

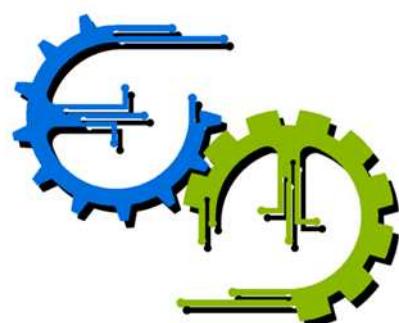


TRABALHO DE GRADUAÇÃO

**SLAM Visual Monocular
baseado em Filtro de Kalman Estendido
para Robôs Aéreos**

De Hong Jung
Rodrigo de Lima Carvalho

Brasília, julho de 2016,



**ENGENHARIA
MECATRÔNICA**
UNIVERSIDADE DE BRASÍLIA

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

TRABALHO DE GRADUAÇÃO

SLAM Visual Monocular baseado em Filtro de Kalman Estendido para Robôs Aéreos

**De Hong Jung
Rodrigo de Lima Carvalho**

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

Banca Examinadora

Prof. Geovany Araújo Borges, ENE/UnB _____
Orientador

Prof. Mariana Costa Bernardes Matias, _____
ENE/UnB
Co-Orientadora

Prof. Antônio Padilha Lanari Bó, ENE/UnB _____
Examinador interno

Prof. Luis Felipe Cruz Figueiredo, ENE/UnB _____
Examinador interno

Brasília, julho de 2016,

FICHA CATALOGRÁFICA

DE HONG, JUNG; RODRIGO, DE LIMA CARVALHO

SLAM Visual Monocular baseado em Filtro de Kalman Estendido para Robôs Aéreos

[Distrito Federal] 2016.

x, 101p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2016). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1. Localização e Mapeamento Simultâneos

2.Filtro de Kalman Estendido

3. Robótica

I. Mecatrônica/FT/UnB

II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

DE HONG, JUNG; RODRIGO, DE LIMA CARVALHO, (2016). SLAM Visual Monocular baseado em Filtro de Kalman Estendido para Robôs Aéreos. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº022, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 101p.

CESSÃO DE DIREITOS

AUTORES: De Hong Jung e Rodrigo de Lima Carvalho

TÍTULO DO TRABALHO DE GRADUAÇÃO: SLAM Visual Monocular baseado em Filtro de Kalman Estendido para Robôs Aéreos.

GRAU: Engenheiro

ANO: 2016

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Trabalho de Graduação e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desse Trabalho de Graduação pode ser reproduzida sem autorização por escrito do autor.

De Hong Jung e Rodrigo de Lima Carvalho

Campus Darcy Ribeiro, SG-11, Universidade de Brasília

70919-970 Brasília – DF – Brasil.

Dedicatórias

Aos meus pais, Jonildo e Margareth.

Rodrigo de Lima Carvalho

*Aos meus falecidos avós, Chang Sun Jung e
Hye Jae Jung Lee.*

De Hong Jung

Agradecimentos

Adradeço primeiramente aos meus orientadores por todo o apoio e ensinamentos dados durante este trabalho. Agradeço à prof. Mariana Bernardes por toda a sua ajuda e paciência em todos os projetos em que participei no LARA. A sua competência e dedicação é algo que não se vê todo dia. Agradeço ao prof. Geovany Borges por todo o seu conhecimento do além que foram essenciais para a execução deste trabalho.

Queria agradecer a todas as amizades que fiz durante o curso. Foram essenciais para a minha sanidade mental e para me dar força para continuar batalhando no curso. Passamos muitas noites sem dormir, fizemos inúmeros trabalhos em cima de hora, festejamos vários finais de semestre, rimos de muitas situações inusitadas e crescemos como pessoa juntos. Em especial, gostaria de agradecer à melhor turma da Mecatrônica de todos os tempos: a Vigésima Sétima Turma de Engenharia Mecatrônica da Universidade de Brasília, em particular meus amigos Pedro Hecksher, Eric Torlig, Cristiana Miranda, Daniel Vincentin, Franscisco Gomes, Vitor Caveira, Gabriel Cataldi, Matheus Takahashi, Marcos Pereira, Marco Emílio, Filipe Caixeta, Jessé Barreto e Rodrigo Teixeira. Não poderia pedir uma turma melhor.

Também agradeço às equipes na qual participei durante o curso, em particular a Equipe Draco Volans e a UnBeatables, onde eu aprendi que não é necessário mais de duas horas de sono por dia. Aprendi com excelentes pessoas, como trabalhar em grupo e lidar com situações totalmente aleatórias e desesperadoras.

Queria agradecer à todos os membros do LARA por todo o apoio e receptividade, que tornaram o ambiente de trabalho um lugar mais agradável de se conviver. Em especial, agradeço aos meus amigos Lucas Rato, Luis Figueiredo, Roberto Baptista, Yuri Rocha, Gabriel Moises, Jhonantans, Raphael, Henrique, Lucas de Levy e Marcela Carvalho por todas as zueras feitas nesse período. Agradeço também à Geordana Maeda por me massagear durante a escrita desse agradecimento.

Agradeço aos meus amigos do ~sudoku~ por sempre me apoiarem e acreditarem nos meus sonhos, em particular, aos meus amigos Foguinho, Coelho, Guto, Juliano, Pedroka, Kabeça e Camila.

Obviamente, agradeço ao Rodrigo Carvalho por ser a melhor dupla de TG que eu poderia ter. O seu foco e dedicação foram essenciais para o desenvolvimento do trabalho.

Por fim, agradeço ao meu pai por todo o apoio que me ofereceu durante todo o meu crescimento como pessoa.

De Hong Jung

Devo agradecer muitas pessoas que estiveram comigo durante esses anos de graduação, afinal de contas não caminhei sozinho. Muitas pessoas foram indispensáveis para a minha formação e, nesse espaço, tentarei ser breve para demonstrar meu agradecimento a algumas delas.

Primeiramente, agradeço aos mestres que me ensinaram o que eu pude aplicar nesse trabalho. Em especial, agradeço meus orientadores, Prof. Geovany Borges e Prof.^a Mariana Bernardes, por terem mostrado o caminho para que esse trabalho pudesse ser realizado. Não posso deixar de agradecer meu amigo e colega de trabalho, De Hong, por estar sempre disponível trabalharmos juntos nesse projeto.

Agradeço ainda os colegas do LARA (Lucas, Roberto, Luis, Pedro, Eric, Gabriel...). Obrigado por toda a paciência e disponibilidade para tirar algumas dúvidas e ajudar no que fosse possível. Todos ali, professores e alunos de graduação, mestrado e doutorado, foram importantes e contribuíram de alguma maneira para que esse trabalho pudesse ser concluído.

Não poderiam faltar os meus queridos colegas engenheiros da 4flyers. Marcela, Artur e Giordano, vocês fazem parte disso também.

Gostaria de agradecer também meus amigos que não são colegas de curso, mas são parceiros para a vida toda e sempre me ajudaram a superar as dificuldades que surgiram durante o período de graduação.

Agradeço a minha namorada, Carolina, por me apoiar, me motivar sempre e vibrar junto comigo as minhas conquistas. Não tenho dúvidas de que ela contribuiu fortemente para que eu tivesse “gás” para ir até o fim.

Por fim, agradeço toda a minha família, em especial meus irmãos, Caio e Gabriel, e meus pais, Jonildo e Margareth, com quem eu compartilhei todas as emoções vividas durante esses anos. Foram eles que me deram o apoio para vencer todas as dificuldades. Obrigado por fazerem eu me sentir tão capaz de ser o que eu sempre sonhei em ser.

Rodrigo de Lima Carvalho

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema de localização e mapeamento simultâneos utilizando informações de imagens de uma câmera monocular e dados de navegação de um quadrirroto de baixo custo em ambientes fechados. O sistema foi baseado no Filtro de Kalman Estendido para realizar correções dinâmicas de posição do veículo e de pontos de interesse do mapa. É aplicado, neste trabalho, um método de detecção de pontos de interesse para extrair informações características do ambiente. Um mapa probabilístico tridimensional é construído através de técnicas de projeção de pontos de imagens. Além disso, é abordado, neste trabalho, uma técnica de casamento entre pontos de interesse observados na imagem com pontos presentes no mapa. O método de mapeamento e localização proposto é validado realizando testes em ambientes desconhecidos com diferentes características. Os resultados obtidos demonstram que o sistema consegue estimar a posição do veículo no ambiente, sendo corrigida recursivamente pelas observações consecutivas de pontos característicos presentes no local, que também têm suas posições atualizadas no mapa.

Palavras Chave: slam, visual, monocular, fke, filtro, quadrirroto, robótica, mapeamento, localização

ABSTRACT

This work describes the development of a simultaneous localization and mapping system using images of a monocular camera and navigation data of a low cost quadrotor in indoor environments. The system was based on the Extend Kalman Filter for dynamic position corrections of the vehicle and features of the map. In this work, it is applied a feature detection method to extract characteristics information of the environment. A probabilistic tridimentional map is created through image projection technics. Besides that, in this work, it is explained a feature matching method between observed points in the image and features in the map. The localization and mapping system method proposed is validated through experiments in unknown environments with different features. The results show that the system can estimate the position of the vehicle in the environment, being recursively corrected by consecutives feature observations, which positions are also updated in the map.

Keywords: slam, visual, monocular, ekf, filter, quadrotor, robotics, mapping, localization

SUMÁRIO

1 INTRODUÇÃO	1
1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO	1
1.2 VEÍCULOS AÉREOS NÃO TRIPULADOS - VANTS	3
1.3 LOCALIZAÇÃO E MAPEAMENTO	4
1.4 DESCRIÇÃO DO PROBLEMA	5
1.5 OBJETIVOS DO PROJETO	6
1.6 RESULTADOS OBTIDOS	6
1.7 APRESENTAÇÃO DO MANUSCRITO	7
2 FUNDAMENTOS	8
2.1 REVISÃO BIBLIOGRÁFICA	8
2.2 AR.DRONE PARROT 2.0	10
2.2.1 PROCESSAMENTO EMBARCADO	12
2.3 FERRAMENTAS COMPUTACIONAIS	13
2.3.1 ROS	13
2.3.2 RVIZ	14
2.4 MODELAGEM MATEMÁTICA	16
2.4.1 SISTEMAS DE COORDENADAS	16
2.4.2 SISTEMA DE ORIENTAÇÃO	16
2.4.3 TRANSFORMAÇÃO DE COORDENADAS	17
2.4.4 ALGORITMO ORB PARA DETECÇÃO DE PONTOS DE INTERESSE	18
2.4.5 PROJEÇÃO DE IMAGENS	20
2.4.6 <i>Bootstrap</i>	21
2.4.7 DISTÂNCIA DE MAHALANOBIS	22
2.4.8 REPRESENTAÇÃO GRÁFICA DE INCERTEZAS NA IMAGEM	23
2.4.9 FILTRO DE KALMAN	24
2.4.10 FILTRO DE KALMAN ESTENDIDO	26
3 DESENVOLVIMENTO	28
3.1 INTEGRAÇÃO DO <i>Hardware</i> COM ROS E RVIZ	28
3.1.1 <i>Driver ardrone_autonomy</i>	28
3.1.2 COMUNICAÇÃO DO RVIZ COM O NÓ PRINCIPAL NO COMPUTADOR DE BASE	29
3.1.3 EXECUÇÃO <i>offline</i>	31

3.2	CALIBRAÇÃO DE CÂMERA.....	32
3.3	REPRESENTAÇÃO DO MAPA	34
3.4	FILTRO DE KALMAN ESTENDIDO PARA SLAM	35
3.4.1	ETAPA DE PREDIÇÃO.....	37
3.4.2	ETAPA DE CORREÇÃO.....	38
3.5	ARQUITETURA.....	39
3.5.1	VISÃO GERAL	40
3.5.2	PROCEDIMENTOS APÓS LEITURAS DE DADOS DE NAVEGAÇÃO.....	40
3.5.3	PROCEDIMENTOS APÓS LEITURAS DA CÂMERA	41
3.6	DESCRIÇÃO DO CÓDIGO.....	42
3.6.1	LINGUAGEM DE PROGRAMAÇÃO	42
3.6.2	BIBLIOTECAS IMPORTANTES.....	42
3.6.3	ESTRUTURA.....	43
4	RESULTADOS	45
4.1	ESTIMATIVA DE POSIÇÃO A PARTIR DOS DADOS DE NAVEGAÇÃO.....	45
4.2	PROJEÇÃO DE UM PONTO 3D DO MAPA PARA A IMAGEM 2D	48
4.3	PROJEÇÃO DE UM PONTO 2D DA IMAGEM PARA UM PONTO 3D NO MAPA	49
4.4	CASAMENTO DE PONTO OBSERVADO COM PONTO CORRESPONDENTE NO MAPA	50
4.5	CASAMENTO ENTRE PONTOS DE INTERESSE BASEADO NA ELIPSE DE INCERTEZA	52
4.6	CORREÇÃO DA POSIÇÃO DE PONTOS DE INTERESSE	54
4.7	COMPARAÇÃO DA ESTIMATIVA ANTES E APÓS CORREÇÃO	55
4.8	SLAM VISUAL MONOCULAR COMPLETO	59
4.8.1	VOO NA COBERTURA DE UM PRÉDIO	59
4.8.2	VOO NO AMBIENTE DE TRABALHO DO LARA	61
5	CONCLUSÕES	66
REFERÊNCIAS BIBLIOGRÁFICAS		68
ANEXOS		70
I	PLANTA DA COBERTURA DO APARTAMENTO	71
II	 DESCRIÇÃO DO CONTEÚDO DO CD	72

LISTA DE FIGURAS

1.1 Ilustração de projeto no qual quadrirotores são utilizados para entrar em prédios abandonados, em perigo de desmoronamento. (Fonte: Projeto <i>DARPA's new Fast Lightweight Autonomy (FLA) program</i>). ¹⁾	2
1.2 Arquitetura básica de veículos aéreos não tripulados.....	3
1.3 Veículos utilizados para mapeamento.(Fonte: Google ²)	5
2.1 AR.Drone Parrot 2.0.....	10
2.2 Dimensões do veículo.....	11
2.3 Estimativa de velocidade linear no eixo x provenientes do processamento embarcado: Estimativa a partir da odometria visual (azul), estimativa da velocidade do modelo aerodinâmico a partir de leituras diretas dos sensores de medição inercial (verde), estimativa após fusão sensorial (vermelho). (Fonte: [1]).....	12
2.4 Plano quadriculado em ambiente RViz.....	14
2.5 Representação de localização e orientação do veículo.....	15
2.6 Representação de pontos de interesse.....	15
2.7 Representação da origem no plano.	15
2.8 Representação dos sistemas de coordenadas	16
2.9 Representação do sistema de orientação <i>roll-pitch-yaw</i>	17
2.10 Representação da transformação de coordenadas usando matrizes de transformação homogêneas	18
2.11 Representação da transformação de coordenada para encontrar a posição relativa do ponto de interesse em relação ao quadrirotores utilizando matriz de transformação homogênea inversa.....	19
2.12 Pontos de interesse detectados utilizando algoritmo ORB	19
2.13 Elipse para representação gráfica de incertezas.	23
2.14 Elipse 3-sigma de uma distribuição de amostras a partir de uma simulação de Monte Carlo.	24
3.1 Representação da integração <i>Hardware</i> , ROS e RViz.	30
3.2 Resultado da integração do <i>Hardware</i> com o ROS e o RViz.....	31
3.3 Mapeamento do controle sem fio Microsoft XBox 360.	32
3.4 <i>Chessboard</i> utilizado para a calibração da câmera	33
3.5 Interface da ferramenta de calibração de câmera do ROS.....	33
3.6 Diferença entre uma imagem com distorção e uma imagem retificada	34

3.7	Arquitetura do SLAM Visual implementado.	41
3.8	Estrutura do código.	43
4.1	Ambiente do LARA onde foi feito o voo teste	46
4.2	Vista superior do resultado de estimativa de posição em 2D. Teste feito em torno das mesas realizando uma volta completa e encerrando o voo um pouco à frente da posição inicial.	46
4.3	Velocidade e aceleração linear no eixo x.	47
4.4	Velocidade e aceleração linear no eixo y.	47
4.5	Ponto de interesse observado (verde) e ponto projetado (vermelho). Tamanho total da imagem $(u_t, v_t) = (598, 292)$.	48
4.6	Distâncias de Mahalanobis e distâncias em pixels de pontos observados com o ponto no mapa.	51
4.7	Distância de Mahalanobis e distância em pixels de pontos observados com o ponto no mapa na terceira situação.	52
4.8	Verificação de casamento entre o ponto de interesse observado e pontos do mapa baseado nas elipses de incerteza	53
4.9	Vista superior do SLAM implementado baseado no casamento usando elipses de incerteza. Em verde, é apresentado o percurso estimado apenas pela predição, e em amarelo, é apresentado o percurso estimado pela predição junto com a correção.	54
4.10	Posição dos pontos de interesse no tempo.	55
4.11	Ambiente onde foi feito o voo do teste de comparação da estimativa antes e após correção.	56
4.12	Visualização do mapa 3D gerado no teste de comparação da estimativa antes e após correção em ambiente RVIZ. Em verde, é apresentado o caminho estimado apenas pela predição, e em amarelo, é apresentado o caminho estimado pela predição e a correção.	57
4.13	Visualização do mapa 2D gerado no teste de comparação da estimativa antes e após correção em ambiente RVIZ. Em verde, é apresentado o caminho estimado apenas pela predição, e em amarelo, é apresentado o caminho estimado pela predição e a correção.	57
4.14	Gráfico de intervalo de confiança do filtro a partir de dados de posição no eixo X de translação.	58
4.15	Gráfico de intervalo de confiança do filtro a partir de dados de posição no eixo Y de translação.	58
4.16	Gráfico de intervalo de confiança do filtro a partir de dados de posição no eixo Z de translação.	59
4.17	Vista do modelo tridimensional da cobertura do prédio. Fonte: Google Earth	60
4.18	Vista superior do mapeamento e da localização do quadriroto em uma cobertura de um prédio	61
4.19	Vista de perspectiva do mapeamento e da localização do quadriroto em uma cobertura de um prédio	62

4.20	Vista superior do mapeamento e da localização do quadrirroto em uma cobertura do prédio sobreposto com a planta original	63
4.21	Vista superior do mapeamento e da localização do quadrirroto no LARA	64
4.22	Vista de perspectiva do mapeamento e da localização do quadrirroto no LARA	65
I.1	Planta da cobertura do apartamento	71

LISTA DE TABELAS

3.1	Mensagem do tópico <i>/navdata</i>	29
4.1	Resultado do teste de projeção de um ponto 3D para um ponto 2D.	48
4.2	Resultado do teste de projeção de um ponto 3D para um ponto 2D.	49
4.3	Comparação entre o caminho percorrido estimado e o caminho medido da cobertura do apartamento através de pontos de referências.	60
4.4	Comparação entre os segmentos de caminho percorrido pela estimação e os caminho medido no LARA	62

LISTA DE SÍMBOLOS

Símbolos Latinos

v	Velocidade linear	[m/s]
a	Aceleração linear	[m/s ²]
P	Matriz de covariâncias	
\mathbf{x}_k	Posição e orientação do veículo e dos pontos de interesse no instante k	
\mathcal{O}	Sistemas de coordenadas	
\mathbf{R}	Matriz de rotação	
\mathbf{p}	Posição 3D relativa	
\mathbf{M}	Matriz de calibração da câmera	
\mathbf{T}	Matriz de transformação homogênea	

Símbolos Gregos

ω	Velocidade angular	[rad/s]
ψ	Rotação no eixo z - <i>Yaw</i>	[°]
θ	Rotação no eixo y - <i>Pitch</i>	[°]
ϕ	Rotação no eixo x - <i>Roll</i>	[°]
σ	Desvio padrão	

Subscritos

ref	referência
fer	ferramenta
sis	sistema
des	desejado

Sobrescritos

- Variação temporal
- Valor médio
- ^ Estimação ou predição

Siglas

VANT	Veículo Aéreo Não Tripulado
GPS	Sistema de Posicionamento Global
LIDAR	<i>Light Detection And Ranging</i>
LARA	Laboratório de Automação e Robótica da Universidade de Brasília
UMI	Unidade de Medição Inercial
SLAM	<i>Simultaneous Localization and Mapping</i> - Localização e Mapeamento Simultâneos
FK	Filtro de Kalman
FKE	Filtro de Kalman Estendido
ORB	<i>Oriented FAST and Rotated BRIEF</i>
ROS	<i>Robot Operating System</i>
RViz	<i>ROS Visualization</i>
UnB	Universidade de Brasília
FAST	<i>Features from Accelerated Segment Test</i>
BRIEF	<i>Binary Robust Independent Elementary Features</i>

Capítulo 1

Introdução

Esse capítulo traz a contextualização e a apresentação dos objetivos desse trabalho, mostrando o que foi atingido. Por fim, será feita a apresentação do manuscrito.

1.1 Contextualização e motivação

Os constantes avanços na robótica no século XXI têm tornado cada vez mais comum a presença de robôs em várias atividades do cotidiano. Robôs são constantemente usados em indústrias, procedimentos médicos ou no transporte de materiais e pessoas. Cada vez mais os robôs ganham capacidade de substituir os seres humanos em tarefas que podem colocar a vida de pessoas em risco, tarefas tediosas ou repetitivas e tarefas que essas máquinas desempenham trabalho mais preciso.

Um dos ramos da robótica, chamado de robótica móvel, envolve dispositivos que são dotados de sistema de locomoção e capazes de transportar cargas com algum nível de autonomia. Dentre os robôs móveis estão os robôs aéreos, que são robôs capazes transportar alguma carga se sustentando no ar. Os robôs aéreos, também chamados de veículos aéreos não tripulados (VANTs), podem ser remotamente guiados, tele-operados¹ ou agir de forma autônoma.

Veículos aéreos não tripulados são utilizados principalmente para realizar tarefas de alto risco para o ser humano ou quando se torna economicamente viável um voo não tripulado. Esses veículos são normalmente dotados de bom poder computacional, para realizar tarefas de estabilização, aquisição de dados e até voos autônomos.

Historicamente o uso de artefatos voadores não tripulados está ligado à ações militares do século XX, quando bombas eram guiadas remotamente para atingir alvos distantes do ponto de lançamento. Com o avanço da tecnologia, foi possível a aplicação de sistemas teleguiados e autônomos em aeronaves não tripuladas para outros fins que não fossem militares.

Cada vez mais os VANTs se tornam acessíveis a uma parcela maior da população, principal-

¹Na teleoperação, o operador não está necessariamente vendo o veículo, mas recebe informações de imagens oriundas de dispositivos embarcados para operá-lo remotamente

mente com o surgimento de veículos aéreos multirrotores, ou seja, veículos que possuem vários motores acoplados à hélices que são capazes de se sustentar no ar.

Os quadrirotores, um tipo específico de veículo multirrotor caracterizado por manter-se no ar utilizando quatro propulsores, vem sendo uma opção relativamente barata e bastante útil para os usuários. O quadrirotores, quando devidamente controlado, é capaz de decolar e pouso na forma vertical, além de pairar no ar sem se mover significativamente no plano horizontal. Por essas características, o quadrirotores possui uma capacidade de realizar voos tanto em locais fechados quanto abertos, oferecendo estabilidade para realizar tarefas.

Quadrirotores atualmente são amplamente utilizados para diversos tipos de serviços, tais como aquisição de imagens aéreas, monitoramento de atividades e levantamento de dados de locais de difícil acesso ou que trazem perigo para o ser humano, como ilustra a Figura 1.1.



Figura 1.1: Ilustração de projeto no qual quadrirotores é utilizado para entrar em prédios abandonados, em perigo de desmoronamento. (Fonte: Projeto *DARPA's new Fast Lightweight Autonomy (FLA) program*.²)

É interessante que, em locais desconhecidos, o quadrirotores seja capaz de se localizar para cumprir seu objetivo com sucesso. No caso de voos autônomos, essa capacidade de localização é imprescindível. Um sistema de localização e mapeamento, que permite o avanço em outras atividades de veículos aéreos autônomos, deve ser capaz de:

- identificar movimentos realizados pelo veículo, utilizando sensores, e estimar a sua localização e atitude, isto é, estimar uma posição relativa à estimativa anterior e a orientação do veículo com relação a todos os eixos de rotação;
- identificar pontos característicos de interesse no ambiente e estimar suas posições em um mapa.

Realizar as duas tarefas simultaneamente pode se tornar uma tarefa complicada, tendo em vista as incertezas envolvidas nos sistemas de medição. Faz-se o uso, nessas situações, de ferramentas

²<http://www.foxnews.com/tech/2014/12/31/drones-to-zoom-into-buildings-and-hunt-inside.html>

de filtragem estocástica, como o filtro de Kalman, para que seja possível melhorar as estimativas.

Tendo em vista que quase todos os quadrirotorros possuem câmeras capazes de transmitir imagens, vê-se a possibilidade de usar esse recurso para auxílio no mapeamento e localização. É possível utilizar algoritmos para identificar pontos de interesse na imagem da câmera, estimar o posicionamento deles e usar essas informações para melhorar as estimativas. Incorporar medições estimadas de pontos de interesse identificados pela câmera à medições dos sensores iniciais ao sistema de filtragem estocástica, torna o sistema de localização e mapeamento mais robusto, quando comparado a um sistema que utiliza apenas uma dessas ferramentas.

1.2 Veículos Aéreos Não Tripulados - VANTs

Os VANTs são basicamente compostos por uma estrutura mecânica, na qual estão embarcados sensores, atuadores e uma unidade de processamento, a qual oferece o poder computacional necessário para processar informações provenientes de sensores, controlar atuadores e se comunicar com um computador de base ou central de operação utilizando um módulo de controle ou comunicação. Uma representação simplificada da arquitetura de VANTs é mostrada no fluxograma da Figura 1.2.

