Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Programa de Pós-Graduação em Informática Aplicada

HEALTHDRONES - Uma Plataforma em Software para Controle de Veículos Não Tripulados e Planejamento de Trajetórias

Alberto Rogério e Silva

Recife

Setembro de 2016

Alberto Rogério e Silva

HEALTHDRONES - Uma Plataforma em Software para Controle de Veículos Não Tripulados e Planejamento de Trajetórias

Orientador: Jones Albuquerque
Co-Orientador: Victor Medeiros

Dissertação de mestrado apresentada ao Curso de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do grau de Mestre em Informática Aplicada.

Recife

Setembro de 2016

A palha flutua na superfície; mas aquele que procura por pérolas deve mergulhar e aprofundar-se. (John Dryden)

Às nossas famílias.

Agradecimentos

Agradeço primeiramente a Deus.

Gostaria de agradecer também ao Programa de Pós-Graduação em Informática Aplicada da Universidade Federal de Pernambuco pelo apoio.

Um especial agradecimento à equipe de gestão e coordenação da Escola Técnica Senai Areias por todo incentivo e valiosa cooperação.

Agradecimento ao financiamento (parcial) desta pesquisa pelo National Institute of Science and Technology for Software Engineering (INES), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08

Resumo

Veículos Aéreos Não Tripulados ou VANTs passaram a fazer parte do cotidiano de entusiastas, pesquisadores, empresas, etc., nas mais variadas aplicações onde a presença de pessoas é dificultada ou têm suas vidas postas em risco. Neste trabalho, busca-se analisar os mecanismos, possibilidades e meios de controle por meio de plataformas em software, que permitem interagir com drones de pequeno porte, de forma direta ou com um mínimo de interferência humana, em diferentes cenários de utilização.

Uma das características dos veículos não tripulados, particularmente os aéreos, que mais tem despertado o interesse de pesquisadores é o fato de tais veículos possibilitarem a execução de tarefas ou missões ditas *autônomas*. Com um mínimo de informação sobre o ambiente que deverão sobrevoar, pode-se configurar completamente as ações e condições que propiciarão a tomada de decisões desses veículos, no sentido de evitarem colisões contra objetos próximos, desviarem e retornarem à trajetória da missão ou mesmo a abortarem. Com ênfase nesse tipo de atuação dos drones, apresenta-se neste trabalho uma abordagem por meio de algoritmo evolucionário, no caso, o algoritmo multiobjetivo NSGA-II [19], para o desenho de possíveis trajetórias que o drone poderá percorrer para realizar sua missão, saindo de um ponto de origem rumo a um ponto de destino sem colidir com obstáculos.

Tal abordagem é fundamentada no levantamento de dois principais e igualmente importantes objetivos: executar a missão consumindo um mínimo de recursos necessários ao voo, como a carga de bateria; e reduzir o risco de colisões contra obstáculos fixos dispostos ao longo do trajeto. Para essa proposta de utilização, o algoritmo NSGA-II mostrou-se bastante eficaz na obtenção de um conjunto de soluções factíveis e conciliando os objetivos conflitantes considerados, fornecendo em um tempo de execução computacional viável para execução de

missões, um *trade-off* de soluções bem distribuídas próximas à chamada frente convexa de Pareto.

Além disso, é feito um levantamento sistemático e quantitativo do controle de navegabilidade permitido pelo drone por meio do desenvolvimento de uma nova aplicação em software, destacando sua precisão e eficiência para o uso em missões autônomas de veículos não tripulados de pequeno porte. Também é demonstrada a possibilidade de execução de programas e rotinas fazendo-se uso de recursos de hardware e software do próprio drone, levando-o à uma operação verdadeiramente autônoma.

Como uma exemplo de aplicação contextualizada do emprego de drones voando de forma autônoma entre pontos de origem e destino e evitando obstáculos fixos, também é proposta, modelada e simulada uma situação onde o drone é utilizado como guia turístico aéreo em ambientes públicos, sendo esta uma das contribuições desse trabalho. Tal simulação, realizada com a ferramenta Arena, pode indicar a formação de filas de espera ou gargalos nos diversos setores do ambiente onde o *tour* se realiza, e que podem ser utilizada para analisar a viabilidade do uso do drone como um guia aéreo.

Abstract

Unmanned Aerial Vehicles or UAVs become part of everyday enthusiasts, researchers, companies, etc., in several kind of applications where the presence of people is difficult or have their lives put at risk. In this study, we try to analyze the mechanisms, possibilities and means of control through software platforms that allow to interact with small drones, either directly or with a minimum of human interference in different usage scenarios.

One of the characteristics of unmanned vehicles, particularly air ones, and that has attracted attention of researchers is the fact that such vehicles make possible the execution of tasks or autonomous missions. With a minimum of environmental information that will fly over, we can completely configure the actions and conditions that will provide the decision-making of these vehicles in order to avoid collisions with nearby objects, divert and return to the path of mission or even aborting. With emphasis on this type of action of drones, this work presents an approach using evolutionary algorithm, specifically the multi-objective algorithm NSGA-II, the design of possible trajectories that the drone can go to accomplish its mission, leaving a point of course rise to a point of destination without colliding with obstacles.

Such an approach is based on the statement of two main and equally important objectives: to execute the mission consuming a minimum of resources necessary for the flight, as the battery charge; and reduce the risk of collisions against fixed obstacles arranged along the path. For this proposed use, the NSGA-II algorithm proved to be very effective in getting a set of feasible solutions and reconciling the conflicting goals considered by providing in a low computational execution time, a trade-off well distributed solutions close to convex Pareto front.

In addition, a systematic and quantitative survey of the navigability control allowed by

the drone is made through the development of a new software application, highlighting its precision and efficiency for use in autonomous missions of small unmanned vehicles. It is also demonstrated the possibility of running programs and routines making use of hardware and software resources of the drone itself, leading to a truly autonomous operation.

As an example of contextualized application of drones flying autonomously, it is also proposed, modeled and simulated an real situation where the drone is used as aerial tour guide in public places, this being one of the contributions of this work. Such simulation was performed with Arena tool, and it may indicate the formation of queues and bottlenecks in many sectors of the environment where the tour takes place, which would be usefull to analyse the efetiveness of the use of drone as an autonomous guide by the air.

Lista de Figuras

1.1 Modelo de drone quadricóptero (Parrot Ar.Drone®). 2

2.1 Tipos de Drones (Fonte g1.com.br). 7

2.2 Tamanhos de Drones (Fonte g1.com.br). 8

2.3 Regras para voo de drones (Fonte g1.com.br). 11

2.4 CCP Co.'s RC NanoDrone (Imagem BLOOMBERG). 12

3.1 AR.Drone com suas querenas para voo *outdoor* e *indoor*. 19

3.2 Execução de movimentos do drone (Adaptação de [68]). 20

3.3 Movimentos do drone, conforme [58]). 20

3.4 AR.Drone executando missão controlado por *Node.js*. 22

3.5 Aplicativo para desenhar trajetos com Yadrone. 23

3.6 Acessando drone com Ar.FreeFlight. 24

3.7 Diagrama de classes do JMetal. 25

4.1 Distâncias consideradas para minimizar percurso e risco de colisão. 30

4.2 Dois possíveis trajetos obtidos da frente de soluções para o conjunto 1 de obstáculos (8 variáveis). 32

4.3 Frente de soluções obtidas para o conjunto 1 de obstáculos com 8 variáveis. . 33

4.4 Duas soluções factíveis da frente de soluções para o conjunto 1 de obstáculos (10 variáveis). 35

4.5 Frente de soluções obtidas para o conjunto 1 de obstáculos com 10 variáveis. 35

4.6 Indicação de duas soluções para o conjunto 2 de obstáculos e vetor de decisão com 10 variáveis. 37

4.7 Frente de Pareto para a disposição 2 de obstáculos. 37

5.1 Drone executando trajeto de linha. 40

5.2 Missão de ida e volta em linha reta. 40

5.3 Drone em execução do movimento retangular. 41

5.4 Configuração do ImageJ mediante a indicação de uma distância de referência conhecida. 42

5.5 Medidas dos retângulos delimitas no piso da sala de testes. 43

5.6 Leituras de percursos realizadas com o ImageJ. 43

5.7 Hélice traseira do AR.Drone danificada. 47

6.1 Variações do tempo em espera dos visitantes. 52

6.2 Variações do tempo em espera dos visitantes (mesma probabilidade de escolha de trajetos). 53

6.3 Mapa de pinos na placa interna do *AR.Drone* 55

6.4 Conexão do Arduino à placa controladora do *AR.Drone*. 56

6.5 Novos sensores conectados ao drone. 56

6.6 Distâncias a obstáculos repassadas ao drone. 58

A.1 Representação da chegada de visitantes ao local do tour. 71

A.2 Realização do atendimento aos visitantes. 71

A.3 Realização do tour pelos visitantes com o drone como guia. 72

A.4 Submodelos destacando as opções do tour. 72

A.5 Finalização do tour. 72

B.1 Sala de testes para filmagens do drone. Cortesia Escola Técnica Senai Areias (Sala 140). 73

B.2 Marcações para manter o posicionamento fixo da bancada e notebook. 74

B.3 Comprimentos principais de quadros no piso. 74

Lista de Tabelas

4.1 Soluções para as funções objetivos com base no primeiro conjunto de obstáculos assumindo-se 4 pontos (8 variáveis). 31

4.2 Soluções para as funções objetivos com base no primeiro conjunto de obstáculos assumindo-se 5 pontos (10 variáveis). 34

4.3 Soluções para as funções objetivos com base no segundo conjunto de obstáculos assumindo-se 5 pontos (10 variáveis). 36

5.1 Resultados para as missões de linha empregando os *frameworks Node.js e Yadrone* como plataformas de desenvolvimento (valores em centímetros). . . 44

5.2 Resultados para a missão de trajeto retangular empregando os *frameworks Node.js e Yadrone* (valores em centímetros). Testes 01 e 02. 45

5.3 Resultados para a missão de trajeto retangular empregando os *frameworks Node.js e Yadrone* (valores em centímetros). Testes 03 e 04. 45

5.4 Resultados para a missão de trajeto retangular com movimentos para frente seguidos de rotação (valores em centímetros). Testes 01 e 02. 46

5.5 Resultados para a missão de trajeto retangular com movimentos para frente seguidos de rotação (valores em centímetros). Testes 03 e 04. 46

6.1 Resultados para as configurações iniciais do modelo, com variação da média do tempo entre chegadas de visitantes entre 10 e 30 minutos. 51

6.2 Resultados para as configurações iniciais do modelo, com variação da média do tempo entre chegadas de visitantes entre 40 e 60 minutos. 51

6.3 Resultados assumindo a escolha de trajetos distintos, com variação da média do tempo entre chegadas de visitantes entre 10 e 30 minutos. 53

6.4 Resultados assumindo a escolha de trajetos distintos, com variação da média do tempo entre chegadas de visitantes entre 40 e 60 minutos. 53

Lista de Abreviaturas e Siglas

ANATEL	Agência Nacional de Telecomunicações
AGL	Altitude Above Ground Level
BVLOS	Beyond Visual Line Of Sight
DECEA	Departamento de Controle do Espaço Aéreo
FAA	Federal Aviation Administration
FPS	Frames Per Second
FPV	First Person View
GPS	Global Positioning System
HD	High Definition
IMU	Inertial Measurement Unit
IOC	Intelligent Orientation Control
LiPo	Lithium Polymer
LOS	Line of Sight
PID	Proporcional-Integral-Derivativo
QVGA	Quarter Video Graphics Array
RC, R/C	Radio Controlled
ROS	Robot Operating System
RPA	Remotely-Piloted Aircraft
RTF	Ready to Fly
SUAS	Small Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
VANT	Veículo Aéreo Não Tripulado

Lista de Símbolos

Δ_A	Variação positiva da velocidade do motor do drone
Δ_B	Variação negativa da velocidade do motor do drone
Ω_H	Velocidade de rotação do motor do drone
ϕ	Ângulo de rolagem (*roll*)
ψ	Ângulo de guinada (*yaw*)
θ	Ângulo de arfagem (*pitch*)

Sumário

1 Introdução **1**

1.1 Apresentação 3

1.2 Justificativa 4

1.3 Objetivos 4

1.4 Organização do Trabalho 5

2 Revisão da Literatura **7**

2.1 VANTs - Veículos Aéreos Não Tripulados 7

2.2 Navegabilidade em Veículos não Tripulados 13

3 Ar.Drone: Controlabilidade e Missões Autônomas **17**

3.1 Parrot AR.Drone 17

3.2 Softwares de Controle do Drone 21

3.2.1 Node.Js 21

3.2.2 Yadrone 22

3.2.3 AR.FreeFlight 23

3.3 *Framework* JMetal 24

4 Otimização Evolucionária de Trajetórias **26**

4.1 Planejamento de Trajetórias 26

4.2 Método Proposto 28

5 Resultados e Discussão **38**

5.1 Controlabilidade por *Frameworks* 38

5.1.1 Preparação do ambiente 38

5.1.2 Detalhamento das missões 39

5.1.3 Resumo de resultados obtidos 42

6 Navegabilidade Autônoma - Estudo de Caso **48**

6.1 Uso de Drones como Guias Aéreos 48

6.1.1 Chegada de entidades (visitantes) 49

6.1.2 Atendimento e verificação de prioridades 49

6.1.3 Realização do *tour* 49

6.1.4 Refinando o *tour* com submodelos 50

6.1.5 Finalizando o *tour*: saída de entidades 50

6.1.6 Simulação do modelo e resultados obtidos 50

6.2 Execução Interna de *Scripts* 54

7 Conclusão **59**

8 Perspectivas **61**

Referências Bibliográficas **63**

A Modelagem e Simulação no Arena **71**

A.1 Chegada de Entidades 71

A.2 Realização de Atendimento 71

A.3 Realização do Tour 72

A.4 Refinamento do Tour 72

A.5 Finalizando o Tour 72

B Ambiente de Testes de Voos **73**

Capítulo 1

Introdução

Veículos Aéreos Não Tripulados (VANTs) estão sendo cada vez mais utilizados nas mais diversificadas funções, tanto em aplicações militares como civis, industriais, comerciais e domésticas. Em quaisquer das operações em que tais veículos são envolvidos, é importante que sejam convenientemente controlados, além disso, podem ser desenvolvidos modelos que permitam descrever, prever e avaliar situações excepcionais de voo e que facilitem o percurso da trajetória pretendida [11].

VANTs de pequeno porte, dotados de pequenos motores e baixa autonomia, são conhecidos como drones, Figura 1.1, e tem despertado grande interesse no âmbito comercial, possibilitando a monitoração em tempo real de regiões e espaços de difícil acesso ou hostis. Dotados de versáteis câmeras de alta resolução e permitindo a troca de informações de forma *wireless*, esses dispositivos conseguem alcançar elevadas altitudes com autonomia de bateria que pode variar de alguns poucos minutos a horas.

Conforme atesta o Departamento de Defesa Americano (*Departament of Defense – DoD*) VANT é definido como "um veículo aéreo motorizado que não transporta um operador humano, usa forças aerodinâmicas para a sustentação aérea, pode voar de maneira autônoma ou ser pilotado por controle remoto, pode ser descartável ou recuperável e pode transportar uma carga útil letal ou não-letal. Veículos balísticos ou semibalísticos, mísseis de cruzeiro e projéteis de artilharia não são considerados veículos aéreos não tripulados" [14].

O termo drone tem origem inglesa, sendo traduzido com zangão ou zumbido, ressaltando um

ruido característico que os drones emitem por meio de seus motores e hélices. Por questões de aerodinâmica, os drones utilizam a rotação controlada de seus rotores para manterem sua estabilidade e condições de voo.

Figura 1.1: Modelo de drone quadricóptero (Parrot Ar.Drone®).

Já há algum tempo e, nos dias atuais de forma bem mais frequente, VANTs têm sido utilizados para fins militares como meio de espionagem a ataques aéreos, poupando a vida de soldados e oficiais em tais operações [17]. Seu uso tem se tornado bastante requisitado para desempenhar funções de vigilância e supervisão de áreas consideradas perigosas ou de difícil acesso. Uma vez que os VANTs são veículos que não necessitam de pilotos, eles são controlados remotamente ou de forma autônoma. São utilizados complexos algoritmos de controle e estabilidade que não requerem a intervenção humana [15].

É creditado a Nicola Tesla os conceitos iniciais de voo não tripulado em 1915, conforme [3]. Já em 1916, Lawrence e Elmer Sperry apresentaram um modelo de veículo aéreo dotado de navegação automática, que ficou conhecido como *torpedo aéreo* [16], mas o primeiro VANT foi apresentado durante a Primeira Guerra Mundial apenas em 1917.

Entretanto, a grande diversidade de sistemas comerciais para veículos aéreos não tripulados possuem suas próprias interfaces de controle de trajetórias, fazendo-se uso de algum tipo de sensoriamento para mapeamento de sua posição, como por exemplo GPS ou GPRS. Isto limita bastante o uso de tais dispositivos para voos autônomos em ambientes relativamente pequenos (com raio de cobertura próximo ao alcance de sinais emitidos por *Wi-Fi*, por exemplo), fechados, remotos ou de difícil acesso, cujo sensoriamento seja prejudicado.

Assim, faz-se necessário a viabilização de mecanismos de navegabilidade autônoma e a disponibilização disso em forma de uma plataforma em software que possa enviar comandos de operação e coordenadas de posicionamento ao drone, permitindo dessa forma um meio alternativo de sensoriamento para o controle de missão.

1.1 Apresentação

Neste trabalho é proposto um controle de trajetória para drones por meio de uma abordagem evolucionária. Diversas técnicas empregando algoritmos genéticos [53] têm sido bastante utilizadas na resolução de diferentes problemas [59, 52], com inúmeras aplicações envolvendo o planejamento e controle de trajetórias para veículos aéreos não tripulados [33, 43, 54, 4], e ampla aplicabilidade para desenvolvimento com aceleradores de hardware e dispositivos programáveis [42]. Neste trabalho, é utilizado o algoritmo evolucionário multiobjetivo NSGA-II para buscar um conjunto de soluções que descrevem as melhores trajetórias a serem percorridas por um VANT de modo a evitar colisões com obstáculos estáticos posicionados ao longo do percurso.

Os diversos modelos de drones existentes atualmente são projetados com a possibilidade de controle de movimentação básica através de comandos diretos e também com a possibilidade de construção de missões consideradas autônomas, ou seja, são configurados pontos ao longo de um percurso ou mapa para que o drone navegue entre eles. Dessa forma, um levantamento de diversos mecanismos de controle (*frameworks*) para drones é apresentado, permitindo observar para cada um deles sua eficiência, modos de operação e controlabilidade. Cada *framework* disponibiliza um conjunto de recursos utilitários que permite estabelecer uma conexão com o drone e enviar-lhe comandos de movimentação bem como recuperar informações diversas acerca do estado em que o drone se encontra como seu *status*, posição, velocidade, carga atual da bateria, altitude, velocidade de rotação dos motores, etc.

Também é apresentado um estudo de caso que analisa através de modelagem, um cenário de aplicação onde um drone é utilizado por um grupo de pessoas como um guia aéreo que as acompanha em um *tour* por diferentes setores de um espaço aberto ou edifício. A simulação do modelo é realizada com o programa Arena® (*Rockwell Automation, Inc*) e reproduz algumas situações de desempenho onde são verificados os tempos gastos em diferentes estágios do *tour*.

