

ELDER Y. NAKASHIMA
RAFAEL H. DA ROCHA

Automação e localização de drone em ambiente indoor

São Paulo
2015

ELDER Y. NAKASHIMA
RAFAEL H. DA ROCHA

Automação e localização de drone em ambiente indoor

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do diploma de conclusão de curso.

Área de Concentração:
Engenharia de Computação

Orientador:
Prof. Dr. Reginaldo Arakaki

Coorientadores:
Leandro Rodrigues de Souza
Marcelo Angelo Pita

São Paulo
2015

FICHA CATALOGRÁFICA

Nakashima et al.

Automação e localização de drone em ambiente indoor/ E. Y. Nakashima, R. H. da Rocha. São Paulo, 2015.
115 p.

Monografia (Graduação em Engenharia de Computação) — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

1. Visão computacional #1.
 2. Veículos guiados remotamente #2.
 3. Robôs #3.
- I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais (PCS). II. t.

Aos nossos familiares e amigos, que nos motivaram e sempre acreditaram em nós.

AGRADECIMENTOS

Ao nosso orientador, professor Reginaldo Arakaki, pelos passos à frente que nos fazem enxergar além.

Aos nossos coorientadores, Leandro Rodrigues de Souza e Marcelo Angelo Pita, pelos diversos caminhos apresentados para passarmos.

À professora Anna Helena Reali Costa, ao Daniel Makoto Tokunaga e ao Milton Hurpia da Rocha, por mostrar quais os caminhos são sem saída e quais podemos andar.

Aos amigos e colegas, pela caminhada compartilhada.

A nossos familiares, por nos ensinarem a caminhar.

*"If I have seen further it is only by
standing on the shoulders of giants."*
Isaac Newton

RESUMO

Na robótica móvel, um dos paradigmas gira em torno da localização do robô em relação ao ambiente em que está inserido. Ou seja, trata-se de um problema de estimativa de posição. Utiliza-se como plataforma o Parrot AR.Drone 2.0 Elite Edition, um quadricóptero que contém diversos sensores, dentre os quais, após detalhada pesquisa aplicada, é escolhida a sua câmera frontal para criar um módulo de localização por visão computacional. Para tal, é desenvolvido um simulador, que possibilita o teste prévio dos algoritmos antes de testá-lo no drone real. Por fim, integra-se ao módulo de localização as bibliotecas de controle e comando do drone, e utilizando um banco de dados, uma interface engloba todas as partes do projeto, possibilitando ao usuário a experiência de controlar o quadricóptero e designar-lhe tarefas, validando o funcionamento da localização em ambiente indoor.

ABSTRACT

In mobile robotics, one of the paradigms is about the location of a robot related to the environment in which it is inserted, in other words, a position estimation issue. The chosen platform is the Parrot AR.Drone 2.0 Elite Edition, a quadcopter containing a variety of sensors. After detailed applied research, the frontal camera is chosen as the candidate which best fits the requirements, able to create a tracking module based on computer vision. To achieve this goal, a simulator is developed, enabling the prior testing of algorithms before testing it in the real drone. Finally, all modules of localization, database, control and libraries are put together using a man-machine interface, in which the user can experience assigning tasks to the drone, validating the localization function in indoor environments.

SUMÁRIO

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

1	Introdução	16
1.1	Motivação	16
1.2	Objetivos	17
1.3	Objetivos específicos	18
2	Plataforma: Parrot AR.Drone 2.0	20
2.1	O drone	20
2.2	Bateria	21
2.3	Câmeras	22
2.4	Sistema de navegação inercial	22
2.5	Movimentação do drone	26
3	Localização Indoor	31
3.1	Visão computacional	32
3.2	Infravermelho	33
3.3	Laser	34

3.4	Ondas sonoras	34
3.5	Radiofrequência	36
3.5.1	Identificação por radiofrequência	36
3.5.2	WLAN	37
3.5.3	ZigBee	37
3.5.4	Bluetooth	37
3.5.5	Célula de origem	38
3.5.6	Reconhecimento de padrões	38
3.5.7	Banda ultralarga	40
3.6	Magnetismo	40
3.7	Conclusões e comparações	41
4	Visão computacional	43
4.1	Definições básicas	44
4.2	Transformações geométricas	44
4.2.1	Coordenadas homogêneas e matrizes relacionadas	44
4.2.2	Calibração da câmera	45
4.3	Extração de características	45
4.3.1	Características de baixo nível	45
4.3.2	Extração por correspondência de forma	46
4.3.3	Descrição de objetos	47
4.4	Marcadores artificiais	47

4.5	Sistemas monoculares e estéreo	48
4.6	Bibliotecas	48
4.7	Filtro de Kalman	49
4.8	Escolhas	50
5	Especificação	51
5.1	Requisitos de localização indoor	51
5.2	Requisitos da interface	54
6	Plano de trabalho	55
6.1	Metodologia	55
6.2	Codificação e versionamento	62
7	Arquitetura	64
7.1	Componentes	64
7.1.1	Drone	64
7.1.2	Terminal	66
7.1.3	Comunicação	66
7.1.3.1	Navdata	66
7.1.3.2	Video stream	67
7.1.3.3	AT commands	67
7.1.3.4	Control port	67
7.2	Módulos	67

7.2.1	Simulador	67
7.2.1.1	Requisitos do simulador	68
7.2.1.2	Alternativas de simulador	69
7.2.1.3	Justificativa de escolha	70
7.2.2	Algoritmo de melhor caminho	71
7.2.2.1	Algoritmo A*	71
7.2.2.2	Sistema	71
7.2.3	Interface	72
7.2.3.1	Localização	72
7.2.3.2	Câmera frontal	73
7.2.3.3	Comandos	73
7.2.3.4	Sistema	74
7.2.4	Drone	75
7.2.5	Localização	76
7.2.6	Sistema final	77
8	Implementação e Testes	78
8.1	Simulador	78
8.1.1	Modelagem	78
8.1.2	Controle	80
8.1.3	Algoritmos	81
8.1.3.1	Algoritmo de ir ao destino	81

8.1.3.2	Testes do algoritmo de ir ao destino	81
8.1.3.3	Algoritmo de seguir rota	83
8.1.3.4	Testes do algoritmo de seguir rota	84
8.2	Algoritmo de melhor caminho	89
8.2.1	Algoritmo	89
8.2.2	Testes	90
8.3	Interface	90
8.3.1	Integração com o simulador	90
8.3.1.1	Localização	90
8.3.1.2	Controle	90
8.3.2	Testes da integração do controle do simulador	93
8.4	Drone	94
8.4.1	Controle por comandos	94
8.4.2	Imagen para o usuário	95
8.5	Localização	96
8.5.1	ARToolKit	96
8.5.1.1	Marcadores	99
8.5.1.2	Integração dos marcadores com a câmera frontal do Drone	100
8.5.1.3	Teste 1: Leitura do marcador	101
8.5.1.4	Teste 2: Multimarcadores	101
8.5.1.5	Teste 3: Ajuste de coordenadas	102

8.5.2	Testes	103
9	Considerações Finais	105
9.1	Dificuldades	105
9.2	Objetivos alcançados	106
9.3	Desafios futuros	107
Referências		109
Apêndice A - Sumário dos AR commands		113
Apêndice B - Configurações gerais do Drone		114
Apêndice C - Testes do pathfinding		115

LISTA DE ILUSTRAÇÕES

1	Imagen ilustrativa do drone utilizado com casco indoor	21
2	Placa com os sensores de movimento	23
3	Placa com ultrassom e bússola	26
4	Eixos de rotação do drone	27
5	Torque em cada motor do quadricóptero	28
6	Torque em modo hover	29
7	Torque para rotação em relação ao eixo z	29
8	Torque para movimentação horizontal	30
9	Organização de trabalho, com tarefas, sprints e milestones.	55
10	Gantter - Ferramenta online para gestão de projetos com planejamento do projeto.	57
11	Trello - Ferramenta online para gestão das tarefas.	57
12	Tarefa colocada na coluna blocked.	59
13	Utiliza-se o Trello para conversas, centralizando a comunicação.	60
14	Informações interessantes são mantidas nos cards, servindo de referência na documentação.	61
15	Metodologia para utilização do git.	62
16	Representação da divisão em camadas da arquitetura de comunicação.	65
17	Arquitetura do sistema com simulador e algoritmo de melhor caminho	71
18	Interface do usuário	72

19	Interface do usuário	73
20	Arquitetura do sistema com simulador, algoritmo de melhor caminho e simulador.	74
21	Arquitetura após a substituição do simulador pelo drone	75
22	Arquitetura final do sistema	76
23	Arquitetura final do sistema	77
24	Ambientes de simulação. À esquerda, mais simplificado para os primeiros testes. À direita, com paredes, para testes do Algoritmo de pathfinding.	79
25	Uma cadeira simplificada, suficiente para testes. Drone também simples, com parte superior chanfrada e em vermelho para não haver confusão de sua orientação.	79
26	Ambiente de programação da Blender Game Engine.	80
27	Fluxograma do primeiro algoritmo do simulador	81
28	Drone no tempo inicial, antes da execução, no teste do algoritmo de ir ao destino	82
29	Drone no tempo final, após a execução do algoritmo	82
30	Fluxograma do algoritmo de seguir rota. Ele reutiliza o algoritmo previamente criado de ir ao destino.	84
31	Drone no tempo inicial, antes da execução, no teste do algoritmo de ir ao destino. O drone está localizado no canto superior esquerdo.	85
32	Drone no tempo final, após a execução do algoritmo. O Drone se encontra no canto superior direito.	86
33	Primeiro mapa e caminho criados pelo código em Processing.	88

34	Interface de controle pelo usuário.	91
35	Interface mostrando a posição atual do Drone.	91
36	Interface de controle pelo usuário, durante o vôo do drone no simulador.	92
37	O simulador funcionando simultaneamente com a interface do usuário.	93
38	Diagrama com a sequencia de operações entre o comando do usuário e a execução pelo drone.	95
39	Drone voando, replicando sua câmera frontal na interface.	96
40	Exemplos de marcadores 2D 3x3	97
41	Diagrama de localização usando ARToolKit	98
42	Teste com a detecção de marcadores durante o vôo do drone.	103
43	Teste de detecção de marcadores, detectando-os parcialmente.	104

LISTA DE TABELAS

1	Relação entre teclas e botões da interface com seus respectivos comandos.	74
---	---	----

LISTA DE ABREVIATURAS E SIGLAS

AoA	Angle of Arrival
BD	Banco de Dados
BGE	Blender Game Engine
CMOS	Complementary Metal-Oxide Semiconductor
CPU	Central Processing Unit
DDP	Data Distribution Protocol
GPS	Global Positioning System
GPU	Graphic Processing item [IHM] Interação Homem-Máquina
iOs	iPhone Operating system
IP	Internet Protocol
MEMS	Micro Electro-Mechanical System
PMBoK	Project Management Body of Knowledge
PTAM	Parallel Tracking and Mapping
QCIF	Quarter Common Intermediate Format
RFID	RadioFrequency IDentification
ROS	Robot Operating System
RSS	Received Signal Strenght
SDK	Software Development Kit
SGBD	Sistema de Gerenciamento de Banco de Dados

SIFT	Scale-invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SMP	Symmetric Multi Processing
SURF	Speeded-up Robust Features
SVM	Support Vector Machine
TCP	Transmission Control Protocol
ToA	Time of Arrival
UDP	User Datagram Protocol
VANT	Veículo Aéreo Não-Tripulado
VGA	Video Graphics Array
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network

1 INTRODUÇÃO

1.1 Motivação

O cenário em que se insere este projeto consiste em drones não-tripulados, os quais possuem aplicações diversas, tanto no âmbito profissional quanto no pessoal, portanto com um grande espectro de utilização.

O primeiro uso conhecido e amplamente utilizado de VANTs (Veículos Aéreos Não-Tripulados) foi na inteligência militar, como apoio aéreo para artilharia, mísseis e atividades de patrulhamento em guerra. Na mesma área, porém mais contemporânea, tem seu uso em missões com risco de morte, na função de substituir a presença humana por máquinas, como no desarmamento de um dispositivo explosivo, ou a fim de vasculhar áreas afetadas pela radiação (FRANKE, 2015).

Num viés similar, o drone pode ser utilizado para demais tipos de monitoramento. Entre eles, pode realizar rondas em estabelecimentos que necessitam de segurança, substituindo novamente a presença humana e diminuindo situações que trazem risco à vida. Outro uso recorrente se faz no monitoramento de vida selvagem, obtendo informações sobre os animais e sobre as condições ambientais - como por exemplo, focos de incêndio em tempo real, auxiliando na tomada de decisões para solução de possíveis problemas (AEROVIORNMENT, 2015).

Comercialmente, a Amazon, empresa multinacional de comércio eletrônico, já se utiliza de drones para fazer entregas de encomendas (AMAZON, 2015).

Mas o uso de drones não se restringe apenas a empresas ou no setor militar; devido à sua fabricação comercial e consequente barateamento, seu uso pessoal já é amplo e, com o auxílio de acessórios como câmeras onboard - ou ego-motion - e sensores, a gama de possibilidades de uso do drone como hobby é diversificado (PHOTOJOJO, 2015).

Com as câmeras, também é possível fazer uso de drones para cobertura jornalística ou documentária, explorando a mobilidade para fazer filmagens e fotografias aéreas, trazendo perspectivas e pontos de vista diferentes (GOHOBBY, 2015). Além disso, vídeos publicitários e de promoção em redes sociais também fazem uso de tomadas aéreas em drones para proporcionar um produto atrativo.

Um possível e provável próximo passo na utilização de drones é a sua automação. Se a princípio o objetivo foi retirar o tripulante de vôo, o desafio então se torna retirar também o controlador à distância, ou seja, dotar o veículo de uma inteligência capaz de utilizar os sensores para obter informações do ambiente e assim prover um grau de automação cada vez maior, dessa forma com interferência humana menor ou em mais alto nível.

1.2 Objetivos

O objetivo principal é prover maior automação no uso de drones, de forma que não seja necessária a presença de uma pessoa como piloto. Temos como exemplos de aplicação:

Entregas automáticas: consiste na possibilidade de tornar o drone uma ferramenta de entrega de encomendas, sendo capaz de se localizar e criar trajetórias para chegar ao destino determinado a ele. Somado a um módulo de reconhecimento de obstáculos, o veículo entregador pode fazer alterações na rota e ainda assim, como se localiza no ambiente, ser capaz de chegar ao endereço programado.

Vigília automática: combinado com o reconhecimento de movimento ou de presença humana, o sistema pode detectar comportamento suspeito no ambiente e avisar sobre uma possível ameaça de roubo ou invasão. Dotado de sensores, como temperatura, calor ou câmeras, variações suspeitas ou detecções provindas destes possibilitam criar alertas automáticos.

Para poder realizar aplicações como estas, se faz necessário um sistema de localização indoor. Este então, é um objetivo do projeto. É importante ressaltar que o GPS (Global Positioning System) não funciona bem em ambiente indoor, por isso não é uma solução viável. A potência do sinal recebido é prejudicada na medida em que há a atenuação do sinal dos satélites pelas paredes, tetos e demais obstáculos físicos similares. Além disso, a precisão padrão dos dispositivos GPS está aquém da desejada para a aplicação em ambientes indoor. Existem diversas tecnologias diferentes para estimar uma localização do drone no ambiente. Também se encaixa como objetivo, então, escolher qual ou quais melhor se adequam ao projeto, de acordo com seus requisitos, premissas e limitações. Uma vez que tais tecnologias são definidas, o objetivo então é implementar e realizar testes que verifiquem a validade do uso deste tipo de veículo aéreo não tripulado para localização em ambientes indoor.

1.3 Objetivos específicos

Desenvolver um sistema de controle e automação do drone, utilizando seus sensores e via de comunicação para enviar e receber comandos, sendo acessível via navegador. Para isso, criar uma interface que informe ao usuário a posição do drone no ambiente e que apresente a ele os comandos necessários.

Para tal, projetar o sistema em Node.js (NODE.JS, 2015), com a interface do usuário desenvolvida com o framework Meteor (METEOR, 2015b), dados armazenados em um banco de dados Mongo (MONGO, 2015) e modelo do drone Parrot AR.Drone 2.0 Elite Edition(PARROT, 2015), sendo esse controlado com suporte da biblioteca node-

ar-drone (GEISENDÖRFER, 2012).

2 PLATAFORMA: PARROT AR.DRONE 2.0

2.1 O drone

O AR.Drone 2.0 é um quadricóptero controlado remotamente produzido pela empresa francesa Parrot SA. Tal controle é feito através de um aplicativo disponível tanto para Android quanto para iOS (iPhone Operating system). Para os demais sistemas operacionais, como bada, Symbian and Windows Phone, existem softwares não-oficiais. Sua primeira versão data de 2010, sendo vencedora de prêmio (CES Innovations Award for Electronic Gaming Hardware). A versão segunda, que é a utilizada neste trabalho, foi lançada em 2012. Mais especificamente, o modelo utilizado é o AR.Drone 2.0 Elite Edition, sendo o mais completo da linha.

A carcaça do drone é feito de material plástico e espuma. A conexão com o drone é realizada utilizando uma WLAN (Wireless Local Area Network) criada por ele mesmo, sendo assim controlável por aparelhos móveis, e guiada através de video streaming por câmera.

Figura 1: Imagem ilustrativa do drone utilizado com casco indoor



Fonte: Parrot

A CPU (Central Processing Unit) do AR.Drone 2.0 é uma OMAP 3630 CPU produzida pela Texas Instruments(TEXAS INSTRUMENTS, 2015). O processador é sustentado por um ARM Cortex A8 32-bits a 1 GHz, a placa gráfica é uma PowerVR SGX530 GPU (Graphic Processing Unit) com frequência de 800 MHz. Possui uma memória DDR2-RAM de 1 GB, que roda a 200 MHz e também uma NAND-Flash de 128 MB. O sistema operacional do drone é um Kernel Linux 2.6.32. Isto possibilita um sistema leve e de rápida reação.

Sem o casco, as dimensões são 451x451 milímetros, pesando 380 gramas. Já com o casco para proteção em uso indoor, o peso aumenta 40 gramas, indo a 420 gramas, e as dimensões aumentam para 517x517 milímetros.

2.2 Bateria

A bateria é constituída de 3 células de lítio-polímero, cuja capacidade é de 1,000 mAh a 11.1 volts e capacidade de descarga de 10C (10 Amps). Pesa 105 gramas; possui um módulo de proteção do circuito contra sobrecarga, sobredescarga e curto-circuitos.

É protegida por um revestimento rígido e atende ao padrão UL2054 de segurança (LIT-TELFUSE, 2012). Em termos de autonomia, é possível realizar até 15 minutos de vôo. Tem dois conectores, um para corrente e outro para balanceamento de carga.

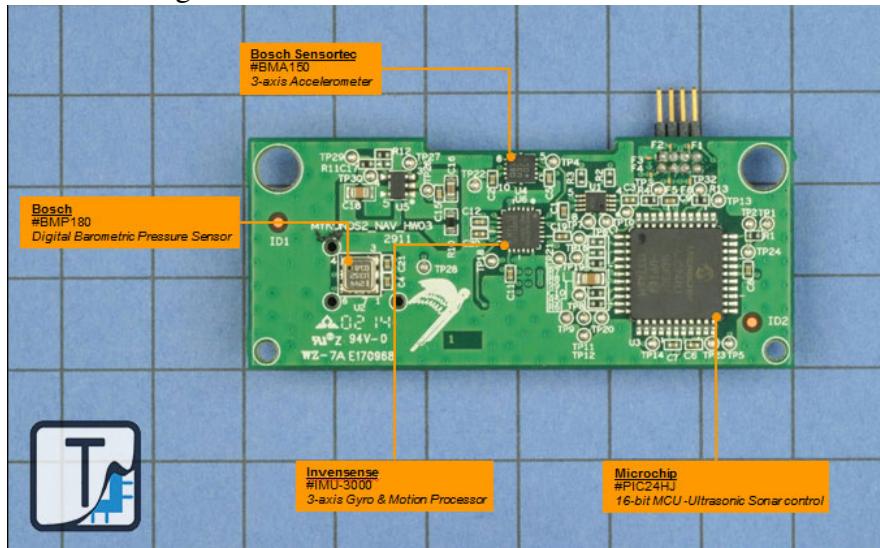
2.3 Câmeras

Há duas câmeras do tipo CMOS (Complementary Metal-Oxide Semicondutor) equipadas no drone, uma localizada na parte frontal e outra na parte inferior. Suportam transmissão ao vivo. A resolução da câmera frontal é de 640x480 pixels, VGA (Video Graphics Array), com um ângulo de visão de 93º. Já a câmera vertical tem resolução menor, de 176x144 pixels, QCIF (Quarter Common Intermediate Format), com ângulo de visão de 64º. A frequência da câmera é de 60 frames por segundo, embora o streaming seja feito ao padrão de 15 frames por segundo, sendo a taxa superior com o propósito de reduzir borrões na movimentação, otimizando assim o uso do vídeo para algoritmos de processamento de imagem. A câmera também é utilizada para estabilização horizontal, na função de *hovering* (em que ele plana no ar) e é capaz de estimar a velocidade do drone.

2.4 Sistema de navegação inercial

Em termos de navegação inercial, o drone provê 6 graus de liberdade, com medição dos ângulos *pitch*, *roll* e *yaw*. Com estes dados de navegação, é possível estabilizar o quadricóptero e controlar sua movimentação. A unidade de medição é um MEMS (Micro Electro-Mechanical System) que contém acelerômetro de giroscópio de 3 eixos.

Figura 2: Placa com os sensores de movimento



Fonte: <http://www.techinsights.com>

Acelerômetro

O modelo do acelerômetro é o BMA150 4, da Bosch Sensortec (BOSCH, 2008).

Ele é capaz de sensoriar inclinação, movimentação, vibração e choques em aparelhos móveis como celulares, interfaces homem-máquina, controladores de jogos e aspectos de realidade virtual. Seu dado de saída é a força G - aceleração relativa à queda livre - como quantidade de aceleração. Sua estrutura funciona de acordo com o princípio da capacidade diferencial, que mede a deflexão da massa de prova através de um sistema de suspensão. Dessa maneira, a saída é dada em termos de 3 eixos perpendiculares. Para obter a aceleração em movimento em relação à Terra, é necessário normalizar uma compensação de gravidade que o acelerômetro tem, corrigindo os efeitos causados pela rotação da Terra em relação ao quadro inercial do capacitor. Dessa forma, ao medir a gravidade, o acelerômetro também pode ser utilizado como sensor de inclinação.

Giroscópio

O drone possui um giroscópio de 3 eixos (INVENSENSE, 2010), medindo a velocidade angular em graus por segundo, baseado nos princípios do momento angular.

Para estimar o ângulo absoluto teta, a velocidade angular deve ser integrada em relação ao tempo. Os erros relativos a este sistema de giroscópios se dão devido aos erros sistemáticos das taxas de sinal, proporcionando um deslocamento do ângulo integrado com o tempo. O sistema utiliza o mesmo sistema MEMS dos acelerômetros para sentir as rotações do drone. As aplicações mais comuns deste componente são para controle de jogos, controles remotos para televisões smart, monitoramento esportivo e de saúde, reconhecimento de movimento e de gestos.

Barômetro

O sensor de pressão utilizado é o BMP180 (BOSCH, 2013), baseado na piezoresistividade, no qual o material semicondutor presente é capaz de variar sua resistência quando submetido a esforços mecânicos. As aplicações típicas deste componente são em ajuda a navegação por GPS, como estimativa de posição, navegação indoor e outdoor, esportes e lazer, previsão do tempo e indicação de velocidade vertical. Como vantagens, o componente possui baixo consumo de energia, baixa tensão, tornando-se vantajoso para utilização em aparelhos como celulares e aparelhos GPS, e sobretudo nos drones, os quais possuem baixa autonomia. Pode trabalhar em 4 níveis de consumo de energia, os quais produzem ruído de 0,25 metro, no nível mais preciso e oneroso, até 0,5 metro no nível mais econômico.

Ultrassom

O sensor de ultrassom (TEXAS INSTRUMENTS, 2009) realiza a medição de altitude, auxiliando na estabilização vertical, além de assistir o controle da velocidade vertical. Para tal, ele é colocado na parte inferior do drone, apontando para baixo a fim de medir a distância até o chão. O ultrassom é emitido pelo transmissor (400ST), bate no chão e reflete, voltando para o receptor (400SR). Dessa forma, o sistema é capaz de calcular a distância.