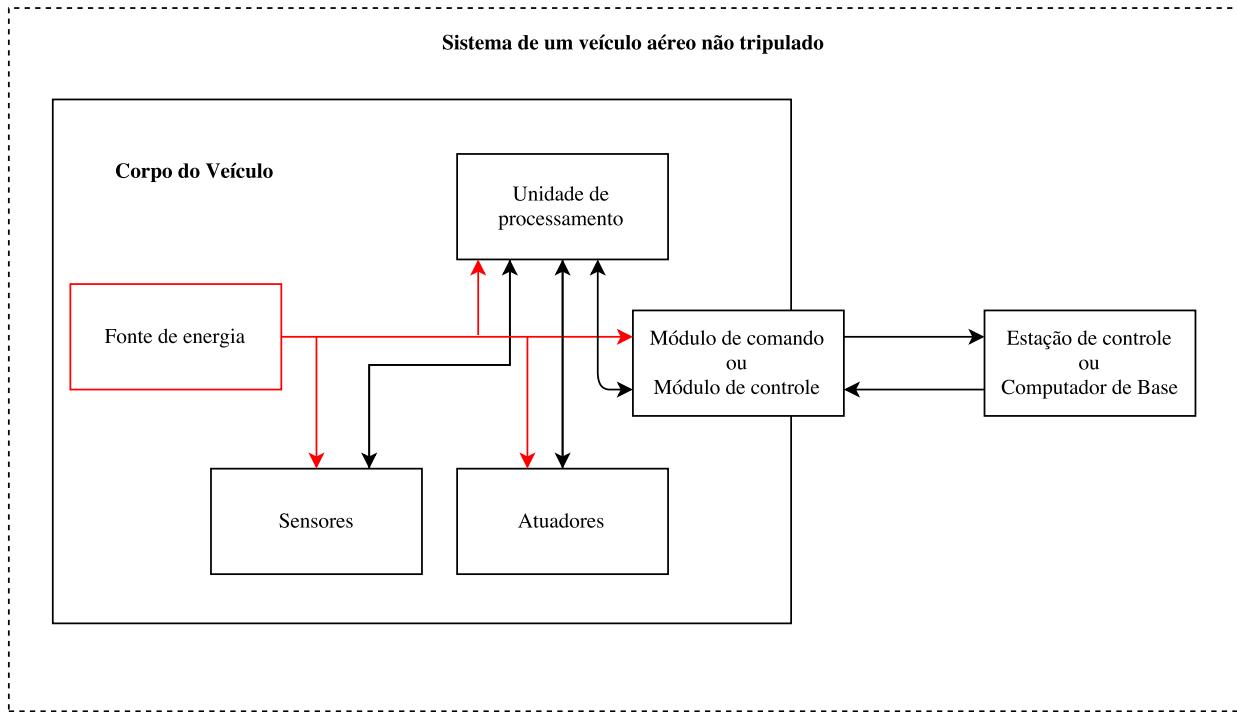


Figura 1.2: Arquitetura básica de veículos aéreos não tripulados.

Todos esses componentes são alimentados por uma fonte de energia. A fonte de energia deve ser escolhida de forma que ofereça autonomia de voo suficiente para a tarefa em questão e, ao mesmo tempo não aumente muito a massa do sistema para não dificultar a sustentação no ar.

A unidade de processamento é formada, principalmente, por microcontroladores e microproces-

sadores, podendo ser também apenas microcontrolador. É na unidade de processamento, associada a conversores analógico/digital, que os dados dos sensores podem ser interpretados.

A sustentação do veículo no ar é feita controlando-se atuadores, através de controladores embarcados. Normalmente, no caso de quadrirrotos, os atuadores são propulsores compostos de hélices conectadas a motores de corrente contínua sem escovas. Os motores de corrente contínua sem escovas têm maior vida útil, quando comparados aos motores de corrente contínua comuns, e conseguem girar em altas rotações. Os fabricantes normalmente realizam combinações de hélices com características específicas, como o passo oferecido pelas pás, e de motores, também com características específicas, de modo que ofereçam máxima economia de energia e o empuxo necessário para voo.

Para que sejam capazes de realizar controle de estabilidade, imprescindível para manterem-se no ar, os VANTs são equipados com sensores iniciais, tais como acelerômetros e giroscópio, que oferecem informações que permitem identificar a movimentação do veículo no ar e fornecer estimativa de posição e orientação.

Determinadas tarefas, como mapeamento, podem ser realizadas por um computador que não esteja embarcado no veículo. Por essa razão, os veículos aéreos, mesmo que autônomos, normalmente têm alguma comunicação sem fio com um operador ou uma central para enviar os dados e receber comandos.

1.3 Localização e mapeamento

Em várias ocasiões, sistemas autônomos precisam se localizar e mapear o ambiente em que se encontram. Existem várias ferramentas que são usadas para permitir a localização de robôs. Essas ferramentas são escolhidas para o projeto de acordo com a tarefa do dispositivo, escolha que é baseada na precisão necessária, no ambiente em que ele se movimenta, no custo do equipamento e nos custos operacionais.

Para ambientes externos, por exemplo, o sistema de posicionamento global (GPS) é muito utilizado. Esse sistema recebe informações de satélites que orbitam na Terra para que se localize. Nesse caso, é estimada a posição de um equipamento com tecnologia GPS com relação à Terra. Por necessitar manter-se conectado à satélites, o GPS não é utilizado em ambientes internos, onde os sinais de comunicação não são suficientes para determinar a localização.

Em ambientes internos, por exemplo, podem ser usados sensores de medição inercial para estimar a posição do veículo. Sensores de medição inercial possuem, assim como quaisquer outros sistemas de medição, uma incerteza atribuída a eles. Por serem utilizados para estimar posicionamento relativo a uma posição inicial, os erros dos sensores são integrados de maneira que o erro da estimativa de posição aumenta significativamente com o aumento do número de medições. Nesse caso, a precisão da estimativa nem sempre é boa o suficiente para que o robô exerça a sua atividade de forma segura. Por esse motivo, podem ser usados outros sensores ou dispositivos que aumentam a precisão da localização.

O mapeamento do ambiente é feito com base na posição relativa estimada de um ponto característico do ambiente. Para isso, podem ser usados outros sensores e dispositivos que identificam objetos e pontos no ambiente, como uma câmera. A posição dos objetos, ou dos pontos de interesse do ambiente, deve ser representada no mapa.

O Google Street View, recurso utilizado pela empresa Google para disponibilizar imagens panorâmicas aos usuários do Google Maps, usa dispositivos diferentes de localização e mapeamento em ambientes externos e internos. O veículo da Figura 1.3(a) é utilizado em ruas e armazena imagens relacionadas à localização em que ele se encontrava.



(a) Veículo utilizado para localização e mapeamento em ambientes externos.
(b) Veículo utilizado para localização e mapeamento em ambientes internos.

Figura 1.3: Veículos utilizados para mapeamento.(Fonte: Google³)

O dispositivo da Figura 1.3(b) percorre ambientes como museus e prédios históricos para disponibilizar uma vista panorâmica interna desses locais.

O uso de apenas um sensor para estimativa de posição normalmente não é suficiente. É necessário aumentar a precisão dessa estimativa de alguma maneira.

Nesse trabalho serão feitos voos em ambientes internos e semi-externos. Dadas todas as incertezas envolvidas e considerando-se um ambiente dinâmico, os dados devem ser tratados por uma ferramenta de filtragem estocástica. Para aumentar a precisão da estimativa de posição por sensores inerciais, utilizam-se imagens da câmera do veículo.

1.4 Descrição do problema

Um dos principais desafios da robótica móvel é a navegação e localização simultâneos de robôs em ambientes totalmente desconhecidos. Robôs devem ser capazes de se localizar no ambiente para que possam então tomar decisões e planejar rotas. É interessante, durante a navegação, guardar dados do ambiente em um mapa, para que depois esses dados sejam usados tanto para auxiliar na

³<https://www.google.com/maps/streetview/understand/>

própria localização, quanto para auxiliar no planejamento de rotas ou qualquer que seja o objetivo do veículo.

O problema de localização e mapeamento normalmente é tratado com o uso de diferentes sensores embarcados no veículo. Existem vários métodos para tratar o problema, dentre eles o método que utiliza sensores que podem estimar posições relativas, tais como sensores ultrassônicos, LIDAR (*Light Detection And Ranging*) ou até câmeras.

Dispõe-se, para esse trabalho, de um quadriroto AR.Drone Parrot 2.0 para implementação de um sistema de localização e mapeamento simultâneos. Trata-se de um veículo aéreo que é operado remotamente, dotado de sensores de medição inercial e câmeras.

A associação de leituras de imagens da câmera com os sensores embarcados no quadriroto torna possível a aplicação de um modelo de Localização e Mapeamento Simultâneos (SLAM) Visual Monocular. Os dados dos sensores do veículo, para esse trabalho, são processados por um computador não embarcado no dispositivo, que recebe informações de navegação e imagens da câmera por comunicação sem fio (Wi-Fi) para fazer todo o processamento.

1.5 Objetivos do projeto

O objetivo principal desse trabalho de graduação é fazer o uso da metodologia do SLAM Visual Monocular em um robô aéreo para que ele seja capaz de se localizar em ambientes internos e mapeá-los. Entende-se por localizar-se no ambiente como sendo indicar posição do veículo com relação à origem do voo e a sua atitude. Para isso, faz-se o uso de dados de uma unidade de medição inercial, composta por acelerômetro, giroscópio e magnetômetros de três eixos cada, e imagens de uma câmera frontal. O sistema deve ser capaz de estimar a posição do veículo em ambientes dinâmicos e gerar um mapa de pontos de interesse identificados pela câmera do equipamento, fazendo o uso do filtro de Kalman estendido como ferramenta de filtragem estocástica.

Tendo em vista a necessidade do veículo aéreo reconhecer o ambiente em que a sua tarefa será realizada, propõe-se nesse trabalho uma forma de localização e mapeamento simultâneos que seja possível de ser aplicado em diferentes tipos de quadrirotos. O mapa gerado deve incluir o caminho do robô durante a sua movimentação e a posição dos pontos de interesse do ambiente.

Espera-se que o resultado desse trabalho possa ser utilizado em outras pesquisas dentro do Laboratório de Automação e Robótica da Universidade de Brasília (LARA), bem como em outras instituições de ensino.

1.6 Resultados obtidos

Com o trabalho desenvolvido, foi possível obter informações de posicionamento do veículo e de pontos de interesse do ambiente durante um voo controlado remotamente. São detectados pontos de interesse que estão próximos ao veículo durante o voo, de maneira que o sistema é aplicável a veículos aéreos capazes de realizar voos de preferência em ambientes fechados.

É possível utilizar dados do resultado do SLAM Visual aplicado desde que sejam tratados *offline*. O sistema desenvolvido não funciona de *online*, de maneira que impossibilita por hora voos autônomos.

1.7 Apresentação do manuscrito

Este trabalho está organizado em cinco capítulos. No Capítulo 2 é apresentada a revisão da literatura existente sobre modelos de SLAM, especialmente os visuais monoculares. Ainda no Capítulo 2, é apresentado o veículo utilizado para aplicação do SLAM, o AR.Drone 2.0, faz-se o embasamento teórico e apresentam-se os algoritmos que são importantes para o entendimento do trabalho. No Capítulo 3 é apresentada a modelagem matemática desenvolvida durante o trabalho, descrevem-se as etapas e a maneira como o SLAM foi implementado. No Capítulo 4 estão os resultados obtidos em diferentes voos do quadrirotor em alguns ambientes. O Capítulo 5 contém as conclusões desse trabalho e propostas para trabalhos futuros utilizando o SLAM Visual aqui apresentado. Em anexo, encontram-se a planta de um ambiente onde foi realizado um voo teste, além uma descrição do CD que contém a versão digital do trabalho.

Capítulo 2

Fundamentos

Esse capítulo tem como objetivo mostrar toda a fundamentação teórica na qual o trabalho foi baseado. É feita a revisão bibliográfica da literatura já existente a cerca do assunto, a descrição do *hardware* utilizado, a descrição das ferramentas e dos algoritmos SLAM Visual Monocular implementado.

2.1 Revisão bibliográfica

Esta seção tem como objetivo introduzir e discutir as abordagens encontradas na literatura para o problema de localização e mapeamento de robôs em ambientes desconhecidos.

Localização e mapeamento simultâneos, tratado como SLAM, de *Simultaneous Localization and Mapping*, incorpora dois módulos para tratar da localização e do mapeamento e são fortemente dependentes um do outro. O módulo de localização tem como objetivo “prover uma estimativa da postura do robô referenciada a um mapa de ambiente” [2]. O outro módulo promove atualizações no mapa do ambiente, que está suscetível a mudanças. Isso é feito através de informações do ambiente, adquiridas a partir sensores, associadas ao sistema de coordenadas do robô usando a estimativa de localização.

Como já foi dito no capítulo anterior, o objetivo desse trabalho é implementar um SLAM Visual Monocular em um quadrírotor. Vê-se que algumas pesquisas já trataram desse assunto, por isso serão mostrados algumas das quais se assemelham ao modelo proposto por esse trabalho.

Analizando a literatura recente sobre localização e mapeamento de robôs com sensores de baixo custo, como câmeras monoculares e sensores iniciais, percebe-se a popularidade no uso de filtros recursivos que auxiliam na fusão sensorial. Dentre os filtros estudados, o Filtro Kalman Estendido, ou FKE, se destaca mais na área da robótica, apresentando uma abordagem estocástica, considerando modelos não lineares para descrever o sistema.

No problema do SLAM, a criação e atualização de um mapa é de grande importância. Foi apresentado por Smith *et al.* [3, 4] a representação do mapa estocástico. Esses trabalhos levaram em consideração a correlação em problemas gerais de localização e mapeamento de robôs com

um único vetor de estados e matriz de covariância, que são atualizados por um Filtro de Kalman Estendido. Essa é a mesma base para o SLAM apresentado nesse trabalho.

A importância de manter correlações estimadas e utilizar a abordagem do FKE para o problema de SLAM foi provada na tese de P. M. Newman [5].

A. J. Davison *et al.* [6] aborda a aplicação de um SLAM Monocular em tempo real, utilizando mapa probabilístico 3D que é matematicamente representado por um vetor de estados e matriz de covariância. Assim como nesse trabalho, A. J. Davison *et al.* também utiliza pontos de referência naturais do ambiente.

A inserção de um ponto de interesse no mapa necessita de uma atualização tanto do vetor de estados quanto da matriz de covariância. O método de atualização da matriz de covariância para inserção de pontos de interesse é realizado através do método de *Bootstrap*. Esse método, introduzido por Efron [7], realiza uma re-amostragem para estimação de distribuições estatísticas baseadas em observações independentes. Neste trabalho, o *Bootstrap* é utilizado para estimar a matriz de covariância do erro de estimação do novo ponto de interesse a ser inserido no mapa.

Por se tratar de SLAM Visual, é necessário o processamento de imagens. Para isso, são feitas parametrizações de pontos de interesse dentro do mapeamento e localização simultâneos de maneira eficiente e precisa. Os trabalhos [8, 9] mostram como são resolvidos os problemas de projeção de pontos de interesse a partir de imagens de câmera monocular para representação no mapa.

A extração de pontos de interesse da imagem desse trabalho é baseada no algoritmo ORB. Esse algoritmo é abordado em [10], onde a sua eficiência é comprovada. O uso desse algoritmo para aplicação de SLAM é feito também em [11].

A método de verificação de casamento entre dois pontos de interesse deste trabalho é baseado na distância de Mahalanobis. Esse método foi abordado inicialmente por P. C. Mahalanobis em 1936 em [12]. A distância de Mahalanobis mede a distância entre um ponto e uma distribuição, levando em consideração a matriz de covariância dos elementos considerados.

A aplicação desse trabalho é voltada principalmente para o quadriroto AR.Drone Parrot, que será apresentado ainda nesse capítulo, contudo as técnicas são válidas para VANTs similares. É feito um detalhamento do hardware utilizado e como suas informações são geradas em [1]. A leitura desse artigo justifica o uso de informações já processadas da unidade de medição inercial provenientes do processador embarcado do quadriroto.

Dijkshoorn [13] também mostra a aplicação de SLAM no AR.Drone Parrot, e detalha a inteligência embarcada nesse veículo para estimativa de estado a partir do modelo aerodinâmico e de odometria visual. Para o trabalho de Dijkshoorn utilizam-se imagens provenientes da câmera vertical inferior para identificação de pontos de interesse do ambiente e estimação de localização do veículo.

Pela literatura, percebe-se que os ângulos de Euler são amplamente utilizados na representação da orientação de robôs no espaço. A representação de movimentação de corpos rígidos escolhida é a matriz de transformação homogênea usando ângulos de Euler para parametrização da matriz de rotação.

Analizando a literatura existente que engloba o assunto de SLAM, percebe-se este ainda não é um problema completamente resolvido. As diversas técnicas existentes estão sendo constantemente aprimoradas. Ainda assim, em muito casos, principalmente em casos de SLAM Visual, o sistema é executado *offline*. Através da literatura, foram escolhidos os métodos e técnicas mais adequados para implementar o SLAM Visual considerando os equipamentos disponíveis. Percebe-se que o Filtro de Kalman Estendido é uma escolha coerente para fusão sensorial de sistema não lineares, como é o caso deste trabalho. Além disso dentre os diversos métodos de detecção de pontos de interesse, o algoritmo ORB apresenta um resultado mais consistente que o SIFT e o FAST, apresentando um desempenho maior, além de ser de código aberto. Para casamentos de pontos de interesse caracterizados apenas por suas posições no espaço e suas incertezas, percebe-se pela literatura que a distância de Mahalanobis apresenta um resultado satisfatório para o problema.

2.2 AR.Drone Parrot 2.0

Primeiramente, é necessário levantar informações do *hardware* disponível no veículo quadrirroto AR.Drone Parrot 2.0 (Figura 2.1).



Figura 2.1: AR.Drone Parrot 2.0.

As especificações técnicas dos sensores embarcados no quadrirroto que será utilizado para o desenvolvimento do trabalho são listadas abaixo:

- Câmera frontal HD (720p 30fps);
- Processador ARM Cortex A8 1GHz 32 bit com 800MHz video DSP TMS320DMC64x;
- Linux 2.6.32;
- 1Gbit DDR2 RAM na frequência de 200MHz;
- Wi-Fi b,g,n;
- Giroscópio de 3 eixos com precisão de 2000°/s;
- Acelerômetro de 3 eixos com precisão de +-50mG;
- Magnetômetro de 3 eixos;

- Sensor de pressão com precisão de +/- 10 Pa;
- Sensor Ultrassônico para medição de baixas altitudes;
- Câmera QVGA de 60fps vertical voltada para o solo com resolução de 320x240.

O veículo dispõe de dispositivo capaz de criar uma rede sem fio (Wi-Fi) para comunicação com um computador externo. De acordo com o fabricante, o alcance máximo desse sinal é de 50m a 100m, variando de acordo com as condições do ambiente. É necessário conectar o computador de base à essa rede para obter os dados dos sensores embarcados.

A câmera frontal permite aquisição de imagens com resolução suficiente para extração de pontos de interesse do ambiente. O sensor giroscópio, com precisão de 2000°/segundo, fornece a variação da orientação do veículo em relação aos três eixos de rotação. O acelerômetro fornece a aceleração do veículo nos três eixos de translação. Esses dados são utilizados para obter informação de velocidade linear do veículo em movimento. O magnetômetro auxilia na obtenção da orientação do veículo com relação ao mundo. O sensor de pressão será utilizado pelo veículo em elevadas altitudes. Ele calcula a pressão atmosférica, logo é possível determinar a altitude sabendo-se a pressão de referência do solo. Esse sensor não possui precisão para determinar pequenas altitudes, nesse caso é utilizado o sensor ultrassônico, que mede a distância com relação aos obstáculos abaixo do veículo. No caso de um voo em solo regular, é possível dizer que a altitude relativa é a medida dessa distância.

As dimensões do veículo utilizando diferentes configurações de voo, que são *Indoor Hull* e *Outdoor Hull*, são mostradas na Figura 2.2. Para todos os voos, nesse trabalho, foi utilizada a configuração *Indoor Hull*, ou seja, o veículo com a proteção de hélices. Nessa configuração, o veículo fica aproximadamente 40g mais pesado, mas oferece maior proteção durante o voo.

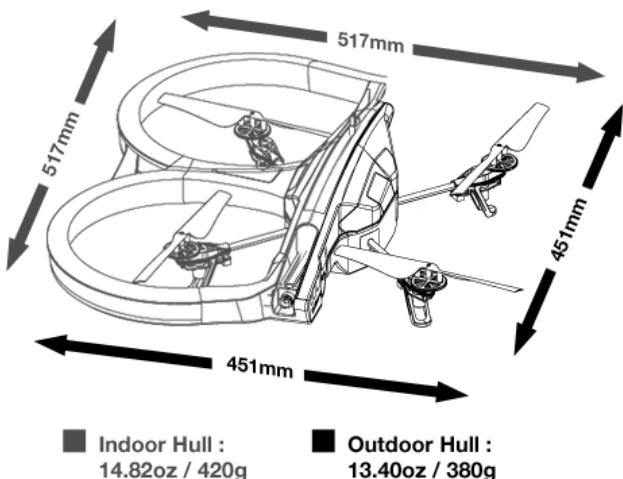


Figura 2.2: Dimensões do veículo.

2.2.1 Processamento embarcado

O quadrirotor utilizado possui um sistema de processamento, graças ao processador embarcado ARM Cortex A8. Esse processamento embarcado permite maior facilidade do controle remoto do veículo, com funções de decolagem, pouso, manutenção de posição e controle de altitude. Além disso, o processamento interno já oferece estimativa do estado do veículo, como atitude e velocidade.

A estimativa de estado feita pelo veículo utiliza os sensores citados anteriormente com algoritmos de fusão sensorial. É visto em [1, 13] que o processamento interno leva em consideração o modelo aerodinâmico do veículo e um algoritmo de odometria visual utilizando a câmera voltada para o solo. O gráfico da Figura 2.3 mostra o resultado da estimativa de velocidade utilizando a fusão sensorial feita pelo processamento embarcado do AR.Drone, comparado com as estimativas usando cada uma das técnicas utilizadas separadamente. Percebe-se que o resultado da estimativa de velocidade apenas pelo sistema de odometria visual apresenta incoerências, e o sistema de estimativa de velocidade utilizando a UMI também mostra muitos ruídos e erros. Por se tratar de estimativa de posição, os erros dos sensores da UMI seriam integrados e gerariam significativa diferença nos resultados.

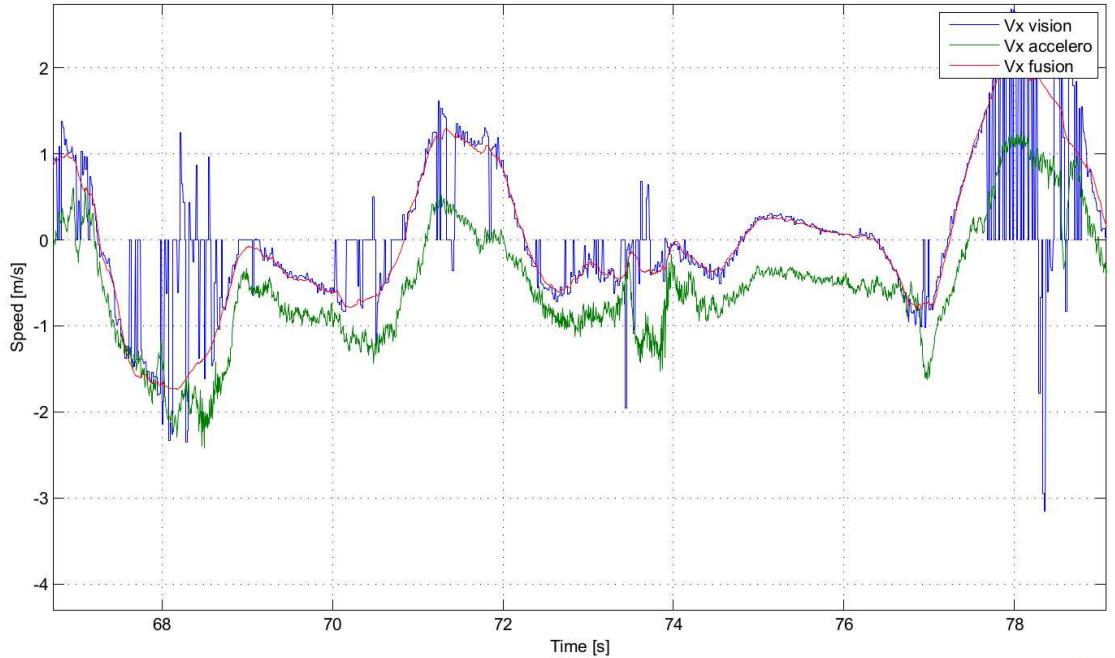


Figura 2.3: Estimativa de velocidade linear no eixo x provenientes do processamento embarcado: Estimativa a partir da odometria visual (azul), estimativa da velocidade do modelo aerodinâmico a partir de leituras diretas dos sensores de medição inercial (verde), estimativa após fusão sensorial (vermelho). (Fonte: [1])

Visto que o resultado da fusão sensorial do processamento embarcado já fornece uma estimativa robusta de velocidade, foi utilizada a informação de velocidade proveniente do AR.Drone para os cálculos de estimativa de posição do veículo.

Os algoritmos de estimativa de velocidade baseados na visão acima citados são particularmente

eficientes quando o solo é suficientemente texturizado [1]. Portanto, os voos para testes desse trabalho serão feitos levando-se em conta esse processamento embarcado feito pelo veículo e, para que esses dados sejam confiáveis, os voos são feitos em superfícies planas e texturizadas.

2.3 Ferramentas computacionais

Foram utilizadas algumas ferramentas para aquisição de dados e simulação. Como o trabalho é feito em ambiente Linux, por oferecerem boas ferramentas para visualização e boas ferramentas para integração de diferentes tarefas durante a execução dos códigos, foram escolhidas as plataformas ROS (*Robot Operating System*) e RViz (*ROS Visualization*). Essas duas plataformas permitem que sejam adquiridos os dados do *hardware* do veículo e, após tratamento, sejam visualizados os resultados em ambiente 3D.

2.3.1 ROS

Robot Operating System (ROS) é uma plataforma utilizada para simplificar a tarefa de criação de softwares para robôs. É uma coleção de estruturas de software (bibliotecas, ferramentas e *drivers*) que permitem a transmissão de mensagens, gerenciamento de pacotes, abstração de *hardware* e outras facilidades. O principal objetivo do ROS é permitir que softwares de interação e controle de robôs sejam desenvolvidos de forma mais robusta em uma plataforma comum.

Foi escolhido o ROS pela facilidade em trabalhar com o AR.Drone, devido a trabalhos publicados também nessa plataforma e por ela fornecer ambiente para fácil gerenciamento de pacotes e transmissão de mensagens entre o *hardware* utilizado e o computador de base. Dessa forma, o foco desse trabalho pode ser no *software* de mais alto nível e no estudo de algoritmos para a implementação do SLAM visual.

