SENAI CIMATEC
ESPECIALIZAÇÃO EM AUTOMAÇÃO, CONTROLE E ROBÓTICA

TRABALHO DE CONCLUSÃO DE CURSO

LOCALIZAÇÃO E MAPEMENTO SIMULTÂNEO (SLAM) COM VISÃO MONOCULAR APLICADO EM ROBÔ AÉREO

Aluno: Danilo Costa Oliveira

Orientador: Prof. Msc. Oberdan Rocha Pinheiro

Salvador, 18 de Junho de 2014.

DANILO COSTA OLIVEIRA

LOCALIZAÇÃO E MAPEMENTO SIMULTÂNEO (SLAM) COM VISÃO MONOCULAR APLICADO EM ROBÔ AÉREO

Trabalho de conclusão de curso apresentado à faculdade Senai Cimatec, como requisito para concessão de grau de especialista em automação, controle e robótica.

Orientador: Prof. Msc. Oberdan Rocha Pinheiro

Salvador, 18 de Junho de 2014.

Ficha catalográfica elaborada pela Biblioteca da Faculdade de Tecnologia SENAI CIMATEC

O48l Oliveira, Danilo Costa

Localização e mapeamento simultâneo (SLAM) com visão monocular aplicado em robô aéreo / Danilo Costa Oliveira – Salvador, 2014.

50 f. : il. color.

Orientador: Prof. Me. Oberdan Rocha Pinheiro.

Monografia (Especialização em Automação, Controle e Robótica) – Programa de Pós-Graduação, Faculdade de Tecnologia Senai - CIMATEC, Salvador, 2014.

Inclui referências bibliográficas.

1. Robótica aérea. 2. Quadrotor. 3. SLAM (Localização e Mapeamento Simultâneo). 4. Câmara monocular. I. Faculdade de Tecnologia Senai – CIMATEC. II. Pinheiro, Oberdan Rocha. III. Título.

CDD: 629.892

NDI - 01

DANILO COSTA OLIVEIRA

LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEO (SLAM) COM VISÃO MONOCULAR APLICADO EM ROBÔ AÉREO

Projeto Final de Curso aprovado com nota 7,5 como requisito de Especialista em Automação e Controle, tendo sido julgado pela Banca Examinadora formada pelos Professores:

_____

Msc. Oberdan Rocha Pinheiro – Orientador

Faculdade de Tecnologia SENAI CIMATEC

_____

Msc. Marcelo Henrique Souza Bomfim – Professor

Faculdade de Tecnologia SENAI CIMATEC

Salvador, 18 de junho de 2014.

RESUMO

A robótica móvel é uma área que vem despertando o interesse da comunidade acadêmica. Com o surgimento e popularização de tecnologias como o MEMS (sistemas micro eletromecânicos), câmeras digitais, sensores RGB-D, etc, foi possível projetar robôs aéreos estáveis em miniatura (MAVs) e de baixo custo, possibilitando a realização segura de voos *indoor*. Nesse cenário, o quadrotor se tornou a plataforma de preferência da comunidade acadêmica, surgindo diversas soluções comerciais. Para controle adequado de robôs móveis é necessário estimar a pose do robô. Uma técnica de estimação abordada em muitos trabalhos é o SLAM (localização e mapeamento simultâneo). Neste trabalho, é realizado um estudo sobre técnicas de SLAM para estimação de pose de robôs móveis aéreos multirrotores em voo *indoor*, usando como sensores uma IMU (unidade de medição inercial) de baixo custo em configuração *strapdown* e uma câmera monocular. Especificamente foi considerado o problema de estimação e controle de um quadrotor *AR.Drone* da empresa Parrot, sem a necessidade de modificação do *firmware* de fábrica. A técnica de SLAM desenvolvida nesse trabalho utiliza um filtro de Kalman desenvolvido pelo autor a detecção de padrões impressos utilizando o *software* de código aberto ARToolkit, assim, o problema de SLAM é simplificado. O método foi implementado em ROS (*Robot Operating System*) e toma proveito dos recursos do *firmware* de fábrica do *AR.Drone*. Por fim, foi avaliada a aplicabilidade da técnica como fonte de realimentação para um controlador de posição.

Palavras chave: Robótica aérea, quadrotor, SLAM, câmera monocular, controle.

Sumário

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API – Interface do processo de aplicação

ASVs – Veículos sobreaquáticos autônomos

AUVs – Veículos subaquáticos autônomos

DGPS – GPS diferencial

ENU – *East north up coordinate frame*

GPGPU – *General Purpose Graphics Processing Unit*

GPS –Sistema de posicionamento global

IMU – Unidade de medição inercial

MAV – Veículo aéreo de escala micro

MEMS –Sistemas micro eletro-mecânicos

NED – *North east down coordinate frame*

RGB-D – *Red Green blue, depth*

ROS – *Robot operating system*

RTK – *Real time kinematics*

SDK – Kit de desenvolvimento de software

SLAM – Localização e mapeamento simultâneo

UAVs – Veículos aéreos não tripulados

UGVs – Veículos terrestre não tripulados

WLAN – Rede sem fio de área local (wi-fi)

WAAS – *Wide area augmentation system*

VTOL – Decolagem e voo vertical

LISTA DE SÍMBOLOS

$S(x)$ – Matriz antissimétrica do vetor $x \in R^3$.

$vex(S(x))$ – Operação inversa de $S(x)$.

SO(3) – Grupo ortogonal especial.

{E }, {B } – Referências da terra e de corpo rígido, consecutivamente.

α, β e γ – Ângulos de Euler para a primeira, segunda e terceira rotação consecutivamente.

θ_y , θ_p e θ_r – Ângulos de roll, pitch e yaw.

$(\theta, \hat{e})$ – Representação da rotação em eixo $\hat{e}$ e ângulo θ.

v_E – Vetor v em coordenadas de {E}.

v_B – Vetor v em coordenadas de {B}.

r – Vetor em coordenadas de {E}.

b – Vetor r em coordenadas de {B}.

R – Matriz de rotação do referencial {E } para o referencial {B }.

${}^{V}R_B$ – Matriz de rotação do referencial {V } para o referencial {B }.

$Rx\ Ry\ Rz$ – Matrizes de rotação nos eixos x, y, e z.

${}^{E}p$ – Ponto p em coordenada de {E}.

$\tilde{p}$ – Forma homogênea do ponto p.

${}^{A}T_B$ – Matriz de transformação homogênea de {A} para {B}.

${}^{A}t_B$ – Vetor de translação de {A} para {B}.

$p_i = (x_i, y_i)$ – Ponto no plano da imagem.

(u,v) – Ponto no plano da imagem em coordenada de pixel.

(u_o , v_o) – Centro da imagem.

ρ_w e ρ_h – Largura e altura dos pixels.

F – Distancia focal da câmera.

C – Matriz de intrínsecos da câmera.

$X \sim N(\mu, \Sigma)$ – Distribuição normal de media μ e matriz de covariância Σ.

$g()$ – Função de predição.

ε_t – Erro de predição.

Σ_ε – Matriz de covariância do erro de predição.

h()– Função de observação.

Z_t – Observação do filtro de Kalman.

δ_t – Erro de observação.

Σ_δ – Matriz de covariância do erro de observação.

G, H, K – Matrizes do filtro de Kalman.

$\{V\}$ – Sistema de coordenada com origem na origem de {B}, mas com eixos x e y paralelos ao solo.

${}^{V}v$ – Velocidade linear em coordenada de {V}.

${}^{Vx}v$ – Velocidade linear na direção do eixo x do sistema de coordenada {V}.

$p(t) = [\, p_x(t) \;\; p_y(t) \;\; p_z(t)]^T$ – Posição da aeronave no espaço.

1 INTRODUÇÃO

Veículos aéreos não tripulados (VANTs) ou *unmaned aerial vehicles (UAVs)* são aeronaves que não possuem pilotos humanos, podendo ser pilotadas remotamente, ou serem autônomas (ou possuírem algum grau de autonomia). A utilização de veículos aéreos não tripulados vem crescendo a cada dia, principalmente no âmbito militar, desde sua primeira utilização no inicio do século 20. A princípio estas aeronaves eram utilizadas como alvos aéreos de teste e projeteis guiados. Com o desenvolvimento de tecnologias de telecomunicações, aviônica (eletrônica embarcada nas aeronaves), e de fotografia e vídeo, foram desenvolvidas aeronaves que podem ser usadas em diversas missões militares e também civis. Na maioria dessas missões o fornecimento de imagens aéreas é o principal objetivo da aeronave. Tais aeronaves podem ser usadas, dentre outras, nas seguintes missões:

- Missões de combate;
- Reconhecimento de território;
- Apoio em busca e salvamento;
- Transporte e logística;
- Apoio de missões em terra;
- Vigilância e patrulha;
- Mapeamento (aerofotogrametria) e levantamento topográfico;
- Inspeção de equipamentos e construções;
- Inspeção de áreas perigosas;
- Coleta de dados meteorológicos;
- Cinema e jornalismo;
- Exploração espacial em planetas com atmosfera.

As características construtivas da aeronave devem ser projetadas levando em consideração a finalidade da aeronave. Algumas das características a serem consideradas são: o tipo de aeronave (asas fixas, asas rotativas, aeróstato etc.), a aerodinâmica, materiais construtivos e a propulsão (tipo de motor utilizado). Segundo Melo (2010), a miniaturização de dispositivos eletromecânicos, o melhoramento na relação carga-massa das baterias e o controle de voo automático para pequenas aeronaves (mini/micro) são fatores que contribuíram para a evolução dos VANTs elétricos. Com isso se popularizaram os veículos

aéreos de escala micro/mini (MAVs), sendo possível realizar voos em interiores (*indoor*) e em espaços restritos.

Em algumas das missões descritas, é vantajoso, e em alguns casos é necessário, que o VANT seja autônomo. Com isso, muitos trabalhos foram desenvolvidos buscando conferir autonomia a veículos aéreos não tripulados. Tais trabalhos utilizam de conceitos de robótica móvel e teoria de controle, sendo inseridos na área de robótica aérea.

A robótica aérea é uma subárea da robótica móvel que trata do projeto de criação, implementação, instrumentação, inteligência e controle de dispositivos robóticos capazes de locomoção aérea. A robótica móvel, por sua vez, e uma subárea da robótica que trata de problemas inerentes a robôs moveis, como por exemplo, localização e orientação no espaço, mapeamento tridimensional do espaço, planejamento de trajetória, identificação e desvio de obstáculos, cooperação entre robôs, dentre outros problemas. Robôs móveis são geralmente autônomos ou são dotados de um certo grau de autonomia, sendo que robôs com alto grau de autonomia necessitam de pouca ou nenhuma intervenção humana. Segundo Mahony, Kumar e Corke (2012), essa área da robótica tem despertado grande interesse da comunidade cientifica e muitos trabalhos foram gerados nos últimos anos. Dentre os trabalhos em robótica aérea, o quadrotor foi a plataforma mais utilizada.

Segundo Asada e Slotine (1986), as primitivas da robótica (funções básicas para o controle do robô) são sentir (*sense*), planejar (*plan*) e agir (*act*). Isso significa que um robô autônomo deve ter meios de adquirir informação do ambiente por meio de sensores, planejar suas ações e atuar sobre o meio.

Os sensores utilizados para adquirir informação do ambiente são ruidosos ou medem as informações indiretamente. Para solucionar esses problemas são utilizadas técnicas de fusão de sensores, para desenvolver estimadores de estados do robô e extrair informações do ambiente, como por exemplo o mapa do ambiente e obstáculos presentes (estáticos ou móveis). Fusão de sensores é uma área do conhecimento que trata a integração de informações sobre uma mesma variável, vindas de fontes diferentes.

A primitiva do sentir é de grande importância para as outras primitivas. No âmbito da primitiva do agir, as informações dos estimadores de estados são utilizadas para realimentação dos controladores. No âmbito do planejamento, por exemplo, o mapa do

ambiente é utilizado para o planejamento de trajetória, e informações sobre obstáculos são usadas para estratégia de desvio de obstáculos.

1.1 O QUADROTOR E O *AR.DRONE*

O quadrotor é uma aeronave de asas rotativas multirrotor que possui quatro rotores. Esse tipo de aeronave foi proposta na década de 1920 e foi um dos primeiros conceitos bem sucedidos de aeronaves de decolagem e pouso vertical (LEISHMAN, 2000). Na época, a eletrônica e os sistemas de controle não estavam avançados o suficiente o que exigia muito esforço do piloto para manter a aeronave estabilizada. Por ser mais fácil de ser controlado manualmente, o helicóptero foi o mais aceito pelo mercado como aeronave de decolagem e pouso vertical entre as aeronaves tripuladas, até hoje. Nas últimas décadas foram desenvolvidas soluções de controle e estabilização automático para aeronaves multirrotores, popularizando o uso dessas aeronaves em voos não tripulados.

Figura 1: Aeronave multirrotor de oito rotores.
Fonte: http://www.unmannedsystemstechnology.com/2012/06/new-video-shows-aerobot-cinestar-vtol-uav-in-action/cinestar/

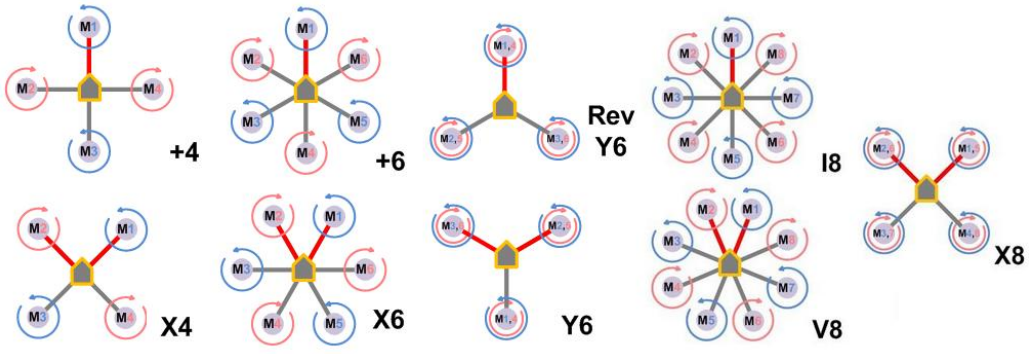


Figura 2: Possíveis configurações de aeronaves multirrotores com quatro, seis ou oito rotores. As setas em azul indicam rotores que giram no sentido anti-horário e as setas em rosa indicam rotores que giram no sentido horário.
Fonte: http://wiki.dji.com/en/index.php/Multi-rotor_Aircraft

Dentre as vantagens que as aeronaves multirrotores apresentam em comparação com helicópteros de mesma escala pode-se citar:

1. é mecanicamente mais simples, dispensando o prato cíclico que varia a inclinação das hélices e controla o movimento linear do helicóptero;
2. usar uma quantidade maior de rotores permite diminuir o diâmetro desses, o que diminui a energia cinética armazenada no movimento de rotação. Isso diminui os riscos caso as hélices batam em algum objeto, sendo mais seguro para voos *indoor*;
3. são mais ágeis (maior aceleração), são mais velozes (alcançam maiores velocidades) e possuem maior manobrabilidade, podendo realizar manobras acrobáticas como *flips* etc;

Por esses motivos, entre outros, a comunidade acadêmica adotou o quadrotor como plataforma padrão para pesquisa em robótica aérea (MAHONY, KUMAR & CORKE, 2012) principalmente para veículos aéreos de escala micro (MAVs).

Os quatro rotores do quadrotor são montados em uma estrutura em cruz, equidistantes do centro de massa. Os eixos de rotação dos rotores são alinhados de forma a exercer força ortogonal ao plano da estrutura, conforme a Figura 3. Dois rotores giram no sentido horário enquanto os outros dois giram no sentido anti-horário, cancelando o torque induzido. Essas aeronaves têm capacidade de decolagem e pouso verticais (VTOL), voo vertical e de pairar no ar. Essas capacidades concedem uma alta versatilidade e manobrabilidade à aeronave, podendo ser essencial para algumas missões, principalmente em missões que exigem voo *indoor*.

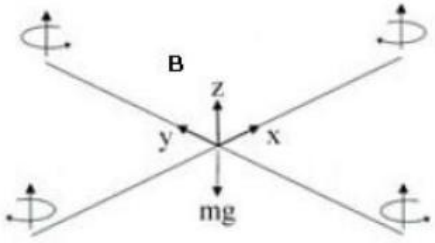


Figura 3: Diagrama construtivo de um quadrotor
Fonte: (BOUABDALLAH, S., MURRIERI., P & SIEGWART, R., 2004).

Segundo Mahony, Kumar e Corke (2012), a maioria dos quadrotores é equipado com uma unidade de medição inercial (IMU) e com um instrumento de medida de altura,

podendo este ser acústico, por infravermelho, barométrico, por laser, ou outro tipo. A IMU consiste de dispositivos eletromecânicos microscópicos (MEMS): um acelerômetro triaxial, um giroscópio triaxial e um magnetômetro triaxial. Com estes sensores é possível estimar os ângulos de atitude da aeronave.

