UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Estudo de Algoritmos para Controle de um Enxame de Quadrotores

Rafael Gomes Braga

Itajubá, Fevereiro de 2019

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Rafael Gomes Braga

Estudo de Algoritmos para Controle de um Enxame de Quadrotores

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação

Área de Concentração: Matemática da Computação

Orientador: Prof. Dr. Alexandre Carlos Brandão Ramos

Fevereiro de 2019
Itajubá - MG

Dedico à minha família e à Natália por todo o apoio durante o desenvolvimento deste trabalho.

Agradecimentos

Agradeço primeiramente a Deus pela saúde e pelas bençãos que me permitiram passar por todo esse processo.

Aos meus pais e familiares pelo apoio incondicional.

Ao meu orientador, Alexandre Carlos Brandão Ramos por todas as conversas, ensinamentos e incentivos, por estar sempre disponível e disposto a me ajudar.

À banca examinadora, pela leitura cuidadosa e pelas valiosas colaborações para a redação final deste texto.

Aos meus colegas da UNIFEI, em especial Adriano, Audeliano, Gustavo e João Paulo, pela disposição e por todas as dicas importantíssimas que me forneceram. Ao Wander pelas dicas relativas à dissertação.

Ao professor Guilherme Souza Bastos pelas dicas, pelo apoio e por todo o encorajamento nos meus momentos de indecisão.

E à Natália, minha namorada, que me deu forças e esteve presente em todos os momentos, batalhando junto comigo.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES). Código de Financiamento 001.

A frase mais empolgante de ouvir em ciência,
a que prenuncia novas descobertas,
não é "Eureka!", mas sim "Isto é estranho..."
(Isaac Asimov)

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO
EM CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Rafael Gomes Braga

Estudo de Algoritmos para Controle de um Enxame de Quadrotores

Dissertação aprovada por banca examinadora em 22 de Fevereiro de 2019, conferindo ao autor o título de **Mestre em Ciência e Tecnologia da Computação**.

Banca Examinadora:
Prof. Dr. Alexandre Carlos Brandão Ramos - UNIFEI (Orientador)
Prof. Dr. Roberto Claudino da Silva - UNIFEI (Coorientador)
Prof. Dr. Hildebrando Ferreira de Castro Filho - ITA
Prof. Dr. Roberto Affonso da Costa Júnior - UNIFEI

Itajubá
2019

Resumo

Um dos desafios mais atuais da Robótica Móvel é o controle de múltiplos VANTs (Veículos Aéreos Não Tripulados) enquanto realizam juntos uma tarefa de forma cooperativa. Diversas estratégias têm sido propostas, cada uma com suas vantagens e desvantagens. Após o estudo de várias estratégias, esse trabalho propõe um algoritmo para controle de enxames de VANTs baseado em comportamentos observados em animais sociais na natureza. Para que os VANTs realizem o comportamento desejado, esse algoritmo, escrito em C++ na plataforma ROS, utiliza as três regras de Reynolds: Coesão, Alinhamento e Separação. Para avaliá-lo, foi construído um ambiente de simulação baseado no simulador Gazebo e diversos testes simulados foram realizados. Para explorar uma possível aplicação, foi proposto um cenário em que o grupo de VANTs deve procurar pela fonte de um vazamento de um gás tóxico. Além disso, para aproveitar as vantagens das diferentes técnicas de controle de enxames, e minimizar suas desvantagens, foram incorporados os conceitos de líder-seguidor e estrutura virtual a fim de se obter também o controle da formação dos VANTs.

Palavras-chaves: Enxames Robóticos, Veículos Aéreos Não Tripulados, Regras Baseadas em Comportamento, ROS, Gazebo.

Abstract

One of the current challenges in Mobile Robotics is the control of multiple UAVs (Unmanned Aerial Vehicles) while they perform a task together cooperatively. Many strategies have been proposed, each one with its advantages and disadvantages. After the study of many strategies, this work proposes an UAV swarm control algorithm based on behaviors observed in social animals in nature. This algorithm, written in C++ in the ROS platform, uses the three Reynolds rules of Cohesion, Alignment and Separation to obtain the expected behavior from the UAVs. To evaluate it, a simulation environment was build, based on the Gazebo simulator, and many tests were conducted. To explore a possible application, a scenario was proposed in which the UAV group has to search for the source of a toxic leaking gas. Further more, to leverage the advantages of the different swarm control techniques, and minimize their disadvantages, the concepts of leader-follower and virtual structure were incorporated in order to obtain also the control of the formation of the UAVs.

Keywords: Robotic Swarms, Unmanned Aerial Vehicles, Behavior Based Rules, ROS, Gazebo.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Glossário

SCC	*Sistema de Coordenadas Cartesianas*
S_i	*SCC Inercial*
S_v	*SCC do Veículo*
S_b	*SCC do Corpo*
$\mathbf{p}_i$	*Posição de um corpo expressa no SCC i*
$\mathbf{o}_i$	*Orientação de um corpo expressa no SCC i*
R_a^b	*Matriz de Rotação do SCC S_a para o SCC S_b*
m	*Massa do quadrotor*
g	*Aceleração da gravidade $g = 9,81m/s^2$*
f_i	*Empuxo produzido pelo i-ésimo rotor*
τ_i	*Torque de reação produzido pelo i-ésimo rotor*
J	*Matriz de Inércia do quadrotor*
ω	*Velocidade angular*
M	*Momento total agindo sobre o quadrotor*
ρ	*Densidade do ar*
A	*Área da hélice*
C_T	*Coeficiente de força aerodinâmica*
C_D	*Coeficientes de momento aerodinâmico*
r	*Raio da hélice*
Ωi	*Velocidade angular do rotor i*
K_f	*Constante de força aerodinâmica*
K_m	*Constante de momento aerodinâmico*
F_B	*Forças não gravitacionais agindo no SCC fixo no corpo*
$Pose_i$	*Pose do i-ésimo quadrotor no exame*

1 Introdução

Este capítulo discutirá as motivações e objetivos do desenvolvimento deste trabalho. Será apresentada uma breve introdução sobre VANTs, suas características, tipos e aplicações, e também serão mencionadas as vantagens do uso de enxames de VANTs. Em seguida serão descritas as contribuições obtidas pelo trabalho. Por fim, a estrutura da dissertação será delineada em detalhes.

1.1 Motivação

Nas últimas duas décadas tem havido um crescente interesse por parte da sociedade e da comunidade científica pelos veículos aéreos em miniatura, conhecidos popularmente como *drones*. Isso se deve ao avanço recente nas tecnologias de computadores e sensores embarcados, que estão se tornando cada vez menores, mais poderosos e baratos, o que permitiu o desenvolvimento de modelos comerciais de *drones* bastante robustos e acessíveis.

Embora o nome mais popular para esse tipo de dispositivo seja *drone* (que em inglês significa zangão), um nome que os representa de forma mais acurada quando se trata de aplicações científicas é Veículo Aéreo Não Tripulado, ou VANT, de forma abreviada. Isso se deve ao fato de que essas aeronaves não carregam pilotos humanos a bordo, sendo controladas remotamente ou recebendo comandos a partir de um computador embarcado. Por esse motivo será adotado o nome VANT nesse trabalho.

Além da disponibilidade de tecnologia, várias características dos VANTs justificam sua popularidade: são capazes de se mover em três dimensões, carregar diversos tipos de sensores, transportar carga, executar manobras, entre outras. Graças a esses fatores, são utilizados em uma gama imensa de aplicações, tanto civis como militares, entre as quais podemos citar: monitoramento e combate de incêndios florestais (MERINO et al., 2010)(OLLERO; DIOS; MERINO, 2006), inspeção de prédios e pontes (CHOI; KIM, 2015), dispersão de agrotóxicos em plantações (PEDERI, 2015), busca e resgate de sobreviventes após um desastre (BAKER et al., 2016), vigilância e monitoramento (GENG et al., 2014) ou mesmo ataques aéreos (LI et al., 2016).

Existem vários tipos de VANTs, cada um apresentando certas vantagens e desvantagens em relação aos outros. Os principais tipos são:

- **VANTs de asa fixa:** Precisam percorrer uma distância em solo antes de decolarem, porém podem voar por um longo tempo e atingirem grandes velocidades. São utilizados principalmente em aplicações meteorológicas, mapeamento e ações militares;

- **Helicópteros:** São capazes de decolar e pousar verticalmente e possuem boa manobrabilidade. Porém, seu controle é muito complexo;

- **Multirotores:** Também possuem a capacidade de decolar e pousar verticalmente, podem ficar suspensos no ar de forma estável (*hover*) e como possuem diversas hélices, são capazes de transportar cargas relativamente pesadas. Porém, devido à esse mesmo motivo, apresentam um consumo de energia maior e portanto menor autonomia.

Entre esses tipos o mais popular é o quadrotor, que é um multirotor que apresenta quatro hélices. Além das características citadas o quadrotor é de fácil construção e controle, sendo também mais barato devido ao número menor de motores.

Uma vez que o controle do quadrotor já é um problema bem compreendido, uma nova área que tem atraído interesse é o uso de um grupo de VANTs em uma missão, ao invés de apenas uma unidade. Essa estratégia traz muitas vantagens, entre as quais podemos citar:

- O grupo carrega muitos sensores e portanto é capaz de obter mais dados e cobrir uma maior área que apenas um robô;

- Pelo mesmo motivo o grupo é mais atento ao que o cerca e portanto é capaz de se defender melhor de ameaças;

- O grupo é mais robusto contra falhas, pois se um robô falhar os outros podem continuar a missão sem ele.

Porém, enquanto essa estratégia traz muitas vantagens, o controle de enxames é muito mais complexo e traz diversos novos desafios. Considerações devem ser feitas a respeito da comunicação entre VANTs, divisão de tarefas, detecção de obstáculos e outros membros do enxame e mesmo o posicionamento espacial dos robôs. Cada um desses tópicos é investigado extensivamente por diversos trabalhos encontrados na literatura. Este trabalho se foca na questão do posicionamento espacial dos VANTs, de forma a evitar colisões e manter formações.

Diversas ferramentas estão disponíveis atualmente para ajudar no projeto e teste desse tipo de aplicação. O ROS e o simulador Gazebo estão se tornando padrões da indústria e da comunidade científica na área da robótica. Essas ferramentas permitem a implementação de algoritmos de controle e testes através de simulações realísticas de forma rápida e padronizada, portanto foram escolhidas para serem utilizadas nesse trabalho.

1.2 Objetivos

Baseado nos trabalhos previamente citados, o objetivo dessa pesquisa é desenvolver um algoritmo de controle de enxames de VANTs capaz de mover um grupo de quadrotores como um grupo coeso enquanto evitam colisões uns com os outros. Para isso foi utilizada uma estratégia baseada em comportamentos conhecida como regras de Reynolds. O grupo pode se mover para uma posição arbitrária ou, utilizando as interações entre os membros do grupo e uma regra baseada em otimização de enxame de partículas, percorrer uma área enquanto procura pela fonte de um vazamento de gás. Indo além do controle básico também foi proposta uma estratégia para controle da formação do grupo, combinando a estratégia baseada em comportamento com estrutura virtual e líder seguidor.