Devido ao tipo de princípio de medição e ao comprimento de onda utilizados,

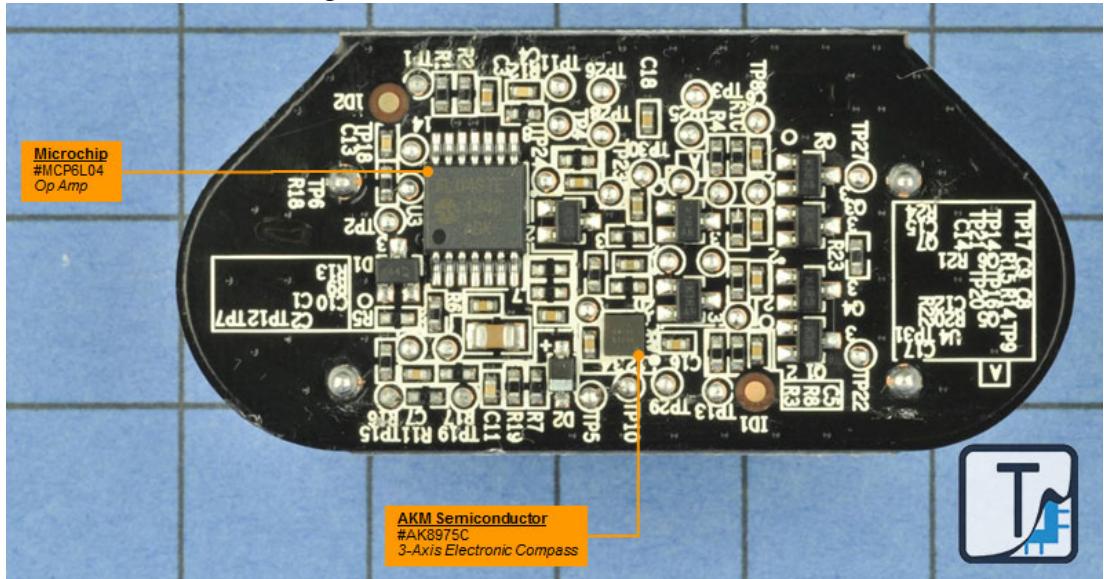
medidas realizadas com ultrassom são restritas a algumas faixas de valores. Estes, porém, ainda são maiores do que os requisitos do projeto pedem: O sensor ultrasom do drone tem um espectro de funcionamento entre 20 centímetros e 6 metros.

Em relação a reconhecimento de objetos, devido à propagação do som em um ângulo restrito, esta propagação é de maneira cônica, o sensor não é capaz de detectar exatamente pontos de profundidade, apenas uma região constante de profundidade. Deve-se ter atenção também para o tipo de superfície sobre a qual o drone irá voar, pois materiais que absorvem as ondas não irão refletir de volta, prejudicando a leitura de altura. Superfícies não-paralelas ao drone também podem ter uma medição incompleta ou ineficiente. Por fim, antes de uma onda ser emitida, a anterior deve ser detectada pelo sensor. Dessa forma, a velocidade mínima de operação do sensor de ultrassom é de 25Hz.

Magnetômetro

No módulo de ultrassom está acoplado também um circuito sensor magnético, AK8975 (ASAHI KASEI MICRODEVICES, 2010), que é uma bússola eletrônica de 3 eixos, que utiliza uma tecnologia de alta sensibilidade baseada em sensor de efeito Hall. Tal efeito é observado ao se inspecionar um condutor na medida em que um campo magnético perpendicular ao fluxo da corrente ser aplicado. Por consequência, cria-se uma diferença de potencial no condutor, a tensão de Hall, sendo esta proporcional à corrente e, também, à densidade de fluxo magnético.

Figura 3: Placa com ultrassom e bússola

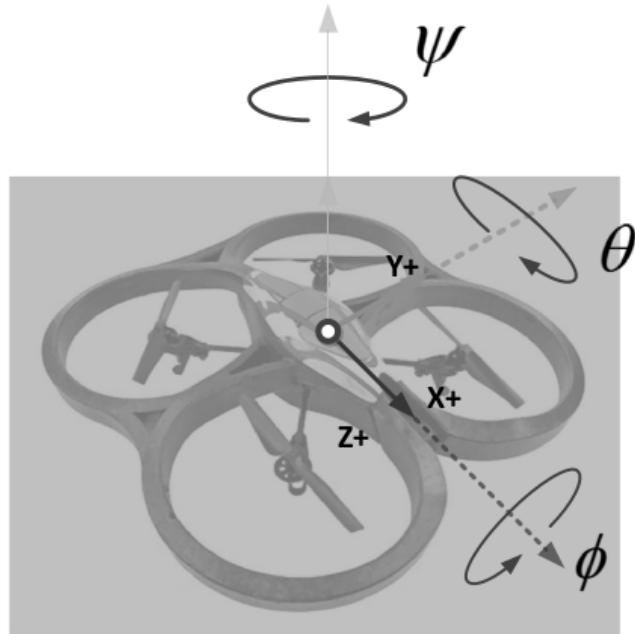


Fonte: <http://www.techinsights.com>

2.5 Movimentação do drone

A movimentação do drone é feita ao configurar individualmente a velocidade de rotação de cada um dos rotores. Considera-se uma formação em cruz em uma estrutura rígida de fibra de carbono. Sobre cada rotor está uma hélice. Rotores diametralmente opostos giram na mesma direção. Dessa forma, dois rotores opostos giram no sentido horário e os outros dois, no sentido anti-horário. Caso todos girassem no mesmo sentido, o quadricóptero ficaria girando sem controle; este par de rotações cancela o efeito e estabiliza o drone pairando no ar. (DIJKSHOORN, 2012)

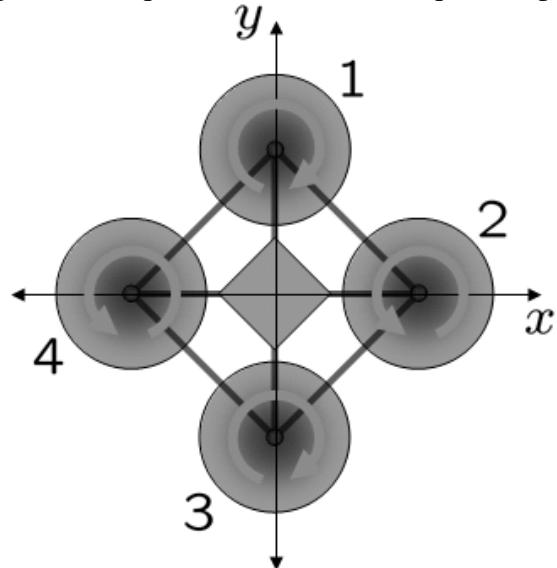
Figura 4: Eixos de rotação do drone



Fonte: KIRKEGAARD, Dennis et al

Veículos aéreos possuem liberdade para rotacionar em três eixos: *pitch*, no qual se inclina a frente do veículo para cima ou para baixo, imaginando um eixo de rotação nas asas de um avião; *roll*, o qual corresponde a uma rotação de um eixo passando pela traseira até a frente; e *yaw*, imaginando um eixo de rotação perpendicular ao drone. Por questões de conformidade e conveniência terminológica, utiliza-se neste texto os termos ingleses. Seguindo a imagem 4, considera-se *phi* como rotação em *roll*, *theta* como rotação em *pitch* e *psi* em *yaw*.

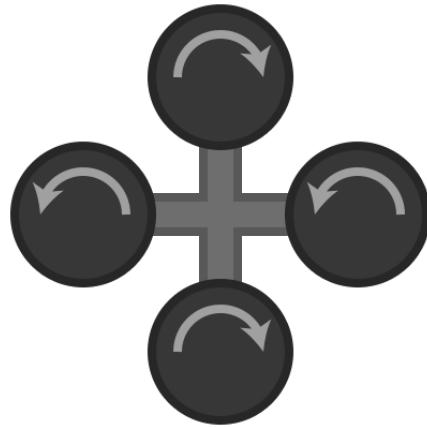
Figura 5: Torque em cada motor do quadricóptero



Fonte: HOFFMAN, Gabriel

No modo em que o drone paira no ar, todos seus rotores possuem a mesma rotação. Para elevar sua altitude, a rotação de todos os rotores é incrementada da mesma maneira. Similarmente, para diminuir a altitude, a rotação de todos os rotores é decrementada do mesmo valor, como mostrado na figura a seguir (QUADCOPTERARENA, 2015).

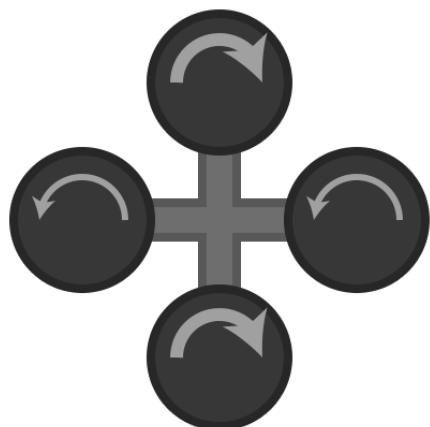
Figura 6: Torque em modo hover



Fonte: <http://quadcopterarena.com/>

Para rotacionar o drone em seu eixo z, se está lidando com o o eixo *yaw*. Para tal, deve-se aumentar a rotação em um dos sentidos - horário ou anti-horário - e decrementar no outro. Em outras palavras, o drone girará horizontalmente.

Figura 7: Torque para rotação em relação ao eixo z

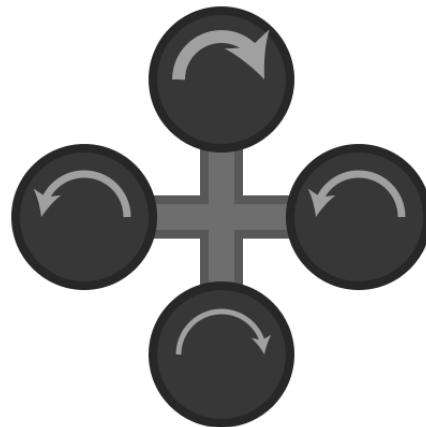


Fonte: <http://quadcopterarena.com/>

Para ajustar seus rotações em *pitch* e em *roll*, basta aumentar a rotação de um rotor

e decrementar a rotação do rotor diametralmente oposto. Tomando-se um rotor como frontal, para mudar a rotação em *roll*, configuram-se os rotores laterais, causando uma movimentação lateral. Já para a rotação em *pitch*, alteram-se as rotações do frontal e do traseiro, causando uma movimentação para frente ou para trás, sempre no sentido do rotor mais lento.

Figura 8: Torque para movimentação horizontal



Fonte: <http://quadcopterarena.com/>

3 LOCALIZAÇÃO INDOOR

No capítulo anterior, descreveu-se o sistema de sensoriamento do drone. Todos esses sensores contribuem para localização do drone em ambiente indoor, no qual o GPS não é capaz de atuar com eficácia, devido às barreiras físicas que atrapalham seu sinal e também à sua precisão limitada para o escopo de uso deste drone em ambientes fechados. Porém, a precisão de todos estes sensores também possui limitações. O ultrassom, por exemplo, irá considerar uma possível desnível causado por obstáculos em seu caminho como ponto de chão, e dessa forma, caso esteja configurado para voar a uma altura pré-determinada do solo, irá subir sua altura conforme reconhecer tal obstáculo e passar por cima dele. O giroscópio e acelerômetro também tem como elementos prejudiciais a instabilidade causada pela perturbação que o vento que o drone produz, que em ambiente fechado, acaba retornando através das paredes e solo e perturbando o próprio drone.

Dessa forma, se faz necessário um estudo das tecnologias possíveis de localização indoor, levando em conta não só a sua precisão, mas a complementaridade entre duas ou mais possibilidades, trazendo à tona um sistema ainda mais híbrido de localização indoor. E embora não faça parte do escopo do projeto, uma de suas utilidades é auxiliar no direcionamento do projeto em atualizações futuras. Uma destas possibilidades é a modularização destes métodos, ficando a critério do usuário escolher aquelas as quais se adequam mais a sua necessidade. Lista-se, então, separadas em tópicos de acordo seu tipo, as tecnologias estudadas e suas características.

3.1 Visão computacional

Sistemas de posicionamento indoor baseados em câmeras e a análise de suas imagens é amplamente utilizado em robótica móvel que, de alguma maneira ou técnica, se utiliza das características físicas ou artificiais do local para obter o posicionamento do item em questão. Seu uso se deve à possibilidade de um grau de precisão elevado em relação às demais técnicas, podendo ser da ordem de milímetros. Seu crescente uso acompanha a maior capacidade computacional, em relação a taxas de transmissão e evolução dos algoritmos de análise de imagem.

O uso de câmeras pode ocorrer de duas maneiras: a primeira é chamada de ego-motion, ou seja, os dispositivos de gravação de imagem são inerentes ao robô. No caso deste projeto, as câmeras do drone se enquadram nesta categoria. A segunda maneira é aquela na qual o ambiente possui câmeras em posição conhecida, de possível rotação porém de localização estática, mapeando a movimentação dos objetos ou dos artifícios técnicos. De qualquer forma, as arquiteturas baseadas em câmera estão relacionadas à obtenção do ângulo de chegada, AoA (Angle of Arrival), no qual cada pixel de um chip CMOS representa um ângulo de incidência vertical e horizontal.

Para obter informações de profundidade, uma câmera monocular - como as disponíveis no drone - necessita utilizar uma sequência de imagens, e a partir dela retirar este tipo de informação, de maneira similar a um sistema estéreo de visão, que conta com mais de uma câmera. Existem diversas técnicas e variações delas que procuram retirar do ambiente e seus objetos de maneira mais otimizada ou especializada as informações necessárias, dependendo do tipo de padrões e características geométricas e cromáticas. Outra maneira de obter profundidade é fundir a análise de imagem com sensores adicionais, como laser ou infravermelho. Tal adição, porém, vem com um custo relacionado ao peso adicional que isso causa ao sistema.

Técnicas de localização indoor baseadas em imagem estão sujeitas a variáveis não-

controláveis como iluminação e sombreamento, que estão passíveis a alterações ao longo do tempo e até aleatoriamente. A fim de reduzir os impactos destas características do ambiente e também de aumentar a robustez e a precisão do sistema, pode-se empregar o uso de marcadores artificiais. Dessa forma, a detecção de pontos é simplificada, uma vez que o padrão de marcador tem a possibilidade de ser visualmente mais simples e detectável computacionalmente; também é possível determinar uma escala para o sistema, uma vez que o tamanho do marcador pode ser especificado; e, por fim, múltiplos marcadores, cada um com seu código próprio, viabilizam a utilização de vários localizadores em locais diferentes do ambiente.

3.2 Infravermelho

Já foi visto que o uso de sensor infravermelho pode ser uma solução híbrida se integrada com um sistema de câmeras. Em relação a este último, a vantagem consiste em ser um sistema menos intrusivo, uma vez que seu comprimento de onda é invisível ao olho humano. As técnicas que utilizam este comprimento de onda serão separados em 3 categorias diferentes:

Beacons ativos: Beacons ativos consistem em posicionar receptores em locais conhecidos e mapeados, e um transmissor móvel, assim podendo realizar medições de distância entre ambos. Para conseguir uma precisão da ordem de metros, porém, é necessária uma configuração com muitos receptores, sendo só assim possível realizar o diferenciamento entre locais com mesmas características para o sistema. Em caso de paredes ou outros obstáculos opacos para a onda, o sinal se torna indisponível.

Radiação ambiente: Sistemas baseados na emissão natural da radiação infravermelha são chamados de passivos, uma vez que não é necessária nenhuma fonte artificial de calor. Dessa forma, com essa arquitetura de localização se possibilita a criação de uma imagem completamente passiva do ambiente ao redor do sensor.

Radiação artificial: Ao contrário dos sistemas baseados em radiação ambiente, esta técnica se utiliza de radiação artificial, por isso sendo chamada de localização ativa. Para tal, são comumente utilizadas câmeras infravermelhas baseadas em LEDs infravermelhos ou marcadores retro-reflexivos. O sensor de movimentos da Microsoft, o Kinect, utiliza uma projeção contínua de luz infravermelha para obter o sensoriamento 3D da movimentação das pessoas.

3.3 Laser

As tecnologias que utilizam laser são baseadas na aquisição de uma nuvem de pontos 3D, na forma de coordenadas polares. Estas coordenadas são posteriormente transformadas em coordenadas cartesianas. Como grande vantagem, está a precisão elevada, da ordem de milímetros até micrômetros, o que justifica o uso na indústria e em modelagem tridimensional. O contraponto se deve ao alto custo deste tipo de sensor, inviabilizando seu custo neste projeto. Seu emprego em projetos envolvendo veículos aéreos não-tripulados, porém, já é feito, como se pode ver em (BRY; BACH-RACH; ROY, 2012) sendo comumente empregado em técnicas de mapeamento e localização simultâneos, ou em inglês, SLAM, Simultaneous Localization and Mapping. Dessa forma, com esse tipo de sensoriamento se descarta a necessidade de conhecer previamente o ambiente, visto que não é preciso realizar uma fase de mapeamento prévia.

3.4 Ondas sonoras

A localização através de ondas sonoras é diferente das outras citadas, uma vez que utiliza ondas mecânicas, que precisam de um meio para se propagar, como o ar, ao invés de ondas eletromagnéticas, que não precisam deste meio. Para tal, os sistemas baseados nesta técnica realizam em sua maioria a multilateração, instalando pontos em paredes e teto e calculando a distância para o ponto móvel. Ondas sonoras são classifi-

cadas em 3 tipos: infrassons (frequência abaixo de 20Hz), ultrassons (acima de 20kHz) e os sons (entre ambas faixas, que é audível pelo ouvido humano). Costuma-se utilizar o ultrassom, porém é possível entrar na faixa dos sons audíveis pelo homem, sendo que uma pequena fração das abordagens se utiliza deste espectro. Como problema comum entre os métodos que se utilizam de ondas sonoras, está a interferência de outras fontes e também ruídos, além de problemas de escalabilidade. A taxa de atualização varia de 1 a 10 Hertz, porém nenhum destes métodos é comumente utilizado em drones voltado à localização, uma vez que este tipo de veículo causa muita perturbação sonora, sendo empregado somente na obtenção de altura do veículo.

Sistemas de dispositivos ativos: Este método utiliza dispositivos móveis que transmitem, ativamente, sinais sonoros. O pré-requisito para os pulsos ultrassônicos é que eles sejam precisamente programados entre os dispositivos, a fim de garantir a precisão do sistema. Ainda é possível, a troco de uma menor taxa de transmissão, obter um certo grau de escalabilidade de dispositivos, caso haja sincronização correta.

Sistemas de dispositivos passivos: Um conjunto de transmissores enviam sua localização para um receptor móvel. Esse receptor calcula sua localização com base nos dados enviados. É possível calcular a distância entre transmissor e receptor pela diferença de chegada dos sinais, sendo enviado um de rádio frequência (velocidade próxima à da luz) e do ultrasom (velocidade do som). Esse método permite escalabilidade de usuários sem risco de interferência de sinais.

Ecolocalização: Este método funciona de maneira similar aos animais que se comunicam e localizam por ondas, como os morcegos. Devido à assinatura da sequência de ondas, é possível identificar o eco que a onda transmitida produz, dessa forma determinando a localização do objeto ou obstáculo do ambiente. A diferença deste para os outros métodos consiste na ausência de beacons, tags ou demais dispositivos auxiliares.

3.5 Radiofrequênciа

Por radiofrequênciа, entende-se da faixa de frequênciа no espectro de 3 kHz a 300 GHz, ou seja, a utilizada por ondas de rádio. No âmbito da localizaçao, os fatores relevantes destas ondas eletromagnéticas são a identificaçao do objeto e do sensor e a medição de sua distância através das características da onda, como potênciа de sinal. Cita-se como exemplo o Bluetooth e etiquetas RFID (RadioFrequency IDentification). Tais tecnologias tem seu alicerce em medidas de características da onda para criar um mapeamento do ambiente. Este mapeamento, porém, é passível de flutuações ao longo do tempo, por mudanças físicas do ambiente como, por exemplo, uma nova configuraçao de móveis ou movimentação humana. Assim sendo, são necessários ajustes matemáticos para obter dados consistentes, conseguindo obter precisões abaixo do desejado para a aplicação presente. Outros fatores complicantes são a demora do sistema para obter a localizaçao, e também o alto custo caso se deseje uma rede sensorial mais numerosa.

3.5.1 Identificaçao por radiofrequênciа

Comumente chamados pela sua sigla, RFID, constituem-se de tags, ativas ou pas-sivas, as quais possuem cada um uma chave única, e quando próximas a um leitor, são identificadas. Seus principais usos estão em controle de estoques, de animais e na identificaçao de automóveis. Nesse tipo de aplicações, o uso do RFID se dá apenas com a proximidade o suficiente para que a tag seja lida, não sendo importante um pa-râmetro preciso de localizaçao. Mesmo assim, é possível utilizar RFID para tal fim, porém mais uma vez esbarrando nas limitações de tecnologias similares, limitações essas como precisão abaixo da desejada, sendo necessários muitos sensores para au-mentar a precisão e variaçao da intensidade do sinal recebido. Existem, basicamente, dois tipos de tag: ativa e passiva. A primeira tem necessidade de uma bateria para transmitir seu sinal, enquanto a última utiliza a energia própria da onda recebida para

retransmitir seus dados, portanto sem necessidade de uma bateria.

3.5.2 WLAN

Redes sem fio locais também podem ser utilizadas para fins de localização indoor. Sua difusão em quase todo tipo de ambientes facilita, já fazendo parte da infraestrutura dos locais. O problema desta tecnologia, porém, se dá no que tange ao nível de precisão: no mínimo 1 metro de inacurácia. Além disso, as abordagens desta tecnologia para o fim de localização está ainda a nível de pesquisa. No contexto atual em que se insere, este tipo de localização não pode ser utilizada como a principal para a aplicação deste projeto, sendo apenas possível utilizá-la como secundária de maneira híbrida.

3.5.3 ZigBee

Outra alternativa estudada é o padrão de tecnologia sem fio ZigBee, cuja grande vantagem se baseia no seu baixo consumo energético, além da baixa complexidade computacional. O ZigBee se baseia na camada física e controle de acesso médio definido no padrão IEEE 802.15.4, sendo considerada uma WPAN (Wireless Personal Area Network), tendo seu uso mais frequente em redes de sensores utilizando microprocessadores, aplicações embarcadas. Pode ser utilizada em qualquer tipo de topologia de rede. Um exemplo de projeto de localização baseado em Zigbee está em (HERNANDEZ et al., 2009) Caso se considere o uso de uma placa microprocessadora com o drone, a comunicação pode ser feita através do ZigBee, e esta pode ser uma solução alternativa também para a localização, a troco do peso que este conjunto traria a mais para o quadricóptero sustentar.

3.5.4 Bluetooth

Outra alternativa de comunicação e localização, como o Zigbee, é o Bluetooth. Esta tecnologia tem níveis de operação de energia, podendo então ser escolhida de

acordo com as necessidades do projeto. Possui como vantagem o fato de ser uma tecnologia de alta segurança em relação as suas similares, e como desvantagem latência da ordem de segundos, o que na prática pode impossibilitar uma aplicação como a localização de um robô. Ainda pesa contra o Bluetooth - e também o ZigBee - o fato que o drone já possui um protocolo WLAN de comunicação definido, com portas especificadas para tipos de dados diferentes, o que acaba por ter grande peso na decisão das tecnologias utilizadas.

3.5.5 Célula de origem

Este método é capaz apenas de identificar o ponto com maior intensidade de sinal recebido e dessa forma localizar o dispositivo na posição deste ponto. Portanto, quanto mais pontos mais precisão se consegue. É evidente notar que tal abordagem se torna onerosa no nível de precisão aceitável para um drone em ambiente indoor.

3.5.6 Reconhecimento de padrões

A técnica de reconhecimento de padrões, também conhecida como *fingerprinting*, é composta por duas partes: online e offline. Na fase offline, acontece o mapeamento do ambiente através das características da onda eletromagnética nos diversos pontos (pré-determinados) do local. Essas são as fingerprints. Dessa forma, o mapa de sinais de rádio é criado. Os parâmetros da onda comumente utilizados são: RSS - Received Signal Strength; ToA - Time of Arrival; e AoA - Angle of Arrival; respectivamente, a força do sinal recebido, o tempo que uma onda eletromagnética demora para chegar ao receptor e o ângulo que a onda forma em relação ao emissor. Na fase seguinte, online, é utilizado algum método que ofereça uma estimativa da localização do objeto através dos dados da onda eletromagnética, de acordo com o mapa previamente confeccionado.

Tal estimativa é feita por algoritmos apropriados. Cabe uma descrição sucinta de

cada um deles:

Probabilístico: Assume-se n candidatos a posição L₁, L₂, ..., L_n e s é a força do sinal observada na fase online. A regra de decisão é: Escolher L_i se a probabilidade de L_i ter força de sinal s é maior que para L_j, sendo j = 1, 2, ..., n e j diferente de i. Se aplica apenas a candidatos a posição discretos.

Vizinho k mais próximo: Baseado no radio map criado na fase online, utilizam-se os valores de RSS para procurar a posição k mais próxima. Podem ser utilizadas variáveis de peso, de acordo com a relevância de determinados pontos de referência.

Redes neurais: assumindo-se que as fingerprints são matematicamente complexas para serem analisadas, pode-se utilizar este método, o qual consiste num conjunto de valores de entrada com um peso associado; somam-se todos estes valores de entrada e uma função de ativação calcula o valor de saída. Por sua vez, este valor está unicamente associado a um valor medido durante a fase online.

SVM - Support Vector Machine: é um método de sensoriamento virtual utilizado também em processos contínuos na automação industrial que serve para um conjunto de métodos de aprendizado supervisionado. Analisam-se os dados e reconhecem-se padrões. A entrada é um conjunto de dados que é classificado binariamente, não probabilístico, através de hiperplanos em um gráfico, que dividem os dados dicotôneamente, facilitando sua classificação e regressão.

