FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

HoverSea: Inspecção de estruturas marítimas utilizando um drone e um catamarã autónomo

Diogo Cardoso

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Nuno ALexandre Lopes Moreira da Cruz

18 de Fevereiro de 2017

Resumo

Hoversea tem como objetivo a realização de missões totalmente autónomas, recorrendo a um drone e a um catamarã. Neste projeto será utilizado o quadricóptero AR Drone 2.0 que deverá ser capaz de levantar voo a partir de uma plataforma presente no catamarã para depois se deslocar até coordenadas GPS previamente definidas por um utilizador e, posteriormente, aterrar de forma segura na plataforma de aterragem. A leitura de sensores e controlo do drone, bem como as bibliotecas de tratamento de imagem, foram realizadas através da utilização do software *Robot Operating System*.

No presente documento é descrito, de forma pormenorizada e exaustiva, todo o processo de identificação do sistema, com explicações detalhadas dos testes efetuados. Apresentam-se ainda duas propostas de algoritmo para a conceção de um controlador proporcional do movimento horizontal e para a fase de aterragem.

Relativamente à plataforma de aterragem, é explanado o processo de escolha do padrão e a sua respetiva deteção.

Abstract

Hoversea purpose is to execute fully autonomous missions with a quadcopter and a catamaran. In this project we use the AR Drone 2.0 that will be able to takeoff from a landing platafrom in the catamaran, fly to pre-defined GPS coordinates and afterwards land in the same platfrom. Relatively to this platform, it will be described the conception of the pattern and the process of detection. To perform the sensor reading, drone movement control and the computer vision processes we will use the software Robot Operating Software.

This document describes the process of identifcation of the drone system and presents a proposal for an algorithm for the design of a proportional controller to control the quadcopter movement and an algorithm for the landing phase.

Agradecimentos

Este espaço é dedicado àqueles que deram a sua contribuição para que esta dissertação fosse realizada. A todos eles deixo aqui o meu agradecimento sincero.

À minha família pela forma como me incentivaram e acompanharam ao longo destes anos.

Ao meu orientador, Professor Nuno Alexandre Lopes Moreira da Cruz, pelo apoio e orientação disponibilizados durante a realização deste trabalho.

Diogo Cardoso

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

ROS	Robot Operating System
RGB	Red Green Blue
ms	Milissegundos
s	Segundos
m	Metros
GPS	Global Position System
Fps	Frames per second

Capítulo 1

Introdução

Com este documento pretende-se, em primeira instância, explicar o processo de elaboração de um algoritmo que possibilite a realização de missões autónomas a partir da utilizaçao de um drone e de um catamarã. Assim sendo, o drone, deverá ser capaz de levantar voo de uma plataforma integrada no catamarã e deslocar-se até coordenadas GPS previamente definidas, com retorno à embarcação, numa aterragem pautada com a máxima segurança.

1.1 Motivação

Verifica-se, atualmente, um interesse crescente na monitorização de ambientes costeiros, face ao impacto negativo das atividades antrópicas [10]. A crescente preocupação, associada à evolução de tecnologias conduziu a novas abordagens numa tentativa de solucionar o referido problema. Cada vez mais a solução passa pela utilização de sistemas autónomos, na medida em que estes permitem a inspeção da costa de um modo mais eficiente, mais económico, mais prático e, consequentemente, permitem aceder a ambientes remotos a que provavelmente nenhum operador humano conseguiria aceder facilmente [6]. Na atualidade, existem projetos que propõem um sistema composto por um *unmanned surface vehicle*, USV, e por um *unmanned aeril vehicle*, UAV [9] [5]. Este sistema demonstra-se particularmente eficiente, uma vez que a cooperação entre estes dois veículos possibilita ampliar as capacidades individuais dos mesmos. Como é do conhecimento geral, os UAV's apresentam tempos de voo relativamente curtos devido ao alto consumo de bateria, enquanto que os USV's apresentam um problema relacionado com o facto de apenas poderem navegar em ambientes aquáticos impedindo assim de proporcionar, quando necessário, outro ponto de vista. Deste modo, a cooperação entre os dois veículos permite colmatar as desvantagens de cada um deles, visto que o UAV pode ser transportado a bordo do USV, evitando assim o uso de bateria, e descolar quando for necessário, proporcionando outra perspetiva do terreno [10].

1.2 Material utilizado

Neste projeto utilizou-se um quadricóptero, modelo Parrot AR Drone 2.0 Power Edition, e um catamarã autónomo desenvolvido na Faculdade de Engenharia da Universidade do Porto.

1.2.1 Drone

AR Drone 2.0 é um quadricóptero desenvolvido pela companhia *Parrot Inc* e é o sucessor do primeiro modelo, o AR Drone 1.0.

O drone apresenta dimensões de 517mmx517mm com 127mm de altura e um peso de 380gramas, com capacidade de carregar um peso extra de 100g. Está equipado com quatro motores, apresentando, cada um, uma hélice de diâmetro de 20cm, para além de duas câmaras, uma horizontal direcionada para a sua frente e outra orientada verticalmente para baixo. A câmara horizontal tem uma resolução de 720p e permite gravações a 30fps. A câmara vertical por sua vez permite capturar imagens com uma resolução de 640x360 pixeis e permite gravações até 60fps. AR Drone 2.0 possui ainda uma interface USB que possibilita conectar o *Parrot AR.Drone 2.0 Flight Recorder* e assim permitir a leitura de coordenadas GPS e a gravação de vídeos durante os voos do drone.

Figura 1.1: AR Drone 2.0

Um preço de retalho acessível, e a facilidade de utilização de software para trabalho de desenvolvimento, chamou a atenção de universidades e centros de investigação, sendo hoje utilizado em inúmeros projetos no âmbito da Robótica e de visão computacional.

Um quadricóptero pode ser definido como um helicóptero que é impulsionado a partir de quatro motores de modo a mover-se, no espaço, nos três eixos. São três os movimentos imprescindíveis para o controlo do drone: a aceleração vertical, obtida pela diminuição ou aumento da velocidade dos motores; a aceleração horizontal, obtida a partir do aumento da velocidade de um motor e diminuição da velocidade do motor oposto, dependendo do sentido desejado; a rotação do drone sobre o eixo vertical, que é alcançado com o aumento de velocidade do motores 1 e 4 e

a diminuição dos 2 e 3, e vice-versa, como é visível na figura 1.2 e no esquema que ilustra os seis graus de liberdade do quadricóptero, três movimentos rotacionais sobre cada um dos eixos e três tipos de movimentos direcionais.

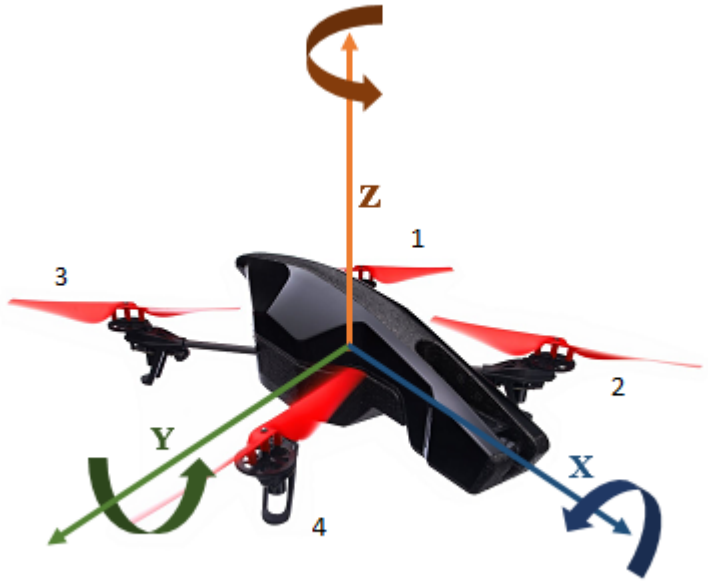


Figura 1.2: Esquema ilustrativo do diferentes graus de liberdade

1.2.2 Catamarã

Zarco é um catamarã de 1.5m de comprimento que tem como objetivo realizar manobras totalmente autónomas em rios e lagos. Este veículo apresenta um peso total de 50 kg, podendo transportar até 50 kg. O catamarã apresenta também espaço suficiente para transportar uma plataforma de dimensão 1m x 1m. Zarco possui dois propulsores elétricos que permitem uma navegação com uma velocidade de até 3 nós. Por sua vez, a sua posição, relativamente aos objetivos das manobras, é assegurada pela utilização de GPS diferencial que apresenta grande precisão.

1.3 Estrutura da Dissertação

Para além da introdução, a presente dissertação, engloba mais 5 capítulos. No capítulo 2, é descrito o estado da arte e são apresentados trabalhos relacionados. No capítulo 3, é explanado o processo de construção da plataforma, bem como o padrão nela expresso e ainda a análise do sistema de visão. Nos capítulos 4 e 5, são abordadas as três fases da missão: descolagem, voo e a aterragem do drone e o estudo do comportamento do drone. Refira-se que a análise do drone, nomeadamente do sistema de visão e o estudo do movimento do drone, foi realizado de forma separada. De forma conclusiva, no capítulo 6 são expostas as respetivas ilações e o possível trabalho a desenvolver, a partir deste projeto.

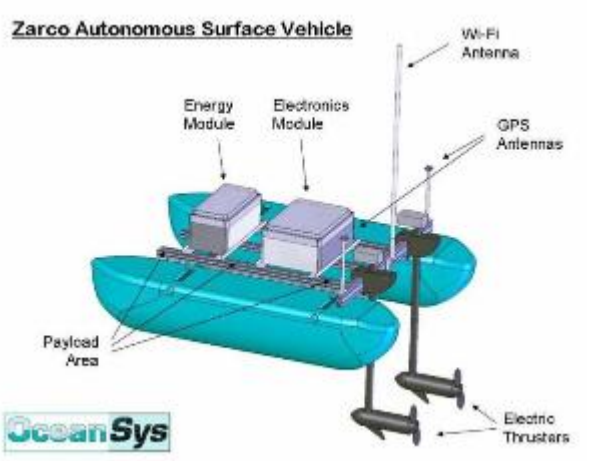


Figura 1.3: Catamarã Zarco

Capítulo 2

Revisão Bibliográfica

2.1 Projetos Semelhantes

2.1.1 Projeto RIVERWATCH

Este projeto [9] é composto por dois veículos, um catamarã com 4.5m de comprimento, que será o USV, e um drone, que por sua vez será o UAV. A cooperação entre os dois veículos permite planear o trajeto de navegação, utilizando um *laser scanner* de grande alcance integrado no catamarã e as imagens capturadas pelo quadricóptero durante o voo, que só poderiam ser obtidas através de satélites, tendo neste caso um custo elevado e podendo estar, eventualmente, desatualizadas.

Um aspeto importante a considerar neste projeto é a sua abordagem à aterragem e descolagem do drone. Para a descolagem ele simplesmente levanta voo rapidamente até uma altura considerável, no entanto, a aterragem é um processo bastante mais complexo. Por norma, é esperado, que o drone detete a plataforma de aterragem no catamarã, contudo, como é descrito no artigo, a deteção desta plataforma nem sempre é bem-sucedida devido a variáveis como a luminosidade. Além disso, é necessário, alguma capacidade computacional, por parte do drone. Para ultrapassar esta dificuldade, foi implementada uma câmara no catamarã direcionada para cima, de modo a detetar a posição do drone e assim assisti-lo na descida, uma vez que sendo o céu o fundo, existiria por isso menos ruído.

Esta abordagem de aterragem demonstrou a sua sensibilidade relativamente à posição do sol durante dias de céu limpo, uma vez que quando aquele se encontra no seu ponto mais alto, influencia a câmara do catamarã impedindo a deteção do drone. Durante a aterragem verificou-se que quando o drone se encontrava perto da plataforma e da câmara não era possível realizar a deteção, uma vez que o campo de visão da câmara não permitiu que o drone estivesse completamente visível. Foi, consequentemente, implementada uma rede de segurança pelo perímetro da plataforma, para evitar que o drone se deslocasse para fora da plataforma tanto na aterragem, como durante a navegação do catamarã enquanto se encontrava parado na plataforma.

