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

ESTUDO DE ALGORITMOS DE VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO E DESVIO DE OBJETOS EM TEMPO REAL: UMA APLICAÇÃO PARA QUADROTORES

WANDER MENDES MARTINS

Itajubá, 18 de dezembro de 2018

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

WANDER MENDES MARTINS

ESTUDO DE ALGORITMOS DE VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO E DESVIO DE OBJETOS EM TEMPO REAL: UMA APLICAÇÃO PARA QUADROTORES

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

Área de Concentração: Sistemas de Computação

Orientador: Prof. Dr. Alexandre Carlos Brandão Ramos

18 de dezembro de 2018
Itajubá

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

ESTUDO DE ALGORITMOS DE VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO E DESVIO DE OBJETOS EM TEMPO REAL: UMA APLICAÇÃO PARA QUADROTORES

WANDER MENDES MARTINS

Dissertação aprovada por banca examinadora em 07 de Dezembro de 2018, conferindo ao autor o título de **Mestre em Ciência e Tecnologia da Computação.**

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

Itajubá
2018

WANDER MENDES MARTINS
ESTUDO DE ALGORITMOS DE VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO E DESVIO DE OBJETOS EM TEMPO REAL: UMA APLICAÇÃO PARA QUADROTORES/ WANDER MENDES MARTINS. – Itajubá, 18 de dezembro de 2018-
92 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Alexandre Carlos Brandão Ramos

Dissertação (Mestrado)
Universidade Federal de Itajubá
Programa de Pós-Graduação em Ciência e Tecnologia da Computação, 18 de dezembro de 2018.

1. Visão Computacional. 2. Desvio de Obstáculos. 3. Inteligência Artificial. I. Orietador Prof. Dr. Alexandre Carlos Brandão Ramos. II. Universidade Federal de Itajubá. III. POSCOMP. IV. Estudo de Algoritmos de Visão Computacional para Identificação e Desvio de Objetos em Tempo Real: Uma Aplicação para Quadrotores

CDU 07:181:009.3

WANDER MENDES MARTINS

ESTUDO DE ALGORITMOS DE VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO E DESVIO DE OBJETOS EM TEMPO REAL: UMA APLICAÇÃO PARA QUADROTORES

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

Trabalho aprovado. Itajubá, 07 de Dezembro de 2018:

Prof. Dr. Alexandre Carlos Brandão Ramos
Orientador

Prof. Dr. Hildebrando Ferreira de Castro Filho, membro externo - ITA

Prof. Dr. Roberto Affonso da Costa Júnior, membro - UNIFEI

Prof. Dr. Roberto Claudino da Silva, membro - UNIFEI

Itajubá
18 de dezembro de 2018

Agradecimentos

Dedico este trabalho a minha querida mãe Maria Aparecida, minhas irmãs Walkíria e Waleska, meus sobrinhos Gustavo e Julliano aos quais agradeço pelo apoio em todos os aspectos de minha vida.

Dedico também a minha adorável esposa Yolanda Daniela e a nossos maravilhosos filhos Brena, Fernanda, Ana Paula, Bárbara, Geovana, Jean, Isabela, Gabriel, Beatriz, Nícolas e, com especial alegria, a Lavínia e Rebeca que nasceram durante este mestrado, e os agradeço por cederem o tempo de minha presença com eles para a realização desse trabalho.

Agradeço a todos os colegas do Laboratório de Multimídia e Interatividade (LMI) e da equipe Black Bee, ambos da Universidade Federal de Itajubá (UNIFEI) e, em especial, aos alunos Rafael Gomes Braga, Caique de Sousa Freitas Duarte e Luciano do Vale Ribeiro, por toda a ajuda e apoio nos experimentos realizados.

Ao meu orientador, Prof Dr. Alexandre Carlos Brandão Ramos, agradeço pela amizade, confiança, atenção, revisões e sugestões desta dissertação, e por me direcionar às oportunidades de publicar artigos em congressos internacionais.

Agradeço ao Prof. Dr. Guilherme Sousa Bastos, coordenador deste programa de mestrado, pelo apoio pessoal em momentos críticos durante o curso e por ter apresentado em Singapura o primeiro artigo fruto do presente trabalho.

Aos membros da Banca agradeço pela disposição em analisar, avaliar e por enriquecer este trabalho com a experiência e inestimada opinião profissional de cada um.

Agradeço, ainda, à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo suporte financeiro concedido.

Finalmente, mas com o mesma gratidão e carinho, agradeço a todos aqueles que de alguma forma colaboraram para a realização deste trabalho, direta ou indiretamente, como os servidores públicos da UNIFEI, terceirizados e outros.

"Sábio é aquele que conhece os limites da própria ignorância."
(Sócrates)

Resumo

A navegação autônoma de robôs e a percepção de ambiente por parte de máquinas, têm motivado muitas pesquisas em diferentes áreas da ciência, graças ao desafio que representam e às possibilidades de suas aplicações em praticamente todas as áreas de interação humana, como indústria, comércio, agropecuária, vigilância e segurança, medicina, militar, serviços, trânsito, *smart city* e muitas outras. Atualmente, muitas aplicações para o controle de veículos autônomos apresentam bons resultados utilizando a visão computacional como parte integrante do conjunto de sensores (BERTOZZI; FASCIOLI, 2000), sempre com duas ou mais câmeras (visão estéreo e/ou combinada) e algoritmos para tratar as imagens obtidas. O crescente número de uso de Aeronaves Remotamente Pilotadas (ARPs) no Brasil e no mundo que, devido a suas reduzidas dimensões e necessidade de minimizar o consumo de baterias aumentando a autonomia de voo, exigem soluções compactas e leves, motivou a pesquisa apresentada nesta dissertação, que implementa uma solução embarcada baseada em visão de computador para que uma ARP consiga identificar obstáculos e áreas livres a sua frente, e se desviar pela área com maior espaço livre, utilizando uma única câmera (monovisão) RGB, sistema de cores (*Red, Green, Blue*) que se combinam em diferentes intensidades reproduzindo um largo espectro cromático, representando imagens no espectro visível (imagens que nossos olhos conseguem ver) e aplicando tratamentos matemáticos clássicos e elementares às mesmas. Foi desenvolvida nesta pesquisa uma solução embarcada que permite o voo autônomo controlado de uma ARP integrando *software*, como os programas *Robot Operating System* (ROS), simulador de robôs Gazebo, e *hardware*, como as controladoas Pixhawk e Raspberry Pi, que foram aplicadas a uma aeronave de asas rotativas do tipo quadrotor, popularmente conhecida como *"drone"*.

Palavras-chaves: ARP, Visão Computacional, Desvio de Obstáculos, Sistemas Embarcados.

Abstract

The autonomous navigation of robots and the perception of the environment by machines have motivated many researches in different areas of science, thanks to the challenge it represents and to the possibilities of its applications in practically all areas of human interaction, such as industry, commerce, security, medicine, the military, services, traffic, smart city, and many others. Nowadays, many of the applications for the control of autonomous vehicles that present good results, use computer vision as part of the set of sensors (BERTOZZI; FASCIOLI, 2000), always with two or more cameras (stereo and / or combined vision), and no-trivial algorithms to treat the images. The growing number of use of Remotely Piloted Aircraft (RPAs) in Brazil and in the world, which, due to its small size and the need to minimize battery consumption, increasing the autonomy of flight, requiring compact and light solutions, motivated the research presented in this dissertation which implements an embedded solution based on computational vision so that a RPA can identify obstacles and free areas ahead of it and deviate through the area with the greatest free space using a single RGB camera, color system, (Red, Green, Blue) that combine different intensities reproducing a wide chromatic spectrum, representing images in the visible spectrum (images that our eyes can see) and applying classic and elementary mathematical treatments to them. In this research, an embedded solution was developed that allows the controlled autonomous flight of an ARP integrating software, such as the Robot Operating System (ROS), Gazebo robot simulator, and hardware, such as the Pixhawk and Raspberry Pi controllers, which were applied to a rotary wing aircraft of the quadrotor type, popularly known as drone.

Key-words: RPA, Computer Vision, Obstacle Avoidance, Embedded Systems, OpenCV.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

2D	- De duas dimensões, bidimensional
3D	- De três dimensões, tridimensional
ANAC	- Agência Nacional de Aviação Civil
ARP	- Aeronave Remotamente Pilotada
ARPs	- Plural de ARP
BA	- Estado brasileiro da Bahia
BSD	- Berkeley Software Distribution
DECEA	- Departamento de Controle do Espaço Aéreo
DOF	- Degree of Freedom
DNG	- Digital Negative
DPI	- Divisão de Processamento de Imagens
DPIs	- Dots per Inch
ESC	- Electronic Speed Controllers
EUA	- Estados Unidos da América
FM	- Frequência Modulada
FPV	- First Person View
Full HD	- Full High Definition
GPS	- Global Positioning System
HDMI	- High-Definition Multimedia Interface
IMU	- Inertial Measurement Unit
INPE	- Instituto Nacional de Pesquisas Espaciais
JPEG	- Joint Photographic Experts Group
LIDAR	- Light Detection and Ranging

LWH - Length, Width and Height

MG - Estado brasileiro de Minas Gerais

OpenCV - Open Source Computer Vision Library

OSHW - Open Source Hardware

PC - Personal Computer

PDI - Processamento Digital de Imagens

PIXEL - Aglutinação de picture e elemen

PWM - Pulse Width Modulation

px - pixels

RAM - Random Access Memory

RGB - Red, Green and Blue

RJ - Estado brasileiro do Rio de Janeiro

ROS - Robot Operating System

RPA - Remotely-Piloted Aircraft

RPAs - Plural de RPA

RSSI - Received Signal Strength Indication

RTH - Return-to-Home

RTL - Return-to-Launch

SANT - Sistema Aéreo Não Tripulado

SONAR - Sound Navigation and Ranging

SP - Estado brasileiro de São Paulo

SVA - Sistema de Visão Artificial

UAV - Unmanned Air Vehicle

UAVs - Plural de UAV

UNIFEI - Universidade Federal de Itajubá

USB - Universal Serial Bus

UUV	- Unmanned Underwater Vehicles
UUVs	- Plural de UUV
Wi-Fi	- Wireless Fidelity
VANT	- Veículo Aéreo Não Tripulado
VANTs	- Plural de VANT

Lista de símbolos

A	- ampere
C	- taxa de descarga C-rate
cm	- centímetro
D	- distância entre duas cores
g	- grama
GB	- giga byte
GHz	- giga hertz
Hz	- hertz
kg	- quilograma
kV	- quilovolt
L	- grau de luminosidade
m	- metro
M	- mega
mm	- milímetro
Mp	- mega pixels
mAh	- miliampere-hora
t	- largura
%	- percentagem ou porcentagem
$*$	- multiplicação
$<-$	- atribuição

Sumário

1 INTRODUÇÃO

Esse capítulo apresenta a motivação e os objetivos dessa pesquisa, números oficiais de registros de ARPs no país, alguns pontos da legislação brasileira relacionados ao tema, e os assuntos tratados pelos demais capítulos desta dissertação. A **Seção 1.1** descreve as necessidades que motivaram o desenvolvimento desse trabalho. A regulamentação e os órgãos estatais brasileiros que regulam o uso de ARPs no território nacional, as classificações de ARPs segundo esses órgãos, e a quantidade de ARPs registradas no país, são apresentados na **Seção 1.2**. A **Seção 1.3** define o escopo da pesquisa, e seus objetivos são descritos na **Seção 1.4**.

1.1 Motivação

Em praticamente todas as aplicações que envolvem ARPs autônomas ou não, é desejável que as aeronaves consigam o máximo de autonomia relacionada a desvio de obstáculos durante o voo. A necessidade de uma solução compacta, leve e econômica motivou este trabalho que implementa uma solução usando o mínimo possível de *hardware*, *software* e visão computacional para identificar objetos e comandar o voo de uma ARP a se desviar dos mesmos.

1.2 Visão Geral

A Agência Nacional de Aviação Civil (ANAC) editou em maio de 2017 um regulamento especial com regras gerais para o uso civil de aeronaves não tripuladas no Brasil. As regras da ANAC são complementares às de outros órgãos, que também devem ser observadas antes de qualquer operação. Dentre eles, destacam-se as normas do Departamento de Controle do Espaço Aéreo (DECEA), do Ministério da Defesa e da Agência Nacional de Telecomunicações(ANATEL) (ANAC, 2018a). A legislação proíbe operações de aeronaves não tripuladas completamente autônomas (que não permitem qualquer tipo de intervenção de um piloto remoto). Partindo desta premissa, nesse trabalho é adotado o termo ARP para designar *drone*, VANT (Veículo Aéreo Não Tripulado), RPA (Remotely Piloted Aircraft), e UAV (*Unmanned Air Vehicles*) sendo que, segundo a ANAC,

> [...]o termo drone é usado popularmente para descrever qualquer aeronave (e até mesmo outros tipos de veículos) com alto grau de automatismo. De forma geral, toda aeronave drone é considerada uma aeronave não tripulada categorizada como Aeromodelo, ARP ou Aeronave Não Tripulada Autônoma.

Corroborando com esta definição, VANTs são pequenas aeronaves que, sem qualquer contato físico direto, possuem a capacidade de executar tarefas como monitorar, mapear, entre outras (MEDEIROS, 2017). De acordo com (MEDEIROS, 2017), essas aeronaves se caracterizam por dois aspectos básicos: não possuir piloto a bordo e carregar equipamentos, normalmente sensores, que lhes permitem cumprir determinadas missões. Essas ARPs são pilotadas ou controladas a distância através de meios eletrônicos e computacionais, supervisionados pelo homem. Atualmente mais de 40 (quarenta) países trabalham com o desenvolvimento de ARPs voltadas para diferentes mercados (JORGE, 2001).

A ANAC define ARPs como sendo "*... aeronaves não tripuladas utilizadas para fins experimentais, comerciais ou institucionais*", e as classifica:

> As ARPs estão divididas em três classes, de acordo com o peso máximo de decolagem, no qual deve ser considerado os pesos da bateria ou combustível do equipamento e de carga eventualmente transportada. A classificação é aplicável apenas para as ARPs e não para os aeromodelos.
>
> Classe 1 – Peso máximo de decolagem maior que 150kg
>
> Classe 2 – Peso máximo de decolagem maior que 25kg e até 150kg
>
> Classe 3 – Peso máximo de decolagem de até 25 kg

Ainda segundo a ANAC, "*..aeromodelos são as aeronaves não tripuladas remotamente pilotadas usadas para recreação e lazer*".

As ARPs de classe 3 estão se tornando bastante populares, principalmente entre pessoas físicas (Fig. 1.1).

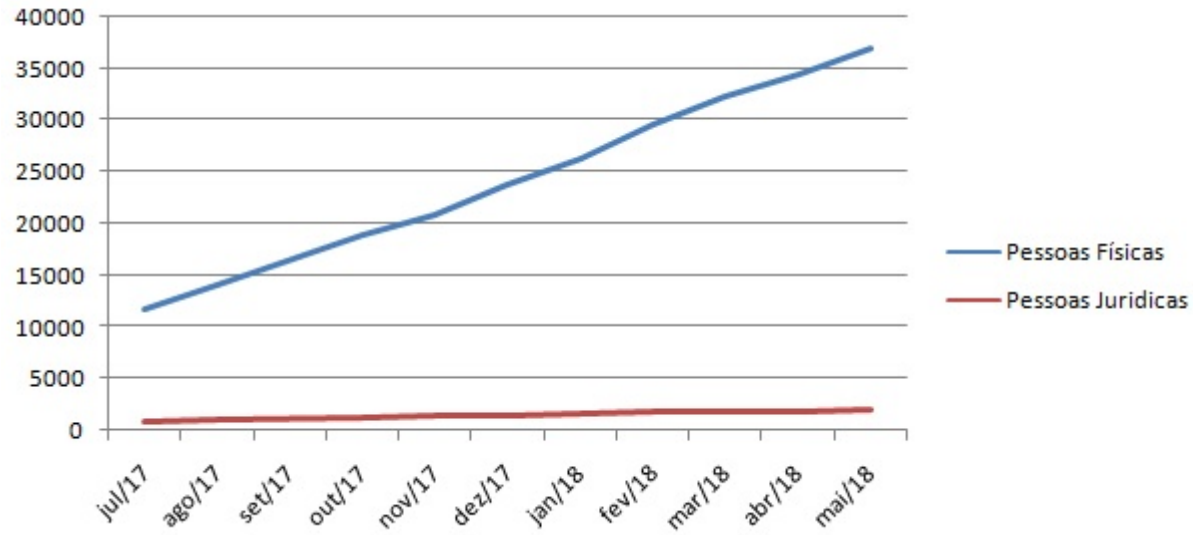

Figura 1.1 – Crescimento de Cadastros de Drones na ANAC - Pessoas Físicas x Jurídicas.
Fonte: Produção do autor com dados obtidos da Tabela 1.1.

No Brasil, mais de 41 mil aeronaves não tripuladas foram cadastradas na ANAC até abril de 2018 (Tabela 1.1) (ANAC, 2018a).

Tabela 1.1 – Quantidade de Cadastros de Drones na ANAC

Cadastro	07/17	08/17	10/17	11/17	12/17	01/18	02/18	03/18	04/18	05/18
Pessoas	12514	15090	20133	22204	25202	27862	31441	34115	36338	38988
Físicas	11693	14102	18884	20827	23659	26205	29624	32237	34375	36902
Jurídicas	821	988	1249	1377	1543	1657	1817	1878	1963	2086
Drones	13256	16567	22087	24295	27313	30087	33675	36331	38453	41338
Profissional	5375	6363	8557	9386	10443	11167	12175	13031	13809	14855
Recreativo	7881	10214	13530	14909	16870	18920	21500	23300	24644	26483
UF										
NI	464	559	763	831	919	1041	1037	1124	1200	1261
AC	13	15	18	23	30	37	39	44	45	46
AL	62	67	91	104	127	139	161	177	185	197
AM	90	124	176	203	236	274	310	337	358	386
AP	20	25	28	32	34	37	44	45	51	60
BA	433	491	676	750	856	951	1063	1178	1275	1406
CE	258	297	370	398	442	495	552	593	629	668
DF	457	548	712	798	923	1005	1139	1231	1306	1403
ES	237	282	365	398	447	480	544	602	645	690
GO	372	437	571	611	679	734	821	881	931	990
MA	119	153	206	224	238	255	277	295	313	338
MG	1297	1519	1993	2182	2453	2729	3032	3232	3391	3636
MS	181	220	284	313	365	402	446	473	503	528
MT	199	234	301	335	378	422	469	504	527	580
PA	143	189	259	285	313	337	361	393	414	447
PB	144	161	214	233	260	288	325	358	377	402
PE	292	339	449	506	572	625	721	792	842	907
PI	63	77	93	99	115	121	133	143	153	167
PR	770	986	1350	1496	1705	1894	2108	2253	2354	2546
RJ	1417	1891	2694	2930	3281	3618	4156	4537	4872	5280
RN	145	177	227	244	275	318	349	372	399	443
RO	57	80	108	119	143	155	166	188	196	217
RR	47	57	69	85	92	96	100	103	107	113
RS	675	825	1029	1147	1277	1395	1580	1701	1804	1933
SC	599	731	980	1091	1238	1354	1530	1642	1743	1898
SE	88	114	163	177	212	230	255	272	291	331
SP	4576	5926	7835	8607	8625	10566	11853	12743	13415	14325
TO	38	44	63	74	78	89	98	112	120	133

Não há dados do mês de setembro/2017. NI = Não informado.

Fonte: Produção do autor com dados divulgados pela ANAC (ANAC, 2018b)

.

Os estados de São Paulo, Rio de Janeiro e Minas Gerais, todos no sudeste do país, lideram, nesta ordem, o número de cadastros e o crescimento de registros de ARPs na Agência (Fig. 1.2).

Embora o crescimento de registros de ARPs na ANAC seja acentuado entre pessoas físicas, observa-se que o registro de aeronaves para uso profissional apresenta um crescimento superior ao do registro daquelas para uso recreativo (Fig. 1.3).

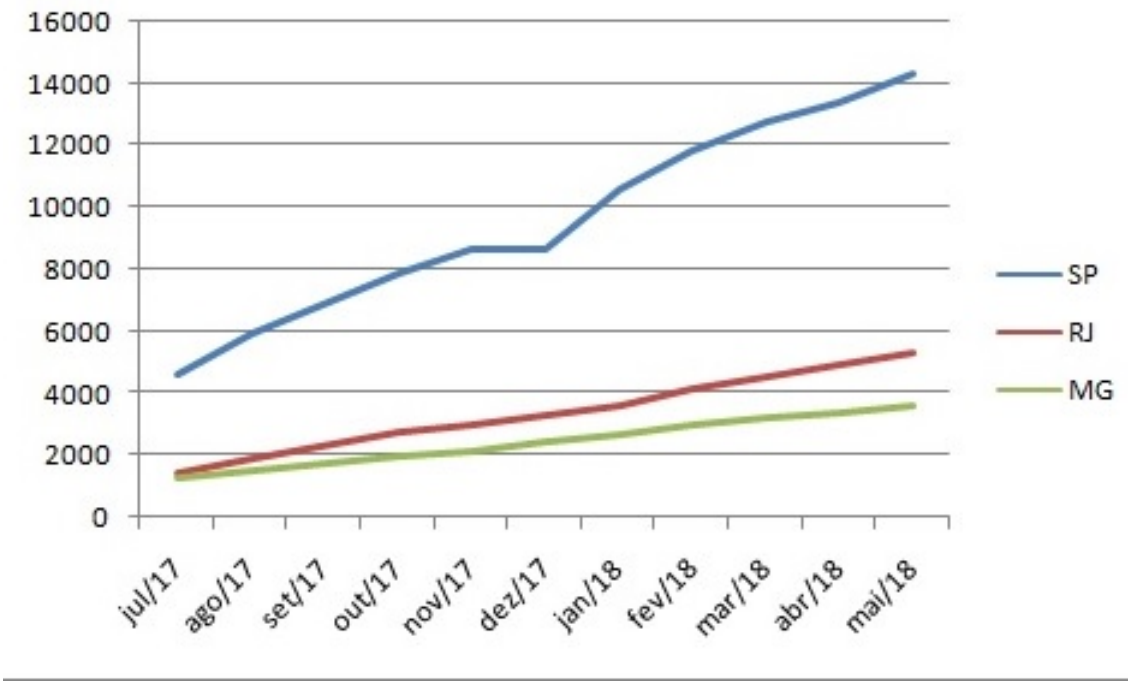

Figura 1.2 – Crescimento do Cadastro de Drones na ANAC - SP, RJ e MG.
Fonte: Produção do autor com dados obtidos da tabela 1.1.