SMP - Symmetric Multi Processing: Utiliza os valores de RSS para procurar o candidato a posição para cada transmissor separadamente. Escolhe as N entradas de dados mais próximas, em termos de força de sinal, e estima a posição baseado na média de coordenadas destes N pontos

3.5.7 Banda ultralarga

De acordo com a FCC (Federal Communications Comission) norte-americana, o nome de Banda Ultralarga (Ultra Wideband) é designado a tecnologia de radiofrequência cuja largura de banda é maior do que 500 MegaHertz ou mais do que 25% de sua largura central. Os diferenciais da banda ultralarga consistem na sua imunidade à propagação por vários caminhos - ou seja, pode se propagar através de paredes e outros tipos de obstáculos - e pela precisão da ordem de centímetros, como em (LABS, 2015). Para usar este sistema, é necessário instalar dispositivos auxiliares nas paredes do ambiente a fim de calcular a posição. Este ainda é, porém, um protótipo e não está no mercado. Dentre as tecnologias similares, como WLAN, Bluetooth ou RFID, a banda ultralarga surge como uma alternativa que se sobressai pela precisão e imunidade à interferências físicas e de outras ondas. Além disso, não sofre interferência das demais faixas e tecnologias sem fio, além de proporcionar um nível de segurança e menor consumo que tecnologias como Bluetooth, por exemplo.

3.6 Magnetismo

É possível obter a localização de um sistema através de campos magnéticos, sejam eles naturais ou artificiais. Tais campos podem ser provenientes de ímãs ou através do controle de corrente elétrica. Da mesma maneira que nos métodos como Bluetooth, beacons e demais similares, é possível utilizar o método de reconhecimento de padrões para localizaçāo, uma vez que cada ponto do espaço possui uma unicidade magnética em relação aos demais. Para aplicar tal método neste sistema, porém, é necessário se considerar a influência magnética que os motores do quadricóptero causam no ambiente e, portanto , no sensoriamento deste. A vantagem deste tipo de tecnologia em relação aos demais consta na não necessidade de ter uma linha de visão entre o sensor e o objeto de medição, uma vez que os campos magnéticos não são atrapalhados por obstáculos como paredes. Em ambientes indoor, uma vez que a disposição dos itens é

mais constante, é uma tecnologia passível de uso para localização.

3.7 Conclusões e comparações

As possibilidades de implementação de localização indoor utilizando radiofrequência tem como vantagem um custo reduzido de implementação, visto que pode se utilizar da infraestrutura já existente em edifícios, como roteadores. Há grande gama de estudos em relação a essas tecnologias, impulsionada pela popularização dos microcontroladores como Arduino e Raspberry Pi, os quais utilizam radiofrequência para comunicação e sensoriamento.

Em contrapartida, o ambiente indoor proporciona muitas barreiras, como paredes e obstáculos, contribuindo assim para reflexões e atenuações do sinal. Além disso, interferência de outros sinais pode prejudicar o funcionamento do sistema. Porém, a desvantagem mais decisiva se refere à falta de um modelo preciso o suficiente para a força dos sinais. A variação ocorre devido a diversas variáveis alheias ao sistema, como, por exemplo, a movimentação humana dentro de um prédio, que faz com que o sinal sofra mudanças em pontos do ambiente, causando assim uma estimativa imprecisa sobre a localização corrente do drone. Para driblar tais complicações, cita-se a banda ultralarga, um candidato em potencial para aplicações no ramo.

Comparativamente, os métodos de localização indoor através de análise de imagem são mais precisos do que os demais. Neste tipo de ambiente fechado, como prédios, salas e corredores, parte-se da premissa que o espaço de vôo é reduzido, sendo então a precisão um requisito prioritário para o projeto. Pesa a favor, também, o fato do drone já possuir duas câmeras monoculares integradas, sendo possível utilizar técnicas de localização monocular ou, alternativamente, adicionar tecnologias no sentido de prover sensores de profundidade que facilitem a estimativa de pose e também de mapeamento do drone quanto ao mundo em que está inserido - ao custo, em contrapartida, de um maior peso do drone e consequente menor tempo de vôo.

De modo a escolher qual ou quais tecnologias melhor se enquadram no escopo e tempo disponível para o projeto, um estudo detalhado deste tipo de tecnologia visual é descrito no próximo capítulo.

4 VISÃO COMPUTACIONAL

A utilização de câmeras, sejam elas onboard ou não, são recorrentes em projetos utilizando drones ou outros tipos de robôs móveis. O poder de processamento atual e de transmissão de dados já possibilita o uso deste tipo de sensores para localização a taxas adequadas.

Em ambientes outdoor, é possível empregar o recurso de GPS, porém ainda assim há locais em que seu sinal pode não estar disponível, abrindo espaço para explorar adaptações dos métodos de localização e mapeamento. Um exemplo motivador para este projeto se encontra em (ALBERS-SCHOENBERG, 2013), no qual o drone é capaz de se localizar nas ruas ao cruzar a imagem advinda da câmera do drone com as imagens do banco de dados do Google Street View. Alguns trabalhos ilustram técnicas tanto de localização quanto de mapeamento simultaneamente, como em (SHEN; MICHAEL; KUMAR, 2011), no qual são acoplados sensores de profundidade - no caso, lasers - que possibilitam e facilitam o mapeamento medindo distâncias.

O projeto contará com dois setores distintos da computação gráfica: a análise e processamento da imagem, para realizar a localização através de marcas do ambiente; e a síntese de imagem, para desenvolver um simulador para o drone e também criação de mapas e caminhos. Uma vez definida a tecnologia principal de localização indoor, são descritas as técnicas, metodologias e o conhecimento necessário para sua implementação no projeto.

4.1 Definições básicas

Define-se por imagem como uma função bidimensional nas quais x e y são coordenadas espaciais que, em cada ponto definido por um par, especifica a intensidade da imagem neste determinado ponto. Havendo a discretização de todos estes pontos da imagem, esta é considerada digital. Através da sequência destas imagens captadas pela câmera, forma-se um vídeo. Cada frame desta sequência é analisada e também a variação entre frames consecutivos, obtendo assim informações sobre a movimentação e localização do drone. Para tal, técnicas procuram por pontos característicos, ou *feature points*, sendo estes pontos e conjuntos deles naturais ou artificiais.

4.2 Transformações geométricas

4.2.1 Coordenadas homogêneas e matrizes relacionadas

Ao se trabalhar com visão computacional, pela facilidade algébrica que o método proporciona, são utilizadas coordenadas homogêneas ao invés das cartesianas da geometria euclidiana. Dessa forma, as transformações projetivas são mais facilmente representadas, como rotação, translação, escalamento, devivo a sua linearidade.

Os dados provenientes da câmera pinhole produzem uma matriz 3x4, descrevendo o mapeamento desta câmera num mundo tridimensional para uma imagem bidimensional. Porém, apenas com esta matriz ainda não se sabe sobre a geometria interna da câmera e nem sobre sua posição. Para contornar essas questões, decompõe-se esta matriz em outras duas: a matriz intrínseca e a extrínseca.

A matriz intrínseca é uma matriz 3x3, triangular superior, cuja função é descrever os parâmetros internos da câmera: distância focal, ponto principal, além da distorção das lentes. Já a matriz extrínseca pode ser dividida em duas partes: a matriz de rotação e a matriz de translação (que é um vetor), especificando a transformação entre os frames de câmera e de mundo, sendo o resultado da utilização dessas matrizes a posição

nas coordenadas da câmera.

4.2.2 Calibração da câmera

A calibração de câmera consiste no procedimento de determinação dos parâmetros da câmera através de um processo de captura de um objeto de referência em diversos ângulos e posições diferentes. No mínimo 6 pontos são necessários para três dimensões, porém um número maior de pontos proporcionará uma calibração mais precisa. Tais capturas fornecerão coordenadas para resolver um sistema linear e então obter a matriz de transformação da câmera.

4.3 Extração de características

Ou *feature extractions*, significa a extração de informações relevantes dentre uma grande quantidade de dados, buscando sua unicidade e não-redundância, reduzindo a dimensão do sistema. Ou seja, busca-se extrair pontos ou regiões relevantes de uma imagem para interpretação posterior, que podem ser buscando cantos, lados, cor, brilho, que os destacam dos demais pontos da vizinhança(NIXON; AGUADO, 2002a). Os métodos devem levar em consideração condições de iluminação, além dos ruídos provenientes da captura das imagens em condições adversas. A seguir, os principais tipos de extração de características

4.3.1 Características de baixo nível

Características de baixo nível são aquelas cujas singularidades são básicas, extraídas diretamente da imagem sem a necessidade de informações espaciais a mais, que também podem ser utilizadas para operar nas técnicas de mais alto nível. As linhas da imagem, ou seja, sequências contínuas de pixels com a mesma intensidade, são o tipo mais primitivo de extração de informação.

Um nível acima, estão as curvas e cantos da imagem, que definem mudanças brus-

cas de direção nas linhas ou graus de curvatura elevados. O fluxo ótico também é um tipo de extrator de características de baixo nível, o qual possibilita evidenciar quais pontos possuem maior movimento em relação aos frames anteriores. Todas estas técnicas de extração darão pontos singulares, que criam um conjunto de pontos específicos caracterizando a imagem, como se queira, os objetos de interesse que ela possui.

4.3.2 Extração por correspondência de forma

Extrair informações da imagem por correspondência de forma consiste em procurar contornos, formas geométricas, modelos e cores, podendo-se variar a posição, orientação e escala do item de interesse. Para tal, busca-se propriedades invariantes nesta imagem. É comum utilizar o método de *thresholding*, o qual transforma uma imagem em escala de cinza em binária, ou seja, cada pixel da matriz é ou preto ou branco. Esse procedimento auxilia a detecção de linhas e contornos, uma vez que evidencia diferenças de brilho.

O nível de *threshold* pode ser alterado, fazendo com que um determinado valor de cinza seja o limite entre os que abaixo serão brancos e os acima, pretos. Outras técnicas podem ser aplicadas antes do *thresholding* para diminuir o ruído e qualificar mais a imagem em relação a ruídos.

Mais uma vez, as técnicas de baixo nível podem ser aplicadas para comparar informações e confirmá-las ou não, contribuindo ou não para o acerto entre template e imagem a ser analisada. Templates que são invariantes quanto à posição são mais facilmente detectáveis, embora ainda assim seja possível, a um custo computacional maior, detectar invariantes quanto à rotação e ao tamanho.

Outra técnica que deteca formas é a **transformada de Hough**, que é capaz de identificar formas geométricas parametrizáveis através de uma matriz chamada acumulador, onde objetos candidatos são obtidos ao verificar pontos de máximo local. Quando não é possível ter um template ou não é possível modelá-lo com precisão

o suficiente, são necessárias técnicas que extraem formas **flexíveis** das imagens, por deformação, simetria ou estatisticamente.

4.3.3 Descrição de objetos

Após a detecção de características, pode-se obter descritores destes. Coleções de pixels representam objetos em uma imagem. Para reconhecer estes objetos, é necessário descrever essas propriedades das coleções em números, chamados descritores de objetos. Basicamente, segundo deve-se comparar objetos reconhecidos às imagens. Para o bom funcionamento de um descritor, deve-se respeitar quatro propriedades(NIXON; AGUADO, 2002b):

Definir um conjunto completo Objetos devem ter o mesmo descritor se e somente se eles compartilham a mesma forma;

Congruência Objetos similares devem ter descritores similares;

Propriedades invariantes Descritores invariantes em relação a tamanho, rotação e posição devem ser úteis para identificar os objetos com respectiva variação;

Conjunto compacto Um descritor deve ser conciso ao representar um objeto;

Dessa forma, deve-se procurar cuidadosamente características dos objetos de maneira a particularmente identificar cada um dos objetos distinta e separadamente.

4.4 Marcadores artificiais

Marcadores artificiais são úteis na medida em que facilitam o processo de extração de características do ambiente. Eles podem ser únicos ou genéricos, sendo que com a primeira opção, a unicidade de cada marcador facilita o reconhecimento do local em questão. Existem diversos tipos de marcadores, e ao escolher se deve levar em conta a quantidade de marcadores que será necessário, pois um número maior requer

marcadores com códigos de identificação mais complexos e, portanto, mais difíceis de identificar.

Também há de se considerar o impacto visual que estes causarão no ambiente, visto que a poluição causada pode prejudicar aspectos na camada de negócio, principalmente. Para contornar este problema, marcadores visualmente agradáveis podem ser utilizados, com o contrapeso da maior complexidade e dificuldade de leitura destes.

4.5 Sistemas monoculares e estéreo

Sistemas monoculares são aqueles os quais possuem apenas uma câmera como sensor; por outro lado, sistemas estéreo são classificados como aqueles que possuem duas ou mais câmeras ou sensores de profundidade. Estes últimos, pela maior capacidade sensorial, produzem mais saídas e informações sobre o ambiente. Porém, em (DAVISON; MONTIEL; STRASDAT, 2012) e (SHRIDHAR; NEO, 2015), há a implementação de um localização e mapeamento simultâneos através de uma câmera monocular, característica comum aos sistemas que naturalmente detectam profundidade.

A tomada de decisões quanto à escolha da técnica de visão computacional passa pela escolha entre as diversas abordagens e combinações entre câmeras monoculares e estéreo e os algoritmos de reconhecimento, mapeamento e localização, levando-se em conta estes fatores, juntamente com a verificação das bibliotecas de processamento de imagem disponíveis.

4.6 Bibliotecas

Para o trabalho com processamento de imagens, a principal tecnologia é o **OpenCV**. É uma biblioteca feita especialmente para processamento de imagens. Porém, sua implementação em baixo nível e complexidade é um fator a ser ponderado

em relação ao tempo de projeto.

Já o **ARToolKit** é uma biblioteca de mais simples utilização e possibilidade de suprir as necessidades do projeto com maior rapidez de familiarização, instalação e entendimento. Esta biblioteca é feita para aplicações na área de realidade aumentada, portanto sua técnica se baseia em marcadores, dessa forma possuindo funções próprias para tal. O ARToolKit é utilizado neste projeto, sendo seus testes documentados em 8.5.1.

Tanto para implementação da localização quanto para simulação, uma alternativa popular e de grande número de bibliotecas baseadas é o ROS (Robot Operating System)(ROBOTICS, 2015), uma plataforma open source que reúne bibliotecas e ferramentas para operação de robôs, sendo diversas destas bibliotecas relacionadas ao AR.Drone, como o tum ardroner(ENGEL; STURM; CREMERS, 2014), que utiliza PTAM (Parallel Tracking and Mapping) e em (MONAJJEMI, 2014)

Ainda se pode citar o uso de noje.js para controlar o drone (GEISENDÖRFER, 2012), e também uma biblioteca que atua acima dela, controlando o drone em nível de execução de missões (ESCHENAUER, 2013).

4.7 Filtro de Kalman

Sistemas dinâmicos lineares, como o caso da estimativa da localização de um drone, podem se utilizar do filtro de Kalman. Consiste em um processo recursivo de maneira que as novas medidas são processadas na medida em que chegam. É conveniente ao drone, pois o análise é online em tempo real, ou seja, as medições advindas do sistema de navegação inercial e das câmeras, que tem um delay estimado, podem contar com o filtro para uma estimativa da posição do drone nos frames seguintes. O estado de um sistema dinâmico determinístico é o menor vetor que resume o passado do sistema na sua totalidade. O conhecimento do estado permite, em teoria, a previsão

da dinâmica futura e saídas do sistema determinista. Se todo o ruído é Gaussiano, o filtro de Kalman minimiza o erro quadrático dos parâmetros estimados.

Porém, para sistemas não lineares como do drone, é possível utilizar o Filtro de Kalman Estendido. Apesar de não ter a mesma otimização de um processo linear, continua sendo o método utilizado em grande parte dos sistemas dinâmicos, como os ligados a robótica móvel. As medições obtidas de vários sensores diferentes servem de entrada para o filtro, que deve ser modelado para obter saídas que possam predizer a pose do drone com precisão. É comum que as bibliotecas que trabalham com estimativas de posição possuam implementações do filtro de Kalman, combinadas com o controle PID quando relacionadas ao drone.

4.8 Escolhas

Para fins de escolha das tecnologias utilizadas, após pesquisa aplicada e colaboração de especialistas, levando-se em conta os fatores de complexidade - de instalação, curva de aprendizado, necessidade de módulos paralelos e compatibilidade com máquina e drone - de acordo com o tempo de projeto disponível, optou-se pelo ARToolKit como biblioteca de localização indoor, o que implica em utilização de marcadores artificiais no ambiente que são lidos pelo processamento de imagem e então provêm informações sobre posição. Para controle do drone, será utilizado o ardrone-autonomy não baseado em ROS, uma vez, também, que sua implementação apresentou problemas nas máquinas.

5 ESPECIFICAÇÃO

Por se tratar de um projeto que envolve diversas áreas de conhecimento e pesquisa, faz-se mais do que necessária a definição e especificação de requisitos, a fim de delimitar as fronteiras do projeto e poder adequá-lo ao cronograma e prazos, com o objetivo de atingir resultados tangíveis e que satisfaçam os objetivos. É interessante, então, separar os requisitos de acordo com as diferentes partes do projeto. Tais parâmetros tem como serventia conseguir classificar e comparar os métodos de localização indoor e guiar quais são os parâmetros relevantes para o tipo de aplicação que se deseje utilizar. Muitos dos requisitos são guiados pelas próprias especificações do drone, uma vez que outros requisitos como autonomia e peso são preponderantes para a aplicação necessária.

5.1 Requisitos de localização indoor

Precisão: Quando se remete a localização em ambientes fechados, precisão é um dos primeiros questionamentos. Uma vez que há paredes, móveis, além do movimento de pessoas ou até outras máquinas, tudo isto em espaços de poucos metros. Ao se falar de robótica móvel, métodos de localização de um robô retornam uma função probabilística no espaço, portanto a incerteza na medição é um requisito importante a se considerar. As dimensões do quadricóptero utilizado para o projeto estão na ordem de 30 centímetros. Considerando-se a sua susceptibilidade a rajadas de vento que ele próprio produz e que refletem nos objetos, além da instabilidade devido a desbalance-

amentos em seus motores e hélices, pondera-se que o drone deve retornar parâmetros de localização com incerteza de medição da ordem de no máximo 0,5 metro.

Autonomia: A autonomia se torna um fator limitante na medida em que os drones em geral possuem um tempo de vôo reduzido, devido à capacidade de suas baterias e também do grande peso, proporcionalmente, que elas possuem em relação ao montante total. Ademais, por medidas de segurança e também para poder avisar previamente ao usuário, o drone utilizado indica o nível de bateria crítico em seu log e então pousa, procurando assim evitar acidentes. De acordo com os testes, as tarefas passadas ao drone devem ter duração máxima de 10 minutos.

Taxas de atualização e latência: Os sensores do drone, como a câmera, o sistema inercial de navegação, o altímetro e os demais, além da taxa de comunicação com o servidor são diferentes e influenciam na qualidade da automação e localização do quadricóptero. Tais frequências são definidas pelo fabricante do drone e são respeitadas e levadas em conta no projeto.

Área de cobertura: A cobertura do sistema é limitada ao alcance da rede wireless criada pelo drone. Os testes feitos utilizam como ambiente, majoritariamente, salas de aula da Escola Politécnica da USP, cujas dimensões de lado são da ordem de 10 metros, e não foi observada nenhuma perda de sinal devido a baixa potência. Então, classifica-se a área de cobertura como local, limitada a uma sala, embora seja possível, em extensões do projeto, promover escalabilidade do sistema através de redes de roteamento até o servidor, ou até mesmo ter área de cobertura global, caso seja acoplado um módulo de conexão (como 4G) ao drone e o envio de seus dados seja através dele.

Peso: A propulsão dos rotores é o item mais custoso em termos de consumo de energia no drone. Quanto mais peso o drone tem, mais bateria irá consumir para manter o quadricóptero em vôo e menos autonomia ele tem. Portanto, embora seja possível manter o drone com anexação de módulos, qualquer adição significa uma perda em termos de tempo de vôo.

Custo: Associado ao peso, está a noção de custo do sistema. Além do custo em termos de energia e autonomia, a não adição de novos módulos ao drone também impacta no custo financeiro do projeto.

Saídas do sistema: Dentre toda a gama de informações advinda do módulo de comunicação do drone, algumas de fato interessam para a localização. São elas os vídeos produzidos, as saídas dos sensores de altitude, ultrassom e também os dados de navegação inercial. É importante diferenciar as saídas do sistema e as saídas esperadas da interface homem-máquina. Basicamente, são desejadas as coordenadas e a orientação em um espaço tridimensional.

Privacidade: Tratando-se de ambientes indoor e do tipo de aplicação os quais o drone pode executar, parte-se da premissa que a privacidade nestes ambientes já é passível a controle por sensores como câmeras, movimento ou calor, não sendo locais que invadam a privacidade pessoal, como estabelecimentos comerciais, escritórios ou indústrias. Dessa forma, permite-se a utilização de imagens

Infraestrutura: Para a localização indoor, serão necessários marcadores externos impressos em folhas de tamanho A4, afixados em paredes e suportes verticais, de modo que sempre haja um ou mais marcadores na visão da câmera.

Robustez: O drone deve operar sempre com o casco indoor, a fim de minimizar os danos em casos de batidas ou quedas. Em caso de comportamento estranho pós-choque, realizar procedimento de calibração.

Aspectos jurídicos: Em relação à legislação, não há definições claras sobre o que pode ou não ser feito com VANTs. Em geral, há recomendações de aeromodelismo (ANAC, 1999), que se resume a pilotar distante de áreas densamente povoadas, ter segurança e autorização ao pilotar próximo à pessoas, evitar áreas impróprias para isso, como hospital ou zonas de decolagem, e voar abaixo de 400 pés (aproximadamente 121,92 metros).

Na legislação, aparelhos com até 25 quilogramas podem ser classificados como aeromodelos. Acima disso, é classificado como aeronave comum. Caso este limite seja ultrapassado, há maior risco, podendo pegar até 5 anos de prisão para o infrator (TERRA, 2015).

A legislação não enquadra o uso de drone neste projeto. É um projeto exclusivamente em ambiente interno e controlado, previamente simulado. Eventuais mudanças na legislação reguladora após o projeto não serão consideradas.

5.2 Requisitos da interface

Informação mostrada: A interface deve mostrar um mapa do local, incluindo suas paredes internas. Dentro deste mapa, deve mostrar o drone em sua localização estimada. De modo a facilitar a visualização, a localização será vista de cima num mapa em duas dimensões. A dimensão em z, altura do drone, é omitida na interface, porém sua medição é realizada da mesma maneira que as outras dimensões. A interface, dessa maneira, deve manter um aspecto limpo e de fácil familiarização.

Botões de comando: O usuário dispõe de botões de controle do drone, para movimentação. Os botões de aterrissagem (land) e decolagem (take off) são coloridos e destacados dos demais, de maneira que o usuário consiga visualmente os localizar em caso de urgência de comando. Os comandos podem ser dados também através do teclado.

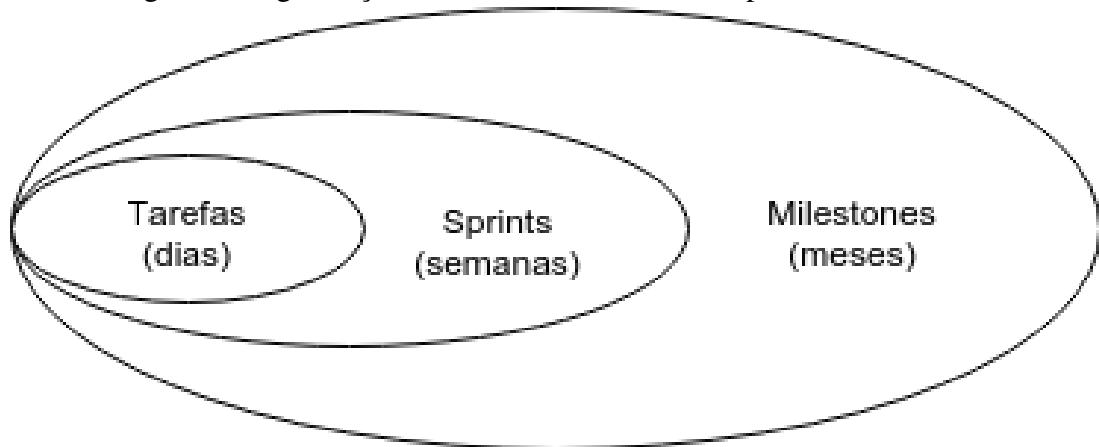
Rota ao clique: A interface deve permitir ao usuário que ele ordene um destino através do clique do cursor em cima de um determinado ponto do mapa.

6 PLANO DE TRABALHO

6.1 Metodologia

O projeto se desenvolve buscando integrar processo ágil com planejamento utilizando o PMBoK (Project Management Body of Knowledge). Para isso, organiza-se as atividades ao longo do projeto em tarefas, sprints e milestones.

Figura 9: Organização de trabalho, com tarefas, sprints e milestones.



Fonte: Autor

Milestone

Antes do início da execução das tarefas em si, é feita uma reunião da equipe se reúne com o intuito de planejar e traçar objetivos específicos no período de dois meses de trabalho. A reunião busca alinhar os membros, evitar redundância de trabalho e favorer o paralelismo e cooperação. Divide-se o trabalho em 5 milestones:

Relatório com objetivos, aspectos conceituais e especificações: O primeiro mi-

ilestone foca na área de pesquisa. Tal pesquisa busca contextualizar a equipe, verificar quais são os problemas de design e prover embasamento teórico, além de fornecer informações sobre as tecnologias aplicadas.