2.1.2 Projeto Sea Robot-Assisted Inspection

Neste projeto [5] foram também utilizados um catamarã e um drone para realizar diversos tipos de missões, nomeadamente a nível militar, como reconhecimento de zonas litorais hostis, para aceder a locais no litoral de difícil acesso ou ainda em manobras na ocorrência de desastres naturais. Este método acarretou um custo reduzido, mais prático e mais seguro comparativamente à alternativa que seria a realização de missões em veículos tripulados.

O controlo dos dois veículos foi realizado através de controlo remoto por dois utilizadores, um responsável por cada veículo. Caso se encontrem fora do campo de visão dos utilizadores, estes podem utilizar uma interface num computador. Não sendo totalmente autónomo este tipo de missões, afasta-se do objetivo presente documento.

Para a programação e implementação do projeto foi utilizado um *software*, baseado em linguagem *Java*, denominado *distributed field robotics architecture*.

2.2 Abordagem de aterragem

O processo de aterragem de um drone numa plataforma de uma forma precisa e segura é um processo delicado, uma vez que exige total estabilidade do drone e um controlo robusto do mesmo [11]. Encontrar esta estabilidade torna-se um processo bastante complicado em ambientes exteriores visto que o drone está suscetível a uma variável bastante imprevisível, o vento. Uma dos principais fases neste processo é sem dúvida a deteção da plataforma, o que pressupôs a realização de uma pesquisa de modo a perceber o tipo de investigação já realizado nesta área.

No artigo [8] foram testados três métodos para a deteção do padrão, nomeadamente *template matching, feature detection and matching* e *blob detection*. *Template macthing* é uma técnica que consiste na deteção de uma parte de uma imagem modelo. Apesar desta abordagem ser rápida, não é ideal para ser aplicada neste projeto, na medida em que este método apresenta problemas na deteção quando a imagem tem uma orientação e escala diferente da original.

Feature detection consiste na identificação na imagem obtida de pontos de interesse e posteriormente análise desses pontos com o intuito de perceber se existe alguma relação entre os pontos e o padrão modelo.

Blob detection por sua vez, baseia-se na identificação de regiões numa imagem que apresentam uma característica semelhante entre si, como por exemplo a cor.

Tendo em consideração estes três métodos existem inúmeros padrões possíveis para a plataforma, como por exemplo, o conhecido símbolo de um heliporto, em forma de H[8]. Neste projeto optou-se por um padrão constituído por dois círculos de cores distintas, vermelho e azul, aplicando um filtro HSV, *Hue Value and Saturation*, é possível isolar os dois círculos e deste modo usa-los como centro da plataforma[8]. Este método no entanto não aparenta ser eficaz para deteções em ambientes exteriores uma vez que a luminosidade influencia consideravelmente a cor do padrão e a consequente deteção por parte do drone.

Outro padrão possível é o demonstrado no artigo [3]. Este padrão é composto por uma série de círculos concêntricos, o que apresenta uma vantagem clara relativamente a padrões com uma forma específica, como por exemplo a de um heliporto [11] ou de seis quadrados brancos num fundo preto [1]. Visto que durante a aterragem do drone o padrão irá ser detetado para diferentes escalas, a algum momento durante a descida o padrão estará apenas parcialmente visível na imagem pondo em causa todo o processo de deteção. Assim, o padrão composto pelos círculos concêntricos de diferentes raios permite ultrapassar esta dificuldade, na medida em que existirá sempre um círculo de raio menor para ser detetado. Uma das desvantagens deste padrão é o facto de não ser possível extrair a orientação do drone.

2.3 Software para implementação

Existem, presentemente, inúmeros *softwares* que permitem programar e implementar aplicações no campo de robótica. A pesquisa desenvolvida visou a recolha de informação útil e relevante na tentativa de procurar compreender qual o *software* mais indicado para o controlo de um drone em geral, e para o AR DRone 2.0 em particulare.

Robot Operating System, é, atualmente, o mais popular para o controlo de aplicações que envolvam robótica [7][16]. Este software baseia-se numa arquitetura baseada na Teoria dos Grafos, onde o processamento é realizado nos nós que podem enviar e receber mensagens. A este software podem ser adicionados módulos que possuem bibliotecas para determinadas áreas.

2.3.1 ROS - Robot Operating System

Robot Operating System é um *software* criado pelo *Standord Artificial Intelligence Laboratory* em 2007 com o objetivo de proporcionar ferramentas e bibliotecas simples para a projeção de aplicações para robôs, sendo desenvolvido posteriormente na *Willow Garage*.

ROS pode ser desenvolvido em C++, *Python* e Lisp e é constituído por quatro conceitos fundamentais, nós, tópicos, mensagens e serviços.

Os nós podem ser definidos como processos que executam algum tipo de computação. A utilização de vários nós individuais acarreta diversas vantagens, nomeadamente a complexidade do sistema, a programação e o facto de, no caso de ocorrer alguma falha, este erro se encontrar isolado no nó, evitando assim a sua propagação[4].

A cada nó existe um nome correspondente que os diferencia entre si, sendo a comunicação assegurada através de mensagens. Estas mensagens são trocadas entre nós utilizando os tópicos. Para isso os nós subscrevem a um determinado tópico, caso pretendam receber informação relevante ou, caso queiram enviar informação, publicam para o respetivo tópico. Para cada tópico podem existir vários publicadores e subscritores. Este método de comunicação apesar de ser eficaz não é apropriado para interações que exijam sincronismo, como tal nestas situações, são utilizados os serviços. Um nó fornece um serviço, utilizando um nome específico, e de seguida aguarda uma interação de um cliente e da sua respetiva mensagem[4].

Os módulos *ardrone_autonomy* [12] [14], *camera_calibration* [13] e *vision_opencv*[17] [15] apresentam ferramentas interessantes para a implementação deste projeto. O primeiro é constituído por bibliotecas que permitem o controlo do AR Drone 2.0 e é baseado no *software* de desenvolvimento oficial da Parrot. O segundo é um módulo que permite utilizar as bibliotecas OpenCV, bibliotecas utilizadas para o desenvolvimento de aplicações na área de visão computacional e processamento de imagem em tempo real. Por fim, o módulo *camera_calibration* permitiu realizar a calibração da câmara vertical. De seguida será explicado em detalhe o módulo *ardrone_autonomy*.

2.3.2 Módulo ardrone_autonomy

Ardrone_autonomy é um módulo ROS que fornece bibliotecas para a programação do AR Drone 1.0 e 2.0. Este módulo foi desenvolvido por *Mani Monajjemi* na Universidade *Simon Fraser*.

Este módulo incluí diversos componentes importantes para este projeto, nomeadamente a leitura de sensores do drone, através da subscrição ao tópico *ardrone/navdata*. A informação obtida pelos sensores é enviada pelo drone a cada 5ms numa estrutura específica. Na tabela 2.1 podemos constatar como é constituída esta estrutura. O eixos referidos na tabela são os eixos descritos na figura 3.

Tabela 2.1: Estrutura da mensagem de leitura de sensores

Nome da variável	Tipo	Definição
header	std_msgs/Header	Cabeçalho da trama de informação
batteryPercent	float32	Percentagem de bateria restante
state	uint32	Estado atual do drone
rotX	float32	Rotação em graus segundo o eixo X
rotY	float32	Rotação em graus segundo o eixo Y
rotZ	float32	Rotação em graus segundo o eixo Z
magX, magY, magZ	int32	Leitura do magnetômetro, em unidades Tesla, para os diferentes eixos
pressure	int32	Leitura, em unidades Pascal, do barômetro
temp	int32	Temperatura exterior atual
wind_speed	float32	Velocidade atual do vento
wind_angle	float32	Ângulo atual do vento
wind_comp_angle	float32	Retificação estimada devido ao ângulo do vento
altd	int32	Altura atual do drone em milímetros
motor1, ..., motor4	int32	Leitura do barômetro
vx, vy, vz	float32	Velocidade linear nos três eixos em milímetros por segundo
ax, ay, az	float32	Acelaração linear nos três eixos em unidades g
tm	float32	Tempo da leitura da informação em microsegundos

O módulo pode funcionar em três tipos diferentes de ciclos, em tempo-real com atualizações de 5 ou 67 ms ou com atualizações em tempo fixo de 20ms. Em modo tempo real a informação é publicada no momento em que é recebida. A escolha do tempo de ciclo é bastante relevante uma vez que no caso do tempo de ciclo do módulo for superior ao tempo de transmissão do drone, alguma informação vai ser perdida.

Ardrone_autonomy permite também aceder às duas câmaras do drone, a frontal e a vertical, subscrevendo, respetivamente, aos tópicos *ardrone/front/image_raw* e *ardrone/bottom/image_raw*.

2.4 Resumo e Conclusões

Da análise de projetos semelhantes ao proposto, conclui-se que na verdade ainda não existem missões completamente autónomas, daí a pertinência do *HoverSea* no sentido de tentar colmatar as falhas que ainda persistem.

Existem soluções para o formato do padrão e a sua respetiva deteção, no entanto, o padrão composto pelos círculos concêntricos de raios diferentes foi o que se demonstrou mais adequado a este projeto, uma vez que permite a identificação de um padrão para diferentes escalas e quando este não se encontra totalmente visível. Uma desvantagem deste padrão consiste na impossibilidade de obter a orientação do drone face à plataforma, no entanto para este projeto a orientação durante a aterragem não é relevante.

O *software* ROS demonstra ser apropriado às necessidades deste projeto devido à simplicidade de utilização e devido ao facto de ter sido desenhado com o objetivo de ser o mais modular possível, permitindo assim aos utilizadores deste software utilizarem apenas os módulos mais úteis para o seu projeto.

Capítulo 3

Plataforma de Aterragem

A escolha do padrão e do material a usar na plataforma de aterragem foi crucial para o sucesso da missão, uma vez que todo o processo dependia de uma boa deteção do padrão. Como tal, para essa seleção, foram considerados vários fatores que poderiam dificultar a deteção, nomeadamente a qualidade da câmara vertical do drone, as condições atmosféricas e o ambiente envolvente.

3.1 Padrão

A identificação da plataforma pode ser realizada através de duas abordagens: a primeira, utilizando o reconhecimento por cor e a segunda, através da análise da forma de um padrão presente na plataforma. A simplicidade, eficácia e rapidez são grandes vantagens que o reconhecimento por cor apresenta relativamente ao outro método, no entanto, apresenta desvantagens que são essenciais, nesta situação em concreto. Este projeto ambiciona a realização de missões em espaços exteriores, o que compromete a análise da imagem, uma vez que esta está sujeita a grandes variações de luminosidade, saturação e brilho, dificultando assim a identificação. A utilização deste método revela ainda como problema a possível existência de "falsos positivos" uma vez que, durante o momento da identificação da plataforma, podem existir objetos com a mesma cor, impossibilitando assim o drone de distinguir o alvo correto. Finalmente, o reconhecimento por cor, impossibilita extrair informação relativamente ao padrão, como por exemplo a identificação da orientação do drone durante a aterragem, o que nesta dissertação pode não ser importante, visto que não interessa como é feita a aterragem, mas para trabalho futuro poderá ser relevante. O reconhecimento por forma permite uma análise mais robusta do que por cor, na medida em que as condições de ambiente e do padrão, como a luminosidade, reflexão ou textura, não interferem de uma maneira tão evidente, visto que neste tipo de identificação são analisadas as propriedades geométricas como ângulos, o tamanho, número de lados, entre outros aspetos. A análise por forma permite ainda obter informações importantes relativamente ao padrão a reconhecer, nomeadamente a rotação e escala. Considerando as vantagens e desvantagens, a deteção do padrão será realizada utilizando a forma do padrão.

3.1.1 Forma

Como já referido na revisão bibliográfica existem diversas propostas para a forma do padrão, nomeadamente com o formato de um heliporto [8], ou composto por 4 quadrados brancos num fundo preto [2]. Estes exemplos, no entanto, apresentam certas debilidades.

Neste projeto, o drone, tem de ser capaz de detetar o padrão para diferentes escalas, uma vez que encontrar-se-á a diferentes alturas durante a descida. Neste sentido, os padrões descritos não são eficazes, na medida em que para alturas baixas o drone não será capaz de detetar a plataforma na sua totalidade. A 1.5 metros de altura o drone é capaz de detetar no máximo 1.35m x 0.75m, visto que a plataforma terá 1m x 1m verificamos que devido à instabilidade do drone a plataforma não estará completamente visível. Reduzir a dimensão do padrão não é uma solução viável, visto que compromete deteções a maiores altitudes.