O ROS cria uma rede *peer-to-peer* (ou ponto-a-ponto) de processos que estão comunicando e compartilhando dados entre si. É necessário entender alguns conceitos básicos utilizados pelo ROS para entender como essa rede funciona, que são explicadas a seguir.

Nó

Um nó, no ambiente ROS, é o nome dado aos processos. Um nó é capaz de ler informações vindas de outros nós e também publicá-las para que sejam tratadas por outros nós. Nesse trabalho, é executado um nó principal, que é responsável pelos cálculos necessários e pela comunicação com o *hardware* e com o *software* de visualização.

Mensagem

Os nós se comunicam uns com os outros através de mensagens. A mensagem é uma estrutura de dados que indica o tipo da informação que será transmitida. São suportados vários tipos de

padrões como inteiro, ponto flutuante, vetores e *booleano*. As mensagens podem incluir estruturas arbitrárias aninhadas e matrizes. O tipo de mensagem garante a compatibilidade das informações interpretadas por diferentes nós da rede.

Tópico

Um tópico é o nome dado para identificar onde os nós publicam o conteúdo das mensagens. Um nó envia informações publicando-as em um tópico (*publish*), ou lê informações desse tópico ao se inscrever nele (*subscribe*). Os tópicos no ROS podem ser lidos também pelo software RViz, que será descrito nesse capítulo. Múltiplas mensagens podem ser publicadas em um mesmo tópico por vários nós diferentes. As mensagens de um mesmo tópico também podem ser adquiridas por diferentes nós.

Usa-se a barra “/” antes do nome para indicar que é um tópico. Por exemplo, */nome_do_tópico*.

Bag

Um *bag* é o formato de um arquivo onde são salvas as informações publicadas em tópicos do ROS. *Bags* gravados podem ser novamente executados com informações de tempo de aquisição de cada dado. É uma ferramenta importante para que sejam salvos os dados e futuramente utilizados novamente pelos nós.

2.3.2 RViz

RViz (ROS *Vizualization*) é uma ferramenta de visualização 3D para o ROS. Essa plataforma permite que sejam visualizados marcadores em um ambiente 3D de acordo com tópicos do ROS. Os nós devem publicar mensagens em tópicos de forma que sejam interpretadas pelo RViz, que por sua vez posiciona os marcadores. O posicionamento dos marcadores em ambiente 3D permite a visualização do mapa que será formado pelo SLAM.

A referência que indica altitude nula, ou seja, o solo, é um plano quadriculado, como indicado na Figura 2.4.

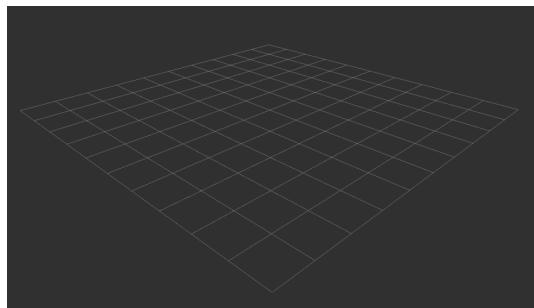


Figura 2.4: Plano quadriculado em ambiente RViz.

Neste trabalho, serão usados vetores de marcadores (*MarkerArray*) para indicar localização e orientação do veículo e as posições dos pontos de interesse no mapa. Esses vetores de marcadores são publicados pelo nó em tópicos que serão lidos pelo RViz.

Para localização do veículo, são adicionadas a posição e a orientação em relação ao mundo ao vetor correspondente à localização a cada nova informação dos dados do veículo. Esse vetor também contém a informação de cor e forma como será representado o marcador, que no caso de localização do veículo é uma seta (Figura 2.5).

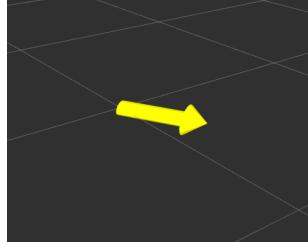


Figura 2.5: Representação de localização e orientação do veículo.

Para representação dos pontos de interesse no mapa são usados cubos (Figura 2.6), visto que não é necessário indicar sua orientação. As informações de posição 3D e cor são adicionadas ou alteradas no vetor correspondente aos pontos de interesse.

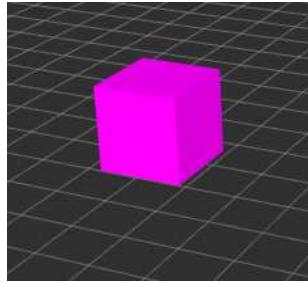


Figura 2.6: Representação de pontos de interesse.

Também será usada a ferramenta de visualização do sistema de coordenadas, chamada “*Axes*”, posicionada no centro do plano para facilitar a indicação da origem da referência. Nessa representação, a cor azul significa o eixo z , a cor verde representa o eixo y e o vermelho o eixo x .

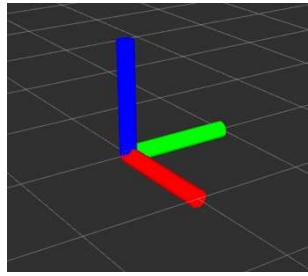


Figura 2.7: Representação da origem no plano.

2.4 Modelagem Matemática

Esta seção tem como objetivo descrever toda a modelagem matemática que foi necessária para o desenvolvimento do SLAM Visual Monocular.

2.4.1 Sistemas de Coordenadas

Neste trabalho, é utilizado um sistema de câmera monocular, constituído de uma câmera que fica fixa na parte frontal do quadrirrotor. Ambas as partes movem-se em conjunto, tornando a transformação de coordenadas entre o quadrirrotor e a câmera um parâmetro fixo do sistema. O sistema de coordenada do quadrirrotor, da câmera e da imagem são representados por \mathcal{O}^r , \mathcal{O}^c e \mathcal{O}^i respectivamente na Figura 2.8.



Figura 2.8: Representação dos sistemas de coordenadas

2.4.2 Sistema de Orientação

Para este trabalho, os ângulos de Euler foram utilizados como o sistema de orientação 3D do quadrirrotor. Utilizou-se a convenção $x - y - z$, em que a rotação sobre esses eixos representam os ângulos de $roll(\phi)$, $pitch(\theta)$ e $yaw(\psi)$, respectivamente. O sistema de orientação adotado pode ser observado na Figura 2.9.

Um dos maiores problemas encontrados no uso dos ângulos de Euler é o *Gimbal Lock*. Esse fenômeno ocorre no momento que o ângulo de *pitch* alcança o ângulo de 90° , causando, assim, uma ambiguidade entre os ângulos de *yaw* e *roll* e, consequentemente, a perda de um grau de liberdade. Entretanto, sabe-se que essa situação nunca será alcançada pelo quadrirrotor na prática. Por esse motivo e por ser de fácil interpretação, escolheu-se os ângulos de Euler como o sistema de orientação.

A matriz de rotação baseada na convenção *roll-pitch-yaw* é descrita por

$$\mathbf{R} = \begin{bmatrix} \cos(\psi)\cos(\theta) & -\sin(\psi)\cos(\phi) + \cos(\psi)\sin(\theta)\sin(\phi) & \sin(\psi)\sin(\phi) + \cos(\psi)\sin(\theta)\cos(\phi) \\ \sin(\psi)\cos(\theta) & \cos(\psi)\cos(\phi) + \sin(\psi)\sin(\theta)\sin(\phi) & -\cos(\psi)\sin(\phi) + \sin(\psi)\sin(\theta)\cos(\phi) \\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix}. \quad (2.1)$$

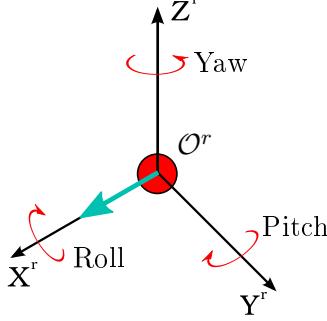


Figura 2.9: Representação do sistema de orientação *roll-pitch-yaw*

2.4.3 Transformação de Coordenadas

Neste trabalho, as posições e orientações do quadrirotor, da câmera e dos pontos de interesse são definidos e atualizados através das matrizes de transformação homogêneas. As definições descritas nesta subseção estão de acordo com [14]. As matrizes de transformação homogêneas, representadas por

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{cw} & \mathbf{p} \\ 0 & 1 \end{bmatrix}, \quad (2.2)$$

são definidas pela concatenação da matriz de rotação da câmera c em relação à origem do voo w (mundo) \mathbf{R}_{cw} , e o vetor de posição 3D \mathbf{p} , em que \mathbf{R}_{cw} é definido conforme a Eq. (2.1) e \mathbf{p} é definido como $(p_x \ p_y \ p_z)^T \in \mathbb{R}^3$.

A transformação de coordenadas de um estado para o outro, representada pela Figura 2.10, é realizada através do produto de duas matrizes de transformação homogêneas, em que a primeira representa a posição e orientação do estado anterior e a segunda representa a posição e orientação do novo estado do quadrirotor. A equação que representa esse cálculo é descrita por

$$\mathbf{T}_{2,0} = \mathbf{T}_{1,0} \cdot \mathbf{T}_{2,1}. \quad (2.3)$$

Um outro caso, representado pela Figura 2.11, é realizado um pouco diferente. Como nessa situação, a posição de um ponto de interesse em relação ao quadrirotor é desejada, deve-se calcular a matriz de transformação homogênea inversa

$$\mathbf{T}_{cw}^{-1} = \begin{bmatrix} \mathbf{R}_{cw}^T & -\mathbf{R}_{cw}^T \cdot \mathbf{p}_{cw} \\ 0 & 1 \end{bmatrix} \quad (2.4)$$

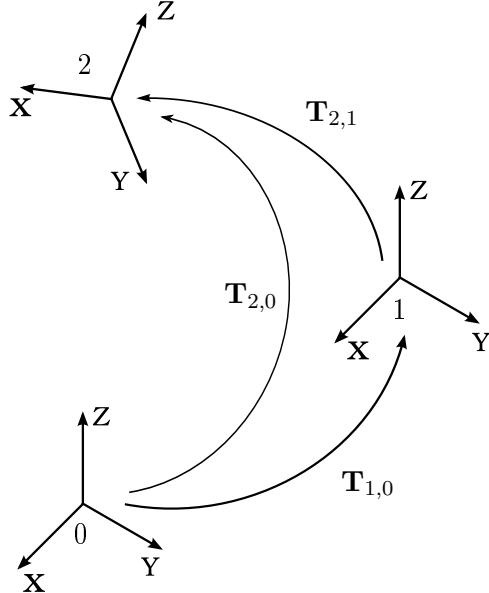


Figura 2.10: Representação da transformação de coordenadas usando matrizes de transformação homogêneas

para realizar a transformação. As equações que representam a transformação desse caso é descrito por

$$\mathbf{T}_{fc} = \mathbf{T}_{cw}^{-1} \cdot \mathbf{T}_{fw} = \begin{bmatrix} \mathbf{R}_{fc} & \mathbf{p}_{fc} \\ 0 & 1 \end{bmatrix} \quad (2.5)$$

e

$$\mathbf{p}_{fc} = \begin{bmatrix} z_x \\ z_y \\ z_z \end{bmatrix}. \quad (2.6)$$

A partir do resultado da Eq. (2.5), é possível concluir que o elemento \mathbf{p}_{fc} é um vetor de posição que representa a distância relativa do ponto de interesse em relação ao quadrirrotor.

2.4.4 Algoritmo ORB para Detecção de Pontos de interesse

Os pontos de interesse são padrões específicos e únicos que os tornam fáceis de serem detectados e comparados em uma imagem. Eles podem ser pontos, objetos específicos, vértices ou arestas. O ORB (*Oriented FAST and Rotated BRIEF*) é um algoritmo de detecção rápida e robusta de pontos de interesses locais, abordado inicialmente por Ethan Rublee[10]. Um exemplo de detecção pode ser observado na Figura 2.12. A detecção é baseada na fusão do algoritmo de detecção FAST [15] e descritores visuais BRIEF [16], ambas com modificações para deixar o desempenho de detecção melhor.

Primeiramente, o algorítimo FAST (*Features from Accelerated Segment Test*) é usado para

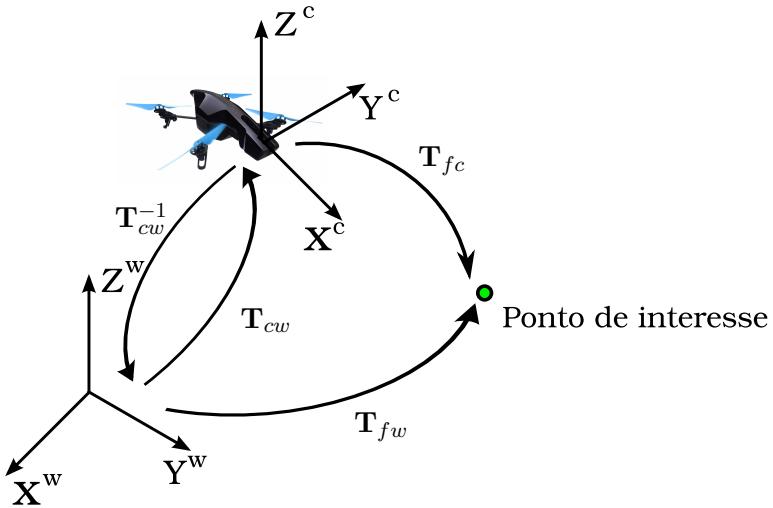


Figura 2.11: Representação da transformação de coordenada para encontrar a posição relativa do ponto de interesse em relação ao quadrirroto utilizando matriz de transformação homogênea inversa

detectar pontos de interesse na imagem e em seguida, é aplicado um algoritmo de detecção de cantos Harris [17] para selecionar os pontos de maior confiança. O FAST, entretanto, não apresenta reconhecimento de orientação dos pontos de interesse, ou seja, ele não é invariante à rotação. O algoritmo ORB trata esse problema, computando a intensidade da centroide ponderada da área onde o canto de encontra. A direção do vetor da posição do canto até a centroide retorna, assim, a orientação. O diferencial do algoritmo ORB é o fato de ser invariante à rotação, resistente à ruídos e rápida performance.

O BRIEF (*Binary Robust Independent Elementary Features*) é um descritor de pontos de interesse que utiliza testes binários simples entre os *pixels* em áreas pequenas. O descritor utilizado possui um desempenho parecido com o SIFT [18], incluindo a robustez à luminosidade, borrão e distorção de perspectiva. Porém, o BRIEF possui um desempenho ruim quanto à rotação. O algoritmo ORB corrige esse problema guiando o BRIEF de acordo com a orientação do ponto de interesse.

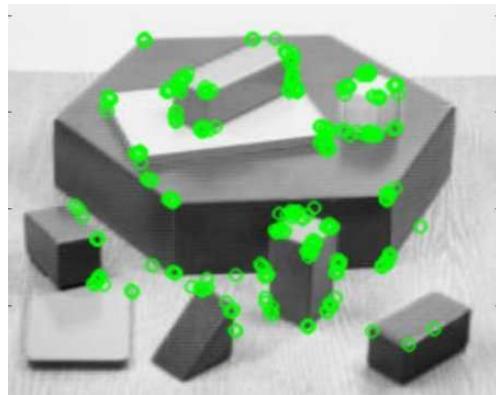


Figura 2.12: Pontos de interesse detectados utilizando algoritmo ORB

2.4.5 Projeção de Imagens

A projeção de imagem é aplicada, neste trabalho, em duas situações distintas: projeção de um ponto de interesse do espaço 3D para o espaço 2D projetado sobre o plano imagem, e projeção de um ponto de interesse no espaço 2D no plano da imagem projetado para o espaço 3D. As equações definidas nesta seção foram retiradas do livro *Computer Vision, A modern approach* dos autores Forsith e Ponce [19].

Em ambas as situações, utiliza-se a matriz de calibração da câmera \mathbf{M} que contém os seus parâmetros intrínsecos, definido por

$$\mathbf{M} = \begin{pmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{pmatrix}. \quad (2.7)$$

em que f_x e f_y são as distâncias focais da câmera e C_x e C_y representam a posição central da imagem.

Projeção do espaço 3D para o 2D

A partir dos parâmetros intrínsecos da matriz representada pela Eq. (2.7), é possível determinar a projeção de um ponto no espaço 3D no plano 2D da imagem da câmera. Os cálculos necessários para esta projeção são descritos por

$$\begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{M} \cdot \frac{\mathbf{p}_{fc}}{z_z} \quad (2.8)$$

em que u e v representam os pontos projetados no plano 2D da câmera. Logo,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} C_x + f_x \cdot \frac{z_x}{z_z} \\ C_y + f_y \cdot \frac{z_y}{z_z} \end{bmatrix}, \quad (2.9)$$

em que $\mathbf{p}_{fc} = \begin{bmatrix} z_x \\ z_y \\ z_z \end{bmatrix}$ e representa a posição relativa do ponto em relação à câmera.

Projeção do espaço 2D para o 3D

Para a projeção do espaço 2D para o 3D, é necessário obter alguma estimativa de profundidade z_z do ponto projetado. Neste trabalho, a estimativa de profundidade é descrita com mais detalhes na Seção 2.4.6 onde a projeção é utilizada. A projeção 3D do ponto pode ser facilmente encontrada invertendo a Eq. (2.9) e isolando as variáveis z_x e z_y , sendo descrita por

$$\mathbf{p}_{fc} = \begin{bmatrix} z_x \\ z_y \\ z_z \end{bmatrix} = \begin{bmatrix} \frac{z_z \cdot (u - C_x)}{f_x} \\ \frac{z_z \cdot (v - C_y)}{f_y} \\ z_z \end{bmatrix}. \quad (2.10)$$

Entretanto, a vetor \mathbf{p}_{fc} encontrado descreve a posição relativa do ponto de interesse em relação ao quadrirotor. Para encontrar a posição do ponto de interesse em relação ao mundo, deve realizar transformações de coordenadas usando o método descrito pela Eq. (2.3). Como o ponto de interesse não possui uma orientação definida, considera-se que a sua matriz de rotação em relação ao quadrirotor é igual a uma matriz identidade \mathbb{I} de dimensão 3×3 . Assim, a equação que descreve a transformada de coordenadas da posição relativa do ponto de interesse em relação ao quadrirotor para uma posição em relação ao mundo é descrita por

$$\mathbf{p}_{fw} = \begin{bmatrix} \mathbf{R}_{cw} & \mathbf{p}_{cw} \\ 0 & 1 \end{bmatrix} \cdot \mathbf{p}_{fc}, \quad (2.11)$$

em que \mathbf{R}_{cw} e \mathbf{p}_{cw} representam a matriz de rotação e a posição no espaço 3D do quadrirotor em relação ao mundo.

2.4.6 Bootstrap

O *Bootstrap* é um método de re-amostragem para estimação de distribuições baseadas em observações independentes. O método é bastante eficiente em diversas situações, sendo uma alternativa até melhor que métodos assintóticos como a tradicional aproximação normal e a expansão de Edgeworth[20].

Neste trabalho, o *Bootstrap* é utilizado para estimar a matriz de incerteza de um ponto de interesse a ser inserido no mapa baseado nas medições das posições dos pontos. Esta matriz é posteriormente inserida na matriz de covariância do erro de estimativa do sistema. A equação que descreve o cálculo de estimativa do *Bootstrap* é apresentada por

$$P' = \frac{1}{N-2} \cdot \sum_{m=1}^N (\mathbf{p}_m - \bar{\mathbf{p}}) \cdot (\mathbf{p}_m - \bar{\mathbf{p}})^T, \quad (2.12)$$

em que $\bar{\mathbf{p}}$ representa uma posição de um ponto no espaço 3D dado por

$$\bar{\mathbf{p}} = \frac{1}{N} \sum_{m=1}^N \mathbf{p}_m. \quad (2.13)$$

O vetor $\begin{pmatrix} u \\ v \end{pmatrix}_m$, descrito por

$$\begin{pmatrix} u \\ v \end{pmatrix}_m \sim \mathcal{N}\left(\mu, \begin{pmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_v^2 \end{pmatrix}\right), \quad (2.14)$$

representa um elemento de posição 2D (neste trabalho, a posição representa a posição do *pixel* do ponto de interesse na imagem) de uma distribuição normal gaussiana, discretizada com N amostras, realizada com os parâmetros u e v e com média μ e covariância $\begin{pmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_v^2 \end{pmatrix}$. Para cada elemento da distribuição, é realizada um projeção da posição no espaço 2D para o espaço 3D, representado por $\mathbf{p}_m = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}_m$. O cálculo realizado para a projeção e obtenção de \mathbf{p}_m é descrito conforme a Eq. (2.10), em que o elemento z_z é estimado o variando entre zero e uma distância máxima estipulada amostrada N vezes. A distância máxima é definida escolhendo um valor máximo de distância que o ponto de interesse pode ser projetado do quadrirotor. Em seguida, com o intuito de se obter a posição dos pontos \mathbf{p}_m em relação ao mundo, é aplicada uma transformação de coordenadas, conforme a Eq. (2.11), para cada elemento da distribuição. O vetor $\bar{\mathbf{p}}$, descrito pela Eq. (2.13), representa a média aritmética das posições dos pontos da distribuição normal realizada conforme a Eq. (2.14).

2.4.7 Distância de Mahalanobis

A distância de Mahalanobis é um método de medição de distância entre um ponto e uma distribuição de algum elemento. O método foi introduzido por P. C. Mahalanobis pela primeira vez em 1936, através de sua publicação [12]. Neste trabalho, a distância de Mahalanobis foi utilizada para verificar se há casamento entre o ponto de interesse observado com os pontos de interesse contidos no mapa.

As equações que descrevem a medição são descritas por

$$d_{ij}^2 = \mu_{ij}^T \cdot \mathbf{P}_{ij}^{-1} \cdot \mu_{ij}, \quad (2.15)$$

em que a matriz de covariância \mathbf{P}_{ij} que caracteriza a incerteza dos pontos de interesse é dada por

$$\mathbf{P}_{ij} = \frac{\partial h_c}{\partial x} |_{x=\hat{\mathbf{x}}_{k-1}} \cdot \left(\mathbf{P}_{\mathbf{k}|\mathbf{k}-1} \cdot \frac{\partial h_c}{\partial x} |_{x=\hat{\mathbf{x}}_{k-1}} \right)^T + \mathbf{R} \quad (2.16)$$

e a diferença em *pixels* de um ponto observado para um ponto projetado no plano da imagem μ_{ij} é dada por

$$\mu_{ij} = \begin{bmatrix} u \\ v \end{bmatrix} - h_c(\hat{\mathbf{x}}_{k|k-1}, \mathbf{M}). \quad (2.17)$$

A função $h_c(\hat{\mathbf{x}}_{k|k-1}, \mathbf{M})$ representa a projeção do ponto de interesse no espaço 3D para o 2D, em que \mathbf{M} é a matriz que contém os parâmetros intrínsecos da câmera descrito pela Eq. (2.7) e \mathbf{R} é um ruído branco.

2.4.8 Representação gráfica de incertezas na imagem

A representação gráfica das incertezas do ponto presente no mapa é feita utilizando uma elipse (Figura 2.13). Essa elipse é formada a partir das informações da matriz de covariância do erro de estimativa de posição dos pontos já presentes mapa.

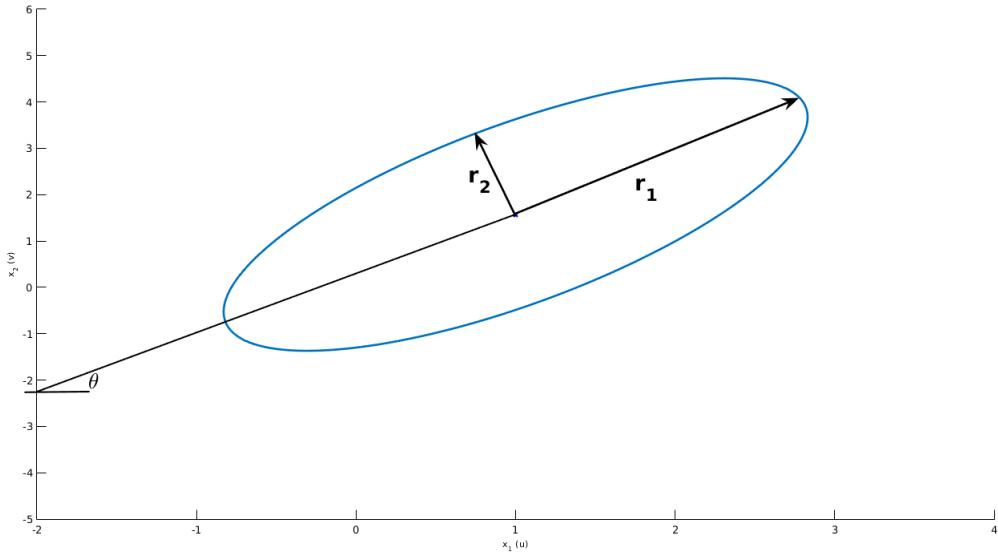


Figura 2.13: Elipse para representação gráfica de incertezas.

Dado uma distribuição conjunta X , sua matriz de covariância $P_x(X)$ é descrita por

$$P_x(X) = \begin{bmatrix} \sigma_{X_1} & \text{cov}\{X_1, X_2\} \\ \text{cov}\{X_1, X_2\} & \sigma_{X_2} \end{bmatrix}. \quad (2.18)$$

Tem-se que o centro da elipse é dado pelo valor esperado $E\{X\}$. O ângulo com relação ao eixo X_1 é dado por

$$\theta = \frac{1}{2} \arctan \left(\frac{2\rho\sigma_{X_1} - \sigma_{X_2}}{\sigma_{X_1}^2 - \sigma_{X_2}^2} \right),$$

em que ρ é o coeficiente de correlação, dado por

$$\rho = \frac{\text{cov}\{X_1, X_2\}}{\sigma_{X_1}\sigma_{X_2}}.$$

Os eixos principais, r_1 (raio menor) e r_2 (raio maior), são respectivamente,

$$r_1^2 = \frac{c\sigma_{X_1}^2\sigma_{X_2}^2(1-\rho^2)}{\sigma_{X_2}^2\cos^2(\theta) - 2\rho\sigma_{X_1}\sigma_{X_2}\sin(\theta)\cos(\theta) + \sigma_{X_1}^2\sin^2(\theta)}$$

e

$$r_2^2 = \frac{c\sigma_{X_1}^2\sigma_{X_2}^2(1-\rho^2)}{\sigma_{X_2}^2\sin^2(\theta) - 2\rho\sigma_{X_1}\sigma_{X_2}\sin(\theta)\cos(\theta) + \sigma_{X_2}^2\cos^2(\theta)}.$$

A elipse obtida é comumente chamada de “elipse de incerteza \sqrt{c} -sigma”, dado $c = \ln\left(\frac{1}{(1-p^2)}\right)$. Isso significa que, para $c = 9$ tem-se uma elipse 3-sigma (ou 3σ) que incorpora 98,8891% da distribuição de X . A elipse 3-sigma é comumente utilizada para representar a incerteza total associada a uma variável.