1.3 Contribuições

A principal contribuição desse trabalho é a criação de um algoritmo capaz de mover um grupo de VANTs juntos como um enxame coeso enquanto evitam colisões uns com os outros. Diferente de outros trabalhos, aqui os VANTs voam na mesma altitude, de forma que a capacidade de evitar colisões é fundamental. O programa foi escrito em C++ na plataforma ROS e está adequado com as principais padronizações adotadas mundialmente pelo ROS. Com isso, esse programa poderá ser utilizado para diversas outras aplicações e servir como base para outras pesquisas.

Além disso, outra contribuição é a criação de um ambiente de simulação de quadrotores baseado no simulador Gazebo. Esse simulador permite o teste de diversos algoritmos de controle em cenários muito próximos da realidade. Todo o código desenvolvido será disponibilizado para que novos alunos e pesquisadores possam testar seus trabalhos de forma rápida e prática no futuro.

1.4 Estrutura do Trabalho

Este trabalho é estruturado da seguinte forma: o capítulo 1 ofereceu uma introdução ao problema tratado e falou sobre as motivações e contribuições deste trabalho. O capítulo 2 apresenta uma revisão da literatura a respeito da robótica de enxames e regras de Reynolds. No capítulo 3 é detalhada a modelagem utilizada para controlar o quadrotor e a modelagem do sistema com múltiplos VANTs. O capítulo 4 mostra os detalhes da implementação do algoritmo proposto, passando pelas considerações de *hardware* e *software*. No capítulo 5 são explicados os experimentos propostos para validar o algoritmo e são apresentados os resultados obtidos. Por fim, o capítulo 6 traz conclusões a respeito dos resultados e discute possíveis trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresentará uma breve revisão da literatura relacionada ao tema desse trabalho. Primeiramente será caracterizado o conceito de enxames robóticos. Em seguida serão detalhados os principais tipos de comportamentos que são desejados nos enxames e as técnicas comumente utilizadas para obter esses comportamentos serão descritas. Por fim o conceito da Otimização por Enxame de Partículas, uma popular técnica de inteligência de enxames, será apresentada de forma resumida.

2.1 Robótica de Enxames

A Robótica de Enxames é uma área de estudos relativamente nova que se preocupa com a utilização de sistemas formados por múltiplos robôs autônomos para realizar tarefas complexas que não podem ser realizadas por indivíduos únicos ou são executadas de forma mais eficiente por grupos. Essa área de estudo busca inspiração na natureza, onde são encontradas espécies de animais que apresentam comportamentos sociais de grupo, como bandos de pássaros, cardumes de peixes ou rebanhos de animais terrestres (TAN; ZHENG, 2013). Nota-se que animais simples conseguem ser bem sucedidos em certas tarefas quando se juntam em grupos, pois os grupos parecem ter algum tipo de "inteligência de enxame"(BONABEAU et al., 1999). Essa expressão se refere às capacidades superiores que os grupos possuem em relação aos seus indivíduos.

O principal objetivo da Robótica de Enxames é o projeto de sistemas formados por múltiplos robôs minimalistas que, através de regras simples e interações locais acabam gerando um comportamento global desejado, mesmo que robôs não tenham sido explicitamente programados para cumprir esse comportamento específico (BRAMBILLA et al., 2013). É dito que o comportamento global emerge a partir das interações locais entre os robôs. Enxames são especialmente efetivos em aplicações envolvendo exploração e busca em ambientes desconhecidos, uma vez que um número grande de robôs pode cobrir uma área maior e mais rapidamente que uma única unidade.

As principais características de um enxame robótico são (BRAMBILLA et al., 2013):

- Robôs são autônomos;
- As capacidades de sensoreamento e comunicação são locais;
- Robôs não possuem um controle central (e podem não carregar conhecimentos globais);
- Os robôs cooperam para executar as tarefas.

Três propriedades são desejadas em enxames robóticos (ŞAHIN, 2004):

- **Robustes:** é a capacidade de continuar operando mesmo que haja a perda de indivíduos. Isso é obtido através de redundância e mantendo os membros do enxame homogêneos (robôs idênticos);
- **Escalabilidade:** é a característica de funcionar bem em grupos de tamanhos diferentes. A performance deve se manter parecida ao se aumentar ou diminuir o número de robôs. Isso é obtido forçando as interações e comunicação a serem locais;
- **Flexibilidade:** é a capacidade de o enxame lidar com diferentes ambientes e tarefas. Isso é obtido através de redundância e simplicidade nos comportamentos.

2.2 Controle de Enxames Baseado em Comportamentos

Uma forma bastante comum de desenvolvimento de enxames robóticos é a chamada "baseada em comportamentos". Essa técnica consiste no estudo de comportamentos simples apresentados por seres vivos na natureza de forma a entendê-los e modelá-los matematicamente, para então implementá-los nos robôs. Espera-se que esses comportamentos simples gerem comportamentos mais complexos.

Em geral é um processo interativo em que o comportamento individual dos robôs é implementado e ajustado até que o comportamento coletivo seja obtido. Costuma envolver tentativa e erro.

Vários comportamentos mais simples podem ser combinados de forma a atacar um problema do mundo real. Vários tipos de comportamento são estudados, envolvendo posicionamento espacial dos robôs, navegação e busca de objetivos específicos.

2.3 Técnicas de Controle de Enxames

A maioria dos algoritmos de controle propostos na literatura pode ser classificado em três tipos: líder-seguidor (GU et al., 2006)(AMBROZIAK; GOSIEWSKI, 2015), estrutura virtual (EGERSTEDT; HU; STOTSKY, 2001)(ASKARI; MORTAZAVI; TALEBI, 2015) ou baseado em comportamento (BALCH; ARKIN, 1998)(VIRÁGH et al., 2014). Na estratégia líder-seguidor um dos VANTs no grupo é escolhido como o líder. Os outros têm que segui-lo e se posicionar de acordo com a sua posição. Essa abordagem possui a vantagem de ser mais fácil para um operador humano pilotar o grupo, tendo que controlar apenas o líder. Porém uma desvantagem é que se o líder parar de funcionar, o grupo todo também para. A estratégia de estrutura virtual consiste em tratar o grupo inteiro como uma única estrutura fixa, com cada VANT representando um ponto que a compõe. O controlador é projetado de forma que cada VANT se mova para criar o comportamento

desejado da estrutura. Finalmente na estratégia baseada em comportamento os VANTs são programados para seguir algum comportamento desejado, como evitar colisões ou mover para mais perto uns dos outros. Na maioria dos casos esse tipo de estratégia é baseado em fenômenos reais observados na natureza e portanto é classificado como bio-inspirado.

2.4 *Flocking*

Flocking é um comportamento observado em bandos de pássaros e cardumes de peixes em que um grande número de indivíduos se move em direção a uma localização alvo comum.

Uma vantagem desse tipo de movimentação é que não existe uma formação específica a ser buscada pelos robôs, eles se movem de uma forma fluida, muito parecida com as contrapartes do reino animal. Isso permite que o enxame se adapte a diferentes tipos de ambientes, sendo uma tarefa útil quando se deseja mover o grupo de robôs por um ambiente desconhecido evitando colisões entre os robôs (e colisões com obstáculos, apesar de essa tarefa não ter sido focada nesse trabalho).

Assim como nos animais, esse tipo de comportamento geralmente é implementado utilizando apenas interações locais (o que pode ser visto como uma vantagem).

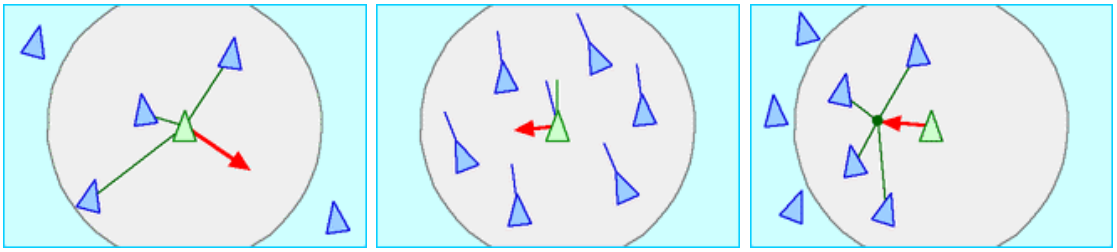
O trabalho mais antigo que modela esse tipo de comportamento foi publicado em 1987 por Craig Reynolds (REYNOLDS, 1987), em que foi apresentado um modelo chamado "boids". Inicialmente desenvolvido para o campo de gráficos computadorizados, esse modelo eventualmente passou a ser utilizado tambérm na área de enxames robóticos.

Reynolds propôs um conjunto de regras comportamentais visando simular o movimento de grupos de animais como bandos de pássaros ou cardumes de peixes. Foi criado um programa de computador em que entidades chamadas boids se movem de acordo com essas regras. Para melhor representar o comportamento de animais reais, Reynolds não criou nenhum controle central, mas ao invés disso programou cada boid para sentir seu próprio ambiente e decidir para onde se mover. Isso resultou em um movimento fluido muito similar ao de bandos reais de pássaros.

As regras de Reynolds são representadas na figura 2.4.1 e podem ser descritas como:

- **Separação:** os boids tentam se mover para longe uns dos outros para evitar colisões;
- **Alinhamento:** os boids tentam se mover na mesma direção que os outros estão se movendo em média;
- **Coesão:** os boids tentam se mover para perto dos outros para formar um bando;

Essas três regras sozinhas são suficientes para fazer os robôs se agruparem e se moverem juntos para direções aleatórias. Porém uma vez que se quer controlar o enxame, foi



(a) Separação (b) Alinhamento (c) coesão

Figura 2.4.1 – Regras de Reynolds (Fonte: Reynolds, 1995)

incorporada uma quarta regra, chamada Migração. Essa regra permite selecionar uma localização chamada ponto de migração para onde os robôs tentam se mover.

2.5 Otimização por Enxame de Partículas

Otimização por Enxame de Partículas (ou PSO, do inglês *Particle Swarm Optimization*) é uma técnica de otimização desenvolvida em 1995 (KENNEDY; EBERHART, 1995) inspirada pelo comportamento de animais sociais como cardumes de peixes ou bandos de pássaros. No PSO, cada solução possível para o problema de otimização é modelada com uma partícula que se move sobre o espaço de busca. As partículas são avaliadas por uma função objetivo e recebem um valor representando sua performance em resolver o problema de otimização. Cada partícula mantém um registro da posição onde obteve seu melhor valor, chamando esse valor de *local best*. Além disso o melhor valor obtido pelo enxame todo é lembrado com *global best*, assim com sua posição correspondente. Em cada iteração as partículas se movem com uma velocidade calculada através da fórmula:

$$v_{t+1} = \omega * v_t + c_1 * r_1 * (l_{\text{best}t} - x_t) + c_2 * r_2 * (g_{\text{best}t} - x_t) \tag{2.1}$$

Onde x_t é a posição da partícula no tempo anterior, ω é o coeficiente inercial, l_{best} e g_{best} são o *local best* da partícula e o *global best* do enxame, c_1 e c_2 são coeficientes de aprendizado e r_1 e r_2 são vetores aleatórios.

