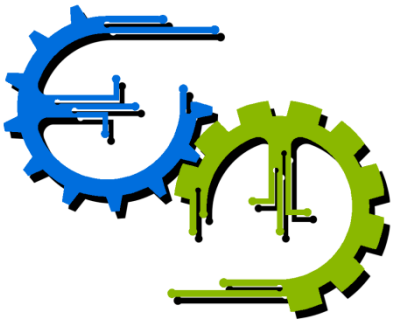
TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO DE CONTROLE DE ROLAGEM E ARFAGEM COM REALIMENTAÇÃO VISUAL APLICADO A UM QUADRIRROTOR COMERCIAL

Por,

Felipe de Paula Lima

Brasília, Abril de 2013



ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO DE CONTROLE DE ROLAGEM E ARFAGEM COM REALIMENTAÇÃO VISUAL APLICADO A UM QUADRIRROTOR COMERCIAL

Por,

Felipe de Paula Lima

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Jones Yudi Mori Alves da Silva, UnB/ENM
(Orientador) _____

Prof. Henrique Cezar Ferreira, UnB/ENE
(Examinador Interno) _____

Prof. Gerardo Antonio Idrobo Pizo, UnB/ FGA
(Examinador Interno) _____

Brasília, Abril de 2013

FICHA CATALOGRÁFICA

FELIPE, LIMA
Implementação de controle de rolagem e arfagem com realimentação visual aplicado a um quadrirrotor comercial,

[Distrito Federal] 2013.

viii, 91p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, Ano). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1.Quadrirrotor 2.Controle
3.Visão 4.Câmera

I. Mecatrônica/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

LIMA, F, (2012). Implementação de controle de rolagem e arfagem com realimentação visual aplicado a um quadrirrotor comercial. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº 14, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 91p.

CESSÃO DE DIREITOS

AUTOR: Felipe de Paula Lima.

TÍTULO DO TRABALHO DE GRADUAÇÃO: Implementação de controle de rolagem e arfagem com realimentação visual aplicado a um quadrirrotor comercial.

GRAU: Engenheiro ANO: 2013

Felipe de Paula Lima
Quadra 17 conjunto C casa 07 – Sobradinho - DF.
73045-173 Brasília – DF – Brasil.

Dedicatórias

Dedico este trabalho à minha família, em especial ao meu pai, Nairon Braz, por ter proporcionado toda a base necessária para manter-me em foco nos estudos e pela herança curiosa desenvolvida desde minha infância.

Felipe de Paula Lima

Agradecimentos

Agradeço ao meu pai pelo amor incondicional, pelo apoio durante toda minha vida acadêmica e principalmente pelo empenho e orgulho demonstrados no decorrer de todos esses anos de estudo.

Agradeço à minha mãe, Rute J. de Paula, pelo amor incondicional, pela presença em todas as reuniões de pais e mestres e principalmente pela dedicação durante todo o período necessário para meu desenvolvimento como pessoa.

Agradeço à minha irmã, Caroline, pelo amor e carinho dedicados durante toda minha vida, e pela presença e apoio nos momentos difíceis.

Agradeço à minha namorada, Thayná, pelo amor, companheirismo, apoio e compreensão em dias difíceis.

Agradeço ao meu tio Valdir, pelo apoio na minha entrada na universidade, ao meu tio Paulo pelo orgulho e empolgação demonstrados a cada vitória e à minha tia Fátima pelo amor demonstrado durante toda minha vida.

Agradeço ao professor Antônio Jacó, pela dedicação em ensinar não só robótica, mas também princípios básicos de convivência em grupo e raciocínio, elementos básicos que me fizeram tomar todas as decisões posteriores.

Agradeço à equipe DROID de competição de robótica inteligente, aos funcionários dos laboratórios de engenharia mecânica e elétrica e aos professores da universidade de Brasília, em especial ao meu orientador, professor Jones Yudi, por todas as experiências adquiridas e pelo grande desenvolvimento acadêmico.

Por ultimo, mas não menos importante, agradeço aos meus colegas de faculdade que contribuíram para minha formação acadêmica e também aos meus amigos que de alguma forma me ajudaram, principalmente ao meu amigo Giordano pela parceria durante todo e qualquer projeto ao longo desse tempo de amizade.

Felipe de Paula Lima

RESUMO

O *Parrot AR.Drone*[1] é um aeromodelo quadrirrotor que pode ser operado remotamente por meio de um dispositivo *handheld* ou um PC comum. O aeromodelo é equipado com diversos sensores: ultrassom; unidade inercial (IMU) com seis graus de liberdade; câmera frontal e câmera vertical. Uma API é fornecida pelo fabricante do aeromodelo, permitindo o livre desenvolvimento de aplicativos de teleoperação. Testes de campo previamente realizados mostraram que em ambientes externos, com a presença de perturbações (vento, por exemplo), a operação do aparelho torna-se uma tarefa complicada. Este projeto propõe o desenvolvimento de um sistema de controle de rolagem e arfagem com realimentação visual, utilizando a câmera como sensor de posição. O aeromodelo envia ao PC remoto as imagens capturadas. O software de controle executa o processamento e envia comandos de correção ao aeromodelo. Desse modo, o sistema possui uma característica de controle em malha fechada.

ABSTRACT

The *Parrot Ar.Drone*[1] is a quadrirrotor model airplane that can be operated remotely via a handheld device or a PC. The model is equipped with several sensors: ultrasound; inertial unit (IMU) with six degrees of freedom; front-facing camera and vertical camera. An API is provided by the manufacturer of model aircraft, allowing the free development of teleoperation applications. Previously conducted field tests have shown that in external environments, in the presence of disturbances (wind, for example), the operation of the appliance becomes a complicated task. This project proposes the development of a system of scroll and pitch control with visual feedback, using the camera as position sensor. The model sends to the remote PC captured images. The control software performs processing and sends correction commands to the model airplane. In this way, the system has a closed loop control feature.

[1] `http://ardrone.parrot.com`

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Este capítulo apresenta considerações gerais preliminares relacionadas à proposta desse projeto, assim como aspectos históricos da área de veículos aéreos não tripulados (VANT's) e controle servo visual.

1.1 ASPECTOS GERAIS

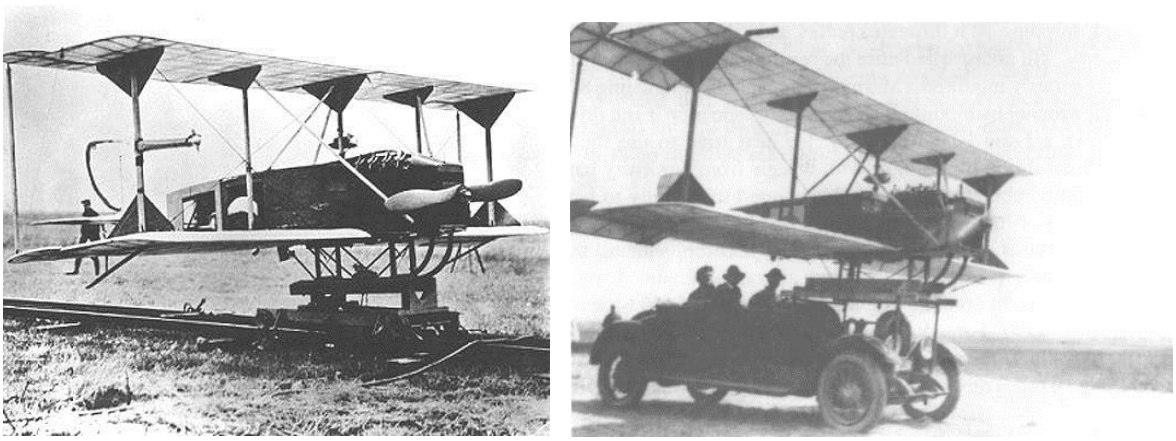
Este relatório tem por objetivo descrever todo o processo envolvido durante a execução deste trabalho de graduação. Começa pela contextualização do que anteriormente já foi feito na área de quadrirrotores e finaliza com os resultados obtidos na resolução do problema.

As áreas de estudo relacionadas ao trabalho serão expostas na profundidade necessária para compreensão requerida. Principalmente, controle servo-visual, que está diretamente relacionado às técnicas de processamento de imagem, visão computacional e teoria de controle. Por se relacionar com conteúdos tão abrangentes, este trabalho fará menções apenas às subáreas que foram utilizadas para o desenvolvimento.

1.2 CONTEXTUALIZAÇÃO HISTÓRICA

Os avanços das tecnologias de sistemas embarcados possibilitaram a implementação de técnicas de controle em dispositivos autônomos. Particularmente no início do século XX, foram criadas as primeiras aeronaves não tripuladas. A precursora de voo foi a aeronave batizada de Curtiss-Sperry "*Flying Bomb*" devido aos seus fins bélicos, projetada para ser como um míssil teleguiado conhecido hoje em dia. Após algumas tentativas falhas, utilizando catapulta, mostrada na Figura 1.1 (a), voou em março de 1918, com um carro como plataforma de lançamento, ilustrado na Figura 1.1 (b) [2].

Um Veículo Aéreo Não Tripulado (*VANT)*, comumente conhecido como *drone*, é uma aeronave sem piloto humano a bordo. Seu voo é controlado autonomamente pelos computadores do próprio veículo, ou pelo comando de um piloto no chão ou em outro veículo. [1]



(a) Com catapulta (b) Com carro

Figura 1.1 - "Flying Bomb". Figura extraída de [2].

O uso dos *VANTs* hoje em dia é bem amplo, são bastante utilizados em áreas como: vigilância de fronteiras, rastreamento e captura de suspeitos [3], agricultura de precisão e pulverização de culturas [4], inspeção de linhas elétricas [5] e combate a incêndios [6].

Os países desenvolvidos vêm dedicando crescentes esforços para integrar os *VANTs* ao espaço aéreo controlado nos últimos anos, visando sua utilização para fins civis. [7]

É importante observar que não só *VANTs* com a arquitetura de aviões de asas fixas podem ter diversos usos.

O primeiro helicóptero experimental bem sucedido foi uma máquina de quatro rotores que era alimentada por um motor rotativo de 180 cavalos de potência. O exército dos Estados Unidos patrocinou o projeto. Realizado por George de Bothezat, um engenheiro de origem russa que emigrou para os EUA, em Dayton em 1922. [8]

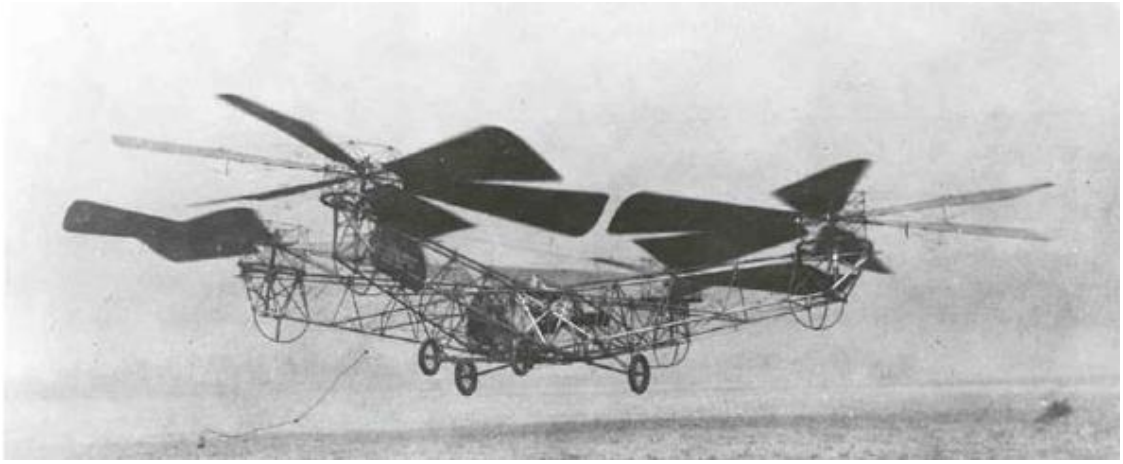


Figura 1.2 - De Bothezat helicóptero.[1] Autor: National Park Service

O desempenho ruim dos primeiros protótipos e a dificuldade para estabilizá-los manualmente impossibilitou o desenvolvimento desses helicópteros como veículos aéreos de transporte. [11]

Com todo o avanço tecnológico nas áreas de eletrônica e computação dos dias atuais, tornou-se possível controlar um helicóptero de quatro motores de forma totalmente autônoma, nascendo assim os *VANTs* quadrirrotores.

O conjunto de quatro rotores permite ao quadrirrotor uma mobilidade enorme, sendo este capaz de mover-se em todas as direções e também girar em torno do seu próprio eixo. [10]

Em contraste, uma desvantagem facilmente identificável dos helicópteros em relação aos aviões de asas fixas é o maior consumo de energia, visto que os aviões possuem o "efeito asa" que fornece sustentação, permitindo que o avião plane sem gastar energia. Helicópteros em geral não podem planar. Porém, existem grandes vantagens como: capacidade de ficarem parados em relação a um ponto fixo em terra; podem também alçar voo sem necessidade de velocidade inicial. Essas entre outras características de voo dos helicópteros, sejam eles de um ou mais rotores, influenciaram

[1] http://commons.wikimedia.org/wiki/File:De_Bothezat_Quadrotor.jpg

diretamente no desenvolvimento dos *VANTs quadricópteros.* No Brasil [9] e no mundo [3] esse tipo de aeronave vem sendo utilizada durante shows, monitoramento de retenção de tráfego e acompanhamento de passeatas e protestos.

1.3 DRONES NA ATUALIDADE

A polícia do Reino Unido já utiliza *VANTs* para auxiliar trabalhos em campo. Usando a tecnologia de imagem térmica e câmera on-board, o operador é capaz de detectar o suspeito por meio de seu calor corporal e patrulhas em terra vão direto para sua localização [3]. Como ilustrado na Figura 1.3.

Figura 1.3 - Drone utilizado pela polícia do Reino Unido. Foto extraída de [3].

No Brasil, um *VANT* desenvolvido no Instituto Tecnológico de Aeronáutica (ITA), está sendo avaliado pela Polícia Militar de Pernambuco para ser utilizado como dispositivo de segurança para eventos de grande porte, como a Copa do Mundo de 2014.

Dotado de duas câmeras que possibilitam um aumento de até 10 vezes, a aeronave desenvolvida no ITA é capaz de capturar imagens de placas de carro, rostos ou detalhes

de qualquer pessoa se estiver voando a uma altitude média de 30 metros [9]. Tornando-se assim uma ótima ferramenta a favor da polícia.

Figura 1.4 - Projeto desenvolvido por alunos do ITA. Foto extraída de [9].

Em tais aplicações os *VANTs* utilizados são menores que as aeronaves comuns, por esse motivo, são conhecidos como veículos aéreos em miniatura (MAV), o que facilita a compra e a produção dos mesmos. [10]

1.4 TÉCNICAS DE CONTROLE

Uma das essenciais tarefas de um sistema autônomo de qualquer tipo é coletar dados a respeito do ambiente ao seu redor. Isto é realizado obtendo medidas através de diversos sensores, e só então interpretar esses dados e extrair informações úteis. [12]

O objetivo de um sistema autônomo é realizar uma tarefa específica de forma controlada. Para realizar esse controle, a fim de manter o sistema sempre funcionando em torno de um padrão, é necessário obter a informação do erro ou desvio da variável medida em relação ao padrão imposto pelo sistema. Após a obtenção da informação de erro, é necessário corrigi-lo, é aí que entram as técnicas de controle.

Vários sensores podem ser utilizados para obtenção de dados a respeito do sistema. A escolha do sensor adequado vai depender das variáveis a serem medidas e das características especificas de cada sensor, bem como, da necessidade ou não de conversores e/ou condicionadores de sinais.

Um dos sensores com a maior capacidade de extrair informações de grande utilidade do ambiente é a câmera, observando a fisiologia humana concluímos que a visão junto com a nossa capacidade de interpretação, proporcionam uma imensa quantidade de dados do entorno permitindo-nos interatuar com ele. [13]

Os primeiros trabalhos que fizeram uso de sistemas de visão para o controle de robôs datam da década de 1960. Era usada uma abordagem em malha aberta, em que o movimento e o controle eram feitos em instantes diferentes, denominada "Ver depois Mover" "*Look-then-Move*" [14]. Isso comprometia a precisão do sistema de controle, pois se acumulavam erros advindos do posicionamento do robô, do sistema de visão, e das transformações entre eixos coordenados. [15]

O termo controle servo-visual foi proposto formalmente em 1973 por Hill e Park [16]. Eles expuseram as vantagens de inserir o erro visual à malha de controle. Porém, a área de controle servo-visual pouco avançou até meados da década de 90, devido principalmente às dificuldades de processamento de imagens em tempo hábil para sua utilização na malha de controle [15]. Hoje em dia isso ainda é uma dificuldade, porém, algo completamente contornável na maioria dos casos, devido à grande evolução tecnológica dos computadores e sistemas embarcados. A maior dificuldade continua sendo a interpretação correta das imagens obtidas. Para isso são utilizadas e desenvolvidas várias ferramentas que auxiliam no processamento das imagens, com o objetivo de obter informações o mais precisas e confiáveis possíveis.

1.5 DEFINIÇÃO DO PROBLEMA

Testes de campo previamente realizados mostraram que em ambientes externos, com a presença de perturbações (vento, por exemplo), a operação do aparelho torna-se uma tarefa complicada, devido ao sistema de controle embarcado no aparelho não ser tão bem elaborado. O drone utilizado no projeto apresenta instabilidade até mesmo em ambientes internos, onde existem poucas interferências diretas ao voo.

Sabendo da subutilização da câmera na maioria dos drones, na maioria das vezes usadas apenas como ferramentas de interação amigável com o usuário, foi proposto então o desenvolvimento de uma ferramenta que aumenta a confiabilidade do controle de rolagem e arfagem do modelo.

1.6 OBJETIVOS

Este projeto propõe o desenvolvimento de um sistema de controle de estabilidade com realimentação visual, utilizando as imagens da câmera como sensor de posição. O aeromodelo envia ao PC remoto as imagens capturadas. O software de controle executa o processamento e envia comandos de correção ao aeromodelo. Desse modo, o sistema possui uma característica de controle em malha fechada, sendo que o intuito é o projeto de um controlador clássico adequado.

O objetivo inicial foi definido como sendo quatro principais atividades:

- Obtenção do modelo dinâmico do sistema;
- Coleta de imagens das câmeras embutida;
- Envio de comandos para o quadrirrotor via wi-fi com o auxilio da API;
- Desenvolvimento de testes.

O objetivo final foi definido como sendo o desenvolvimento e testes do controlador de estabilidade, utilizando o modelo dinâmico, as imagens da câmera e o envio de comandos para o drone obtidos na etapa anterior.

1.7 APRESENTAÇÃO DO MANUSCRITO

O capítulo 2 descreve a plataforma utilizada para o desenvolvimento do trabalho, ou seja, o quadrirrotor, bem como, seus componentes e estrutura de programação em geral, além dos softwares utilizados para realizar calibração e simulação.

O capítulo 3 explicita toda a parte de visão computacional utilizada, o que inclui: câmera; calibração e algoritmos de estimação de movimento.

Toda a modelagem do sistema, tanto teórica quanto experimental, bem como, o projeto dos controladores, são tratados no capítulo 4.

A validação, testes realizados e resultados são descritos no capítulo 5.

Por fim, a conclusão e possíveis trabalhos futuros estão detalhados no capítulo 6, seguido das referências bibliográficas e anexos.

2 PLATAFORMA

Este capítulo busca descrever todas as ferramentas que serão utilizadas para colocar em prática os objetivos deste trabalho. Sendo dividido em três principais tópicos: descrição do quadrirrotor; descrição da biblioteca OpenCV e detalhamento da estrutura geral do sistema.

2.1 DESCRIÇÃO DO QUADRIRROTOR

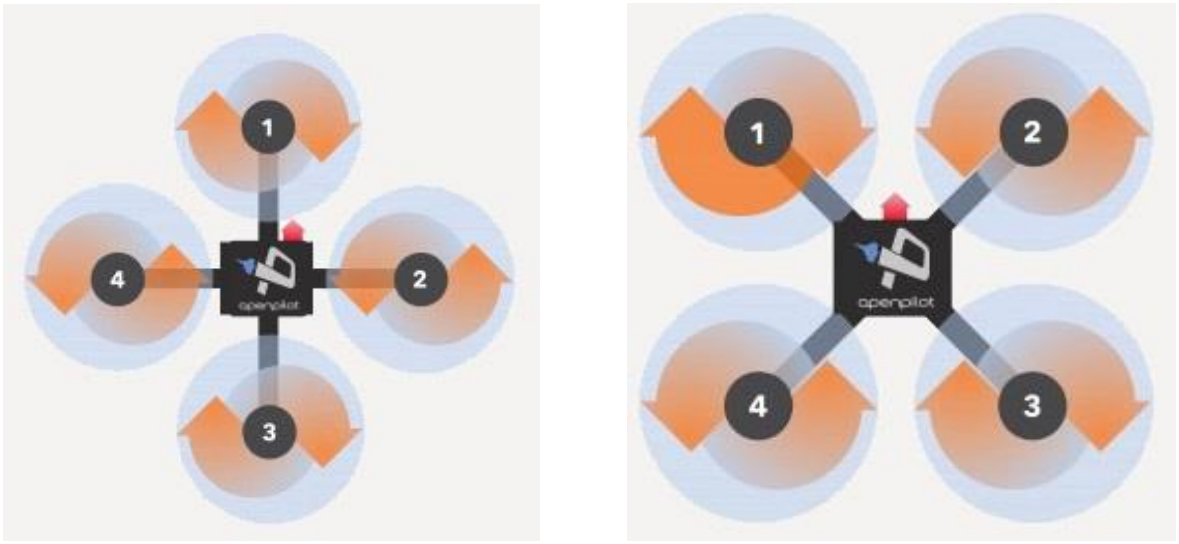
Para colocar em prática os objetivos desse trabalho, utilizaremos um quadrirrotor comercial bem conhecido no meio acadêmico devido à sua acessibilidade, equipamentos sensoriais e uma biblioteca aberta para se realizar a comunicação [10]. O *Parrot AR.Drone*[1] já foi utilizado, por exemplo, para experimentos e pesquisas em navegação autônoma [17], estabilização usando aprendizagem de máquina [18], interação homem-máquina [19] e também em assistência e treinamento de atletas [20]. O drone utilizado está ilustrado na Figura 2.1.

Figura 2.1 - Quadrirrotor *Parrot AR.Drone* [1].

[1] `http://ardrone.parrot.com`

2.1.1 CARACTERÍSTICAS DE FUNCIONAMENTO

A estrutura mecânica do *AR.Drone* é composta por quatro rotores anexados às quatro extremidades de um cruzamento a que estão acoplados a bateria e o hardware RF. Cada par de rotores opostos gira da mesma forma. Um par gira no sentido horário e o outro no sentido anti-horário [21], como ilustrado nas Figuras 2.2 (a) e (b).



(a) + **(b)** X

Figura 2.2 – Movimentação dos rotores. Fonte: *OpenPilot Foundation* [2]

Quadrirrotores podem ter basicamente duas configurações simétricas, em forma de + ou em forma de X, como ilustrado na Figuras 2.2 (a) e (b) respectivamente. A maioria dos quadrirrotores na atualidade utiliza a configuração em X. A movimentação desses drones ocorre de acordo com a variação da rotação dos motores, como mostrado de forma bem detalhada na Figura 2.3.

[2] `http://wiki.openpilot.org/display/Doc/Multirotor+Mixer+Settings`

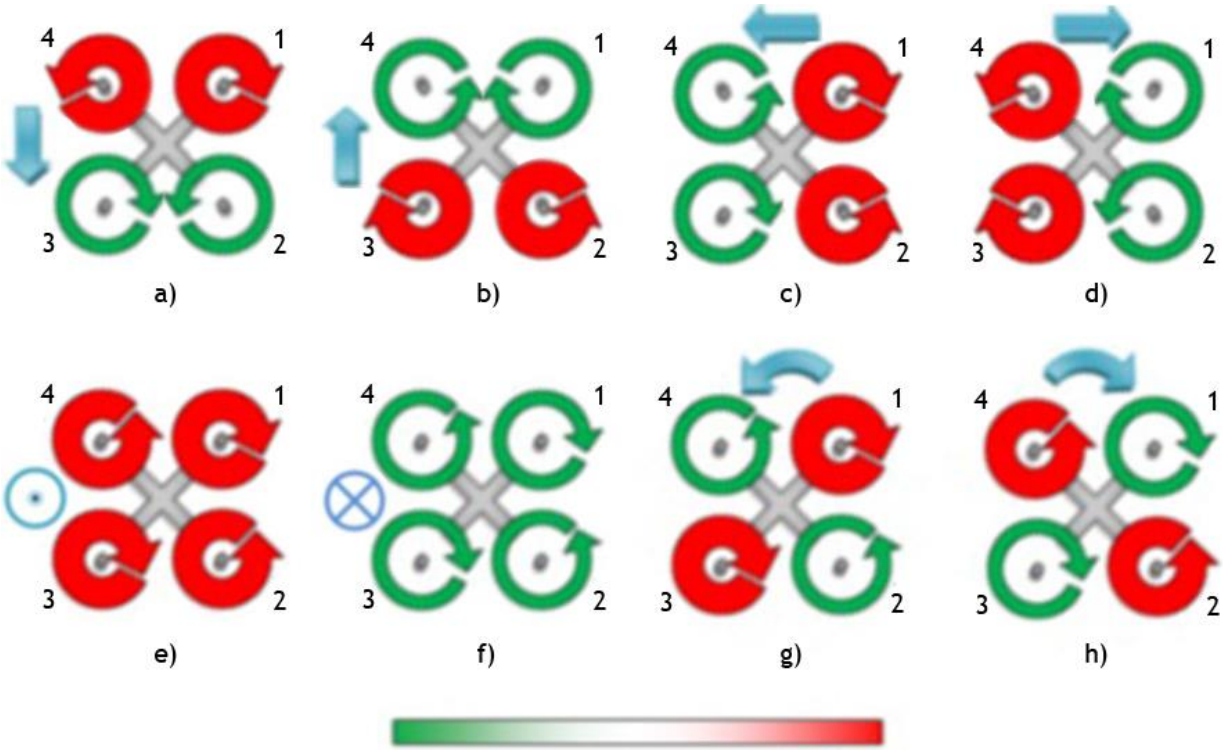


Figura 2.3 - Movimentação do drone na configuração X. Figura editada de [10].

As cores na legenda ilustram a velocidade de rotação dos rotores, variando de verde a vermelho, sendo verde baixa rotação e vermelho, alta.