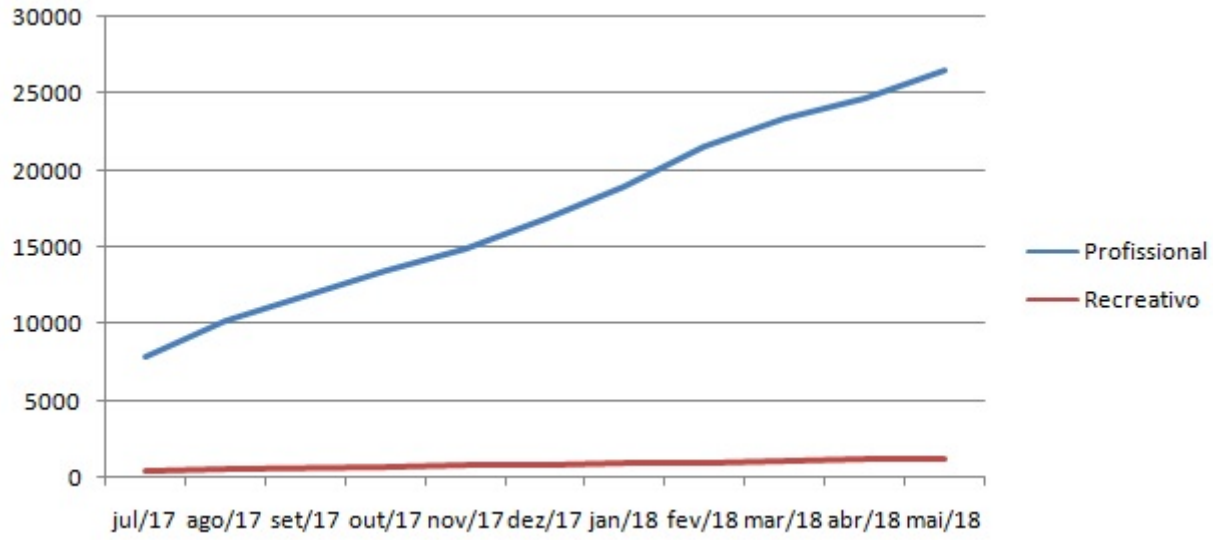

Figura 1.3 – Crescimento do Cadastro de Drones na ANAC Segundo o Tipo de Uso
Fonte: Produção do autor com dados obtidos da tabela 1.1

É possível que este crescimento justifique a quantidade atual de pesquisas voltadas para aplicações de ARPs, observado na Fig. 1.4.

Uma ARP com algum grau de automação deve perceber o ambiente a sua volta. Para isto, podem ser usados recursos como odometria, sonar, laser e visão computacional, entre outros.

Como exemplos de pesquisas em aplicações que utilizam visão computacional em ARPs, podemos citar:

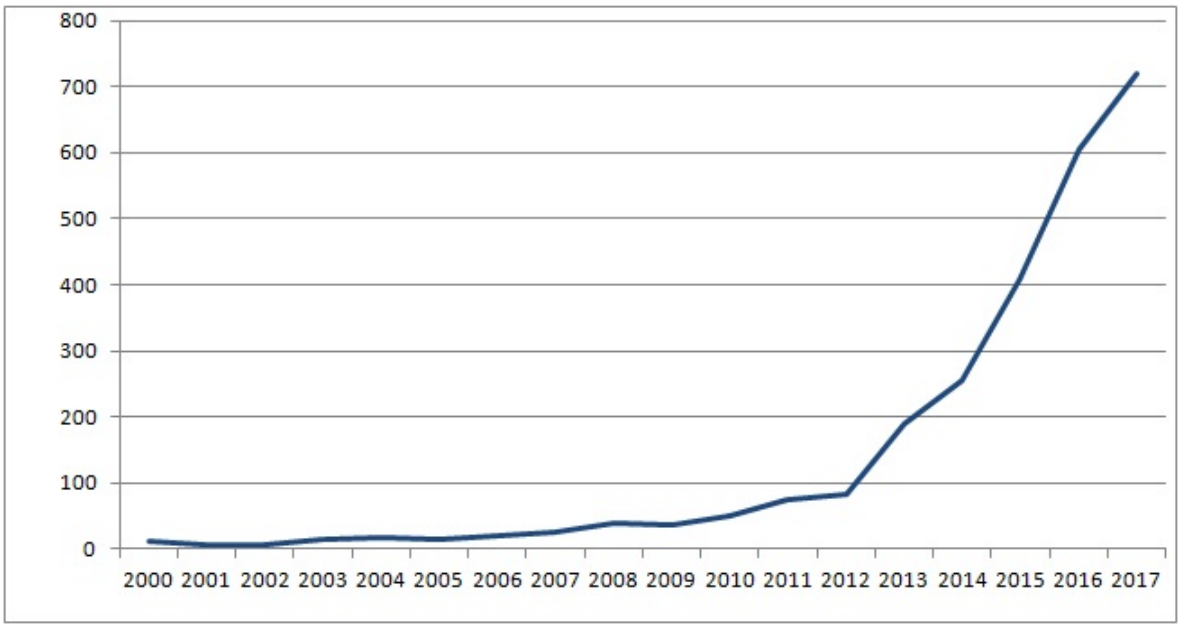

Figura 1.4 – Produção Nacional de Artigos sobre ARP
Fonte: O Autor com dados do Google Acadêmico.

- Inspeção de segurança em canteiro de obras (MELO, 2016);
- Monitoramento de grandes estruturas como pontes (METNI; HAMEL, 2006);
- Identificação de plantação de maconha (GLOBOG1, 2018);
- Obtenção de fotografias aéreas (FILHO; AGOSTINHO, 2011);
- Navegação Autônoma por Imagens (SILVA, 2016), (BRAGA, 2018) e (CONTE; DOHERTY, 2009);
- Monitoramento das condições ambientais do entorno das rodovias, controle de avalanches ao longo das mesmas, monitoramento de acidentes e resgate, mapeamento, inspeção de segurança e monitoramento de estruturas e pavimentos (PURI, 2005) e (KARAN et al., 2014).

Segundo (CRESTANI; FIGUEIREDO; VON-ZUBEN, 2002), há enorme interesse científico no estudo de sistemas de navegação em ambientes desconhecidos e sujeitos ao atendimento de múltiplos objetivos.

Soluções de baixo custo usando ARPs para esses tipos de problemas são muito interessantes para serem usados em diversos projetos globais de grande escala (AMENYO et al., 2014), tornando a tecnologia mais popular (RAHMAN et al., 2016).

Um sistema autônomo deve ser capaz de executar por si mesmo tarefas específicas e pré-definidas sem nenhum controle externo (FABRO, 1996), através de soluções que controlem seu comportamento (CRESTANI; FIGUEIREDO; VON-ZUBEN, 2002), fazendo com que se comporte conforme planejado ou desejado.

1.3 Escopo da Pesquisa

Essa dissertação de mestrado apresenta uma solução para o problema de navegação autônoma embarcada que se opõe a outras soluções que exigem processamento complexo citadas por (CRESTANI; FIGUEIREDO; VON-ZUBEN, 2002), uma vez que ARPs de pequeno porte precisam embarcar soluções que demandem *hardware* de peso mínimo possível e processar qualquer algoritmo em tempo hábil para permitir movimentos em voo a tempo de identificar e se desviar dos obstáculos em seu caminho evitando colisões. Esta necessidade de uso de soluções simples que empreguem o mínimo de *hardware* aumenta à medida que se produzem ARPs cada vez menores.

Há muitas maneiras de fazer uma ARP perceber obstáculos em seu caminho e evitá-los, como o uso de câmeras e sensores (GARCIA et al., 2016) como sonares e GPS, além do emprego de estratégias como rotas previamente conhecidas e marcadores de posição (*checkpoints* e *landmarks*), dentre outras.

Nesse trabalho é proposto e desenvolvido um sistema autônomo de navegação embarcado em uma ARP do tipo quadrotor fundamentado exclusivamente no tratamento de imagens em tempo real que identifica obstáculos durante o voo e controla os movimentos da aeronave desviando-a pela área com maior espaço livre.

1.4 Objetivos

Essa seção descreve os objetivos gerais e específicos da presente pesquisa.

1.4.1 Objetivos Gerais

Esta pesquisa de mestrado tem como objetivo geral estudar e implementar um método capaz de perceber obstáculos e áreas livres a frente de uma ARP em movimento, em ambientes simulado e real, sem utilizar GPS ou qualquer outro sensor ativo, empregando apenas uma câmera embarcada e técnicas de visão por computador.

Essa percepção é baseada na captura de imagens aéreas pela aeronave durante o voo, no processamento dessas imagens em tempo real, na identificação de obstáculos e áreas livres à frente da ARP, na tomada de decisão quanto a melhor opção de desvio e no comando dos movimentos da aeronave de forma a conduzi-la pelo caminho definido, evitando obstáculos.

1.4.2 Objetivos Específicos

Os objetivos específicos desta pesquisa podem ser enumerados como:

(a) Produzir a simulação de uma ARP autônoma.

(b) A ARP deve ter uma câmera embarcada e capturar imagens a sua frente durante seu voo.

(c) A ARP deve ser capaz de receber comandos de pilotagem em tempo real de um algoritmo embarcado.

(d) A pesquisa deverá produzir um algoritmo que processe as imagens capturadas aplicando os conceitos de visão por computador para percepção de objetos e obstáculos, de forma a identificar, em cada imagem, obstáculos e áreas livres à frente da ARP, em tempo real durante seu voo.

(e) O algoritmo deverá ser capaz de comandar o voo da ARP de forma a se desviar de obstáculos através da área de maior espaço livre a sua frente.

(f) A solução deve incluir a integração com o software ROS, por ser um ambiente desenvolvido para o controle de robôs e possuir ampla biblioteca de funções específicas para esta aplicação.

(g) O algoritmo desenvolvido, aplicado, testado e aperfeiçoado em ambiente simulado deve ser posteriormente embarcado em uma ARP em voos reais e controlados.

(i) A eficiência da solução deve ser avaliada tanto em ambiente simulado quanto em ambiente real, e os resultados devem ser publicados.

1.5 Organização do Trabalho

O restante desta dissertação de mestrado está dividido em 6 outros capítulos e um apêndice organizados como segue:

O **Capítulo 2** apresenta outros trabalhos de pesquisa relacionados com os temas visão por computador e desvio autônomo de obstáculos.

No **Capítulo 3** são descritos os materiais usados e os métodos aplicados nesta pesquisa.

No **Capítulo 4** é apresentada a estratégia de identificação e desvio de obstáculos usando visão computacional.

O **Capítulo 5** descreve os experimentos realizados em ambiente simulado e aqueles em ambiente real.

O **Capítulo 6** apresenta uma análise dos resultados obtidos pelos experimentos realizados.

As conclusões e propostas de trabalhos futuros são descritas no **Capítulo 7**.

Ao final dos capítulos, o **Apêndice A** descreve os componentes de uma ARP do tipo quadrotor.

2 TRABALHOS E APLICAÇÕES RELACIONADOS

2.1 Introdução

A presente pesquisa lida com dois temas distintos: visão computacional, para a identificação de objetos, e o desvio desses objetos propriamente dito. Nesse trabalho esses dois temas foram combinados para solucionar o problema de desvio autônomo de obstáculos por uma ARP.

Ambos os temas possuem vasta quantidade de aplicações e pesquisas relacionadas(Figs. 2.1 e 2.2). Diversas pesquisas para solucionar o problema de desvio de obstáculos, utilizando ou não visão computacional, vêm sendo realizadas no mundo todo. Inúmeras técnicas como emprego de sensores e conhecimento prévio da região ou ambiente têm sido exploradas e apresentadas à comunidade científica, fato observável pela quantidade crescente de produção de artigos relacionados (Fig. 2.3).

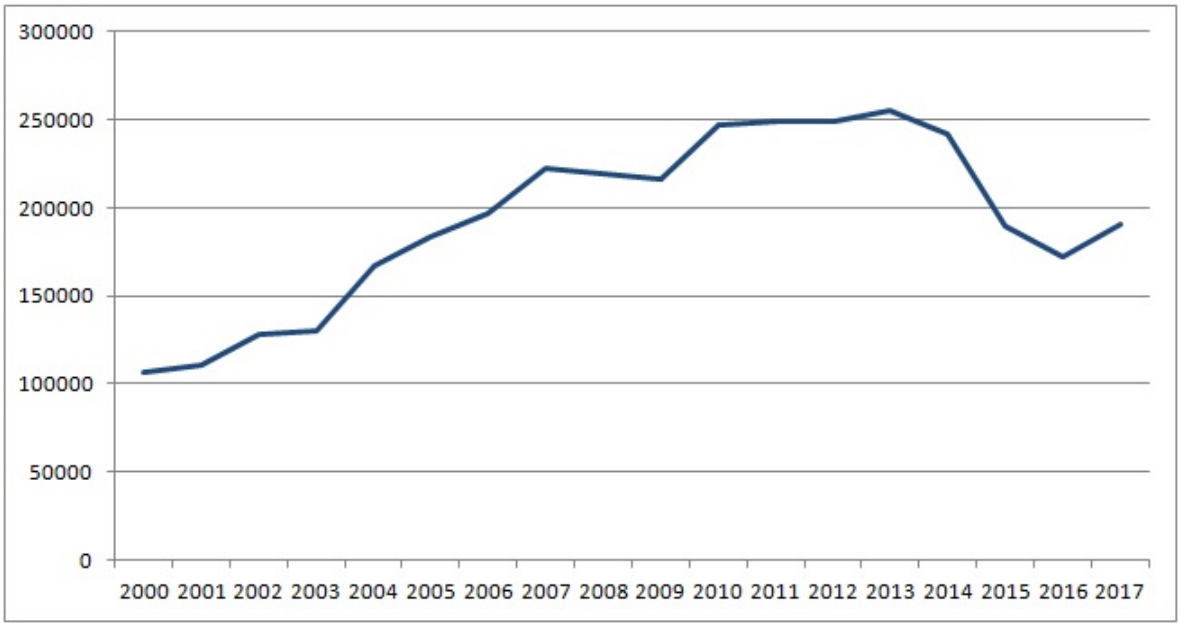

Figura 2.1 – Produção Mundial de Artigos sobre Visão Computacional
Fonte: O Autor com dados do Google Acadêmico.

Nesse capítulo são mencionados alguns trabalhos que empregam visão computacional, e/ou sensores, para a percepção do ambiente ou tratam de desvio de obstáculos em tempo real, e outros que combinam os dois assuntos.

A Seção 2.2 relaciona alguns trabalhos sobre o tema enquanto a Seção 2.3 apresenta algumas aplicações.

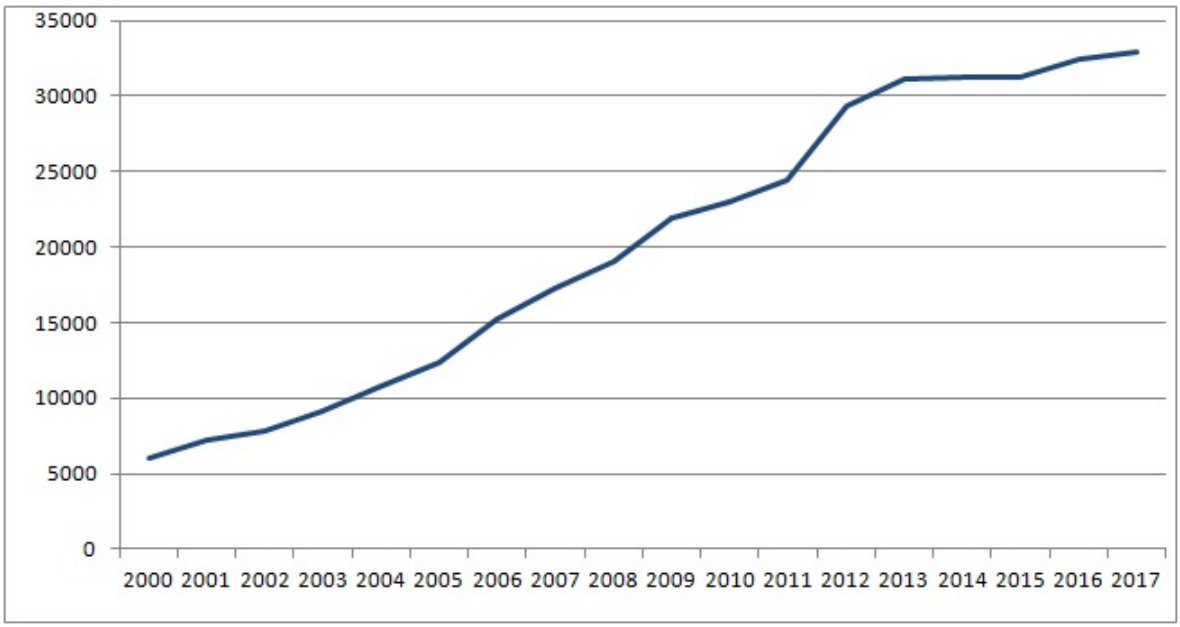

Figura 2.2 – Produção Mundial de Artigos sobre Desvio de Obstáculos
Fonte: O Autor com dados do Google Acadêmico.

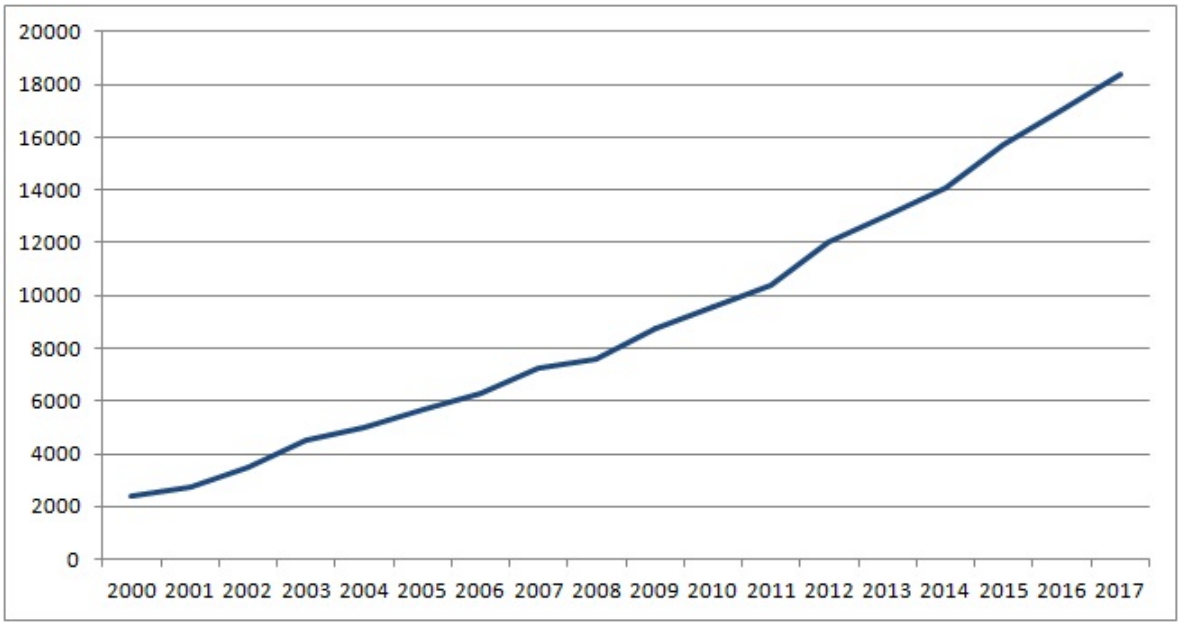

Figura 2.3 – Produção Mundial de Artigos sobre ARP
Fonte: O Autor com dados do Google Acadêmico.

2.2 Trabalhos Relacionados

Um sistema de navegação para ARPs em ambiente interno (*indoor*), usando visão por computador, foi proposto por (ANDRESEN; JONES; CROWLEY, 2001). O sistema necessitava do conhecimento antecipado do ambiente sobrevoado. Esse conhecimento era fornecido por meio de imagens adquiridas e armazenadas previamente para serem então processadas durante o voo. Nesse tipo de solução as imagens capturadas pela ARP são comparadas com as do banco de dados de imagens armazenadas. O algoritmo procura

identificar pontos em comum (âncoras) nas imagens que possam indicar a localização da aeronave, atuando como referências físicas espaciais (SHIGUEMORI; MARTINS; MONTEIRO, 2007).

Uma solução em tempo real usando visão computacional estéreo (emprego de duas câmeras) foi desenvolvida por (NISTER; NARODITSKY; BERGEN, 2006). A solução se aplicava à navegação autônoma de um robô terrestre, localizando e fazendo o robô se desviar de objetos em seu caminho Essa solução usava um algoritmo para detectar os contornos das imagens e identificar os obstáculos e opções de desvio desses obstáculos no caminho do robô.

Outra solução de desvio autônomo de obstáculos é proposta por (SAUNDERS; BEARD; MCLAIN, 2007), usando vários caminhos circulares, obtidos com o emprego de laser que, combinados, definiam a presença ou ausência de obstáculos no ambiente e mensuravam a distância entre o robô e os obstáculos identificados.

Uma solução utilizando linguagem de programação *Python* é proposta por (HERMíNIO et al., 2010). Um pequeno protótipo móvel percorria um percurso pré-definido desviando-se de obstáculos usando uma câmera de vídeo de um celular embarcado. As imagens eram transmitidas em tempo real para um computador, via rede *wi-fi*, que as processava e enviava comandos de movimento para um microcontrolador embarcado no protótipo.

O uso de diversos sensores é a solução proposta por (WAGSTER et al., 2012) para fazer com que uma ARP se desvie de obstáculos voando a uma velocidade constante pelo ambiente. A combinação de múltiplos sensores aumenta a eficiência nas definições de pose (posicionamento da aeronave) e identificação e localização dos objetos.

2.3 Aplicações Relacionadas

O uso de ARP para escanear fachadas de prédios, por meio de uma câmera digital de alta resolução, é apresentado por (ESCHMANN et al., 2012). Nesta solução, em um primeiro momento é feita a aquisição de imagens 2D (de duas dimensões) pela ARP em voos horizontal e vertical. Depois, as imagens coletadas são unidas e procura-se por sinais de rachaduras. Um trabalho similar, para detectar fissuras, usando imagens 3D (de três dimensões) foi apresentado por (TOROK; GOLVARPAR-FARD; KOCHERSBERGER, 2014).

Uma solução para desvio de obstáculos por uma ARP usando grafos foi proposta por (WEISS et al., 2014). A conectividade entre os nós indicava a presença ou ausência de obstáculos.

Uma solução que usa GPS embarcado e um dispositivo laser é criada por (STUB-

BLEBINE et al., 2015) para fazer uma ARP se desviar de obstáculos em tempo real.

Segundo (NETO, 2016), um dos setores da economia brasileira que pode tirar grande proveito da tecnologia de ARPs com visão computacional capazes de executar algumas tarefas programadas é a agricultura comercial, pois com uso das aeronaves é possível realizar mapeamento do solo cultivado, verificar quais indivíduos (plantas) precisam de mais nutrientes e ainda verificar a saúde dos mesmos. Em sua solução, (NETO, 2017) usou monovisão para orientar um robô terrestre no desvio de obstáculos.