Como pode ser visto, a velocidade é composta por três fatores. O primeiro é o componente inercial, que faz a partícula continuar se movendo na direção que ela vinha se movendo previamente. O segundo fator é o componente cognitivo, que faz a partícula se mover na direção de l_{best}, seu melhor valor local. O último é chamado de componente social, e faz com que a partícula se mova em direção a g_{best}, o maior valor obtido por qualquer partícula no enxame. Os vetores aleatórios são aplicados para prevenir a partícula de parar em ótimos locais.

2.6 Trabalhos Relacionados

A maioria dos trabalhos encontrados na literatura se inspira nos comportamentos de animais sociais encontrados na natureza (MOESLINGER; SCHMICKL; CRAILSHEIM, 2011). Como já citado, Craig Reynolds foi o primeiro a propor um conjunto de regras comportamentais para simular o movimento de um bando de pássaros (REYNOLDS, 1987). Hauert et al. (2011) usou as regras de Reynolds para criar um bando de 10 VANTs de asa fixa tanto em simulação como na realidade. Porém esses VANTs voavam em altitudes diferentes, portanto eles não precisavam evitar colisões uns com os outros. Em (QUINTERO; COLLINS; HESPANHA, 2013) 3 VANTs voaram juntos, porém novamente em altitudes diferentes.

Kushleyev et al. (2013) desenvolveu um impressionante bando de 20 quadrotores em miniatura capazes de assumir várias formações diferentes, usando uma estratégia de controle centralizado. Bürkle, Segor e Kollmann (2011) criou um enxame de quadrotores que voavam em ambiente aberto também utilizando processamento central em uma estação em terra. Porém o controle centralizado é algo que não existe em bandos reais de pássaros e por essa razão outros trabalhos tendem a utilizar controle descentralizado, programando cada VANT para tomar suas próprias decisões. Um exemplo notável é o trabalho de Vásárhelyi et al. (2014), em que um enxame de 10 quadrotores voou em um ambiente aberto usando as regras de Reynolds. Uma aplicação das regras de enxame também pode ser vista em De Benedetti et al. (2016), onde um grupo de VANTs simulados é utilizado para monitorar uma área.

3 Modelagem

A modelagem e controle do quadrotor são problemas altamente investigados na literatura, sendo que diversas soluções já foram propostas e testadas. Nesse capítulo a modelagem do quadrotor será discutida de forma breve, conforme descrita nas seguintes fontes: (BEARD; MCLAIN, 2012), (RAZA; GUEAIEB, 2010) e (HABIB et al., 2014). Em seguida, será detalhada a metodologia utilizada para representar as informações dos múltiplos VANTs que fazem parte de um enxame robótico.

3.1 O Quadrotor

O quadrotor é um VANT de asas rotativas composto por quatro rotores fixos nas pontas de uma estrutura em forma de cruz chamada *frame*. É um veículo dinamicamente instável, necessitando de métodos de controle adequados para atingir a estabilidade. Porém essa instabilidade, quando controlada, o torna um veículo altamente ágil. Em comparação com as aeronaves de asa fixa é um veículo muito manobrável, capaz de se mover com 6 graus de liberdade, decolar e pousar na vertical (classificado como VTOL, do inglês *Vertical Take Off and Landing*) e pairar no ar. É também capaz de carregar uma maior carga do que os veículos de asa fixa.

A Figura 3.1.1 representa a estrutura física do quadrotor. Ao girar, cada rotor gera uma força de empuxo vertical ao longo do eixo Z e um torque de reação no sentido contrário ao do giro do rotor. Para balancear os torques criados pelos rotores, os rotores 1 e 3 giram no sentido anti-horário enquanto que os rotores 2 e 4 giram no sentido horário.

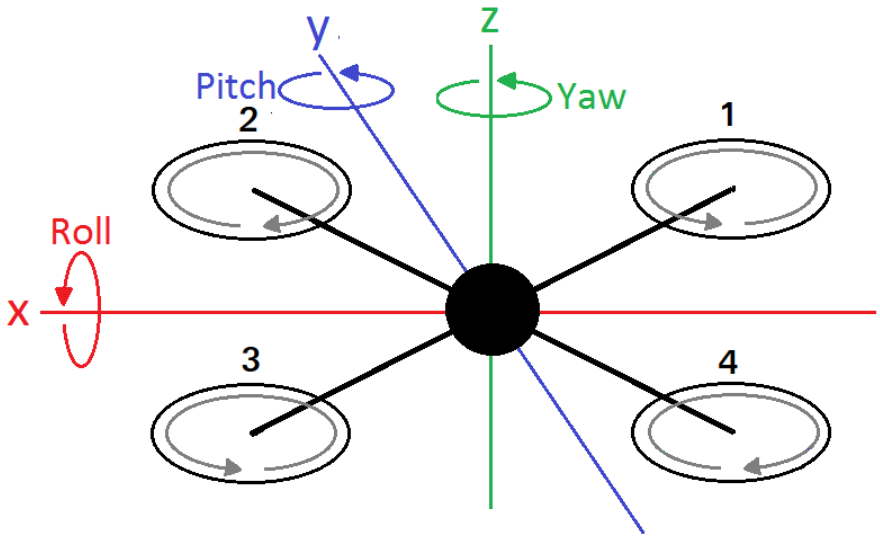


Figura 3.1.1 – Modelo do quadrotor (Fonte: autor)

Controlando-se a velocidade de giro dos rotores é possível controlar o empuxo e o torque gerados por cada um deles e consequentemente mover o quadrotor, através de quatro manobras distintas:

- **Subida e Descida (*Throttle*)**: Aumentando ou diminuindo a velocidade dos quatro rotores ao mesmo tempo é criada uma força na vertical que move o quadrotor verticalmente ao longo do eixo Z.
- **Rolagem (*Roll*)**: Aumentando (ou diminuindo) a velocidade do rotor 4 e diminuindo (ou aumentando) a velocidade do rotor 2 enquanto a velocidade dos outros dois rotores é mantida constante é criado um torque ao redor do eixo X, fazendo com que o quadrotor gire ao redor desse eixo. A força vertical criada pelos rotores é inclinada, passando a ter um componente na direção do eixo Y, o que faz com que o quadrotor se movimente ao longo desse eixo.
- **Arfagem (*Pitch*)**: Aumentando (ou diminuindo) a velocidade do rotor 1 e diminuindo (ou aumentando) a velocidade do rotor 3 enquanto a velocidade dos outros dois rotores é mantida constante é criado um torque ao redor do eixo Y, o que faz com que o quadrotor gire ao redor desse eixo. A força vertical criada pelos rotores é inclinada, passando a ter um componente na direção do eixo X, fazendo com que o quadrotor se mova ao longo desse eixo.
- **Guinada (*Yaw*)**: Aumentando (ou diminuindo) a velocidade dos rotores 1 e 3 e diminuindo (ou aumentando) a velocidade dos rotores 2 e 4 o torque de reação gerado por eles se desequilibra, e o quadrotor passa a ter um torque resultante ao redor do eixo Z. Isso faz com que o quadrotor gire ao redor desse eixo.

As quatro manobras são representadas na Figura 3.1.2.

3.1.1 Sistemas de Coordenadas

Para modelarmos o quadrotor é interessante definir alguns Sistemas de Coordenadas Cartesianas (SCC). Seja um quadrotor se movendo no espaço, conforme representado pela Figura 3.1.3.

O SCC Inercial $S_i = (\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i)$ é fixo ao solo, com sua origem na posição da estação em solo. Por convenção o vetor $\mathbf{x}_i$ aponta para o norte, o vetor $\mathbf{y}_i$ aponta para o leste e o vetor $\mathbf{z}_i$ aponta para baixo, razão pela qual esse SCC é comumente conhecido como NED (*North East Down*).

O SCC do veículo $S_v = (\mathbf{x}_v, \mathbf{y}_v, \mathbf{z}_v)$ tem sua origem fixa no centro de massa do quadrotor, porém seus eixos são paralelos aos do SCC inercial. O SCC do veículo é relacionado ao SCC inercial por uma translação.

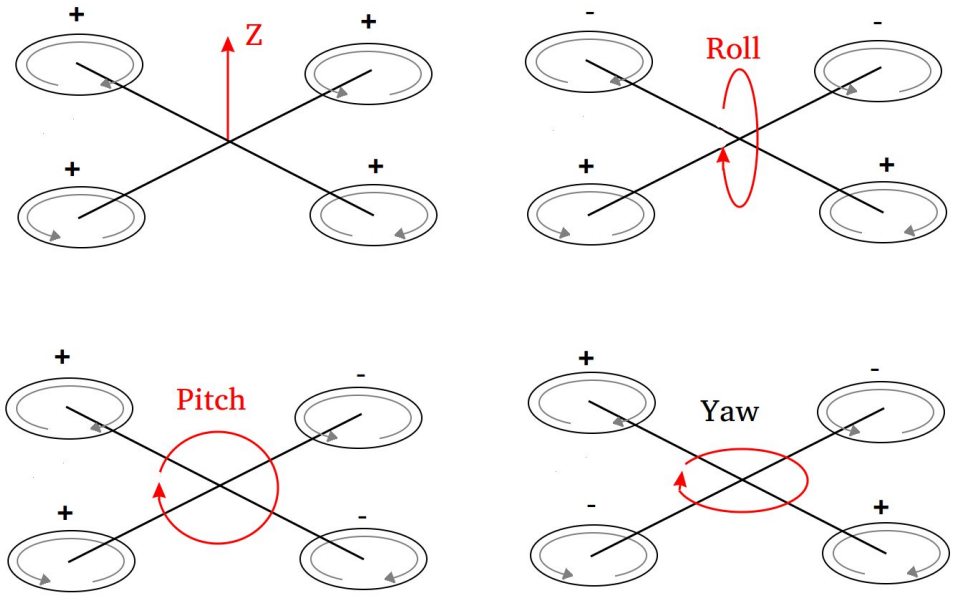


Figura 3.1.2 – Manobras realizadas pelo quadrotor. Símbolos de + simbolizam um aumento na velocidade de rotação do rotor. Símbolos de - representam uma diminuição na velocidade de rotação (Fonte: autor)

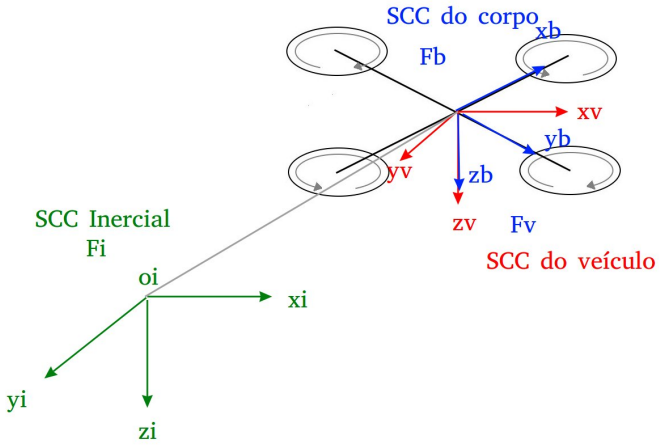


Figura 3.1.3 – Os Sistemas de Coordenadas Inercial, do Veículo e do Corpo (Fonte: autor)

O SCC do corpo $S_b = (\mathbf{x}_b, \mathbf{y}_b, \mathbf{z}_b)$ é fixo no centro de massa do quadrotor, porém seus eixos estão alinhados com a estrutura física do quadrotor. O vetor $\mathbf{x}_i$ está alinhado com o braço do rotor 1, o vetor $\mathbf{y}_i$ está alinhado com o braço do rotor 2 e o vetor $\mathbf{z}_i$ aponta para baixo. O SCC do corpo é relacionado ao SCC do veículo por uma rotação.

