MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

CARLA SZE COSENZA
LUCAS BASTOS GERMANO

CONTROLE DE DRONES ATRAVÉS DO RECONHECIMENTO DE IMAGENS DO TERRENO

RIO DE JANEIRO
2020

CARLA SZE COSENZA
LUCAS BASTOS GERMANO

CONTROLE DE DRONES ATRAVÉS DO RECONHECIMENTO DE IMAGENS DO TERRENO

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia da Computação.

Orientador(es): Paulo Fernando Ferreira Rosa, Ph.D.
Erick Menezes Moreira, D.Sc.
Marlos de Mendonça Corrêa, M.Sc.

Rio de Janeiro

2020

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP: 22290-270

Cosenza, Carla Sze; Germano, Lucas Bastos.
Controle de drones através do reconhecimento de imagens do terreno / Carla Sze Cosenza e Lucas Bastos Germano. – Rio de Janeiro, 2020.
63 f.

Orientador(es): Paulo Fernando Ferreira Rosa, Erick Menezes Moreira e Marlos de Mendonça Corrêa.

Projeto de Final de Curso (graduação) – Instituto Militar de Engenharia, Engenharia da Computação, 2020.

1. navegação autônoma. 2. visão computacional. 3. scene matching. i. Rosa, Paulo Fernando Ferreira (orient.) ii. Moreira, Erick Menezes (orient.) iii. Corrêa, Marlos de Mendonça (orient.) iv. Título

CARLA SZE COSENZA
LUCAS BASTOS GERMANO

Controle de drones através do reconhecimento de imagens do terreno

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia da Computação.

Orientador(es): Paulo Fernando Ferreira Rosa, Erick Menezes Moreira e Marlos de Mendonça Corrêa.

Aprovado em Rio de Janeiro, 29 de outubro de 2020, pela seguinte banca examinadora:

Prof. **Paulo Fernando Ferreira Rosa** - Ph.D. do IME - Presidente

---

Prof. **Erick Menezes Moreira** - D.Sc. do IME

Prof. **Marlos de Mendonça Corrêa** - M.Sc. do CTEx

Prof. **Ricardo Choren Noya** - D.Sc. do IME

Prof. **Julio Cesar Duarte** - D.Sc. do IME

Rio de Janeiro
2020

Este trabalho é dedicado a todos os nossos amigos e família,
que nos ajudaram a chegar até aqui.

AGRADECIMENTOS

Agradecemos a todas as pessoas que nos incentivaram, apoiaram e possibilitaram esta oportunidade de ampliar nossos horizontes. Somos especialmente gratos às pessoas que estavam conosco diariamente nessa jornada: Oracília, Carlos, Jose Carlos, Salpen, Sarah, Lucas, Matheus Mello, Karol, Mateus e Roomie.

Nossos familiares e mestres que nos deram boas memórias e ensinamentos durante esses 5 anos, dos quais iremos lembrar para sempre.

Agradecemos ao Yugo Nihari por nos ajudar a montar o componente de espelho para o drone, que foi crucial para o trabalho.

E por fim, aos nossos Orientadores Paulo Rosa, Maj Menezes e Maj Mendonça, por suas disponibilidades e atenções, até mesmo em horários que deveriam estar com suas famílias, sempre nos ajudavam no que fosse preciso.

"Se vi mais longe foi por estar sobre os ombros de gigantes"
(Isaac Newton, 1676)

RESUMO

A indústria de drones já possuí um valor bilionário, e mesmo assim prevê um grande crescimento nos próximos anos. Isto é devido ao fato que essas aeronaves estão cada vez mais sendo utilizadas para realização de tarefas comerciais, como entregar compras e monitorar plantações. Conseguir navegar é uma tarefa essencial para estes robôs e suas novas funcionalidades estão exigindo cada vez mais dessa tecnologia. Atualmente, vem-se confiando demasiadamente na utilização dos sistemas de geolocalização por satélite; todavia, seu erro de aproximação impossibilita tarefas de localização com requisitos de acurácia adequados e sua falta de estabilidade, em certos ambientes, demanda outras tecnologias para garantir a movimentação precisa de aeronaves remotamente pilotadas. Este trabalho propõe uma solução para a navegação através do reconhecimento de imagens do terreno. Nele, um veículo em movimento é equipado com uma câmera para capturar imagens e um algoritmo de *scene matching* identifica pontos chaves dessas imagens, relacionando-as com um conjunto de imagens previamente adquirido, localizando o drone no terreno. Os resultados foram avaliados em uma simulação de pequena escala de uma base petrolífera real, proveniente do desafio Petrobras, que é motivado por tarefas de interesse da indústria de óleo e gás. Obteve-se um software, que gera a navegação correta em ambientes fechados com controle de luz e um mapa pequeno.

Palavras-chave: navegação autônoma. visão computacional. scene matching.

ABSTRACT

The drone industry is already worth billions and studies still predict a considerate growth in the upcoming years. This is due to the fact that these aircrafts are increasingly being used to perform jobs such as delivering good and monitoring plantations. Being able to navigate is an essential task for these robots, and their new functionalities are demanding more from this technology. Geolocation systems have been deeply replied on, though their estimation error makes it impossible to perform tasks that require precision and their lack of stability in certain environments shows the necessity of needing other technologies to guarantee the precise movement of remotely piloted aircrafts. This project presents a solution for navigation through the recognition of images of the terrain. The robot in movement is equipped with a camera to capture images and a scene matching algorithm identifies key points of that image, matching them with a set o previously acquired images, locating the drone the in terrain. The obtained results were evaluated in a small scale simulation of a real oil base, based on the Petrobras challenge, which is inspired by real tasks of interest of the oil and gas industry. A robust software was obtained, that generates the correct navigation for close environments with light control and a small map.

Keywords: autonomous navigation. computer vision. scene matching.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IME	Instituto Militar de Engenharia
CTEx	Centro Tecnológico do Exército
SDK	Standard Development Kit
PC	Personal Computer
GPS	Global Positioning System
GNSS	Global Navigation Satellite Systems
SURF	Speed Up Robust Feature
SIFT	Scale Invariant Feature Transform
BRIEF	Robust Independent Elementary Features
FAST	Features from Accelerated Segment Test
ORB	Oriented FAST, Rotated BRIEF
IMU	Inertial Measurement Unit
API	Application Programming Interface
DoG	Difference of Gaussians
UDP	User Datagram Protocol
GPU	Graphics Processing Unit
FPS	Frames per second

LISTA DE SÍMBOLOS

Σ	Letra grega maiúscula sigma
σ	Letra grega minúscula sigma
$\in$	Relação de pertinência

SUMÁRIO

1 INTRODUÇÃO

Os veículos aéreos não tripulados (VANTs), também conhecidos como aeronave remotamente pilotadas (ARPs), ou ainda drones, são aeronaves que não necessitam de um piloto humano a bordo, podendo ser controladas por um computador a bordo, a distância por um operador humano ou por outro computador (1).

A história dos drones data desde de 1849, quando foi usado através de balões por forças austríacas contra a cidade de Veneza (2), porém nessa época ainda não existia a tecnologia necessária para criar nenhum tipo de inteligência embarcada nas aeronaves. Foi somente a partir do começo do século XX que as inovações começaram, momento onde foram desenvolvidas as primeiras aeronaves rádio-controladas e também os primeiros drones para espionagem.

Foi também no fim do século XX que os primeiros sistemas de navegação global por satélite (GNSS) surgiram. Vários países investiram muito nessa tecnologia, que hoje é utilizada mundialmente. Os GNSS mais conhecidos são:

- GPS: criado pelos Estados Unidos e globalmente operacional desde 1994, a constelação conta com 31 satélites em órbita, sendo o GNSS mais utilizado no mundo atualmente
- GLONASS: desenvolvido inicialmente pela União Soviética e mantido pela Rússia, teve seu primeiro lançamento de satélites em 1982, mas só vindo alcançar cobertura global em 2011
- BeiDou: o sistema chinês começou a ser lançado em 2000 e em 2020 se tornou globalmente operacional quando a fase 3 foi finalizada
- Galileo: projeto realizado pela União Européia e pela Agência Espacial Européia, começou a ser operacional em 2016, porém, até o ano de 2020, ele não possui total cobertura global

Apesar dos drones serem inicialmente idealizados para fins militares, eles ganharam os mais diversos fins com o advento da tecnologia e da industrialização, sendo utilizados em, por exemplo: fotografia profissional, lazer, agricultura e vigilância (3).

Pode-se considerar então, que os drones possuem um papel importante no ambiente tecnológico militar e civil, já que eles são uma opção segura de se realizar tarefas, pois não põe em risco vidas humanas.

1.1 Motivação

Dada essa importância que os drones trouxeram para o mundo civil e militar, eles se tornaram um tópico de pesquisa essencial em todo o mundo, sendo utilizados nas mais diversas áreas de pesquisa.

Uma da áreas que mais está crescendo atualmente é a da movimentação autônoma de drones, onde os drones não são controlados por um operador humano. Isso necessita que os drones usem outra forma para se localizar com precisão, como utilizar um dos sistemas de navegação global por satélite (GNSS).

Por diversos motivos, a utilização do GPS ou qualquer outro serviço de geolocalização por satélites pode não ser viável ou eficiente, por exemplo: realização de tarefas em ambientes fechados, onde o sinal dos satélites é fraco; interferência no sistema ou ainda pode existir a impossibilidade de usar o serviço de satélites de outros países por motivos políticos. Por essas razões, faz-se necessário a pesquisa de outras formas para se garantir a eficiência e a precisão na movimentação das aeronaves não tripuladas, sendo uma dessas formas a navegação através do reconhecimento de imagens do terreno, tema central deste trabalho.

1.2 Objetivos

O presente projeto teve por objetivo elaborar, implementar e testar um sistema de localização baseado em imagens para um drone de pequeno porte com payload reduzido.

O sistema de navegação utiliza o algoritmo *Scale-Invariant Feature Transform* (SIFT) para fazer o reconhecimento das imagens dos locais predeterminados, localizando o drone no mapa de referência, a partir das imagens adquiridas pelo drone no voo. O SIFT é um algoritmo de *scene matching* baseado na identificação de pontos chaves, que foi usado para procurar o local que o drone está, usando as imagens capturadas pela sua câmera, no mapa. Dessa forma, é possível encontrar a posição do drone no mapa de coordenadas, que é passado como entrada. Com esse dado, o sistema escolhe o destino do veículo e determina *waypoints* a serem seguindos, fazendo assim a navegação. A figura 1 mostra esse procedimento. O VANT voa pela região delimitada pelo mapa, capturando imagens usando sua câmera. Essas imagens são passadas para o sistema, que usa o algoritmo SIFT para procurar sua localização no mapa, assim determinando a posição do drone e permitindo a determinação dos *waypoints* a serem seguidos.

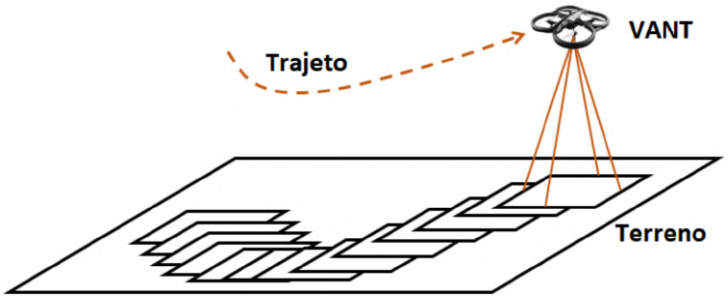


Figura 1 – Diagrama da navegação por imagens. Adaptado de: Luiz Junior(1).

1.3 Justificativa

O cenário mundial indica que o mercado de drones ainda irá crescer bastante nos próximos anos, paralelo a isso, também espera-se que no Brasil o desenvolvimento científico, na área de drones, ainda expanda muito. Como se pode observar em outras potências militares, como Estados Unidos, Rússia e China, que fazem um largo emprego de drones em atividades militares, espera-se que, num futuro próximo, os drones estejam envolvidos em diversas atividades militares de âmbito nacional, como: monitoramento de fronteiras e mapeamento de terrenos.

Pensando nessa possibilidade de expandir a produção nacional, científica e militar, na área de drones, surgiu um projeto do Instituto Militar de Engenharia (IME) em parceria com o Centro Tecnológico do Exército (CTEx), que trabalha justamente em uma das áreas mais importantes desse meio, que é a do controle de drones.

O conceito do projeto é implementar um sistema de navegação por meio de imagens obtidas diretamente da câmera do drone, fazendo-se que não tenha a necessidade de ter um sistema de geolocalização por satélites embarcado no drone ou que se esse sistema, como o GPS, não estiver disponível em certas áreas, o drone ainda possa se movimentar de forma precisa.

Analisando esse contexto, verifica-se que a movimentação por imagens do terreno é uma importante ferramenta no controle de drones, podendo ser aplicada também em conjunto com os GNSS. Além disso, esse sistema pode ser empregado tanto no meio militar quanto no civil, pois o controle preciso dos drones é necessário em qualquer meio que os empreguem.

1.4 Contribuições

A principal contribuição desse trabalho é um sistema de navegação baseado em imagens que pode ser adaptado para outros veículos e locais. Com ele é possível navegar em um mapa de referência da área de interesse, a partir de imagens adquiridas pelo drone durante o voo. Todo o código desenvolvido está disponibilizado, permitindo sua utilização

e adaptação a outros cenários e por outros colaboradores.

Além disso, este trabalho foi fruto de uma parceria do IME com o CTEx, deixando assim como contribuição um compartilhamento de conhecimento entre as duas instituições.

Esta monografia também permitiu aprimorar o conhecimento do grupo de pesquisa em VANTs e algoritmos de *scene matching* do laboratório de robótica e inteligência computacional do IME, contribuindo em seus projetos de cooperação internacional com outros centros de pesquisa.

1.5 Metodologia

O projeto está dividido em duas partes: a localização por visao computacional e a navegação. A navegação é responsável por fazer o deslocamento preciso do drone entre dois pontos, controlando a velocidade, aceleração e altitude. Já a localização por visao computacional é responsável pelos algoritmos de *scene matching* que serão executados durante o voo para localizar o drone em uma fotografia de referência.

Para avaliar a navegação, foram realizados três experimentos, cada um com uma foto de referência diferente, visando avaliar a precisão do voo com o algoritmo de localização habilidado. Em cada experimento, realizou-se a navegação sem a utilização das imagens e com a utilização das imagens, assim podendo comparar a precisão do trajeto realizdo.

1.6 Estrutura da monografia

No capítulo 2, é apresentada a fundamentação teórica, com os conceitos chaves relacionados a este trabalho, como as especificações do drone, os algoritmos utilizados e os resultados de trabalhos relacionados a este. No capítulo 3, encontra-se uma descrição do desafio proposto pela Petrobras, cujo cenário inspirou os experimentos realizados. O capítulo 4 descreva a proposta para a solução do problema. No capítulo 5 são apresentados os experimentos realizados e seus resultados. Por último, no capítulo 6, encontra-se as conclusões, com considerações, dificuldades encontradas e perspectivas futuras.

