em relação a μ e Σ, igualarmos a derivada a zero e, em seguida, resolvermos os parâmetros usando a equação de probabilidade logarítmica (28.1) que definimos (BISHOP M CHRISTOPHER, 2006), obtemos:

$$\mu_k^* = \frac{\sum_{n=1}^{N}\gamma(z_{nk})x_n}{\sum_{n=1}^{N}\gamma(z_{nk})}, \qquad \sum_k^* = \frac{\sum_{n=1}^{N}\gamma(z_{nk})(x_n-\mu_k)(x_n-\mu_k)^T}{\sum_{n=1}^{N}\gamma(z_{nk})} \quad (47)$$

Em seguida, usaremos esses valores revisados para determinar γ na próxima iteração EM e assim por diante, até vermos alguma convergência no valor de verossimilhança. Podemos usar a equação (29) para monitorar a probabilidade logarítmica em cada etapa e sempre temos a garantia de atingir um máximo local (BISHOP M CHRISTOPHER, 2006).

Na figura 48 mostra a aplicação do algoritmo *Mixture of Gaussians*.

Figura 48: Aplicação do algoritmo *Mixture of Gaussians*.



Fonte: MAKLIN (2019)

3.3 COMPARAÇÃO DOS ALGORITMOS DE CLUSTERIZAÇÃO

O quadro 3 mostra as vantagens e desvantagens de algoritmo de clusterização, a tabela está construída baseada em fórmulas matemáticas de probabilidade, pois os algoritmos tem como princípio, o modelo de distribuição, conectividade, centroides, densidade e similaridade. Além disso, a tabela apresenta se o algoritmo tem a necessidade de definir o número de cluster k, e o grau de complexidade do algoritmo.

Quadro 3 - Comparação dos algoritmos de clusterização.

Algoritmo	Vantagem	Contra
Affinity Propagation	Definir o número de clusters: **Não** Principios de similaridade Usada em formato matricial Equação base: $a(k,k) \leftarrow \sum_{i'tal\ que\ i'\neq k} \max\{0, r(i',k)\}$	É inferior ao algoritmo *K-Means e DBSCAN*
Hierarchical clustering	Definir o número de clusters: **Não** Principios de similaridade Funciona bem em qualquer situação de distribuição dos pontos. Equação base: $L(r,s) = \min(D(x_{ri}, x_{si}))$ $L(r,s) = \max(D(x_{ri}, x_{si}))$ $L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{si})$	Porém sua complexidade não é linear, sendo $O(n^3)$.
BIRCH	Definir o número de clusters: **Não** Agrupamento hierárquico Principios de similaridade Grande conjuntos de dados. Equação base: $x_0 = \frac{1}{n} \sum_{i=1}^{n} x_i$	Só processa atributos métricos, sendo assim só é possivel aplicar se o valores poderem ser representados no espaço euclidiano, desta forma nenhum atributo categórico deve estar presente.
DBSCAN	Definir o número de clusters: **Não** Conexão por Densidade Aplicável em diferentes tipos de distribuição de dados. Equação base: $N_{\epsilon}(p) = \{q\ em\ D |\ dist(p,q) < \varepsilon\}.$	Não funciona tão bem quando a densidade dos dados é variável. Pórem é um algoritmo de muito sucesso.

K-Means	Definir o número de clusters: **Sim** Modelos de centroídes e semelhanças. Fácil implementação Equação base: $$F_s(P) = \sum_{i=1}^{k} \sum_{x_j \in C_i} D(x_j, y_i)$$ $$i^* = \underset{i=1,2,\ldots,k}{\operatorname{argmin}} \|x_j - y_i\|_2^2$$	É preciso estimar o número de cluster para aplicar o algoritmo, além de ser lento para grandes conjuntos de dados, e poder convergir para mínimos locais.

Mini-Batch K-Means	Definir o número de clusters: **Sim** Modelos de centroídes e semelhanças. Fácil implementação Grande número de clusters e de dados. Equação base: $$F_s(P) = \sum_{i=1}^{k} \sum_{x_j \in C_i} D(x_j, y_i)$$ $$i^* = \underset{i=1,2,\ldots,k}{\operatorname{argmin}} \|x_j - y_i\|_2^2$$	Não é tão eficiente para pequenos grupos de dados.
Mean Shift	Definir o número de clusters: **Não** Conexão por Densidade Modelos Centróides Equação base: $$y_i = \frac{1}{n_i} \sum_{x_j \in C_i} x_j,$$	A seleção do tamanho da janela/raio "r" pode não ser trivial, isto é, se os parâmetros não definidos com um valor próxima do ideal, os clusters não saem como esperado.
OPTICS	Definir o número de clusters: **Não** Conexão por Densidade Gráfico de acessibilidade Equação base: $$dist(p, q) < \varepsilon.$$	A complexidade de tempo do algoritmo é $O(n^2)$. Para bases com milhares de dados, a dimensionalidade, faz com que os pontos se tornem esparsos e se perca regiões densas que caracterizam grupos.
Spectral Clustering	Definir o número de clusters: **Não** Aplicável para conjuntos pequenos. Usada em formato matricial Particionamento de grafos. Equação base: $$D_{ij} = \begin{cases} d_i, i = j \\ 0, i \neq j \end{cases}$$	Não recomendável para conjuntos de dados grandes. Pois como não são escalares, A construção de matrizes e cálculo de autovalores e autovetores, podem ser demoradas.

	$L = D - A$	
Mixture of gaussians	Definir o número de clusters: **Não** Mistura Gaussiana Probabilidade de mistura π Dimensão $\mathbb{R}^2$ Equação base: $\mathcal{N}(x\|\mu,\sum) = \frac{1}{(2\pi)^{\frac{D}{2}}\|\sum\|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(x-\mu)^T\sum^{-1}(x-\mu)\right)$ $\sum_{k=1}^{K}\pi_k = 1$	Convergência lenta. Converge para o ótimo local só.

Fonte: Autoria própria (2022)

4 APLICAÇÃO DOS ALGORITMOS E SIMULAÇÃO

Conforme foi mostrado na tabelas 3 e 3.1 cada algoritmo tem suas características específicas, e desta forma cada um tem sua aplicação para determinando tipo de distribuição dos dados, ou seja, não existe algoritmo ruim, cada um deles tem suas vantagens e desvantagens, sendo melhores aplicado em determinados tipos de distribuição. Para a aplicação em um sistema de radar automotivo, os algoritmos com as caracteristicas semelhantes ao tipo de distribuição dos dados (points clouds) gerados pelo radar são: *K-Means, Mean Shift e DBSCAN.*

Para todas as simulações, será utilizado o *MATLAB R2021a*, pois o mesmo tem a opção de gerar código e também a opção de transferir para o diagrama de blocos do *Simulink*, o que facilita muito na hora de aplicar em uma situação real na indústria automotiva, porém a aplicação da clusterização em um conjunto de dados não relacionados com a indústria automotiva, pode ser feita em outros programas e linguagens de programação, para realização da simulação serão utilizados códigos disponibilizados no GitHub.

4.1.1 Simulação *K-Means*

O algoritmo *K-Means* é um dos mais utilizados, pois é simples de entender e fácil de aplicar, para a simulação em questão foi colocado aleatoriamente um grupo de pessoas com diferentes idades e a sua renda, desta forma em cima desse grupo foi aplicado o algoritmo de agrupamento *K-means.*

O quadro 4 mostra os dados utilizados na aplicação do algoritmo *K-means,* como o nome, idade e a renda possuída.

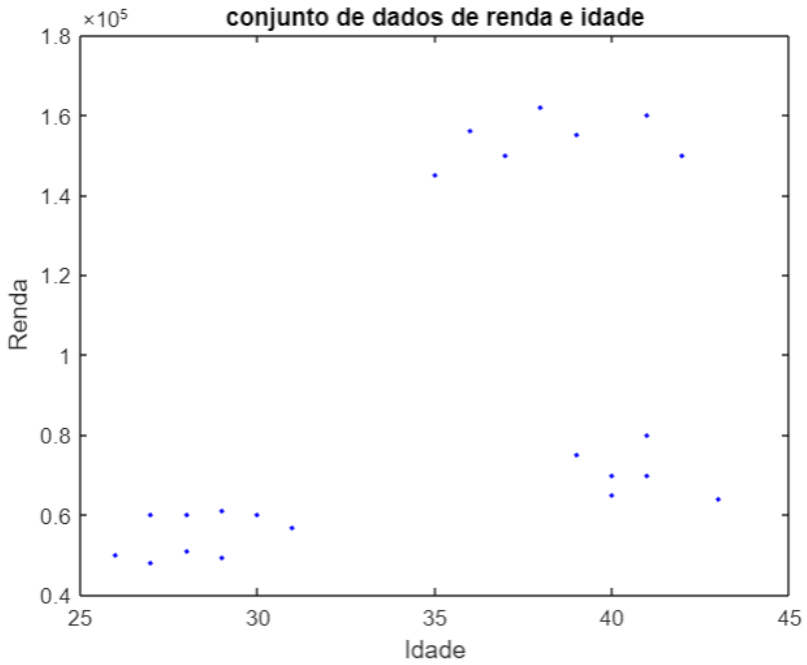
Quadro 4: Dados para aplicação do *K-means*

Nome	Idade	Renda
Daniel	27	60000
Michael	30	60000
Joao	29	61000
Carlos	28	60000
Gabriel	42	150000
Gustavo	39	155000
Davi	41	160000
Andrea	38	162000
Jose	36	156000
Angelina	35	145000
Donald	37	150000
Tom	26	50000
Arnold	27	48000
Isaque	28	51000
Lucas	29	49500
Rogerio	31	57000
Antônio	40	65000
Bruno	41	70000
Matheus	43	64000
Leonardo	39	75000
Rodrigo	41	80000
Claudio	40	70000

Fonte: Autoria própria (2022)

A primeira simulação é realizada, colocando o número de entrada de clusters igual a 1.A figura 49 mostra a distribuição dos dados.

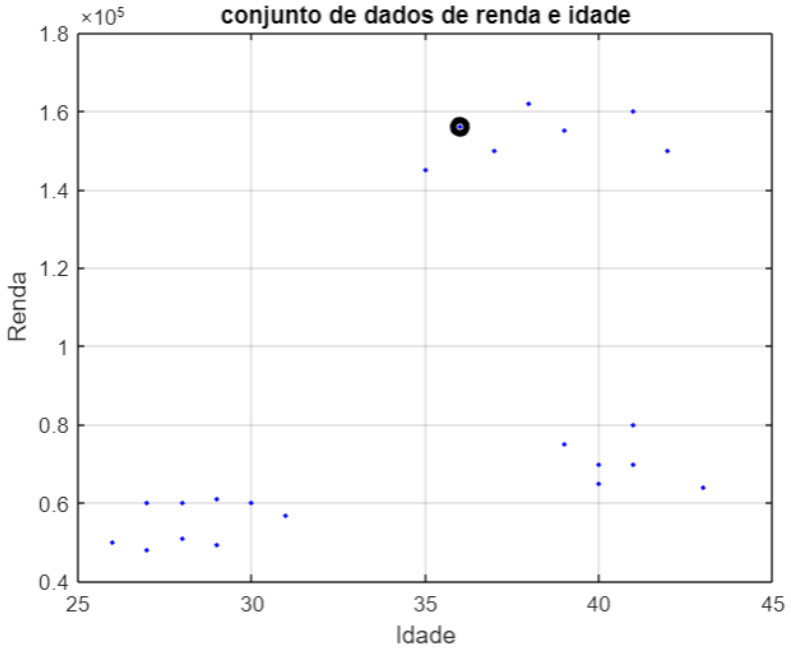
Figura 49: Distribuição dos dados



Fonte: Autoria própria (2022)

A figura 50 mostra o centroide para K=1.

