INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS TERESINA CENTRAL
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DANIEL MESSIAS MENDES CAMPELO PITOMBEIRA

DESENVOLVIMENTO DE ALGORITMO LÍDER-SEGUIDOR APLICADO EM ENXAME DE ROBÔS

TERESINA, PIAUÍ

2020

DANIEL MESSIAS MENDES CAMPELO PITOMBEIRA

DESENVOLVIMENTO DE ALGORITMO LÍDER-SEGUIDOR APLICADO EM ENXAME DE ROBÔS

Projeto apresentado à Banca Examinadora como requisito para aprovação na disciplina de Trabalho de Conclusão de Curso II do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí.

Orientador: Prof. Me. Francisco Marcelino Almeida de Araújo

TERESINA, PIAUÍ

2020

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD

Pitombeira, Daniel Messias Mendes Campelo

P685d Desenvolvimento de algoritmo líder-seguidor aplicado em enxame de robôs / Daniel Messias Mendes Campelo Pitombeira. - 2019.
30 f.: il. color.

Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Teresina Central, 2019.
Orientador : Prof Me. Francisco Marcelino Almeida de Araújo.

1. Inteligência. 2. ROS (Robot Operation System). 3. Lider-seguidor. I.Título.

CDD - 004.21

Elaborado por Tanize Maria Sales CRB 3/1024

DANIEL MESSIAS MENDES CAMPELO PITOMBEIRA

DESENVOLVIMENTO DE ALGORITMO LÍDER-SEGUIDOR APLICADO EM ENXAME DE ROBÔS

Projeto apresentado à Banca Examinadora como requisito para aprovação na disciplina de Trabalho de Conclusão de Curso II do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí.

Aprovado pela banca examinadora em _____.

BANCA EXAMINADORA:

Prof. Me. Francisco Marcelino Almeida de Araújo
Instituto Federal de Educação, Ciência e Tecnologia do Piauí - IFPI

Prof. Me. Duany Dreyton Bezerra Sousa
Instituto Federal de Educação, Ciência e Tecnologia do Piauí - IFPI

Dr. Antônio Sales Oliveira Coelho
Universidade Federal do Piauí - UFPI

Me. Juan de Aguiar Gonçalves
Universidade Estadual do Piauí - UESPI

TERESINA, PIAUÍ

2020

RESUMO

Este trabalho apresenta o uso de inteligência de enxame em robôs utilizando as bibliotecas oferecidas por ROS (Robot Operation System), com foco no uso no desenvolvimento de um algoritmo que usa a estratégia de líder-seguidor, fazendo assim que trabalhem em conjunto. Essa abordagem pode ser aplicada com sucesso em uma ampla variedade de trabalhos: estudo de características biológicas, resgate, limpeza de vazamento de petróleo, etc. Neste projeto foi criado um conjunto de robôs onde um agente líder supervisiona o grupo de agentes seguidores e os coordena para alcançar o objetivo do enxame.

Palavras-chaves: Inteligência de enxame, ROS, líder-seguidor.

ABSTRACT

This work presents the use of swarm intelligence in robots using the libraries offered by ROS (Robot Operation System), focusing on the use of an algorithm that uses the leader-follower strategy, thus working together, this approach can be applied successfully in a wide variety of works, study of biological characteristics, rescue, cleaning of oil leak, etc. In this project a set of robots was created where a leader agent supervises the group of follower agents and coordinates them to reach the swarm's goal.

Key-words: Swarm intelligence, ROS, leader-follower.

LISTA DE ILUSTRAÇÕES

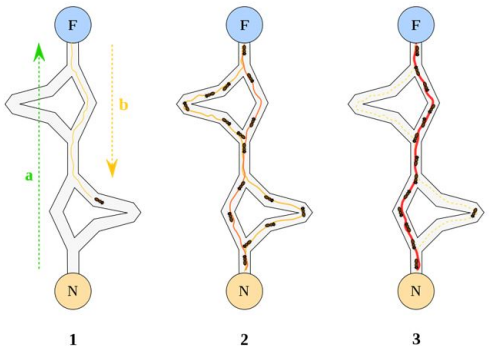
Figura 1 – Inteligencia de enxame em formigas 7
Figura 2 – Formigas unidas em prol de um objetivo final 8
Figura 3 – Fluxograma 12
Figura 4 – Simulação líder-seguidor com obstáculo 13
Figura 5 – Simulação da formação líder-seguidor 14
Figura 6 – Modelos de TurtleBot3 15
Figura 7 – Estratégia do fluxo de tópicos utilizada 16
Figura 8 – Diagrama de classe 18
Figura 9 – Modelo utilizado pelo ROS 18
Figura 10 – Antes do Algoritmo 19
Figura 11 – Depois do Algoritmo 20
Figura 12 – Mapa simulado utilizando *matplotlib* (15 partículas) 21
Figura 13 – Diagrama de classe - Mapa 22
Figura 14 – Diagrama de classe - Particula 23
Figura 15 – Inicio - 5 partículas, 1 interação 24
Figura 16 – Fim sem PSO - 5 partículas, 1 interação 24
Figura 17 – Fim com PSO - 5 partículas, 1 interação 25
Figura 18 – Inicio - 10 partículas, 3 interações 26
Figura 19 – Fim sem PSO - 10 partículas, 3 interações 26
Figura 20 – Fim com PSO - 10 partículas, 3 interações 27

SUMÁRIO

1	**INTRODUÇÃO**	**7**
1.1	Justificativa	9
1.2	Objetivos	9
1.2.1	Objetivo Geral	9
1.2.2	Objetivos Específicos	9
2	**FUNDAMENTAÇÃO TEÓRICA**	**10**
2.1	Inteligência de enxame	10
2.2	ROS	10
2.3	Gazebo	11
2.4	Particle swarm optimization	11
3	**TRABALHOS RELACIONADOS**	**12**
3.1	Swarm bots: System design for ECHOLOCATION	12
3.2	Planning for multi-agent teams with leader switching	13
3.3	Leader-follower formation control of underactuated surface vehicles based on sliding mode control and parameter estimation	14
4	**METODOLOGIA**	**15**
4.1	Metodologia inicial	15
4.2	Fluxo antigo	15
4.3	Lógica Matemática	16
4.4	Lógica computacional	17
4.5	Simulação	19
4.5.1	Testes	19
4.6	Metodologia final	20
4.7	Lógica final	21
4.8	Resultados final	23
5	**CONCLUSÃO**	**28**
	REFERÊNCIAS	**29**

1 INTRODUÇÃO

Figura 1 – Inteligencia de enxame em formigas



Fonte: [SINGH; KUMAR; SINGH, 2011]

Foi constatado que ao observar a natureza pode-se extrair características e aprendizados em que é possível utilizá-los em diversas situações rotineiras. [VERLEKAR; JOSHI, 2018] demonstra em seu trabalho que a natureza sempre nos surpreendeu, e ao observá-la pode-se notar sequencias lógicas que podem ser simuladas em nossos robôs, ajudando a realizar tarefas complexas com êxito.

Um desses aprendizados é a inteligência de enxame (SI) que se baseia em uma combinação de práticas baseado no comportamento coletivo de sistemas descentralizados e auto-organizados, introduzido por Gerardo Beni e Jing Wang em 1989 no contexto de robótica, [GARG et al., 2009]. Sendo assim composta por um grupo de indivíduos que interagem entre si e com seu ambiente fazendo com que juntos consigam chegar a um objetivo final, fazendo uso de interações robô-robô e ambiente-robô por meio de uma comunicação local e trocas de informações, [G; A; HENCY, 2018]. Um exemplo bem claro desse comportamento é mostrado na Figura 1, na qual temos formigas que ao trabalharem em conjunto conseguem se locomover de uma folha até a outra.

Figura 2 – Formigas unidas em prol de um objetivo final

Fonte: [CARACIOLO, 2009]

Segundo [TAN; ZHENG, 2013], SI é inspirada em enxames encontrados na natureza, como insetos sociais, peixes ou mamíferos, que se comunicam uns com os outros em um enxame da vida real. Variando em tamanho, complexidade e modos de organização, existindo colônias altamente estruturadas ocupando uma ampla extensão, atingindo milhões de seres.