A simulação apresentada na Figura 2.13 mostra uma distribuição gerada da simulação de Monte Carlo, a partir da qual obtém-se uma matriz de covariância e representa-se a elipse 3-sigma de incerteza associada à essa variável aleatória.

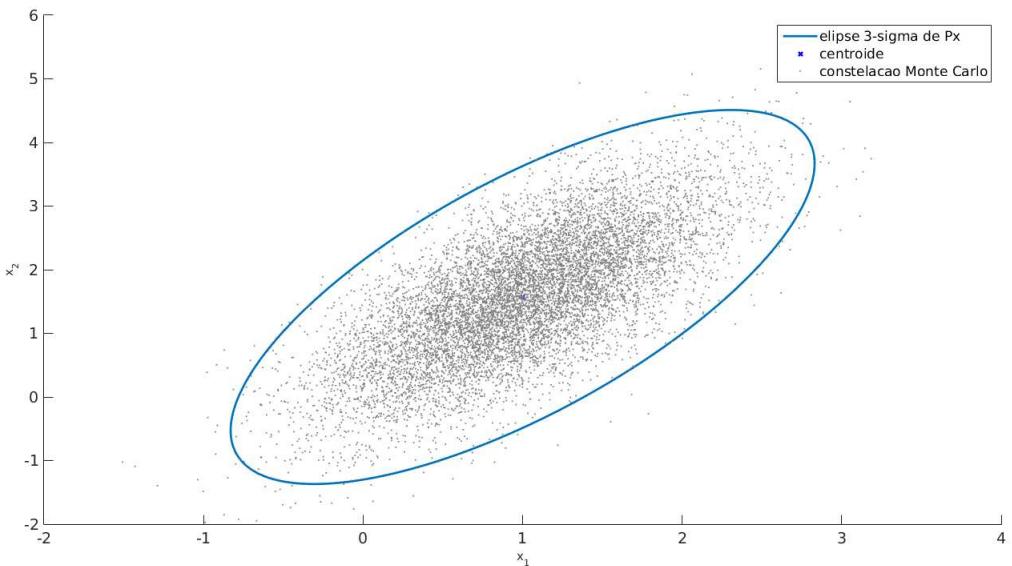


Figura 2.14: Elipse 3-sigma de uma distribuição de amostras a partir de uma simulação de Monte Carlo.

2.4.9 Filtro de Kalman

O filtro de Kalman, ou FK, é um método de estimação Bayesiana recursivo de estados de um sistema dinâmico linear a partir de medições de observações ao longo do tempo, cuja evolução segue um modelo incerto com ruídos estatísticos. A teoria do filtro foi criado e implementado por Rudolf Kalman em 1960 em [21]. O método possui diversas aplicações na área de tecnologia, sendo aplicadas em sistemas de navegação, orientação e controle de veículos.

O modelo do sistema dinâmico linear em tempo discreto é descrito pelas equações

$$\mathbf{x}_k = \mathbf{F}_{k-1} \cdot \mathbf{x}_{k-1} + \mathbf{G}_k \cdot \mathbf{u}_k + \mathbf{w}_k \quad (2.19)$$

e

$$\mathbf{y}_k = \mathbf{H}_k \cdot \mathbf{x}_k + \mathbf{v}_k. \quad (2.20)$$

em que a primeira descreve o modelo de processo e a segunda representa o modelo de medição. Os vetores \mathbf{x} , \mathbf{u} e \mathbf{y} representam o vetor de estados do sistema, o vetor de controle e o vetor de medição. As matrizes lineares \mathbf{F} , \mathbf{G} e \mathbf{H} representam o modelo de transição de estados, o modelo das entradas de controle e o modelo de observação, respectivamente. Os elementos \mathbf{w} e \mathbf{v} representam ruídos de processo e de observação, para os quais assume-se que são ruídos brancos gaussianos de média nula.

O estado do filtro de Kalman é representado por duas variáveis. A primeira é a estimativa do vetor de estados $\hat{\mathbf{x}}_{k|k}$ para o próximo estado sabendo a estimativa do estado atual. A segunda é a matriz de covariância do erro de estimação $\mathbf{P}_{k|k}$ do próximo estado, sabendo a estimativa atual.

O método de filtragem consiste em duas etapas principais: a predição e a correção. A etapa de predição utiliza a estimativa do estado anterior corrigido para realizar a estimativa do estado atual baseado na dinâmica do sistema. A etapa de correção realiza a predição com a ajuda das observações atuais medidas para corrigir e melhorar a estimativa do estado atual. As equações que descrevem a etapa de predição são representadas por (2.21) e (2.22) e a etapa de correção é descrita pelas Eq. (2.23)-(2.25), em que \mathbf{K} representa o ganho ótimo de Kalman, responsável por gerar as estimativas de mínimo erro quadrático.

Etapa de Predição

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_{k-1} \cdot \hat{\mathbf{x}}_{k-1} + \mathbf{G}_k \cdot \mathbf{u}_k \quad (2.21)$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1} \cdot \mathbf{P}_{k-1} \cdot \mathbf{F}_{k-1}^T + \mathbf{Q}_k \quad (2.22)$$

Etapa de Correção

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \cdot \mathbf{H}_k^T \cdot (\mathbf{H}_k \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \quad (2.23)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot (\mathbf{y}_k - \mathbf{H}_k \cdot \hat{\mathbf{x}}_{k|k-1}) \quad (2.24)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{H}_k) \cdot \mathbf{P}_{k|k-1} \cdot (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{H}_k)^T + \mathbf{K}_k \cdot \mathbf{R}_k \cdot \mathbf{K}_k^T \quad (2.25)$$

2.4.10 Filtro de Kalman Estendido

O método do filtro de Kalman se baseia em modelos lineares para realizar tanto a predição como as medições das observações e a correção do estado. No entanto, no mundo real os modelos lineares não satisfazem tão bem as dinâmicas dos sistemas, sendo a maioria não-lineares. O filtro de Kalman Estendido, ou FKE, é uma versão modificada do filtro de Kalman que se baseia em funções não-lineares para realizar a filtragem. A ideia do FKE é aplicar uma linearização de primeira ordem do modelo não-linear do sistema através da expansão da série de Taylor truncada, também conhecido como série de Taylor de primeira ordem.

As modificações realizadas em relação ao filtro de Kalman estão relacionadas com a função que determina a estimativa do estado na etapa de predição e a função que determina a medição das observações na etapa de correção. Estas duas etapas são trocadas por uma função não-linear $f(\mathbf{x}_{k-1}, \mathbf{u}_k)$ e por uma outra função não-linear $h(\mathbf{x}_k)$. Assim, o modelo do sistema é descrito por

Modelo Não-Linear

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \quad (2.26)$$

$$\mathbf{y}_k = h(\mathbf{x}_k) + \mathbf{v}_k. \quad (2.27)$$

Além disso, o FKE utiliza Jacobianos, representados por \mathbf{F}_x e \mathbf{H}_x , para substituir as matrizes lineares \mathbf{F} e \mathbf{H} do filtro de Kalman. Os Jacobianos relacionados às funções $f(\mathbf{x}_{k-1}, \mathbf{u}_k)$ e $h(\mathbf{x}_k)$ são representadas por

$$\mathbf{F}_x = \frac{\partial f(\mathbf{x}_{k-1}, \mathbf{u}_k)}{\partial \mathbf{x}} \Big|_{\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k}, \quad \mathbf{H}_x = \frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k} \Big|_{\hat{\mathbf{x}}_{k|k-1}}. \quad (2.28)$$

As etapas de predição e correção do FKE são similares ao FK, se diferenciando apenas no uso das jacobianas representadas pela Eq. (2.28). As equações que descrevem as etapas de predição e correção do filtro de Kalman Estendido são representadas por (2.29)-(2.33).

Etapa de Predição

$$\hat{\mathbf{x}}_{k|k-1} = f(\mathbf{x}_{k-1}, \mathbf{u}_k) \quad (2.29)$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{x_{k-1}} \cdot \mathbf{P}_{k-1} \cdot \mathbf{F}_{x_{k-1}}^T + \mathbf{Q}_k \quad (2.30)$$

Etapa de Correção

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \cdot \mathbf{H}_{x_k}^T \cdot (\mathbf{H}_{x_k} \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}_{x_k}^T + \mathbf{R}_k)^{-1} \quad (2.31)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_{x_k} \cdot (\mathbf{y}_k - \mathbf{H}_{x_k} \cdot \hat{\mathbf{x}}_{k|k-1}) \quad (2.32)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{H}_{x_k}) \cdot \mathbf{P}_{k|k-1} \cdot (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{H}_{x_k})^T + \mathbf{K}_k \cdot \mathbf{R}_k \cdot \mathbf{K}_k^T \quad (2.33)$$

Capítulo 3

Desenvolvimento

Esse capítulo descreve como foi feita a integração das ferramentas que são utilizadas no trabalho, além de mostrar como foi feita a implementação do SLAM Visual no AR.Drone Parrot 2.0, indicando como é feita a representação do mapa e como funciona a arquitetura do modelo.

São apresentados os dados que são utilizados do processamento embarcado do veículo, assim como a unidade de medida de cada um deles. É descrito ainda nesse capítulo como foi estruturado o código e as ferramentas que são aplicadas nele.

3.1 Integração do *Hardware* com ROS e RViz

Como já foi dito no Capítulo 2, o veículo faz a transmissão dos dados através de uma rede Wi-Fi para um computador de base. O computador de base, por sua vez, executa o *framework* ROS , onde são criados tópicos específicos com esses dados. O nó principal pode, assim, realizar o *subscriber* dos tópicos para processá-los.

A comunicação do AR.Drone com o ROS é possível graças ao projeto *ardrone_autonomy*¹. Esse projeto contém o *driver* do AR.Drone Parrot 2.0 para o ROS, necessário para que sejam publicados os tópicos com as informações do veículo.

3.1.1 *Driver ardroner autonomy*

O computador de base deve ter instalado o ROS e o *driver* *ardrone_autonomy*. Dessa forma, os tópicos com as mensagens do veículo são publicados e podem ser lidos pelo nó principal. Dentre os tópicos publicados pelo AR.Drone, são usados os tópicos */navdata* e */front/image_raw*. A mensagem publicada no tópico */navdata* contém as informações presentes na Tabela 3.1. A mensagem no tópico */front/image_raw* é do tipo *sensor_msgs/Image*, padrão do ROS.

¹<http://ardrone-autonomy.readthedocs.io/en/latest/index.html>

Tabela 3.1: Mensagem do tópico */navdata*.

Nome da informação no tópico	Descrição da informação	Unidade de medida
header	Cabeçalho da mensagem no ROS.	-
batteryPercente	Nível da bateria.	[]
state	Estado do vôo (em solo, decolando, pousando, navegando...).	-
rotX, rotY, rotZ	Rotação em torno dos eixos x, y e z, respectivamente.	[°]
magX, magY, magZ	Leitura do magnetômetro em torno dos eixos x, y e z, respectivamente	-
pressure	Pressão medida pelo barômetro	[Pa]
temp	Temperatura	-
wind_speed	Velocidade do vento estimada	-
wind_angle	Angulação do vento estimada	-
wind_comp_angle	Compensação do vento estimada	-
altd	Altitude estimada	[mm]
motor1, motor2, motor3, motor4	Valor do PWM dos motores 1 a 4	-
vx, vy, vz	Velocidade linear nos eixos x, y e z, respectivamente	[mm/s]
ax, ay, az	Aceleração linear nos eixos x, y e z, respectivamente	[g]
tm	Medição do tempo - <i>Timestamp</i>	[ms]

- Não se aplica ou não é utilizado

Nesse trabalho, utilizam-se os dados de velocidade (v_x, v_y, v_z), aceleração (a_x, a_y e a_z) e orientação (ϕ, θ, ψ) estimados pelo veículo.

3.1.2 Comunicação do RViz com o nó principal no computador de base

Um nó, nomeado como nó principal, é executado para realizar os cálculos descritos nas Seções 3.2 e 3.3 desse capítulo, a partir das informações dos tópicos gerados pelo AR.Drone no ROS. Os resultados desses cálculos geram o vetor de estados, que contém a informação de posição do veículo e dos pontos de interesse presentes no ambiente.

A posição dos pontos de interesse e do veículo devem ser adicionados no RViz para que possam ser visualizados. Para que isso seja possível, o nó principal faz a publicação das mensagens com essas informações, isto é, as mensagens que contém as posições e as formas dos marcadores que serão adicionados no ambiente 3D do RViz. Os tópicos usados para essa comunicação são */pose3D* e */features*.

O fluxograma da Figura 3.1 mostra como é feita a integração dos elementos que compõem o processo de formação do mapa 3D dos pontos de interesse e localização do veículo.

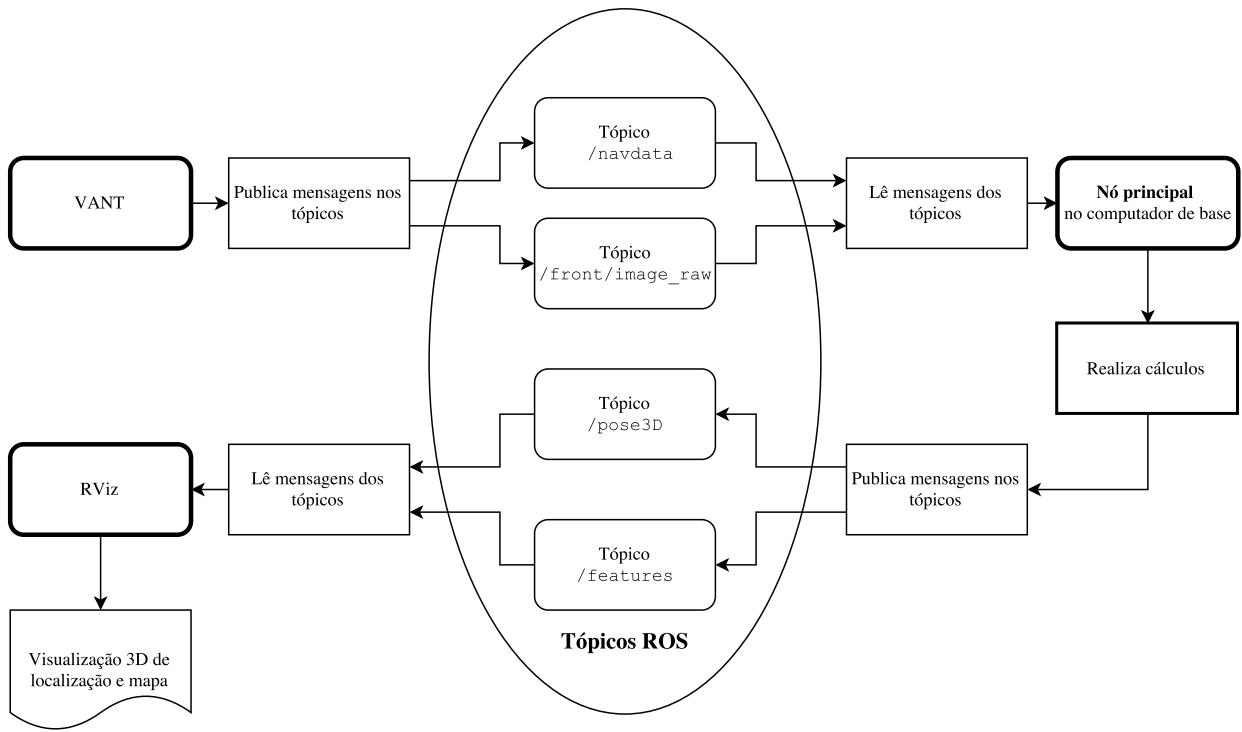


Figura 3.1: Representação da integração *Hardware*, ROS e RViz.

A integração do *Hardware* com o ROS e o RViz irá resultar no mapa visualizado no RViz, no qual os cubos representam pontos no mapa e as setas representam localização e orientação do veículo no decorrer da sua movimentação no ambiente, de acordo como descrito na Seção 2.3. A janela do RViz no computador, após a integração, é mostrada na Figura 3.2.

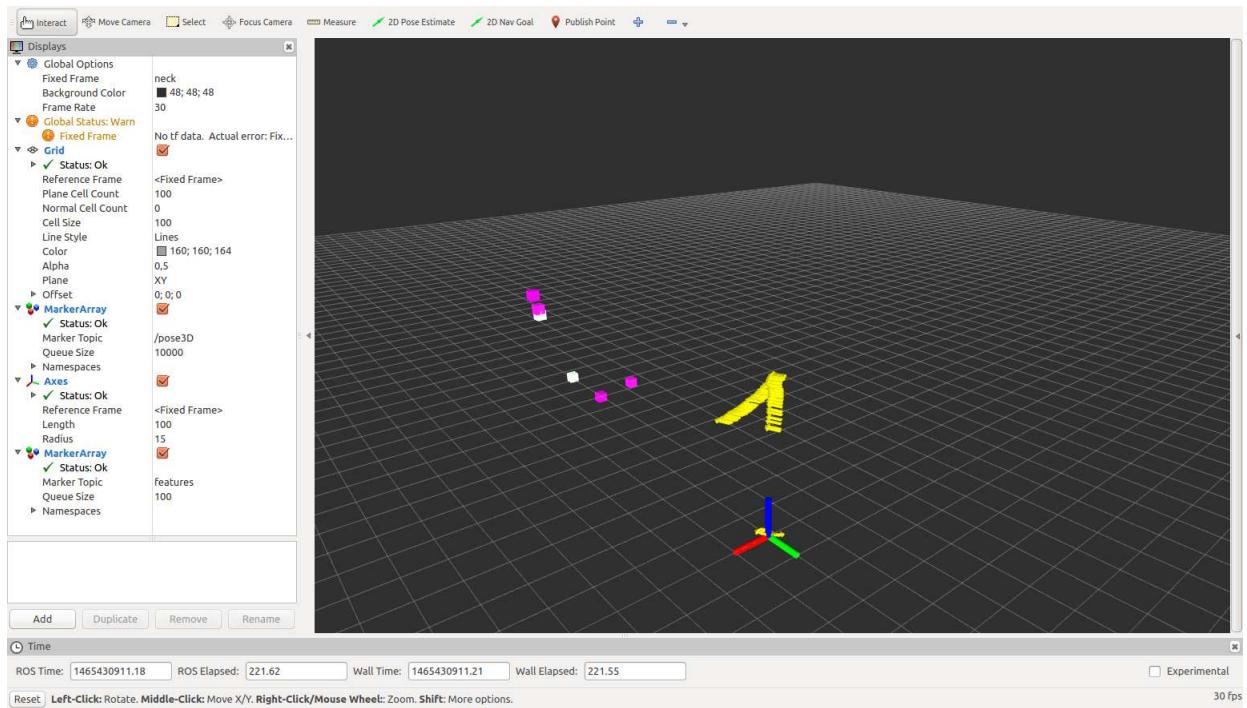


Figura 3.2: Resultado da integração do *Hardware* com o ROS e o RViz.

Entendido como é gerado o mapa, pode-se explicar agora como é representado o mapa e o funcionamento da arquitetura do SLAM Visual Monocular aplicado nesse trabalho.

3.1.3 Execução *offline*

O sistema desenvolvido nesse trabalho foi feito para execução *offline*, isto é, o sistema é executado a partir de dados colhidos em um voo real do veículo (mas não em tempo real). Os dados são obtidos e gravados a partir da função *rosbag record*, oferecida pelo ROS. Essa função grava um arquivo do tipo *bag* o qual contém gravados os tópicos escolhidos que foram publicados pelo AR.Drone durante o voo.

Os tópicos */navdata* e */front/image_raw* foram os escolhidos para todas as gravações. Esses tópicos são novamente publicados no ROS usando a função *rosbag play*, também oferecida pelo ROS.

A execução de um *bag* funciona como se o voo estivesse acontecendo em tempo real, fazendo com que as simulações representem fielmente a execução do código como se estivesse realmente sendo feito um voo do veículo.

Para realizar um voo, o veículo é operado remotamente utilizando um nó do ROS chamado *joystick_controller*², o qual envia comandos para o AR.Drone. Os comandos permitem controle de orientação (*roll*, *pitch* e *yaw*) e também de altitude. Para facilitar os comandos feitos pelo operador, mapeiam-se os botões de um controle sem fio Microsoft XBox 360 (Figura 3.3) com o

²https://github.com/mikehamer/ardrone_tutorials

intuito de enviar os comandos de controle de voo para o veículo.



Figura 3.3: Mapeamento do controle sem fio Microsoft XBox 360.

3.2 Calibração de Câmera

O uso de câmeras tem sido bastante utilizadas nos últimos tempos em dispositivos como celulares, *tablets* e *webcams*. Entretanto, essas câmeras possuem uma distorção significante, dificultando a sua aplicação em projetos de engenharia que envolvem projeção de imagens da câmera e o mundo real, ou vice e versa. Neste projeto, projeções de imagens são usadas constantemente, sendo necessário o uso de métodos de calibração e retificação das imagens vindas da câmera. Para este trabalho, utiliza-se um modelo físico simplificado de câmeras, chamado de *pinhole*. Esta seção não tem como objetivo descrever com detalhes como é feita a calibração³, mas descrever as ferramentas e como elas foram utilizadas para a calibração da câmera do AR.Drone Parrot 2.0.

Para realizar a calibração da câmera utilizou-se o *framework* ROS, que possui uma ferramenta para calibração de câmera. Para o funcionamento da calibração, é necessário realizar diversas medições de um *chessboard*, representado pela Figura 3.4, em diversas posições e orientações. A medida que as medições vão sendo realizadas, a ferramenta mostra o nível de medição de cada parâmetro de medida. A interface da ferramenta é representada pela Figura 3.5.

Após realizada a calibração, a ferramenta retorna duas matrizes, \mathbf{M} e \mathbf{M}_d , em que a primeira contém os parâmetros intrínsecos e a segunda contém os parâmetros extrínsecos da câmera. Os elementos que constituem a matriz \mathbf{M} representam a distância focal da lente da câmera (f_x e f_y) e a posição central da imagem (c_x e c_y). Já a matriz \mathbf{M}_d , contém apenas elementos que descrevem a distorção da lente. As matrizes são descritas por (3.1) e (3.2).

³Para mais detalhes: http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html#goal

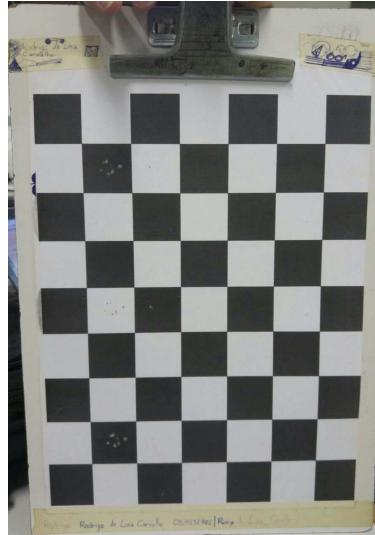


Figura 3.4: *Chessboard utilizado para a calibração da câmera*



Figura 3.5: Interface da ferramenta de calibração de câmera do ROS

$$\mathbf{M} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

$$\mathbf{M}_d = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix} \quad (3.2)$$

Utilizando uma função específica da biblioteca OpenCV⁴, é possível remover a distorção presente nas imagens, passando como entrada as matrizes de parâmetros intrínsecos e extrínsecos da câmera. É possível observar o resultado da retificação da imagem com distorção através da Figura 3.6.

Para simplificar os cálculos de projeção de imagem e de Jacobianas utilizadas no filtro de Kalman estendido, calibriou-se a imagem duas vezes. A primeira calibração é realizada apenas para

⁴Documentação da função de retificação: [http://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#void undistort\(InputArray src, OutputArray dst, InputArray cameraMatrix, InputArray distCoeffs, InputArray newCameraMatrix\)](http://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#void undistort(InputArray src, OutputArray dst, InputArray cameraMatrix, InputArray distCoeffs, InputArray newCameraMatrix))



(a) Imagem com distorção

(b) Imagem retificada

Figura 3.6: Diferença entre uma imagem com distorção e uma imagem retificada

remover as distorções presentes na imagem utilizando a função de retificação. Ao realizar a segunda distorção por cima da imagem retificada, é possível perceber que a matriz de parâmetros extrínsecos contém elementos bem próximos de zero, representando ausência de distorção na imagem. Assim, com a matriz de parâmetros intrínsecos obtido na segunda calibração e a imagem retificada pela primeira calibração, é possível realizar as projeções de imagens, descritas na Seção 2.4.5, sem considerar a distorção presente na lente da câmera.

3.3 Representação do Mapa

O mapeamento do SLAM desenvolvido neste trabalho é realizado através de um mapa 3D probabilístico baseado em pontos de interesse. O mapa é constituído de elementos que representam as estimativas de posições e incertezas dos pontos de interesse identificados no ambiente. A medida que as estimativas são realizadas, o mapa dinamicamente, sendo corrigido pelo filtro de Kalman estendido. As estimativas dos estados do veículo e dos pontos de interesse são atualizados durante a sua movimentação e observações dos pontos de interesse e dos dados de navegação.

O mapa probabilístico é representado por um vetor de estados \mathbf{x}_k e uma matriz de covariância dos erros de estimativa \mathbf{P} descritos por

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_v \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{P}_{xy_1} & \mathbf{P}_{xy_2} & \cdots \\ \mathbf{P}_{y_1x} & \mathbf{P}_{y_1y_1} & \mathbf{P}_{y_1y_2} & \cdots \\ \mathbf{P}_{y_2x} & \mathbf{P}_{y_2y_1} & \mathbf{P}_{y_2y_2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad (3.3)$$

em que $\mathbf{x}_v = (x_c \ y_c \ z_c \ \phi \ \theta \ \psi)^T$ representa a posição e orientação do veículo no espaço nos três eixos de translação, e \mathbf{y}_k representa as posições no espaço 3D dos pontos de interesse inseridos no mapa.