É possível utilizar também outros sensores. Alguns dos sistemas mais frequentemente utilizados são: *scanner* de ambiente com medidor de distancia a laser, câmeras instaladas na aeronave, sistema baseados em imagem como VICON[1], sistema de posicionamento global (GPS) e triangulação de sinais. Alguns desses sistemas são externos, limitando a aplicabilidade e autonomia do robô. Buscando conferir autonomia em qualquer ambiente, muitos trabalhos foram desenvolvidos focando na utilização de sensores embarcados no robô, sendo geralmente utilizadas câmeras.

Existem muitas soluções comerciais e até soluções de código e *hardware* abertos para se construir um quadrotor. Entre as empresas que comercializam essas soluções pode-se citar *Parrot, 3D Robotics, Dji, Draganfly* etc. O *AR.Drone* é um quadrotor lançado pela empresa *Parrot* para o público geral, com a finalidade de entretenimento (BRISTEAU *et al.* , 2011). A aeronave pode ser pilotada através de um aplicativo para celulares e aparelhos móveis utilizando uma conexão Wi-Fi. A plataforma atraiu a atenção da comunidade acadêmica por ser uma alternativa de baixo custo em comparação a outras plataformas existentes no mercado e possuir um sistema de sensores e *firmware* que implementa alguns controles básicos. Recentemente foi lançada a versão *AR.Drone* 2.0 com algumas melhorias, como a substituição de sensores por outros mais sensíveis, substituição de câmeras por outras de maior resolução etc.

Segundo o fabricante[2], o *AR.Drone* possui uma câmera frontal e uma câmera vertical (apontada para baixo), e as imagens podem ser enviadas pela conexão Wi-Fi. O *firmware* também utiliza as imagens para fazer estimação de velocidade linear da aeronave. Possui também uma unidade de medição inercial (IMU) com giroscópio de 3 eixos, e acelerômetro de 3 eixos, (na versão 2.0 possui também magnetômetro de 3 eixos), sensor de ultrassom para medir a altitude (na versão 2.0 possui também um barômetro para melhorar a estimação de altitude). Para acionamento, possui 4 rotores com motores *brushless* e caixa de

[1] Em: <http://www.vicon.com/>. Acesso em 22 janeiro 2014.
[2] Em: < http://ardrone2.parrot.com/>. Acesso em 22 janeiro 2014.

redução, *drivers* e circuitos de controle de velocidade dos rotores. Para processamento possui um processador ARM córtex A8 32 bits de 1GHz com placa de processamento digital de sinal de vídeo de 800MHz, com sistema operacional de tempo real baseado em Linux. A versão 2.0 ainda suporta um receptor de GPS externo que pode ser conectado em uma porta USB do *AR.Drone*.

Segundo Bristeau *et al.* (2011), o *firmware* integrado executa algoritmos de estimação de estado do quadrotor (estimação de velocidade linear e estimação de atitude), de controle e estabilização, de calibração dos sensores, de proteção contra situações de emergência, e de reconhecimento de determinados padrões nas imagens das câmeras. O *firmware* também conta com uma API[3] para comunicação com aplicativos externos, como os aplicativos oficiais do fabricante para Android e iOS.

Contudo, o *firmware* do *AR.Drone* é fechado, o que limita a implementação de alguns algoritmos. O *firmware* original não permite o acesso direto aos motores, e aceita como entrada de controle através de sua API apenas referências de ângulos de pitch e roll, velocidade angular de yaw e velocidade vertical.

Existem alternativas de código aberto para substituir o *firmware* original. (PERQUIN, 2011) criou um *firmware* de código aberto para o *AR.Drone* 1.0 onde é implementada a estimação e estabilização de atitude. Em 2012, alunos da universidade de tecnologia *DELFT* portaram o sistema *PAPARAZZI* de voo autônomo para o *AR.Drone*, implementando um controle de posição baseado em GPS[4]. A empresa Pixhawk comercializa o sistema de controle de voo *open-source* e *open-hardware* PX4, que pode substituir toda a eletrônica do *AR.Drone*, mantendo apenas os motores e controladores dos motores e a estrutura física[5].

Caso se deseje utilizar o *firmware* original, o fabricante do *AR.Drone* disponibiliza um kit de desenvolvimento de software (SDK) para desenvolvimento de aplicações utilizando a API original.

Existem também algumas soluções utilizando o ROS (*robot operating system*) que é um *framework* para programação de robôs desenvolvido pela empresa *Willow Garage*. Em

[3] Interface de programação de aplicativos
[4] Em: < http://wiki.paparazziuav.org/wiki/TU_Delft_-_Autonomous_Quadrotor>. Acesso em 22 janeiro 2014
[5] Em: < http://pixhawk.org/platforms/multicopters/ar_drone>. Acesso em 22 janeiro 2014

2012 uma equipe do laboratório *Autonomy Lab* da universidade de Simon Fraser no Canadá lançou um pacote para ROS que serve de *driver* para aplicações ROS com o *AR.Drone*. Esse *driver*, chamado de *AR.Drone autonomy*[6], é baseado na SDK oficial do *AR.Drone*, e utiliza o *firmware* original embarcado no *drone*. Em 2012 uma equipe da Universidade Técnica de Munique (TUM) lançou um pacote para ROS que utiliza o *driver AR.Drone autonomy* e implementa um sistema de estimação de posição e controle baseado em navegação visual[7].

Para esse trabalho, foi utilizado o *driver AR.Drone autonomy*, e desenvolvido um pacote para ROS para implementar os algoritmo de SLAM e controle da pose do quadrotor. A escolha foi feita por haver uma documentação melhor detalhada deste *driver*.

Por vezes a aeronave será referida como corpo rígido quase como um sinônimo, visto que a representação da pose e os algoritmos de SLAM podem ser aplicados em qualquer corpo rígido orientado a partir de um referencial fixo, como por exemplo AUVs (veículos subaquáticos autônomos), ASVs (veículos sobreaquáticos autônomos), UGVs (veículos terrestre não tripulado), ligações (*links*) de robôs articulados, peças sendo manipuladas pelo robô etc.

1.2 O PROBLEMA DE LOCALIZAÇÃO E MAPEAMENTO (SLAM)

Existem muitos sensores e métodos utilizados para realizar a localização de um objeto no espaço tridimensional. O mais comum e mais conhecido sistema de localização é o GPS que é um sistema de navegação por satélite. Para os receptores GPS mais comuns e de uso civil, o erro de localização é da ordem de 5 a 15 metros e pode ser agravado por fatores atmosféricos, reflexões do sinal de GPS e obstrução do sinal por copas de árvores e outras estruturas[8]. Além disso existe um atraso para recepção do sinal e uma baixa frequência de estimação da posição. Alguns sistemas também podem ser utilizados para diminuir o erro de localização do GPS como o sistema RTK (*real time kinematic*), DGPS (*Differential Global Positioning System*) e o sistema WAAS (*wide area augmentation system*). Ainda assim, esses sistemas não estão disponíveis em todos os países. Em sistemas RTK e DGPS o receptor deve estar perto de uma estação base para receber um sinal de correção da

[6] Em: < http://wiki.ros.org/ardrone_autonomy>. Acesso em 22 janeiro 2014.
[7] Em: < http://wiki.ros.org/tum_ardrone>. Acesso em 22 janeiro 2014.
[8] Em: < http://en.wikipedia.org/wiki/Global_Positioning_System>. Acesso em 23 janeiro 2014.

localização[9], e o sistema WAAS cobre apenas a America do Norte[10]. Em ambos os sistemas também é necessário uma visada limpa ou pouco obstruída dos satélites (ou do céu), assim o sinal de GPS não é acessível no interior de casas, edifícios, túneis e cavernas.

Em ambientes onde o sinal de GPS não é acessível pode-se utilizar um conjunto de estações base de localizações conhecidas, emitindo um sinal que pode ser eletromagnético ou acústico, e realizando uma técnica de trilateração (localização pela medição da distância do receptor para cada uma das estações base) ou *resectioning* (localização pela medição dos ângulos de orientação em relação às estações base)[11]. Existem muitas soluções comerciais geralmente utilizados em aplicações industriais (como o iGPS da NIKON[12]) e aplicações subaquáticas (como os sistemas de posicionamento acústico da SONARDYNE[13]). Contudo, esses equipamentos são caros e não permitem a localização do robô em ambientes desconhecidos, sendo limitados à operação no ambiente de trabalho.

Outro sistema muito utilizado, principalmente em trabalhos com quadrotores, (KLEIN & MURRAY, 2007), é um sistema externo de captura de movimento por imagem (como o sistema comercial VICON). Esse sistema é bastante preciso e rastreia a posição de marcadores especiais fixados no robô utilizando os princípios de visão computacional descritos na seção 2.4. Por ter alta precisão e frequência de amostragem, os trabalhos que utilizam esse sistema são focados em controle de precisão com resultados impressionantes como o equilíbrio de um pêndulo invertido sobre a aeronave (HEHN & D'ANDREA, 2011), trajetórias agressivas sobre pequenas aberturas (MELLINGER & KUMAR, 2011), e até jogar ping-pong e fazer malabarismo com bolas (MÜLLER, LUPASHIN, & D'ANDREA, 2011). Porém, como o sistema de captura é externo, se restringe ao uso em laboratório e não é apropriado para realizar tarefas em outros ambientes, como por exemplo a exploração de ambientes desconhecidos.

Pensando nisso uma grande parte da comunidade científica aborda o problema de localização utilizando apenas sensores embarcados no robô. A técnica mais simples e intuitiva é a de *deadreckoning* que é a integração das estimações da velocidade linear. A velocidade linear de uma aeronave pode ser estimada com os sensores da IMU ou com

[9] Em: < http://en.wikipedia.org/wiki/Differential_GPS>. Acesso em 23 janeiro 2014
[10] Em: < http://en.wikipedia.org/wiki/GNSS_augmentation> Acesso em 23 janeiro 2014.
[11] Em: < http://en.wikipedia.org/wiki/Local_positioning_system>. Acesso em 23 janeiro 2014.
[12] Em: < http://www.nikonmetrology.com/en_US/>. Acesso em 23 janeiro 2014.
[13] Em: < http://www.sonardyne.com/products/positioning.html>. Acesso em 23 janeiro 2014.

câmeras utilizando técnicas de odometria visual como a técnica *optical flow*. Pode-se também empregar técnicas de fusão de sensores para melhorar a estimação usando ambos sensores IMU e câmera (BRISTEAU ET AL., 2011). Porém, a integração de uma medição de velocidade com erro e ruído de medição, gera um desvio e aumento ilimitado do erro com o tempo.

Para eliminar esse problema é necessário cruzar as informações dos sensores com um mapa previamente armazenado no robô ou fazer o mapeamento e localização simultâneo (SLAM). Para isso geralmente é utilizado, em conjunto com a IMU, um sistema de imageamento como câmeras monoculares (BRISTEAU *et al*., 2011), câmera estéreo, sensores RGB-D como o *kinect* (FALLON, JOHANNSSON & LEONARD, 2012), *laser range scanners* (WEI, CAPPELLE, & RUICHEK, 2011) e *sonar scanners* (DIOSI & KLEEMAN, 2004).

Os sonares são muito pesados e não são usados em robôs aéreos de pequenas dimensões e os laser range scanners consomem muita energia, o que diminui o tempo de duração da bateria (BILLS, CHEN, & SAXENA, 2011). O sistema RGB-D consegue medir a distância do sensor para superfícies reflexivas (não funciona com vidros transparentes e espelhos), utilizando uma câmera RGB e uma câmera infravermelha. As imagens da câmera infravermelha são usadas para calcular a distância dos objetos na imagem utilizando um projetor laser infravermelho que projeta um padrão de malha estruturada no objeto, gerando uma nuvem de pontos tridimensionais que representam a imagem. Essa técnica é chamada de *structured light 3d scanning* (ZHANG, CURLESS, & SEITZ, 2002). O algoritmo de localização utiliza a nuvem de pontos adquirida pelo sensor e compara com um mapa ou constrói o próprio mapa do ambiente (SCHERER & ZELL, 2013).

As câmeras estéreo conseguem localizar pontos de interesse no espaço 3d por técnicas de extração de características (*features*) como o algoritmo *FAST corner detector* e estimação de profundidade por minimização dos erros das restrições geométricas. Sabendo a *baseline* (distância) exata entre as duas câmeras do conjunto estéreo (*stereo rig*) é possível modelar um conjunto de restrições relacionadas com a posição onde determinada ponto no espaço será projetado nas imagens (restrições epipolares). Assim, sabendo a posição de determinada *feature* nas duas imagens do conjunto estéreo é possível estimar a posição daquele ponto no espaço 3d. Os pontos de interesse (*features*) geralmente são pontos de alto

contraste como quinas e arestas, assim, o ambiente a ser capturado deve ser estruturado, para que a câmera consiga captar vários pontos de interesse. Ambientes muito homogêneos e de baixo contraste, são problemáticos e o algoritmo não terá uma boa precisão. Assim como no RGB-D o sensor forma uma nuvem de pontos. O algoritmo de localização compara essa informação com um mapa ou constrói o próprio mapa dos pontos de interesse (LEMAIRE *et al*., 2007)

Uma técnica parecida pode ser utilizada com câmeras monoculares. Utilizando câmeras estéreo se tem duas imagens do ambiente tiradas com uma distância (*baseline*) conhecidas entre elas. Com câmeras monoculares também pode-se obter imagens estéreo simplesmente transladando a câmera, porém, nesse caso não se conhece a *baseline*. Caso seja possível identificar um conjunto de pontos de interesse nas duas imagens obtidas de posições diferentes (pelo menos cinco pontos), é possível estimar a pose relativa entre as imagens e estimar também a profundidade dos pontos de interesse. Essa técnica é conhecida como *structure from motion* e surgiu como uma técnica off-line para obter informação tridimensional apartir de imagens. Baseado nessa técnica e utilizando técnicas de computação paralela e GPGPU (*General Purpose Graphics Processing Unit*) alguns algoritmos foram desenvolvidos para realizar esse procedimento em tempo real, e assim aplicar em robótica. Essa técnica apresenta um problema por não ser possível determinar a escala, ou seja, tanto a pose relativa quanto a profundidade estimada dos pontos é dada em função de um termo de escala normalizado. Para resolver o problema de SLAM utilizando câmera monocular é necessário utilizar sensores adicionais como a IMU, sensores ultrassônicos, ou até mesmo GPS, para estimar a escala do mapa. A estimação é feita por uma técnica de fusão de sensores, geralmente um filtro de Kalman ou uma variação do mesmo, como o filtro de Kalman estendido (ENGEL, STURM, & CREMERS, 2012). Apesar de necessitar de outros sensores e de um algoritmo de estimação de escala, a utilização de câmera monocular para SLAM tem como vantagem o menor peso e menor consumo de bateria que os outros sensores listados.

Uma técnica alternativa de localização por câmera monocular que não utiliza a extração de características utiliza técnicas de reconhecimento de padrões para identificar *landmarks* (também chamados de fiduciais) de escala previamente conhecida (EBERLI *et al*., 2010). Dessa forma sabendo a escala do objeto no mundo real e o tamanho e posição do

objeto na imagem é possível saber a localização do objeto em relação à câmera. Essa é uma forma mais simples e que simplifica muitos dos problemas das técnicas anteriores. Por exemplo, não é necessário ter duas imagens para estimar a profundidade dos pontos, nem é necessário identificar e rastrear pontos entre as duas imagens, sendo possível estimar a profundidade com apenas uma imagem. Porém essa técnica depende da distribuição e conhecimento prévio das *landmarks* e o mapa gerado não representa todos os objetos e obstáculos do ambiente. Porém, por ser mais simples e exigir menos esforço computacional, esta técnica será utilizada para desenvolvimento do algoritmo de localização.

1.3 ESTRUTURA DO TRABALHO

Nesse trabalho, é feito um levantamento das técnicas de SLAM utilizando como sensor principal uma câmera monocular. Como base para o desenvolvimento do trabalho é utilizada o quadrotor *AR.Drone* do fabricante PARROT. São estudados métodos de estimação da pose da aeronave em relação a um referencial fixo na terra e a aplicabilidade desses métodos como realimentação de controle de atitude e posição da aeronave. É feita a implementação de um método que utiliza o reconhecimento de padrões impressos utilizando o *software* de código aberto ARToolkit e um filtro de Kalman desenvolvido pelo autor. O método implementado parte das ferramentas embarcadas no *firmware* do *AR.Drone* não sendo necessário trocar o *firmware* de fábrica por um de código aberto.

No Capítulo 2, é feita uma revisão das ferramentas matemáticas necessárias para o desenvolvimento do trabalho. Nesse capítulo são apresentados conceitos gerais sobre matemática e sistemas dinâmicos. Formas de representação de atitude e pose de corpo rígido, bem como as equações da dinâmica de translação e rotação. Também são apresentados conceitos de visão computacional.

No Capítulo 3, é apresentado a teoria do filtro de Kalman e o desenvolvimento de um método de SLAM utilizando o filtro de Kalman para mapear marcos artificiais (padrões previamente conhecidos e impressos em folhas de papel. Este método utiliza o *software* de código aberto ARToolkit e simplifica bastante o problema de SLAM. O algoritmo desenvolvido é aplicado como fonte de realimentação para um controlador de pose do quadrotor e seu desempenho é avaliado.