Diante disso, em seu trabalho [VERLEKAR; JOSHI, 2018] cita o uso de enxame em robôs como um método de inspiração biológica moderna em que é simulado o comportamento de especies biológicas em robôs.

Muitas dos comportamentos biológicos são refletidos em propriedades de enxame, como citadas por [SAHIN; SPEARS; WINFIELD, 2007]:

- Estão situados no ambiente e podem agir para modificá-lo;
- Os recursos de sensoriamento e comunicação dos robôs são locais;
- Os robôs não têm acesso ao controle centralizado e/ou ao conhecimento global;
- Cooperam para lidar com uma determinada tarefa.

Sendo a inteligência de enxame algo que abrange um grande campo de atividades, variando de acordo com o objetivo a ser concluído, segundo [G; A; HENCY, 2018] algumas de suas aplicações são: agrupamento (formação de grupos), forrageamento (coleta e entrega), exploração (para explorar área máxima, eles se distribuem), controle de bando (comportamento de grupo altamente coordenado como pássaros e peixes), classificação de tarefas (construção de ninhos de vespas e cupins).

1.1 JUSTIFICATIVA

A justificativa do tema se dá pela dificuldade que robôs individuais têm em fazer tarefas complexas, como um transporte de cargas pesadas, manobras em terrenos complexos, dentre outros, [Shia; Bastani; Yen, 2011], além de prevenção de falhas caso algum robô esteja danificado ou sem bateria, [MANSOR; ADOM; RAHIM, 2015]. Assim a inteligencia de enxame vem sendo uma porta de entrada para realização de trabalhos determinantes, com uma gama de aplicações, podendo ser aplicada em missões de busca, exploração espacial, além de poder fazer tarefas que robôs individuais teriam grande dificuldade [Shia; Bastani; Yen, 2011].

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver algoritmos de estratégia líder-seguidor por meio de ferramentas de inteligência de enxame com robôs realizando uma análise sobre a influência do método de PSO em uma técnica líder-seguidor, contribuindo com futuras aplicações de estratégias mestre-escravo com robôs.

Este trabalho visa o desenvolvimento e comparação de dois algoritmos líder-seguidor utilizando um enxame de robôs, onde será visto quais os benefícios da implantação de uma estratégia de PSO num contexto de líder-seguidor. Trazendo a geração de um sistema inteligente para realizar tarefas específicas lideradas por um único robô.

1.2.2 Objetivos Específicos

- Desenvolver algoritmo de líder-seguidor;
- Aplicar técnicas de inteligência de enxame em robôs;
- Desenvolver algoritmo líder-seguidor com uso de PSO; e
- Analisar resultados obtidos na simulações em um ambiente controlado.

2 FUNDAMENTAÇÃO TEÓRICA

Para melhor compreender a finalidade desse trabalho, é necessário entender alguns conceitos relacionados.

2.1 INTELIGÊNCIA DE ENXAME

"O termo swarm intelligence (inteligencia de enxame) foi proposto no fim da década de 1980, quando se referia a sistemas robóticos compostos por uma coleção de agentes simples em um ambiente interagindo de acordo com regras locais.", [ATTUX; ZUNBEN, 1980].

No trabalho de [MANCUSO, 2010] ele fala sobre o comportamento similar da internet com as raízes das plantas, por conseguinte no reino animal pode ser observado muitos comportamentos que podem ser espelhados na computação, segundo [KRAUSE; RUXTON; KRAUSE, 2010], "Há uma abundância de exemplos de comportamento social entre insetos e animais na natureza que exibem propriedades inteligentes emergentes".

Todavia, os seres humanos também tem essa prática, salienta [KRAUSE; RUXTON; KRAUSE, 2010], "Inteligencia de enxame também pode ser observado nos humanos. As pessoas aprendem umas com as outras. O conhecimento se espalha de pessoa para pessoa. Cultura emerge das populações, nossa sociedade tem essa notável capacidade de se auto-organizar e se adaptar."

2.2 ROS

Robot Operating System é um framework utilizado para desenvolver aplicações para robôs, ele oferece uma série de ferramentas e bibliotecas que vêm para facilitar a tarefa de criar um comportamento específico em plataformas robóticas, [ROS, 2019c]. Para entendermos mais a fundo como funciona o ROS, temos que falar um pouco sobre nós e tópicos.

No seu conceito mais básico, um nó é um processo que realiza computação, uma das funções mais importantes do mesmo é a comunicação entre outros nós pelo uso de tópicos. A divisão por nós se mostra mais eficaz em relação a tolerância sobre falhas e redução da complexidade do código. Partes fundamentais de um robô são dividas em nós como: o controle sobre suas rodas, controle sobre os sensores de proximidade, controle de planejamento de um caminho a ser seguido. [ROS, 2019a]

Visto que a comunicação dos nós se faz pelo uso de tópicos, os nós que geram dados criam publicadores que apenas disparam as informações (publisher) no tópico, já os nós que estão interessados em dados assinam (subscriber) o tópico desejado, assim podendo receber os dados publicados. Pode haver vários publishers e subscribers em um tópico.

Os tópicos são destinados a comunicação unidirecional de fluxo contínuo. Os nós que precisam executar chamadas de procedimento remoto, ou seja, receber uma resposta a uma solicitação, devem usar os serviços. [ROS, 2019e]

Analisando o mecanismo de comunicação do ROS, pode-se, segundo [VILLA et al., 2017], afirmar que se trata de nós e tópicos, que partilham informação sobre a estrutura de dados, cada nó pode publicar mensagens em algum tópico, quanto pode receber informações assinando o tópico pretendido. Devido a esse tipo de descentralização, ocorre uma melhora no desempenho do sistema e permite a execução de pacotes em diferentes estações.

2.3 GAZEBO

Gazebo é um software projetado para reproduzir com precisão os ambientes dinâmicos que um robô pode encontrar. Todos os objetos simulados têm massa, velocidade, atrito e dentre outros atributos que lhes permitem se comportar de forma realista quando movimentado. [Koenig; Howard, 2004].

É um simulador gratuito que permite testar algoritmos em modelos 3D, projetar robôs, realizar testes de regressão e treinar sistemas de IA usando cenários realistas. O Gazebo oferece a capacidade de simular com precisão e eficiência enxame de robôs em ambientes internos e externos complexos com ótimas interfaces gráficas e programáticas.

Por ser um software tão robusto, ele acaba tendo seus contras, como ser um software custoso em máquinas não tão poderosas e sua outra limitação é fato de só estar disponível apenas para linux.

2.4 PARTICLE SWARM OPTIMIZATION

Segundo [LI; CLERC, 2019], Particle Swarm Optimization (Optimização por enxame de partículas) é uma técnica meta-heurística inspirada nos comportamentos sociais observados em animais, insetos e humanos. Desde a sua criação, o PSO teve uma aceitação generalizada entre pesquisadores e profissionais como uma técnica robusta e eficiente para resolver vários problemas de otimização.

Desde 1995, várias versões do algoritmo sugiram, entretanto, todas elas se baseia na ideia inicial de um enxame de partículas que interagem entre si a procura de um ótimo global. De acordo com [Doctor; Venayagamoorthy; Gudise, 2004], isso se deve graças a persistência ao longo do tempo na influência de um indivíduo sobre o outro e do grupo sobre o indivíduo. A força do PSO depende da seleção adequada dos parâmetros do enxame. Um exemplo desses parâmetros seria a prioridade da partícula estar mais perto dos vizinhos ou do líder. Deste modo, não existe nenhum conjunto de parâmetros que seja ideal para todas as aplicações universalmente.