Deste modo propôs-se um padrão que permitisse a deteção para diferentes escalas. Na figura 3.1 pode observar-se o padrão utilizado.

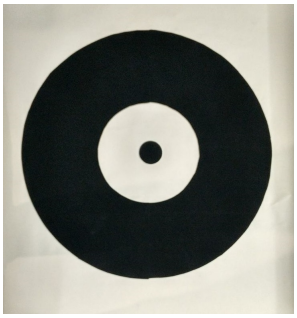


Figura 3.1: Padrão utilizado com círculos concêntricos de raios 3.5cm, 15cm e 37.5cm

O padrão apresentado é constituído por três círculos concêntricos de raios diferentes, permitindo assim, para diversas alturas, detetar pelo menos um deles. O alvo de aterragem seria o centro dos círculos que corresponde ao centro da plataforma.

Visto que podemos determinar a altura a que se encontra o drone durante a deteção, através da leitura dos sensores, e sabendo a relação entre as medidas na projeção da imagem na câmara e as medidas reais, podemos prever o raio do círculo que esperamos detetar na imagem. Deste modo, podemos prevenir, a deteção de falsos positivos.

O padrão utilizado apresenta raios de valor 37.5 cm, 15cm e 3.5cm. Sabendo estas medidas apenas é necessário calcular a relação entre centímetros e os pixeis na imagem e prever assim o raio do padrão na imagem.

Colocando o drone a uma altura de 1 metro verificamos que no eixo horizontal os 640 pixeis correspondem a 0,90 metros. Deste modo podemos obter a seguinte relação:

$$Dist\_Max\_Detetada = \frac{0,90 \times altura\_atual}{1} \tag{3.1}$$

$$raio = \frac{raio\_real \times PxlMaxHorizontal}{Dist\_Max\_Detetada} = \frac{raio\_real \times 640}{Dist\_Max\_Detetada} \tag{3.2}$$

A variável *Dist_Max_Detetada* representa a distância máxima detetada no eixo horizontal em metros. *Raio_real* será o valor do raio na realidade. O valor de *raio* calculado será o tamanho em pixeis do círculo a detetar.

De modo a verificar estes princípios realizaram-se testes para os três raios. Para o círculo de raio 3.5cm e de 15cm colocou-se o drone a 1 metro de altura, prevendo-se por isso um raio na imagem de 25 pixeis e 107 pixeis, respetivamente. Para o círculo de raio maior, 37.5cm, colocou-se o drone a 3.5 metros de altura, esperando-se assim detetar um círculo de raio 76 pixeis. Nas figuras 3.2, 3.3 e 3.4 podemos observar o raio dos círculos detetados a cor azul e o raio previsto do círculo a cor verde.

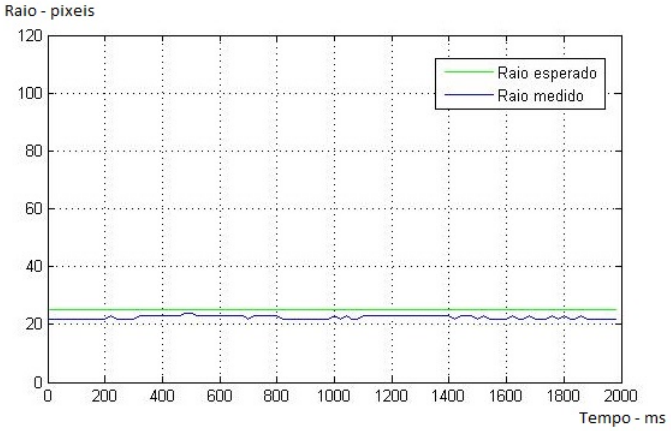


Figura 3.2: Resultado do teste da previsão do raio de 3.5cm do círculo

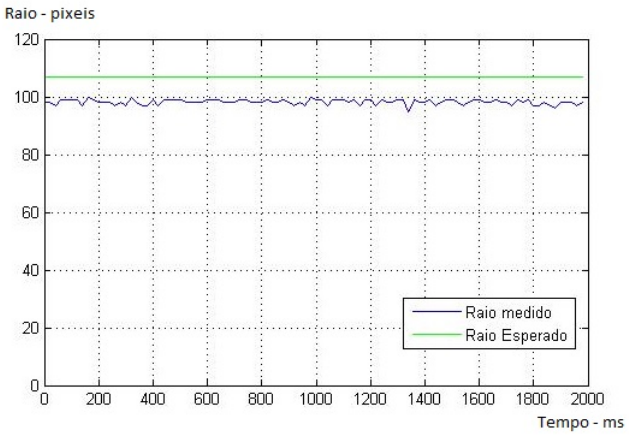


Figura 3.3: Resultado do teste da previsão do raio de 15cm do círculo

Constata-se existir um erro associado nesta previsão. De modo a perceber o comportamento deste erro realizou-se o tratamento de informação que pode ser observado na tabela 3.1.

Na figura 3.5 observa-se a relação linear entre a dimensão do raio a detetar e o erro obtido. Verifica-se que o erro entre o raio previsto e o raio obtido na imagem é proporcional com o aumento

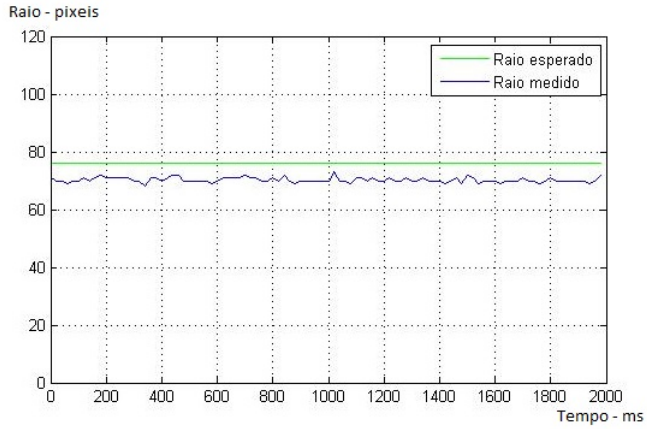


Figura 3.4: Resultado do teste da previsão do raio de 37.5cm do círculo

do raio. Realizando uma regressão linear podemos constatar que esta relação tem valor $\frac{7}{80}$, ou seja podemos corrigir o raio previsto utilizando a seguinte expressão:

$$raio_corrigido = \frac{7}{80} \times raio \tag{3.3}$$

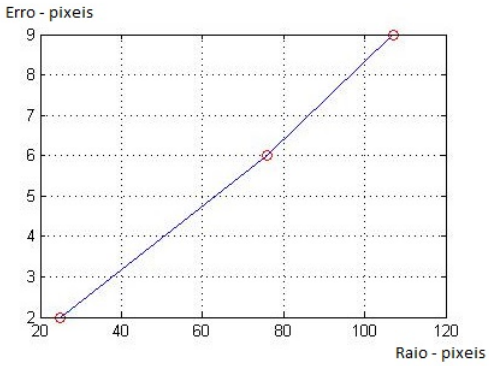


Figura 3.5: Relação linear entre a dimensão do raio e o erro

Uma vez que as medições apresentam um desvio padrão de 1 pixel para os três raios, impôs-se um limite superior e limite inferior na ordem de 4 pixeis, correspondente a quatro desvios padrões.

Concluindo, os círculos detetados terão de respeitar a seguinte condição:

$$raio\_corrigido - 4 < raio\_obtido\_na\_imagem < raio\_corrigido + 4 \tag{3.4}$$

3.1.2 Material

Durante os primeiros testes para a deteção utilizou-se uma versão impressa do padrão proposto. Os resultados destes testes, não foram satisfatórios, devido à reflexão da luz no papel, pelo que houve necessidade de construir um novo modelo constituído por papel de veludo.

Tabela 3.1: Tratamento de informação relativo aos testes de previsão de raio a detetar

Raio (cm)	Raio previsto (pixeis)	Média dos resultados (pixeis)	Desvio padrão (pixeis)	Erro (pixeis)
3.5	25	23	1	2
15	107	98	1	9
37.5	76	70	1	6

3.2 Calibração das câmaras

A calibração das câmaras é um processo essencial quando se pretende estimar a localização de um objeto através das imagens capturadas pela câmara. Neste caso pretende-se detetar o padrão concebido, utilizando a câmara vertical do drone.

Este processo foi realizado utilizando o módulo ROS *Camera_calibration* que permite a calibração de câmaras utilizando apenas um padrão axadrezado. Para funcionamento do módulo é necessário determinar três parâmetros de entrada:

1. O tipo de padrão que se irá utilizar (*chessboard, circles e acircles)*

2. Dimensão do padrão

3. Dimensão dos quadrados do padrão caso o padrão utilizado seja *chessboard*

Uma vez que para a missão apenas é utilizada a câmara vertical, só foi realizada a calibração para esta. Para este processo utilizou-se um padrão axadrezado constituído por 4x5 quadrados com dimensões 35mm x 35mm. Na figura 3.6 podemos ver o padrão utilizado e na figura 3.7 a respetiva calibração.

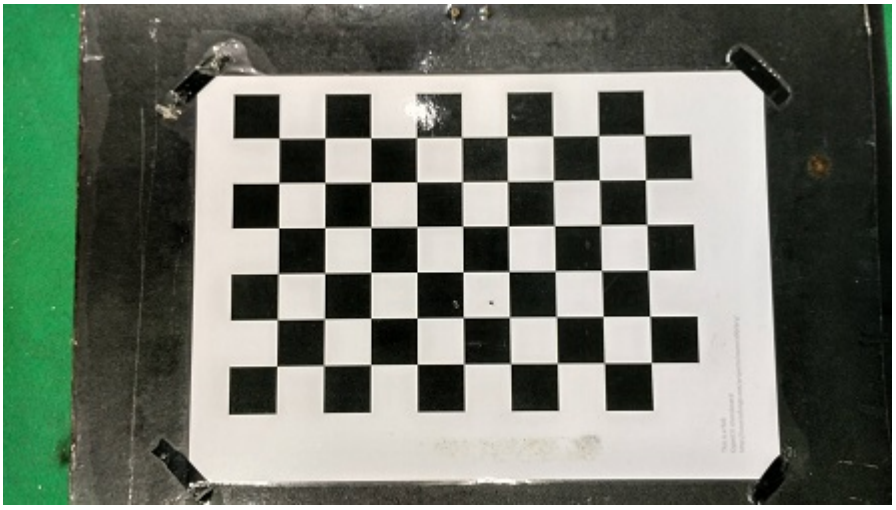


Figura 3.6: Padrão de calibração

Depois de realizada a calibração o módulo cria um ficheiro, no diretório da câmara do módulo *ardrone_autonomy*, composto por uma matriz de parâmetros da câmara e o vetor de distorção. A primeira é uma matriz 3x4 e descreve os pontos 3D no mundo para uma imagem 2D, enquanto que o vetor de distorção contém os coeficientes de distorção.

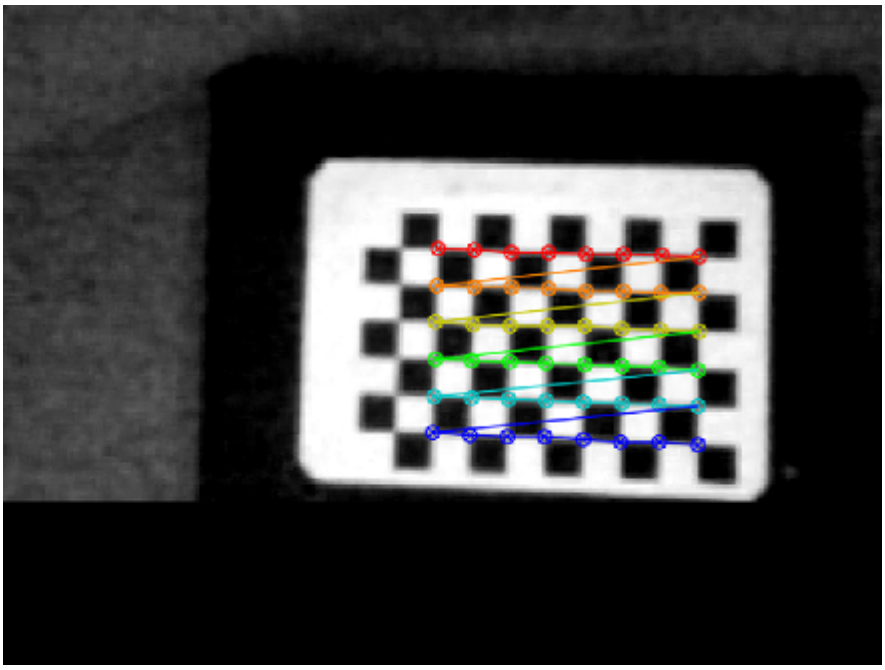


Figura 3.7: Imagem capturada durante a calibração

Este procedimento é de grande importância especialmente em projetos que necessitem de precisão em deteção de *features*, na medida em que estes parâmetros permitem realizar a correspondência dos pontos do espaço tridimensional com os pixeis das imagens.