No Capitulo 4, são apresentadas as conclusões e perspectivas de trabalhos futuros.

2 FUNDAMENTOS MATEMÁTICOS

Nesse capítulo, será revisada a base matemática que será usada no desenvolvimento e análise das leis de controle e estimação. Na seção 2.1 é apresentado o conceito de matriz antissimétrica. Na seção 2.2 são descritas as formas de representação de atitude de corpo rígido e as operações algébricas que se pode fazer sobre essas representações. Na seção 2.3, são descritas as formas de representação da pose de um corpo rígido e na seção 2.4, são apresentados conceitos de visão computacional.

2.1 MATRIZ ANTISSIMÉTRICA

Uma matriz antissimétrica é uma matriz quadrada A tal que $A^T = -A$. Desta forma, os elementos da diagonal principal de uma matriz antissimétrica são nulos.

Seja o vetor $x = [x_1 \quad x_2 \quad x_3]^T \in R^3$, a matriz antissimétrica $S(x)$ associada ao vetor x é dada por:

$$S(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}. \tag{2.1}$$

A operação inversa é dada pelo operador $vex(\cdot)$, que retorna o vetor associado a matriz antissimétrica. Seja a matriz $A = S(x)$, então $vex(A) = x$ e $S\big(vex(A)\big) = A$.

Dessa forma algumas propriedades podem ser traçadas:

$$S(x)^T = -S(x), \tag{2.2}$$

$$S(x)^2 = xx^T - x^T x\, I, \tag{2.3}$$

$$\lambda(S(x)^2) = [0 \quad -\|x\|^2 \quad -\|x\|^2]^T, \tag{2.4}$$

sendo I é a matriz identidade, $\lambda(A)$ é o vetor de auto valores da matriz quadrada A.

Seja $y \in R^3$ e $x \times y$ é o produto vetorial de x e y, então também valem as seguintes propriedades:

$$S(x)y = -S(y)x = x \times y, \tag{2.5}$$

$$S(x)\mathrm{x} = [0\ 0\ 0]^{\mathrm{T}}, \tag{2.6}$$

$$S(Rx) = \mathrm{R}S(x)\mathrm{R}^{\mathrm{T}}, \mathrm{R} \in \mathrm{SO}(3), \tag{2.7}$$

onde $SO(3)$ é o grupo de rotação definido por $SO(3) := \{ R \in R^{3\times3} | \, det(R) = 1 | \, R^T R = RR^T = I_{3x3} \}$, que será descrito nas seções seguintes. Também valem:

$$x^T y = \frac{1}{2} traço(\, S(x)^T \, S(y) \,), \tag{2.8}$$

$$traço(S(x)\, B) = 0, \tag{2.9}$$

onde $traço\ (A)$ é a soma dos elementos da diagonal principal da matriz A e B é uma matriz simétrica, $B \in R^{3\times3}$, $B = B^T$.

2.2 REPRESENTAÇÃO DA ATITUDE DE CORPO RÍGIDO

Para representar a atitude da aeronave (corpo rígido) serão utilizados dois sistemas de coordenadas (referenciais inerciais):

- Referencial inercial da terra {E} : Um referencial fixado em determinado ponto da superfície da terra. Esse referencial pode ser em coordenada NED (eixo $\boldsymbol{x}$ voltado para o norte, eixo $\boldsymbol{y}$ voltado para o leste e eixo **z** voltado para baixo na direção da aceleração da gravidade) ou em coordenada ENU (eixo **x** voltado para o leste, eixo $\boldsymbol{y}$ voltado para o norte e eixo $\boldsymbol{z}$ voltado para cima na direção contrária à aceleração da gravidade)
- Referencial do corpo rígido {B} : um referencial fixado ao centro de gravidade da aeronave. O referencial pode ser fixado de forma que a base ortonormal tem eixo $\boldsymbol{x}$ voltado para a frente da aeronave, eixo $\boldsymbol{y}$ voltado para o lado direito da aeronave e eixo **z** voltado para baixo na direção contraria ao impulso (*thrust*) da aeronave, ou de forma que o eixo $\boldsymbol{x}$ seja voltado para frente da aeronave, o eixo $\boldsymbol{y}$ voltado para o lado esquerdo da aeronave e eixo $\boldsymbol{z}$ voltado para cima na direção do impulso da aeronave.

Em muitas das aplicações envolvendo aeronaves são utilizados os referenciais com o eixo **z** voltado para baixo, porém, nesse trabalho serão utilizados os referenciais com o eixo **z** voltado para cima.

A atitude é a representação da orientação do corpo rígido {B } em relação ao referencial da terra {E}, sem considerar a translação. Serão apresentados diferentes descrições matemáticas para representar tal orientação.

2.2.1 Ângulos de Euler

Os Ângulos de Euler foram introduzidos por Leonhard Euler para representar uma sequencia de 3 rotações elementares sobre os eixos de um sistema de coordenada, dessa forma pode-se representar qualquer orientação de um corpo rígido em relação a um referencial. As rotações podem ser sobre os eixos do referencial global fixo (rotação extrínseca), ou no referencial do corpo rígido, que muda a cada rotação elementar, inicialmente alinhado ao referencial global (rotação intrínseca). Existem 12 possíveis sequencias de eixos de rotação, também chamadas de convenções, seja para rotação extrínseca ou intrínseca, que são divididas em 2 grupos:

- Ângulos de Euler (ou Ângulos de Euler clássicos) = z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y
- Ângulos de Tait-Bryan = x-y-z, y-z-x, z-x-y, x-z-y, z-y-x, y-x-z

Ângulos de Tait-Bryan também são chamados de ângulos de Cardan, ângulos náuticos, *heading, elevation e bank,* ou ***yaw, pitch*** e ***roll,*** sendo que esses últimos são também os eixos da aeronave (eixos z, y e x do referencial {B }). A notação dos ângulos de Euler geralmente é α, β e γ para a 1ª 2ª e 3ª rotação consecutivamente, independente da convenção utilizada. Observe que para convenções diferentes a representação de uma mesma rotação irá ter valores de ângulos diferentes. Uma convenção muito utilizada na literatura é a $z - y - x$ intrínseca e os ângulos de Euler nesse caso são comumente chamados de yaw, pitch e roll (ou θ_y , θ_p e θ_r). Também como convenção, os ângulos positivos seguem a regra da mão direita.

Para qualquer convenção, se os ângulos da primeira e terceira rotação forem contidos em intervalos de range igual a 2π, como por exemplo $[-\pi, \pi]$, e o ângulo da segunda rotação for contido em um intervalo de range igual a π, como por exemplo no intervalo $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, pode-se determinar qualquer rotação. Também, qualquer rotação pode ser

representada por uma única solução de valores dos ângulos nesses intervalos, com exceção dos casos de singularidade. Nas convenções de Ângulos de Euler clássicos, a singularidade ocorre para $\beta = n\pi\,, n = 0,1,2,3$ ... Quando n é par tem-se que $(\alpha + \gamma)$ tem solução única, mas α e γ tem infinitas soluções, já nos casos em que n é impar tem-se que $(\alpha - \gamma)$ tem solução única, mas α e γ tem infinitas soluções. Nas convenções de ângulos de Tait-Bryan a singularidade ocorre para $\beta = \frac{(n+1)\pi}{2}$.

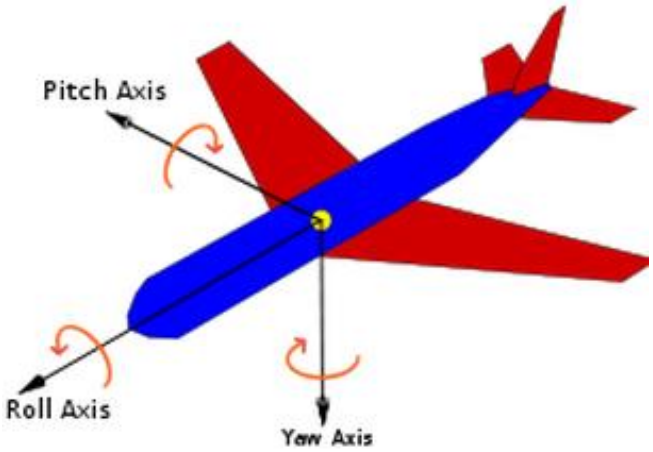


Figura 4: Ângulos yaw pitch e roll de uma aeronave
Fonte: Wikipédia, disponível em: http://en.wikipedia.org/wiki/Tait-Bryan_angles

As singularidades podem causar problemas de *gimbal lock* em giroscópios giratórios[14] e punhos de três graus de liberdade em robôs manipuladores. Por isso deve-se ter cuidado ao utilizar essa representação. No caso da maioria das aeronaves, a singularidade representa uma atitude incomum (pitch = 90º, para a convenção $z - y - x$ intrínseca), que dificilmente será alcançada durante um voo (CORKE, 2011).

2.2.2 Eixo-ângulo

Essa representação parametriza a rotação de um corpo rígido usando um vetor unitário $\hat{e} \in R^3\,, \|\hat{e}\| = 1$, escrito nas coordenadas do referencial global que representa o eixo de rotação do corpo rígido, e um ângulo θ que representa a magnitude da rotação sobre o eixo, seguindo a regra da mão direita. Também pode ser parametrizado na forma mais compacta de vetor de rotação e = $\theta\hat{e}$

[14] Para mais informações consultar o apêndice A.

Para se calcular o vetor resultado de uma rotação ou o mapeamento de um vetor para uma outra base de coordenadas pode-se usar a formula de Rodrigues[15]:

$$v_B = v_E \cos\theta + (\hat{e} \times v_E)\sin\theta + \hat{e}(\hat{e} \cdot v_E)(1 - \cos\theta). \quad (2.10)$$

Nessa representação, uma rotação descrita por $(\theta, \hat{e})$ equivale as rotações descritas por $(2\pi n - \theta, -\hat{e})$ e $(\theta + 2\pi n, \hat{e})$, $n \in Z$. Se o valor do ângulo de rotação for limitado em um intervalo com range igual a π, como por exemplo $[0, \pi]$, qualquer rotação terá representação única (valores únicos de θ e $\hat{e}$) com exceção do caso em que $\theta = \pi$. Nesse caso o eixo de rotação tem duas soluções ($\hat{e}$ e $-\hat{e}$).

2.2.3 Matriz de rotação

A matriz de rotação, ou *direct-cosine matrix*, é uma matriz ortogonal de 3 dimensões, contida no grupo de rotação SO(3). O grupo de rotação SO(3) é definido pelas matrizes $R \in R^{3\times3}$, ortogonais $(R^T R = RR^T = I_{3x3})$, que representem uma transformação linear de um vetor no espaço euclidiano preservando o seu comprimento e sentido. Para uma matriz ortogonal $det(R^T) = det(R)$ isso implica em $det(R)^2 = 1$, e assim, $det(R) = \pm 1$. Para preservar o sentido o determinante deve ser positivo, logo:

$$SO(3) := \{ R \in R^{3\times3} \mid det(R) = 1 \mid R^T R = RR^T = I_{3x3} \} \quad (2.11)$$

Esse conjunto é utilizado pois oferece uma representação global e única (não existindo múltiplas soluções) da orientação do corpo rígido. Ou seja, a matriz de rotação é um mapeamento bijetor (um-para-um) do espaço $SO(3)$ para o espaço euclidiano de três dimensões. Esse conjunto pode ser também chamado de espaço de rotação. Uma matriz $R \in SO(3)$ pode ser usada para mapear as coordenadas de um vetor de uma base referencial para outra, por exemplo, se R descreve a rotação do referencial $\{B\}$ para o referencial $\{E\}$, então qualquer vetor r expresso em coordenadas de $\{B\}$, pode ser expresso em coordenadas de $\{E\}$ multiplicando-se:

$$r = Rb. \quad (2.12)$$

[15] Em <http://en.wikipedia.org/wiki/Rodrigues'_rotation_formula>. Acesso em 23 janeiro 2014.

Em alguns trabalhos, a convenção para multiplicação do vetor é tal que a matriz de rotação é multiplicada pela direita, conforme a equação (2.13). Essa convenção não será utilizada nesse trabalho.

$$r^T = b^T R. \tag{2.13}$$

A matriz de rotação pode ser escrita em termos dos ângulos de Euler, sendo que as rotações elementares podem ser representadas por $Rx, Ry\ e\ Rz$. Usando a convenção $z-y-x$ de rotações intrínsecas, a convenção de multiplicação pela esquerda, e ângulos de Euler α, β e γ, a matriz de rotação pode ser composta:

$$Rx = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\gamma & -sen\,\gamma \\ 0 & sen\,\gamma & cos\,\gamma \end{bmatrix}, \tag{2.14}$$

$$Ry = \begin{bmatrix} cos\,\beta & 0 & sen\,\beta \\ 0 & 1 & 0 \\ -sen\,\beta & 0 & cos\,\beta \end{bmatrix}, \tag{2.15}$$

$$Rz = \begin{bmatrix} cos\,\alpha & -sen\,\alpha & 0 \\ sen\,\alpha & cos\,\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.16}$$

$$R = Rz\,Ry\,Rx. \tag{2.17}$$

Observe que se for usada uma convenção diferente a ordem de multiplicação das matrizes irá mudar (ex.: para $z-y-x$ extrínseco, $R = Rx\,Ry\,Rz$). Porém, caso representem uma mesma rotação, duas convenções diferentes irão ter valores de ângulos de Euler diferentes, o que resultará na mesma matriz de rotação.

Para representar a orientação relativa entre vários referenciais pode-se utilizar a notação com subíndice e superíndice. Assim, a matriz ${}^{A}R_{B}$ representa a atitude de $\{B\}$ em relação a $\{A\}$, e a matriz ${}^{B}R_{C}$ representa a atitude de $\{C\}$ em relação a $\{B\}$. As matrizes podem ser associadas para representar a orientação relativa entre quaisquer referenciais. Logo, ${}^{A}R_{C} = {}^{A}R_{B}\ {}^{B}R_{C}$. Também é verdade que ${}^{B}R_{A} = \left({}^{A}R_{B}\right)^{-1} = \left({}^{A}R_{B}\right)^{T}$.

Além dessas formas de representação de atitude ainda é possivel representa-la na forma de quatérnios unitários[16]. Esta forma não será abordada nesse trabalho.

2.3 REPRESENTAÇÃO DA POSE DE UM CORPO RÍGIDO

[16] Para mais informações consultar o apêndice B

Para representar a pose de um corpo rígido em relação a um referencial deve-se levar em consideração a atitude e a translação entre os dois sistemas de coordenadas. Por isso se diz que um sistema que pode transladar nas três dimensões do espaço e rotacionar no espaço $SO(3)$, tem 6 graus de liberdade (três para a translação e três para a rotação). Para representar a translação num espaço tridimensional pode-se usar um vetor $t = (x, y, z)$ tal que para dois sistemas $\{A\}$ e $\{B\}$, sendo $\{B\}$ transladado de $\{A\}$ pelo vetor t, o ponto ${}^{A}p$ em coordenadas de $\{A\}$ satisfaz a equação:

$$ {}^{A}p = {}^{B}p + t. \tag{2.18}$$

Seja o referencial $\{B\}$ transladado de $\{A\}$ pelo vetor t, com matriz de rotação ${}^{A}R_{B}$, então o ponto ${}^{A}p = (x, y, z)$ em coordenadas de $\{A\}$ satisfaz a equação:

$$ {}^{A}p = {}^{A}R_{B}\ {}^{B}p + t. \tag{2.19}$$

A operação de translação e rotação é chamada de transformação euclidiana, e para combinar rotação e translação em uma representação de pose pode-se usar a representação em par de quatérnio e vetor, um par matriz de rotação e vetor (como na equação (2.19)), ou a matriz de transformação homogênea. A matriz de transformação homogênea é a mais usada em aplicações de robótica (CORKE, 2011), e por isso, será usada nesse trabalho.

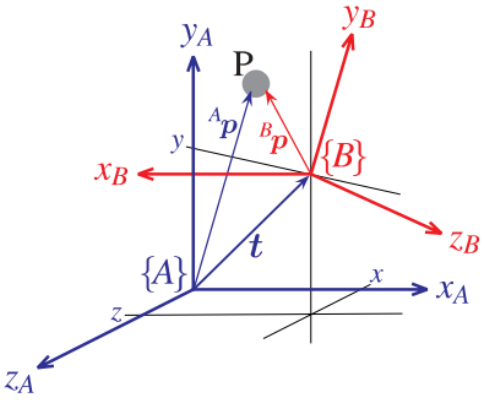


Figura 5: Diferentes sistemas de coordenada utilizados para representar o ponto P. O referencial $\{B\}$ está transladado de $\{A\}$ pelo vetor t, com matriz de rotação ${}^{A}R_{B}$
Fonte: CORKE, 2011, p.25

A matriz de transformação homogênea utiliza o sistema de coordenadas homogênea. O sistema de coordenadas homogêneas representa pontos de um espaço n dimensional por um vetor de $n + 1$ dimensões. No caso do espaço tridimensional um ponto p representado em coordenadas cartesianas pelo vetor $p = (x, y, z)$, pode ser representado em coordenadas homogêneas pelo vetor $\tilde{p} = (xw, yw, zw, w)$, sendo w qualquer número real. Para passar um

vetor em coordenadas cartesianas para coordenadas homogêneas normalmente escolhe-se $w = 1$, logo, $\tilde{p} = (x, y, z, 1)$.