Simulação e localização indoor: Uma vez definido o escopo do projeto, este milestone tem como objetivo principal o desenvolvimento de um simulador para testes, aumentando a produtividade dos testes. Paralelamente, estudam-se as tecnologias de localização indoor de maneira mais incisiva do que no primeiro milestone, com o objetivo de selecionar quais são as tecnologias possíveis dentro das especificações.

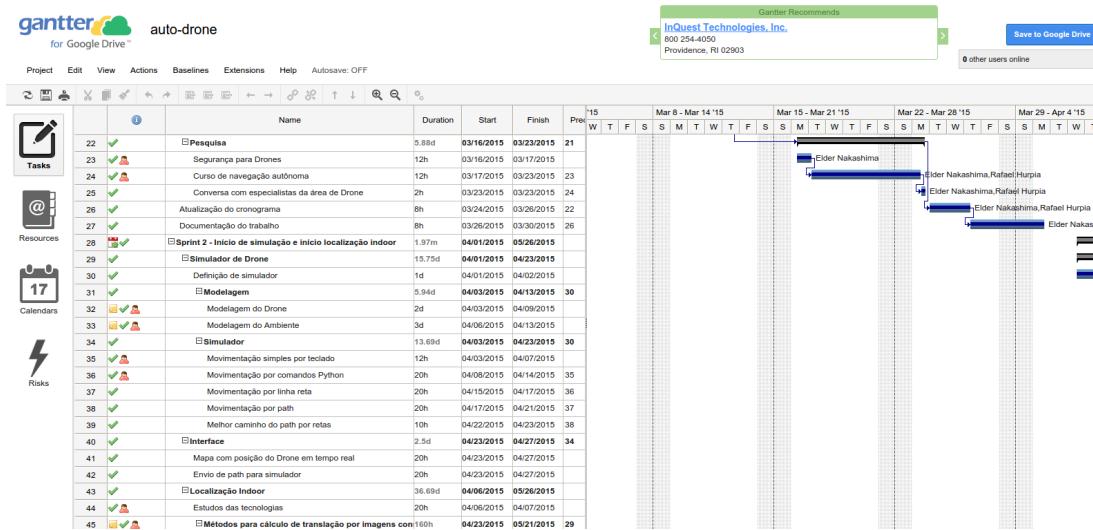
Visão computacional e navegação: Nesta etapa são iniciados os testes com o drone real além do simulador, procurando validar os testes virtuais e também entendendo o funcionamento e os problemas do drone em questão. As tecnologias candidatas para localização indoor são testadas, tanto no drone quanto na sua compatibilidade e instalação no sistema.

Integração e testes: Os módulos de localização e controle do drone são integrados, além de baterias de testes e depuração de problemas. A interface também é integrada e os testes também são realizados controlando-se o drone por ela.

Documentação e apresentação: Embora todos os milestones incluam a confecção da monografia, esta última etapa visa revisar, corrigir e consolidar a documentação, além de preparar as apresentações e demais documentos de entrega.

O planejamento do projeto é feito construindo uma EAP (Estrutura Analítica do Projeto), estimando o tempo de cada atividade em homem-hora e inserindo-as no Gantter (TECHNOLOGIES, 2015), ferramenta online de gestão de projetos.

Figura 10: Gantter - Ferramenta online para gestão de projetos com planejamento do projeto.

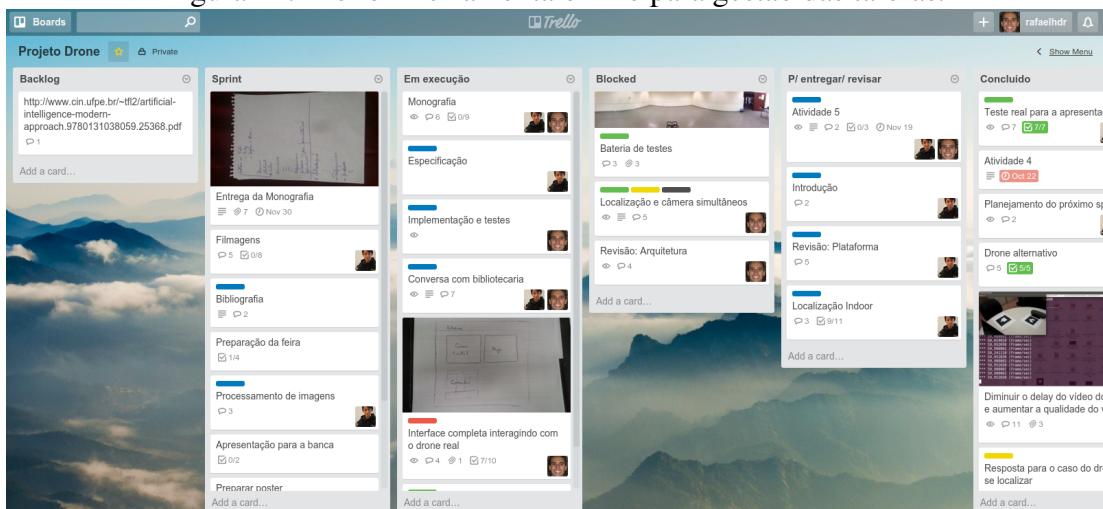


Fonte: Autor

Sprint

O sprint é um aglomerado de tarefas a ser realizado num período de duas semanas. São atividades de curta duração e podem ser divididas em subtarefas menores. Para gestão dos sprints, utiliza-se o Trello (TRELLO, 2015).

Figura 11: Trello - Ferramenta online para gestão das tarefas.



Fonte: Autor

Nesta ferramenta, as atividades são representadas por cartões (cards) e são divi-

didas em listas, representadas em formato de colunas. Explica-se, então, cada uma delas:

O **backlog** reúne as atividades agendadas para o próximo milestone, portanto é preenchida com tarefas no início de cada uma delas, embora eventualmente novas tarefas criem a necessidade de serem adicionadas nesta lista conforme se evolui o projeto.

A tabela **sprint** reúne as atividades agendadas a serem realizadas nos dias seguintes, mais especificamente em duas semanas. Conforme a disponibilidade e sequência das atividades, um membro da equipe se aloca em uma destas tarefas e transfere o cartão para a lista **em execução**, significando que esta tarefa está sendo executada no presente instante. Por motivos organizacionais e de produtividade, o número de tarefas desta lista deve ser limitada e datas de limite de entrega devem ser estabelecidas.

Outro caminho que a tarefa pode seguir é para a lista **bloqueado**, na qual se alocam as tarefas que não podem ser realizadas por algum motivo, como a dependência de outra tarefa ou impossibilidade de material ou local.

Uma vez que a tarefa seja executada, ela deve passar pela coluna **Para entregar/revisar**, na qual outros membros (que podem ser marcados e avisados sobre) serão encarregados de revisão. Assim que essa revisão for realizada, a tarefa passa para a coluna **concluído**, ficando registradas para eventuais consultas e documentação.

Tarefa

As tarefas estão inclusas nos sprints. São definidas pela equipe ao início dos milestones e também dos sprints. Elas devem ser de fácil entendimento, e servir de comunicação entre a equipe. Dessa forma, não se perde conversas, pois estão todas documentadas em cada cartão.

Figura 12: Tarefa colocada na coluna blocked.

Add

- Members
- Labels
- Checklist
- Due Date
- Attachment

Actions

- Move
- Copy
- Subscribe
- Archive

[Share and more...](#)

rafaelhdr
Estou acessando a câmera do Drone, não faço ideia como. Agora vou estudar esse código melhor.
Jul 30 at 3:02 PM - [Edit](#) - [Delete](#)

rafaelhdr attached P_20150730_145818.jpg.jpeg to this card

Jul 30 at 3:01 PM

rafaelhdr moved this card from Blocked to Em execução Jul 30 at 3:00 PM

rafaelhdr
Preciso do Drone.
Jul 28 at 5:03 PM - [Edit](#) - [Delete](#)

rafaelhdr joined this card Jul 28 at 5:03 PM

rafaelhdr moved this card from Sprint to Blocked Jul 28 at 5:02 PM

Elder Nakashima added this card to Sprint Jul 23 at 2:02 PM

Fonte: Autor

Figura 13: Utiliza-se o Trello para conversas, centralizando a comunicação.

The screenshot shows a Trello board with a single column containing five messages. The right side of the interface includes 'Add' and 'Actions' sections.

- Add:**
 - Members
 - Labels
 - Checklist
 - Due Date
 - Attachment
- Actions:**
 - Move
 - ✉ Copy
 - 👁 Subscribe
 - 📝 Archive
- Share and more...**

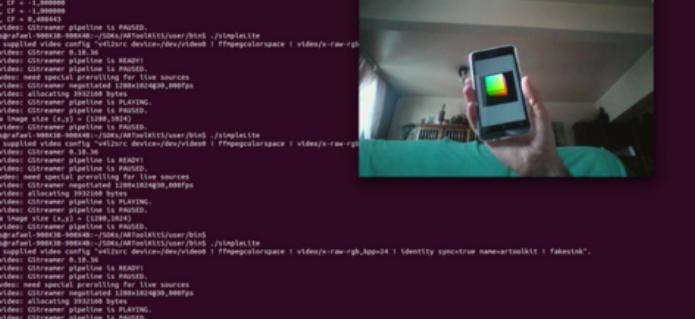
Messages:

- rafaelhdr (Mar 17 at 8:25 PM) - Não sei se ajuda, mas peguei um tutorial de Game (carrinho) <http://www.cgmasters.net/free-tutorials/python-scripting/>
Super simples, e fácil de entender.
- rafaelhdr (Mar 17 at 5:57 PM) - Certo. Vou dar uma estudada nele. Recomenda algo?
- Elder Nakashima (Mar 13 at 4:35 PM) - @rafaelhdr eu falei com o professor ontem rapidamente, ele realmente entende muito pouco de blender... o que eu sei de blender não é o programa em si, mas um pouco de bgui e bge, que é blender graphic user interface e blender game engine, ou seja, tudo em código :)
- rafaelhdr (Mar 13 at 4:33 PM) - @eldernakashima Me ensina Blender um dia? Algo bem simples, mas pra perder menos tempo que no tutorial deles, e otimizarmos o tempo.
- rafaelhdr (Mar 13 at 4:14 PM) - Vi o paparazi. Parece bem legal, mas não indoor, certo? Me pareceu até que ele usa um Google Maps.

Fonte: Autor

Figura 14: Informações interessantes são mantidas nos cards, servindo de referência na documentação.

rafaelhdr attached [artoolkit-simpleLite.png](#) to this card



Jul 17 at 2:54 PM

rafaelhdr **Funcionando :**

O simpleLite funciona alterando a linha 540 para
char vconf[] = "v4l2src device=/dev/video0 ! ffmpegcolorspace ! video/x-raw-rgb,bpp=24 ! identity sync=true name=artoolkit ! fakesink";

O teste simples funciona com
.simpleTest v4l2src device=/dev/video0 ! ffmpegcolorspace ! video/x-raw-rgb,bpp=24 ! identity sync=true name=artoolkit ! fakesink

Jul 17 at 10:52 AM - [Edit](#) - [Delete](#)

rafaelhdr artoolkit com câmera funcionando por 1 frame
.simpleTest v4l2src device=/dev/video0 ! ffmpegcolorspace ! identity name=artoolkit ! autovideosink

Jul 17 at 10:02 AM - [Edit](#) - [Delete](#)

Add

Members

Labels

Checklist

Due Date

Attachment

Actions

Move

Copy

Subscribe

Archive

Share and more...

Fonte: Autor

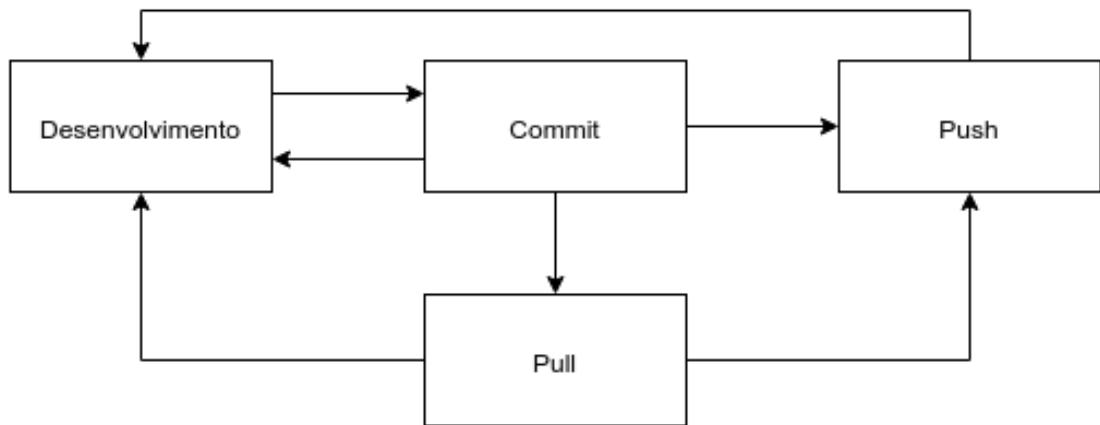
Como ferramentas auxiliares, é possível criar etiquetas, ou labels, os quais facilitam a categorização das tarefas e sua visualização. Membros podem ser alocados, datas de limite estipuladas, *checklists* para subdivisão de tarefas podem ser criadas e arquivos podem ser anexados aos cartões de tarefa.

6.2 Codificação e versionamento

O código fonte do projeto foi completamente versionado utilizando o Git (CONSERVANCY, 2015), incluindo a documentação. Esta ferramenta serve também como backup dos registros, já que o projeto se torna mais seguro em casos de danificação das máquinas ou roubo. Se houver perda, o conteúdo está todo na nuvem. Apenas o que não foi enviado para a ferramenta será perdido, por isso se deve buscar uma atualização constante, além de possibilitar à equipe estar com o projeto também atualizado. A equipe trabalha em paralelo, e com o Git, a troca de códigos é feita automaticamente, havendo trabalho apenas quando há conflitos nos códigos enviados para a nuvem, cabendo ao último que fez a atualização ajustar os conflitos.

O uso da nuvem é realizada pelos serviços do BitBucket (ATLASSIAN, 2015).

Figura 15: Metodologia para utilização do git.



Fonte: Autor

Desenvolvimento: Desenvolver o projeto, seja codificando código, ou mesmo documentando. Esse processo é feito na máquina do responsável. Quando termina, é necessário fazer o commit.

Commit: O commit é a finalização da tarefa, versionando o código. Após o commit, é possível continuar com outra tarefa (evitando interação com a nuvem), fazer o pull do código que está na nuvem (quando há algo de fora para trazer), ou então

push do código para a nuvem (quando não há algo de fora, e deseja atualizar com as atualizações feitas).

Pull: É preciso estar com o commit feito, para ter seu conteúdo salvo antes de trazer o novo, e com este comando são trazidas da nuvem as atualizações feitas.

Push: Após commits ou pull, o desenvolvedor envia para a nuvem todo o trabalho feito com este comando.

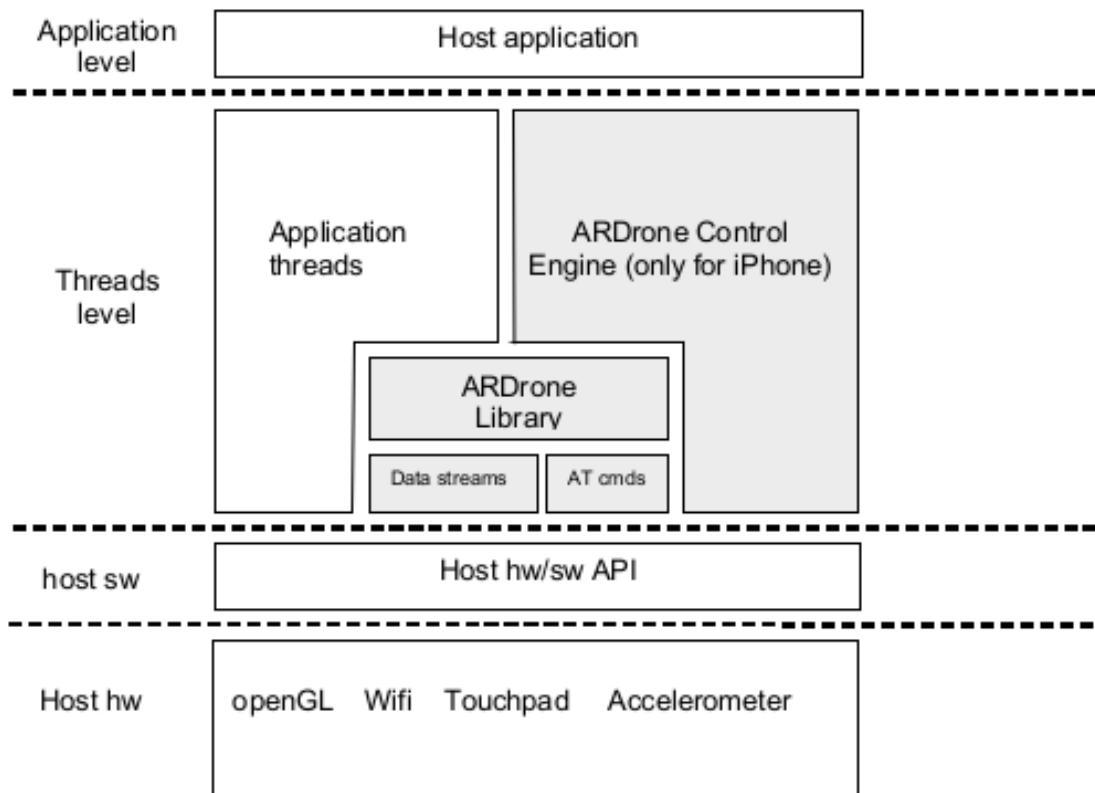
7 ARQUITETURA

7.1 Componentes

7.1.1 Drone

As especificações do drone são citadas em 2. A utilização de seus componentes é feita utilizando o SDK disponibilizado pela fabricante (PISKORSKI et al., 2012), mas abstraído com a utilização de libraries disponibilizadas publicamente, como em (GEISENDÖRFER, 2012; ESCHENAUER, 2013).

Figura 16: Representação da divisão em camadas da arquitetura de comunicação.



Fonte: <http://ardrone2.parrot.com/>

Das diversas camadas da figura 7.1.1, utiliza-se apenas a **ARDrone Library** para o projeto, pois é onde está a atuação do SDK disponibilizado. É possível alterar o software embarcado, mas não há suporte do fabricante para essa solução.

"However, this SDK does NOT support : rewriting your own embedded software - no direct access to the drone hardware (sensors, engines) is allowed." (PISKORSKI et al., 2012)

O conteúdo disponibilizado pela AR Drone Library é mostrado a seguir:

Soft : Serviços de comunicação, controle e dados de navegação;

VLIB : Biblioteca de video para o AR Parrot Drone 1.0;

FFMPEG : Biblioteca de video para o AR Parrot Drone 2.0;

ITTIAM : Decodificador de video para Android e iPhone;

VPSDK : Bibliotecas de diversas finalidades, incluindo processamento de video, wrappers multiplataforma para funções de sistema, helpers para pipelines de video.

7.1.2 Terminal

O terminal é utilizado como interface entre o usuário e o drone. Para o projeto, utiliza-se como terminal o notebook Samsung Series 9 900X3B, com as seguintes configurações: processador Intel Core i5 ULV, memória RAM de 4 GB DDR3 1,333 MHz, placa gráfica Intel HD graphics 3000, utilizando sistema operacional GNU-Linux com a distribuição Ubuntu 14.04, sendo essa versão escolhida por ser Long Term Service (LTS), havendo alguns softwares que são produzidos apenas para essa versão (SAMSUNG, 2012).

A interação com o usuário é feita por uma interface. Seu objetivo é ser de fácil utilização para pessoas que já estão acostumadas com a utilização de um navegador web. Seu funcionamento será aprofundado em 7.2.3.

7.1.3 Comunicação

A comunicação entre o terminal e o drone é feita via WLAN. Para tal, utilizam-se 4 portas, cada uma delas responsável por um tipo diferente de dados.

7.1.3.1 Navdata

Dados de navegação (como posição, velocidade, velocidade de cada motor, entre outras informações) são enviadas a uma taxa de 15 Hz, para o modo de demonstração, e a 200 Hz, modo *debug*. A transmissão é feita via UDP (User Datagram Protocol) utilizando a porta 5554.

7.1.3.2 Video stream

A transmissão do video é feita pela porta 5555 utilizando protocolo TCP (Transmission Control Protocol). Isso pode acarretar problemas de delay, contornado com um mecanismo de redução de latência (*latency reduction mechanism*) que envia apenas o último frame ao trabalhar com video em tempo real.

A câmera frontal tem capacidade de 720p com 30 frames por segundo (720p-30fps). Ela pode ser configurada para enviar em 360p ou 720p.

7.1.3.3 AT commands

Ardrone Text Strings commands são as sequências de caracteres enviados, as quais são traduzidas em comandos pelo drone. O terminal envia pacotes UDP pela porta 5556 a uma frequência de 30 Hz.

O anexo A contém a lista de comandos que podem ser executados.

7.1.3.4 Control port

A porta de controle é utilizada para transmissão de dados sensíveis pela porta 5559 utilizando protocolo TCP. Os tipos de informação são documentados no manual da fabricante (PISKORSKI et al., 2012) e o anexo B apresenta alguns exemplos.

7.2 Módulos

São as aplicações desenvolvidas no terminal para realização do projeto. Estão organizadas em sequência de criação.

7.2.1 Simulador

Projetado para representar o drone físico no terminal durante a etapa de desenvolvimento. Ele simula os comandos do drone e os algoritmos programados.

Diversos são os motivos que trazem a necessidade de um ambiente de simulação antes de testar os algoritmos implementados e sensores diretamente com o drone. Embora o AR.Drone Parrot 2.0 possua protetores para as hélices justamente para navegação indoor, uma bateria de testes prévia é útil no sentido de evitar acidentes com objetos do ambiente, danificação do próprio drone ou até mesmo uma indesejável fuga sem controle do quadricóptero, cujas consequências podem ser desastrosas e perigosas.

Outro fator importante a se considerar ao modelar um simulador é que uma das limitações do drone está em seu tempo de vôo. A bateria do drones, em geral, dão autonomia inferior a 30 minutos, sendo que após esse tempo, a recarga seria necessária caso não haja peças de reposição sobressalentes. Ademais, possibilita um paralelismo no qual os membros podem testar simultaneamente o comportamento do drone

Além disto, o nível de ruído do drone é elevado devido a movimentação das hélices e dos seus rotores. Desta forma, em ambientes os quais não possuam isolamento acústico se traduzem em limitações de horário e local para aplicação de seus testes.

Porém, por limites de custo e orçamento, determinados testes serão executados apenas com o drone, visto que ele acopla diversos sensores os quais não serão disponíveis como peças separadas, e não se considera a hipótese de desmontar a estrutura principal do drone para separação destes sensores. Ainda se considera o peso do drone como fator limitante para a inserção de novos itens embarcados, visto que seu peso é diretamente proporcional ao consumo de energia.

7.2.1.1 Requisitos do simulador

Emulação dos comandos: O simulador deve receber comandos da mesma maneira que o drone recebe, e responder a estes com um comportamento similar. Para isto, a implementação de algoritmos de leitura de entradas deve ser incremental em termos de nível de complexidade, com o intuito de facilitar a interação com a ferramenta de modelagem e seu consequente aprendizado e familiarização.

Placa de vídeo: Os simuladores requerem placas de vídeo para processar graficamente os ambientes de simulação. Deste modo, se faz necessária uma placa de vídeo suficientemente robusta para lidar com a carga gráfica demandada.

Similaridade visual com o AR.Drone Parrot 2.0: O simulador deve ter um modelo similar ao drone utilizado, de forma que seja possível reconhecer na imagem que se trata do objeto a ser controlado, e não seja confundido com demais objetos componentes do ambiente. Tal requisito visa facilitar a busca por um modelo que se adeque ao propósito de simulação sem que se perca a garantia de reconhecimento visual que é um drone em vôo que está em questão. Deve ser possível identificar a orientação do drone, ou seja, qual é o lado frontal (que leva a câmera frontal) e se ele está "em pé"(com a câmera vertical para baixo).

Autonomia: Para os testes em que se leva em questão a autonomia, retorno a ponto seguro e cálculos de tempo de vôo, deve-se levar em conta esta variável. Para os demais testes em que não se leva em conta a autonomia, o drone poderá voar indefinidamente a fim de não interromper o processo.

Ambiente: O sistema de simulação deve ser dotado de um ou mais ambientes indoor, atendendo assim ao escopo do projeto. Por ambiente indoor, entende-se, nos requisitos para simulador de vôo, um espaço rodeado totalmente por paredes, portas (abertas ou fechadas) e janelas fechadas; piso predominantemente plano; e teto predominantemente plano, podendo ser dotado de aparelhos de iluminação ou ventilação.

Obstáculos: O sistema de simulação deve possuir paredes internas, as quais são intransponíveis pelo drone.

7.2.1.2 Alternativas de simulador

Blender: Software amplamente usada para desenho gráfico em 3D, mas também com possibilidade de ser usada para games. Uma grande vantagem é sua popula-

ridade, sendo a alternativa mais conhecida. Aparenta possuir uma menor curva de aprendizado, por ter mais pessoas contribuindo, tendo alto grau de customização.

Mas essa grande customização acaba sendo uma desvantagem também. Por possuir diversos projetos com quadcopteros, mas não exatamente um projeto com o AR Parrot 2.0.