3.3 Deteção

A deteção dos círculos na imagem será efetuado usando o método de *feature extraction*, *Circle Hough Transform*. Este método aplica o mesmos princípios da *Hough Transform* mas na identificação de círculos em imagens.

3.3.1 Circle Hough Transform

Um círculo pode ser descrito utilizando os seguintes parâmetros, as coordenadas do seu centro, (a, b), e o seu raio, r.

$$r^2 = (x-a)^2 + (y-b)^2 \tag{3.5}$$

Circle Hough Transform tem como objetivo encontrar pontos na imagem que satisfaçam estes parâmetros. Uma limitação imediata na utilização deste método é a infinidade de raios que teriam de ser testados, tornando o processo demorado e pesado computacionalmente. Contudo, como visto anteriormente, podemos prever a dimensão do raio a detetar.

A primeira fase antes de utilizar a transformada consiste na aplicação de um *edge detector*, como o *Canny*, permitindo assim calcular as linhas de uma imagem como podemos ver na figura 3.8.

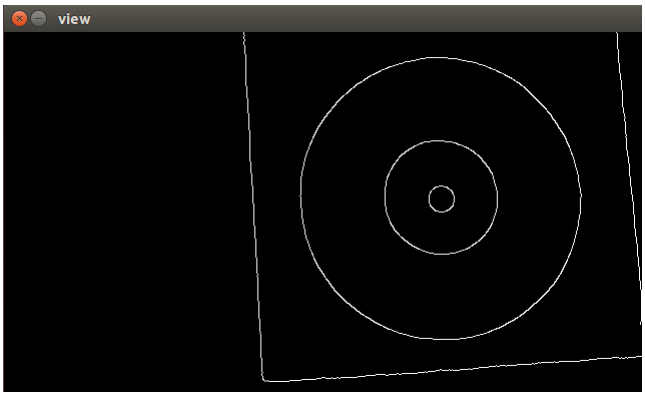


Figura 3.8: Imagem depois de aplicado um filtro edge detector

Obtida agora a imagem com os contornos é desenhado um círculo para cada ponto dos contornos com o raio pretendido. Incrementando no acumulador, vetor com o tamanho igual ao número de pontos nos contornos, todos os pontos cujo o perímetro do círculo cruze. Por fim, no seguimento deste processo é feito o mapeamento utilizando o acumulador e o seu máximo, ou máximos no caso da presença de mais de um círculo na imagem.

Para aplicar esta transformada utilizamos a função *HoughCircles* presente na biblioteca OpenCV. Esta função aceita os seguintes parâmetros:

1. Imagem obtida pelo drone;

2. Vetor que irá ser preenchido com os parâmetros dos círculos detetados;

3. Método de deteção;

4. Inverso do rácio entre a resolução do acumulador e a imagem;

5. Mínima distância entre os centros dos círculos;

6. Valor do *threshold* mais alto utilizado no *edge detector*;

7. *Threshold* do acumulador para a deteção dos centros do círculo;

8. Raio mínimo a detetar;

9. Raio máximo a detetar.

De realçar que a imagem utilizada na função para a deteção é convertida para uma escala de cinzentos e posteriormente é aplicado um desfoque Gaussiano de modo a diminuir o ruído da imagem.

Os parâmetros de entrada utilizados na deteção de círculos neste projeto foram os seguintes:

1. Imagem capturada;

2. Vetor, cujo cada posição é constituída por dois valores, o raio e centro do círculo;

3. *CV_HOUGH_GRADIENT*, uma vez que é o único implementado na biblioteca *OpenCV*;

4. 1, ou seja a resolução do acumulador é igual à resolução da imagem;

5. 50;

6. 200, valor *standard*;

7. 100;

8. Limite inferior de deteção do círculo corrigido já referido neste capítulo;

9. Limite superior de deteção do círculo corrigido já referido neste capítulo;

O valor do *threshold* do acumulador para a deteção dos centros do círculo foi definido em 100, o que torna o acumulador bastante rigoroso na deteção parcial de círculos. Como já referido, o padrão é composto por círculos concêntricos de diferentes raios, deste modo a não deteção de círculos quando este se encontra parcialmente visível não é um problema, uma vez que existe sempre um círculo mais pequeno para ser detetado.

Na figura 3.9 podemos observar um exemplo de uma deteção depois de aplicado o procedimento.

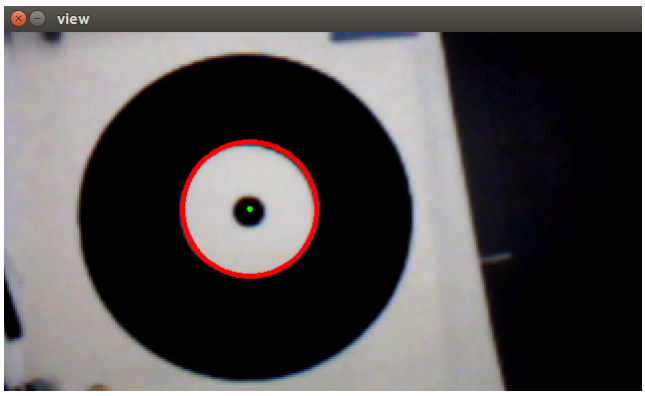


Figura 3.9: Exemplo de deteção

3.3.2 Intervalo de tempo entre deteções

O intervalo de tempo entre deteções é sem dúvida um fator importante de análise, uma vez que este tem grande influencia sobre o controlo do movimento do drone. Deste modo realizaram-se testes que consistiam na colocação do drone a alturas distintas, enquanto era calculado o tempo entre deteções. Podemos observar os resultados nas figuras 3.10, 3.11, 3.12 e 3.13 e uma sucinta análise dos mesmos na tabela 3.2.

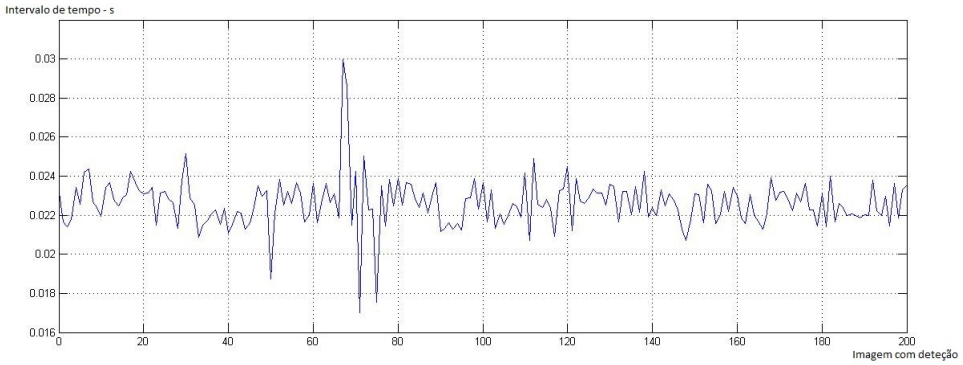


Figura 3.10: Intervalo de tempo entre deteções a um altura de 1metro

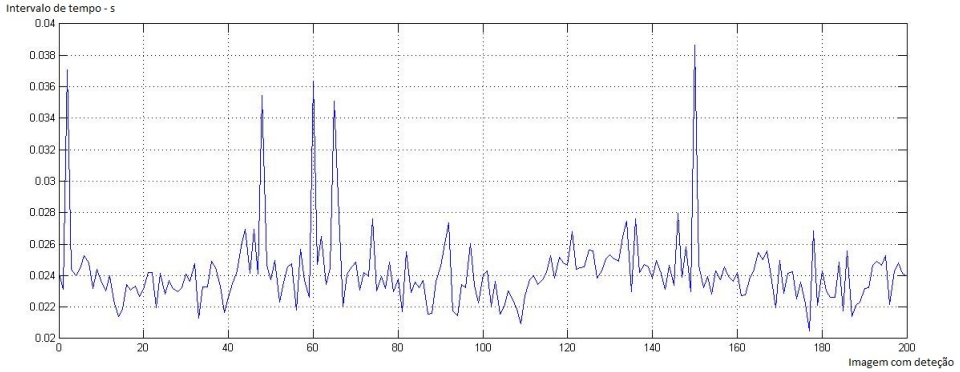


Figura 3.11: Intervalo de tempo entre deteções a um altura de 2metros

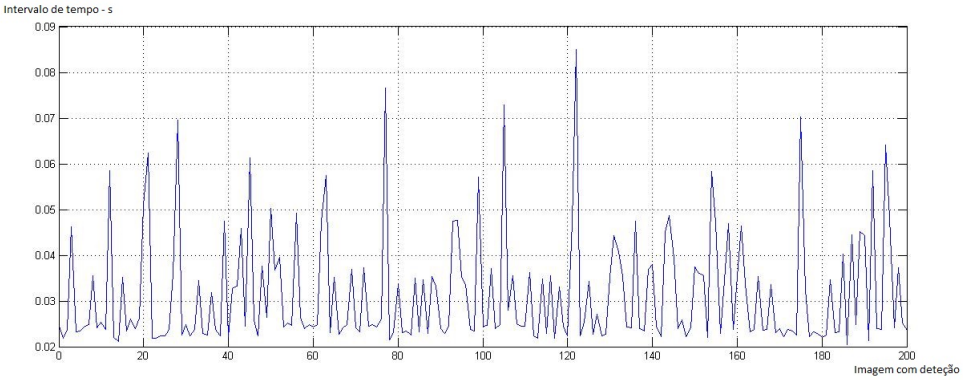


Figura 3.12: Intervalo de tempo entre deteções a um altura de 3.5metros

Analisando as figuras constata-se que o intervalo de deteções é tanto maior quanto maior for a altura do drone, o que vai ao encontro das expectativas iniciais. Podemos verificar que, na pior das situações, o intervalo de tempo entre duas deteções é de 112.5 milissegundos, o que é um resultado bastante satisfatório. Podemos concluir que o desvio padrão dos intervalos é proporcional ao aumento da altura.

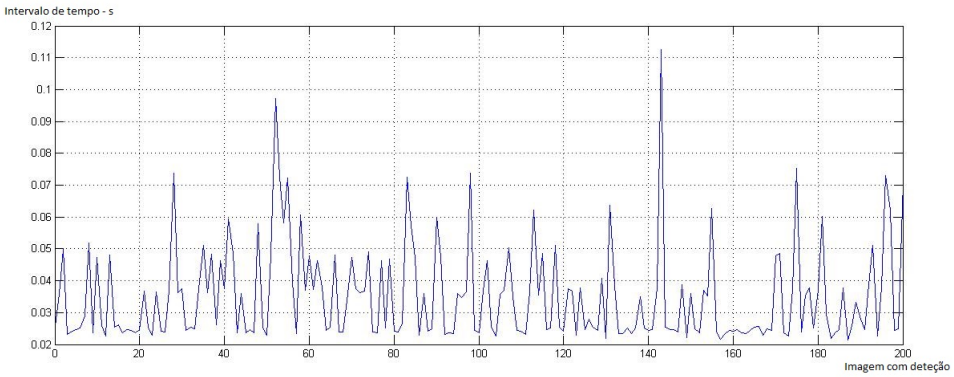


Figura 3.13: Intervalo de tempo entre deteções a um altura de 7metros

Tabela 3.2: Análise dos resultados

Altura (m)	Média de intervalos de tempo (ms)	Desvio padrão (ms)	Máximo (ms)
1	22.6	1.3	30.0
2	24.2	2.4	38.6
3,5	31.5	12.2	84.9
7	35.0	15.1	112.5

Capítulo 4

Descolagem e Missão

4.1 Processo de descolagem

O catamarã, durante a sua missão, comunicará com o drone para assim este iniciar o procedimento de descolagem. Podemos observar o esquema de comunicação entre o computador de controlo presente no catamarã e o drone na seguinte figura 4.1.