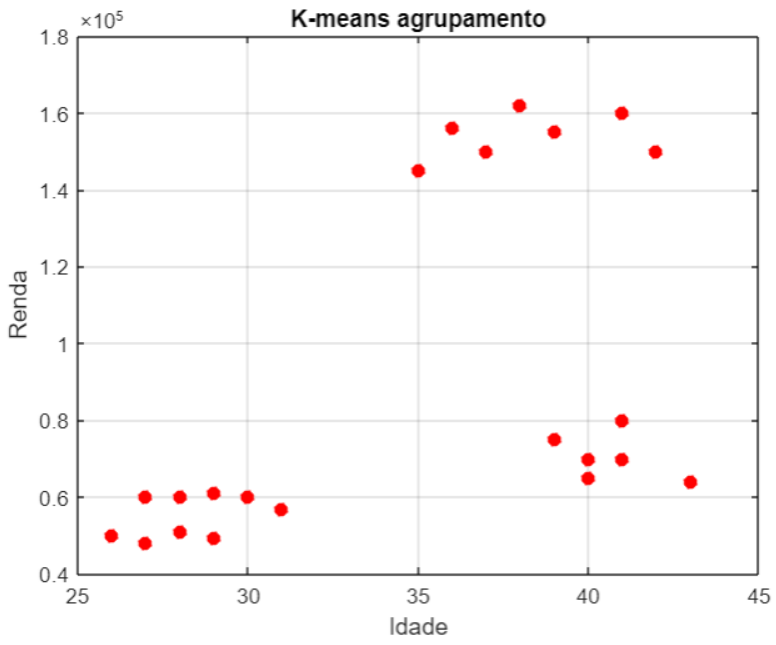
Figura 50: Centroide para K=1.



Fonte: Autoria própria (2022)

Na figura 51 mostra o resultado da aplicação do *K-Means* para K=1.

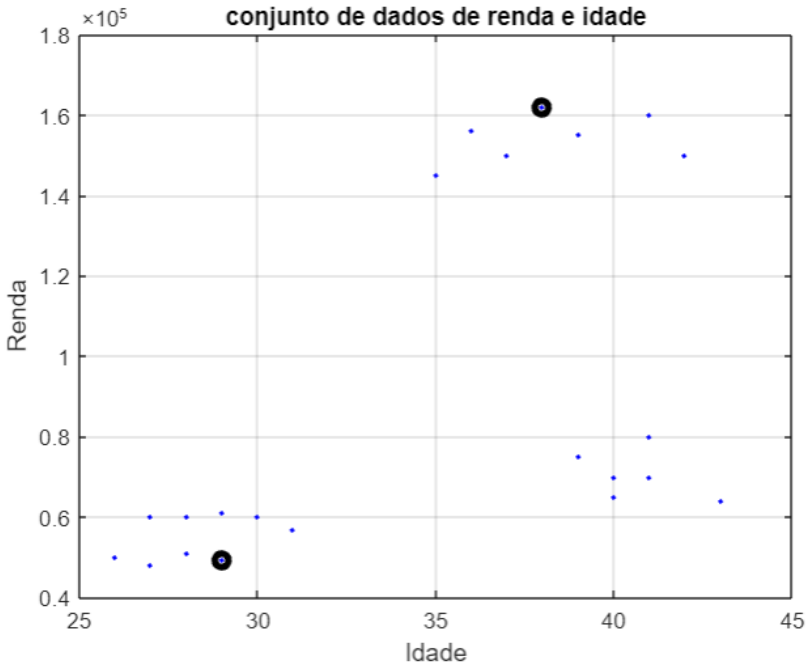
Figura 51: Resultado aplicação *K-means*, K=1



Fonte: Autoria própria.

A segunda simulação é realizada, colocando o número de entrada de clusters igual a 2.A figura 52 apresenta a localização dos centroides para K=2.

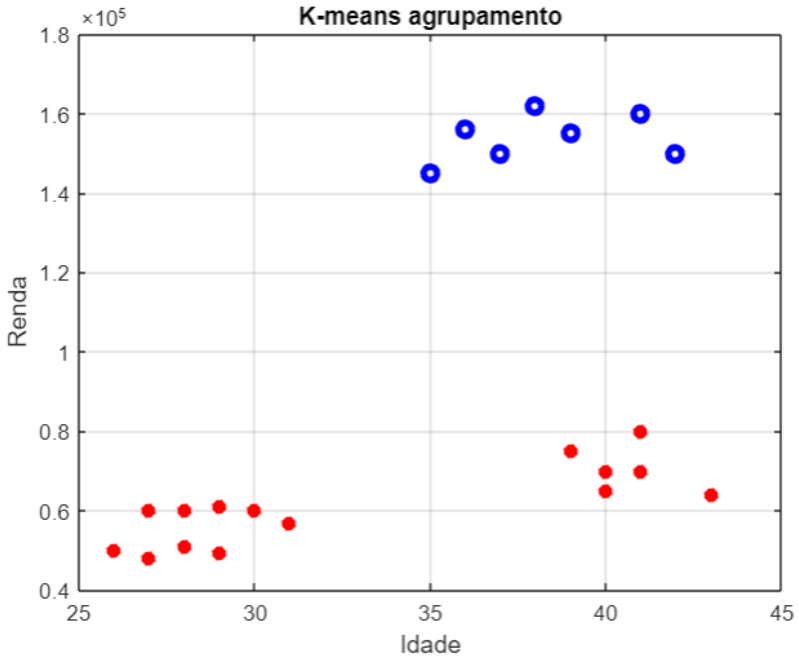
Figura 52: Centroides para K=2



Fonte: Autoria própria (2022)

Na figura 53 mostra o resultado da aplicação do *K-Means* para K=2.

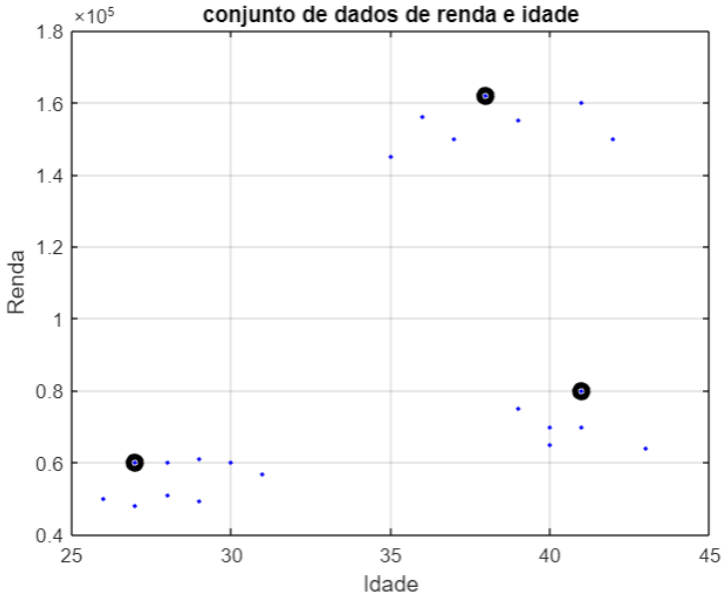
Figura 53: Resultado da aplicação *K-means,* K=2



Fonte: Autoria própria.

A terceira simulação é realizada, colocando o número de entrada de clusters igual a 2.A figura 54 mostra a localização dos centroides para K=3.

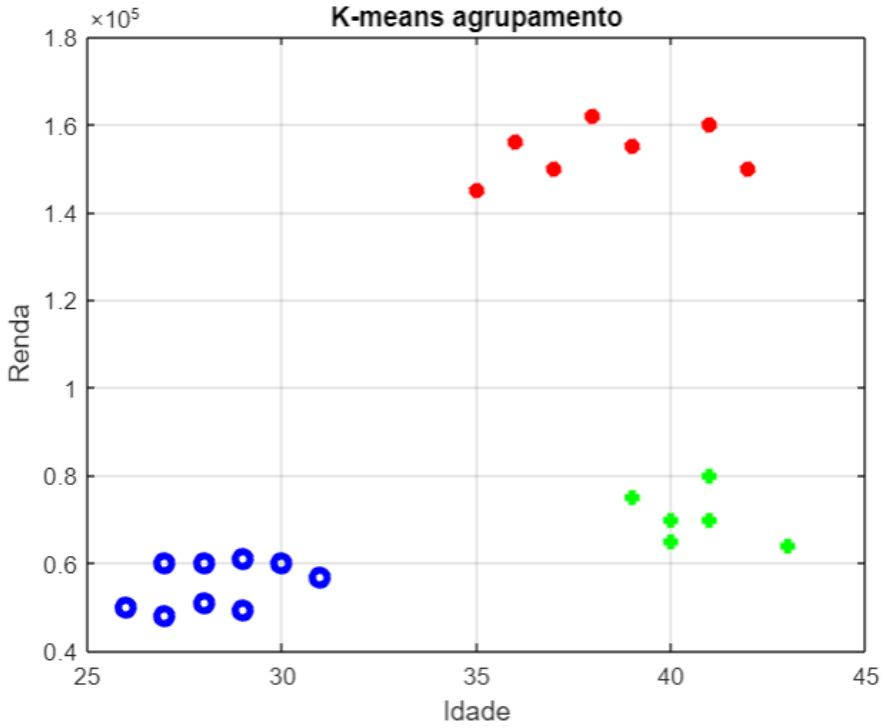
Figura 54: Centroides para K=3



Fonte: Autoria própria (2022)

Na figura 55 mostra o resultado da aplicação do *K-Means* para K=3.

Figura 55: Resultado da aplicação *K-means*, K=3.



Fonte: Autoria própria (2022)

4.1.2 Simulação Mean Shift

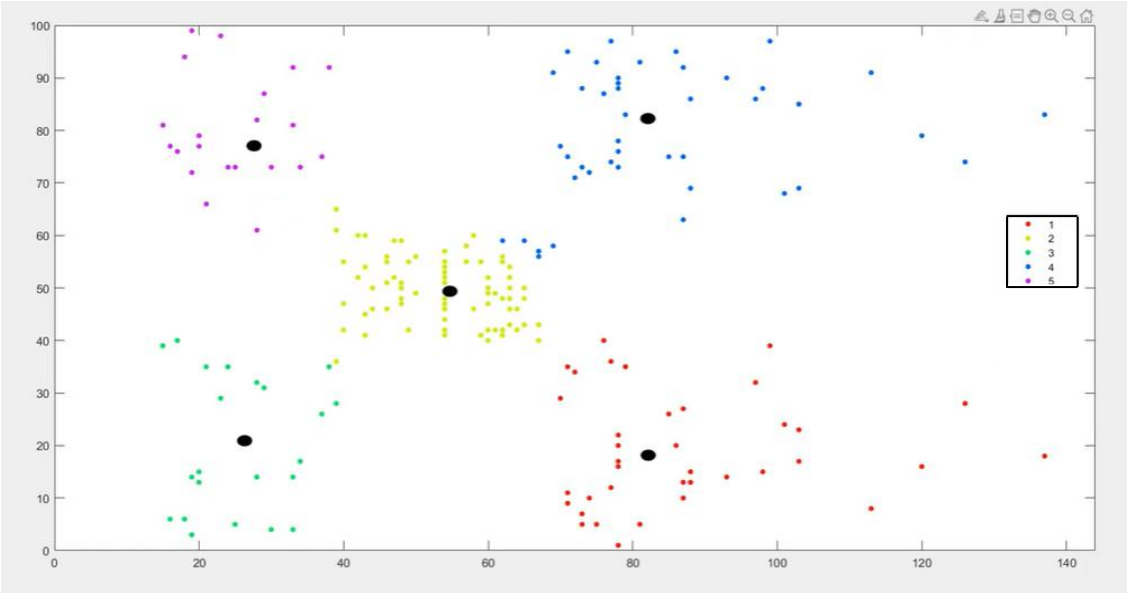
Como vimos na fundamentação teórica o algoritmo *Mean Shift* é outro algoritmo bastante interessante, a figura 56 mostra o conjunto de dados em que recebera a clusterização do algoritmo *Mean Shift*.

Figura 56: Conjunto de dados *Mean Shift*.

Fonte: Autoria própria (2022)

Na figura 57 vemos o resultado na aplicação do algoritmo Mean Shift no conjunto de dados mostrados acima.