FlightGear: Desenvolvido especialmente para vôos. Esse software é desenvolvido para ensino de pilotagem de avião. É uma ferramenta bem completa para esse tipo de simulação. Isso é uma grande vantagem, por ser bem específico. Além disso, há um projeto de unificação de controle de UVA, incluindo o AR Parrot 2.0 (PAPARAZZI, 2015).

Porém, a curva de aprendizado parece ser maior. Além disso, suas vantagens de vôo se devem principalmente à vôos em ambiente externo, sendo o projeto focado para interno.

TUM Simulator: Software feito especialmente para simulação interna do AR Parrot 2.0. Possui diversos artigos para seu bom funcionamento.

É um software mais antigo, e tem maior dependência de uma boa placa de vídeo, sendo que seu funcionamento não foi possível nas máquinas disponíveis para utilização.

X-Plane: Ferramenta semelhante ao FlightGear, mas mais conhecida. Possui vantagens semelhantes, com exceção de ter um projeto suportando o AR Parrot 2.0. Além disso, é uma ferramenta paga, assim possuindo como desvantagem o aumento do custo.

7.2.1.3 Justificativa de escolha

Dentre as opções, o **Blender** apresenta melhor adequação bem ao terminal utilizado 7.1.2, que possui a placa de vídeo - Intel HD graphics 3000 - como componente gráfico limitante. Além disso, é customizável para as necessidades encontradas, e pos-

sui bibliotecas de atuação e controle em python.

7.2.2 Algoritmo de melhor caminho

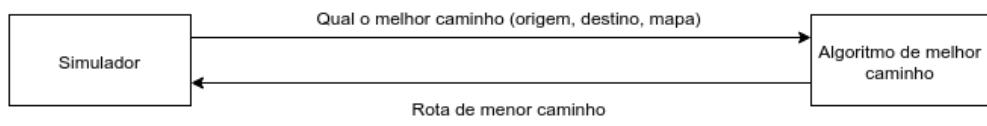
Esse módulo é utilizado para calcular a melhor rota entre a posição inicial do drone e seu destino final. Para sua implementação, foi utilizada uma library pública que implementa pathfinding com diversos algoritmos de pathfinding na linguagem Node.js (XU, 2011). Dos diversos algoritmos disponibilizados, foi escolhida a implementação utilizando o algoritmo A*. A razão para isso é sua performance ser superior entre os demais implementadas.

7.2.2.1 Algoritmo A*

O algoritmo A* (lê-se A estrela) é uma solução amplamente utilizada em jogos eletrônicos. Após seu surgimento, houveram diversas publicações visando sua otimização. É comprovado que o A* encontrará um caminho, caso ele exista, e ele será o melhor, de acordo com a heurística dada (CUI; SHI, 2011).

7.2.2.2 Sistema

Figura 17: Arquitetura do sistema com simulador e algoritmo de melhor caminho



Fonte: Autor

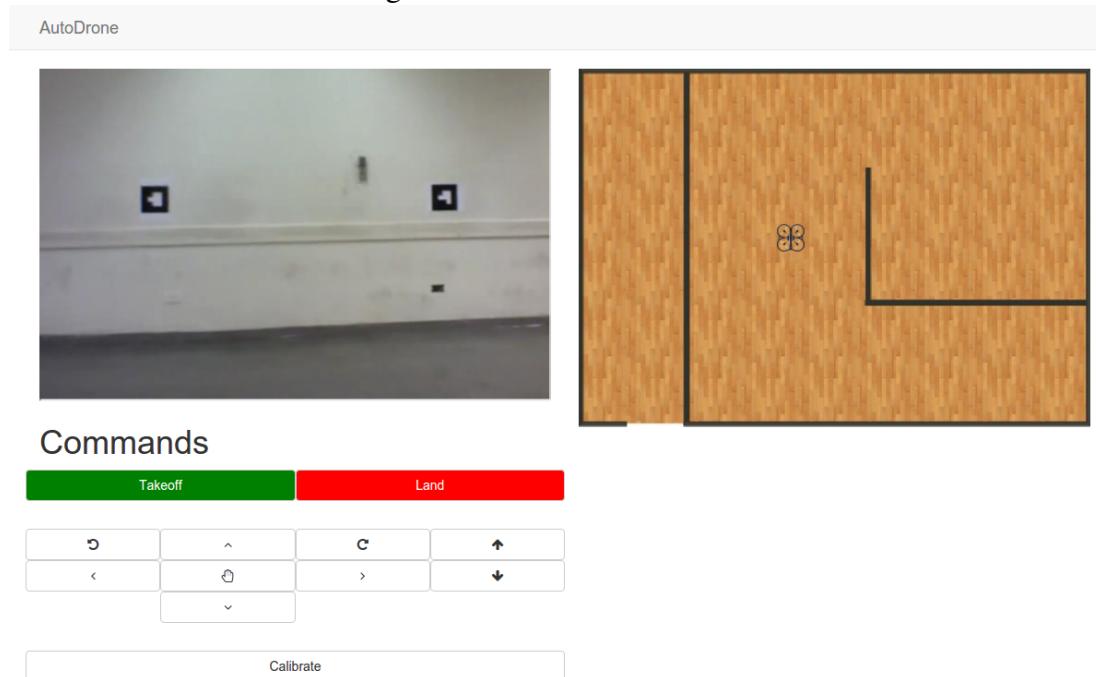
Nessa etapa, o sistema é composto apenas do módulo simulador e algoritmo de melhor caminho. Sua comunicação consiste no simulador solicitando o melhor caminho, passando como argumentos sua origem, o destino final desejado e o mapa do ambiente. A resposta é a rota do menor caminho, composta pela lista de posições que o drone deve passar para chegar ao destino. Se não houver menor caminho, deve ser retornado vazio (null em node.js, que deverá ser interpretado como None em python).

Se já estiver em seu destino, deve ser retornado uma lista vazia.

7.2.3 Interface

A IHM (Interface Homem-Máquina) é realizada com o terminal que possuir o sistema. Através de um navegador web, pode-se visualizar a posição do drone, sua câmera frontal, e executar comandos.

Figura 18: Interface do usuário



Fonte: Autor

7.2.3.1 Localização

A localização do Drone é transmitida utilizando o banco de dados como intermediário. O simulador salva no banco de dados da interface sua posição atual e a interface detecta as alterações e então atualiza a posição do drone. Essa mudança acontece sem a necessidade de recarregar a página com o uso do Meteor Livequery (METEOR, 2015d).

7.2.3.2 Câmera frontal

O drone transmite o video de sua câmera frontal via Wi-Fi. Seus dados são interpretados através da library Drone Stream (WEISSHUHN, 2012) e a interface a replica para o usuário.

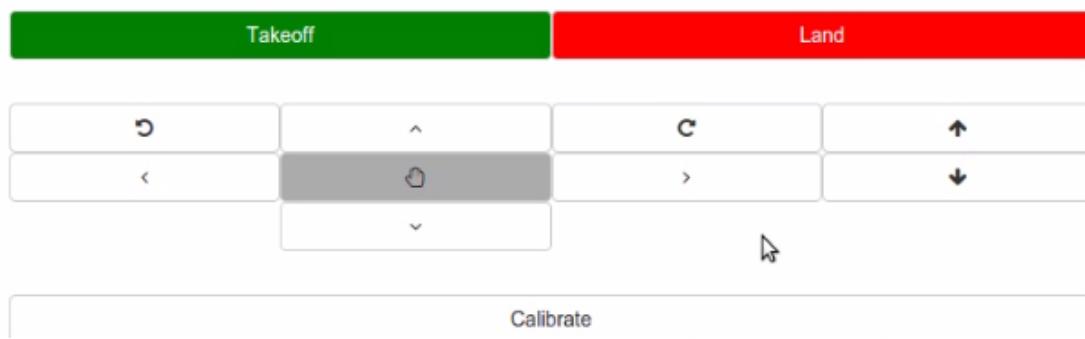
Uma possível alternativa é utilizar o player de video do navegador, seguindo HTML5. Porém seu delay é alto, chegando a 10 segundos, diferente da biblioteca Drone Stream, que possui delay menor de 1 segundo.

7.2.3.3 Comandos

O usuário envia as instruções para o drone através dos comandos disponíveis pela interface. Esses comandos podem ser disparados ao clicar em botões na tela, ou utilizando atalhos do teclado. A figura 7.2.3.3 mostra os comandos disponíveis na tela do usuário e a tabela 1 relaciona os botões aos atalhos, explicando também o que faz cada um desses comandos.

Figura 19: Interface do usuário

Commands



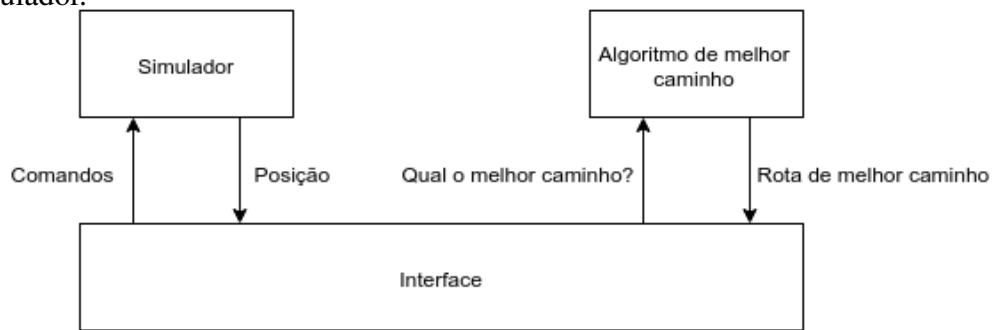
Fonte: Autor

AT commands	Tecla	Descrição
Takeoff	K	Decolar o Drone
Land	L	Pousar o Drone
↗	W	O drone vai para frente
↖	A	O drone vai para o lado esquerdo
↗	D	O drone vai para o lado direito
↘	S	O drone vai para trás
✋	X	O drone pára seu movimento
↻	E	O drone gira em sentido horário
↺	Q	O drone gira em sentido anti-horário
↑	R	O drone vai para cima
↓	F	O drone vai para baixo
Calibrate	C	É feita uma calibração

Tabela 1: Relação entre teclas e botões da interface com seus respectivos comandos.

7.2.3.4 Sistema

Figura 20: Arquitetura do sistema com simulador, algoritmo de melhor caminho e interface.



Fonte: Autor

Nessa etapa do projeto, há 3 módulos integrados, com as operações centralizadas na interface. Ela utiliza as operações anteriores para conexão com o algoritmo de melhor caminho da mesma forma que o simulador se comunicava antes, como visto em 7.2.2.2.

A conexão entre interface e simulador é feita por DDP (Protocolo de Distribuição

da Dados) (METEOR, 2015c), utilizando um socket entre a aplicação em Python do simulador e o Meteor da interface. O usuário envia um comando para a interface, que envia o comando ao simulador, que executa. Paralelamente, o simulador envia a posição do drone sempre que há uma mudança.

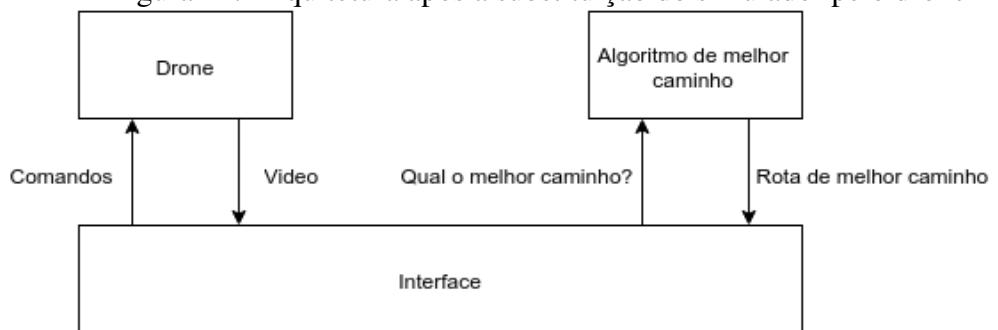
DDP: É um protocolo que visa fazer requisição ao banco de dados, trazendo os resultados, e enviar mudanças assim que houver alterações. Sua especificação define duas operações: Chamadas remotas pelo cliente para o servidor e cliente se inscrevendo em conjuntos de documentos para receber suas atualizações (DEBERGALIS, 2015; METEOR, 2015c; METEOR, 2015a).

python-meteor: Utiliza-se a library python-meteor para a comunicação entre o simulador e a interface. No simulador, há um script específico para isso (meteor.py), onde são implementadas funções de atualização da localização (do simulador para a interface), é informado o estado do drone (em vôo ou parado) e se obtém pedidos da interface (utilizado para seguir rota) (HARNISCH, 2014; TIPLING; COHEN, 2013; TIPLING; COHEN, 2015).

7.2.4 Drone

Após familiarização com o ambiente e os testes feitos, substitui-se o módulo simulador pelo drone real.

Figura 21: Arquitetura após a substituição do simulador pelo drone



Fonte:

Autor

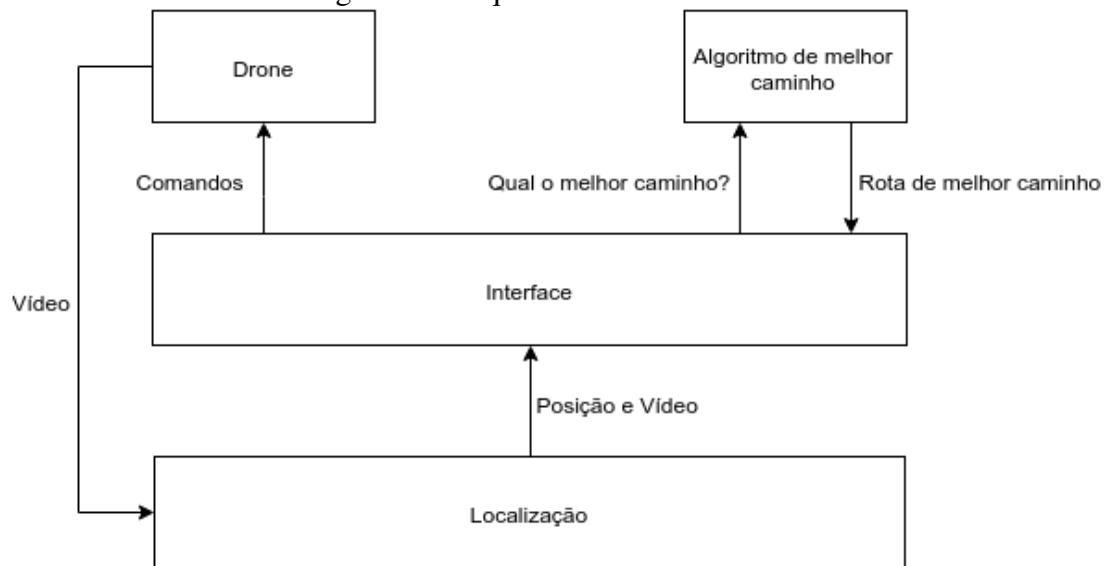
Com essa mudança, troca-se a funcionalidade de localização, em que antes era feita pelo BGE (BLENDER, 2015) e será substituída pelo módulo de localização através da câmera frontal do drone.

7.2.5 Localização

A localização indoor é feita utilizando as imagens da câmera frontal do drone. As imagens são coletadas pelo ARToolKit (VAUGHAN; LAMB; YOUNG, 2015), que utiliza o GStreamer para acessar a porta 5555 no IP 192.168.1.1, e à partir da detecção dos marcadores, cuja localização é previamente sabida pelo sistema, se estima a posição global do drone.

Com a localização, a arquitetura é finalizada como na figura a seguir.

Figura 22: Arquitetura final do sistema



Fonte: Autor

O video não é mais enviado diretamente para interface, mas sim para o script de localização com o ARToolKit. Sua posição é calculada e o video é retransmitido.

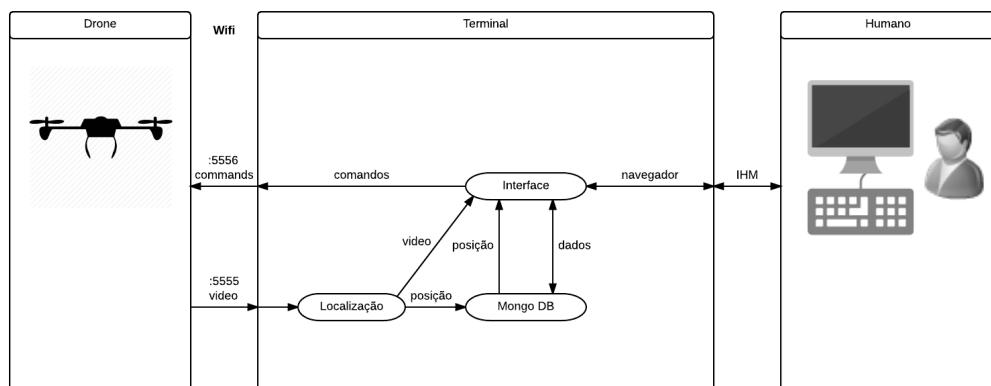
A posição é transmitida da mesma forma que o simulador. A posição final é salva no banco de dados, que atua como intermediário, como explicado na seção 7.2.5.

O video é retransmitido utilizando o GStreamer (GSTREAMER, 2015) que duplica os frames, enviando para o ARToolKit e também para a interface.

7.2.6 Sistema final

Os módulos atuam no terminal do usuário. A figura 23 mostra a arquitetura incluindo as interações com o drone e humano, sendo o terminal um intermediário.

Figura 23: Arquitetura final do sistema



Fonte: Autor

Em relação ao que foi construído, acrescenta-se a seta dupla "dados", que se refere à todos as informações armazenadas no banco de dados, como é comumente feito em sistemas que utilizam um SGBD (Sistema de Gerenciamento de Banco de Dados).

8 IMPLEMENTAÇÃO E TESTES

8.1 Simulador

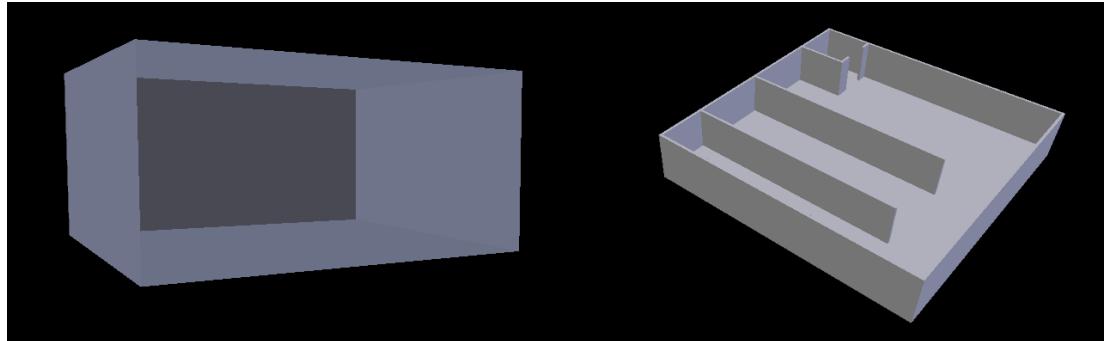
O software utilizado para o simulador foi o Blender, conforme explicado em 7.2.1. Esta seção foca em explicar sua construção, desde a modelagem do ambiente e do Drone, assim como a programação do controle e algoritmos implementados.

8.1.1 Modelagem

O primeiro passo do software é modelar o ambiente e o drone. Esse é um passo necessário, mas o detalhamento gráfico não o é, pois não auxiliará em nenhum dos requisitos listados em 7.2.1.1, e sim contraprodutivo pois trará mais dificuldade no processamento da placa de video. Por esse motivo, a modelagem é bastante simplificada, de forma a atender às necessidades do projeto.

O primeiro ambiente era um cubo, com o intuito de familiarização com o software. Para testes do algoritmo de pathfinding foi necessário sua melhoria, com paredes e obstáculos.

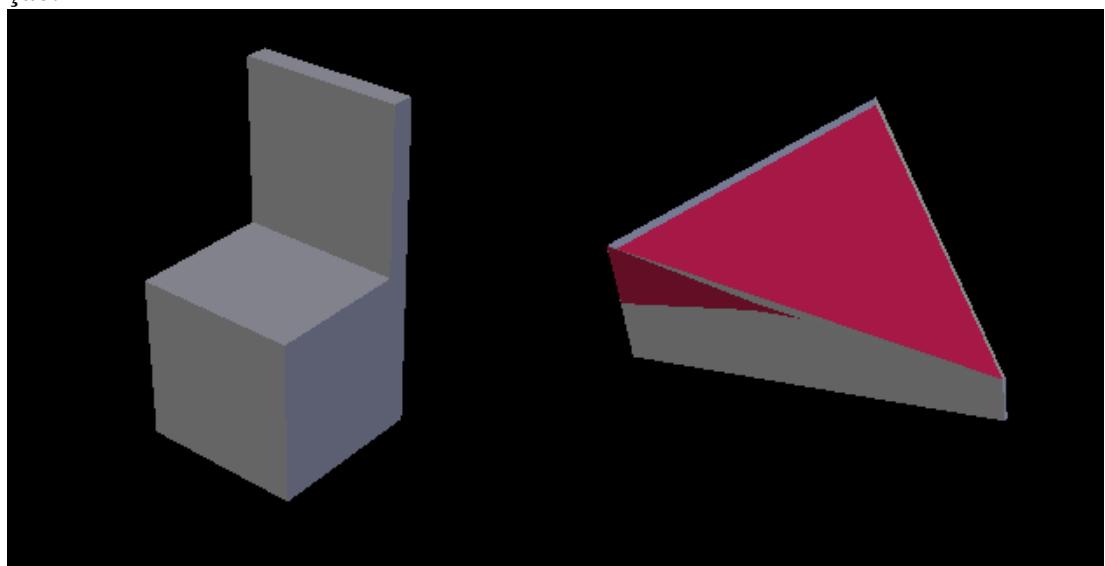
Figura 24: Ambientes de simulação. À esquerda, mais simplificado para os primeiros testes. À direita, com paredes, para testes do Algoritmo de pathfinding.



Fonte: Autor

O único obstáculo para os testes era uma cadeira, para que seja desviado durante o vôo. O drone se diferencia para saber sua orientação, com um chanfro na parte superior, além de estar em vermelho.

Figura 25: Uma cadeira simplificada, suficiente para testes. Drone também simples, com parte superior chanfrada e em vermelho para não haver confusão de sua orientação.

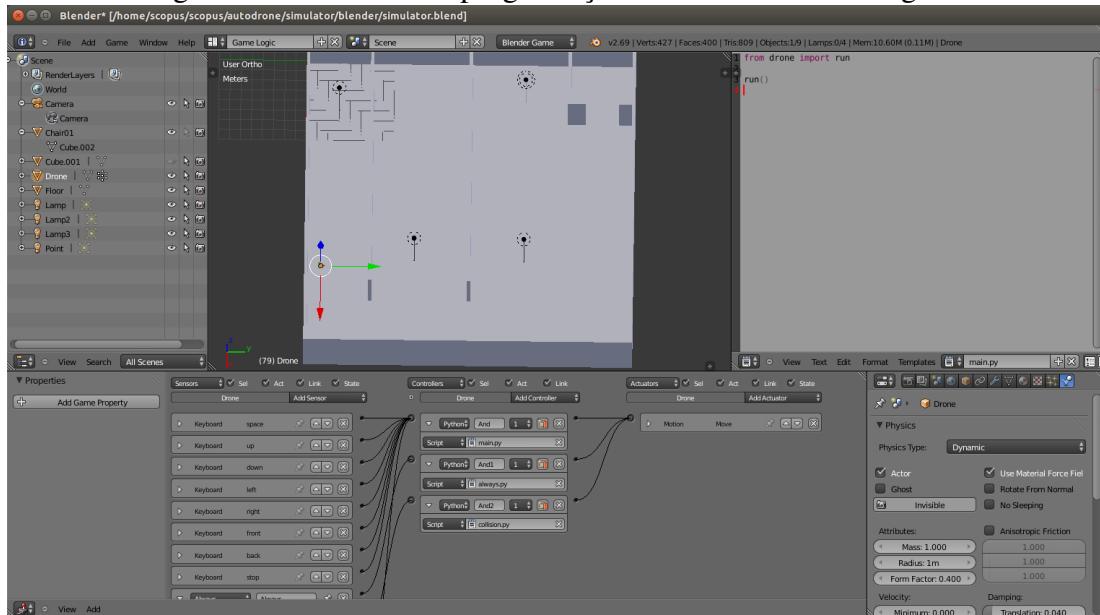


Fonte: Autor

8.1.2 Controle

O controle da movimentação é feito com o teclado. Para isso, foi utilizado o BGE (Blender Game Engine), ferramenta com foco em desenvolvimento de jogos, podendo ser utilizado para desenvolvimento de qualquer software interativo (BLENDER, 2015). A figura 26 mostra o painel geral com a lógica de jogo, em que se associa as teclas aos seus respectivos comandos.

Figura 26: Ambiente de programação da Blender Game Engine.



Fonte: Autor

A implementação do algoritmo é desenvolvida em Python, que é interpretada pelo BGE. Há dois programas principais:

drone.py: Responsável pela movimentação do Drone, interpretando e aplicando os comandos do usuário ao programa, e também na implementação dos algoritmos de rota.