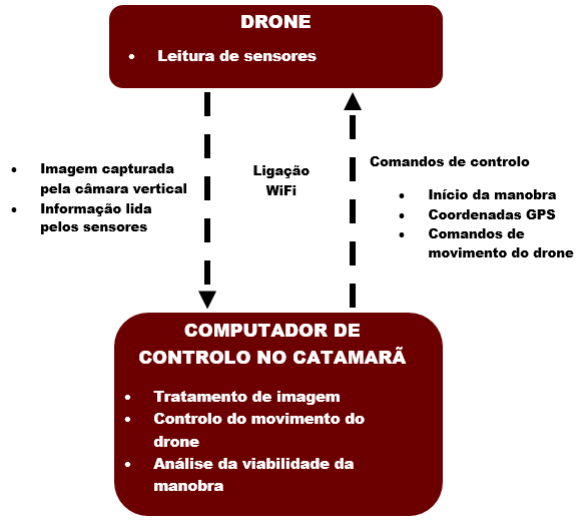


Figura 4.1: Esquema de comunicação entre o computador de controlo e o drone

O processo de descolagem é realizado através da publicação de uma mensagem para o tópico *takeoffdrone* composta por um vetor de quatro elementos do tipo *float* com a estrutura representada na tabela 4.1. De realçar que o drone deslocar-se-á entre as duas coordenadas a uma velocidade constante de 0.5m/s.

Tabela 4.1: Estrutura do vetor com a informação da missão

Latitude	Longitude	Altura	Orientação

No momento em que o drone receber esta mensagem, é analisada a carga atual de bateria, de modo a perceber se é capaz de realizar a missão, sem correr o risco de ficar sem energia durante o voo. Na seguinte secção é explicado como é feita esta análise.

4.1.1 Análise de energia

Uma vez que não é possível parar durante a missão, é crucial verificar antes da descolagem se é possível efetuar a missão com a energia atual do drone. Como tal, foram realizadas alguns testes de modo a perceber o comportamento da descarga de energia durante o voo. A energia da bateria segundo as especificações é de:

$$Energia = 11.1V \times 1500mAh = 16.65Wh \tag{4.1}$$

Os testes foram realizadas no exterior onde o drone se deslocava a uma velocidade de 0.5m/s, uma vez que será esta a velocidade do drone durante as missões, a 1m de altura durante um minuto. Podem observar-se os resultados na figura 4.2. Os resultados no gráfico e os cálculos foram realizados utilizando a percentagem de bateria restante, na medida em que é mais percetível nesta representação o estado atual da bateria.

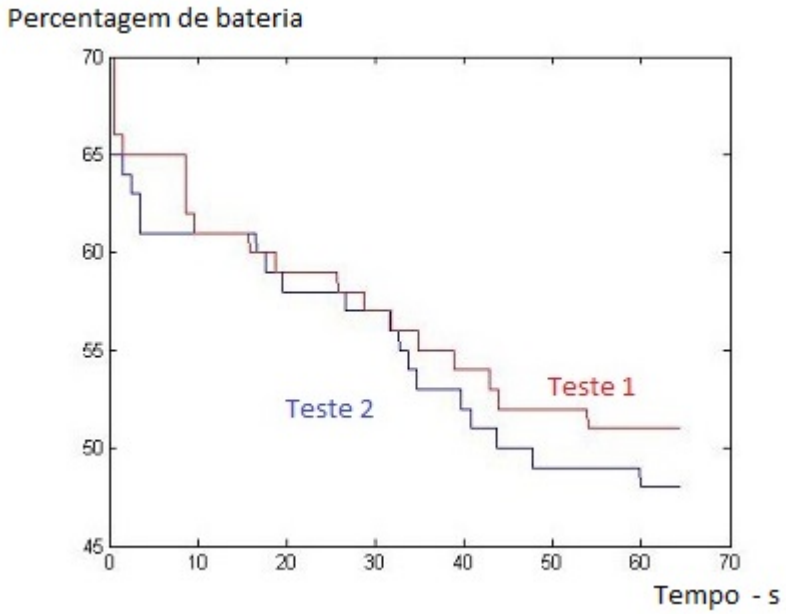


Figura 4.2: Comportamento da percentagem de bateria restante em função do tempo em segundos

Realizando regressão linear para os testes determina-se a relação entre a percentagem de bateria e o tempo decorrido. Obteve-se para o teste 1 uma relação de -0.25 e para o teste 2 uma relação -0.27. Calculada a média para os dois valores podemos determinar a relação entre o tempo de voo e a percentagem de bateria. A média dos dois valores é -0.26, o que significa que podemos escrever a seguinte função para prever o tempo de voo máximo:

$$percentagemFinal - percentagemInicial = -0.26 \times TempoVoo \tag{4.2}$$

Assim, caso a bateria esteja totalmente carregada, o drone apresenta um tempo de voo máximo de 377 segundos, ou seja 6 minutos e 17 segundos, a uma velocidade constante de 0.5m/s. Consultando as especificações oficiais do drone verifica-se que este apresenta tempos de voo entre 12 e 18 minutos. A discrepância entre este valor e o valor obtido pode ser explicado por dois fatores, o primeiro pelo facto de bateria já apresentar algum uso e pelo facto do cálculo do valor de 6 minutos e 17 segundos ter sido realizado para uma velocidade constante de 0.5m/s. É expectável que o drone consuma menos energia quando se encontra parado, uma vez que exige menos potência dos seus motores, e um consumo maior para velocidades também elas maiores.

Sabendo o comportamento da bateria, as coordenadas finais e iniciais da missão pode prever-se se o drone conseguirá chegar ao final da sua missão. A distância entre duas coordenadas GPS foi calculada pelo seguinte algoritmo.

$$Ponto1 = (Latitude1, Longitude1)$$

$$Ponto2 = (Latitude2, Longitude2)$$

$$RaioTerra = 6373Km$$

$$\Delta Longitude = Longitude2 - Longitude1 \tag{4.3}$$

$$\Delta Latitude = Latitude2 - Latitude1 \tag{4.4}$$

$$a = \sin(\Delta Latitude/2)^2 + \cos(Latitude1) * \cos(Latitude2) * \sin(\Delta Longitude/2)^2 \tag{4.5}$$

$$b = 2 * atan2(\sqrt{a}, \sqrt{1-a}) \tag{4.6}$$

$$Dist = RaioTerra * b \tag{4.7}$$

Importante realçar que esta distância calculada é apenas metade do percurso total, uma vez que ele necessita de regressar à embarcação depois de chegar às coordenadas definidas. Por fim, é verificado se o drone possui energia suficiente para realizar a missão e uma aterragem segura.

$$TempoVoo = \frac{percentagemFinal - percentagemInicial}{-0.26} \tag{4.8}$$

$$DistMax = TempoVoo \times Velocidade \tag{4.9}$$

$$DistMax > 2 \times Dist \tag{4.10}$$

Para o drone realizar a missão, a distância máxima que ele pode efetuar com a percentagem de bateria atual terá de ser superior ao dobro da distância entre as coordenadas atuais do drone e as coordenadas objetivo. De realçar que a percentagem final está definida previamente com o valor igual a 25, permitido reservar 25% da energia da bateria para efetuar as manobras de aterragem.

O procedimento efetuado de análise de energia foi realizado para uma velocidade constante de 0.5m/s, no entanto este algoritmo pode ser aplicado para diversos valores de velocidade.

No caso da energia da bateria não ser suficiente é publicado no tópico *batteryInsufficient* uma mensagem, do tipo booleana de valor *True* caso não consiga realizar a missão. Caso contrário a mensagem tem valor *False* e o drone começa o processo de descolagem.

4.2 Missão

Nesta fase da missão o drone desloca-se para as coordenadas GPS, que foram enviadas pelo catamarã. Para isso, é utilizado um ramo do módulo *ardrone_autonomy*, o ramo *gps-waypoint*.

Este ramo apenas pode ser utilizado se o *Parrot GPS Flight Recorder* estiver conectado ao drone, uma vez que este dispositivo permite a publicação de informação GPS para o tópico *ardrone/navata_gps*.

Para utilizar o ramo *gps-waypoint* é enviado para o serviço *ardrone/setgpstarget* uma mensagem estruturada no seguinte molde:

1. Identificação da mensagem;

2. Coordenadas GPS do objetivo - vetor composto por valores, latiude, longitude e altitude;

3. Instruções adicionais - vetor composto por dois valores, velocidade linear durante o voo em m/s e orientação do drone relativamente ao objetivo.

Definidos os valores e posterior envio da mensagem é apenas necessário ativar o modo de voo *ardrone/setautoflight* e o drone autonomamente deslocar-se-á até ao objetivo.

De realçar que a referência das coordenadas GPS utilizada foi a referência *World Geodetic System* 84, WGS 84.

Capítulo 5

Aterragem

A fase da aterragem é, indubitavelmente, a etapa mais crítica de toda a missão, na medida em que existem diversas variáveis que a condicionam. Na seguintes secções serão apresentados o dimensionamento e a implementação de um sistema de aterragem baseado na visão computacional.

Realizada a missão, o drone, deslocar-se-á até as coordenadas GPS do catamarã recorrendo ao mesmo sistema de navegação usado anteriormente. Utilizando a câmara vertical e tratamento de imagem, é adquirida a posição da plataforma relativamente ao drone, como já referido neste documento, sendo posteriormente realizado o controlo da descida aplicando um controlador proporcional.

5.1 Precisão do sistema de navegação GPS

À utilização de um sistema GPS está inerente um erro de precisão que poderá afetar a deteção do padrão, uma vez que apesar de o drone se encontrar nas coordenadas que ele julga serem as corretas, na realidade está posicionado com um certo erro. Neste sentido, foram realizados testes de modo a aferir a ordem deste erro.

Os testes consistiram na fixação do drone numa coordenada GPS no exterior, latitude de 41.177842 (Norte) e longitude de -8.595404 (8.595404 Oeste), e posteriormente era lido no módulo GPS a coordenada e assim aferida a discrepância entre as duas coordenadas. Podem observar-se os resultados na figura 5.1, onde no eixo vertical está representado a longitude e no eixo horizontal a latitude. Podemos ainda verificar a coordenada real representada pelo ponto vermelho.

De modo aferir o erro em metros calculou-se a distância entre a coordenada recebida e a coordenada real. Na figura 5.2 podemos observar o erro em função do tempo.

Analisando o resultado dos testes verifica-se que o sistema de GPS apresenta um erro máximo de 3 metros, distância máxima aproximada entre a coordenada real e as coordenadas lidas pelo sensor GPS. Aferido o erro, determinou-se a altura a que o drone se deve aproximar da plataforma de modo a que mesmo existindo erro, possa detetar a plataforma de aterragem, sendo para isso necessário calcular o ângulo de visão da câmara vertical do drone.

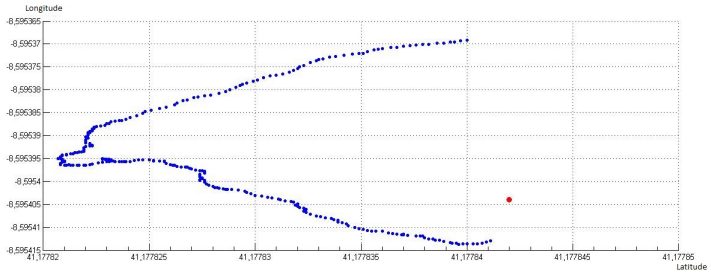


Figura 5.1: Gráfico de dispersão das coordenadas GPS

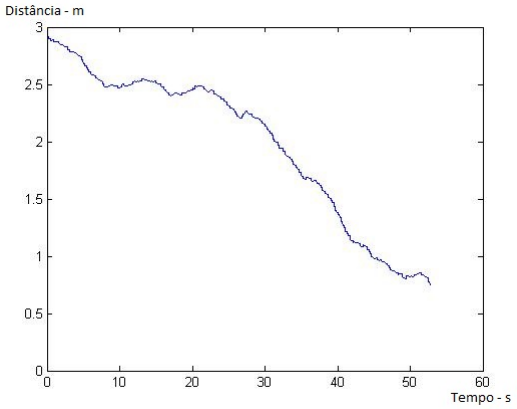


Figura 5.2: Erro em metros entre as duas coordenadas em funçao do tempo

A uma altura de 1 metro, é detetado até 50 cm de largura e 90 cm no comprimento. Tendo isto em consideração é calculado ângulo de visão, λy, para o pior caso, neste caso para a largura.

$$\lambda y = 2 \times \arctan(\frac{0.45}{1}) = 48.45 graus \tag{5.1}$$

Deste modo para detetar até 3 metros o drone precisa de estar a 6,67 metros de altura.