Figura 57: Aplicação *Mean Shift*.



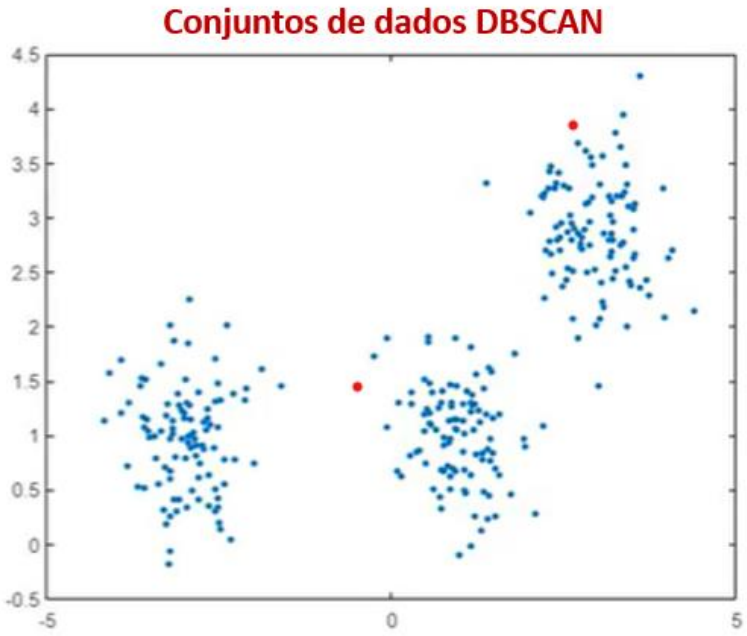
Fonte: Autoria própria (2022)

4.1.3 Simulação DBSCAN

Para realizar a simulação DBSCAN, o arquivo utilizado como o conjunto de dados, está disponível no GitHub, e para realizar a simulação é preciso instalar a *toolbox* DBSCAN, Função estatística e *Machine Learning Toolbox* DBSCAN.

A figura 58 mostra o conjunto de dados, em que o algoritmo DBSCAN será aplicado.

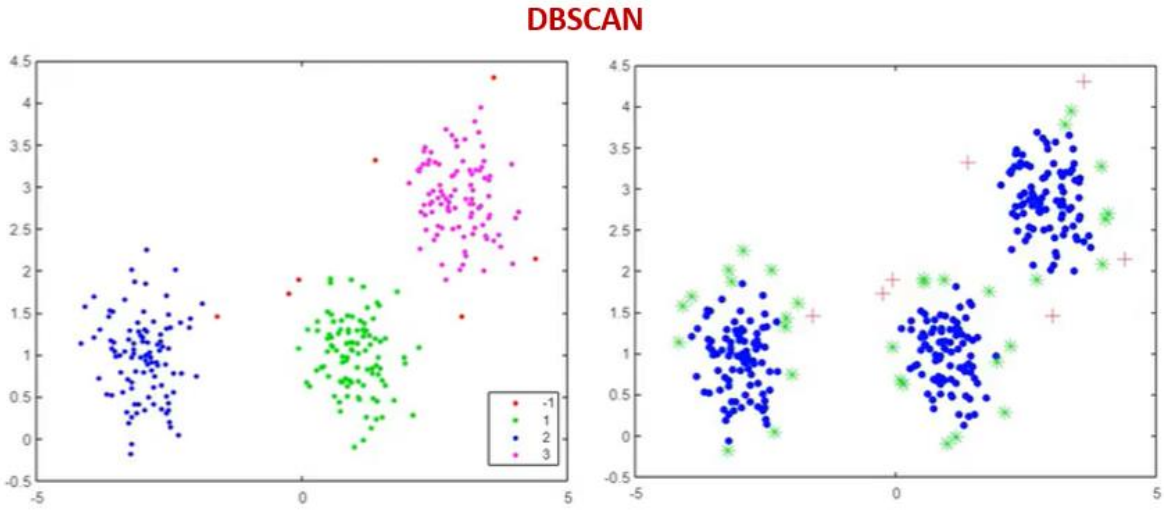
Figura 58: Conjunto de dados DBSCAN.



Fonte: Autoria própria (2022)

A figura 59 mostra a aplicação do método DBSCAN em um conjunto de dados.

Figura 59: Aplicação DBSCAN.

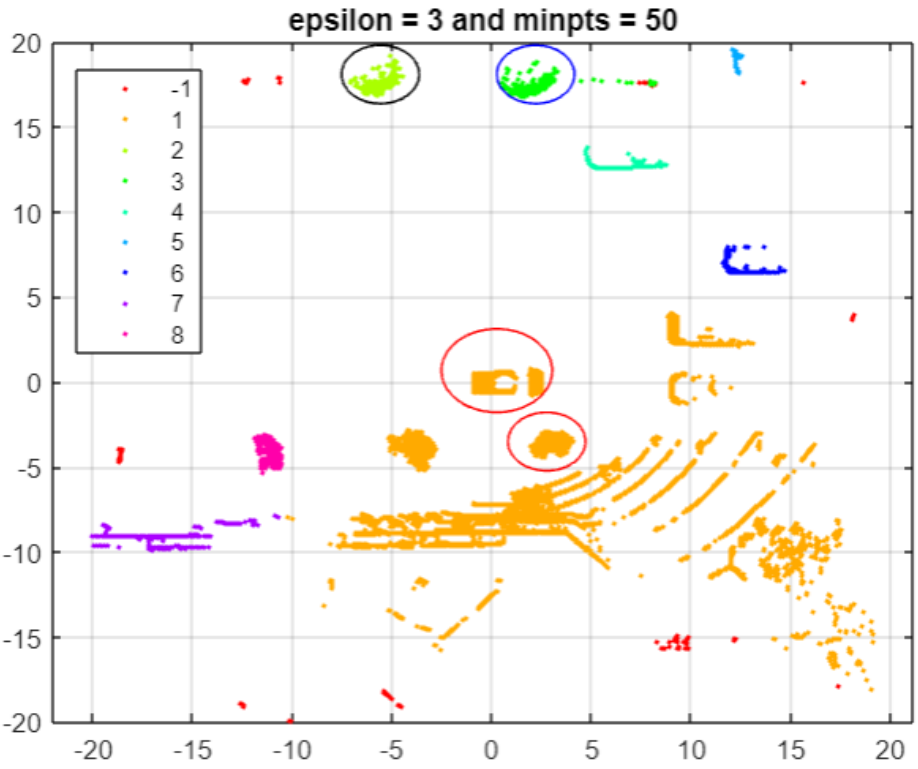


Fonte: Autoria própria (2022)

O algoritmo DBSCAN é baseado em densidade, e é muito interessante para aplicação em sistema automotivos, outra vantagem é que não precisamos estimar o número de clusters para a aplicação do algoritmo, vejamos como seria a aplicação DBSCAN, em um sistema de radar, exemplo que encontramos no MATLAB R2021a, onde é fornecido um arquivo com nuvens de pontos de um sensor e é feita a clusterização, é um modelo bastante próximo para a aplicação em veículos autônomos, porém para aplicar em um sistema de radar real seria preciso fazer mais algumas etapas descritas na Figura 21, para que o reconhecimento de objetos fosse feito com sucesso.

É visível na figura 60 que a clusterização não sai como o esperado dependendo do valor de épsilon, veja a seguir o valor para épsilon = 3.

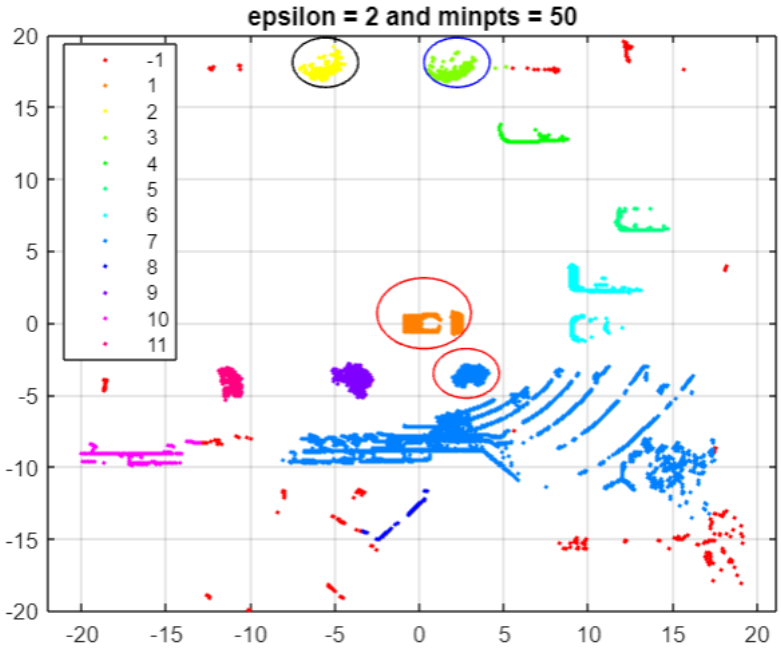
Figura 60: Exemplo DBSCAN, épsilon=3.



Fonte: Autoria própria (2022)

Com épsilon = 2 o resultado é superior ao da figura 60, como mostrado na figura 61.

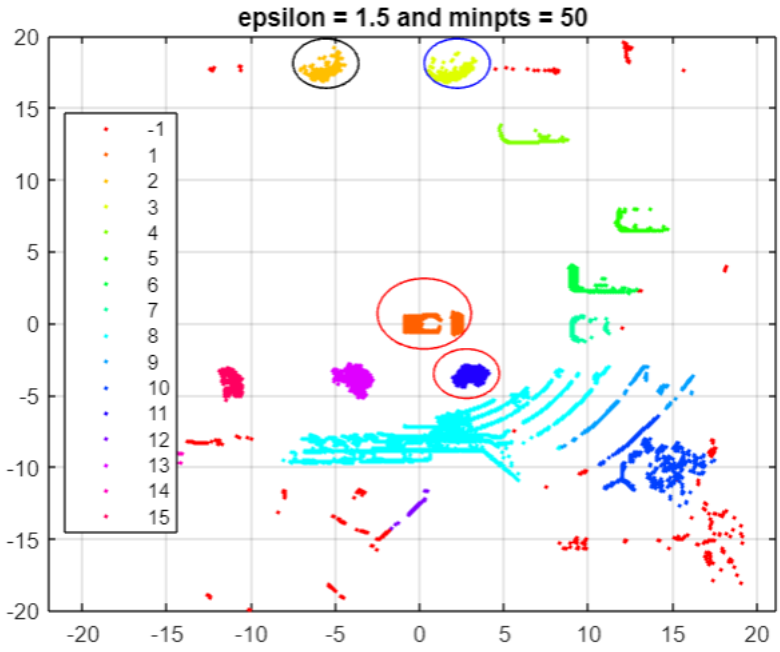
Figura 61: Exemplo DBSCAN, épsilon=2.



Fonte: Autoria própria (2022)

O resultado mais próximo do real, com épsilon=1.5 é mostrado na figura 62.

Figura 62: Exemplo DBSCAN, épsilon=1.5.



Fonte: Autoria própria (2022)

5 APLICAÇÃO DE ALGORITMO DE CLUSTERIZAÇÂO PARA RECONHECIMENTO DE OBJETOS

Nesta parte do trabalho, faremos algumas simulações na ferramenta Driving Scenário Design do MATLAB, mostrando como o radar detecta os objetos a sua frente, como pedestre, ciclista, um veículo parado e em movimento, veremos também como o radar detecta vários objetos ao mesmo tempo, isto é pedestre, ciclista, veículo juntos, para isso utilizaremos o MATLAB R2021a. Será feita a comparação entre os 3 algoritmos, *K-Means, Mean Shift* e DBSCAN.

Para vamos adotar as seguintes nomenclaturas:

- Caminhão – T → *Truck*
- Carro – V → *Vehicle*
- Motocicleta – M → *Motorcycle*
- Pedestre – P → *Pedestrian*
- Ciclista – C → *Ciclist*

Não será considerado as condições de tempo, manhã, tarde, noite, chuva fraca, chuva forte e neblina, essas condições serão consideradas em trabalhos futuros com aplicação de MIMO radar.