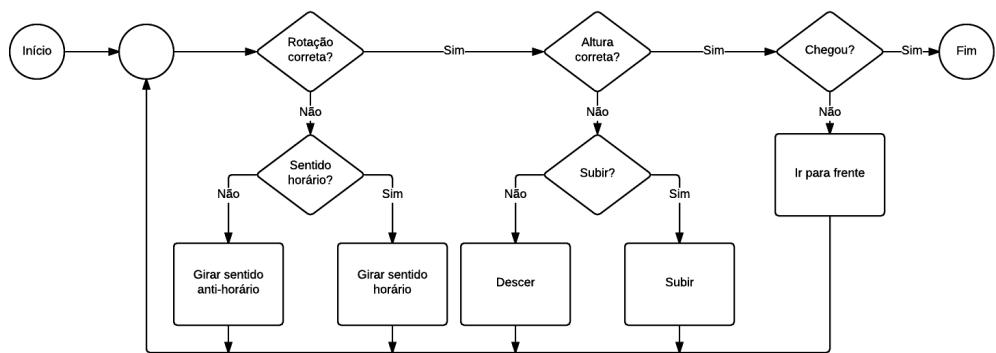
meteor.py: Responsável pela interação com a interface. Coleta e salva as informações no Banco de Dados.

8.1.3 Algoritmos

8.1.3.1 Algoritmo de ir ao destino

Seu objetivo é fazer com que o Drone se localize e se dirija para algum destino, ou seja, dado a posição atual (X_0, Y_0, Z_0) ele deve ir para (X_1, Y_1, Z_1). A figura 30 mostra o fluxograma do programa que realiza o algoritmo de ir ao destino.

Figura 27: Fluxograma do primeiro algoritmo do simulador



Fonte: Autor

Sua simplicidade consiste em ignorar a existência de obstáculos, seguindo retas até o destino.

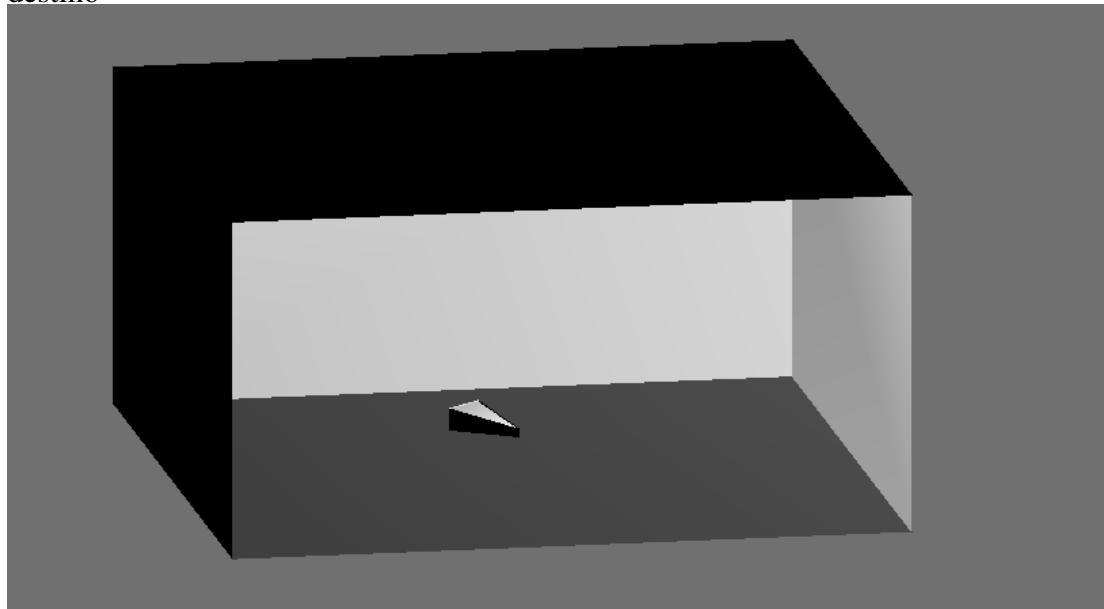
8.1.3.2 Testes do algoritmo de ir ao destino

Para obter a localização do objeto, inicialmente se detecta visualmente, mas para maior precisão, utiliza-se um debugger com o seguinte comando:

```
print(bge.logic.getCurrentController().owner.worldPosition)
```

Esse comando imprime no terminal a posição atual do objeto corrente (no caso, o drone). Para dispará-lo, o associamos a tecla D (de debug).

Figura 28: Drone no tempo inicial, antes da execução, no teste do algoritmo de ir ao destino

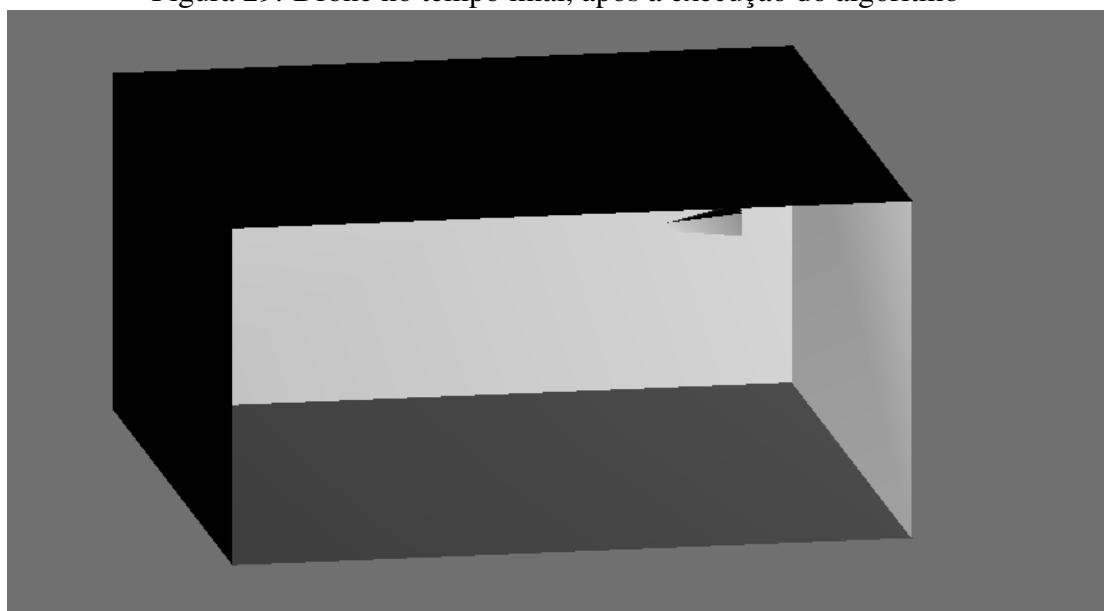


Fonte: Autor

A figura 28 mostra o drone em sua posição inicial (-0.4314, 0.0000, 1.0000).

Em seguida, roda-se o programa para ir ao destino (7.0, 7.0, 10.0). Ao final, ele se encontra na posição (7.0672, 7.0608, 10.0000), como visualizado na figura 29.

Figura 29: Drone no tempo final, após a execução do algoritmo



Fonte: Autor

Após o teste inicial, outros testes foram feitos, com posição inicial e destino final gerados aleatoriamente. A seguir, seus resultados:

Posição inicial	Destino final	Destino final real
(8.1, 7.8, 11.7)	(-6.2, 0.2, 2.0)	(-6.2045, 0.1974, 2.0000)
(-9.8, 2.4, 9.8)	(1.8, -3.8, 13.4)	(1.8406, -3.8225, 13.4000)
(0.8, 1.9, 8.7)	(-6.3, 0.5, 1.4)	(-6.3616, 0.4863, 1.4000)
(0.5, 7.8, 3.5)	(1.9, -6.2, 11.2)	(1.8948, -6.1297, 11.2000)
(6.7, -9.0, 3.7)	(8.2, -3.8, 13.2)	(8.2112, -3.7562, 13.2000)

Para a escolha dos pontos iniciais e finais, foi feito um script em python que gerava posições aleatórias, exemplificado a seguir:

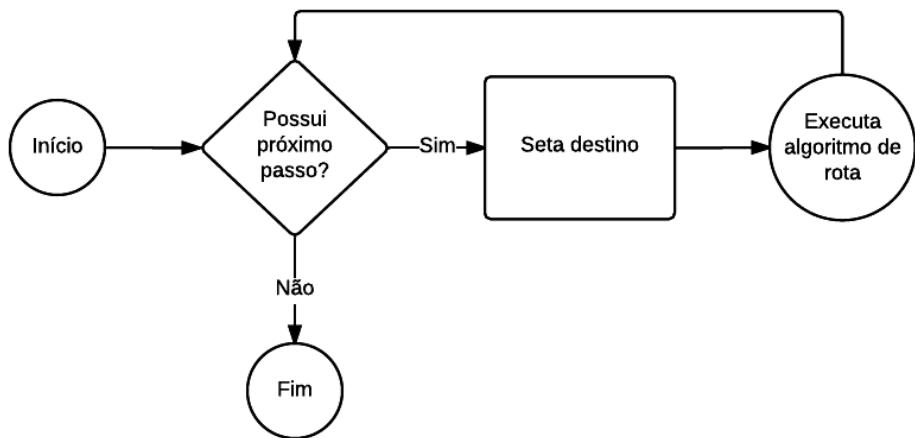
```
import random

print((float(str(random.randint(-9,9)) + "." + str(random.randint(0,9))), \
float(str(random.randint(-9,9)) + "." + str(random.randint(0,9))), \
float(str(random.randint(1,14)) + "." + str(random.randint(0,9))))
```

8.1.3.3 Algoritmo de seguir rota

A criação da rota é feita externamente, sendo seu funcionamento explicado em 8.2. Com a rota dada, o drone deve ir ao destino utilizando retas geradas pelo algoritmo, evitando obstáculos previamente mapeados. O algoritmo segue a lógica do fluxograma abaixo:

Figura 30: Fluxograma do algoritmo de seguir rota. Ele reutiliza o algoritmo previamente criado de ir ao destino.



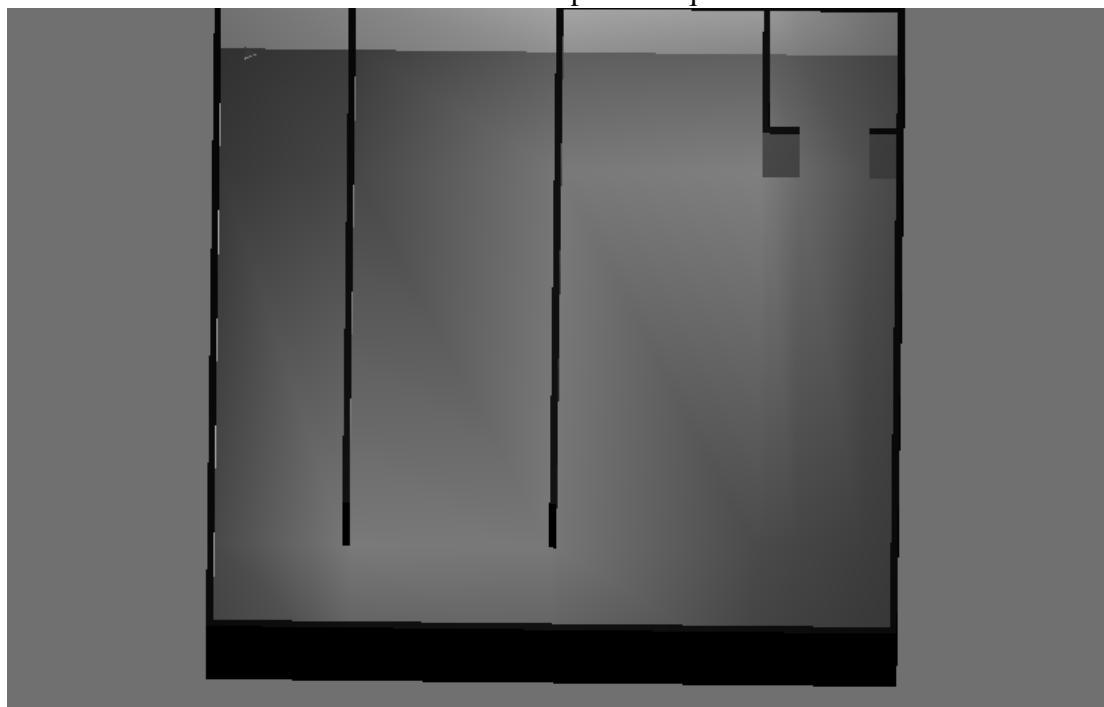
Fonte: Autor

O algoritmo de seguir ao destino, explicado em 8.1.3.1, é reutilizado. O algoritmo de rota busca o próximo destino, vai ao destino da reta, e então aloca o próximo destino, até alcançar o destino final.

8.1.3.4 Testes do algoritmo de seguir rota

A figura 31 mostra o drone saindo do ponto (1.5, 1.5, 2.0).

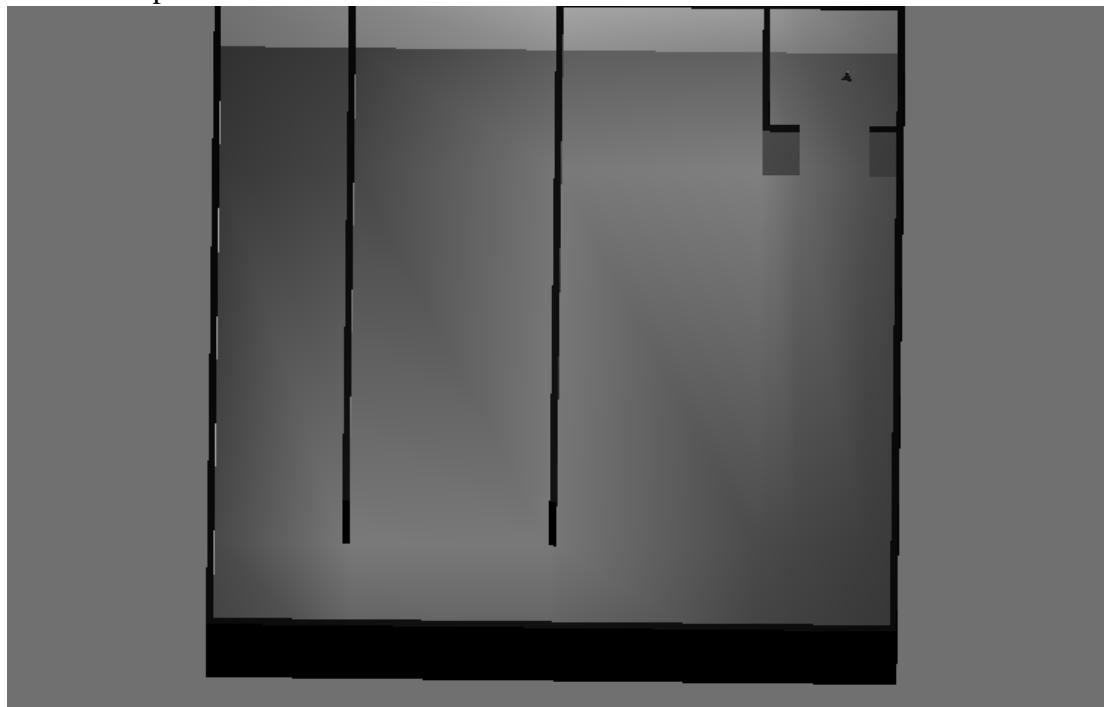
Figura 31: Drone no tempo inicial, antes da execução, no teste do algoritmo de ir ao destino. O drone está localizado no canto superior esquerdo.



Fonte: Autor

Seu destino é $(2.5, 27.5, 2.0)$. Após a execução do algoritmo, ele se encontra na posição $(2.4504, 27.5032, 2.0)$, em que pode ser visualizado na figura 32.

Figura 32: Drone no tempo final, após a execução do algoritmo. O Drone se encontra no canto superior direito.



Fonte: Autor

A altura é fixada em 2.0, pois o mapeamento 2D é mais prático de se fazer manualmente, e já atende o objetivo do projeto de automatizar o drone. Além disso, os pontos de início e final estão sempre com um valor inteiro adicionado de meia unidade. Isso se deve à discretização do mapa em quadrados, e então o drone deve ter seu ponto de chegada ou início sempre no centro de um quadrado.

A seguir, o resultado de mais testes feitos com o algoritmo de seguir rota.

Posição inicial	Destino final	Destino final real
(17.5, 23.5)	(25.5, 14.5)	(25.5016, 14.4004)
(23.5, 17.5)	(19.5, 4.5)	(19.4028, 4.5037)
(4.5, 26.5)	(17.5, 23.5)	(17.5155, 23.5012)
(23.5, 4.5)	(7.5, 26.5)	(7.5490, 26.4494)
(25.5, 4.5)	(11.5, 23.5)	(11.4803, 23.5193)
(12.5, 23.5)	(25.5, 5.5)	(25.4983, 5.5014)
(1.5, 1.5)	(25.5, 11.5)	((25.4986, 11.5004))
(25.5, 9.5)	(17.5, 23.5)	(17.4527, 23.5459)
(25.5, 4.5)	(21.5, 4.5)	(21.4001, 4.4996)
(25.5, 9.5)	(14.5, 23.5)	(14.5531, 23.4479)

A escolha de pontos iniciais e finais agora é mais restrita, pela discretização dos pontos possíveis. O script a seguir gera os possíveis pontos de início e fim.

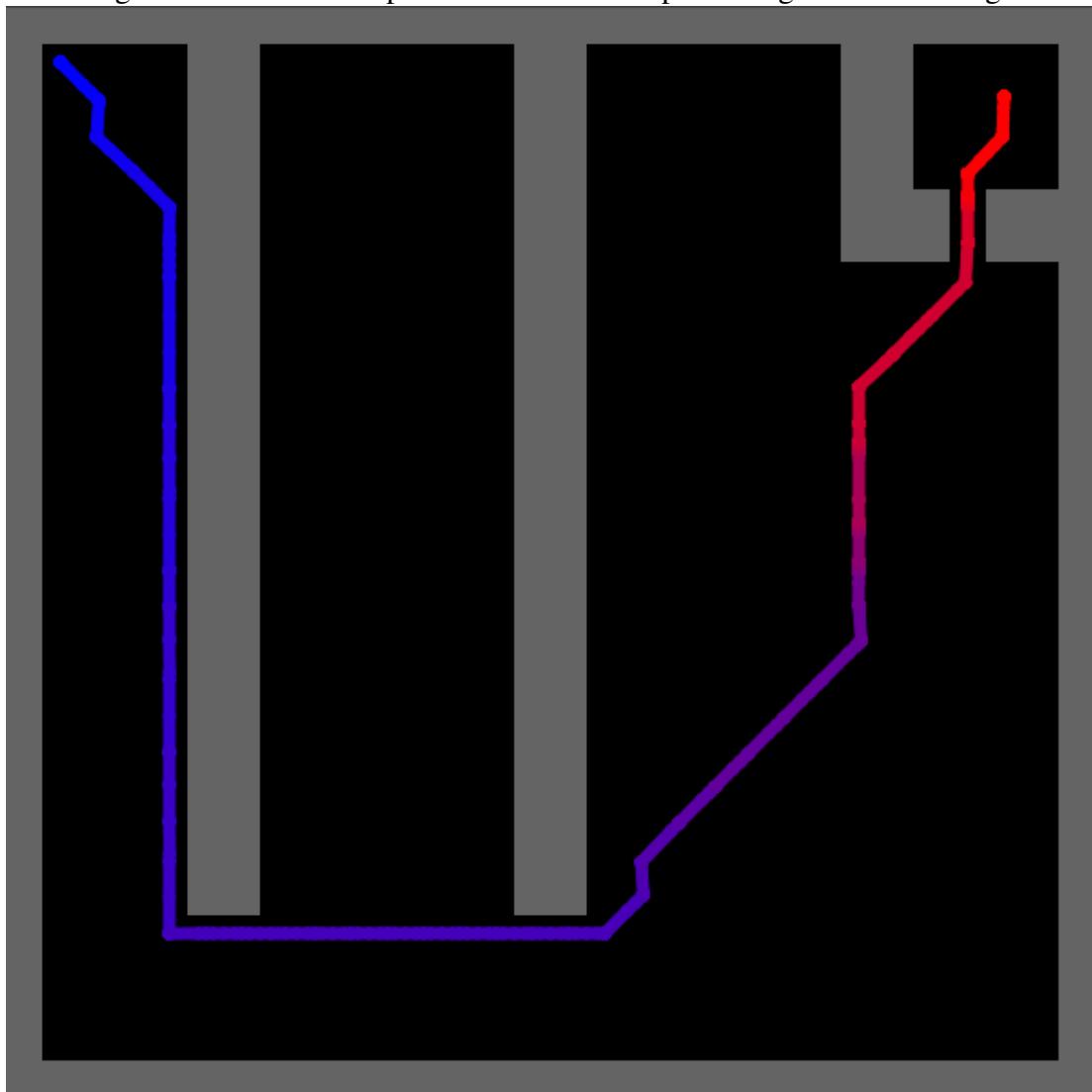
```
# all_points já possui todos os pontos possíveis
import random
print(random.choice(all_points))
```

Processing Processing é uma linguagem de programação de código aberto, com comunidade online. Sua saída pode ser em 2D ou 3D, e pode ser interativa e integrada a OpenGL, além de rica em bibliotecas e bem documentada. A fim de proporcionar ao usuário visualização do caminho feito pelo drone e também para depuração de testes, foi feito um código em Processing que seria o caminho feito pelo drone. Sua primeira versão lê um arquivo json que corresponde a um mapa e outro que corresponde aos pontos os quais o drone passou na simulação, ilustrados na figura seguinte: O caminho se inicia sempre na parte mais azul da linha, e se encerra sempre na parte mais

vermelha. As cores e sua cromatização são facilmente cambiáveis, embora esta combinação de cores seja de fácil detecção de tons intermediários, facilitando a detecção da sequência do caminho percorrido

Em suma, o programa necessita como entradas os arquivos json referentes ao mapa e ao caminho traçado, e proporciona como saída uma visualização gráfica deste caminho, com escala de cores conforme ele é percorrido.

Figura 33: Primeiro mapa e caminho criados pelo código em Processing.



Fonte: Autor

8.2 Algoritmo de melhor caminho

Dos diversos algoritmos de pathfinding, foi escolhido o A*, como explicado em 7.2.2. Ele é desenvolvido em node.js, linguagem diferente do simulador, mas a mesma na interface.

8.2.1 Algoritmo

O algoritmo é uma forma generalizada do algoritmo Dijkstra, como representado à seguir:

1. Adiciona o ponto inicial à lista aberta.
2. Repetir os seguintes passos:
 - (a) Procurar o nó com menor caminho entre à lista aberta.
 - (b) Colocar esse nó na lista fechada.
 - (c) Para cada nó alcançável desse novo nó:
 - i. Se estiver na lista fechada, ignorá-lo.
 - ii. Se não estiver na lista aberta, adicioná-lo, marcando o nó atual como seu pai.
 - iii. Se já estiver na lista aberta, alterar pai e custo de chegada caso seu custo seja menor que o atual
 - (d) Terminar quando:
 - i. O destino for colocado na lista fechada.
 - ii. A lista aberta estiver vazia.
3. Se foi encontrado destino, traçar caminho com os pais armazenados.

Fonte: (CUI; SHI, 2011)

Sua melhoria se refere à heurística utilizada, o que resulta em menor quantidade de passos realizada em nos testes realizados.

8.2.2 Testes

A biblioteca utilizada possui uma rotina de testes. Executando essa rotina, resultou-se no sucesso de todos os 57 testes executados. Ressalta-se que os testes do mapa e do algoritmo de A* estão inclusos. O anexo C demonstra a saída dos testes realizados.

8.3 Interface

8.3.1 Integração com o simulador

8.3.1.1 Localização

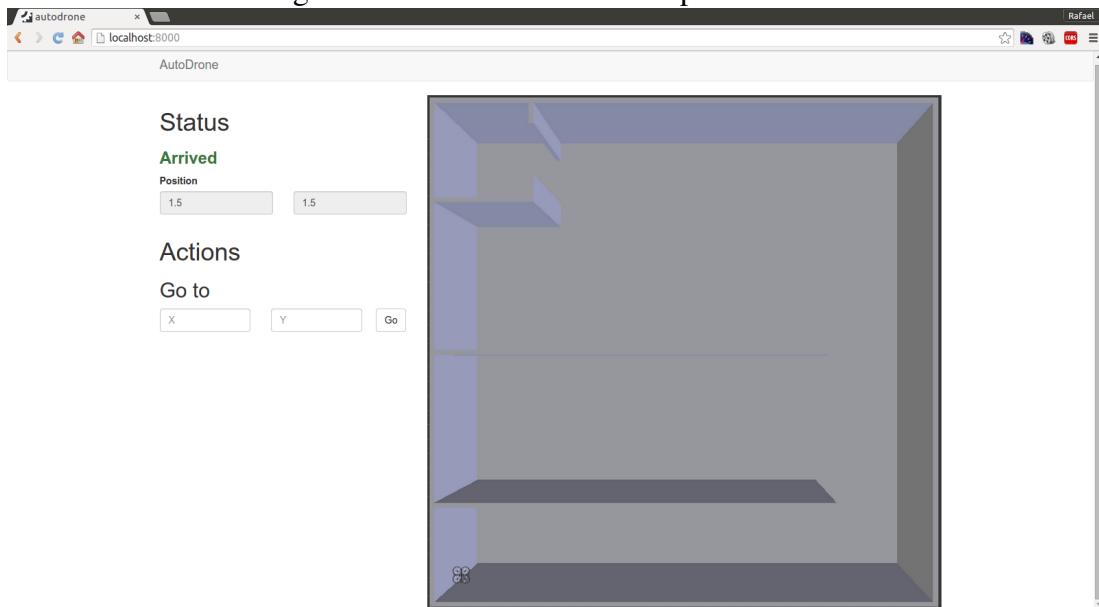
O simulador possui um sistema de localização preciso e absoluto. O desenvolvimento consiste em troca de informações entre a localização do drone no simulador, e a apresentação para o usuário.