No caso dos movimentos de translação, em x e y, ilustrados nas Figuras 2.3 (a), (b), (c) e (d), para movimentar o quadrirrotor em uma direção qualquer, basta manter os rotores dianteiros em relação ao sentido do movimento em uma rotação constante e aumentar a rotação dos motores traseiros. O movimento na direção y (esquerda e direita) é obtido através da variação do ângulo φ (phi), sendo chamado de *roll.* Já o movimento na direção x (frente e trás) é obtido através da variação do ângulo θ (theta), e é chamado de *pitch.*

Para os movimentos de ascensão e descida, eixo z, ilustrados nas Figuras 2.3 (e) e (f), é preciso aumentar ou diminuir a rotação dos quatro motores igualmente. Esse movimento é chamado *throttle.*

Quando o objetivo é um movimento de rotação em torno do eixo z, como mostrado nas Figuras 2.3 (g) e (h), basta aumentar a rotação de dois rotores opostos e diminuir a rotação dos outros dois. Esse movimento varia a velocidade angular, representada pela letra grega Ψ' (psi) e é chamado de *yaw*.

Nesse trabalho, como o drone será comandado a partir de uma API (*Application programming interface*) em alto nível, o tipo de configuração passa a não ter importância. Esse assunto será abordado mais detalhadamente na sessão 2.4 desse capítulo.

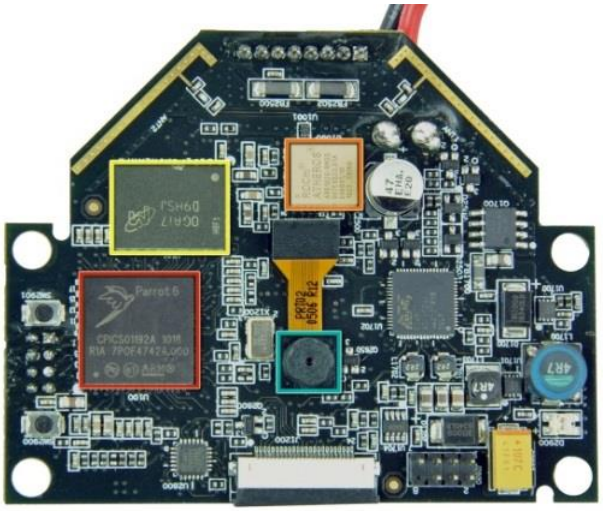
2.1.2 COMPONENTES

As informações abaixo listadas foram extraídas de [10] e [21].

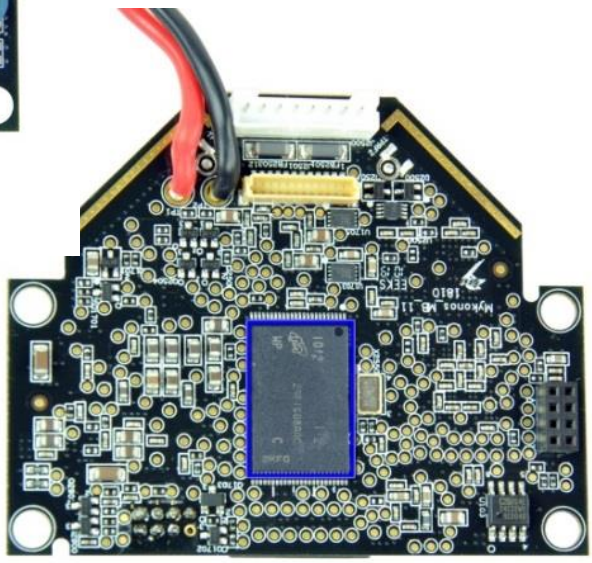
O *Parrot AR.Drone*, quadrirrotor utilizado neste trabalho, é dotado das seguintes tecnologias:

1. Sensores:
    - Câmera frontal (QVGA – 15fps) e vertical (QCIF – 60fps).
    - Três acelerômetros *MEMS* (*Micro-Electro-Mechanical Systems*).
    - Altímetro ultrassônico.
    - Girômetro *MEMS* de dois eixos.
    - Girômetro *MEMS* de precisão de um eixo.
2. Motores trifásicos sem escova (*Brushless*).
3. CPU licenciado da ARM com Linux embarcado.
4. Comunicação digital via Wi-Fi.

Ambas as câmeras usam sensores CMOS. A câmera vertical, indicada na Figura 2.4 (a) por um quadrado verde acoplado à placa mãe do quadrirrotor, é uma câmera de alta velocidade, com 60 fps, resolução QCIF (176 × 144 pixels) e abertura de 64°. Já a câmera frontal, mostrada na Figura 2.5, é uma câmera com resolução QVGA (320 × 240 pixels), 15 fps e abertura de 90°.



(a) Parte da frente

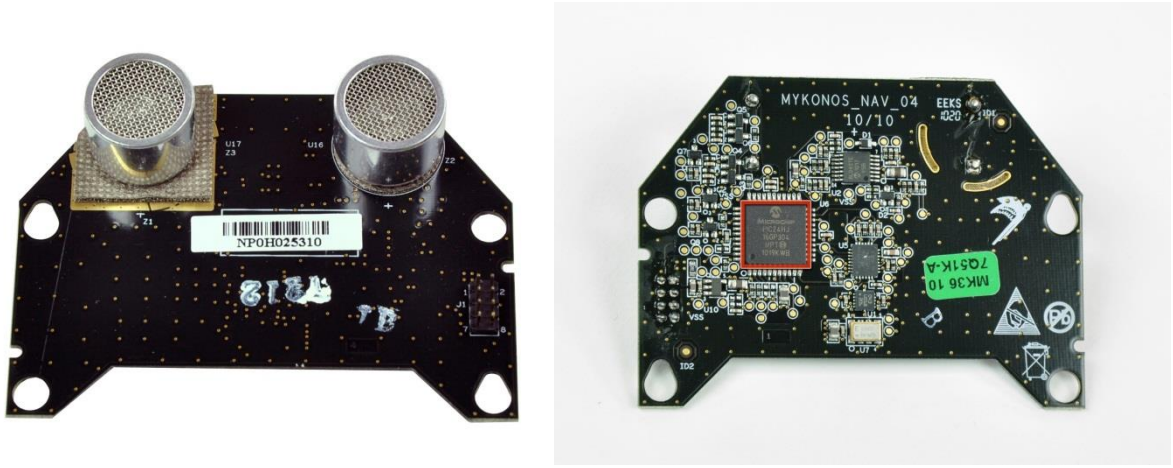


(b) Parte de trás

Figura 2.4 - Placa mãe do quadrirrotor. Figura extraída de [10].

Figura 2.5 - Câmera frontal do quadrirrotor. Foto extraída de [10].

A placa mãe do quadrirrotor é onde se encontra o *CPU* Parrot P6 ARM926, pode ser vista na Figura 2.4. Podem ser vistos também nessa figura o circuito integrado ROCm Atheros AR6102G-BM2D, que permite a comunicação Wi-Fi nos padrões 802.11b e 802.11g, o circuito integrado Micron 29F1G08AAC, a memória Flash de 128MB, o circuito integrado Micron OGA17 D9HSJ e a memória DDR SDRAM de 128MB.



(a) Altímetro ultrassônico. (b) Microcontrolador PIC24HJ.

Figura 2.6 - Placa de navegação do quadrirrotor. Figura extraída de [10].

A placa de navegação do quadrirrotor está ilustrada na Figura 2.6. Mostrados nessa figura estão: o altímetro ultrassônico, composto por um emissor e um receptor que trabalham na frequência de 40khz, o micro controlador PIC24HJ, que processa os

dados dos sensores, o circuito integrado Invensense IDG[3], que é o girômetro de dois eixos e o circuito integrado XV-3500CB [4], que é o girômetro de um eixo.

Os quatro motores responsáveis pelo voo são motores trifásicos sem escovas, devido à alta confiabilidade, capacidade de operar em altíssimas rotações e vida útil mais longa devido à ausência de desgaste das escovas. As hélices foram projetadas pela empresa francesa Novadem[5], com o objetivo de tornar mais eficiente o consumo de energia e o impulso do quadrirrotor. Esses componentes podem ser vistos na Figura 2.7.

Figura 2.7 - Conjunto motor, hélice e circuito controlador. Figura extraída de [10].

É importante ressaltar que o *AR.Drone* utiliza uma bateria LiPo de 1000mAh e 11.1V de tensão para voar, que corresponde a uma autonomia de aproximadamente doze minutos de voo. Durante o voo a tensão da bateria diminui da carga total (11.1 Volts) para baixa carga (9 Volts). O *AR.Drone* monitora a tensão da bateria e

[3] http://www.invensense.com/

[4] http://www.epsontoyocom.co.jp/english/gyroportal/product_xv3500cb.html

[5] http://www.novadem.com/

converte em uma porcentagem da carga total (100% se a bateria está cheia, 0% se a bateria estiver descarregada). Quando o robô detecta uma tensão baixa da bateria, primeiro ele envia uma mensagem de aviso para o usuário, então, automaticamente pousa. Se a tensão atinge um nível crítico, todo o sistema é desligado para evitar qualquer comportamento inesperado.

2.1.3 COMUNICAÇÃO

O *AR.Drone* pode ser controlado a partir de qualquer dispositivo que suporte o modo *ad-hoc Wi-fi*. O processo a seguir é seguido automaticamente [21]:

1. O *AR.Drone* cria uma rede *Wi-Fi* com um *ESSID* normalmente chamado adrone_xxx, onde xxx é um identificador de seis números de cada ardrone, e auto aloca um endereço de *IP*.
2. O usuário conecta o dispositivo cliente nesta rede *ESSID*.
3. O cliente requisita um endereço *IP* do servidor *DHCP* do drone.
4. O servidor *DHCP* concede ao cliente um endereço *IP,* que é o próprio endereço *IP* do drone, acrescido de 1.
5. O cliente pode iniciar então a rede *ad-hoc Wi-fi*.

Depois de estabelecida a comunicação, é possível ao cliente controlar e configurar o quadrirrotor mandando *comandos AT* na porta *UDP 5556*. A API fornecida pelo fabricante envia esses *comandos AT* através de uma função específica. O drone, por sua vez, envia pacotes denominados *navdata* na porta *UDP 5554*, contendo informações sobre o estado atual do quadrirrotor, sobre a porcentagem de bateria restante e sobre a atitude, medidos pelos sensores.

Vale ressaltar que apesar da capacidade de controle direto dos ângulos de Euler associados ao quadrirrotor e da velocidade vertical do mesmo [10] através do programa

alto nível utilizado, não podemos ter o controle diretamente de cada motor. Isso foi uma medida de segurança adotada pelo próprio fabricante. É apresentada na Tabela 2.1 uma lista dos principais *comandos AT*.

Tabela 2.1 - Principais *comandos AT*. Tabela editada de [10].

Comando AT	Descrição
`AT*REF`	Decola, pousa ou parada de emergência
`AT*PCMD`	Controla os movimentos do quadrirrotor (roll, pitch, gaz, e yaw)
`AT*FTRIM`	Ajusta a referência horizontal
`AT*ANIM`	Move o quadrirrotor num movimento pré-definido
`AT*COMWDG`	Informa se a conexão está ativa

Será utilizado internamente, em especial o comando `AT*PCMD`, que além de selecionar outros modos de voo do controlador do quadrirrotor, envia as entradas de referência para o controlador [10], ou seja, é responsável pela movimentação do drone.

2.1.4 ESTRUTURA DE PROGRAMAÇÃO

Todo o desenvolvimento de softwares para atuar juntamente com o *AR.Drone* começa a partir do *SDK* oficial, que fornece bibliotecas para a comunicação entre o quadrirrotor e o cliente, o qual se conecta ao servidor do mesmo [10]. Mais especificamente, foi utilizada a versão 2.0 do *SDK*. Composto pela biblioteca *ARDroneLib*, e alguns exemplos de clientes.

A Figura 2.8 ilustra através de uma árvore a estrutura da biblioteca *ARDroneLib*.

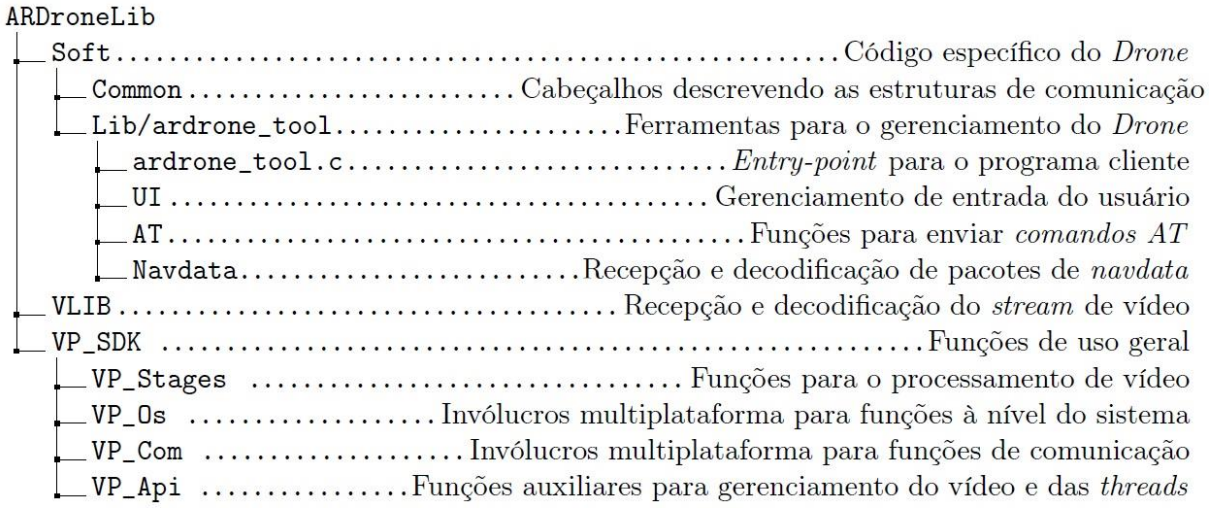


Figura 2.8 - Estrutura da biblioteca. Figura extraída de [10].

A biblioteca de comunicação *ARDroneLib,* bem como os *comandos AT,* não serão mais detalhados nesse trabalho, devido ao seu nível inferior nessa aplicação específica. Isso só ocorre, pois um programa de mais alto nível no conceito do projeto foi implementado, a fim de facilitar a comunicação direta com o drone. Este programa é chamado *Autopylot*. Seu funcionamento será mais bem detalhado na sessão 2.3.1.

2.2 DESCRIÇÃO DA BIBLIOTECA OPENCV

A *OpenCV* é uma biblioteca *open source* de ferramentas e algoritmos para processamento de imagens desenvolvida inicialmente pela *Intel*. Esta biblioteca é composta por algoritmos otimizados, de tal forma que utiliza os recursos do sistema para atingir o menor custo computacional [22]. Como no caso deste trabalho é utilizado o conceito de tempo real, as ferramentas dessa biblioteca se tornam muito úteis. Essa biblioteca atualmente possui mais de 500 funções de processamento com aplicações dentre as quais se destacam: operações entre imagens, filtros, transformações

morfológicas, calibração de câmeras, *tracking*, estimação de pose, reconhecimento e identificação de faces, gestos e objetos [23].

Escrita em C, C++ e Python para os sistemas operacionais Linux, Windows e Mac OS X, foi escolhida a *OpenCV* como ferramenta principal para o desenvolvimento dos algoritmos de visão do sistema. As bibliotecas adotadas foram em C, devido a maior familiaridade com a linguagem.

2.3 PROGRAMAS AUXILIARES

No decorrer do trabalho, fez-se necessário o uso de programas adicionais, além do *SDK* do *AR.Drone* e da biblioteca *OpenCV.* O *AutoPylot* foi utilizado impreterivelmente no desenvolvimento deste trabalho. Já as ferramentas Camera *Calibrarion ToolBox for MATLAB®*[7] e *Simulink®* foram utilizadas para fins de atestar resultados, bem como modelar o sistema de forma experimental.

2.3.1 *AutoPylot*[6]

O software *AutoPylot*, foi desenvolvido pelo professor Simon D. Levy, com o intuito de facilitar ao programador o desenvolvimento de um piloto automático para o quadrirrotor, ou simplesmente para interação direta e fácil com o drone. O piloto automático é ativado quando a variável global e externa, `g_autopilot`, assume o valor `TRUE`. Isso ocorre no código quando é pressionado um botão do *joystick*, configurado no arquivo `gamepad.c`, ou quando é atribuído o valor `TRUE` a ela diretamente no programa.

[6] `http://home.wlu.edu/~levys/software/ardrone_autopylot/.`

[7] `http://www.mathworks.com/products/matlab/`

Assim que o piloto automático é ativado, uma função especifica é chamada toda vez que o cliente recebe um pacote do quadrirrotor, contendo o quadro de vídeo disponível, bem como, informações dos sensores, etc. Essa função pode ser rodada em alto nível como um script de *Python* [8], MATLAB®, ou na linguagem C.

No caso do Python, é a função `agent` do arquivo `autopylot_agent.py`. Em MATLAB, é a função `autopylot_action` do arquivo `autopylot_agent.m`. Já no caso do C, é a função `agent_comm_act` do arquivo `autopylot_c_agent.c`. Além dessa função, neste arquivo também contém as funções `agent_comm_init` e `agent_comm_close`, chamadas no início e no término do piloto automático, respectivamente. [10]

Os parâmetros de entrada das funções são semelhantes para as três linguagens: Existe um vetor de elementos do tipo *unsigned char*, representando os *bytes* do *stream* de vídeo, a largura e a altura do quadro de vídeo sendo recebido, um valor booleano que é `TRUE` se o quadro é da câmera vertical e `FALSE`, caso seja da câmera frontal, e os valores do *navdata* correspondente ao estado do quadrirrotor, como: porcentagem da bateria restante, ângulos de Euler, altitude, velocidades lineares estimadas nos três eixos e a defasagem angular em relação ao norte terrestre. [10]

A função retorna ao quadrirrotor valores para habilitar ou desabilitar o piloto automático, efetuar a troca entre a câmera vertical e frontal e também os valores de ângulo e velocidade, modificados, para servirem como parâmetros do comando `AT*PCMD` que atua diretamente no drone.

2.3.2 CAMERA CALIBRATION TOOLBOX FOR MATLAB® [9]

Como o nome já diz, essa é uma ferramenta do MATLAB capaz de aplicar vários exemplos de calibração de câmeras, desde algoritmos mais simples para uma única

[8] `http://www.python.org/`
[9] `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html#examples`

câmera, até implementações em sistemas de visão estéreo.

Desenvolvida pelo professor Jean-Yves Bouguet, do Instituto de Tecnologia da Califórnia, para auxiliar no problema de calibração de câmera, isto é, a partir de imagens obtidas com a câmera, é possível especificar os parâmetros intrínsecos e extrínsecos da mesma. [10]

O conceito de calibração de câmeras, bem como sua necessidade de implementação e aplicação neste trabalho, serão detalhados no próximo capítulo.

A biblioteca *OpenCV* possui funções capazes de realizar essa tarefa, porém, por motivos de praticidade, foi escolhido realizar a calibração da câmera através desta *toolbox*.

2.3.3 *Simulink* ® [10]

Simulink ® é um ambiente de diagrama de bloco para simulação multi-domínios e projeto baseado em modelo. Ele oferece suporte a projeto em nível de sistema, simulação, geração automática de código, e teste contínuo de verificação de sistemas embarcados. O *Simulink* fornece um editor gráfico, bibliotecas de bloco personalizável para modelagem e simulação de sistemas dinâmicos. É integrado com o MATLAB ®, permitindo-lhe incorporar algoritmos do MATLAB em modelos e exportar resultados de simulação para o MATLAB para análise posterior.

2.4 ESTRUTURA GERAL DO SISTEMA

Como já detalhado anteriormente, as formas de enviar comandos para o quadrirrotor variam de acordo com os níveis de programação do *AR.Drone,* vão desde

[10] `http://www.mathworks.com/products/simulink/`

complexos *comandos AT,* enviados diretamente ao aeromodelo, até uma simples troca de valor em uma variável da função agent_comm_act do arquivo `autopylot_c_agent.c`.

Foi escolhido trabalhar no mais alto nível possível, ou seja, a partir dos programas *AutoPylot,* foi instalado no computador as bibliotecas do *OpenCV* e adicionadas, as que seriam utilizadas, ao programa `autopylot_c_agent.c`. Como já explicitado, os programas *AutoPylot* comunicam-se diretamente com o *SDK* do *AR.Drone.* Com tudo isso associado, foi restringido todo o acesso aos programas envolvidos para comunicar-se com o drone e controlá-lo ao programa `autopylot_c_agent.c`. Tendo acesso à todas as informações necessárias de sensores, bem como, imagens das câmeras e com capacidade de processar imagens e interpreta-las, devido à adição das bibliotecas *OpenCV.*

As entradas do sistema são enviadas como valores em ponto flutuante, normalizados por uma razão de -1 a 1 do máximo ângulo de arfagem, máximo ângulo de rolamento, máxima velocidade angular do ângulo de guinada e a máxima velocidade vertical [10]. As entradas são representadas respectivamente pelos símbolos, θ_{AT} , φ_{AT} , Ψ'_{AT} e $\mathrm{Vz}_{\mathrm{AT}}$.

Serão utilizados os valores padrões de configuração: ângulo máximo de arfagem e rolamento de 14,32°, velocidade vertical máxima de 700mm/s e velocidade de rotação em torno do eixo de guinada de 100°/s. Ou seja, denotando θ_{REF} para o ângulo de arfagem de referência, em graus, φ_{REF} para o ângulo de rolamento de referência, em graus, Ψ'_{REF} para a velocidade angular de referência, em graus por segundo e $\mathrm{Vz}_{\mathrm{REF}}$ para a velocidade vertical de referência, em metros por segundo, logo, tem-se as seguintes relações [10]:

$$\theta_{\mathrm{REF}} = 14{,}32^{\circ}\,\theta_{\mathrm{AT}} \tag{2.1}$$

$$\varphi_{\mathrm{REF}} = 14{,}32^{\circ}\,\varphi_{\mathrm{AT}} \tag{2.2}$$

$$\Psi'_{REF} = 100^{\circ}/\mathrm{s}\,\Psi'_{AT} \tag{2.3}$$

$$\mathrm{Vz_{REF}} = 0{,}7\mathrm{m/s}\ \mathrm{Vz_{AT}} \tag{2.4}$$

Em contrapartida, o quadrirrotor envia pacotes, denominados *navdata,* neles estão contidas informações de status do quadrirrotor como: percentagem de bateria restante, altitude e ângulos medidos pelos sensores internos, e por fim, estimativas de velocidades de translação horizontal.

São ilustrados na Figura 2.9, os sistemas de coordenadas definidos pelos sensores do drone. O sistema de eixos ortogonais B (i, j, k) está fixo ao quadrirrotor e o sistema de referência de eixos ortogonais E (x, y, z) está fixo à terra. O sistema de referência x, y e z é definido com base nos dados dos sensores, através do comando `AT*FTRIM`, utilizado antes da decolagem do quadrirrotor, quando o mesmo se encontra no chão [10]. Esse comando é realizado internamente no sistema.

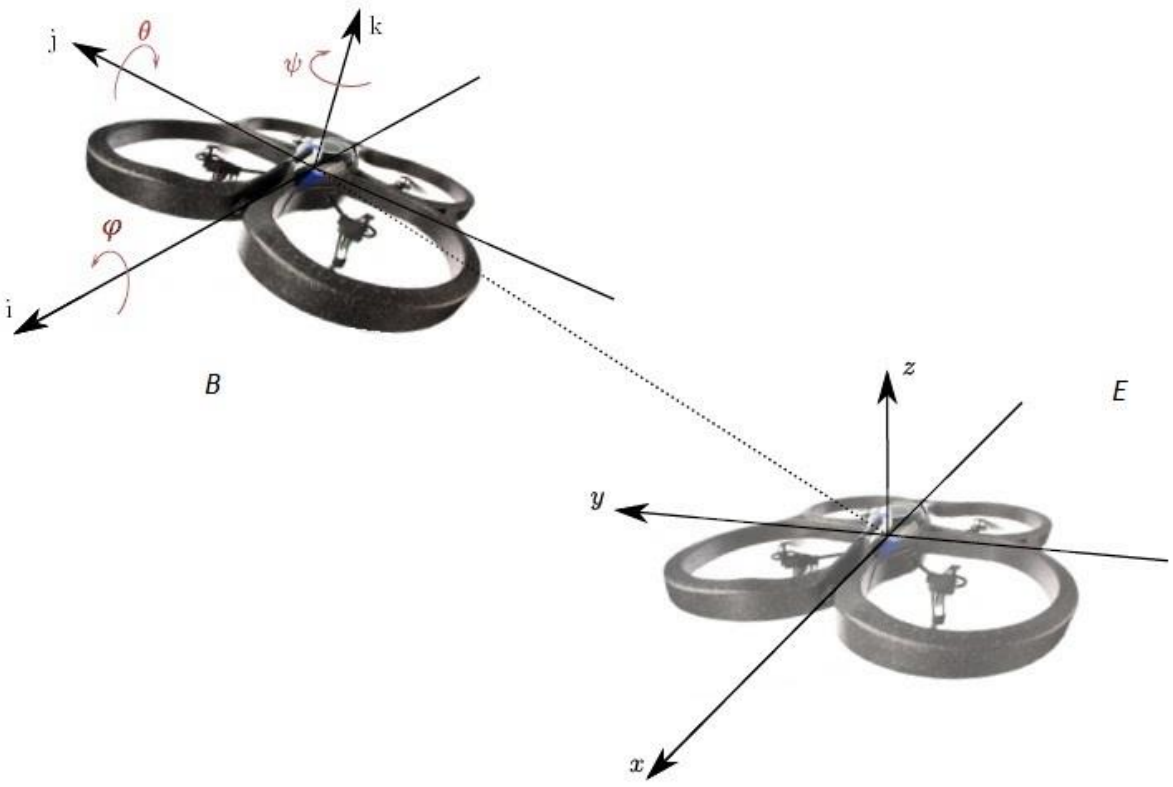


Figura 2.9 - Sistemas de coordenadas. Figura extraída de [10].

Os ângulos de arfagem θ e rolagem φ serão nulos se o plano ij, for paralelo ao plano xy. Já o ângulo de rotação Ψ, será nulo se os planos ik ou jk forem respectivamente paralelos aos planos xz ou yz. Os ângulos θ, φ e Ψ ilustrados na Figura 2.9 serão positivos se girarem nos sentidos indicados pela figura, serão negativos caso contrário. Já as componentes de velocidade linear do drone são definidas em relação aos eixos x e y do sistema de referência, de modo que Vx e Vy serão positivas no sentido positivo de seus respectivos eixos. [10]

A partir dessa estrutura, torna-se possível desenvolver o controlador de voo, objetivo final deste trabalho.