2 REFERENCIAL TEÓRICO E REVISÃO DE LITERATURA

Neste capitulo, serão apresentados os conceitos que foram usados para o desenvolvimento do projeto, que será apresentado nos capítulos seguintes. Seu objetivo é ambientar o leitor nos tópicos teóricos e práticos nos quais o estudo de alicerça. Além disso, esse capítulo também contém a revisão de literatura que foi utilizada como base para este projeto.

2.1 Tópicos tutoriais

Nesta seção, são abordados os principais tópicos teóricos que foram utilizados no desenvolvimento deste trabalho. De forma resumida, os temas encontradas nesta seção são:

- *Scene Matching* e os algoritmos utilizados neste trabalho
- As bibliotecas essenciais para o desenvolvimento do trabalho
- O drone utilizado, assim como a *Application Programming Interface* (API) usada para controlá-lo

2.1.1 Scene Matching

No decorrer dos anos, houve um aumento nos estudos e desenvolvimento de robótica e câmeras de alta resolução. Devido a isso, surgiu a necessidade de ter uma visão computacional parecida com a de um humano, capaz de entender uma cena e fazer comparações. Esta área de pesquisa tem produzido uma serie de técnicas para alcançar este objetivo, que são capazes de analisar movimento, reconstruir uma cena, restaurar uma cena, comparar imagens, etc (4).

Neste projeto, o foco será em técnicas que resolvem o problema de registro de imagens, também conhecido como *scene matching*. *Scene matching* é o grupo de problemas na qual se quer descobrir qual região de uma imagem descreve a mesma cena que outra imagem. O objetivo é conseguir identificar a mesma cena em imagens diferentes, que podem ter características diferentes devido ao ângulo na qual a imagem foi feita, a iluminação do momento, reflexão de objetos, etc. Uma das soluções mais simples para isso é chamado *template matching* (5).

A figura 2 exemplifica o resultado do *scene matching*. A imagem da esquerda é a imagem base enquanto a imagem da direita é a imagem a ser comparada. O problema é conseguir encontrar a correspondência do que está sendo mostrado na imagem base na imagem da direita. Pode-se notar que há o registro da imagem perceptível pelo retângulo vermelho, indicando a região da imagem a ser comparada que corresponde à imagem base.

Figura 2 – Resultado de um algoritmo de *scene matching*. Fonte: Juan, Qing-song e Jing-hua(6).

2.1.2 Scale-invariant feature transform (SIFT)

Existem vários algoritmos para *scene matching*, cada um com suas vantagens e desvantagens. Para este projeto, a técnica utilizada é baseada no *Scale-Invariant Feature Transform* (SIFT), um algoritmo baseado em pontos chaves (que será descrito com maior detalhes na seção 2.2.1).

O SIFT foi desenvolvido especificamente para ser invariante a escala, como seu próprio nome diz, pois naquela época, já haviam algoritmos que eram invariantes a rotação, porém não a escala.

O algoritmo possui cinco passos principais: detecção de extremos, localização precisa de pontos chaves, atribuição da orientação dos descritores, construção do descritor local e casamento de pontos em comum (7). Cada passo será explicado nas próximas subseções.

2.1.2.1 Detecção de extremos

Como o SIFT é invariante a escala, o primeiro passo do algoritmo é procurar por pontos que sejam invariantes a mudanças de escala. Esses pontos serão possíveis pontos chaves do algoritmo. Para fazer isso, procura-se características estáveis em diferentes escalas, usando uma função chamada de *scale space*. No caso do SIFT, essa função é a Gaussiana.

Logo, uma imagem $I(x, y)$ pode ser representada por $L(x, y, \sigma)$, sendo calculada por:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.1}$$

onde $G(x, y, \sigma)$ representa a função Gaussiana na escala σ.

Procurar pontos chaves nessas representações seria muito custoso computacio-

nalmente, portanto o SIFT faz uma adaptação do cálculo do $L(x, y, \sigma)$, calculando sua aproximação usando a função *Difference of Gaussians* (DoG), que é formada pela diferença de imagens em escalas próximas, separadas por uma constante k. Esse processo é feito para diferentes oitavos da imagem na pirâmide Gaussiana, como mostra a figura 3.

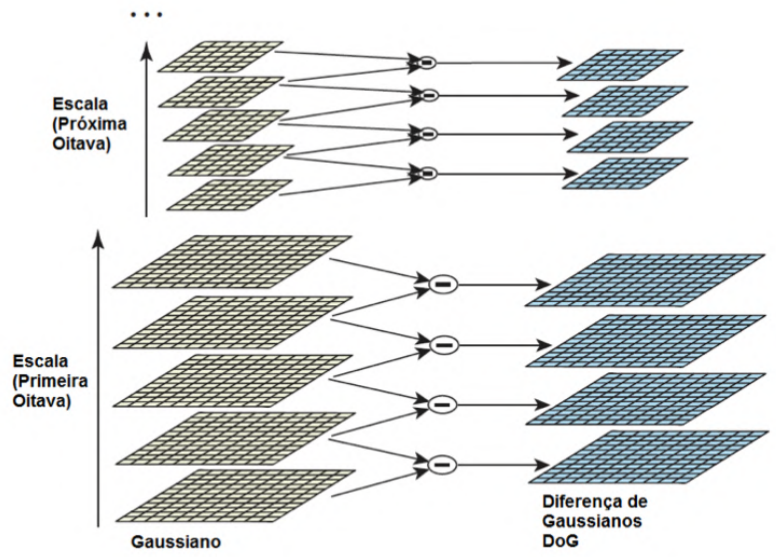


Figura 3 – Processo de obtenção do DoG. Fonte: PUC-Rio(8).

Após calcular as DoGs, a procura de pontos de mínimos e máximos locais é realizada. Porém, como a escala entra como dimensão da figura, a comparação para a verificação desses pontos é realizada com seus 8 píxels vizinhos, mas também com seus 9 píxels vizinhos na escala anterior e na próxima escala, como mostra a figura 4 (7).

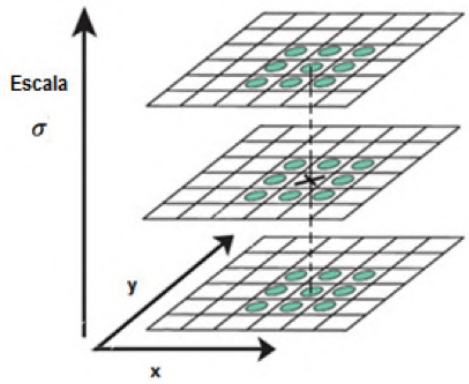


Figura 4 – Análise dos pontos de mínimos e máximos locais. Fonte: PUC-Rio(8).

2.1.2.2 Localização de pontos chaves

O passo descrito anteriormente serve para identificar pontos que serão invariantes a escala, porém pode haver uma quantidade muito grande de pontos que satisfazem essa

condição, portanto é necessário escolher quais desses pontos realmente serão usados como pontos chaves. A partir do conjunto de pontos gerados para o passo, primeiro remove-se os pontos que estão perto da borda da imagem, usando a função Hessiana. Depois, remove-se os pontos que possuem um contraste baixo, calculado o seu contraste em relação aos píxels ao seu redor. Os pontos restantes são utilizados como pontos chaves (7).

2.1.2.3 Atribuição da orientação

Para cada ponto chave é necessário calcular a sua orientação para que o algoritmo possa ser invariante a rotação. Isso é feito analisando píxels vizinhos. A magnitude do gradiente é calculada para esses pontos e a média ponderada desses valores é usado como orientação do ponto chave (7).

2.1.2.4 Construção do descritor local

Como o ponto chave está fortemente ligado aos seus vizinhos, é necessário fazer a descrição deles também. Para isso uma região de 16x16 ao redor do ponto é levada em consideração. Essa região é quebrada em 16 blocos, e para cada bloco se calcula sua orientação. Essas orientações formarão o descritor local (7).

2.1.2.5 Casamento de pontos chaves

Após realizar os primeiros quatro passos, obtém-se os pontos chaves da imagem, logo só é necessário fazer o registro correto, ou *matching*, com os pontos das imagens analisadas. Vale ressaltar que é necessário fazer os quatro passos anteriores para as duas imagens a serem comparadas.

Uma das formas mais simples de fazer isso é pela localização dos vizinhos mais próximos. Se dois pontos chaves são similares, analisa-se os seus vizinhos, medindo sua distância e seu grau de similaridade (7).

2.1.3 Speeded up robust features (SURF)

O *Speeded-Up Robust Feature* (SURF), é outro algoritmo de *scene matching* baseado em pontos chaves. Os três passos principais do algoritmo são: detectar pontos chaves, descrever os pontos chaves e casar os pontos chaves. No diagrama 5 é possível ver o fluxo do algoritmo, desde a aplicação dos passos de detecção de pontos chaves em cada imagem até o registro das duas (6). Cada passo será explicado nas próximas subseções.

2.1.3.1 Detectando pontos chaves

Como mencionado anteriormente, o SURF é baseado em pontos chaves. Esses pontos são caracterizados por ter um alto contraste quando comparado aos seus vizinhos,

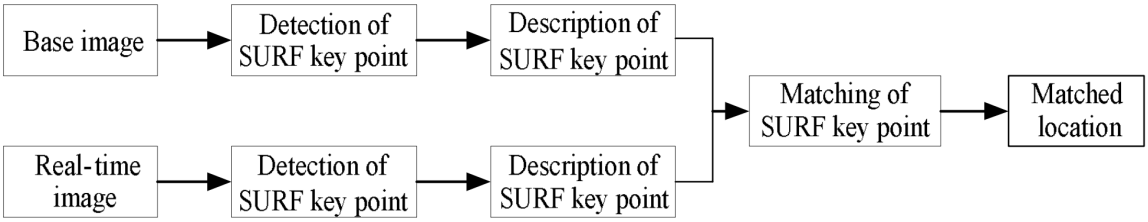


Figura 5 – Diagrama mostrando o fluxo do algoritmo SURF. Fonte: Juan, Qing-song e Jing-hua(6).

tornando-se pontos destaques na imagem. Eles são calculados de forma similar ao SIFT. O cálculo é feito da seguinte maneira: dado um ponto $X = (x, y)$ que está contido na imagem I, sua matriz Hessiana na escala σ é dada por:

$$H(X, \sigma) = \begin{bmatrix} L_{xx}(X, \sigma) & L_{xy}(X, \sigma) \\ L_{yx}(X, \sigma) & L_{yy}(X, \sigma) \end{bmatrix} \tag{2.2}$$

onde $L_{xx}(X, \sigma)$ representa a convolução da derivada de segunda ordem da Gaussiana da imagem I no ponto X, analogamente para o resto (6). Qualitativamente, a Hessiana descreve como que o ponto está mudando em relação ao seus vizinhos, mesmo que haja variação de escala, pois a convolução considera esse elemento. Essa forma de calcular os pontos pode ser um pouco demorada, por isso recorre-se a um filtro de caixa de tamanho 9x9, que são aproximações de uma Gaussiana com $\sigma = 1,2$ e representam a menor escala para computar mapas de resposta de *blobs*. Eles serão representandos por D_{xx}, D_{yy} e D_{xy}. Assim, pode-se simplificar o cálculo do determinante da Hessiana a equação 2.3, onde w representa o peso relativo da resposta do filtro.

$$det(H_{aproximado}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{2.3}$$

Através de cálculos e experimentos, foi concluído que $w = 0,9$ pode ser usando sem impacto nos resultados. Logo, a equação final é:

$$det(H_{aproximado}) = D_{xx}D_{yy} - (0,9 * D_{xy})^2 \tag{2.4}$$

O determinante aproximado da matriz Hessiana representa a resposta do *blob* na posição X da imagem. Esses valores são guardados e seus pontos de mínimo local são utilizados para fazer a comparação entre as imagens (9).

2.1.3.2 Descrevendo os pontos chaves

Para descrever os pontos, analisa-se a orientação dominante ao redor do ponto chave. Para isso, pega-se uma região de tamanho 20 píxels. Essa região é então dividida em

quatro blocos 4x4 e em cada um é calculada a resposta de wavelet dx e dy e depois esses valores são somados ($dx + dy$). Também são analisados os valores absolutos dessa resposta. Logo, para cada bloco, um vetor de tamanho quatro é calculado, como o representando na 2.5:

$$V = (\Sigma dx, \Sigma dy, \Sigma|dx|, \Sigma|dy|) \tag{2.5}$$

Assim, para cada ponto chave extraído da imagem, haverá 4x(4x4) descrições para ele (6).

2.1.3.3 Relacionando pontos chaves

Depois que os pontos chaves são detectados e descritos, tanto na imagem base quanto na imagem a ser comparada, é necessário fazer o casamento desses pontos. Para isso, utiliza-se primeiro um algoritmo mais granular e depois outro mais refinado de comparação. Se três pontos chaves são casados corretamente, pode-se concluir que a imagem base está contida na imagem a ser comparada (6).

2.1.3.3.1 Comparação granular

Este algoritmo é baseado no método do vizinho mais perto bidirecionalmente. Suponha que temos dois conjuntos de pontos chaves do SURF, P e Q, sendo P da imagem base e Q da imagem a ser comparada. Se o par de pontos (p_i, q_j) satisfaz um dos critérios, então pode-se dizer que o par (p_i, q_j) está casado.

1. $d(p_i, q_j) = \min_{q_l \in Q} d(p_i, q_l) = \min_{p_k \in P} d(p_k, q_j)$
2. $d(p_i, q_j) \leq \min_{q_l \in Q, l \neq j} d(p_i, q_l) * \alpha$
3. $d(p_i, q_j) \leq \min_{p_k \in P, k \neq i} d(p_k, q_j) * \alpha$

Perceba que é necessário que o limiar $\alpha \leq 1$ e que quanto menor é o valor de α, mais preciso será o algoritmo, porém menos pares de pontos casados serão detectados. Com a comparação granular, pode-se eliminar pontos que claramente não terão um par na imagem base. Depois, usa-se a comparação mais refinada para melhorar a precisão do algoritmo como um todo (6).

2.1.3.3.2 Comparação refinada

O método descrito na subseção anterior não filtra os pontos com uma precisão adequada. Por isso, após a primeira comparação com um algoritmo não tão preciso mas

rápido, usa-se um algoritmo de maior exatidão (6). Um exemplo de algoritmo seria o *RANdom SAmple Consensus* (RANSAC), que consiste em utilizar o menor conjunto possível para caracterizar os dados e ir aumentando esse conjunto graduativamente (10).

2.1.4 Open Source Computer Vision Library (OpenCV)

O OpenCV é uma biblioteca de código aberto para visão computacional, processamento de imagem, aprendizagem de máquina, entre outros. Ele inicialmente foi desenvolvido na linguagem C++ mas atualmente possui interface para Python, Java e MATLAB. Ele suporta multiplos *threads* e processamento utilizando a GPU (*Graphics Processing Unit*). Isso dá à biblioteca a capacidade de resolver problemas em tempo real que necessitam de mais capacidade computacional. O OpenCV também é versátil em termos de plataforma, podendo ser usando no Windows, Linux, Android, MacOS e sistemas embarcados (11).