Atualização do mapa

O mapa probabilístico é atualizado constantemente pelo filtro de Kalman estendido, alterando os valores dos elementos contidos na Eq. (3.3). A medida que os pontos de interesse são observados, o mapa insere novos pontos e também elimina outros.

A inserção de pontos de interesse no mapa é realizada quando não há um casamento de um ponto de interesse observado na imagem com algum ponto presente no mapa. A etapa de inserção consiste na execução de dois passos: aumento do vetor de estados e o aumento da matriz de covariância. O aumento de \mathbf{x}_k é efetuado inserindo um novo elemento de \mathbf{y}_k ao final do vetor de estados através da projeção do ponto de interesse do espaço 2D para o 3D, usando as equações descritas pelas Eq. (2.10) e (2.11). Já a matriz de covariância \mathbf{P} , é aumentada através do cálculo de uma nova matriz de covariância \mathbf{P}' , utilizando o *Bootstrap* conforme a Eq. (2.12), que representa a incerteza da estimativa da posição do ponto de interesse em questão. A nova matriz calculada é, então, inserida na matriz \mathbf{P} , representado por

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{k-1} & 0 \\ 0 & \mathbf{P}' \end{bmatrix} \quad (3.4)$$

A eliminação de pontos de interesse no mapa acontece quando não há casamento de um ponto de interesse do mapa com um ponto observado, sendo que o ponto do mapa está projetado na imagem, cinquenta vezes consecutivas. Este número foi escolhido através de testes empíricos verificando-se um consistência na eliminação dos pontos de interesse. A projeção do ponto de interesse na imagem é descrito pela Eq. (2.6). A etapa de eliminação consiste na retirada da posição 3D do ponto de interesse \mathbf{y}_e do vetor de estados \mathbf{x}_k , em que e representa a numeração do ponto de interesse a ser eliminado, e a retirada das sub-matrizes de covariância que possuem elementos do ponto de interesse a ser eliminado presentes em \mathbf{P} . A equação

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{P}_{xy_1} & \cdots & \cancel{\mathbf{P}}_{xy_e} & \cdots & \mathbf{P}_{xy_n} \\ \mathbf{P}_{y_1x} & \mathbf{P}_{y_1y_1} & \cdots & \cancel{\mathbf{P}}_{y_1y_e} & \cdots & \mathbf{P}_{y_1y_n} \\ \vdots & \vdots & \ddots & / & \dots & \vdots \\ \cancel{\mathbf{P}}_{y_ex} & \cancel{\mathbf{P}}_{yey_1} & \ddots & \cancel{\mathbf{P}}_{yey_e} & \ddots & \cancel{\mathbf{P}}_{yey_n} \\ \vdots & \vdots & \vdots & / & \ddots & \vdots \\ \mathbf{P}_{y_nx} & \mathbf{P}_{y_ny_1} & \cdots & \cancel{\mathbf{P}}_{y_ny_e} & \cdots & \mathbf{P}_{y_ny_n} \end{bmatrix} \quad (3.5)$$

descreve o método de retirada das sub-matrizes de covariância do ponto de interesse da matriz de covariância do sistema.

3.4 Filtro de Kalman Estendido para SLAM

Por se tratar de um sistema não linear, este trabalho utiliza o FKE (filtro de Kalman estendido) como estrutura básica do SLAM Visual Monocular para a fusão dos dados de navegação com as imagens fornecidas pela câmera frontal do quadrirotor. São utilizados dados de velocidade linear,

aceleração linear e orientação enviados pelo controlador de navegação do próprio quadrirroto. Mais detalhes sobre o filtro de Kalman podem ser vistos no Capítulo 2.

O vetor de estados \mathbf{x}_k é composto pela posição do veículo em relação ao mundo (x_c, y_c, z_c), a orientação (ϕ, θ, ψ) e as posições dos pontos de interesses inseridos no mapa ($x_{f_1}, y_{f_1}, z_{f_1}$, ..., $x_{f_N}, y_{f_N}, z_{f_N}$). O vetor de controle \mathbf{u}_k é composto pelos dados recebidos de velocidade e aceleração linear nos três eixos do quadrirroto. O vetor de medição \mathbf{y}_k é composto pela estimativa de posição projetada nas coordenadas da câmera (u, v) em *pixels*. O vetor de medição \mathbf{z}_k é composto pelos dados de orientação recebidos do controlador de navegação ($\phi_k, \theta_k, \varphi_k$). As equações

$$\mathbf{x}_k = \begin{bmatrix} x_c \\ y_c \\ z_c \\ \phi \\ \theta \\ \psi \\ x_{f_1} \\ y_{f_1} \\ z_{f_2} \\ \vdots \\ x_{f_N} \\ y_{f_N} \\ z_{f_N} \end{bmatrix}_k , \quad (3.6)$$

$$\mathbf{u}_k = \begin{bmatrix} v_x \\ v_y \\ v_z \\ a_x \\ a_y \\ a_z \end{bmatrix}_k , \quad (3.7)$$

$$\mathbf{y}_k = \begin{bmatrix} u \\ v \end{bmatrix}_k \quad (3.8)$$

e

$$\mathbf{z}_k = \begin{bmatrix} \phi \\ \theta \\ \varphi \end{bmatrix}_k \quad (3.9)$$

ressaltam os vetores mencionados , que formam a base do filtro adotado.

3.4.1 Etapa de Predição

A etapa de predição do filtro de Kalman estendido consiste em dois passos: estimativa do vetor de estados \mathbf{x}_k e estimativa da matriz de covariância \mathbf{P}_k . O modelo que descreve os passos dessa etapa são descritos por

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) \quad (3.10)$$

e

$$\mathbf{P}_{k|k-1} = \frac{\partial f}{\partial x} |_{x=\hat{\mathbf{x}}_{k-1}} \cdot \mathbf{P}_{k-1} \cdot \frac{\partial f}{\partial x}^T |_{x=\hat{\mathbf{x}}_{k-1}} + \mathbf{Q}_v \quad (3.11)$$

em que

$$\frac{\partial f}{\partial x} |_{x=\hat{\mathbf{x}}_{k-1}} = \mathbb{I}_n \quad (3.12)$$

A estimativa do vetor de estados \mathbf{x}_k é determinada pela função $f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k)$, que é representada por

$$f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) = \begin{bmatrix} \hat{x}_{c_{k-1}} + v_{x_k} t + a_{x_k} \frac{t^2}{2} \\ \hat{y}_{c_{k-1}} + v_{y_k} t + a_{y_k} \frac{t^2}{2} \\ \hat{z}_{c_{k-1}} + v_{z_k} t + a_{z_k} \frac{t^2}{2} \\ \hat{\phi}_{k-1} \\ \hat{\theta}_{k-1} \\ \hat{\psi}_{k-1} \\ \hat{x}_{f_{1_{k-1}}} \\ \hat{y}_{f_{1_{k-1}}} \\ \hat{z}_{f_{2_{k-1}}} \\ \vdots \\ \hat{x}_{f_{N_{k-1}}} \\ \hat{y}_{f_{N_{k-1}}} \\ \hat{z}_{f_{N_{k-1}}} \end{bmatrix} \quad (3.13)$$

A função consiste em equações lineares que descrevem estimativas dos elementos de \mathbf{x}_k a partir dos valores do estado anterior \mathbf{x}_{k-1} e do vetor de controle \mathbf{u}_k .

No FKE, a estimativa do novo estado deve ser acompanhado pelo aumento de incerteza representado pela matriz de covariância \mathbf{P}_k . Esta é calculada através da incerteza do estado anterior e da Jacobiana $\frac{\partial f}{\partial x}$, conforme a Eq. (3.12), sendo \mathbf{Q}_v um ruído branco.

3.4.2 Etapa de Correção

A etapa de correção do filtro adotado ocorre em duas situações distintas. Caso o dado recebido seja proveniente da navegação do quadrirroto, a correção ocorre baseada na observação dos valores de orientação. Caso contrário, se o dado recebido for proveniente da imagem da câmera, a correção ocorre baseada na observação dos pontos de interesse detectados na imagem. A etapa de correção consiste na atualização do vetor de estados \mathbf{x}_k a partir no cálculo do ganho de Kalman. Além disso, a matriz de covariância do erro de estimativa \mathbf{P}_k também é atualizado.

Correção pela navegação

A correção baseada nos dados de navegação consiste em realizar a observação da orientação dada pelo quadrirroto \mathbf{z}_k (Eq. (3.9)) e atualizar tanto o vetor de estados \mathbf{x}_k pela equação

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{G}_k \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_{k|k-1}), \quad (3.14)$$

quanto a matriz de covariância pela equação

$$\mathbf{P}_k = (\mathbb{I} - \mathbf{G}_k \cdot \mathbf{H}) \cdot \mathbf{P}_{k|k-1} \cdot (\mathbb{I} - \mathbf{G}_k \cdot \mathbf{H})^T + \mathbf{G}_k \cdot \mathbf{R} \cdot \mathbf{G}_k^T, \quad (3.15)$$

sendo que o Jacobiano \mathbf{H} e o ganho de Kalman \mathbf{G}_k são representados por

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \quad (3.16)$$

e

$$\mathbf{G}_k = \mathbf{P}_{k|k-1} \cdot \mathbf{H}^T \cdot (\mathbf{H} \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}^T + \mathbf{R})^{-1} \quad (3.17)$$

respectivamente, em que \mathbf{R} é a matriz que representa a incerteza das medições da orientação do quadrirroto.

Correção pela câmera

A correção baseada na câmera consiste em um processo similar à correção pela navegação, porém, a observação é medida através dos pontos de interesse \mathbf{y}_k (Eq. (3.8)) detectados na imagem através do algoritmo ORB.

A função de medição $h_c(x_k, M)$, descrita por

$$h_c(\mathbf{x}_k, \mathbf{M}) = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} C_x + f_x \cdot \frac{z_x}{z_z} \\ C_y + f_y \cdot \frac{z_y}{z_z} \end{bmatrix}, \quad (3.18)$$

é calculada realizando a projeção dos pontos de interesse inseridos no mapa (coordenadas 3D) no *frame* da câmera (coordenadas de imagem) a partir dos parâmetros intrínsecos da câmera e da posição e orientação do quadrirotor. A matriz que contém os parâmetros intrínsecos \mathbf{M} é descrita por

$$\mathbf{M} = \begin{pmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{pmatrix}, \quad (3.19)$$

em que f_x e f_y são as distâncias focais e C_x e C_y representam a posição central da imagem. Já as variáveis z_x , z_y e z_z representam a distância relativa do ponto de interesse em relação ao quadrirotor. Os cálculos realizados para encontrar a distância relativa são descritos na Seção 2.4.3.

Nesta etapa, a correção é executada apenas no momento em que ocorre o casamento do ponto de interesse observado com algum ponto de interesse já inserido no mapa. Os casamentos entre os pontos de interesse foram realizados através do cálculo da distância de Mahalanobis descrito na Seção 2.4.7. As equações que descrevem essa etapa de correção são descritas por

$$\hat{\mathbf{x}}^{(m)} = \hat{\mathbf{x}}^{(m-1)} + \mathbf{G}^{(m)} \cdot \left(\mathbf{y}_c^{(m)} - h_c \left(\hat{\mathbf{x}}^{(m-1)} \right) \right) \quad (3.20)$$

e

$$\mathbf{P}^{(m)} = (\mathbb{I} - \mathbf{G}^{(m)} \cdot \mathbf{H}_c^{(m)}) \cdot \mathbf{P}^{(m-1)} \cdot (\mathbb{I} - \mathbf{G}^{(m)} \cdot \mathbf{H}_c^{(m)})^T + \mathbf{G}^{(m)} \cdot \mathbf{R} \cdot \mathbf{G}^{(m)T}, \quad (3.21)$$

em que m representa o m -ésimo casamento de pontos de interesse do mapa e que o Jacobiano $\mathbf{H}_c^{(m)}$ e o ganho de Kalman $\mathbf{G}^{(m)}$ são representados por

$$\mathbf{H}_c^{(m)} = \frac{\partial h_c}{\partial x} \Big|_{x=\hat{\mathbf{x}}^{(m-1)}} \quad (3.22)$$

e

$$\mathbf{G}^{(m)} = \mathbf{P}^{(m-1)} \cdot \mathbf{H}_c^{(m)T} \cdot \left(\mathbf{H}_c^{(m)} \cdot \mathbf{P}^{(m-1)} \cdot \mathbf{H}_c^{(m)T} + \mathbf{R}^{(m)} \right)^{-1} \quad (3.23)$$

respectivamente, em que \mathbf{R} é a matriz que representa a incerteza das medições dos pontos de interesse detectados na imagem da câmera.

A estrutura do filtro de Kalman estendido com a modelagem matemática desenvolvida para este trabalho pode ser observado pelo Algoritmo 3.1.

3.5 Arquitetura

A arquitetura do SLAM Visual Monocular é baseada em medições realizadas pela unidade de medição inercial (UMI) e pelas leituras de imagens da câmera, aplicadas a um filtro de Kalman

Algorithm 3.1 Algoritmo do filtro de Kalman estendido do SLAM Visual Monocular

Inicialização: $\hat{\mathbf{x}}_0, \mathbf{P}_0$

Parâmetros ajustáveis: \mathbf{Q}_v, \mathbf{R}

Laço Principal

Se há dados não processados , **então**

Se dados provenientes da Navegação , **então**

Etapa de Predição :

$$\begin{aligned}\hat{\mathbf{x}}_{k|k-1} &\leftarrow f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) \\ \mathbf{P}_{k|k-1} &\leftarrow \frac{\partial f}{\partial x} \Big|_{x=\hat{\mathbf{x}}_{k-1}} \cdot \mathbf{P}_{k|k-1} \cdot \frac{\partial f}{\partial x} \Big|_{x=\hat{\mathbf{x}}_{k-1}}^T + \mathbf{Q}_v\end{aligned}$$

Etapa de Correção :

$$\begin{aligned}\mathbf{G}_k &\leftarrow \mathbf{P}_{k|k-1} \cdot \mathbf{H}^T \cdot (\mathbf{H} \cdot \mathbf{P}_{k|k-1} \cdot \mathbf{H}^T + \mathbf{R})^{-1} \\ \hat{\mathbf{x}}_k &\leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{G}_k \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_{k|k-1}) \\ \mathbf{P}_k &\leftarrow (\mathbf{I} - \mathbf{G}_k \cdot \mathbf{H}) \cdot \mathbf{P}_{k|k-1} \cdot (\mathbf{I} - \mathbf{G}_k \cdot \mathbf{H})^T + \mathbf{G}_k \cdot \mathbf{R} \cdot \mathbf{G}_k^T\end{aligned}$$

Se dados provenientes da Câmera , **então**

Etapa de Correção :

$$\begin{aligned}\mathbf{G}^{(m)} &\leftarrow \mathbf{P}^{(m-1)} \cdot \mathbf{H}_c^{(m)T} \cdot \left(\mathbf{H}_c^{(m)} \cdot \mathbf{P}^{(m-1)} \cdot \mathbf{H}_c^{(m)T} + \mathbf{R}^{(m)} \right)^{-1} \\ \hat{\mathbf{x}}^{(m)} &\leftarrow \hat{\mathbf{x}}^{(m-1)} + \mathbf{G}^{(m)} \cdot \left(\mathbf{y}_c^{(m)} - h_c(\hat{\mathbf{x}}^{(m-1)}, \mathbf{M}) \right) \\ \mathbf{P}^{(m)} &\leftarrow (\mathbf{I} - \mathbf{G}^{(m)} \cdot \mathbf{H}_c^{(m)}) \cdot \mathbf{P}^{(m-1)} \cdot (\mathbf{I} - \mathbf{G}^{(m)} \cdot \mathbf{H}_c^{(m)})^T + \mathbf{G}^{(m)} \cdot \mathbf{R} \cdot \mathbf{G}^{(m)T}\end{aligned}$$

estendido (FKE) para que sejam feitas as correções necessárias. Os dados de navegação e leituras da câmera chegam ao nó no computador de base em momentos distintos. Os procedimentos são realizados de acordo com qual medição foi feita no instante k .

3.5.1 Visão geral

A arquitetura de todo o SLAM é representada pelo fluxograma da Figura 3.7. Cada etapa será descrita nas próximas subseções.

3.5.2 Procedimentos após leituras de dados de navegação

Diz-se que houve leitura de dados quando são lidas informações de velocidade de translação (v), aceleração linear (a) e orientação (ϕ, θ, ψ) do tópico `/navdata`. Nessa ocasião, é feita a etapa de predição do FKE. Isso significa que será estimada a posição do veículo de acordo com as informações dos sensores.

Após a estimativa do estado feita pelas medições dos sensores, é feita a correção da orientação baseada na observação da nova orientação (ϕ_k, θ_k, ψ_k) e um ganho G_k calculado com base na matriz de covariância P_k . Baseado também nessa observação e no ganho G_k , a matriz de covariância P é atualizada. Os cálculos que descrevem essa correção são descritos na Seção 3.3.

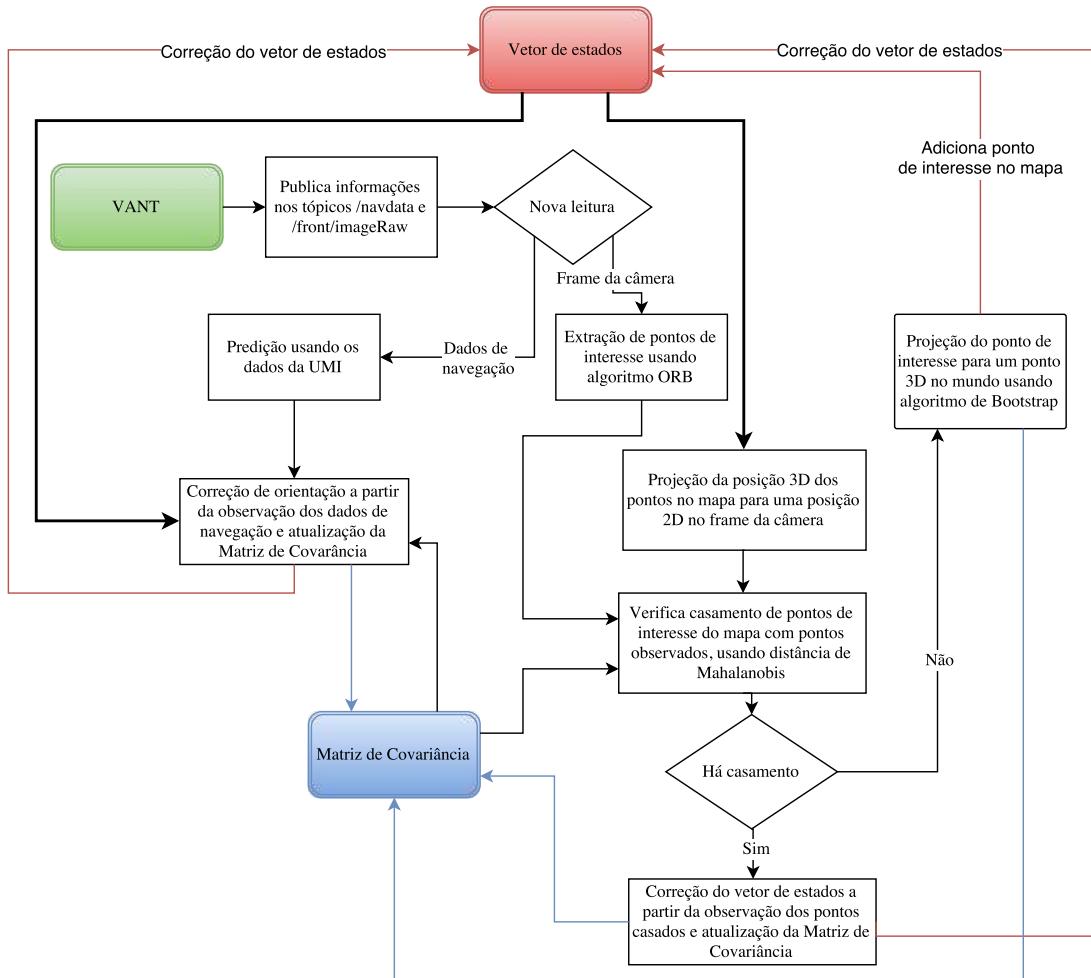


Figura 3.7: Arquitetura do SLAM Visual implementado.

3.5.3 Procedimentos após leituras da câmera

Quando há leitura de imagem da câmera frontal do veículo, pode-se então fazer correção da posição do veículo e dos pontos presentes no mapa ou adicionar novos pontos no mapa. As informações da câmera são publicadas no tópico */front/image_raw* pelo AR.Drone. Para decidir se será feita a correção ou adicionado um novo ponto, é necessário que seja feito o casamento dos pontos de interesse do mapa com os pontos de interesse observados no *frame* da câmera.

3.5.3.1 Extração de pontos de interesse do *frame* da câmera

É utilizado o algoritmo ORB para extração de pontos de interesse do *frame* da câmera. Esse algoritmo foi escolhido dentre outros testados, visto que foram obtidos pontos de interesse de maior confiabilidade.

Diz-se que os pontos de interesse extraídos têm maior confiabilidade quando eles são detectados várias vezes seguidas no decorrer da troca de *frames* do vídeo. Após extraídos esses pontos, é feito o casamento deles com os ponto que já estão presentes no mapa.

3.5.3.2 Casamento de pontos de interesse

É dito que há casamento entre os pontos de interesse observado e os que estão presentes no mapa caso a distância de Mahalanobis entre esses pontos seja menor que um valor de referência. Usando a interpretação probabilística referida no Capítulo 2, é escolhido um valor de referência que represente a verosimilhança entre os pontos.

3.5.3.3 Adição de novos pontos de interesse no mapa

Caso não haja casamento do ponto de interesse observado pela câmera, esse ponto é adicionado no mapa. É feita a projeção da posição do ponto do *frame* da câmera para uma posição no mapa de três dimensões. Essa projeção é feita baseada no algoritmo de *Bootstrap*, descrito na Seção 2.4.6. O algoritmo de *bootstrap* também atualiza a matriz de covariância P , adicionando o erro de estimativa inicial do novo ponto observado.

3.5.3.4 Correção do vetor de estados baseado no casamento de pontos de interesse

Caso haja casamento entre os pontos de interesse, a correção é feita baseada em um ganho G_k calculado a partir da matriz de covariância P_k , na observação da posição 2D dos pontos de interesse do *frame* da câmera e na posição do ponto em questão no mapa, projetado no *frame* da câmera. Esses cálculos são descritos na Seção 3.4.

3.6 Descrição do código

3.6.1 Linguagem de programação

Todo o código é feito utilizando a linguagem *Python* de programação. *Python* é uma linguagem de alto nível, interpretada e orientada a objetos. Foi utilizada essa linguagem pela facilidade em realizar operações matriciais, constantemente presente no código, através de bibliotecas específicas. Além dessa vantagem, o código em *Python* é de fácil entendimento para usuários de outras linguagens de programação orientadas a objetos, sendo possível também o seu uso no ambiente do ROS.

3.6.2 Bibliotecas importantes

Algumas bibliotecas foram utilizadas para facilitar operações e tratamentos de informações. A biblioteca OpenCV, por exemplo, foi usada para implementar o algoritmo ORB de extração de pontos de interesse da imagem. Outras informações a respeito dessa biblioteca são encontradas no site oficial⁵.

⁵<http://opencv.org>

A biblioteca roslib e rospy permitem a comunicação do código em *Python* com o ambiente ROS. Outras bibliotecas básicas, como numpy e math são usadas para fazer os cálculos matriciais e operações usando seno, cosseno e raiz quadrada. O código completo com todas as bibliotecas utilizadas encontra-se no conteúdo do CD, que é descrito no anexo desse trabalho.

3.6.3 Estrutura

O código é estruturado de acordo com o sistema apresentado na Figura 3.8.

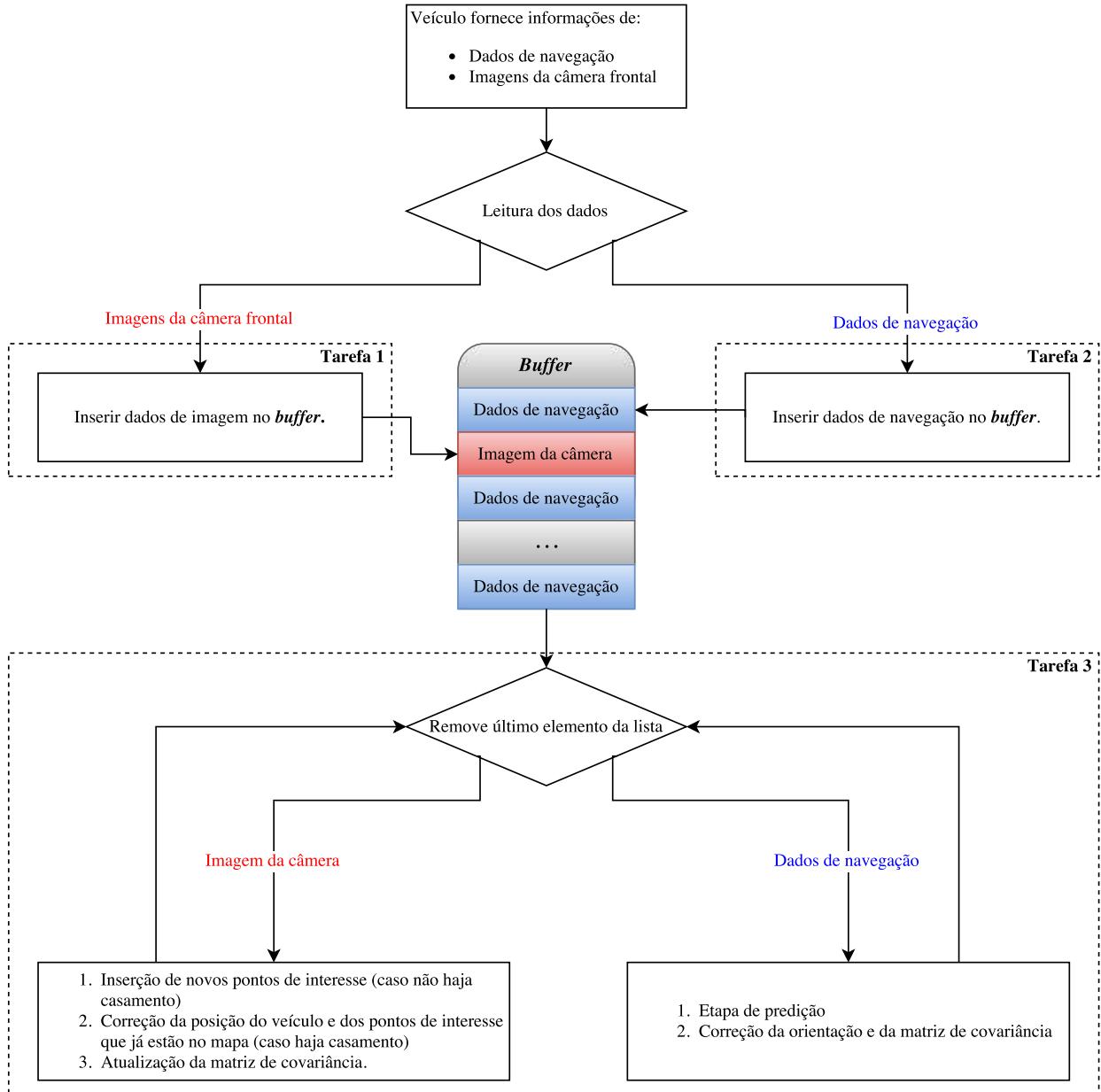


Figura 3.8: Estrutura do código.