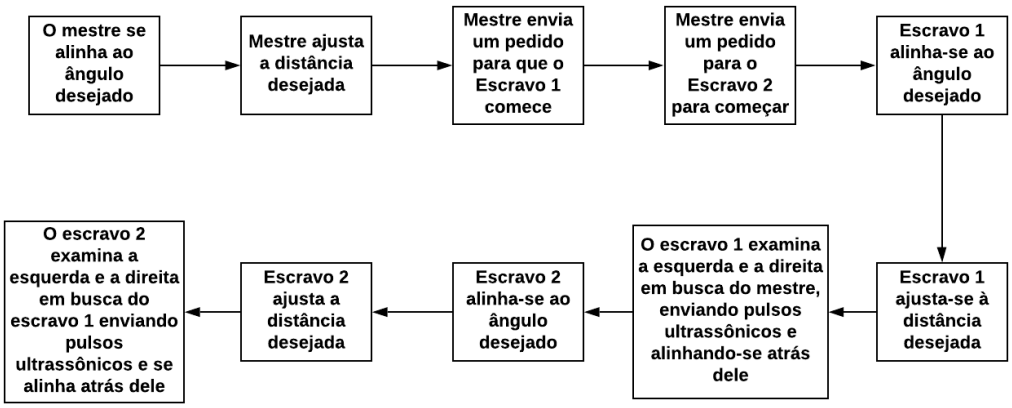
3 TRABALHOS RELACIONADOS

Como base para estudo do trabalho em questão, foram utilizados como bases alguns trabalhos acadêmicos, para o melhor entendimento foram realizados resumos sobre os temas de alguns trabalhos relacionados.

3.1 SWARM BOTS: SYSTEM DESIGN FOR ECHOLOCATION

No trabalho de [Wahi; Raj; Zhun Fan, 2017], traz o uso de inteligência de enxame em robôs inspirado na forma de comunicação usada pelos morcegos, a ecolocalização. O algoritmo descrito no trabalho usa o conceito de localização de eco para se alinharem um atrás do outro. O Master ou Líder funciona no modo broadcast, enviando sinais para ambos os escravos. Os escravos recebem as mensagens, mas apenas respondem aquelas que são designadas a ele. Antes de tudo os robôs se alinham a um certo ângulo em relação ao campo magnético da Terra. Depois disso, os dois escravos se movem para a esquerda e para a direita, lendo e armazenando valores ultra-sônicos até encontrarem os outros. Quando um chega perto do outro a distância captada pelo sensor ultrassônico cai drasticamente, e em seguida ele para. O fluxograma seguido pelo algoritmo é exposto no trabalho como:

Figura 3 – Fluxograma



Fonte: Elaborada pelo Autor, 2019

Algumas restrições desse projeto são a necessidade de um espaço fechado, já que o trabalho é realizado utilizando o conceito de SONAR, sem esse espaço fechado os sensores ultrassônicos não teriam a precisão exigida para a conclusão do mesmo, e um terreno liso, uma vez que os robôs usam o sensor ultrassônico para guiá-los e o design dos robôs inclui uma bateria pesada.

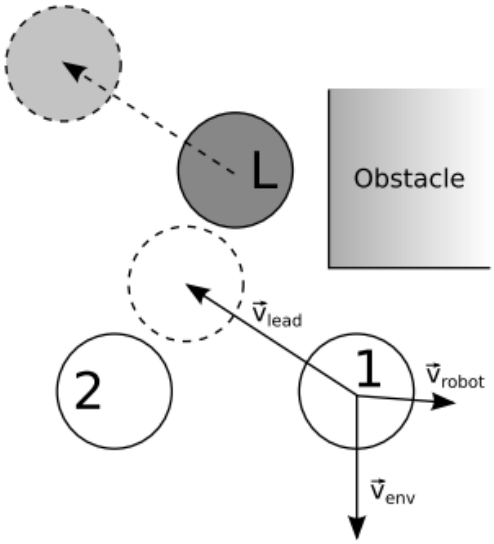
3.2 PLANNING FOR MULTI-AGENT TEAMS WITH LEADER SWITCHING

No estudo de [Swaminathan; Phillips; Likhachev, 2015], ele traz uma técnica de líder-seguidor com uma abordagem de planejamento que descobre quando trocar de líderes no decorrer do caminho. Os autores citam como principal explicação para o uso de um líder substituível o fato de que em certos momentos um seguidor ser mais eficaz seguindo um líder diferente ou se tornando o próprio líder.

No trabalho em questão são aplicadas técnicas de busca heurística convertendo o problema de elaboração de movimento em um grafo onde os estados (instante onde o robô está em cada iteração) se tornam as vértices no grafo e os movimentos em dois estados são as arestas unindo os vértices. Nos testes feitos, houve um desempenho significativamente melhor e uma taxa de sucesso de 61% para 97%.

Sobre cada movimento de cada seguidor foi gerado uma política que gera uma resposta a um movimento executado pelo líder. Na implementação do projeto, foi gerado uma política por meio da soma ponderada de vetores de velocidade retornados de três variáveis diferentes. Isso é ilustrado na Figura 3 pelos autores.

Figura 4 – Simulação líder-seguidor com obstáculo



Fonte: [Swaminathan; Phillips; Likhachev, 2015]

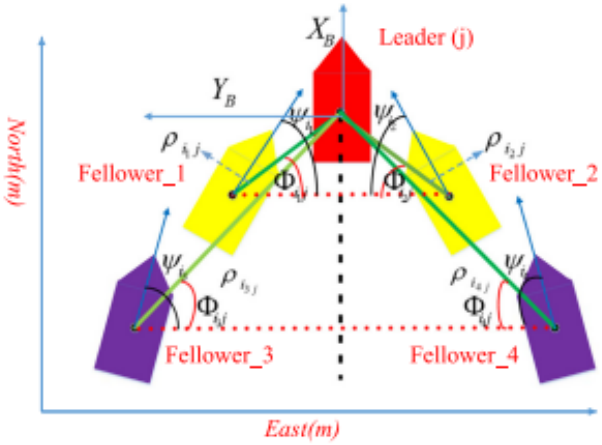
Onde $\vec{v}_{lead}$ se trata da força vetorial que o líder exerce sobre o seguidor, $\vec{v}_{env}$ é o comportamento que empurra os robôs para longe dos obstáculos do ambiente calculado por um gradiente exponencial negativo que se afasta dos obstáculos, obtendo uma repulsão suave e ao mesmo tempo exponencialmente crescente em relação a proximidade do obstáculo, e $\vec{v}_{robot}$ corresponde a ação de um robô repelir o outro, a velocidade resultante do robô é gerada por essas três forças.

3.3 LEADER-FOLLOWER FORMATION CONTROL OF UNDERACTUATED SURFACE VEHICLES BASED ON SLIDING MODE CONTROL AND PARAMETER ESTIMATION

O trabalho de [SUN et al., 2018] traz o conceito de formação líder-seguidor utilizando barcos, levando em consideração perturbações ambientais como ondas e tempestades que poderiam mudar o curso do mesmo. Para combater tais perturbações foram usadas técnicas como backstepping, que se baseia em tentar estabilizar o controle para a origem definida.

Informações como o ângulo que o seguidor necessita para alcançar o líder, e a orientação que o líder se encontra são necessárias para a conclusão do trabalho. Na Figura 4, o autor demonstra visualmente o ângulo calculado para alcançar o objetivo.

Figura 5 – Simulação da formação líder-seguidor



Fonte: [SUN et al., 2018]

No final do trabalho há analises sobre como o projeto se comportou ao ser influenciado por pertubações externas, resultados e possíveis continuações do projeto.