Esta biblioteca é muito poderosa e possui mais de 2500 algoritmos, abrangendo diversas áreas da visão computacional, como identificação faces e rastreamento de objetos. Neste trabalho, o foco é em algoritmos capazes de detectar a mesma cena em imagens diferentes. Esse tipo de problema se chama *scene matching* (1).

2.1.5 Ambientes Virtuais do Python

Python é uma linguagem muito utilizada em diversas aplicações, e por isso, muitos programas escritos nessa linguagem dependem de bibliotecas e pacotes que não estão disponíveis na biblioteca padrão da linguagem. Isso faria com que fosse necessário fazer uma configuração em cada máquina que compilasse o código do programa para garantir que o software funcionasse. Uma forma de resolver este problema é usar os *virtual environments* do Python, ambientes virtuais em português. Esses ambientes são diretórios isolados que possuem uma versão específica do Python, além de bibliotecas adicionais. Logo, é possível criar um ambiente para cada programa com as bibliotecas específicas necessárias (12). Para este projeto foi utilizado um ambiente virtual, de forma que os desenvolvedores possam usar facilmente as mesmas versões das bibliotecas, como do OpenCV. No apêndice **??** encontra-se o arquivo com as bibliotecas especificadas.

2.1.6 O drone Tello

O drone que será usado nos experimentos é o drone Tello, da empresa DJI. Esse modelo foi escolhido pois, além do laboratório do IME já possuir um exemplar, ele é pequeno, de fácil manuseio, tem um vôo estável e também possui uma qualidade de foto e vídeo suficientemente boa para o trabalho proposto. Uma imagem do drone pode ser

vista na figura 6. A especificação completa do drone pode ser visto no website da DJI (13). Algumas características importantes estão relacionadas na tabela 1.

Figura 6 – Drone DJI Tello.

Tabela 1 – Especificações do drone Tello.

Característica	**Valor**
Dimensões	98 x 92.5 x 41 mm
Tempo máximo de vôo	13 minutos
Distância máxima de vôo	100 metros
Altura máxima de vôo	30 metros
Qualidade do vídeo	720p (HD)
Qualidade da foto	5MP (2592x1936)
Peso	80 gramas
Velocidade máxima	8 m/s

Uma característica física do drone Tello é que sua câmera é apontada para frente, o que não é adequado para o objetivo desse trabalho. Então, faz-se necessário recorrer à uma adaptação por meio de um suporte com um espelho, de tal forma que a câmera obtenha imagens que estão diretamente abaixo do drone. Uma imagem do drone com o suporte já encaixado pode ser vista na figura 7. Apesar dessa limitação, o Tello foi escolhido para este trabalho, pois já se encontrava disponível para uso no laboratório de robótica do IME, já tendo sido utilizado com sucesso por outros alunos, de graduação e pós-graduação, em outros trabalhos.

O suporte para o espelho é uma peça impressa utilizando uma impressora 3D com um filamento de PLA. Além disso, também é necessário um espelho quadrado de 18 mm de lado, para ser colado no suporte. O modelo 3D do suporte pode ser encontrado em (14).

2.1.7 A API de controle do drone Tello

O Tello possui um Standard Development Kit (SDK) que permite ao usuário controlar a aeronave usando comandos de texto. A arquitetura de comunicação padrão utiliza Wi-Fi, que é compartilhada pelo próprio drone. Após o dispositivo, que pode ser um computador ou um smartphone, estar conectado na rede, é necessário criar um socket com o protocolo *User Datagram Protocol* (UDP) para mandar comandos e receber dados

Figura 7 – Drone com o suporte para o espelho encaixado. Fonte: Claye(14).

do drone. Além disso, os dados de vídeo da câmera também são recebidos por um socket UDP.

Caso o usuário se conecte por um smartphone, ele pode usar o aplicativo Tello, que está disponível para Android e iOS. O aplicativo possui uma interface de usuário amigável e pode ser usado para voar o drone e ver o vídeo da câmera de forma simultânea. Caso contrário, se o usuário se conectar ao drone por um computador, ele tem que criar o socket UDP manualmente, utilizando alguma linguagem de programação.

Essa API possui uma limitação de não permitir movimentos, em qualquer direção, menores que 20 centímetros. Essa limitação é imposta diretamente no SDK do Tello, não sendo possível ser ultrapassada (15). Dessa forma, os menores movimentos que o drone poderá fazer são de 20 centímetros em qualquer direção.

O SDK fornece uma lista de comandos que pode ser utilizada pelo usuário. A lista completa pode ser visualizada no website da DJI (13) e alguns comandos podem ser vistos na tabela 2.

Tabela 2 – Alguns dos comandos disponíveis no SDK do Tello.

Comando	**Descrição**
takeoff	Decolagem automática
land	Pouso automático
up x	Deslocar "x"cm para cima
left x	Deslocar "x"cm para baixo

Um esquema no qual a comunicação entre o dispositivo controlador e o drone é simplificada, pode ser visto na figura 8.

2.1.8 A biblioteca DJITelloPy

Inicialmente foi feita uma pesquisa entre as bibliotecas *open-source* disponíveis e chegou-se a conclusão de que seria usada a biblioteca TelloPy (16). Porém, com o desenvolvimento do projeto, foi visto que nas funções disponíveis na biblioteca não existia

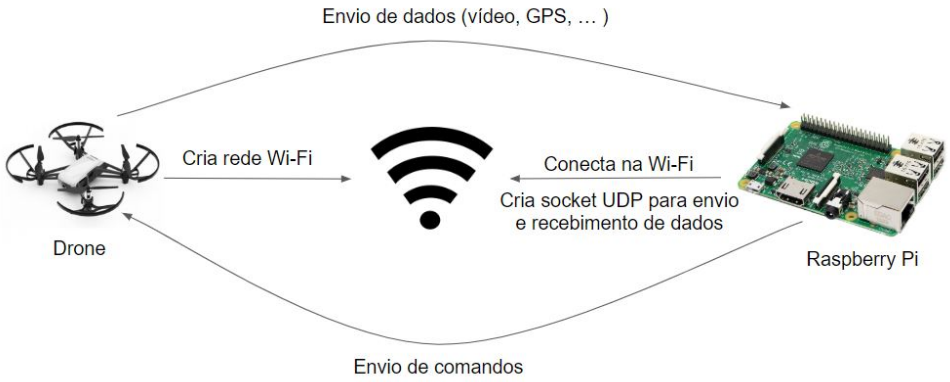


Figura 8 – Esquema simplificado da comunicação entre o drone e o dispositivo controlador.

nenhuma que pudesse movimentar o drone por uma distância fixa no espaço, só sendo possível passar uma velocidade para o drone em qualquer direção e controlar, no código deste trabalho, quanto tempo o drone ficaria se movimentando nessa direção. Por isso, apesar do trabalho ser possível com essas funções, ele geraria mais código desnecessário e provalmente acarretaria em perda de precisão. Então, foi pesquisada uma nova biblioteca que pudesse além de proporcionar uma determinada velocidade ao drone em qualquer direção, também conseguisse movimentar o drone de um ponto a outro do espaço só sendo necessário passar as diferenças nas coordenadas entre esses dois pontos. A biblioteca encontrada que possui todas essas funções é a DJITelloPy, que pode ser encontrada em (17).

Essa biblioteca é basicamente uma abstração de todas as funcionalidades presentes no SDK do Tello. A API fornece funções que facillitam a movimentação do drone além de permitir uma captura de imagem da câmera de forma mais fácil. Além disso, ela também possui algumas funcionalidades extras que ajudarão no projeto, são elas:

- Controlar o drone com o teclado do computador
- Transmissão de vídeo ao vivo, com a opção de adicionar filtros e efeitos ao vídeo
- Exemplos básicos de código: decolar, pousar e transmissão de dados

Essa biblioteca será executada em um computador que permanecerá em solo, sendo esse computador o responsável por enviar os comandos de movimentação para o drone e por receber todos os dados relativos ao vôo, como: altitude, velocidade e imagem da câmera.

O código da bilioteca, assim como instruções de uso e atualizações, pode ser encontrado em (16).

2.2 Revisão de Literatura

Nesta seção, serão mostrados trabalhos de outros autores que foram utilizados como base para esta monografia.

2.2.1 Escolha do algoritmo de *scene matching*

Como dito no capítulo anterior, a grande demanda pela evolução tecnológica da visão computacional acarretou no desenvolvimento de diversos algoritmos para solucionar vários problemas da área. Em específico, há várias técnicas para resolver grupo de problemas designado de *scene matching*. Os algoritmos mais conhecidos são: *template matching*, detecção de *blobs*, SURF, SIFT e *Oriented FAST and rotated BRIEF* (ORB) (4).

A escolha do algoritmo utilizado nesta monografia foi baseado nos trabalhos de (4) e (18). O primeiro faz uma comparação entre o *template matching*, detecção de *blobs* e o SURF enquanto o segundo faz a comparação entre o SIFT, SURF e ORB.

2.2.1.1 Comparação do *template matching*, detecção de *blobs* e SURF

O *template matching* é considerado um método de alto nível de visão computacional. Ele consiste em ter um conjunto de imagens que será o *template*, ou o que se está buscando em nas outras imagens. Para se procurar a imagem, primeiro são passados diversos filtros na imagem para conseguir extrair objetos diferentes. Esses objetos então são comparados com os *templates* através de uma função de correlação. Esta técnica é muito utlizada para encontrar letras ou palavras em imagens de texto. O algorimto de detecção de *blobs*, por outro lado, procura regiões onde há contraste em características, como luz e sombra. Um *blob* é considerado uma região onde as caracterísiticas são relativamente uniforme. O método consiste em encontrar esses *blobs* nas imagens e assim fazer a comparação (4). A técnica SURF consiste em encontrar pontos chaves nas imagens e comparar estes pontos. O SURF é descrito com mais detalhes na seção 2.1.3.

A comparação entre os três foi feita testando vários conjuntos de imagens, que consistiam em diversos tipos de imagem como textos, objetos e gestos. O resultado está resumido na tabela 3.

Chegou-se a conclusão que em termos de precisão, o SURF foi o destaque porém vale ressaltar que ele é o mais complexo de ser implementado dos três (4).

2.2.1.2 Comparação do SIFT, SURF e ORB

O algoritmo SIFT é semelhante ao SURF. Ele consiste em procurar pontos chaves nas imagens e fazer a comparação desses. O SURF possui o mesmo fluxo porém, a forma de cálcular esses pontos é diferente (para mais detalhes do SURF, veja a seção 2.1.3). O

Tabela 3 – Comparação do *template matching*, detecção de *blobs* e SURF. Adaptado de: Jayanthi e Sreedevi(4).

Pontos de comparação	**SURF**	***Template Matching***	**Detecção de blobs**
Precisão	85-95%	40-50%	10-20%
Velocidade	Rápido	Intermediário	Muito devagar
Limite de gestos	Até 30	N/A	Até 5
Invariante a escala	Sim	Não	N/A
Invariante a rotação	Sim	Sim	N/A
Complexidade	Mais complexo	Menos complexo	Menor complexidade
Restrinções	Funciona bem com vários tipos de imagens	Cor do fundo precisa ser diferente da cor do texto	Cor do fundo precisa ser branco ou preto

ORB, por outro lado, é a composição da detecção de pontos chaves usando o *features from accelerated segment test* (FAST) e da sua descrição usando o método *Binary Robust Independent Elementary Features* (BRIEF) (18).

A comparação entre os três foi feita utilizando uma imagem com variação de cor, intensidade, rotação, escala e ruído. As tabelas 4, 5 e 6 resumem o resultado encontrado.

Tabela 4 – Comparação do SIFT, SURF e ORB em relação ao registro de uma imagem com variação em intensidade. Adaptado de: Karami, Prasad e Shehata(18).

	Tempo (segundos)	Kpnts1	Kpnts2	Registros	Taxa de registro (%)
SIFT	0,13	248	229	183	76,7
SURF	0,04	162	166	119	72,6
ORB	0,03	261	267	168	63,6

Tabela 5 – Comparação do SIFT, SURF e ORB em relação ao registro de uma imagem com ela alterada em rotação. Adaptado de: Karami, Prasad e Shehata(18).

Ângulo	0	45	90	135	180	225	270
SIFT	100	65	93	67	92	65	93
SURF	99	51	99	52	96	51	95
ORB	100	46	97	46	100	46	97

Os resultados mostram que o ORB é o mais rápido entre os três enquanto o SIFT é o mais lento. O SIFT chega a ser duas vezes mais devagar que o SURF. O SURF possui

Tabela 6 – Comparação do SIFT, SURF e ORB em relação ao registro de uma imagem com adição um ruído. Adaptado de: Karami, Prasad e Shehata(18).

	Tempo (segundos)	Kpnts1	Kpnts2	Registros	Taxa de registro (%)
SIFT	0,115	248	242	132	52,8
SURF	0,059	162	385	108	39,48
ORB	0,027	261	308	155	54,48

pior precisão quando há ruído, mas nos outros casos ficou na média. O SIFT apresentou um padrão de precisão bem estável nos experimentos (18).

2.2.1.3 Escolha do algoritmo de *Scene Matching*

No início do projeto, optou-se por usar o SURF como algoritmo de *scene matching*, devido ao fato que os artigos analisados mostraram que ao mesmo tempo que a técnica demonstrava uma boa precisão, ela também conseguia ser rápida.

Porém, durante os testes iniciais da visão nos mapas utilizados, notou-se que o SIFT se adequou melhor, como explicado na seção 5.1.1.

3 DESCRIÇÃO DO PROBLEMA

Como o objetivo deste trabalho é elaborar, implementar e testar um sistema de localização baseado em imagens, foi preciso utilizar algum ambiente que simule as interações do drone com um problema já existente no mundo, de tal forma que o drone possa, no contexto desse problema, reconhecer determinadas áreas por meio de sua câmera e se movimentar até os locais escolhidos.

É importante destacar que este trabalho não foi feito para resolver exclusivamente o problema que será explicado nesse capítulo, podendo ser usado para diversos problemas que necessitem de um sistema de localização baseado em imagens. Para isso, é importante que o sistema seja alimentado com todas as informações pertinentes a cada situação específica. Quais são e como deverão ser passadas essas informações ficará mais claro nos capítulos seguintes.

O problema escolhido é baseado em um desafio proposto pela maior empresa de petróleo do Brasil, a Petrobras. Além disso, existe uma competição em que equipes de toda a América Latina podem participar para tentar ganhar o máximo de pontos possível neste desafio. Portanto, esse problema é quase inteiramente uma tradução do desafio citado e o original pode ser visto no website da competição latino-americana de robótica, encontrado em (19).

O principal objetivo desse problema é fornecer casos de teste para verificar a eficiência do software desenvolvido neste trabalho, o que significa usar o sistema de navegação proposto para resolver uma tarefa do desafio, presente no mundo real.

Para isso, o problema estuda a operação e o voo do drone sobre dutos petrolíferos. O campo de testes simula essas instalações, contendo pontos especificados onde o drone deve decolar e pousar. Durante todos os testes, o drone deve agir de forma autônoma, sem nenhum controle externo ou interação humana.