São basicamente três tarefas executadas pelo nó no computador de base. A primeira é responsável pela leitura de imagens provenientes da câmera frontal do veículo. Essa tarefa está ligada ao tópico */front/image_raw*, já mencionado nesse capítulo. Os dados de imagem são adicionados

a um retentor (*buffer*), uma região da memória que armazena temporariamente os dados lidos do veículo enquanto eles ainda não são tratados pela Tarefa 3.

A Tarefa 2 realiza leitura dos dados de navegação do tópico */navdata*. Esses dados também são adicionados ao *buffer*.

Por fim, a Tarefa 3 remove elementos do *buffer* para realizar os procedimentos responsáveis pelo SLAM, descritos matematicamente na Seção 3.3.

Capítulo 4

Resultados

Esse capítulo apresenta diversos experimentos que ilustram a eficácia dos métodos descritos nos Capítulos 2 e 3 desse trabalho. Os experimentos mostram:

- a estimativa de posição do veículo a partir dos dados de navegação do veículo;
- a projeção da posição no espaço 3D de um ponto conhecido para uma posição no espaço 2D no *frame* da imagem;
- a projeção da posição no espaço 2D para o espaço 3D de um ponto de interesse observado;
- o casamento de com o ponto de interesse observado na imagem com um ponto conhecido e já presente no mapa através da distância de Mahalanobis;
- o casamento de com o ponto de interesse observado na imagem com um ponto conhecido e já presente no mapa através das elipses de incerteza;
- a correção da posição de pontos de interesse presentes no mapa no decorrer do tempo;
- a comparação da estimativa de posição do veículo utilizando só os dados de navegação com a posição corrigida pelo FKE.

Por fim, serão apresentados os voos feitos que mostram o resultado do sistema completo em funcionamento.

4.1 Estimativa de posição a partir dos dados de navegação

O primeiro teste feito comprova a eficácia dos métodos utilizados para transformação de coordenadas e cálculos de estimativa de posição do veículo a partir de informações de velocidade (v), aceleração (a) e rotação no eixo z (φ). Para isso, foi feito um voo com o veículo no Laboratório de Automação e Robótica (LARA) da UnB. O veículo percorreu, nesse caso, um percurso em volta das mesas centrais do laboratório, que foram aproximadamente um retângulo de 3.6m por 3m.

A Figura 4.1 mostra uma imagem do ambiente de trabalho do LARA onde foi realizado o voo. Durante o voo, foram gravados os dados de navegação que são usados para a estimativa de posição.



Figura 4.1: Ambiente do LARA onde foi feito o voo teste

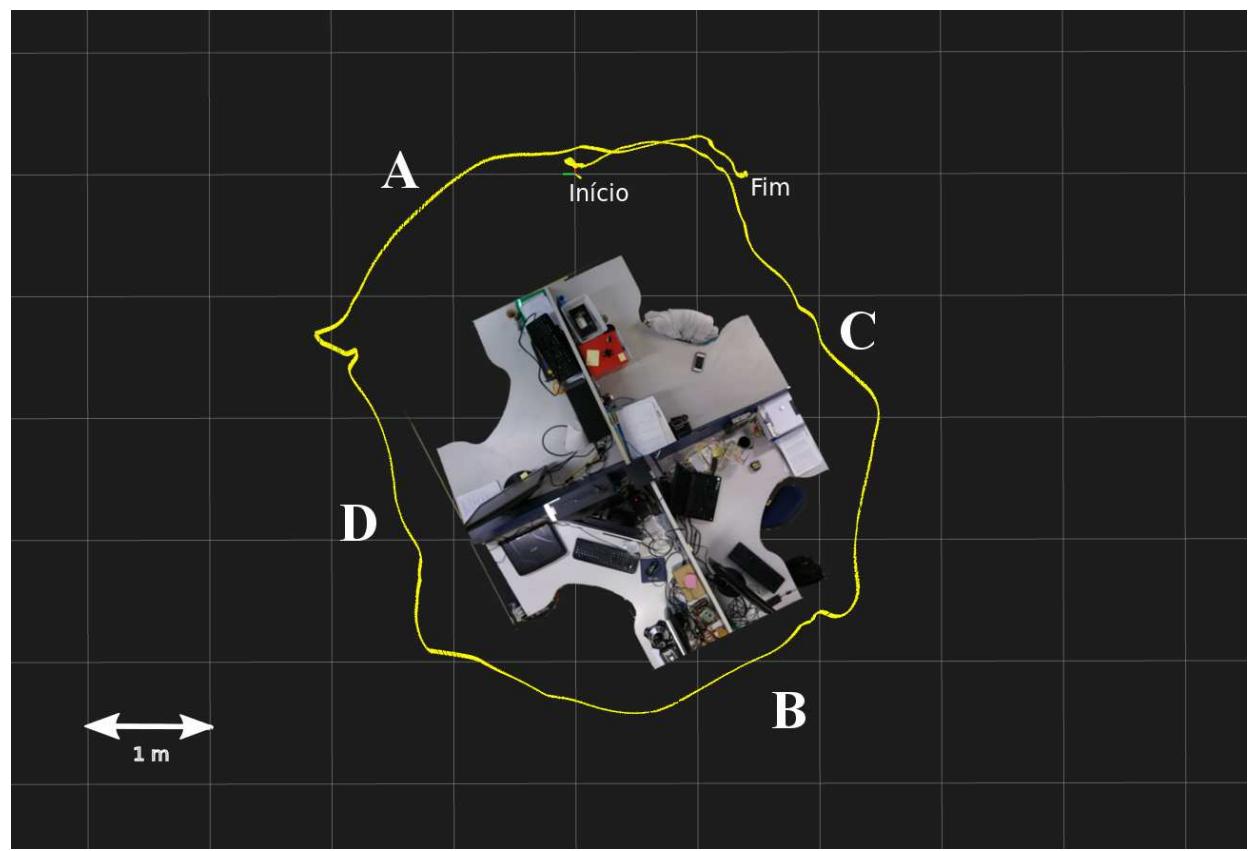


Figura 4.2: Vista superior do resultado de estimativa de posição em 2D. Teste feito em torno das mesas realizando uma volta completa e encerrando o voo um pouco à frente da posição inicial.

Pela Figura 4.2 é visto que a distância entre os pontos A e B é de aproximadamente $4.3m$, e a distância entre os pontos C e D é de aproximadamente $3.3m$. Considerando o espaço ocupado pelas mesas, constata-se que a estimativa de velocidade e aceleração provinda do AR.Drone Parrot 2.0 é confiável o suficiente para ser usada na etapa de predição do FKE.

Os gráficos das figuras 4.3 e 4.4 mostram as velocidades e acelerações lineares em dois eixos de translação para o teste apresentado nesta seção.

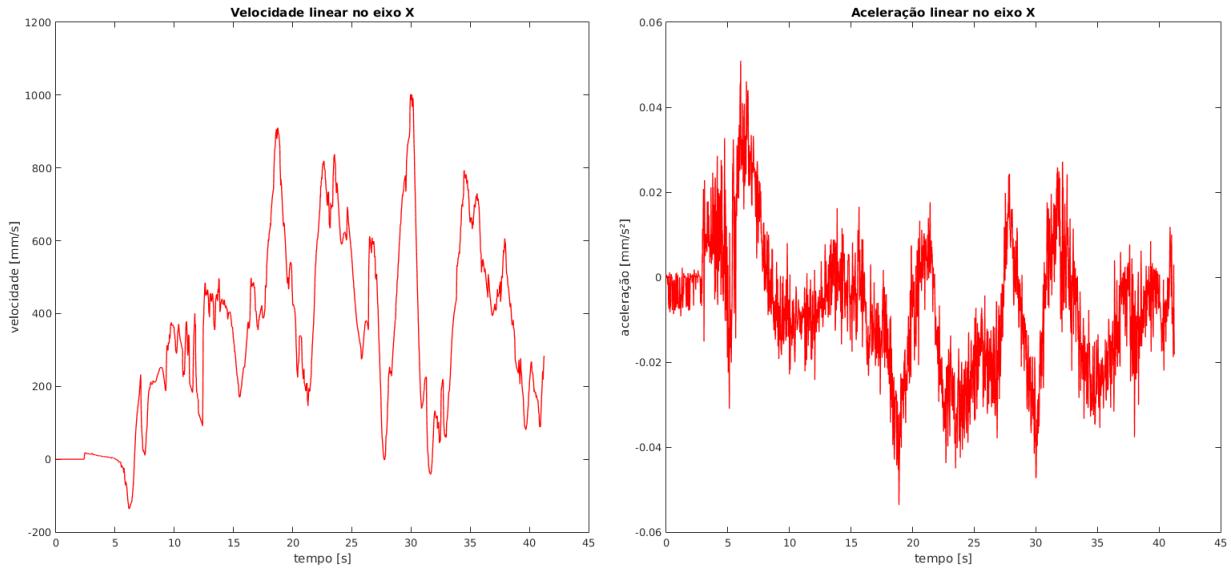


Figura 4.3: Velocidade e aceleração linear no eixo x.

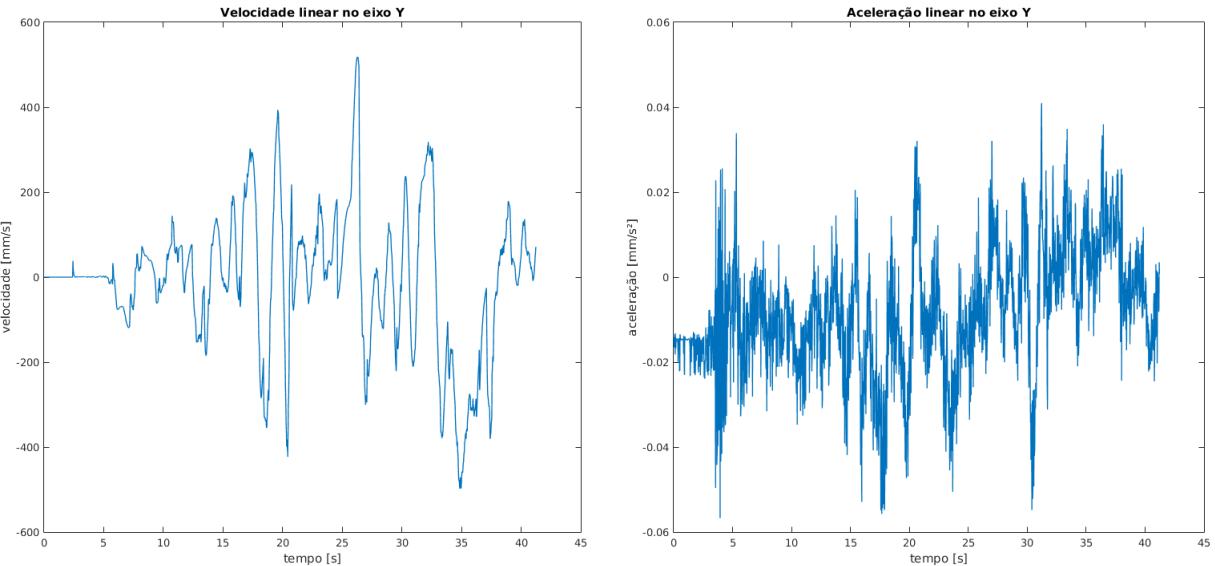


Figura 4.4: Velocidade e aceleração linear no eixo y.

4.2 Projeção de um ponto 3D do mapa para a imagem 2D

Com o intuito de verificar a metodologia de projeção de imagens de pontos no espaço 3D para pontos na imagem 2D da câmera, foi realizado um teste no qual se utiliza apenas os dados da imagem frontal do veículo. Nesse caso, é adicionado um ponto no espaço 3D no mapa de posição conhecida ($x = 0,92m$; $y = 0,08m$; $z = 0,0m$), e esse ponto é então projetado na imagem da câmera de acordo com os cálculos da Seção 2.4.5.

Para esse teste, o veículo não levantou voo, apenas obteve imagens da câmera durante alguns segundos.

A Figura 4.5 mostra o resultado do teste. A posição do ponto de interesse observado na imagem através do algoritmo de detecção de pontos de interesse ORB, em verde, é comparada à posição do ponto de interesse projetado na imagem, em vermelho, a partir da posição 3D conhecida na Tabela 4.1.

Tabela 4.1: Resultado do teste de projeção de um ponto 3D para um ponto 2D.

Ponto de interesse observado na imagem(<i>pixels</i>).	$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 240 \\ 128 \end{bmatrix}$
Ponto 3D conhecido projetado na imagem (<i>pixels</i>).	$h_c \left(\begin{bmatrix} 0.92 \\ 0.08 \\ 0.0 \end{bmatrix}, \mathbf{M} \right) = \begin{bmatrix} 238 \\ 131 \end{bmatrix}$
Distância entre os pontos em <i>pixels</i> .	$ \mu_{ij} = \left\ \begin{bmatrix} u \\ v \end{bmatrix} - h_c(\hat{\mathbf{x}}_{k k-1}, \mathbf{M}) \right\ = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$



Figura 4.5: Ponto de interesse observado (verde) e ponto projetado (vermelho). Tamanho total da imagem $(u_t, v_t) = (598, 292)$.

Observa-se pela Tabela 4.1 que a diferença máxima entre o ponto observado e o ponto conhecido projetado na imagem é de 3 *pixels*. Essa diferença é bastante aceitável, considerando que 3 *pixels* representa aproximadamente 1% e 0.5% do tamanho máximo da imagem nos eixos vertical e horizontal, respectivamente. Através deste teste, concluiu-se que o método de projeção de imagens do espaço 3D para o espaço 2D é suficientemente aceitável para a sua aplicação no SLAM.

4.3 Projeção de um ponto 2D da imagem para um ponto 3D no mapa

Para verificar o método de projeção de pontos do espaço 2D das imagens da câmera para uma posição no espaço 3D, faz-se um teste no qual um ponto de interesse da imagem é projetado utilizando as Eq. 2.10 e 2.11 da Seção 2.4.5, considerando que a posição z_z varia de 0 a 1.5m. Para identificar a validade dos valores, é realizada a medida da distância do ponto real no espaço ao veículo.

Para esse teste, assim como no teste da Seção 4.2, o veículo não levantou voo, apenas obteve imagens da câmera durante alguns segundos. O resultado pode ser visto na Tabela 4.2.

Tabela 4.2: Resultado do teste de projeção de um ponto 3D para um ponto 2D.

Ponto de interesse observado na imagem(<i>pixels</i>).	$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 240 \\ 128 \end{bmatrix}$
Posição 3D real do ponto observado (metros).	$\mathbf{p}_{fr} = \begin{bmatrix} 0,92 \\ 0,08 \\ 0,0 \end{bmatrix}$
Posição em 3D estimada do ponto observado (metros).	$\hat{\mathbf{p}}_{fr} = \begin{bmatrix} 0,9575 \\ 0,0714 \\ 0,004 \end{bmatrix}$
Distância entre os pontos em metros.	$ \mathbf{p}_{fr} - \hat{\mathbf{p}}_{fr} = \begin{bmatrix} 0,0375 \\ 0,0086 \\ 0,004 \end{bmatrix}$

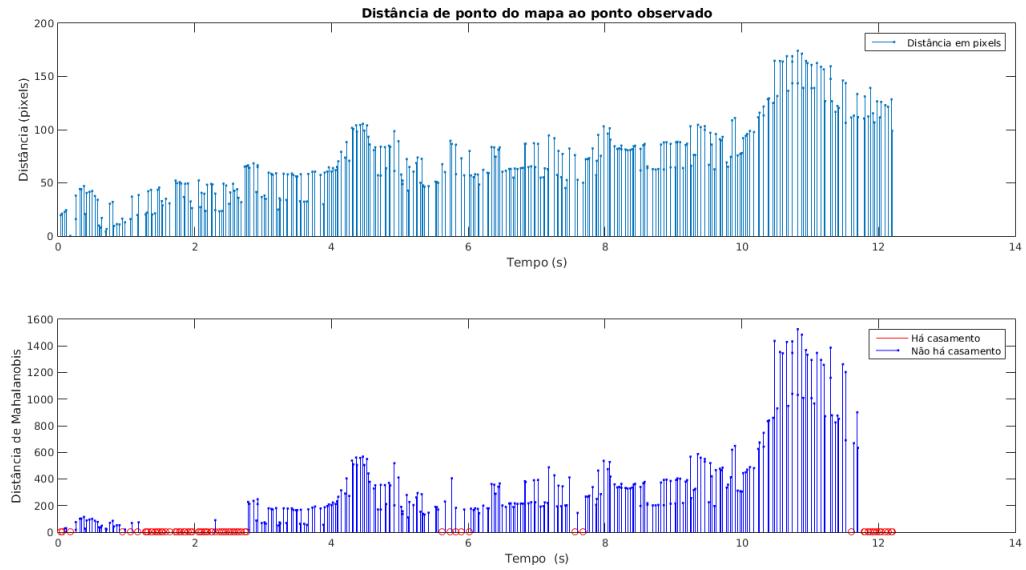
Observa-se pela Tabela 4.2 que a projeção do ponto de interesse observado na imagem para o espaço 3D se mostra bem próximo do valor da posição real do ponto observado, apresentando uma diferença pequena. A distância no eixo x nas coordenadas \mathcal{O}^r do ponto em relação à câmera, nesse caso, é uma estimativa incerta, que será corrigida posteriormente pela etapa de correção do FKE.

4.4 Casamento de ponto observado com ponto correspondente no mapa

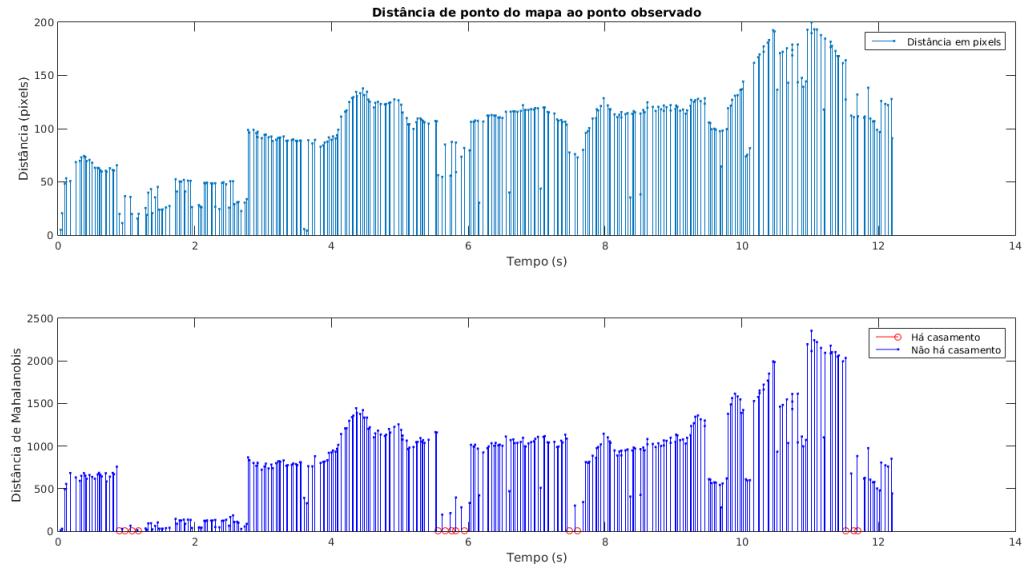
Como é dito na Seção 2.4.4, utiliza-se para esse trabalho o algoritmo ORB de extração de pontos de interesse na imagem. O casamento de pontos de interesse identificados na imagem, a partir do algoritmo ORB, com os pontos já gravados no mapa, é feito calculando-se a distância de Mahalanobis, descrita na Seção 2.4.7. Faz-se um teste no qual um ponto da imagem é projetado no mapa e, em seguida, calcula-se a distância de Mahalanobis entre o ponto observado na imagem e o ponto projetado. Os gráficos das Figuras 4.6 e 4.7 mostram como se comportam a distância de Mahalanobis e a distância em *pixels* de um ponto de interesse do mapa comparado à pontos observados na imagem da câmera, em três situações diferentes. Na primeira e na segunda situação, o veículo manteve a câmera, na maior parte do tempo, voltada para o ponto de interesse em questão. Na terceira situação, o ponto é visto apenas nos instantes iniciais e finais do teste.

No primeiro caso, assim como no segundo, o ponto presente no mapa é constantemente projetado na imagem. Em alguns momentos, há o casamento com algum ponto de interesse observado, em outros não.

No terceiro caso, é visto que o ponto foi projetado na imagem nos instantes iniciais e finais do período de teste. Em todas as vezes que ele foi projetado houve o casamento com algum ponto que foi observado.



(a) Distâncias na primeira situação.



(b) Distâncias na segunda situação.

Figura 4.6: Distâncias de Mahalanobis e distâncias em pixels de pontos observados com o ponto no mapa.

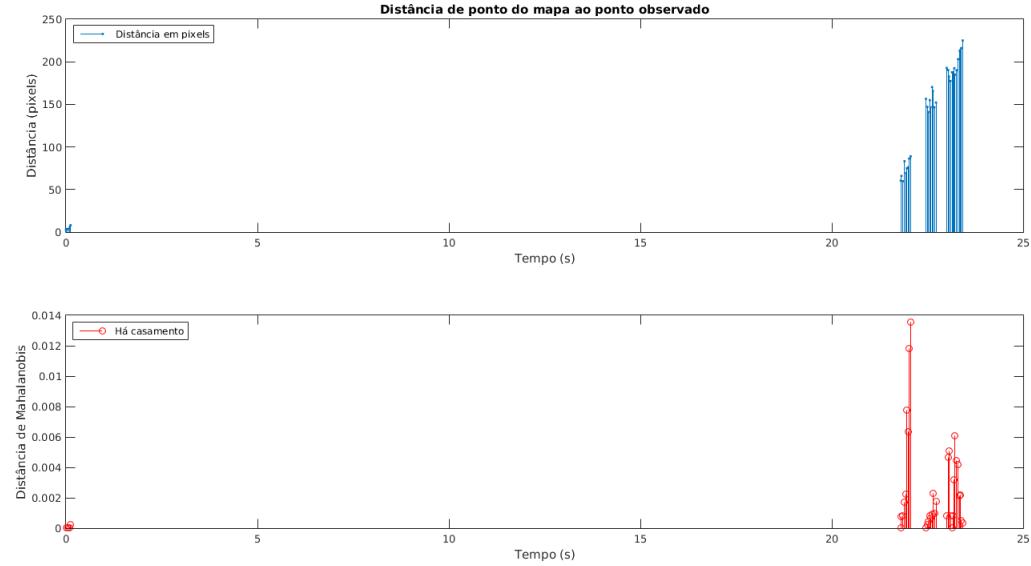


Figura 4.7: Distância de Mahalanobis e distância em pixels de pontos observados com o ponto no mapa na terceira situação.

4.5 Casamento entre pontos de interesse baseado na elipse de incerteza

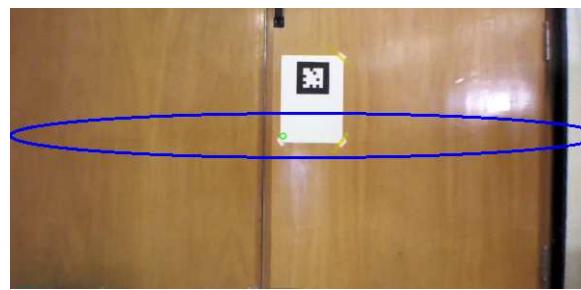
Além do método da distância de Mahalanobis para verificar casamento de pontos de interesse, também testou-se o método da elipse de incerteza, descrito na Seção 2.4.8. Nesta técnica, o casamento é feito quando o ponto de interesse observado na imagem está contido dentro da elipse de incerteza gerada pelos pontos de interesse e suas matrizes de covariância presentes no mapa. Para este teste, realizou-se um voo parado, em que o quadrirotor voava em uma mesma posição e direcionado para uma parede com um marcador.

O resultado deste teste pode ser visto pelas imagens da Figura 4.8. O ponto verde representa o ponto de interesse observado na imagem através do algoritmo ORB. As elipses azuis representam o casamento entre o ponto de interesse observado e alguma ponto de interesse presente no mapa. Já as elipses vermelhas, representam a ausência de casamento entre os pontos de interesses. Observa-se pelas imagens adquiridas que o resultado foi coerente com o teste proposto.

Entretanto, apesar do teste ter sido bem sucedido, o casamento baseado na elipse de incerteza não apresentou um resultado satisfatório ao utilizá-lo no teste de SLAM completo. Na Figura 4.9, percebe-se que o caminho estimado pelo SLAM, em amarelo, se diverge completamente do caminho estimado somente pela etapa de predição, em verde. Baseado nessa análise, o casamento baseado nas elipses de incerteza não foi utilizado neste trabalho. O método da distância de Mahalanobis mostrou-se mais consistente durante os testes, o tornando a melhor escolha para o casamento de pontos de interesse no SLAM implementado.