O formalismo de Newton-Euler pode ser utilizado para derivar os modelos cinemáticos e dinâmicos que descrevem a movimentação do quadrotor.

3.2 Modelo Cinemático

Seja $\mathbf{p}_S^T = [p_x, p_y, p_z]$ e $\mathbf{r}_S^T = [\phi, \theta, \psi]$ respectivamente os vetores que denotam a posição e orientação do quadrotor expressos no SCC S.

3.2.1 Movimento Rotacional

Normalmente as velocidades angulares do quadrotor ao redor dos eixos do SCC S_b são obtidas diretamente através das medidas dos seus sensores embarcados. Essas velocidades serão representadas por $\omega = \begin{bmatrix} p & q & r \end{bmatrix}^T$. As velocidades angulares no SCC S_i, que são conhecidas como taxas de Euler, são representadas por $\dot{\eta} = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$, e podem ser obtidas a partir das velocidades angulares ω através da seguinte relação:

$$\omega = R_b^i \dot{\eta} \tag{3.1}$$

Onde R_b^i é chamada de Matriz de Rotação e transforma vetores do SCC S_b para o SCC S_i. A Matriz de Rotação é dada por:

$$R_b^i = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{bmatrix} \tag{3.2}$$

3.2.2 Movimento Translacional

A relação entre a velocidade do quadrotor nos três SCs definidos é:

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix}_{S_i} = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix}_{S_v} = R_v^b \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix}_{S_b} \tag{3.3}$$

3.3 Modelo Dinâmico

A movimentação do quadrotor pode ser dividida em dois subsistemas: translacional e rotacional. O subsistema rotacional é totalmente atuado enquanto o translacional é sub atuado.

3.3.1 Equações de Movimento Rotacional

Usando o formalismo de Newton-Euler as equações de movimento rotacional podem ser definidas no eixo fixo ao corpo do quadrotor como:

$$J\dot{\omega} + \omega \times J\omega = M \tag{3.4}$$

Onde:

J Matriz de Inércia do quadrotor

ω Velocidade angular

M É o Momento total agindo sobre o quadrotor

Os primeiros dois termos da equação, $J\dot{\omega}$ e $\omega \times J\omega$ representam a taxa de mudança do momento angular no SCC fixo ao corpo. O motivo de derivar as equações rotacionais no SCC do corpo e não no SCC inercial é ter a matriz de inércia independente do tempo.

3.3.1.1 Matriz de Inércia

Devido à simetria do quadrotor, sua matriz de inércia é diagonal, na forma:

$$J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.5}$$

Onde I_{xx}, I_{yy} e I_{zz} são os momentos de inércia ao longo dos eixos principais do SCC fixo ao corpo.

3.3.1.2 Momentos Agindo no Quadrotor (M)

Para o último termo da equação 3.4 é necessário definir dois efeitos físicos que são as forças e momentos aerodinâmicos produzidos pelos rotores. Como um efeito da rotação, há uma força gerada chamada de força aerodinâmica e um momento gerado chamado de momento aerodinâmico. As equações 3.6 e 3.7 descrevem a força aerodinâmica F_i e o momento aerodinâmico M_i produzidos pelo i-ézimo rotor:

$$F_i = \frac{1}{2}\rho A C_T r^2 \Omega i^2 \tag{3.6}$$

$$M_i = \frac{1}{2}\rho A C_D r^2 \Omega i^2 \tag{3.7}$$

Onde:

ρ Densidade do ar

A Área da hélice

C_T, C_D Coeficientes aerodinâmicos

r Raio da hélice

Ωi Velocidade angular do rotor i

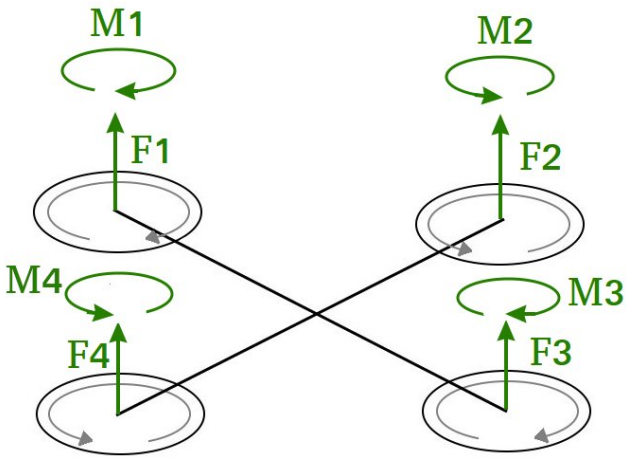


Figura 3.3.1 – Forças e momentos agindo no quadrotor (Fonte: autor)

Claramente as forças e momentos aerodinâmicos dependem da geometria da hélice e da densidade do ar. Como no caso de quadrotores a altitude máxima é limitada, a densidade do ar pode ser considerada constante, e as equações 3.6 e 3.7 podem ser reescritas como:

$$F_i = K_f \Omega i^2 \tag{3.8}$$

$$M_i = K_m \Omega i^2 \tag{3.9}$$

Onde K_f e K_m são as constantes de força aerodinâmica e momento aerodinâmico, respectivamente. Essas constantes podem ser determinadas experimentalmente para cada tipo de hélice.

Identificando as forças e momentos gerados pelas hélices, nós podemos estudar os momentos M que agem no quadrotor. A Figura 3.3.1 mostra as forças e momentos agindo no quadrotor. Cada rotor i produz uma força para cima F_i e um momento M_i na direção oposta à da rotação do rotor i.

Começando com os momentos ao redor do eixo x, F_2 multiplicada pelo comprimento do braço do quadrotor l gera um momento negativo ao redor do eixo x, enquanto que, da mesma maneira, F_4 gera um momento positivo. Assim, o momento total no eixo x pode ser expresso como:

$$\begin{aligned} M_x &= -F_2 l + F_4 l \\ &= -(K_f \Omega_2^2) l + (K_f \Omega_4^2) l \\ &= l K_f (-\Omega_2^2 + \Omega_4^2) \end{aligned} \tag{3.10}$$

Para os momentos ao redor do eixo y, o rotor 1 gera um momento positivo enquanto o rotor 3 gera um momento negativo. O momento total pode ser expresso como:

$$\begin{aligned} M_y &= F_1 l - F_3 l \\ &= (K_f \Omega_1^2) l - (K_f \Omega_3^2) l \\ &= l K_f (\Omega_1^2 - \Omega_3^2) \end{aligned} \tag{3.11}$$

Para o eixo Z, a força produzida pelos rotores não causa momento. Por outro lado, momentos são gerados pela rotação dos rotores conforme a equação 3.9. O momento total no eixo Z pode ser expresso como:

$$\begin{aligned} M_z &= M_1 - M_2 + M_3 - M_4 \\ &= K_m \Omega 1^2 - K_m \Omega 2^2 + K_m \Omega 3^2 - K_m \Omega 4^2 \\ &= K_M (\Omega 1^2 - \Omega 2^2 + \Omega 3^2 - \Omega 4^2) \end{aligned} \tag{3.12}$$

Combinando as três equações na forma de vetor nós temos:

$$M = \begin{bmatrix} l K_f (-\Omega_2^2 + \Omega_4^2) \\ l K_f (\Omega_1^2 - \Omega_3^2) \\ K_M (\Omega 1^2 - \Omega 2^2 + \Omega 3^2 - \Omega 4^2) \end{bmatrix} \tag{3.13}$$

3.3.2 Equações de Movimento Translacional

As equações de movimento translacional são derivadas a partir da segunda lei de Newton e são expressas no SCC inercial através da equação:

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R F_B \tag{3.14}$$

Onde:

$r = \begin{bmatrix} x & y & z \end{bmatrix}^T$	Distância do quadrotor ao SCC inercial
m	Massa do quadrotor
g	Aceleração da gravidade $g = 9,81 m/s^2$
R	Matriz de rotação de S_B para S_I
F_B	Forças não gravitacionais agindo no SCC fixo no corpo

3.3.2.1 Forças não Gravitacionais Agindo no Quadrotor

Quando o quadrotor está numa orientação horizontal a única força não gravitacional agindo nele é o empuxo produzido pelos rotores conforme a equação 3.8. Assim, as forças não gravitacionais agindo no quadrotor F_B podem ser expressas como:

$$F_B = \begin{bmatrix} 0 \\ 0 \\ K_f(\Omega 1^2 + \Omega 2^2 + \Omega 3^2 + \Omega 4^2) \end{bmatrix} \tag{3.15}$$

As duas primeiras linhas do vetor são zeros porque os rotores não produzem nenhuma força nos eixos X e Y. A última linha é apenas a soma das forças produzidas por todos os rotores.

F_B é multiplicada pela matriz de rotação R na equação 3.14 para transformar a força expressa no SCC fixo ao corpo para o SCC inercial, de forma que a equação pode ser aplicada para qualquer orientação do quadrotor.

3.4 Sistema com Múltiplos VANTs

O sistema desenvolvido é formado por um grupo de robôs que voam sobre uma área e uma estação em terra com a qual eles se comunicam através de um link de telemetria. A estação em terra é responsável por coletar dados de voo de cada robô, mas não controlá-los. O algoritmo de controle é distribuído e roda em cada VANT. O objetivo é mover o enxame para um ponto alvo usando as regras de enxame para evitar colisões e otimizar o movimento.