O desvio de obstáculos em tempo real por ARPs encontra aplicações na inspeção de áreas urbanas, como propõem (MOTA et al., 2014) e (FELIZARDO; RAMOS; CAMINO, 2015), incluindo a inspeção de linhas de transmissão. Uma análise sobre a eficácia do emprego de mini-helicópteros para esta finalidade foi apresentada por (HRABAR; MERZ; FROUSHEGER, 2010).

Na área oceânica, o desvio autônomo de obstáculos também vem sendo explorado, como em trabalhos para controle de trajetória de veículos submarinos não tripulados (*Underactuated Unmanned Underwater Vehicles - UUVs*) (YAN et al., 2015) incluindo múltiplos veículos (CAI; ZHANG; ZHENG, 2017), podendo ser aplicados em inspeções de cabos e dutos de petróleo submersos (YAN et al., 2018).

A percepção e o desvio autônomo de obstáculos também têm aplicações em enxames (*swarm*) de ARPs, onde várias aeronaves voam juntas de forma cooperativa na realização de tarefas, como propõe (BRAGA et al., 2018), em uma abordagem em que as aeronaves constantemente se atraem (para manter a formação) e se retraem (para evitar colisões) por meio de cálculos resultantes de vetores de forças físicas virtuais ponderadas, sem perder o foco da tarefa a ser realizada em conjunto.

Por sua vez, esse trabalho apresenta uma solução embarcada de desvio de obstáculos para ARPs do tipo quadrotor, que usa uma câmera para aquisição de imagens durante o voo, processa essas imagens em tempo real dentro da própria aeronave, aplica funções da biblioteca gráfica OpenCV (do inglês *Open Source Computer Vision Library*), para identificar objetos e áreas livres de obstáculos na cena do voo por meio de técnicas de visão por computador, e comanda de forma autônoma o desvio da aeronave destes obstáculos pela área de maior espaço livre, integrando a solução ao ROS.

3 MATERIAIS E MÉTODOS

3.1 Introdução

Nesse capítulo são apresentados os materiais e métodos empregados no desenvolvimento da solução proposta no presente trabalho. A Seção 3.2 apresenta os conceitos de visão por computador, sistema de visão artificial (SVA) e a forma como foram abordados nesta pesquisa. Na Seção 3.3 é apresentada a biblioteca de tratamento de imagens OpenCV e suas funções empregadas na solução proposta. A Seção 3.4 descreve a simulação por computador, o simulador Gazebo e o framework ROS, também utilizados no desenvolvimento da solução.

3.2 Visão por Computador

Para (MASCARENHAS; VELASCO, 1989), Sistema de Visão Artificial (SVA) é definido como "*...um sistema computadorizado capaz de adquirir, processar e interpretar imagens correspondentes a cenas reais*". A estrutura de um SVA é proposta por (MARQUES; VIEIRA, 1999) (Fig. 3.1).

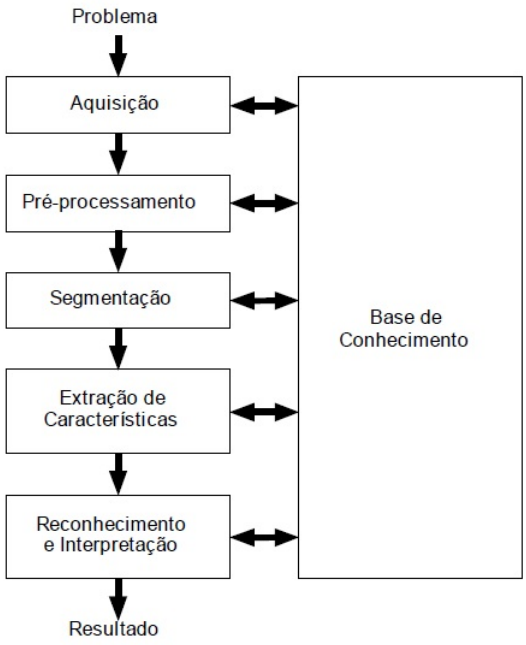

Figura 3.1 – Um Sistema de Visão Artificial (SVA) e suas principais etapas.
Fonte: (MARQUES, VIEIRA, 1999), *Processamento Digital de Imagens*, p.9.

Seguindo a estrutura proposta por (MARQUES, VIEIRA, 1999), o SVA dessa

dissertação pode ser definido como:

Problema: Ambiente desconhecido à frente de uma ARP em voo.

Aquisição: Emprego de uma câmera embarcada a frente da ARP para adquirir imagens durante seu voo.

Pré-processamento: Aplicar operações matemáticas relacionadas a visão por computador sobre as imagens adquiridas, que facilitem a identificação de obstáculos e de áreas livres a frente da aeronave.

Segmentação: Definir as regiões de interesse na imagem, que levem a identificação de obstáculos e áreas livres de obstáculos.

Extração de Características: Obter informações das regiões segmentadas na busca pela definição de obstáculos e áreas livres a frente da aeronave.

Reconhecimento e Interpretação: Identificar a presença ou ausência de obstáculos e áreas livres a frente da aeronave.

Resultado: Obstáculos e áreas livres à frente da aeronave, identificados.

Base de Conhecimento: Nesse trabalho não foram usados marcadores ou imagens conhecidas antecipadamente. A base de conhecimento nesta pesquisa pode ser entendida como a tecnologia e a estratégia que foram empregadas para que cada fase funcionasse perfeitamente, assim como a divisão da imagem resultante em três áreas de interesse a serem comparadas em busca daquela que apresente a menor quantidade de obstáculos.

Neste ponto, o tema visão por computador se completa e o tema desvio de obstáculos se inicia.

A partir dos resultados obtidos pelo SVA, a solução desenvolvida nessa pesquisa irá tomar a decisão quanto ao melhor caminho a ser percorrido pela ARP no desvio dos obstáculos identificados a sua frente.

Uma vez decidido o caminho de desvio, a solução irá gerar os comandos necessários aos componentes eletrônicos da ARP de forma a comandar os componentes elétrico-mecânicos que movimentam a ARP conduzindo-a em conformidade com o trajeto desejado.

A cada novo avanço, mínimo que seja, o SVA é acionado, capturando e processando novas imagens, gerando novos resultados, decisões e movimentos de desvio, num processo que se repete ininterruptamente, corrigindo a trajetória de forma autônoma durante o voo, passando por um cenário com vários obstáculos, até que um operador humano interrompa o avanço da aeronave.

O sistema ROS é executado em um computador do tipo notebook na estação de controle em solo.

O SVA, o algoritmo que interpreta os resultados produzidos pelo SVA, o algoritmo de tomada de decisões de desvio e o algoritmo que define os comandos de desvio da ARP são executados por um sistema Raspberry Pi embarcado na aeronave. Todos estes recursos também podem ser processados no notebook da estação de controle, como nos experimentos em ambiente real realizados na ARP do tipo Bebop2 que não permite que um sistema Raspberry Pi seja embarcado.

Os comandos de voo gerados no sistema Raspberry Pi são gravados em um nó do ROS que simula o rádio controle TX. o ROS envia estes comandos de rádio por *wi-fi* para a controladora Pixhawk embarcada que comanda os motores da ARP no sentido de obedecer os comandos de voo produzidos pelo algoritmo embarcado. Estes passos ocorrem exatamente sempre da mesma forma, não importando se a ARP possui ou não o sistema Rasbperry Pi embarcado.

3.2.1 Visão Computacional em Robôs Autônomos

A visão computacional tem como meta *"utilizar computadores para emular a visão humana, incluindo o aprendizado e a capacidade de fazer inferências e agir com base em informações visuais.* (GONZALEZ; WOODS, 2008).

Ainda segundo (GONZALEZ; WOODS, 2008), *"não existe um acordo geral entre os autores em relação ao ponto em que o processamento de imagens termina e outras áreas relacionadas, como a análise de imagem e a visão computacional, começam"*.

A visão computacional permite que máquinas consigam aperceber-se de algumas informações físicas do ambiente em que estão inseridas, processando imagens capturadas por câmeras de vídeo. Estas informações permitem reconhecer, manipular e pensar sobre objetos que compõem uma imagem (MILANO; HONORATO, 2010).

A navegação baseada em computador aplicada a robôs móveis autônomos tem sido objeto de mais de três décadas de pesquisa e vem sendo intensamente estudada (WAXMAN; LEMOIGNE; SCINVASAN, 1987) e (THROPE et al., 1988). Para as ARPs, o processamento de imagens em tempo real é fundamental para a tomada de decisão durante o voo e pode ser feito com uma câmera simples (DAVISON, 2003).

A forma como se organiza uma solução de visão computacional depende da aplicação, e se existe ou não alguma parte de aprendizagem durante a operação. Segundo (MILANO; HONORATO, 2010) existem funções que são comuns a toda solução de visão computacional, a saber, aquisição de imagem, pré-processamento, extração de características, detecção e segmentação e processamento de alto nível.

A Figura 3.2 ilustra o processo desde a aquisição de uma imagem real até a sua representação digital.

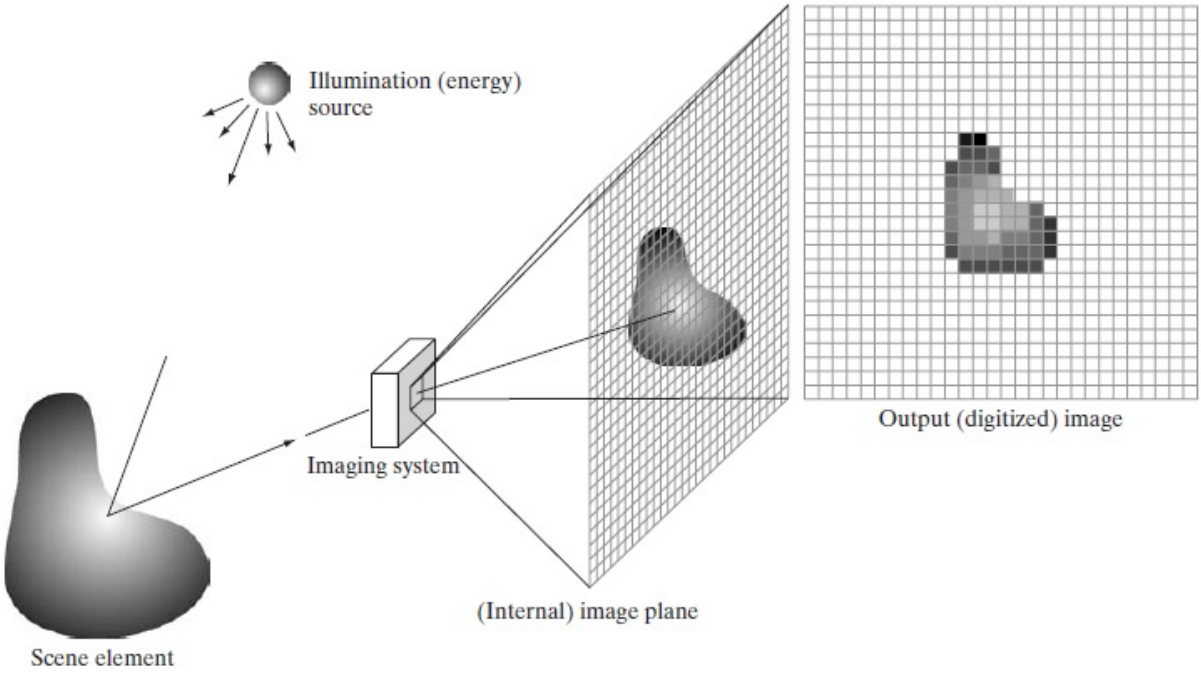

Figura 3.2 – Exemplo do Processo de Aquisição de Imagens Digitais.
Fonte: (GONZALEZ, WOODS, 2008) *Digital Image Processing*, p.51

Não existe uma teoria geral de aprimoramento de imagem. Quando uma imagem é processada para interpretação visual, o espectador é o juiz final de quão bem um método particular funciona (GONZALEZ; WOODS, 2008). No presente trabalho foram realizadas tentativas de aplicações de diversos métodos e ajustes dos mesmos até ser encontrado um que atendesse aos objetivos previamente estabelecidos e descritos no parágrafo anterior.

3.2.2 Imagem

Segundo (MARQUES; VIEIRA, 1999),

> *"...uma imagem monocromática pode ser descrita matematicamente por uma função f(x,y) da intensidade luminosa, sendo seu valor, em qualquer ponto de coordenadas espaciais (x,y), proporcional ao brilho (ou nível de cinza) da imagem naquele ponto."*

Para (GONZALEZ; WOODS, 2008),

> *"...uma imagem pode ser definida como uma função bidimensional, f(x,y) em que x e y são coordenadas espaciais (plano), e a amplitude de f em qualquer par de coordenadas (x,y) é chamado de intensidade ou nível de cinza da imagem nesse ponto. Quando x,y e os valores de intensidade de f são quantidades finitas e discretas, chamamos de imagem digital. (...) Observe que uma imagem digital é composta de um número finito de elementos, cada um com localização e valor específicos. Esses valores são chamados de elementos pictóricos, elementos de imagem, pels e pixels. Pixel é o termo mais utilizado para representar os elementos de uma imagem digital."*

Portanto, é possível definir uma imagem bidimensional, como é o caso das fotos capturadas pela ARP na solução desenvolvida nesta pesquisa, como o conjunto de pixels que a compõem, expressos em uma função f(x,y), onde cada pixel possui um endereço cartesiano (x,y) com o valor de um atributo f qualquer para aquele pixel, como por exemplo, o nível de luminosidade ou de cinza em uma escala de cinzas do pixel naquele ponto. E pixel como sendo a menor unidade de medida de uma imagem digital, contendo características como cor, brilho, intensidade e posição na imagem.

3.2.3 Aquisição de Imagem

A aquisição ou captura de uma imagem consiste em representar numericamente uma imagem do mundo real obtida por meio de algum dispositivo físico que converta energia no espectro eletromagnético em sinal elétrico analógico proporcional à quantidade de energia percebida. A conversão desse sinal em conjuntos de dígitos 0s e 1s (bits), compondo uma imagem digital é chamada de quantização, e pode pode ser bidimensional, uma cena tridimensional ou ainda uma sequência de imagens (na forma de um vídeo ou não). Cada pixel acessado na imagem vai indicar o atributo intensidade da luz em uma ou várias faixas de cor (que forma imagens em tons de cinza ou coloridas) na função de atributos f(x,y).

3.2.4 Quantização

Quantização é a transformação de uma imagem com tons contínuos em uma outra imagem com tons discretos, permitindo o armazenamento ou transmissão digital, com cada tom representado por um código binário (MASCARENHAS; VELASCO, 1989). *"A digitalização dos valores de amplitude é chamada de quantização"* (GONZALEZ; WOODS, 2008). É preciso definir a faixa, amplitude ou quantidade de valores possíveis de serem atribuídos à função f(x,y) que representa a imagem digital. Neste trabalho, a faixa de valores para o atributo intensidade de cinza foi definida entre 0 e 255.

3.2.5 Pré-processamento

Uma imagem adquirida (capturada) precisa ser preparada de forma a permitir a extração de informações de maior relevância para a conveniência da aplicação desejada, reduzindo ruídos (informações de má qualidade), aumentando o contraste, tornando mais fácil e preciso detectar o tipo de informação desejada, suavizando a imagem pela redução da amplitude de informações no domínio f(x,y). Uma imagem pré-processada para atender a uma aplicação poderá não ter nenhuma utilidade se usada para atender outras aplicações com objetivos distintos, sendo este um dos inúmeros desafios da visão computacional.

3.2.6 Processamento de Alto Nível

Identificar a relevância das informações para o que se pretende, interpretando os objetos detectados em diferentes categorias, como em obstáculos e áreas livres.

Para (MILANO; HONORATO, 2010), o problema da visão computacional é determinar se uma imagem contém ou não um dado objeto ou característica de interesse, aplicando diferentes estratégias e técnicas, não havendo uma aplicável a todas as aplicações possíveis, envolvendo objetos aleatórios em situações aleatórias. Na melhor das hipóteses, cada solução é útil para objetos e cenários específicos, como poliedros, faces humanas, letras escritas à mão ou veículos, placas de veículos, espécies de animais e de plantas, sendo necessário considerar outras informações, como a iluminação o fundo fixo e a pose (sentido, direção e posição) dos objetos em estudo.

3.2.7 Extração de Características

Extrair características é, em primeira análise, aperceber-se, em uma imagem, daqueles detalhes relevantes a determinado propósito. A técnica consiste em agrupar os pixels (x,y) da imagem em elementos mais complexos, como bordas ou cantos, através das relações dos valores de f(x,y) entre os pixels vizinhos.

3.2.8 Detecção e Segmentação

Segmentação de uma imagem significa separar os diversos componentes (objetos) existentes, reunindo os pixels pertencentes a um dado objeto ou região, através de um critério de semelhança, como por exemplo, sua intensidade de luminosidade, como adotado neste trabalho.

3.2.9 Identificação de Objetos

Uma ocorrência de um objeto previamente conhecido pode ser identificada em imagens, como a face de uma pessoa ou sua impressão digital. Neste trabalho o interesse está em identificar a presença ou ausência de obstáculos (quaisquer objetos) na imagem a frente da ARP.

3.2.10 Vizinhança de Pixel e Convolução

Convolução significa aplicar uma máscara para transformar uma imagem, pixel a pixel, determinando características, como bordas, extraindo ruídos (como reflexos) e identificando formas. A convolução atua em cada pixel com base nos atributos de seus pixels vizinhos, modificando o valor original do pixel na imagem resultante. Isto suaviza e aproxima os valores dos atributos de pixels vizinhos.

A vizinhança de um pixel pode se dar de duas formas:

Vizinhança de 4 pixels: onde se consideram apenas os pixels mais próximos na vertical e horizontal;

Vizinhança de 8 pixels: Nesse caso são consideradas as diagonais além dos pixels horizontais e verticais conforme apresentado na Fig. 3.3.

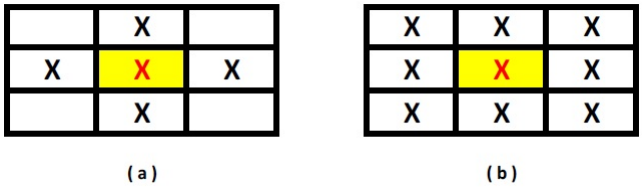

Figura 3.3 – Vizinhanças de 4 (a) e de 8 (b) Pixels em uma Imagem Digital.
Fonte: Produção do autor.

A máscara resultante é chamada de filtro. No processo de detecção de bordas, a vizinhança de 8 pixels é empregada. Existem diversos filtros para detecção, como os de Robert, Laplace, além dos de Sobel e Canny (ANTONELLO, 2010), que foram usados neste trabalho.

Os filtros transformam a imagem pixel a pixel, e combinam o nível ou valor de escala de cinza de um determinado pixel com o de seus vizinhos (DPI.INPE, 2010). Neste trabalho foi empregada a técnica de detecção de bordas para identificar os obstáculos nas imagens a frente da linha de voo da ARP. Os algoritmos mais comuns para detecção de bordas são os operadores Sobel, Canny e as variações de ambos (ANTONELLO, 2010), em que a detecção de bordas se faz através da identificação do "gradiente", que são variações abruptas na intensidade dos valores dos pixels de uma região da imagem.

3.2.11 Detector de Bordas Sobel

O algoritmo de detecção de bordas Sobel possui formulação matemática mais simples que a do Canny. Emprega o cálculo do gradiente de um pixel em relação aos seus vizinhos. Quando este valor de gradiente for acima de um limiar então é detectada a presença de borda (GONZALEZ; WOODS, 2008).

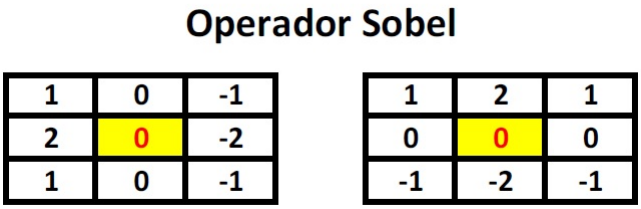

Figura 3.4 – Filtro Sobel.
Fonte: Produção do autor.

Neste trabalho foi aplicada sobre a imagem convertida a tons de cinza a convolução das máscaras apresentadas na Fig. 3.4, que ilustra a aplicação do operador Sobel, que usa uma matriz (Mx) para detectar bordas verticais e outra matriz (My) para detectar bordas horizontais.

Como resultado da aplicação das duas máscaras, são obtidas duas novas matrizes de intensidade (Ix e Iy). O módulo da intensidade do gradiente resultante (I) é obtido através da raiz quadrática da soma dos quadrados de Ix e Iy.

$$|I| = \sqrt{I_x^2 + I_y^2} \tag{3.1}$$

Este módulo de intensidade do gradiente resultante (I) é mais suave, menos rico ou com menor amplitude entre os valores do atributo intensidade de cor do pixel, que as informações originais, gerando uma imagem com valores f(x,y) para luminosidade mais homogênea, facilitando a identificação de regiões delimitadoras, como é o caso de bordas de objetos.

3.2.12 Detector de Bordas Canny

Desenvolvido por John F. Canny (CANNY, 1986), o detector de bordas Canny, ou *Canny Edge Detector*, ou ainda "perador Canny", aplica múltiplas vezes outras técnicas, como o filtro Sobel, para identificar linhas de bordas contidas na imagem, que podem estar delineando objetos ou obstáculos. O processo consiste em aplicar um filtro gaussiano para suavizar a imagem e remover o ruído (*blur*), encontrar os gradientes de intensidade da imagem e determinar bordas potenciais ao aplicar Sobel duplo, verificar se o pixel faz parte de uma borda "forte" suprimindo todas as outras bordas que são fracas e não conectadas à bordas fortes. Uma borda forte é aquela em que existe uma forte relação de valores entre os pixels vizinhos, ou seja, os valores da função f(x,y) dos pixels estão bastante próximos uns dos outros e acima de um determinado limiar.

Este algoritmo está implementado na biblioteca OpenCV e foi usado neste trabalho.

3.3 A Biblioteca Gráfica OpenCV

A OpenCV (*Open Source Computer Vision Library*) (OPENCV.ORG, 2018) é uma biblioteca de funções computacionais para tratamento de imagens, visão computacional e reconhecimento de padrões.

A biblioteca armazena imagens em tipos de dados comuns como matrizes e vetores. Para manipular suas informações, lida com processamento de imagens, realiza transfor-

mações geométricas, filtragens, tratamentos de cor, e outras. Fornece diversas funções para manipular vídeos, como avaliação de movimento, análise de fluxo ótico e rastreio de objetos. Permite a calibração de câmeras, extração de características, detecção de objetos, *Highgui* (módulo de controle de interface e dispositivos de entrada), para tratamento facilitado de criação de interfaces gráficas, entre diversos outros recursos.