3 VISÃO COMPUTACIONAL

Este capítulo procura expor toda a abordagem na área de processamento de imagem e visão computacional implementada neste trabalho, com o intuito de obter informações adicionais ao sistema de controle do quadrirrotor.

3.1 ASPECTOS GERAIS

A implementação de um sistema de controle em um robô móvel é uma tarefa complexa. Diversas restrições quanto ao tempo de resposta devem ser consideradas para que este seja um sistema factível. Em aplicações com exigência de um curto tempo de resposta, o sistema de controle utilizando visão só será possível através da implementação de algoritmos complexos de processamento de imagem. Isso se deve ao fato de que as características da imagem exigem varias etapas de processamento antes que elas possam ser transformadas em dados uteis para o controlador. Dessa forma o software deve considerar todos esses requisitos em seu desenvolvimento. [24]

3.2 IMAGENS DE CÂMERAS

As imagens de uma câmera podem ser definidas como uma representação visual em um plano bidimensional de cenas do mundo real, ou seja, é uma projeção ou mapeamento de um espaço 3D num plano 2D. Com essa transformação de espaços, são perdidas algumas informações durante o processo, a mais importante delas é a profundidade, assim sendo, dependendo do posicionamento, planos no espaço 3D podem se apresentar como linhas no espaço 2D, bem como linhas podem ser vistas como pontos. [24]

Utilizando conhecimentos da física envolvida na captura e da transformação realizada pelos instrumentos utilizados, pode-se fazer uma reconstrução aproximada do espaço 3D, a partir das imagens, sob várias perspectivas, de uma câmera. [25]

3.2.1 REPRESENTAÇÃO DA IMAGEM

Uma imagem pode ser representada de diversas maneiras, isso dependerá da sua aplicação. As representações mais comuns são: níveis de cinza, *RGB* e *HSV*. Uma imagem em escala de cinza pode ser representada por uma função contínua $I(x,y)$ que indica a intensidade de luz refletida em cada pixel (x,y), da imagem. A discretização da imagem é realizada se a mesma for capturada por uma câmera que possui uma matriz de mxn elementos sensores de largura s_x e altura s_y. O resultado dessa captura é uma imagem amostrada com resolução espacial de mxn pixels, $I(k,j)$ (Figura 3.1) quantizada em intensidade com uma precisão de 2^b, onde b representa o número de *bits* utilizados para armazenar o valor de intensidade. Já as imagens coloridas como: *RGB* e *HSV*; diferenciam-se por possuírem três dimensões para intensidade. [25]

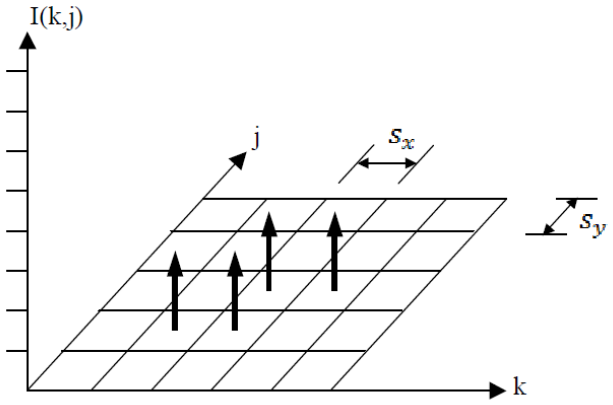


Figura 3.1 - Representação da imagem. Figura extraída de [24].

A representação de imagens coloridas *RGB* utiliza como base o sistema visual do ser humano, que através dos cones e bastonetes, células localizadas na retina do olho, são capazes de detectar três níveis distintos de cores: vermelho, verde e azul *(Red,*

Green e Blue); e através da sua combinação, variando continuamente a intensidade, obtêm-se infinitas tonalidades (Figura 3.2). Esta representação tem como vantagem a sua simplicidade, tornando-a mais intuitiva e entre suas desvantagens está a susceptibilidade a mudanças na iluminação. [24]

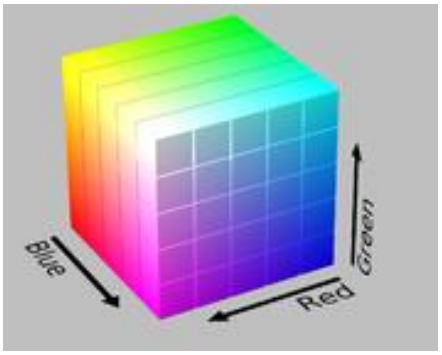


Figura 3.2 - Representação RGB. [11]

Diferente da representação *RGB*, a representação de imagens coloridas *HSV* (Figura 3.3) baseia-se em parâmetros intuitivos de cor e é representada por um subconjunto de um sistema de coordenadas cilíndricas. Esses parâmetros são denominados: *Hue,* H, especifica um ângulo em torno do eixo vertical do cilindro, variando de 0° a 360º; *Saturation,* S, medida ao longo do eixo radial, especifica a pureza relativa (diluição com a cor branca) da cor, varia de 0 a 1; *Value,* V, é a medida ao longo do eixo do cilindro que possui valor 0 para o preto e 1 para o branco, especifica a intensidade de luz na cor. Uma de suas vantagens está na capacidade de identificar cores independentemente de iluminação e entre suas desvantagens está a maior complexidade na representação. [24]

[11] `http://en.wikipedia.org/wiki/HSL_and_HSV`

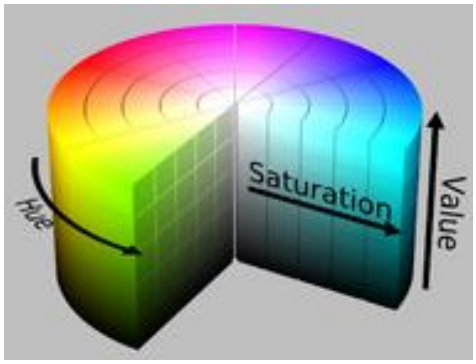


Figura 3.3 - Representação HSV. [11]

3.2.2 MODELO DE CÂMERA

Um modelo que representa bem as características de uma câmera é conhecido como *pinhole* (Figura 3.4). É modelado a partir de uma entrada de luz por um orifício do tamanho de um buraco de agulha onde do lado de dentro está uma placa fotossensível plana ou não, que fica marcada devido à incidência da luz. Esse modelo, já antigo, permite fazer análises mais diretas do comportamento do sistema mesmo para as câmeras atuais e suas lentes. [25]

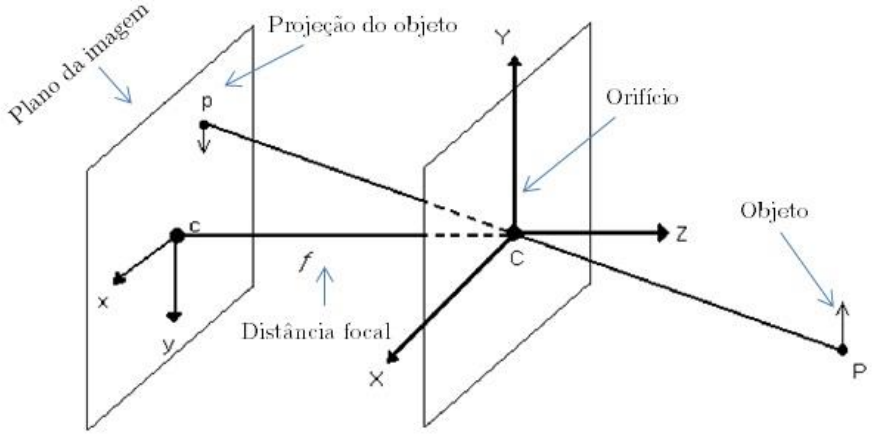


Figura 3.4 - Representação do modelo *pinhole*. Figura extraída de [24].

A Figura 3.4 ilustra bem o funcionamento do modelo de câmera *pinhole*. A luz incidente sobre um objeto é refletida na direção da câmera, mas apenas os raios que chegam ao orifício, ou ponto focal, localizado em C na imagem, conseguem chegar ao

plano da imagem. Assim sendo, a imagem capturada fica invertida em relação ao objeto real. [26]

Os pontos obtidos podem ser representados através da seguinte relação:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = f \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix} \tag{3.1}$$

Escrito em coordenadas homogêneas pode ser representado em forma matricial como:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{3.2}$$

onde f é a distância focal, X, Y e Z são as coordenadas dos pontos do objeto no espaço tridimensional e x, y e z as coordenadas do ponto correspondente no plano da imagem.

Em posse de informações como: distância focal e a distância em Z; pode-se utilizar a relação 3.2 para calcular quanto um *pixel* representa no espaço real, o que permite estimar o tamanho real de objetos. A distância focal é um parâmetro que pode ser medido e depende somente da câmera, de acordo com o modelo *pinhole*. Já a distância em Z, conhecida como profundidade, é uma das informações perdidas na conversão 3D→2D. É possível recuperá-la usando relações encontradas entre duas imagens através da geometria epipolar. [26]

O modelo de um CCD (*charge coupled device*) deve ser introduzido ao modelo *pinhole* para que imagem luminosa, contínua, seja transformada em dados, discretos, armazenáveis e processáveis. Esse dispositivo, composto por vários sensores, localiza-se no plano de formação da imagem, onde no modelo *pinhole* é colocada à placa fotossensível, e transforma a luminosidade em valores digitais. [24]

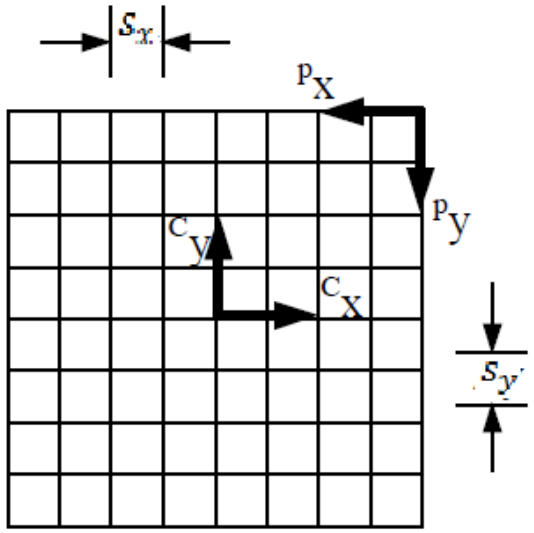


Figura 3.5 - Modelo de CCD. Figura extraída de [24].

A modelagem de um CCD pode ser feita por um plano, onde as coordenadas da imagem (x, y) estão dadas com respeito ao ponto central (Figura 3.5). A fim de expressar a imagem de um ponto em coordenadas de pixel (u, v) o seguinte mapeamento pode ser feito:

$$u = s_x x + c_x,$$
$$v = s_y y + c_y, \tag{3.3}$$

onde s_x e s_y [em mm/pixel] são os tamanhos dos pixels nas direções x e y respectivamente, e c_x e c_y são as coordenadas em pixels do ponto central. Logo, a matriz de projeção de uma câmera, em coordenadas homogêneas, é expressa por:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & 0 & c_x & 0 \\ 0 & fs_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{3.4}$$

onde,

$$K = \begin{bmatrix} fs_x & 0 & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

e corresponde a matriz de parâmetros intrínsecos da câmera. [26]

3.3 PROCESSAMENTO DIGITAL DE IMAGENS

Pode-se definir processamento de imagem como sendo qualquer forma de processamento de dados no qual a entrada é uma imagem, e necessita-se trabalhar essa entrada a fim de realçar ou ofuscar certas características para um determinado fim.

Dentre os nossos sentidos, a visão é o mais complexo e desenvolvido, assim, é fácil perceber que a imagem é a mais importante ferramenta na percepção humana do mundo. Porém, diferente dos humanos, limitados pelo espectro visível da luz, o processamento digital de imagens consegue cobrir todo o espectro de luz através de imagens geradas por fontes que não formam imagens pela percepção humana [24]. Exemplo disso é o avanço da astronomia e astrofísica a partir da espectroscopia do espaço.

3.3.1 DETECÇÃO DE VÉRTICES E BORDAS

Vértices, ou cantos, e bordas são conhecidos por serem ótimas características a se observar numa imagem, quando se deseja extrair informações dela. A estrutura de um vértice ou borda pode ser caracterizada através da matriz C, na vizinhança Q de um ponto P extraído da imagem, onde E_x é a derivada em x da imagem, E_y é a derivada em y. [24]

$$C = \begin{bmatrix} \sum E_x{}^2 & \sum E_x * E_y \\ \sum E_x * E_y & \sum E_y{}^2 \end{bmatrix} \tag{3.6}$$

Aplicando esse operador em toda a imagem, obtém-se em cada ponto a matriz C. Diagonalizando a matriz C obtemos os autovalores λ_1 e λ_2. A partir daí é possível utilizar as condições mostradas a seguir, a fim de identificar os vértices na imagem. [24]

$$C = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{3.7}$$

Se $\lambda_1 = \lambda_2 = 0$; Q é perfeitamente uniforme.

Se $\lambda_2 = 0$; $\lambda_1 > 0$; Q contém borda de degrau ideal, branco e preto e o autovetor de λ_1 é paralelo ao gradiente da imagem.

Se $\lambda_1 > \lambda_2 > 0$; Q contém o vértice de um quadrado preto em fundo branco (ou vice-versa).

Após a detecção desses vértices ou bordas, eles podem ser aplicados em algoritmos de fluxo óptico, calibração de câmeras e até mesmo no cálculo de homografia. Nesse trabalho será de extrema importância para aplicação nos algoritmos de estimação de movimento sem marcadores, detalhados na sessão 3.6.2 deste capítulo.

3.4 CALIBRAÇÃO DE CÂMERAS

Como afirma [22], em teoria é possível imaginar uma lente que não introduza distorções à imagem. Já na prática, nenhuma lente é perfeita. Assim, precisamos considerar as distorções da lente que ocorrem em dois tipos principais: distorções radiais e tangenciais; o primeiro tipo ocorre em razão do formato das lentes, enquanto que o segundo ocorre devido ao processo de montagem da câmera. As distorções em câmeras reais ocorrem nos locais dos pixels perto das extremidades da lente. Esse fenômeno é o responsável pelo efeito conhecido como olho de peixe. Desse modo, à medida que nos afastamos radialmente do centro da lente, aumenta a distorção, por isso é chamada distorção radial. A distorção tangencial ocorre quando as lentes não estão perfeitamente paralelas ao plano de projeção (CCD).

Devido a essas distorções na imagem de uma determinada câmera, fez-se necessário a criação de um procedimento de calibração, em que essas distorções pudessem ser

identificadas e corrigidas para a obtenção de uma imagem o mais próximo da realidade possível.

Como explicitado em [23] o processo de calibração de uma câmera tem como saída tanto um modelo da geometria da câmera quanto um modelo de distorção de sua lente. A partir desses dois modelos, são extraídos os parâmetros intrínsecos da câmera, os quais são usados para corrigir matematicamente as distorções de lente.

3.4.1 CALIBRAÇÃO REALIZADA

O procedimento de calibração adotado faz uso de várias imagens de *chessboards* ou "tabuleiros de xadrez" que são feitos usando a câmera que se quer calibrar, no caso deste trabalho, a câmera inferior do drone. A partir das imagens extraídas da câmera, o algoritmo de calibração calcula os parâmetros intrínsecos da câmera.

O algoritmo de calibração usado faz parte do pacote *Camera Calibration Toolbox for* MATLAB®, descrito na seção 2.3.2 do capítulo 2.

Primeiramente foram tiradas vinte fotos, de diferentes ângulos em relação ao plano do *chessboard*, como ilustrado na Figura 3.6.

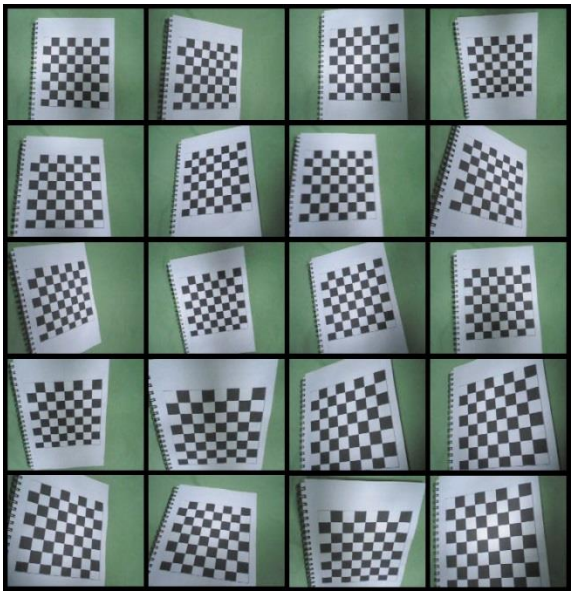


Figura 3.6 – Mosaico de imagens da câmera inferior do drone.

Após a inserção no programa das imagens tiradas da câmera do drone, são identificados os cantos do tabuleiro de xadrez de cada imagem (Figura 3.7). O algoritmo de calibração realiza a associação dos respectivos cantos em cada imagem distinta, com o objetivo de extrair os parâmetros da câmera.

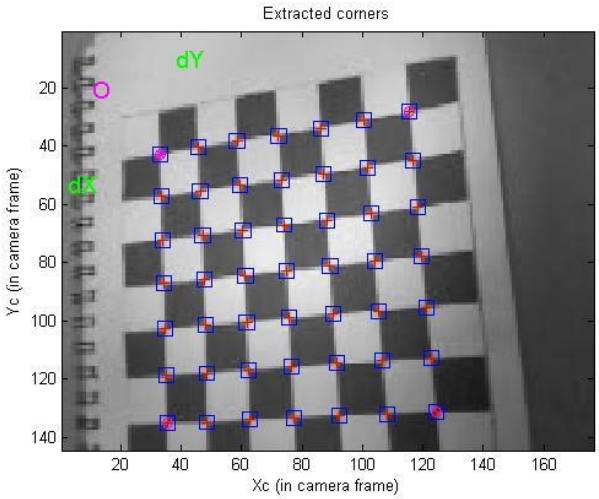


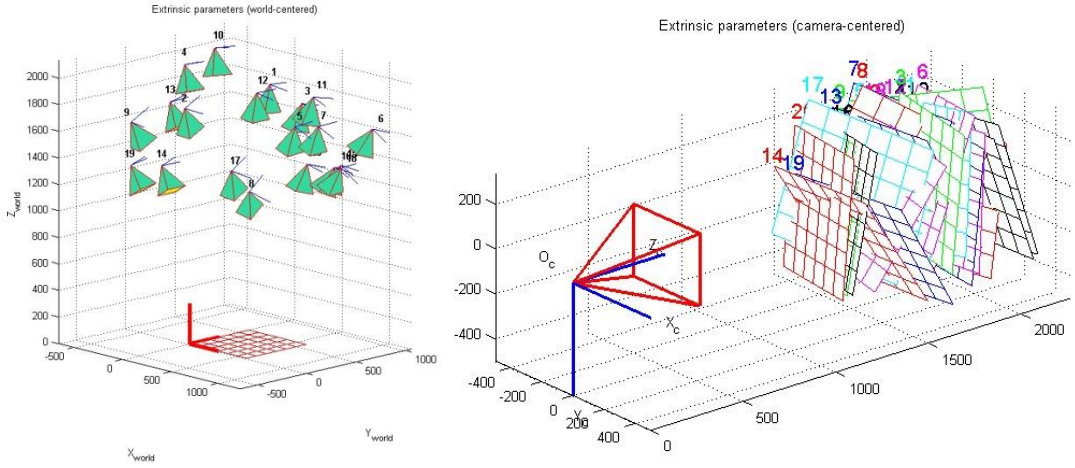
Figura 3.7 - Identificação dos cantos no *chessboard*.

Com a relação entre os cantos nas imagens, o algoritmo de calibração obtém os parâmetros intrínsecos (Tabela 3.1) e extrínsecos (Figura 3.8) da câmera.

Tabela 3.1 – Parâmetros Intrínsecos da câmera

Parâmetros	Antes da Calibração	Depois da Calibração
Distância Focal	fc=[204.47439 204.47539]	fc=[204.88358 222.24123]± [2.12092 2.28993]
Ponto Principal	cc=[87.5 71.5]	cc=[84.50501 70.60671]± [2.29382 2.61901]
Inclinação	alpha_c=[0] =>ângulo do pixel = 90°	alpha_c=[0]±[0] =>ângulo do pixel = 90°±0°
Distorção	kc=[0 0 0 0 0]	kc=[0.34366 -1.28192 -0.00410 -0.00577 0.00000]±[0.06225 0.52925 0.00705 0.00647 0.00000]
Erro do Pixel	-	err=[0.12049 0.17586]

Depois de identificar os parâmetros intrínsecos da câmera, o algoritmo de calibração pode então, utilizando a matriz de distorção, corrigir as imagens, a fim de criar uma imagem já calibrada e sem distorções. Essa correção é bem ilustrada na Figura 3.9.



(a) A imagem como centro (b) A câmera como centro

Figura 3.8 - Parâmetros extrínsecos da câmera

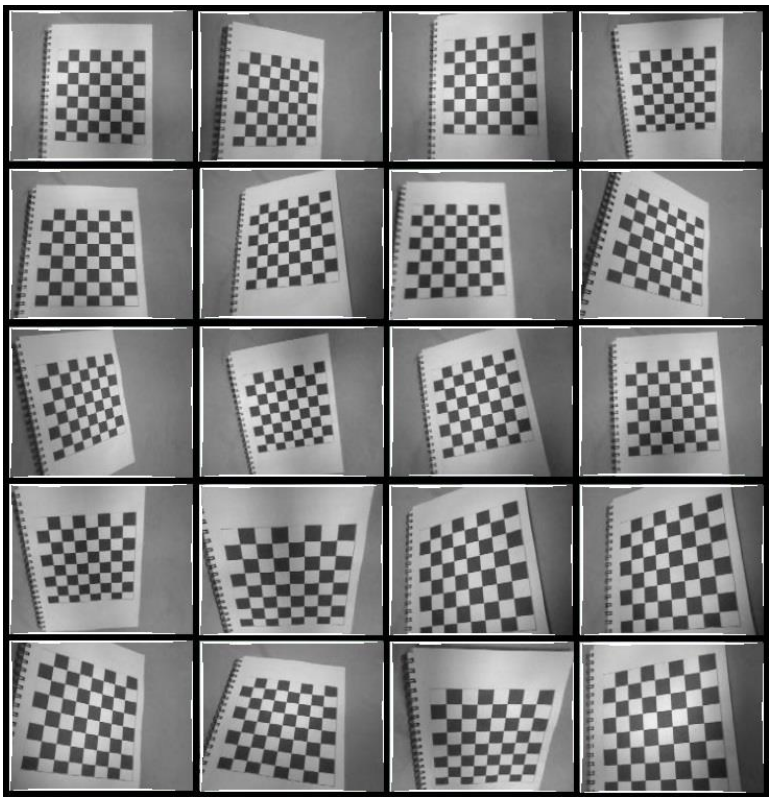


Figura 3.9 - Mosaico das imagens corrigidas

Em posse das imagens originais e das corrigidas, pode-se então definir o erro para cada imagem, como sendo, a diferença entre a imagem original e a sem distorção. Foi desenvolvido então um *script* em MATLAB para ilustrar bem esse erro (Figura 3.10).

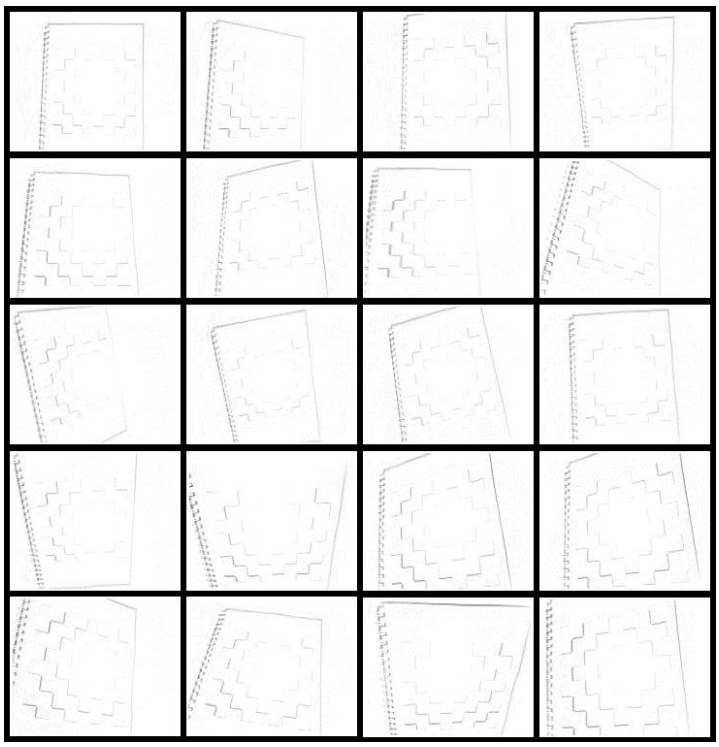


Figura 3.10 - Mosaico das distorções em cada imagem.

Fica claro a partir da Figura 3.10 que as distorções da câmera são relativamente pequenas, tendo em vista também a distância em que as fotos foram tiradas e a distorção em pixels.

É claro que a precisão é um aspecto importante neste trabalho, afinal, pequenas perturbações podem gerar grandes problemas para um voo estável do quadrirrotor, porém, levando em consideração as pequenas distorções encontradas, bem como a necessidade de se trabalhar com respostas em tempo real, ficou evidente que o custo computacional envolvido em corrigir quadro a quadro do pacote de imagens recebidos pelo drone é bem superior ao efeito real dessas distorções.

Assim sendo, não foi adotado nenhum procedimento de correção das imagens recebidas.

3.5 RELAÇÃO MILÍMETROS POR PIXEL

Uma etapa importante para o trabalho é a obtenção da relação entre um pixel na imagem com o comprimento real desse pixel no ambiente. Sabe-se que essa relação depende apenas da distância focal e da distância entre a câmera e o objeto em estudo.