O desafio original possui quatro fases, porém este trabalho irá focar na primeira fase, com algumas adaptações, onde pode-se melhor avaliar a localização e mapeamento, que é o foco de estudo desse trabalho.

3.1 O campo de testes

O campo de testes, ou arena, que será usada possui uma área de 64 m^2 com locais previamente marcados que serão usados como bases. Existem três tipos diferentes de bases:

- Base costeira: o lugar de onde o robô deve sair e voltar de suas tarefas

- Bases suspensas e terrestres

Além das bases suspensas, terrestres e costeiras, a arena também possui um grande oleoduto, na cor laranja, transversal à arena. Essas instalações podem ser vistas no esquema da figura 9.

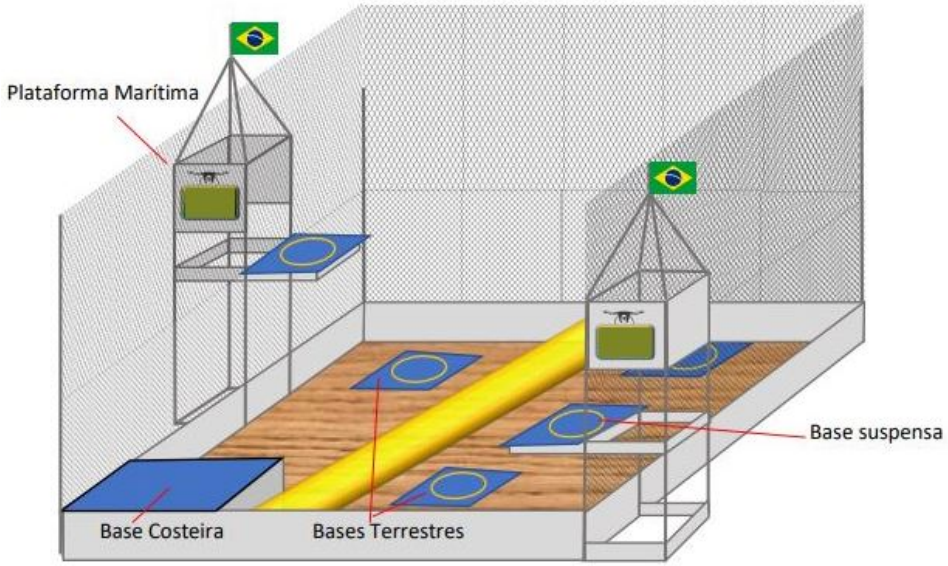


Figura 9 – Representações das instalações no campo de testes. Fonte: LARC(19).

Mais informações sobre o desafio, como a especificação dos tamanhos das bases e dos símbolos que definem como o robô identifica cada base podem ser encontrados no anexo A.

A tarefa

A tarefa do problema é: dado um mapa com a coordenada de todas as bases da arena, o drone deve decolar da base costeira e pousar uma única vez em cada base terrestre e suspensa, utilizando-se da estimação de movimento espacial, dada pela *Inertial Measurement Unit* (IMU), e da estimação de posição por imagens, dada pelo módulo de visão computacional do sistema. A descrição técnica de como essa navegação funciona se encontra no capítulo 4.

Então, pode-se fazer um casamento entre o objetivo deste trabalho e a tarefa proposta. De acordo com a seção 1.2 e o que foi descrito nessa seção, a tarefa servirá para avaliar a implementação do sistema de localização baseado em imagens. Assim, o sistema poderá ser testado em um ambiente controlado e os resultados poderão ser analisados.

4 SOLUÇÃO PROPOSTA

Este trabalho tem como objetivo desenvolver um sistema capaz de fazer uma navegação baseada em imagens. Para isso, a solução proposta foi divida em duas partes: navegação e a localização. A junção desses segmentos irá resultar no navegador por imagens.

Para este projeto, o código será executado em um computador que ficará no solo próximo ao local onde o drone está voando, pois o Tello não possui um computador de bordo embarcado. Entretanto, em outros projetos pode ser usado um computador de bordo, como um Raspberry Pi, que poderá embarcar o algoritmo junto com o drone. Caso o código seja executado de forma embarcada, deve se atentar para a velocidade de processamento do código. Apesar desse trabalho não estabelecer um critério para a taxa de atualização mínima do algoritmo, o código poderia apresentar funcionamento inesperado caso seja executado em taxas mais baixas do que o normal. A frequência de geração da localização e sua precisão, obtidas neste trabalho, são apresentadas e discutidas no capítulo de resultados.

4.1 Navegação do drone

O objetivo principal da navegação é fazer com que o drone pouse em cada base uma única vez. Para isso, o drone deverá decolar da base inicial, se deslocar de forma autônoma até a base mais próxima, que não foi visitada ainda, e pousar nela.

Para essas tarefas, algumas variáveis serão definidas e determinadas previamente. Primeiramente, será admitido que existe um eixo coordenado xyz, sendo que o centro da base costeira está posicionada exatamente na origem do sistema de coordenadas xy, assim como mostra a figura 10.

Dessa maneira, a origem da arena será sempre o ponto de partida do drone. Além disso, as coordenadas das bases são dadas pela posição do centro de cada base nesse eixo coordenado.

No começo de cada experimento de navegação, será dado como entrada ao sistema um mapa com as coordenadas de todas as bases. O drone então, baseando-se na sua estimativa de posição dada pelo seu sistema de navegação inercial (IMU) e do navegador de imagens, deve conseguir pousar uma vez em cada base e voltar à base inicial. A lógica de movimentação será explicada com mais detalhes na seção 4.1.2.

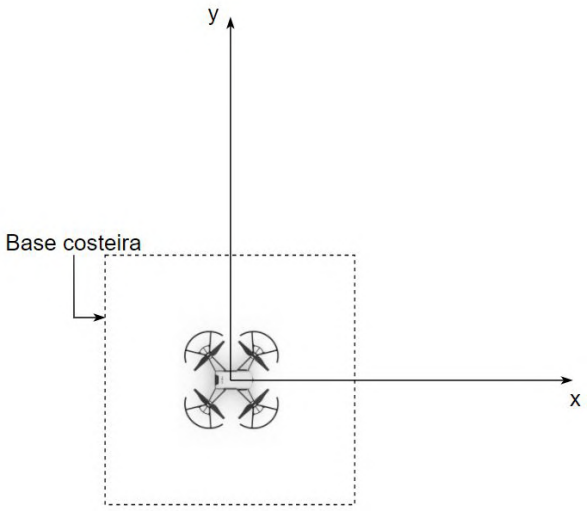


Figura 10 – Representação da base costeira e dos eixos coordenados na arena.

4.1.1 Estimação da posição

Durante o voo, o drone precisará estimar sua posição tentando obter a maior precisão possível. Como esta monografia trabalha com o fato dos sistemas de geolocalização por satélites não estarem viáveis a todo tempo, pode-se usar dois tipos de estimação da posição: a estimativa de posição dada pela IMU, que é embarcada no drone, e a estimativa da posição baseada nas imagens aéreas.

4.1.1.1 Estimação baseada na IMU

A IMU (*Inertial Measurement Unit*) é um dispositivo eletrônico que consegue medir aceleração e rotação utilizando acelerômetros e giroscópios. De forma geral, a IMU envia, com uma determinada taxa de frequência, a velocidade que o drone possui em todos os eixos (incluindo a velocidade angular).

Especificamente para o caso do Tello, o drone possui um *hardware* embarcado que faz automaticamente o cálculo do quanto o drone se moveu em um determinado intervalo de tempo, possibilitando assim o envio de comandos por distância para o drone, como por exemplo: movimentar-se 30 centímetros para qualquer direção desejada.

Esse sistema no entanto possui limitações quanto a sua precisão, é de se esperar que o drone não consiga se movimentar de forma satisfatória utilizando somente seu sistema de navegação inercial. Tal tópico será discutido com mais detalhes na seção de resultados, onde são apresentados experimentos feitos utilizando somente a IMU como estimador da posição do drone no espaço.

4.1.1.2 Estimação baseada em imagens aéreas

Durante seu trajeto entre as bases, o drone deve utilizar a câmera para ajudá-lo a corrigir sua posição no ambiente, essa tarefa será feita pelo módulo de visão computacional do sistema.

Esse módulo será descrito com maior detalhes no próximo capítulo, mas basicamente a visão computacional deve analisar as imagens da câmera e informar ao sistema uma estimativa de onde o drone está de acordo com o mapa de referência obtido antes do voo.

4.1.2 Lógica de movimentação

Após decolar do ponto de partida, o drone deve seguir um comportamento para que possa, iterativamente, pousar e decolar de cada base existente no mapa, até voltar para o ponto de partida e terminar a missão. Esse comportamento pode ser melhor entendido no diagrama de lógica de movimentação da figura 11.

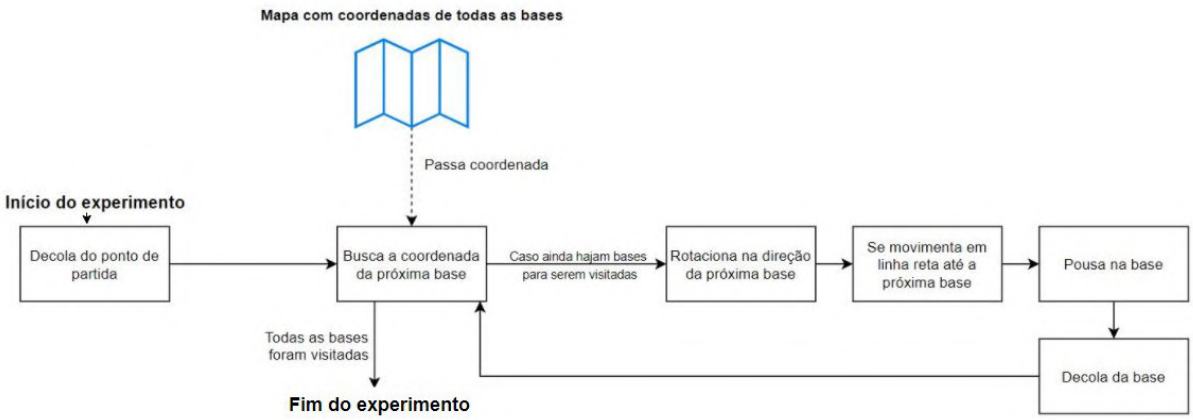


Figura 11 – Diagrama de lógica de movimentação do drone ao longo do experimento.

Então, segundo o que pode ser visto no diagrama, o drone realizará um comportamento repetitivo para visitar todas as bases. Dado que o drone está em uma base qualquer no meio do experimento, ele irá decolar, procurar a coordenada da próxima base, rotacionar o seu eixo para que sua frente esteja virada para a próxima coordenada e então se movimentar de tal forma que fique em cima da base de destino, para que possa pousar e repetir o comportamento inteiro novamente.

Vale lembrar que apesar do drone saber exatamente a distância entre as bases, a movimentação baseada somente nas estimativas dadas pela IMU possui erros de medição que são acumulados caso não exista alguma correção. No caso desse trabalho, essa correção será feita pelo estimador baseado nas imagens aéreas que será descrito na seção 4.2.

4.2 Visão computacional

O objetivo principal do sistema de visão computacional a ser desenvolvido neste trabalho consiste em realizar as seguintes tarefas:

1. Identificar locais através das imagens do drone
2. Localizar o drone a partir dessas imagens

O diagrama da figura 12 exibe os componentes do sistema. Os módulos serão explicados em mais detalhes nas próximas seções.

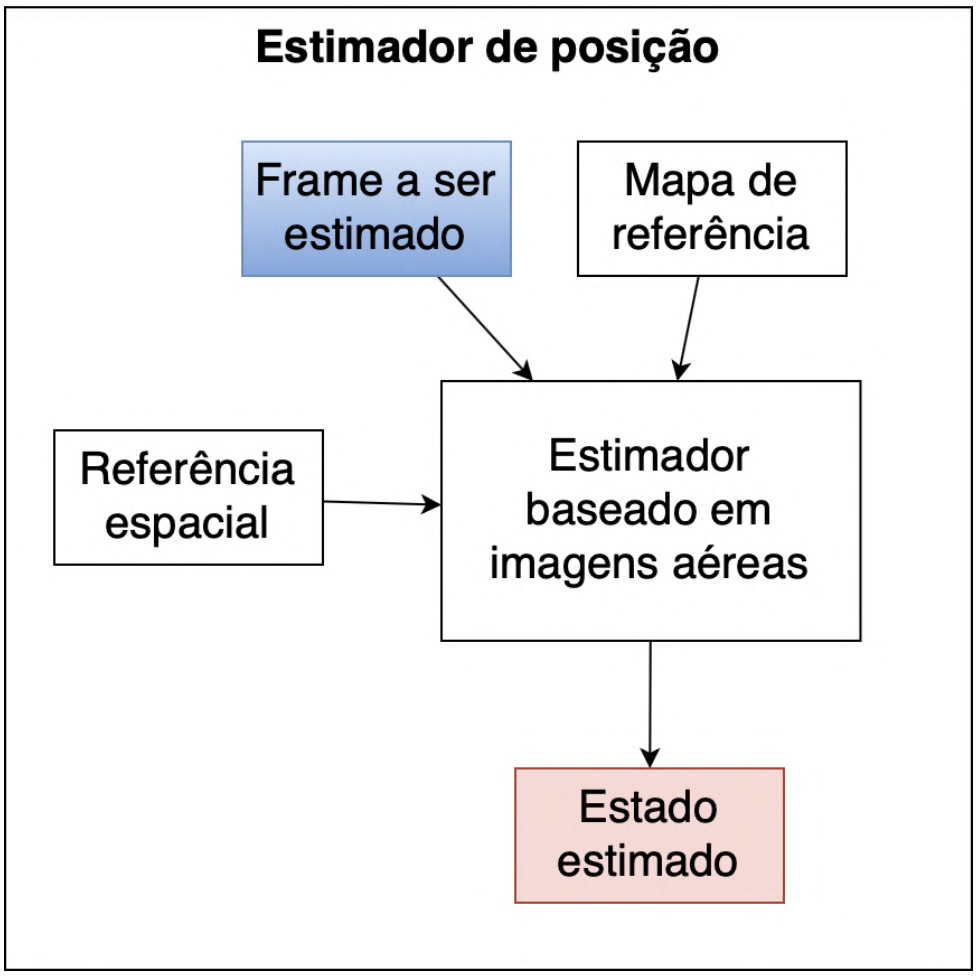


Figura 12 – Diagrama de componentes do sistema de visão computacional.

4.2.1 Capturador de frames

O módulo capturador é responsável por adquirir, manter e descartar os frames a serem estimados, como visto na figura 12, independente de onde eles tenham originado. O sistema desenvolvido possui três opções de captura de imagem: vídeo, câmera do computador e câmera do drone. A opção de vídeo traz frames de um vídeo pré-gravado, no

formato mp4, enquanto a opção de câmera do computador procura frames em tempo real da câmera do computador que está executando o sistema, caso este possua uma. A última opção, a câmera do drone, busca frames da câmera do drone, caso este esteja conectado ao computador. A figura 13 mostra o diagrama do capturador, com cada componente descrito anteriormente. No apêndice **??** encontra-se o código do capturador.