De realçar que nestes cálculos não foi considerado o erro do sistema GPS do catamarã, uma vez que este tipo de embarcações está equipado com sistema de GPS diferencial. Este sistema apresenta uma precisão na ordem dos 10 centímetros sendo por isso desprezável.

5.2 Sistema de controlo

5.2.1 Sensores

Detetada a plataforma de aterragem é necessário começar a aproximação. Capturada a imagem e detetado o centro da plataforma é calculado o erro, ou seja a distância entre o centro do padrão e o centro da imagem, que coincide com o centro do drone. Na figura 5.3 pode ver-se o erro representado nos respetivos eixos, Δx e Δy.

As coordenadas 2D do centro do círculo (x, y), obtidas utilizando a função *HoughCircles*, já explicada neste documento, são relativas ao canto superior direito da imagem, em pixeis. Como

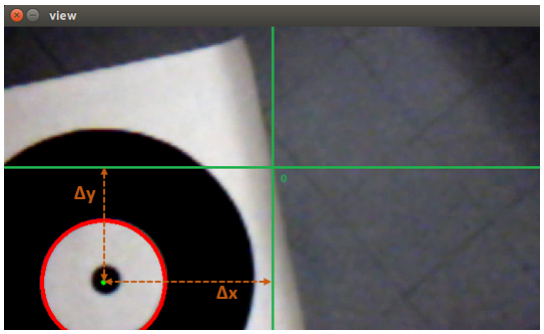


Figura 5.3: Erro entre o centro do padrão e o centro da imagem

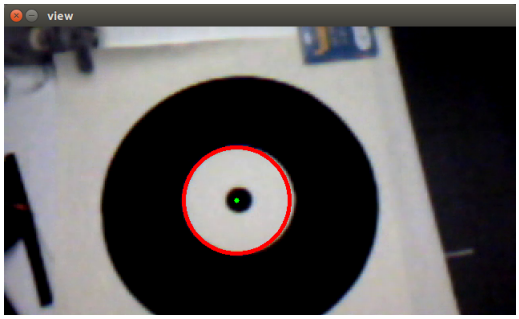
tal é necessário alterar as coordenadas de modo a usar o referencial com a origem no centro da imagem. Para isso usaram-se as seguintes expressões:

$$\Delta x = X - \frac{ImageLength}{2} \tag{5.2}$$

$$\Delta y = \frac{ImageWidth}{2} - Y \tag{5.3}$$

Quando o drone está estabilizado, de facto, estas coordenadas são precisas, contudo, durante a deslocação do drone, existe um erro associado, uma vez que no momento em que o drone se inclina sobre os eixos para se mover as coordenadas na imagem não correspondem à posição real do drone relativamente à plataforma. Na figura 5.4 podemos observar este efeito.

(a) Esquema



(b) Screenshot da câmara

Figura 5.4: Esquema e imagem obtida pela câmara do drone

Através de princípios de trigonometria é possível determinar a equação que permite calcular a distância real entre o drone e o alvo. A expressão formulada é a seguinte, sendo α e θ na expressão os ângulos indicado na figura 5.5:

$$Distancia(A,B) = h \times \tan(\alpha + \theta) \tag{5.4}$$

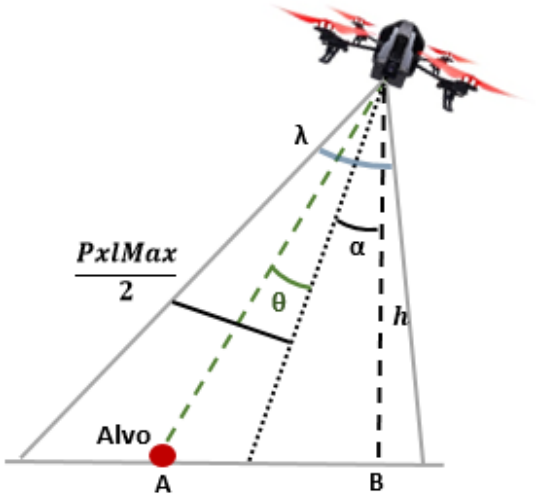


Figura 5.5: Esquema do efeito da inclinação na deteção

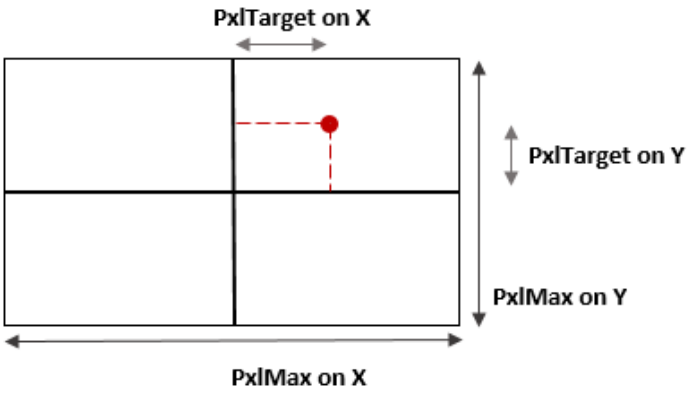


Figura 5.6: Projeção 2D da imagem obtida

O ângulo α pode ser obtido através da leitura dos sensores do drone, por outro lado o θ necessita de cálculos intermédios. Através da figura 5.7 podemos escrever as seguintes equações:

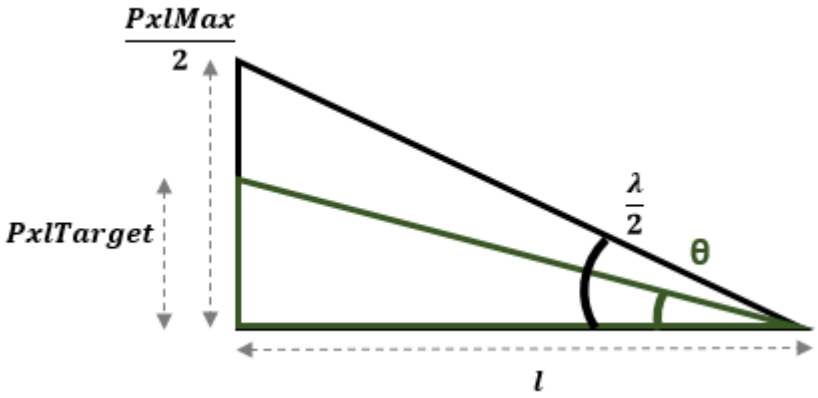


Figura 5.7: Esquema ilustrativo das relações trigonométricas

$$\tan(\frac{\lambda}{2}) = \frac{\frac{pxlMax}{2}}{l} \tag{5.5}$$

$$\tan(\theta) = \frac{pxlTarget}{l} \tag{5.6}$$

$$\tan(\frac{\lambda}{2}) = \frac{\frac{pxlMax}{2}}{\frac{pxlTarget}{\tan(\theta)}} \tag{5.7}$$

Colocando o θ em evidência, podemos utilizar a seguinte fórmula para calcular a posição real em pixeis.

$$\theta = arctan(\frac{tan(\frac{\lambda}{2}) \times pxlTarget}{\frac{pxlMax}{2}}) \tag{5.8}$$

Determinada a equação, podemos aplica-la para os dois eixos. Consultando a ficha técnica do quadricóptero podemos determinar os valores presentes na tabela 5.1. O valor pxlTarget será o erro no respetivo eixo, relativamente ao centro da câmara, obtido depois do tratamento de imagem.

Tabela 5.1: Valores de ângulos de visão e resolução da imagem capturada

Ângulo de visão no eixo X	λy	48 graus
Ângulo de visão no eixo Y	λx	56 graus
Tamanho máximo da imagem no eixo X	PxlMaxX	640
Tamanho máximo da imagem no eixo Y	PxlMaxY	360

Obtido por fim o erro entre o alvo e o drone é necessário diminuí-lo através do controlo do drone. A biblioteca *ardrone_autonomy* permite o controlo de movimento do drone através da

publicação de mensagens de estrutura *geometry_msgs::Twist* para o tópico *cmd_vel*. A estrutura é constituída por dois vetores ambos compostos por três valores (x, y, z), o primeiro responsável pelo movimento linear e o segundo pelo movimento angular. Os valores para estas mensagens podem variar entre valores de -1 e 1.

No caso do movimento linear x e y os valores correspondem à percentagem do ângulo máximo de inclinação do drone, definido previamente. Por exemplo, enviando um valor de 0.5 para o movimento linear x, significa que o drone inclinar-se-á $0.5 \times AnguloMximo$ para a sua frente. No caso do movimento linear z, este valor representa a percentagem do valor máximo de velocidade vertical, definido previamente. Os valores usados neste projeto para o ângulo máximo de inclinação e velocidade vertical máxima, são respetivamente 12 graus e 700 mm/s, valores sugeridos pelo fornecedor.

Para o movimento angular, os valores x e y servem meramente para ativar ou desativar o modo *auto hover*, para valores arbitrários diferentes de zero o modo é desativado. Os valores para o movimento angular z representam no entanto a percentagem da velocidade máxima de rotação sobre o eixo vertical. O valor da velocidade máxima de rotação usado foi 100 graus por segundo.

Para definir o sentido dos movimentos é usado o sinal do valor. Valores positivos para os movimentos lineares x, y e z o drone desloca-se, respetivamente para a sua frente, para a sua esquerda e para cima, no caso de valores negativos o sentido é o inverso.

Valores positivos para o movimento angular z induzem um movimento rotacional no sentido inverso dos ponteiros do relógio, enquanto valores negativos provocam um movimento no sentido inverso.

5.2.2 Identificação do sistema

A função de transferência G(s) representa matematicamente o comportamento de um sistema, nesta situação o sistema é o drone. Para um determinado *input* deste bloco é gerado um *output*, neste caso, o *input* são os valores de inclinação, que por sua vez, geram no *output* valores de velocidade.

Para determinar esta função foram realizados ensaios de modo a aferir o comportamento do drone para diferentes valores de inclinação.

Numa primeira fase, realizaram-se testes que consistiam na descolagem do drone durante 4 segundos, de modo a atingir os 700mm de altura. Posteriormente, o drone, voava linearmente para a sua frente durante 8s, com valores de inclinação de 0.1, e por fim era feita a aterragem durante 4s. Espera-se por isso uma velocidade na ordem de 1m/s, uma vez que a velocidade máxima é 10m/s. Os resultados foram adquiridos pela da leitura da velocidade através da subscrição do tópico *ardrone/navdata*, como já referido no documento, e posterior registo num ficheiro de texto, *.txt*, segundo a nomenclatura "tempo_de_leitura velocidade altura".

Na figura 5.8 pode ver-se o tratamento de resultados de três testes cumprindo estas normas, utilizando o software *MatLab*. O teste 1, 2 e 3 são representados respetivamente pelas cores vermelho, verde e azul. No eixo horizontal é representado o tempo e no eixo vertical é representado a velocidade do drone em mm/s.