4 METODOLOGIA

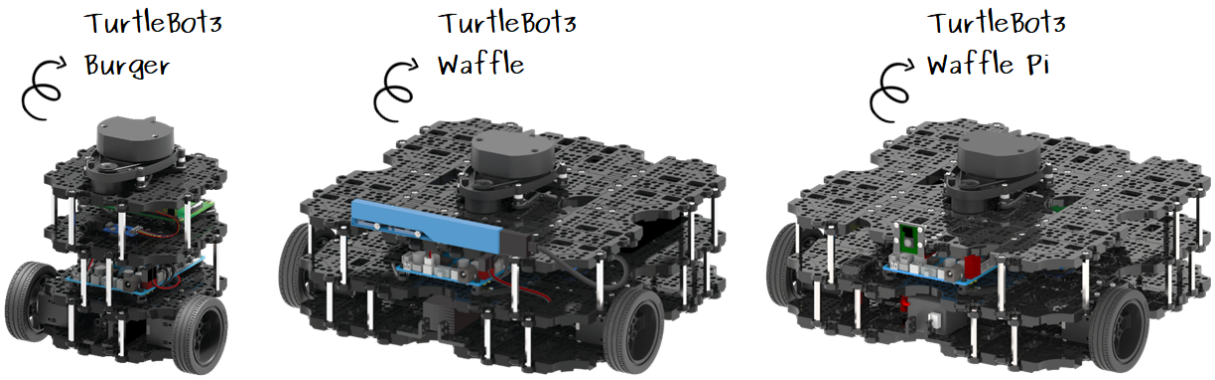
A metodologia foi divida em duas etapas: 1) etapa inicial onde foi desenvolvimento do algoritmo líder-seguidor utilizando ROS e o Gazebo para as simulações, 2) etapa final na qual houve a comparação entre o algoritmo líder-seguidor com e sem PSO, agora utilizando Python e simulando um mapa utilizando matriz. Esta mudança se deve pelo fato da simulação utilizando Gazebo ser muito custosa e os testes finais teriam grandes estruturas de repetições e um número grande de robôs.

4.1 METODOLOGIA INICIAL

A comunicação dos robôs é algo muito importante para a conclusão desta pesquisa. Para ocorrer essa comunicação precisaremos de um computador externo que possa ser o core para a execução do ROS, [ROS, 2019b].

Para conclusão desta primeira parte foi utilizada a linguagem Python junto com bibliotecas e ferramentas disponíveis pelo ROS, nas simulações serão utilizadas bibliotecas oferecidas da integração de ROS com Gazebo, que traz consigo algumas simulações de robôs, como TurtleBot3, gerando assim todo um esquema de comunicação. Na figura 5, temos os modelos disponíveis na biblioteca TurtleBot3, onde neste trabalho foi realizado utilizando o modelo Burger. [E-MANUAL, 2019]

Figura 6 – Modelos de TurtleBot3

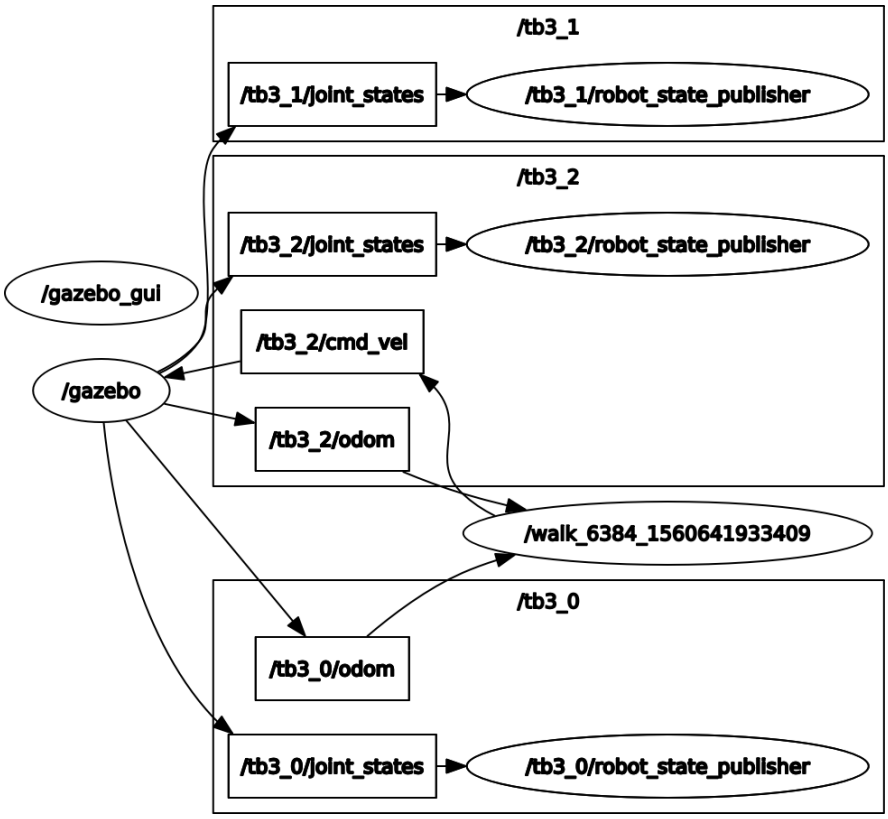


Fonte: [ROBOTIS, 2019]

4.2 FLUXO ANTIGO

No processo de desenvolvimento do projeto foi utilizado um plugin chamado rqt_graph que permite a visualização de um gráfico representando os nós e os tópicos que estão inscritos ou sendo publicados [ROS, 2019d].

Figura 7 – Estratégia do fluxo de tópicos utilizada



Fonte: Elaborada pelo Autor, 2019

Dentro do fluxo do projeto demonstrado na Figura 6 é possível observar que foi criado um novo nó que cuidará dos procedimentos expostos neste trabalho, denominado *walk*, junto dele temos um ID gerado automaticamente pelo próprio ROS, legendado na imagem acima por um círculo, dentro deste nó serão armazenadas todas as informações significativas para a realização da técnica líder-seguidor deste projeto.

Nesta simulação foram utilizado três robôs: tb3_0, tb3_1 e tb3_2, onde em cada robô temos a informação *odom* (Odometria) que oferece dados sobre a orientação e posição do robô em relação ao mundo simulado de forma precisa e em tempo real.

4.3 LÓGICA MATEMÁTICA

Para realização desta metodologia precisamos que o seguidor calcule a orientação necessária em relação ao líder para que o percurso feito por ele esteja correto. Diante disso, sua velocidade angular (velocidade necessária para o robô se deslocar em torno do seu eixo) é calculada a partir do ângulo calculado pela função *athan2* disponível na biblioteca math do Python onde é passado a diferença entre o Y do líder e o Y seguidor, X do líder e o X do seguidor retornando o ângulo que o seguidor precisa estar para chegar ao líder.

Ao calcular o ângulo que o robô precisa estar, foi preciso definir uma velocidade angular para que de fato o robô fizesse a rotação em seu eixo, para ter conhecimento sobre a orientação

atual do seguidor ser de fato a desejada, é necessário ser capaz de conhecer a orientação real do mesmo.

Para isso temos que transformar os valores de orientação recebidos em quatérnios pelo tópico *odom* para ângulos de Euler utilizando a biblioteca *tf.transformations*, assim retirando uma dimensão e podendo trabalhar diretamente com a orientação real do robô.

Desse modo foi feito o calculo da velocidade do seguidor por:

$$velAngular = (angleToMove - theta) * K_a$$

$arcToMove$ = Ângulo que o seguidor precisa estar;

$theta$ = Ângulo atual do seguidor; e

K_a = Constante de velocidade angular inicial.

$$velLinear = distancia * K_l$$

$distancia$ = Distancia entre líder e seguidor; e

K_l = Constante de velocidade linear inicial.

Para calcular o critério de parada do seguidor é calculada a distancia entre o líder e o seguidor utilizando a distância euclidiana fazendo a diferença entre o eixo X e eixo Y do líder e do seguidor.