Usando a matriz de transformação homogênea ${}^{A}T_{B}$ e a representação em coordenadas homogêneas, a equação (2.19) se resume a:

$$ {}^{A}\tilde{p} = \begin{bmatrix} {}^{A}p \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{A}R_{B} & t \\ 0_{1x3} & 1 \end{bmatrix} \begin{bmatrix} {}^{B}p \\ 1 \end{bmatrix} = {}^{A}T_{B} \; {}^{B}\tilde{p} \, . \tag{2.20} $$

A matriz de transformação homogênea é uma matriz 4x4 e tem uma estrutura bem especifica, pertencendo ao grupo euclidiano especial de dimensão 3, ou $T \in SE(3) \subset R^{4\times4}$. A representação de pose com a matriz de transformação homogênea facilita na representação de pose relativa a vários sistemas de coordenadas.

$$ {}^{A}T_{C} = {}^{A}T_{B} \; {}^{B}T_{C} = \begin{bmatrix} {}^{A}R_{B} & t_1 \\ 0_{1x3} & 1 \end{bmatrix} \begin{bmatrix} {}^{B}R_{C} & t_2 \\ 0_{1x3} & 1 \end{bmatrix} = \begin{bmatrix} {}^{A}R_{B} \; {}^{B}R_{C} & t_1 + {}^{A}R_{B} \, t_2 \\ 0_{1x3} & 1 \end{bmatrix} \tag{2.21} $$

$$ {}^{B}T_{A} = \left({}^{A}T_{B} \right)^{-1} \tag{2.22} $$

2.4 CONCEITOS DE VISÃO COMPUTACIONAL

A câmera é um sensor muito utilizado em robótica e nessa seção serão abordados conceitos básicos de visão computacional e geometria projetiva.

Uma câmera é um dispositivo eletromecânico composto por um sistema óptico (lentes e diafragmas) e um sensor para captura de imagens bidimensionais. Para discutir o processo de formação de uma imagem bidimensional de um objeto no espaço tridimensional, primeiramente considere uma câmera sem lentes, chamada de câmera estenopeica ou câmera *pinhole*. Nessa câmera a luz passa através de um pequeno buraco de forma que todos os raios de luz que incidem no plano de imagem passam aproximadamente pelo mesmo ponto.

Seja {C} um sistema de coordenada com origem no centro do orifício de passagem da luz, e eixos coordenados dispostos como na Figura 6. Seja {E} um sistema de coordenadas fixado na terra sendo a pose da câmera representada pela matriz ${}^{E}T_{C}$. Seja P um objeto em um ponto ${}^{C}P = ({}^{C}X, {}^{C}Y, {}^{C}Z)$ em coordenadas de $\{C\}$. Seja ${}^{E}P$ o mesmo ponto, agora representado em coordenadas de {E}. O raio de luz que é refletido por P e passa pelo orifício da câmera é projetado no ponto $p_i = (x_i, y_i)$ do plano da imagem. Pela relação entre os triângulos é possível relacionar os pontos pela equação (2.23), sendo F a distância

focal, ou a distância entre o plano da imagem e a origem de {C}. Pode-se também reescrever a equação (2.23), em coordenadas homogêneas, como na equação (2.24).

$$x_i = F\frac{{}^{C}X}{{}^{C}Z}, \quad y_i = F\frac{{}^{C}Y}{{}^{C}Z} \tag{2.23}$$

$$\tilde{p} = \begin{bmatrix} \widetilde{x_i} \\ \widetilde{y_i} \\ \widetilde{w_i} \end{bmatrix} = \begin{bmatrix} F & 0 & 0 & 0 \\ 0 & F & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^{C}X \\ {}^{C}Y \\ {}^{C}Z \\ 1 \end{bmatrix} = C\ {}^{C}\tilde{P} = C\ {}^{E}T_{C}^{-1}\ {}^{E}\tilde{P} \tag{2.24}$$

$$\widetilde{w_i} = {}^{C}Z\,, \quad x_i = \frac{\widetilde{x_i}}{\widetilde{w_i}}\,, \quad y_i = \frac{\widetilde{y_i}}{\widetilde{w_i}} \tag{2.25}$$

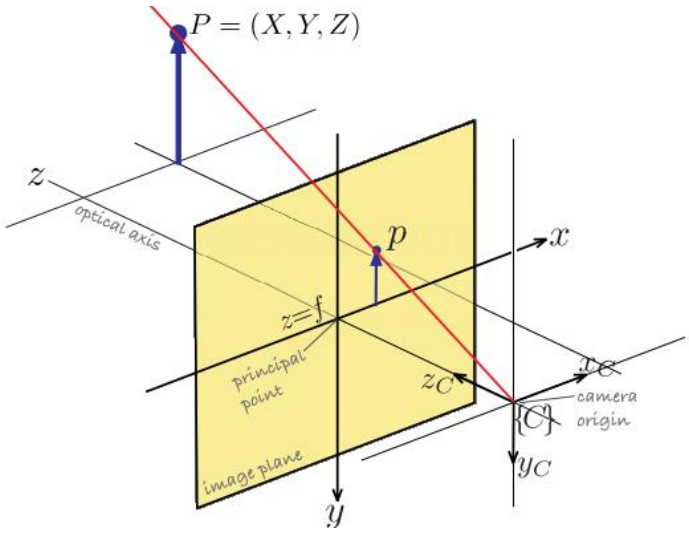


Figura 6: Sistemas de coordenadas para modelagem da formação da imagem.
Fonte: (CORKE, 2011, p. 252)

A matriz C é chamada de matriz de projeção da câmera *pinhole*. Nas câmeras modernas é utilizado lentes no lugar de um orifício pequeno, dessa forma uma quantidade maior de luz é incidida no plano de imagem do sensor. Câmeras com lentes possuem duas lentes convexas e um diafragma móvel para ajuste do foco. Diferente das câmeras *pinhole*, que projetam uma imagem focada de todos os objetos da cena, independente da sua distância, as câmeras com lentes precisam ajustar o foco para visualizar os objetos de interesse. Esse foco irá fazer com que a imagem de objetos a determinada distância da câmera estejam focadas, e a medida que se afastam ou se aproxima da câmera, perdem o foco. O conjunto de distâncias para as quais os objetos formariam imagens focadas é chamado de profundidade de campo. Câmeras com pequenas lentes se aproximam da câmera

pinhole e possuem uma profundidade de campo maior. Em contrapartida menos luz incide no sensor, o que prejudica a imagem. Para melhores imagens pode-se aumentar o tempo de exposição, mas isso é ruim para capturar imagens de objetos em movimento.

A equação (2.24) dá a posição da projeção em um sistema métrico, já em câmeras digitais geralmente as coordenadas no plano da imagem são dadas em pixels e o ponto $(0,0)$ é o canto superior esquerdo da imagem, ao invés do centro da imagem. Seja ρ_h e ρ_w a altura e largura do pixel no plano do sensor, seja o ponto (u, v) um ponto em coordenadas de pixel no plano da imagem. Então a relação entre a coordenada métrica e a de pixels é dada na equação (2.26), onde (u_o, v_o) é o ponto em que o eixo z de {C} intercepta o plano da imagem (geralmente o centro da imagem a não ser que o sensor esteja desalinhado).

$$u = \frac{x_i}{\rho_w} + u_o, \qquad v = \frac{y_i}{\rho_h} + v_o \tag{2.26}$$

Assim, a matriz de projeção também absorve esses parâmetros e é chamada de matriz de intrínsecos da câmera. Em contrapartida a matriz ${}^{E}T_{C}^{-1}$ é chamada de matriz de extrínsecos.

$$C = \begin{bmatrix} \frac{F}{\rho_w} & 0 & u_o & 0 \\ 0 & \frac{F}{\rho_h} & v_o & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.27}$$

A projeção perspectiva tem as seguintes características:

1. Faz o mapeamento de um espaço tridimensional para um espaço bidimensional..
2. Linhas retas no espaço tridimensional são projetadas em linhas retas no espaço bidimensional.
3. Linhas paralelas no espaço tridimensional são projetadas em linhas retas que se interceptam em um mesmo ponto no horizonte.
4. Figuras cônicas são projetadas em figuras cônicas, ou seja um circulo no espaço tridimensional é projetado em um circulo ou uma elipse no espaço bidimensional, a depender da matriz de extrínsecos.
5. O mapeamento não preserva os ângulos.

6. O mapeamento não é bijetor, e não existe uma solução única para inversa, ou seja não se pode determinar ${}^{C}P$ apartir de p.

Existem infinitas soluções ${}^{C}P$, mesmo para um conjunto de pontos no espaço tridimensional, e a relação entre as soluções é um fator de escala ou de perspectiva. Por isso objetos grandes e distantes da câmera podem gerar imagens iguais a objetos pequenos e próximos da câmera, desde que os objetos tenham a mesma forma, com diferentes fatores de escala, como mostra a Figura 7.

Figura 7: Perspectiva forçada. O fator de escala dos objetos em miniatura faz com que os objetos pareçam mais longe, quando na verdade estão perto, porém, pequenos.
Fonte: Michael Paul Smith, disponível em: <http://www.flickr.com/photos/24796741@N05/>

Há duas maneiras de fazer o mapeamento inverso e achar ${}^{C}P$ atravez da projeção. Em uma delas é necessário conhecer o tamanho do objeto no ambiente real. Já na outra maneira utiliza-se imagens estéreo (duas imagens do mesmo objeto porém captradas em posições diferentes). A primeira maneira é de fácil verificação, já a segunda faz uso de restrições geométricas.

Considere as câmeras $C1$ e $C2$ com translação e rotação relativa entre elas dada pela matriz R e o vetor t, como ilustrado na Figura 8. Considere que estas câmeras observam o ponto p no espaço tridimensional real e que esse ponto é projetado nos pontos $x1$ e $x2$ das imagens. Observa-se que qualquer ponto da reta r seria projetado no ponto $x1$ da imagem da camera $C1$. Já os pontos da reta r seriam projetados ao longo da linha $l2$ que é chamada de linha epipolar de $x1$. A linha epipolar $l2$ é dada pela interseção entre o plano da imagem de $C2$ e o plano formado pelos pontos $x1$, $C1$ e $C2$, onde $C1$ e $C2$ são os centros do obturador da câmera. Seja $\hat{x}1 = (x1 - C1)$ o vetor que aponta de $C1$ para $x1$ e seja $\hat{x}2 = (x2 - C2)$ o

vetor que aponta de $C2$ para $x2$, então pelos conceitos de geometria analítica é possivel obter uma relação matematica entre os pontos $x1$ e $x2$ dada pela equação abaixo:

$$(\hat{x}2)^T \cdot [S(t)R] \cdot \hat{x}1 = (\hat{x}2)^T \cdot E \cdot \hat{x}1 = 0 \tag{2.28}$$

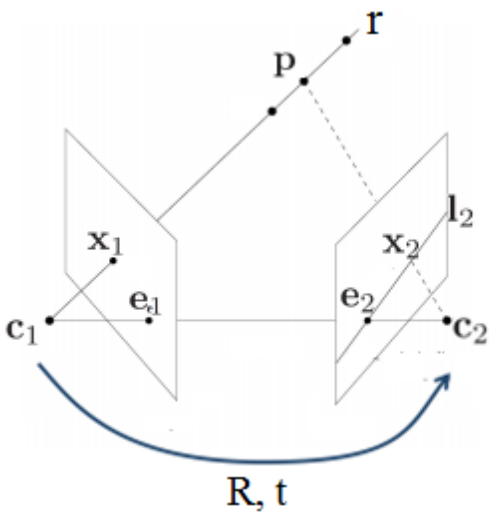


Figura 8: Geometria epipolar

A matriz E é chamada de matriz essencial e a equação é chamada de restrição epipolar. Assim conhecendo-se R e t, ao se encontrar um ponto de interesse em uma imagem basta buscar pelo ponto correspondente ao longo da linha epipolar, e assim encontrar a solução para o ponto p. Caso R e t não sejam conhecidas o problema é encontrar R, t e p. É possivel solucionar esse problema caso se encontre pelo menos 5 pontos correspondentes nas duas imagens. Para achar pontos correspondentes é necessario identificar e distinguir os pontos encontrados, podendo utilizar, por exemplo, o descritor SIFT (*Scale-invariant feature transform*). Para cada correspondencia encontrada tem-se uma restrição epipolar, como há erros e ruidos nem todas as restrições poderão ser satisfeitas sempre, mas pode-se encontrar uma solução satisfatória pela minimização do erro.

3 SLAM COM FILTRO DE KALMAN

Neste capítulo será apresentado a teoria do filtro de Kalman e o desenvolvimento de um filtro de Kalman para realizar mapeamento e localização. Na seção 3.11 serão abordados alguns algoritmos embarcados no *firmware* de fábrica do *AR.Drone* e como as informações geradas por esses algoritmos podem ser usadas para desenvolvimento de um algoritmo de localização. Na seção 3.22 será discutido o desenvolvimento do algoritmo de localização utilizado nesse trabalho que utiliza técnicas de visão computacional e um filtro de Kalman.

A localização é feita em termos de um referencial qualquer, geralmente fixado à terra, como foi assumido na seção 2.3. Para o desenvolvimento desse trabalho, o autor não tinha disponível um equipamento de medição de posição, com precisão certificada para avaliar a precisão do algoritmo de localização. O algoritmo desenvolvido foi diretamente implementado em ROS (*robot operating system*), sem realizar simulações, apesar de existirem ferramentas poderosas para isso, como o software de código aberto GAZEBO[17] e o *toolbox* de robótica para Matlab (CORKE, 2011).

3.1 ALGORITMOS DO *FIRMWARE* DO *AR.DRONE* E DO ARTOOLKIT

O *firmware* embarcado no *AR.Drone* implementa alguns algoritmos que serão úteis para o desenvolvimento do algoritmo de SLAM e as informações geradas por esses algoritmos podem ser acessadas pela API do *firmware*.

Segundo Bristeau et al. (2011) o firmware realiza a estimação dos ângulos de atitude utilizando a IMU. Além disso, realiza também uma estimação de velocidade linear utilizando informações proveniente da câmera vertical (apontada para baixo) e da IMU. Utiliza-se então uma técnica de fusão de sensores para melhorar a estimativa da velocidade.

A estimativa da velocidade apenas pela câmera é realizada por dois algoritmos complementares. A depender do conteúdo da cena capturada pela câmera e/ou da qualidade esperada da estimação, um dos algoritmos é escolhido para fazer a estimativa. O primeiro algoritmo calcula o fluxo óptico (*optical flow*) que é a velocidade de deslocamento da imagem capturada utilizando o método de Lucas e Kanade. O segundo algoritmo calcula o deslocamento de pontos de interesses (*trackers*) entre duas imagens consecutivas. Os pontos

[17] Em: < http://gazebosim.org/>. Acesso em 23 janeiro 2014.

de interesse são extraídos pelo método FAST corner detector e um algoritmo de minimização dos erros quadráticos é usado para identificar cada ponto nas duas imagens e calcular o deslocamento da figura.

Para ambos os algoritmos assume-se que a superfície do chão é plana e que a profundidade da cena é uniforme, ou seja, assume-se que o chão é paralelo ao plano do sensor da câmera. O segundo algoritmo tem boa precisão , no entanto, precisa que a cena tenha bastante contraste e que a velocidade não seja muito alta. Já o primeiro algoritmo tem menor precisão , contudo, tem resultado satisfatório em ambientes de pouco contraste, ou em altas velocidades. A estimação da velocidade pela câmera não possui erro sistemático (bias), mas possui erro aleatório (ruído) e tem uma frequência de atualização pequena em relação à dinâmica da aeronave.

A estimativa da velocidade pela IMU leva em consideração o modelo aerodinâmico da aeronave e o fenômeno de *blade flapping*, que provoca uma força de arrasto na aeronave proporcional a velocidade linear entre a aeronave e o vento. Em ambientes internos a velocidade do vento em relação ao referencial da terra pode ser desprezada, logo a força de arrasto é proporcional a velocidade linear da aeronave. Essa força de arrasto pode ser medida diretamente pelo acelerômetro e, considerando o modelo aerodinâmico, pode-se estimar a velocidade. Essa estimativa possui erro sistemático , porém, a frequência de *update* é maior que a do método anterior. Ao utilizar o método de fusão de sensores pode-se estimar o erro sistemático da estimação pela IMU, melhorar a estimação final da velocidade, e garantir redundância da estimação. Em situações que a estimação pela câmera não está disponível usa-se a estimação da IMU e vice-versa.