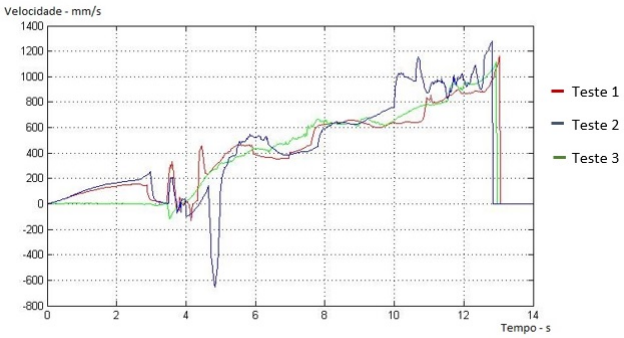


Figura 5.8: Resultados dos testes 1, 2 e 3

Apesar de algum ruído presente nas medições, é possível, pela análise do gráfico, verificar que existe uma semelhança no comportamento da velocidade entre os três testes. Esta semelhança pode ser descrita por uma função de transferência de primeira ordem, no entanto pela interpretação da figura constata-se que o drone nos testes 1 e 2 não chegam a estabilizar na velocidade expectável. No teste 3, no entanto, verifica-se uma estabilização nos 1000mm/s aos 10s, ou seja demorou 6s até atingir a velocidade desejada partindo de uma posição estacionária. Estes resultados permitiram verificar a velocidade lenta do processo de aceleração do drone, sendo necessário realizar testes diferentes, uma vez que estes não permitem aferir o comportamento do mesmo.

Numa segunda fase, efetuaram-se dois testes diferentes, ambos com tempos de descolagem e aterragem de 4s e 12s de voo linear, mas para valores de inclinação diferentes. No teste 4 foi utilizado um valor de inclinação 0.03 e no teste 5 um valor de inclinação de 0.06.

Nestes testes, ao aumentar o tempo de voo, espera-se observar melhor a estabilização na velocidade esperada, 300mm/s para o teste 4 e 600mm/s para o teste 5. A diminuição do valor de inclinação para valores diferentes tem como objetivo perceber se o comportamento de aceleração do drone é o mesmo do verificado nos primeiros testes. Na figura 5.9 podemos observar os resultados dos dois testes, o teste 4 representado pela gráfico de cor azul e o teste 5 representado pela cor vermelha.

Como podemos observar na figura 5.9, para velocidades mais baixas, o tempo para atingir a velocidade desejada é também ele mais baixo. Como podemos constatar para o teste 4, o drone demora cerca de 1,5 segundos para atingir o regime permanente enquanto que no teste 5 este demora 6s. Mais uma vez podemos verificar em ambos os testes um ruído significativo, dificultando assim determinar a função de transferência.

Visto que os resultados obtidos não eram satisfatórios realizou-se outro tipo de testes. Desta vez, para além de se analisar o comportamento do drone em movimento linear para a sua frente, registou-se também a velocidade do drone enquanto recuava. O protocolo deste teste consistiu na descolagem do drone durante 4 segundos, voo para a sua frente durante 15 segundos e posterior paragem imediata. O drone esperava em *hover* durante 3 segundos e de seguida realizava outro voo, desta vez a recuar durante 15 segundos acabando por fim por aterrar durante 3 segundos. Os valores de inclinação para os dois voos foram 0.03 e -0.03 respetivamente. Na figura 5.10 podemos

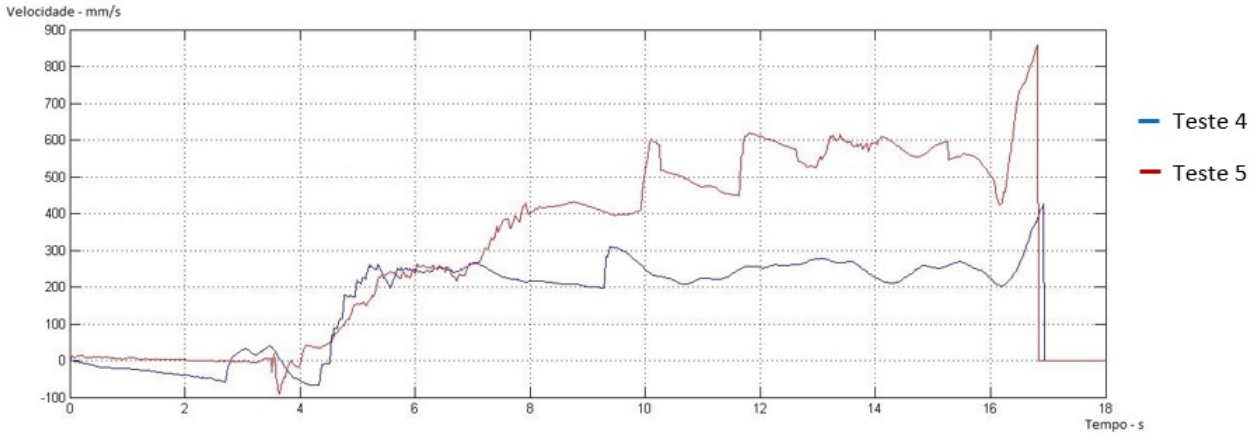


Figura 5.9: Resultados dos testes 4 e 5

ver os resultados obtidos pelos dois testes, o teste 6 representado pela cor azul e o teste 7 pela cor vermelha.

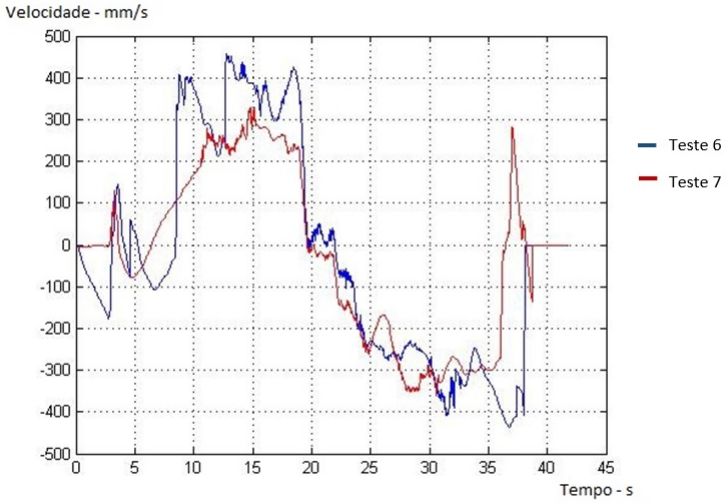


Figura 5.10: Resultados dos testes 6 e 7

Analisando a figura 5.10 verificamos na fase de recuo um comportamento de uma função de transferência de primeira ordem quando o drone inicia o voo partindo da posição *hover* aos 22 segundos do teste. Deste modo dimensionou-se o controlador utilizando esta fase, uma vez que nos outros voos verificou-se a existência de muito ruído.

Uma função de transferência pode ser descrita pela seguinte expressão:

$$G(s) = \frac{k}{\tau s + 1} \tag{5.9}$$

O valor τ representa o tempo de subida, ou seja o tempo que o drone demora a atingir a velocidade desejada partindo de uma posição estacionária. Analisando a figura 5.10 verificamos que este tempo é cerca de 2s. O k representa o ganho da função, neste caso, como o valor de

inclinação enviado foi -0.03, a velocidade do drone seria -300mm/s, deste modo o ganho será 10000. Concluindo, a função de transferência utilizada foi a seguinte:

$$G(s) = \frac{10000}{2s+1} \tag{5.10}$$

Utilizando as funcionalidades do Matlab colocou-se no mesmo gráfico, figura 5.11, três curvas, a primeira a representar o valor da velocidade pretendido, curva de cor vermelha, a segunda representar os resultados obtidos no teste 7 e por último a curva azul que reflete a função de transferência calculada.

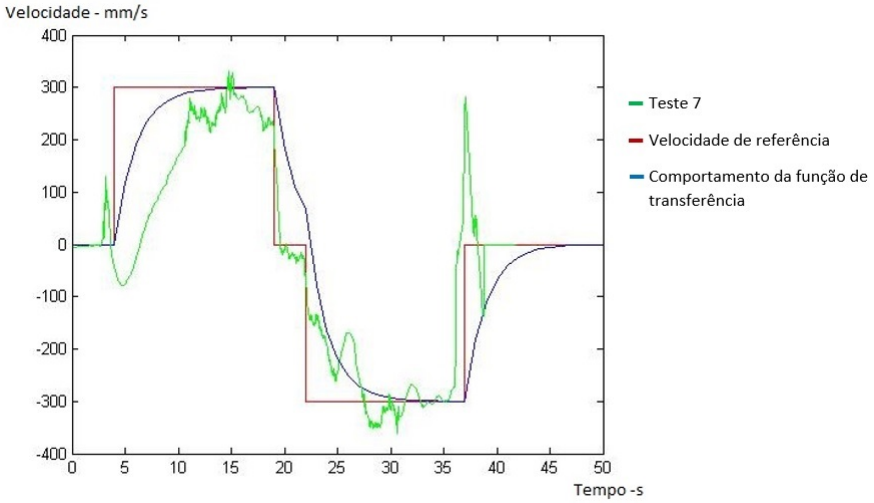


Figura 5.11: Sobreposição das curvas de velocidade pretendida, leituras de velocidade durante o teste 7 e representação da função de transferência calculada.

Como podemos observar, a função de transferência calculada, representa de forma satisfatória o comportamento do sistema, uma vez que, observando os resultados obtidos nos testes, verifica-se uma grande variabilidade na leitura da velocidade, sendo por isso necessário um controlador simples para o controlo do movimento do drone.

5.2.3 Controlador Proporcional

Determinada a função de transferência é possível dimensionar o controlador proporcional. Para isso, foi construído em ambiente *Simulink*, uma ferramenta do *software Matlab* utilizada para simulação e análise de sistemas de controlo, uma simulação para o valor do controlador proporcional. Na figura 5.12 podemos observar o sistema de controlo.

O erro é medido pela distância até ao alvo e não pela velocidade, e como tal, à saída da função de transferência é colocado um integrador, obtendo-se assim a distância em função dos valores de inclinação gerados pelo controlador. Na saída é também colocado um ganho de valor 0.001 visto que a velocidade está representada em mm/s e o cálculo do erro é feito em metros.

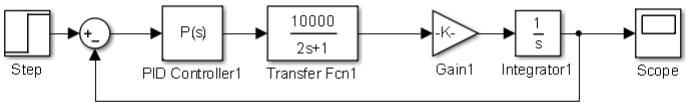


Figura 5.12: Sistema de controlo

A utilização de um controlador proporcional é adequado ao problema em questão, na medida em que o sistema é sempre estável uma vez que este apresenta os dois pólos em malha aberta no semiplano esquerdo, em $-\frac{1}{2}$ e na origem, como podemos observar pela função:

$$G(s) = \frac{5}{s(s+\frac{1}{2})} \tag{5.11}$$

Construindo o lugar geométrico de raízes para esta função em malha fechada podemos constatar que o *Kp* que permite um tempo de subida mais rápido sem causar *overshoot* é o valor que coloca os pólos no eixo real no valor $-\frac{1}{4}$. Este valor de *Kp* é 0,014. A função em malha fechada é representada pela seguinte expressão:

$$T(s) = \frac{Kp \times \frac{10000}{2s+1} \times K' \times \frac{1}{s}}{1 + Kp \times \frac{10000}{2s+1} \times K' \times \frac{1}{s}} = \frac{5Kp}{s^2 + \frac{1}{2}s + 5Kp} \tag{5.12}$$

No entanto este valor de *Kp* apresenta um tempo de subida demasiado grande como podemos observar na figura 5.13. De modo a melhorar este comportamento admitiu-se um valor máximo de *overshoot* de 10%.

O valor de *overshoot* em sistemas de segunda ordem é calculado por:

$$ln(overshoot) = \frac{-\xi\pi}{\sqrt{1-\xi^2}} \tag{5.13}$$

Utilizando esta expressão verificamos que para um valor de *overshoot* de 10% ξ, o coeficiente de amortecimento, toma um valor igual a 0.58. Uma vez que o sistema de segunda ordem é descrito por:

$$T'(s) = \frac{wn^2}{s^2 + 2\xi ws + w^2} \tag{5.14}$$

Podemos obter Kp pela seguinte expressão:

$$s^2 + 2\xi ws + w^2 = s^2 + \frac{1}{2}s + 5Kp \tag{5.15}$$

Deste modo podemos extrapolar o valor de *Kp*. *Kp* terá um valor de 0,037. Na figura 5.13 podemos observar o comportamento do sistema para valores de $Kp = 0.014$ e $Kp = 0.037$.