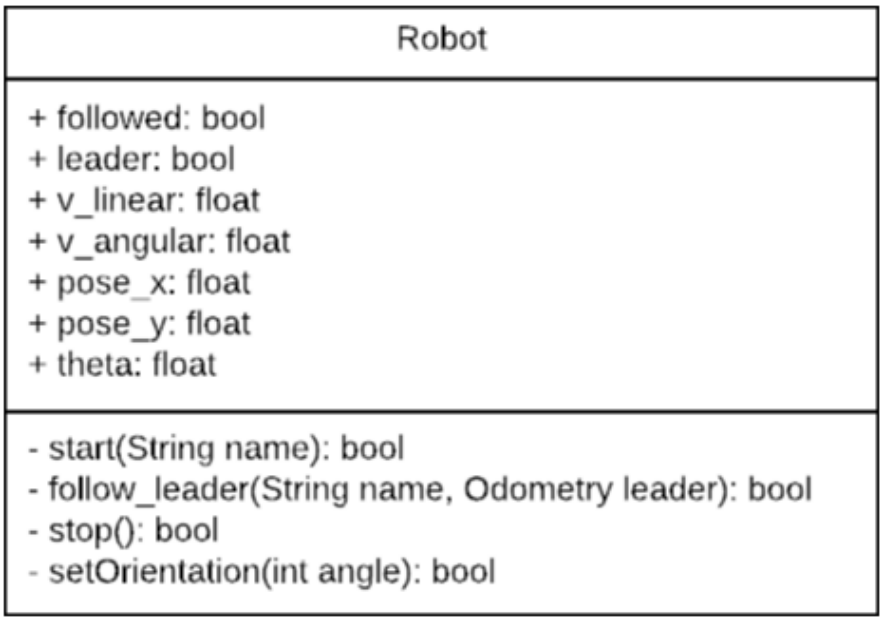
$$D = \sqrt{(L_x - S_x)^2 + (L_y - S_y)^2}$$

L_x = Posição X do Líder L_y = Posição Y do Líder
S_x = Posição X do Seguidor S_y = Posição Y do Seguidor

4.4 LÓGICA COMPUTACIONAL

Para realizar estas operações foram utilizados conceitos de POO para conclusão desta metodologia, se baseando no conceito de objetos, possibilitando obter uma melhor visão do projeto devido a abstração necessária para realização do mesmo, assim podendo gerar artefatos de engenharia de software que auxiliam no desenvolvimento do projeto, demonstrado na Figura 7.

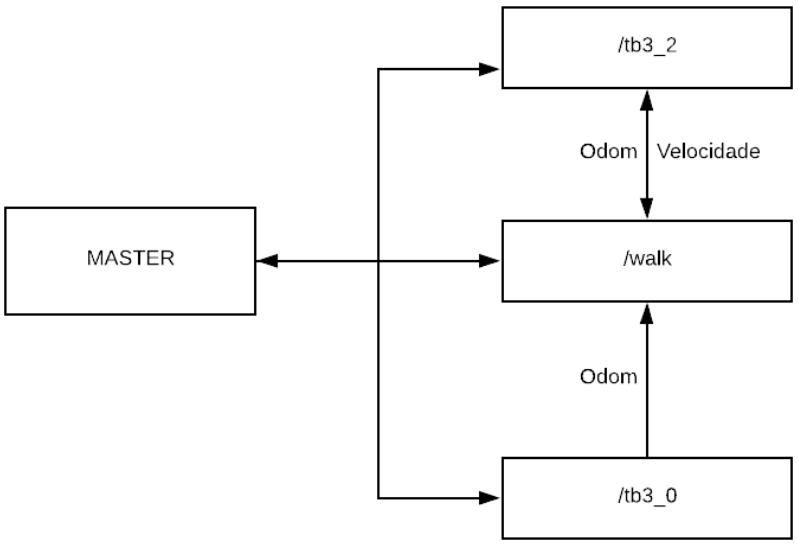
Figura 8 – Diagrama de classe



Fonte: Elaborada pelo Autor, 2019

Foi utilizado princípios de POO pois se faz similar ao modelo utilizado pelo ROS visto na Figura 8. Fazendo uma analogia, o Master seria como um objeto e cada nó seria como um método. Em contrapartida não teria a ligação entre os métodos, assim apenas os métodos servindo o objeto.

Figura 9 – Modelo utilizado pelo ROS



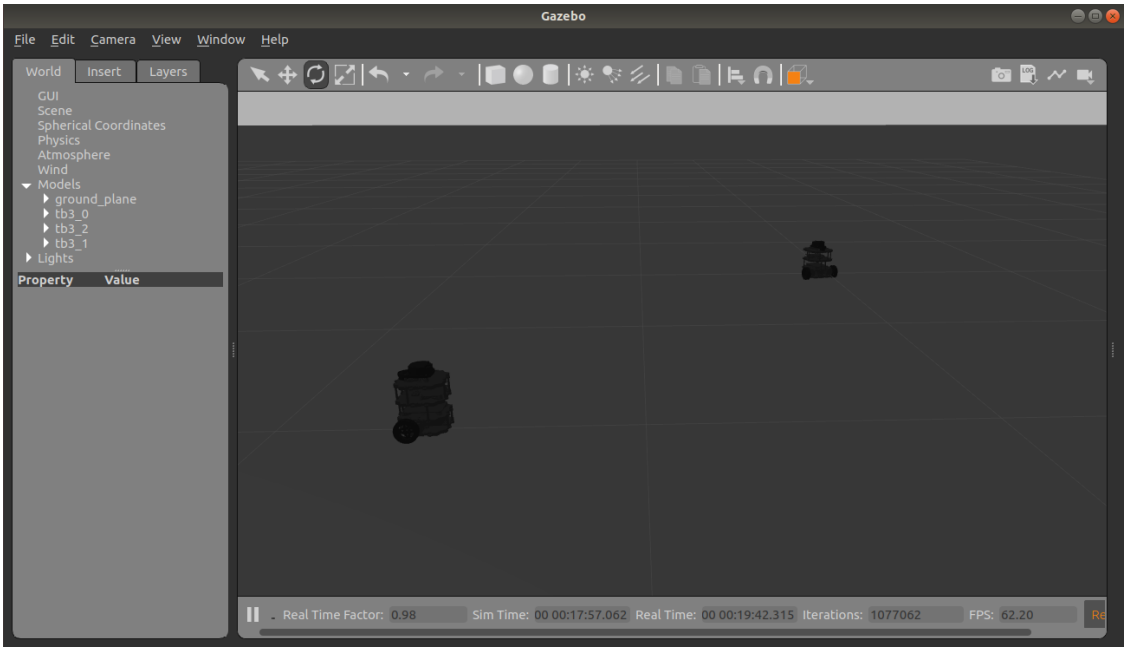
Fonte: Elaborada pelo Autor, 2019

4.5 SIMULAÇÃO

4.5.1 Testes

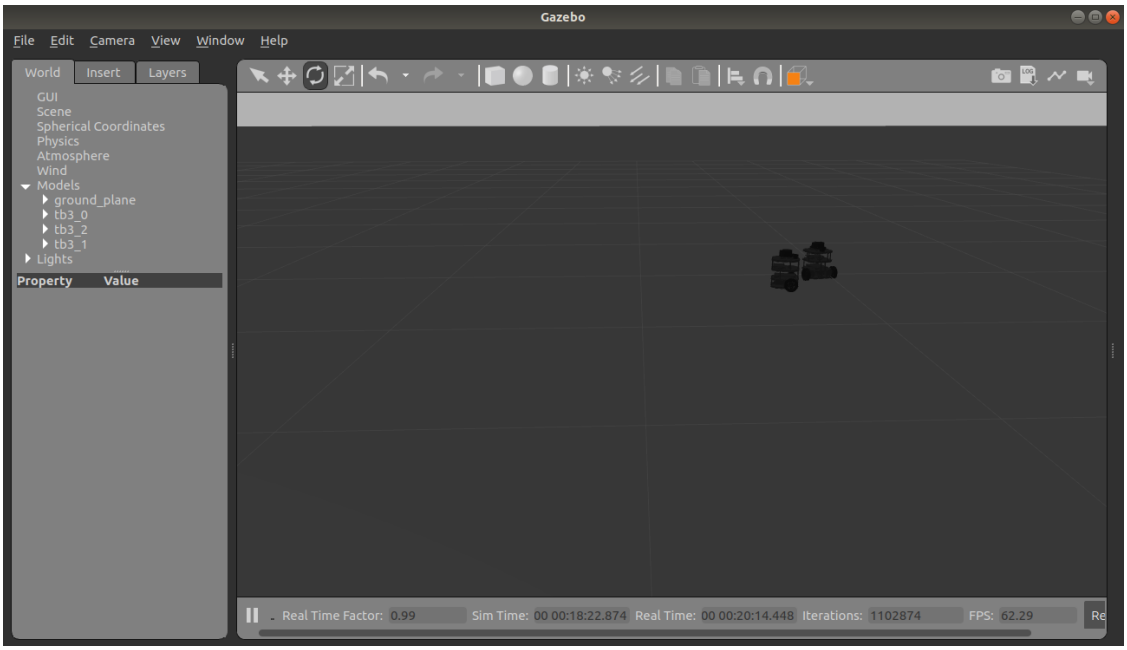
Para este trabalho foi criado uma simulação com um ambiente aberto e sem obstáculos. Mesmo com um ambiente básico houve dificuldades devido ao ponto (0,0) do mapa ser no centro do mesmo. Os resultados obtidos antes e depois do algoritmo foram ilustrados nas figuras 9 e 10, respectivamente.