Sua licença é na modalidade *Berkeley Software Distribution (BSD)*, uma licença de código aberto que impõe poucas restrições quando comparada àquelas impostas por outras licenças como a *General Public License (GNU)* ou a *copyright*, se aproximando do modelo "domínio público".

3.3.1 Função Blur - Suavização da Imagem

Blur, smoothing, blurring, desfoque ou suavização de uma imagem é "borrar" ou "embaçar", digitalmente a imagem, como em uma foto "fora de foco", alterando a cor de cada pixel, misturando sua cor com as cores dos pixels a seu redor (seus vizinhos) pelo processo de convolução. Há várias maneiras de obter o novo valor de cada pixel, como suavização pelo valor da média simples, pela gaussiana, pela mediana e por filtro bilateral. Neste trabalho o valor do pixel foi substituído pelo valor da média simples dos valores dos seus vizinhos. Alguns autores chamam esta vizinhança de "janela de cálculo", núcleo ou *kernel* (núcleo, em inglês) (ANTONELLO, 2010).

Este procedimento aumenta a eficiência em algoritmos de identificação de objetos e de detecção de bordas, ao diminuir a riqueza de detalhes da imagem, tornando-a mais simples em termos de estímulos visuais, ou seja, diminuindo a quantidade de valores possíveis nas informações contidas na imagem a ser processada, uma vez que aproxima os valores de todos os pixels.

Em outras palavras, o desfoque da imagem é obtido pela convolução da imagem com um *kernel* de filtro de baixa passagem. É útil para remover ruídos, no caso, removendo conteúdo de alta frequência como bordas da imagem, que é "borrada" (ou suavizada) nesta operação.

A biblioteca OpenCV fornece quatro tipos de técnicas de suavização, uma para cada maneira de obter o valor do pixel no centro do núcleo: (a) pela média dos valores dos pixels vizinhos (*Averaging*), (b) desfocagem gaussiana (*Gaussian-Blurring*), (c) desfocagem mediana (*Median Blurring*) e (d) filtragem bilateral *Bilateral-Filtering*. Nesse trabalho foi usada a função *Blur* que utiliza a técnica *Averaging* por ser a que emprega a formulação matemática mais simples.

3.3.2 Função FindContourn - Definição de Contornos

Para identificar obstáculos a frente da ARP, é preciso definir os contornos de todos os objetos contidos em cada imagem capturada durante o voo. Digitalmente, contorno é uma lista de pixels representando uma curva em uma imagem. Na OpenCV os contornos são representados por sequências em que cada entrada na mesma codifica a informação sobre a localização do próximo ponto na curva. Neste trabalho, após a suavização da imagem pela função *Blur*, foi usada a função *FindContourn* para identificar os contornos de possíveis obstáculos.

3.3.3 Função Operador Sobel - Filtro de Gradiente

Atualmente a biblioteca OpenCV disponibiliza a implementação de três filtros de gradiente (*high-pass filters*): Sobel, Scharr e Laplacian. As respectivas funções são: *Sobel*, *Scharr* e *Laplacian*. Nesse trabalho foi aplicada a função *Sobel*, mais uma vez buscando produzir resultados com as soluções mais elementares possiveis.

3.3.4 Função Operador Canny - Detector de Bordas

Desenvolvido por John F. Canny (CANNY, 1986), o detector de bordas Canny, ou *Canny Edge Detector*, ou ainda "operador Canny", aplica múltiplas vezes outras técnicas, como o filtro Sobel, para identificar linhas de bordas contidas na imagem, que podem estar delineando objetos ou obstáculos. O processo consiste em aplicar um filtro gaussiano para suavizar a imagem e remover o ruído (*blur*), encontrar os gradientes de intensidade da imagem e determinar bordas potenciais ao aplicar Sobel duplo, verificar se o pixel faz parte de uma borda “forte” suprimindo todas as outras bordas que são fracas e não conectadas à bordas fortes. Uma borda forte é aquela em que existe uma forte relação de valores entre os pixels vizinhos, ou seja, os valores da função f(x,y) dos pixels estão bastante próximos uns dos outros e acima de um determinado limiar.

3.4 Simulação por Computador

Para provar ou testar um conceito, ou para projetar uma solução ou sistema de controle, de medição ou atuação, é possível reproduzir seu comportamento e eficiência em um ambiente virtual simulado em computador, usando modelagem matemática para predizer as condições esperadas no mundo real onde o sistema irá operar. Isto evita acidentes e custos com testes para o aperfeiçoamento da solução. Estudar e avaliar comportamentos complexos em robôs reais pode demandar muito tempo com testes caros e complexos, dificultando ou inviabilizando a implementação diretamente no mundo real. Entretanto, uma simulação nunca substitui a observação em ambiente real que possui características

extremamente difíceis de serem modeladas computacionalmente com exatidão e completude (CAZANGI, 2004). Nesse trabalho foram simulados cenários com obstáculos e com a aplicação de um algoritmo desenvolvido, que comandou o desvio de uma ARP por entre os obstáculos em um ambiente virtual, antes de aplicar a solução em um cenário real.

3.4.1 Simulador Gazebo

Gazebo é um *software* gratuito e, segundo o site do fabricante, um simulador com a capacidade de simular de forma precisa e eficiente populações de robôs em complexos ambientes internos e externos. Possui um robusto "motor" de física, gráficos de alta qualidade e convenientes interfaces gráficas programáveis. A simulação de robôs é uma ferramenta essencial para todo desenvolvedor de soluções aplicáveis em robótica. Um simulador bem projetado permite realizar testes rápidos, projetar robôs, realizar testes de regressão e treinar sistemas de inteligência artificial usando cenários realistas.

3.4.2 Framework ROS

O algoritmo desenvolvido nessa pesquisa foi implementado em linguagem de programação C++ e executado na plataforma ROS, uma coleção de *frameworks* de código aberto criada para ajudar no desenvolvimento de aplicações robotizadas. O ROS fornece ferramentas e instalações que auxiliaram no aprimoramento da solução desenvolvida ao permitir ajustar a velocidade, altura e ângulos de desvio através da observação do comportamento da ARP simulada antes da solução ser levada para um voo real.

O ROS é uma rede de funções computacionais representada em uma arquitetura de grafos (Fig. 3.5). Contém "nós" que se comunicam entre si usando um modelo de troca de mensagens por meio de publicação e assinatura (consumo) em locais chamados "tópicos". Sensores de controle, de estado, planejamento e atuador, são exemplos de nós que publicam (escrevem) em locais (tópicos) e formas (tipos) específicos, mensagens que poderão ser lidas (consumidas) por qualquer nó inscrito nestes locais. Assim, o *driver* (*software* que traduz o informações do *hardware* ou de um dispositivo para que um programa de computador possa processar) de um determinado sensor pode ser implementado como um nó, que publicará dados do sensor em um fluxo de mensagens em um tópico nomeado. Essas mensagens podem ser consumidas por quaisquer outros nós que se inscrevam neste tópico, incluindo filtros, registradores e sistemas de nível superior, como orientação, localização de caminho e qualquer aplicação. Cada nó é um programa independente no sistema. E existe ainda um nó mestre (*Master*) para gerenciar toda a rede.

Na Fig. 3.5, ROS-MASTER é o nó principal que controla todos os demais nós de uma solução implementada em ROS e, está embarcada no robô (que pode ser uma ARP) juntamente com os nós *Camera-Node* e *Image-Processing-Node*, sendo executados,

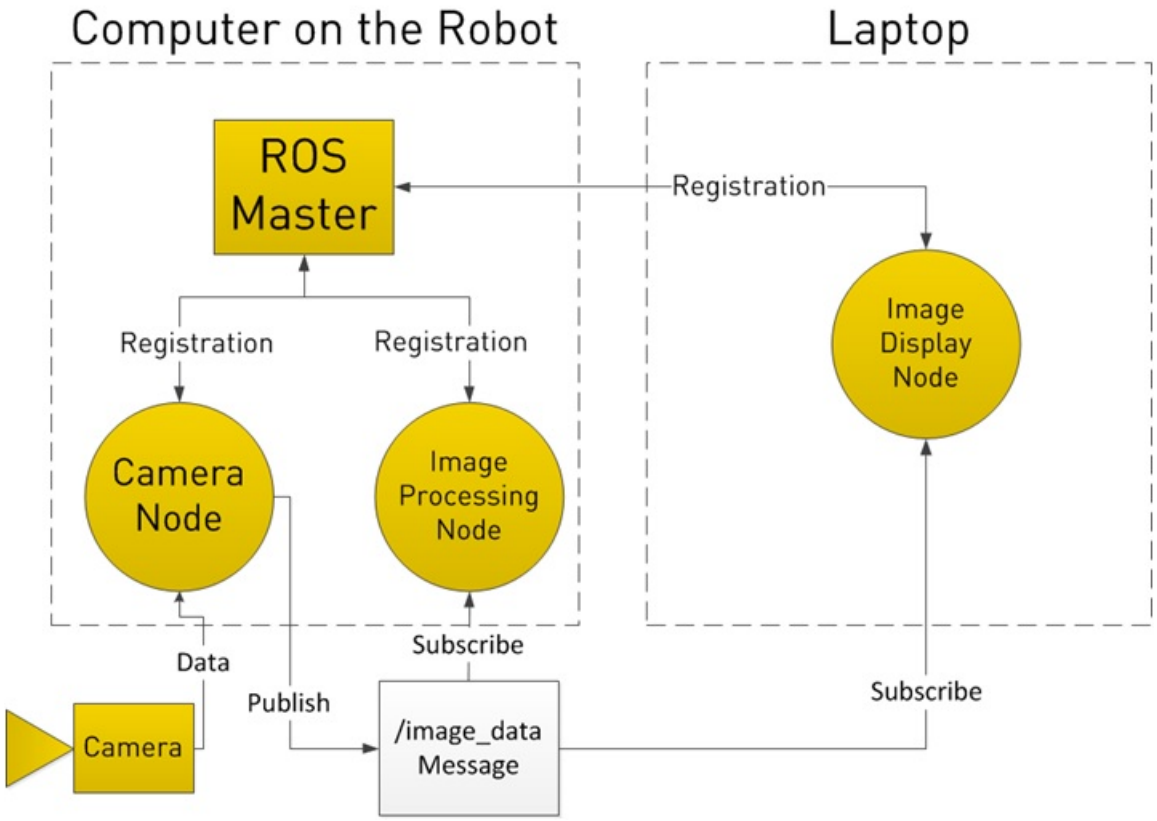

Figura 3.5 – Estrutura de Nos do ROS.
Fonte: (ROS.ORG, 2015)

por exemplo, em uma placa Raspberry PI. O nó *Image-Display-Node* está executando em um laptop externo. A comunicação entre nós embarcados e externos pode se dar via rede *wi-fi*. Os dados da câmera são publicados (enviados e armazenados) em *Camera-Node* que as publica no tópico */image_data_Message*. O nó *Image-Processing-Node* está subscrito neste tópico e, portanto, tem acesso às imagens ali publicadas pelo nó *Camera-Node*. Este nó manipula as imagens conforme a aplicação determinar. Outro nó, o *Image-Display-Node* também está subscrito no tópico */image_data_Message*, tendo, portanto, acesso às imagens ali armazenadas. Este nó tem a função de exibir em tela as imagens para um observador humano. O ROS e os nós *Camera-Node* e *Image-Processing-Node*, podem ser implementados externamente em um computador ou laptop juntamente com o nó *Image-Display-Node*. Nesta pesquisa estes dois cenários foram explorados e implementados.

Na solução desenvolvida, foi criado um nó chamado *image_subscriber*, que se inscreve no tópico em que as mensagens de imagem da câmera estão sendo publicadas. O nó criado é responsável por processar as imagens e decidir como e quando a ARP deve se mover. Nos experimentos em ambientes reais, as mensagens de imagem vem de um nó conectado diretamente ao *hardware* da câmera, enquanto nos experimentos em ambientes simulados, foi usado o simulador Gazebo, publicando os dados da câmera simulada.

Outro nó, chamado Mavros, é responsável pela comunicação com a placa contro-

ladora Pixhawk, descrita na seção A.9.3.

mavros/OverrideRCIn Message

File: mavros/OverrideRCIn.msg

Raw Message Definition

```
# Override RC Input
# Currently MAVLink defines override for 8 channel

uint16 CHAN_RELEASE=0
uint16 CHAN_NOCHANGE=65535

uint16[8] channels
```

Compact Message Definition

```
uint16 CHAN_RELEASE=0
uint16 CHAN_NOCHANGE=65535
uint16[8] channels
```

autogenerated on Wed, 26 Aug 2015 12:29:18

Figura 3.6 – Mensagem do Tipo mavros/msgs/OverrideRCIn.
Fonte: (ROS.ORG, 2015)

Quando o nó image_subscriber decide para onde mover a ARP, ele cria uma mensagem de tipo mavros/msgs/OverrideRCIn (Fig. 3.6) que representa um comando de um controlador de rádio, incluindo seus canais, preenche a mensagem com os valores correspondentes para causar o movimento desejado e a publica em um tópico em que o nó Mavros está inscrito.

O Nó Mavros cria uma mensagem do tipo MAVLink que contém essa informação e a envia para a controladora Pixhawk (**seção A.9.3**) através de uma conexão do tipo "serial"

Figura 3.7 – Diagrama da solução no ROS.
Fonte: Produção do autor.

A Fig. 3.7 mostra um diagrama gerado por `rqt_/graph`, uma ferramenta ROS que mostra os nós e os tópicos relevantes.

A Fig. 3.8 ilustra a solução desenvolvida.

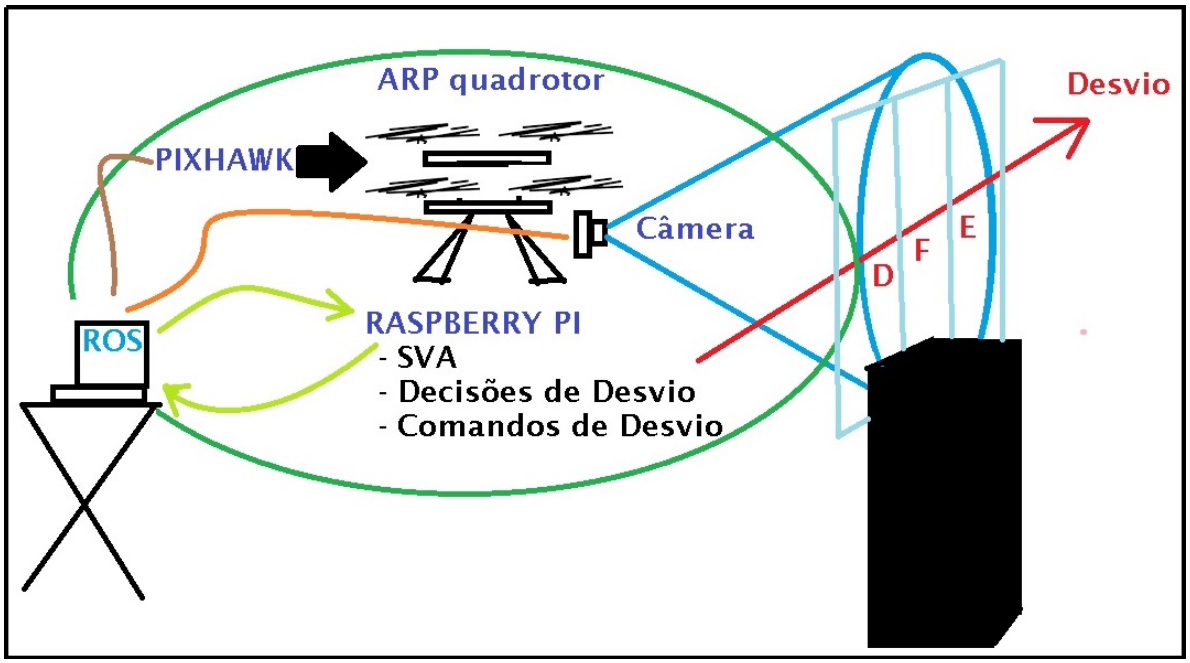

Figura 3.8 – Ilustração da Solução Desenvolvida.
Fonte: Produção do autor.

(1) O algoritmo embarcado na Raspberry Pi comanda a captura das imagens pela câmera, também embarcada.

(2) As imagens são gravadas em um tópico do ROS.

(3) Essas imagens são lidas pelo algoritmo desenvolvido que aplica SVA, que as manipula aplicando os tratamentos matemáticos necessários e gerando o resultado programado.

(4) Baseado no resultado fornecido pelo SVA, o algoritmo decide a forma de desvio, se para o lado D (direito), F (à frente) ou lado E (esquerdo), e avança na direção comandada.

(5) Essa forma de desvio é enviada na forma de mensagem de formato específico para um tópico previamente criado para este objetivo, no formato de comandos de rádio-controle TX.

(6) O ROS envia os comandos de rádio-controle para a ARP (simulada ou não) via *wi-fi*.

(7) A ARP recebe estas mensagens através de seu receptor (RX) e os converte em comandos de aceleração de motores, pela Pixhawk.

(8) A ARP se movimenta conforme comandado.

(9) O movimento muda a ARP de posição espacial e todo o processo dos items (1) a (9) se repetem, corrigindo e reposicionando a aeronave numa trajetória de desvio de obstáculos identificados.

4 IDENTIFICAÇÃO E DESVIO DE OBJETOS

4.1 Introdução

Nesse trabalho foi usada apenas a visão do computador, através do tratamento de imagens coletadas em tempo real, a partir de uma câmera embarcada. A vantagem desta solução é poder ser implementada usando poucos recursos de software e *hardware*. A solução desenvolvida foi eficiente nos testes executados do voo simulado e mais limitada nos testes reais.

Foi desenvolvido um algoritmo, posteriormente codificado em linguagem C++, para adquirir imagens, processá-las, reconhecer obstáculos e áreas livres e comandar o desvio da ARP para a área de maior espaço livre, se houver.

Para os experimentos simulados, o código foi integrado com ROS e Gazebo como ilustra a Fig. 4.1.

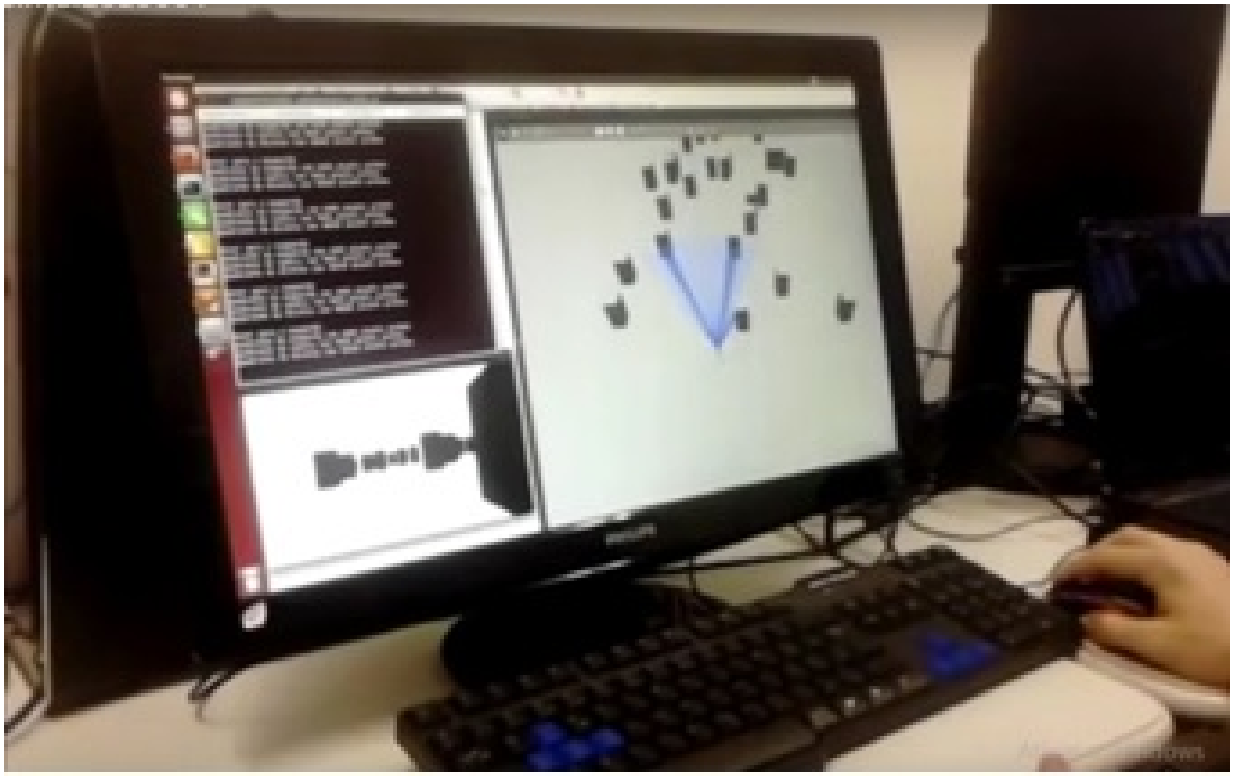

Figura 4.1 – Simulação com Múltiplos Obstáculos Usando ROS e Gazebo.
Fonte: Produção do autor.

A plataforma considerada é uma ARP do tipo quadrotor que é semelhante a um helicóptero, mas tem quatro rotores. A plataforma quadrotor é atraente devido à sua

simplicidade mecânica, agilidade e dinâmica bem compreendida (SREENATH; LEE; KUMAR, 2013).

A técnica usada neste trabalho consiste na aplicação das etapas descritas nas Seções 4.3 a 4.10.

4.2 Imagens ROS x OpenCV

O ROS trabalha com um formato de imagem diferente do formato usado pela biblioteca OpenCV. Enquanto ROS publica e acessa imagens em formato próprio, em uma mensagem to tipo `sensor_msgs_/Image`, o OpenCV usa o formato de matriz cv::Mat para lidar com imagens. O ROS possui uma biblioteca nativa, chamada *CvBridge*, que converte imagens de um formato para outro, e que foi usada neste trabalho para converter imagens entre os formatos do ROS e da OpenCV (Fig. 4.2).