Foi considerado que cada robô é capaz de se localizar no SCC Inercial usando seus sensores embarcados. Para simplificar a estratégia de controle a altitude foi fixada em um valor predeterminado, o que é válido para a maioria das aplicações de vigilância e busca e resgate. Dada a altitude predeterminada, os valores de latitude e longitude podem ser convertidos em coordenadas cartesianas X e Y. A direção de movimento é dada pelo ângulo de guinada, chamado em inglês de *Yaw*. Assim, a pose de cada robô i a qualquer momento é dada pela tupla:

$$\mathrm{Pose_i} = (\ \mathrm{X_i},\ \mathrm{Y_i},\ \mathrm{Yaw_i}\ )$$

Para essa estratégia de controle funcionar cada robô deve determinar sua própria pose e também a pose dos seus vizinhos. O GPS e sensores embarcados são suficientes para encontrar a pose do robô. Porém a localização por GPS tem uma precisão bastante baixa e não é confiável para ser usada para evitar colisões. Porém, as regras de Reynolds requerem apenas que um robô saiba a posição dos vizinhos relativa a si mesmo, não a sua posição global. Isso permite que sejam utilizados sensores de distância, que possuem precisão melhor que a do GPS, para detectar robôs próximos. Essa estratégia é mais

próxima de como bandos reais de pássaros funcionam e também elimina a necessidade de comunicação entre os robôs.

O objetivo da missão é atingir uma determinada localização chamada Ponto de Migração definida por X_{mp} e Y_{mp}. Uma interface que é executada na estação em terra permite a um operador criar um ponto de migração durante a missão. A estação em terra envia essa informação para todos os robôs. De novo, uma vez que a precisão do GPS é pobre, foi definido um raio de tolerância R_T ao redor do ponto de migração. O objetivo é considerado atingido pelo VANT quando ele entra nessa área.

4 Implementação

Este capítulo detalha a implementação do algoritmo para controle de enxames. O capítulo começa com uma descrição dos robôs utilizados em laboratório. Em seguida, as duas principais aplicações que foram criadas para testar o algoritmo proposto são apresentadas: a busca por vazamento de gás e o controle de formação. O texto então se aprofunda no software que foi implementado. O funcionamento do ROS é explicado em detalhes e são apresentados os pseudocódigos de cada regra de comportamento que foi criada. Por fim considerações sobre a complexidade computacionais são traçadas.

4.1 Hardware

Os robôs que se deseja controlar com o algoritmo proposto são pequenos quadrotores com 250mm de diâmetro, cada um usando uma Pixhawk como placa controladora. A Pixhawk é um dispositivo de código aberto (PIXHAWK, 2013) equipada com muitos sensores como giroscópio, acelerômetros, magnetômetro e barômetro e também pode ser conectada a um módulo de GPS externo. Ela executa um software poderoso que implementa as rotinas básicas de controle do quadrotor, juntamente com muitas outras funções úteis. Na aplicação proposta, a principal função da Pixhawk é providenciar estabilização de baixo nível e controle do VANT, além da estimação da sua posição. A Figura 4.1.1 mostra uma foto de um dos VANTs utilizados no laboratório.

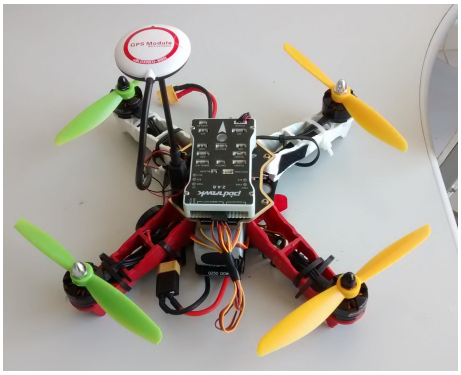


Figura 4.1.1 – VANT utilizado no laboratório (Fonte: autor)

Originalmente a Pixhawk foi feita para ser controlada por um operador humano através de um controle via rádio (RC) ou receber comandos de uma estação em terra. Porém ela também é capaz de se comunicar com outros dispositivos através de um protocolo chamado MAVLink. Esse protocolo foi utilizado para enviar comandos de um computador Linux embarcado (Raspberry Pi) que também é carregado pelo VANT. Dessa forma pode-se programar o VANT para voar de forma autônoma, sendo ainda capaz de retomar

controle manual a qualquer momento. A Figura 4.1.2 mostra uma vista esquemática dos componentes do VANT.

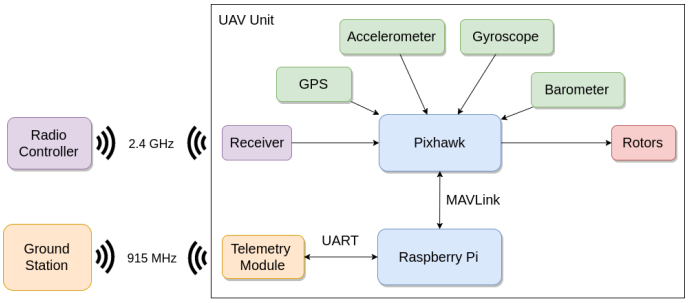


Figura 4.1.2 – Componentes do VANT (Fonte: autor)

4.2 Missão de Busca por Vazamento de Gás

A fim de explorar uma possível aplicação do sistema desenvolvido, foi proposta uma missão da seguinte forma: após um desastre (como por exemplo um terremoto) um gás tóxico começa a vazar em uma área aberta e um grupo de quadrotores será liberado para navegar pela área em busca da fonte de gás. Em adição aos sensores previamente mencionados, cada quadrotor carrega um sensor de gás, que retorna um valor representando a concentração de gás na sua posição atual.

O algoritmo da Otimização por Enxame de Partículas é naturalmente adequado para ser usado em aplicações onde um enxame de robôs se move sobre uma área buscando por alguma posição alvo. Essa característica inspirou a adição de alguns conceitos do PSO juntamente com as regras de enxame à estratégia de controle. Assim foi implementada uma quarta regra chamada Seguir Gás, que deve fazer os VANTs seguirem o rastro do gás até a fonte do vazamento.

Assumindo que cada VANT é equipado com um sensor capaz de detectar o gás e medir sua concentração no ar, e assumindo que a concentração se torna maior quanto mais próximo o VANT está da fonte, considerando os valores maiores como lbest, pode-se usar a Equação 2.1 para calcular um vetor que moverá o VANT em direção aos pontos com maior concentração de gás, eventualmente atingindo a fonte do vazamento.Porém, uma vez que as regras de enxame já fazem os VANTs se moverem juntos como um bando, pode-se ignorar o componente social da Equação 2.1, pois a interação social virá das próprias regras de enxame. Além disso é possível ignorar o componente inercial uma vez que os VANTs já possuem uma inercial física. Assim, foi definida a regra de Seguir Gás como:

$$v = c_1 * r_1 * (lbest_t - x_t) \tag{4.1}$$

4.3 Controle de Formação

A fim de explorar outras técnicas de controle de enxames, também foi proposta uma estratégia para controlar a formação dos VANTs. Nesse cenário os VANTs receberão dados que representam uma formação e deverão assumir as posições adequadas a fim de manter essa formação.

Cada VANT recebe um ID único, que é um número inteiro positivo começando em 0 e aumentando em incrementos de 1. A formação é definida como um vetor de posições relativas a um sistema de coordenadas da formação, S_F. A posição designada para cada VANT é aquela cujo índice no vetor da formação é igual ao ID do VANT. Nos experimentos foram definidas três formações: uma linha horizontal, uma coluna vertical e um triângulo.

Cada VANT computa sua posição desejada relativa ao sistema de coordenadas da formação. Foram definidos dois métodos para fazer esse cálculo:

- **Método do líder-seguidor:** Nesse método um VANT é considerado o líder do grupo (aquele com ID 0). O líder ignora sua posição na formação e ao invés disso pode se mover livremente, sendo controlado por um operador humano ou seguindo uma sequência de pontos. Os outros VANTs calculam sua posição desejada relativa ao líder, considerando que a posição dele é a origem do sistema de coordenadas da formação.

- **Método de Waypoints:** Nesse método, todos os VANTs recebem um waypoint para seguir e tratam esse waypoint como a origem do sistema de coordenadas da formação. Não há líder nesse método. Assim, essa abordagem é mais tolerante a falhas dos VANTs que a primeira, porém é mais difícil para um operador humano controlar o grupo.

A posição de cada VANT i no sistema de coordenadas S_F é representada por p_i^F. No primeiro método de cálculo da posição, a origem do sistema de coordenadas S_F é a posição do líder, e então a posição desejada de cada VANT i no sistema de coordenadas global é dada por:

$$p_{iD}^G = p_0^G + p_i^F \tag{4.2}$$

onde p_{iD}^G é a posição desejada do VANT i no sistema de coordenadas S_G e p_0^G é a posição atual do líder no sistema de coordenadas S_G. No segundo método, o sistema de coordenadas S_F é centralizado no próximo waypoint, portanto a posição desejada do VANT i no sistema de coordenadas global é dada por:

$$p_{iD}^G = p_w^G + p_i^F \tag{4.3}$$

onde p_w^G é a posição do próximo waypoint.

4.4 Robot Operating System

O algoritmo foi implementado em C++ na plataforma ROS. ROS (*Robot Operating System*) é uma plataforma de código aberto criado para ajudar pesquisadores no desenvolvimento de aplicações robóticas (ROS, 2007). O ROS fornece muitas ferramentas e estruturas de dados padrões que foram muito úteis no trabalho desenvolvido.

Uma aplicação em ROS é uma rede de programas independentes chamados nós, que se comunicam uns com os outros. Essa rede é gerenciada por outro programa chamado ROS Master. Essa comunicação acontece na forma de trocas de mensagens; nós que geram informações publicam essa informação em tópicos na forma de mensagens, enquanto nós que precisam dessa informação subscrevem aos tópicos correspondentes e recebem as mensagens publicadas. Os tipos de mensagens representam estruturas de dados comuns utilizadas em robótica, como leituras de sensores ou comandos de velocidade.

4.4.1 Aplicação em ROS - Estrutura Geral

Dois nós são executados em cada VANT. O primeiro é chamado *mavros*, que é um nó padrão que conecta o sistema à Pixhawk através de uma conexão serial. O nó *mavros* é capaz de converter os dados contidos em mensagens do ROS para mensagens MAVLink e vice-versa. Isso significa que o *mavros* é o responsável por fazer a ponte entre nosso sistema ROS e a Pixhawk.

O outro nó é o programa principal de controle, chamado *swarm_controller_node*. Esse nó subscreve a alguns dos tópicos onde o *mavros* publica dados recebidos da Pixhawk. Usando informações do sensor de GPS e do estimador de estados interno da Pixhawk o *swarm_controller_node* determina a posição do VANT no sistema de coordenadas global e publica essa informação em um tópico chamado *uav_positions*. Esse nó também subscreve a esse mesmo tópico, para receber os dados publicados pelos outros VANTs. Dessa forma, todos os VANTs são capazes de saber a posição global de todos os membros do enxame. Essa informação é armazenada em um array interno.

Usando a informação obtida o *swarm_controller_node* utiliza as regras comportamentais para decidir para qual direção o VANT deve se mover. O nó então publica essa informação na forma de uma mensagem do tipo *geometry_msgs/TwistStamped* no tópico *mavros/setpoint_velocity/cmd_vel*. O nó *mavros* recebe essa informação, cria a mensagem MAVLink correspondente e envia para a Pixhawk, que seguirá esse comando de acordo.