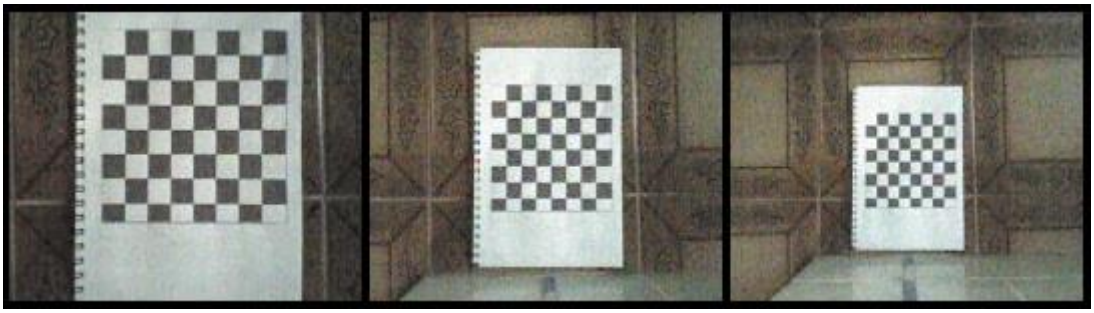
Para isso foi realizado um experimento colhendo três imagens do mesmo tabuleiro de xadrez utilizado no procedimento de calibração, onde previamente sabe-se o comprimento de cada quadrado preto por exemplo. A distância focal não foi utilizada nesse caso. As fotos foram tiradas de distâncias distintas, para isso foi utilizada uma fita métrica na parede para garantir boas medições, como ilustrado na Figura 3.11.

Figura 3.11 – Foto tirada com uma câmera externa para ilustrar o experimento.

As imagens foram colhidas a distâncias de 40 cm, 60 cm e 80 cm em relação ao chão (Figura 3.12). Além da imagem e da distância medida pela fita métrica, também foi extraído o valor do sensor ultrassônico para cada uma das distâncias, com o objetivo final de identificar o deslocamento do quadrirrotor de acordo com sua distância ao solo.

A relação entre um pixel na imagem e seu tamanho no mundo físico pode ser afetada diretamente pelas inclinações φ e θ do quadrirrotor, pertencentes ao sistema de coordenadas B, ilustrado na Figura 2.9 da seção 2.4 deste trabalho, devido ao efeito de

perspectiva. Como para mover-se nas direções x e y o quadrirrotor precisa se inclinar, as medidas da relação entre pixels e milímetros são afetadas de qualquer maneira, logo essas distorções serão descartadas, levando em consideração os pequenos ângulos de inclinação, e o alto custo computacional demandado para solucionar o problema.



(a) 40 cm (b) 60 cm (c) 80 cm

Figura 3.12 – Imagens extraídas da câmera inferior do quadrirrotor.

Para melhorar a precisão das medidas, foi utilizada uma distância fixa maior que um quadrado do tabuleiro, essa distância D = 156 mm é ilustrada na Figura 3.13.

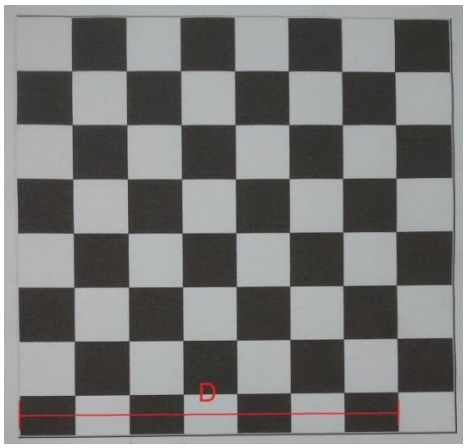


Figura 3.13 – Tabuleiro de xadrez utilizado, com destaque para a distância D.

Os dados obtidos nesse experimento estão indicados na Tabela 3.2.

Tabela 3.2 – Dados obtidos no experimento.

Altura (cm)	Qtd. Pixels em D	Valor Sensor
40	81	370
60	54	570
80	40	770

Tem-se então que a relação mm/pixel para cada uma das alturas é como ilustrado na Tabela 3.3.

Tabela 3.3 – Relação mm/pixel.

Altura (cm)	Relação (mm/pixel)
40	$\frac{156\ mm}{81\ pixels} = 1{,}926$
60	$\frac{156\ mm}{54\ pixels} = 2{,}890$
80	$\frac{156\ mm}{40\ pixels} = 3{,}900$

Graficamente tem-se:

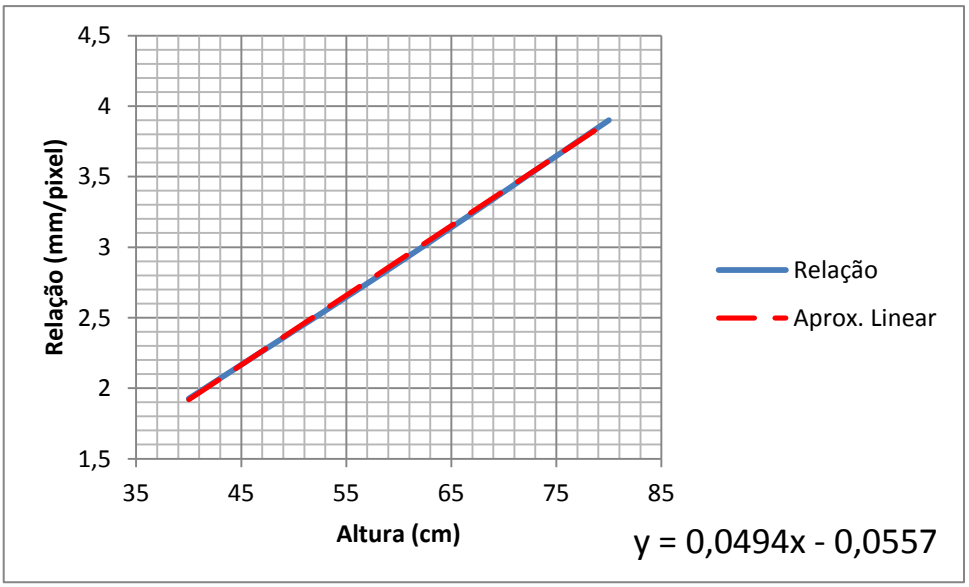


Figura 3.14 – Relação mm/pixel de acordo com a altura.

De acordo com a aproximação linear obtida, tem-se então a equação 3.8.

$$R_{mm/pixel} = 0{,}0494a - 0{,}0557, \tag{3.8}$$

onde $R_{mm/pixel}$ é a relação mm/pixel e a é a altura do quadrirrotor.

A partir dos dados do sensor indicados na Tabela 3.2 e dos dados da relação mm/pixel, pode-se então encontrar a relação de mm/pixel em função do valor do sensor, como ilustra a Figura 3.15.

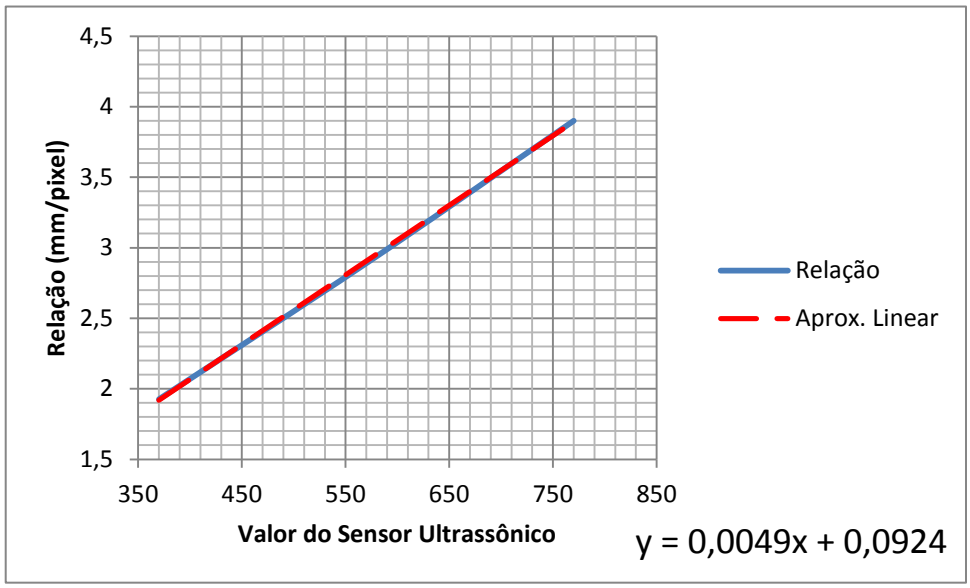


Figura 3.15 – Relação mm/pixel de acordo com o valor do sensor.

De acordo com a aproximação linear obtida, tem-se então a equação 3.9.

$$R_{mm/pixel} = 0{,}0049V_s + 0{,}0924 \ , \tag{3.9}$$

onde V_s é o valor do sensor para uma determinada altura.

3.6 ALGORITMOS DE ESTIMAÇÃO DE MOVIMENTO

A estimação de movimento é feita localizando no quadro de referência, qual bloco mais se assemelha ao bloco atual. Um bloco é um conjunto de pixels. Assim que o bloco é encontrado, a estimação de movimento gera um vetor indicando módulo, direção e sentido que este bloco se movimentou no quadro de referência. Esse vetor é chamado de vetor de movimento.

Nessa área de estimação de movimento, existem diversos algoritmos para se localizar os blocos que mais se assemelham entre o quadro de referência e o que está

sendo processado. São esses algoritmos que, após localizarem o bloco, geram o vetor de movimento correspondente. [27]

Marcadores podem ser inseridos nas imagens para que os algoritmos possam encontrar mais facilmente as características da imagem de referência na atual.

3.6.1 COM MARCADORES

Geralmente, em algoritmos de perseguição são utilizados marcadores, assim fica mais fácil encontrar a localização do determinado marcador na imagem atual, sabendo características desse marcador como: cor, textura, etc. Porém, quando se quer saber, além da localização, informações como: sentido em relação ao marcador; a tarefa se torna mais complexa.

Algoritmos com marcadores, que procuram encontrar mais de uma característica do marcador, geralmente são utilizados em realidade aumentada, onde são essenciais informações como direção e sentido, esses marcadores são conhecidos como *tags.*

No inicio do projeto, foi cogitada a utilização de marcadores, porém, como o objetivo é tornar o voo mais estável independente do local, é importante não restringir tanto as condições do ambiente.

3.6.2 SEM MARCADORES

Algoritmos de estimação de movimento sem marcadores, ou melhor, sem características pré-definidas, geralmente utilizam técnicas de fluxo óptico, de natureza esparsa ou densa. Esse tipo de técnica possibilita a identificação de movimento entre sequência de frames sem que se conheça a priori o conteúdo destes. Tipicamente, o movimento em si indica que algo de interessante está acontecendo [28].

Os algoritmos que não utilizam técnicas de fluxo óptico também são eficazes para algumas aplicações, os mais conhecidos são: *Meanshift* e *Camshift* (*Continuously Adaptive Mean -SHIFT*). O *Camshift* é um algoritmo desenvolvido para o rastreamento de cor, possibilitando também o rastreamento de faces. É baseado numa técnica estatística onde se busca o pico entre distribuições de probabilidade em gradientes de densidade. Esta técnica é chamada de "média por deslocamento" (*mean shift*) e foi adaptada no *Camshift* para tratar a mudança dinâmica das distribuições de probabilidade das cores numa sequência de vídeo. Pode ser usada no rastreamento de objetos e no rastreamento de faces.

Algoritmos de fluxo óptico de natureza esparsa consideram algum conhecimento prévio sobre os pontos que se deseja rastrear, como por exemplo, os cantos descritos na seção 3.3.1. Os algoritmos densos, por sua vez, associam um vetor de velocidade ou de deslocamento a cada pixel na imagem, sendo, portanto desnecessário o conhecimento prévio de pontos específicos da imagem. Para a maioria das aplicações práticas, entretanto, as técnicas densas possuem um custo de processamento muito alto, sendo preferíveis, portanto, as técnicas esparsas. O conjunto de bibliotecas *OpenCV* possuem funções que implementam técnicas de detecção de movimento esparsas e densas. A Tabela 3.4 resume as funções de detecção de movimento presentes no *OpenCV*.

Tabela 3.4 – Algoritmos de Fluxo Óptico do *OpenCV*.

Algoritmo	Tipo	Comando
Lucas-Kanade	Esparso	`cvCalcOpticalFlowLK`
Lucas-Kanade Piramidal	Esparso	`cvCalcOpticalFlowPyrLK`
Gunnar Farneback	Denso	`cvCalcOpticalFlowFarneback`
Horn-Shunk	Denso	`cvCalcOpticalFLowHS`
Block Matching	Denso	`cvCalcOpticalFlowBM`

É importante na maioria dos casos utilizar algoritmos já consolidados e implementados no *OpenCV,* pois eles já são otimizados e assim o custo computacional em seus processamentos são menores.

3.6.3 COMPARATIVO GERAL

Como já dito anteriormente, a estimação de movimentos pode ser feita utilizando, ou não, marcadores. Quando o objetivo é seguir um objeto que se move, é melhor utilizar algoritmos com marcadores, que são mais diretos e fáceis de implementar, porém, quando o cenário está mudando a todo instante, é necessário utilizar outra abordagem.

Quando o objetivo é detectar o movimento realizado pela câmera, comparando uma imagem com um *background,* utilizar marcadores não é uma boa ideia. Isso limitaria muito o sistema. Para isso, existem os algoritmos que não utilizam marcadores. *Camshift* como citado anteriormente utiliza uma abordagem por cores, as quais podem sofrer muitas alterações. Já os algoritmos de fluxo óptico, utilizam características de uma vizinhança de pixels para tentar encontrar essas mesmas características em outra imagem, definindo assim um vetor de movimento. A vantagem dos algoritmos de fluxo óptico está no seu uso mais genérico, que pode ser aplicado em diversas ocasiões, desde situações em que se quer obter simplesmente um vetor de deslocamento entre duas imagens, até em sistemas de *tracking* bem complexos.

Para situações em que se trabalha em tempo real é importante utilizar algoritmos esparsos de fluxo óptico, devido ao seu menor custo computacional. É importante também escolher as melhores características de uma imagem a serem buscadas em uma

imagem de referência para obter um vetor que exprima de forma mais exata o movimento entre as imagens.

3.7 ALGORITMOS UTILIZADOS

A partir de toda análise realizada na parte de visão computacional, decidiu-se então utilizar algoritmos de fluxo óptico, por serem os que melhor atendiam as expectativas quanto à obtenção de um vetor que definisse o movimento entre imagens da forma mais exata possível.

Apesar de algoritmos densos terem um grande custo computacional e a implementação no caso deste projeto precisar da informação de posição quase de forma instantânea, a câmera inferior, que usaremos para implementar o algoritmo, tem uma resolução muito baixa (QCIF - 176x144) e uma taxa de atualização alta de 60fps, logo, implementar algoritmos densos não se torna tão dispendioso.

O algoritmo Lucas-Kanade foi a primeira opção de implementação no projeto, por sua simplicidade, porém, logo na explicitação do algoritmo no guia do *OpenCV,* a informação era que esse algoritmo já estava obsoleto, e indicava o uso de outros algoritmos como: Lucas-Kanade Piramidal e Farneback.

Foi implementado então primeiramente, por sua simplicidade e eficiência, o algoritmo denso de Gunnar Farneback.

A fim de chegar à conclusão de qual algoritmo atende de forma mais satisfatória o objetivo de conseguir um vetor de movimento coerente, foi implementado posteriormente o algoritmo esparso Lucas-Kanade Piramidal.

3.7.1 Algoritmo de Gunnar Farneback

O algoritmo desenvolvido por Gunnar Farneback é uma análise de fluxo óptico denso que produz um campo de deslocamento entre dois frames de vídeo sucessivos.

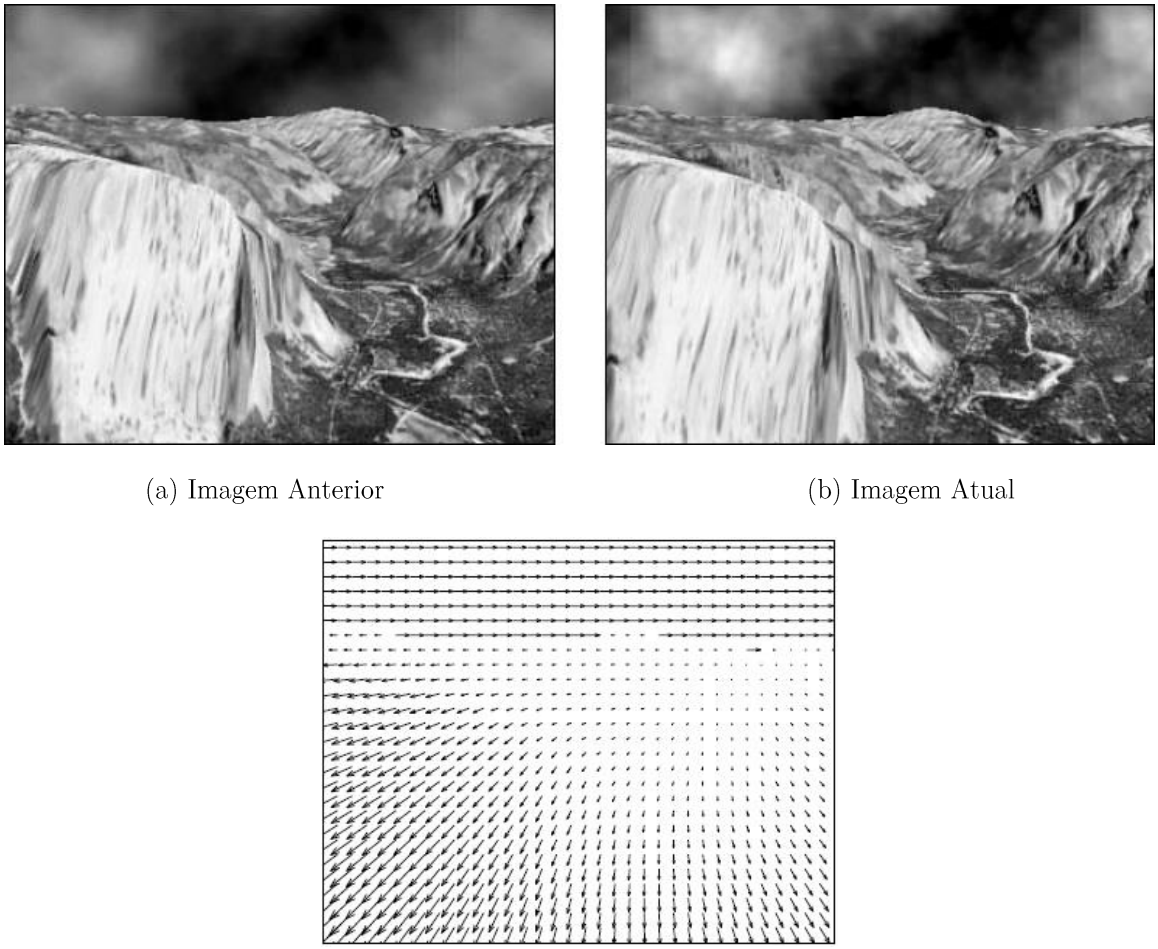
Cada vizinhança de ambos os frames são aproximadas por polinômios quadráticos, o que pode ser feito com eficiência usando uma transformação de expansão polinomial. A partir daí, começa-se a analisar o que acontece quando o polinômio passa por uma translação ideal [29]. Ao final do algoritmo, chega-se então a um pixel no segundo frame correspondente ao mesmo pixel do primeiro frame, resultando assim em um vetor de deslocamento entre os dois frames.

Esse algoritmo, por ser denso, é aplicado a todos os pixels da imagem, e para cada pixel, em uma determinada vizinhança. Essa vizinhança normalmente é pequena, logo, o algoritmo fica restrito a pequenos deslocamentos entre as imagens. A solução para esse problema é aumentar a vizinhança do pixel, o que ocasiona em um aumento do custo computacional.

Na função do *OpenCV* que implementou esse algoritmo no drone, foi utilizada uma vizinhança com 15 pixels, e 5 iterações do algoritmo, a fim de convergir a um bom resultado. Vizinhanças maiores foram testadas, porém, resultados satisfatórios para grandes deslocamentos não foram obtidos.

Com o objetivo de eliminar ruídos, pequenos vetores de deslocamento (inferiores a dois pixels) foram descartados. Para diminuir o custo computacional envolvido, foram utilizados vetores de deslocamento de forma espaçada de 15 em 15 pixels na imagem para obter o vetor de média.

Um exemplo de implementação do algoritmo está na Figura 3.16.



(a) Imagem Anterior

(b) Imagem Atual

(c) Fluxo Óptico

Figura 3.16 - Frames selecionados da sequencia Yosemite e campo de velocidade. Figura extraída de [29]

3.7.2 Algoritmo Lucas-Kanade Piramidal

O algoritmo Lucas-Kanade (LK), como originalmente proposto em 1981, foi uma tentativa de produzir resultados densos. Pelo fato de o método ser facilmente aplicado a um subconjunto de pontos na imagem de entrada, passou a ser uma importante técnica esparsa. O algoritmo de LK pode ser aplicado em um contexto esparso porque depende apenas da informação local que é derivada de alguma pequena vizinhança que cerca cada um dos pontos de interesse. [23]

A desvantagem de usar pequenas janelas locais é que grandes movimentos podem mover pontos para fora da janela e assim tornar impossível para o algoritmo encontrar novamente esse ponto. Esse problema que levou ao desenvolvimento do algoritmo LK

piramidal, que monitora a partir do nível mais alto de uma pirâmide de imagem (menos detalhes) até níveis inferiores (mais detalhes). O acompanhamento com pirâmides de imagem permite que grandes movimentos sejam detectados pela janela local.

A ideia básica do algoritmo LK assenta em três pressupostos:

- Constância do brilho. Um pixel da imagem de um objeto na cena não altera na aparência, enquanto se move (possivelmente) de quadro a quadro.
- Persistência temporal ou "pequenos movimentos". O movimento da imagem ocorre lentamente.
- Coerência espacial. Pontos vizinhos em uma cena pertencem à mesma superfície, têm movimentos semelhantes e são projetados para pontos próximos no plano da imagem.

Tendo como base esses pressupostos, o algoritmo realiza uma análise matemática sofisticada, utilizando o fato da invariância ou pequena variância no tempo, para aplicar os conceitos de derivada. No caso do brilho constante, por exemplo, a derivada da intensidade do brilho é igual à zero, e a partir do pressuposto de pequenos movimentos, pode-se trabalhar com termos diferenciais.

Para implementação do algoritmo utilizando a função correspondente do *OpenCV,* fez-se necessária a obtenção de pixels com vizinhanças com boas características a se rastrear, tais como: cantos ou vértices e bordas. Essas características foram obtidas a partir de uma função do *OpenCV*, chamada `cvGoodFeaturesToTrack`, que retorna uma matriz com esses pontos de interesse. Resultando assim em vetores de movimento que variam de acordo com a variação na imagem. A Figura 3.17 ilustra bem a aplicação da função `cvGoodFeaturesToTrack` e do algoritmo.

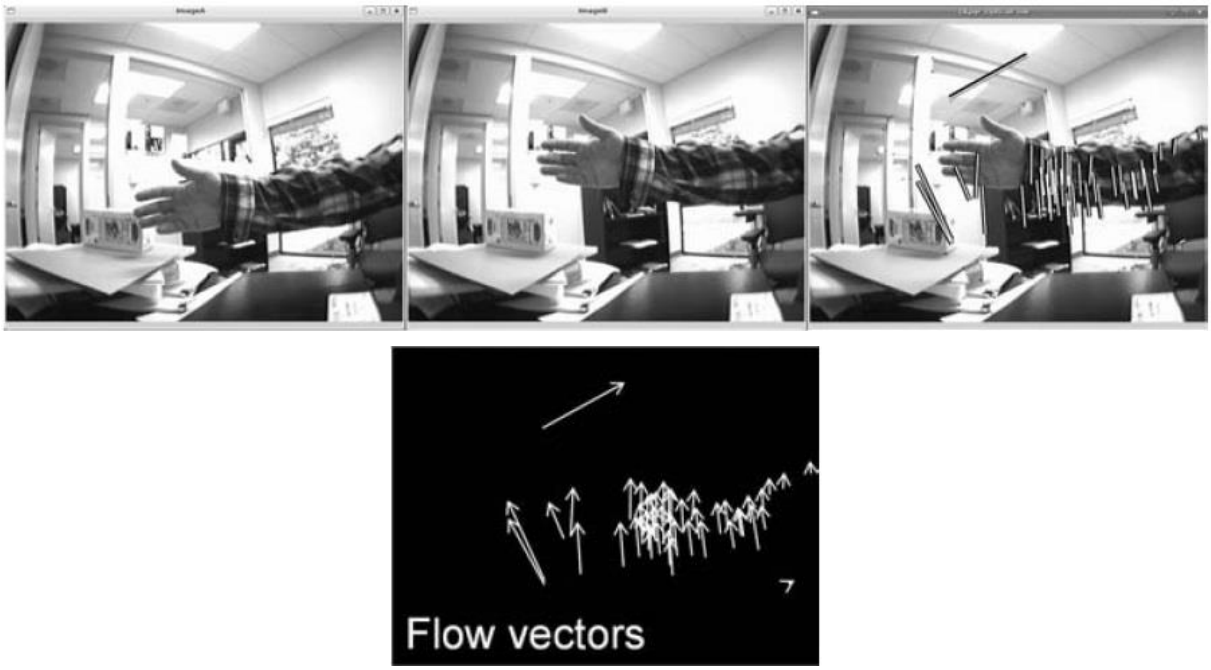


Figura 3.17 - Implementação do algoritmo Lucas-Kanade Piramidal. Figura extraída de [23].

4 MODELO, IDENTIFICAÇÃO E CONTROLE DO SISTEMA

Neste capitulo serão expostos: as equações de movimento que descrevem a dinâmica do drone; o levantamento experimental da função de transferência que modela o sistema; o projeto e implementação de controladores; bem como uma simulação de todo o conjunto.

4.1 MODELAGEM DO SISTEMA

Todo o equacionamento matemático descrito nesta seção é utilizado para fins teóricos. A modelagem enfatizada é o levantamento experimental das respostas do sistema descrito nas próximas seções.

4.1.1 DINÂMICA DO SISTEMA

Para modelar a dinâmica do sistema do quadrirrotor, foi utilizado o modelo descrito em [30] *apud* [31], baseado no modelo de Newton-Euler para um corpo genérico de seis graus de liberdade.

As equações de movimento são definidas no sistema B, ilustrado na Figura 2.9, pelas seguintes razões:

- A matriz de inércia é invariante no tempo;
- A simetria do corpo pode ser utilizada para simplificar as equações;
- As forças exercidas pelos motores são dadas neste sistema;
- A aceleração medida pelo acelerômetro também é dada neste sistema.

Assim sendo, a equação 4.1 descreve o movimento do sistema:

$$v = [V^B \quad \omega^B] = [u \quad v \quad w \quad p \quad q \quad r]^T, \tag{4.1}$$

onde V^B descreve a velocidade linear (em suas componentes $u, v\ e\ w$) e ω^B descreve a velocidade angular do corpo (em suas componentes $p, q\ e\ r$), todas dadas no sistema de coordenadas B, como mencionado anteriormente.