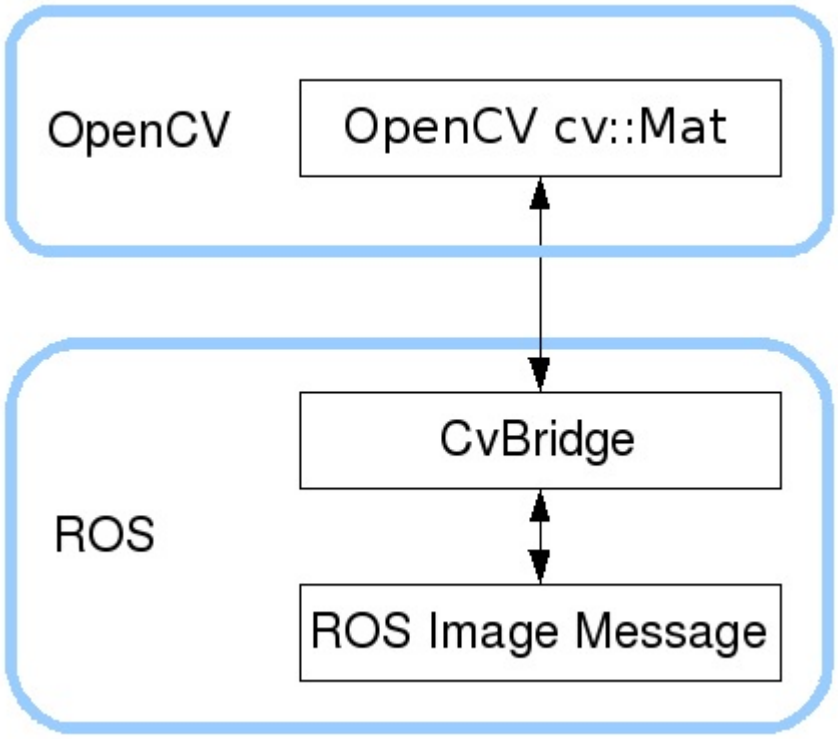

Figura 4.2 – Conversão de imagens OpenCV para formato ROS.
Fonte: ROS.org.

CvBridge define um tipo registro chamado CvImage, cujos atributos são uma imagem OpenCV, a codificação desta imagem e um cabeçalho no padrão ROS. Por CvImage conter a mesma informação que `sensor_msgs_/Image`, é possível converter de uma representação para outra (ROS.ORG, 2017).

4.3 Captura de Imagens

A ARP utilizada nos experimentos tem uma câmera frontal embutida que é ativada pelo código em linguagem C++ desenvolvido para capturar uma imagem do ambiente a sua frente. Essa imagem original sem qualquer tipo de tratamento é chamada nesse trabalho de imagem primitiva.

Figura 4.3 – Imagem Primitiva.
Fonte: Produção do autor.

A figura 4.3 mostra um exemplo de imagem primitiva (RGB) (vermelho, verde, azul) não tratada com uma resolução de 360x640px (pixels).

4.4 Aplicar Escala de Cinza à Imagem Capturada

O primeiro tratamento matemático que a imagem primitiva recebe é ter seu número de cores reduzido a uma faixa mais estreita, limitada a tons ou graus ou escala de cinza. Esta técnica trata a distância (D) entre cores RGB, dada pela equação (4.1). Quanto menor a distância entre duas cores, mais similares são (BUENO, 2011). No formato de cores RGB, as cores de dois pixels quaisquer (1 e 2) podem ser definidas respectivamente como $Cor_1(R_1, G_1, B_1)$ *e* $Cor_2(R_2, G_2, B_2)$.

$$D = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2} \tag{4.1}$$

O grau de luminosidade (L) de cada pixel também é considerado, o que indica o quanto o olho humano pode perceber sua cor, e pode ser definido pela equação (4.2).

$$L = R * 0,2126 + G * 0,7152 + B * 0,0722 \tag{4.2}$$

A Fig. 4.4 mostra o pseudo-código da conversão:

```
Para cada Pixel da imagem faça
inicio
  | Calcular sua luminosidade (L);
  | Pintar o pixel com o brilho encontrado;
fim
```

Figura 4.4 – Algoritmo para Gerar Imagem em Escala de Cinza por Brilho.
Fonte: Produção do autor.

Ainda é possível converter qualquer cor em seu nível de cinza aproximado ao obter as suas primitivas vermelha, verde e azul (da escala RGB) e adicionando 30% a mais de vermelho, 59% a mais de verde e 11% a mais de azul, independentemente da escala usada (0,0 a 1,0; 0 a 255; 0% a 100%.)

```
Para cada Pixel da imagem faça
inicio
  | Pixel_RGB[0] <- 1.30 * Pixel_RGB[0]; // Vermelho
  | Pixel_RGB[1] <- 1.59 * Pixel_RGB[1]; // Verde
  | Pixel_RGB[2] <- 1.11 * Pixel_RGB[2]; // Azul
fim
```

Figura 4.5 – Algoritmo para Gerar Imagem em Escala de Cinza por Ajuste.
Fonte: Produção do autor.

O nível resultante é o cinza de valor desejado. Tais porcentagens estão relacionadas à sensibilidade visual real do olho humano convencional para as cores primárias (LIMA, 2010). A Fig. 4.5 mostra o pseudo-código desta aplicação.

O código desenvolvido usou a função "cvtColor" da biblioteca gráfica de código aberto OpenCV para converter a foto primitiva (Fig. 4.3) em uma foto em escala de cinza (Fig. 4.6).

Figura 4.6 – Imagem em Escala de Cinza.
Fonte: Produção do autor.

Uma imagem com a função f(x,y) para cores, empobrecida (com menos informações) de colorida para tons de cinza, facilita a identificação de objetos e limites como bordas.

4.5 Determinar o Limiar

O segundo e último tratamento que a imagem captada pela câmera recebe é ter cada um de seus pixels ativado (255) ou desativado (0). Define-se então, um limiar para dispositivos monocromáticos, podendo ser, por exemplo, a luminosidade média de pixel da imagem primitiva (BUENO, 2011). Nesta pesquisa, por tentativa e erro, foi arbitrado o valor 127 para o limiar. Este valor pode variar de um ambiente para outro ou de uma aplicação para outra e deve ser ajustado em conformidade com o objetivo desejado, sendo, portanto, mais um dos inúmeros desafios da visão por computador. Pixels com valor da função f(x,y) iguais ou acima deste limiar terão seus valores igualados a 255 (ligados), enquanto os demais pixels da imagem, cujos valores sejam inferiores ao limiar, terão seus valores igualados a zero (desligados). Esta convenção pode ser aplicada ao inverso ou com com o "inclusive" sendo aplicado para desligar e não para ligar os pixels, sem que isso mude o resultado final.

4.6 Binarizar a Imagem

É atribuída a cada uma das 3 cores RGB de cada pixel da imagem em escala de cinza, os valores 0 (totalmente preto) ou 255 (totalmente branco). se o valor de cor RGB for, respectivamente, inferior ou igual ao limiar ou superior ao limiar.

Isso equivale a dizer que são pintados de preto ("apagados") os pixels de cor cinza mais escuro, e de branco ("acesos") os pixels de cor cinza mais claro. Após este tratamento, é obtida a imagem "binarizada" da Fig. 4.7 que possui apenas pixels nas cores preto (0) ou branco (255).

Figura 4.7 – Imagem Binarizada.
Fonte: Produção do autor.

A Fig. 4.8 mostra o pseudo-código para a binarização da imagem em escalas de cinza.

```
Para cada Pixel da imagem faça
inicio
  | Se {Luminosity(L) >= Limiar}
  |     Ligar o Pixel: aplicar cores RGB(255, 255, 255)
  | Senão
  |     Desligar o Pixel: aplicar cores RGB(0, 0, 0)
fim
```

Figura 4.8 – Algoritmo para Gerar Imagem Binarizada.
Fonte: Produção do autor.

4.7 Segmentar a Imagem

Uma vez obtida a imagem binarizada, esta é dividida verticalmente em três áreas de mesmo tamanho (altura e largura) chamadas de esquerda, centro e direita.

Nessa pesquisa, em que foi adotada uma segmentação da imagem em três seções de interesse, a largura (t) de cada área é um terço da largura total (*width*) da imagem binarizada (eq. 4.3).

$$t = div(width/3) \tag{4.3}$$

No caso de mais divisões, a proporcionalidade deverá ser mantida.

A Fig. 4.9 ilustra esta divisão.

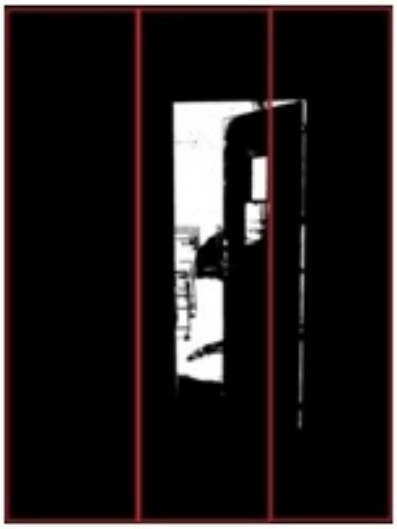

Figura 4.9 – Imagem Binarizada Segmentada em 3 Áreas de Interesse.
Fonte: Produção do autor.

Assim são definidas as três áreas de interesse.

Área(0): 360 x (000 a 213) pixels (lado esquerdo da imagem).

Área(1): 360 x (214 a 426) pixels (centro da imagem).

Área(2): 360 x (427 a 640) pixels (lado direito da imagem).

4.8 Contagem de Pixels

Definidas as áreas, são contadas as quantidades de pontos “desligados” (negros) em cada área, representados por A(n), onde n é a n-ésima área e 0 <= n <= 2.

A Fig. 4.10 mostra o pseudo-código desta contagem.

```
Para cada Área (n de 0 a 2) da imagem faça
inicio
 |    Contar a quantidade de pixels OFF A(n)
fim
```

Figura 4.10 – Algoritmo para Contar o Número de Pixels Off.
Fonte: Produção do autor

4.9 Determinar o Caminho de Desvio

O código interpreta que a maneira mais apropriada para a ARP avançar é se deslocando para a área A(n) com menor quantidade de pontos negros, uma vez que o código associa os pixels pretos a obstáculos e os pixels brancos a áreas livres com alguma possibilidade de permitir a passagem da ARP.

A Tabela 4.1 mostra a regra de decisão de desvio de obstáculos que o código desenvolvido emprega.

Tabela 4.1 – Tabela de Decisões de Desvio de Obstáculos

Se	E	Ação
A(0) > A(1)	A(0) > A(2)	Desviar a Esquerda
A(1) > A(0)	A(1) > A(2)	Avançar a Frente
A(2) > A(0)	A(2) > A(1)	Desviar a Direita

Fonte: Produção do autor

Caso duas ou mais áreas tenham a mesma quantidade de pixels negros, ou RGB (0,0,0), o código aplica a Tabela 4.2 de decisão.

Tabela 4.2 – Tabela de Decisões para Áreas Livres Concorrentes

Se	Ação
A(n) = A(1)	Avançar a Frente
A(0) = A(1)	Desviar a Direita

Fonte: Produção do autor

Desvios para cima e para baixo de obstáculos não foram implementados por fugirem do escopo desta pesquisa.

4.10 Comandar o Movimento da ARP

A decisão de mudança (para a direita, frente ou esquerda) é traduzida em comandos de voo que são transmitidos para a ARP que os executa.

Uma vez passados os comandos de voo para a ARP, descartar a imagem processada e substituir por uma nova imagem (*frame*) é capturada no tempo t+1, agora com o novo posicionamento da ARP, resultante do desvio sugerido e comandado.

As etapas definidas nas seções 4.3 a 4.10 são repetidas indefinidamente até que se deseje pousar a ARP.

5 EXPERIMENTOS

5.1 Introdução

O desenvolvimento do algoritmo da solução proposta nessa pesquisa foi feito em quatro etapas.

Etapa 1 - Teste Estático em Ambiente Real sem ARP: Primeiro, o código foi desenvolvido e testado em um notebook com uma câmera (*webcam*). O código captura as imagens obtidas pela câmera, as processa, identificava regiões com mais e menos obstáculos em cada uma e sugere decisões de desvio de obstáculos exibindo no monitor frases como "mover para a direita", "mover para a esquerda", "avançar"ou "parar ", dependendo da posição do obstáculo encontrado. Para criar obstáculos, foram colocados objetos na frente da câmera e, posteriormente, pessoas.

Etapa 2 - Teste Dinâmico em Ambiente Real sem ARP: Caminhando com o notebook e a *webcam* através dos corredores da Universidade, e recebendo do algoritmo instruções sobre como evitar pessoas e objetos que estavam a frente ou foram alcançados durante o percurso.

Etapa 3 - Teste Dinâmico em Ambiente Simulado com ARP: Num segundo momento, o código foi executado em um ambiente simulado e foi testado com aeronave e obstáculos virtuais. Esta etapa foi realizada em dois cenários, um contendo um único obstáculo e outro contendo vários obstáculos.

Etapa 4 - Teste Dinâmico em Ambiente Real com ARP: Finalmente, o código foi embarcado em uma ARP e testado em ambiente real.

Esse capítulo apresenta os experimentos das etapas 3 e 4.

5.2 Experimentos em Ambiente Simulado

5.2.1 Introdução

Para testar e avaliar a estratégia proposta, foram realizadas simulações de dois cenários diferentes: um simples em que a ARP tem que evitar apenas um obstáculo, e um mais complexo, onde a ARP voou por um cenário contendo 35 obstáculos.

Os experimentos simulados incluem uma placa controladora Pixhawk virtual. A ARP simulada está equipada com uma câmera, montada em sua parte frontal, para capturar as imagens do caminho a medida em que se desloca.

Em cada cenário, a ARP deveria avançar para os obstáculos e evitá-los o máximo possível. Os dados de posição da ARP foram coletados e plotados em gráficos 2D usando o software MATLAB (MATHWORKS, 1994) para uma melhor visualização física da simulação.

5.2.2 Software

Para o desenvolvimento, foi usado o sistema operacional Linux® Ubuntu® *Desktop* 14.04 LTS. O código foi escrito em linguagem C++ usando a biblioteca de código aberto (*open source*) OpenCV® 2.8 (OPENCV.ORG, 2018). A simulação foi realizada no mesmo sistema operacional, com o *firmware* ROS Indigo (ROS.ORG, 2015) e o simulador Gazebo 8.0.0 (GAZEBOSIM.ORG, 2018).

5.2.3 Hardware

Foi utilizado um notebook com processador Intel® *Pentium* N3700 com *clock* de 1.84GHz, 4.00GB de memória RAM e, para a captura de imagens, uma *webcam* com resolução de 1.3M pixels. Os experimentos em ambiente simulado foram realizados em um computador *desktop* equipado com processador Intel Core i7-4770 com *clock* de 3.40GHz x8 (8 núcleos) e placa gráfica *Gallium* 0.4 NVE4.

5.2.4 Um Único Obstáculo

Para a realização do primeiro tipo de experimento, em um ambiente simulado, foi construído no simulador Gazebo um cenário virtual contendo uma ARP e apenas um obstáculo a sua frente, conforme a Fig. 5.1.

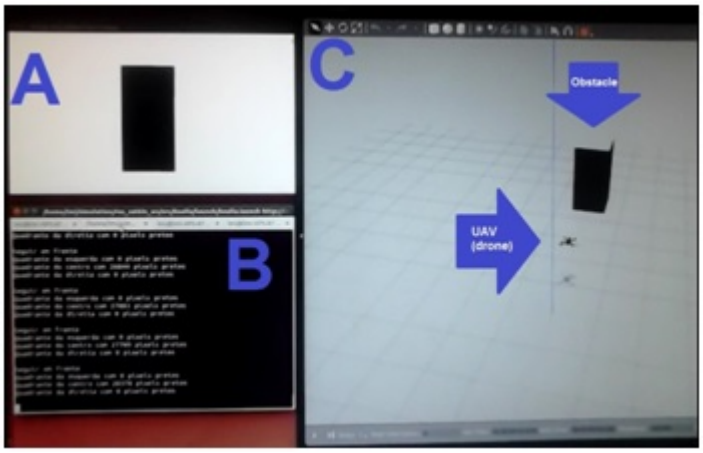

Figura 5.1 – Experimento Simulado com um Único Obstáculo.
A é a visão da câmera frontal embarcada na ARP
B é a tela de comandos de desvios e análise dos espaços livres
C é o experimento simulado.
Fonte: Produção do autor.

O obstáculo é uma caixa de 1 m de comprimento, 1 m de largura, 2 m de altura, posicionada a 5m da posição inicial da ARP, que então recebeu comandos para voar a uma altura constante de 1 m (em relação ao solo).

5.2.5 Múltiplos Obstáculos

Neste experimento, o Gazebo foi instruído a gerar 35 obstáculos dispostos aleatoriamente em uma área de 35x21m (Fig. 5.2).

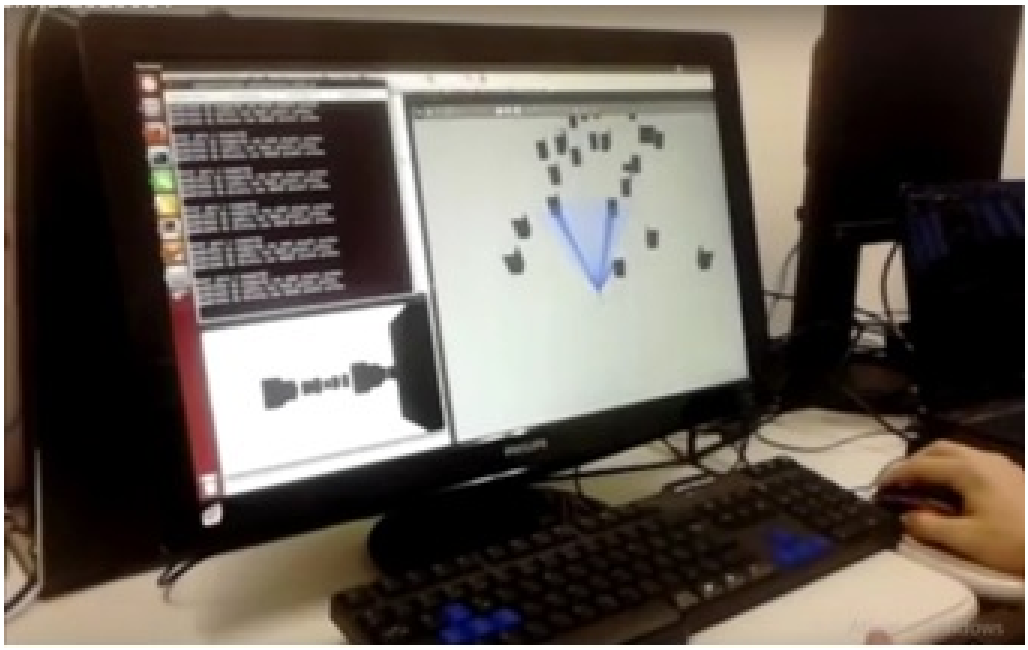

Figura 5.2 – Experimento Simulado com Múltiplos Obstáculos.
Fonte: Produção do autor.

Todos os obstáculos do cenário virtual possuem a mesma medida do único obstáculo do primeiro experimento (1x1x2m), entretanto, uma vez que eles são posicionados aleatoriamente, a distância entre cada um é desconhecida.

Da mesma forma que no experimento com apenas um obstáculo, novamente a ARP virtual recebeu comandos para voar a uma altura constante de 1 metro em relação ao solo.

Observou-se que, mesmo com um número maior de obstáculos, a aeronave foi capaz de evitar colisões enquanto navegava pelo cenário repleto de obstáculos.

5.3 Experimentos em Ambiente Real

5.3.1 Introdução

A solução proposta foi projetada para ser executada por um sistema Raspberry Pi enviando comandos de desvio para uma controladora de voo Pixhawk, estando ambos os dispositivos embarcados em uma ARP do tipo quadrotor de 450 mm, equipada com uma câmera montada em sua parte frontal, para capturar as imagens do caminho a medida em que se desloca pelo ambiente.

A estratégia empregada nos experimentos em ambientes reais é a mesma empregada nos experimentos em ambientes simulados. Entretanto, a riqueza de informações visuais em um ambiente real, que inclui brilho, sombras, reflexos e inúmeras outras características que se transformam em informações a serem tratadas, é um desafio para a visão computacional.

Para minimizar esta dificuldade, foram empregados mais tratamentos nas imagens nos experimentos em ambiente real do que naqueles em ambiente simulado.

Nos experimentos em ambiente simulado, o emprego da função cvtColor, da biblioteca OpenCV, que transforma uma imagem colorida em uma imagem em escala de cinza, foi suficiente na etapa de pré-processamento para produzir resultado útil objetivando perceber objetos.

Já os experimentos em ambiente real exigiram a aplicação de mais duas outras funções, Blur e Canny.

5.3.2 Captura da Imagem

A Fig. 5.3 mostra uma imagem capturada pela câmera embarcada na ARP durante um experimento em ambiente real externo. Trata-se de uma outra ARP também em voo.

Figura 5.3 – Imagem Primitiva Capturada Durante um Voo Real da ARP.
Fonte: Produção do autor.

5.3.3 Aplicação de Grey Scale à imagem capturada

O primeiro tratamento matemático que a imagem primitiva recebeu foi a redução do número de cores para uma banda mais estreita, limitada a tons ou graus ou escala de cinza, usando a função cvtColor.

A imagem primitiva originalmente capturada (Fig. 5.3) é convertida para uma imagem em Gray Scale (Fig. 5.4).

Figura 5.4 – Aplicação de Gray Scale à Imagem Primitiva.
Fonte: Produção do autor.

5.3.4 Desfocando a Imagem

A função blur da biblioteca OpenCV é usada para desfocar a imagem, suavizando-a de forma a diminuir o desvio padrão das cores dos pixels.

A Fig. 5.5 mostra o resultado da aplicação da suavização Blur à imagem em escala de cinza.

Figura 5.5 – Efeito Blur sobre a Imagem Gray Scale.
Fonte: Produção do autor.

5.3.5 Percebendo Contornos

A função Canny da biblioteca OpenCV é usada para aplicar o algoritmo Canny86 apresentado na seção 3.2.12, para a detecção de bordas da imagem.

O operador Canny86 objetiva atender a três critérios principais: (a) baixa taxa de erro através de uma boa detecção de apenas contornos realmente existentes. (b) boa localização: a distância entre os pixels detectados e os pixels do contorno real são mínimos. (c) resposta mínima: apenas uma resposta detectada por contorno (OPENCV, 2004).

A Fig. 5.6 mostra o resultado da aplicação da detecção de bordas Canny86 e Find Contours na Fig. 5.5.

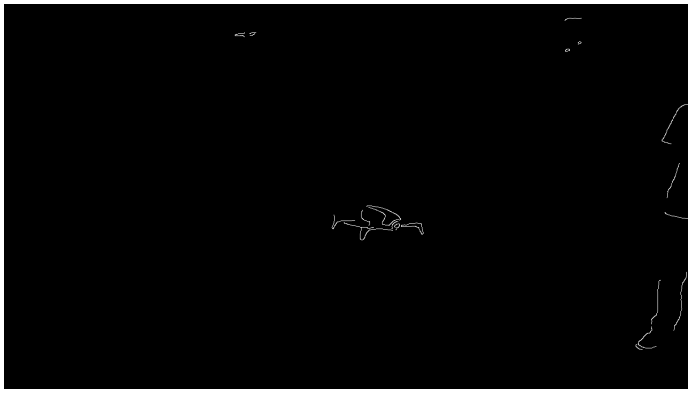

Figura 5.6 – Efeito Canny86 sobre a Imagem Blur.
Fonte: Produção do autor.