(a)



(b)



(c)



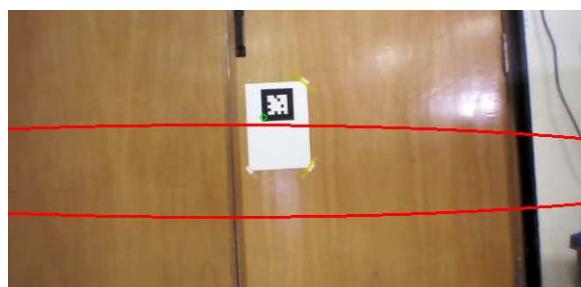
(d)



(e)



(f)



(g)



(h)

Figura 4.8: Verificação de casamento entre o ponto de interesse observado e pontos do mapa baseado nas elipses de incerteza

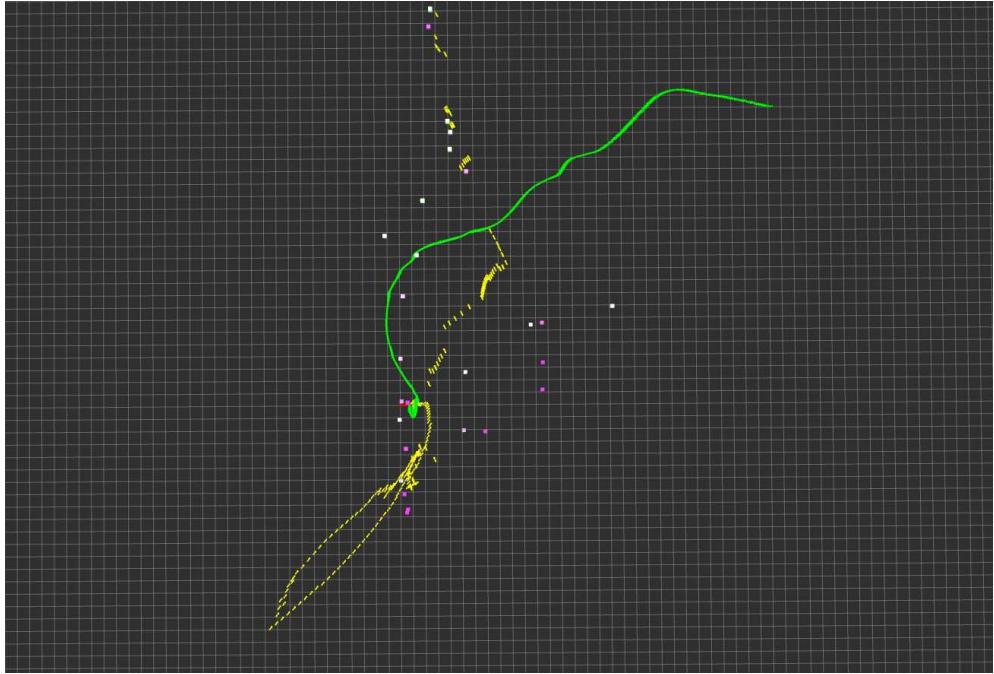


Figura 4.9: Vista superior do SLAM implementado baseado no casamento usando elipses de incerteza. Em verde, é apresentado o percurso estimado apenas pela predição, e em amarelo, é apresentado o percurso estimado pela predição junto com a correção.

4.6 Correção da posição de pontos de interesse

Para averiguar a etapa de correção do mapa do SLAM, realizou-se um voo teste com o quadrirrotor posicionado em uma pequena área e direcionado para uma parede com um marcador fixo. Este teste tem como objetivo analisar o comportamento das posições do espaço 3D dos cinco primeiros pontos de interesse presentes no mapa.

Os gráficos da Figura 4.10 mostram a posição nos três eixos dos cinco primeiros pontos de interesse do mapa no decorrer do tempo. É possível ver que em todas as estimativas de posição dos pontos observados convergem para um valor final. Além disso, pode-se ver que as estimativas demoram, em média, de 2 a 3 segundos para estabilizarem. Como o quadrirrotor realizou um voo em que sua posição se concentrava em uma pequena área ao seu redor, o comportamento das posição dos pontos de interesse foi coerente com o experimento.

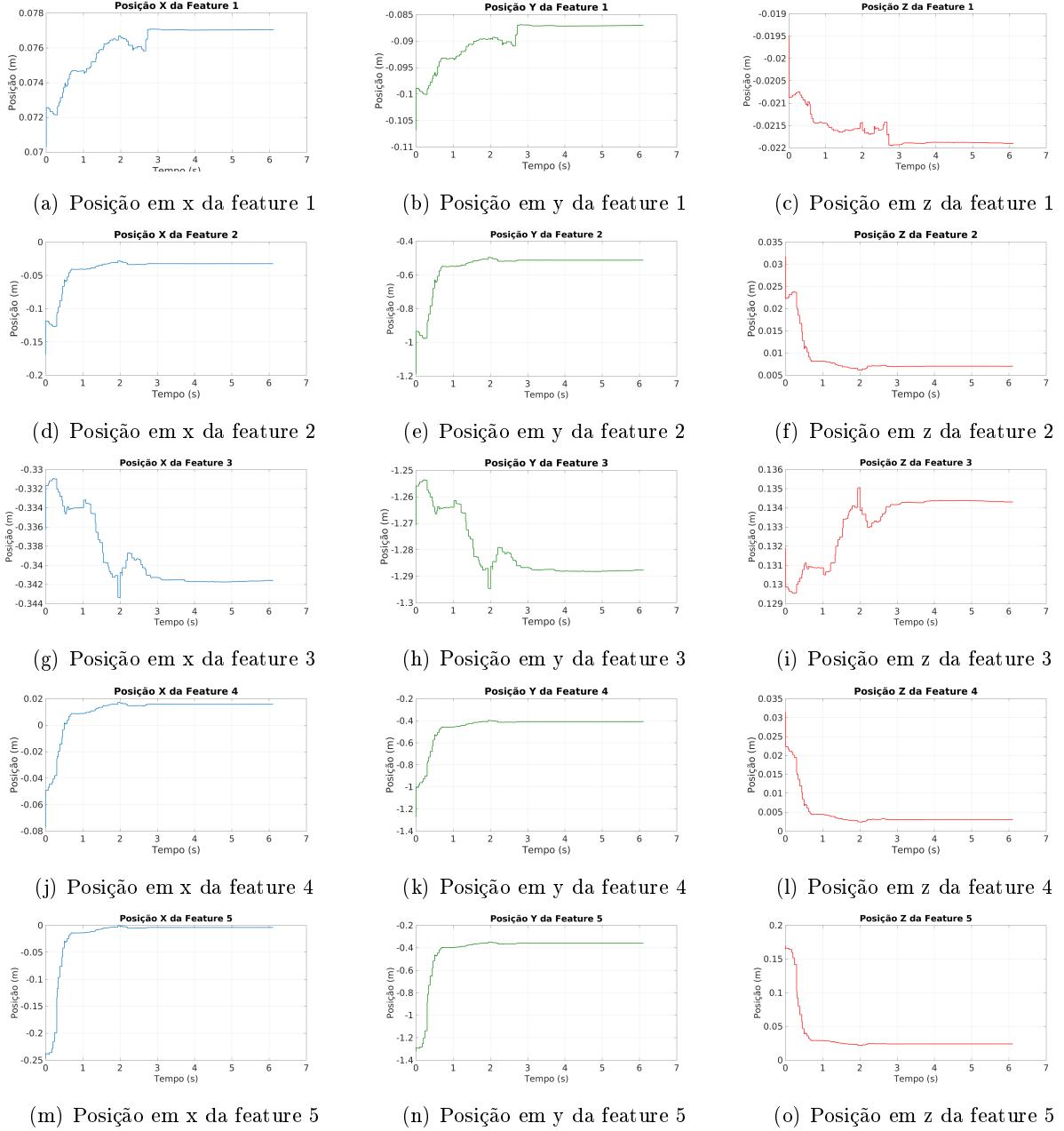


Figura 4.10: Posição dos pontos de interesse no tempo.

4.7 Comparação da estimativa antes e após correção

Para o teste de comparação entre a estimativa de posição antes da correção e após a correção, todo o modelo de SLAM é aplicado. O veículo realizou um voo em um corredor do prédio SG-11 no Campus Darcy Ribeiro da Universidade de Brasília, representado pela Figura 4.11. Foram colocadas marcações na parede apenas com o intuito de o sistema ser capaz de identificar pontos de interesse, tendo em vista que a parede branca não oferece pontos de interesse necessários para o SLAM. Além disso, as marcações servem para saber se os pontos inseridos no mapa possuem posições parecidas com as posições reais deles no espaço.

O resultado do teste é representado pelas Figuras 4.12 e 4.13, mostrando a visão superior e a visão em perspectiva da estimativa de localização do quadrirrotores e a posição dos pontos de interesse inseridos no mapa. O percurso de cor verde representa o caminho estimado apenas com os dados de navegação, ou seja, não há correção. O percurso de cor amarelo representa o caminho estimado pelo sistema do SLAM completo, ou seja, a localização é estimada tanto pelos dados de navegação como dados provindos da câmera através da etapa de predição e correção.

Percebe-se pelo resultado obtido que o caminho sem correção se diferencia do caminho com correção na distância máxima percorrida na ida e por apresentar uma navegação mais ruidosa. Ambas as diferenças são resultados pelas constantes correções realizadas pelo filtro de Kalman estendido.

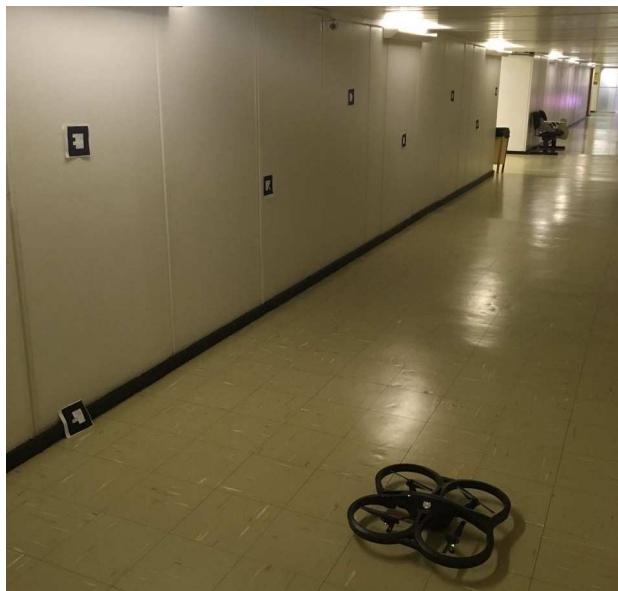


Figura 4.11: Ambiente onde foi feito o voo do teste de comparação da estimativa antes e após correção.

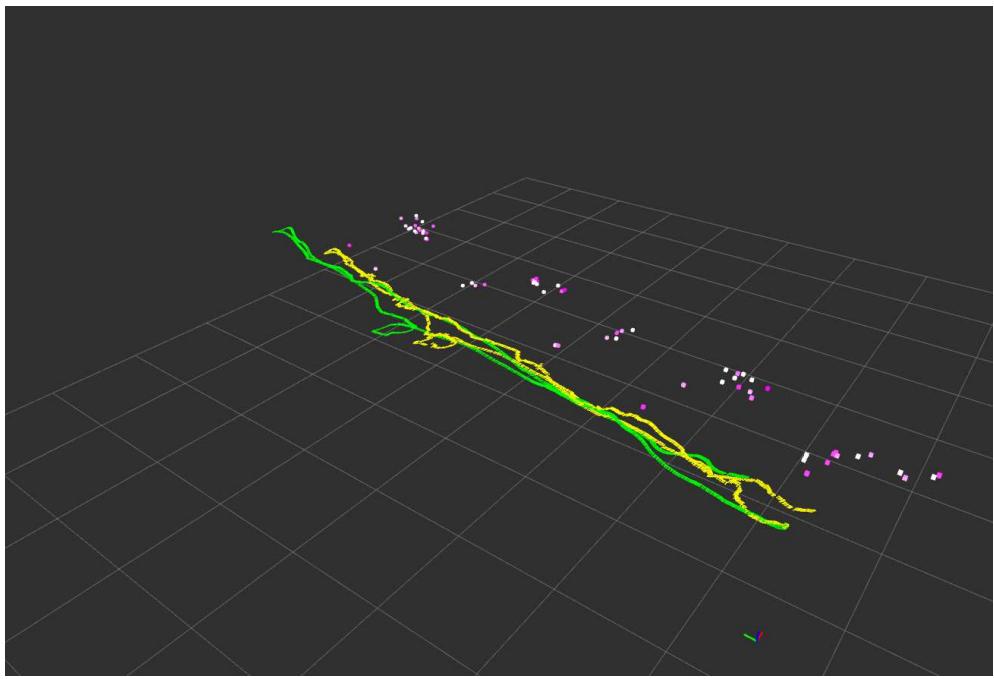


Figura 4.12: Visualização do mapa 3D gerado no teste de comparação da estimativa antes e após correção em ambiente RVIZ. Em verde, é apresentado o caminho estimado apenas pela predição, e em amarelo, é apresentado o caminho estimado pela predição e a correção.

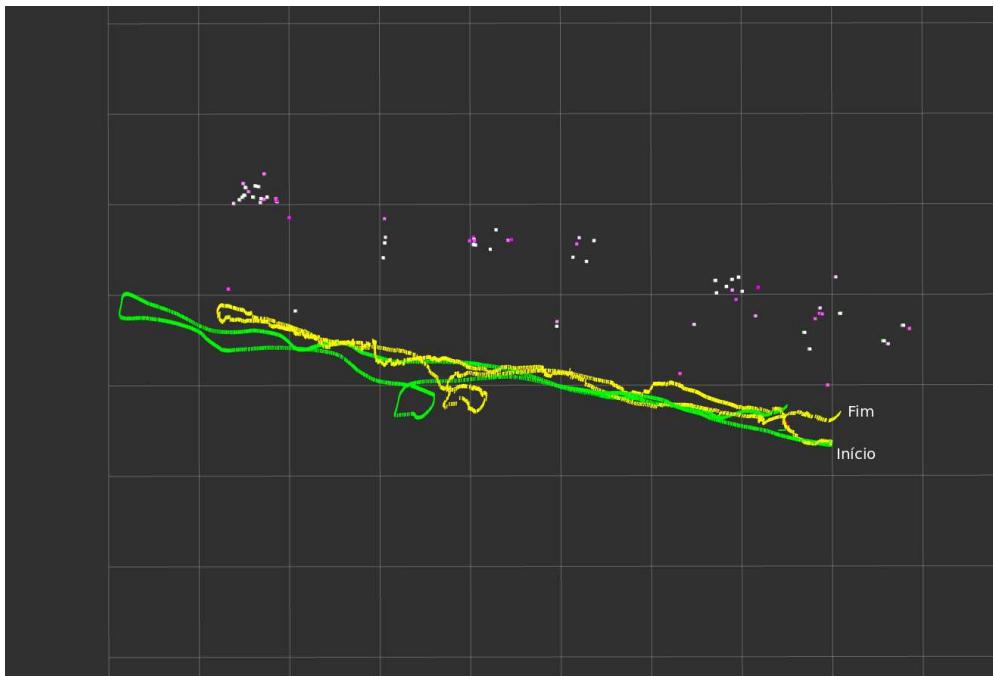


Figura 4.13: Visualização do mapa 2D gerado no teste de comparação da estimativa antes e após correção em ambiente RVIZ. Em verde, é apresentado o caminho estimado apenas pela predição, e em amarelo, é apresentado o caminho estimado pela predição e a correção.

Os dados da matriz de covariância, cuja diagonal apresenta valores do desvio padrão σ dos

elementos do vetor de estados, foram gravados durante o voo para que possam ser plotados os gráficos das Figuras 4.14, 4.15 e 4.16, que permitem a análise de coerência do filtro utilizando dados dos três eixos de translação do veículo. Para que haja coerência no filtro, deve ser observado que a correção da posição não extrapola o intervalo 3σ . Os gráficos mostram que os limites do intervalo 3σ não são ultrapassados durante a execução do sistema, indicando coerência do filtro.

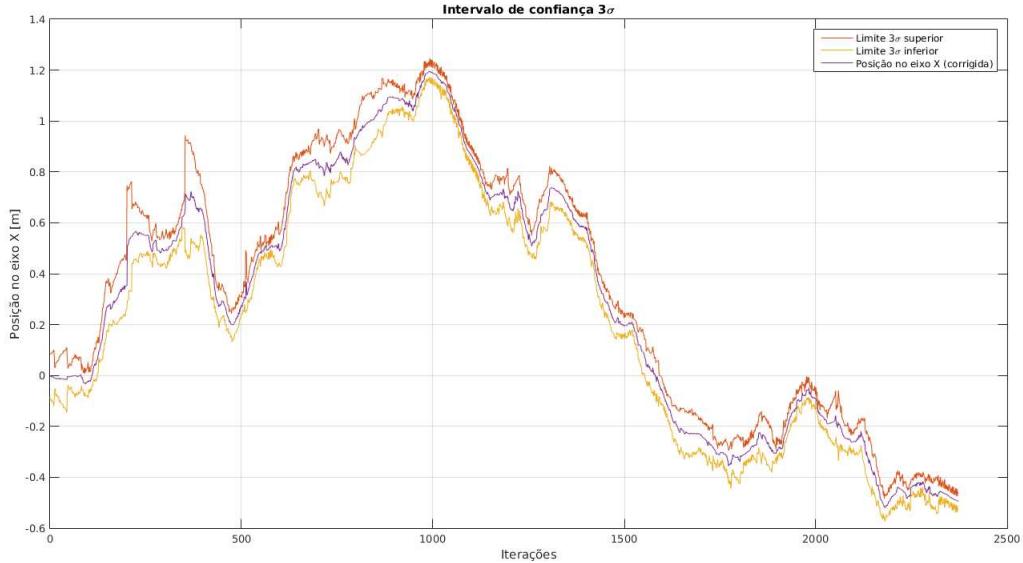


Figura 4.14: Gráfico de intervalo de confiança do filtro a partir de dados de posição no eixo X de translação.

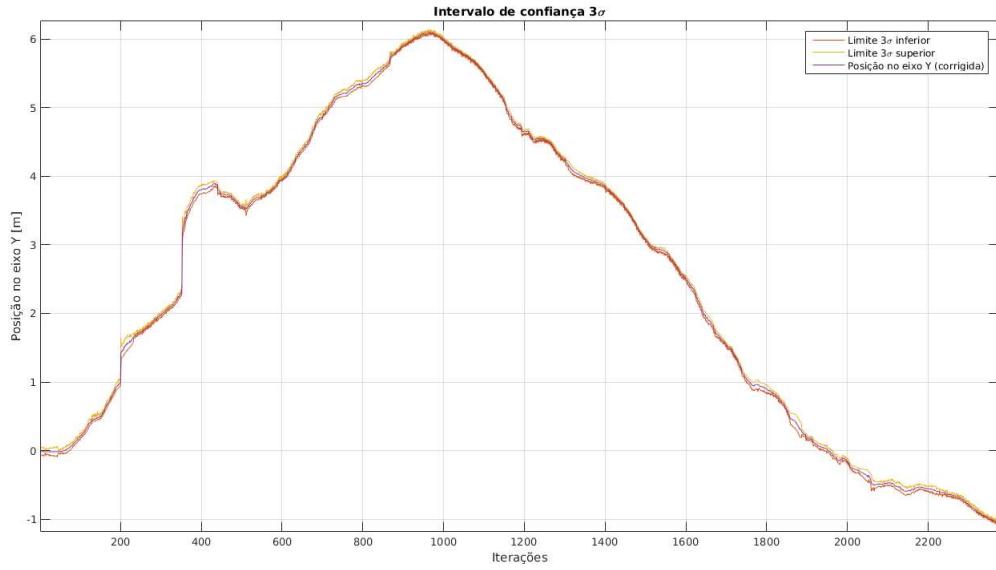


Figura 4.15: Gráfico de intervalo de confiança do filtro a partir de dados de posição no eixo Y de translação.

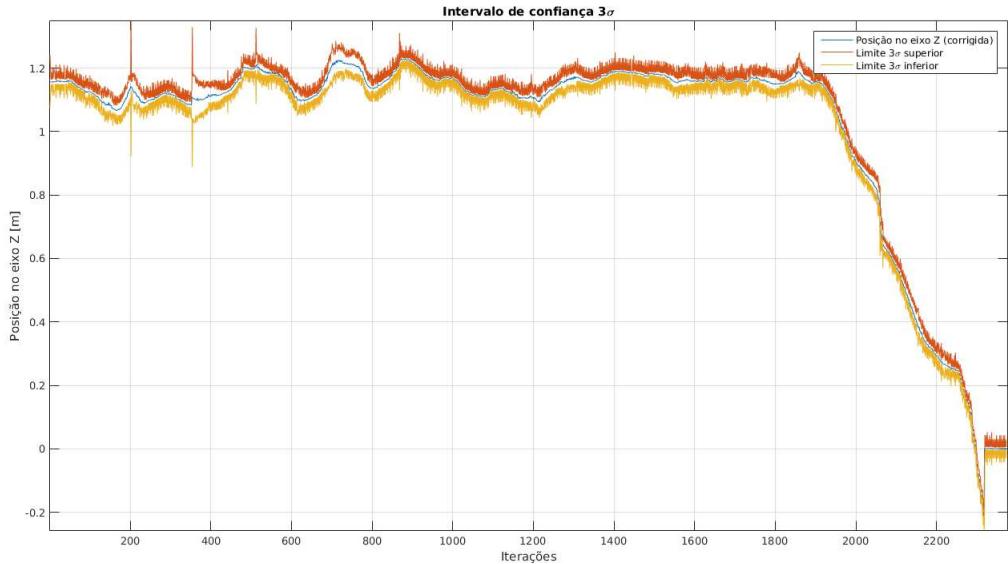


Figura 4.16: Gráfico de intervalo de confiança do filtro a partir de dados de posição no eixo Z de translação.

4.8 SLAM Visual Monocular completo

Para averiguar o resultado do SLAM Visual Monocular baseado em filtro de Kalman estendido implementado neste trabalho, realizou-se o voos em dois ambientes diferentes:

- Cobertura de apartamento
- Ambiente de trabalho do LARA

Para cada um dos testes, fixou-se para três o número máximo de pontos de interesse detectados por *frame*. Esse limite foi escolhido porque o aumento desse valor exigia um poder computacional muito grande, deixando o sistema muito lento.

4.8.1 Voo na cobertura de um prédio

Este voo foi realizado por volta das 14h30 do dia 27 de Junho de 2016 em um dia ensolarado com presença de ventos. A Figura 4.17 mostra um modelo tridimensional da cobertura do prédio onde foi realizado o voo. A cobertura escolhida possui paredes altas o suficiente para que não houvesse perigo do quadrirotor cair. Com o intuito de comparação entre o ambiente real e o mapeado, obteve-se a planta original da cobertura, que pode ser visto no Anexo I. O resultado da localização e do mapeamento de pontos de interesse podem ser observados pelas Figuras 4.18 e 4.19.

Considerando as marcações representadas por letras maiúsculas na Figura 4.18, a sequência do caminho percorrido pelo quadrirotor foi: \overline{AB} , \overline{BC} , \overline{CD} e \overline{DE} . Como não foi possível medir

Caminho	Distância Estimada	Distância Medida
\overline{AB}	8.0m	8.5m
\overline{BC}	19.0m	20.5m
\overline{CD}	13.0m	12.5m
\overline{DE}	14.0m	13.5m

Tabela 4.3: Comparação entre o caminho percorrido estimado e o caminho medido da cobertura do apartamento através de pontos de referências.

a posição real do quadrirotor durante o voo, mediu-se a distância entre os pontos de referência percorridos para realizar uma comparação com as distâncias estimadas. A Tabela 4.3 mostra a relação entre as distâncias estimadas e as distâncias medidas entre os pontos de referência. Os dados da tabela mostram que as distâncias estimadas são coerentes com as distâncias medidas. Entretanto, como as distâncias medidas não representam a distância real que o veículo percorreu, não é possível realizar uma comparação quantitativa dos dados obtidos. Para facilitar a visualização do resultado, é mostrado na Figura 4.20 a vista superior do caminho realizado pelo quadrirotor e da posição dos pontos de interesse presentes no mapa sobreposto na planta da cobertura. Mesmo com a presença de ventos no ambiente, o SLAM foi capaz de estimar bem a posição do veículo no ambiente.



Figura 4.17: Vista do modelo tridimensional da cobertura do prédio. Fonte: Google Earth

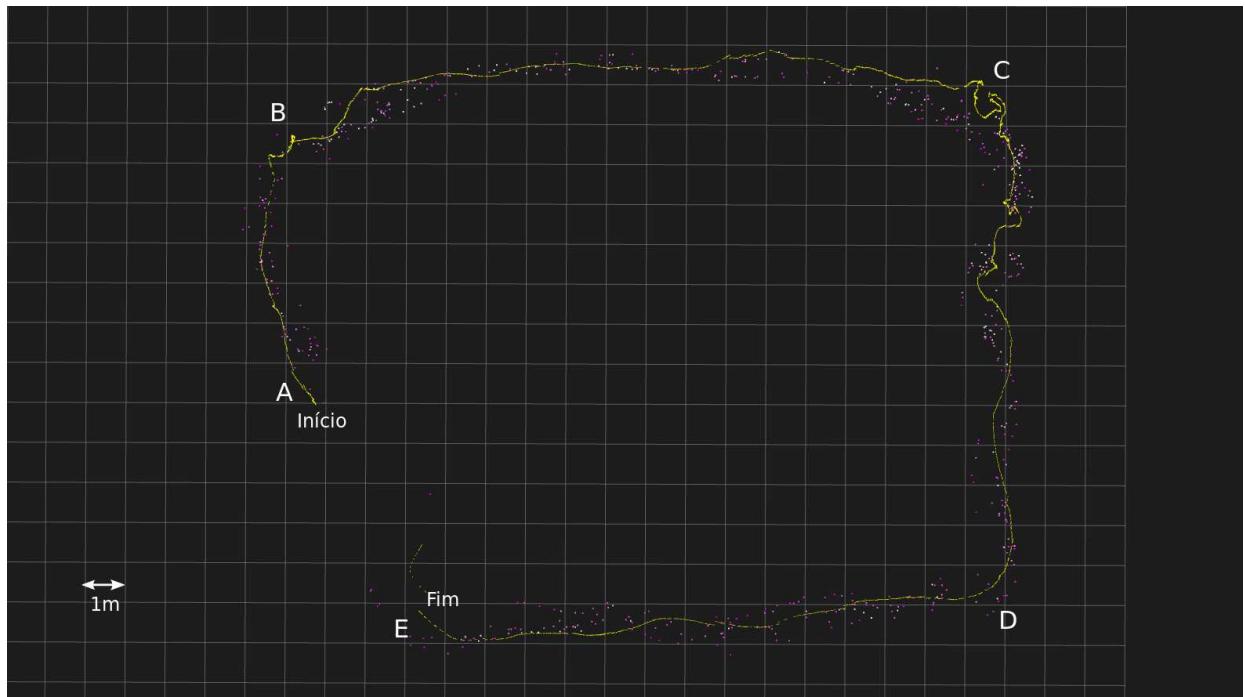


Figura 4.18: Vista superior do mapeamento e da localização do quadrirroto em uma cobertura de um prédio

4.8.2 Voo no ambiente de trabalho do LARA

Este voo foi realizado por volta de 16h00 do dia 3 de Julho de 2016. Ao contrário do teste realizado na cobertura do apartamento, o ambiente do laboratório não possui muitas perturbações como o vento e a luz do sol. Além disso, o LARA apresenta um ambiente com presença maior de elementos que caracterizam um ponto de interesse como cantos e bordas. O resultado da localização e do mapeamento de pontos de interesse podem ser observados pelas Figuras 4.21 e 4.22.