Para modelar a dinâmica de um corpo de seis graus de liberdade, são levadas em conta: a massa do corpo e a sua matriz de inércia. Esta dinâmica está descrita na equação 4.2.

$$\begin{bmatrix} mI_{3x3} & 0_{3x3} \\ 0_{3x3} & J_{1x3}I_{3x3} \end{bmatrix} \begin{bmatrix} \dot{V} \\ \dot{\omega^B} \end{bmatrix} + \begin{bmatrix} \omega^B \times (mV^B) \\ \omega^B \times (I\omega^B) \end{bmatrix} = \begin{bmatrix} F^B \\ \tau^B \end{bmatrix}, \tag{4.2}$$

onde m é a massa do corpo, I_{3x3} é a matriz identidade de dimensão três, 0_{3x3} é uma matriz quadrada de dimensão três com zeros em todas as posições, J_{1x3} é a matriz de inercia, F^B é o vetor de forças atuantes sobre o sistema e τ^B é o vetor de torques atuantes sobre o sistema, ambos dados em B.

Escrevendo a equação 4.2 na forma matricial, obtêm-se a equação 4.3.

$$M_B\dot{v} + C_B(v)v = \Lambda, \tag{4.3}$$

onde M_B é a matriz generalizada de inércia do corpo, C_B é a matriz que leva em consideração a aceleração centrípeta de Coriolis e Λ é o vetor generalizado de forças do corpo. Definidos, respectivamente, nas equações 4.4, 4.5 e 4.6.

$$M_B = \begin{bmatrix} mI_{3x3} & 0_{3x3} \\ 0_{3x3} & J_{1x3}I_{3x3} \end{bmatrix} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{ii} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{jj} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{kk} \end{bmatrix} \tag{4.4}$$

$$C_B(v) = \begin{bmatrix} 0_{3x3} & -mS(V^B) \\ 0_{3x3} & -S(I\omega^B) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & mw & -mv \\ 0 & 0 & 0 & -mw & 0 & mu \\ 0 & 0 & 0 & mv & -mu & 0 \\ 0 & 0 & 0 & 0 & I_{kk}r & -I_{jj}q \\ 0 & 0 & 0 & -I_{kk}r & 0 & I_{ii}p \\ 0 & 0 & 0 & I_{jj}q & -I_{jj}p & 0 \end{bmatrix} \tag{4.5}$$

$$\Lambda = [F^B \quad \tau^B]^T = [F_i \quad F_j \quad F_k \quad \tau_i \quad \tau_j \quad \tau_k]^T \tag{4.6}$$

A equação 4.3 é genérica e válida para qualquer corpo que obedeça às condições estabelecidas previamente. Porém, para o sistema em questão, pode-se dividir o vetor Λ em três componentes, de acordo com a natureza das forças atuantes sobre o quadrirrotor.

Uma componente é relacionada ao vetor gravitacional G_B, dado a partir da aceleração devida à gravidade g [m/s^2] e estabelecido na equação 4.7.

$$G_B = \begin{bmatrix} F_G^B \\ 0_{3x1} \end{bmatrix} = \begin{bmatrix} R_\theta^{-1} F_G^E \\ 0_{3x1} \end{bmatrix} = \begin{bmatrix} R_\theta^T \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \\ 0_{3x1} \end{bmatrix} = \begin{bmatrix} mgs_\theta \\ -mgc_\theta s_\varphi \\ -mgc_\theta s_\varphi \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{4.7}$$

onde s_θ representa o $\operatorname{sen}\theta$, c_θ representa $\cos\theta$ e s_φ representa o $\operatorname{sen}\varphi$.

A segunda contribuição advém dos efeitos giroscópicos produzidos pela rotação das hélices. Como duas estão rodando no sentido horário e as outras duas no sentido anti-horário, existe um desequilíbrio quando a soma algébrica das velocidades dos rotores não é igual à zero. Além disso, se os ângulos de rolagem e arfagem forem diferentes de zero, o quadrirrotor sofre torques giroscópicos de acordo com a equação 4.8. Nesta equação, Ω é a soma algébrica das velocidades dos rotores, dado pela equação 4.9, onde Ω_k é a velocidade do motor k.

$$O_B(v)\Omega = \left[-\sum_{k=1}^{4} J_{TP} \left(\begin{matrix} 0_{3x1} \\ \omega^B \mathrm{x} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \end{matrix} \right) (-1)^k \Omega_k \right]$$

$$= \begin{bmatrix} 0_{3x1} \\ J_{TP} \begin{bmatrix} -q \\ p \\ 0 \end{bmatrix} \end{bmatrix} = J_{TP} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ q & -q & q & -q \\ -p & p & -p & p \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.8}$$

$$\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \tag{4.9}$$

A última componente se deve ás forças e torques produzidos diretamente pelos motores. Obtêm-se através de considerações aerodinâmicas que estas são diretamente proporcionais ao quadrado da velocidade de rotação dos rotores. Estas contribuições são descritas por:

$$U_B(\Omega) = E_B \Omega^2 = \begin{bmatrix} 0 \\ 0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ bl(\Omega_4^2 - \Omega_2^2) \\ bl(\Omega_3^2 - \Omega_1^2) \\ d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \end{bmatrix}, \tag{4.10}$$

onde b é o coeficiente de empuxo, d é o coeficiente de arrasto e l é a distância entre os rotores e o centro de massa do quadrirrotor.

Logo, é possível descrever a dinâmica do sistema a partir da seguinte equação matricial:

$$M_B \dot{v} + C_B(v)v = G_B(\xi) + O_B(v)\Omega + E_B \Omega^2 \tag{4.11}$$

Isolando a derivada do vetor velocidade obtém-se:

$$\dot{v} = M_B^{-1}(-C_B(v)v + G_B(\xi) + O_B(v)\Omega + E_B \Omega^2) \tag{4.12}$$

Representando a equação 4.12 como um sistema de equações, obtém-se:

$$\begin{cases} \dot{u} = (vr - wq) + gs_\theta \\ \dot{v} = (wp - ur) - gc_\theta s_\varphi \\ \dot{w} = (uq - vp) - gc_\theta s_\varphi + \frac{U_1}{m} \\ \dot{p} = \frac{I_{jj} - I_{kk}}{I_{ii}} qr + \frac{J_{TP}}{I_{ii}} p + \frac{U_2}{I_{ii}} \\ \dot{q} = \frac{I_{kk} - I_{ii}}{I_{jj}} pr + \frac{J_{TP}}{I_{jj}} p + \frac{U_3}{I_{jj}} \\ \dot{r} = \frac{I_{ii} - I_{jj}}{I_{kk}} pq + \frac{U_4}{I_{kk}} \end{cases} \tag{4.13}$$

A velocidade das hélices é dada a partir de:

$$\begin{cases} U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_2 = bl(\Omega_4^2 - \Omega_2^2) \\ U_3 = bl(\Omega_3^2 - \Omega_1^2) \\ U_4 = d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \\ \Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \end{cases} \tag{4.14}$$

As três primeiras equações presentes no sistema de equações 4.13 representam as acelerações medidas pelo acelerômetro, pois, de acordo com o sistema de coordenadas *B*, $u, v \; e \; w$ são as velocidades de translação do quadrirrotor.

Para calcular a inclinação do sistema foi realizado o seguinte equacionamento:

Sejam φ, θ e ψ respectivamente as rotações em i, j e k necessárias para deslocar o sistema de coordenadas *B* de modo que coincida em orientação com o sistema *E*. Desta forma, a matriz de rotação completa é dada por:

$$R_\theta = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\varphi + c_\psi s_\theta s_\varphi & s_\psi s_\varphi + c_\psi s_\theta c_\varphi \\ s_\psi c_\theta & c_\psi c_\varphi + s_\psi s_\theta s_\varphi & -c_\psi s_\varphi + s_\psi s_\theta c_\varphi \\ -s_\theta & c_\theta s_\varphi & c_\theta s_\varphi \end{bmatrix} \tag{4.15}$$

Assim, pode-se obter o vetor velocidade angular em relação ao sistema de coordenadas *E*, chamada de $\dot{\theta_B}$, multiplicando a matriz de rotação pela matriz de velocidade angular ω_B descrita na equação 4.1, obtém-se:

$$\dot{\theta_B} = R_\theta \omega_B \tag{4.16}$$

Agora, deve ser encontrada uma maneira de relacionar as três primeiras equações do sistema de equações 4.13 com os ângulos em relação ao sistema de coordenadas *E*.

Isolando os ângulos φ e θ no sistema de equações 4.13, obtêm-se:

$$\varphi = \frac{\sin^{-1} \ddot{y}}{g \cos\theta} \tag{4.17}$$

$$\theta = \frac{\sin^{-1} \ddot{x}}{g} \tag{4.18}$$

De modo a simplificar os cálculos, ignoram-se as contribuições dos componentes de Coriolis centrípetas e de inércia, por serem muito menores que a contribuição do vetor gravidade, principalmente quando os ângulos de inclinação ou as velocidades angulares são próximos à zero.

A partir das equações 4.17 e 4.18, pode-se então estimar a inclinação do sistema a partir somente dos valores lidos pelo acelerômetro.

4.1.2 MODELAGEM DO MOTOR *BRUSHLESS*

O circuito elétrico equivalente de um motor *brushless* é constituído de três fases simétricas e equilibradas, sendo cada uma constituída por uma resistência de enrolamento, uma autoindutância, resultante da interação entre sua indutância própria e as indutâncias mútuas das outras fases, e uma fonte de tensão que representa a força eletromotriz (f.e.m.) trapezoidal produzida pelo deslocamento do fluxo do rotor nos enrolamentos do estator [32].

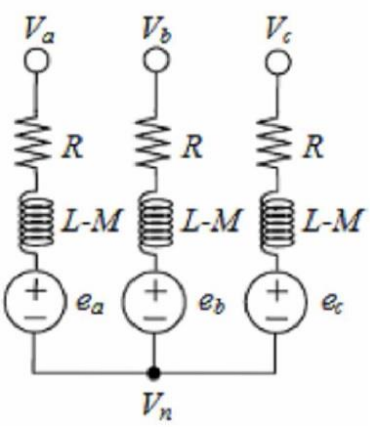


Figura 4.1 – Circuito equivalente de um Motor *brushless* em Y. Figura extraída de [32].

A partir da lei das tensões de Kirchhoff, o valor instantâneo da tensão da fase a do motor pode ser representado pela seguinte expressão:

$$V_a = e_a + Ri_a + (L - M)\frac{di_a}{dt} - V_n \ , \tag{4.19}$$

onde e_a é o valor instantâneo da f.e.m. induzida pela excitação do ímã permanente em uma única fase do enrolamento de armadura, i_a é a corrente instantânea na fase a, R é resistência da armadura por fase, L é a indutância por fase e M é a indutância mútua.

Assim, a equação matricial do motor é:

$$V_{3x1} = e_{3x1} + RI_{3x3}i_{3x1} + [L - M]I_{3x3}\frac{di}{dt}_{3x1} - V_n 1_{3x1}, \tag{4.20}$$

onde I_{3x3} é a matriz identidade e 1_{3x1} é uma matriz coluna preenchida com 1.

As equações do torque eletromagnético de um motor *brushless* e sua equação mecânica são dadas respectivamente por:

$$T_e = \frac{e_a i_a + e_b i_b + e_c i_c}{\omega_r} \ , \tag{4.21}$$

$$\frac{d\omega_r}{dt} = \frac{T_e - T_L - B\omega_r}{J_m} \ , \tag{4.22}$$

onde T_e é o torque eletromagnético, ω_r é a velocidade angular do motor, T_L é o torque da exercido pela carga acoplada ao eixo do motor, B é o coeficiente de atrito viscoso e J_m é o momento de inércia do rotor.

Considerando um motor *brushless* ideal com as suas fases conectadas em Y, em que as comutações são perfeitas, as formas de onda das correntes quadradas e a fonte de alimentação é ideal, é possível representar sua tensão terminal como ([34] *apud* [33]):

$$U = E + 2Ri, \tag{4.23}$$

onde E é a soma das f.e.m. das fases conectadas em série.

A equação 4.23 é a mesma para um motor CC convencional. A queda de tensão através dos comutadores eletrônicos é desprezada, considerando-os como ideais. A partir da equação 4.23 e a equação do torque eletromagnético (4.21), as características de torque e velocidade podem ser obtidas por:

$$\omega = \omega_0 \left[1 - \frac{T}{T_0}\right], \tag{4.24}$$

onde a velocidade angular em vazio, o torque e a corrente com o rotor bloqueado são dados respectivamente por:

$$\omega_0 = \frac{V}{k\Phi}\ rad/s \tag{4.25}$$

$$T_0 = k\Phi I_0 \tag{4.26}$$

$$I_0 = \frac{V}{2R} \tag{4.27}$$

4.1.3 MODELAGEM DAS HÉLICES

O comportamento de uma hélice pode ser baseado em três parâmetros: o Coeficiente de *Thrust (Impulso)* C_T; o Coeficiente de Potência C_P e o Raio da Hélice r.

Esses parâmetros são utilizados no cálculo do Impulso T e da Potência P de uma hélice, explicitados respectivamente nas equações 4.28 e 4.29 ([35] *apud* [36]).

$$T_H = C_T \frac{4\rho r^4}{\pi^2} \omega^2, \tag{4.28}$$

$$P_H = C_P \frac{4\rho r^5}{\pi^3} \omega^3, \tag{4.29}$$

onde ρ é a densidade do ar e ω_H a velocidade angular da hélice.

4.2 INCLINAÇÃO vs. TRANSLAÇÃO

O primeiro experimento teve por objetivo a obtenção da curva Ângulo de Inclinação vs. Velocidade de Translação para o ângulo de rolamento. Tendo em vista a simetria do drone, foi concluído que o experimento para o ângulo de arfagem seria o mesmo, não necessitando assim a repetição do mesmo.

O experimento foi realizado enviando comandos de φ_{AT} para o drone por um determinado tempo, determinado pela variável CT, e medindo com uma trena milimetrada a distância percorrida pelo quadrirrotor.

O tempo de um ciclo de recebimento de pacotes do drone, CT, foi obtido experimentalmente também através do programa utilizado, medido por um cronômetro o tempo total gasto na execução de 50CT, ou seja, 50 ciclos, e dividindo esse tempo por 50. Esse experimento foi colocado à prova em um programa com mais instruções, e não variou o valor obtido, concluindo-se que o gargalo de tempo está no recebimento dos pacotes, e não na execução do programa. O tempo obtido foi de CT=0,056s, logo, o tempo gasto é:

$$t = 50.0{,}056 = 2{,}8\ segundos$$

Foram realizadas cinco medidas de distâncias para cada um dos valores de entrada, φ_{AT} = 0,10, 0,20, 0,35, 0,50 e 1,00, e descartada a pior medida, obtendo assim os dados da Tabela 4.1.

Tabela 4.1 – Dados do primeiro experimento

φ_{AT}	D1 (m)	D2 (m)	D3 (m)	D4 (m)	Média	Desv. Pad
0,1	1,65	1,625	1,85	1,49	1,65375	0,148513
0,20	3,63	3,25	3,65	3,64	3,54250	0,195171
0,35	4,68	4,77	4,82	4,60	4,71750	0,097425
0,50	5,00	4,60	4,50	5,00	4,77500	0,262996
1,00	5,96	5,46	5,22	5,24	5,47000	0,344287

Levando em consideração o tempo gasto para percorrer as distâncias mostradas na Tabela 4.1 e a relação entre φ_{AT} e φ_{REF} dada pela equação 2.2, obtém-se a Tabela 4.2, ilustrada graficamente através da Figura 4.2.

Tabela 4.2 – Dados manipulados do primeiro experimento

φ_{REF}	V1(m/s)	V2(m/s)	V3(m/s)	V4(m/s)	Média	Desv. Pad
1,432°	0,589286	0,580357	0,660714	0,532143	0,590625	0,053041
2,864°	1,296429	1,160714	1,303571	1,300000	1,265179	0,069704
5,012°	1,671429	1,703571	1,721429	1,642857	1,684821	0,034795
7,160°	1,785714	1,642857	1,607143	1,785714	1,705357	0,093927
14,32°	2,128571	1,950000	1,864286	1,871429	1,953571	0,122960

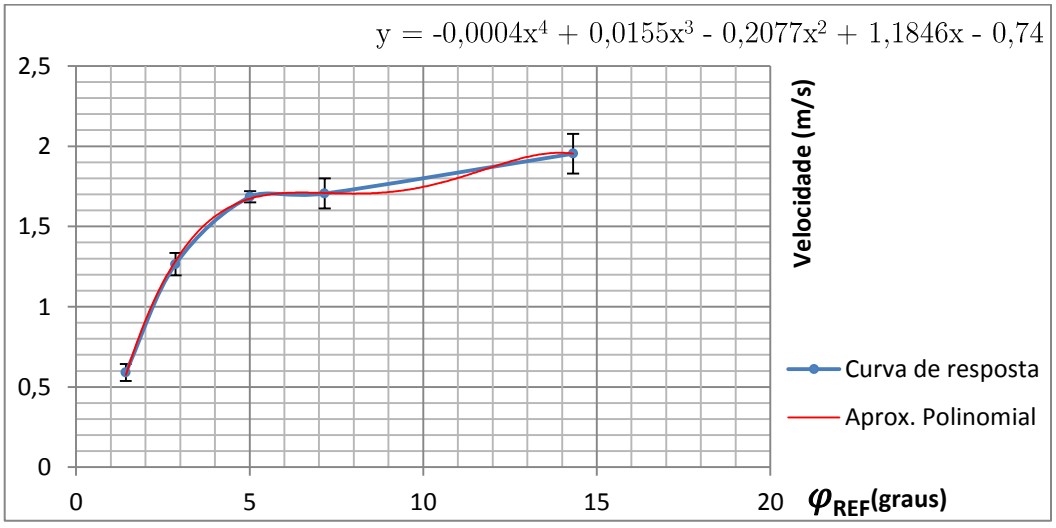


Figura 4.2 – Gráfico do Ângulo de Referência vs. Velocidade de Translação do drone.

Abaixo estão ilustrados na Figura 4.3 o gráfico da velocidade de translação vs. ângulo de referencia e sua aproximação polinomial, pois serão utilizados posteriormente.

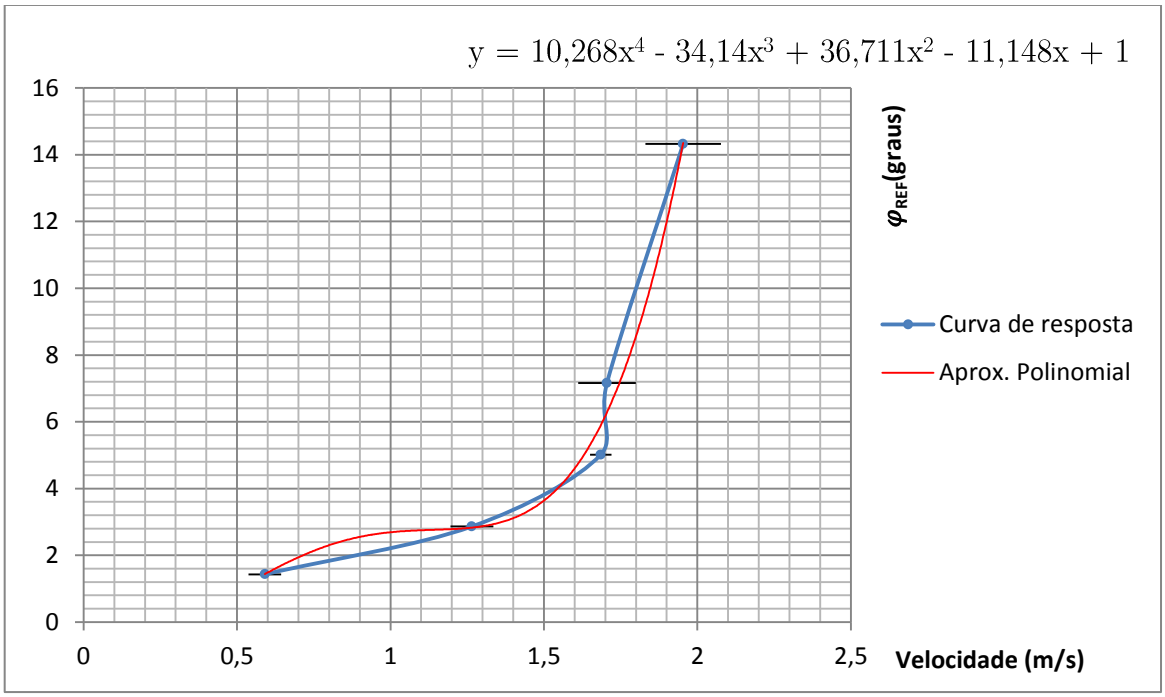


Figura 4.3 – Gráfico da Velocidade de Translação do drone vs. Ângulo de Referência.

4.3 FUNÇÃO DE TRANSFERÊNCIA

Para obtenção da função de transferência do sistema, onde a entrada é dada pelo ângulo de referência enviado ao quadrirrotor (arfagem ou rolamento) e a saída é o ângulo de inclinação em resposta ao comando descrito na sessão 2.4 do segundo capítulo deste trabalho através das equações 2.1, 2.2, 2.3 e 2.4, foi realizado o experimento detalhado a seguir.

Foram utilizados os valores dos sensores internos do drone para o ângulo de rolamento φ (pela simetria, para movimentos em θ a modelagem seria idêntica), que retorna um valor em ponto flutuante entre -1 e 1, simbolizando um ângulo entre -90° e 90° teoricamente, isso porque manualmente (girando o drone na mão) ele pode assumir valores entre -90° e 90°, porém, através dos comandos enviados, o máximo ângulo de arfagem ou rolamento é 14,32°, como explicitado na seção 2.4 desse trabalho.

Novamente utilizando o tempo de ciclo CT = 0,056s, nesse caso funcionando como um tempo de amostragem do sistema discreto, a cada ciclo o programa realiza uma medida de inclinação (φ) do drone, de acordo com uma entrada degrau de amplitude φ_{REF}, nos dando assim a resposta da saída do sistema no tempo.

Foram realizadas, em um intervalo de 3,36 segundos, três coletas de dados do sensor de inclinação φ com um tempo de amostragem CT igual a 0,056s, para entradas φ_{AT} = 0,3 e 0,5 e 1,0, equivalentes a φ_{REF} = 4,296°, 7,16° e 14,32° respectivamente. As respostas no tempo para a entrada degrau φ_{REF} = 4,296° estão ilustradas na Figura 4.4.

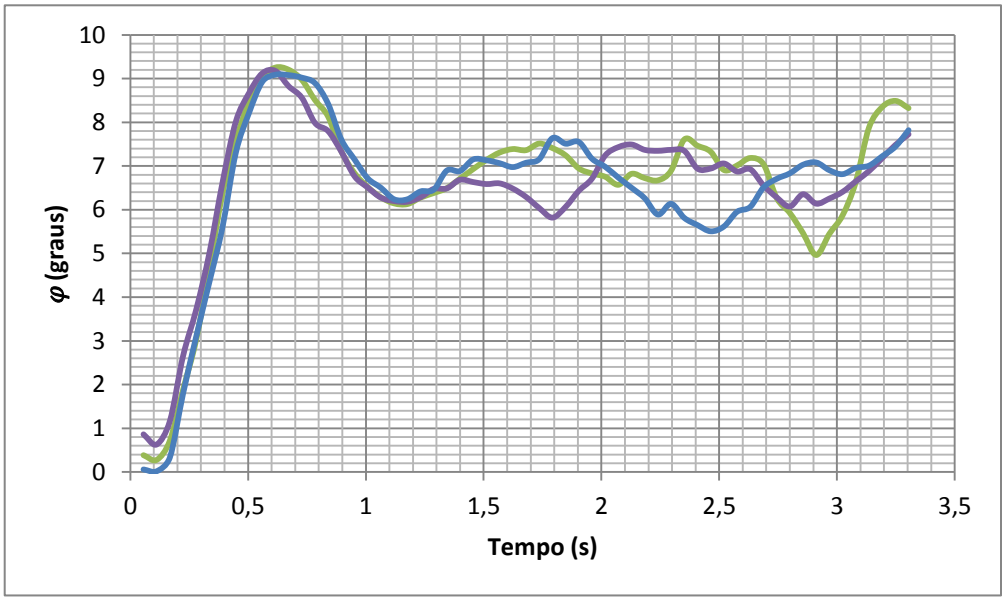


Figura 4.4 – Resposta no tempo para uma entrada φ_{REF} = **4,296°**.

A resposta transitória se mostrou constante nas três coletas de dados para a resposta no tempo, porém, a resposta em regime permanente se mostrou oscilante de forma aleatória, isso se explica pelo fato do sistema ser muito susceptível a diversas interferências. Mesmo com essas oscilações aleatórias, a média dos valores é quase constante a partir de um segundo, como ilustra a Figura 4.5.

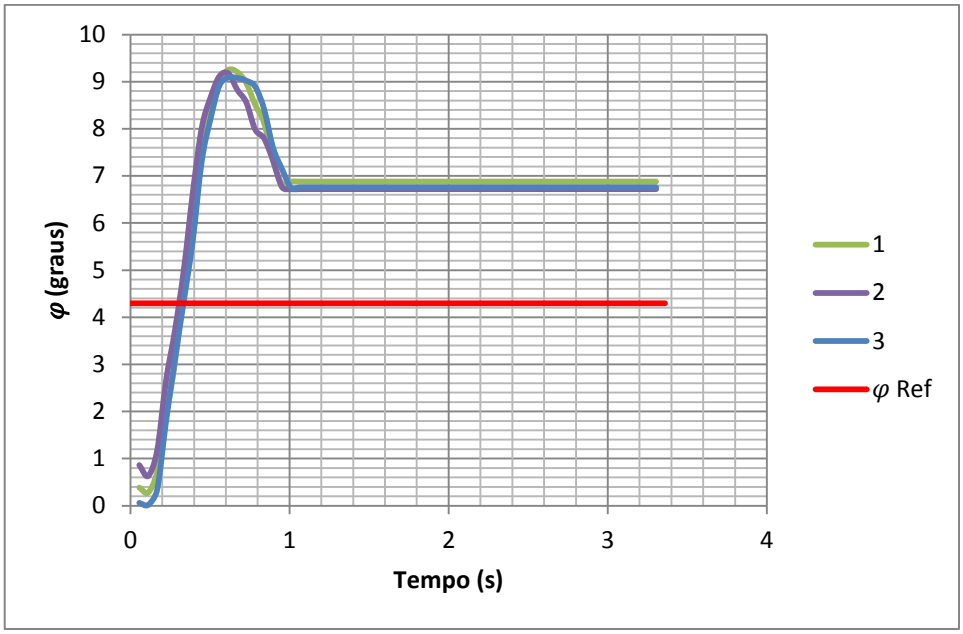


Figura 4.5 – Resposta no tempo para uma entrada $\varphi_{\text{REF}} = 4{,}296°$ com média em regime permanente.