Uma vez que cada VANT executa instâncias dos mesmos nós, é necessário executá-los sob diferentes *namespaces* para evitar conflitos. A cada VANT é designado um *namespace* na forma */uavn*, onde n é igual ao ID daquele VANT. Um diagrama simplificado do sistema resultante é mostrado na Figura 4.4.2.

Nas aplicações de busca por vazamento de gás e controle de formação existem algumas características exclusivas que serão analisadas com mais detalhes nas próximas seções.

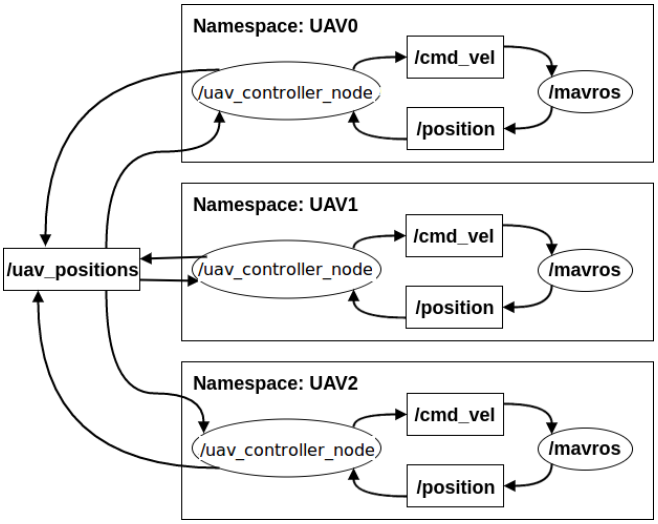


Figura 4.4.1 – Diagrama simplificado da aplicação em ROS. (Fonte: autor)

4.4.2 Busca por Vazamento de Gás

Para o nó de controle ser capaz de detectar a fonte de vazamento do gás ele precisa receber os dados de concentração de gás a partir de algum sensor. Como os testes foram realizados em simulação, essa medida foi gerada através de uma função que retorna um valor de concentração para cada posição onde o VANT se encontra.

4.4.3 Controle de Formação

Nesse caso o nó de controle precisa receber informações sobre a formação desejada que os VANTs façam. Assim, o *swarm_controller_node* subscreve a um tópico chamado /formation, onde a estação em terra publica mensagens do tipo *geometry_msgs/PoseArray*. Essas mensagens contém um *array* de poses representando a estrutura virtual que descreve a formação. Cada vez que uma nova mensagem chega, o nó de controle atualiza um *array* interno que armazena a formação desejada, de forma que a formação possa ser mudada durante o voo.

4.5 Implementação do Algoritmo de Controle

Nessa seção é discutida em detalhes a implementação de cada uma das regras que controlam os VANTs de forma que eles executem os comportamentos desejados. Também será explicado como essas regras são integradas no programa principal.

A estratégia de controle proposta é descentralizada, portanto cada VANT executa seu próprio nó de controle. A cada iteração esse nó deve: *i.* Obter a pose do próprio VANT a partir do sensor de GPS e do estimador de estados da Pixhawk; *ii.* Obter a pose dos vizinhos através de comunicação; *iii.* Usar as regras de enxame para determinar

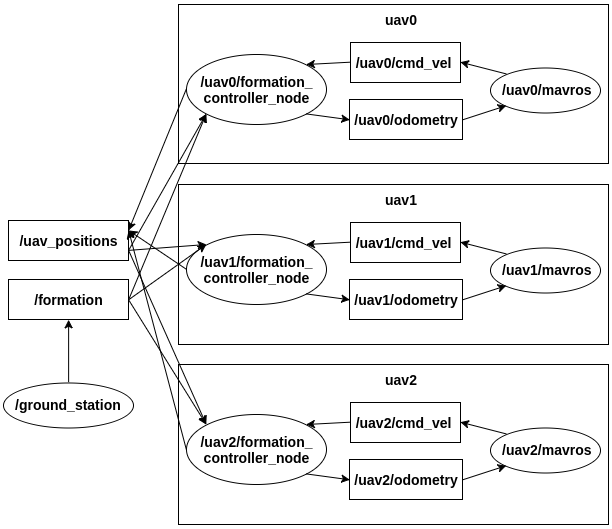


Figura 4.4.2 – Diagrama da aplicação de controle de formação. (Fonte: autor)

qual a direção para a qual o VANT deve se mover de forma a seguir os comportamentos desejados.

O looping principal do algoritmo de controle é dado pelo Algoritmo 1.

Algoritmo 1: Loop principal do algoritmo de enxame

```
1  início
2      getPose()
3      getNeighbors()
4      v1 ← separation()
5      v2 ← alignment()
6      v3 ← cohesion()
7      v4 ← migration()
8      vres ← r1 * v1 + r2 * v2 + r3 * v3 + r4 * v4
9      move(vres)
10 fim
```

Primeiro o programa chama a função *getPose()*, que calcula a pose do VANT e inicializa duas variáveis, *this.position* e *this.heading*, representando as coordenadas XY e direção de movimento do veículo. Então, o programa chama outra função *getNeighbors()* que lê as informações recebidas através da comunicação com os outros robôs para determinar as posições e direções de movimento desses outros robôs. Essa informação é armazenada em um array chamado *neighbors*.

A partir daí o programa começa a chamar as funções onde as regras de enxame estão de fato implementadas. Cada função retorna um *array* representando o vetor velocidade apontando para a direção para a qual aquela regra decide que o VANT deve se mover. Na linha 8 os quatro vetores são combinados através de uma média ponderada para gerar a direção final para a qual o robô deve se mover. O peso aplicado a cada regra determina o quanto aquela regra influencia no resultado final. Os valores dos pesos podem ser

trocados para obter diferentes resultados. Regras podem ser até mesmo removidas do calculo modificando o valor do seu peso para zero.

Na linha 9 a direção resultante calculada é passada para outra função para mover o VANT. Essa função usa as funções do pacote *mavros* para enviar mensagens MAVLink para o piloto automático, movendo o VANT para a direção correta.

Regra de Separação: essa regra tenta mover o robô para longe dos outros de forma a evitar colisões. Isso é feito criando um vetor para cada um dos vizinhos apontando para a direção oposta daquele vizinho. Esses vetores são então combinados em um vetor resultante e retornados para o programa principal.

Algoritmo 2: Regra de Separação

```
1  início
2      Vector v ← 0
3      para todos os vizinhos n faça
4          vn ← this.position − n.position
5          vn.normalize()
6          vn ← vn ∗ distance(this, n)
7          v ← v − n
8      fim
9      retorna v
10 fim
```

Também é desejado que o robô se mova mais rápido quanto mais perto ele estiver do vizinho. Isso é obtido ajustando a magnitude dos vetores. Primeiro o vetor é normalizado, de forma que sua magnitude passe a ser 1. A magnitude é então dividida pela distância entre os dois robôs.

Regra de Alinhamento: Essa regra representa um esforço em mover o robô para a mesma direção média que seus vizinhos estão se movendo. O robô executa uma iteração por cada um dos vizinhos e calcula a direção média para a qual eles estão se movendo. Esse vetor é então retornado para a função principal.

Algoritmo 3: Regra do Alinhamento

```
1  início
2      Vector v ← 0
3      para todos os vizinhos n faça
4          v ← v + n.velocity
5      fim
6      v ← v/neighbors.length
7      retorna v
8  fim
```

Regra de Coesão: Essa regra tenta mover o robô para o centro de massa dos outros robôs, de modo a formar um enxame. A implementação é similar à da regra do

Alinhamento mas, ao invés de calcular a direção média, é calculada a posição média dos vizinhos.

Algoritmo 4: Regra da Coesão

```
1 início
2     Vector v ← 0
3     para todos os vizinhos n faça
4         v ← v + n.position
5     fim
6     v ← v/neighbors.length
7     retorna v
8 fim
```

Regra de Migração: Essa é a regra usada para mover o enxame. Uma vez que o ponto de migração tenha sido definido pelo operador, os robôs tentam se mover para aquele ponto. É uma regra simples que apenas retorna um vetor apontando para o ponto de migração.

Algoritmo 5: Regra da Migração

```
1 início
2     Vector v ← 0
3     se migrationPoint é definido então
4         v ← migrationPoint − this.position
5     fim
6     retorna v
7 fim
```

4.5.1 Busca por Vazamento de Gás

Nessa aplicação os VANTs não devem se mover para um ponto de migração definido por um operador, mas sim se mover de forma autônoma em direção à fonte de um vazamento de gás. Para isso a regra de Migração foi substituída por outra, a regra Seguir Gás.

Regra Seguir Gás: Como explicado na seção 2.5 essa regra tenta fazer o VANT se mover em direção à fonte de um vazamento de gás, e é baseada no algoritmo PSO. A

implementação é mostrada no Algoritmo 6.

Algoritmo 6: Regra Seguir Gás

```
1  início
2      concentration ← getConcentration()
3      se concentração é maior que lbest então
4          lbest ← concentration
5          bestPositon ← this.position
6      fim
7      Vector v ← 0
8      Vector r ← rand()
9      v ← c * r * (bestPosition − this.position)
10     retorna v
11 fim
```

Primeiro a função *getConcentration()* é chamada e retorna a concentração atual de gás medida pelo sensor. Esse valor é comparado com o valor prévio de lbest e se a nova concentração for maior, lbest e bestPosition são atualizados. O restante da função implementa a Equação 4.1. Essa equação retorna um vetor apontando para a melhor localização portanto o VANT vai sempre ter uma tendência de se mover para locais onde a concentração de gás é maior. Para evitar máximos locais um fator aleatório também é aplicado. Isso fará com que o VANT se mova para fora de pontos de máximo local.

4.5.2 Controle de Formação

Nessa aplicação é desejado não apenas que os VANTs se movam como um enxame, mas que eles assumam e mantenham uma determinada formação. O programa principal é um pouco diferente, e também as regras comportamentais.

Em cada iteração do algoritmo de controle, ele segue os seguintes passos:

1. Obter a pose do próprio VANT a partir do estimador de estados interno da Pixhawk;
2. Determinar a posição dos vizinhos baseado nas mensagens de posição recebidas dos outros VANTs;
3. Obter o array de formação a partir do tópico */formation*;
4. Usar as regras comportamentais de Coesão e Separação para determinar a direção para a qual o VANT deve se mover.

O programa principal é dado pelo Algoritmo 7.

Algoritmo 7: Loop principal do algoritmo de controle de formação

```
1 início
2     getPose()
3     getNeighbors()
4     getFormation()
5     v1 ← separation()
6     v2 ← cohesion()
7     vres ← r1 * v1 + r2 * v2
8     move(vres)
9 fim
```

Regra de Formação: Essa regra tenta mover o robô para sua posição desejada de forma a manter a formação. A implementação é mostrada no algoritmo 8. Primeiro foi testado qual método está sendo usado para determinar a posição desejada. Então é checado se o VANT é o líder testando se o seu ID é 0. Se o método que está sendo utilizado é o líder-seguidor, o líder segue o *waypoint* e os seguidores determinam suas posições desejadas de acordo com a Equação 4.2. Se o método de *waypoints* está sendo usado, todos os VANTs utilizam a Equação 4.3 para determinar suas posições desejadas. A função então gera um vetor apontando para a posição calculada.