Considerando as marcações representadas por letras maiúsculas na Figura 4.21, a sequência do caminho percorrido pelo quadrirroto foi: \overline{AC} , \overline{BC} , \overline{CD} , \overline{DE} , \overline{EB} , \overline{BC} , \overline{CF} e \overline{FG} . Como não foi possível medir a posição real do quadrirroto durante o voo, mediu-se a distância entre os pontos de referência percorridos para realizar uma comparação com as distâncias estimadas. A Tabela 4.4 mostra a relação entre as distâncias estimadas e as distâncias medidas para cada segmento de caminho. É importante frisar que a distância medida se baseia em pontos de referência, não sendo possível realizar uma análise de erro quantitativo. Entretanto, é possível perceber que, pelos dados da tabela e da Figura 4.21, o resultado foi coerente com o medido, apresentando um caminho com um formato esperado. Ainda assim, é possível perceber que houve erros de percurso, principalmente nos trechos \overline{DE} e \overline{EB} . Esses erros são justificados pelas constantes mudanças de orientação *yaw* durante esse voo e pelo *drift* dos valores da UMI. É possível observar também que, entre os trechos \overline{EB} e \overline{BC} , o SLAM corrigiu a orientação do veículo, o colocando de volta para o percurso esperado, seguindo um percurso semelhante ao realizado anteriormente.

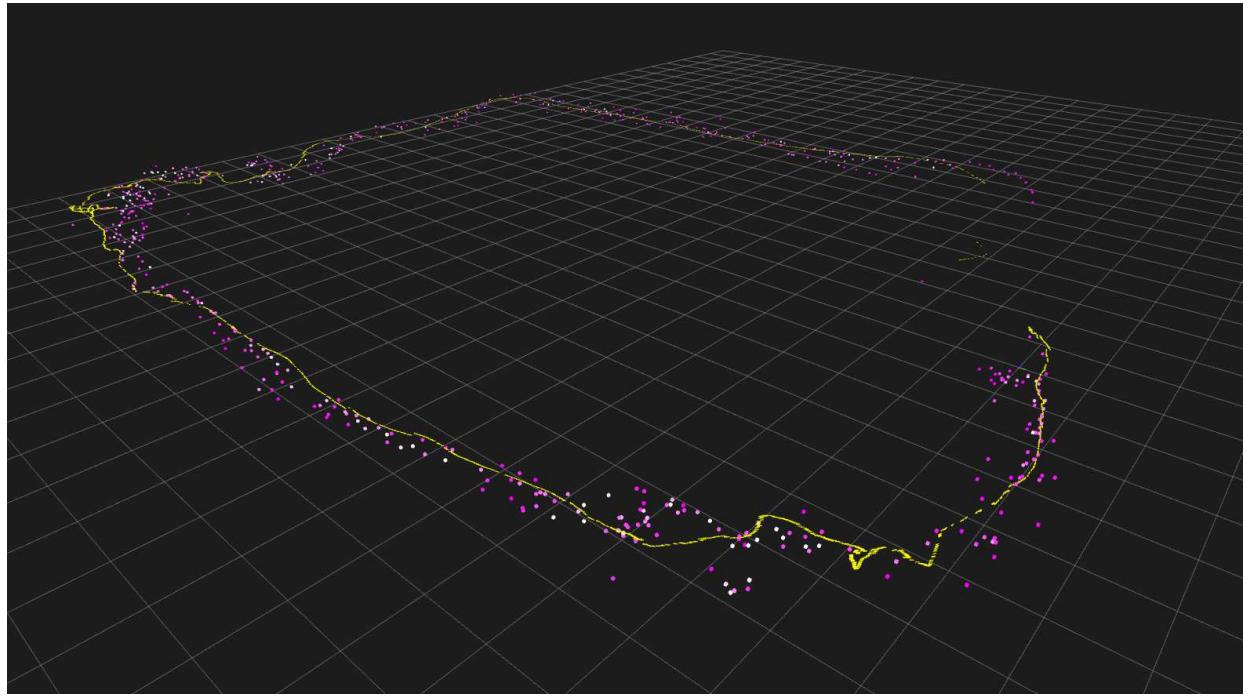


Figura 4.19: Vista de perspectiva do mapeamento e da localização do quadríroto em uma cobertura de um prédio

Caminho	Distância Estimada	Distância Medida
\overline{AC}	8.0m	7.8m
\overline{BC}	4.0m	3.6m
\overline{CD}	4.3m	4.8m
\overline{DE}	2.7m	3.6m
\overline{EB}	5.0m	4.8m
\overline{CF}	10.7m	9.9m
\overline{FG}	8.0m	7.8m

Tabela 4.4: Comparaçāo entre os segmentos de caminho percorrido pela estimação e os caminho medido no LARA

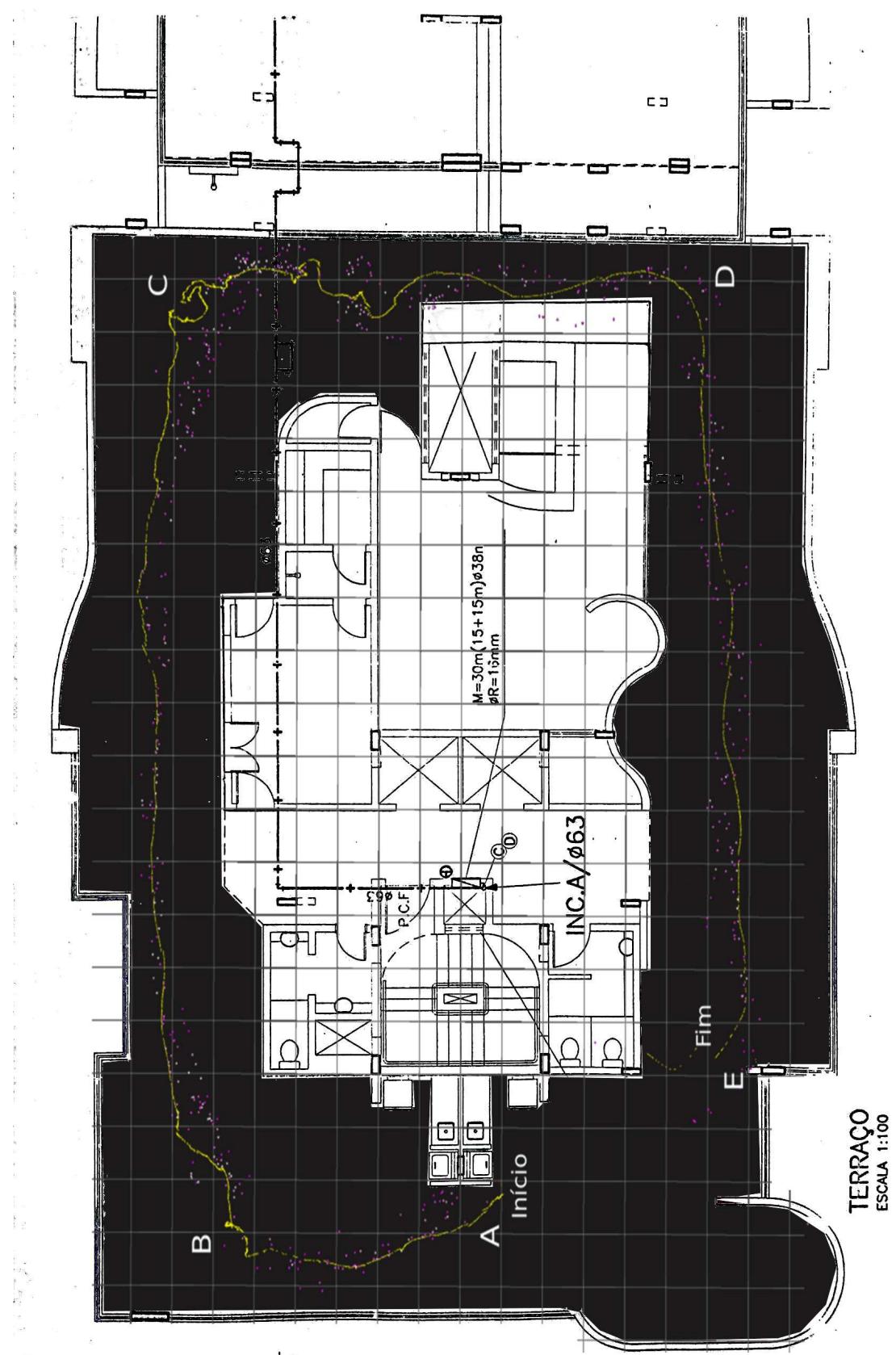


Figura 4.20: Vista superior do mapeamento e da localização do quadrirroto em uma cobertura do prédio sobreposto com a planta original

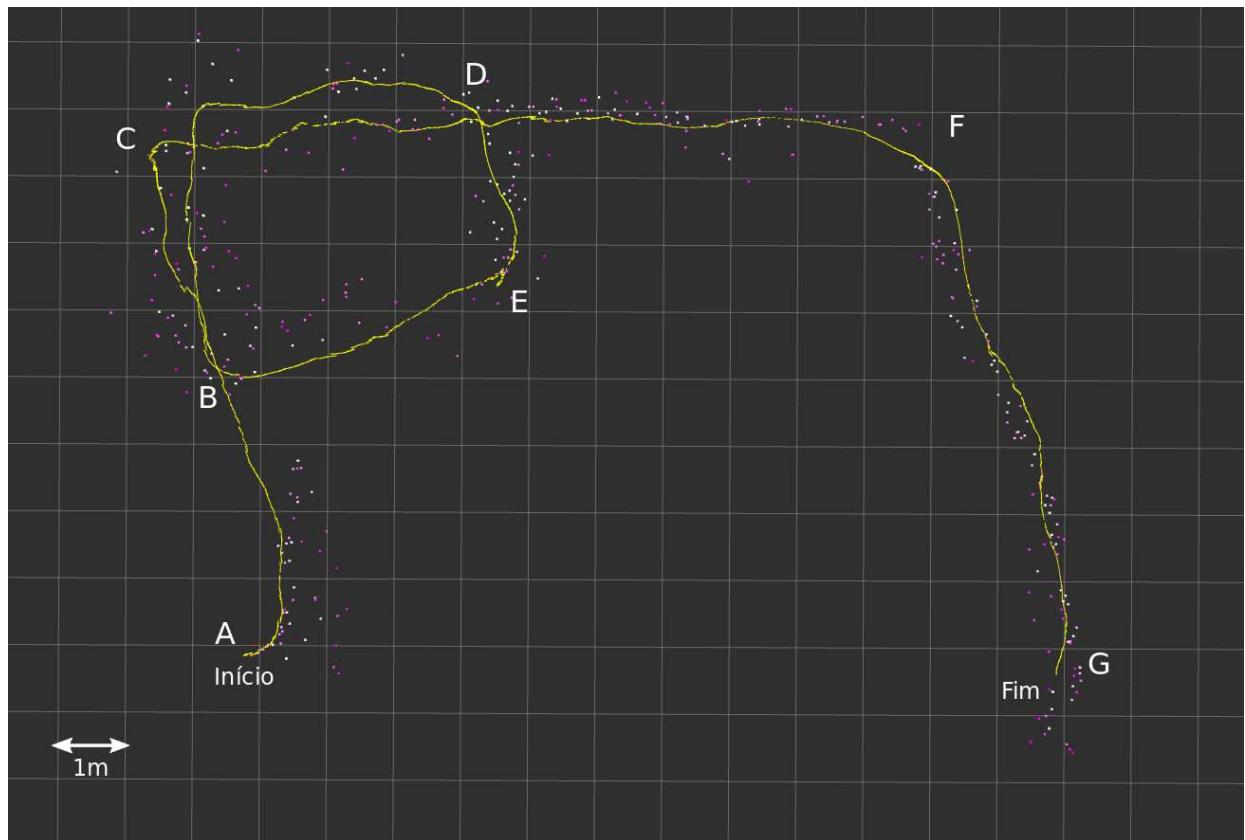


Figura 4.21: Vista superior do mapeamento e da localização do quadrirroto no LARA

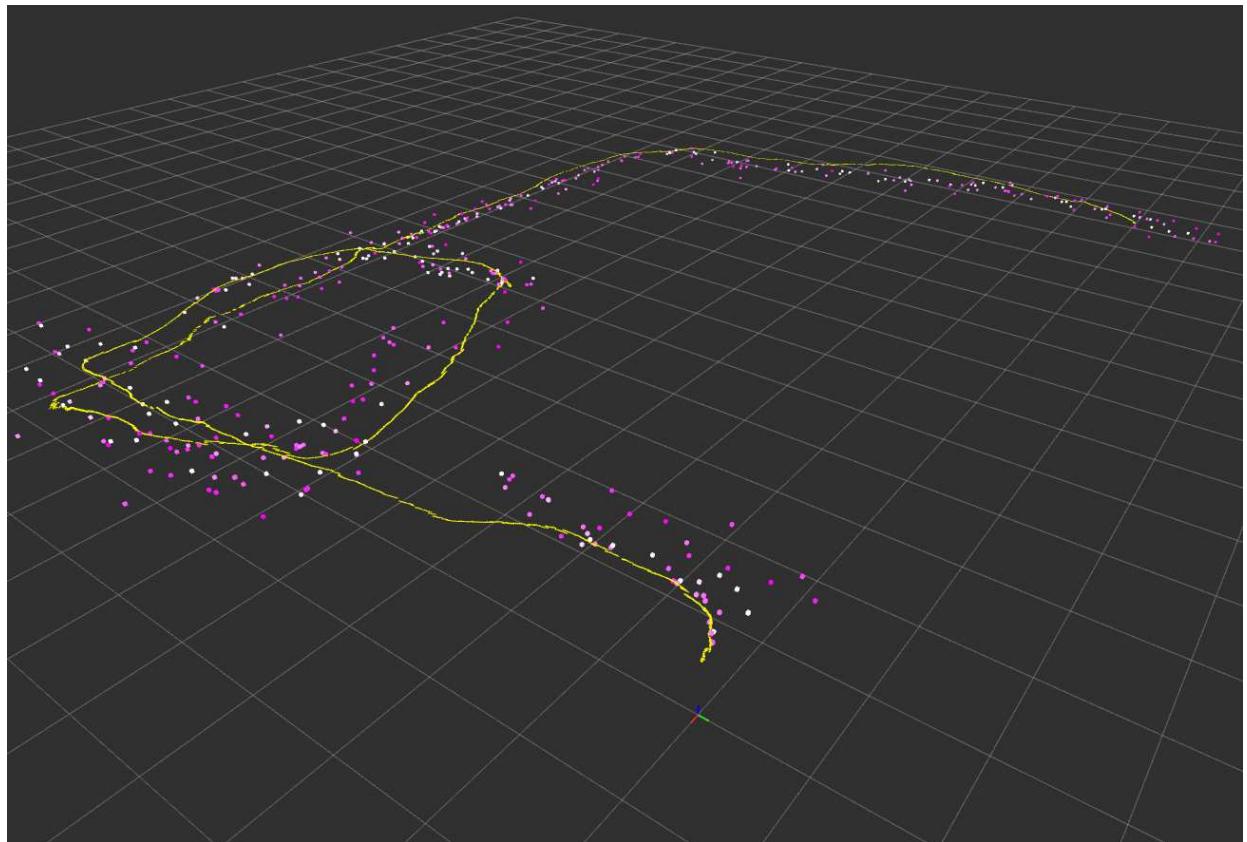


Figura 4.22: Vista de perspectiva do mapeamento e da localização do quadrirroto no LARA

Capítulo 5

Conclusões

Esse trabalho apresentou o desenvolvimento de um sistema de mapeamento e localização simultâneos baseado em informações de imagens do ambiente e também de dados de navegação de um quadrirrotoor do modelo AR.Drone Parrot 2.0. Foi feita uma revisão bibliográfica acerca do tema, indicando alguns trabalhos já existentes nos quais esse trabalho se baseou. São indicadas ainda algumas informações importantes acerca de como os dados do quadrirrotoor são gerados.

O sistema foi feito utilizando informações de velocidade, altitude e orientação pré-processados pelo *hardware* do veículo. Essas informações são tratadas por um computador de base, que é responsável pela aplicação do Filtro de Kalman Estendido que realiza as correções de posicionamento tanto do veículo quanto de pontos de interesse observados no ambiente.

Foi verificado dificuldade em aplicar o sistema em ambientes que possuem o solo pouco texturizado, tendo em vista que as informações de velocidade do veículo são geradas com importante contribuição da odometria visual feita com a câmera vertical. Além disso, fatores como iluminação, ventos e presença de muitos pontos característicos no ambiente influenciaram bastante nos resultados. Por esses motivos, ajustaram-se os parâmetros de incerteza que atribuem o nível de confiança no modelo de predição e na medição dos pontos de interesse de acordo com cada situação.

Os resultados obtidos mostram que o sistema é capaz de estimar de maneira *offline* a posição do veículo e dos pontos de interesse durante um voo realizado controlado remotamente. O mapa gera pontos de interesse que estavam próximos ao veículo durante o voo. Os resultados são satisfatórios e indicam que o sistema pode ser incorporado em veículos aéreos que desejam se localizar em ambientes desconhecidos. Os dados são publicados em tópicos do ROS, portanto permite que outros nós sejam executados para realizar outros processos que podem ser úteis para a tarefa desejada.

Para trabalhos futuros, é proposto que o sistema seja desenvolvido de maneira mais eficiente do ponto de vista computacional, tendo em vista que os códigos desenvolvidos nesse trabalho ainda não são capazes de realizar SLAM Visual *online*. A capacidade de processar os dados *online* seria importante para que seja aplicado em um veículo autônomo. Além disso, propõe-se que sejam estimados as informações de velocidade diretamente da unidade de medição inercial, sem que seja

utilizado a odometria visual do AR.Drone. Dessa maneira, o sistema teria propósito mais geral e então poderia ser utilizado em outros veículos aéreos que não dispõem de câmera vertical apontada para o solo, ou em ambientes que não possuem solo suficientemente texturizado.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] BRISTEAU, P. J. et al. The Navigation and Control technology inside the AR.Drone micro UAV. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, v. 18, n. PART 1, p. 1477–1484, 2011.
- [2] BORGES, G. A. Mapeamento e localização simultâneos. *Robótica Móvel*, p. 208–242, 2014.
- [3] SMITH, R.; SELF, M.; CHEESEMAN, P. A stochastic map for uncertain spatial relationships. *Proceedings of the 4th international symposium on Robotics Research*, n. 0262022729, p. 467–474, 1988.
- [4] SMITH, R.; SELF, M.; CHEESEMAN, P. Estimating Uncertain Spatial Relationships in Robotics. *Autonomous Robot Vehicles*, v. 4, n. April, p. 167–193, 1990.
- [5] NEWMAN, P. On the Structure and Solution of the Simultaneous Localisation and Map Building Problem. *Doctoral diss., University of Sydney*, p. 1999, 1999.
- [6] DAVISON, A. J. et al. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 29, n. 6, p. 1052–1067, 2007.
- [7] EFRON, B. Bootstrap methods: another look at the jackknife. In: *Breakthroughs in Statistics*. [S.l.]: Springer, 1992. p. 569–593.
- [8] CIVERA, J.; DAVISON, A. J.; MONTIEL, J. M. M. Inverse Depth Parameterization for Monocular {SLAM}. *IEEE Transactions on Robotics*, v. 24, n. 5, p. 932–945, 2008.
- [9] MENG, X.; HONG, F.; CHEN, Y. Monocular simultaneous localization and mapping with a modified covariance Extended Kalman Filter. *Proceedings - 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2009*, v. 2, p. 900–903, 2009.
- [10] RUBLEE, E.; BRADSKI, G. ORB - an efficient alternative to SIFT or SURF. 2011.
- [11] MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, v. 31, n. 5, p. 1147–1163, 2015.
- [12] MAHALANOBIS, P. C. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, v. 2, p. 49–55, 1936.
- [13] DIJKSHOORN, N. Simultaneous localization and mapping with the AR . Drone. *Universiteit Van Amsterdam*, p. 239–244, 2012.

- [14] SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. *Robot modeling and control*. [S.l.]: Wiley New York, 2006.
- [15] ROSTEN, E.; DRUMMOND, T. Machine learning for high-speed corner detection. In: SPRINGER. *European conference on computer vision*. [S.l.], 2006. p. 430–443.
- [16] CALONDER, M. et al. Brief: Binary robust independent elementary features. In: SPRINGER. *European conference on computer vision*. [S.l.], 2010. p. 778–792.
- [17] HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: CITESEER. *Alvey vision conference*. [S.l.], 1988. v. 15, p. 50.
- [18] LOWE, D. G. Object recognition from local scale-invariant features. In: IEEE. *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. [S.l.], 1999. v. 2, p. 1150–1157.
- [19] FORSYTH, D. A.; PONCE, J. *Computer vision: a modern approach*. [S.l.]: Prentice Hall Professional Technical Reference, 2002.
- [20] HALL, P. *The bootstrap and Edgeworth expansion*. [S.l.]: Springer Science & Business Media, 2013.
- [21] KALMAN, R. E. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, American Society of Mechanical Engineers, v. 82, n. 1, p. 35–45, 1960.
- [22] EFRON, B. Bootstrap methods: Another look at the jackknife. In: *Springer Series in Statistics*. [S.l.]: Springer Science Business Media, 1992. p. 569–593.
- [23] THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic robotics*. [S.l.]: MIT press, 2005.

ANEXOS

I. PLANTA DA COBERTURA DO APARTAMENTO

Segue neste anexo a planta da cobertura do apartamento utilizado para realizar o voo teste descrito na Seção 4.8.1.

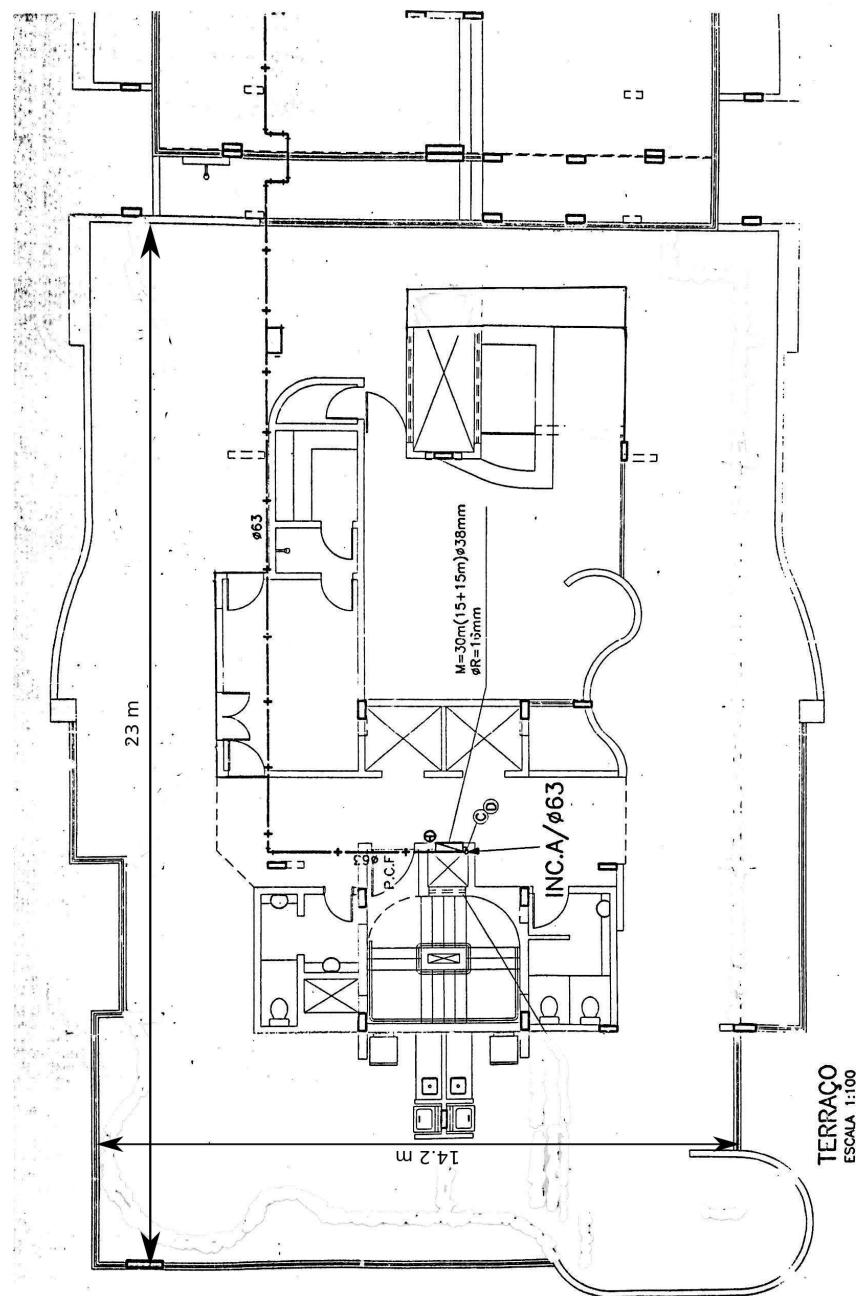


Figura I.1: Planta da cobertura do apartamento

II. DESCRIÇÃO DO CONTEÚDO DO CD

No CD entregue com o trabalho é disponibilizada uma cópia digital do relatório, bem como uma pasta com todos os arquivos utilizados para a elaboração do mesmo. Além disso, o arquivo texto README descreve como executar os códigos utilizados no trabalho.