Além da análise para uma entrada $\varphi_{\text{REF}} = 4{,}296°$ também foram levantadas as respostas para $\varphi_{\text{REF}} = 7{,}16°$ e $\varphi_{\text{REF}} = 14{,}32°$, como dito anteriormente. Os resultados foram semelhantes aos com a entrada $\varphi_{\text{REF}} = 4{,}296°$, diferenciando-se em módulo obviamente, porém, pode-se observar algo não previsto anteriormente. Essas respostas são ilustradas nas Figuras 4.6 e 4,8, já com a média em regime permanente nas Figuras 4.7 e 4,9.

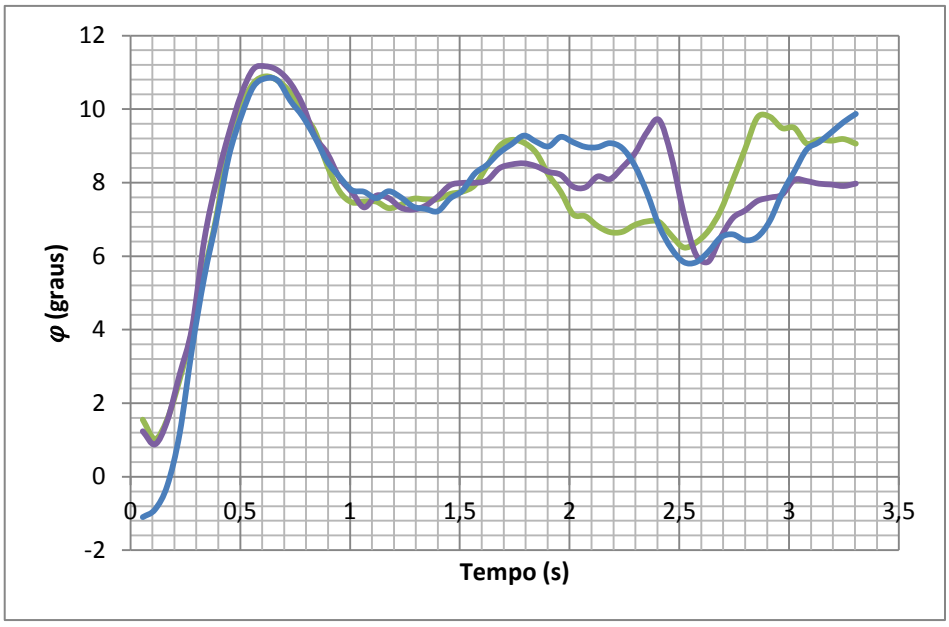


Figura 4.6 – Resposta no tempo para uma entrada $\varphi_{\text{REF}} = 7{,}16°$.

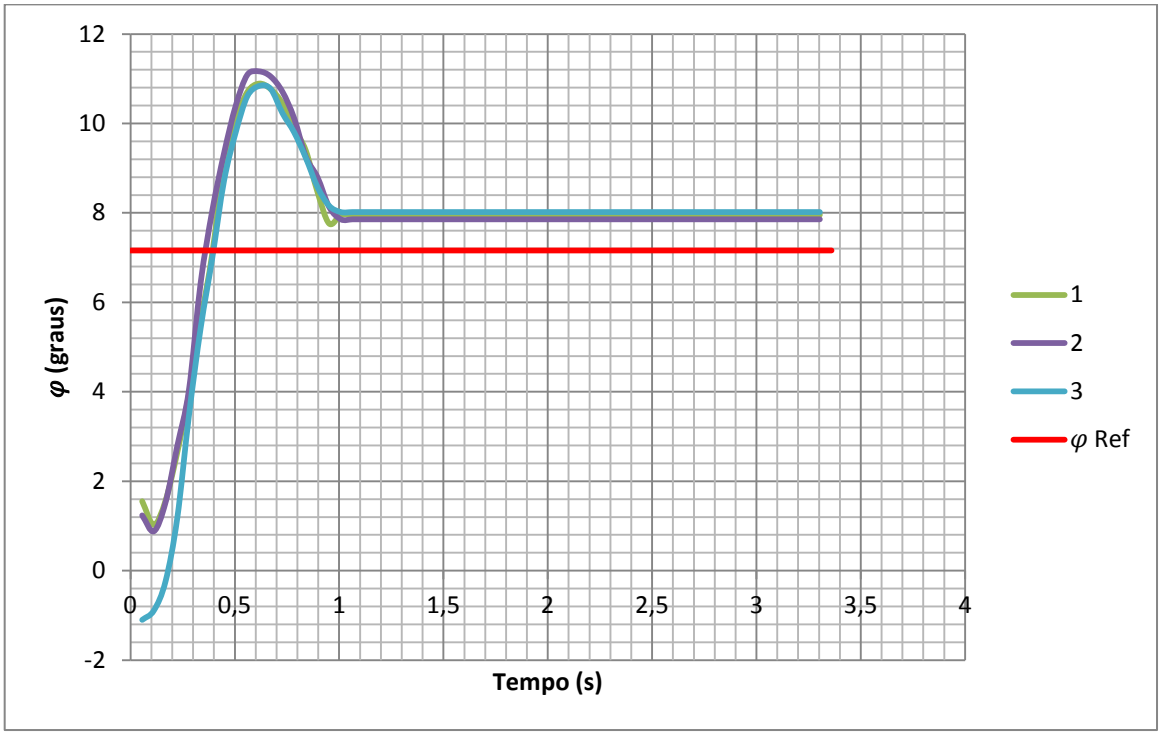


Figura 4.7 – Resposta no tempo para uma entrada $\varphi_{\text{REF}} = 7{,}16°$ com média em regime permanente.

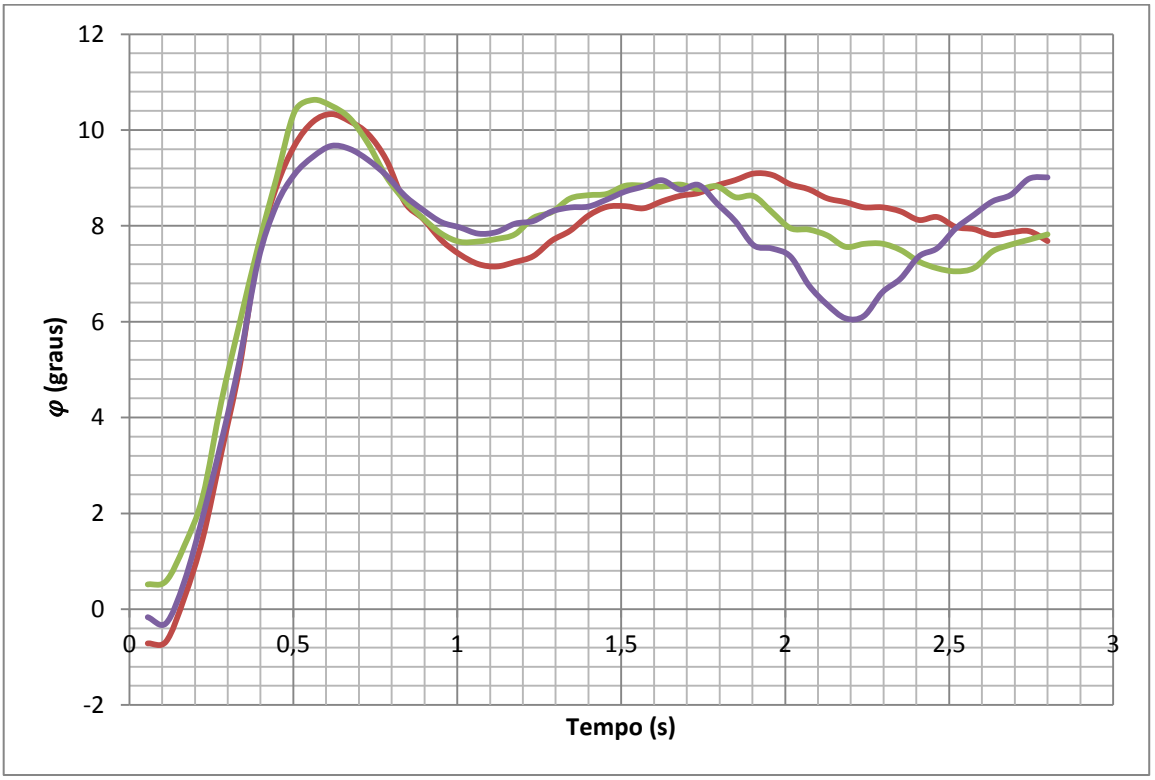


Figura 4.8 – Resposta no tempo para uma entrada $\varphi_{\text{REF}} = 14{,}32°$.

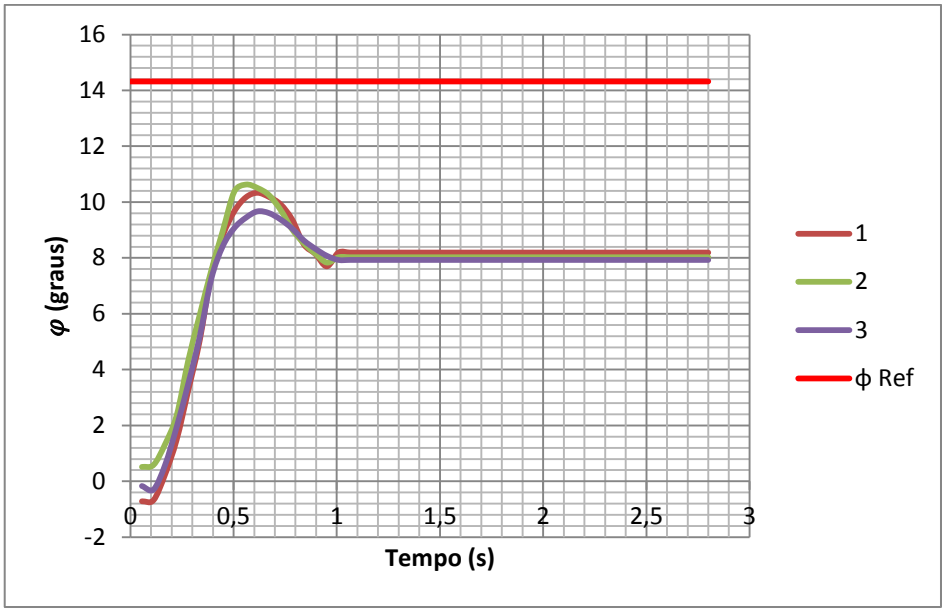


Figura 4.9 – Resposta no tempo para uma entrada $\varphi_{\mathrm{REF}} = 14{,}32°$ com média em regime permanente.

A diferença foi observada ao realizar o experimento para uma entrada $\varphi_{\mathrm{AT}} = 0{,}5$ e 1,0, respectivamente $\varphi_{\mathrm{REF}} = 7{,}16°$ e 14,32°. Observou-se que, para os valores de entrada φ_{REF} igual a 7,16° e 14,32°, a resposta teve a mesma média de saída em $\varphi = 8°$. Logo, 8° é o máximo ângulo de arfagem e rolamento observados experimentalmente.

Levando em consideração, o ganho em regime permanente para a entrada $\varphi_{\mathrm{REF}} = 4{,}296°$ igual a $\frac{6{,}8°}{4{,}296°} = 1{,}583$ e o ângulo máximo encontrado experimentalmente de 8°, tem-se que $\varphi_{REF} = \frac{8°}{1{,}583} \approx 5°$ equivalente a $\varphi_{AT} = \frac{5°}{14{,}32°} \approx 0{,}35$ é a entrada máxima que surte efeito sobre o drone. Isso pode ser constatado observando a Figura 4.2. Assim, a faixa de operação da entrada do sistema deve estar entre $\varphi_{\mathrm{REF}} = -5°$ e $\varphi_{\mathrm{REF}} = 5°$.

Depois de obter a resposta no domínio do tempo do sistema, torna-se simples a obtenção da função de transferência.

Uma função de transferência de segunda ordem padrão pode modelar de forma razoável o sistema. Foi utilizada a seguinte forma padrão:

$$G(s) = k\frac{\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2} \tag{4.30}$$

A Figura 4.10 ilustra uma resposta no domínio do tempo para um sistema de segunda ordem e uma entrada do tipo degrau. Serão utilizados o sobressinal M_p e o instante de pico t_p para encontrar os valores da frequência natural de oscilação do sistema ω_n e da constante de amortecimento ξ, através das equações 4.31 e 4.32.

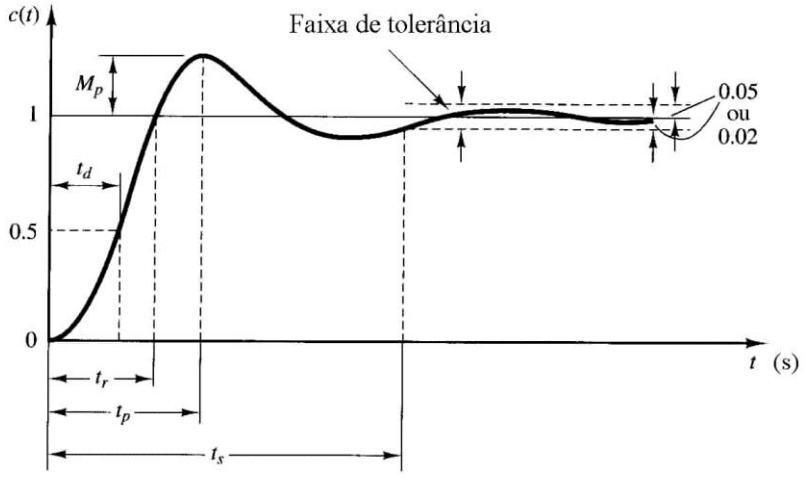


Figura 4.10 – Resposta no tempo para uma entrada degrau com parâmetros ilustrados. [12]

$$M_p = e^{\frac{-\xi\pi}{\sqrt{1-\xi^2}}} = \frac{c(t_p)-c(\infty)}{c(\infty)} \tag{4.31}$$

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\xi^2}} \tag{4.32}$$

Experimentalmente, através das respostas obtidas e ilustradas nas Figuras 4.5, 4.7 e 4.9, obtêm-se os dados da Tabela 4.3.

Tabela 4.3 – Dados obtidos a partir das respostas ao degrau.

φ_{AT}	M_p	t_p
0,3	$M_p \approx \frac{9{,}2° - 6{,}8°}{6{,}8°} \approx 0{,}353$	$t_p \approx 0{,}65s$
0,5	$M_p \approx \frac{11° - 8°}{8°} \approx 0{,}375$	$t_p \approx 0{,}65s$
1,0	$M_p \approx \frac{10{,}8° - 8°}{8°} \approx 0{,}350$	$t_p \approx 0{,}65s$

[12] http://www.lee.eng.uerj.br/~jpaulo/Contri/sistemas-de-segunda-ordem.html

Foi utilizado um sobressinal médio $M_{pm} \approx 0{,}36$.

Isolando ξ na equação 4.31, obtém-se a equação 4.33.

$$\xi = \frac{-\ln M_p}{\sqrt{\pi^2 + (\ln M_p)^2}} \tag{4.33}$$

Substituindo M_p por $\boldsymbol{M_{pm}}$, obtém-se $\xi \approx 0{,}31$.

Substituindo o valor de ξ e t_p na equação 4.32, obtém-se $\omega_n = 5{,}08\, rad/s$.

Por fim, substituindo o valor de ξ e ω_n na equação 4.30, obtém-se a função de transferência do sistema:

$$G(s) = \frac{\varphi(s)}{\varphi_{REF}(s)} = k\frac{25{,}83}{s^2 + 3{,}14s + 25{,}83} \tag{4.34}$$

Utilizando um tempo de amostragem CT = 0,056s, como já explicitado anteriormente, obtém-se também através do MATLAB a função de transferência a tempo discreto do sistema, indicada na equação 4.35, utilizando o método com segurador de ordem zero (*ZOH*), além da resposta ao degrau unitário ilustrada na Figura 4.11.

$$H(z) = \frac{\varphi(z)}{\varphi_{REF}(z)} = k\frac{0{,}03797z + 0{,}03581}{z^2 - 1{,}765z + 0{,}8388} \tag{4.35}$$

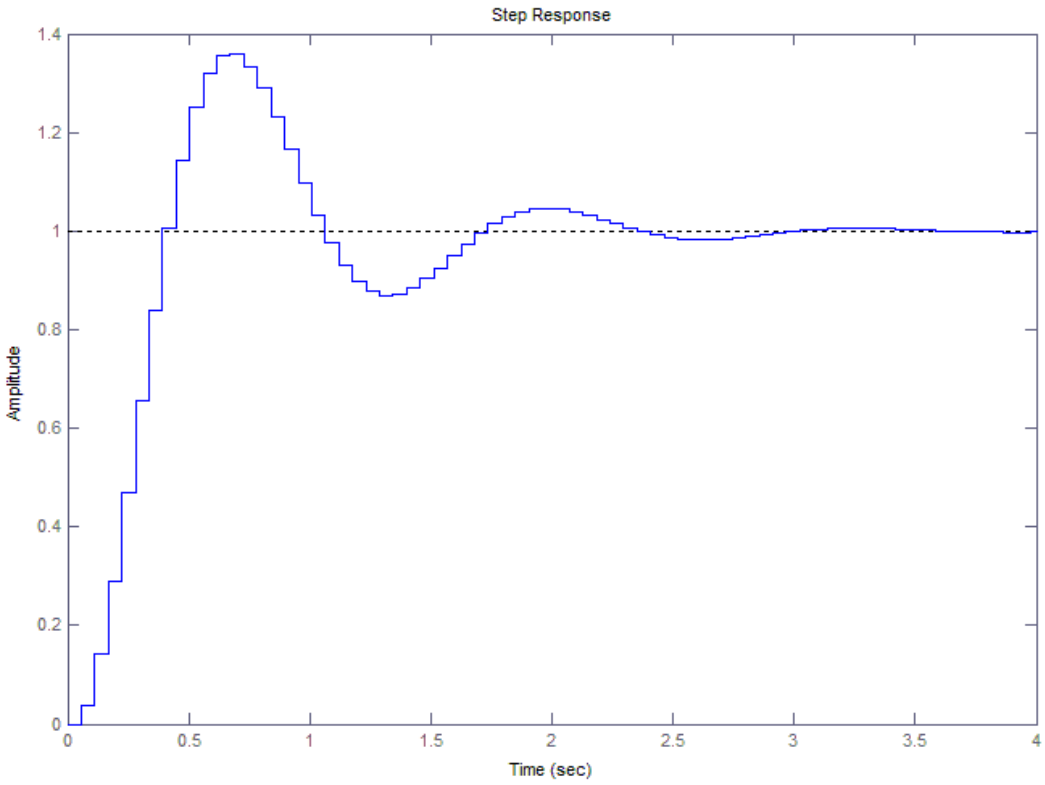


Figura 4.11 – Resposta do sistema à entrada degrau unitário.

Porém, o ganho em regime permanente para essa função de transferência é unitário, ou seja, o erro em regime estacionário é zero, o que não acontece na prática como vemos nas Figuras 4.5, 4.7 e 4.9. Deve-se realizar uma aproximação então e considerar o ganho em regime igual ao ganho para a entrada $\varphi_{\mathrm{REF}} = 4{,}296°$ calculado anteriormente que se encontra dentro da faixa de operação do sistema. Assim, a nova função de transferência a tempo discreto (equação 4.36) e a nova resposta para a entrada $\varphi_{\mathrm{REF}} = 4{,}296°$ (Figura 4.12) são:

$$G(z) = 1{,}583\,\frac{0{,}03797z+0{,}03581}{z^2-1{,}765z+0{,}8388} = \frac{0{,}06011z+0{,}05668}{z^2-1{,}765z+0{,}8388} \tag{4.36}$$

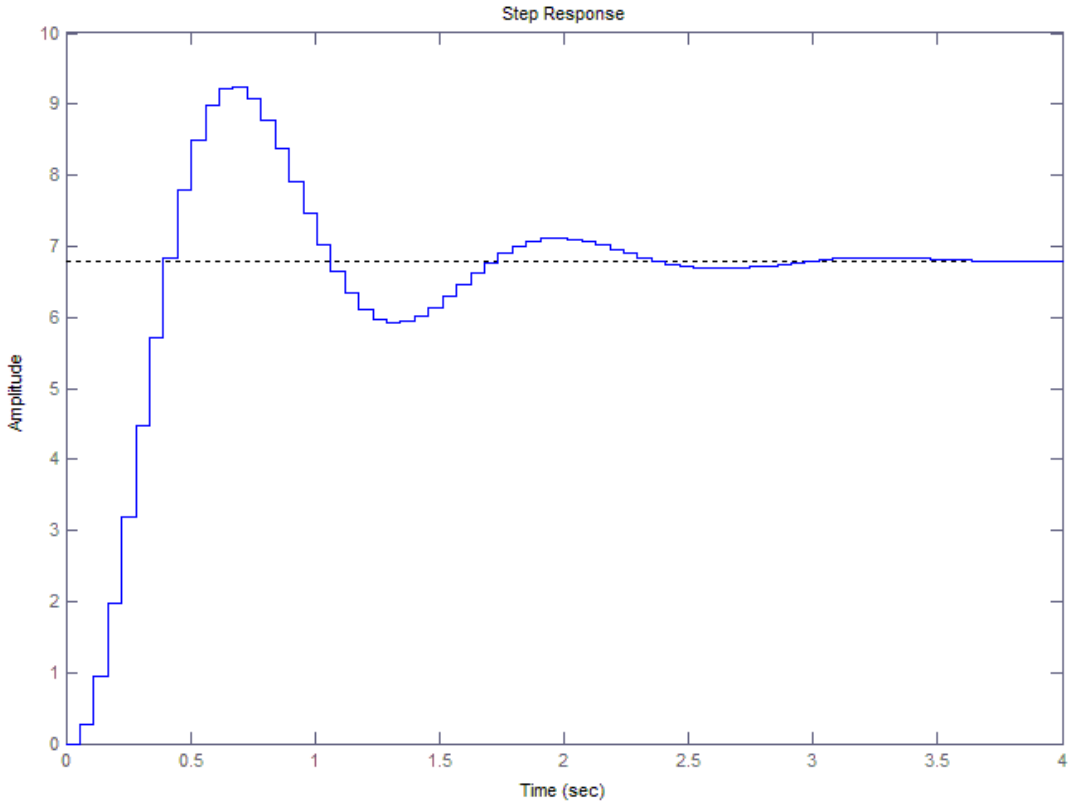


Figura 4.12 – Resposta do sistema com ganho não unitário à entrada degrau de amplitude 4,296°.

4.4 SISTEMA A SER CONTROLADO

A função de transferência obtida experimentalmente é tratada como função de transferência a malha aberta do sistema, afinal, mesmo sabendo que em mais baixo nível o quadrirrotor realiza um controle, o objetivo desse trabalho é justo aprimorar esse controle adicionando um controlador servo-visual externamente ao sistema.

A ideia então é utilizar o vetor de deslocamento obtido pelos algoritmos de fluxo óptico para realimentar o sistema. O modelo à malha fechada do sistema é bem ilustrado pelo diagrama de blocos da Figura 4.13.

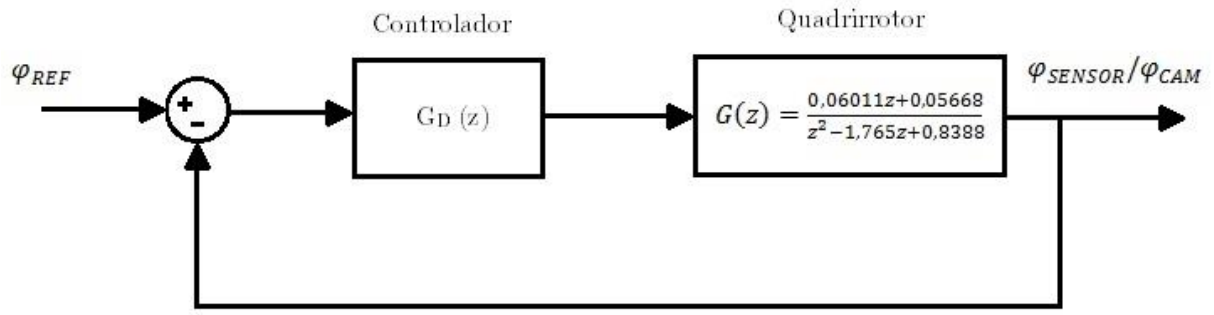


Figura 4.13 – Diagrama de blocos do sistema com controlador.

φ_{REF} é dado através da relação descrita na equação 2.2, G_D (z) é o controlador que será projetado a fim de melhorar a resposta do sistema, φ_{SENSOR} é obtido multiplicando-se o valor de inclinação φ por 90° e por fim, φ_{CAM} é obtido da seguinte maneira:

- Multiplica-se o valor em pixels do vetor de deslocamento do algoritmo de fluxo óptico pela relação mm/pixel, de acordo com a altura do quadrirrotor, definida pela equação 3.9;
- Divide-se o deslocamento em milímetros encontrado pelo tempo de amostragem para obter a velocidade do quadrirrotor para o instante em estudo;
- E finalmente é utilizada a equação polinomial obtida através da relação velocidade de translação vs. inclinação ilustrada na Figura 4.3. Esse valor realimentará o sistema, fechando assim a malha.

4.5 PROJETO DO CONTROLADOR

Decidiu-se realizar dois projetos de controladores, um utilizando o LGR – Lugar Geométrico das Raízes – e outro com resposta *DeadBeat*. Para tanto, utilizou-se das ferramentas do MATLAB para auxiliar no projeto e desenvolver controladores mais eficazes. As simulações foram realizadas com o *Simulink*.

4.5.1 PROJETO LGR

Os requisitos para o projeto no LGR são:

$$M_p = 10\% \quad e \quad t_r = 0{,}2\ s,$$

onde M_p é o máximo sobressinal definido pela equação 4.31 e t_r é o tempo de subida definido por:

$$t_r = \frac{\pi - \cos^{-1}\xi}{\omega_n\sqrt{1-\xi^2}} \tag{4.37}$$

A partir dos requisitos de projeto, utilizando as equações 4.33 e 4.37, é encontrado um coeficiente de amortecimento $\xi = 0{,}59$ e uma frequência natural $\omega_n = 13{,}635$.

$$s = -\xi\omega_n \pm j\omega_n\sqrt{1-\xi^2} \tag{4.38}$$

De acordo com a equação tem-se que os pólos desejados de malha fechada a tempo contínuo são:

$$s = -8{,}045 \pm j11{,}000$$

Logo, sabendo que:

$$z = e^{sT}, \tag{4.39}$$

onde T é o tempo de amostragem do sistema.