No quadro 5 é mostrado os cenários de testes que serão realizados, os cenários de testes serão feitos com caminhão, carro, motociclista, pedestre e ciclista, partindo com a análise de três algoritmos de clusterização, *K-Means, Mean Shift,* DBSCAN.

Quadro 5. Cenários de testes

	Caminhão	Carro	Motocicleta	Pedestre	Ciclista
K-Means	SCE-1.1	SCE-1.2	SCE-1.3	SCE-1.4	SCE-1.5
Mean Shift	SCE-2.1	SCE-2.2	SCE-2.3	SCE-2.4	SCE-2.5
DBSCAN	SCE-3.1	SCE-3.2	SCE-3.3	SCE-3.4	SCE-3.5

Onde SCE → Cenário.

As métricas para avaliar o desempenho e comparação entre os algoritmos foram listadas tomando como referência os órgãos de regulamentação de segurança automotiva, como a Euro NCAP e a Latin NCAP, as métricas são as seguintes:

a) Distância em que identificam e reconhecem objetos. (Quanto antes reconhecer, melhor o algoritmo).

b) Qual distância real dos objetos. (Quando mais longe estiverem os objetos melhor)

c) Velocidade dos objetos. (Caso o objeto esteja em baixa velocidade, o nosso veículo autônomo

saberá e terá melhor tempo pra reação caso possa haver uma colisão).

O teste será o mesmo para todos, o radar usado na simulação terá o range de 50 m, será usado o teste do AEB de 50 metros, disponível no Driving Scenario Design no caminho:

Open >PrebuiltScenario>

C:\ProgramFiles\MATLAB\R2021a\toolbox\shared\drivingscenario\PrebuiltScenarios.

Quadro 5.1 Avaliação de Desempenho para K-Means

Velocidade Veículo Autônomo	Métricas De Avaliação	SCE-1.1	SCE-1.2	SCE-1.3	SCE-1.4	SCE-1.5
20km/h	a)	45 m	45 m	40 m	45 m	45 m
	b)	45 m	45 m	40 m	45 m	45 m
	c)	Parado	Parado	Parado	Parado	Parado
40km/h	a)	40 m	40 m	40 m	45 m	42 m
	b)	40 m	40 m	40 m	45 m	42 m
	c)	40 km/h	40 km/h	40 km/h	2 km/h	20 km/h
60km/h	a)	40 m	40 m	40 m	45 m	40 m
	b)	40 m	40 m	40 m	45 m	40 m
	c)	60 km/h	60 km/h	60 km/h	4 km/h	30 km/h

Fonte: Autoria própria (2022)

Quadro 5.2 Avaliação de Desempenho para Means Shift

Velocidade Veículo Autônomo	Métricas De Avaliação	SCE-2.1	SCE-2.2	SCE-2.3	SCE-2.4	SCE-2.5
20km/h	a)	40 m	40 m	40 m	40 m	40 m
	b)	40 m	40 m	40 m	40 m	40 m
	c)	Parado	Parado	Parado	Parado	Parado
40km/h	a)	40 m	40 m	40 m	40 m	40 m
	b)	40 m	40 m	40 m	40 m	40 m
	c)	40 km/h	40 km/h	40 km/h	2 km/h	40 km/h
60km/h	a)	40 m	40 m	40 m	40 m	40 m
	b)	40 m	40 m	40 m	40 m	40 m
	c)	60 km/h	60 km/h	60 km/h	4 km/h	30 km/h

Fonte: Autoria própria (2022)

Quadro 5.3 Avaliação de Desempenho para DBSCAN

Velocidade Veículo Autônomo	Métricas De Avaliação	SCE-3.1	SCE-3.2	SCE-3.3	SCE-3.4	SCE-3.5
20km/h	a)	47 m	47 m	45 m	45 m	45 m
	b)	47 m	47 m	45 m	45 m	45 m
	c)	Parado	Parado	Parado	Parado	Parado
40km/h	a)	47 m	47 m	45 m	45 m	45 m
	b)	47 m	47 m	45 m	45 m	45 m
	c)	40 km/h	40 km/h	40 km/h	2 km/h	20 km/h
60km/h	a)	47 m	47 m	45 m	45 m	45 m
	b)	47 m	4 m	45 m	45 m	45 m
	c)	60 km/h	60 km/h	60 km/h	4 km/h	30 km/h

Fonte: Autoria própria (2022)

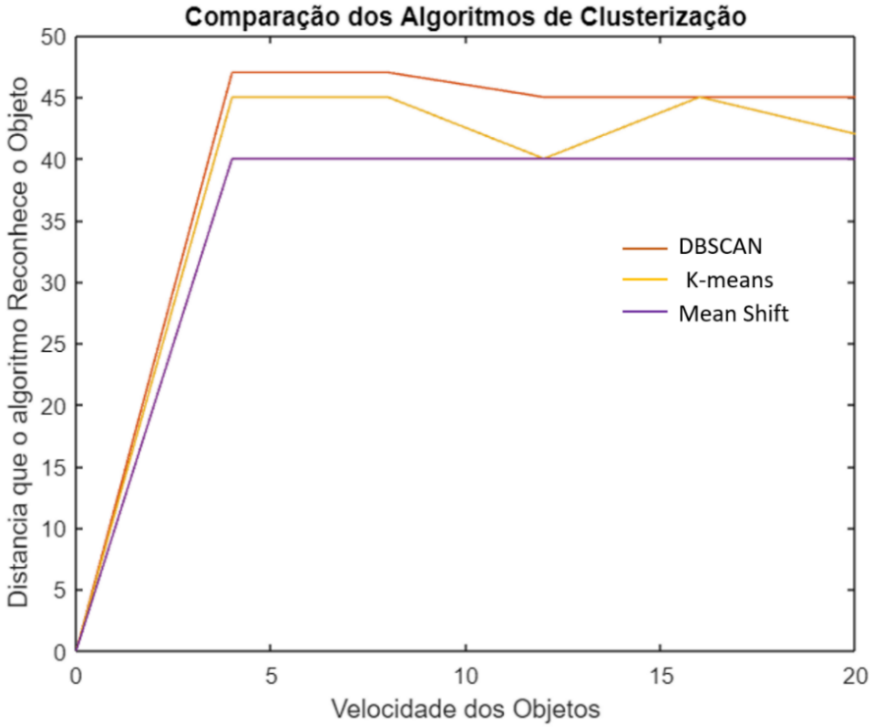
Quadro 5.4 Melhor Cenário de cada algoritmo

Velocidade Veículo Autônomo/ Algortimo	Métricas De Avaliação	SCE-1.1 SCE-2.1 SCE-3.1	SCE-1.2 SCE-2.2 SCE-3.2	SCE-1.3 SCE-2.3 SCE-3.3	SCE-1.4 SCE-2.4 SCE-3.4	SCE-1.5 SCE-2.5 SCE-3.5
20km/h DBSCAN	a)	47 m	47 m	45 m	45 m	45 m
	b)	47 m	47 m	45 m	45 m	45 m
	c)	Parado	Parado	Parado	Parado	Parado
20km/h *K-Means*	a)	45 m	45 m	40 m	45 m	42 m
	b)	45 m	45 m	40 m	45 m	42 m
	c)	Parado	Parado	Parado	Parado	Parado
20km/h *Mean shift*	a)	40 m	40 m	40 m	40 m	40 m
	b)	40 m	40 m	40 m	40 m	40 m
	c)	Parado	Parado	Parado	Parado	Parado

Fonte: Autoria própria (2022)

A figura 63 mostra o gráfico da análise comparativa do melhor desempenho de cada algoritmo, o cenário do melhor desempenho dos 3 algoritmos foi com o objeto parado, e os veículos com os radares a 20 km/h.

Figura 63: Gráfico Comparativo dos algoritmos de Clusterização

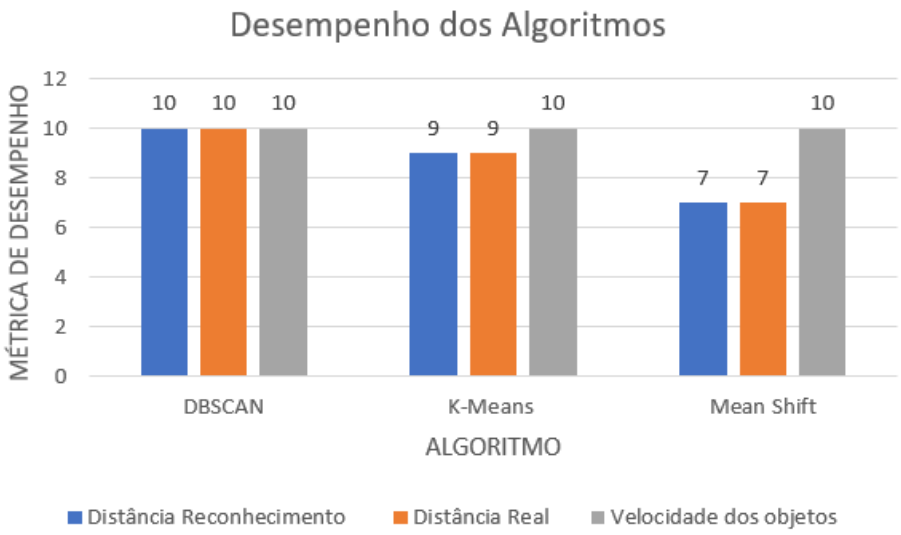


Fonte: Autoria própria (2022)

Observando o gráfico e os resultados apresentados nos quadros, o algoritmo com melhor desempenho no reconhecimento de objetos segundo as métricas de avaliação, é o DBSCAN, como já vimos na fundamentação teórica por ser um algoritmo de densidade, é um algoritmo de bastante sucesso para diversas aplicações, inclusive aplicações em sistemas de radares automotivos para o reconhecimento de objetos.

A Figura 64 mostra o desempenho dos algoritmos baseado nas notas (0 a 10), atribuídas a cada métrica.

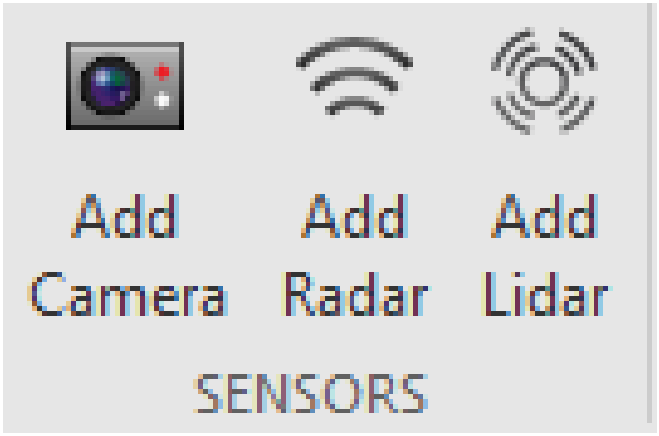
Figura 64: Gráfico do desempenho dos algoritmos.



Fonte: Autoria própria (2022)

A figura 65 mostra a aba que é utilizada para adicionar sensores no Driving Scenário.