1.2 Justificativa

Existem atualmente disponíveis inúmeros softwares aplicativos que permitem o controle de drones de todos os tipos. Esses controles não se limitam a simples movimentações em torno dos eixos x, y e z mas fornecem ao controlador meios de configurar missões e executá-las. Alguns desses softwares são desenvolvidos para controle de drones por meio de conexão *Wi-Fi*, o que lhes dá um alcance limitado (entre 50 e 100m). Qual o nível de controle e precisão que estas plataformas em software possibilitam? Que tipos de operações e particularidades de comandos estão disponíveis para a atuação confiável do drone? É possível executar missões com o uso de drones sem intervenção humana alguma, ou seja, é possível a execução de tarefas fazendo-se uso do sistema e memória incorporados à estrutura interna do drone? Nesse trabalho, verifica-se que modelos de drones, como o *Parrot AR.Drone 2.0*, possibilitam esse tipo de operação ao permitirem a execução de *scripts* ou rotinas em seu sistema operativo. Essas são algumas perguntas que esse trabalho busca responder.

Outros questionamentos surgem ao se empregar planejadores de trajetórias para a obtenção de rotas para veículos não tripulados. Por meio de uma abordagem evolucionária, justifica-se nesse trabalho que eficientes algoritmos multiobjetivos podem ser empregados de forma tão confiável quanto se deseje para o repasse ao drone de coordenadas a serem seguidas em suas missões, tal que tais missões sejam completadas com um risco reduzido no que se refere à possibilidade de choques contra obstáculos, e com economia de recursos importantes.

1.3 Objetivos

Como objetivo geral desse trabalho, pretende-se:

- Analisar e descrever uma nova plataformas em software para controlabilidade de veículos aéreos de pequeno porte do tipo drones, no que se refere ao envio de comandos e coordenadas de posicionamento para a execução de trajetórias planejadas.

Para alcançar tal objetivo, os seguintes objetivos específicos são trabalhados:

- Apresentar o modo de atuação de *frameworks* desenvolvidos para enviarem comandos a drones por meio de comunicação por *Wi-Fi*;
- Comprovar a eficiência de comandos de voo desses *frameworks* na forma de uma nova plataforma de controle para a execução de trajetos simples;
- Utilizar, por meio de uma abordagem evolucionária, um planejador de trajetórias de modo a repassar ao drone coordenadas a serem percorridas;
- Confirmar o processamento de *scripts* contendo tarefas por meio do sistema operativo do próprio drone;
- Apresentar e simular uma situação aplicável ao uso de drones como veículos autônomos em missões planejadas;

1.4 Organização do Trabalho

Este trabalho está organizado da seguinte forma.

O **Capítulo 1** contempla uma introdução e apresentação do trabalho, justificativa para seu desenvolvimento e objetivos buscados.

O **Capítulo 2** apresenta a revisão da literatura e do estado de arte, destacando trabalhos relevantes já desenvolvidos e especificações técnicas sobre drones.

O **Capítulo 3** descreve os modos de controle de missões de drones com base no modelo *Parrot Ar.Drone 2.0*. Destaca-se também neste capítulo os softwares utilizados para envio de comandos ao drone e implementação de algoritmos evolucionários.

O **Capítulo 4** relata o método proposto para a obtenção de trajetos otimizados por abordagem evolucionária e ilustra parte das soluções obtidas em diferentes cenários.

No **Capítulo 5** são apresentados os resultados obtidos para os testes realizados de controlabilidade e planejamento de trajetórias.

No **Capítulo 6** é apresentado um breve estudo de caso para o uso de drones como guias

aéreos em ambientes públicos. Nesse capítulo verifica-se também a possibilidade da execução de programas fazendo uso do sistema de controle incorporado ao drone.

O **Capítulos 7** traz a conclusão e uma panorâmica sobre o trabalho desenvolvido e seus resultados.

No **Capítulo 8** são apresentadas algumas propostas de trabalhos futuros.

Capítulo 2

Revisão da Literatura

2.1 VANTs - Veículos Aéreos Não Tripulados

De uma forma geral, os VANTs são classificados em dois grandes grupos: o de asas fixas e de asas rotativas ou giratórias, Figura 2.1. Uma grande vantagem dos veículos de asas rotativas é a possibilidade de efetuar movimentos verticais para cima e para baixo bem como de permanecerem parados no voo (pairar). No Brasil, conforme pode ser verificado em [57], a legislação proíbe o uso de tais aparelhos de forma completamente autônoma ou seja, quando são usados para desenvolverem missões sem intervenção humana, sendo permitido apenas o uso de VANTs do tipo RPA (*Remotely-Piloted Aircraft*).

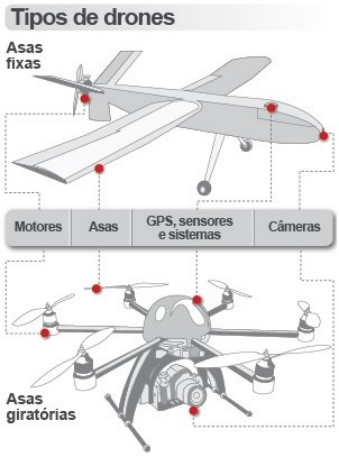


Figura 2.1: Tipos de Drones (Fonte g1.com.br).

Atualmente os drones são construídos e projetados com os mais variados tamanhos e propósitos, não havendo uma especificação padronizada com base em tipos e modelos. Com relação ao tamanho, como representado na Figura 2.2, existe uma classificação que normalmente é empregada, conforme segue abaixo:

- Drones muito pequenos (*Very small drones*): Drones que não ultrapassam 50cm de comprimento. São encontrados de duas formas: Nano ou micro drones e mini drones.
- Drones pequenos (*Small drones*): A maioria desses drones são do tipo asa fixa, não ultrapassando 2m de comprimento.
- Drones médios (*Medium drones*): Esses drones são maiores e mais pesados que os tipos anteriores, mas ainda são menores que aeronaves comuns. Com envergadura entre 5 e 10m podem transportar cargas de até 200kg.
- Drones grandes (*Large drones*): Apresentam tamanho comparável a algumas aeronaves e possuem aplicação tipicamente militar [62].

Figura 2.2: Tamanhos de Drones (Fonte g1.com.br).

A ANAC (Agência Nacional de Aviação Civil) apresentou uma proposta para regulamentação do uso de drones por civis. A divulgação, ocorrida em Setembro de 2015, sugere uma

série de regras que devem ser observadas por todos que desejam operar aeronaves remotamente pilotadas (RPA). A seguir é apresentado um resumo da proposta sugerida, conforme veiculado em [23]:

Classe 1 (peso maior que 150 kg) – Aeronaves deverão ser certificadas pela ANAC, serão registradas no Registro Aeronáutico Brasileiro (RAB) e pilotos deverão possuir Certificado Médico Aeronáutico (CMA), licença e habilitação. Todos os voos deverão ser registrados.

Classe 2 (peso menor ou igual a 150 kg e maior que 25 kg) – Aeronaves não precisarão ser certificadas, mas os fabricantes deverão observar os requisitos técnicos exigidos e ter o projeto aprovado pela Agência. Também deverão ser registradas no RAB e pilotos deverão possuir CMA, licença e habilitação. Todos os voos também deverão ser registrados.

Classe 3 (peso menor ou igual a 25 kg) – Se operados até 400 pés acima do nível do solo (aproximadamente 120 metros) e em linha visada visual, serão apenas cadastrados (apresentação de informações sobre o operador e o equipamento). Não será requerido CMA nem será necessário registrar os voos. Licença e habilitação somente serão requeridas para quem pretender operar acima de 400 pés. As operações de RPA até 25 kg só poderão ocorrer a uma distância mínima de 30 metros de uma pessoa. A distância pode ser menor no caso de pessoas anuentes (aquelas que concordarem expressamente com a operação) ou de pessoas envolvidas na operação. Em áreas urbanas e aglomerados rurais, as operações serão de no máximo 200 pés acima do nível do solo (aproximadamente 60 metros).

Idade mínima – Os pilotos de RPA das três classes deverão ser maiores de 18 anos.

Seguro – Será exigido seguro com cobertura de danos a terceiros para todos os RPA (das três classes), com exceção de órgãos de segurança pública e defesa civil.

Atividades ilícitas ou invasão de privacidade – Atividades ilícitas ou invasão de privacidade com uso de RPA serão naturalmente tratadas pelas autoridades de segurança pública competentes.

Defesa civil e segurança pública – Órgãos de segurança pública e defesa civil poderão operar em quaisquer áreas, sob responsabilidade do órgão (ou do operador que estiver a serviço deles), desde que observadas as demais exigências da futura norma. Essas operações

não precisarão possuir seguro com cobertura de danos a terceiros.

Aeromodelos – No caso de aeromodelos (que são aeronaves destinadas à recreação), não haverá necessidade de autorização da ANAC, mas deverá ser observada a distância mínima de 30 metros de pessoas não anuentes. No caso de pessoas anuentes (que concordem expressamente), essa distância não precisará ser observada. Pela proposta, não há idade mínima para os pilotos de aeromodelos nem obrigatoriedade de seguro contra danos a terceiros.

Regras atuais (até entrada em vigor do novo regulamento) – Atualmente, a legislação (Lei nº. 7.565/86) determina que, para operar, qualquer aeronave deve ser autorizada. No âmbito da ANAC, a Instrução Suplementar (IS nº 21-001) de 2012 prevê a emissão de autorização para uso de VANT (RPA) somente para pesquisa e desenvolvimento e treinamento de pilotos. Essas autorizações da ANAC não excluem a necessidade de anuência de outros agentes públicos como DECEA e ANATEL. Para o uso de aeromodelos, vigora hoje a Portaria DAC n° 207/STE/1999, na qual os equipamentos devem respeitar a restrição de não operar nas zonas de aproximação e decolagem de aeródromos e nunca ultrapassar altura superior a 400 pés (aproximadamente 120 metros) mantendo-se o equipamento sempre ao alcance da visão do piloto.

Atualmente os VANTs de pequeno porte têm sido utilizados em operações cotidianas que não exigem grande habilidade do operador ou regras. O uso desses veículos não se limitam a simples filmagem, ou captura de imagens. Como observado em [26], as aplicações onde os VANTs podem ser encontrados são organizadas principalmente em quatro grupos: aplicações ambientais, aplicações de segurança, aplicações de comunicação e aplicações de monitoramento e vigilância.

A Figura 2.3 resume algumas das regras de voo para drones conforme descrito pelo Departamento de Controle do Espaço Aéreo.

Figura 2.3: Regras para voo de drones (Fonte g1.com.br).

Diversos modelos de drones estão disponíveis para serem utilizados no âmbito comercial e de lazer. Muitos deles vem incorporados com sofisticados controles de voo e estabilidade em conjunto com um aparato profissional de filmagens com captura e processamento de imagens. Em Junho deste ano, o **RC Nanodrone** foi apresentado na *Tokyo Toy Show*, Figura 2.4. Descrito como o menor drone do mundo, possui 3.5cm de largura com peso de 9g (11g quando equipado com sua câmera). Capaz de filmar 4 minutos durante o voo.

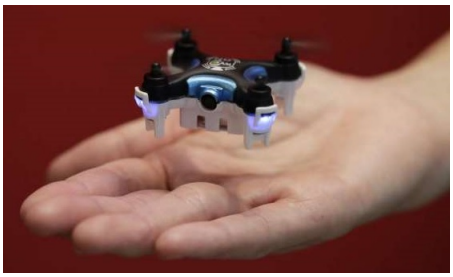


Figura 2.4: CCP Co.'s RC NanoDrone (Imagem BLOOMBERG).

Em 2015, pesquisadores da Universidade de Oakland apresentaram o *Looncopter*, um drone que, além do tradicional voo ao ar livre, também pode se mover submerso em água.

O próprio modelo *AR.Drone* desenvolvido e comercializado pela empresa **Parrot**, tem sido amplamente utilizado em vários campos de pesquisas. No ano de 2012, um estudo considerando a ação conjunta entre *software* e *hardware* com ênfase na autonomia aérea foi proposta, conforme elucidado em [47]. Nesse estudo foi utilizado o *AR.Drone* como quadricóptero experimental executando missões autônomas. Um ponto bastante interessante da pesquisa foi a utilização de um algoritmo de localização por meio de técnicas de **SURF** (*Speeded Up Robust Features*) [6], onde câmeras forneciam as imagens necessárias à execução de tarefas autônomas.

Apesar de um número cada vez mais crescente de pesquisadores que decidem construir por inteiro seus próprios veículos não tripulados [69], estudos experimentais que fazem o uso de VANTs, optam pelo modelo quadricóptero já pronto. Este modelo é amplamente encontrado no comércio, permitindo facilmente sua modificação, como a incorporação de hardware para permitirem ao veículo por exemplo, a utilização de novos tipos de sensores, [8].

Em 2013, conforme especificado em [65], um *hardware* baseado no modelo de quadricóptero foi construído de maneira a demonstrar a viabilidade operacional de controles autônomos embarcados. O sistema em si empregou uma placa controladora *ArduPilot* Mega® em conjunto com um microcontrolador **STM32F4**, representando uma abordagem funcional do controle autônomo incorporado ao projeto de VANTs.

Missões autônomas com drones, simuladas em software, também têm sido objeto de estudo por vários pesquisadores, como em [9], que propõe um controlador não linear subatuado para

tarefas de voo em três dimensões. Uma plataforma de simulação desenvolvida empregando o MATLAB® e baseada na teoria de Lyapunov [10] foi utilizada para demonstrar a funcionalidade do controlador para operar VANTs, sendo destacada sua eficiência para manter a estabilidade do veículo.

2.2 Navegabilidade em Veículos não Tripulados

Um grande número de estudos e experimentos envolvendo o planejamento de trajetórias para veículos não tripulados tem sido feitos ao longo do tempo. Tais estudos levam em consideração uma ampla variedade de diferentes fatores que favorecem a obtenção de trajetos segundo alguns critérios como, distância total percorrida, autonomia de voo, presença de obstáculos, dentre outros [38, 5].

Uma abordagem empregando algoritmos genéticos para planejamento de trajetórias *off-line* é demonstrada em [60]. O objetivo primordial do estudo é propor trajetos para um dado ponto alvo, minimizando a exposição a ameaças e garantindo a manutenção de recursos em níveis seguros, como o consumo de combustível.

Uma metodologia empregando algoritmos bio inspirados para o planejamento de percursos é apresentada em [72], o qual propõe um método baseado em otimização de colônia de formigas para o desenho de caminhos para VANTs. O estudo é baseado na forma com que formigas buscam naturalmente seus alimentos, chegando a um ponto alvo, passando por vários outros pontos no cenário de navegação.

O cenário descrito em [50], utiliza algoritmos genéticos baseados em recozimento simulado (*Simulated Annealing*) para o planejamento de pistas de voo para VANTs. Neste trabalho, o chamando Mapa de Elevação Digital (DEM), é processado em uma superfície com um mínimo de ameaças. De maneira a obter uma superfície de voo mais suavizada, o mapa de elevação original é processado em quatro direções, onde o algoritmo de recozimento simulado é utilizado para o planejamento de rotas tridimensionais.

O planejamento de trajetórias para VANTs em ambientes 3D por meio de algoritmos genéticos paralelos é apresentado em um promissor estudo em [73]. Os pesquisadores discutem

como o planejamento de trajetos pode ser feito evitando ameaças provenientes de uma rede de radares com diferentes raios de detecção. Uma abordagem fazendo uso de algoritmos genéticos paralelos é apresentada de modo a reduzir o tempo computacional no cálculo de caminhos. O mapeamento em três dimensões do ambiente é feito por meio da ferramenta *open source World Wind* [66], um programa desenvolvido pela NASA que apresenta ao usuário imagens em 3D do globo terrestre, semelhante ao Google Earth®. O estudo provou ser uma maneira conveniente de encontrar percursos rápidos e seguros para voos autônomos de veículos não tripulados.

Algoritmos genéticos e outras abordagens evolucionárias têm encontrado uma grande variedade de aplicações envolvendo o planejamento de trajetórias por permitir incorporar ao problema de buscar trajetos em meio a obstáculos, alguns objetivos conflitantes [55, 48, 61, 30], em particular o algortitmo NSGA-II [19]. Este algoritmo é uma evolução da primeira versão chamada NSGA (*Non-dominated Sorting Genetic Algorithm*) desenvolvido pelo professor Kalyanmoy Deb[1] para a resolução de problemas de otimização não convexas multiobjetivos e de objetivo único. Suas principais características são descritas abaixo:

- Um procedimento de ordenação não-dominada onde todos os indivíduos são ordenados conforme o nível de não-dominância, ou seja, busca por soluções próximas a frente de Pareto.
- Implementação de elitismo que guarda todas as soluções não-dominadas, melhorando assim as propriedades de convergência.
- Apresenta um mecanismo automático funcional baseado na distância populacional (*crowding distance*) de maneira a garantir diversidade e espalhamento de soluções.
- Emprega uma definição modificada de dominância para a implementação de restrições sem lançar mão de funções de penalidade.

Em [31], são apresentadas técnicas de otimização avançadas para planejamento de missões de aeronaves com o objetivo de detectar e monitorar esporos e agentes patogênicos em plantas. São analisados e comparados dois tipos de otimizadores multiobjetivos: O NSGA-II, e um

[1]RID: B-1563-2009 - Michigan State Univeristy, MSU

algoritmo baseado em estratégias de jogos híbridos (*Hybrid Game*). Tais algoritmos devem fornecer um conjunto de trajetórias livres de colisão em um espaço de três dimensões. Os trajetos são então representados por meio de curvas *splines* do ponto de início ao ponto final e deste ao ponto de início novamente. As duas abordagens são então comparadas em relação a seu custo computacional e complexidade de projeto.

Outro importante estudo envolvendo múltiplos veículos aéreos não tripulados é descrito em [7], onde é proposto um planejador de caminhos também baseado em algoritmos evolucionários em cenários realísticos. Múltiplos critérios são levados em consideração para a obtenção de percursos, como restrições de terreno, radares e mísseis. Um ponto relevante a ser mencionado é a possibilidade do funcionamento online do planejador, que recalcula trechos do caminho original para evitar riscos inesperados. Os testes e simulações realizados comprovam a eficácia do planejador proposto, tornando-o apropriado para o uso em missões reais.

Diferentes aspectos do estudo aqui proposto diferem das abordagens apresentadas e contribuem na busca de formas eficientes para a obtenção de trajetórias otimizadas de drones de pequeno porte autônomos. Dentre esses aspectos destacam-se a utilização de simples equações baseadas em cálculos de distâncias para a obtenção da função objetivo empregada no algoritmo evolucionário. A equação leva em consideração a distância média entre pontos da trajetória encontrados pelo algoritmo e os pontos que representam obstáculos presentes no cenário. Outro aspecto importante diz respeito à possibilidade de execução de rotinas fazendo-se uso do próprio sistema operativo utilizado pelo drone, o que mostrou-se possível para o modelo empregado, *Parrot Ar.Drone 2.0 Elite Edition*, por meio do acesso à sua placa controladora, sem que isto afetasse o desempenho a controlabilidade do mesmo; tal acesso também permitiu o uso de novos sensores conectados ao drone por meio de um módulo *Arduino Uno* bem como a troca de informações com este.

Uma nova metodologia também foi utilizada para a verificação e comprovação do controle de veículos não tripulados por meio de alguns *frameworks* baseados em diferentes linguagens de programação e plataformas de execução. Tal metodologia é inteiramente baseada em testes de voos que foram executados utilizando-se computadores do tipo *notebooks* e dispositivos móveis como *smartphones* e *tablets*. A análise da eficiência de controle, confiabilidade e precisão dos *softwares* empregados foi efetuada por meio da investigação de imagens prove-

nientes de vídeos de gravação das missões de voo do drone. Destaca-se assim, de que forma uma dada plataforma em software é propícia para o controle de veículos não tripulados do tipo drones, bem como quão controlável ou autônomo tal veículo se apresenta.