Os pólos desejados de malha fechada a tempo discreto são:

$$z = 0{,}520 \pm j0{,}368$$

Foi desenhado então o LGR do sistema sem controlador (Figura 4.14), que considera o sistema com realimentação como ilustrado na Figura 4.13, porém, só

variando o ganho. Percebe-se que o sistema é estável para pequenos ganhos, mas o LGR é em sua maior parte instável, pois se encontra fora do circulo de raio unitário.

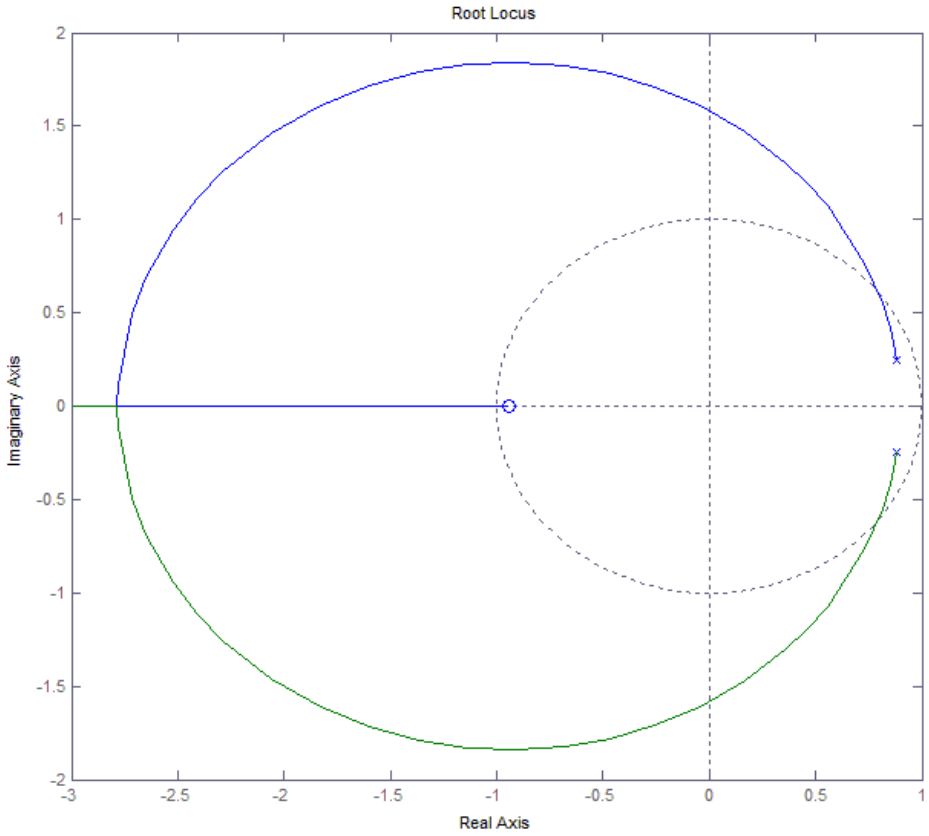


Figura 4.14 – LGR do sistema sem controlador.

O diagrama do *Simulink* do sistema em malha fechada sem controlador é ilustrado na Figura 4.15.

Para o ganho unitário, o sistema é estável, porém, se aproxima do circulo de raio unitário, deixando-o bem oscilante, como ilustra a Figura 4.16 com a resposta ao degrau do sistema com realimentação unitária e controlador proporcional unitário.

Para projetar o controlador, foi utilizada a ferramenta *sisotool* do MATLAB, que serve para auxiliar no projeto de controladores no LGR e na frequência.

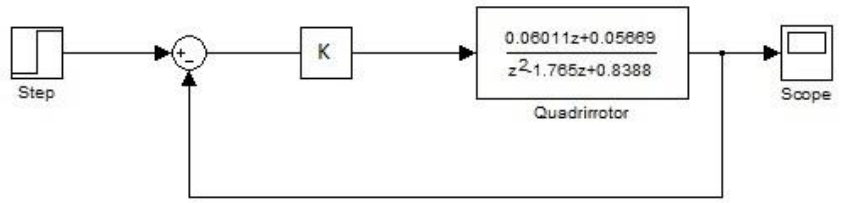


Figura 4.15 – Diagrama do *Simulink* para o sistema em malha fechada sem controlador.

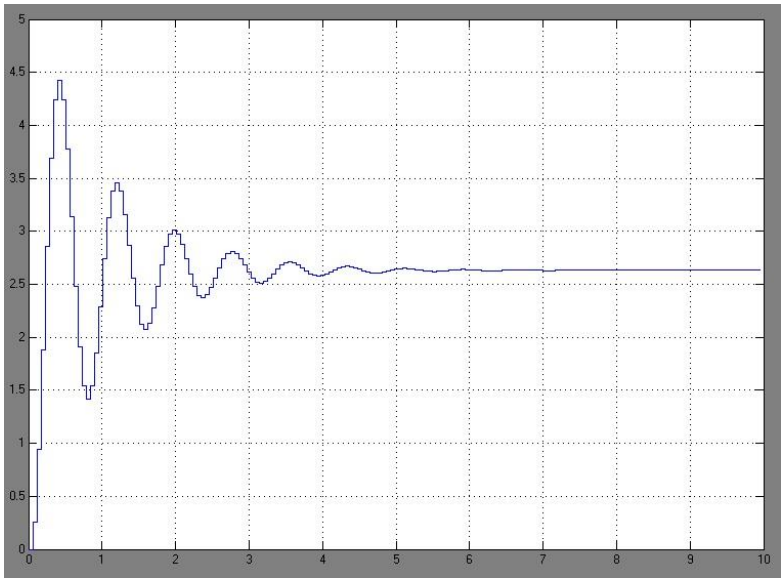


Figura 4.16 – Resposta ao degrau de amplitude 4,296° em malha fechada sem controlador.

Em posse dos pólos desejados e da função de transferência à malha aberta do sistema, foi projetado então um controlador, com função de transferência dada pela equação 4.40.

$$G_D(z) = \frac{Y(z)}{U(z)} = 14{,}2\,\frac{(z-0{,}684)(z-0{,}824)}{(z-1)(z+0{,}984)} = \frac{14{,}2z^2-21{,}4136z+8}{z^2-0{,}0527z-0{,}948} \qquad (4.40)$$

Para implementação no programa foi utilizada a equação de diferenças do controlador dada por: (4.41)

$$y[k] = 0{,}052y[k-1] + 0{,}948y[k-2] + 14{,}2u[k] - 21{,}4136u[k-1] + 8u[k-2],$$

onde $\boldsymbol{y}$ representa a saída do controlador (entrada da função de transferência do quadrirrotor), $\boldsymbol{u}$ a entrada ($\varphi_{\mathrm{REF}} - \varphi_{\mathrm{CAM}}$), k representa o instante atual, [k-1] o instante anterior e [k-2] dois instantes atrás.

A Figura 4.17 mostra o LGR do sistema associado ao controlador projetado. Diferentemente do LGR sem o controlador, o sistema controlado é em sua maior parte estável, pois se encontra dentro do circulo de raio unitário. É possível ver que os requisitos de projeto foram atendidos observando os dados ilustrados na imagem onde mostra ganho unitário, pólos, máximo sobressinal, coeficiente de amortecimento e frequência natural, quase idênticos aos do requisito.

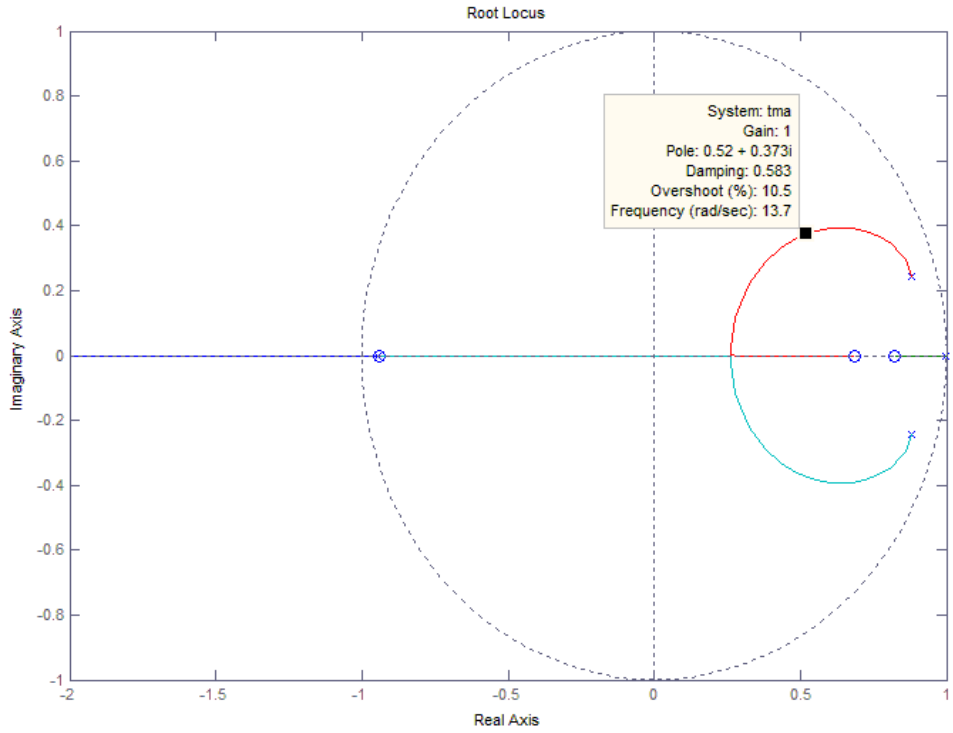


Figura 4.17 – LGR do sistema com o controlador.

O diagrama do *Simulink* é ilustrado pela Figura 4.18 e a resposta ao degrau do sistema controlado é dada pela Figura 4.19. Observando a imagem, fica claro o aumento na velocidade de resposta do sistema com o controlador, já o máximo sobressinal passou de aproximadamente 36% sem controlador para 22% com o controlador, não atingindo exatamente o requisito, porém, tornou-se menor e aceitável.

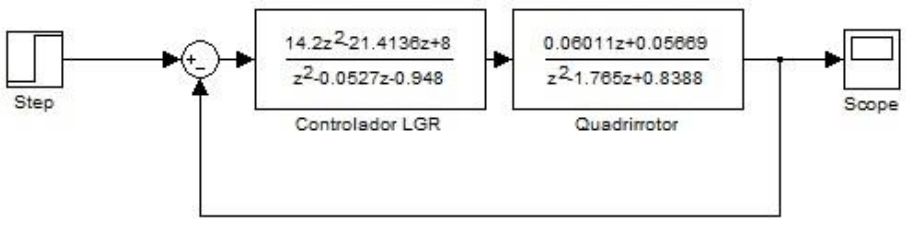


Figura 4.18 – Diagrama do *Simulink* para o sistema em malha fechada com controlador LGR.

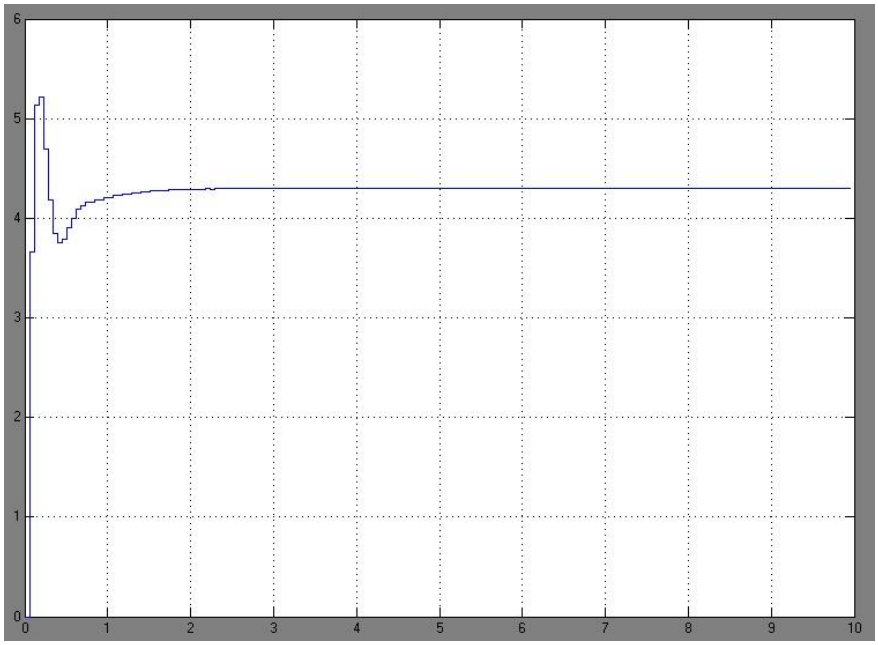


Figura 4.19 – Resposta ao degrau de amplitude 4,296° em malha fechada com controlador LGR.

4.5.2 PROJETO COM RESPOSTA *DeadBeat*

Resposta *DeadBeat* ocorre quando a saída alcança o valor de referência o mais rápido possível e sem oscilações.

Um controlador com resposta *DeadBeat* é dado por:

$$G_D(z) = \frac{1}{G(z)} \frac{M(z)}{1-M(z)}, \tag{4.42}$$

onde $M(z)$ é a função de transferência de malha fechada do sistema controlado, dada por:

$$M(z) = \frac{G_D(z)G(z)}{1+G_D(z)G(z)} \tag{4.43}$$

Assim, foi projetado um controlador que atinge o valor de referência no menor tempo possível e sem sobressinal, dado pela equação (4.44).

$$G_D(z) = \frac{Y(z)}{U(z)} = \frac{z^2-1{,}765z+0{,}8388}{0{,}06011z^2-0{,}00342z-0{,}05669} \tag{4.44}$$

A equação de diferenças do controlador *DeadBeat* é:

$$y[k] = 0{,}0569y[k-1] + 0{,}943y[k-2] + 16{,}636u[k] - 29{,}363u[k-1] + 13{,}954u[k-2] \tag{4.45}$$

O diagrama do *Simulink* e a resposta ao degrau do sistema controlado são ilustrados pelas Figuras 4.20 e 4.21 respectivamente. A simulação ocorre exatamente como proposta, sem sobressinal e com tempo de subida igual a 0,056 segundos que é o menor tempo possível.

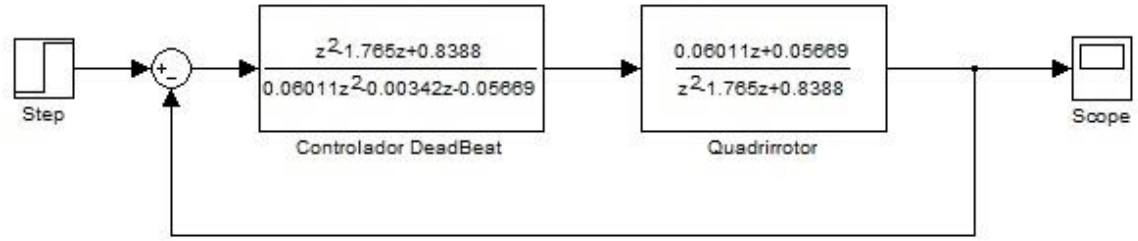


Figura 4.20 - Diagrama do *Simulink* para o sistema em malha fechada com controlador *DeadBeat*.

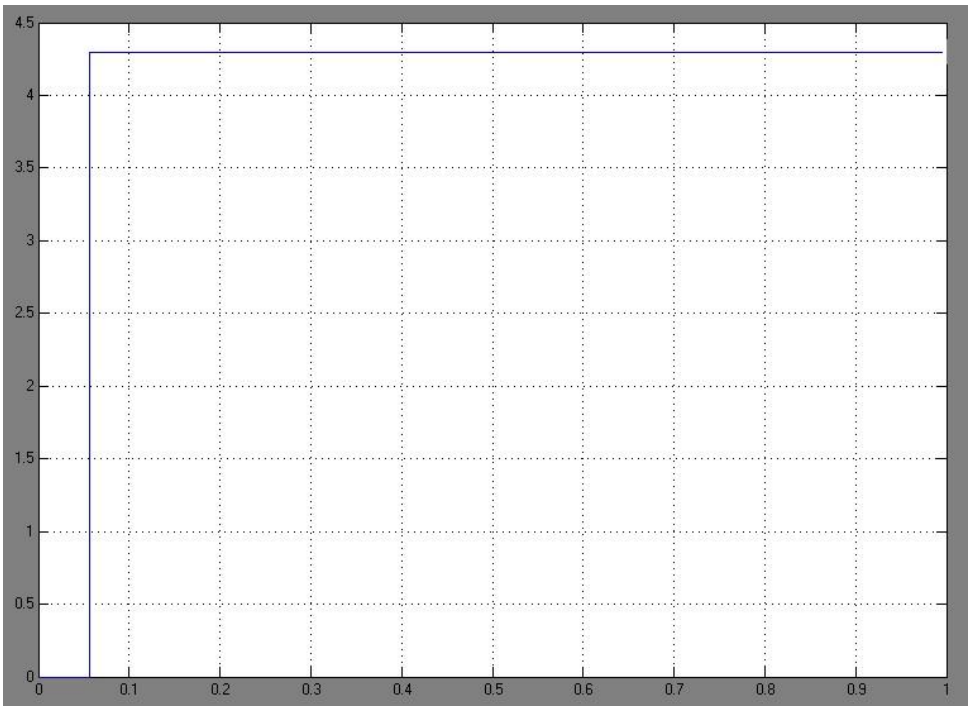


Figura 4.21 – Resposta ao degrau de amplitude 4,296° em malha fechada com controlador *DeadBeat*.

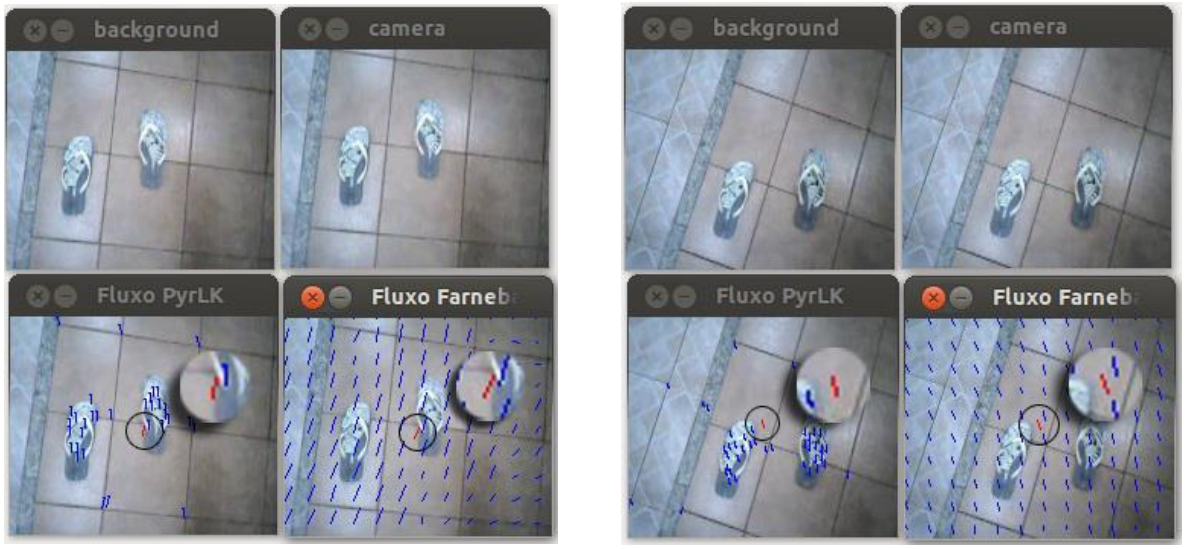
5 TESTES E VALIDAÇÃO

Neste capítulo serão descritos os procedimentos de teste e validação para o sistema controlado, bem como os resultados obtidos.

5.1 TESTES INICIAIS

O primeiro teste realizado na parte prática do projeto foi o de implementar os algoritmos de fluxo óptico a fim de utilizá-los na realimentação do sistema. Foram implementados com êxito os algoritmos de estimação de movimento de Gunnar Farneback e Lucas-Kanade Piramidal. O algoritmo Horn-Schunck foi implementado posteriormente, mas por falta de ajuste dos parâmetros não indicava a direção correta do movimento, tornando-o assim inviável para essa aplicação.

A Figura 5.1 ilustra bem o funcionamento dos dois algoritmos implementados. No centro da imagem de cada um dos fluxos existe um vetor vermelho que indica a média



(a) LK-Piramidal FarneBack (Instante 1)

(b) LK-Piramidal FarneBack (Instante 2)

Figura 5.1 – Algoritmos de estimação de movimentos implementados no quadrirrotor.

dos vetores de deslocamento na imagem, percebe-se que no Instante 1 os vetores de média dos dois algoritmos se assemelham, mas não coincidem, já no Instante 2 eles praticamente se coincidem.

Ao longo desse mesmo teste ficou claro que só havia um algoritmo de estimação dentre os dois que poderia ser utilizado para realimentar o sistema, o algoritmo de Gunnar Farneback, pois mesmo com boa indicação de movimento o algoritmo Lucas-Kanade Piramidal se mostrou muito instável em suas indicações, variando a direção do vetor de deslocamento médio a todo momento, porém, sempre em torno de uma direção e sentido fixos. Essa oscilação fez com que esse algoritmo fosse descartado para a aplicação, tendo em vista que isso causaria instabilidade no voo do drone. Por fim, o algoritmo de Gunnar Farneback foi o utilizado na realimentação.

5.2 PROCEDIMENTO DE VALIDAÇÃO

Para validação dos dados obtidos através do algoritmo de fluxo óptico, foi realizado um breve comparativo entre os dados de voo obtidos pelo algoritmo e os dados dos sensores do drone. Para isso utilizou-se de alguns dos procedimentos descritos na seção 4.4 deste trabalho. Os resultados para esse experimento de validação é ilustrado através da Figura 5.2.

Para realizar o procedimento de validação dos controladores projetados primeiramente eles tiveram que ser implementados, através de uma função no programa em linguagem C. Para tanto, utilizou-se as equações de diferenças dos controladores LGR e *DeadBeat* 4.41 e 4.45 respectivamente.

Após a implementação dos controladores, pôde-se, realizar o mesmo procedimento descrito na seção 4.3 para obtenção da resposta ao degrau de amplitude $\varphi_{\text{REF}} = 4{,}296°$ com cada um dos controladores e com a malha fechada sem controlador. O objetivo desse procedimento é obter resultados práticos, para comparar com os teóricos, obtidos através da simulação no *Simulink*, ilustrados nas Figuras 4.16, 4.19 e 4.21.

Três coletas de dados foram realizadas novamente para cada uma das respostas a malha fechada sem controlador, com controlador LGR e controlador com resposta *DeadBeat.* Ilustradas pelas Figuras 5.3, 5.4 e 5.5 respectivamente.

5.3 RESULTADOS

Com os procedimentos de validação descritos anteriormente, foram obtidos resultados que serão mostrados nesta seção.

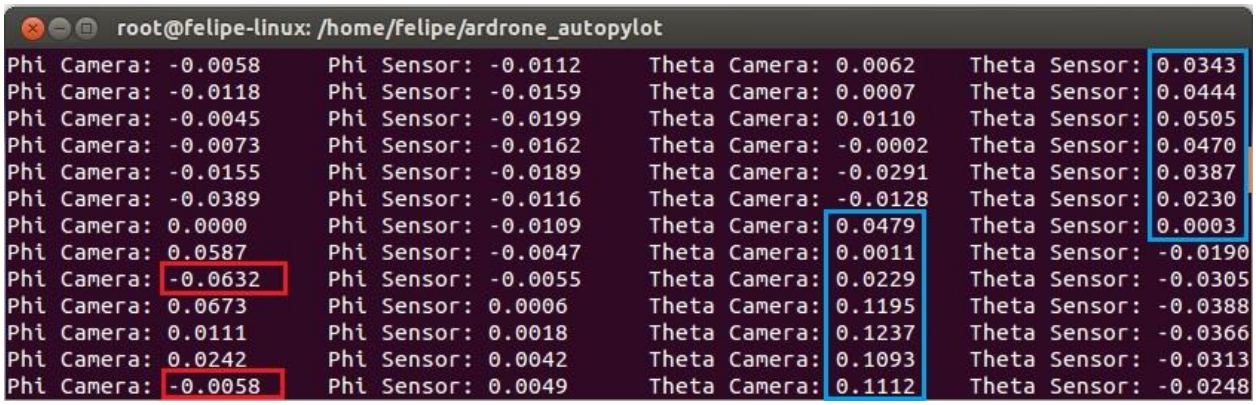


Figura 5.2 – Comparativo entre os dados obtidos pela câmera e os dados do sensor.

A Figura 5.2 mostra que os dados dos ângulos (-1 a 1) obtidos com a câmera e os mesmos dados obtidos através do sensor inercial do drone são em geral equivalentes em sinal, com exceção aos dados destacados pelos retângulos vermelhos, por exemplo, que são disparidades devido ao erro do algoritmo de detecção de movimento, e em módulo se assemelham na maioria dos pontos, exceto por uma defasagem dos dados da imagem em relação aos dados do sensor, como ilustrado nos blocos de dados dentro dos retângulos azuis. Logo, é factível o uso desses dados na realimentação do sistema.

No procedimento de validação das respostas para o sistema realimentado, foram obtidos resultados satisfatórios que coincidem muito com a simulação.

A resposta ao degrau do sistema em malha fechada com controlador proporcional observada na simulação, como ilustrada na Figura 4.16, é observada também no experimento realizado com o quadrirrotor. A Figura 5.3 mostra os dados obtidos nesse experimento, onde a resposta tem um sobressinal elevado e é bem oscilante, exatamente como previsto pela teoria.