A função findContours de biblioteca OpenCV é usada para encontrar os limites dos objetos na imagem. Essa função recupera contornos da imagem binária usando o algoritmo "Suzuki85". Contornos são uma ferramenta útil para a análise de formas, detecção e reconhecimento de objetos.

As diferentes cores observadas nos contornos na Fig. 5.6 se deve ao agrupamento que essa função faz dos pixels pelas semelhanças de atributos.

A função OpenCV drawContours desenha contornos preenchidos por pontos (x,y) semelhantes.

A Fig. 5.7 mostra o último resultado do processamento da imagem após a aplicação de *Draw Contours Suzuky85* sobre *findContours*.

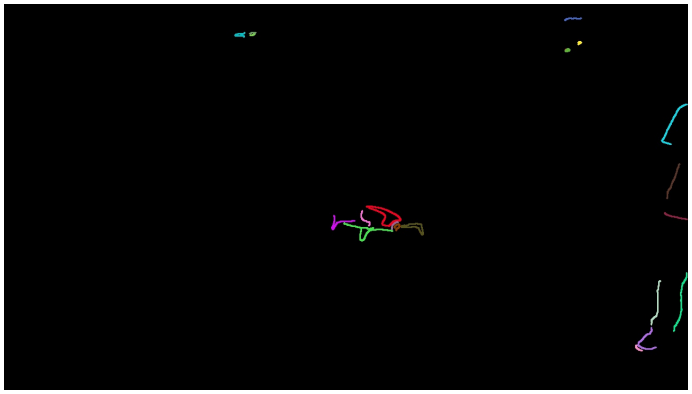

Figura 5.7 – Aplicação de *Draw Contours Suzuky05* após *Find Contours*.
Fonte: Produção do autor.

5.3.6 Identificando Obstáculos e Áreas Livres

Neste ponto, a imagem possui apenas dois tipos de pixels quanto a cor (ou luminosidade): os pretos (RGB 0,0,0) e os não pretos.

O algoritmo desenvolvido converte os pixels pretos em brancos (RGB 255,255,255, chamados *ON*) e converte todos os pixels não pretos em pixels pretos, exceto os pixels pertencentes aos contornos que terão suas cores mantidas, ou seja, os pixels pretos dentro dos contornos terão suas cores inalteradas.

Após esta operação, os pixels brancos estarão representando espaços livres e os pretos os objetos e obstáculos identificados, conforme pode ser visto na Fig 5.8.

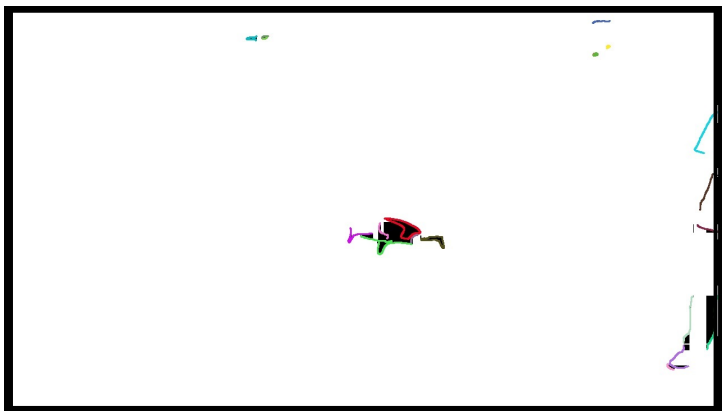

Figura 5.8 – Objeto Identificado.
Fonte: Produção do autor.

Dividindo a área em três áreas, interpretamos como sendo o caminho mais apropriado para o avanço da ARP aquele que a leva à área com mais pontos brancos.

A figura 5.9 mostra a imagem dividida em 3 áreas de interesse (segmentos). Neste exemplo, a área esquerda é a mais livre. A ARP deve se desviar do obstáculo identificado usando essa área.

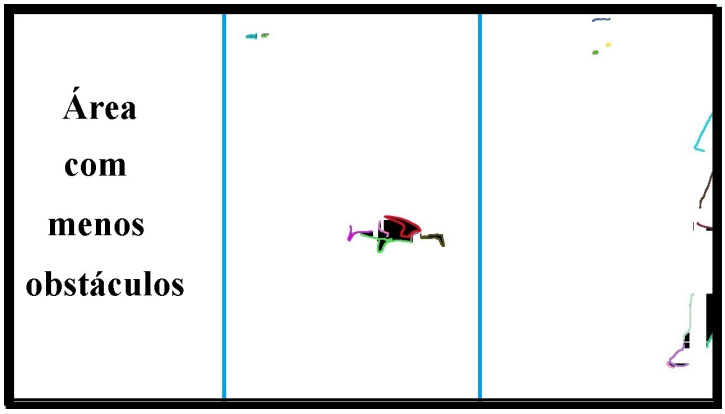

Figura 5.9 – Área Livre Identificada.
Fonte: Produção do autor.

Neste ponto, a aplicação de visão por computador está completa.

A partir deste ponto, entra em ação a segunda parte do algoritmo que comandará os componentes eletro-eletrônicos a fazer com que a aeronave se desvie de obstáculos pela área de maior espaço livre selecionada.

5.3.7 Conduzindo a ARP

Uma vez identificada a área mais livre de obstáculos, o algoritmo produz comandos que fazem com que a ARP avance passando por esta área. Os comandos geram movimento de desvio, manipulando o eixo longitudinal (ROLL), e de avanço, atuando sobre o eixo lateral (PITCH) (Fig. 5.10). O eixo vertical (YAW) não recebe nenhum comando.

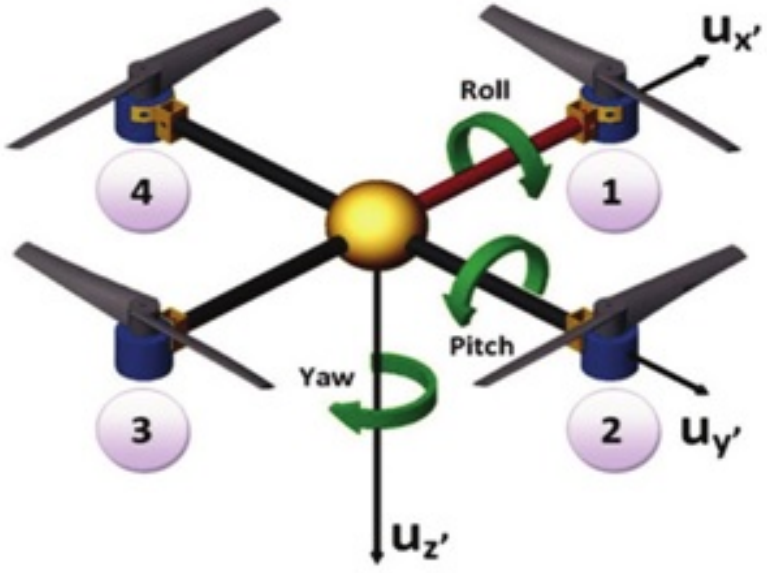

Figura 5.10 – Graus de liberdade de uma ARP do tipo quadrotor.
Fonte: (VIDEOBLOCKS, 2018).

A decisão de movimento é traduzida nestes comandos de voo e transmitida para a ARP que os executa movimentando-se como desejado, desviando-se de obstáculos e voando entre os objetos.

A Fig. 5.11 mostra o código em linguagem C++ que implementa o comando de desvio da ARP.

As variáveis msg.channels (0..3) simulam os canais do rádio controle (TX). O Canal [0] comanda o movimento de rolagem (ROLL). O canal (1), o de arfagem (PITCH). O canal (2) é o acelerador do rotor. A orientação YAW, canal (3), que não foi implementada neste trabalho, recebe o valor zero.

```
//Define Constantess}
#define FACTOR   0.6
#define BASERC  1500
  //Movimento Padrão: Não Movimento - A ARP pára}
  rollAction = 0;
  pitchAction = 0;

  //Define movimentos de desvio ou avanço
  If {desviar_a_esquerda} {
        rollAction = 100;
  }

  If (avançar_a_frente) {
        pitchAction = 200;
  }

  If {desviar_a_direita} {
     rollAction = -100;
  }

  // Determina movimento
  Roll  = BASERC - rollAction * FACTOR;
  Pitch = BASERC - pitchAction * FACTOR;

  // Preenche informaçoes dos canais do Rádio Controle
  msg.channels[0] = Roll;
  msg.channels[1] = Pitch;
  msg.channels[2] = BASERC;

  // Yaw (Não implementados desvios verticais)
  msg.channels[3] = 0;

return
```

Figura 5.11 – Código C++ para Comandar os Movimentos da ARP.
Fonte: Produção do autor.

Esta solução permite que uma ARP identifique objetos dentro do campo de visão da câmera, a uma distância maior do que nas soluções que usam outros sensores como sonar e laser, que precisam estar mais próximos dos obstáculos para que o receptor receba

de volta os estímulos sonoros ou luminosos, respectivamente, dos estímulos enviados por transmissores, que se chocam com os obstáculos e voltam para o receptor.

Combinado com estas outras soluções, a solução proposta nesta pesquisa pode ampliar as aplicações de desvio de obstáculos de ARPs podendo ser aplicada num primeiro momento e, ao se aproximar de obstáculos, usar outros sensores para aumentar a precisão e eficácia dos movimentos e decisões.

6 RESULTADOS

6.1 Introdução

Esse capítulo apresenta uma avaliação dos resultados obtidos com a solução desenvolvida quando aplicada a ambiente simulado (seção 5.2) e em ambiente real (seção 5.3).

6.2 Resultados em Ambiente Simulado

Foi incluído no código o recurso de salvar em arquivo texto as coordenadas (x,y) da trajetória da ARP durante seu movimento no ambiente simulado. Estes dados foram compilados no *software* MatLab para plotar um gráfico espaço em função do tempo, representando o deslocamento da ARP durante o voo.

A Fig. 6.1 apresenta um exemplar deste gráfico ilustrando o percurso feito pela ARP durante o experimento em ambiente com um único obstáculo.

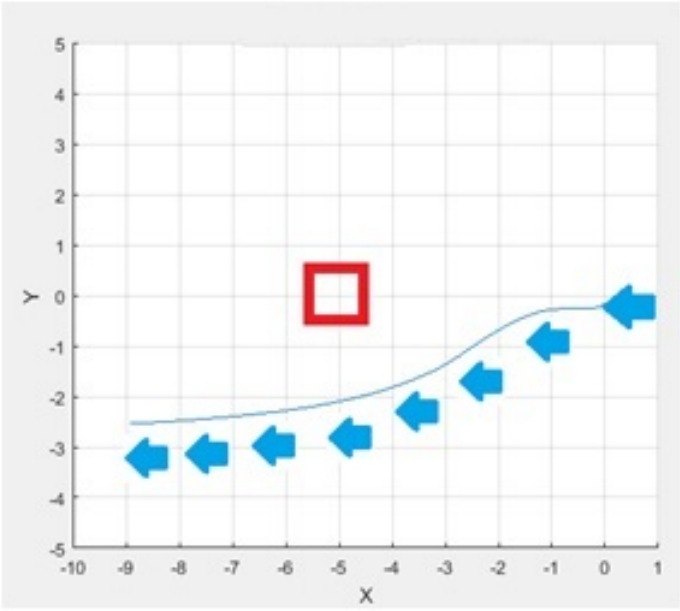

Figura 6.1 – Trajetória (em metros) do Percurso da ARP ao Desviar-se de um Obstáculo. Vermelho representa o obstáculo e azul a trajetória de desvio realizada pela ARP
Fonte: Produção do autor.

A solução desenvolvida conduziu a ARP de forma a evitar o obstáculo mantendo uma distância de pelo menos 1,5 m do mesmo.

Esta distância pode ser viável em ambientes externos (*outdoor*) mas ser inconveniente para ambientes menores, internos (*indoor*). Foi verificado nos experimentos que, reduzindo a velocidade e/ou ângulo de fuga, pode-se obter desvios com maior proximidade de objetos identificados, porém com maior chance da aeronave tocar nestes objetos. Para ambientes externos, no mundo real, com maior influência de agentes como o vento, manter uma distância de segurança mostrou ser uma medida prudente na prevenção de colisões com os objetos do cenário.

Os testes mostram que o algoritmo foi eficiente ao mover a ARP para longe de todos os obstáculos que encontrou em seu caminho.

Em ambiente simulado a solução se mostrou eficiente na quase totalidade dos experimentos, principalmente pelo fato de ser possível criar um ambiente de cores bem definidas, ao contrário do que é comumente encontrado no mundo real.

Para demonstrar os experimentos em ambiente simulado, foram produzidos dois vídeos mostrando o voo da ARP. Estes vídeos são públicos e estão disponibilizados na plataforma do YouTube. O vídeo mostrando o voo da ARP durante o experimento em ambiente com múltiplos obstáculos está disponível em (MARTINS, 2017a), enquanto o outro vídeo, mostrando o voo da ARP no experimento em ambiente com um único obstáculo, pode ser visto em (MARTINS, 2017b).

6.3 Resultados em Ambiente Real

Já em ambiente real, a solução foi eficiente em encontrar caminhos que desviassem a ARP de obstáculos a sua frente, mas encontrou dificuldade em perceber o quão próximos os obstáculos se encontravam da aeronave, causando na maioria dos experimentos um fenômeno de “vai pra esquerda”, “volta pra direita”, entrando em um loop infinito causando um voo de vai e volta no plano horizontal sem conseguir avançar ou interromper o loop. O vídeo demonstrando este comportamento pode ser visto em (MARTINS, 2017d).

Ainda foi observado em experimentos em ambiente não simulado que a solução não é eficiente quando o obstáculo é da mesma cor ou em tonalidade próxima daquela do fundo da imagem. Nestes casos, a ARP não foi capaz de perceber o objeto, e apresentou dois comportamentos distintos e indesejados. Em alguns experimentos a ARP avançou sobre o obstáculo sendo necessário que o operador remoto assumisse o controle para evitar uma colisão. Em outros experimentos a ARP iniciou movimento de escape para a direita, voando até encontrar fundo de cor distinta do que estava a sua frente no início do voo. O vídeo demonstrando este comportamento pode ser visto em (MARTINS, 2017c).

6.4 Publicação de Artigos

Além destes resultados, e daqueles descritos nos Capítulos 5.2 e 5.3, esta pesquisa gerou duas publicações em congressos internacionais, ambos valiados pela CAPES com estrato Qualis B1.

A pesquisa em ambiente simulado permitiu a publicação do artigo *"Computer Vision in Remotely Piloted Aircraft (RPA) to Avoid Obstacles During Flight"* na *"18th International Conference on Advanced Robotics (ICAR 2017)*, em Hong-Kong, China, DOI: 10.1109/ICAR.2017.8023626 (MARTINS et al., 2017).

Já o trabalho em ambiente real permitiu a publicação do artigo *Computer Vision Based Algorithm for Obstacle Avoidance (Outdoor Flight)"* na *"IEEE 15thInternational Conference on Information Technology: New Generations (ITNG 2018)*, Las Vegas, EUA, p. 1-6, DOI: 10.1109/ICAR.2017.8023626 (MARTINS; MORA-CAMINO; RAMOS, 2018).

6.5 Capítulos em Livro

Os dois artigos publicados em congressos internacionais, compuseram dois capítulos de um livro publicada em 2017 pela editora Omniscriptum Publishing Group, Saarbrücken, Alemanha, de autoria do orientador desta dissertação de mestrado, Prof. Dr. Alexandre Carlos Brandão Ramos e outros, intitulada *Drone Development and Applications: a Research Study at UNIFEI*, com ISBN 978-620-2-07716-3 (RAMOS et al., 2017).

7 CONCLUSÕES E TRABALHOS FUTUROS

7.1 Introdução

O objetivo deste trabalho foi propor uma solução embarcada em ARP que a faça se desviar de obstáculos durante o voo. A solução deve fazer uso exclusivamente de visão computacional. Para isto, a ARP deve estar equipada com uma câmera frontal. A solução deve ser capaz de perceber o cenário a frente da ARP, identificando objetos em rota de colisão. Também deve perceber áreas com menor quantidade de obstáculos, ou seja, mais livres. Deve conduzir a ARP de forma a desviar-se dos objetos identificados, voando para a área mais livre e evitando colisões.

Para atingir estes objetivos, um protótipo foi desenvolvido e testado primeiramente em um ambiente simulado e, posteriormente, em ambiente real.

Fez parte dos resultados esperados integrar a solução com o software ROS, desenvolvido para controlar sistemas autônomos em ambientes simulado e real.

7.2 Limitações da Solução Desenvolvida

A solução desenvolvida possui várias limitações que são elencadas nesta Seção.

7.2.1 Grau de Liberdade

O algoritmo desenvolvido é capaz de perceber obstáculos e comandar as ações de desvio da ARP do objeto a sua frente, no plano horizontal, e não vertical. Portanto, os movimentos possíveis são desvios a esquerda e a direita, avançar a frente e parar. Ações para cima e para baixo exigem uma percepção por parte do algoritmo da aproximação do solo ou chão e do teto ou outros obstáculos acima da aeronave controlada. para trabalhos futuros.

7.2.2 Cores

O algoritmo desenvolvido é incapaz de perceber obstáculos da mesma cor que o fundo em que estão. É preciso que os obstáculos tenham cor distinta da cor do ambiente em que se encontram.

7.2.3 Profundidade 3D

A solução proposta é incapaz de perceber a distância entre a ARP e o obstáculo detectado e, portanto, inicia o movimento de desvio tão logo o algoritmo perceba qualquer obstáculo a frente da aeronave, não importando a sua distância entre eles.

7.2.4 Ambiente

A solução apresentada não lida com imagens descontínuas, como as que contém folhagens e arbustos. Neste tipo de cenário, uma área pode conter vários pequenos espaços livres de obstáculos, porém espalhados, desconexos, podendo ser interpretada como uma área livre de obstáculos, quando o número total de pixels indicar uma possibilidade de ser um espaço livre para a ARP avançar. Neste caso, entretanto, o arranjo espacial físico poderia impedir o avanço da aeronave sem colisões.

7.2.5 Dimensões

A solução descrita nesta pesquisa não trata as dimensões da ARP, ou seja, é incapaz de mensurar se o espaço livre identificado a frente da aeronave é suficientemente grande para que ela possa passar pelo caminho sugerido, podendo causar colisões.

7.2.6 Distância dos Objetos

A solução desenvolvida não mensura a distância entre a ARP e os objetos identificados a sua frente, iniciando o desvio dos mesmos tão logo sejam percebidos pelo algorítimo. Nos experimentos em ambiente simulado isto não apresentou nenhuma inconveniência, mas nos experimentos em ambiente real causou, na maioria das vezes, um desvio prematuro. Por não haver o mesmo tratamento de percepção e desvio de obstáculos lateralmente, isto pode levar a uma colisão com objetos posicionados nas laterais da aeronave ou mesmo em paredes no caso de voo *indoor*.

7.3 Dificuldades

Como é comum em visão computacional, as dificuldades se relacionam com as aplicações em um mundo real, principalmente quanto a ruídos, causados por reflexos, iluminação, sombras, e outros aspectos que interferem nas imagens obtidas pela ARP em tempo real.

Também é um desafio para a visão computacional concluir se a ARP caberá em uma área supostamente livre ou se suas dimensões impedem que se use esta área identificada como uma possibilidade real de desvio. Na prática são usados outros sensores

como o laser para estimar dimensões, à medida em que a ARP, encaminhada para a área identificada como a mais livre de obstáculos, se aproxime desta região sugerida.

Estavam disponíveis para os experimentos em ambiente real, duas ARPs. Uma de 450 mm, montada na Universidade, de arquitetura aberta (seção A.2), e outra, modelo Bebop2 (seção A.10), de arquitetura fechada. A primeira, que estava equipada com o sistema Raspberry Pi e controladora Pixhawk, não conseguiu embarcar a solução desenvolvida pois apresentou panes e não conseguia força suficiente para o voo. Não foi possível solucionar o problema a tempo de concluir a dissertação. Por este motivo, foi usada a ARP Bebop2, mas sem o uso da Raspberry Pi e da Pixhawk, uma vez que esta aeronave não permite embarcar estes componentes. A solução foi enviar apenas os comandos de rádio controle do ROS via wi-fi para a ARP que executou os desvios comandados.

Entretanto, para o ROS, não importa se a solução é simulada ou real. Cada nó desenvolvido grava e lê mensagens nos tópicos e em formatos de específicos. Esta é uma das vantagens do uso do ROS em desenvolvimento de protótipos robóticos.

Portanto, o que foi desenvolvido é totalmente aplicável às especificações de uma ARP como a simulada, contendo o sistema Raspberry Pi e a controladora Pixhawk.

7.4 Conclusões Finais

O algoritmo desenvolvido neste trabalho, mesmo fazendo uso de menos recursos que outras propostas similares, mostrou-se eficiente na condução de uma ARP através de obstáculos no ambiente simulado sem causar colisões. Existem, no entanto, várias limitações, principalmente para o ambiente real, listadas na seção 7.2, e possíveis melhorias sugeridas na seção 7.5, que permitem outras pesquisas relacionadas que podem enriquecer o presente trabalho na condução eficiente de uma ARP de forma autônoma controlada, por ambientes internos e externos.

A pesquisa propiciou um aprofundamento no aprendizado dos temas tratamento de imagens, visão por computador, framework ROS de criação de protótipos robóticos, criação de ambientes simulados na plataforma Gazebo, mecânica de voo de ARPs, legislação vigente, programação de embarcados, ferramentas de robótica, além da oportunidade de participação em eventos internacionais da área.

7.5 Trabalhos Futuros

Uma das características das ARPs do tipo quadrotor é poder girar no próprio eixo, ou eixo vertical (YAW), aumentando a versatilidade de operações.

O algoritmo proposto é capaz de mover a ARP apenas horizontalmente (esquerda,

direita e a frente). Como os ARPs do tipo quadrotor são capazes de movimento vertical e 3D completos, uma melhoria seria estender o algoritmo também para impulsionar a ARP sob ou sobre os obstáculos e em mais graus de liberdade. Para isto podem ser empregadas outras duas câmeras, localizadas sob e sobre a aeronave, ou o emprego de sensores como barômetro, laser, sonar, para permitir movimentos da ARP no eixo vertical.

Como as soluções de visão por computador são sensíveis às mudanças na iluminação, trabalhos futuros podem explorar a solução apresentada em outras condições relacionadas a iluminação.