Capítulo 3

Ar.Drone: Controlabilidade e Missões Autônomas

Diversos estudos e projetos têm sido apresentados para demonstrar de maneira eficiente a controlabilidade de veículos aéreos não tripulados. O modelo AR.Drone da empresa francesa Parrot™, tem sido usado por vários pesquisadores em diferentes cenários como objeto de modelagem, controle, navegação autônoma [49, 27], por apresentar inúmeras vantagens como a possibilidade de comunicação por *Wi-Fi*, o que permite a estes aparelhos receberem comandos e instruções de *tablets*, aparelhos celulares, computadores pessoais, etc.

3.1 Parrot AR.Drone

O drone utilizado neste trabalho foi o modelo *quadrotor AR.Drone 2.0 - Elite Edition* desenvolvido pela empresa Parrot™ (Figura 1.1). Este drone apresenta várias características que o tornam amplamente utilizado em vários cenários de pesquisa e estudo. Dentre essas característica, destacam-se:

- Cascos de Polipropileno Expandido (PPE).
- Peças de plástico de nylon com 30% de fibra de alta qualidade.
- Bateria recarregável de 1.000 mA/H LiPo com 3 elementos.

- Pairagem de emergência controlada por software.
- Processador A8 Cortex ARM de 32 bits, 1GHz com vídeo de 800MHz, DSP, TMS320DMC64x.
- 1Gbit DDR2 de RAM a 200MHz.
- Wi-Fi™: b/g/n.
- Acelerômetro de 3 eixos +/- 50mg de precisão.
- Giroscópio de 3 eixos com precisão de 2000°/segundo.
- Sensor de pressão +/- 10 Pa de precisão (80 cm no nível do mar).
- Câmera HD frontal de 30 FPS com resolução 1280x720 pixels (720p).
- Câmera QVGA vertical de 60 FPS para medição da velocidade terrestre.
- Magnetômetro de 3 eixos, 6° de precisão.
- Sensores de ultra-som para medição da altitude terrestre.
- USB 2.0 de alta velocidade para extensões.

Apesar desse modelo apresentar uma estrutura à base de polipropileno expandido bastante leve, o que dificulta seu controle em ambientes de média ou alta perturbação [63], muitos *frameworks* foram projetados para permitir a troca de informações com o drone, enviando comandos e recebendo dados enquanto a conexão é mantida. Para incentivar e permitir o desenvolvimento de aplicações , a Parrot disponibilizou um *SDK (Software Development Kit)* [58] , que possibilita interagir facilmente com o drone por meio de sistemas *Android, iOS* e vários outros.

Além de vários recursos voltados para os desenvolvedores, o *AR.Drone* também conta com um *kernel linux* 2.6.32, o que propicia um ambiente robusto para a operação de programas que executam no próprio drone. Alguns estudos demonstrando esta particularidade já foram comprovados e alcançaram boa eficácia [51].

Apresentado em Janeiro de 2010 como um brinquedo de alta tecnologia e projetado com sofisticados mecanismos de controle para jogos empregando realidade aumentada, rapidamente o *AR.Drone* tornou-se um importante recurso de pesquisa por diversas universidades, sendo bastante utilizado nos campos de robótica, modelagem matemática, inteligência artificial, visão computacional, processamento de imagens e inúmeras outras áreas [44].

Dependendo do ambiente onde se deseja controlar o drone, é ainda possível operá-lo fazendo-se o uso das querenas exterior (*outdoor*) e interior (*indoor*) construídas com polipropileno expandido (Figura 3.1), o que, além de proteger a estrutura interna do drone e suas partes elétricas ao evitar o contato acidental com pessoas, animais e objetos próximos, também garante ao drone uma leveza extraordinária.

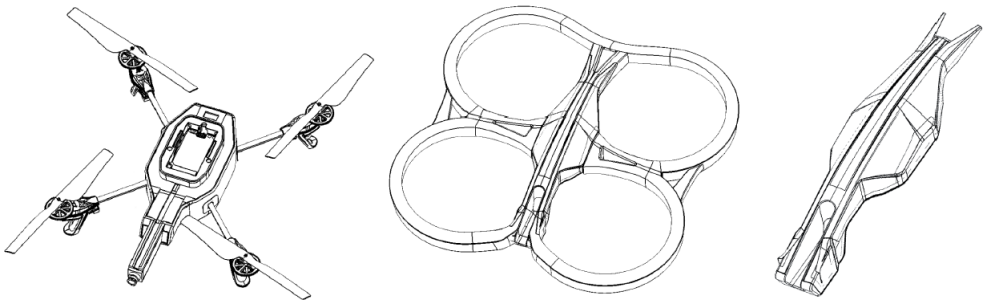


Figura 3.1: AR.Drone com suas querenas para voo *outdoor* e *indoor*.

O drone tipo *quadrotor* é composto por quatro motores DC (corrente contínua) sem escovas nos quais são conectadas quatro hélices, onde um par rotaciona no sentido horário e outro par rotaciona no sentido anti-horário, de tal forma que hélices adjacentes rotacionem em sentidos opostos. Esta disposição de motores e hélices favorece o modo como o drone executas seus diversos movimentos, além de eliminar o torque proveniente dos próprios motores [39].

Para que o drone realize movimentos variados, é preciso controlar precisamente o modo como os motores operam, tanto em relação ao sentido de giro quanto à velocidade empregada. A Figura 3.2 adaptada de [68] apresenta o sentido e intensidade do giro de cada motor para a realização de movimentos.

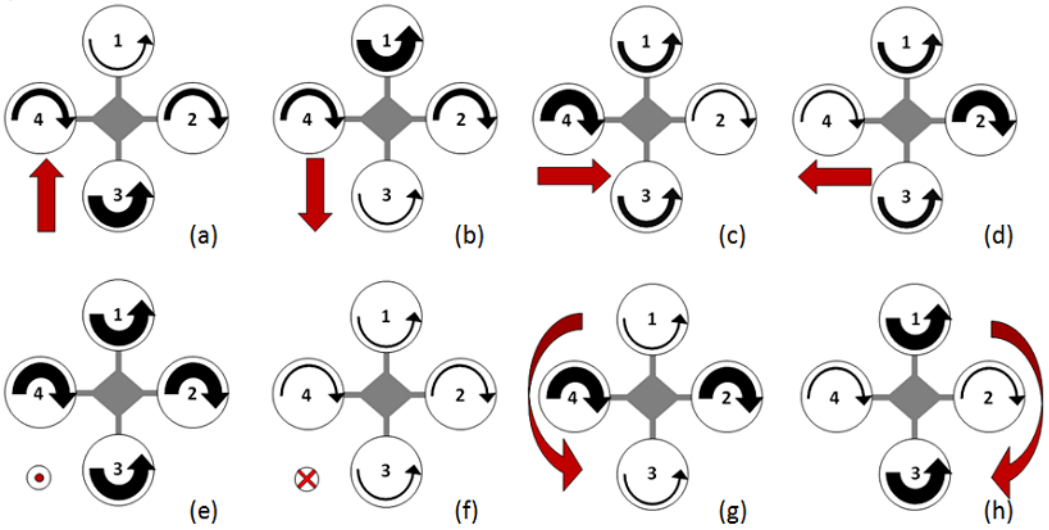


Figura 3.2: Execução de movimentos do drone (Adaptação de [68]).

A Figura 3.3 ilustra os modos de operação dos motores conforme descrito em [58]. Pela imagem, destacam-se quatro importantes movimentos. O movimento realizado em torno do eixo x é representado por ϕ, sendo conhecido como ângulo de rolagem ou *roll*. O movimento realizado em torno do eixo y é representado por θ e é conhecido como ângulo de arfagem ou *pitch*. O movimento em torno do eixo z é representado por ψ, sendo conhecido por ângulo de guinada ou *yaw*. O movimento realizado ao longo do eixo z, caracteriza o deslocamento vertical para cima ou para baixo do drone.

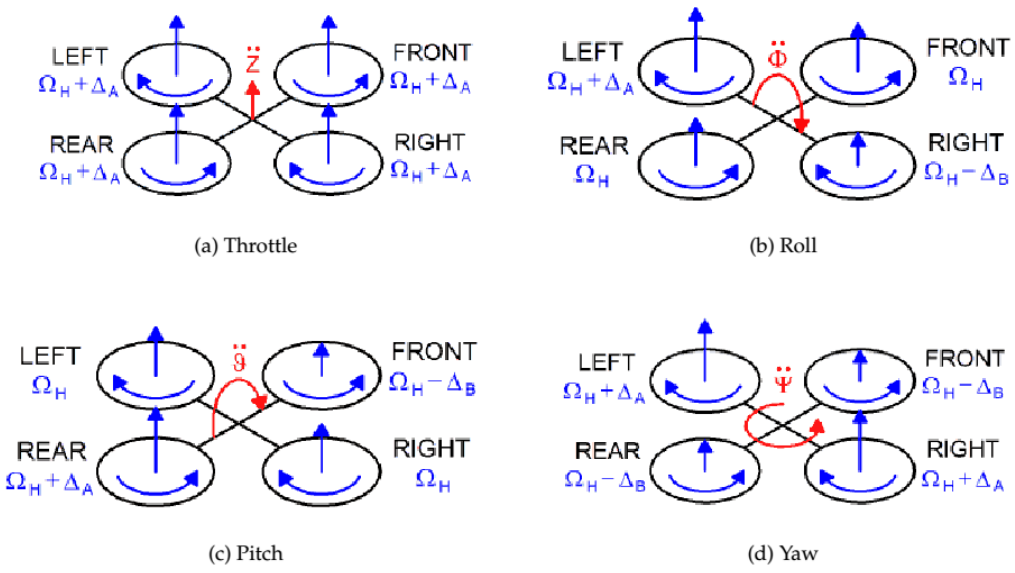


Figura 3.3: Movimentos do drone, conforme [58]).

Como pode ser observado das ilustrações apresentadas na Figura 3.2 e Figura 3.3, o movimento vertical do *quadrotor* é conseguido por meio de um incremento Δ_A (movimento para cima) ou decremento Δ_B (movimento para baixo) gradativo e de igual intensidade nas rotações (Ω_H) dos quatro motores simultaneamente, Figuras 3.2-e, 3.2-f e 3.3-a. A movimentação para a esquerda (direita) se dá por meio de uma leve rolagem no drone através de um decremento (incremento) na intensidade de rotação do motor esquerdo (*left ou motor 4*) acompanhado de um incremento (decremento) na intensidade de rotação do motor direito (*right ou motor 2*) com os demais motores *front e rear (1 e 2 respectivamente)* sem sofrerem alterações, Figuras 3.2-c, 3.2-d e 3.3-b. A movimentação para frente (trás) é obtida com uma leve arfagem no quadrirotor por meio de um incremento (decremento) na intensidade de rotação do motor traseiro (*rear ou motor 3*) em conjunto com um decremento (incremento) na intensidade de rotação do motor frontal (*front ou motor 1*) com as rotações dos outros dois motores *left e right (4 e 2 respectivamente)* mantidas, Figuras 3.2-a, 3.2-b e 3.3-c. A rotação ao redor do eixo z é conseguida por meio do movimento de guinada, variando-se a intensidade da velocidade de cada par de motores de maneira oposta, Figuras 3.2-g, 3.2-h e 3.3-d.

3.2 Softwares de Controle do Drone

Para demonstrar a eficiência do controle de drones e a possibilidade do planejamento e execução de voos autônomos fazendo uso de comunicação *Wi-Fi*, foram utilizados alguns *frameworks* como meios de interface de comunicação para envio de comandos e recebimento de dados do drone.

3.2.1 Node.Js

O *framework* Node.js [29] é uma plataforma desenvolvida sobre a *engine JavaScript V8*, orientada a eventos, desenvolvida pelo *Google*^{TM} para seu navegador *WEB Google Chrome*. Além de propiciar um ambiente de execução do lado do servidor que executa código em *JavaScript* com extrema rapidez [51], diversas bibliotecas como *ar-drone e ardrone-autonomy* têm sido desenvolvidas para permitir o controle de drones controlados por *Wi-Fi*. O *Node.js*

é um projeto *open source*, podendo ser executado em sistemas *Mac OSX, Windows ou Linux*.

A biblioteca *ardrone-autonomy* [28] é um *driver ROS* para quadricópteros Parrot AR-Drone 1.0 e 2.0 baseado no kit de desenvolvimento de softwares (SDK) do AR.Drone. Algumas características úteis dessa biblioteca são listadas abaixo:

- Filtro de *Kalman Extendido*: Utilizado para melhorar a estimativa de estados e formas de detecção.
- Projeção de câmera: Utilizado para estimar a posição de objetos detectados pela câmera.
- Controlador PID: Utilizado para controlar a posição do drone de maneira autônoma.
- Preparador de Missões: Utilizado para preparar um plano ou tarefa de voo e executá-lo.

A Figura 3.4 ilustra o AR.Drone executando uma missão para avançar e retroceder em linha reta, controlador pelo *Node.js* em conjunto com a biblioteca *ardrone-autonomy*.

Figura 3.4: AR.Drone executando missão controlado por *Node.js*.

3.2.2 Yadrone

O *Yadrone* [71], é um *beta framework* de código aberto desenvolvido para controle do AR.Drone 2.0. Escrito em Java, sua proposta é voltada tipicamente para atividades de ensino e pesquisa. Algumas características do *Yadrone* são apresentadas a seguir.

- Suporte a leitura de sensores e configurações de vídeo do drone.
- Possui uma central de controle para recuperação de dados e o controle via teclado.
- Suporte a *Android* (sem opção de vídeo).
- *Framework* escrito 100% em linguagem Java.

Por ser facilmente extensível, o *Yadrone* permite a desenvolvimento de aplicações *desktop* ou móvel (*Android*) para controlar o drone, permitindo-se criar, dentre outros aspectos, cenários de missões autônomas. A Figura 3.5 apresenta uma das telas do aplicativo *Android* desenvolvido para o traçado de trajetórias e envio de comandos ao drone.

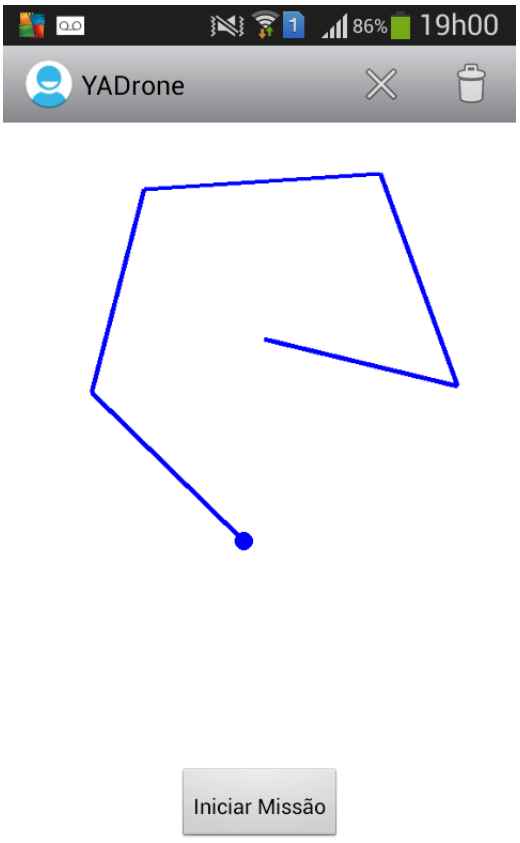


Figura 3.5: Aplicativo para desenhar trajetos com Yadrone.

3.2.3 AR.FreeFlight

O AR.FreeFlight [35] é um aplicação desenvolvida pela empresa *Parrot SA* para controle do *AR.Drone*. Com versões para *Mac OS, iPhone, Windows e Android*, é possível obter

total controle sobre o drone, além de recuperar em tempo real imagens de vídeo diretamente no *tablet* ou *smartphone*. Também é possível gravar vídeos e salvar imagens fotográficas no próprio aparelho por meio das inúmeras opções fornecidas em sua interface de controle, conforme ilustrado na Figura 3.6.

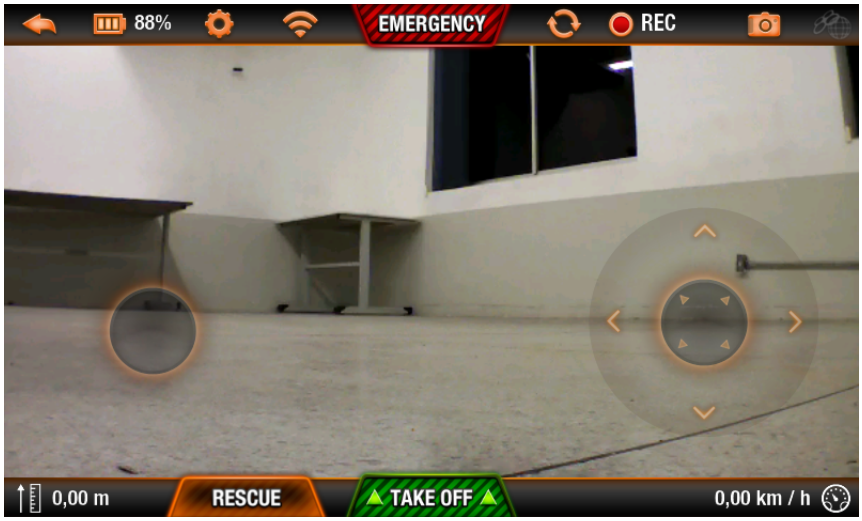


Figura 3.6: Acessando drone com Ar.FreeFlight.

O *Ar.FreeFlight* apresenta uma grande vantagem com relação a outros softwares controladores por ser desenvolvido pela *Parrot*, própria empresa fornecedora do *AR.Drone*, explorando dessa forma todos os recursos de operação e controle. É possível empregar vários meios de controle através do *tablet/smartphone*, como o controle absoluto (*absolut control*), que permite operar o drone sem a preocupação com orientação, uma vez que é utilizado o movimento do aparelho controlador para guiar o drone. Outro recurso importante é o piloto automático, que mantém o drone em modo *hover* (pairar) quando o usuário retira o dedo da tela do *smartphone* ou o aterrissa quando uma chamada telefônica é recebida.

3.3 *Framework* JMetal

Para implementação do algoritmo evolucionário NSGA-II foi utilizado o *framework* para otimização multiobjetivo *JMetal* (*Metaheuristic Algorithms in Java*) [25, 24]. Este *framework* é inteiramente desenvolvido em linguagem Java e conta com várias características que o tornam uma excelente opção para a avaliação de problemas multiobjetivos, como

representação de variáveis em distintos formatos (binário, real, inteiro). O JMetal é protegido sob a *GNU Lesser General Public License* e pode ser adquirido gratuitamente no site *http://jmetal.sourceforge.net*. A Figura 3.7 representa o diagrama de classes do *framework*.

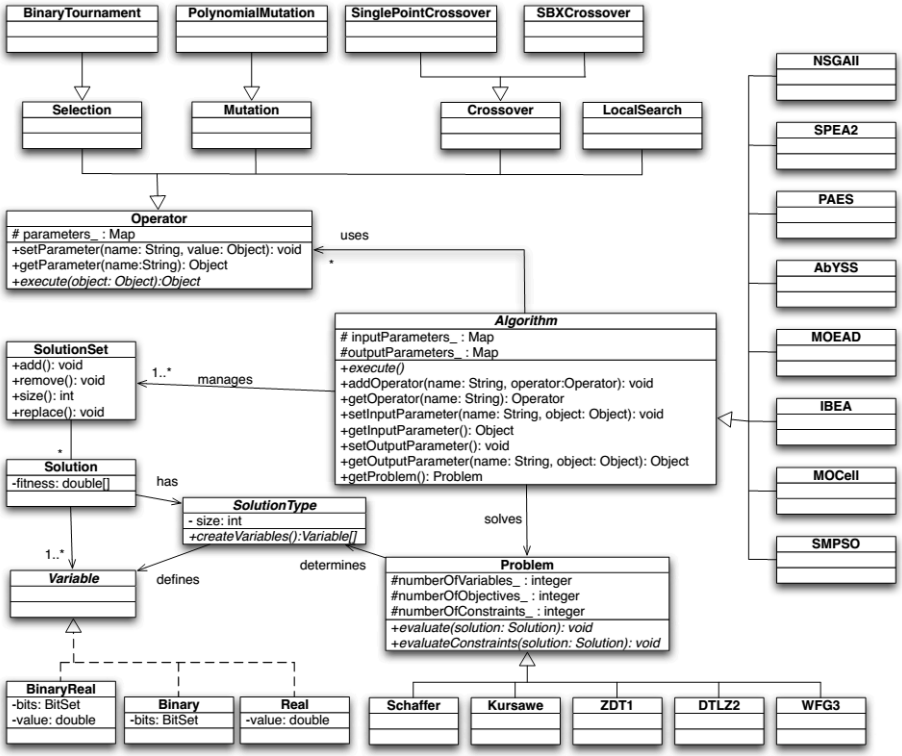


Figura 3.7: Diagrama de classes do JMetal.