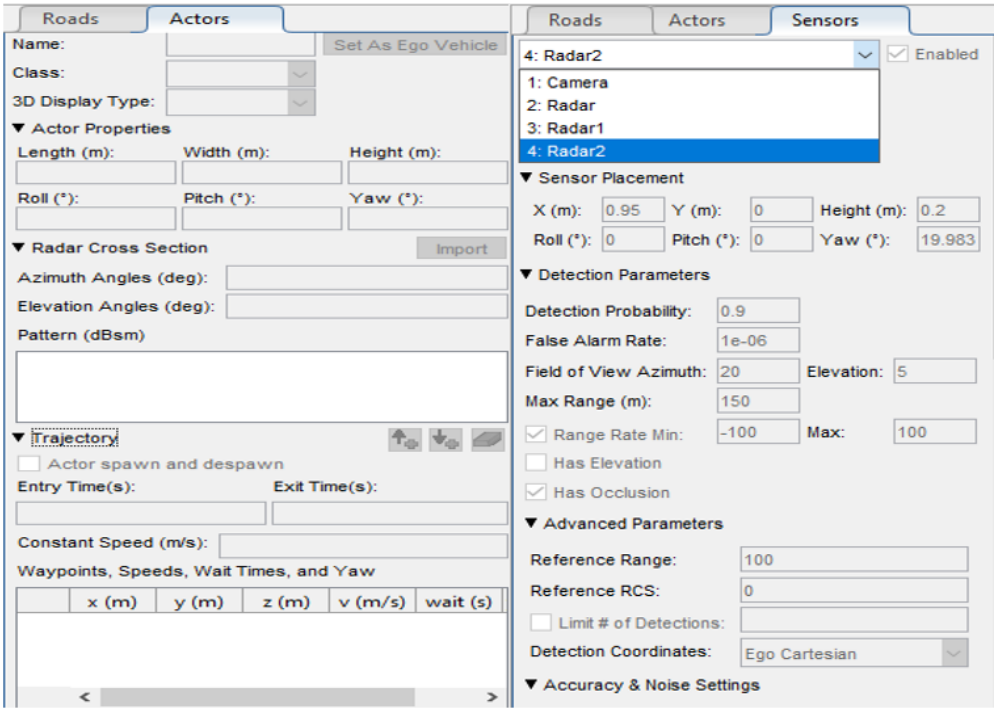
Figura 65: Aba de sensores Driving Scenário



Fonte: Autoria própria (2022)

A figura 66 mostra a aba *Actors* e *Sensors* utilizadas para alterar os parâmetros dos sensores que é utilizada para adicionar sensores e ver quais sensores estão presentes na simulação na ferramenta Driving Scenário.

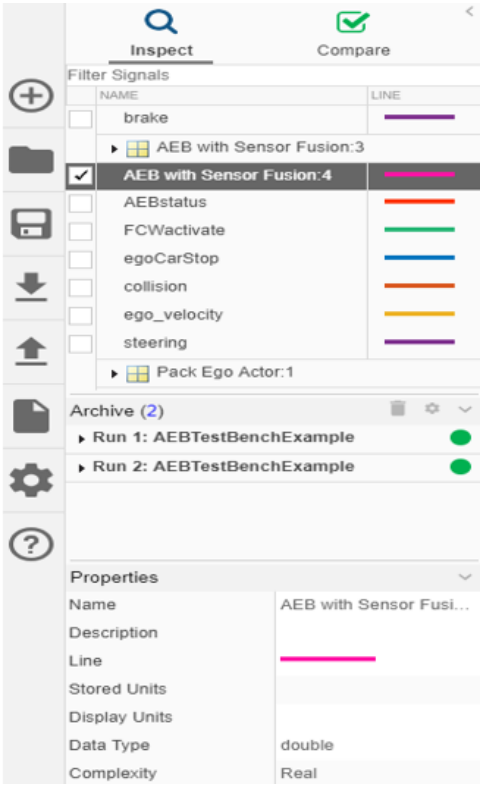
Figura 66: Aba de Actors e Sensors Driving Scenário



Fonte: Autoria própria (2022)

A figura 67 mostra a aba utilizadas para verificar os sinais presentes durante a simulação, nela é possível extrair todos os dados durante a simulação, como por exemplo, verificar o tempo de frenagem do AEB.

Figura 67: Aba de sinais da simulação Driving Scenário

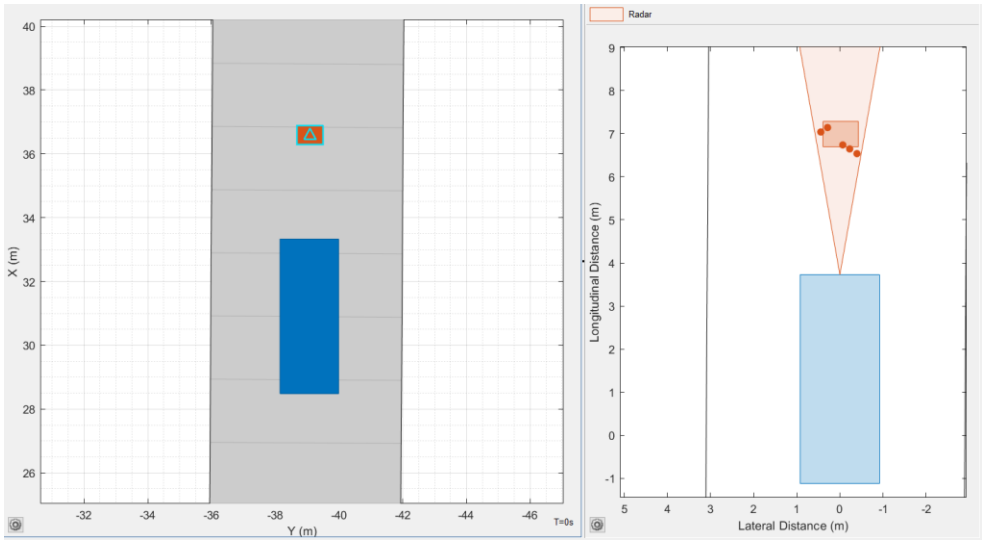


Fonte: Autoria própria (2022)

Abaixo será mostrado, como funciona a visão do radar para o reconhecimento de objetos, como pedestre, caminhão, carro, ciclista, em uma nuvem de pontos *(Point Clouds)* na simulação do Driving Scenario Design, dentro do MATLAB

Na figura 68 mostra como o radar detecta um pedestre a 3 metros distância, podemos ver que quando o veículo se aproxima do pedestre o radar detecta 5 pontos.

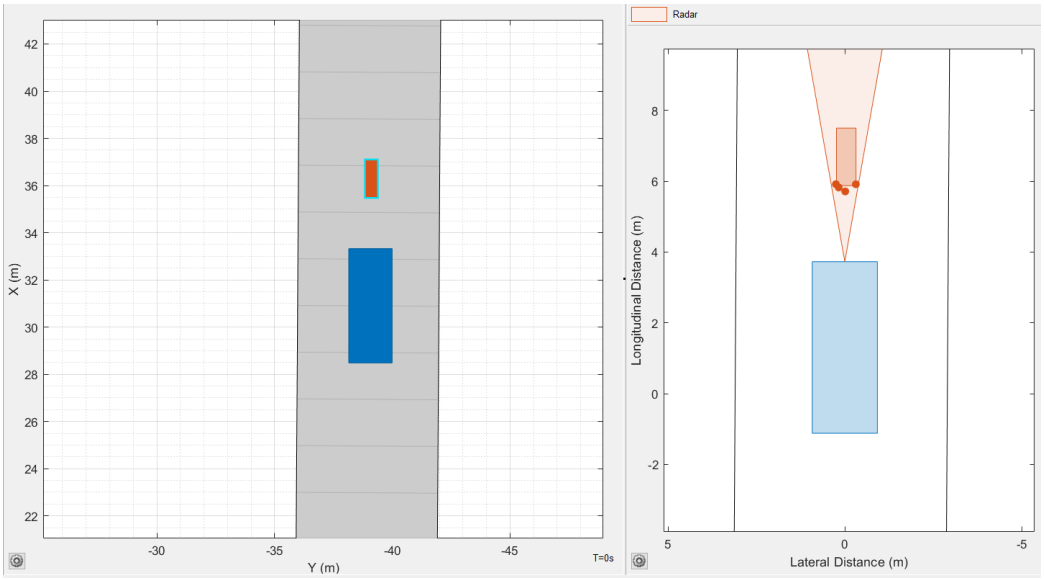
Figura 68: Radar detectando pedestre Driving Scenário Design



Fonte: Autoria própria (2022)

Na figura 69 mostra como o radar detecta um ciclista a 2 metros de distância, podemos ver que quando o veículo se aproxima do ciclista o radar detecta 4 pontos.

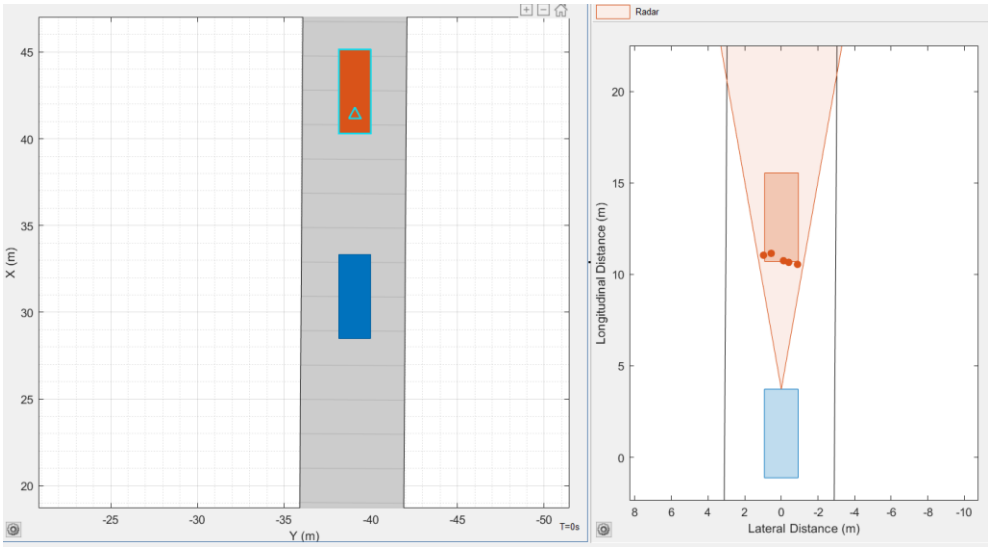
Figura 69: Radar detectando ciclista Driving Scenário Design.



Fonte: Autoria própria (2022)

Na figura 70 mostra como o radar detecta um veículo parado a 6 metros de distância, quando o veículo se aproxima do veículo parado o radar detecta 5 pontos.

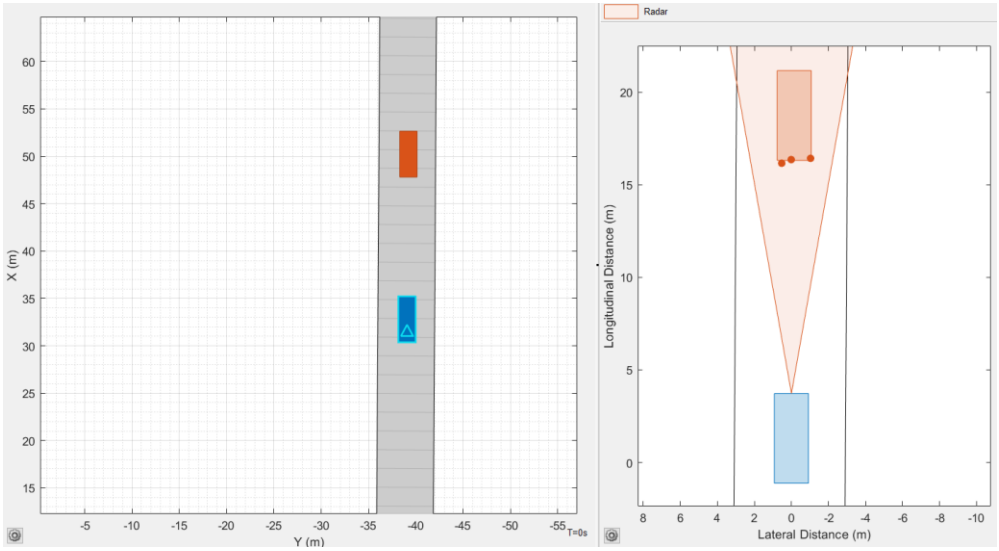
Figura 70: Radar detectando veículo parado Driving Scenário Design.



Fonte: Autoria própria (2022)

Na figura 71 mostra como radar detecta um veículo em movimento, quando o veículo se aproxima do veículo em movimento o radar detecta 3 pontos.