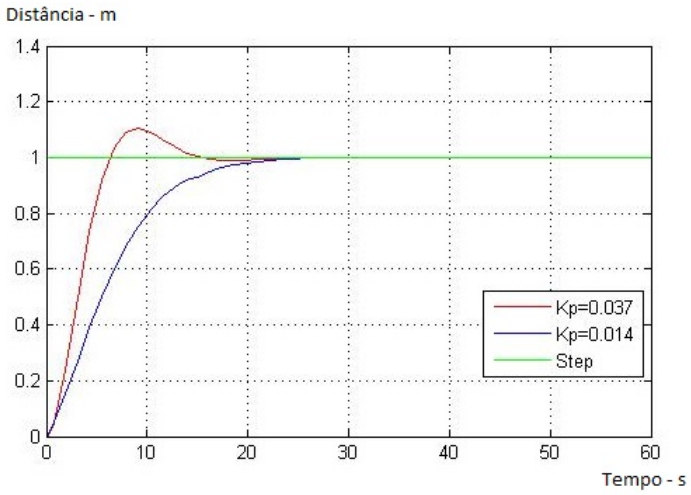


Figura 5.13: Comportamento do sistema para dois valores de Kp, 0,014 e 0,037

5.3 Processo de aproximação

Estabelecido o processo para estabilizar o drone no centro do padrão foi projetado um método para realizar a descida e deste modo completar a aterragem com sucesso.

Como não é possível ser preciso ao ponto de drone estar perfeitamente alinhado com o padrão, implementou-se uma zona de segurança para a descida. Como podemos observar na figura 5.14, quando o centro do padrão se encontra dentro da zona delimitada pelo círculo verde, o drone pára o seu movimento horizontal e procede à descida. A descida é efetuada a uma velocidade de 350mm/s. Se por algum motivo o drone sai desta zona é suspensa a descida e é realizado o controlo horizontal novamente.

Para concluir a aterragem, quando o drone se encontra a uma altura de 30 cm da plataforma, são desligados os motores, uma vez que a proximidade da plataforma torna o controlo do drone bastante difícil. Isto não acarreta nenhum problema, na medida em que o drone já se encontra suficientemente próximo da plataforma para aterrar e está preparado para uma queda desta altura.

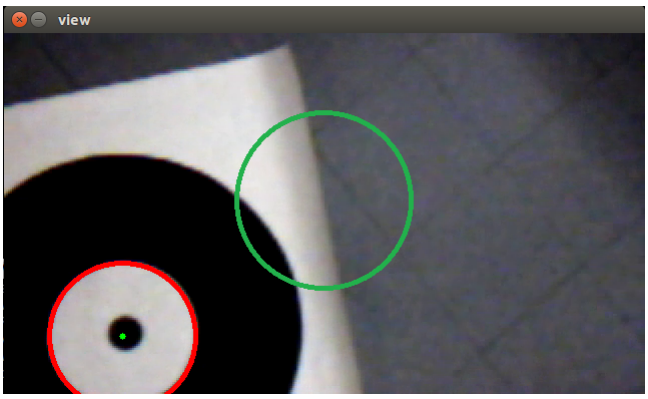


Figura 5.14: Zona de segurança para a descida

Foi finalmente implementado um procedimento de segurança caso o drone saísse do alcance do padrão por algum motivo. Como visto anteriormente no capítulo 3, verifica-se que o tempo máximo entre deteções é cerca de 112 ms. Tendo isto em consideração, se durante 2s não existir deteção é ativado o estado de segurança. Como sabemos a última posição detetada do padrão, sabemos em que zona ele se encontra, deste modo é enviado um comando para o drone se deslocar lentamente para essa zona enquanto sobe, também a uma velocidade mais lenta. Ao subir lentamente estamos a aumentar o campo de visão da câmara e assim facilitar a deteção do padrão. Detetado de novo o padrão é repetido o processo de aterragem. Se o drone durante este processo atingir a altitude de aproximação definida na navegação utilizando as coordenadas GPS, significa que o drone se afastou consideravelmente do catamarã, neste caso é utilizado de novo o sistema de navegação GPS.

O drone no fim da manobra de aterragem comunica com o catamarã através da publicação de uma mensagem *True*, do tipo booleana, no tópico *landing*.

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Satisfação dos Objetivos

Projetos que ambicionaram a realização de missões com recurso a um drone e a uma embarcação para a inspeção, monitorização e vigilância de ambientes costeiros são, cada vez mais, uma realidade. As soluções atualmente apresentadas, ficam, contudo, ainda aquém das expectativas, na medida em que não são, para já, completamente autónomas.

O presente projeto pressupôs a correção de um protocolo perspetivado para a exequibilidade de missões autónomas, através da utilização de um drone comercial *lowcost*.

O cumprimento dos objetivos que a seguir se elencam, contribui para a concretização da presente dissertação.

- Projeto de uma plataforma de aterragem;
- Dimensionamento de um algoritmo para a deteção do padrão presente na plataforma;
- Protocolo de comunicação com o drone;
- Conceção de um algoritmo para determinar a viabilidade de o drone realizar a missão tendo em consideração a percentagem de energia restante na bateria;
- Extração da posição real do drone relativamente à plataforma de aterragem utilizando apenas a câmara vertical;
- Dimensionamento de um controlador proporcional para o controlo do drone;
- Conceção de um algoritmo de aterragem na plataforma.

Do trabalho encetado, retiram-se as seguintes conclusões.

O padrão proposto neste documento apresenta resultados bastante positivos, como podemos observar nos testes efetuados, provando a viabilidade de aplicar o padrão na plataforma de aterragem e assim realizar o controlo no drone baseado nessa deteção.

A escolha do material utilizado no padrão demonstrou ser bastante relevante, uma vez que a deteção do padrão exibiu claras debilidades face à reflexão de luz no mesmo.

A análise da descarga de bateria permitiu concluir que a bateria presente no drone não permite voos de duração superiores a 6 minutos e 17 segundos, não sendo por isso adequado para missões que exijam percorrer grandes distâncias.

O cálculo da função de transferência demonstrou ser o processo mais desafiante desta dissertação, na medida em que se verificou durante a leitura de dados ruído significativo, tornando difícil realizar uma aproximação do sistema. O ruído detetado durante estas medições demonstra a grande sensibilidade do drone a pequenas perturbações.

Neste documento são apresentados algoritmos para o dimensionamento de um projeto deste tipo, nomeadamente a conceção do padrão presente na plataforma, análise da descarga de energia da bateria, confirmação da precisão do módulo GPS, identificação do sistema e do controlador proporcional.

O *software ROS* escolhido para a implementação demonstrou-se ideal para este projeto, na medida em que proporcionou todas as ferramentas necessárias para a realização do mesmo.

Após a conclusão deste trabalho é possível observar duas principais contribuições.

A primeira consiste num algoritmo que permite a deteção do padrão e a exclusão de falsos positivos. Neste documento é explanado o processo de deteção de círculos utilizando a *Circle Hough Transform*, nomeadamente a previsão do raio dos círculos do padrão na imagem capturada pelo drone, sendo assim possível excluir círculos detetados na imagem que não possuem os raios previstos.

A segunda principal contribuição baseia-se no processo de dimensionamento de um controlador proporcional para um drone que apresenta grande variabilidade de leitura de medidas, nomeadamente de velocidade. Para a identificação do sistema foram propostos testes específicos com o objetivo de perceber o comportamento do mesmo. Devido à grande variabilidade das leituras, como podemos constatar pelos testes efetuados, foi proposto um controlador proporcional, na medida em que um controlador mais complexo exigiria uma abordagem para a identificação do sistema também ela mais complexa.

6.2 Trabalho Futuro

Em termos de trabalho futuro, o documento final prevê uma flexibilidade de manobra suficiente para que algumas linhas de desenvolvimento possam vir a ser alvo de interesse, nomeadamente uma análise mais extensiva do comportamento de descarga de energia da bateria do drone, nomeadamente para diferentes valores de velocidade.

Relativamente à fase de aterragem seria interessante integrar um algoritmo que permitisse prever a posição do drone relativamente à plataforma de aterragem, corrigindo assim a sua posição caso não seja detetado o padrão. Esta previsão seria possível, uma vez que tanto a última posição detetada do padrão como os valores de velocidade enviados podem ser obtidos. Ainda sobre esta

fase, poderia-se desenvolver um protocolo para situações em que o drone não detete a plataforma no momento em que chega às coordenadas GPS do catamarã.

No que toca à plataforma de aterragem, seria interessante conceber um padrão e desenvolver um algoritmo de deteção que permitisse obter a orientação do drone face ao catamarã, sendo deste modo possível efetuar a descida numa perspetiva pré-determinada.

Por fim, outra linha de desenvolvimento seria aperfeiçoar o controlador proporcional. Neste sentido, seria realizada uma identificação do sistema utilizando outro tipo de abordagem, que não envolva os sensores de velocidade do drone, uma vez que estes demonstraram grande variabilidade nos resultados obtidos.

Bibliografia

[1] Courtney S Sharp, Omid Shakernia e S Shankar Sastry. “A vision system for landing an unmanned aerial vehicle”. Em: *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*. Vol. 2. Ieee. 2001, pp. 1720–1727.

[2] Z Yuan, Z Gong, J Chen e J Wu. “A vision-based method for autonomous landing of a rotor-craft unmanned aerial vehicle”. Em: *Applied Bionics and Biomechanics* 3.3 (2006), pp. 171–177.

[3] Sven Lange, Niko Sünderhauf e Peter Protzel. “Autonomous landing for a multirotor UAV using vision”. Em: *SIMPAR 2008 Intl. Conf. on Simulation, Modeling and Programming for Autonomous Robots*. 2008, pp. 482–491.

[4] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler e Andrew Y Ng. “ROS: an open-source Robot Operating System”. Em: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe. 2009, p. 5.

[5] Michael Lindemuth, Robin Murphy, Eric Steimle, William Armitage, Karen Dreger, Tim Elliot, Michael Hall, Dmitry Kalyadin, Jeffrey Kramer, Mayur Palankar et al. “Sea robot-assisted inspection”. Em: *IEEE robotics & automation magazine* 18.2 (2011), pp. 96–107.

[6] Matthew Dunbabin e Lino Marques. “Robots for environmental monitoring: Significant advancements and applications”. Em: *IEEE Robotics & Automation Magazine* 19.1 (2012), pp. 24–39.

[7] Ricardo Mendonça, Pedro Santana, Francisco Marques, André Lourenço, Joao Silva e José Barata. “Kelpie: A ROS-based multi-robot simulator for water surface and aerial vehicles”. Em: *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*. IEEE. 2013, pp. 3645–3650.

[8] Roman Barták, Andrej Hrasko e David Obdrzálek. “On Autonomous Landing of AR. Drone: Hands-On Experience.” Em: *FLAIRS Conference*. 2014.

[9] Eduardo Pinto, Francisco Marques, Ricardo Mendonça, André Lourenço, Pedro Santana e José Barata. “An autonomous surface-aerial marsupial robotic team for riverine environmental monitoring: Benefiting from coordinated aerial, underwater, and surface level perception”. Em: *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*. IEEE. 2014, pp. 443–450.

[10] Eduardo Pinto, Pedro Santana, Francisco Marques, Ricardo Mendonça, André Lourenço e José Barata. “On the design of a robotic system composed of an unmanned surface vehicle and a piggybacked vtol”. Em: *Doctoral Conference on Computing, Electrical and Industrial Systems*. Springer. 2014, pp. 193–200.

[11] Joao Silva, Ricardo Mendonça, Francisco Marques, Paulo Rodrigues, Pedro Santana e José Barata. “Saliency-based cooperative landing of a multirotor aerial vehicle on an autonomous surface vehicle”. Em: *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*. IEEE. 2014, pp. 1523–1530.

[12] *ardrone_autonomy — ardrone_autonomy indigo-devel documentation*. http://ardrone-autonomy.readthedocs.io/en/latest/. Accessed: 2017-02-7.

[13] *camera_calibration - ROS Wiki*. http://wiki.ros.org/camera_calibration. Accessed: 2017-02-7.

[14] *GitHub - AutonomyLab/ardrone_autonomy: ROS driver for Parrot AR-Drone 1.0 and 2.0 quadrocopters*. https://github.com/AutonomyLab/ardrone_autonomy. Accessed: 2017-02-7.

[15] *OpenCV | OpenCV*. http://opencv.org/. Accessed: 2017-02-7.

[16] *ROS.org | Powering the world’s robots*. Accessed: 2017-02-7. URL: http://www.ros.org.

[17] *vision_opencv - ROS Wiki*. http://wiki.ros.org/vision_opencv. Accessed: 2017-02-7.