Como descrito em sua documentação, a arquitetura básica do *JMetal* utiliza um algoritmo (*Algorithm*) para resolver um problema (*Problem*) usando um ou mais conjuntos de soluções (*SolutionSet*) e um conjunto de objetos representando seus operadores (*Operator*), como *crossover*, mutação, seleção e o operador de busca local.

Capítulo 4

Otimização Evolucionária de Trajetórias

4.1 Planejamento de Trajetórias

Nos últimos 20 anos, problemas envolvendo otimização evolucionária multiobjetivo (*multi-objective optimization - EMO)* tornaram-se bastante atrativos em inúmeras áreas. Os algoritmos evolucionários fazem uso dos meios associados ao conceito da evolução das espécies por seleção natural dos indivíduos mais aptos, onde mais de uma solução participa das rotinas de iteração do algoritmo e evolui para uma nova população de soluções.

Segundo [18], as razões para essa popularidade dizem respeito ao fato de otimização evolucionária não requerer informações derivativas, ser relativamente fácil de implementar e possuir grande flexibilidade e alcance em diversas aplicações. De um modo geral, o campo da robótica industrial vem apresentando crescentes avanços, com a utilização de máquinas inteligentes e autônomas, capazes de navegar em ambientes repletos de obstáculos e tomarem decisões. O planejamento de trajetórias para tais máquinas deve garantir seu livre percurso pelo ambiente, otimizando recursos de consumo como energia elétrica e evitando a colisão contra obstáculos.

Algoritmos para preparação e planejamento de movimentos têm sido amplamente estudados, como os trabalhos citados em [46, 34, 36, 45, 21, 13]. Grande parte desses trabalhos têm por objetivo encontrar os caminhos mais curtos até o destino, não levando em consideração

outros objetivos práticos que são importantes em aplicações de robótica móvel, como visibilidade, suavidade do percurso e segurança, cujo sentido não deve ser definido apenas como trajetos livres de colisão, mas também levando em consideração outros importantes efeitos do ambiente [2].

Ainda segundo [2], para a implementação bem sucedida de qualquer algoritmo de otimização multiobjetivo, o requisito mais importante é a representação significativa do caminho em termos de variáveis. Seguindo essa linha de abordagem, várias metodologias têm sido empregadas para problemas multiobjetivos e de objetivo único, como pode ser verificado em [67, 40, 22].

Em problemas práticos reais, normalmente existem pontos conflitantes que precisam ser otimizados para obtenção de possíveis soluções. Nesses casos, diversos objetivos devem ser considerados ao mesmo tempo. Problemas desse tipo são usualmente conhecidos como *Multiobjective Optimisation Problems (MOPs)* [12] e são formalmente definidos como

$$min[f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})] \tag{4.1}$$

onde $\vec{x} \in \Omega \subseteq \Re^n$ é chamado de vetor de variáveis de decisão. As soluções encontradas para esses tipos de problemas constituem um conjunto de soluções ditas soluções não-dominadas (*non-dominated solutions*). Uma dada solução x_1 domina outra solução x_2 se x_1 não é pior que x_2 em todos os objetivos e x_1 é melhor que x_2 em ao menos um objetivo. Em outras palavras, a relação de dominância, representada por $\prec$, pode ser definida como:

$$\begin{aligned} \vec{x_1} \prec \vec{x_2} \Leftrightarrow \forall i \in \{1, ..., k\}, f_i(\vec{x_1}) \leq f_i(\vec{x_2}) \\ e \qquad \exists i \in \{1, ..., k\}, f_i(\vec{x_1}) < f_i(\vec{x_2}) \end{aligned} \tag{4.2}$$

Empregando a relação de dominância de Pareto, os elementos mínimos são denominados *Pareto-Optimal*. O conjunto de vetores *Pareto-Optimal* é conhecido como conjunto ótimo de Pareto (*Pareto-Optimal set*). Ao serem plotados no espaço de objetivos (*objective space*), esses conjuntos definem a verdadeira frente de Pareto (*Pareto-Front* - PF_{true}). O objetivo então de algoritmos para solucionar problemas de otimização multiobjetivos (*Multiobjective*

Optimisation Problems - MOPs) é gerar uma frente de Pareto conhecida com boa aproximação da verdadeira frente de Pareto [32].

4.2 Método Proposto

Para o desenvolvimento de trajetos, foi considerado um percurso realizado pelo drone em um plano XY (2D), admitindo que ele se mantém em uma altura fixa, ou seja, durante o percurso não era exigido uma mudança de altura da aeronave. O desenho dos obstáculos ao longo do cenário foi realizado por meio da indicação de pontos, através de suas coordenadas, no plano cartesiano. Esses obstáculos são também considerados fixos, e não se leva em conta o surgimento de algum obstáculo novo durante o voo.

A configuração dos parâmetros usados no algoritmo NSGA-II através do *framework JMetal*, levou em consideração a resolução de um novo problema chamado *PathDrone*. A representação desse problema é obtida por meio da construção de uma classe que descreve todas as particularidades associadas ao problema que se deseja solucionar por meio do algoritmo. Para a representação dos dados, foi utilizado o formato de numeração real. Várias execuções do algoritmo foram realizadas alterando-se a quantidade de variáveis desejadas para a solução. Outros parâmetros usados são listados a seguir.

- O cenário para o voo é considerado como um plano cartesiano com limite inferior esquerdo no ponto (0,0) e limite superior direito no ponto (20,20).
- Os pontos que representam coordenadas de um trajeto têm limite inferior 2 e limite superior 15.
- O tamanho da população igual a 100.
- Número máximo de avaliações igual a 2500.
- Operador de Mutação Polinomial.

Os pontos de origem e destino do trajeto (missão) foram considerados fixos em relação à coordenada x com valores iguais a 1 para o ponto de origem, e 17 para o ponto que representa

o destino do movimento. Por essa razão, os limites para as coordenadas calculadas pelo algoritmo deveriam se manter entre 2 e 15.

O operador de mutação polinomial é uma derivação mais especializada para ser empregado em algoritmos genéticos voltados à resolução de problemas de otimização multiobjetivos [32], favorecendo a aproximação dos descendentes dos pais. Esse operador foi introduzido em [1] e posteriormente melhorado em [20].

Considera-se nesse trabalho a minimização de duas funções objetivos com igual importância. Para definição da primeira função objetivo, admite-se que o veículo aéreo não tripulado possui um conjunto limitado de recursos que precisam ser poupados, como por exemplo, sua carga de bateria. É assumido que o drone, durante seu trajeto, não é abastecido ou recarregado. Dessa forma, conhecidos os pontos de origem e destino, é esperado que o percurso desenvolvido pelo veículo seja próximo da distância mínima entre os dois pontos.

Dessa forma, a primeira função objetivo f_1 a ser considera, leva em conta a distância total percorrida pelo drone, a qual deve ser tão próxima da mínima quanto possível. Tal distância é obtida pelo soma das distâncias entre pontos discretos sucessivos dos vetores de decisão encontrados. Outro objetivo de igual relevância, está associado à segurança com que a missão é executada, ou seja, deseja-se evitar a colisão do veículo não tripulado com quaisquer pontos pertencentes ao mapeamento de obstáculos impostos ao cenário da missão. Para isto, foi definida uma nova função f_2 que representa o risco de colisão do drone, como o inverso (uma vez que se deseja minimizar a função) das somas das distâncias de cada ponto do obstáculo à reta suporte que conecta dois pontos consecutivos da trajetória proposta como solução.

Assim, a minimização do risco de uma colisão corresponde à uma maximização das distâncias entre o drone e os obstáculos, equações (4.3) e (4.4). A Figura 4.1, destaca as distâncias consideradas para as funções descritas. Os pontos circulares representam obstáculos a serem evitados e os pontos quadriculares representam soluções propostas para a trajetória a ser percorrida.

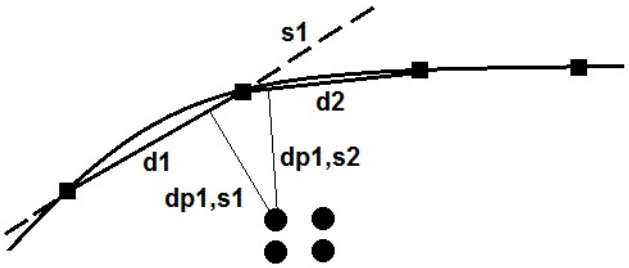


Figura 4.1: Distâncias consideradas para minimizar percurso e risco de colisão.

$$f_1 = \sum_{i=1}^{N-1} d_i = \sum_{i=1}^{N-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{4.3}$$

$$f_2 = \left(\sum_{k=1}^{M} \sum_{j=1}^{N-1} d_{Pk,sj} \right)^{-1} = \left(\sum_{k=1}^{M} \sum_{j=1}^{N-1} \frac{|a_j x_k + b_j y_k + c_j|}{\sqrt{a_j^2 + b_j^2}} \right)^{-1} \tag{4.4}$$

Na equação (4.3), d_i representa a i-ésima distância entre os N pontos da trajetória apresentados como solução. A equação (4.4) emprega a distância de uma reta, descrita em sua forma geral como $ax + by + c = 0$, a um dado ponto do espaço de obstáculos. Nesta equação, a_j, b_j e c_j representam os coeficientes da reta suporte que une dois pontos consecutivos da trajetória. As coordenadas (x_k, y_k) representam os valores do k-ésimo ponto que compõe o conjunto de M obstáculos e $d_{Pk,sj}$ corresponde à distância entre o ponto obstáculo P_k e a reta s_j, como pode ser verificado através da Figura 4.1.

Pode-se assumir qualquer disposição para os pontos que representam obstáculos. Foram escolhidos para apresentação nesse trabalho duas disposições desses pontos, representando os conjuntos de obstáculos 1 e 2.

A Tabela 4.1 apresenta algumas soluções encontradas para as funções objetivos descritas. Como pode ser observado, o conjunto de soluções retrata uma ampla variação em relação aos objetivos conflitantes, deixando claro a condição de desproporcionalidade entre eles. Foram apontadas soluções que claramente refletem a otimização de um objetivo em detrimento ao

outro, ou seja, existem soluções que representam trajetórias não tão curtas, mas com risco de colisão bastante reduzido. Por outro lado, algumas soluções são bem mais arriscadas (voos passando próximos aos obstáculos), no entanto, descrevem percursos rápidos.

As linhas destacadas na Tabela 4.1 foram obtidas por meio das coordenadas (variáveis) de duas das soluções calculadas.

Tabela 4.1: Soluções para as funções objetivos com base no primeiro conjunto de obstáculos assumindo-se 4 pontos (8 variáveis).

f1	**f2**
30.36205938609736	0.9284017496586583
33.481548748921696	**0.7631308298597251**
48.401812735693554	0.5136858250606751
29.304740197878907	1.1590328607850897
32.42559693563289	0.825153976182511
29.42690238440023	1.110309601417625
39.630231669472636	0.6406470928222838
33.160438529557545	0.8050193261141501
30.590767797245917	0.8995305358853349
50.157950185539	0.4848487742822668
39.541734717918054	0.644744374861641
46.875852882187225	0.5970481930441053
17.47791720516956	6.472524270216029
25.09694410925024	**1.3718289936432027**
43.33880741002497	0.6187149582875555
18.842286295960946	4.223003840227774
24.100945681398706	1.5818457996147544
62.95956410870232	0.42002473298316995
30.36205938609736	0.9284017496586583
16.714710707617566	8.888648641699218

A Figura 4.2 representa o desenho das trajetórias para as coordenadas das duas soluções

realçadas na Tabela 4.1, considerando-se um primeiro padrão de pontos representando obstáculos. Para este conjunto de soluções foram utilizadas 8 (oito) variáveis ou coordenadas, o que corresponde a 4 (quatro) pontos, com tempo total de execução computacional médio de 140ms e interpolação por *splines* cúbicas. Os pontos representando a origem e o destino são fixos e não integram o conjunto de soluções encontrado pelo algoritmo, porém, esses pontos são também considerados nos cálculos para a composição completa das funções objetivos.

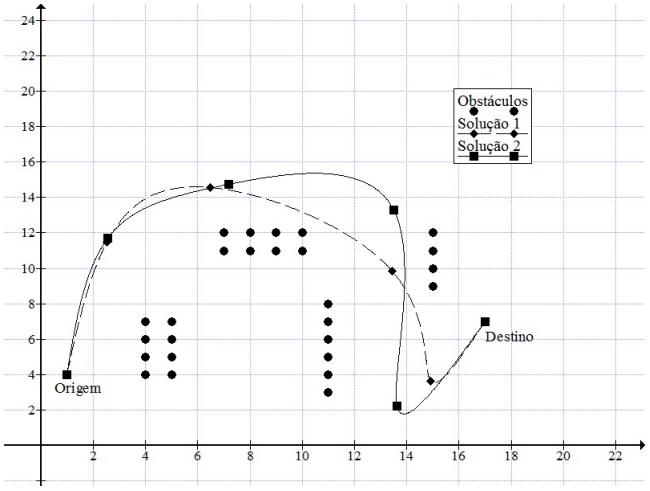


Figura 4.2: Dois possíveis trajetos obtidos da frente de soluções para o conjunto 1 de obstáculos (8 variáveis).

Marcações circulares na figura representam pontos de obstáculos que devem der evitados durante o trajeto. Outras marcações representam pontos das soluções encontradas para o percurso. A distribuição das soluções encontradas podem ser verificadas conforme ilustrado na Figura 4.3.

Para a representação gráfica dos pontos representativos de obstáculos e coordenadas de soluções de trajetos, foi utilizado o software *open source Graph* [37], desenvolvido por Ivan Johansen[1] para o traçado de funções matemáticas em um sistema de coordenadas. Além de permitir a indicação de uma família de pontos para sua representação no plano, o *Graph* oferece algumas opções de interpolação como a linear ou *splines* cúbicas.

A Figura 4.3 ilustra o relacionamento entre os valores das funções objetivos para um dado conjunto de soluções encontrado pelo algoritmo evolucionário. Esse conjunto compõe uma

[1]http://publisher.brothersoft.com/ivan-johansen.html

frente de soluções próxima à frente verdadeira de Pareto.

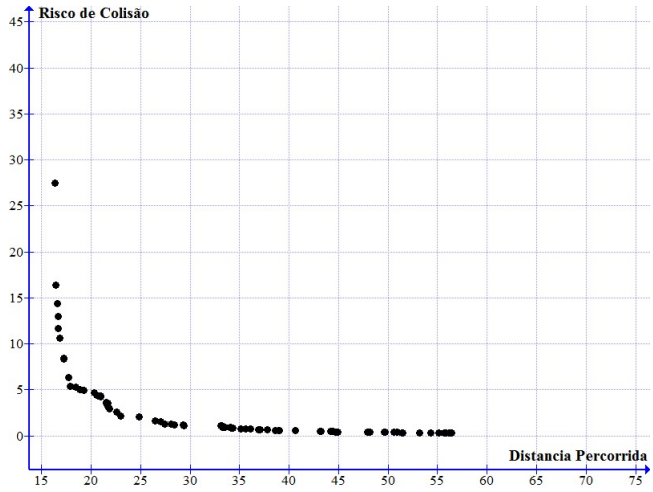


Figura 4.3: Frente de soluções obtidas para o conjunto 1 de obstáculos com 8 variáveis.

Da Figura 4.3, percebe-se a forma convexa da frente de Pareto, fornecendo um conjunto bem espalhado de possíveis soluções que buscam a negociação conjunta de uma trajetória relativamente curta para o voo do drone e com o risco de colisão com algum ponto do conjunto de obstáculos também reduzido.

Assumindo agora um conjunto maior de variáveis para compor o conjunto de soluções para a primeira organização de obstáculos, nota-se a variação de trajetórias ao longo do obstáculos. A Figura 4.4 ilustra duas soluções factíveis para um total de 10 (dez) variáveis, ou seja, 5 (cinco) pontos.

A Tabela 4.2 exibe um novo conjunto de soluções para o primeiro padrão de obstáculos, assumindo-se cinco pontos para a composição da trajetória. Vale salientar que os pontos de origem e destino não fazem parte da solução encontrada, mas participam dos cálculos a cada iteração do algoritmo que avaliam as funções objetivos.

Tabela 4.2: Soluções para as funções objetivos com base no primeiro conjunto de obstáculos assumindo-se 5 pontos (10 variáveis).

f1	**f2**
36.692455360646726	1.513708588332979
23.701800116689064	4.1837759019957685
36.692455360646726	1.513708588332979
23.046761400590746	4.643636738121742
19.981946442159575	7.618817151604425
21.749943915693894	**7.14661477473291**
53.65872626096549	0.6870713233582615
18.06942752132347	**11.708622930321116**
33.32741810468355	1.5856153891140063
70.84963182603056	0.4967462018603824
18.050264233265555	12.034196874345575
23.79907705458881	4.149412786672304
29.437723384773015	2.870210584474992
28.24595611322008	2.947835654339937
77.79988167996173	0.431408475396118
53.407949411728445	0.7352900308442815
36.98675445869358	1.5104772565237257
59.87323570481409	0.6134702464175344
78.8261813712293	0.4167846489286293
77.79988167996173	0.431408475396118

As duas linhas destacadas da Tabela 4.2 representam soluções provenientes das coordenadas de duas possíveis trajetórias e são ilustradas no gráfico da Figura 4.4.

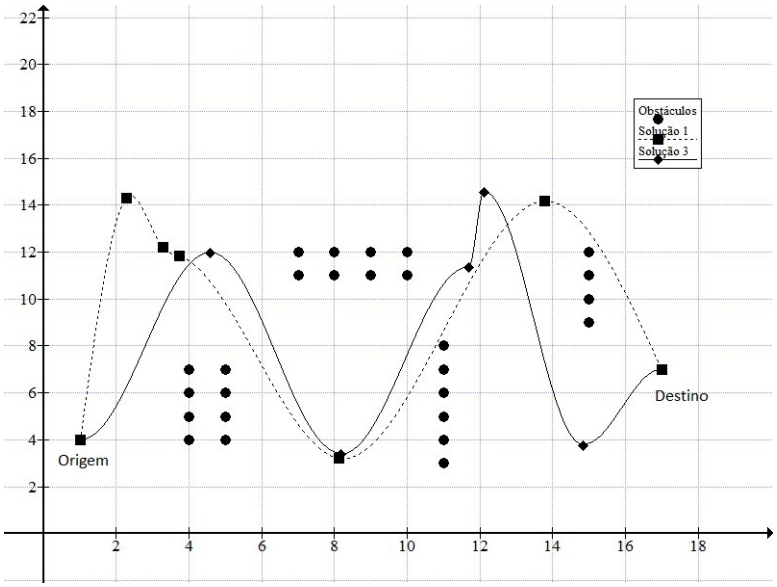


Figura 4.4: Duas soluções factíveis da frente de soluções para o conjunto 1 de obstáculos (10 variáveis).