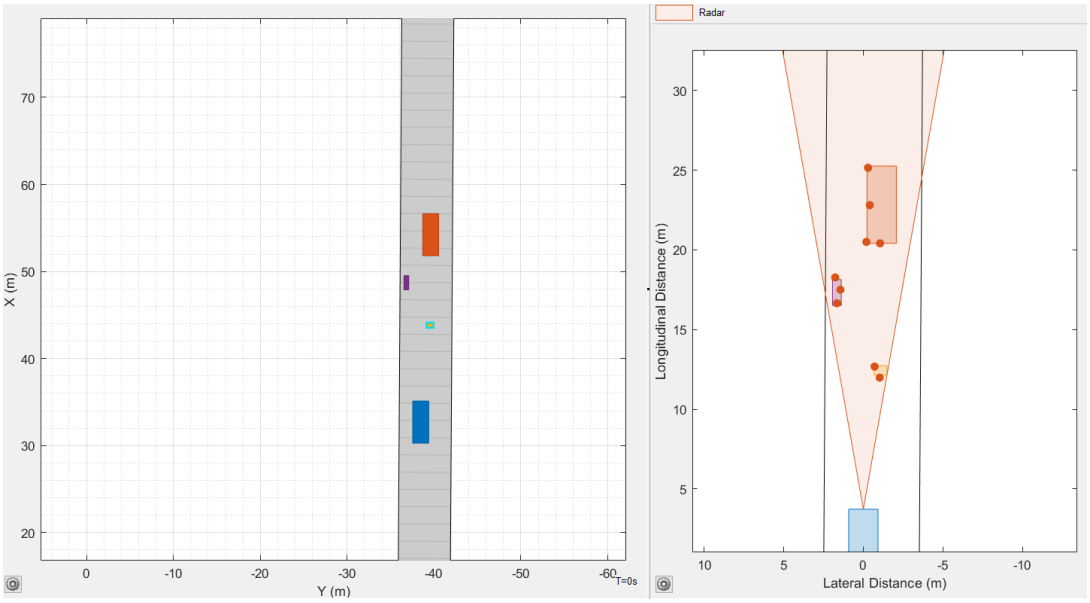
Figura 71: Radar detectando veículo em movimento Driving Scenário Design.



Fonte: Autoria própria (2022)

A figura 72 ilustra como o radar detecta um veículo parado, um ciclista e um pedestre, quando o veículo se aproxima dos objetos o radar detecta os pontos.

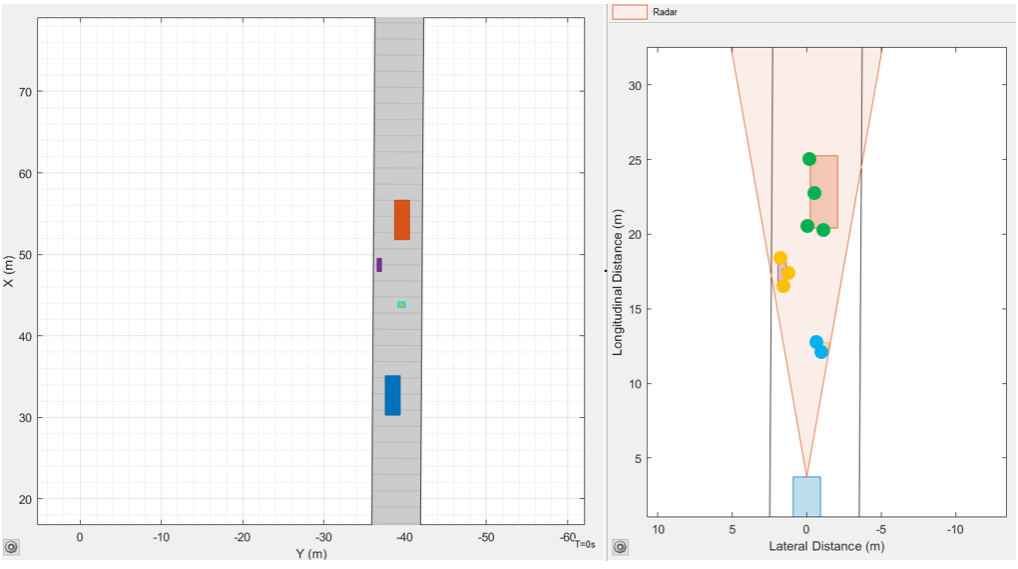
Figura 72: Radar detectando vários objetos Driving Scenário Design.



Fonte: Autoria própria (2022)

Na figura 73 é mostrado como o algoritmo DBSCAN, reconhece os pontos quando se trata de vários objetos, como o número de pontos é maior, é melhor para aplicação do algoritmo.

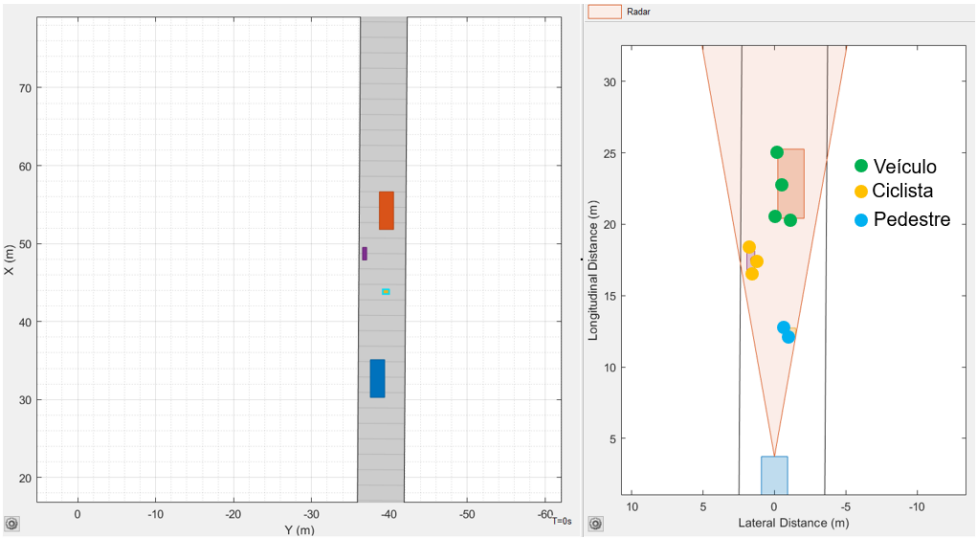
Figura 73: DBSCAN reconhecendo vários objetos.



Fonte: Autoria própria (2022)

Na figura 74 é apresentado como fica o reconhecimento dos objetos, feita pelo algoritmo DBSCAN.

Figura 74: DBSCAN reconhecendo e classificando os objetos.

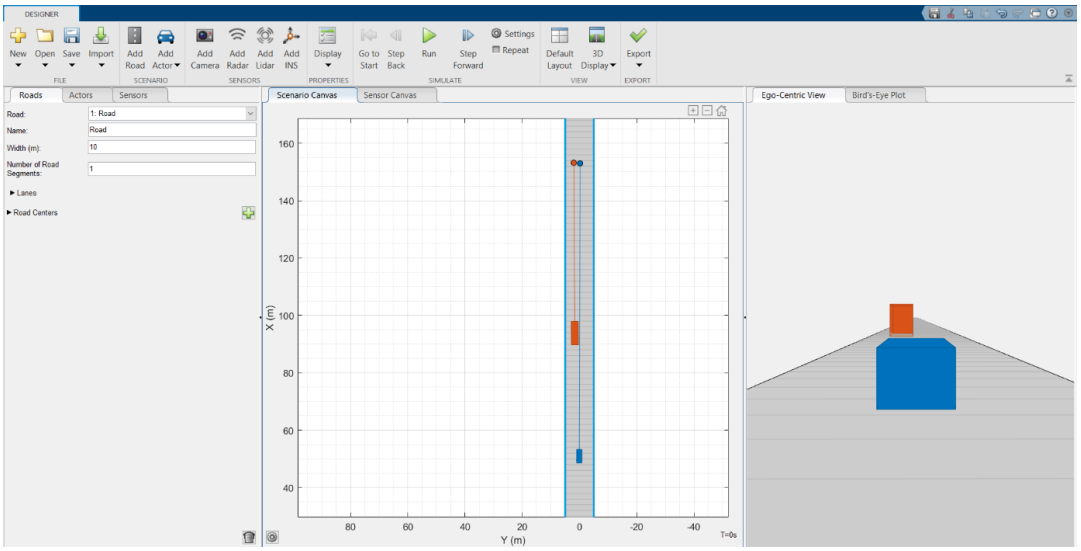


Fonte: Autoria própria (2022)

Quando mais distante está o objeto menos pontos o radar consegue capturar, e mais o objeto se aproxima, melhor o radar captura os pontos, o caminhão por exemplo quando está distante o radar capta 5 pontos pois o objeto é maior, e quando se aproxima o radar captura 7 ou 8 pontos.

Agora veremos como funciona a detecção de objetos pelo Driving Scenario Design do MATLAB. Primeiro é necessário carregar o Cenário desejado e após isso, colocar o radar no veículo como mostra a figura 75.

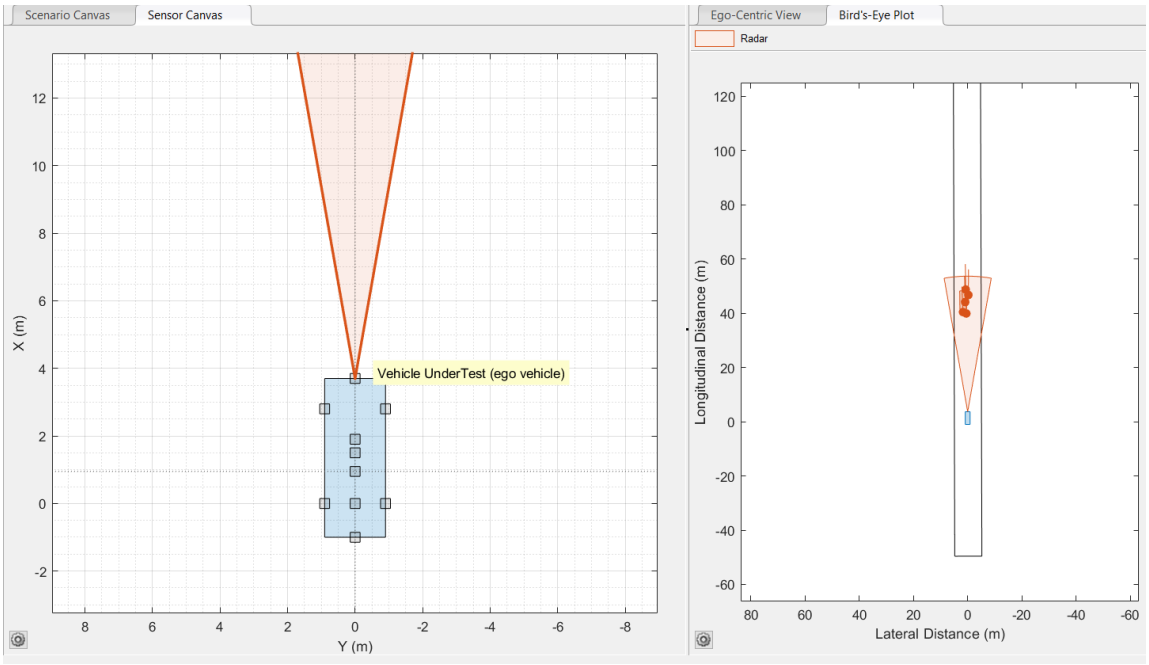
Figura 75: Carregando o Cenário Driving Scenario MATLAB



Fonte: Autoria própria (2022)

A figura 76 mostra onde podemos colocar o radar no veículo, o recomendado é sempre na parte frontal do veículo, para ter maior alcance dos objetos a sua frente.