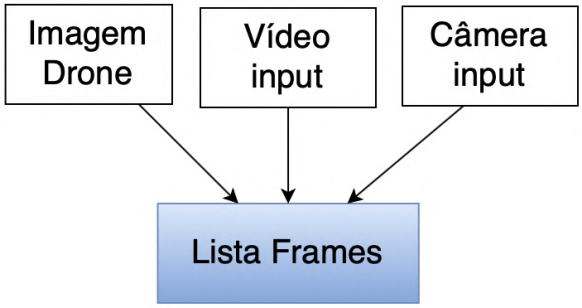


Figura 13 – Diagrama do capturador.

4.2.2 Estimador da posição

O estimador é o componente mais importante do sistema de visão computacional. Este módulo é responsável por fazer a identificação dos locais desejado e calcular a posição do drone baseado nesses locais, além de determinar a posição para a qual o drone deve se deslocar. As funções do estimador estão ilustradas na figura 14.

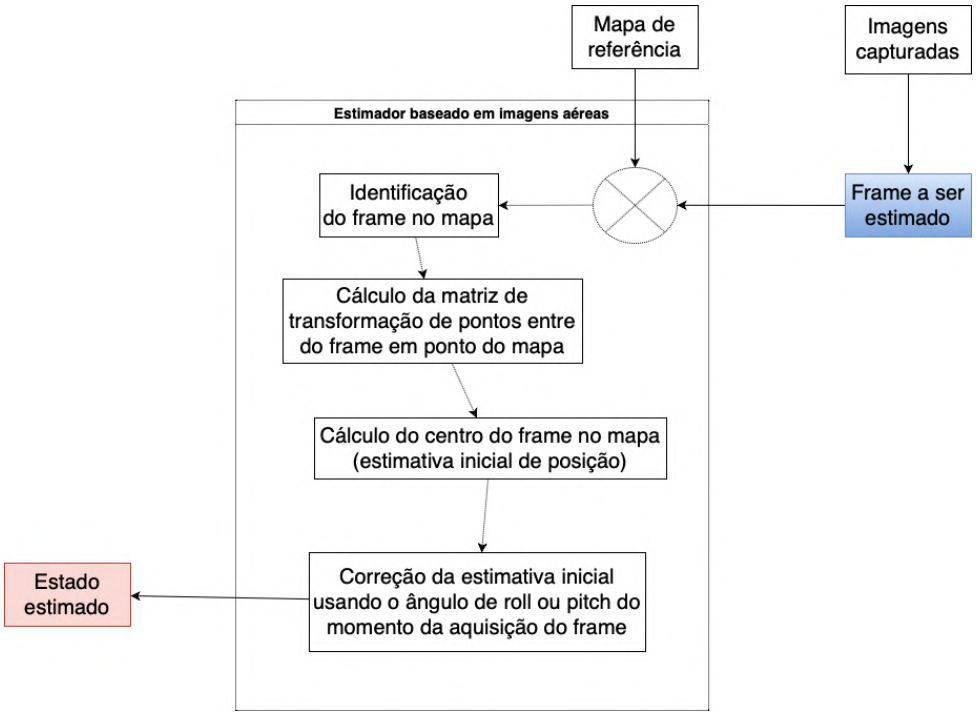


Figura 14 – Diagrama do estimador

A primeira etapa do estimador é fazer a identificação dos locais do mapa de referência no frame a ser analisado. Esta etapa envolve usar o algoritmo de *scene matching*, conseguindo assim fazer a comparação e localização dos templates de referência na imagem passada pelo drone.

Conseguindo identificar esses pontos, é necessário fazer a transformação entre o mapa de referência e o frame atual, conseguindo assim fazer uma correspondência entre cada ponto do frame para sua posição no mapa. Com isso, e os dados de altitude do drone, é possível identificar a posição do veículo baseado no ponto central do frame.

Junto, todos estes dados compõem o estado estimado do drone, a partir da qual serão definidas as futuras posições do robô.

4.3 Navegação por imagens

O navegador por imagens é a junção do sistema de navegação com o sistema de visão que foi divido em duas *threads*, assim como mostra a figura 15. Em uma *thread* se encontra o sistema de visão computacional, que constantemente atualiza a posição do drone para a navegação. Em outra *thread* se encontra o sistema de movimentação, que sofreu algumas adaptações.

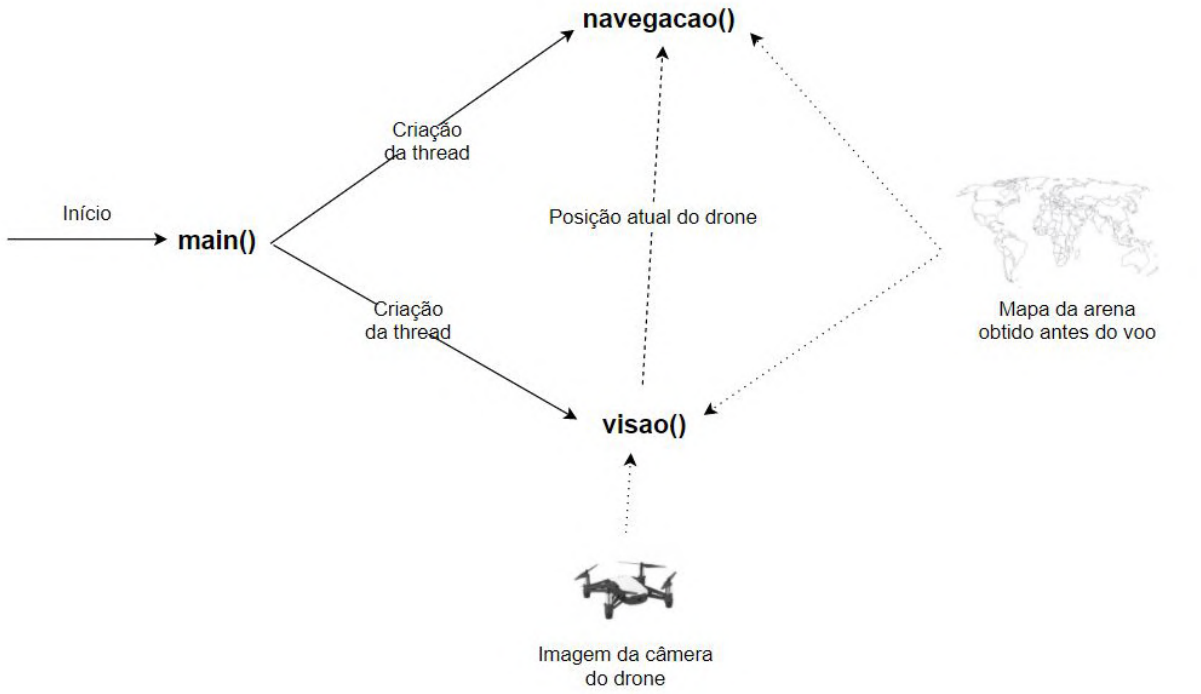


Figura 15 – Representação das *threads* do navegador de imagens

Após se movimentar para um *waypoint*, o sistema analisa sua posição atual, a que é calculada pelo algoritmo de visão computacional, e determinava seu o deslocamento

necessário para corrigir sua posição atual baseando-se onde o drone deveria estar caso tivesse se movimentado corretamente.

Para facilitar o funcionamento dos algoritmos de *scene matching*, o drone pausa sua navegação algumas vezes por cerca de 1 segundo, sendo que esse tempo depende da taxa de atualização que a thread do estimador de posição está sendo executada. Essa pausa ocorre durante a movimentação entre uma base e outra, e serve para que a visão consiga calcular com mais precisão a sua posição no mapa. Essas paradas são necessárias pois a taxa de atualização do algoritmo de visão é muito menor que a taxa de atualização do IMU, significando que o drone pode se movimentar durante um percurso relativamente longo sem receber *feedback* do sistema de visão. Além disso, o sistema de visão computacional também pode não conseguir identificar imediatamente onde o drone está localizado no mapa, isso pode acontecer pois nem sempre o algoritmo de *scene matching* consegue combinar *key points* suficientes para se localizar. Essa dificuldade pode ser atribuída a condições de aquisição da imagem, como por exemplo: a câmera e o suporte para o espelho estarem balançando enquanto o drone está em voo ou a luminosidade do ambiente estar diferente entre o momento que o mapa foi obtido e o momento do experimento de voo.

A correção de movimentação juntamente com os pontos de parada podem ser melhor vistos na figura 16, onde após passar pelo 1º ponto de parada o drone erra o caminho para o 2º ponto de parada, terminando numa posição que deve ser corrigida pela visão.

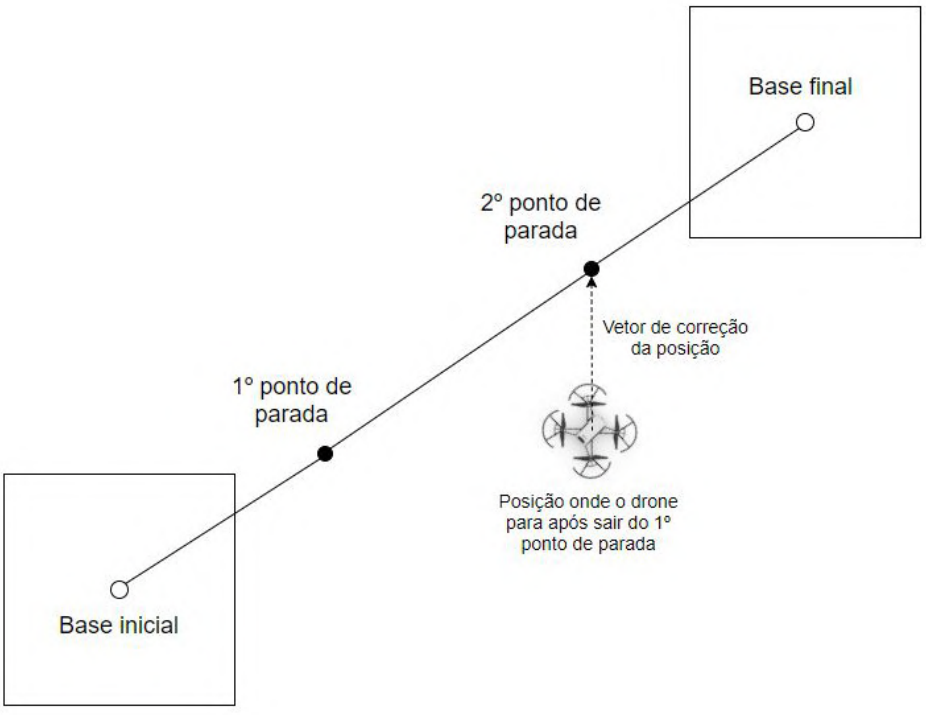


Figura 16 – Representação da correção de posição que o drone deve fazer após realizar um movimento não preciso durante os pontos de parada.

5 EXPERIMENTOS E RESULTADOS

Durante o desenvolvimento do projeto, vários testes e experimentos foram realizados, tanto para determinar parâmetros a serem usados quanto para medir o desempenho do sistema desenvolvido. Este capítulo descreve esses testes e explica os seus resultados e as decisões tomadas.

5.1 Testes do sistema de visão computacional

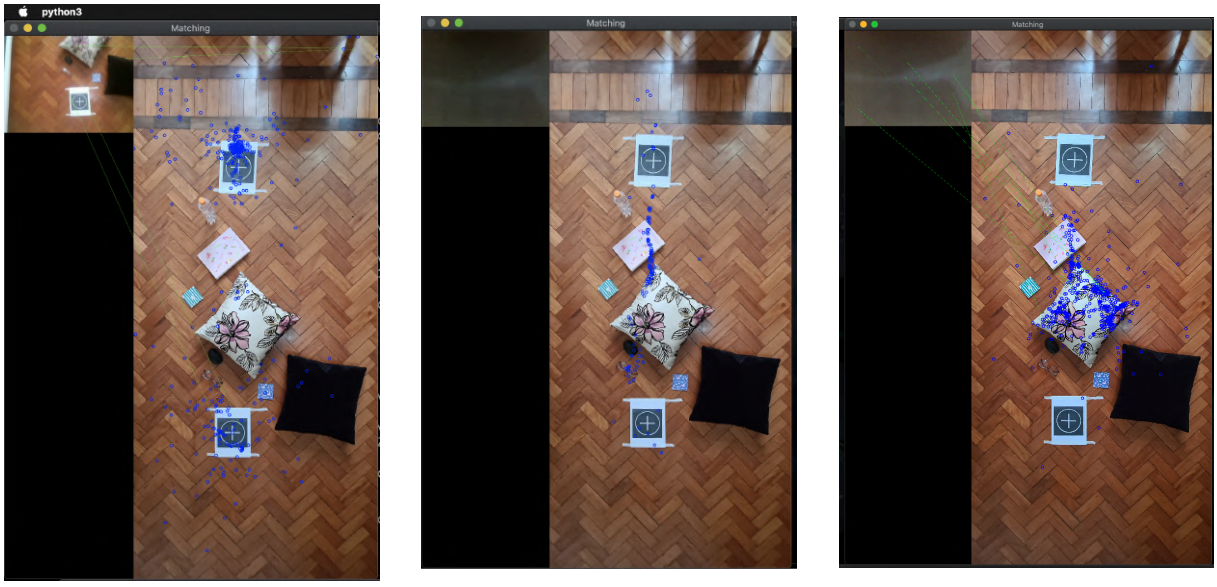
Antes de desenvolver o navegador por imagens, foi necessário garantir o funcionamento adequado do sistema de visão, ou seja, o sistema precisava se localizar de forma precisa e rápida. Para fazer isso, uma série de testes foram realizados para determinar métricas e parâmetros a serem usados. Estes experimentos serão descritos nas próximas subseções.

5.1.1 Algoritmo escolhido

Durante a pesquisa teórica, decidiu-se usar o SURF como algoritmo de *scene matching*. Porém, nos primeiros testes realizados, o SURF apresentou uma baixa precisão, não se localizando adequadamente. Como o software foi implementando para que fosse extremamente fácil trocar entre os algoritmos de *scene matching* baseados em pontos chaves (SURF, SIFT e ORB), foi possível realizar a comparação entre a precisão desses três algoritmos para o caso específico dos mapas.

Para fazer o teste de comparação, o drone realizou um voo de uma base para outra, sendo controlado por controle remoto, e gravando um vídeo do trajeto. Esse vídeo foi usado para simular o voo na visão, realizando o *scene matching* com cada algoritmo e observando a precisão deles em relação a trajetória original o seu cálculo foi. A figura 17 mostra as posições, representadas por círculos azuis, calculadas por cada algoritmo, que deveria gerar o trajeto percorrido. As almofadas no mapa foram utilizadas para deixar o mapa menos uniforme.