Figura 10 – Antes do Algoritmo



Fonte: Elaborada pelo Autor, 2019

Figura 11 – Depois do Algoritmo



Fonte: Elaborada pelo Autor, 2019

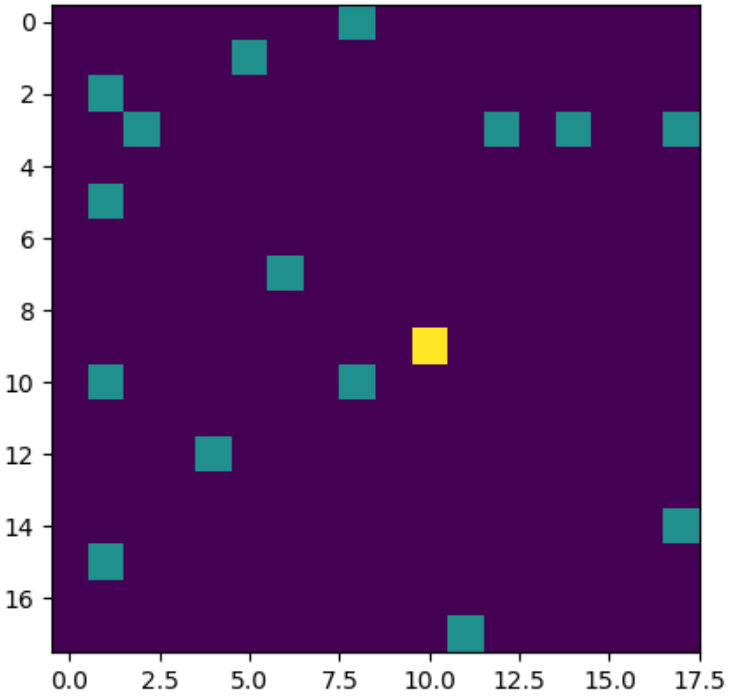
4.6 METODOLOGIA FINAL

A partir deste momento vamos chamar os robôs de partículas e a movimentação de cada uma como casas de um xadrez, logo teremos que aplicar a optimização por enxame de partículas em nosso algoritmo líder-seguidor, utilizando critérios para movimentação impostos em cada partícula, tendo como inspiração um bando de pássaros temos tais critérios:

1. Cada partícula tende a se movimentar na direção da sua inércia;
2. Cada partícula tende a se movimentar em direção a seu vizinho (pbest); e
3. Cada partícula tende a se movimentar em direção ao grupo (gbest).

Para implementação desses critérios de movimentação foi utilizada a linguagem Python junto com bibliotecas e ferramentas como *numpy* e *matplotlib*, teremos um mapa onde será formado por uma matriz onde foi criado um padrão para os valores da mesma representarem as partículas e o alvo. Sendo assim caso tenha o valor igual a 1 significa que é uma partícula, caso tenha valor igual a -1 significa que é o alvo e caso tenha valor igual a 0 significa que é um lugar vazio no mapa, visualmente teremos o mapa deste modo:

Figura 12 – Mapa simulado utilizando *matplotlib* (15 partículas)



Fonte: Elaborada pelo Autor, 2019

4.7 LÓGICA FINAL

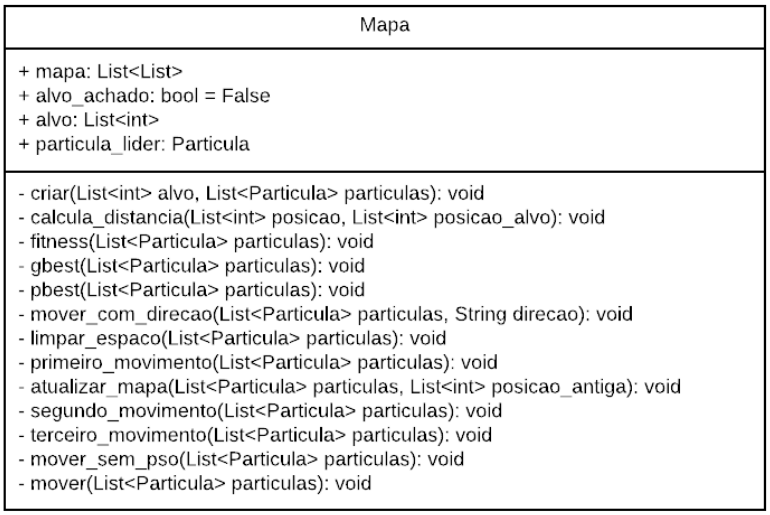
Como dito na metodologia temos 3 fatores que influenciam na movimentação da partícula e que são atualizados a cada interação que é realizada, o primeiro é feito movendo uma casa na orientação na qual a partícula se encontra, orientação sendo uma variável aleatória onde se pode ter o valor de 0, 90, 180, 270. Já o segundo é calculado para cada partícula verificando qual partícula que não é ela mesma está mais perto dela, em seguida o terceiro é calculado fazendo a média de todas as posições.

Para realização deste trabalho precisamos definir um líder, o método usado para definição de líder foi determinando qual partícula está mais perto do alvo (em amarelo no mapa). Logo caso usássemos o PSO puramente dito teríamos apenas uma aglomeração de partículas onde nunca iriam ir em direção ao líder, para isso foi feito uma alteração onde o gbest (melhor posição do enxame) é dito como a posição do líder e tem uma prioridade maior no fator da deslocação.

Contudo por consequência da utilização de um mapa de autoria própria acaba por trazer a necessidade de algumas validações e tratamentos, como caso a partícula tente ir para um lugar fora do alcance do mapa. Já que o mesmo não é faz uso de colisões, além da devida criação do mapa, setando as posições iniciais de cada partícula e do alvo.

Para que haja abstração vantajosa de tais operações, foram criadas classes que fazendo uso dos princípios de POO, gerando artefatos que podem ser vistos nas Figuras 12 e 13.

Figura 13 – Diagrama de classe - Mapa



Fonte: Elaborada pelo Autor, 2019

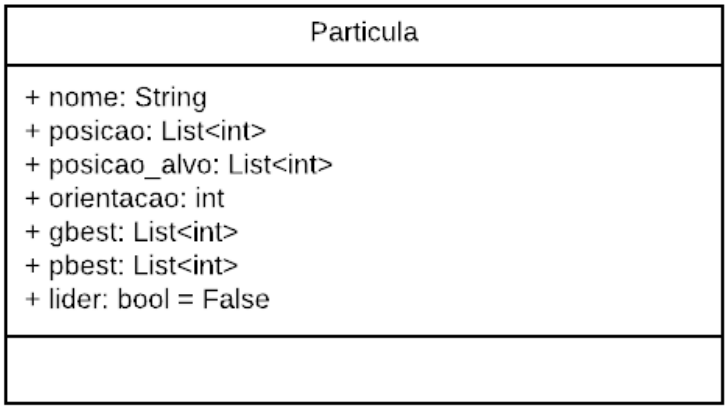
Na Figura 12 temos o diagrama de classe do Mapa que foi criado, a lógica em cima do mesmo deu-se na criação de 4 atributos, todos estes atributos estão situados no Mapa por conta do grande principio do enxame de robôs, onde o conhecimento é globalizado e cada robô tem controle mínimo sobre os recursos do meio, desta forma temos atributos como *"alvo_achado"* que serve como critério de início do uso do PSO, *"alvo"* que como seu nome já remete, é a posição do alvo e *"particula_lider"* refere-se a partícula que vai se tornar o *gbest* das outras partículas. Temos aqui as funções mais importantes na abstração:

- **criar()**: responsável por setar as posições tanto das partículas quanto do alvo, definir se o alvo foi achado, verificar qual partícula é a líder.
- **fitness()**: faz a chamada do **pbest()** e **gbest()**, os atualizando para a próxima interação.
- **limpar_espaco()**: primeira função chamada ao iniciar cada etapa de movimentação, sendo encarregado de fazer a liberação do espaço no qual a partícula ocupava antes de se mover.
- **atualizar_mapa()**: ultima função chamada ao inicio cada etapa de movimentação, sendo encarregada de fazer a ocupação do espaço no qual a partícula se moveu, com as devidas validações sobre já ter algo na posição que ela quer ir, caso seja o fim do mapa ou uma outra partícula ele acaba por seguir para a direita. Caso seja o alvo o mapa e partículas mudam de cor para sinalizar que o líder chegou ao alvo.