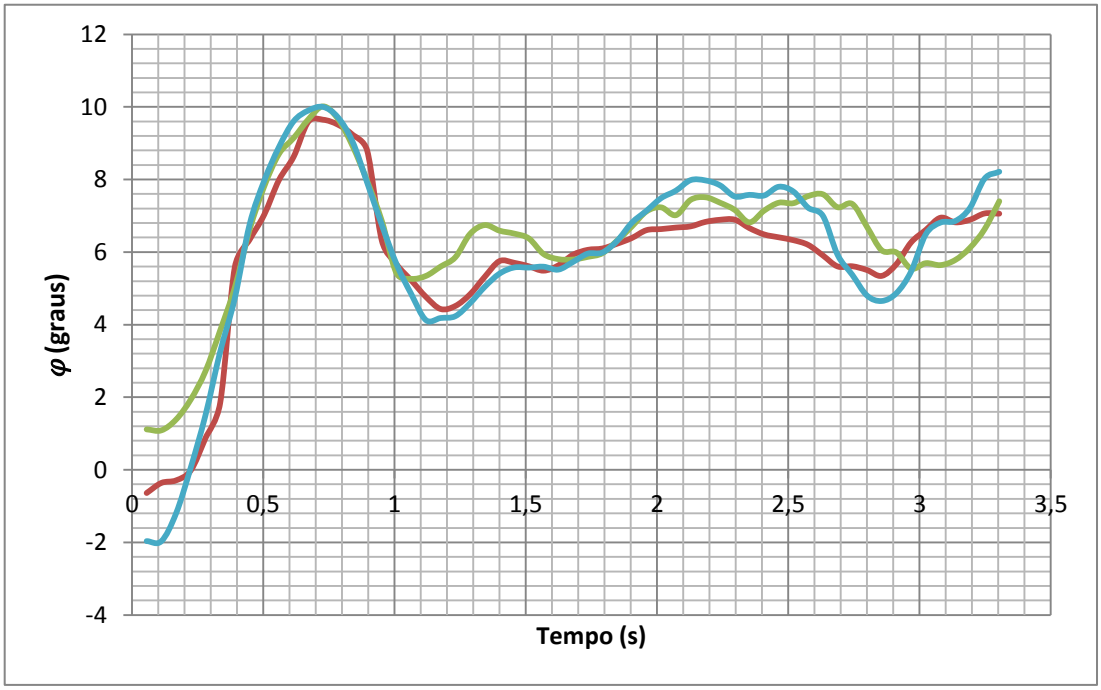


Figura 5.3 – Resposta ao degrau de amplitude $\boldsymbol{\varphi}_{\mathrm{REF}} = 4{,}296°$ no sistema a malha fechada com controlador proporcional.

Para o sistema em malha fechada com o controlador projetado no LGR, as características da resposta ao degrau observadas na simulação, como ilustrada na Figura 4.19, são observadas também no experimento realizado com o quadrirrotor. A Figura 5.4 mostra os dados obtidos nesse experimento. Analisando a resposta experimental é observado um sobressinal de aproximadamente 13%, que chega próximo ao de 10% dos requisitos de projeto, porém o requisito de tempo de subida de 0,2

segundos não foi atendido, isso porque estruturalmente (tempo de resposta dos motores e inércia) o drone não consegue responder de forma mais rápida do que a observada.

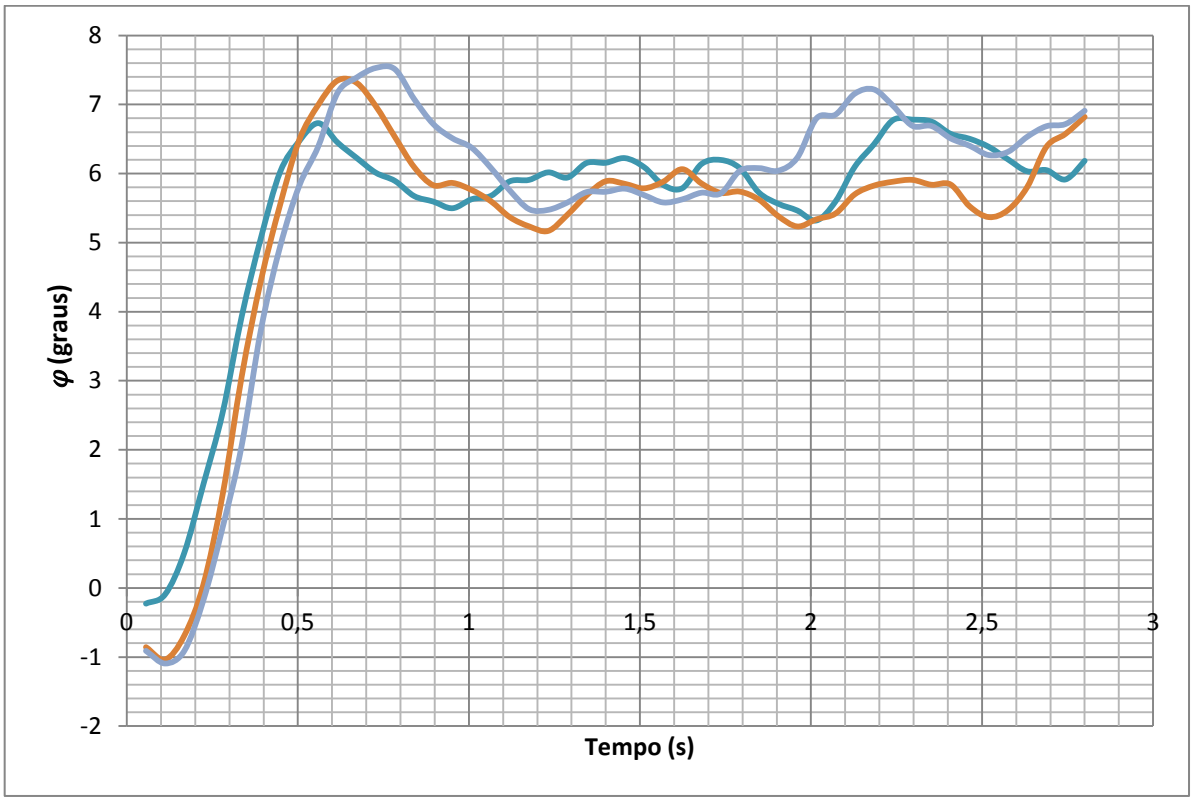


Figura 5.4 – Resposta ao degrau de amplitude $\boldsymbol{\varphi_{\mathrm{REF}} = 4{,}296°}$ no sistema com controlador LGR.

No sistema em malha fechada com o controlador projetado com resposta *DeadBeat*, as características da resposta ao degrau na simulação (Figura 4.21) são: tempo de subida bem pequeno e sem sobressinal. É observado na Figura 5.5 que o tempo de subida continua o mesmo como dito anteriormente, devido a limitações estruturais, e na média não existe sobressinal de fato, pois o pico no inicio que é observado tem a mesma amplitude dos posteriores, significando assim uma oscilação natural do sistema devido às não linearidades.

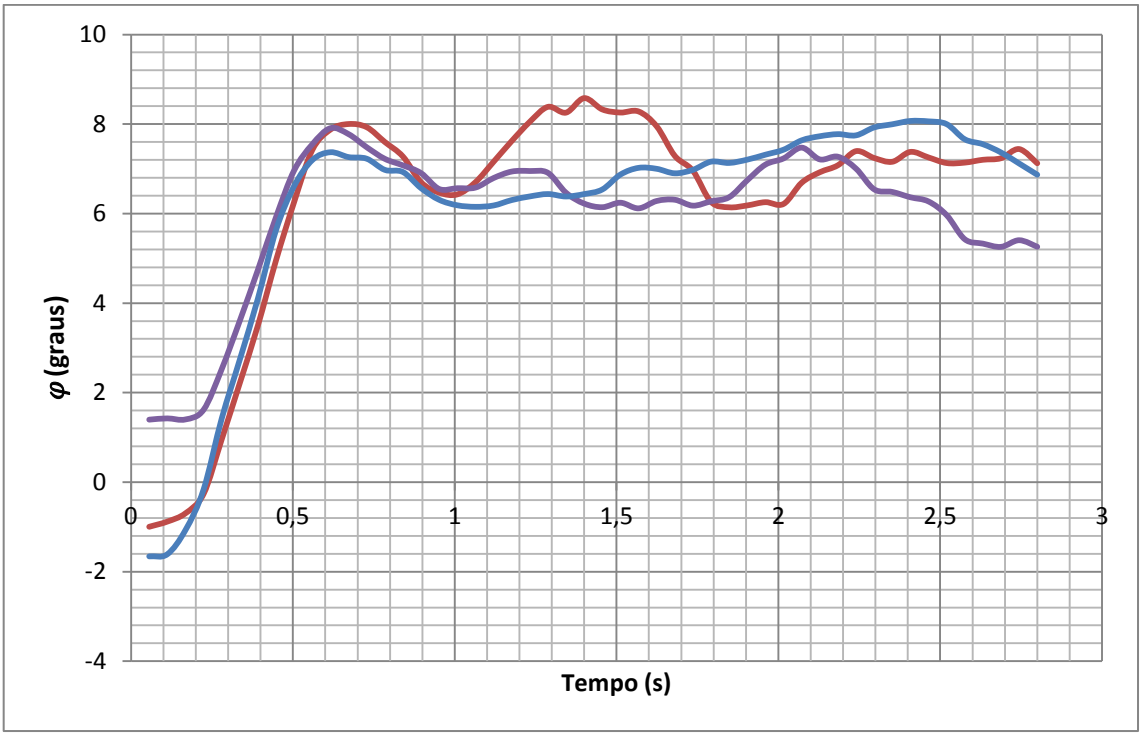
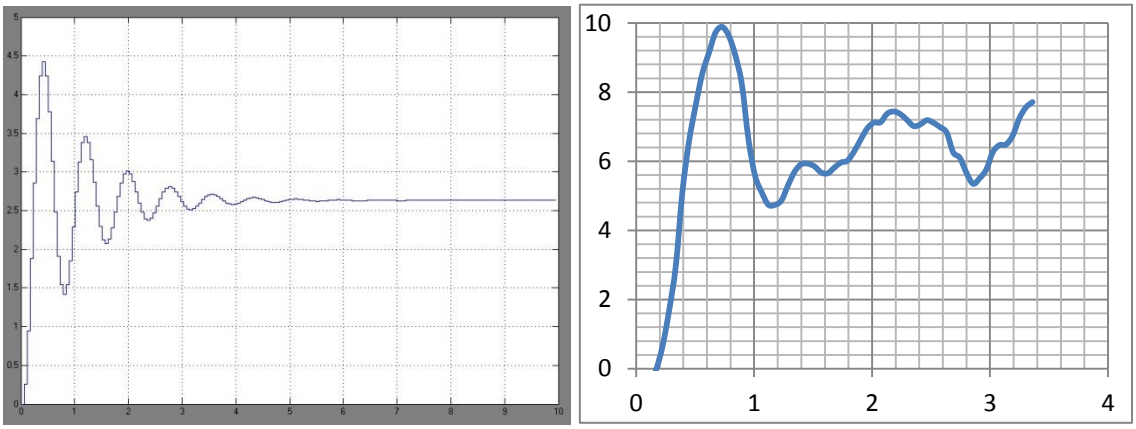


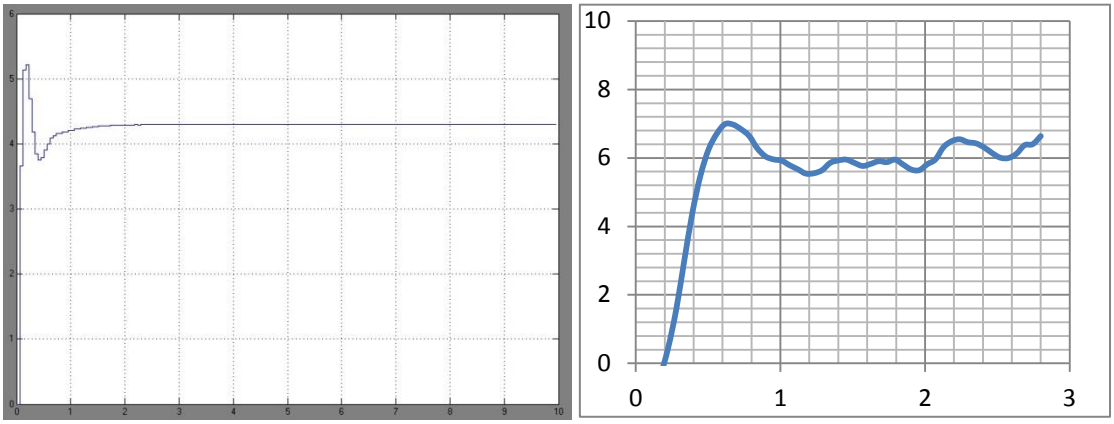
Figura 5.5 – Resposta ao degrau de amplitude $\boldsymbol{\varphi_{\mathrm{REF}} = 4{,}296°}$ no sistema com controlador *DeadBeat*.

Para melhor ilustrar os resultados obtidos, as Figuras 5.6, 5.7 e 5.8 comparam qualitativamente os resultados das simulações com a média dos resultados práticos obtidos para o sistema com realimentação unitária e sem controlador, para o sistema com controlador LGR e para o sistema com o controlador *DeadBeat* respectivamente.



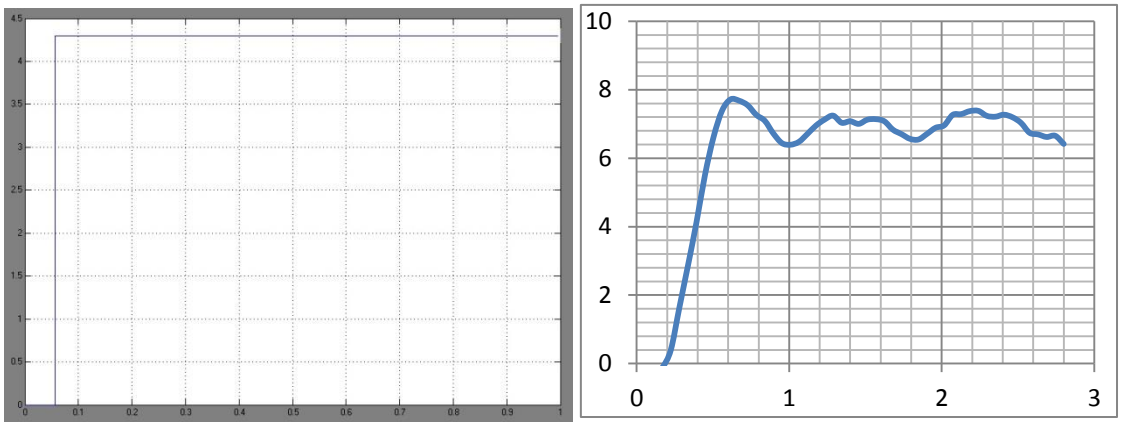
(a) Resultado simulação (b) Resultado prático

Figura 5.6 – Respostas com realimentação e com controlador proporcional unitário.



(a) Resultado simulação (b) Resultado prático

Figura 5.7 – Respostas com realimentação e com controlador LGR.



(a) Resultado simulação (b) Resultado prático

Figura 5.8 – Respostas com realimentação e com controlador *DeadBeat*.

6 CONCLUSÃO

Neste capítulo serão expostas as conclusões do trabalho e comentários finais, além de propor formas de continuação da linha de pesquisa relacionada ao projeto.

6.1 CONCLUSÕES E COMENTÁRIOS FINAIS

Primeiramente, neste trabalho é discutida a importância do uso dos drones na atualidade, sua história, e os objetivos deste trabalho neste âmbito, que é justo melhorar a implementação atual adicionando o controle de posição com realimentação visual, realizado atualmente apenas com sensores inerciais. Nesta conclusão serão descritos os objetivos alcançados ao longo do desenvolvimento do trabalho. Propostas para pesquisas posteriores serão descritas na próxima seção deste capítulo de conclusão.

A plataforma se mostrou muitas vezes instável, em se tratando de robustez mecânica na maioria das vezes. As hélices tiveram que ser trocadas para que o voo se tornasse menos oscilante, elas se deterioraram ao longo de testes realizados anteriormente a este trabalho. No inicio muitos problemas de comunicação surgiram, outros problemas parecidos apareceram até que toda a implementação de software tivesse sido feita da forma mais adequada. A câmera de não tão boa qualidade dificultou a obtenção de um vetor de movimento bem definido utilizando os algoritmos de fluxo óptico, por outro lado sua baixa resolução viabilizou a implementação de algoritmos densos.

O desenvolvimento na área de visão computacional deste trabalho foi muito satisfatório, afinal, o objetivo de extrair um vetor que exprimisse a movimentação real do quadrirrotor utilizando apenas a imagem da câmera inferior do drone foi atingindo.

Infelizmente, o uso de mais algoritmos de detecção de movimentos para realizar a realimentação não foi possível. A correção das distorções na imagem foi determinada via calibração, mas não foi utilizada na implementação do controlador. Já a relação entre um pixel na imagem e seu tamanho real em milímetros foi de extrema importância para toda parte de controle com realimentação visual do sistema, sem essa relação de nada adiantaria um vetor de movimento em pixels do drone.

A modelagem matemática do sistema só serviu no ponto de vista teórico, para estudos posteriores de aprimoramento das técnicas de controle utilizadas neste trabalho. Já a modelagem experimental foi essencial na implementação dos controladores e obtenção dos bons resultados. Não só o levantamento da função de transferência experimental, mas também a obtenção da relação Inclinação vs. Translação foram experimentos bem elaborados que permitiram desenvolver os objetivos do trabalho. O projeto dos controladores só foi possível com algumas aproximações. Sabe-se que o sistema é não linear e que a velocidade de deslocamento do drone não é constante quando submetido a uma entrada degrau, porém, foi necessário aproximá-lo a um sistema linear que se move a velocidade constante no intervalo de amostragem, além de outros ajustes descritos durante o desenvolvimento do trabalho, para que fosse possível atingir os objetivos limitados pelos conhecimentos de graduação.

Os testes e experimentos para validação dos controladores obtiveram resultados muito bons, considerando as aproximações realizadas. Os controladores com realimentação visual, tanto utilizando o LGR quanto com resposta *DeadBeat,* se mostraram eficazes, exceto pelas limitações físicas do drone. A função de transferência obtida experimentalmente que modela o sistema se mostrou coerente com o resultado esperado, de acordo com a teoria, nas respostas do sistema a malha fechada.

Por fim, acredita-se que esse trabalho de graduação tenha atingindo todos os objetivos pré-determinados e que sirva de base para muitas pesquisas futuras na associação de visão computacional ao controle de quadrirrotores.

6.2 TRABALHOS FUTUROS

Para trabalhos futuros, algumas sugestões são listadas abaixo.

- Realizar procedimentos semelhantes ao deste trabalho, porém utilizando os sensores inerciais (giroscópios e acelerômetros) para a realimentação. Após isso, poder-se-ia explorar a fusão das duas técnicas de realimentação, considerando suas vantagens e desvantagens.
- Repetir o processo de projeto com realimentação utilizando a câmera frontal, possibilitando explorar o controle do ângulo de rolamento e a translação no plano dessa câmera.
- Desenvolver técnicas de geração e seguimento de trajetórias, de forma a validar o trabalho realizado em ambientes sujeitos a perturbações externas, como ventos laterais.
- Implementar técnicas de estimação e filtragem com o objetivo de melhorar a estimativa de velocidade.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] **AMAANI, L.** Air Force officials announce remotely piloted aircraft pilot training pipeline. *News*, 9 de Jun, 2010. Disponível em: <http://www.af.mil/news/story.as-p?storyID=123208561>

[2] **NEWCOME, L.** *Unmanned aviation: a brief history of unmanned aerial vehicles*. [S.l.]: AIAA, 2004.

[3] **HULL, L.** Drone makes first UK 'arrest' as police catch car thief hiding under bushes. *MailOnline*, 12 de Fev. 2010. Disponível em: <http://www.dailymail.co.uk-/news/article-1250177/Police-make-arrest-using-unmanned-drone.html>.

[4] **SUGIURA, R**. et al. Field information system using an agricultural helicopter towards precision farming. In: IEEE. *Advanced Intelligent Mechatronics, 2003. AIM 2003. Proceedings. 2003 IEEE/ASME International Conference on*. [S.l.], 2003. v. 2, p. 1073–1078.

[5] **OLIVEIRA F.A.R. E DE ARAUJO DIAS, A**. de. *Verificação de Falhas em Linhas de Transmissão por Redes Neurais Artificiais a partir dos Espectros de Freqüência de Imagens*. Dissertação (Trabalho de graduação) — Universidade de Brasília, 2007.

[6] **MERINO, L**. et al. An unmanned aircraft system for automatic forest fire monitoring and measurement. *Journal of Intelligent & Robotic Systems*, Springer, p. 1–16, 2011.

[7] **CORRÊA, M. JÚNIOR, J.** *Estudos de veículos aéreos não tripulados baseado em sistemas multi-agentes e sua interação no espaço aéreo controlado.* — TG- Universidade de São Paulo, 2008.

[8] **The Ohio Academy of Science**, *HeartLandScience*. First Successful Helicopter, 2005. Disponível em: <http://www.heartlandscience.org/aeroav/helicopter.htm>.

[9] **CALEGARINI, R.** Polícia Militar/PE testa VANT para monitoramento em grandes eventos, *PilotoPolicial,* 29 de Set, 2012. Disponível em: < http://www.pilotopolicial.com.br/policia-militar-testa-aeronave-eletrica-para-monitoramento-em-grandes-eventos/>.

[10] **ARAÚJO, ER, KUSSABA, H**, 2011. Sistemas de Perseguição Baseado em Quadrirrotores. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº 23, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 123p.

[11] **LEISHMAN, J**. Principles of Helicopter Aerodynamics. [S.l.]: Cambridge University Press, 2000.

[12] **SIEGWART, roland.** *Introduction to Autonomus Mobile Robots.* s.l. : The MIT Press, 2004.

[13] **MUÑOZ, GLORIA LILIANA LÓPEZ.** *Análise Comparativa das técnicas de controle servo-visual baseado em posição e controle servo-visual baseado em imagem.* Brasilia : UnB, Dissertação de mestrado-2011.

[14] **Corke, Peter I.** *Visual control of robot manipulators: a review. Visual Servoing.* Singapore : World Sci, 1993, Vol. 7, pp. 1-31.

[15] **Bernades, Mariana.** *Controle Servo-visual para Aproximação de Portas.* Brasilia : UnB, Dissertação de mestrado-2009.

[16] **J. Hill and W. T.Park.** *Real time control of a robot with a mobile camera Proceedings of 9th ISIR*, pages 233 – 246, March 1979.

[17] **BILLS, C.; CHEN, J.; SAXENA, A.** Autonomous MAV flight in indoor environments using single image perspective cues. In: IEEE. Robotics and Automation (ICRA), 2011 IEEE International Conference on. [S.l.], 2011. p. 5776–5783.

[18] **BILLS, C.; YOSINSKI, J.** *MAV stabilization using machine learning and onboard sensors.* [S.l.], 2010.

[19] **NG, W.; SHARLIN, E.** *Collocated interaction with flying robots.* In: IEEE. RO-MAN, 2011 IEEE. [S.l.], 2011. p. 143–149.

[20] **HIGUCHI, K.; SHIMADA, T.; REKIMOTO, J**. *Flying sports assistant: external visual imagery representation for sports training.* In: ACM. Proceedings of the 2nd Augmented Human International Conference. [S.l.], 2011. p. 7:1–7:4.

[21] **PARROT.** AR.Drone Developer Guide. [S.l.], Fevereiro 2011. SDK 1.7. Disponível em https://projects.ardrone.org/.

[22] **PORFIRIO, R.** *Estimação de Pose por Visão Monocular para Aterrissagem de VANTs Utilizando ARTOLKIT com Multimarcas.* Brasilia, jun de 2012. UnB.

[23] **BRADSKI, D. G. R.; KAEHLER, A**. *Learning opencv, 1st edition*. First. [S.l.]: O'Reilly Media, Inc., 2008. ISBN 9780596516130.

[24] **BEVILAQUA, D.** *Desenvolvimento de Sistema de Visão para Movimentação de Robô Móvel Pioneer P3-AT*. TG -Brasília, jul de 2011. UnB

[25] **GONZALES.** *Processamento de Imagens Digitais*. s.l. : Prentice Hall, 2002.

[26] **Shah, Mubarak.** *Fundamental of computer Vision*. s.l. : University of Central Florida, 1997.

[27] **ROSA, L.** *Investigação sobre Algoritmos para a Estimação de Movimento na Compressão de Vídeos Digitais de Alta Definição: Uma Análise Quantitativa*. Pelotas, 2007. TCC - Universidade Federal de Pelotas.

[28] **Marengoni, M; Stringhini, D.** *Tutorial: Introdução à Visão Computacional usando OpenCV*.

[29] **Farneback, G.** *Two-Frame Motion Estimation Based on Polynomial Expansion*. Linkoping, Sweden.

[30] **BRESCIANI, T.** *Modelling, Identification and Control of a Quadrotor Helicopter*. Tese (Mestrado) - Lund University, 2008.

[31] **ENOMOTO, J.** *Controle de Inclinação Utilizando Lógica Fuzzy*. Curitiba, 2008. TCC - Universidade Federal do Paraná.

[32] **MONTEIRO, J. R. B. A.** *Estratégias de Acionamento e Controle em Máquinas CA de Ímã Permanente com Fluxo Não Senoidal*. 1997. 120 f. Dissertação – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 1997.

[33] **TRINDADE, R.** *ESTUDO DE MÁQUINAS ELÉTRICAS NÃO-CONVENCIONAIS: Motor Brushless DC*. Escola de Engenharia de São Carlos da Universidade de São Paulo.TCC São Carlos, 2009.

[34] **GIERAS, J. F.; WING, M**. *Permanent Magnet Motor Technology. First Edition*. New York: Marcel Dekker, Inc, 2002. 589 p.

[35] **McCormick, W. Barnes and W.** *Aerodynamics, Aeronautics and Flight Mechanics. s.l.* : New York: Wiley 2nd editon, 1995.

[36] **VIEIRA, J.** *Plataforma Móvel Aérea QuadRotor.* Universidade do Minho – Escola de Engenharia. Outubro, 2011.

ANEXOS

I. DESCRIÇÃO DO CONTEÚDO DO CD

Os arquivos, diretórios e subdiretórios do CD são divididos da seguinte forma:

- Resumo.pdf
- Trabalho de Graduação – Versao Final.pdf
- Apresentação
- Codigos
    - ardrone_autopylot
    - ARDrone_SDK_2_0
    - OpenCV-2.4.3
- Referências
- Resultados Calibração

II. INSTRUÇÕES PARA O PROGRAMA

Todos os programas foram testados na seguinte plataforma:

- Sistema operacional Ubuntu 12.04, baseado em Linux.
- OpenCV versão 2.4.3.
- Firmware do AR.Drone versão 1.7.2.
- SDK versão 2.0.

Todos os programas desenvolvidos ao longo do projeto estão na pasta "`ardrone_autopylot`" com o prefixo "`autopylot_c_agent`" e de extensão "`.c`". Somente o arquivo "`autopylot_c_agent.c`" é rodado quando é dado o `make` na pasta "`ardrone_autopylot`", logo, é necessário renomear o programa desejado para que o mesmo seja compilado.

O joystick utilizado para ativar os controladores no quadrirrotor foi o do *Sony PlayStation 3*, porém vários outros podem ser utilizados. É necessário apenas configura-los da forma correta por meio do arquivo "`gamepad.c`" conforme descrito no guia do desenvolvedor do SDK[21, pg. 96].