A figura 17a mostra a trajetória que o SURF calculou para o teste. Foi possível notar que o algoritmo conseguia identificar a posição quando o drone estava nas bases, porém entre as bases houve poucos pontos de detecção, como mostram as concentrações de pontos azuis nas bases. O ORB teve um resultado contrário ao do SURF, como visto na figura 17c, pois conseguiu achar a sua posição entre as bases mas não conseguiu se localizar nas bases em si, como mostram os pontos disperos nas almofadas. Já o SIFT teve o melhor resultado, como mostra a figura 17b. Além de conseguir fazer a identificação da



(a) SURF (b) SIFT (c) ORB

Figura 17 – Trajetos calculados por algoritmos de *scene matching*.

posição tanto nas bases quanto entre elas, sua precisão foi muito boa, como pode ser visto pelo fato que os pontos desenhados formam uma caminho, demarcado pela sequência de pontos azuis, gerando o trajeto que o drone realizou.

Estes testes foram realizados usando os parâmetros padrões do SURF, SIFT e ORB dentro do OpenCV. A fim de analisar a razão desta discrepância entre os algoritmos, decidiu-se testar o SURF novamente, porém com uma mudança nos seus parâmetros. Portanto, o parâmetro *hessianThreshold*, que normalmente possui um valor de 100, foi alterado para 50. A figura 18 mostra o trajeto calculado pelo algoritmo, do mesmo teste. Comparando esta figura com a 17a, é possível notar que essa única mudança acarretou em um algoritmo com mais precisão, devido ao fato de haver uma concentração maior de pontos entre as duas bases, especialmente no segmento que as liga. mostrando que a visão computacional é tão sensível que precisa ser adaptativa, variando os parâmetros de execução do algoritmo.

Este teste evidenciou que para poder usar o SURF, como decidido previamente, seria necessário calibrar os parâmetros para o caso de uso específico dos experimentos a serem realizados. A escolha inicial do SURF tinha sido devido ao fato que ele possui uma taxa de precisão alta ao mesmo tempo que processava as imagens de forma rápida. Nos testes realizados especificamente nos ambientes de testes, o SIFT apresentou uma velocidade adequada (mais detalhes na seção 5.1.3). Como não havia a necessidade de fazer a calibração do SIFT e sua velocidade estava dentro do esperado, decidiu-se usar o SIFT para a realização dos experimentos.

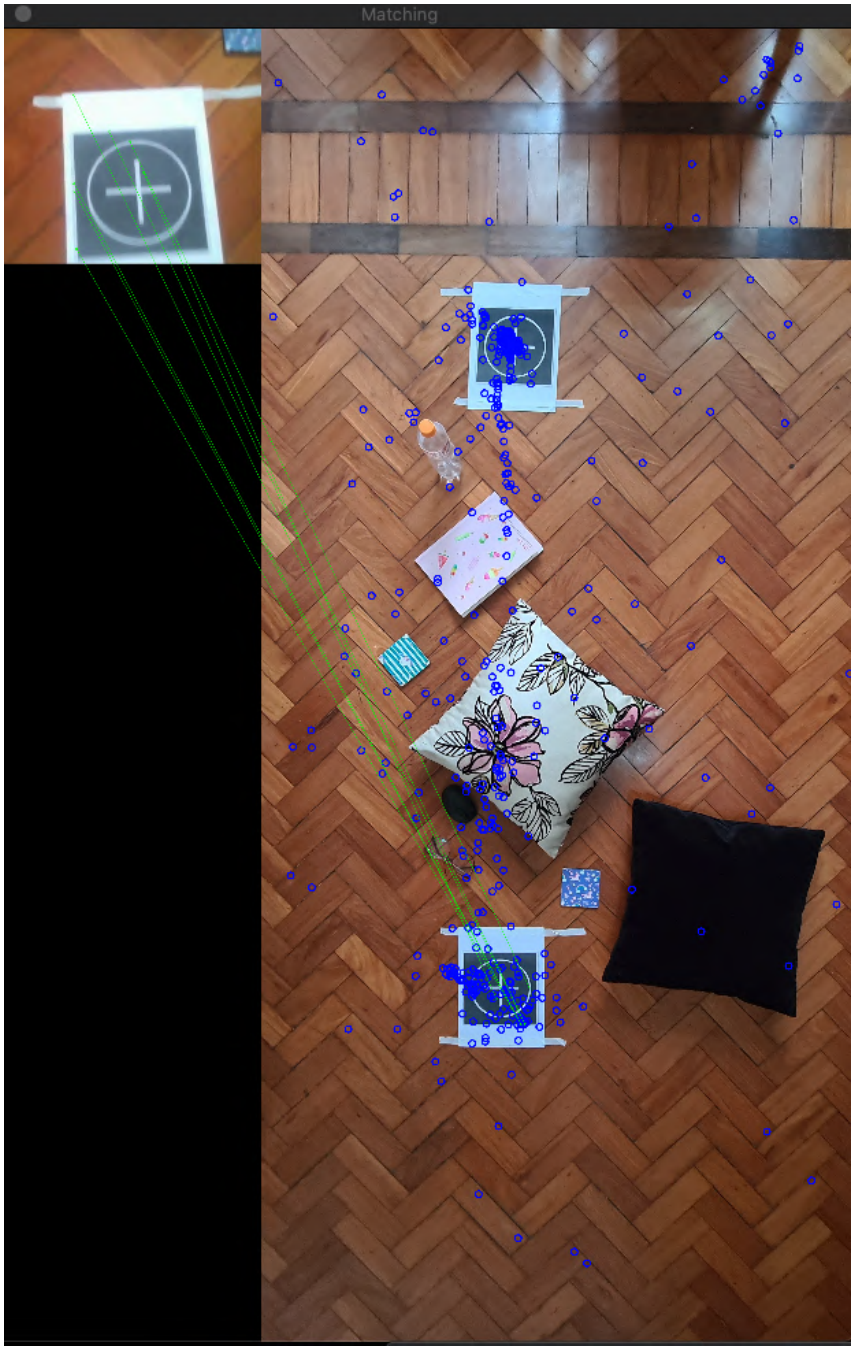
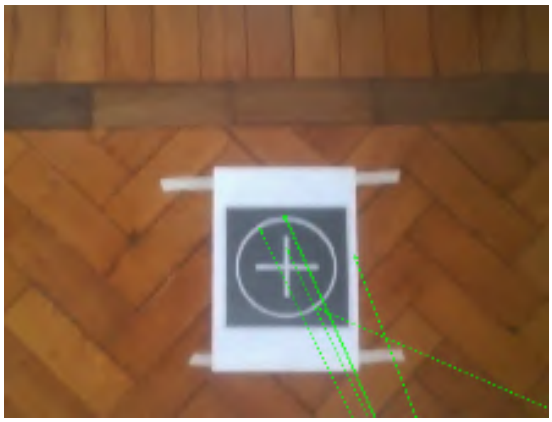


Figura 18 – Trajetória calculada pelo SURF após mudança de parâmetro.

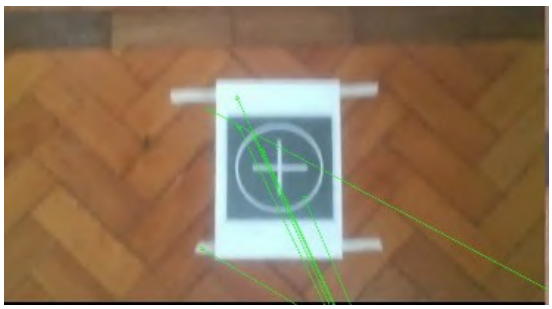
5.1.2 Centro do drone e centro do frame

Para calcular a posição do drone no mapa, desejava-se usar o centro do frame como posição central do drone e assim achar sua matriz de transformação em relação ao mapa real para calcular a posição, como descrito na seção 4.2.2. Porém, devido a posição da câmera no drone e também da posição do espelho, o centro do frame capturado pelo robô não representava sua posição. Para resolver isso, foi necessário medir a distância do espelho do drone em relação ao centro dele e fazer uma analogia ao que seria o deslocamento do centro do frame em relação a posição do drone, modificando o frame para que esses

centros coincidam. A figura 19 mostra essa diferença. Ambos os frames mostrados foram capturados quando o drone estava posicionado acima da base, ou seja, idealmente, o centro da base deveria estar no centro do frame. Na figura 19a, que mostra o frame capturado nessa posição sem alteração, é fácil ver que o centro da base não é o centro do frame. Fazendo as modificações no frame como descrito anteriormente, obteve-se a figura 19b, onde esses dois pontos coincidem.



(a) Frame sem alteração



(b) Frame com alteração

Figura 19 – Frame do drone ao estar em cima da base.

Portanto, na hora de fazer o cálculo da posição no mapa real, usou-se o centro do frame alterado, e não o frame normal. O frame normal foi usado para fazer o *scene matching* pois ao remover uma parte do frame, se perderiam alguns pontos chaves, impactando a precisão do algoritmo, já que haveria menos pontos para fazer a comparação.

5.1.3 Qualidade do mapa

Os algoritmos de *scene matching* utilizados precisavam de um *template* para fazer sua localização. Nos testes e experimentos, o *template* utilizado foi o frame, pois se procurava onde no mapa estava se encontrava o que o drone estava visualizado. A imagem gerada pelo drone possui uma dimensão de 960 por 720 píxels, resultando em uma qualidade de imagem normal. Porém, o mapa utilizado foi obtido tirando foto do local usando um celular, gerando uma imagem de aproximadamente 3000 por 4000 píxels. Por causa disso, o mapa possuía uma grande quantidade de pontos chaves, e fazer a comparação com o os pontos chaves do frame consumia muito tempo, resultando em um código que rodava em aproximadamente 1,5 *frames per second* (FPS).

Com o objetivo de esse tempo de execução, decidiu-se reduzir a quantidade de píxels do mapa e analisar o quanto isso impactava na velocidade do algoritmo e no seu casamento. Portanto, realizou-se o seguinte experimento: gerou-se novos mapas, com qualidade de 100%, 75%, 50%, 25% e 12% em relação a imagem original, e para cada imagem gerada, realizou-se o cálculo das posições de um voo perfeito pelo drone, como no

teste da seção 5.1.1. As métricas utilizadas para analisar os mapas foram a taxa de FPS do *scene matching* e a razão entre os frames que não tiveram posições encontradas e a quantidade de frames totais. Os resultados estão exibidos na tabela 7.

Tabela 7 – Análise do impacto da qualidade do mapa.

Qualidade do mapa	**FPS médio**	**Frames em que a posição não foi encontrada**	**Número total de frames**	**Razão de falhas**
100%	1,5	382	608	0,62
75%	1,5	383	608	0,62
50%	3	225	552	0,40
25%	6	270	600	0,45
12%	8	342	682	0,50

O teste mostrou que quanto maior a quantidade de píxels da imagem, mais tempo foi necessário para fazer o algoritmo de *scene matching*, pois era possível extrair mais pontos chaves do mapa, consequentemente tendo que realizar mais comparações com os pontos do frame. Por outro lado, foi mais fácil encontrar a posição no mapa quando a qualidade do mapa era de 50% da original. Isso se deve ao fato que a qualidade do mapa em 50% é o que mais se aproxima a qualidade do frame capturado pelo drone. Como o mapa de qualidade 25% possuiu um nível de acerto parecido com o de 50% e ainda foi duas vezes mais rápido, optou-se por usar sempre o mapa em 25%.

Vale ressaltar que estes testes foram realizados em um MacBook Pro de 8GB de RAM e com processador 3,1 GHz Dual-Core Intel Core i5 e que o algoritmo de *scene matching* utilizado foi o SIFT. A realização destes mesmos testes em outras máquinas pode gerar resultados divergentes, devido às especificações do computador.

5.2 Experimentos de navegação

Para analisar o sistema de navegação desenvolvido, três experimentos foram realizados. Cada experimento possuía uma configuração de mapa diferente. Devido a falta de espaço, as bases tiveram seu tamanho reduzido, em relação as bases oficiais do desafio. Para melhor simular o ambiente oficial, cada mapa possui uma linha de cadernos, que representa o duto do desafio.

Além disso, para cada experimento foram realizadas seis simulações, três usando a navegação baseada em imagens e as outras três usando navegação baseada no IMU do drone. Isso providenciou uma forma de verificar se o sistema de navegação desenvolvida é mais preciso que o sistema padrão do drone Tello, comparando a posição final do drone com

a posição esperada. Nas próximas seções, serão descritos os experimentos e os resultados obtidos.

5.2.1 Experimento 1

Para o experimento 1 usou-se duas bases. O ponto de saída foi a base posicionada em (0, 0) e desejava-se chegar na base localizada em (150, 0). O duto deste mapa foi posicionado na diagonal, cruzando-lo. A figura 20 mostra o mapa utilizado para este experimento, além de pontos de referência usados para fazer a estimação da posição.

Figura 20 – Mapa usado no experimento 1.

A trajetória a ser seguida pelo drone, de acordo com o algoritmo de navegação descrito na seção 4.1, era a seguinte: decolar da base inicial, voar para a base final, e pousar na base final.

Primeiro realizou-se o experimento usando a navegação baseada em imagens. Na primeira simulação, o drone pousou na posição (159, -1), que ficava a uma distância de 9,0 cm da posição desejada. A segunda simulação teve um desempenho um pouco pior, finalizando o voo na posição (167, 13), ou seja, a uma distância de 21,4 cm do local desejado. Por último, a simulação 3 teve o melhor resultando, ficando a uma distância de 6,3cm da base final, terminando na posição (144, 2) do mapa. Por outro lado, os resultados das simulações da navegação usando IMU não tiveram tanta precisão. Na primeira simulação, o drone pousou no ponto (110, -22), ficando a uma distância de 45,7 cm da posição desejada. O desempenho da segunda simulação foi um pouco melhor, pousando em (129, -33), que ficava a uma distância de 39 cm da base final. Já a última simulação obteve o melhor resultado dos três, ficando a uma distância de 16,3 cm, terminando na posição (153, -16). Os resultados de todas as simulação estão sumarizados na tabela 8.

Tabela 8 – Resultados experimento 1.

Tipo de navegação	**Número da simulação**	**Posição final**	**Distância até a base final (cm)**	**Distância média até a base final (cm)**
Baseada em imagens	Simulação 1	(159, -1)	9,05	12,25
	Simulação 2	(167, 13)	21,4	
	Simulação 3	(144, 2)	6,3	
Baseada no IMU	Simulação 1	(110, -22)	45,7	33,7
	Simulação 2	(129, -33)	39,1	
	Simulação 3	(153, -16)	16,3	

Este experimento foi realizado usando apenas iluminação artificial, que resultou em uma maior instabilidade na movimentação do drone, como descrito na seção 5.4.1. Este fator introduziu um erro adicional na movimentação, que foi bastante evidenciado na decolagem, pois o drone subia mas acabava se deslocando um pouco para trás.

Nas simulações com navegação usando IMU, esse erro foi propagado. Na primeira e segunda simulações ocorreu esse deslocamento, ou *drift*, resultando no fato que o robô não conseguiu chegar até a base. Porém, na simulação 3, o drone teve uma decolagem mais precisa, ficando quase perfeitamente alinhado com a primeira base, e consequentemente chegando muito próximo do ponto objetivo.

O *drift* na decolagem também aconteceu nas simulações da navegação baseada em imagens. Porém, como o drone tinha um referencial novo e que não propagava erros, a localização gerada pelas imagens, ele conseguiu corrigir sua posição durante o seu voo. A figura 21 mostra a trajetória realizada pelo drone na simulação 1, sendo possível assim ver que no início a aeronave estava um pouco deslocada porém esse erro foi corrigido durante o voo.