Ainda limita a solução proposta o fato de não tratar profundidade 3D nas imagens capturadas, sendo impossível determinar a distância entre a ARP e os obstáculos a sua frente. Uma solução clássica para esse problema é o emprego de visão *stereo*, que usa duas câmeras ao invés de monovisão, adotada nesta pesquisa, que usa apenas uma câmera.

O presente trabalho não trata das dimensões da própria ARP, nem do ambiente a sua volta, de forma a permitir mensurar se a aeronave cabe entre os espaços identificados como livres de obstáculos. Uma abordagem neste sentido também pode ser desenvolvida em novas pesquisas.

Esta pesquisa também não trata o voo da ARP em ambientes com imagem fragmentada, como no caso de folhagens, nem resolve o problema de obstáculos na mesma cor que o restante do ambiente, quando a detecção é mais difícil. Estes desafios também são temas que podem ser explorados por outras pesquisas.

Esse trabalho limitou-se a evitar obstáculos, mas não a executar um plano de voo. Conduzir a ARP de volta a sua rota original depois de esquivar-se de um ou mais obstáculos não esta no seu escopo mas pode enriquecer esta pesquisa gerando novos resultados.

A divisão da área da imagem em três partes foi arbitrada de maneira estática e poderia ser feito de uma forma dinâmica aplicado uma varredura horizontal ou vertical de forma a isolar os espaços da imagem com maior ou menor espaço livre para, em seguida, aplicar novos tratamentos e análises apenas na área de maior interesse, permitindo uma maior precisão e acerto do processo de desvio de obstáculos e navegação.

Outros recursos de Inteligência Artificial poderiam ser aplicados, melhorando as decisões, mas foge do escopo deste trabalho.

Finalmente, é possível aprimorar a eficiência da ARP no desvio de obstáculos empregando outros sensores combinados, como sonar e *laser*. Todavia, neste caso, distanciando-se um pouco do objetivo primário desta pesquisa que é o uso de poucos recursos e basear-se exclusivamente no emprego de visão de computador para fazer com que uma aeronave identifique e se desvie de obstáculos.

Apêndices

APÊNDICE A – COMPONENTES DE UMA ARP DO TIPO QUADROTOR

A.1 Introdução

Há diversos tipos e subtipos de ARP's segundo suas características aerodinâmicas, como as de asas fixas (aviões) e as de asas rotativas (helicópteros e multirotores) (ANGELOV, 2012).

É possível dividir o *hardware* de uma ARP do tipo asa rotativa, como os quadrotores empregados nesta pesquisa, em duas partes: a plataforma e os sistemas embarcados. A função da plataforma (balões brancos na Fig. A.1) é transportar os embarcados (balões amarelos) garantindo um voo estabilizado e seguro. O sistema embarcado é o conjunto combinado de um ou mais equipamentos ou sensores, como as câmeras e transmissores de video responsáveis por capturar e transmitir as informações do ambiente durante o voo, e acessórios como protetores de hélices, alarmes, ganchos, e pequenas redes para o transporte de objetos.

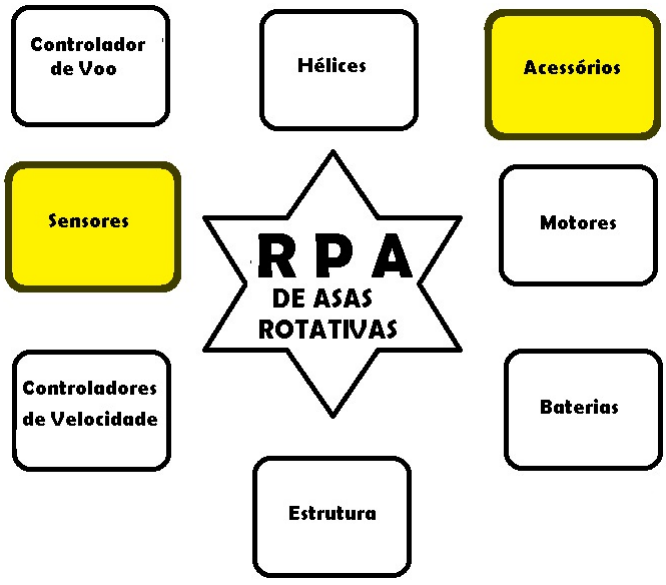

Figura A.1 – Componentes de uma ARP de Asas Rotativas.
Fonte: Produção do Autor.

A seguir são descritos alguns dos principais componentes de uma ARP.

A.2 Estrutura

A estrutura, *frame* ou "chassi"(Fig A.2), é a base física da ARP e é nela que todos os demais componentes estão fixados.

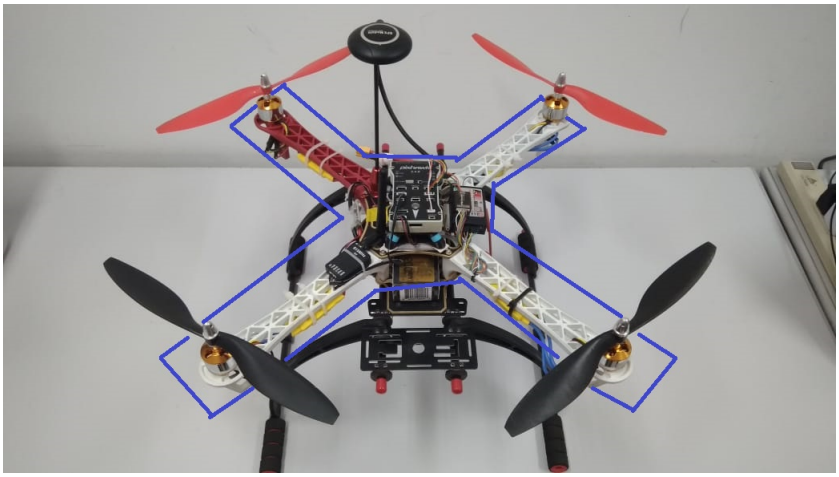

Figura A.2 – Frame de uma ARP Quadrotor (destaque em azul).
Fonte: Produção do Autor.

Há várias categorias de frames, confome a aplicação da ARP, como os de cinematografia aérea, capazes de levantar câmeras profissionais de maior peso, os esportivos leves para facilitar movimentos rápidos e radicais em acrobacias, os FPV, visão em primeira pessoa) que permitem acomodar vários acessórios como câmeras, transmissores e antenas para passar a impressão ao operador de que ele está voando a bordo da aeronave, e os mini frames, bem menores e resistentes a quedas.

A.3 Baterias

A bateria (Fig. A.3) fornece a energia para o funcionamento dos motores e de todos os componentes eletro-eletrônicos embarcados na ARP, sendo, portanto, um item vital para seu voo.

Figura A.3 – Bateria Tattu 5200 usada na ARP.
Fonte: Produção do autor.

Infelizmente, a alimentação elétrica ainda é um ponto crítico para todos os usuários de ARPs de asas rotativas, tanto para uso doméstico quanto profissional, uma vez que praticamente nenhuma aeronave atualmente consegue oferecer autonomia superior a 30 minutos de voo, como mostra a Tabela A.1.

Tabela A.1 – Os Dez Drones Comerciais com Maior Autonomia de Voo (2017)

Produto	Tempo de Voo	Alcance de Controle
Blade Chroma Quadcopter Drone	30 min	2500 m
Sim Too Pro	30 min	1000 m
DJI Phatom 4	28 min	3500 m
DJI Mavic Pro	27 min	7000 m
DJI Inspire 2	27 min	7000 m
Parrot Bebop 2	25 min	3200 m
DJI Phantom 3 Standard	25 min	1500 m
DJI Phantom 3 Pro	23 min	3000 m
3DR Solo	22 min	500 m
Yuneec Q500+	22 min	2000 m

Conforme as especificações de seus fabricantes. Fonte:Filmora (LOPES, 2018)

Chuva, neblina, ventos e até a pressão atmosférica influenciam no consumo de energia e tempo de voo da ARP. Além disto, esforços para manobras e para a estabilização da aeronave custam uma carga extra da bateria. No caso da pressão atmosférica, quanto mais altitude mais rarefeito fica o ar, implicando em maior necessidade de rotações do motor para manter a ARP em voo, resultando em maior consumo de energia da bateria.

Para maximizar a autonomia de voo, é preciso usar bateria que combine satisfatoriamente sua capacidade de carga, capacitância e consequentemente capacidade de descarga. A harmonia entre estas características vai proporcionar uma bateria mais leve e com melhor capacidade de alimentação. A quantidade de ciclos (recargas) que uma bateria já gastou também influencia no tempo de voo. Baterias mais velhas não carregam completamente ou não balanceiam suas células perfeitamente.

O tamanho, formato e material de que as hélices são feitas também são itens que influenciam o tempo de voo da ARP. Hélices muito pequenas resultam em menos capacidade de carga obrigando os motores a fazerem um esforço extra para manter a aeronave em voo, enquanto hélices muito grandes sobrecarregam e superaquecem o motor, diminuindo sua eficiência e o tempo de voo, com risco de queima de motor e consequente queda da ARP, o que pode causar acidentes e danos com consequências imprevisíveis.

A.4 Motores

Os motores para ARPs (Fig. A.4) são do tipo *brushless* (sem escova ou *outrunners*).

Por não possuírem escovas duram mais que motores *brushed* (com escova). Enquanto nesses o núcleo de bobina do motor gira, nos *brushless* o núcleo de bobina permanece parado, enquanto todo o exterior gira. Por esse motivo são chamados *outrunner*.

Figura A.4 – Motor para ARPs.
Fonte: Produção do Autor.

Além disso, os *brushless* são motores de passo, ou seja, a combinação de diversos passos em alta velocidade se traduzem na eficiência desse tipo de motor. ARPs profissionais ou de alta performance usam motores *brushless*.

A.5 Hélices

A hélice de uma ARP consiste em duas ou mais pás (Fig. A.5) conectadas ao cubo central na raiz da pá, ao qual essas pás são fixadas. Cada pá é essencialmente uma asa rotativa, e toda pá é um perfil aerodinâmico capaz de gerar uma sustentação. Essa força de sustentação no plano em que a pá se desloca recebe o nome de tração ou propulsão.

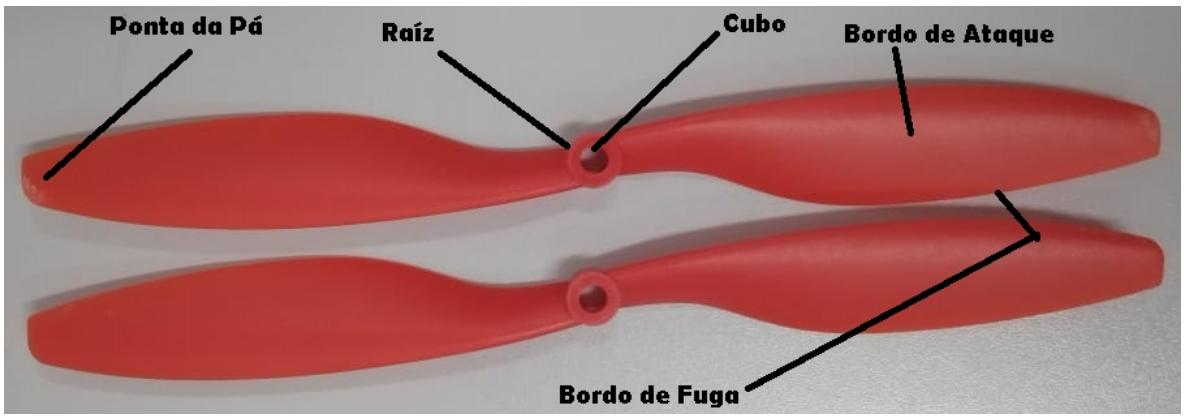

Figura A.5 – Componentes de uma Pá.
Fonte: Produção do Autor.

Com a rotação, as pás da hélice "cortam" o ar e criam um efeito aerodinâmico, como os da sustentação da asa ou da estrutura da aeronave, ou seja, o deslocamento

circular da pá no ar provocará uma baixa pressão no dorso da pá, e uma alta pressão na face, provocando a tração.

As hélices é que fazem a aeronave sair do solo, voar e pousar, e a eficiência de uma hélice está na razão direta entre a força colocada pelos motores (*input power*) e o resultado obtido (*output power*). A força colocada é a força que o motor precisará fazer para girar a hélice. Quanto mais força o motor fizer maior será o consumo da bateria. O resultado obtido é a capacidade da hélice de produzir empuxo (*thrust*) sob uma determinada velocidade de vento (*air speed*).

As hélices podem ser de uma ou mais pás. As mais comuns são as confeccionadas em plástico (ABS, Nylon, etc) e as de fibra de carbono, existindo ainda as confeccionadas com outros materiais, como as de madeira, mais caras e incomuns.

Em quadrotores, as hélices mais utilizadas são as de plástico (diversos tipos de materiais plásticos) e são as mais baratas e indicadas para iniciantes, uma vez que no início do aprendizado da condução de um quadrotor é muito comum serem danificadas. As hélices de plástico fazem mais barulho que as hélices de fibra de carbono, possuem uma performance inferior e vibram mais. Por serem flexíveis, dobram em alta rotação, sendo claramente uma desvantagem.

Hélices de fibra de carbono vibram menos, sendo indicadas para filmagens aéreas. Fazem menos barulho, e são mais estáveis. Entretanto, são altamente perigosas, pois se transformam em verdadeiras lâminas cortantes quando em alta rotação. Não é indicada para iniciantes devido ao custo, que é bem superior ao das hélices de plástico e do perigo de corte oferecido. São rígidas, não flexionam quando em alta rotação, e exigem mais dos motores, podendo consumir um pouco mais que as hélices de plástico. Por serem feitas de fibra de carbono, são mais leves que as de plástico (MOLRC1, 2016).

Existem hélices híbridas de plástico e fibra de carbono, numa tentativa de aumentar a qualidade a um preço mais acessível.

A.6 ESC

ESCs, do inglês, *Electronic Speed Controllers* (Controladores Eletrônicos de Velocidade) (Fig. A.6) são mecanismos que transformam os pulsos PWM (*Pulse Width Modulation*) recebidos da placa controladora de voo em rotação dos motores, enviando-lhes pulsos elétricos fazendo os motores girem em determinada velocidade.

Figura A.6 – ESC para ARPs.
Fonte: Produção do Autor.

ESCs são compostos por transistores de potência e um microcontrolador. Para cada categoria de motor existe uma categoria específica de ESCs, pois variam em potência, taxa de atualização em hertz (*refresh rate*), aplicações por tipo de motor (brushless ou *brushed*), suporte a determinados tipos de baterias, e outros recursos.

A Fig. A.7 ilustra a disposição dos ESCs montados em uma ARP, com destaque em azul.

Figura A.7 – ESCs montados em uma ARP.
Fonte: Produção do Autor.

Os ESCs são classificados segundo suas potências em amperes (A), existindo ESCs de 12A, 20A, 30A, etc.

A frequência de atualização (taxas de atualização) em Hz implica diretamente na velocidade de resposta da RPA aos comandos enviados para os motores. ESCs mais antigos trabalhavam com taxas de 50Hz, enquanto os mais atuais operam em taxas de atualização de 400Hz (MOLRC2, 2016).

Alguns modelos possuem sistema de reversão, capazes de fazer os motores/hélices girarem ao contrário, permitindo grande desaceleração da aeronave, fazendo com que toque o solo suavemente em baixíssima velocidade e em um espaço reduzido (ECODRONES, 2018).

A.7 Rádio Controle - TX

Os rádios controles (Fig. A.8) são aparelhos que permitem controlar o voo e manobras de uma ARP por um operador a partir do solo e podem substituir a estação de controle em caso de necessidade.

Figura A.8 – Rádio Controle - TX.
Fonte: Produção do Autor.

São compostos de chaves de posição e alavancas de controles chamados de *Sticks*, cada um deles geralmente ligado a um canal da frequência operada.

Nos multirotores, ao manipular os comandos do rádio controle (TX), o operador transmite a informação para o receptor (RX) embarcado na ARP, que a leva para o controlador de voo que a interpreta e atua nos ESCs de forma a mudarem a velocidade dos motores produzindo o movimento comandado remotamente (TX).

A.8 Piloto Automático

O piloto automático da ARP permite diversas operações, como estabilização da aeronave e alguns controles de voo. A Pixhawk é um exemplo de piloto automático que possui vários modos de voo pré-definidos, como os listados a seguir, dentre outros:

Modo *Stabilize*: Esse modo permite o nivelamento automático da ARP, de forma que o operador (piloto) da ARP não tenha que se preocupar com o nivelamento da aeronave, mas apenas com a sua altitude e direção.

Modo *Alt Hold*: Mantém a altitude da ARP. A altitude é mantida automaticamente por sensores como barômetro, sonar ou outro método, dependendo da ARP. Este modo é recomendado para iniciantes por facilitar a operação da aeronave.

Modo *Loiter*: Esse modo é também conhecido como GPS ou "*Position Hold*". Utiliza GPS para manter a posição, barômetro para manter a altitude, e a bússola para manter a direção. A ARP se mantém relativamente fixa em uma mesma posição no espaço, dispensando o operador de qualquer necessidade de interação com o rádio controle enquanto a ARP permanece praticamente "parada"(imóvel) no ar.

Modo *RTL (Return-to-Launch)*: Também conhecido como modo RTH (*Return-to-Home*), esse modo faz com que a ARP retorne para o local de partida (onde iniciou o voo).

Modo *Auto*: Esse modo executa uma missão planejada através de um sistema ou *software Mission Planner*.

Modo *Guided*: Esta função está disponível apenas através do uso de uma estação de solo executando o *software Mission Planner* e a telemetria. Permite que o operador controle a ARP através de um clique no mapa do *Mission Planner*. A ARP se deslocará para o local apontado pelo operador.

Para conseguir executar as funções mencionadas, o piloto automático conta com alguns sensores próprios, a saber:

A.8.1 IMU

IMU (*Inertial Measurement Unit*) é um dispositivo que agrega vários sensores, como acelerômetro, giroscópio e sensor de temperatura. Os dois primeiros sensores com três eixos de medição permitem resposta de saída em seis valores, permitindo seis graus de liberdade, cujo acrônimo é DOF (*Degree Of Freedom*).

A.8.2 Barômetro / Altímetro

Altímetro é um tipo de barômetro, uma ferramenta que permite ler a pressão do ar. Quanto mais alto a aeronave está, mais rarefeita é a atmosfera e, portanto, menor a pressão do ar, permitindo, assim, um algoritmo estimar a altitude da ARP (distância entre a aeronave e o nível do mar).

A.8.3 Acelerômetro

O acelerômetro é um dispositivo com sensores que, quando deslocados de cima para baixo ou para os lados, no espaço de eixo 3D, por movimentos de voo, percebe a direção e velocidade do deslocamento.

A.8.4 Giroscópio

O giroscópio mede a velocidade angular. Um exemplo de medição seria 1 volta por segundo, o que representaria 360 graus ou, 360/1. A direção em que o aparelho está girando também é reconhecida. Mas dependendo da aplicação, pode não haver giro, mas apenas deslocamento de graus, comum em ARPs. Detectando pequenas variações é possível estabilizar o voo.

A.8.5 Bússola

Uma bússola giroscópica possui 3 eixos, enquanto a magnética aponta apenas e sempre para o norte. A bússola giroscópica mecânica corrige as inclinações causadas por morros, influências marítimas e inclinação de aeronaves. A bússola digital ou bússola eletrônica utiliza a tecnologia magneto-indutiva, lendo as variações no campo magnético da Terra. Por se tratar de um dispositivo sensível a campos magnéticos, diversos cuidados periféricos devem ser tomados, como por exemplo considerar a massa metálica próxima do sensor, campos eletro-magnéticos e interferências causadas por circuitos ou fontes próximas a ele (OPENCV, 2017).

A.8.6 Telemetria

Sistema de comunicação em duas vias (tanto recebe quanto transmite informações) sem fio via antenas (Figs. A.9 e A.10) e que permite troca de dados entre a controladora de voo embarcada e a estação de controle (Fig. A.8).

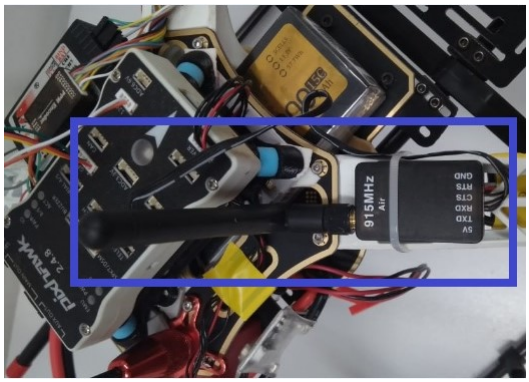

Figura A.9 – Antena Wi-fi Embarcada.
Fonte: Produção do Autor.

A telemetria oferece ao piloto informações e métricas do voo enviando a um receptor em solo todos vários tipos de informações.

Uma das das informações importantes que a telemetria fornece para o controlador em solo é a RSSI, que indica a intensidade do sinal do rádio, útil para perceber quando a aeronave está se aproximando do limite máximo de alcance do rádio controle, ou a inclinação está desfavorável para a transmissão de dados pelas antenas.

A tensão de carga das baterias é outra informação transmitida pela telemetria para o rádio controle, útil para estimar o quanto ainda resta de autonomia de voo da ARP.

Figura A.10 – Antenas Wi-fi.
Fonte: Produção do Autor.

Embora seja comum ver operadores usando cronômetros para estimar o tempo de voo da aeronave, fatores como a intensidade de vento contrário pode exigir esforço adicional dos motores e consequente aumento de consumo de energia das baterias diminuindo a autonomia do voo, o que torna esta prática desaconselhável.

A.9 Sensores

Para auxiliar a estabilidade e permitir o voo livre (quando o sinal de GPS está indisponível), as ARPs contam com uma série de equipamentos internos e externos como bússola, altímetro, giroscópio e barômetro. Uma ARP pode usar vários sensores embarcados de muitas formas distintas.

A.9.1 Sonar

Dispositivo eletrônico que permite detectar a distância entre a ARP e o solo ou entre a ARP e obstáculos a sua frente, dependendo de onde o aparelho é instalado, abaixo ou à frente da aeronave. Este sensor combinado com o altímetro permite uma precisão maior na detecção de altitude.

A.9.2 Laser

Sensores ou *scanners* a *laser* permitem os escaneamento embarcado em ARP's, possibilitando perceber obstáculos e relevos durante o voo. Este método de percepção do ambiente é chamado de sistema LIDAR (*Light Detection and Ranging*) e utiliza a emissão de laser pulsado com elevada frequência de repetição para medir distâncias através do registro do reflexo destes pulsos, permitindo a obtenção de informações tridimensionais acerca de superfícies ao montar uma tabela de dados de posição (x, y) e de elevação (z), gerando uma nuvem de pontos.

A.9.3 Outros Embarcados

Outros equipamentos e acessórios podem ser embarcados em uma ARP, como o módulo/antena GPS (Fig. A.11), o sistema Raspberry PI, o controlador Pixhawk e outros.