A Figura 4.5 apresenta a frente de Pareto obtida com a indicação de 10 variáveis.

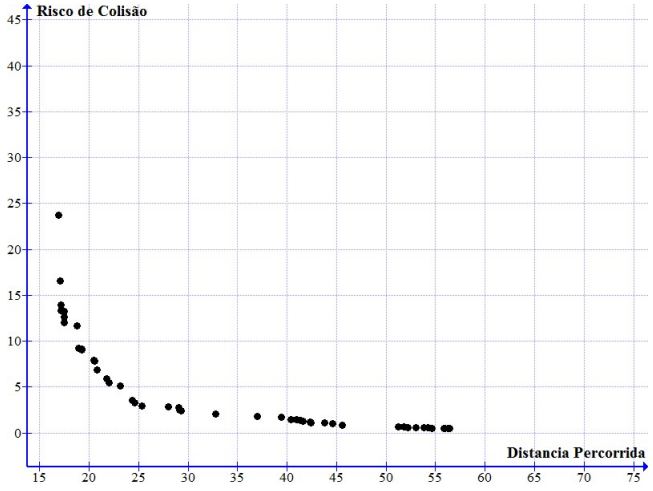


Figura 4.5: Frente de soluções obtidas para o conjunto 1 de obstáculos com 10 variáveis.

Levando-se em consideração um outro padrão para a disposição dos obstáculos (conjunto 2 de obstáculos), diversas execuções foram processadas com a indicação de soluções factíveis bem distribuídas. Essas soluções são apresentadas na Tabela 4.3.

Tabela 4.3: Soluções para as funções objetivos com base no segundo conjunto de obstáculos assumindo-se 5 pontos (10 variáveis).

f1	**f2**
47.71867606714276	0.6078371575816043
61.63999464011124	0.41598633877581154
28.4038660126391	2.7000705214483776
23.066259043935617	3.981037393166835
38.8334189937134	1.1822467346508523
16.593926361283224	**15.10777844872722**
16.971477554697337	12.081814250072524
43.02624643682548	0.8972366502123477
16.575838357874602	16.345870027768353
27.774552979966067	2.9071439505188894
40.60444062771266	0.9349385826443094
42.57068342455618	0.9272264846665739
48.02984585765057	0.5535253657086906
18.42379382968615	**4.9514417689370065**
43.186800453937	0.8536398116377326
40.36083870937064	0.9981279635869137
57.259247638692045	0.46307522496710896
64.25402643330894	0.37847814714005806
33.89889742089827	1.3762592676779284
40.51552197234061	0.9408584599164528

A Figura 4.6 apresenta duas soluções obtidas (linhas destacadas da Tabela 4.3), assumindo-se um total de 10 variáveis (5 pontos) para o vetor de decisão. Os pontos que representam os obstáculos para esse novo padrão, foram propositadamente mantidos afastados de tal forma a permitir um trajeto mais direto entre destino e origem. O objetivo dessa formação e constatar a eficiência e eficácia das soluções apontadas pelo algoritmo. Como pode ser verificado da Figura 4.6, de fato, as trajetórias representam percursos rápidos (sem variações e curvas excessivas) e seguros (distantes de obstáculos).

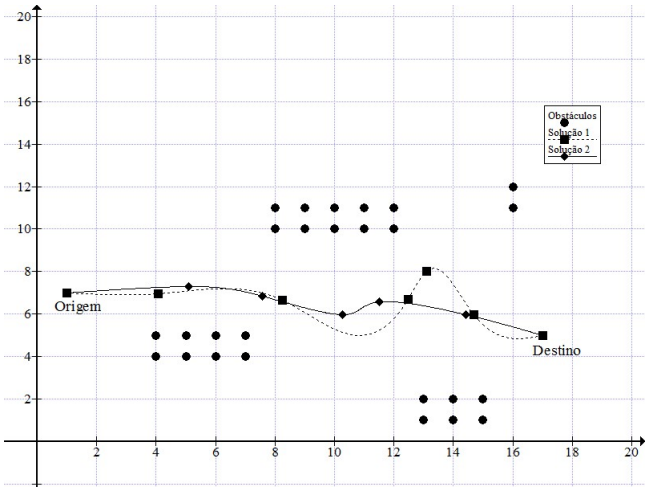


Figura 4.6: Indicação de duas soluções para o conjunto 2 de obstáculos e vetor de decisão com 10 variáveis.

Para esta segunda organização de pontos obstáculos, é assumido o ponto (1,7) como origem do trajeto e o ponto (17,5) como destino. A frente de Pareto para esta disposição de obstáculos é representada na Figura 4.7.

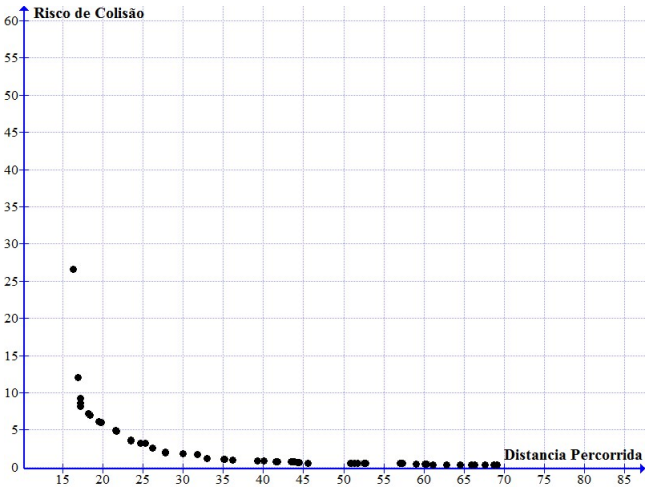


Figura 4.7: Frente de Pareto para a disposição 2 de obstáculos.

Vale ressaltar que a interpolação por *splines* não leva em consideração uma curvatura de raio mínimo que é exigida para o drone se mover suavemente em curva. Dessa maneira e, segundo a indicação de um valor para o raio mínimo que garanta a realização dessas curvas, a aproximação por *splines* pode sofrer variações [56].

Capítulo 5

Resultados e Discussão

5.1 Controlabilidade por *Frameworks*

A experimentação de soluções em software para atuarem como meios de controle para drones quadrirotores de pequeno porte, como o Parrot AR.Drone 2.0, foi realizada mediante filmagens de diferentes situações submetidas ao drone, e posteriormente, uma análise da eficiência e precisão com que o drone obedece comandos recebidos pelas diferentes plataformas.

5.1.1 Preparação do ambiente

Para a realização das filmagens de voos e testes com o drone, utilizou-se uma sala vazia, cujo piso se encontrava previamente delimitado por marcações quadriculares, conforme pode ser observado nas figuras listadas no Apêndice B (Figura B.1). As filmagens foram realizadas com auxílio de um *notebook*, por meio de sua *webcam* integrada. Para manter as distâncias com que as filmagens seriam feitas, padronizou-se um posicionamento fixo para o computador através de marcações no piso e na bancada onde ele deveria permanecer, Figura B.2.

Uma vez obtidas as imagens e vídeos apresentando o desempenho do drone em algumas tarefas preestabelecidas, utilizou-se o software *open source ImageJ* [70], desenvolvido por um time de desenvolvedores ao redor do mundo para a comunidade científica como uma alternativa *open source* para o processamento de imagens.

A configuração padrão do estado do AR.Drone para a realização das tarefas (missões) foi estabelecida e mantida de acordo com a listagem a seguir:

- Flip desativado.
- Limite de altitude: 3 m.
- Velocidade vertical máxima: 497 mm/s.
- Velocidade máxima de rotação: 69 graus/s.
- Ângulo máximo de inclinação: 10 graus.
- Casco exterior desligado.
- Modo joypad e controle absoluto desligados.
- Modo canhoto desligado.

5.1.2 Detalhamento das missões

O drone foi submetido a determinadas tarefas ou missões, a partir dais quais foram feitas medições acerca do movimento executado e distância percorrida. A seguir são apresentadas capturas de imagens provenientes dos diversos vídeos gravados, destacando a controlabilidade obtida por meio do uso de diferentes *frameworks*. Tais *frameworks*, uma vez conectados ao drone, foram configurados para enviarem-lhe comandos de movimentação em todas as direções ou comandos de missões completas.

A sequência a seguir ilustra os movimentos executados pelo drone sob comando dos aplicativos *AR.FreeFlight, Node.js e Yadrone*, com estes dois últimos *frameworks* servindo como plataformas base para o desenvolvimento de um novo sistema de navegabilidade autônoma, proposto neste trabalho. O objetivo do drone correspondente à primeira missão, limita-se à execução de um voo em linha reta ao longo de três quadros delimitados no piso. Tais quadros totalizam um distância total de aproximadamente 337 cm, como pode ser verificado no Apêndice B, Figura B.3.

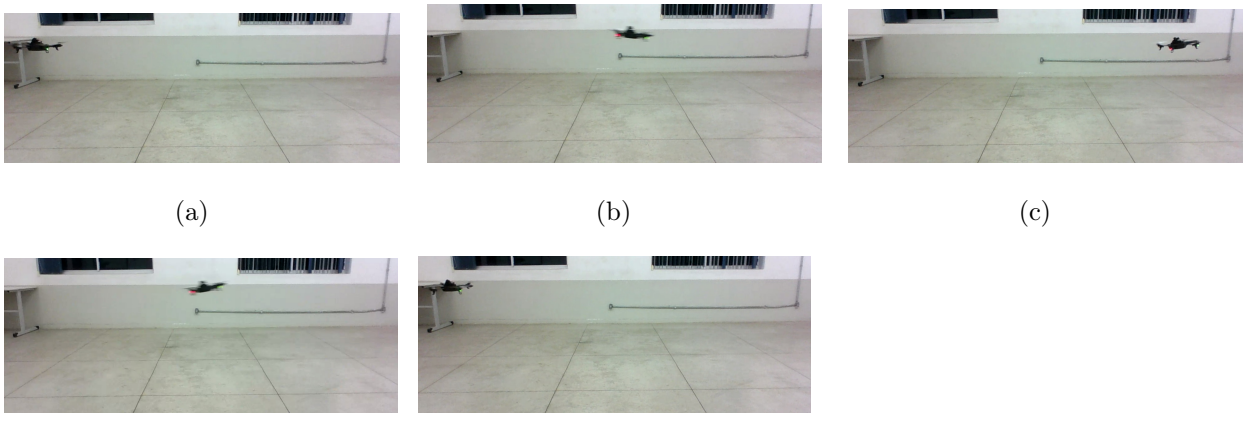
O que se deseja comprovar a partir das missões impostas ao drone, é a precisão da execução de seus movimentos, a eficiência da troca de informações com a aplicação, a realização com sucesso da missão do início ao fim, para que se possa constatar quão seguro e controlável é o drone para ser submetido a voos autônomos.

(a) (b) (c)

Figura 5.1: Drone executando trajeto de linha.

Na Figura 5.1(a), o drone inicia sua tarefa partindo de um estado estável, no qual permanece pairando. Em seguida, a execução do percurso é realizada com sucesso, sendo finalizada com o drone em um estado novamente estável é pronto para uma nova operação, Figuras 5.1(b) e 5.1(c).

A segunda missão que deverá ser desenvolvida pelo drone, tem por objetivo o movimento de ida e volta, ao ponto de partida, por meio de comandos para frente (*forward*) e para trás (*backward*). A sequência a seguir exibe detalhes da tarefa executada.



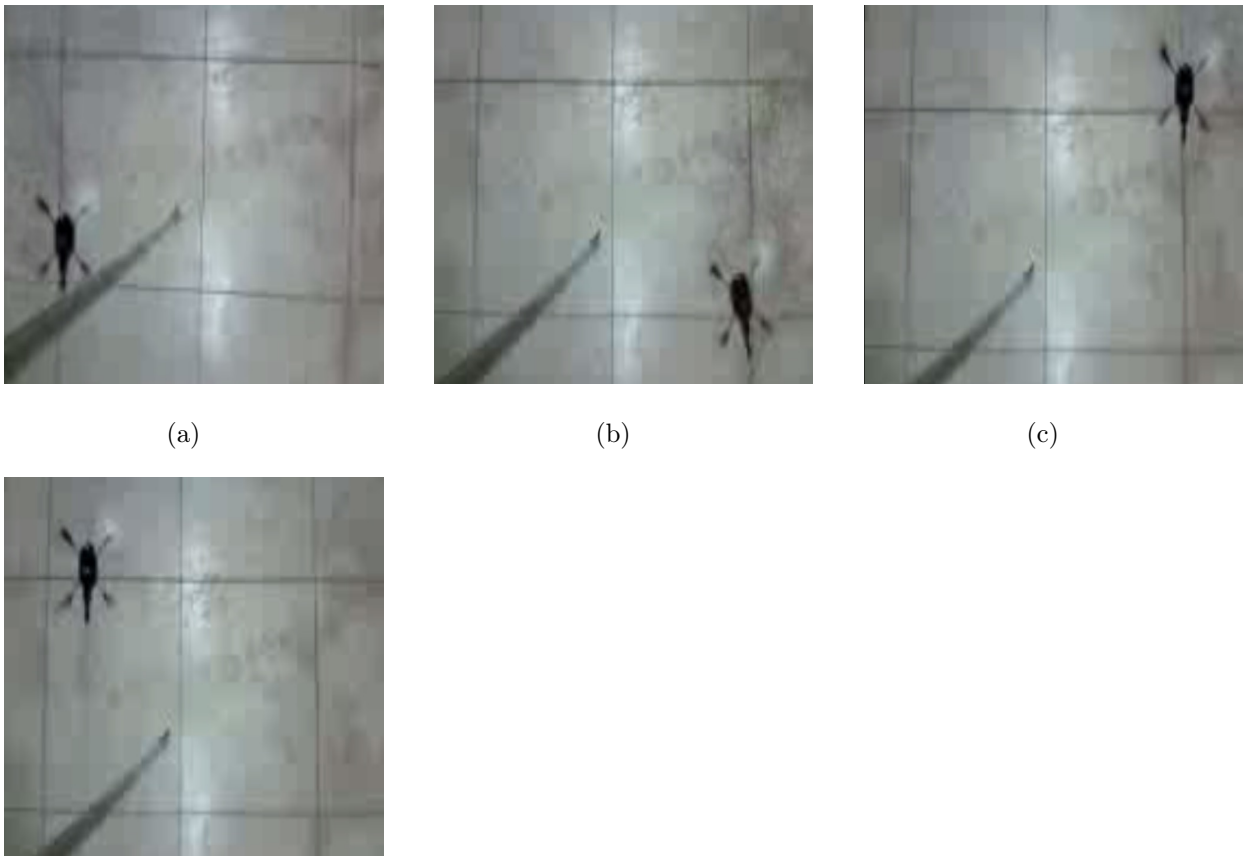
(a) (b) (c)

(d) (e)

Figura 5.2: Missão de ida e volta em linha reta.

A terceira missão submetida ao drone, é a realização de um voo retilíneo a uma mesma altura do solo, de quatro movimentos horizontais na forma de um retângulo. Nenhum comando de

rotação ao redor do eixo vertical é passado ao drone, apenas ações para ele se deslocar para frente, para a direita, para trás e para a esquerda, Figuras 5.3(a), 5.3(b), 5.3(c) e 5.3(d), respectivamente.



(a) (b) (c)

(d)

Figura 5.3: Drone em execução do movimento retangular.

A filmagem desses movimentos foram feitas mediante o posicionamento de um aparelho celular próximo ao teto da sala com auxílio de um bastão de *selfie* retrátil. Localizado em uma posição fixa próximo ao teto, o aparelho pôde enquadrar bem os dois principais retângulos demarcados no piso, cujas medidas são conhecidas, [57]. Como o drone se manteve sempre à mesma altura em relação ao piso (aproximadamente um metro), a presença do bastão não apresentou ameaças à validade e precisão dos movimentos e das medidas.

A quarta é última missão corresponde aos movimentos para execução de um retângulo a uma altura fixa do solo, porém, diferentemente da missão anterior, nesta o drone deverá realizar rotações ao redor do eixo vertical, de forma que os comandos de controle necessários

para a execução da missão se limite a movimentos para frente seguidos de rotações, onde a quantidade de graus a rotacionar é especificado.

5.1.3 Resumo de resultados obtidos

A verificação da eficácia das ações de controle passadas ao drone pelos *frameworks*, bem como a precisão dos movimentos realizados por ele, foram obtidos por meio da análise das gravações realizadas durante cada missão. A leitura das distâncias percorridas pelo drone entre seus movimentos, pode ser verificada por meio do software *ImageJ*, através da indicação de uma distância de referência conhecida, Figura 5.4. Como o piso da sala onde os testes foram feitos é totalmente delimitado por retângulos de medidas conhecidas, Figura 5.5, tornou-se possível, através da indicação destas distâncias ao *ImageJ*, a leitura dos comprimentos dos percursos de interesse, Figura 5.6.

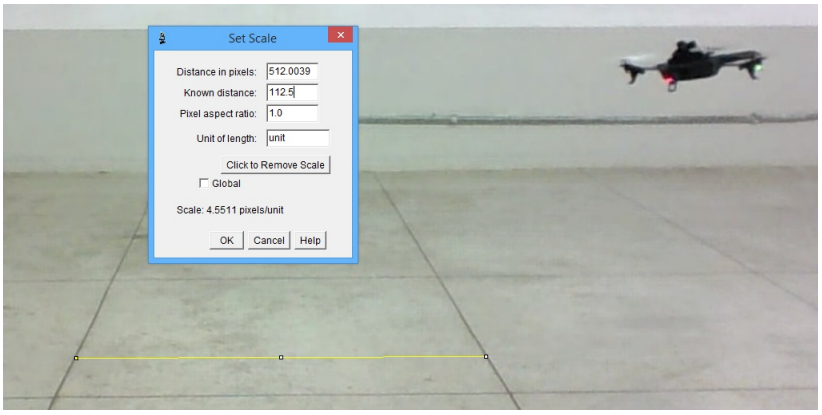


Figura 5.4: Configuração do ImageJ mediante a indicação de uma distância de referência conhecida.

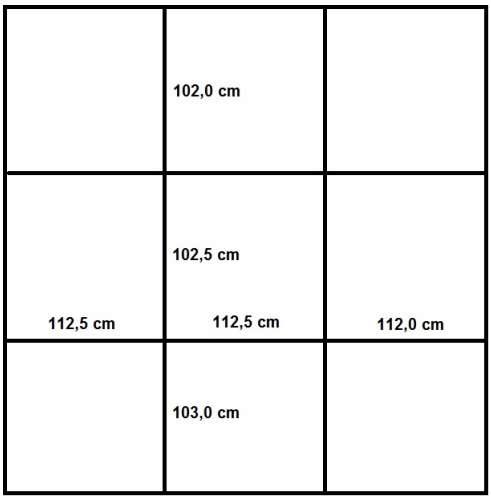


Figura 5.5: Medidas dos retângulos delimitas no piso da sala de testes.