Algoritmo 8: Regra da Formação

```
1  início
2      Vector v ← 0
3      Vector DesiredPosition ← 0
4      se líder-seguidor então
5          se ID == 0 então
6              DesiredPosition ← waypoint
7          senão
8              DesiredPosition ← leader.position + Formation[ID]
9          fim
10     senão
11         DesiredPosition ← waypoint + Formation[ID]
12     fim
13     v ← DesiredPosition − this.position
14     retorna v
15 fim
```

4.6 Preocupações Acerca da Comunicação e da Complexidade Computacional

Uma importante preocupação ao se projetar uma aplicação de controle multi-robô é a elevação da complexidade computacional com o aumento no número de robôs. Em especial, em uma abordagem baseada em comportamento como a que foi utilizada, cada indivíduo tem que iterar por todos os outros indivíduos para computar a influência de suas regras comportamentais na sua movimentação. Se um controle centralizado é usado, isso representa uma complexidade computacional de $O(n^2)$, onde n é o número de robôs. Porém em uma abordagem descentralizada, cada robô executa seu próprio algoritmo de controle com uma complexidade computacional de O(n). Isso significa que uma abordagem descentralizada pode fazer o sistema mais escalável, com a adição de mais robôs trazendo um menor peso ao processo.

Outra preocupação é a influência de atraso nas comunicações. Na aplicação desenvolvida os robôs precisam se comunicar uns com os outros para serem capazes de localizar seus vizinhos. Uma vez que os testes foram realizados em simulação, o atraso na comunicação não era um problema. Porém, em implementações reais o sistema seria negativamente influenciado por esse atraso. Na seção 5 é discutido como esse problema poderia ser atacado em trabalhos futuros.

5 Experimentos e Resultados

Aqui são relatados os experimentos realizados e os resultados obtidos. O capítulo inicia com uma explicação sobre o simulador Gazebo. Então, são descritos os cenários de simulação propostos para testar o funcionamento do algoritmo. Em cada um deles, os resultados obtidos são apresentados e discutidos.

Para testar o desempenho do algoritmo foi decidido realizar testes utilizando o simulador Gazebo (GAZEBO, 2014). O Gazebo é um software de código aberto capaz de simular ambientes tridimensionais e robôs que se movem nesse ambiente. Além disso ele é capaz de simular diversos tipos de sensores e gerar os valores adequados de medidas desses sensores. Todos os elementos da simulação podem ser editados através de arquivos XML que definem as características do mundo de simulação. Além disso o Gazebo permite o desenvolvimento de *plug-ins*, que interagem com a simulação e permitem integrá-la a outros softwares. Dessa forma, foi possível utilizar software fornecido pela equipe que desenvolve o projeto Ardupilot, que simula quadrotores executando o software da Pixhawk e integrando-o com o ROS. Dessa forma foi possível testar a aplicação ROS no simulador como se estivessem sendo utilizados VANTs reais.

A Figura 5.0.1 mostra um VANT simulado, da forma como ele é visível no simulador.

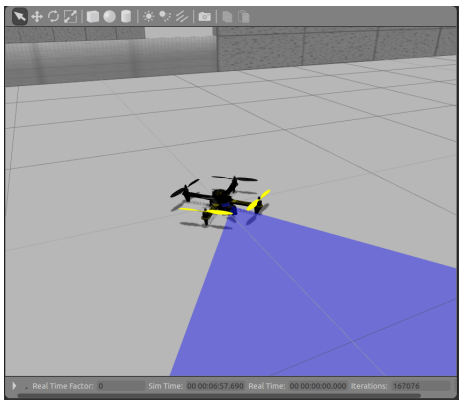


Figura 5.0.1 – VANT simulado com um sensor de distância apontado para frente. (Fonte: autor)

O software RViz também foi utilizado para gerar visualizações dos robôs. A Figura 5.0.2 mostra uma captura de tela desse software.

5.1 Voo em Enxame

Esse teste visou avaliar se as regras de Reynolds utilizadas eram suficientes para mover o enxame de VANTs como um grupo coeso, sem causar colisões entre os membros. Um enxame composto por três VANTs foi definido no simulador Gazebo, com a missão de voar

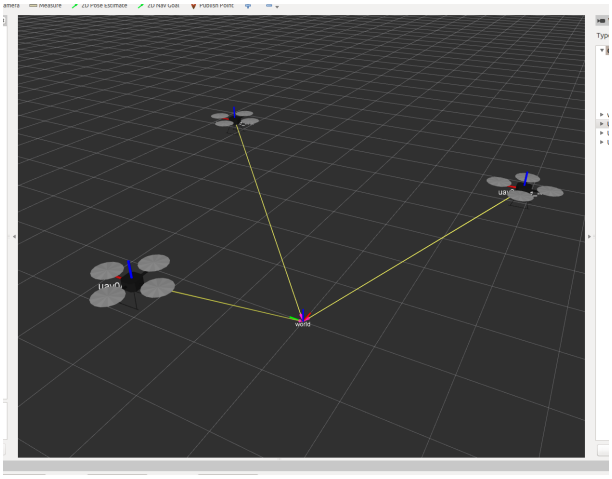


Figura 5.0.2 – Visualização dos VANTs simulados no RViz (Fonte: autor)

através de uma área de 80 por 80 metros, seguindo pontos objetivos que eram enviados pela estação em terra. O algoritmo de controle levava em consideração as três regras de Reynolds e também a regra de migração, que é responsável por mover os robôs em direção aos objetivos. A figura 5.1.1 mostra a trajetória traçada pelos VANTs conforme se moviam em direção a três objetivos.

Foi observado que rapidamente os VANTs se agrupavam, assumindo posições como os vértices de um triângulo, justamente o comportamento esperado da regra da coesão. Devido à inércia, eles acabavam se aproximando bastante uns dos outros, o que fazia com que a influência da regra da separação passasse a dominar, fazendo com que os robôs se afastassem uns dos outros. Esse movimento de aproximação e afastamento se repetia algumas vezes até que os VANTs se estabilizavam a uma distância fixa uns dos outros. Com isso foi possível observar a influência das regras de coesão e separação no movimento do enxame.

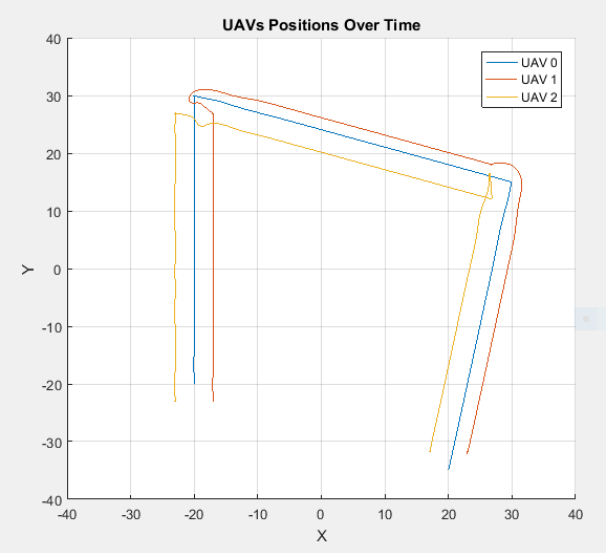


Figura 5.1.1 – Trajetória dos VANTs se movendo em enxame (Fonte: autor)

Ao mesmo tempo que esses comportamentos ocorriam, o enxame começava a se mover em direção ao próximo objetivo. Os robôs paravam de se mover ao atingirem o objetivo e permaneciam estáveis ao redor dele. Uma vez que esse estado era detectado, um novo objetivo era enviado pela estação em terra e o enxame começava a se mover. Dessa vez os movimentos de aproximação e afastamento não eram observados, pois os VANTs já começavam o percurso estando nas suas posições estáveis dentro do enxame.

Durante todo o percurso nenhuma colisão foi observada.

5.2 Busca por Vazamento de Gás

Nesse teste foi definido um cenário em que três VANTs deveriam procurar pela fonte de um vazamento de gás em uma área de 100m por 100m. As posições iniciais dos VANTs e a fonte de gás foram escolhidos de uma forma que os robôs começam de um lado da área enquanto que a fonte fica do outro lado, nas coordenadas X e Y (20, 35). Os VANTs são comandados a decolar e voar a uma altitude fixa de 2 metros. Assim que eles estabilizam nessa altitude, é iniciado o nó de controle e eles começam a se mover. A figura 5.2.1 mostra as trajetórias dos VANTs enquanto buscam pela fonte, que é marcada por um X preto.

Foi observado que a princípio a regra mais prevalente é a regra da Coesão, e os VANTs começam a se mover para perto uns dois outros. Assim que eles atingem uma posição próxima, a regra da Separação começa a agir e eles estabilizam em uma distância pequena uns dos outros. Então a regra do Alinhamento e a regra de Seguir Gás começam a ter uma maior influência no movimento e os VANTs começam a se mover juntos em direção à fonte do gás. Assim que eles chegam mais perto eles começam a oscilar devido aos vetores

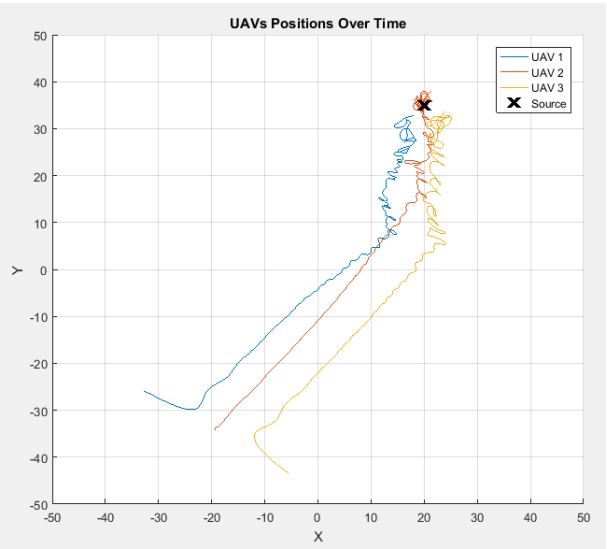


Figura 5.2.1 – Trajetórias desenvolvidas pelos VANTs até estabilizarem ao redor da fonte de gás (Fonte: autor)

aleatórios que são aplicados na regra de Seguir Gás. Após um tempo eles convergem ao redor da fonte, apresentando apenas pequenos movimentos aleatórios.

5.3 Controle de Formação

Na próxima sequência de testes foi utilizada a regra de controle de formação. Foi definido um cenário em que três VANTs, com IDs 0, 1 e 2, têm que se mover juntos mantendo uma formação. Os UAVs decolam e estabilizam a uma altitude de 2 metros. Assim que eles estabilizam é iniciado o nó de controle e eles começam a se mover. Para avaliar a performance do algoritmo foram executados dois testes principais.