O firmware também realiza o reconhecimento de determinados padrões pelas imagens da câmera. Esses padrões são de adesivos e marcadores que são incluídos na caixa do *AR.Drone*, originalmente com o intuito de utilizar em aplicativos de realidade aumentada. O algoritmo sabe o tamanho do padrão no espaço real, logo é possível estimar a distância desse padrão para a câmera, que é aproximadamente a componente ^{C}Z da posição do objeto em coordenadas de $\{C\}$. Um dos padrões é chamado de *oriented roundel* e ao reconhecer esse padrão o *firmware* também identifica o ângulo de orientação do padrão. Assim, quatro informações são disponibilizadas pelo *firmware*: posição do padrão no plano da imagem (x_i, y_i), distância do padrão para a câmera (Z), e ângulo de orientação do padrão.

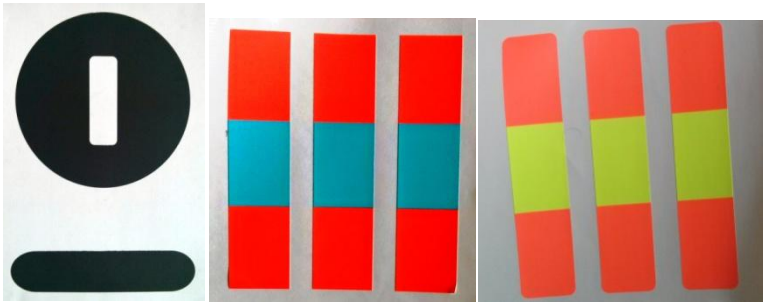


Figura 9: Padrões inclusos na caixa do *AR.Drone* que podem ser reconhecidos pelo *firmware*. Na esquerda o *oriented roundel*, no meio e na direita adesivos coloridos originalmente para serem colados no *AR.Drone* para reconhecimento de aeronaves inimigas.
Fonte: do autor.

O software ARToolkit também realiza detecção e localização relativa de padrões em uma imagem. Para isso deve-se alimentar o software com as figuras dos padrões que devem ser detectados e as dimensões da mesma. Através da teoria de projeção e visão computacional, o software retorna a posição relativa dos padrões. Na Figura 10 tem-se exemplos de padrões detectados pelo ARToolkit.

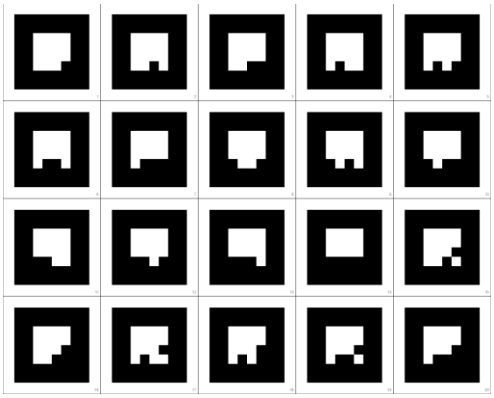


Figura 10: Padrões reconhecidos e rastreados pelo ARToolkit.
Fonte: do autor.

3.2 FILTRO DE KALMAN

O SLAM será realizado utilizando um filtro de Kalman. O filtro de Kalman é um filtro bayesiano para sistemas com modelos probabilísticos que possuam distribuições normais. O filtro utiliza uma serie de observações ruidosas contendo informações do estado do sistema e estima o estado de maneira ótima (para o modelo probabilístico), que tende a ser uma estimação melhor do que se fosse estimado por uma única observação (ENGEL,

STURM, & CREMERW, 2012). É, portanto, uma técnica de fusão de sensores que é vastamente usado em problemas de engenharia.

O filtro de Kalman opera em 2 passos. O primeiro passo é o passo de predição, onde o estado é estimado a partir da estimação no instante de tempo anterior e de um modelo dinâmico do sistema. É estimada também a incerteza inerente a estimação predita utilizando o modelo probabilístico do sistema, que numa distribuição normal é uma matriz de covariância. O modelo probabilístico do sistema considera os erros de medição dos sensores e incertezas do modelo dinâmico do sistema. O estado estimado pelo passo de predição é chamado de *priori* pois não utiliza as informações das observações no tempo atual para fazer a estimação.

O segundo passo é o passo de inovação (ou atualização) que utiliza as observações no estado atual para atualizar a estimação do estado e das incertezas. O estado estimado após a inovação é chamado de *posterior*. O passo de inovação geralmente é executado seguido do passo de predição, mas, caso não haja observação disponível esse passo pode ser pulado. Assim, o passo de predição é executado até que haja observação disponível.

O filtro assume que todas as funções do modelo probabilístico tem distribuição normal $X \sim N(\mu, \Sigma)$, que tem como função de densidade de probabilidade como na equação (3.1).

$$X \sim N(\mu, \Sigma)\,; \qquad P(X = x) = \frac{1}{(2\,\pi)^{d/2}\,|\Sigma|^{1/2}}\; e^{-\frac{(x-\mu)^T \Sigma^{-1}(x-\mu)}{2}} \tag{3.1}$$

Sendo x $\in R^d$ a variável amostrada, $\mu \in R^d$ a média da função e $|\Sigma|$ o determinante da matriz de covariância $\Sigma \in R^{dxd}$, $\Sigma = \Sigma^{\mathrm{T}}$, $\Sigma > 0$. A matriz de covariância tem que ser definida positiva, ou seja $\Sigma > 0$. A distribuição normal tem as seguintes propriedades.

1. Seja $X \sim N(\mu, \Sigma)$ e $Y \sim AX + B$ sendo A e B matrizes compatíveis com a operação, então $Y \sim N(\mu_y, \Sigma_{\mathrm{y}})$, onde:

$$\mu_y = A\mu + B \quad ; \quad \Sigma_{\mathrm{y}} = \mathrm{A}\,\Sigma\,\mathrm{A}^{\mathrm{T}} \tag{3.2}$$

2. Seja $X_1 \sim N(\mu_1, \Sigma_1)$ e $X_2 \sim N(\mu_2, \Sigma_2)$. Então $X_1 \cap X_2 \sim N(\mu_R, \Sigma_{\mathrm{R}})$, onde:

$$\mu_R = \frac{\Sigma_2}{\Sigma_2 + \Sigma_1}\mu_1 + \frac{\Sigma_1}{\Sigma_2 + \Sigma_1}\mu_2 \quad ; \quad \Sigma_R = \frac{1}{{\Sigma_2}^{-1} + {\Sigma_1}^{-1}} \tag{3.3}$$

Para um processo estocástico linear o estado estimado é dado por uma distribuição normal no espaço de estados. Seja $x_t \in R^n$ o vetor de estados do sistema no tempo t, pode-se modelar $x_t \sim N(\mu_t, \Sigma_t)$ sendo μ_t, Σ_t a média e covariância da estimação no tempo t. o sistema evolui no tempo linearmente seguindo o modelo no espaço de estados:

$$x_t = G\, x_{t-1} + B\, u_t + \varepsilon_t \tag{3.4}$$

Sendo A e B matrizes do modelo, u_t o vetor de entrada de controle, e $\varepsilon_t \sim N(0, \Sigma_\varepsilon)$ uma perturbação ou erro de medição de distribuição probabilística conhecida ($\varepsilon_t \in R^n$ e $\Sigma_\varepsilon \in R^{nxn}$). Uma observação do estado $Z_t \in R^k$ pode ser medida por um sensor ou estimada, e pode-se modelar a observação como um modelo linear.

$$Z_t = h(x_t) = H\, x_t + \delta_t \tag{3.5}$$

Sendo e $C \in R^{kxn}$ e $\delta_t \sim N(0, \Sigma_\delta)$ o ruído de medição ou incerteza de estimação, $\delta_t \in R^k$ com $\Sigma_\delta \in R^{nxn}$. Usando as propriedades da distribuição normal e o modelo dinâmico do sistema chega-se as equações usadas no passo de predição:

$$\overline{x_t} \sim N(\overline{\mu_t}, \overline{\Sigma_t}) \tag{3.6}$$

$$\overline{\mu_t} = G\mu_{t-1} + B\, u_t \tag{3.7}$$

$$\overline{\Sigma_t} = G\Sigma_{t-1}G^T + \Sigma_\varepsilon \tag{3.8}$$

Sendo $\overline{x_t}$ a estimação do estado *priori*. Aplicando o modelo da observação chega-se as equações usadas no passo de inovação, sendo K_t o ganho de Kalman do filtro.

$$\mu_t = \overline{\mu_t} + K_t(Z_t - H\overline{\mu_t}) \tag{3.9}$$

$$\Sigma_t = (I - K_t H)\overline{\Sigma_t} \tag{3.10}$$

$$K_t = \overline{\Sigma_t} H^T (H\overline{\Sigma_t} H^T + \Sigma_\delta)^{-1} \tag{3.11}$$

3.2.1 Odometria (predição)

Considere o sistema de coordenada $\{B\}$ fixo na aeronave conforme a Figura 11 e o sistema de coordenada $\{E\}$ fixo na terra, sendo ${}^E R_B = R_z(\theta_y) R_y(\theta_p) R_x(\theta_r)$, a matriz de rotação que representa a atitude da aeronave e ${}^E T_B = \begin{bmatrix} {}^E R_B & p \\ 0_{3x1} & 1 \end{bmatrix}$ a matriz de transformação homogênea que representa a pose da aeronave. Considere o sistema de coordenada $\{V\}$ com a origem na origem de $\{B\}$, porém, com os eixos x e y paralelos ao chão, ou seja a rotação

do referência $\{E\}$ para o referencial $\{V\}$ é dada pela matriz de rotação ${}^{E}R_{V} = R_z(\theta_y)$ e a matriz de rotação de $\{V\}$ para $\{B\}$ é ${}^{V}R_{B} = R_y(\theta_p)R_x(\theta_r)$. Assim, ${}^{E}T_{V} = \begin{bmatrix} {}^{E}R_{V} & p \\ 0_{3x1} & 1 \end{bmatrix}$, e ${}^{V}T_{B} = \begin{bmatrix} {}^{V}R_{B} & 0_{1x3} \\ 0_{3x1} & 1 \end{bmatrix}$. O estimador de velocidade linear desconsidera os ângulos de pitch e roll e assume que o plano da cena é uniforme e paralelo ao plano da imagem. Considera-se também que a velocidade vertical é nula, ou seja, a altitude não varia. Logo o estimador estima ${}^{V}v = \left[{}^{Vx}v\,, {}^{Vy}v\,,\ 0\right]^{T}$, que é a velocidade linear em coordenadas de $\{V\}$.

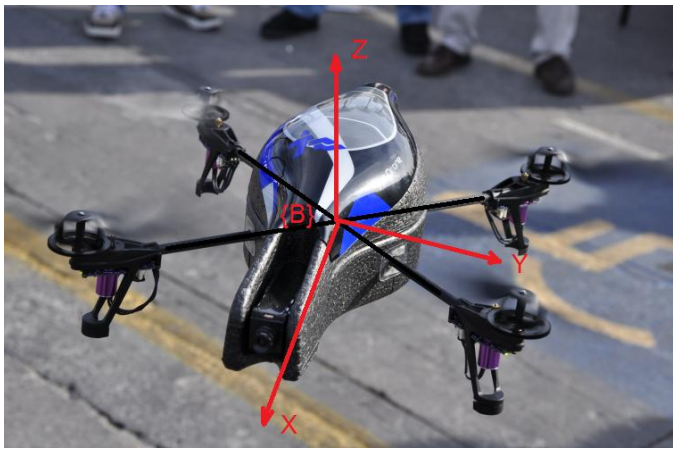


Figura 11: Referencial de corpo rígido utilizado pelo firmware do *AR.Drone*
Fonte: BRISTEAU et al. , 2011, p. 1478, adaptada pelo autor.

Para achar a velocidade linear em coordenada de $\{E\}$ basta fazer a rotação pelo ângulo de yaw.

$$ {}^{E}v = {}^{E}R_{V}\ {}^{V}v = \begin{bmatrix} \cos\theta_y & -sen\ \theta_y & 0 \\ sen\ \theta_y & \cos\theta_y & 0 \\ 0 & 0 & 1 \end{bmatrix} {}^{V}v = \begin{bmatrix} {}^{Vx}v\cos\theta_y - {}^{Vy}v\ sen\ \theta_y \\ {}^{Vx}v\ \text{sen}\ \theta_y + {}^{Vy}v\ cos\ \theta_y \\ 0 \end{bmatrix} \tag{3.12} $$

Para o filtro de Kalman será considerado o vetor de estado:

$$ x_t = [\ p_x(t)\ \ p_y(t)\ \ p_z(t)\ \ x_1\ \ y_1\ \ \ldots\ \ x_m\ \ y_m\ ]^{T} \tag{3.13} $$

Onde os três primeiros termos são a posição da aeronave em coordenada de $\{E\}$, e os outros termos são a posição mapeada dos marcos. Sabe-se a priori que os marcos estão dispostos no chão plano, assim estão a zero de altura. Logo para o passo de predição será feita a estimação da posição da aeronave por *dead reckoning*, bastando integrar a velocidade linear. Pode-se usar qualquer método numérico de integração. Como a estimação da

velocidade é ruidosa e de alta frequência de amostragem, não é necessário um método rigoroso de integração, e pode-se escolher o método mais simples, como a integração trapezoidal. Assim, as equações de atualização da posição são dadas abaixo:

$$p(t) = p(t-\Delta t) + {}^{E}v\,\Delta t \tag{3.14}$$

$$\begin{bmatrix} p_x(t) \\ p_y(t) \\ p_z(t) \end{bmatrix} = \begin{bmatrix} p_x(t-\Delta t) + ({}^{Vx}v\cos\theta_y - {}^{Vy}v\,sen\,\theta_y)\,\Delta t \\ p_y(t-\Delta t) + ({}^{Vx}v\,\text{sen}\,\theta_y + {}^{Vy}v\,cos\,\theta_y)\,\Delta t \\ p_z(t-\Delta t) + {}^{Vz}v\,\Delta t \end{bmatrix} \tag{3.15}$$

O termo $p_z(t)$ é medido pelo *firmware* pelo sensor ultrassom, contudo, foi observado que em determinadas situações existia um erro sistemático grande e variável no tempo (variação lenta). O *firmware* do *AR.Drone* faz a estimação da velocidade vertical, porém, o *driver* (ou o SDK) não disponibiliza essa informação. Não foi detectado o motivo que causa o erro de estimação de altitude com ultrassom nem o motivo pelo qual o *driver* não transmite a velocidade vertical. Logo pode-se estimar ${}^{Vz}v$ derivando a altitude estimada com o ultrassom e estimar melhor a altitude com a observação. Logo para o filtro de Kalman termos:

$$x_t = G\,x_{t-1} + B\,u_t + \varepsilon_t = x_{t-1} + \begin{bmatrix} ({}^{Vx}v\cos\theta_y - {}^{Vy}v\,sen\,\theta_y)\,\Delta t \\ ({}^{Vx}v\,\text{sen}\,\theta_y + {}^{Vy}v\,cos\,\theta_y)\,\Delta t \\ {}^{Vz}v\,\Delta t \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \varepsilon_t \tag{3.16}$$

$$G = I \tag{3.17}$$

$$\bar{\mu_t} = \mu_{t-1} + \begin{bmatrix} ({}^{Vx}v\cos\theta_y - {}^{Vy}v\,sen\,\theta_y)\,\Delta t \\ ({}^{Vx}v\,\text{sen}\,\theta_y + {}^{Vy}v\,cos\,\theta_y)\,\Delta t \\ {}^{Vz}v\,\Delta t \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.18}$$

$$\bar{\Sigma_t} = \Sigma_{t-1} + \Sigma_\varepsilon \tag{3.19}$$

Pode-se modelar ε_t pelo erro de medição da velocidade. Como não foi utilizado um sensor de velocidade com certificação de erro de medição, visto que a velocidade é estimada pelo *firmware* do AR.Drone, e como essa estimação depende bastante das condições de luminosidade do ambiente, da textura e relevo do chão, da própria velocidade e da atitude do

quadrotor, não é possível modelar a distribuição do erro em distribuição normal. Porém, baseado nos experimentos realizados, observou-se que o algoritmo funciona corretamente se for considerado um erro de medição de 10%. Então:

$$\Sigma_\varepsilon = \begin{bmatrix} \frac{\left|{}^{Vx}v\cos\theta_y - {}^{Vy}v\,sen\,\theta_y\right|}{10} & 0 & 0 & 0 & \cdots \\ 0 & \frac{\left|{}^{Vx}v\,\mathrm{sen}\,\theta_y + {}^{Vy}v\,cos\,\theta_y\right|}{10} & 0 & 0 & \cdots \\ 0 & 0 & \frac{\left|{}^{Vz}v\right|}{10} & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \Delta t \tag{3.20}$$

Para esse modelo não foi considerado o erro de estimação do ângulo de yaw θ_y , no entanto, este tem erro de estimação pequeno comparado com o erro de estimação de velocidade. Também não foi considerado o atraso de transporte.