A cada mudança de posição e/ou rotação do drone, o programa meteor.py se responsabiliza em atualizar o banco de dados da interface. A interface detecta as mudanças ocorridas 7.2.5 e então atualiza para o usuário sem a necessidade de recarregamento da página.

8.3.1.2 Controle

A seguir, a imagem que o usuário visualiza da interface.

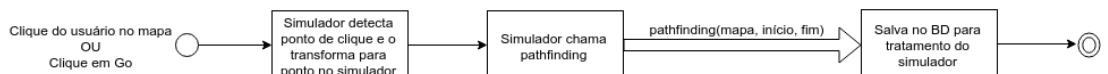
Figura 34: Interface de controle pelo usuário.



Fonte: Autor

A partir dela o usuário pode controlar o drone na interface. Para isso, basta clicar em algum ponto do mapa, ou preencher os campos de "Go to". Ao realizar uma dessas ações, o programará executará o seguinte fluxo de operações:

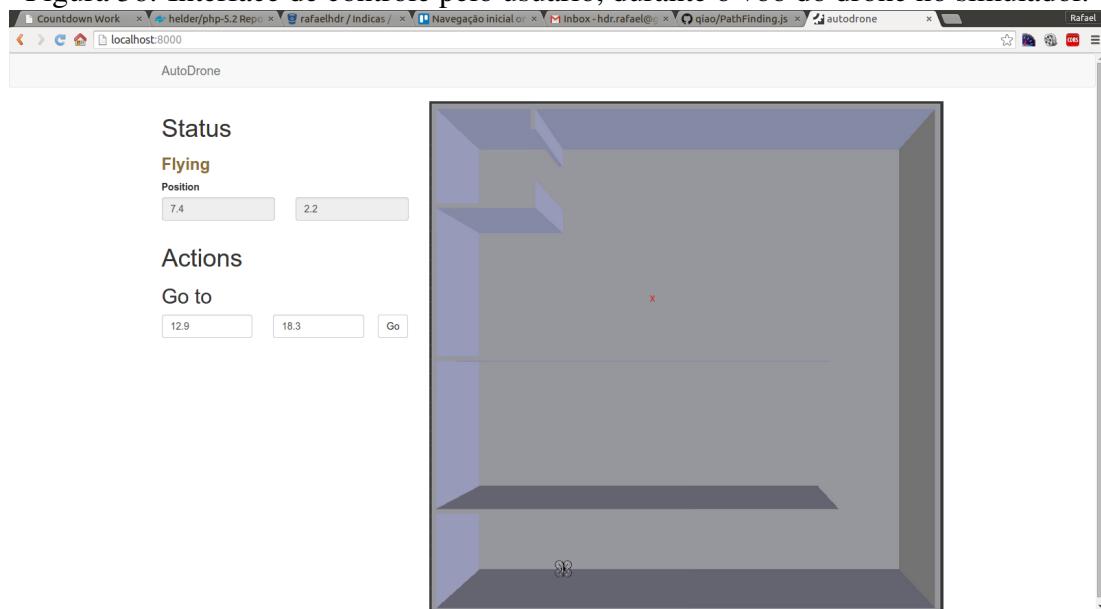
Figura 35: Interface mostrando a posição atual do Drone.



Fonte: Autor

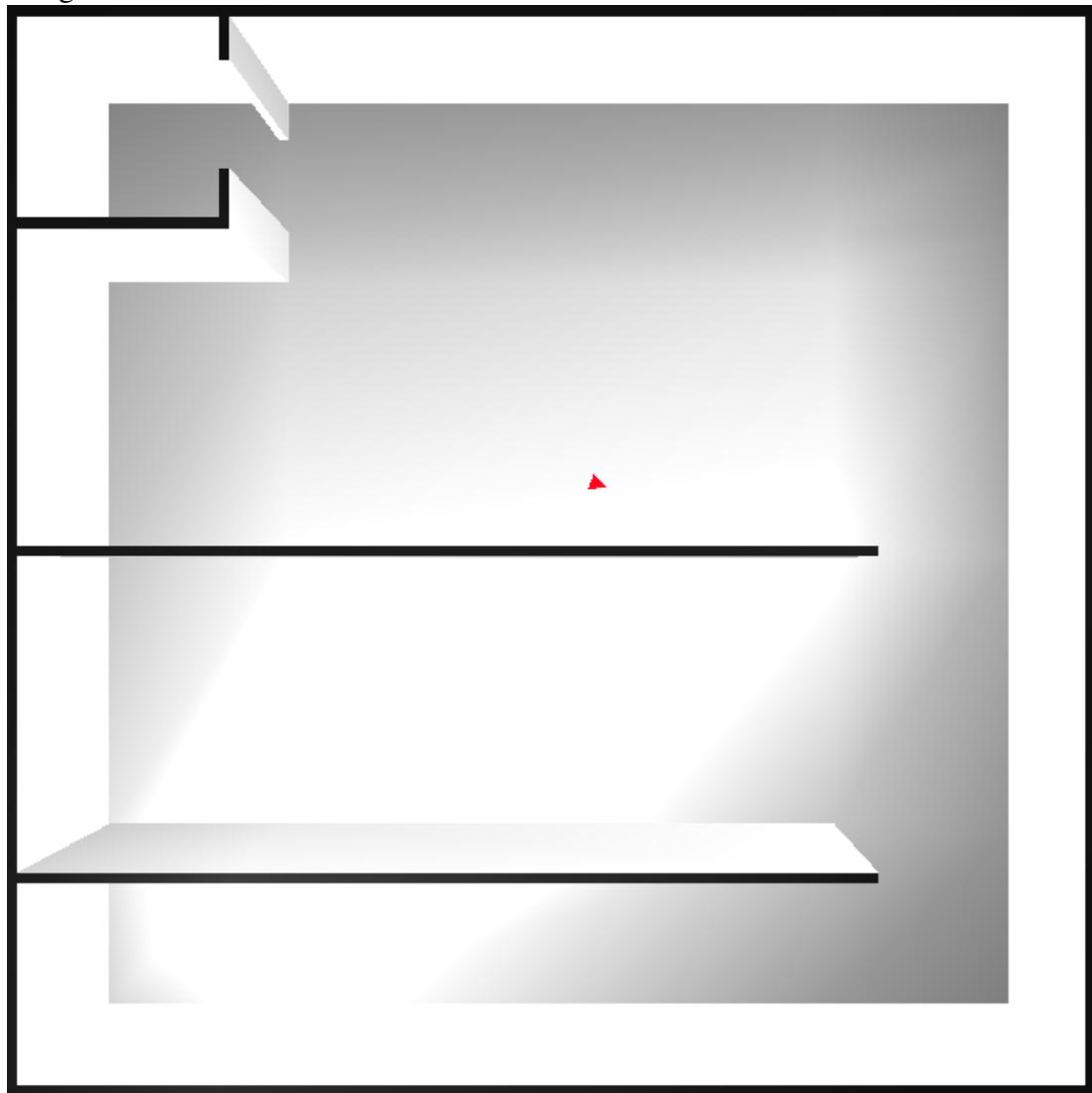
Inicialmente, o sistema detecta para onde o drone deve ir. Devido às diferentes escalas, calcula-se onde foi clicado (em pixels, sendo a origem no canto inferior esquerdo do mapa) e onde deve ir (em absoluto, para o simulador). O passo seguinte é chamar o algoritmo de pathfinding, que retornará a rota a ser percorrida. Ao final, essa rota é salva no banco de dados, sendo esta mudança detectada pelo simulador, e então executando os passos explicados na seção 8.2.

Figura 36: Interface de controle pelo usuário, durante o vôo do drone no simulador.



Fonte: Autor

Figura 37: O simulador funcionando simultaneamente com a interface do usuário.



Fonte: Autor

8.3.2 Testes da integração do controle do simulador

Para esses testes, os dados impressos são extraídos pela própria interface, de forma semelhante aos testes da seção 8.1.3.4.

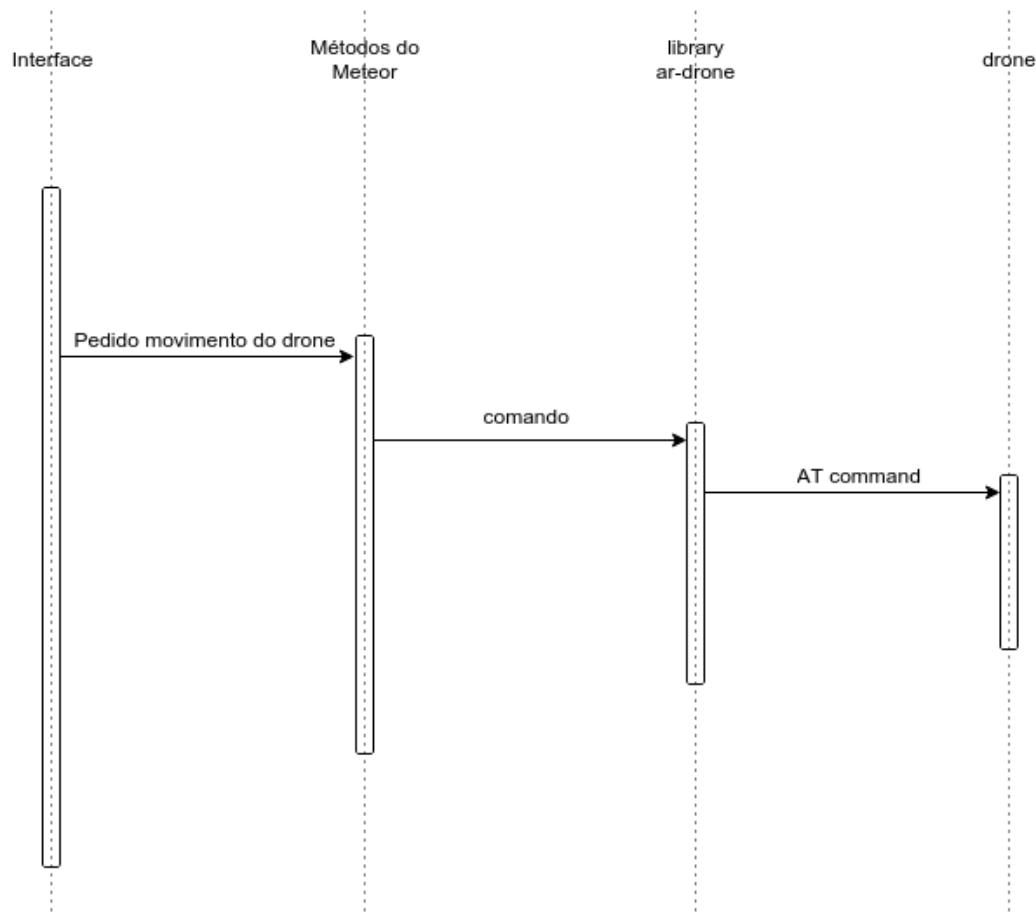
Posição inicial	Destino final	Destino final real
(17.5, 23.5)	(25.5, 14.5)	(25.5016, 14.4004)
(23.5, 17.5)	(19.5, 4.5)	(19.4028, 4.5037)
(4.5, 26.5)	(17.5, 23.5)	(17.5155, 23.5012)
(23.5, 4.5)	(7.5, 26.5)	(7.5490, 26.4494)
(25.5, 4.5)	(11.5, 23.5)	(11.4803, 23.5193)
(12.5, 23.5)	(25.5, 5.5)	(25.4983, 5.5014)
(1.5, 1.5)	(25.5, 11.5)	((25.4986, 11.5004)
(25.5, 9.5)	(17.5, 23.5)	(17.4527, 23.5459)
(25.5, 4.5)	(21.5, 4.5)	(21.4001, 4.4996)
(25.5, 9.5)	(14.5, 23.5)	(14.5531, 23.4479)

8.4 Drone

8.4.1 Controle por comandos

A interface do cliente possui diversos botões, listados em 1. Teclas do teclado são configuradas como atalho, que ao serem clicadas realizam o clique nos botões já criados, evitando duplicação de código. Ao ser clicado, chama-se um método para reconhecer o botão e enviá-lo ao drone, através da library ar-drone (GEISENDÖRFER, 2012).

Figura 38: Diagrama com a sequencia de operações entre o comando do usuário e a execução pelo drone.



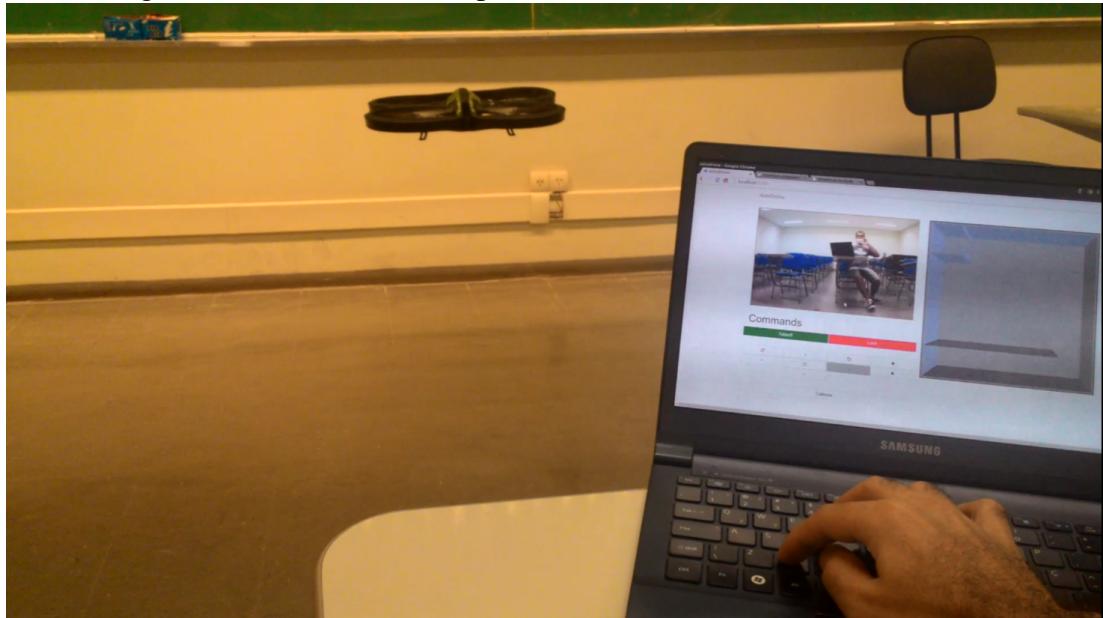
Fonte: Autor

8.4.2 Imagem para o usuário

Para replicar a imagem do drone na interface do usuário, foi utilizada a library drone stream (WEISSHUHN, 2012). Ela funciona criando um servidor, utilizando node.js, que renderiza uma página html com o vídeo. A interface replica essa página utilizando em seu interior um frame (tag html <frame>).

Tentativa com HTML5: A primeira tentativa de replicar envolve HTML5, utilizando tag video, que não funcionou com sucesso, devido à alta taxa de atraso.

Figura 39: Drone voando, replicando sua câmera frontal na interface.



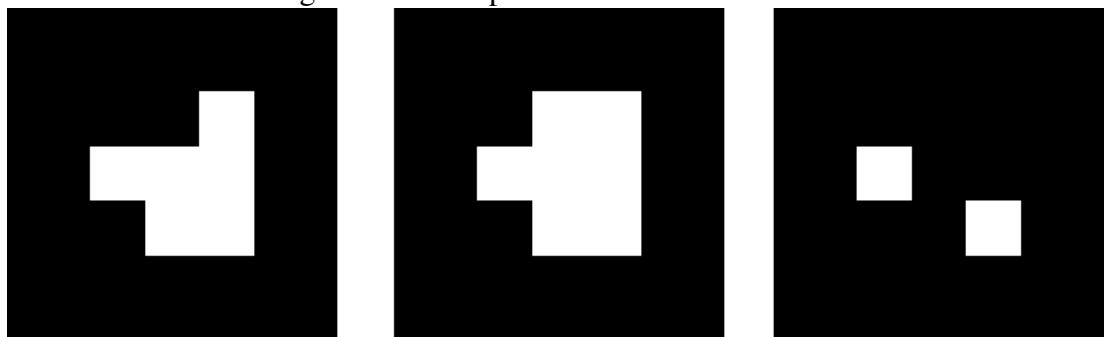
Fonte: Autor

8.5 Localização

8.5.1 ARToolKit

Os frames do vídeo são processados pelo ARToolKit, buscando identificar marcadores pré-mapeados em relação ao mapa do local. Estes marcadores são bidimensionais, e a dimensão destes pode ser aumentada de maneira a possibilitar maiores números de marcadores conforme a demanda. Para este projeto, marcadores 3x3 são suficientes, e portanto, utilizados. Essa configuração, se usada sem código de paridade, possibilita 64 marcadores distintos. Segue abaixo exemplos destes marcadores:

Figura 40: Exemplos de marcadores 2D 3x3



Fonte: ARToolKit

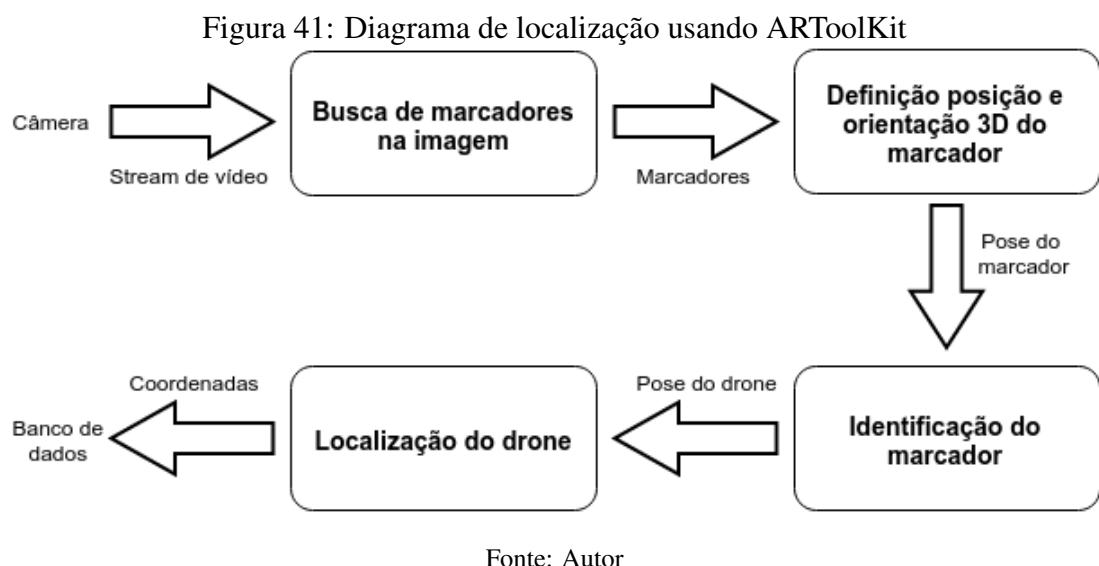
Divide-se, então, o processo em duas partes: mapeamento (offline) e localização (online).

A fase **offline** tem o seguinte procedimento:

- Escolha do local adequado - Deve-se escolher um local com iluminação conveniente, ou seja, que não atrapalhe a captura de vídeo, como luz excessiva ou insuficiente, padrões de imagem similares aos marcadores e pontos cegos que impossibilitam a colocação de um marcador, além dos pré-requisitos para um vôo seguro do drone.
- Mapeamento real e virtual - São tiradas as medidas do local (ou uma planta é utilizada como referência) para posterior mapeamento virtual, dotando-se de uma escala de acordo com o tamanho reservado ao mapa na interface.
- Posicionamento dos marcadores - Os marcadores são posicionados em locais de melhor visualização para o drone. Ou seja, com altura próxima àquela de estabilização, e distante de pontos cegos do ambiente.
- Atualização do arquivo de configuração - Uma vez fisicamente posicionados, os marcadores são adicionados ao arquivo config.dat, o qual contará com uma lista de marcadores, a posição e orientação de cada um e suas dimensões. Todos estes marcadores estarão referenciados a uma origem, em relação aos ângulos de

rotação em nos eixos euclidianos, e a distância entre a origem de cada marcador em relação à origem absoluta.

Na fase online, o sistema em execução utilizará o ARToolKit para realizar a leitura dos dados provenientes da câmera. O diagrama a seguir demonstra o processo de localização:



O ARToolKit busca, em princípio, identificar as bordas dos marcadores e através da sua pose, estima-se a pose em relação à câmera. Nesta etapa, também se calibra o nível de *threshold* que melhor se adequa à iluminação do ambiente. Em seguida, é identificado o marcador em questão, ou seja, é processada a imagem dentro das bordas e comparado aos arquivos base que contém o conjunto de marcadores previamente definidos. Assim, nesta etapa o sistema, de conhecimento da posição deste marcador, calcula a posição global da câmera e, por consequência, do drone.

Problemas: Os primeiros contatos com o ARToolKit se mostraram de difícil entendimento e utilização, e de pouca frequência de reclamação em fóruns. A solução encontrada foi o aprofundamento no estudo do GStreamer, que é uma biblioteca para manipulação de componentes de mídia. Seu melhor entendimento possibilitou a volta ao ARToolKit e continuidade de sua utilização.

8.5.1.1 Marcadores

O teste inicial com os marcadores constituiu na tentativa de que a câmera do drone, integrada ao sistema em fase também inicial, identificasse os marcadores. Para tal, foram impressos 11 marcadores barcodes 2D de dimensão 3x3 em folhas A4. O objetivo do teste foi verificar as condições e fatores limitantes de reconhecimento do marcador, tal como distância e ângulo da câmera em relação a ele.

O programa utilizado para este teste se vale de uma adaptação das bibliotecas de localização e de múltiplos patterns, que devolve a matriz de rotação e translação com a localização do drone, uma vez que um marcador virtual é configurado como ponto de coordenadas zero, e outro marcador real foi medido a uma distância de (2, 3, 0) em metros em relação ao zero, e dessa forma movimentou-se o drone arbitrariamente, observando a resposta do programa quanto à identificação e localização.

A primeira conclusão observada se deve a influência da luz: ambientes cuja iluminação é mais acentuada prejudicam o reconhecimento do pattern na medida em que os pontos escuros, como os da borda do pattern, aparecem no sistema serem pontos mais claros. Em outras palavras, o sistema, que procura por altos valores de cinza no formato de um quadrado, não os identificará pois eles terão um valor mais próximo ao branco. Ao controlar a luz do ambiente de forma que ela não 'estoure' a luz nas imagens, o contraste entre branco e preto fica mais acentuado e assim o reconhecimento se torna mais eficiente. A maior distância coberta registrada foi de aproximadamente 7 metros entre o marcador e o drone, em ângulo de 30 graus em relação ao centro da imagem, enquanto a menor distância é limitada pelo enquadramento do marcador no quadro de imagem, o que significa cerca de 30 centímetros de distância em linha reta. Ressalta-se que mesmo em distâncias curtas, próximas a esta distância mínima de reconhecimento, a iluminação excessiva causa o não reconhecimento do código.

Dada tal conclusão, fica evidente que a iluminação é um fator limitante e que deve

ser controlado junto à escolha dos locais de teste e utilização, influenciando direta e decisivamente na distância entre os marcadores e proporcionalmente no número de marcadores a serem utilizados, lembrando que para barcodes 3x3 sem checksum o número máximo possível é de 64 códigos diferentes.

Em relação à localização efetiva, foi possível verificar que o programa consegue realizar medições coerentes com os 2 metros em x e 3 metros em y que foram configurados entre a referência e o marcador real. Ajustes matemáticos são necessários em relação à rotação e orientação dos marcadores. O resultado geral, neste aspecto, foi considerado satisfatório e evolutivo, e a integração entre câmera do drone e sistema, concluído.

A última observação referente aos testes se dá a existência de um atraso de imagem, principalmente nos primeiros segundos de execução e alguns travamentos do sistema aleatórios, observação esta que se levará em conta nas próximas atualizações dos códigos.

8.5.1.2 Integração dos marcadores com a câmera frontal do Drone

Antes dos testes efetivos, deve-se realizar a calibração dos parâmetros da câmera. Para tal, utiliza-se o módulo de calibração já proveniente do ARToolKit, cujo procedimento inclui um tabuleiro de xadrez cujo tamanho de lado deve ser especificado como entrada do sistema. A câmera, então, é ativada e se deve posicionar o tabuleiro de xadrez em diferentes posições, distâncias e ângulos relativos a câmera, buscando cobrir a maior amplitude possível de possibilidades. O programa detecta todos os quadrados do tabuleiro e, quando o faz, pinta todos estes pontos na cor vermelha, significando que consegue identificar todos os quadrados do tabuleiro. Então, uma vez satisfeita essa condição e os operadores satisfeitos com o posicionamento na tela, é capturada a tela nesta condição, e os dados retirados do processamento da imagem considerados para a confecção das matrizes características da câmera em uso.