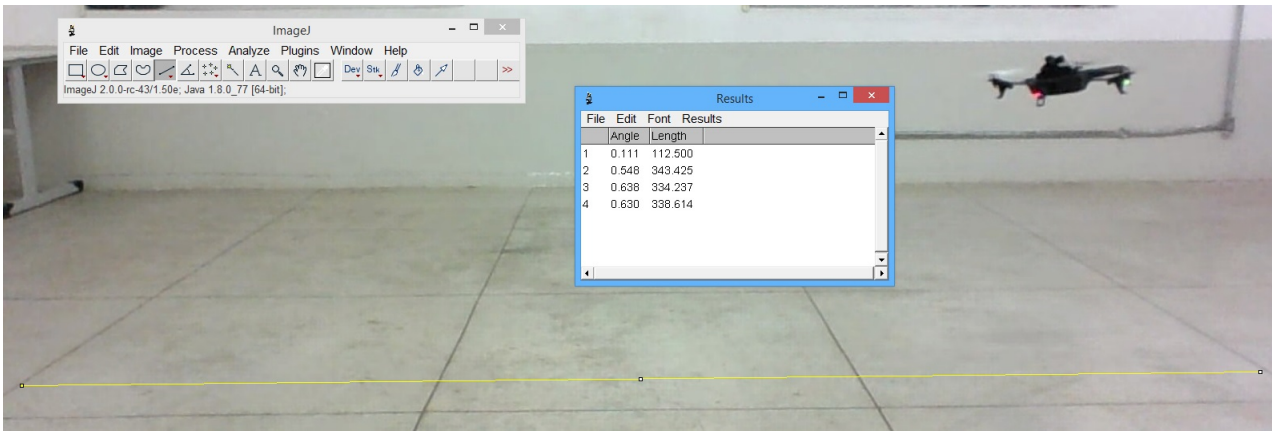


Figura 5.6: Leituras de percursos realizadas com o ImageJ.

As linhas amarelas nas Figuras 5.4 e 5.6, indicam o percurso que se deseja medir. A janela *Results*, registra os valores obtidos para os percursos indicados.

A Tabela 5.1 apresenta os resultados para quatro testes realizados para as missões de trajeto em linha considerando apenas a ida (primeira missão) e também o percurso de ida e volta (segunda missão). Vale ressaltar que cada uma das quatro missões foi assim executada quatro vezes, o que corresponde a quatro testes. Como pode ser observado da Figura 5.5, o movimento de ida tem um comprimento total aproximado de 337 cm (ao longo dos retângulos de bases 112,5 cm, 112,5 cm e 112,0 cm) e o comprimento total de ida e volta, aproximadamente o dobro daquele valor, aproximadamente 674 cm.

Os valores de referência então empregados correspondem à 337 cm para a primeira missão e 674 cm para a segunda missão.

Tabela 5.1: Resultados para as missões de linha empregando os *frameworks Node.js e Yadrone* como plataformas de desenvolvimento (valores em centímetros).

	Teste 01	**Teste 02**	**Teste 03**	**Teste 04**
Linha (Node.js)	340,777	334,693	334,255	332,608
Linha (Yadrone)	332,327	333,796	335,035	339,212
Linha ida e volta (Node.js)	674,137	671,913	673,865	677,202
Linha ida e volta (Yadrone)	673,445	677,221	673,914	673,339

De acordo com a tabela A Tabela 5.1, a execução de missões de linhas mostrou-se bem sucedida para ambas as plataformas, uma vez que os valores encontrados por meio da análise das imagens com o programa *ImageJ* se mantiveram dentro de uma faixa razoável de tolerância considerada inferior a 5%. Vale ressaltar que, durante os trajetos, o drone não recebeu quaisquer comandos para a alteração de sua altura, apenas de movimentação para frente e para trás, conforme funções de controle disponibilizadas em cada plataforma.

A verificação da conformidade com relação à missão de executar uma trajetória retangular (terceira missão), levou em consideração o percurso total desenvolvido pelo drone e sua permanência sobre a linha demarcada em solo, durante seu trajeto. Através de medições realizadas por meio do *ImageJ*, verificou-se quanto o drone afastou-se do seu percurso. As Tabelas 5.2 e 5.3, exibem os resultados obtidos para quatro testes executados para a terceira missão.

Os movimentos impostos ao drone são representados pelas iniciais *Fre* (movimento para frente), *Dir* (movimento para direita), *Tra* (movimento para trás) e *Esq* (movimento para esquerda). Para cada um desses deslocamentos, foram medidos o comprimento do percurso e quanto o drone se afastou da marcação representada pela linha no solo, com as medidas em centímetros. Para a execução desta missão, foram considerados dois retângulos centrais, compondo as seguintes medidas de referência, conforme ilustra a Figura 5.5: frente (112,5 cm), direita (103,0 + 102,5 = 205,5 cm), trás (112,5 cm) e esquerda (102,5 + 103,0 = 205,5 cm).

Os valores lidos dos afastamentos do drone durante seus movimentos, com relação à linha indicativa no solo, são medidas aproximadas e arredondadas para um valor inteiro mais próximo, também em centímetros. Mais uma vez, importante salientar que cada missão é executada quatro vezes, o que corresponde a quatro testes.

Tabela 5.2: Resultados para a missão de trajeto retangular empregando os *frameworks Node.js e Yadrone* (valores em centímetros). Testes 01 e 02.

	Teste 01				**Teste 02**			
	Fre	**Dir**	**Tra**	**Esq**	**Fre**	**Dir**	**Tra**	**Esq**
Node.js	$111,4 \pm 9$	$206,8 \pm 8$	$110,1 \pm 7$	$204,9 \pm 6$	$109,9 \pm 8$	$207,4 \pm 5$	$110,5 \pm 5$	$208,1 \pm 6$
Yadrone	$114,8 \pm 6$	$210,6 \pm 8$	$113,4 \pm 5$	$210,9 \pm 8$	$112,2 \pm 7$	$206,6 \pm 9$	$111,4 \pm 8$	$204,8 \pm 8$

Tabela 5.3: Resultados para a missão de trajeto retangular empregando os *frameworks Node.js e Yadrone* (valores em centímetros). Testes 03 e 04.

	Teste 03				**Teste 04**			
	Fre	**Dir**	**Tra**	**Esq**	**Fre**	**Dir**	**Tra**	**Esq**
Node.js	$109,7 \pm 5$	$204,6 \pm 5$	$113,1 \pm 6$	$208,2 \pm 6$	$114,2 \pm 8$	$212,7 \pm 9$	$101,3 \pm 8$	$210,8 \pm 8$
Yadrone	$118,8 \pm 4$	$207,1 \pm 5$	$112,2 \pm 6$	$210,5 \pm 6$	$111,7 \pm 6$	$206,6 \pm 5$	$108,3 \pm 6$	$205,3 \pm 6$

Os resultados apresentados nas Tabelas 5.2 e 5.3 destacam que, para os testes realizados, a trajetória retangular foi bem desenvolvida, no entanto, os movimentos foram acompanhados por um leve afastamento do trajeto. Algumas das razões para isto estão associadas a possíveis alterações nas velocidades dos motores, o que afetaria as operações de rolagem (movimentos para esquerda e direita) e arfagem (movimentos para frente e para trás); também pequenos danos no conjunto motores e hélices do drone, devido a choques sofridos, o que poderia ao longo do tempo e de forma cumulativa, *afrouxar* seus contatos.

Apesar desses afastamentos, o erro de alguns centímetros ao longo de um percurso de um ou dois metros é condizente com o fato do não uso de meios de sensoriamento como GPS. Dessa forma, as plataformas estudadas, *Node.js e Yadrone*, que serviram de base para o mecanismo de controlabilidade do drone utilizado, apresentaram desempenho dentro dos limites esperados, mantendo o drone sob controle durante toda a missão.

A quarta e última missão é similar à anterior no que se refere ao trajeto retangular, porém

aqui o drone executará apenas movimentos para frente e movimentos de rotação de 90 graus ao redor do eixo vertical. O mesmo procedimento de medidas empregados na missão anterior são utilizados para esta tarefa. As Tabelas 5.4 e 5.5, apresentam os resultados obtidos, onde **Nod** e **Yad** se referem respectivamente aos *frameworks Node.js e Yadrone*.

Tabela 5.4: Resultados para a missão de trajeto retangular com movimentos para frente seguidos de rotação (valores em centímetros). Testes 01 e 02.

	Teste 01				**Teste 02**			
	Fre	**Dir**	**Tra**	**Esq**	**Fre**	**Dir**	**Tra**	**Esq**
Nod	$114,3 \pm 6$	$211,5 \pm 17$	$104,8 \pm 12$	$201,1 \pm 9$	$103,2 \pm 15$	$204,4 \pm 8$	$116,1 \pm 9$	$200,3 \pm 8$
Yad	$112,9 \pm 9$	$205,7 \pm 9$	$117,1 \pm 7$	$199,1 \pm 6$	$106,8 \pm 7$	$201,2 \pm 6$	$103,3 \pm 8$	$203,5 \pm 8$

Tabela 5.5: Resultados para a missão de trajeto retangular com movimentos para frente seguidos de rotação (valores em centímetros). Testes 03 e 04.

	Teste 03				**Teste 04**			
	Fre	**Dir**	**Tra**	**Esq**	**Fre**	**Dir**	**Tra**	**Esq**
Nod	$116,2 \pm 6$	$202,9 \pm 25$	$101,1 \pm 9$	$219,0 \pm 16$	$113,2 \pm 18$	$209,4 \pm 8$	$115,7 \pm 11$	$219,6 \pm 8$
Yad	$111,1 \pm 8$	$206,4 \pm 8$	$109,5 \pm 8$	$214,3 \pm 10$	$120,0 \pm 10$	$215,4 \pm 8$	$109,3 \pm 8$	$200,1 \pm 9$

Verifica-se dos resultados apresentados nas Tabelas 5.4 e 5.5, uma melhor precisão de movimentos ao se utilizar o *Yadrone* como plataforma comparado ao *Node.js*. Deve-se no entanto ser observado que, o desempenho associado a cada plataforma está relacionado à forma como os comandos são processados pelo próprio drone e como este executará o movimento seguinte, uma vez que ao término de cada movimento, o drone empregará meios próprios para se manter estável e a uma altura fixa do solo. Foi percebido que, nos testes realizados utilizando-se o *Node.js* como plataforma de controle, o drone perdeu bastante tempo entre cada movimento, pois, ao término de cada um desses movimentos, o veículo procurava se manter em estado *hover* (pairado) e, nesse momento, flutuações de movimentos ocorriam, retirando o drone do seu percurso ou vagarosamente rotacionando-o de forma não esperada.

Outro problema notado durante os últimos testes, foi a forma irregular com que o AR.Drone executava o comando de decolagem *takeoff*, passando a realizá-lo de forma inclinada para trás em relação à linha do solo, ao invés de fazê-lo de forma vertical. Sabe-se que o movimento para trás do drone se dá por meio da ação de arfagem, ou seja, um aumento na intensidade

de rotação do motor frontal acompanhado de uma diminuição da rotação do motor traseiro. No entanto, o comando de decolagem deve mover o drone na direção vertical para cima, o que é realizado por meio de uma aceleração gradual e de igual intensidade em seus quatro rotores. Se, durante a decolagem, o drone se move para cima e para trás, como observado nos últimos testes, confirma-se uma queda de desempenho do motor traseiro, proveniente de falha mecânica ou estado danificado de sua hélice. A Figura 5.7 destaca a hélice do motor traseiro do drone danificada, o que resultou posteriormente em sua perda completa. Na Figura 5.7, as setas destacam as regiões de envergamento excessivo.

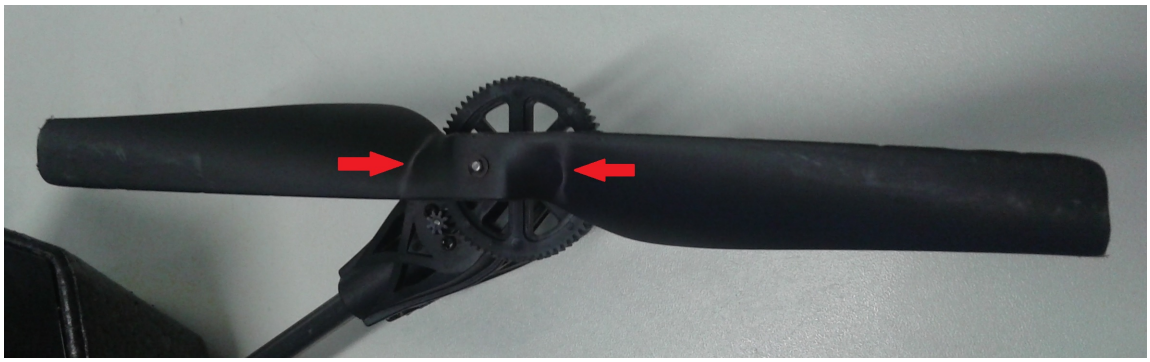


Figura 5.7: Hélice traseira do AR.Drone danificada.

Capítulo 6

Navegabilidade Autônoma - Estudo de Caso

6.1 Uso de Drones como Guias Aéreos

Como possibilidade de aplicação do planejamento de trajetórias para veículos aéreos não tripulados de pequeno porte, apresenta-se nesta seção um estudo das implicações de se utilizar um drone como guia aéreo turístico em ambientes públicos. Uma vez representado o cenário de circulação do drone por meio da indicação de obstáculos fixos e dos pontos de origem e chegada, deseja-se saber se problemas como espera em demasia ou formação de filas podem surgir com base na evolução do *tour*.

Uma situação genérica ou habitual de pessoas que chegam e se deslocam em um dado ambiente é sugerida e o modelo que descreve e simula o comportamento dinâmico desses indivíduos em movimentação guiados por um drone é apresentado com auxílio do software Arena, um programa de simulação de eventos discretos bastante utilizado por especialistas e profissionais que desejam efetuar a análise profunda de cenários e simulações de processos. O Apêndice A destaca a representação de cada etapa da modelagem realizada.

O software Arena [64] possui recursos para análise estatística, modelagem de processos, animação, análise de resultados, etc. Através da análise dinâmica e da interação entre os

elementos do sistema, é possível determinar gargalos, melhores condições de operação, visualizar tamanhos de filas, ocupação de recursos e verificar qual é o comportamento conjunto do sistema. A seguir são descritas cada etapa do processo de modelagem e simulação desenvolvidas.

6.1.1 Chegada de entidades (visitantes)

As entidades (visitantes) chegam ao ambiente segundo uma distribuição de probabilidade exponencial com média inicialmente considerada igual a 10 minutos. Nesta etapa, verifica-se através de um bloco de decisão, se os indivíduos possuem hora previamente marcada. Esta verificação ocorre com uma chance de 20% de ser verdadeira. Estas configurações de parâmetros representam uma abordagem inicial empírica. Outras relações, bem como as análises métricas associadas, serão consideradas posteriormente.

6.1.2 Atendimento e verificação de prioridades

Nesta etapa as entidades serão preparadas para a realização do *tour*. Verifica-se a disponibilidade de drones e a presença de elementos prioritários no cenário. Além disso, o modelo considera que elementos não prioritários e que não tenham efetuado prévio agendamento, aguardem a formação de grupos de 5 (cinco) pessoas. De modo geral, o tempo gasto para atendimento de indivíduos é considerado inicialmente segundo uma distribuição triangular com parâmetros mínimo, máximo e média iguais a 1, 7 e 5 minutos, respectivamente. Assume-se, neste trabalho, a existência de apenas um drone como recurso necessário para a execução do *tour*. Dessa forma, será realizado um *tour* por vez, devendo os próximos grupos aguardarem a liberação daquele recurso.

6.1.3 Realização do *tour*

Devidamente orientados, os grupos seguem para a execução efetiva do *tour*. Neste ponto, escolhe-se qual trajetória o *tour* seguirá. É assumido a possibilidade de um *tour* principal e mais longo por todas as dependências do espaço público ou um simples acompanhamento a

um determinado ambiente, como salas e demais setores, totalizando 5 escolhas distintas. A decisão sobre o percurso a ser efetuado é controlado no modelo por um elemento de decisão do tipo *N-way by Chance*, com chance de 60% das escolhas serem para a realização do *tour* principal (mais demorado). Através de submodelos, representa-se a execução do tour escolhido, levando-se em consideração tempos gastos em cada setor (*checkpoint*) do trajeto.

6.1.4 Refinando o *tour* com submodelos

Submodelos no Arena representam um forma de gerenciar e organizar melhor o modelo, pois permitem particioná-lo em partes menores que podem interagir entre si ou não [41]. Em cada submodelo, verifica-se inicialmente a possibilidade do drone completar o *tour* desejado por meio de um teste realizado sobre uma variável que armazena a capacidade (como a carga) disponível do equipamento. A cada trecho percorrido, esta variável é atualizada para refletir o processo de descarga do drone.

6.1.5 Finalizando o *tour*: saída de entidades

Esta etapa do modelo finaliza a percurso realizado pelas entidades, liberando o recurso alocado para o próximo grupo. Também é verificado aqui, se é preciso realizar uma nova recarga ou ajuste no drone. Os grupos são finalmente desfeitos e as entidades direcionadas para a saída do cenário.

6.1.6 Simulação do modelo e resultados obtidos

Os dados a seguir apresentam os principais resultados das simulações, considerando-se as configurações iniciais do modelo, conforme explicitadas anteriormente, com exceção do tempo entre chegadas de visitantes. Tal tempo, como mencionado, é considerado segundo uma distribuição exponencial com sua média variando entre 10 e 60 minutos, com intervalo de 10 minutos. Para todas as simulações efetuadas, considera-se o tempo de replicação como 30 (trinta) dias. A unidade de tempo está em horas (h) e os tempos indicados representam valores médios.

Considera-se que:

- T1 - Tempo em *tour*g
- T2 - Tempo em espera (filas)
- T3 - Tempo em trânsito
- T4 - Outros tempos

Tabela 6.1: Resultados para as configurações iniciais do modelo, com variação da média do tempo entre chegadas de visitantes entre 10 e 30 minutos.

Replicação	**Tempo entre chegadas (min)**		
30 dias	**expo(10)**	**expo(20)**	**expo(30)**
T1 (h)	0.3614	0.3578	0.3653
T2 (h)	1.7822	0.7927	1.0332
T3 (h)	0.07239	0.07181	0.07052
T4 (h)	0.0	0.0	0.0
Total (h)	2.2160	1.2222	1.4691

Tabela 6.2: Resultados para as configurações iniciais do modelo, com variação da média do tempo entre chegadas de visitantes entre 40 e 60 minutos.

Replicação	**Tempo entre chegadas (min)**		
30 dias	**expo(40)**	**expo(50)**	**expo(60)**
T1 (h)	0.3520	0.3635	0.3530
T2 (h)	1.3394	1.8366	2.0078
T3 (h)	0.07351	0.07295	0.07383
T4 (h)	0.0	0.0	0.0
Total (h)	1.7649	2.2731	2.4346

O tempo em trânsito corresponde ao tempo gasto pelos visitantes em deslocamentos que não envolvem o *tour* propriamente dito, como o trânsito entre guichês de atendimento. De

acordo com os dados exibidos, verifica-se que é exigido dos visitantes um menor tempo de permanência no recinto, quando o tempo entre chegadas dos mesmos obedece um distribuição de probabilidade exponencial com média de 20 minutos, uma vez que para esta distribuição o tempo em espera em filas diminui.

É possível verificar que as variações temporais mais acentuadas ocorrem com relação aos tempos em espera em filas, sendo este portanto, um fator que pode representar situações de atrasos que interferirão no andamento adequado do serviço proposto. A Figura 6.1 a seguir ilustra as oscilações do tempo em espera dos visitantes. Nota-se que, para o cenário de simulação proposto, um tempo entre chegadas de visitantes menor, resulta em formações de filas mais demoradas, como é o caso do tempo entre chegadas com média de 10 minutos. O melhor quadro se apresenta para um tempo entre chegadas segundo uma distribuição exponencial de média igual a 20 minutos. Como sugestão, a presença de um segundo drone poderia inicialmente suprir a deficiência de esperas prolongadas. Tempos entre chegadas maiores resultam em filas demoradas devido à necessidade da formação de grupos com um número mínimo de pessoas, para se dar início ao *tour*.

Vale ressaltar que as simulações executadas não consideram prováveis mudanças de trajetória sujeitas ao drone por meio da transferência até ele de novas coordenadas. Assume-se que o drone, ao dar início à execução do tour, já tenha sido previamente configurado com as coordenadas a serem seguidas.