- **mover()**: faz a chamada de cada parte do movimento, sempre verificando se a partícula a se mover não é o líder e a posição não esta perto do alvo ou for o líder e a posição for diferente da posição do alvo, ao final da movimentação que é chamado o método **fitness()**.

À vista disso temos a classe Particula na Figura 13 que só cuidará de saber basicamente valores sobre sua localização e conhecer se é o líder ou não.

Figura 14 – Diagrama de classe - Particula



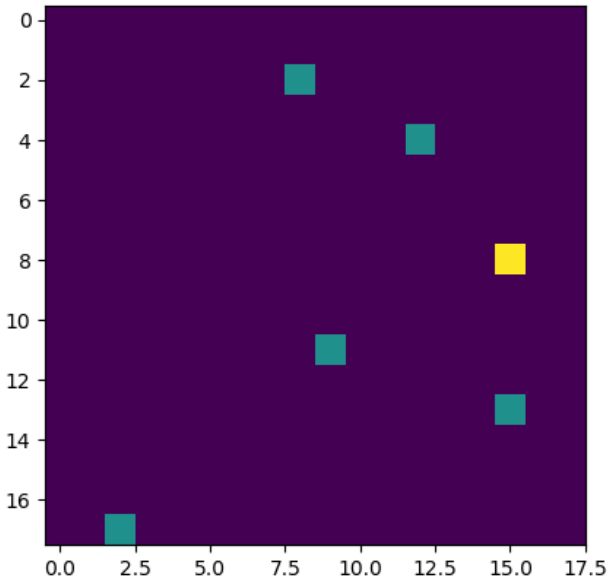
Fonte: Elaborada pelo Autor, 2019

4.8 RESULTADOS FINAL

Para fazer a comparação entre a técnica líder-seguidor com e sem o uso de PSO foram realizados uma bateria de testes com diferentes quantidades de partículas e quantidade de interações diferentes, tendo como padrão 5 testes para cada modo, variando de 5, 10, 15 partículas e 1, 2, 3 interações.

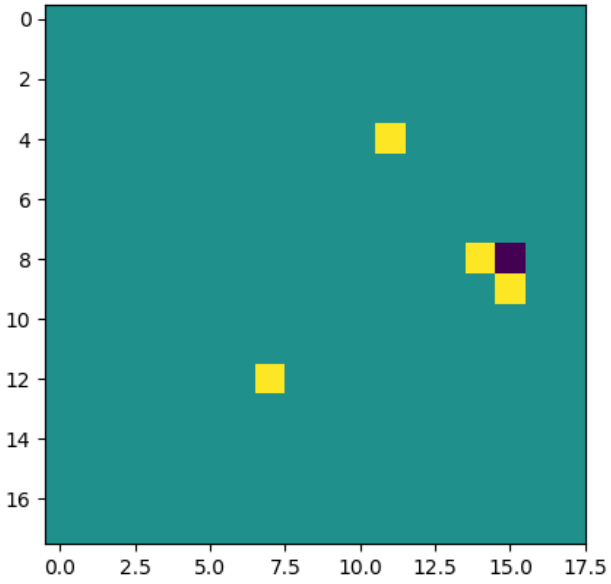
Sendo assim, dentre os testes com 5 partículas houve dois teste com 1 interação onde o algoritmo sem a utilização do PSO foi mais proveitoso, fazendo com que o líder chegasse no alvo, como podemos ver nas figuras abaixo:

Figura 15 – Inicio - 5 partículas, 1 interação



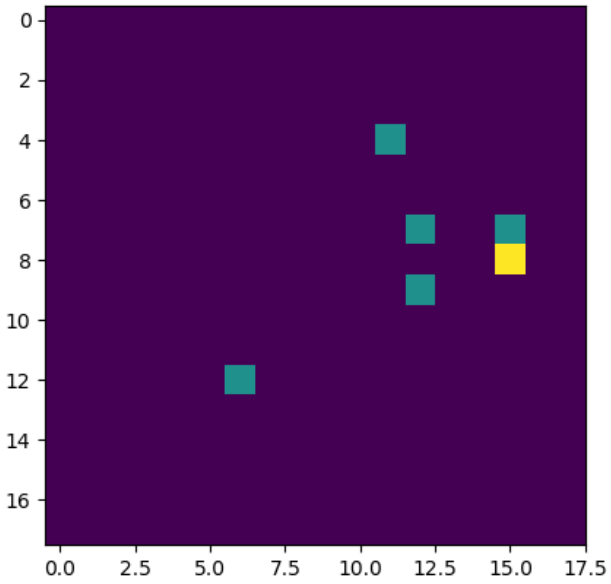
Fonte: Elaborada pelo Autor, 2019

Figura 16 – Fim sem PSO - 5 partículas, 1 interação



Fonte: Elaborada pelo Autor, 2019

Figura 17 – Fim com PSO - 5 partículas, 1 interação

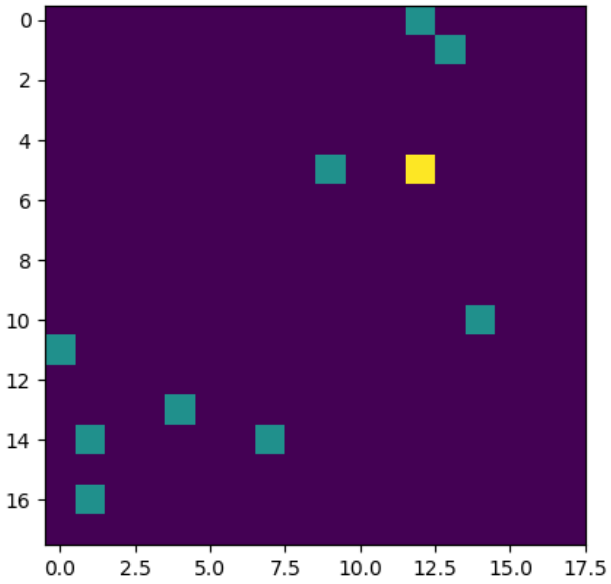


Fonte: Elaborada pelo Autor, 2019

Como podemos ver acima, sendo o alvo mostrado em amarelo, não tendo uma distinção visível para qual indivíduo é o líder e caso o líder esteja na posição do alvo o mapa muda de cor sinalizando que o líder não realizará mais movimentos e o seguidores se tornam da cor amarela. Já em um teste com 2 interações e 15 partículas, houve um caso onde o PSO foi mais conveniente, lembrando que a orientação e a posição de cada partícula são fatores aleatórios para cada cliclo de teste.