8.5.1.3 Teste 1: Leitura do marcador

Foi testada a identificação dos marcadores nos seguintes formatos e tamanhos:

- A4 único - Um marcador grande em folha A4, impresso em condições comuns.
Tamanho: 18,5 cm.
- A4 duplo - Dois marcadores médios em folha A4, impressos em condições comuns. Tamanho: 13 cm
- Tablet e celular - Visualização através da tela de um smartphone ou tablet. Tamanho: variável.

Ao realizar a simples impressão em papel sulfite A4, o marcador não possui a rigidez da qual se espera para a leitura de um drone. Nos testes iniciais, então, foram utilizados materiais adesivos para esticar a folha. Para os testes futuros, considerando-se a precisão, é mandatório utilizar folhas de material mais rígido, como papel cartão.

Já nos aparelhos eletrônicos, os testes ocorreram como esperado, embora a reflexão causada pela tela possa prejudicar a leitura, este procedimento consiste apenas em alternativa de teste, portanto descartada para o produto final.

O fator mais relevante para a identificação, inclusive a princípio mais influente que a distância, é a iluminação. Ao posicionar o marcador em local tanto em alta quanto baixa iluminação prejudica o contraste da imagem, dificultando o reconhecimento do marcador. Dessa forma, ao realizar o mapeamento do local, o posicionamento dos marcadores ao longo do ambiente deve levar em conta os fatores de iluminação como altamente relevantes.

8.5.1.4 Teste 2: Multimarcadores

O teste consiste em utilizar a função de multimarcadores do ARToolKit, cujo objetivo é verificar a precisão do sistema de localização, além de proporcionar noções

iniciais de distância e medições em relação aos marcadores.

Para tal, um arquivo de configuração config.dat provê as informações de distância e ângulo em relação a uma origem definida, além de seu tamanho. Então, dispõe-se o conjunto de marcadores no ambiente de acordo com o que foi descrito no arquivo de configuração. O sistema deve desenhar em realidade aumentada um cubo azul do mesmo tamanho do marcador logo acima dele. Caso o cubo se apresente vermelho, ele não está sendo identificado corretamente, porém entende-se do arquivo de configuração que ele deveria estar naquela posição desenhada.

A partir deste teste, foi possível obter informações do funcionamento do ARTool-Kit de maneira prática, além do manuseio do arquivo de configuração. Outra descoberta importante deste teste consiste no entendimento dos referenciais, como a origem e a câmera. Tal entendimento é necessário para o manuseio das matrizes de transformação, assim proporcionando na saída localizações coerentes. A princípio, o sistema dá a matriz de transformação em relação à câmera como origem das coordenadas. O propósito do projeto é justamente o contrário, saber a posição da câmera do drone em relação ao sistema de coordenadas previsto pelo sistema de marcadores. Desta forma, inverte-se a matriz de transformação e então se obtém o desejado.

8.5.1.5 Teste 3: Ajuste de coordenadas

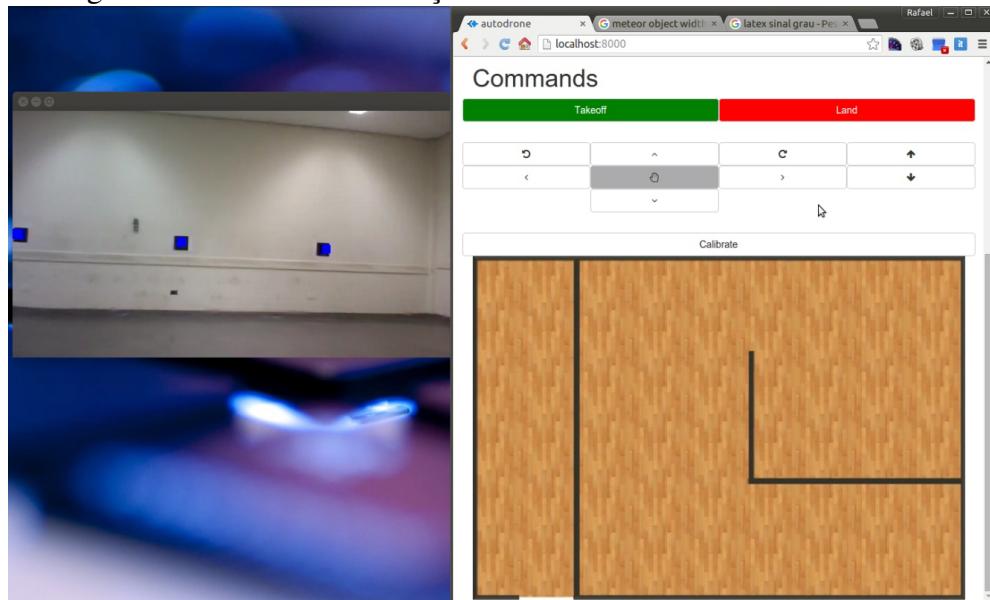
O terceiro teste - que está ligado aos dois anteriores - consiste em configurar manualmente o arquivo config.dat com a posição de diversos marcadores e posicionar manualmente o drone em locais arbitrários, comparando a medição manual, com trenas e régua - com a medição de saída do programa de teste. O objetivo do teste é verificar o grau de imprecisão da câmera, bem como identificar distâncias e espaçamentos ideais para a distribuição dos marcadores no ambiente.

8.5.2 Testes

ARToolKit Após um aprofundamento nos exemplos disponíveis, foram feitas as devidas adaptações para sua utilização neste projeto.

A seguir, a captura de tela com rastreamento da imagem pelo ARToolKit.

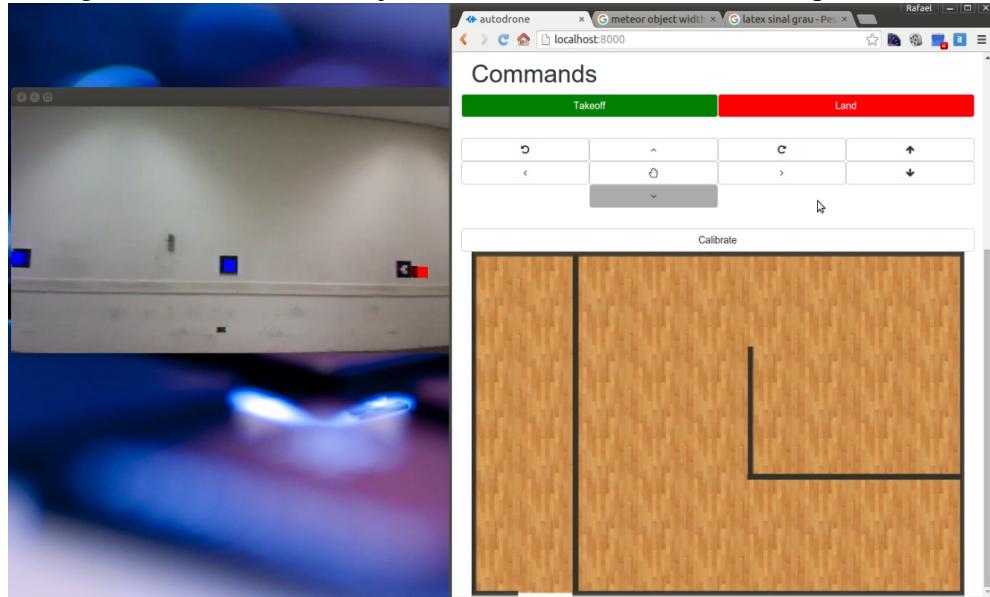
Figura 42: Teste com a detecção de marcadores durante o vôo do drone.



Fonte: Autor

Inicialmente, é coletada a imagem, e então processada para encontrar os marcadores. Encontrando no mínimo um marcador, é possível localizar a câmera e então saber onde se encontra o Drone.

Figura 43: Teste de detecção de marcadores, detectando-os parcialmente.



Fonte: Autor

A seguir, a matriz de transformação relacionando o primeiro marcador e o drone.

$$\begin{array}{cccc}
 0,780977 & 0,066419 & -0,621018 & 100,626791 \\
 -0,077211 & -0,976434 & -0,201531 & 34,078591 \\
 -0,619769 & 0,205341 & -0,757444 & 1484,833967
 \end{array}$$

Com ela, é possível saber onde o drone se encontra em um ambiente previamente mapeado.

9 CONSIDERAÇÕES FINAIS

9.1 Dificuldades

Com o decorrer do projeto, naturalmente obstáculos foram enfrentados, os quais são listados a seguir, com o intuito de auxiliar no direcionamento de projetos decorrentes deste ou similares.

Definição do escopo: Na fase de pesquisa, a grande varieadade de tecnologias e aplicações no que tange à localização indoor dificulta a definição de escopo. Mesmo buscando definir requisitos e adequar as escolhas a eles, técnicas de localização e mapeamento simultâneos foram cogitados. A busca de profissionais experientes na área se mostrou fundamental, contribuindo significativamente para o encurtamento do processo de entendimento do problema, seu contexto e as dificuldades associadas que, em um primeiro contato, o pesquisador tem dificuldade para enxergar.

OpenCV: Durante o desenvolvimento de uma tecnologia de localização indoor, a primeira tentativa consistiu em calcular a translação entre frames consecutivos de uma câmera monocular. Os primeiros testes identificaram imprecisões consideráveis na localização. Assim como outras abordagens vistas posteriormente (e descartadas pelo motivo citado), o uso de sistemas estéreo, ou seja, que são capazes de medir a profundida dos pontos da imagem, é mandatório.

Estabilidade: Apesar de possuir diversos sensores, além ainda de um sistema de controle PID (proporcional-integral-derivative controller) nas bibliotecas utilizadas, observou-se grande instabilidade do drone. Em testes realizados em ambientes abertos,

o drone é muito suscetível a variações devido ao vento, sendo raros momentos em que o drone consegue estabilidade a ponto de apenas pairar no ar. Foram utilizados os aplicativos tanto em Android quanto em iOs, e ambos apresentaram dificuldades para aprendizado e controle. Com este tipo de controlador, porém, ainda se pode evitar batidas. Tal fato não foi observado ao passar o controle para sistema, em seus primeiros testes. Ao aproximar o drone da parede, o deslocamento de ar causado pelas hélices acaba por refletir e desestabilizar o veículo, por muitas vezes se chocando. Outro drone do mesmo modelo foi utilizado e seu comportamento foi comparado, porém sem sucesso. O projeto, então, buscou minimizar os efeitos da instabilidade e controlar o ambiente de vôo de modo a garantir a segurança dos observadores. Calibragens também foram usadas para diminuir os problemas de estabilidade.

Bateria: A duração da bateria é bastante curta para ambientes de utilização contínua, além da demora na recarga. Para de fato implementar um sistema, é necessário procurar alternativas de bateria e recarregamento, além de explorar formas de otimizar o processamento dos dados.

Integração e instalações: Foram enfrentadas dificuldades de integração dos módulos e de instalação das aplicações e bibliotecas, devido a incompatibilidades entre estes e as máquinas utilizadas.

9.2 Objetivos alcançados

O resultado final atende aos objetivos especificados, validando o uso do ARToolKit para a localização indoor com precisão satisfatória, dentro dos parâmetros especificados. Os marcadores foram devidamente reconhecidos, validando os testes preparatórios. A interface, por fim, integra os módulos do projeto, proporcionando ao usuário a experiência de controlar o drone através de um novo canal de comunicação.

Em relação à pesquisa, ressalta-se sua abrangência. Conta com diversos métodos

de localização indoor, pathfinding para geração de rotas em um ambiente 2D, desenvolvimento de ferramentas de computação gráfica do simulador e técnicas de controle. Espera-se contribuir com a comunidade através das informações reunidas e também produzidas pela documentação.

9.3 Desafios futuros

Localização sem marcadores A escolha do ARToolKit para localização se deve principalmente à sua simplicidade de implementação. Este fato limita o projeto para ambientes que permitam a instalação de marcadores em suas paredes. Um possível passo adiante é a possibilidade de utilizar ferramentas que não necessitem de marcadores. Uma das ferramentas encontradas é o PTAM (Parallel Tracking and Mapping) (KLEIN, 2008). Há também métodos de localização que procuram aspectos inerentes ao ambiente, como ortogonalidades, em (WERNECK; COSTA, 2010).

Desvio de objetos O projeto não possui nenhum algoritmo de identificação e desvio de objetos não-mapeados. Técnicas de visão computacional diferentes podem ser aplicadas para tal, ou ainda adição de sensores de profundidade. Uma alternativa para este tema está em (BARRY; TEDRAKE, 2014).

Segurança - Controle em caso de perda a conexão Ao perder a conexão, o drone se comporta de maneira a permanecer voando, e então pousar quando sua bateria se esgota. A possibilidade de controle do drone, como alteração no sistema ou adição de um módulo microcontrolador são alternativas, porém não há suporte por parte do fabricante para tal. Por parte do sistema de controle, é viável criar um sistema de aviso ao usuário em caso de perda de conexão, informando também a última posição em que foi visto.

Modularização Um método de localização por câmera monocular foi implementada. Uma possibilidade é modularizá-la, de forma que um ou mais sistemas de loca-

lização funcionem paralelamente, havendo maior redundância de dados, ou seja, mais entradas para localização precisa. Esta modularização pode incluir, evidentemente, outros sensores, mas também pode ter portabilidade, ou seja, funcionar em outros drones ou veículos robóticos.

REFERÊNCIAS

- AEROVIROIMENT. *Public Safety Commercial UAS Solutions - AeroVironment, Inc.* 2015. Disponível em: <https://www.avinc.com/public-safety>. Acesso em: março de 2015.
- ALBERS-SCHOENBERG, Y. *Urban MAV Localization from Google Street View Data* — University of Zurich, Março 2013.
- AMAZON. *Amazon Prime Air*. 2015. Disponível em: <http://www.amazon.com/b?node=8037720011>. Acesso em: março de 2015.
- ANAC. *Estabele as regras para a operação do aeromodelismo no Brasil*. 1999. Disponível em: <http://www2.anac.gov.br/biblioteca/portarias/port207STE.pdf>. Acesso em: março de 2015.
- ASAHI KASEI MICRODEVICES. *AK8975/AK8975C 3-axis Electronic Compass*. [S.I.], 2010.
- ATLASSIAN. *Bitbucket is the Git solution for professional teams*. 2015. Disponível em: <https://bitbucket.org/>. Acesso em: março de 2015.
- BARRY, A. J.; TEDRAKE, R. Pushbroom stereo for high-speed navigation in cluttered environments. Julho 2014.
- BLENDER, F. *Introduction to Game Engine*. 2015. Disponível em: http://www.blender.org/manual/game_engine/. Acesso em: junho de 2015.
- BOSCH. *BMA150 Data Sheet*. [S.I.], 2008.
- _____. *BMP180 Data Sheet*. [S.I.], 2013.
- BRY, A.; BACHRACH, A.; ROY, N. State estimation for aggressive flight in gps-denied environments using onboard sensing. *IEEE International Conference on Robotics and Automation*, 2012.
- CONSERVANCY, S. F. *Git*. 2015. Disponível em: <https://git-scm.com/>. Acesso em: março de 2015.
- CUI, X.; SHI, H. A*-based pathfinding in modern computer games. *IJCSNS International Journal of Computer Science and Network Security*, v. 11, n. 1, Janeiro 2011.
- DAVISON, A. J.; MONTIEL, J.; STRASDAT, H. Scale drift-aware large scale monocular slam. 2012.

- DEBERGALIS, M. *Introducing DDP*. 2015. Disponível em: <http://info.meteor.com/blog/introducing-ddp>. Acesso em: agosto de 2015.
- DIJKSHOORN, N. *Simultaneous localization and mapping with the AR.Drone*. Dissertação (Mestrado) — Universiteit van Amsterdam, 2012.
- ENGEL, J.; STURM, J.; CREMERS, D. *Scale-Aware Navigation of a Low-Cost Quadrocopter with a Monocular Camera*. [S.I.], 2014.
- ESCHENAUER, L. *Provides key building blocks to create autonomous flight applications with the nodecopter (AR.Drone)*. 2013. Disponível em: <https://github.com/eschnou/ardrone-autonomy>. Acesso em: abril de 2015.
- FRANKE, U. E. *Civilian Drones: Fixing an Image Problem?* Janeiro 2015. Disponível em: <http://isnblog.ethz.ch/security/civilian-drones-fixing-an-image-problem>. Acesso em: maio de 2015.
- GEISENDÖRFER, F. *A node.js client for controlling Parrot AR Drone 2.0 quad-copters*. 2012. Disponível em: <https://github.com/felixge/node-ar-drone>. Acesso em: abril de 2015.
- GOHOBBY. *Uso Profissional | GoHobby*. 2015. Disponível em: <http://www.gohobby.com.br/Uso-Profissional>. Acesso em: março de 2015.
- GSTREAMER. *Open source multimedia framework*. 2015. Disponível em: <http://gstreamer.freedesktop.org/>. Acesso em: julho de 2015.
- HARNISCH, H. *A meteor client for python*. 2014. Disponível em: <https://github.com/hharnisc/python-meteor>. Acesso em: agosto de 2015.
- HERNANDEZ, O. et al. Position location monitoring using ieee 802.15.4/zigbee technology. 2009.
- INVENSENSE. *IMU-3000 Motion Processing Unit Product Specification Rev 1.0*. [S.I.], 2010.
- KLEIN, G. *Parallel Tracking and Mapping for Small AR Workspaces (PTAM)*. Outubro 2008. Disponível em: <http://www.robots.ox.ac.uk/~gk/PTAM/>. Acesso em: junho de 2015.
- LABS, P. *Pozyx - Accurate positioning*. [S.I.], 2015.
- LITTELFUSE. *Use of Low Resistivity Surface Mount PPTC in Li-ion Polymer Battery Packs*. [S.I.], 2012.
- METEOR, D. G. *DDP Specification*. 2015. Disponível em: <https://github.com/meteor/meteor/blob/devel/packages/ddp/DDP.md>. Acesso em: agosto de 2015.
- _____. *Meteor*. 2015. Disponível em: <http://meteor.com/>. Acesso em: abril de 2015.

- _____. *Meteor DDP*. 2015. Disponível em: <https://www.meteor.com/ddp>. Acesso em: agosto de 2015.
- _____. *Meteor Livequery*. 2015. Disponível em: <https://www.meteor.com/livequery>. Acesso em: abril de 2015.
- MONAJJEMI, M. *ardrone autonomy*. [S.I.], 2014.
- MONGO. *Mongo DB*. 2015. Disponível em: <http://www.mongodb.org/>. Acesso em: abril de 2015.
- NIXON, M. S.; AGUADO, A. S. *Feature Extraction and Image Processing*. [S.I.]: Newnes, 2002.
- _____. *Feature Extraction and Image Processing*. [S.I.]: Newnes, 2002.
- NODE.JS, F. *Node.js*. 2015. Disponível em: <http://nodejs.org/>. Acesso em: abril de 2015.
- PAPARAZZI. *Paparazzi The Free Autopilot*. 2015. Disponível em: <http://wiki.paparazziuav.org/>. Acesso em: maio de 2015.
- PARROT. *AR.Drone 2.0. Parrot new wi-fi quadricopter - AR.Drone.com - HD Camera - Civil drone - Parrot*. 2015. Disponível em: <http://ardrone2.parrot.com/>. Acesso em: fevereiro de 2015.
- PHOTOJOJO. *Use Photo Drones for Fun Family Portraits*. 2015. Disponível em: <http://content.photojojo.com/guides/family-photos-with-drones/>. Acesso em: março de 2015.
- PISKORSKI, S. et al. *AR.Drone Developer Guide*. 2.0. ed. [S.I.], Dezembro 2012. Disponível em: <https://svn.ardrone.org/wiki/ardrone-api>. Acesso em: março de 2015.
- QUADCOPTERARENA. *The history of drones and quadcopters*. 2015. Disponível em: <http://quadcopterarena.com/the-history-of-drones-and-quadcopters/>. Acesso em: março de 2015.
- ROBOTICS, F. O. S. *Robot Operational System*. [S.I.], 2015.
- SAMSUNG. *Samsung Notebook Series 9*. [S.I.], 2012. Disponível em: http://www.samsung.com/hk/business-images/resource/data-sheet/2013/01/Series_9-1.pdf. Acesso em: março de 2015.
- SHEN, S.; MICHAEL, N.; KUMAR, V. *Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV*. Maio 2011. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.460.8016&rep=rep1&type=pdf>.
- SHRIDHAR, M.; NEO, K.-Y. Monocular slam for real-time applications on mobile platforms. 2015.

TECHNOLOGIES, I. *Gantter - web-based project scheduling made easy*. 2015. Disponível em: <http://www.gantter.com/>. Acesso em: março de 2015.

TERRA, P. *Realidade de drones no Brasil, mas não em sua legislação*. 2015. Disponível em: <http://tecnologia.terra.com.br/drones-no-brasil-ja-sao-uma-realidade-mas-a-legislacao-nao,dc85020d81bca410VgnVCM3000009af154d0RCRD.html>. Acesso em: março de 2015.

TEXAS INSTRUMENTS. *1 MHz, 85 microA Op Amps*. [S.l.], 2009.

_____. *OMAP3630*. 2015. Disponível em: <http://www.ti.com/product/omap3630>. Acesso em: abril de 2015.

TIPLING, B.; COHEN, G. *A meteor client for python*. 2013. Disponível em: <http://meteorpedia.com/read/Meteorpedia>. Acesso em: agosto de 2015.

_____. *DDP Clients*. 2015. Disponível em: http://meteorpedia.com/read/DDP_Clients#Python. Acesso em: agosto de 2015.

TRELLO, I. *Trello is the free, flexible, and visual way to organize anything with anyone*. 2015. Disponível em: <http://www.trello.com/>. Acesso em: março de 2015.

VAUGHAN, B.; LAMB, P.; YOUNG, W. *Open Source Augmented Reality SDK | ARToolKit.org*. 2015. Disponível em: <http://artoolkit.org/>. Acesso em: julho de 2015.

WEISSSHUHN, B. *Realtime video feed from ar.parrot 2.0 drones in pure javascript. Successor of nodecopter-stream*. Dezembro 2012. Disponível em: <https://github.com/bkw/node-dronestream>. Acesso em: junho de 2015.

WERNECK, N. L.; COSTA, A. H. R. Monocular visual mapping with the fast hough transform. *VI Workshop de Visão Computacional*, 2010.

XU, X. *A comprehensive path-finding library for grid based games*. 2011. Disponível em: <https://github.com/qiao/PathFinding.js>. Acesso em: junho de 2015.

APÊNDICE A - SUMÁRIO DOS AR COMMANDS

AT command	Arguments ¹	Description
AT*REF	input	Takeoff/Landing/Emergency stop command
AT*PCMD	flag, roll, pitch, gaz, yaw	Move the drone
AT*PCMD_MAG	flag, roll, pitch, gaz, yaw, psi, psi accuracy	Move the drone (with Absolute Control support)
AT*FTRIM	-	Sets the reference for the horizontal plane (must be on ground)
AT*CONFIG	key, value	Configuration of the AR.Drone 2.0
AT*CONFIG_IDS	session, user, application ids	Identifiers for AT*CONFIG commands
AT*COMWDG	-	Reset the communication watchdog
AT*CALIB	device number	Ask the drone to calibrate the magnetometer (must be flying)

ardrone_{developer_guide},dk20

APÊNDICE B - CONFIGURAÇÕES GERAIS DO DRONE

8.4 General configuration

All the configurations are given accordingly to the *config_keys.h* file order.

In the API examples, the *myCallback* argument can be a valid pointer to a callback function, or a NULL pointer if no callback is needed by the application.

GENERAL:num_version_config _____ *CAT_COMMON | Read only*

Description :
Version of the configuration subsystem (Currently 1).

GENERAL:num_version_mb _____ *CAT_COMMON | Read only*

Description :
Hardware version of the drone motherboard.

GENERAL:num_version_soft _____ *CAT_COMMON | Read only*

Description :
Firmware version of the drone.

GENERAL:drone_serial _____ *CAT_COMMON | Read only*

Description :
Serial number of the drone.

GENERAL:soft_build_date _____ *CAT_COMMON | Read only*

Description :
Date of drone firmware compilation.

GENERAL:motor1_soft _____ *CAT_COMMON | Read only*

Description :
Software version of the motor 1 board. Also exists for motor2, motor3 and motor4.

GENERAL:motor1_hard _____ *CAT_COMMON | Read only*

Description :
Hardware version of the motor 1 board. Also exists for motor2, motor3 and motor4.

GENERAL:motor1_supplier _____ *CAT_COMMON | Read only*

Description :
Supplier version of the motor 1 board. Also exists for motor2, motor3 and motor4.

APÊNDICE C - TESTES DO PATHFINDING

Resultados parcial dos testes automatizados de pathfinding. O resultado foi positivo em todos os testes.

Utiliza-se a library PathFinding.js (XU, 2011).

```
Grid
  generate without matrix
    ✓ should have correct size
    ✓ should set all nodes' walkable attribute
  generate with matrix
    ✓ should have correct size
    ✓ should initiate all nodes' walkable attribute
    ✓ should be able to set nodes' walkable attribute
    ✓ should return correct answer for position validity query
    ✓ should return correct neighbors
  generate with matrix and no width or height
    ✓ should have correct size

AStar
  ✓ should solve maze 1
  ✓ should solve maze 2
  ✓ should solve maze 3
  ✓ should solve maze 4

BreadthFirst
  ✓ should solve maze 1
  ✓ should solve maze 2
  ✓ should solve maze 3
  ✓ should solve maze 4

Dijkstra
  ✓ should solve maze 1
  ✓ should solve maze 2
  ✓ should solve maze 3
  ✓ should solve maze 4
```

Fonte: Autor