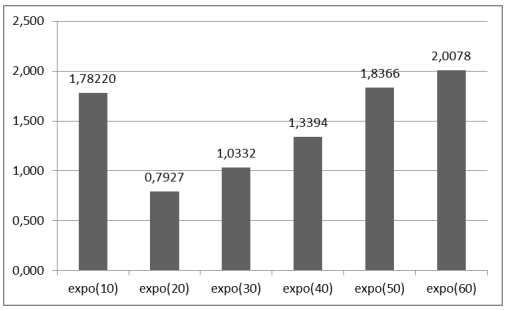


Figura 6.1: Variações do tempo em espera dos visitantes.

Considera-se agora a existência de 5 possíveis trajetos a serem escolhidos para a realização de um *tour*, mas com igual probabilidade de escolha (20%). Esta situação procura representar um cenário mais aproximado com a realidade, onde pessoas realizam diferentes *tours* em certas localidades apenas como atividades de lazer.

Tabela 6.3: Resultados assumindo a escolha de trajetos distintos, com variação da média do tempo entre chegadas de visitantes entre 10 e 30 minutos.

Replicação	**Tempo entre chegadas (min)**		
30 dias	**expo(10)**	**expo(20)**	**expo(30)**
T1 (h)	0.2937	0.2913	0.2889
T2 (h)	0.6441	0.7544	1.0412
T3 (h)	0.08460	0.08376	0.08401
T4 (h)	0.0	0.0	0.0
Total (h)	1.0224	1.1295	1.4141

Tabela 6.4: Resultados assumindo a escolha de trajetos distintos, com variação da média do tempo entre chegadas de visitantes entre 40 e 60 minutos.

Replicação	**Tempo entre chegadas (min)**		
30 dias	**expo(40)**	**expo(50)**	**expo(60)**
T1 (h)	0.3018	0.2906	0.2907
T2 (h)	1.3174	1.5664	1.9689
T3 (h)	0.08367	0.08468	0.08432
T4 (h)	0.0	0.0	0.0
Total (h)	1.7030	1.9417	2.3440

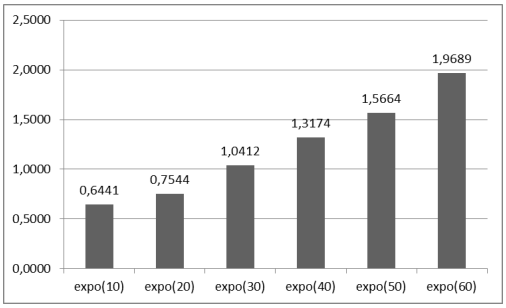


Figura 6.2: Variações do tempo em espera dos visitantes (mesma probabilidade de escolha de trajetos).

A Figura 6.2 apresenta como o tempo de espera dos visitantes varia com a suposição de igual probabilidade de escolha do *tour* para diferentes setores do espaço considerado. Também para esta simulação não são consideradas mudanças de trajetos provenientes de alterações nas coordenadas do *tour*, ou seja, não é levada em consideração a modificação externa do trajeto do *tour* via aparelho celular, no entanto, é importante salientar que há situações onde o controle de coordenadas externo pode ser valioso, em particular nos casos de emergência, vigilância ou busca por pessoas.

6.2 Execução Interna de *Scripts*

Missões autônomas executadas com o drone ou mesmo outros servo mecanismos, refletem um nível de controle onde se deseja um mínimo de interferência humana nas operações de manobras e tomadas de decisão [43]. A maior parte, no entanto, de ações de controle ditas *autônomas*, executam externamente ao veículo controlado todo o processamento de informações necessárias à missão, sendo repassadas ao veículo não tripulado dados já prontos e formatados.

O modelo *AR.Drone* disponibiliza em sua placa controladora, um *kernel linux* que possibilita a execução de *scripts* e programas no próprio drone, o que o torna verdadeiramente autônomo no que diz respeito a processamento de informações e tomadas de decisão em pleno voo.

Além de trazer consigo vários sensores e permitir a leitura de grande quantidade de dados a respeito do seu modo de operação, o *AR.Drone* permite a conexão de outros dispositivos de leitura, como sensores de temperatura ou de gás e, por meio destes, disponibiliza ainda mais informações a cerca do ambiente que o rodeia. A execução interna de rotinas permite, dentre outras possibilidades, uma extensão de hardware, agregando novos recursos ao drone, o que não é possível conseguir através de aplicativos de controle externos. Para demonstrar esta característica, um ensaio foi realizado no intuito de conectar um sensor ultrassônico que repassasse informações sobre a distância de obstáculos diretamente ao drone.

O sensor ultrassônico utilizados foi o modelo *HC-SR04*, um componente bastante comum em projetos eletrônicos, capaz de efetuar medidas de distância entre 2cm e 4m com precisão de

3mm. Seu funcionamento se baseia no envio de pulsos ultrassônicos que, ao interceptarem um obstáculo, retornam ao sensor depois um dado intervalo de tempo. Com base nesse tempo entre envio e retorno, é possível calcular a distância entre o sensor e o obstáculo detectado. Através de uma placa *Arduino UNO*[1], foi estabelecida a comunicação serial entre esta e o drone, por meio de sua interface de comunicação RX, TX, disponível na placa controladora localizada em sua estrutura interna. A Figura 6.3, representa a identificação de pinos na placa interna do drone.

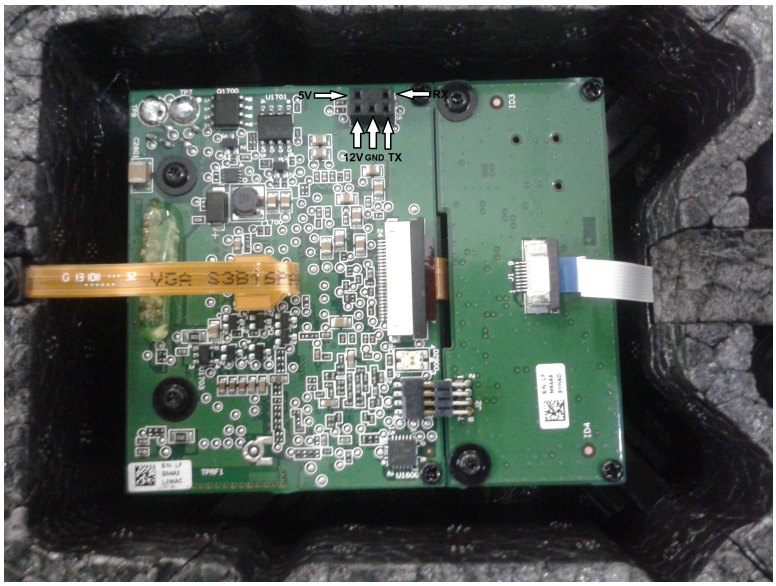


Figura 6.3: Mapa de pinos na placa interna do *AR.Drone*

Tem-se então disponíveis na placa principal de controle do drone, os contatos RX e TX para efetuar a comunicação serial com outros dispositivos, e os contatos de alimentação 5V e *GND (ground)* necessários ao funcionamento de placas que trabalhem com este nível de tensão de alimentação, como a placa *Arduino* que, por sua vez, alimentará novos sensores conectados a ele.

Por meio de uma unidade USB como um pendrive comum, é possível repassar ao sistema do *AR.Drone*, o *framework* de controle *Node.js* e suas bibliotecas, tornando possível a execução de *scripts* contendo comandos para manobrar o drone. A Figura 6.4, ilustra a conexão do *Arduino* com a placa controladora do drone.

O *pendrive* armazena o software controlador *Node.js*, suas bibliotecas para operação do drone

[1]https://www.arduino.cc/

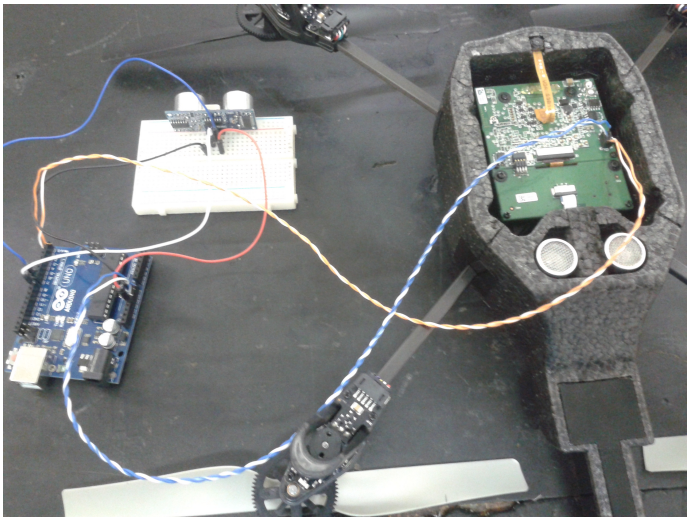


Figura 6.4: Conexão do Arduino à placa controladora do *AR.Drone*.

e os *scripts* que serão executados através do *kernel linux* instalado nele. Um desses *scripts* se encarrega de recuperar os dados provenientes do *Arduino*, através da comunicação serial fornecida pelas interfaces RX e TX. A Figura 6.5 apresenta o conjunto completo em operação.

A cada segundo, o *Arduino* calcula a distância para os obstáculos situados frontalmente ao sensor e, através de sua interface de comunicação serial, repassa essas distâncias ao drone. Por meio do *Node.js* executando no sistema *linux* do *AR.Drone*, um *script* consegue efetuar a leitura de dados disponibilizados pela serial do drone, e executar operações com base nessas leituras.

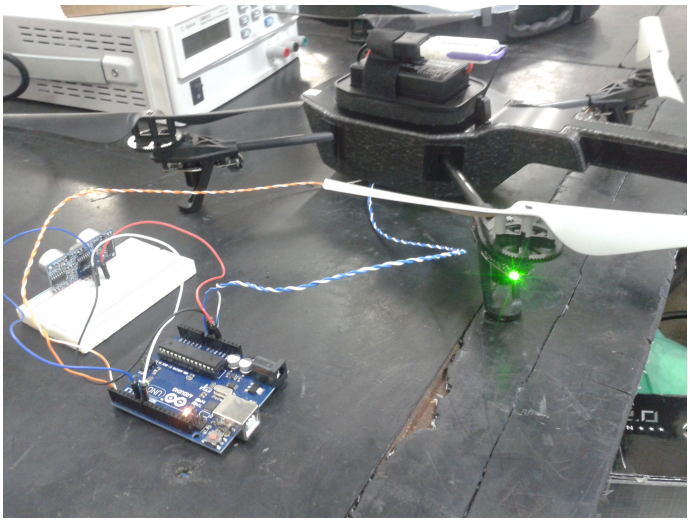


Figura 6.5: Novos sensores conectados ao drone.

Os *scripts* executados no *Node.js* são escritos em *javascript* e fornecem um maneira bastante simples de criar diversos tipos de missões. A listagem abaixo apresenta um *script* para a leitura de dados pela porta serial do drone. A biblioteca *node-serialport* desenvolvida para o *Node.js*, permite estabelecer um canal de comunicação para acessar dados provenientes da porta serial do drone, a uma taxa de 9600 bits por segundo (*baud rate: 9600 bps*). Quaisquer dados disponibilizados pela serial são então recuperados, podendo ser utilizados ao longo do *script* para operações de controle do drone, como desvios de obstáculos detectados à uma distância mínima.

```
var serialport = require('node-serialport')
var serial = new serialport.SerialPort("/dev/ttyO3", {
    parser: serialport.parsers.raw,
    baud: 9600
    })
    serial.on('data', function(chunk) {
        console.log(chunk.toString())
    })
```

A Figura 6.6 apresenta informações recuperadas pelo drone por meio de um acesso *Telnet* feito diretamente ao *AR.Drone*, através seu endereço de IP *192.168.1.1*. Dessa forma, é possível atualizar o drone quanto à presença próxima de obstáculos, fazendo-o tomar alguma decisão com base nos critérios da missão.

Telnet ou Protocolo de Terminal Virtual, é um protocolo padrão de rede cliente-servidor que permite acessar remotamente uma outra máquina que esteja executando o módulo servidor. Criado pelas Forças Armadas Americanas em 1969 para transmissão de dados entre suas bases, o *Telnet* foi disponibilizado ao público apenas em 1977.

De um modo geral, fica constatada a possibilidade da execução de *scripts* por meio do sistema de controle incorporado ao drone, utilizando para isso recursos de *hardware* e *software*, como memória e processamento, provenientes do próprio modelo, isto poderia de certa forma,

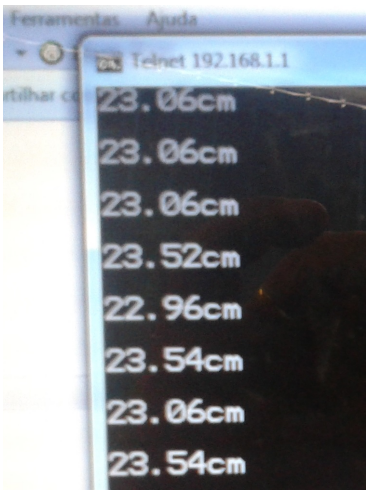


Figura 6.6: Distâncias a obstáculos repassadas ao drone.

automatizar a execução de várias missões impostas ao drone, uma vez que tais missões já estariam configuradas e carregadas nele, havendo a necessidade apenas de passar ao drone qual missão executar. Sob tal ponto de vista, vale salientar que a execução de missões ditas autônomas, torna-se mais pertinente ao próprio veículo não tripulado, levando a um mínimo de interferência humana.

Ressalta-se que os experimentos relatados foram realizados com o modelo *Parrot Ar.Drone 2.0 Elite Edition*, dessa forma, outros modelos de drones podem não disponibilizar ou permitir os mesmos mecanismos de controle e operação, tanto em nível de *hardware* como de *software*.

Capítulo 7

Conclusão

Quer seja no solo, na água ou no ar, os veículos não tripulados têm marcado presença em inúmeras atividades no cotidiano de amadores e profissionais em todo o mundo. A maneira como tais veículos é controlado tem, a cada dia, melhorado significativamente, com o aprimoramento de novos algoritmos e técnicas de controle. Existem demonstrações de drones que equilibram hastes enquanto voam, respondem a movimentos gestuais, acompanham e seguem pessoas, fazem entregas de encomendas e várias outras aplicações que comprovam que esses pequenos veículos, podem aceitar qualquer nível de controle que se deseje.

Esse trabalho apresentou um estudo sobre navegabilidade de drones sem meios de sensoriamento ou posicionamento externos como GPS, fazendo-se uso da transferência de comandos e coordenadas entre o aplicativo de controle e veículo não tripulado, sem a necessidade do repasse por parte deste, de sua atual posição no ambiente ao aplicativo. Demonstrou-se a eficácia de plataformas de desenvolvimento para a construção de soluções em software capazes de operar drones de pequeno porte como o *Parrot AR.Drone 2.0* ou mesmo acompanhá-los em missões autônomas, enviando-lhes comandos e recebendo dados sobre seu estado ou sobre o ambiente ao seu redor.

Foi construída uma aplicação funcional capaz de operar em dispositivos móveis para o controle manual de drones do tipo *Ar.Drone 2.0*, ou para configurar um percurso a ser obedecido por ele, confirmando ser viável para ser utilizada como ferramenta de apoio à navegabilidade por indicação de coordenadas, o que seria bastante útil em situações onde a câmera do drone

não pudesse ser utilizada ou em ambientes escuros. No entanto, a solução sugerida não representa, em seu sentido mais amplo, uma plataforma operacional no sentido de não possibilitar sua execução de forma embarcada e por não especificar uma arquitetura de hardware.

A necessidade do planejamentos de trajetórias ótimas que levem em conta o estado do drone e sua própria conservação tornou-se um requisito obrigatório, para as atuais atividades em que veículos não tripulados são empregados. Assim, uma nova abordagem por meio de algoritmos evolucionários para o desenho de trajetórias navegáveis foi apresentada e demonstrada neste trabalho.

O uso de algoritmos multiobjetivos é bastante apropriado ao se utilizar drones em missões autônomas, uma vez que é fundamental negociar objetivos conflitantes para encontrar soluções factíveis que os satisfaçam. Para drones e outros veículos não tripulados, dependendo da missão que precisa ser executada, inúmeros objetivos podem ser levados em consideração, particularmente aqueles envolvendo questões de segurança e autonomia da missão, tornando a abordagem evolucionária uma excelente opção a ser considerada.

Capítulo 8

Perspectivas

Como principais propostas futuras que possam agregar funcionalidades às soluções apresentadas neste trabalho, destaca-se a utilização de novos algoritmos evolucionários ou bio inspirados levando em consideração a cooperação entre drones, uma vez que a ação conjunta entre servo mecanismos tem encontrado grandes aplicações e servido como solução para diversos tipos de problemas. Outra situação que tem despertado o interesse de inúmeros pesquisadores é a suposição de obstáculos que se movem ou que surgem inesperadamente, tornando a simples missão de percorrer uma dada trajetória mais problemática, dado que o veículo precisa realizar repetidas leituras sobre o ambiente ao seu redor.

Estudos futuros podem ser direcionados também à utilização de outros tipos de drones, não necessariamente fazendo uso de comunicação por *Wi-Fi*, o que limita em demasia o raio de alcance desses veículos. A grande maioria dos drones encontrados comercialmente hoje, fazem uso de comunicação por rádio controle, com enlaces de comunicação ultrapassando centenas e até milhares de metros.

O uso de drones por empresas e indústrias de vários segmentos é um campo bastante promissor de futuras aplicações desses veículos. Companhias de energia elétrica têm usado drones para supervisionarem áreas de difícil acesso ou localizarem danos em suas redes. Grandes empresas distribuidoras já têm feito testes para a entrega de vários tipos de mercadorias a seus clientes por meio de drones, como a *Amazon*®. Dessa maneira, torna-se indispensável em tais aplicações, a utilização de sofisticados algoritmos de controle, visão computacional,

cooperação entre drones, etc. Outra proposta de estudos futuros diz respeito à autonomia, quanto ao consumo de energia, dos veículos não tripulados. É sabido que este é um problema particularmente acentuado para veículos aéreos como drones, que precisam de grande quantidade de energia e cuja autonomia dura apenas alguns poucos minutos. Lança-se então a proposta para a construção de drones que possuam formas alternativas e mais duráveis de energia, fazendo-se uso por exemplo, de energia eólica ou solar.

Referências Bibliográficas

[1] Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[2] Faez Ahmed and Kalyanmoy Deb. Multi-objective path planning using spline representation. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 1047–1052. IEEE, 2011.

[3] U. S. Army. *Unmanned Aircraft System: Roadmap 2010-2035*. Fort Rucker, Alabama, 2010.

[4] G.J. Barlow. Design of autonomous navigation controllers for unmanned aerial vehicles using multi-objective genetic programming. Master's thesis, North Carolina State University, Raleigh, NC, 2004.

[5] Gregory J Barlow, Choong K Oh, and Edward Grant. Incremental evolution of autonomous controllers for unmanned aerial vehicles using multi-objective genetic programming. In *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, volume 2, pages 689–694. IEEE, 2004.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[7] E. Besada-Portas, L. de la Torre, J. M. de la Cruz, and B. de Andrés-Toro. Evolutionary trajectory planner for multiple uavs in realistic scenarios. *IEEE Transactions on Robotics*, 26(4):619–634, Aug 2010.

[8] S. Bouabdallah and R. Siegwart. Full control of a quadrotor. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 153–158, Oct 2007.

[9] Alexandre S. Brandão. Modelagem e controle não linear subatuado de um quad-rotor: Parte 1. In *CONGRESSO BRASILEIRO DE AUTOMÁTICA, 19., 2012, Campina Grande. Anais do XIX Congresso Brasileiro de Automática (CBA 2012)*, pages 449–454. UFCG, 2012.