3.2.2 Observação (inovação)

Considere o sistema de coordenada $\{C\}$ fixo na câmera da aeronave de acordo com a Figura 6 e os sistemas de coordenada $\{B\}$ e $\{E\}$ descritos na seção anterior. A câmera está fixa na estrutura da aeronave logo a matriz de transformação homogênea ${}^{B}T_{C}$ é constante. Considere a matriz intrínseca da câmera C dada pela equação (2.27).

Considera-se que existe uma *tag* (padrão, por exemplo o *oriented roundel*) no ponto ${}^{E}P$ em coordenadas de $\{E\}$. Essa *tag* servirá de marco de referência (*landmark*) para a localização, podendo existir mais de uma. Considere que a câmera está apontada para a *tag* e que a *tag* foi reconhecida pelo ARToolkit, tendo como informação observada a posição da tag em coordenadas da câmera ($Z_t = {}^{C}P = \begin{bmatrix} {}^{C}X & {}^{C}Y & {}^{C}Z \end{bmatrix}^T$). Pode-se achar uma expressão de ${}^{C}P$ em função dos elementos do vetor de estado (${}^{E}P$ e p):

$${}^{C}\tilde{P} = {}^{C}T_{B}\ {}^{B}T_{E}\ {}^{E}\tilde{P} = \begin{bmatrix} {}^{C}R_{B} & {}^{C}t_{B} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{B}R_{E} & -{}^{B}R_{E}\ p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 0 \\ 1 \end{bmatrix} \tag{3.21}$$

$${}^{C}P = {}^{C}R_{E} \begin{bmatrix} x_i \\ y_i \\ 0 \end{bmatrix} - {}^{C}R_{E}\ p + {}^{C}t_{B} \tag{3.22}$$

Os termos ${}^{C}R_{B}$ e ${}^{C}t_{B}$ são constantes e conhecidos, e o termo ${}^{B}R_{E}$ é dado pelo *firmware* do *AR.Drone*. Para o *AR.Drone* 2.0, foi considerado os seguintes valores para a matriz ${}^{B}T_{C}$ (considerando $\{B\}$ conforme a figura 11).

$$ {}^{B}T_{C} = \begin{bmatrix} 0 & -1 & 0 & -0.055 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -0.02 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ para a câmera vertical.} \tag{3.23} $$

$$ {}^{B}T_{C} = \begin{bmatrix} 0 & 0 & 1 & 0.21 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ para a câmera frontal.} \tag{3.24} $$

Definindo as seguintes matrizes

$$ H_{B} = -{}^{C}R_{E} = -\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{3.25} $$

$$ H_{i} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \\ R_{31} & R_{32} \end{bmatrix} \tag{3.26} $$

Então, podemos definir a matriz H da equação (3.5):

$$ H = [H_{B} \quad 0_{3\times2} \quad \cdots \quad H_{i} \quad \cdots \quad 0_{3\times2}] \tag{3.27} $$

Neste caso a estimação da posição da aeronave é influenciada pelos erros de estimação de atitude, erros de mapeamento do marco (ou dos marcos, caso haja mais de um), erro de estimação da posição do marco em coordenadas de $\{C\}$, e erro de determinação de ${}^{B}T_{C}$. A incerteza de observação Σ_{δ} também não pode ser modelada como uma distribuição normal, visto que há interferência de condições de iluminação, sombras, má impressão dos padrões, tamanho físico dos padrões, e identificação errada dos marcos (por exemplo, detectar marco onde não há, ou confundir um marco com outro). Novamente, baseado nos experimentos realizados, foi considerado um erro de 30cm para qualquer direção, sendo que o marco utilizado tem 15cm de largura e comprimento. Foi observado que considerar um erro pequeno pode comprometer o mapeamento quando ocorrer uma identificação errada de um marco. Assim foi considerada a seguinte matriz de covariância para a observação:

$$ \Sigma_{\delta} = \begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 0.3 \end{bmatrix} \tag{3.28} $$

Caso seja observado um marco que ainda não tenha sido mapeado deve-se extender o vetor de estados e a matriz de covariância. O vetor de estado extendido é dado abaixo:

$$\overline{\mu_t} = [\mu_{t-1} \quad f(\overline{x_t}, Z_t)] = y(\overline{x_t}, Z_t) \tag{3.29}$$

Onde $y()$ é a função de extensão do vetor de estados e $f() = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$ é a função inversa da função $h()$. Tem-se que:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = {}^{E}T_B \; {}^{B}T_C \begin{bmatrix} {}^{C}X \\ {}^{C}Y \\ {}^{C}Z \\ 1 \end{bmatrix} \tag{3.30}$$

Desta equação pode-se derivar duas expressões para $f()$:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = {}^{E}T_C \begin{bmatrix} {}^{C}X \\ {}^{C}Y \\ {}^{C}Z \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{E}R_C & {}^{E}t_C \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{C}X \\ {}^{C}Y \\ {}^{C}Z \\ 1 \end{bmatrix} \tag{3.31}$$

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = {}^{E}R_C \begin{bmatrix} {}^{C}X \\ {}^{C}Y \\ {}^{C}Z \end{bmatrix} + {}^{E}t_C = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} {}^{C}X \\ {}^{C}Y \\ {}^{C}Z \end{bmatrix} + \begin{bmatrix} {}^{E}x_C \\ {}^{E}y_C \\ {}^{E}z_C \end{bmatrix} \tag{3.32}$$

$$f() = \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \end{bmatrix} \begin{bmatrix} {}^{C}X \\ {}^{C}Y \\ {}^{C}Z \end{bmatrix} + \begin{bmatrix} {}^{E}x_C \\ {}^{E}y_C \end{bmatrix} \tag{3.33}$$

Ou então:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = {}^{E}T_B \begin{bmatrix} {}^{B}X \\ {}^{B}Y \\ {}^{B}Z \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{E}R_B & p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{B}X \\ {}^{B}Y \\ {}^{B}Z \\ 1 \end{bmatrix} \tag{3.34}$$

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = {}^{E}R_B \begin{bmatrix} {}^{B}X \\ {}^{B}Y \\ {}^{B}Z \end{bmatrix} + p \tag{3.35}$$

$$f() = \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left({}^{E}R_B \begin{bmatrix} {}^{B}X \\ {}^{B}Y \\ {}^{B}Z \end{bmatrix} + p \right) \tag{3.36}$$

Para extender a matriz de covariância e determinar a incerteza inicial do mapeamento do novo marco é necessário obter a matriz jacobiana de $y()$, ou seja, a matriz

de derivadas parciais $Y = {}^{\partial y}/_{\partial \overline{x_t}}$. Das equações (3.33) e (3.36) pode-se definir as matrizes F_Z e F_x dadas pelas equações abaixo e compor a matriz jacobiana Y.

$$F_Z = \frac{\partial f(\overline{x_t}, Z_t)}{\partial Z_t} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \end{bmatrix} \tag{3.37}$$

$$F_x = \frac{\partial f(\overline{x_t}, Z_t)}{\partial \overline{x_t}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.38}$$

$$Y = \frac{\partial y}{\partial \overline{x_t}} = \begin{bmatrix} I_{n\times n} & 0_{n\times 3} \\ F_x & 0_{2\times n-3} \quad F_Z \end{bmatrix} \tag{3.39}$$

Sendo n a dimensão do vetor de estado antes da extensão ($\overline{x_t} \in R^n$). Assim, a matriz de covariancia extendida é dada pela equação abaixo:

$$\Sigma_t = Y \begin{bmatrix} \overline{\Sigma_t} & 0_{n\times 3} \\ 0_{3\times n} & \Sigma_\delta \end{bmatrix} Y^T \tag{3.40}$$

Para maiores detalhes sobre a extensão da matriz de covariância o autor indica ao leitor consultar o trabalho de Corke (2011).

3.3 IMPLEMENTAÇÃO E RESULTADOS PRÁTICOS

O algoritmo proposto foi implementado em um pacote para ROS, junto com um controlador PD (proporcional derivativo) que foi proposto por Oliveira (2013)[18]. O pacote desenvolvido foi disponibilizado em código aberto para download em repositório github[19]. Com esse pacote é possivel fazer um AR.Drone seguir uma trajetória predefinida sem ter conhecimento previo do mapa. Também é possivel salvar e carregar mapas predefinidos. Um video com os resultados de testes praticos foi disponibilizado no youtube[20].

Para realizar o mapeamento pode-se fazer com que o AR.Drone siga uma trajetória circular ao redor do ponto de decolagem para visualizar todos os marcos. Foram distribuidos 16 marcos dispostos como na Figura 12 a). A aeronave sobrevoou os marcos a uma altitude de 1.5m de modo que não era possivel visualizar todos os marcos de uma só vez. O resultado do mapeamento está ilustrado na Figura 12 b). A comparação entre o resultado do mapeamento e o mapa real (levantado com uma trena) está ilustrado na Figura 13. Para esse

[18] Disponivel em <www.myrobotwork.blogspot.com>
[19] https://github.com/DaniloCosta/ArDroneControl
[20] https://www.youtube.com/watch?v=c6fLsAwWbE0

teste foi verificado um erro RMS de mapeamento de 15.85cm e um erro máximo de 25.67cm.

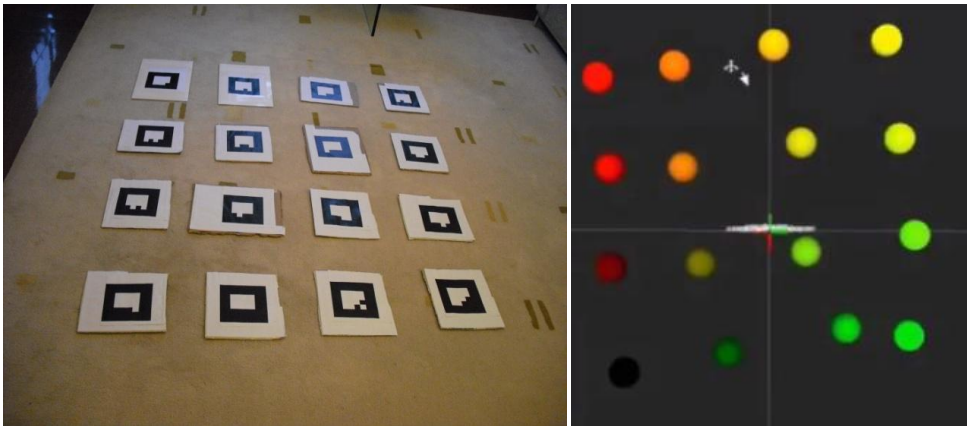


Figura 12: a) Marcos a serem mapeados. b) Mapa realizado

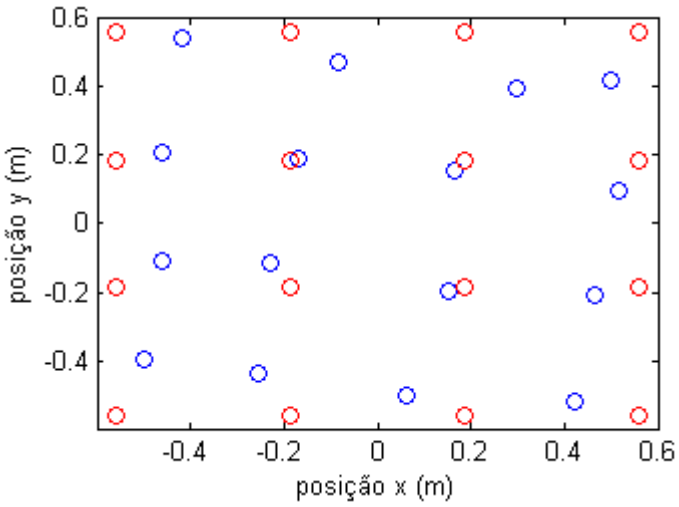


Figura 13: Comparação entre o mapa gerado (em azul) e o mapa real (vermelho).

Em seguida o quadrotor foi colocado para seguir uma trajetória em quadrado. A trajetória realizada está ilustrada na Figura 14 e os gráficos no tempo para esse teste podem ser vistos na Figura 15. Verificou-se que o algoritmo é capaz de realizar seguimento de trajetória.

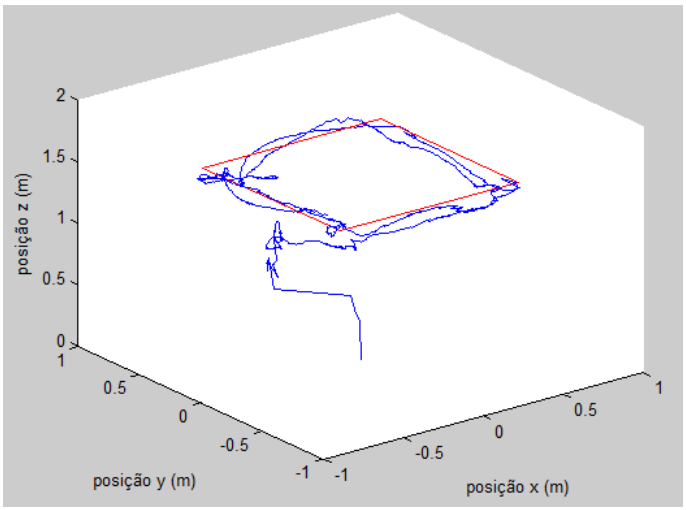


Figura 14: Trajetória realizada durante seguimento de trajetória

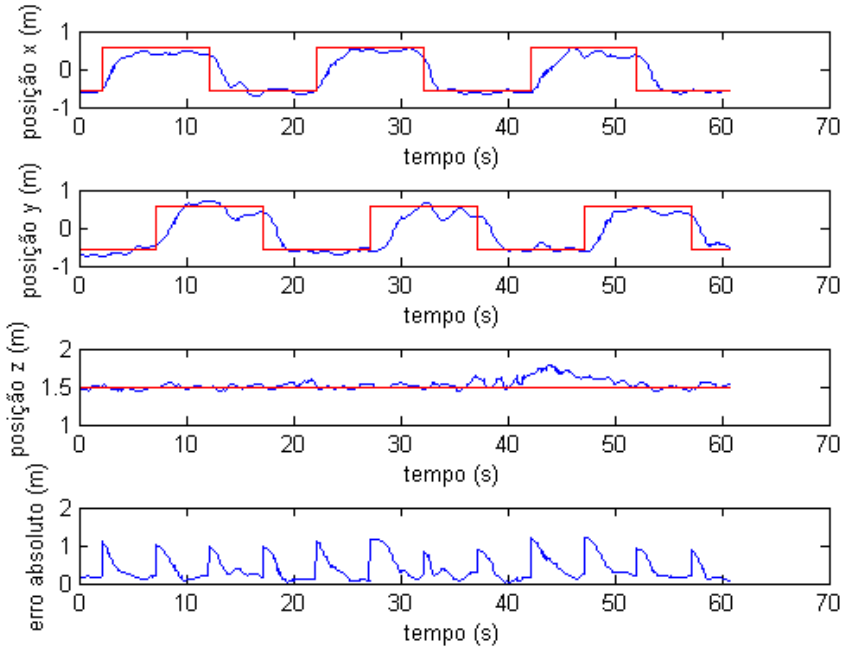


Figura 15: Gráficos da posição durante seguimento de trajetória

Também foi testada a estabilidade e rejeição de perturbações utilizando os algoritmos desenvolvidos. A trajetória realizada neste experimento está ilustrada na Figura 16 e um grafico das posições no tempo está ilustrado na Figura 17. O objetivo é manter a aeronave parada na posição $(0, 0, 1.5)$ mesmo sobre influencia de perturbações externas, como tapas e empurrões. Os picos nos gráficos são resultados de tais perturbações.