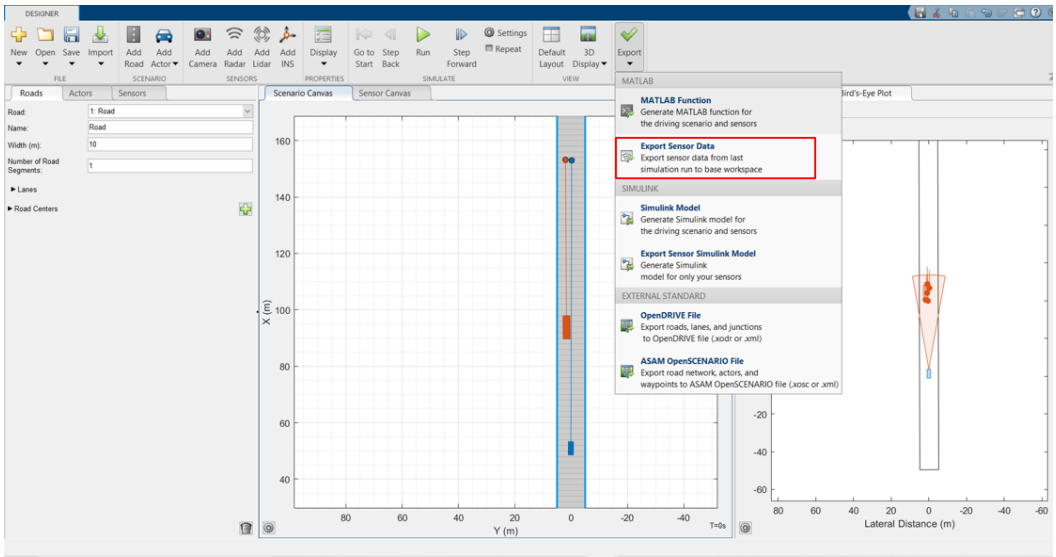
Figura 76: Colocando o radar no veículo.



Fonte: Autoria própria (2022)

Após isso, inicie a simulação em ***"RUN",*** rode o cenário de teste e depois exporte o arquivo no botão ***"Export"*** em seguida ***Export*** **Sensor Data,** figura 77.

Figura 77: Exportando o arquivo para o MATLAB.

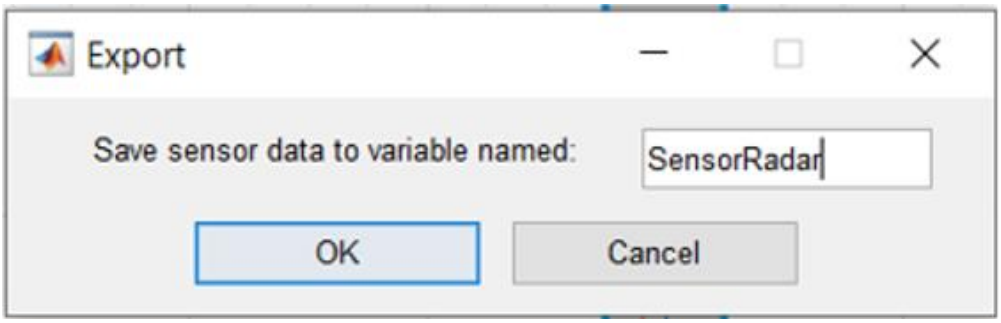


Fonte: Autoria própria (2022)

Após isso, de um nome ao arquivo que será exportado, depois disso no *Command*

Window no MATLAB digite o nome dado ao arquivo, como mostrado na figura 78.

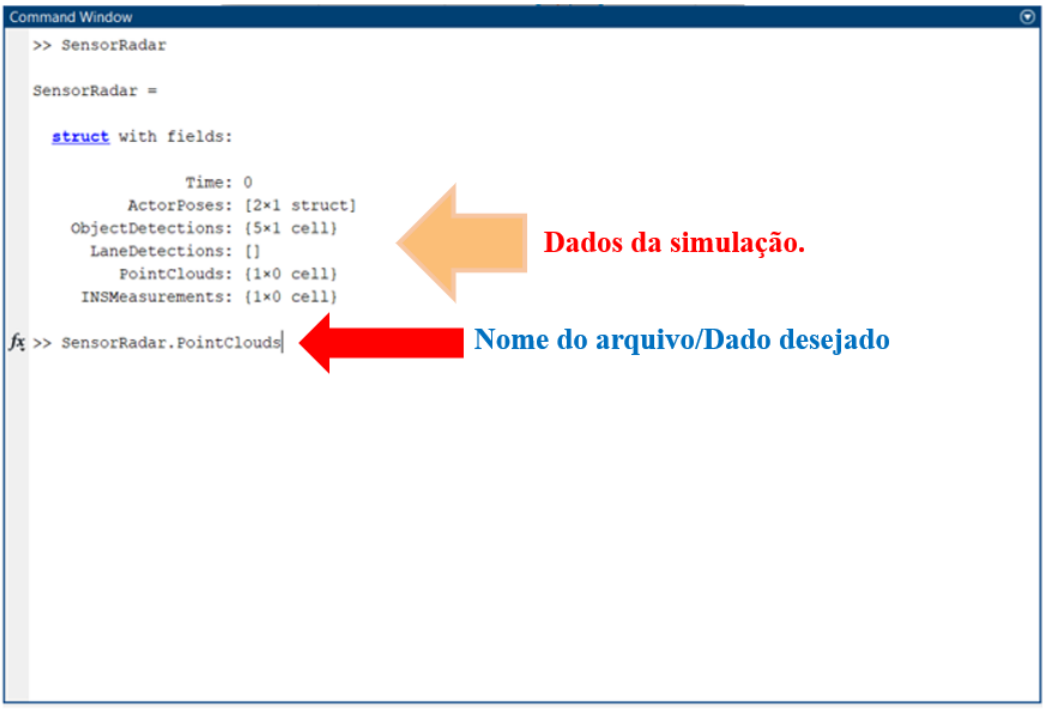
Figura 78: Nomeando o arquivo exportado.



Fonte: Autoria própria (2022)

Logo após, basta digitar, o **"Nome do arquivo. O dado que estamos buscando"** neste caso o PointClouds.

Figura 79: Extraindo dados do arquivo para o MATLAB.



Fonte: Autoria própria (2022)

Logo após basta pegar os PointClouds e aplicar o algoritmo desejado. Bom esses passos foram para aplicação em um cenário de simulação no MATLAB muito próximo do real, porém para aplicar o algoritmo em uma varredura de nuvem de pontos de um radar real que armazenada como uma coleção de pontos 2D ou 3D dependendo da geração do radar, aqui vale ressaltar que quanto mais moderno melhor a resolução e a qualidade dos pontos captados pelo radar como explicado no tópico sobre radares, que contém as coordenadas dos objetos ao redor de um veículo, basta ter um arquivo ". mat" e aplicar no algoritmo, veja o exemplo.

Carregue as coordenadas *x, y, ou x, y, z* dos objetos, depois defina a área de interesse, se desejar visualize o gráfico em 2D, aplique o algoritmo desejado, e se desejar circule os objetos de interesse.

Figura 80: Esboço da aplicando em Varredura de Radar real MATLAB.

```
% Carregar o Arquivo
load('radar_subset.mat')
loc = radar_subset;

% defina a região de interesse
% para abranger 20 metros à esquerda e à direita do veículo,
% 20 metros à frente e atrás do veículo
% e a área acima da superfície da estrada.
xBound = 20; % em metros
yBound = 20; % em metros
zLowerBound = 0; % em metros

indices = loc(:,1) <= xBound & loc(:,1) >= -xBound ...
    & loc(:,2) <= yBound & loc(:,2) >= -yBound ...
    & loc(:,3) > zLowerBound;
loc = loc(indices,:);

%Observar em gráfico 2D
scatter(loc(:,1),loc(:,2),'.');
annotation('ellipse',[0.48 0.48 .1 .1],'Color','red')

D = pdist2(loc,loc);

%Aplicar o Algoritmo nesse caso DBSCAN
[idx, corepts] = dbscan(D,2,50,'Distance','precomputed');

% Fazer o reconhecimento dos Objetos
% Circulando os objetos de interesse
numGroups = length(unique(idx));
gscatter(loc(:,1),loc(:,2),idx,hsv(numGroups));
annotation('ellipse',[0.54 0.41 .07 .07],'Color','red')
grid
```

Fonte: Autoria própria (2022)

É desta forma que os algoritmos de clusterização podem ser aplicados em sistema de radar automotivo para o reconhecimento de objetos, depois dessa etapa do reconhecimento manual circulando os objetos, basta reconhecer os padrões que os pontos gerados pelo radar formam para cada objeto e fazer um treinamento de *IA Machine Learning,* rede neural como por exemplo a YOLO v3, treinar a rede para o reconhecimento desses padrões de pontos e associar aos objetos para que o próprio radar reconheça os objetos partindo da clusterização e dos pontos gerados pelo radar, e assim teremos mais uma ferramenta que trará segurança e confiabilidade na utilização do reconhecimento de objetos para veículos autônomos.

6 CONCLUSÃO

6.1 ANÁLISE DOS RESULTADOS FINAIS

Conforme mostrado nesse trabalho. Conclui-se que não existe algoritmo de clusterização bom ou ruim, cada algoritmo deve ser aplicado em situações específicas de distribuição de dados, claro que um algoritmo pode mostrar melhor desempenho em mais de um tipo distribuição de agrupamento, como por exemplo o método DBSCAN que é um método de muito sucesso, e com certeza pode ser utilizado na aplicação de um sistema de radar automotivo, vimos também que o equipamento utilizado para captar a nuvem de pontos influência bastante no resultado, como vimos, os diferentes tipos de radar e suas aplicações, as gerações de radares, a evolução dos radares que antes mediam a velocidade e posição de um veículo e hoje é capaz de identificar objetos, quando aplicamos em sua nuvem de pontos, algoritmos de clusterização, essa evolução dos radares ajuda e muito a tecnologia de veículos autônomos, pois com essa tecnologia, o carro autônomo é capaz de detectar objetos em tempo real e em qualquer tipo de situação climática, visto que outros sensores como câmeras por exemplo não funcionam tão bem, em dias chuvosos ou com neblinas, infelizmente não foi possível realizar a simulação com arquivo de nuvem de pontos gerados por autoria própria, mesmo estando em parceria com o grupo GSA do programa Rota 2030 da UTFPR-PG, pois ainda não temos o sensor devidamente instalado para realizar esses testes. Do mais, esse trabalho pode trazer uma grande contribuição na parte de simulação sobre essa nova tecnologia que vem sendo desenvolvida.

6.2 LIMITAÇÕES

As limitações encontradas na realização desse trabalho, foram as dificuldades para achar bons materiais sobre os algoritmos de clusterização e, principalmente sobre as tecnologias de radares, pois é algo que está em constante desenvolvimento e é propriedade intelectual das empresas fabricantes, da mesma forma a junção do trabalho em si, pois os algoritmos de clusterização para reconhecimento de objetos em sistema de radar automotivo é um tema recente que ainda está sendo desenvolvido e estudado. Outro fator que trouxe limitações, foi a questão da simulação, pois o programa utilizado para realizar as simulações exige um alto custo computacional.

6.3 TRABALHOS FUTUROS

- Estudo e Aplicação de Radar 4G
- Utilização de modelos de radar sintético para maior exatidão e precisão;
- Métodos de testes sob diferentes condições climáticas considerando falsos positivos e falsos negativos;
- Utilização de múltiplos radar num veículo para ampliar o campo de visão;
- Melhoria e otimização de algoritmos de clusterização;
- Aplicação de algoritmos de clusterização em MIMO radar;
- Aplicação de inteligência artificial para melhoras a exatidão e precisão de MIMO radar;
- Geração de dataset de MIMO radar para desenvolvimento e testes de novos algoritmos e estratégias de clusterização.

REFERÊNCIAS

Rabideau, DJ (2003). **Ubiquitous MIMO multifunction digital array radar**. A Trigésima Sétima Conferência Asilomar sobre Sinais, Sistemas e Computadores.

Rabideau, DJ (2003). **Ubíquo MIMO Multifunction Digital Array Radar** ... and the Role of Time-Energy Management in Radar (PDF). CENTRO DE INFORMAÇÕES TÉCNICAS DE DEFESA.