[10] Shui-Nee Chow and James A Yorke. Lyapunov theory and perturbation of stable and asymptotically stable systems. *Journal of Differential Equations*, 15(2):308–321, 1974.

[11] Halil Cicibas, Kadir Alpaslan Demir, Murat M.Gunal, and Nafiz Arica. A simulation model for analysing unmanned aerial vehicle flight paths. *Proceedings of the European Modeling and Simulation Symposium*, pages 543–548, 2012.

[12] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007.

[13] Marco A Contreras-Cruz, Victor Ayala-Ramirez, and Uriel H Hernandez-Belmonte. Mobile robot path planning using artificial bee colony and evolutionary programming. *Applied Soft Computing*, 30:319–328, 2015.

[14] M. A. Corrêa. *Modelo de Veículos Aéreos não Tripulados Baseado em Sistemas Multi-agentes.* PhD thesis, DTese (Doutorado em Engenharia Elétrica) – Escola Politécnica (Poli), Universidade de São Paulo (USP) – São Paulo, 2008.

[15] S. E. A. P. Costa. Controlo e simulação de um quadrirotor convencional, dissertação para obtenção do grau de mestre em engenharia aeroespacial. Master's thesis, Universidade Técnica de Lisboa, 2008.

[16] T. H. Cox. *Civil UAV Capability Assessment.* NASA and CSM, Inc. [S.l.], 2004.

[17] Konstantinos Dalamagkidis, Kimon P. Valavanis, and Les A. Piegl. *On Integrating Unmanned Aircraft Systems into the National Airspace System: Issues, Challenges, Operational Restrictions, Certification, and Recommendations*, volume 36 of *Intelligent Systems, Control and Automation: Science and Engineering.* Springer-Verlag, 2009.

[18] Kalyanmoy Deb. *Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction*, pages 3–34. Springer London, London, 2011.

[19] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[20] Kalyanmoy Deb and Santosh Tiwari. Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research*, 185(3):1062–1087, 2008.

[21] Carmelo Di Franco and Giorgio Buttazzo. Energy-aware coverage path planning of uavs. In *Autonomous Robot Systems and Competitions (ICARSC), 2015 IEEE International Conference on*, pages 111–117. IEEE, 2015.

[22] Gerry Dozier, Shaun McCullough, Abdollah Homaifar, Eddie Tunstel, and Loretta Moore. Multiobjective evolutionary path planning via fuzzy tournament selection. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 684–689. IEEE, 1998.

[23] DroneShow. Anac apresenta as regras para uso de drones e aeromodelos. preprint (2015), disponível em `http://www.droneshowla.com/`.

[24] J.J. Durillo, A.J. Nebro, and E. Alba. The jmetal framework for multi-objective optimization: Design and architecture. In *CEC 2010*, pages 4138–4325, Barcelona, Spain, July 2010.

[25] Juan J. Durillo and Antonio J. Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42:760–771, 2011.

[26] J. Lopes E. Pastor and P. Royo. Uav payload and mission hardware/software architecture. *IEEE A&E Systems Magazine*, June 2007.

[27] J. J. Engel. Autonomous camera-based navigation of a quadrocopter. Master's thesis, Fakultät Für Informatik der Technischen Universität München, 2011.

[28] Laurent Eschenauer. ardrone-autonomy. `https://github.com/eschnou/ardrone-autonomy`. Acessado: 20-05-2015.

[29] Node.js Foundation. Node.js. `https://nodejs.org/en/`. Acessado: 20-06-2015.

[30] Si-Yao Fu, Li-Wei Han, Yu Tian, and Guo-Sheng Yang. Path planning for unmanned aerial vehicle based on genetic algorithm. In *Cognitive Informatics & Cognitive Computing (ICCI* CC), 2012 IEEE 11th International Conference on*, pages 140–144. IEEE, 2012.

[31] Luis F. Gonzalez, Dong-Seop Lee, and Rodney A. Walker. Optimal mission path planning (mpp) for an air sampling unmanned aerial system. In S. Scheding, editor, *2009 Australasian Conference on Robotics & Automation*, pages 1–9, Sydney, N.S.W, 2009. Australian Robotics & Automation Association.

[32] Mohammad Hamdan. On the disruption-level of polynomial mutation for evolutionary multi-objective optimisation algorithms. *Computing and Informatics*, 29(5):783–800, 2010.

[33] S. Hota and D. Ghose. Optimal path planning for an aerial vehicle in 3d space. pages 4902–4907. Proceedings of the 49th IEEE Conference on Decision and Control, 2010.

[34] Yong K Hwang and Narendra Ahuja. Gross motion planning—a survey. *ACM Computing Surveys (CSUR)*, 24(3):219–291, 1992.

[35] Inc Informer Technologies. Ar.freeflight 2.4.15. `http://ar-freeflight.android.informer.com/`. Acessado: 10-02-2015.

[36] Atikah Janis and Abdullah Bade. Path planning algorithm in complex environment: A survey. *Transactions on Science and Technology*, 3(1):31–40, 2016.

[37] Ivan Johansen. Graph plotting of mathematical functions. `http://www.padowan.dk`. Acessado: 15-06-2015.

[38] Levi Jones. *Coordination and control for multi-quadrotor UAV missions*. PhD thesis, Monterey, California. Naval Postgraduate School, 2012.

[39] M Kamran Joyo, D Hazry, S Faiz Ahmed, M Hassan Tanveer, Faizan A Warsi, and AT Hussain. Altitude and horizontal motion control of quadrotor uav in the presence of air turbulence. In *Systems, Process & Control (ICSPC), 2013 IEEE Conference on*, pages 16–20. IEEE, 2013.

[40] Hu Jun and Zhu Qingbao. Multi-objective mobile robot path planning based on improved genetic algorithm. In *Intelligent computation technology and automation (ICICTA), 2010 international conference on*, volume 2, pages 752–756. IEEE, 2010.

[41] W. David Kelton, Randall P. Sadowski, and Deborah A. Sadowski. *Simulation with Arena*. McGrawHill, 2nd edition, 2002.

[42] J. Kok, L.F. Gonzalez, and N.A. Kelson. Fpga implementation of an evolutionary algorithm for autonomous unmanned aerial vehicle on-board path planning. *IEEE Transactions on Evolutionary Computation*, 2012 (In press).

[43] Jonathan Kok, Troy S. Bruggemann, and Luis F. Gonzalez. An evolutionary computation approach to three-dimensional path planning for unmanned aerial vehicles with tactical and kinematic constraints. In *15th Australian International Aerospace Congress (AIAC 15)*, Melbourne Convention Centre, Melbourne, VIC, February 2013.

[44] Tomás Krajník, Vojtech Vonásek, Daniel Fiser, and Jan Faigl. Ar-drone as a platform for robotic research and education. In David Obdrzálek and Achim Gottscheber, editors, *Eurobot Conference*, volume 161 of *Communications in Computer and Information Science*, pages 172–186. Springer, 2011.

[45] Shupeng Lai, Kangli Wang, Hailong Qin, Jin Q Cui, and Ben M Chen. A robust online path planning approach in cluttered environments for micro rotorcraft drones. *Control Theory and Technology*, 14(1):83–96, 2016.

[46] S LaValle. Planning algorithms, cambridge univ, 2006.

[47] N. Berezny; L. Greef; B. Jensen; K. Sheely; M. Sok; D. Lingenbrink and Z. Dodds. Accessible aerial autonomy. technologies for practical robot applications (tepra). In *IEEE International Conference*, pages 53–58. IEEE, 2012.

[48] John A Marin, Robert Radtke, David Innis, Donald R Barr, and Alan C Schultz. Using a genetic algorithm to develop rules to guide unmanned aerial vehicles. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 1, pages 1055–1060. IEEE, 1999.

[49] G. Martin. Modelling and control of the parrot ar.drone vehicle on-board path planning. Master's thesis, School of Engineering and Information Technology, UNSW Canberra, 2012.

[50] Hao Meng and Guizhou Xin. Uav route planning based on the genetic simulated annealing algorithm. In *Mechatronics and Automation (ICMA), 2010 International Conference on*, pages 788–793. IEEE, 2010.

[51] J. Messias. Framework para sistemas de navegação de veículos aéreos não tripulados. Technical report, Universidade Federal de Ouro Preto, 2014.

[52] Z. Michalewicz. *Genetic algorithms, numerical optimization, and constraints*. Morgan Kaufmann, 1995.

[53] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1998.

[54] S. Mittal and K. Deb. Three-dimensional online path planning for uavs using multiobjective evolutionary algorithms. pages 3195–3202. Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, September 2007.

[55] Shashi Mittal and Kalyanmoy Deb. Three-dimensional offline path planning for uavs using multiobjective evolutionary algorithms. In *IEEE Congress on Evolutionary Computation*, pages 3195–3202. IEEE, 2007.

[56] Ioannis K Nikolos, Kimon P Valavanis, Nikos C Tsourveloudis, and Anargyros N Kostaras. Evolutionary algorithm based offline/online path planner for uav navigation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 33(6):898–912, 2003.

[57] André Oliveira. Drones e vants: O que é permitido e proibido no brasil. `http://awinformaticastm.blogspot.com.br/2016/03/drones-e-vants-o-que-e-permitido-e.html`. Acessado: 10-08-2016.

[58] S. Piskorski, N. Brulez, and P. Eline. *AR.Drone Developer Guide, SDK 1.7*. Parrot S.A, 1st edition, 2011.

[59] H. Qiu, W. Zhou, and H. Wang. A genetic algorithm-based approach to flexible job-shop scheduling problem. *In Natural Computation, 2009. ICNC '09. Fifth International Confer- ence on*, 4:81–85, August 2009.

[60] Yao-hong Qu, Quan Pan, and Jian-guo Yan. Flight path planning of uav based on heuristically search and genetic algorithms. In *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005.*, pages 5–pp. IEEE, 2005.

[61] David Rathbun, Sean Kragelund, Anawat Pongpunwattana, and Brian Capozzi. An evolution based path planning algorithm for autonomous motion of a uav through uncertain environments. In *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*, volume 2, pages 8D2–1. IEEE, 2002.

[62] Sam. Explore the various types of drones. preprint (2016), disponível em `http://www.topdronesforsale.org/types-of-drones/`.

[63] Michael David Schmidt. Simulation and control of a quadrotor unmanned aerial vehicle. 2011.

[64] Paragon Decision Science. Arena. `http://www.paragon.com.br/softwares/arena/`. Acessado: 05-03-2015.

[65] Kleber L. Silva. Hardware para controle avançado de veículo aéreo não tripulado do tipo quadricóptero. In *CONFERÊNCIA DE ESTUDOS EM ENGENHARIA ELÉTRICA, 11., 2013, Uberlândia. Anais da XI Conferência de Estudos em Engenharia Elétrica*, pages 01–06. UFU, 2013.

[66] Softonic. Nasa world wind. `http://nasa-world-wind.softonic.com.br/`. Acessado: 22-04-2016.

[67] Kazuo Sugihara and John Smith. Genetic algorithms for adaptive planning of path and trajectory of a mobile robot in 2d terrains. *IEICE TRANSACTIONS on Information and Systems*, 82(1):309–317, 1999.

[68] R. C. Sá. Construção, modelagem dinâmica e controle pid para estabilidade de um veículo aéreo não tripulado do tipo quadrirotor. Master's thesis, Universidade Fede-

ral do Ceará, Centro de Tecnologia, Departamento de Engenharia de Teleinformática, Fortaleza, 2012, 2012.

[69] Rejane C. Sá. Construction and pid control for stability of an unmanned aerial vehicle of the type quadrotor. In *IEEE LATIN AMERICAN ROBOTICS SYMPOSIUM (LARS/LAC), 2013, Arequipa. Proceedings of the 2013 IEEE Latin American Robotics Symposium*, pages 95–99. IEEE, 2013.

[70] Curtis Rueden Wayne Rasband. Imagej an open platform for scientific image analysis. `http://imagej.net/Welcome`. Acessado: 10-01-2016.

[71] Shigeo Yoshida and Rian Wouters. Yadrone. `https://github.com/MahatmaX/YADrone/tree/master/YADrone`. Acessado: 10-02-2015.

[72] Chao Zhang, Ziyang Zhen, Daobo Wang, and Meng Li. Uav path planning method based on ant colony optimization. In *2010 Chinese Control and Decision Conference*, pages 3790–3792. IEEE, 2010.

[73] N. Özalp and O. K. Sahingoz. Optimal uav path planning in a 3d threat environment by using parallel evolutionary algorithms. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 308–317, May 2013.

Apêndice A

Modelagem e Simulação no Arena

A.1 Chegada de Entidades

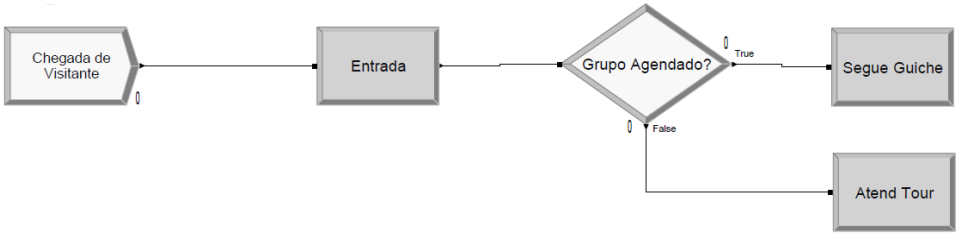


Figura A.1: Representação da chegada de visitantes ao local do tour.

A.2 Realização de Atendimento

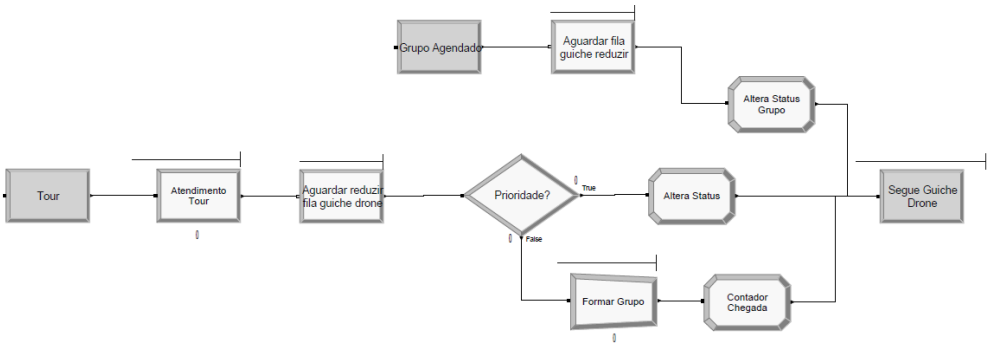


Figura A.2: Realização do atendimento aos visitantes.

A.3 Realização do Tour

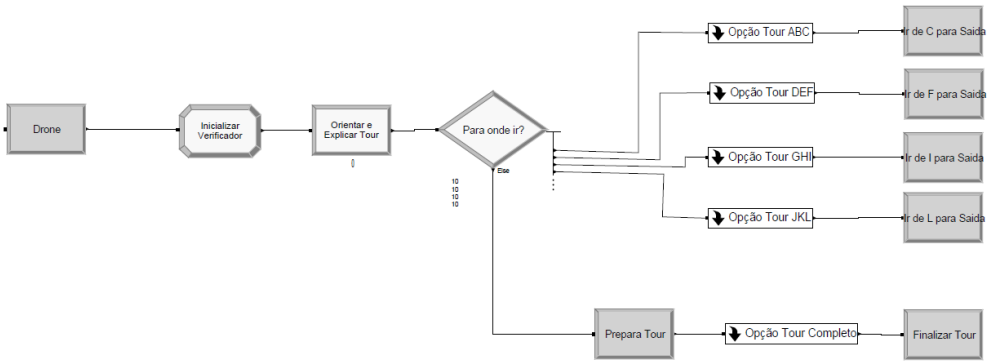


Figura A.3: Realização do tour pelos visitantes com o drone como guia.

A.4 Refinamento do Tour

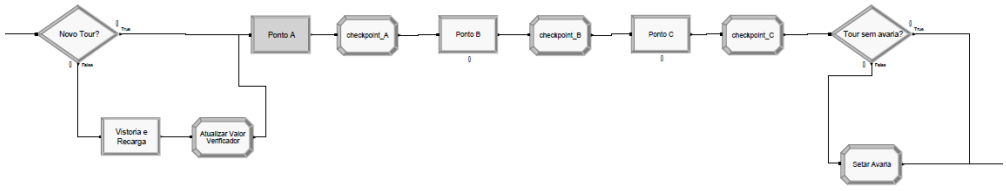


Figura A.4: Submodelos destacando as opções do tour.

A.5 Finalizando o Tour

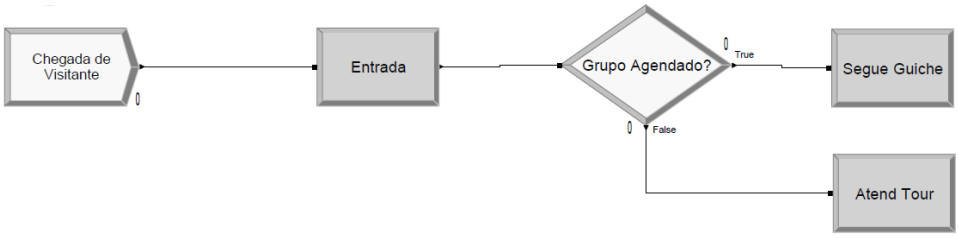


Figura A.5: Finalização do tour.

Apêndice B

Ambiente de Testes de Voos

Figura B.1: Sala de testes para filmagens do drone. Cortesia Escola Técnica Senai Areias (Sala 140).

Figura B.2: Marcações para manter o posicionamento fixo da bancada e notebook.

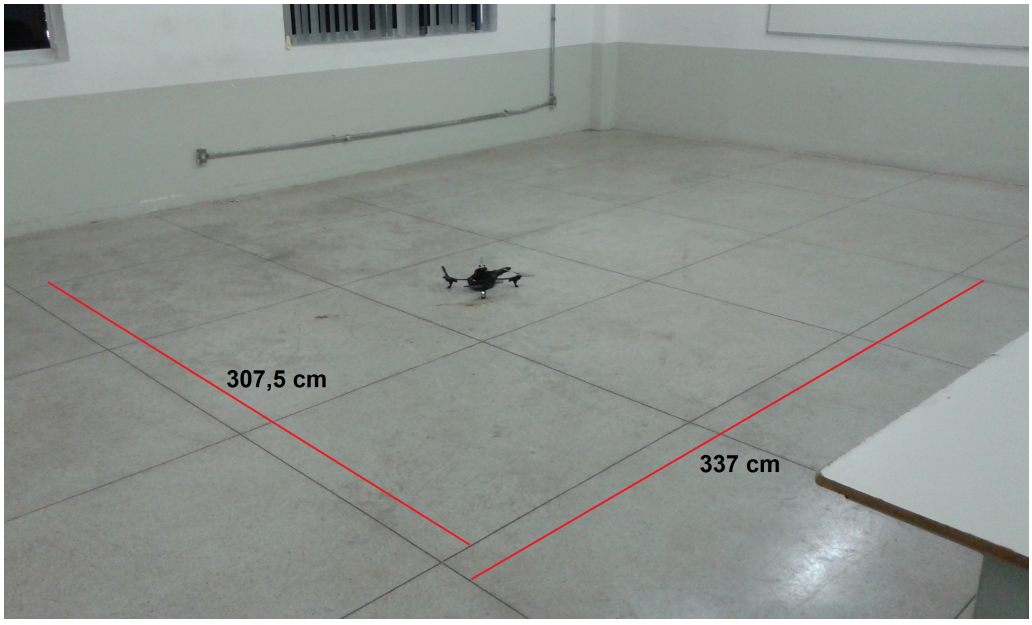


Figura B.3: Comprimentos principais de quadros no piso.