5.2.2 Experimento 2

O mapa do experimento 2 possuía duas bases. A primeira delas, o ponto de partida do experimento, estava localizada em (0, 0) e a segunda, o destino final, em (150, 80). O duto deste mapa a atravessava horizontalmente. A figura 22 mostra o mapa utilizado, além de pontos de referência usados para fazer a estimação da posição.

A trajetória a ser seguida pelo drone, de acordo com o algoritmo de navegação descrito na seção 4.1, era a seguinte: decolar da base inicial, rotacionar para se alinhar com a base final, voar para a base final e pousar na base final.

Primeiro realizou-se as simulações utilizando a navegação baseada em imagens. Na simulação 1, o drone pousou na posição (157, 82), ficando a uma distância de 7,3 cm do ponto final. Já na segunda simulação houve um pouco mais de imprecisão, o drone terminou a uma distância de 15,8 cm do destino, em (155, 65). A última simulação teve

Figura 21 – Trajetória realizada pelo drone na simulação 3 usando navegação por imagens.

Figura 22 – Mapa usado no experimento 2

um desempenho quase perfeito, finalizando o voo na posição (145, 80), a uma distância de 5 cm do objetivo. Depois realizou-se as simulações utilizando a navegação baseada no IMU. Na primeira, o drone pousou no ponto (120, 68), ficando a uma distância de 32,3 cm da posição desejada. A segunda simulação ficou a uma distância de 30,4 cm do objetivo, ao aterrissar em (145, 50). A última simulação teve a melhor precisão, terminando o seu voo na posição (142, 64) e ficando a uma distância de 17,9 cm. A tabela 9 sumariza os

resultados de todas as simulações deste experimento.

Tabela 9 – Resultados experimento 2.

Tipo de navegação	**Número da simulação**	**Posição final**	**Distância até a base final (cm)**	**Distância média até a base final (cm)**
Baseada em imagens	Simulação 1	(157, 82)	7,3	9,4
	Simulação 2	(155, 65)	15,8	
	Simulação 3	(145, 80)	5	
Baseada no IMU	Simulação 1	(120, 68)	32,3	26,9
	Simulação 2	(145, 50)	30,4	
	Simulação 3	(142, 46)	17,9	

Este experimento foi realizado usando apenas iluminação artificial, igual ao experimento 1, que resultou em uma maior instabilidade na movimentação do drone, como descrito na seção 5.4.1. Entretanto, diferentemente do experimento descrito na subseção anterior, não foi observado um tipo específico de interferência devido a este problema.

Por outro lado, foi observado que o movimento de rotacionar o drone para se alinhar com a base final influenciava o desempenho da simulação, especialmente nas simulações em que o IMU foi utilizado para a navegação, devido ao fato que não havia correção depois, caso a rotação não fosse realizada corretamente. Nas simulações com a localização pela visão computacional, foi possível corrigir possíveis erros durante o voo.

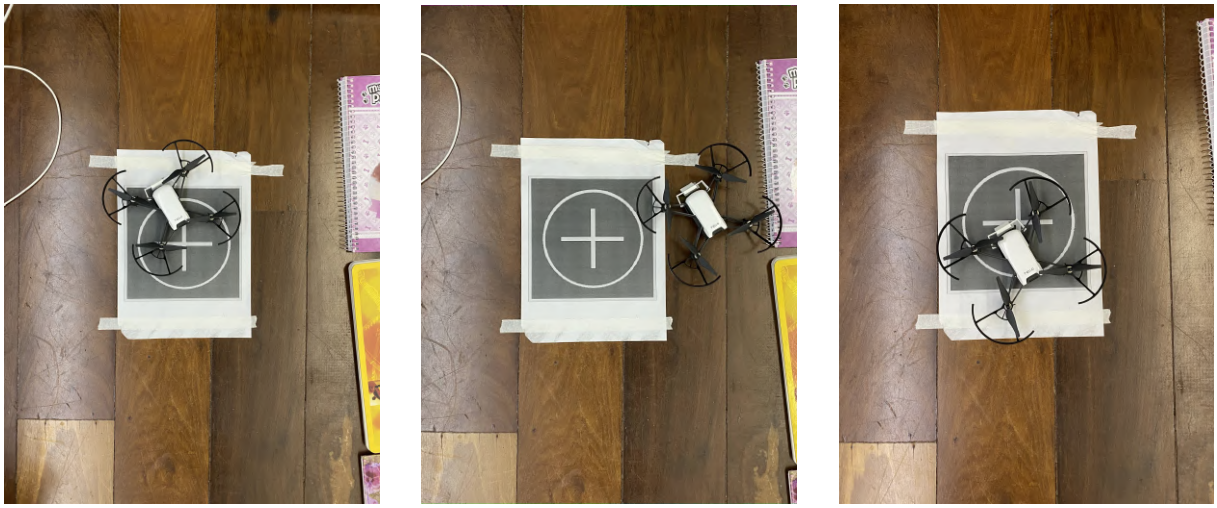
Em relação as simulações que usaram a navegação baseada em imagens, notou-se que a segunda simulação teve um resultado destoante das outras. Isso pode ser atribuído ao comando de pouso, que, assim como na decolagem possui um intervalo onde a estabilização da posição é feita por sensores embarcados e não pelo sistema de visão.

A figura 23 mostra imagens das posições finais de cada simulação que usou a navegação baseada em imagens, enquanto a figura 24 mostra imagens das posições finais de cada simulação que usou a navegação baseada no IMU. Comparando as duas, é possível perceber que a que usou a visão computacional obteve uma navegação mais precisa.

5.2.3 Experimento 3

O mapa do experimento 3 possuía 3 bases. A primeira delas, o ponto de partida do experimento, estava localizada em (0, 0), a intermediária em (80, 80) e a última, o destino final, em (150, 0). Como a configuração para este experimento contou com uma base intermediária, analisou-se o pouso correto nessa posição durante a availação da navegação. A figura 25 mostra o mapa utilizado,além de pontos de referência usados para fazer a estimação da posição.

A trajetória a ser seguida pelo drone, de acordo com o algoritmo de navegação descrito na seção 4.1, era a seguinte: decolar da base inicial, rotacionar para se alinhar com



(a) Simulação 1 (b) Simulação 2 (c) Simulação 3

Figura 23 – Posições finais do drone nas simulações usando navegação baseada em imagens do experimento 2.

(a) Simulação 1 (b) Simulação 2 (c) Simulação 3

Figura 24 – Posições finais do drone nas simulações usando navegação baseada no IMU do experimento 2.

a base intermediária, voar para a base intermediária, pousar nela, decolar dela, rotacionar para alinhar-se com a base final, voar até a base final e pousar nela.

Primeiro realizou-se as simulações utilizando a navegação baseada em imagens. Na simulação 1, o drone pousou na posição (156, 12), ficando a uma distância de 13,4 cm da base final e durante o seu percurso, a aeronave pousou parcialmente na base intermediária, ou seja, ela ficou na região branca do papel que representa a base. Já na segunda simulação, a posição final do drone foi (131, 2), a uma distância de 19,1 cm do objetivo, mas desta vez houve o pouso correto da na base intermediária. A última simulação teve um desempenho muito bom, em relação à posição final, finalizando o voo no ponto (146, 8), que ficava a uma distância de 8,9 cm da base final, porém o pouso na base intermediária foi parcial.

Figura 25 – Mapa usado no experimento 3.

Depois realizou-se as simulações utilizando a navegação baseada no IMU. Na primeira, o drone terminou seu percurso no ponto (94, -24), ficando a uma distância de 60,9 cm mas sem conseguir pousar na base intermediária. A segunda simulação ficou a uma distância de 54,6 cm do objetivo, ao finalizar seu trajeto em (96, -8), porém também sem pousar na base intermediária. A última simulação teve a melhor precisão em relação a distância, terminando o percurso na posição (124, -36), a uma distância de 44,4 cm, porém sem pousar na base intermediária. A tabela 10 sumariza os resultados de todas as simulações deste experimento.

Tabela 10 – Resultados experimento 3.

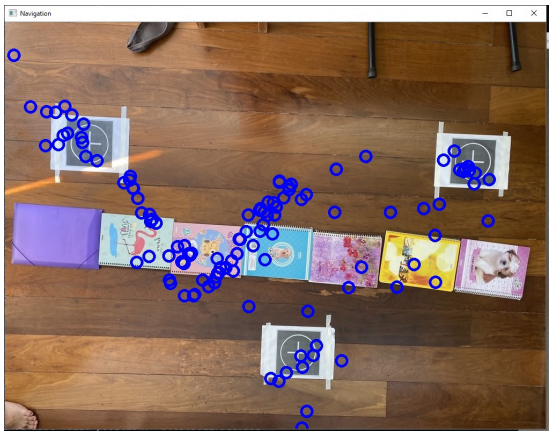
Tipo de navegação	**Número da simulação**	**Pousou na base do meio?**	**Posição final**	**Distância até a base final (cm)**	**Distância média até a base final (cm)**
Baseada em imagens	Simulação 1	Parcialmente	(156, 12)	13,4	13,8
	Simulação 2	Sim	(131, 2)	19,1	
	Simulação 3	Parcialmente	(146, 8)	8,9	
Baseada no IMU	Simulação 1	Não	(94, -24)	60,9	53,3
	Simulação 2	Não	(96, -8)	54,6	
	Simulação 3	Não	(124, -36)	44,4	

Diferentemente dos outros experimentos, este foi realizado utilizando luz natural como fonte principal de iluminação. Isto deveria ter proporcionado uma navegação mais estável, como explicando na seção 5.4.1, porém como foi necessário abrir as janelas do ambiente para deixar a luz entrar, houve um pouco de vento, causando um leve *drift*. Esse *drift* não teve um impacto perceptível na navegação baseada em imagens, pois a

distância média até a base final deste experimento ficou similar aos outros, além do fato que todas as simulações com esta navegação conseguiram chegar perto da base intermediária. Entretanto, foi possível detectar uma maior dificuldade em chegar nas bases usando a navegação baseada no IMU. Nenhuma simulação conseguiu chegar perto da base intermediária e a distância média até a base final ficou consideravelmente maior que nos outros experimentos.

A figura 26 mostra as trajetórias calculadas pela visão durante o voo da simulação 3, usando a localização por imagens e da simulação 2, usando a IMU. É possível ver que a forma do percurso da figura 26b é similar a da figura 26a, porém o uso da IMU não foi suficiente para garantir que o drone voasse a distância necessária para chegar nas bases.

(a) Trajetória da simulação 3 usando navegação baseada em imagens



(b) Trajetória da simulação 2 usando navegação baseada no IMU

Figura 26 – Trajetórias do experimento 3.

5.3 Resultados gerais

Em geral, a navegação baseada em imagens teve uma precisão melhor que a navegação usando apenas o IMU.

Os experimentos 1 e 2 mostraram que mesmo em uma navegação mais simples, que consistia em rotacionar e voar em linha reta, a utilização da localização por imagem apresentou uma precisão melhor, como foi possível ver na comparação das figuras 23 e 24. Essa precisão foi devido, principalmente, ao fato que a posição determinada pela visão computacional não propagava erro, pois o cálculo da localização do momento não depende de outros cálculos passados. O mesmo não se pode dizer sobre a IMU, que ao sofrer um desvio, continuava com esse erro sem correção.

Isso ficou evidenciado no experimento 3, onde nenhuma simulação usando apenas o IMU conseguiu chegar na base intermediária, impossibilitando a finalização correta do percursos, como visto na figura 26.

Abordando especificamente a navegação baseada em imagens, vale ressaltar que os experimentos 1, 2 e 3 geraram uma distância média de 12,25 cm, 9,4 cm e 13,8 cm respectivamente. Esses valores são muito similares, indicando que essa técnica de navegação consegue manter sua precisão em diferentes cenários. Além disso, a base do desafio Petrobras possui um tamanho de 1m por 1m, ou seja, o erro médio desta navegação garante o pouso correto na base costeira.

5.4 Dificuldades encontradas

Durante a realização deste projeto, houve diversas dificuldades encontradas. As principais serão citadas e explicadas nas próximas subseções, contendo também as ações que foram adotadas.

5.4.1 Luminosidade

A movimentação do drone Tello é muito dependente da luminosidade do local. O mais indicado é realizar voos em ambientes com locais iluminados com a luz do sol, o que não foi possível sempre.

Por isso, optou-se em realizar as simulações de cada experimento em ambientes parecidos de luz, para manter a uniformidade. O experimento 1 e 2 foram realizados usando apenas iluminação artificial, em uma sala bastante clara. Porém, o aplicativo do Tello, mesmo assim, indicava que não havia luz adequada para voo.

Já o experimento 3 foi realizado usando luz natural. Notou-se que o drone ficou mais estável durante o voo com esse tipo de iluminação, porém a visão ficou prejudicada. Quando a luz do sol estava muito forte, algumas regiões ficaram muito iluminadas, e ao passar por cima dessa regiões, a câmera do drone capturava apenas um frame muito claro, tornando quase impossível fazer a localização.

5.4.2 Bateria do drone

O drone Tello é um drone de pequeno porte, e consequentemente, sua bateria também possui um tamanho reduzido, resultando em um tempo de voo muito pequeno, como descrito na seção 2.1.6. Por causa disso, não foi possível realizar todas as simulações uma atrás da outra, pois era necessário parar para carregar o drone, atrapalhando muito o desenvolvimento, pois quando não era utilizado um cabo de alta velocidade para o carregamento, este processo demorava bastante.

Além disso, o uso do drone com pouca bateria resultava em voos bastante instáveis. O experimento 2 inicialmente foi conduzido com pouca bateria, e o drone demonstrou

um comportamento inesperado diversas vezes. Por causa disso, foi necessário repetir o experimento todo utilizando a bateria carregada.

5.4.3 Experimentos executados em ambientação inadequada

Devido ao fato que esta monografia foi desenvolvida no ano da pandemia do COVID-19, não foi possível montar uma arena que realmente simulasse e fosse das mesmas proporções que a arena do desafio. Para conseguir adaptar isso, foi necessário utilizar locais muito menores que o previsto e adequar o ambiente disponível. Por isso as bases utilizadas são extremamente menores que as oficiais e a linha de cadernos representa o duto do mapa.

Por causa do espaço pequeno, houve a necessidade de utilizar os algoritmos mais precisos possíveis, pois um desvio de 20 cm nos mapas dos experimentos é suficiente para fazer com que o drone não pouse na base, enquanto esse deriva não proporciona um erro tão grande no mapa oficial, que possui bases com dimensões de 1 m por 1 m.

5.4.4 Isolamento social

Outra dificuldade gerada pela pandemia foi a falta de encontros entre os autores. Quando se iniciou o processo de junção do sistema de visão com a movimentação, os autores optaram por tentar realizar essa atividade virtualmente, que não funcionou direito. Logo foi necessário fazer encontros para realizar o desenvolvimento final e os testes, mesmo não sendo o mais indicado no cenário da pandemia.