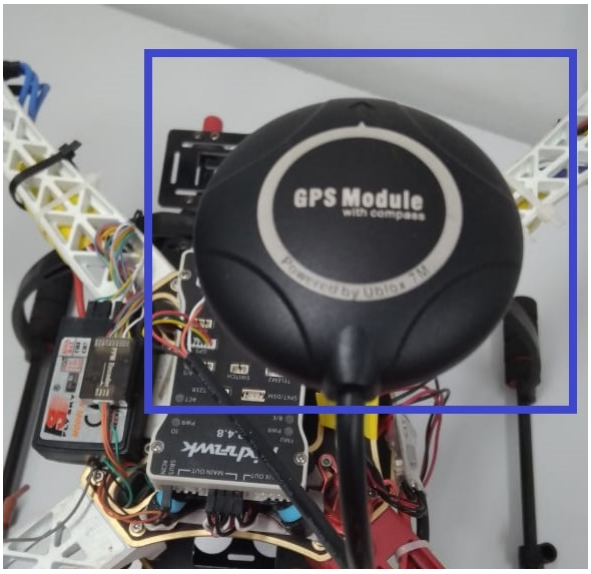

Figura A.11 – Módulo GPS embarcado.
Fonte: Produção do Autor.

Raspberry Pi (Fig. A.12) é "mini-microcomputador"similar à placa mãe (*mother board*) encontrada no interior do gabinete de um microcomputador PC (*Personal Computer*) convencional, contendo processadores, *slot* de expansão, interfaces USB (*Universal*

Serial Bus), saídas digitais de áudio e Vídeo HDMI (*High-Definition Multimedia Interface*), memória RAM (*Random Access Memory*), conector do tipoRJ45 pra cabo de rede e entrada para alimentação de energia elétrica.

Figura A.12 – RaspBerryPi.
Fonte: Produção do autor.

O computador Raspberry Pi foi desenvolvido pela Fundação Raspberry, com objetivo de levar a tecnologia para o ensino de jovens e crianças.

Pixhawk (Fig. A.13) é um dispositivo de *hardware* e *software* que implementa um piloto automático OSHW (*Open Source Hardware*) capaz de controlar as operações básicas de uma RPA, como se estabilizar, decolar e pousar, entre várias outras.

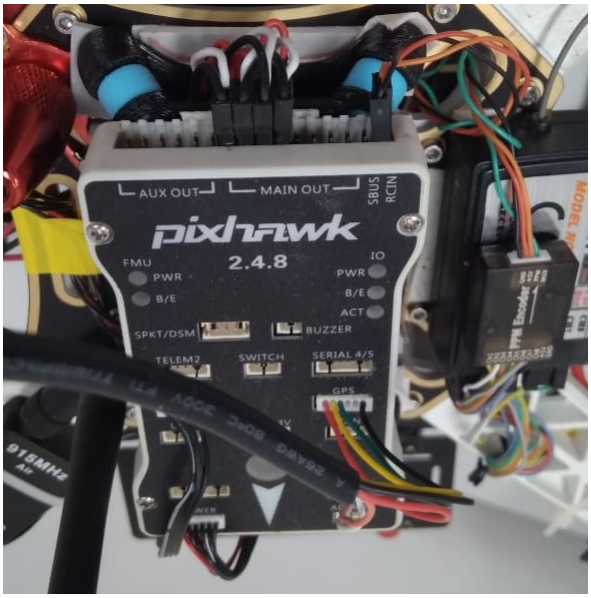

Figura A.13 – Pixhawk.
Fonte: Produção do Autor.

O sistema Pixhawk é multi-tarefa e compartilha recursos que podem ser executados em conjunto ou individualmente.

Possui um textitsoftware incorporado que implementa as funções básicas de controle de voo.

Oferece um ambiente de programação compatível com sistemas operacionais Unix e Linux e funções de piloto automático integrado com *logs* (registros) detalhados de missões e comportamento de voo. Está equipada com giroscópio, acelerômetro, magnetômetro e barômetro e pode ser conectada a acessórios externos como um módulo GPS.

Uma característica útil do Pixhawk é a capacidade de se comunicar com outros dispositivos através de um protocolo chamado MAVLink, que foi desenvolvido especificamente para aplicações em ARP, podendo ser usado ao alcance do controle remoto da aeronave autônoma, executando os algoritmos de controle em um pequeno computador portátil, como um Raspberry Pi, que é embarcado na ARP e envia comandos para a Pixhawk via mensagens no protocolo MAVLink.

A.10 ARP Parrot Bebop2

Para os experimentos em ambiente real desta pesquisa, foi utilizada uma ARP modelo "drone Bebop 2 FPV"(Fig. A.14) da marca Parrot disponibilizada pela equipe BlackBee da Universidade Federal de Itajubá (UNIFEI) (BLACKBEE, 2018).

Figura A.14 – ARP Parrot Bebop 2.
Fonte: Produção do Autor.

Segundo o site do fabricante, esta aeronave pesa 500 g. Possui autonomia para até 25 minutos de operação. Pode voar a até 2 km de distância do rádio-controle. Um único toque de botão no rádio-controle faz com que a aeronave decole e aterrisse. Possui recurso

Return-to-Home que a faz voltar ao ponto de onde decolou. É equipada com sistema de estabilização digital de três eixos que garante vídeos estáveis e sem tremores. Pode ser controlada por aparelhos iPhone ou iPad. Possui câmera grande-angular integrada que grava vídeos full HD a 1080 p e captura fotos de 14 Mp. Grava vídeos em formato RAW, JPEG e DNG. Usa bateria de 2700mAh, de íon de lítio recarregável, com tempo de recarga de aproximadamente 55 minutos. Possui 32,77cm de altura (12,9 pol), 38,1 cm de comprimento (15 pol) e 8,89 cm de largura (3,5 pol) (PARROT, 2018).

Referências

AMENYO, J.-T. et al. *MedizDroids Project: Ultra-low cost, low-altitude, affordable and sustainable UAV multicopter drones for mosquito vector control in malaria disease management*. 2014. IEEE Global Humanitarian Technology Conference (GHTC 2014), 2014. 24

ANAC. *Notícias no website da ANAC: Regulamentação da ANAC sobre drones completa um ano em vigor*. 2018. <http://www.anac.gov.br/noticias/regulamentacao-da-anac-sobre-drones-completa-um-ano-em-vigor>, acessado em 11/05/2018. 20, 22

ANAC. *Quantidade de Cadastros*. 2018. <http://www.anac.gov.br/assuntos/paginas-tematicas/drones/quantidade-de-cadastros>, acessado em 11/05/2018. 22

ANDRESEN, C.; JONES, S. D.; CROWLEY, J. L. *Appearance Based Processes for Visual Navigation*. 2001. Symposium on Robotic Systems – SIRS 97, p. 227-236, 1997. 28

ANGELOV, A. *Sense and avoid in UAS: research and applications*. 2012. First edition. United Kingdom: Wiley, 2012. 73

ANTONELLO, R. *Introdução a Visão Computacional com Python e OpenCV*. 2010. <http://professor.luzerna.ifc.edu.br/ricardo-antonello/wp-content/uploads/sites/8/2017/02/Livro-Introdu%C3%A7%C3%A3o-a-Vis%C3%A3o-Computacional-com-Python-e-OpenCV.pdf>, acessado em 16/09/2018. 37, 39

BERTOZZI, A. B. M.; FASCIOLI, A. *Vision-based intelligent vehicles: state of the art perspectives*. 2000. 8, 9

BLACKBEE. *Equipe BlackBee Drones - UNIFEI*. 2018. Disponivel em <https://blackbee.unifei.edu.br/>, 2018, acessado em 115 de setembro de 2018. 85

BRAGA, J. R. G. *Navegação Autônoma de VANT por Imagens LIDAR*. 2018. Tese (Doutorado em Computação Aplicada) - Curso de Pós-graduação em Computação Aplicada, Instituto Nacional de Pesquisas Espaciais - INPE, São José dos Campos, SP, 2018. 24

BRAGA, R. G. et al. *Collision avoidance based on Reynolds Rules: a case study using quadrotors*. 2018. Information Technology-New Generations (ITNG). 773-780, 2018. 30

BUENO, A. *Fundamentos da Computação Gráfica*. 2011. Pontifícia Universidade Católica, Rio de Janeiro,RJ, 2011. 48, 50

CAI, W.; ZHANG, M. Y.; ZHENG, Y. R. *Task Assignment and Path Planning for Multiple Autonomous Underwater Vehicles Using 3D Dubins Curves*. 2017. Sensors 2017, 17, 1607. 30

CANNY, J. *A Computational Approach to Edge Detection*. 1986. IEEE - Transactions on Pattern Analysis and Machine Intelligence, Vol PAMI-8 pp 679-698, 1986. 38, 40

CAZANGI, R. R. *Uma proposta evolutiva para controle inteligente em navegação autônoma de robôs*. 2004. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Estadual de Campinas (UNICAMP), Campinas, SP, 26p, 2004. 41

CONTE, G.; DOHERTY, P. *Vision-Based Unmanned Aerial Vehicle Navigation Using Geo-Referenced Information*. 2009. 1-18 p. <http://liu.diva-portal.org/smash/record.jsf?pid=diva2%3A234039&dswid=-5448>, acessado em 13/05/2018. 24

CRESTANI, P.; FIGUEIREDO, M.; VON-ZUBEN, F. *Sistemas inteligentes de navegação autônoma: uma abordagem modular e hierárquica com novos mecanismos de memória e aprendizagem*. 2002. XXII Congresso da Sociedade Brasileira de Computação (SBC 2002), Concurso de Teses e Dissertações (CTD 2002), 2002, Florianópolis. Anais do XXII Congresso da Sociedade Brasileira de Computação (SBC 2002), 2002. v1. p599 – 604. 24, 25

DAVISON, A. *Real-Time Simultaneous Localization and Mapping with a Single Camera*. 2003. IEEE International Conference on Computer Vision, pp. 1403-1410, 2003. 33

DPI.INPE. *Teoria : Processamento de Imagens*. 2010. <http://www.dpi.inpe.br/spring/teoria/filtrage/filtragem.htm>, acessado em 16/09/2018. 37

ECODRONES. *E386 desenvolvido pela Event 38 Unmanned Systems*. 2018. Disponivel em <https://store.ecodrones.com.br/rpas-e386-mapeamento-profissional-laser-rangefinder>, 2018, acessado em 11 de setembro de 2018. 79

ESCHMANN, C. et al. *Unmanned aircraft systems for remote building inspection and monitoring*. 2012. 6th European workshop on structural health monitoring. 2012. 29

FABRO, J. A. *Grupos neurais e sistemas nebulosos: aplicação à navegação autônoma*. 1996. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Estadual de Campinas (UNICAMP), Campinas, SP, 75p, 1996. 24

FELIZARDO, L. F.; RAMOS, A. C. B.; CAMINO, F. A. C. M. *Integration of ANN and UAV for Aerial Inspection*. 2015. International Micro Air Vehicles Conference and Flight Competition (IMAV). 2015. 30

FILHO, P. C. de O.; AGOSTINHO, F. *Obtenção de fotografias aéreas de pequeno formato e videografia por meio de aeromodelo artesanal adaptado*. 2011. 151-157 p. <https://periodicos.pucpr.br/index.php/cienciaanimal/article/view/11762/11099>, acessado em 11/05/2018. 24

GARCIA, S. et al. *Indoor SLAM for Micro Aerial Vehicles Control Using Monocular Camera and Sensor Fusion*. 2016. International Conference on Autonomous Robot Systems and Competitions (ICARSC 2016), 2016. 25

GAZEBOSIM.ORG. *sitio do software Gazebo*. 2018. <http://gazebosim.org/>,acessado em 09/12/2018. 56

GLOBOG1. *Com uso de drone, polícia descobre e destrói plantação com 1,5 tonelada de maconha na Bahia*. 2018. <https://g1.globo.com/ba/bahia/noticia/com-uso-de-drone-policia-descobre-e-destroi-plantacao-com-15-tonelada-de-maconha-na-\bahia.ghtml>, acessado em 11/05/2018. 24

GONZALEZ, R.; WOODS, E. *Processamento de Imagens Digitais*. 2008. São Paulo, Brasil, 2008. 33, 34, 35, 37

HERMíNIO, R. P. S. et al. *Visão Computacional aplicado a um Protótipo Elétrico*. 2010. Mostra Nacional de Robótica - Instituto Federal de Educação, Ciência e Tecnologia da Bahia, Vitória da da Conquista, BA, 2010. 29

HRABAR, S.; MERZ, T.; FROUSHEGER, D. *Development of an autonomous helicopter for aerial powerline inspections*. 2010. 1st International Conference on Applied Robotics for the Power Industry (CARPI 2010). 1–6, 2010. 30

JORGE, L. de C. *Determinação da cobertura de solo em fotografias aéreas do Projeto Arara*. 2001. Dissertação (Mestrado em Ciências da Computação) - Universidade de São Paulo, São Carlos, 2001. 21

KARAN, E. et al. *A comprehensive matrix of unmanned aerial systems requirements for potential applications within a department of transportation*. 2014. Construction Research Congress, p. 964-973, 2014. 24

LIMA, L. *Processando Imagens em Grayscale e Negativo em C*. 2010. Disponível em <https://lazarolima.wordpress.com/2010/08/19/processando-imagens-em-grayscale-e-negativo-em-c/>,2010, acessado em 14 de maio de 2018. 49

LOPES, S. *Os 10 Melhores Drones com Maior Tempo de Voo de 2017*. 2018. Disponivel em <https://filmora.wondershare.com/pt-br/drones/drones-with-longest-flight-time.html>, 2018, acessado em 05 de setembro de 2018. 75

MARQUES, F. O.; VIEIRA, N. H. *Processamento Digital de Imagens*. 1999. Rio de Janeiro, Brasport, p.25, 1999. 31, 34

MARTINS, W. M. *Video da Simulação com Múltiplos Obstáculos*. 2017. Disponível em <https://youtu.be/kd-RwH0f7uc>, acessado em 13 de março de 2017. 66

MARTINS, W. M. *Video da Simulação com um Obstáculo*. 2017. Disponível em <https://youtu.be/hNciCGZWBho>, acessado em 13 de março de 2017. 66

MARTINS, W. M. *Video de Experimento em Ambiente Real, efeito ARP deslocando sempre para a direita*. 2017. Disponível em <https://youtu.be/RgivmMHnaiY>, acessado em 25 de maio de 2017. 66

MARTINS, W. M. *Video de Experimento em Ambiente Real, efeito ARP parada*. 2017. Disponível em <https://youtu.be/tWa12tqA_ds>, acessado em 25 de maio de 2017. 66

MARTINS, W. M. et al. *Computer Vision in Remotely Piloted Aircraft (RPA) to Avoid Obstacles During Flight*. 2017. IEEE 18th International Conference on Advanced Robotics (ICAR 2017), Hong Kong, China, p. 1-6, 2017. 67

MARTINS, W. M.; MORA-CAMINO, F.; RAMOS, A. C. B. *Computer Vision Based Algorithm for Obstacle Avoidance (Outdoor Flight)*. 2018. IEEE 15thInternational Conference on Information Technology: New Generations (ITNG 2018), Las Vegas, EUA, p. 1-6, 2018. 67

MASCARENHAS, N. D. A.; VELASCO, F. R. D. *Processamento Digital de Imagens*. 1989. IV Escola Brasileiro-Argentina de Informática, Universidad Católica de Santiago del Estero, Termas de Rio Hondo-Argentina, 2a edição, p. 2.4, 1989. 31, 35

MATHWORKS. *MATLAB*. 1994. <https://www.mathworks.com/products/matlab.html>, acessado em 09/12/2018. 56

MEDEIROS, F. A. *Desenvolvimento de um veículo aéreo não tripulado para aplicação em agricultura de precisão*. 2017. Dissertação (Mestrado em Engenharia Agrícola) - Universidade Federal de Santa Maria, Santa Maria, 2007. 21

MELO, R. R. S. de. *Diretrizes de Inspeção de Segurança em Canteiros de Obra Por Meio de Imageamento com Veículo Aéreo Não Tripulado (VANT)*. 2016. Dissertação (Mestrado em Engenharia Civil) - Programa de Pós-graduação em Engenharia Civil, Escola Politécnica, Universidade Federal da Bahia, Salvador, BA, 2016. 24

METNI, N.; HAMEL, T. *A UAV for bridge inspection: Visual servoing control law with orientation limits*. 2006. 534–540 p. <https://pdfs.semanticscholar.org/464d/0a342fa5507e12f69d992a0243fa92c85469.pdf>, acessado em 11/05/2018. Disponível em: <https://doi.org/10.1109/GHTC.2014.6970335>. 24

MILANO, D. D.; HONORATO, L. B. *Visão Computacional*. 2010. Limeira, SP, 2010. 33, 36

MOLRC1. *Hélices: plástico e fibra de carbono*. 2016. Disponivel em <https://www.molrc.com/?p=888>, 2016, acessado em 10 de setembro de 2018. 77

MOLRC2. *Como montar um drone você mesmo: ESCs*. 2016. Disponivel em <https://www.molrc.com/?p=1119>, 2016, acessado em 10 de setembro de 2018. 78

MOTA, R. L. M. et al. *Expanding Small UAV Capabilities with ANN: A Case Study for Urban Areas Inspection*. 2014. British Journal of Applied Science Technology, 4(2), 387-398, 2014. 30

NETO, A. de M. *Navegação de robôs autônomos baseada em monovisão*. 2017. Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecanica, Campinas, SP, 2017. 30

NETO, M. da S. *Agricultura de Precisão com Drones*. 2016. Disponivel em <http://blog.droneng.com.br/agricultura-de-precisao-com-drones/>, 2016, acessado em 05 de setembro de 2018. 30

NISTER, D.; NARODITSKY, O.; BERGEN, J. *Visual Odometry for Ground Vehicle Applications*. 2006. Princeton NJ 08530 USA, 2006. 29

OPENCV. *OpenCV 2.4.13.2 documentation (imgproc module). Image Processing, Canny Edge Detector*. 2004. <http://docs.opencv.org/2.4/doc/tutorials/imgproc\/imgtrans/canny_detector/canny_detector.html?highlight=canny>, acessado em 27 de abril de 2017. 60

OPENCV. *Embarcados, Linux, programação e IoT*. 2017. <https://www.dobitaobyte.com.br/acelerometro-giroscopio-bussola-altimetro-barometro-imu/>, acessado em 27 de abril de 2017. 81

OPENCV.ORG. *Site OpenCV*. 2018. <http://www.opencv.org>, acessado em 27 de abril de 2017. 38, 56

PARROT. *Site Parrot*. 2018. <https://www.parrot.com/global/drones/parrot-bebop-2-fpv>, acessado em 27 de abril de 2017. 86

PURI, A. *A Survey of Unmanned Aerial Vehicles (UAV) for Traffic Surveillance*. 2005. Department of computer science and engineering, University of South Florida. 2005. 24

RAHMAN, M. F. A. et al. *Implementation of quadcopter as a teaching tool to enhance engineering courses*. 2016. IEEE 8th International Conference on Engineering Education (ICEED 2016), 2016. 24

RAMOS, A. C. B. et al. *Drone Development and Applications: A Research Study at UNI-FEI*. 2017. ISBN: 978-620-2-07716-3, Editora Lameber Academic Publushing, Alemanha, 2017. 67

ROS.ORG. *mavros/OverrideRCIn Message*. 2015. <http://docs.ros.org/hydro/api/mavros/html/msg/OverrideRCIn.html>, acessado em 16/09/2018. 42, 43, 56

ROS.ORG. *cv_bridge Tutorials Using CvBridge To Convert Between ROS Images And OpenCV Images*. 2017. <http://wiki.ros.org/cv_bridge/Tutorials/UsingCvBridgeToConvertBetweenROSImagesAndOpenCVImages>, acessado em 16/09/2018. 47

SAUNDERS, J.; BEARD, R.; MCLAIN, T. *Obstacle Avoidance Using Circular Paths*. 2007. AIAA Guidance, Navigation and Control Conference and Exhibit, 2007. 29

SHIGUEMORI, E. H.; MARTINS, M. P.; MONTEIRO, M. V. T. *Landmarks recognition for autonomous aerial navigation by neural networks and Gabor transform*. 2007. IPAS, v. 6497, n. 12, p. 1–9, 2007. 29

SILVA, W. da. *Navegação Autônoma de VANT em Período Noturno com Imagens Infravermelho Termal*. 2016. Dissertação (Mestrado em Computação Aplicada) - Curso de Pós-graduação em Computação Aplicada, Instituto Nacional de Pesquisas Espaciais - INPE, São José dos Campos, SP, 2016. 24

SREENATH, K.; LEE, T.; KUMAR, V. *Geometric Control and Differential Flatness of a Quadrotor UAV with a Cable-Suspended Load*. 2013. IEEE Conference on Decision and Control (CDC), p 2269–2274, Florence, Itália, '562013. 47

STUBBLEBINE, A. et al. *Laser-Guided Quadrotor Obstacle Avoidance*. 2015. AIAA Infotech @ Aerospace. 2015. 30

THROPE, C. et al. *Vision and navigation for the Carnegie-Mellon Navilab*. 1988. IEEE Trans. Pattem Anal. Mach. Intell., vol.PAM1- 10, No.3. pp.362-373, 1988. 33

TOROK, M. M.; GOLVARPAR-FARD, M.; KOCHERSBERGER, K. B. *Image-Based Automated 3D Crack Detection for Post-disaster Building Assessment*. 2014. Journal of Computing in Civil Engineering, v. 28, n.5, 2014. 29

VIDEOBLOCKS. *Graus de liberdade de um drone*. 2018. <https://www.videoblocks.com/video/drone-yaw-left-right-hjn4anrvipgt0fjz/>,acessado em 01/12/2018. 62

WAGSTER, J. et al. *Obstacle Avoidance System for a Quadrotor UAV*. 2012. Infotech@Aerospace 2012, Infotech@Aerospace Conferences, 2012. 29

WAXMAN, A.; LEMOIGNE, J.; SCINVASAN, B. *A visual navigation system for autonomous land vehicles*. 1987. IEEE J.Robotics Auto., vol.RA-3, No.2, pp124-141, 1987. 33

WEISS, A. et al. *Safe Positively Invariant Sets for Spacecraft Obstacle Avoidance*. 2014. Journal of Guidance, Control, and Dynamics, Vol. 38, No. 4 : pp. 720-732, 2014. 29

YAN, Z. et al. *A Real-Time Reaction Obstacle Avoidance Algorithm for Autonomous Underwater Vehicles in Unknown Environments*. 2018. Sensors 2018, 18, 438; doi:10.3390/s18020438. 30

YAN, Z. P. et al. *Globally Finite-Time Stable Tracking Control of Underactuated UUVs*. 2015. Ocean Eng. 2015, 107, 132–146. 30