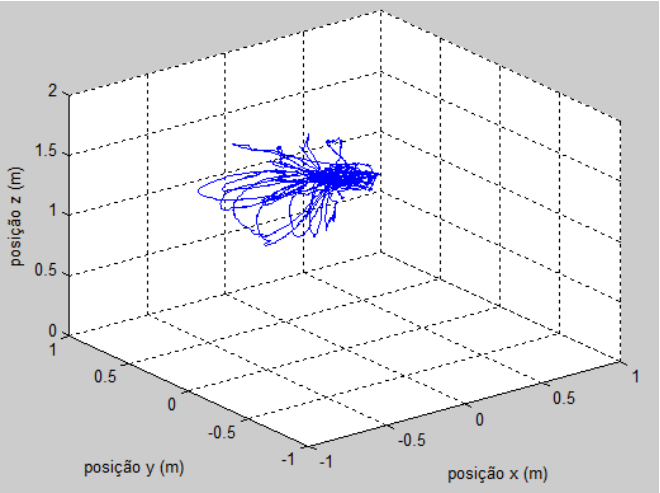


Figura 16: Trajetória realizada no teste de estabilidade e rejeição de perturbação

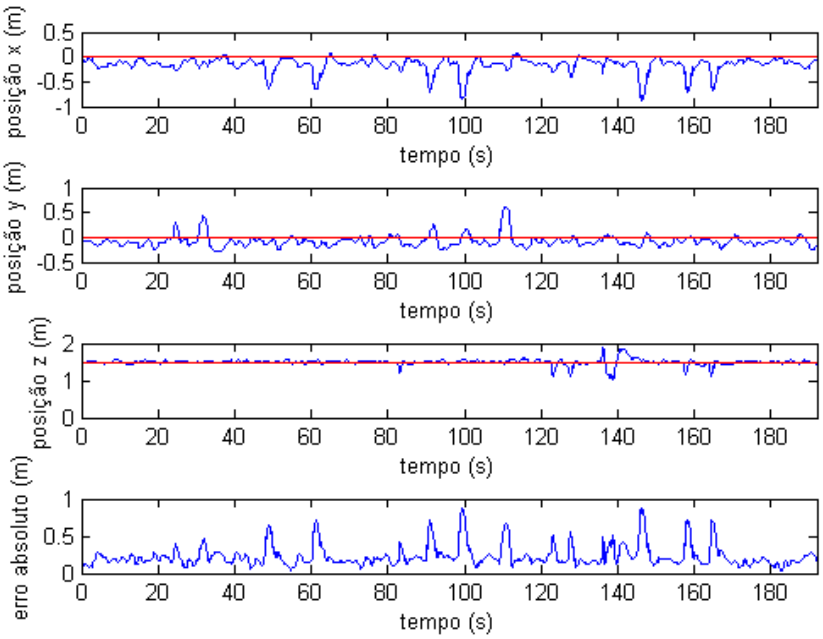


Figura 17: Gráfico de trajetória realizada no teste de estabilidade e rejeição de perturbação

Na Figura 18 observa-se um trecho dos dados coletados nesse experimento. Nesse trecho não foi desferido nenhum golpe contra a aeronave, no entanto o controlador foi capaz de estabilizar a aeronave, mantendo o erro absoluto de posição abaixo de 30 centímetros. Na Figura 19 observa-se outro trecho dos dados deste mesmo experimento onde o controlador rejeita a perturbação causada por um tapa. Observa-se que o controlador consegue rejeitar a perturbação e voltar para a condição estável com erro absoluto abaixo de 30 centímetros em apenas 4 segundos.

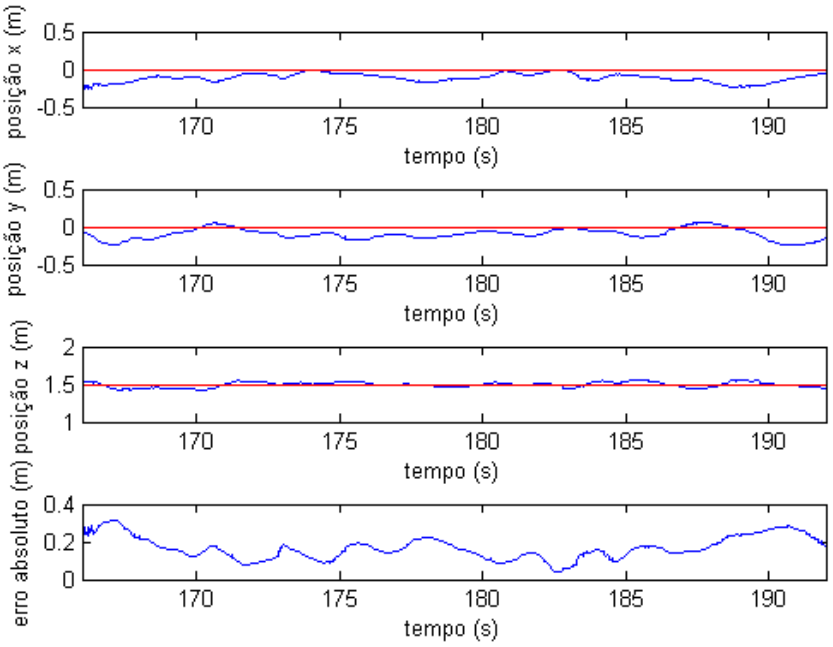


Figura 18: Trecho do teste de estabilidade e rejeição de perturbação mostrando que o algoritmo é capaz de manter erro absoluto de posição menor que 30cm

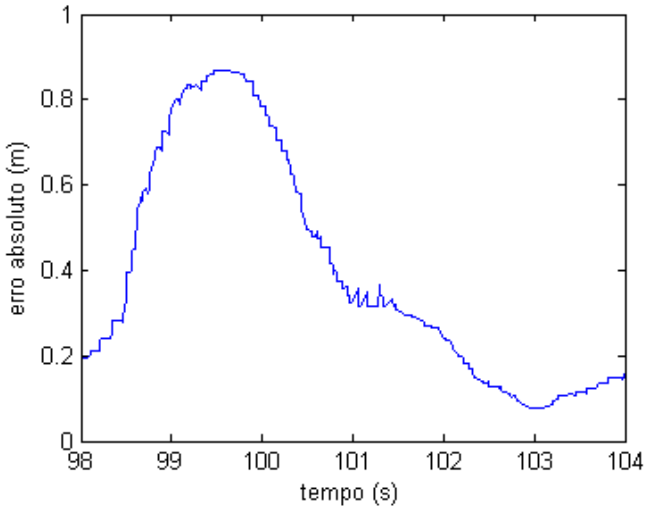


Figura 19: Trecho do teste de estabilidade e rejeição de perturbação mostrando que o algoritmo é capaz de rejeitar perturbações e voltar rapidamente à estabilidade

3.4 CONSIDERAÇÕES SOBRE O ALGORITMO PROPOSTO

Para esse método foi considerado que apenas um marco é visualizado por vez, ou seja, mesmo que haja mais de um marco mapeado na imagem, apenas um deles será utilizado na etapa de observação.

A utilização do padrão e do algoritmo de identificação embarcado no *firmware* simplificaram uma parte do problema de SLAM com visão. Em algoritmos de SLAM por visão é necessário identificar na imagem os pontos mapeados e distinguir entre os pontos visualizados, relacionando cada ponto na imagem com a posição no mapa. Utilizando a identificação do *ARToolkit* esse problema é solucionado. O problema também é simplificado pois é possivel utilizar como conhecimento prévio as dimensões do padrão, a posição do padrão no eixo Z (todos os marcos estão no chão plano), e a estimação dos ângulos de atitude dados pela IMU.

O filtro de Kalman é uma ferramenta útil para fusão de sensores e realização de SLAM, sendo utilizada em diversos trabalhos publicados nos ultimos anos. Este tem como vantagem a simplicidade do algoritmo e da modelagem, e o conhecimento da incerteza da estimação. Porém realizar SLAM com filtro de Kalman fica limitado a mapas de baixa complexidade (mapas esparsos) e pouco extensos, pois com o aumento do numero de pontos mapeados aumentam-se a complexidade computacional e o uso da memória ao operar matrizes de covariancia de grandes dimensões (HOLMES, KLEIN & MURRAY, 2009).

4 CONCLUSÃO

Este trabalho apresentou os resultados de um estudo sobre localização e mapeamento simultâneo aplicado a controle de pose de robôs móveis aéreos multirrotores em voo *indoor*, usando como sensores uma IMU de baixo custo em configuração *strapdown* e uma câmera monocular. Especificamente foi considerado o problema de estimação e controle de um quadrotor *AR.Drone* da empresa Parrot, sem a necessidade de modificação do *firmware* de fábrica. Citou alguns métodos de localização encontrados na literatura e em soluções comercias, especificando em quais situações cada método pode ser utilizado. Justificou a utilização da câmera monocular e da IMU como sensores para localização em voo autônomo indoor. Desenvolveu um filtro de Kalman para SLAM utilizando estimações de velocidade linear e conceitos de visão computacional. O método desenvolvido foi implementado em ROS e toma proveito dos recursos do *firmware* de fábrica do *AR.Drone*.

Para validação do algoritmo foi avaliado o desempenho e aplicabilidade como realimentação para um controlador de pose de um quadrotor. O algoritmo apresentou bom desempenho possibilitando a estabilização da pose do quadrotor e possibilitando o seguimento de trajetórias. Um pacote para ROS contendo o algoritmo de localização e o controlador foi disponibilizado em código aberto para *download* na internet[21].

4.1 PERSPECTIVAS FUTURAS

O algoritmo desenvolvido utiliza a leitura direta da atitude dada pela IMU, porém em trabalhos anteriores do autor foi constatado que essa informação não tem boa precisão, pois a medição do angulo de yaw tende a desviar ilimitadamente no tempo. Para aumentar a precisão e robustez do algoritmo pode-se inserir o angulo de yaw como um estado do vetor de estados do filtro de kalman. Nesse caso as equações do modelo serão não lineares por conta da rotação e o filtro deverá ser um filtro de kalman extendido (EKF).

O algoritmo desenvolvido também assume que os marcos estão distribuidos no chão plano e não considera a possibilidade de desnivel. Sugere-se o desenvolvimento de um algoritmo que não tenha tal restrição, assim pode-se utilizar também a câmera frontal do AR.Drone ao invéz da camera vertical.

[21] https://github.com/DaniloCosta/ArDroneControl

Para aumentar ainda mais a flexibilidade do quadrotor pode-se eliminar a necessidade de utilização de padrões e realizar a localização e mapeamento utilizando *features* das imagens. Sugere-se o estudo e desenvolvimento de um método de SLAM utilizando extratores de *features* e dispensando os padrões.

O método de localização por imagem se mostra adequado para utilização em voos indoor, porém, tem baixo desempenho em voos outdoor, o que também limita a atuação do quadrotor. Como solução pode-se utilizar também um sistema GPS em um algoritmo de fusão de sensores para tornar o quadrotor capaz de realizar ambos voos *indoor* e *outdoor*.

As trajetórias geradas neste trabalho para teste do controlador em caso servo não levam em consideração a limitação do quadrotor. Foram geradas referências em degrau, o que é não realizável pela aeronave, gerando altos sinais de controle e sobressinais. Para melhorar a performance do seguimento de trajetória é indicado a geração de referências suaves. Sugere-se o desenvolvimento de um algoritmo para geração de trajetórias suaves.

Para o seguimento de trajetórias também não foi considerado a presença de obstáculos. Caso haja algum obstáculo entre o quadrotor e o ponto que se deseje alcançar é necessário o planejamento da trajetória para contornar o obstáculo.

REFERÊNCIAS

BILLS, C.; CHEN, J.; & SAXENA, A. Autonomous MAV flight in indoor environments using single image perspective cues. In: International conference on robotics and automation, 2011. Disponível em: < http://www.cs.cornell.edu/~asaxena/MAV/saxena_MAV_perspectivecues_stairs.pdf>. Acesso em 24 janeiro 2014.

BRISTEAU, P.; CALLOU, F.; VISSIERE, D.; & PETIT, N. The navigation and control technology inside the AR.Drone micro UAV. In: IFAC world congress, Milão, 2011, p.1477-1484.

CORKE, P. Robotics vision and control – fundamental algorithms in matlab. Springer, 2011, 556p.

DIOSI, A.; & KLEEMAN, L. Advanced sonar and laser range finder fusion for simultaneous localization and mapping. In: Internation Conference on Intelligent Robots and Systems, 2004, v.2, 1854-1859.

EBERLI, D.; SCARAMUZZA, D.; WEISS, S.; & SIEGWART, R. Vision based position control for MAVs using one single circular landmark. In: Journal for intelligent robot systems, Springer, 2010, 18 p.

ENGEL, J.; STURM,J. & CREMERS, D. Camera-Based Navigation of a Low-Cost Quadrocopter. In Proc. of the International Conference on Intelligent Robot Systems (IROS), 2012. Disponível em: < http://vision.in.tum.de/_media/spezial/bib/engel12iros.pdf>. Acesso em 24 janeiro 2014.

ENGEL, J.; STURM,J.; & CREMERS, D. Accurate Figure Flying with a Quadrocopter Using Onboard Visual and Inertial Sensing. In Proc. of the Workshop on Visual Control of Mobile Robots (ViCoMoR) at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS), 2012. Disponivel em:< http://vision.in.tum.de/_media/spezial/bib/engel12vicomor.pdf>. Acesso em 24 janeiro 2014.

FALLON, M.; JOHANNSSON, H.; & LEONARD, J. Efficient Scene Simulation for Robust Monte Carlo Localization using an RGB-D Camera. IEEE International Conference on Robotics and Automation, 2012.

HEHN, M.; & D'ANDREA, R. A flying inverted pendulum. In: Internetional conference on robotics and automation, Shangai, IEEE, 2011, p.763-770.

HUANG, H.; HOFFMANN, G.; WASLANDER, S.; & TOMLIN, C. Aerodynamics and control of autonomous quadrotor helicopter on aggressive maneuvering. In: Internation conference on robotics and automation, Kobe, IEEE, 2009, p.3277-3282.

KLEIN, G.; & MURRAY, D. Parallel tracking and mapping for small AR workspaces. In: IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2007.

LEMAIRE, T.; BERGER, C.; JUNG, I.; & LACROIX, S. Vision-based SLAM: stereo and monocular approaches. In: International journal of computer vision, Springer,v.74, n.3, 2007, p. 343-364.

LUGO, J.; & ZELL, A. Framework for autonomous onboard navigation with the AR.Drone. In: International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2013, p. 575-583.

MAHONY, R.; KUMAR, V.; & CORKE, P. Multirotor aerial vehicles modeling estimation and control of a quadrotor. In: Robotics & automation magazine, IEEE, 2012, p.20-32.

MELLINGER, D.; MICHAEL, N.; & KUMAR, V. Trajectory generation and control for precise aggressive maneuvers with quadrotors. In: the international journal of robotics research, 2012, p. 664-674.

MELLINGER, D.; & KUMAR, V. Minimum snap trajectory generation and control for quadrotors. In: Internetional conference on robotics and automation, Shangai, IEEE, 2011, p.2520-2525.

MÜLLER, M.; LUPASHIN, S.; & D'ANDREA, R. Quadrocopter Ball Juggling, IEEE/RSJ International Conference on Intelligent Robots and Systems, 20101, p.5113–5120..

OLIVEIRA, D. Estimação de estado e controle de robô aéreo. 2014. 127 p. Monografia (graduação) – Departamento de Engenharia Elétrica, Universidade Federal da Bahia, Salvador, Bahia, Brasil. Disponivel em <www.myrobotwork.blogspot.com>

ROBERTS, A. Attitude estimation and control of VTOL UAVs. 2011. 209 p. Tese (doutorado) – School of graduate an postdoctoral studies, University of Western Ontario, Lonndon, Ontario, Canada, 2011.

SCHERER, S.; & ZELL, A. Efficient onboard RGBD-SLAM for autonomous MAVs. In: International Conference on Intelligent Robots and Systems, Tokyo, 2013, IEEE/RSJ, p. 1062-1068.

WEY, L.; CAPPELLE, C.; & RUICHEL, Y. Unscented information filter based multi-sensor data fusion using stereo camera, laser range finder and GPS receiver for vehicle localization. In: International IEEE Conference on Intelligent Transportation Systems (ITSC), 2011, 1923-1928.

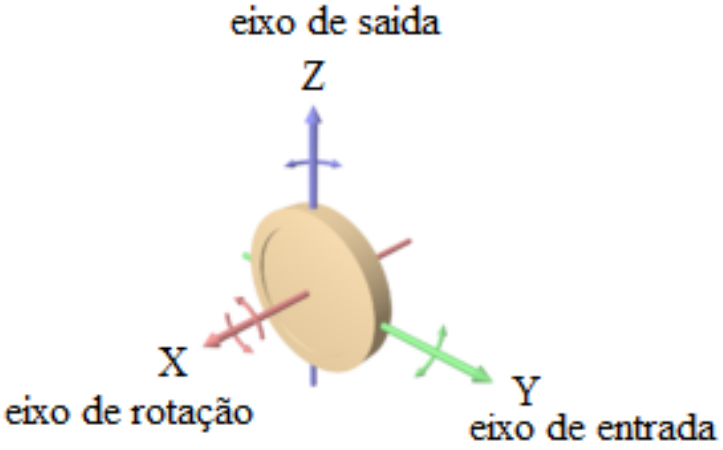
ZHANG, L.; CURLESS, B.; & SEITZ, S. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In: First International Symposium on 3D Data Processing Visualization and Transmission, 2002.

APÊNDICE A: GIROSCÓPIO GIRATÓRIO E GIMBALL LOCK

O giroscópio giratório é um dispositivo para manter determinada orientação e rejeitar as rotações do corpo no qual este está fixado. Baseia-se nos princípios de momento angular, que diz que um disco girando sobre um eixo X com momento angular L, se for exercido um torque τ em um eixo perpendicular ao eixo de rotação (eixo Y), isso resultará em uma rotação sobre um eixo Z perpendicular a ambos X e Y. Esse movimento é chamado de precessão e a velocidade angular de precessão Ω_P é dada pela equação (A.1).

$$\tau = \Omega_P \times L \tag{A.1}$$

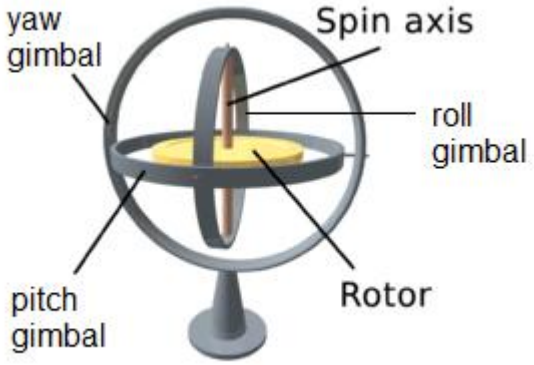
Figura 20: Eixos de rotação, de entrada e saída, do movimento de precessão.