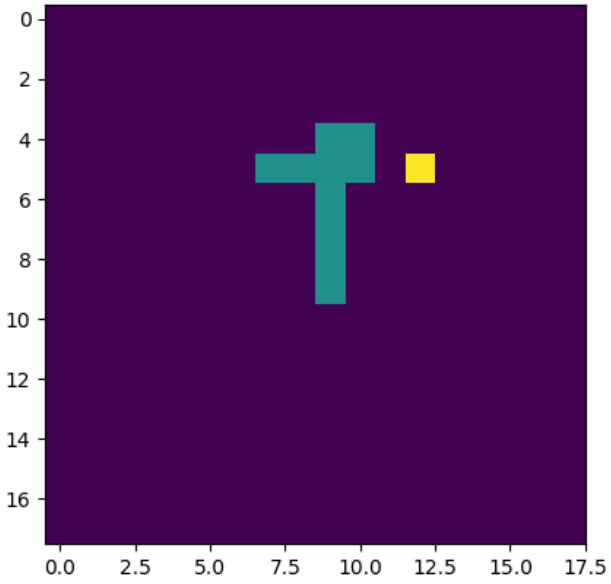
Contudo no geral, o algoritmo utilizando o PSO tende a perder um pouco em relação ao algoritmo isolado de líder-seguidor pois além das partículas terem que se locomover em direção ao líder, elas também se movem em direção aos seus vizinhos e em direção a sua inércia, perdendo vantagem caso no primeiro movimento sua orientação seja contraria a posição do líder. Por outro lado, foi observado que à medida que a quantidade de interações vai aumentando, mais o algoritmo com PSO encaminha-se para um contexto vantajoso, assim descrito nas Figuras abaixo:

Figura 18 – Inicio - 10 partículas, 3 interações



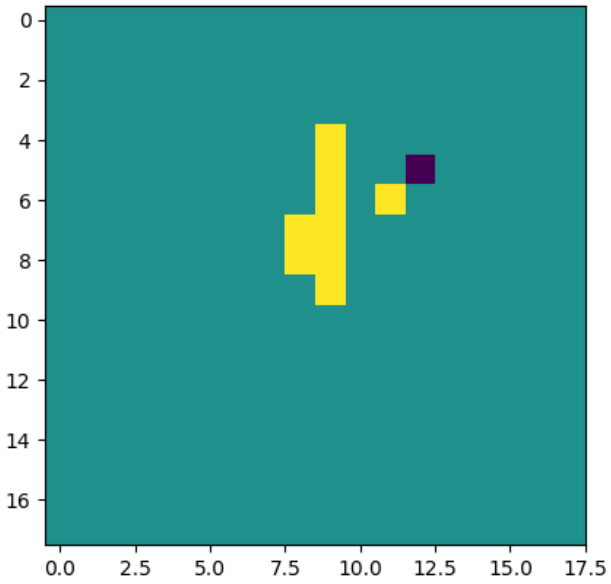
Fonte: Elaborada pelo Autor, 2019

Figura 19 – Fim sem PSO - 10 partículas, 3 interações



Fonte: Elaborada pelo Autor, 2019

Figura 20 – Fim com PSO - 10 partículas, 3 interações



Fonte: Elaborada pelo Autor, 2019

5 CONCLUSÃO

Com o ROS sendo uma ferramenta open-source muito prática e funcional, acaba por abrir portas para a robôtica para pessoas que não tem muito conhecimento na parte da mecânica, sendo assim, não só técnicas líder-seguidor e otimizações por partículas podem ser exploradas e integradas a inteligencia de enxame.

Algumas utilizações de técnicas líder-seguidor podem ser desde a criação de carros autônomos até o carregamento de cargas pesadas, podendo haver uma integração de robôs e drones para um objetivos em comum, outra possível melhora seria um protocolo de formação de grupo, dividir a liderança ou alterar entre os líderes.

REFERÊNCIAS

ATTUX, R. R. F.; ZUNBEN, F. J. V. Inteligência de Enxame. p. 1–80, 1980. Citado na página 10.

CARACIOLO, M. *Introdução à Inteligência de Enxame - Otimização por Enxame de Partículas (PSO)*. [S.l.], 2009. <http://engineering.purdue.edu/~mark/puthesis> [Acessado em Maio. 15, 2019]. Citado na página 8.

Doctor, S.; Venayagamoorthy, G. K.; Gudise, V. G. Optimal pso for collective robotic search applications. In: *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*. [S.l.: s.n.], 2004. v. 2, p. 1390–1395 Vol.2. Citado na página 11.

E-MANUAL. *TurtleBot3*. 2019. Disponível em: <http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/>. Acesso em: 19 abr. 2019. Citado na página 15.

G, S.; A, A. S.; HENCY, V. A survey on swarm robotic modeling, analysis and hardware architecture. *Procedia Computer Science*, v. 133, p. 478–485, 07 2018. Citado 2 vezes nas páginas 7 e 8.

GARG, A. et al. An insight into swarm intelligence. *International Journal of Recent Trends in Engineering*, v. 2, 01 2009. Citado na página 7.

Koenig, N.; Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. [S.l.: s.n.], 2004. v. 3, p. 2149–2154 vol.3. Citado na página 11.

KRAUSE, J.; RUXTON, G. D.; KRAUSE, S. *Swarm intelligence in animals and humans*. 2010. Citado na página 10.

LI, X.; CLERC, M. Swarm Intelligence. In: *International Series in Operations Research and Management Science*. [S.l.: s.n.], 2019. Citado na página 11.

MANCUSO, S. *The Roots of Plant Intelligence*. 2010. Disponível em: <https://www.ted.com/talks/stefano_mancuso_the_roots_of_plant_intelligence?language=pt-br>. Acesso em: 19 abr. 2019. Citado na página 10.

MANSOR, H.; ADOM, A.; RAHIM, N. A. Development of leader and follower strategy for swarm robot applications. *Jurnal Teknologi*, v. 77, 12 2015. Citado na página 9.

ROBOTIS. *Specifications*. 2019. Disponível em: <http://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/>. Acesso em: 19 abr. 2019. Citado na página 15.

ROS. *Nodes*. 2019. Disponível em: <http://wiki.ros.org/Nodes>. Acesso em: 19 abr. 2019. Citado na página 10.

ROS. *roscore*. 2019. Disponível em: <http://wiki.ros.org/roscore>. Acesso em: 19 abr. 2019. Citado na página 15.

ROS. *ROS.org | About ROS*. 2019. Disponível em: <http://www.ros.org/about-ros/>. Acesso em: 19 abr. 2019. Citado na página 10.

ROS. $rqt_graph.2019.Disponvelem$: <>. Acesso em: 19 abr. 2019. Citado na página 15.

ROS. *Topics*. 2019. Disponível em: <http://wiki.ros.org/Topics>. Acesso em: 19 abr. 2019. Citado na página 11.

SAHIN, E.; SPEARS, W.; WINFIELD, A. *Swarm Robotics*. [S.l.: s.n.], 2007. Citado na página 8.

Shia, A.; Bastani, F. B.; Yen, I. A highly resilient framework for autonomous robotic swarm systems operating in unknown, hostile environments. In: *2011 Tenth International Symposium on Autonomous Decentralized Systems*. [S.l.: s.n.], 2011. p. 147–153. ISSN 1541-0056. Citado na página 9.

SINGH, A.; KUMAR, A.; SINGH, G. Solid waste routing by exploiting ant colony optimization. In: . [S.l.: s.n.], 2011. Citado na página 7.

SUN, Z. et al. Leader-follower formation control of underactuated surface vehicles based on sliding mode control and parameter estimation. *ISA Transactions*, v. 72, p. 15 – 24, 2018. ISSN 0019-0578. Disponível em: <http://www.sciencedirect.com/science/article/pii/S001905781730616X>. Citado na página 14.

Swaminathan, S.; Phillips, M.; Likhachev, M. Planning for multi-agent teams with leader switching. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.: s.n.], 2015. p. 5403–5410. ISSN 1050-4729. Citado na página 13.

TAN, Y.; ZHENG, Z. yang. Research advance in swarm robotics. *Defence Technology*, v. 9, n. 1, p. 18 – 39, 2013. ISSN 2214-9147. Disponível em: <http://www.sciencedirect.com/science/article/pii/S221491471300024X>. Citado na página 8.

VERLEKAR, H.; JOSHI, K. Ant & bee inspired foraging swarm robots using computer vision. *International Conference on Electrical, Electronics, Communication Computer Technologies and Optimization Techniques, ICEECCOT 2017*, v. 2018-January, p. 191–195, 2018. Citado 2 vezes nas páginas 7 e 8.

VILLA, M. A. M. et al. Ros-based human leader and robot follower using a pioneer 3-dx robot. *Revista UIS Ingenierías*, v. 15, p. 63–71, 01 2017. Citado na página 11.

Wahi, G.; Raj, A. N. J.; Zhun Fan. Swarm bots: System design for echolocation. In: *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*. [S.l.: s.n.], 2017. p. 229–232. Citado na página 12.