5.3.1 Movendo em Formação

O primeiro teste objetiva avaliar a capacidade dos VANTs de manterem uma formação enquanto se movem juntos. Os robôs são comandados a assumirem uma formação e então se moverem para dois *waypoints*. Esse teste foi repetido esse teste para cada método de cálculo de posição e para três formações diferentes: uma linha horizontal, uma coluna vertical e um triângulo. Em todos os experimentos os VANTs foram capazes de manter a formação e nenhuma colisão ocorreu.

Uma característica interessante do sistema desenvolvido é a possibilidade de controlar a formação em três dimensões. Na Figura 5.3.1 os três VANTs são mantidos numa formação vertical, assumindo diferentes posições no eixo Z.

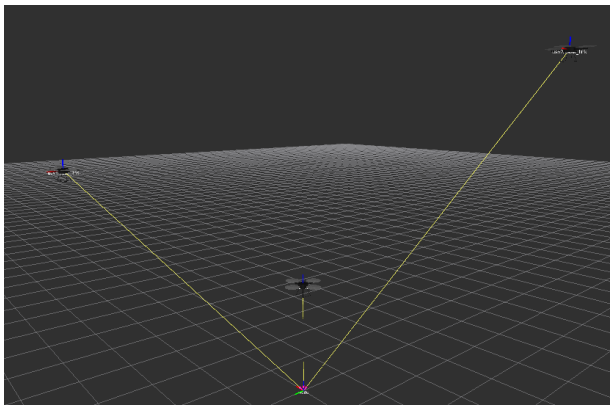


Figura 5.3.1 – VANTs executando uma formação tridimensional

5.3.2 Trocando Formações

Nesse teste objetivou-se observar a capacidade dos VANTs de trocar entre formações sem colidir uns com os outros. Os VANTs começam na formação em linha, então trocam para outras formações seguindo a sequência: coluna, triângulo, linha, triângulo, coluna e

então linha. A Tabela 5.3.1 mostra o tempo gasto em cada troca de formação. Graças ao comportamento de Separação, nenhuma colisão foi observada.

Transição	**Tempo [s]**
linha → coluna	7.16
linha → triângulo	3.66
coluna → triângulo	4.90
coluna → linha	4.70
triângulo → linha	3.53
triângulo → coluna	7.17

Tabela 5.3.1 – Tempo gasto em cada mudança de formação

É interessante notar que os tempos mais longos aparecem nas transições para a formação em coluna. Isso é causado pelo fato de que essa formação é vertical, portanto os VANTs precisam lutar contra a gravidade para ganhar altitude e assumir suas posições desejadas.

6 Conclusão

Esse trabalho apresentou o desenvolvimento de um algoritmo para o controle de enxames de VANTs do tipo quadrotor utilizando a Pixhawk como placa controladora. Usando as regras de *flocking* de Craig Reynolds, juntamente com uma nova regra chamada Migração, o algoritmo é capaz de mover um grupo pequeno de VANTs para um objetivo comum enquanto evitam colisões uns com os outros. Algumas variações do algoritmo também foram propostas, permitindo obter o controle de formação dos VANTs e também fazer os VANTs se moverem de forma autônoma em direção a uma fonte de vazamento de gás. Em especial, a aplicação para busca de vazamento de gás mostrou as regras de *flocking* e a técnica de Otimização por Enxame de Partículas podem ser integradas de forma bastante natural e com bons resultados. O algoritmo foi implementado em C++ e é baseado na plataforma ROS.

Para testar e avaliar o algoritmo proposto, um ambiente de simulação baseado no simulador Gazebo foi preparado. Com esse ambiente objetivou-se recriar as condições ambientais e de *hardware* encontradas no mundo real da forma mais fiel possível. Por isso o simulador Gazebo foi utilizado em conjunto com o *software* ArduPilot, o mesmo que é executado na placa controladora Pixhawk.

Essa implementação, porém ainda é bem simples e muitas melhorias ainda podem ser feitas no futuro. Em primeiro lugar nossa implementação requer que os VANTs voem em uma altitude fixa, limitando o enxame a um movimento bidimensional. Mais trabalhos podem ser feitos para eliminar essa limitação, obtendo um enxame totalmente tridimensional. Além disso, o algoritmo proposto algoritmo depende da comunicação entre os VANTs para que eles possam localizar uns aos outros. Essa abordagem traz diversas desvantagens que poderiam ser eliminadas caso a localização dos vizinhos fosse realizada através de sensores de distância, o que também permitiria que os VANTs detectassem obstáculos. Asssim, uma nova regra de evasão de obstáculos pode ser desenvolvida para que os VANTs evitem colisões com obstáculos fixos como paredes.

Bibliografia

AMBROZIAK, L.; GOSIEWSKI, Z. Two stage switching control for autonomous formation flight of unmanned aerial vehicles. *Aerospace Science and Technology*, Elsevier Masson SAS, v. 46, p. 221–226, 2015. ISSN 12709638. Disponível em: <http://dx.doi.org/10.1016/j.ast.2015.07.015>.

ASKARI, A.; MORTAZAVI, M.; TALEBI, H. A. UAV Formation Control via the Virtual Structure Approach. v. 28, n. 1, p. 1–9, 2015.

BAKER, C. A. B. et al. Planning search and rescue missions for UAV teams. *Frontiers in Artificial Intelligence and Applications*, v. 285, p. 1777–1782, 2016. ISSN 09226389.

BALCH, T.; ARKIN, R. C. Behavior-Based Formation Control for Multirobot Teams. *Ieee Transactions on Robotics and Automation*, v. 14, n. 6, p. 926–939, 1998.

BEARD, R. W.; MCLAIN, T. W. *Small unmanned aircraft: Theory and practice*. [S.l.]: Princeton university press, 2012.

BONABEAU, E. et al. *Swarm intelligence: from natural to artificial systems*. [S.l.]: Oxford university press, 1999.

BRAMBILLA, M. et al. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, v. 7, n. 1, p. 1–41, 2013. ISSN 19353812.

BÜRKLE, A.; SEGOR, F.; KOLLMANN, M. Towards autonomous micro UAV swarms. *Journal of Intelligent and Robotic Systems: Theory and Applications*, v. 61, n. 1-4, p. 339–353, 2011. ISSN 09210296.

CHOI, S. S.; KIM, E. K. Building crack inspection using small UAV. *International Conference on Advanced Communication Technology, ICACT*, v. 2015-Augus, p. 235–238, 2015. ISSN 17389445.

De Benedetti, M. D. et al. UAV-based Aerial Monitoring: A Performance Evaluation of a Self-Organising Flocking Algorithm. *Proceedings - 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2015*, n. ii, p. 248–255, 2016.

EGERSTEDT, M.; HU, X.; STOTSKY, A. Control of mobile platforms using a virtual vehicle approach. *IEEE Transactions on Automatic Control*, v. 46, n. 11, p. 1777–1782, 2001. ISSN 00189286.

GAZEBO. *Gazebo*. 2014. Disponível em: <http://gazebosim.org/>. Acesso em: 26 out 2017.

GENG, L. et al. UAV surveillance mission planning with gimbaled sensors. *IEEE International Conference on Control and Automation, ICCA*, p. 320–325, 2014. ISSN 19483457.

GU, Y. G. Y. et al. Design and Flight Testing Evaluation of Formation Control Laws. *IEEE Transactions on Control Systems Technology*, v. 14, n. 6, p. 1105–1112, 2006. ISSN 1063-6536. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1709935>.

HABIB, M. K. et al. Dynamic modeling and control of a quadrotor using linear and nonlinear approaches. 2014.

HAUERT, S. et al. Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate. *IEEE International Conference on Intelligent Robots and Systems*, p. 5015–5020, 2011. ISSN 2153-0858.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference on*, v. 4, p. 1942–1948 vol.4, 1995. ISSN 19353812.

KUSHLEYEV, A. et al. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, v. 35, n. 4, p. 287–300, 2013. ISSN 09295593.

LI, T. et al. Mission Planning for Multiple UAVs Based on Ant Colony Optimization and Improved Dubins Path *. n. 1, p. 954–959, 2016.

MERINO, L. et al. Automatic Forest Fire Monitoring and Measurement using Unmanned Aerial Vehicles. *VI International Conference on Forest Fire Research*, p. 15, 2010.

MOESLINGER, C.; SCHMICKL, T.; CRAILSHEIM, K. A minimalist flocking algorithm for swarm robots. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 5778 LNAI, n. PART 2, p. 375–382, 2011. ISSN 03029743.

OLLERO, A.; DIOS, J. Martínez-de; MERINO, L. Unmanned aerial vehicles as tools for forest-fire fighting. *Forest Ecology and Management*, v. 234, n. October, p. S263, 2006. ISSN 03781127.

PEDERI, Y. A. Unmanned Aerial Vehicles and New Technological Methods of Monitoring and Crop Protection in Precision Agriculture. p. 298–301, 2015.

PIXHAWK. *Pixhawk - Flight Controller Hardware Project*. 2013. Disponível em: <https://pixhawk.org/>. Acesso em: 26 out 2017.

QUINTERO, S. A. P.; COLLINS, G. E.; HESPANHA, J. P. Flocking with fixed-wing UAVs for distributed sensing: A stochastic optimal control approach. *American Control Conference (ACC)*, p. 2025–2031, 2013. ISSN 0743-1619. Disponível em: <https://www.infona.pl//resource/bwmeta1.element.ieee-art-000006580133>.

RAZA, S. A.; GUEAIEB, W. Intelligent flight control of an autonomous quadrotor. In: *Motion Control*. [S.l.]: InTech, 2010.

REYNOLDS, C. W. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, v. 21, n. 4, p. 25–34, 1987. ISSN 00978930. Disponível em: <http://portal.acm.org/citation.cfm?doid=37402.37406>.

ROS. *ROS | Powering the world's robots*. 2007. Disponível em: <http://www.ros.org/>. Acesso em: 26 out 2017.

ŞAHIN, E. Swarm robotics: From sources of inspiration to domains of application. In: SPRINGER. *International workshop on swarm robotics*. [S.l.], 2004. p. 10–20.

TAN, Y.; ZHENG, Z.-y. Research Advance in Swarm Robotics. *Defence Technology*, Elsevier Taiwan LLC, v. 9, n. 1, p. 18–39, 2013. ISSN 22149147. Disponível em: <http://www.sciencedirect.com/science/article/pii/S221491471300024X>.

VÁSÁRHELYI, G. et al. Outdoor flocking and formation flight with autonomous aerial robots. *IEEE International Conference on Intelligent Robots and Systems*, n. Iros, p. 3866–3873, 2014. ISSN 21530866.

VIRÁGH, C. et al. Flocking algorithm for autonomous flying robots. *Bioinspiration & Biomimetics*, v. 9, n. 2, p. 025012, 2014. ISSN 1748-3182. Disponível em: <http://stacks.iop.org/1748-3190/9/i=2/a=025012?key=crossref.93f6214b4940b7fa7468c73d13d633e5>.