Bliss, DW; Forsythe, KW (2003). **"Radar de múltiplas entradas e saídas múltiplas (MIMO) e imagens: graus de liberdade e resolução"**. The Trinta e Sétima Conferência Asilomar sobre Sinais, Sistemas e Computadores, 2003 . Pacific Grove, CA, EUA: IEEE: 54–59.

Bouveyron, et. al, **Model-Based Clustering and Classification for Data Science: With Applications in R: 50**, Cambridge University Press 25 julho 2019.

Halliday, Resnick, Walker J, **Fundamentos de Física – Eletromagnetismo**, volume 3, LTC; 10ª edição, 30 junho 2016.

Javier Béjar , **K-means vs Mini Batch K-means: A comparison**, Departament de Llenguatges i Sistemes Informàtics Universitat Politècnica de Catalunya, 2013.

KW Forsythe, DW Bliss e GS Fawcett. **Radar de múltiplas entradas e saídas múltiplas (MIMO): problemas de desempenho**. Conference on Signals, Systems and Computers, 1: 310–315, novembro de 2004.

Gao, Xin, et al. **"Sobre as abordagens derivadas da MÚSICA de estimativa de ângulo para radar MIMO bistático."** Redes sem fio e sistemas de informação, 2009. WNIS'09. Conferência Internacional em. IEEE, 2009.

Li, Jian e Petre Stoica . **"Radar MIMO com antenas posicionadas."** IEEE Signal Processing Magazine 24.5 (2007): 106-114.

Sturm, Christian, et al. "**Sinais de múltiplas portadoras espectralmente intercalados para aplicações de rede de radar e radar de múltiplas entradas e múltiplas saídas."** IET Radar, Sonar & Navigation 7.3 (2013): 261-269.

Chen, Chun-Yang e PP Vaidyanathan. **"Propriedades de ambigüidade do radar MIMO e otimização usando formas de onda de salto de frequência."** IEEE Transactions on Signal Processing 56.12 (2008): 5926-5936.

Zhang T, Ramakrishnan R, Livny M, SIGMOD. **BIRCH: An Efficient Data Clustering Method for Very Large Databases**. T Conferência do ACM Special Interest Group on Management of Data. 1996.

Danzer A, et al. ***2D Car Detection in Radar Data with PointNets***. IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 2019, pp. 61-66.

Campello, Ricardo J. G. B.; Moulavi, Davoud; Zimek, Arthur; Sander, Jörg (2015).

"Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection". *ACM Transactions on Knowledge Discovery from Data*. **10** (1): 1–51.

HAN, J. & KAMBER, M. Data Mining: Concepts and Techniques. 2nd. 2006 ROCHA, T., PERES, S. M., BÍSCARO, H. H., MADEO, R. C. B., BOSCARIOLI, C. **Tutorial sobre Fuzzuc-Means e Fuzzy Learning Vector Quantization: Abordagens Híbridas para Tarefas de Agrupamentos e Classificação. Revista de Informática Teórica e Aplicada**, vol.9, n.1, 2012. COSTA, J. A. F. Classificação Automática e Análise de Dados por Redes Neurais Auto-Organizáveis. Tese de Doutorado. Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas. 1999.

Ankerst, Mihael, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. **"OPTICS: ordering points to identify the clustering structure."** In ACM Sigmod Record, vol. 28, no. 2, pp. 49–60. ACM, 1999.

Ankerst, Mihael, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. **"OPTICS-OF: Identifying Local Outliers"** in J.M. Zytkow and J. Rauch (Eds.): PKDD'99, LNAI 1704, pp. 262–270, 1999

Kriegel, Hans-Peter; Schubert, Erich; Zimek, Arthur (2016). **"The (black) art of runtime evaluation: Are we comparing algorithms or implementations?".** Knowledge and Information Systems. 52 (2): 341–378.

RENESAS ELECTRONICS. **Why do we need radar?**. Renesas 2022. Disponível em < https://www.renesas.com/us/en/blogs/radar-transceivers-key-component-adas-autonomous-driving-blog-1-why-do-we-need-radar>. Acesso em 03 de março de 2022.

Brendan J. Frey and Delbert Dueck, **"Clustering by Passing Messages Between Data Points"**, Science Feb. 2007.

Jian Li, **"MIMO Radar Signal Processing"** Wiley-IEEE Press, 2009.

Bach, F., Jordan, M.: Learning spectral clustering. In: Thrun, S., Saul, L., Schölkopf, B. (eds.) **Advances in Neural Information Processing Systems** 16 (NIPS), pp. 305–312. MIT Press, Cambridge (2004)

Nicolas S, Appenrodt1 N, Dickmann J, e Sick B. **A *Multi-Stage* Clustering Framework for Automotive Radar Data.** Conference: 2019 IEEE Intelligent Transportation Systems Conference – ITSC.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). **A density-based algorithm for discovering clusters in large spatial databases with noise**. In *Kdd* (Vol. 96, №34, pp. 226–231).

Friedman, Jerome (2009). **"14.3.12 Hierarchical clustering". The Elements of Statistical Learning (2nd ed.)**. New York: Springer. pp. 520–8.

Nielsen, Frank (2016). **"8. Hierarchical Clustering".** ***Introduction to HPC with MPI for Data Science***. Springer. pp. 195–211.

Maimon, Oded; Rokach, Lior (2006). **"Clustering methods". Data Mining and Knowledge Discovery Handbook.** Springer. pp. 321–352.

Kaufman, L.; Rousseeuw, P.J. (1990). **Finding Groups in Data: An Introduction to Cluster Analysis** (1 ed.). New York: John Wiley.

Mukherjee S, "**Mean Shift**". 2019. Disponível em: <github.com/SatadruMukherjee/Dataset/blob/main/MeanShiftCluster.m>. Acesso em: 05 de março de 2022.

LEARN, S. **Affinity Propagation**. 2015. Disponível em: <http://scikit- learn.org/stable/modules/clustering.html>. Acesso em: 01 de fevereiro de 2022.

LEARN, S. **MiniBatchKMeans**. scikit-learn 2015. Disponível em: <http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html>. Acesso em: 03 de fevereiro de 2022.

LEARN, S. **DBSCAN**. 2021. Disponível em: <http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html>. Acesso em: 03 de fevereiro de 2022.

KHOSLA, S. **ML-K-Means Algorithm.** 2021. Disponível em: <https://www.geeksforgeeks.org/ml-k-means-algorithm/>. Acesso em 07 de fevereiro de 2022.

MATLAB, **DBSCAN**. 2021. Disponível em: <https://www.mathworks.com/help/stats/dbscan.html>. Acesso em 04 de março de 2022.

MATLAB, **DBSCAN**. 2021. Disponível em: < https://www.mathworks.com/help/map/ref/azimuth.html>. Acesso em 04 de março de 2022.

MATLAB, **DBSCAN**. 2021. Disponível em: < https://www.mathworks.com/help/stats/dbscan-clustering.html>. Acesso em 10 de março de 2022.

BÉJAR, J. **K-Means vs Mini Batch K-Means: A comparison.** 2020. Disponível em: <https://upcommons.upc.edu/bitstream/handle/2117/23414/R13-8.pdf>**.** Acesso em 08 de fevereiro de 2022.

CARRASCO, C. O. **Gaussian Mixture Models Explained**. 2019. Disponível em: <https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95>. Acesso em 08 de fevereiro de 2022.

*Bishop, Christopher M. "***Pattern Recognition and Machine Learning"** (2006) Springer-Verlag Berlim, Heidelberg. P (111-655).

Murphy, Kevin P. ***Machine Learning: A Probabilistic Perspective*** (2012), Cambridge, Massachusetts Institute of Technology. P (339-386).

PUC-RIO. **O Método DBSCAN.** 2018. Disponível em:< https://www.maxwell.vrac.puc-rio.br/24787/24787_6.PDF>. Acesso em 08 de fevereiro de 2022.
SACRAMENTO, G. **Clusterização de dados**. 2021. Disponível em:< https://blog.somostera.com/data-science/clusterizacao-dedados >. Acesso em 10 de fevereiro de 2022.

Ester, M., H.-P. Kriegel, J. Sander, and X. Xiaowei. **"A density-based algorithm for discovering clusters in large spatial databases with noise."** In *Proceedings of the Second International Conference on Knowledge Discovery in Databases and Data Mining*, 226-231. Portland, OR: AAAI Press, 1996

IDTechEX, "**Automotive Radar**" 2020. Disponível em: < https://textile-future.com/archives/48679>. Acesso em: 13 de março de 2022

IDTechEX, "**Automotive Radar Comparision**" 2020. Disponível em: < https://www.systemplus.fr/reverse-costing-reports/automotive-radar-comparison-2018/ >. Acesso em: 13 de março de 2022.

Renesas, "**Automotive Radar Industrial**" 2018. Disponível em: < http://www.yole.fr/AutomotiveRadar_IndustryUpdate.aspx#.YiVkA3rMLIU >. Acesso em: 13 de março de 2022.

Renesas, "**Radar Transceivers/ADAS**" 2018. Disponível em: < https://www.renesas.com/br/en/blogs/radar-transceivers-key-component-adas-autonomous-driving-blog-1-why-do-we-need-radar>. Acesso em: 13 de março de 2022.

IDTechEX, "**Automotive Radar**" 2022. Disponível em: < https://www.idtechex.com/en/research-report/automotive-radar-2022-2042/850 >. Acesso em: 13 de março de 2022.

Psmarketresearch, "**Automotive Radar Market**" 2022. Disponível em: < https://www.psmarketresearch.com/market-analysis/automotive-radar-market >. Acesso em: 13 de março de 2022.

YOLE, "**Radar Industry**" 2022. Disponível em: < http://www.yole.fr/Radar_Industry_Market_Update.aspx>. Acesso em: 13 de março de 2022.

APTIV, "**4D Imaging Radar**" 2022. Disponível em: < https://www.aptiv.com/en/insights/article/what-is-4d-imaging-radar >. Acesso em: 03 de março de 2022.

BOSCH, "**Sensor radar**" 2022. Disponível em: < https://www.bosch-mobility-solutions.com/en/solutions/sensors/corner-radar-sensor/ >. Acesso em: 05 de março de 2022.

VALEO, "**Comfort Driving Assistance Systems**" 2022. Disponível em: < https://www.valeo.com/en/comfort-driving-assistance-systems/>. Acesso em: 05 de março de 2022.

DENSO, "**Sensor Radar News**" 2022. Disponível em: < https://www.denso.com/global/en/news/newsroom/2017/20170810-g01/>. Acesso em: 06 de março de 2022.

VEONEER, "**Radar**" 2022. Disponível em: < https://www.veoneer.com/en/radar>. Acesso em: 06 de março de 2022.

PSMARLETRESEARCH, **"Automotive Radar Market"**, 2022. Disponível em: https://www.psmarketresearch.com/market-analysis/automotive-radar-market.Acesso em: 13 de março de 2022.

Dorin Comaniciu, Peter Meer, **"Mean Shift: A robust approach toward feature space analysis"**. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2002. pp. 603-619.

BACH, F. and JORDAN, M., **"Learning spectral clustering"**, In: Thrun, S., Saul, L., Schölkopf, B. (eds.) Advances in Neural Information Processing Systems 16 (NIPS), pp. 305–312. MIT Press, Cambridge (2004).

R. Bishop, **"Application Areas"**, in Intelligent Vehicle Technology and Trends, Artech House ITS Library, pp. 25 – 37, 2005.

S. Beer, G. Adamiuk e T. Zwick, **"Novel Antenna Concept for Compact Millimeter-Wave Automotive Radar Sensors"**, *IEEE Antenas and Wireless Propagation Letters*, vol. 12, pp. 771 – 774, 2009.

A. Fischer, Z. Tong et al., **"Transceptor de radar multicanal de 77 GHz com antena em pacote"**, IEEE Transactions on Antenas And Propagation, vol. 62, pp. 1386 – 1393, 2014.