Fonte: Wikipedia, disponível em: http://en.wikipedia.org/wiki/Gyroscope
adaptado pelo autor.

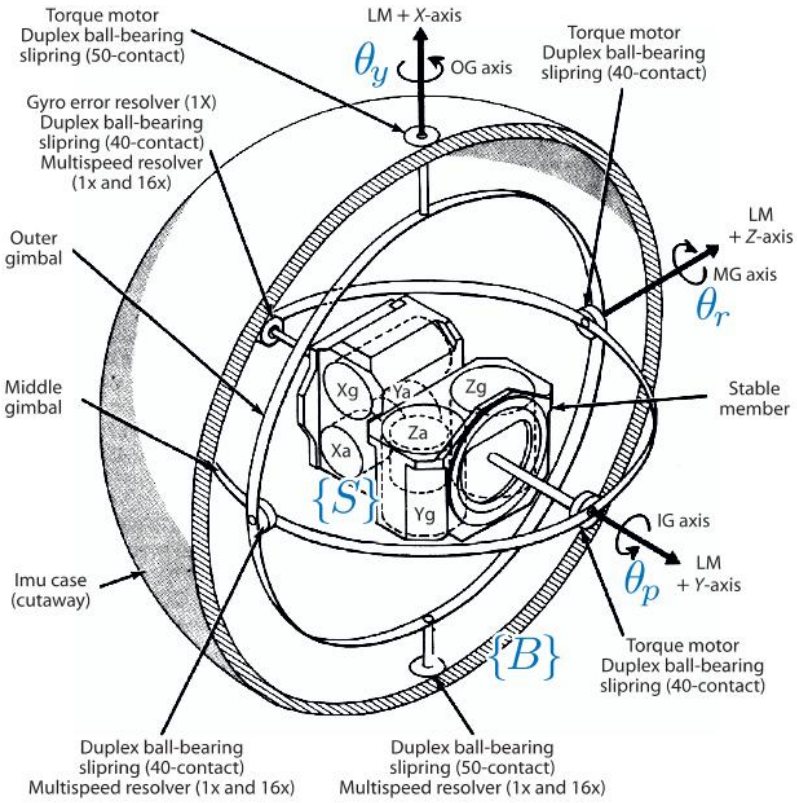
Desse modo ao se montar uma roda girante (rotor) em suporte com juntas tipo cardan (*gimbals*) de três eixos, como mostra a Figura 21, o eixo de rotação tenderá a permanecer na mesma direção, mesmo ao rotacionar a estrutura externa. Os ângulos yaw, pitch e roll da atitude podem então ser medidos pelos ângulos entre as juntas cardan. Outros instrumentos também podem ser montados na plataforma onde o rotor está girando. Unidades de medição inercial antigas montavam acelerômetros na plataforma de modo a medir as acelerações da aeronave em coordenadas do referencial da terra, como mostra a Figura 22.

Figura 21: Giroscópio livre em suporte com juntas tipo cardan



Fonte: Wikipedia, disponível em: http://en.wikipedia.org/wiki/Gyroscope adaptada pelo autor

Figura 22: Esquemático do giroscópio da Apollo 11. $\{B\}$ é o sistema de coordenada da espaçonave e $\{S\}$ é o sistema de coordenada da plataforma estável que tem atitude aproximadamente igual ao do referencial da terra. Xg Yg e Zg são os 3 giroscópios montados na plataforma e Xa,Ya e Za são os 3 acelerômetros medido a aceleração linear em coordenadas de $\{S\}$.



Fonte: CORKE, 2011, p.32

Apesar da atitude da plataforma estável do giroscópio tender a se manter estável sem variação, na prática há um deslocamento com o tempo (*drift*). Caso o referencial inercial $\{E\}$ estiver fixado na terra, este irá rotacionar junto com a terra, logo o referencial $\{S\}$ da

plataforma irá ter atitude diferente de $\{E\}$ ao longo do dia. Esta é a chamada precessão aparente e pode ser compensada pelo modelo da rotação da terra. Já outros fatores como desbalanceamento das juntas cardan, atrito nos rolamentos, e inércia dos anéis causam um deslocamento lento da leitura de atitude. Então é necessário utilizar uma outra forma de medição de atitude em intervalos de tempo para compensar esse erro.

Além disso, esse giroscópio sofre do problema de *gimbal lock*. Quando as juntas de pitch e yaw se alinham ($pitch = 90^o$) os eixos de yaw e roll ficam paralelos. Nessa situação, variações do ângulo de roll e yaw da aeronave, aplicarão torques nos mesmos eixos do suporte e o giroscópio não conseguirá compensar de maneira correta as rotações, levando a uma leitura errada da atitude. Esse problema é conhecido como *gimbal lock* e gera problemas como o incidente com a Apollo 11[22]. Uma maneira de contornar esse problema é adicionando mais um eixo cardan acionado por motor para evitar a singularidade. Outra maneira é utilizar motores nos eixos cardan e movimentar os eixos quando houver alinhamento e reajustar a referência de ângulos.

As práticas modernas consistem em não utilizar mais a estrutura de eixos de cardan e fixar os sensores da IMU (acelerômetro e magnetômetro) diretamente na estrutura da aeronave (configuração *strapdown*). Nessa configuração utiliza-se outras tecnologias de giroscópio (diferente do giroscópio giratório) para medir a velocidade angular de rotação da aeronave, ao invés de medir a orientação diretamente.

[22] Em: < http://en.wikipedia.org/wiki/Gimbal_lock>. Acesso em 22 janeiro 2014

APÊNDICE B: REPRESENTAÇÃO DE ATITUDE COM QUATÉRNIOS UNITÁRIOS

Quatérnios são números hiper complexos $Q \in H$, descobertos por William Hamilton em 1843, sendo H uma extensão do conjunto dos números complexos possuindo uma parte real e três partes imaginarias. Assim, $Q = u + xi + yj + zk$, sendo u, x, y e $z \in R$, e i, j e k são unidades imaginarias. O conjunto forma uma álgebra associativa tal que:

$$i^2 = j^2 = k^2 = -1, \tag{B.1}$$

$$i\,j = k \;\; ; \;\; j\,i = -k, \tag{B.2}$$

$$j\,k = i \;\; ; \;\; k\,j = -i, \tag{B.3}$$

$$k\,i = j \;\; ; \;\; i\,k = -j. \tag{B.4}$$

O conjugado de um quatérnio pode ser escrito como:

$$\bar{Q} = u - x\,i - y\,j - z\,k. \tag{B.5}$$

Um quatérnio pode ser escrito como um par de um escalar u e um vetor $q = [x, y, z]^{\mathrm{T}}$, com a notação $Q = (u, q)$. A multiplicação de quatérnios, é igual a multiplicação de números complexos, no entanto, de ordem maior e seguindo as equações (B.1) até a (B.4). Utilizando a forma vetorial pode-se escrever a multiplicação (operador $\odot$) de dois quatérnios $Q_1 = (u_1, q_1)$ e $Q_2 = (u_2, q_2)$:

$$\begin{aligned} Q_1 \odot Q_2 &= (u_1 u_2 - {q_1}^T q_2 \,,\, u_1 q_2 + u_2 q_1 + S(q_1) q_2) \\ &= (u_1 u_2 - q_1 \cdot q_2 \,,\, u_1 q_2 + u_2 q_1 + q_1 \times q_2). \end{aligned} \tag{B.6}$$

Sendo $q_1 \times q_2 = S(q_1) q_2$ o produto vetorial entre os vetores q_1 e q_2, e $q_1 \cdot q_2 = {q_1}^T q_2$ o produto escalar entre os vetores. A multiplicação de quatérnios tem como inversa $Q^{-1} = (u, -q)$ (que é igual ao de conjugado Q^*) e tem elemento neutro $Q^{-1} \odot Q = Q \odot Q^{-1} = (1, [0\ 0\ 0]^{\mathrm{T}})$. O produto é não comutativo logo manipulação algébrica tem que levar em conta o lado da multiplicação, como em álgebra matricial. Ou seja. se $Q_3 = Q_1 \odot Q_2$ então:

$$Q_1 = Q_3 \odot {Q_2}^{-1} \neq {Q_2}^{-1} \odot Q_3, \tag{B.7}$$

$$Q_2 = {Q_1}^{-1} \odot Q_3 \neq Q_3 \odot {Q_1}^{-1}, \tag{B.8}$$

$${Q_3}^{-1} = {Q_2}^{-1} \odot {Q_1}^{-1}. \tag{B.9}$$

A multiplicação também pode ser feita na forma matricial:

$$Q_1 \odot Q_2 = (u_3, q_3), \tag{B.10}$$

$$\begin{bmatrix} u_3 \\ q_3 \end{bmatrix} = \begin{bmatrix} u_1 & -{q_1}^T \\ q_1 & u_1 I + S(q_1) \end{bmatrix} \begin{bmatrix} u_2 \\ q_2 \end{bmatrix}. \tag{B.11}$$

A norma de um quatérnio pode ser definida por:

$$| \ |Q| \ | = \sqrt{u^2 + x^2 + y^2 + z^2}. \tag{B.12}$$

Assim, um quatérnio unitário é definido como $Q \in S^3$ tal que:

$$S^3 := \{ Q \in R \times R^3 ; \ | \ |Q| \ | = 1 \}. \tag{B.13}$$

Quatérnios unitários podem ser usados para representar rotações. O conjunto S^3 é uma hiperesfera em R^4, de raio unitário e centro na origem, por isso é chamado de hiperesfera de rotações. Assim como na representação em matriz de rotação existem duas convenções e as duas tem a seguinte relação de equivalência. O quatérnio Q em uma das convenções é equivalente ao quatérnio Q^{-1} na outra convenção. A adoção da convenção é importante pois as fórmulas das relações entre as formas de representação da rotação mudam. Será considerada a seguinte convenção.

Seja ${}^B Q_E \in S^3$ o quatérnio equivalente a ${}^B R_E$ (convenção de multiplicação pela esquerda) e representa a rotação do referencial $\{E\}$ para o referencial $\{B\}$. Então qualquer vetor v_E expresso em coordenadas de $\{E\}$, pode ser expresso em coordenadas de $\{B\}$ multiplicando-se:

$$(0, v_B) = \left({}^B Q_E \right)^{-1} \odot (0, v_E) \odot {}^B Q_E \, . \tag{B.14}$$

Uma fórmula compacta para a rotação de vetores é a seguinte:

$$v_B = q {v_E}^T q \ + u^2 v_E \ + \left(2\mathrm{u}\mathrm{I} \ - S(q) \right) S(v_E) q. \tag{B.15}$$

Sendo $(0, v_B)$ a representação em quatérnio do vetor v_B , tal que $v_B = {}^B R_E \, v_E$. Seja também ${}^C Q_B \in S^3$ o quatérnio equivalente a ${}^C R_B$ e ${}^C Q_E \in S^3$o quatérnio equivalente a ${}^C R_E = {}^C R_B \, {}^B R_E$, então:

$$ {}^C Q_E = {}^B Q_E \odot {}^C Q_B \, . \tag{B.16}$$

Uma outra convenção possível é fazer $(0, v_B) = {}^B Q_E \odot (0, v_E) \odot \left({}^B Q_E \right)^{-1}$. Nesse caso ${}^C Q_E = {}^C Q_B \odot {}^B Q_E$ e a relação entre uma convenção e outra é $Q = Q^{-1}$. Nesse trabalho será usada a primeira convenção.

A partir da representação em quatérnios $Q = (u, q)$ pode-se achar a matriz de rotação equivalente $R(Q)$ pela fórmula de *Rodrigues:*

$$R(Q) = I + 2\,S(q)^2 - 2uS(q) = (u^2 - q^T q)I_3 + 2qq^T - 2uS(q), \quad \text{(B.17)}$$

$$R(Q) = I + 2\,S(q)^2 + 2uS(q) = (u^2 - q^T q)I_3 + 2qq^T + 2uS(q). \quad \text{(B.18)}$$

Também é possível achar o quatérnio de uma rotação a partir da matriz de rotação. Existem quatro fórmulas equivalentes para se achar um dos quatérnios de uma matriz de rotação e outras quatro para se achar o negativo do mesmo, lembrando que os dois representam a mesma rotação. Uma das fórmulas é a seguinte:

$$u = \frac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}}, \quad \text{(B.19)}$$

$$q = \frac{1}{4u}[(R_{23} - R_{32})\ (R_{31} - R_{13})\ (R_{12} - R_{21})]^T, \quad \text{(B.20)}$$

$$u = \frac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}}, \quad \text{(B.21)}$$

$$q = \frac{1}{4u}[(R_{32} - R_{23})\ (R_{13} - R_{31})\ (R_{21} - R_{12})]^T. \quad \text{(B.22)}$$

Se for considerada uma rotação de ângulo θ sobre o eixo $\hat{e}$, seguindo a regra da mão direita, nesse caso o quatérnio unitário que representa essa rotação pode ser escrito como:

$$Q = \left(cos\left(\frac{\theta}{2}\right), sin\left(\frac{\theta}{2}\right)\hat{e}\right), \quad \text{(B.23)}$$

$$Q = \left(cos\left(\frac{\theta}{2}\right), -sin\left(\frac{\theta}{2}\right)\hat{e}\right). \quad \text{(B.24)}$$

Essa relação é útil para revelar que no caso particular em que a parte real do quatérnio é nula, $Q = (0, q)$, tem-se que $cos(\theta/2) = 0$. Logo, $\theta = \pi\,(2n + 1)$, $n \in Z$, descrevendo uma rotação de 180° sobre um eixo de rotação unitário q , $||q|| = 1$. Essa condição será de particular interesse na análise das leis de estimação da atitude e das leis de controle.

Assim como a representação em eixo-ângulo, a representação em quatérnio usa quatro parâmetros (um escalar e um vetor em R^3), o que é uma sobre-parametrização do espaço de rotação SO(3), já que pode-se representar a mesma rotação com três parâmetros (usando, por exemplo, ângulos de Euler). Na representação eixo-ângulo, uma rotação descrita por $(\theta, \hat{e})$ equivale as rotações descritas por $(2\pi n - \theta, -\hat{e})$ e $(\theta + 2\pi n, \hat{e})$, $n \in Z$.

Para qualquer destas soluções o quatérnio equivalente poderá ter apenas dois valores, Q ou $-Q$. Ou seja, a transformação $S^3 \rightarrow SO(3)$ é um mapeamento não injetivo dois-para-um.

$$\begin{aligned} Q &= \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\hat{e}\right) = \left(\cos\left(\frac{\theta + 2\pi n}{2}\right), \sin\left(\frac{\theta + 2\pi n}{2}\right)\hat{e}\right) \\ &= \left(\cos\left(\frac{2\pi n - \theta}{2}\right), \sin\left(\frac{2\pi n - \theta}{2}\right)(-\hat{e})\right) \ , n = 0,2,4,6 \ldots \end{aligned} \tag{B.25}$$

$$\begin{aligned} -Q &= \left(-\cos\left(\frac{\theta}{2}\right), -\sin\left(\frac{\theta}{2}\right)\hat{e}\right) = \left(\cos\left(\frac{\theta + 2\pi n}{2}\right), \sin\left(\frac{\theta + 2\pi n}{2}\right)\hat{e}\right) \\ &= \left(\cos\left(\frac{2\pi n - \theta}{2}\right), \sin\left(\frac{2\pi n - \theta}{2}\right)(-\hat{e})\right) \ , n = 1,3,5,7 \ldots \end{aligned} \tag{B.26}$$

Pode-se demonstrar isso também pela fórmula de Rodrigues:

$$\begin{aligned} (-q)^T(-q) &= q^T q \\ S(-q) &= -S(q) \ ; \ (-\mathrm{u})S(-q) = uS(q) \\ R(-Q) &= \left((-u)^2 - (-q)^T(-q)\right)I_3 + 2(-q)(-q)^T - 2(-\mathrm{u})S(-q) \\ &= (u^2 - q^T q)I_3 + 2qq^T - 2uS(q) = R(Q) \end{aligned} \tag{B.27}$$

Usar a representação em quatérnio pode ser vantajoso por ser de simples implementação computacional e muitas vezes simplificar provas matemáticas (ROBERTS, 2011). Para implementar um algoritmo de estimação ou controle de atitude deve-se conservar as propriedades da rotação (por exemplo, um vetor rotacionado deve preservar o valor do módulo). Utilizando quatérnios basta que a norma unitária do quatérnio seja preservada. Assim, um algoritmo que atualiza a estimação da atitude mantém as propriedades da rotação com uma simples normalização do quatérnio após a atualização. Isso é muito mais simples do que preservar as propriedades do $SO(3)$ em matrizes de rotação que necessitam de algoritmos de ortonormalização como o de Gran-Schmidt (ROBERTS,2011).