6 CONCLUSÃO

O uso de drones tem crescido intensamente no últimos anos, e é previsto que seu uso aumente ainda mais no futuro. Por isso está sendo visto um número considerável de publicações e pesquisas realizadas nesta área, como mostra (3). Também é certo que a movimentação por imagens é uma área de pesquisa muito importante, visto que não é utilizada somente em drones, mas também em carros autônomos por exemplo.

Nesse contexto, o presente trabalho teve como finalidade desenvolver um sistema de localização baseado em imagens para um drone de pequeno porte, além de testar esse sistema em uma simulação de uma base petrolífera, que teve que ser simplificada pelas limitações impostas pela pandemia da COVID-19.

Nessa monografia é apresenta como esse sistema foi modelado, dividindo-se o desenvolvimento em duas partes principais: a navegação e a visão. Esses dois subsistemas têm papéis distintos, mas se conectam para cumprir o objetivo do trabalho, que é justamente a navegação por visão. Além da modelagem, encontram-se descritos os experimentos realizados para avaliar a eficácia do sistema, onde são feitos dois tipos de experimento: utilizando a navegação por imagens e utilizando somente o sistema de navegação inercial do drone.

Nesses experimentos, esperava-se que a movimentação realizada somente pelo sistema de navegação inercial tivesse mais erros do que a navegação feita por imagens, devido ao fato do drone não possuir nenhum mecanismo para correção da sua posição, durante todo o trajeto. Tal hipótese foi comprovada pelos resultados obtidos nos experimentos, que foram realizados com diferentes configurações, onde em todas as execuções a navegação por imagem foi mais precisa.

Além disso, também pode-se perceber que nos experimentos sem navegação por imagem, existe um efeito que cascateia os erros à medida que distância maiores são percorridas, como pode ser visto no fato que o experimento 3 possui a maior diferença entre as distâncias dos dois tipos de experimentos.

Pode-se dizer então, que o trabalho conseguiu atingir o objetivo estabelecido, fornecendo um sistema capaz de movimentar um drone de pequeno porte e localizá-lo numa região por meio de imagens obtidas durante o voo.

Com relação aos trabalhos futuros, sugere-se a implementação de um estimador de posição que utilize algoritmos de *scene matching* com parâmetros que se adaptam automaticamente ao local de voo do drone, de forma a obter a melhor estimação possível com um dado algoritmo.

Além disso, pode-se também procurar uma forma de unir os GNSS à navegação por imagens, fazendo com que se tenha um sistema ainda mais robusto e preciso. Nesse caso, a estimação da posição se daria por 3 meios: pelo sistema de geolocalização por satélites, pelas imagens do terreno e pelo IMU do drone.

A precisão de um GPS comum em um *smartphone* é de aproximadamente 4,9 metros (20), podendo ser pior, caso o aparelho esteja perto ou dentro de um edifício. Como as precisões dos GPS utilizados em *smartphones* e em drones são aproximadamentes as mesmas, esse sistema poderia não só ser utilizado em situações em que os GNSS não estão disponíveis, mas também para obter uma maior precisão mesmo com a utilização dos GNSS, cuja precisão pode não ser suficiente, a depender do objetivo da missão a ser praticada e do *hardware* utilizado.

REFERÊNCIAS

1 Luiz Junior, F. *Estratégia líder-seguidor utilizando VANTs para a navegação através do reconhecimento de imagens do terreno*. 119 p. Mestrado em Sistemas e Computação — Instituto Militar de Engenharia, Rio de Janeiro, 2020. 24 jan. de 2020. Disponível em: <http://www.comp.ime.eb.br/pos/?l=0&p=29&q=2020_1>. 16, 18, 26

2 BUCKLEY, J. *Air Power in the Age of Total War*. [S.l.]: Routledge, 2006. 269 p. 16

3 RESEARCH, G. S. *Drones: Reporting for Work*. 2016. 17 maio de 2020. Disponível em: <https://www.goldmansachs.com/insights/technology-driving-innovation/drones/>. 16, 56

4 JAYANTHI, N.; SREEDEVI, I. Comparison of image matching techniques. *International Journal of Latest Trends in Engineering and Technology*, v. 7, p. 396–401, 10 2018. 19 de maio 2020. Disponível em: <https://www.ijltet.org/journal/147599146852.1167.pdf>. 20, 30, 31

5 HALL, E. Scene matching and recognition. In: _____. *Computer Image Processing and Recognition*. Cambridge, Massachussets: Academic Press, 1979. p. 468–550. 20

6 JUAN, S.; QING-SONG, X.; JING-HUA, Z. A scene matching algorithm based on surf feature. In: *2010 International Conference on Image Analysis and Signal Processing*. [s.n.], 2010. (IASP, 10), p. 434–437. 19 maio de 2020. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5476080&tag=1>. 21, 23, 24, 25, 26

7 OPENCV. *Introduction to SIFT (Scale-Invariant Feature Transform)*. 2020. 21 out. de 2020. Disponível em: <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html>. 21, 22, 23

8 PUC-RIO. *SIFT (Scale Invariant Feature Transform)*. 2020. 21 out. de 2020. Disponível em: <https://www.maxwell.vrac.puc-rio.br/17050/17050_5.PDF>. 22

9 BAY, H.; ESS, A.; TUYTELAARS, T.; GOOL, L. V. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, v. 110, p. 346–359, 2008. 18 maio de 2020. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1077314207001555>. 24

10 DERPANIS, K. G. *Overview of the RANSAC Algorithm*. 2010. 22 out. de 2020. Disponível em: <http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac.pdf>. 26

11 OPENCV. *OpenCV Library*. 2020. 14 abr. de 2020. Disponível em: <https://opencv.org>. 26

12 PYTHON. *Virtual Environments and Packages*. 2020. 3 ago. de 2020. Disponível em: <https://docs.python.org/3/tutorial/venv.html>. 26

13 ROBOTICS, R. *Tello specifications*. 2020. 12 maio de 2020. Disponível em: <https://www.ryzerobotics.com/tello/specs>. 27, 28

14 CLAYE, J. *Tello Mirror Clip*. 2020. 9 ago. de 2020. Disponível em: <https://www.thingiverse.com/thing:2911427/files>. 27, 28

15 ROBOTICS, R. *Tello SDK*. 2020. 12 maio de 2020. Disponível em: <https://www.ryzerobotics.com/tello/downloads>. 28

16 HANYAZOU. *Biblioteca TelloPy*. 2020. 18 maio de 2020. Disponível em: <https://github.com/hanyazou/TelloPy>. 28, 29

17 FUENTES, D. *Biblioteca DJITelloPy*. 2020. 18 maio de 2020. Disponível em: <https://github.com/damiafuentes/DJITelloPy>. 29

18 KARAMI, E.; PRASAD, S.; SHEHATA, M. Image matching using sift, surf, brief and orb: Performance comparison for distorted images. In: *2015 Newfoundland Electrical and Computer Engineering Conference*. [s.n.], 2015. (NECEC, 24). 19 maio de 2020. Disponível em: <https://arxiv.org/pdf/1710.02726.pdf>. 30, 31, 32

19 LARC. *Desafio de Robótica Petrobras*. 2020. 18 maio de 2020. Disponível em: <http://www.cbrobotica.org/?page_id=1303>. 33, 34, 62, 63

20 DIGGELEN, F. van; ENGE, P. The world's first gps mooc and worldwide laboratory using smartphones. In: *Proceedings of the 28th International Technical Meeting of the Satellite Division of The Institute of Navigation*. Tampa, Florida: ION, 2015. (ION, 28), p. 361–369. 57

ANEXO A – O DESAFIO PETROBRAS

A.1 As fases do desafio

O problema possui 4 fases, sendo que cada fase é um problema que o drone deve tratar autonomamente. Entretanto, algumas fases permitem interações humanas em casos muito específicos.

A.1.1 Fase 1: Localização e mapeamento

Essa fase exige que o robô faça o reconhecimento da arena e mapeie o ambiente, entendendo onde estão as bases de aterrissagem e decolagem, o oleoduto e a base costeira.

A tarefa

O robô deve deixar a base costeira, atravessar a arena e pousar uma vez em cada base existente, suspensa ou terrestre. Depois disso, o drone deve retornar à base costeira.

As bases suspensas são fixas e as bases terrestres são colocadas aleatoriamente na arena para cada rodada desta tarefa.

A.1.2 Fase 2: Prevenção e sensoriamento

Para avançar para as próximas fases, o robô deve ter as bases, o oleoduto e todos os outros componentes da arena já mapeados. Para isso, as bases terrestres estarão nas mesmas posições definidas na fase 1. Na fase 2, a tarefa é identificar quais sensores de monitoramento estão relatando defeitos no oleduto. Para isso, a tubulução terá cinco sensores de monitoramento, representados por quadrados coloridos, de 5 centímetros de largura, em vermelho ou verde, onde o sensor que detecta o defeito tem a cor vermelha.

A tarefa

O robô deve verificar a tubulação e detectar os sensores que estão relatando defeitos. Para cada sensor vermelho detectado, o robô deve executar algum movimento, som ou piscar de luzes e gravar os dados de cada sensor em um arquivo. Depois de passar pelos 5 sensores, o robô deve retornar autonomamente à base costeira.

A.1.3 Fase 3: Manutenção de gás metano (CH_4)

Nesta fase, cada base suspensa e uma base terrestre recebem uma etiqueta referente a um mostrador digital de gás que possui dois valores, um indicando a porcentagem de gás e o outro indicando o ajuste do zero. Esse é um procedimento padrão nas empresas de petróleo, que precisa ser feito periodicamente. Essa fase visa verificar a porcentagem correta de gás metano (CH_4) nas bases e para isso o robô voador deve indicar quais bases precisam de calibração e por quê.

A tarefa

O robô deve visitar as bases suspensas e terrestres para detectar e verificar as porcentagens de metano (CH_4). Em cada base existem dois monitores digitais e o robô deve relatar se ambos, ou um deles, ou nenhum deles está em conformidade com os padrões estabelecidos:

- Percentual de gás: entre 45% e 55% inclusive.
- Ajuste do zero: entre -5% e 5% inclusive.

Um exemplo do mostrador que o drone deve procurar pode ser visto na figura 27

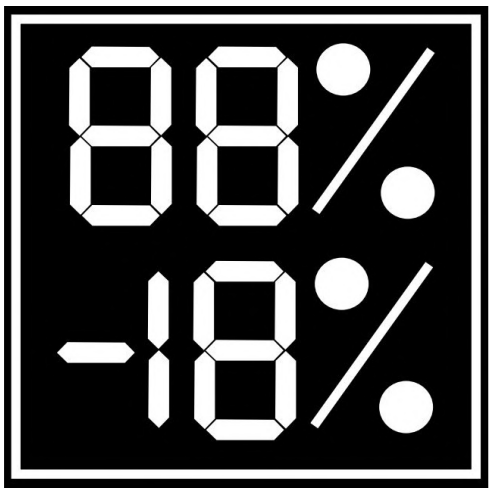


Figura 27 – Display digital com 2 valores, o valor superior identifica a porcentagem de gás e o valor inferior identifica o ajuste zero. Tamanho da etiqueta: 11cm x 11cm

A.1.4 Fase 4: Transporte de equipamentos

Além de garantir que as operações sejam monitoradas e protegidas, os robôs voadores devem poder transportar rapidamente equipamentos entre as bases.

A tarefa

Nesta fase, o robô precisa transportar pacotes de uma base para outra. O robô deve deixar a base costeira, procurar pacotes nas bases suspensas e/ou terrestres e verificar para qual base o pacote deve ser enviado.

As bases são mapeados, sendo identificadas pelas letras: A, B, C, D e E (o mapeamento ocorre antes da execução dos experimentos). Cada pacote possui, no topo, um código QR com a letra da base onde o pacote deve ser entregue. Os exemplos de código QR podem ser vistos na figura 28:

Figura 28 – QR codes que identificam cada base

Os pacotes serão cubos com lados de 10 centímetros, todos em preto, com um código QR colocado no topo. O código QR terá um fundo branco. Os cubos serão feitos de isopor, com uma fina placa de metal na parte superior, todos cobertos com papel preto fosco, e não deverão pesar mais de 50 gramas.

A.2 Especificação dos tamanhos das bases do campo de testes

A especificação dos tamanhos das bases, assim como o formato dos símbolos que definirão como o robô identificará cada base, pode ser encontrado na figuras 29 e 30, todas essas imagens foram retiradas do desafio original, que pode ser acessado no website da competição latino-americana de robótica (19).

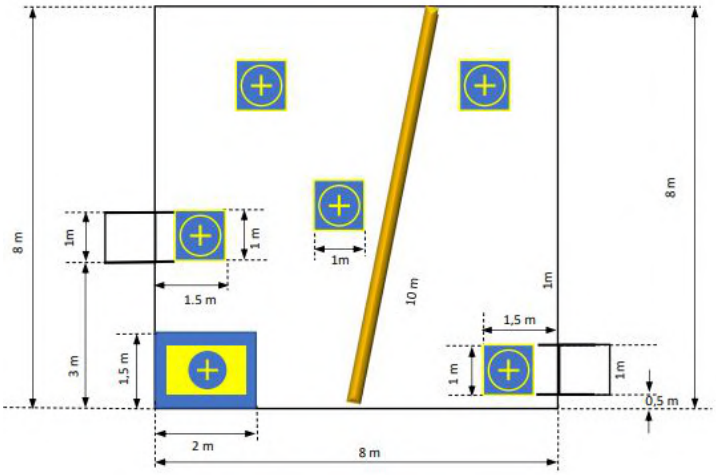


Figura 29 – Dimensões do campo de testes. Fonte: LARC(19)

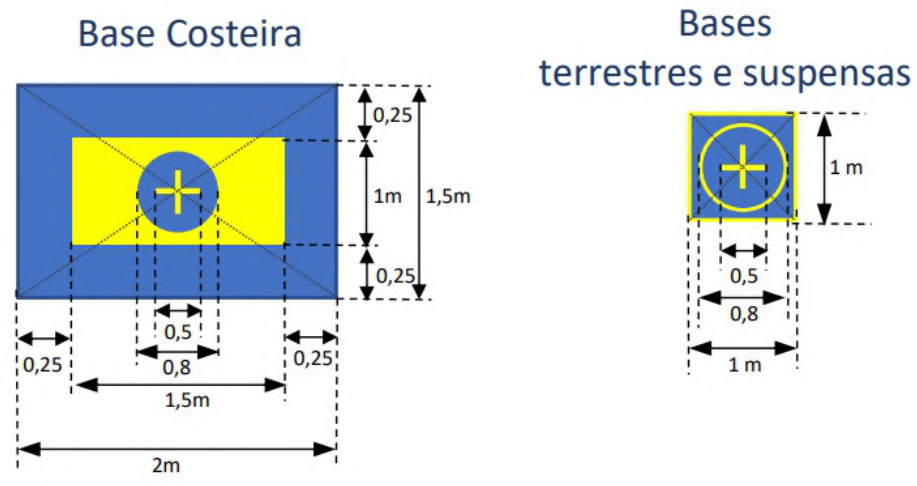


Figura 30 – Dimensões das bases. Fonte: LARC(19)