UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Mecânica

Hebert Wandick Parreira

Desenvolvimento de um rastreador estrelar para determinação de atitude de cubesats

Campinas

2022

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Mecânica

Hebert Wandick Parreira

Desenvolvimento de um rastreador estrelar para determinação de atitude de cubesats

Trabalho apresentado à Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Engenheiro de Controle e automação.

Orientador: Prof. Dr. Rodrigo Moreira Bacurau

Este exemplar corresponde à versão final da monografia apresentada pelo aluno Hebert Wandick Parreira, e orientada pelo Prof. Dr. Rodrigo Moreira Bacurau

Campinas

2022

Resumo

O objetivo deste trabalho é realizar o desenvolvimento de um sistema de rastreamento estrelar, com baixo custo, baixo consumo de energia e com volume mássico e peso reduzido. Estas restrições e objetivos se devem a aplicação desejada ao sistema, que será utilizado em cubesats.

Para a realização deste objetivo, foi realizada uma ampla pesquisa na literatura científica acadêmica a respeito do assunto em questão, além disso foi utilizado um conjunto de técnicas e ferramentas de programação para garantir a qualidade do sistema final.

Os resultados do trabalho estão divididos em duas partes, com a primeira constituindo um simulador estrelar e a segunda sendo o sistema de rastreamento estrelar em si. A primeira parte foi desenvolvida para auxiliar no desenvolvimento e averiguar a acurácia do sistema de rastreamento estrelar em ambiente simulado.

Palavras-chaves: rastreador estrelar; cubesat; determinador de atitude.

Abstract

The objective of this work is to develop a star tracker system, with low cost, low energy consumption and with reduced mass and weight. These restrictions and objectives are due system's desired application, which will be used in cubesats.

To achieve this objective, a wide research in the academic scientific literature about the subject was carried out, in addition, a set of programming techniques and tools was utilized to ensure the quality of the final system.

The results of the work are divided into two parts, with the first constituting a star simulator and the second being the star tracker system itself. The first part was developed to assist in the development and verify the accuracy of the star tracker system in a simulated environment.

Keywords: star tracker; cubesat; attitude determination.

Lista de ilustrações

Figura 2.1 – Comparação de Star Trackers comerciais. (a) Relação entre preço e precisão, (b) Relação entre energia consumida e precisão, (c) Relação entre massa e precisão. Fonte: (DIAZ, 2006) 8
Figura 2.2 – Etapas de funcionamento de um Star Tracker. Fonte: (FIALHO, 2017) 9
Figura 2.3 – Codificação de coordenadas na esfera celeste. Fonte: (CARVALHO, 2001) 11
Figura 2.4 – Inclinação órbita terrestre. Fonte: (PORTAL. . . ,) 11
Figura 2.5 – Frame inercial terrestre. Fonte: (DIAZ, 2006) 12
Figura 2.6 – Referencial inercial. Fonte: (CARVALHO, 2001) 12
Figura 2.7 – Sistema equatorial de coordenadas. Fonte: (CARVALHO, 2001) 13
Figura 2.8 – Sistema de coordenadas vetorial- cartesiano e sistema equatorial de coordenadas. Fonte: (CARVALHO, 2001) 13
Figura 2.9 – Referência do Cubesat. Fonte: (DIAZ, 2006) 14
Figura 2.10–Camera Frame. Fonte: (DIAZ, 2006) 15
Figura 2.11–Resumo de impetração da parte esquerda do catálogo Hipparcos. Fonte: (Esa, 1997) 16
Figura 2.12–Características de matriz representativa do catálogo estelar. Fonte: (CARVALHO, 2001) 16
Figura 2.13–Catálogo estelar representação em 2D. Fonte: (DIAZ, 2006) 17
Figura 2.14–Catálogo estelar representação em 3D. Fonte: (DIAZ, 2006) 17
Figura 2.15–Distorção radial. Fonte: (OZCAKIR, 2020) 18
Figura 2.16–Distorção tangencial. Fonte: (OZCAKIR, 2020) 19
Figura 2.17–Relação entre estrelas no FOV da câmera, Fonte: Autoria própria . . . 21
Figura 2.18–Relação entre estrelas no FOV da câmera, Fonte: Autoria própria . . . 22
Figura 2.19–Parte do arquivo de banco de dados de áreas e momentos angulares, Fonte: Autoria própria 23
Figura 2.20–Parte do arquivo de banco de dados de ângulos entre estrelas, Fonte: Autoria própria 23
Figura 2.21–Parte do arquivo de banco de dados de posição estrelar, Fonte: Autoria própria 24
Figura 2.22–Exemplo de Octo Tree, Fonte: (APPLE, 2020) 24
Figura 2.23–Exemplo de Octo Tree, Fonte: (APPLE, 2020) 25
Figura 2.24–Exemplo da construção de um K vector. Fonte: (MORTARI; NETA, 2000) 26

Figura 2.25–Casos Covid-19. Fonte: (VALENTE, 2020) 26
Figura 2.26–Curva de regressão, Fonte: Autoria Própria 26
Figura 2.27–Possíveis dias para um certo numero de casos, Fonte: Autoria Própria . 27
Figura 2.28–Pyramid, Fonte: (FIALHO, 2017) 28
Figura 2.29–Exemplo de Pyramid, Fonte: (FIALHO, 2017) 28
Figura 2.30–Exemplo de Pyramid, Fonte: Autoria Própria 29
Figura 3.1 – Simulação em projetor, Fonte (TAPPE, 2009) 32
Figura 3.2 – Meu do simulador, Fonte (CARVALHO, 2001) 33
Figura 3.3 – Visualização Frustum rectangular, Fonte: (FARLEX, 2019) 34
Figura 3.4 – Visualização da câmera na posição inicial 35
Figura 3.5 – Coordenadas do frame da camera. Fonte: (FARLEX, 2019) 35
Figura 3.6 – Camera frame at position,declination 0º, ascensão 0º, roll 0º. Fonte: Autoria própria 36
Figura 3.7 – Camera ang 30 37
Figura 3.8 – Camera ang 20 37
Figura 3.9 – Simulação em 3D, Fonte: Autoria própria 38
Figura 3.10–Simulação em 2D, Fonte: Autoria própria 38
Figura 4.1 – Visualização do erro espacial de distorção radial. Fonte: Autoria própria 40
Figura 4.2 – Visualização do frame da camera com estrelas conhecidas. Fonte: Autoria própria 41
Figura 4.3 – Visualização do erro espacial de distorção radial. Fonte: Autoria própria 42
Figura 4.4 – Resultado da correção radial. Fonte: Autoria própria 42
Figura 4.5 – Resultado da aplicação do algoritmo k vector para o banco de dados de ângulos. Fonte: Autoria própria 43
Figura 4.6 – Resultado obtido para o caso em que o sistema detectou incorretamente as estrelas presentes no FOV, Fonte: Autoria própria 46
Figura 4.7 – Resultado obtido para o caso em que o sistema detectou incorretamente as estrelas presentes no FOV, Fonte: Autoria própria 47
Figura 4.8 – Resultado obtido para o caso em que o sistema detectou incorretamente as estrelas presentes no FOV, Fonte: Autoria própria 47
Figura 4.9 – Resultado obtido para o caso em que o sistema detectou corretamente as estrelas presentes no FOV, Fonte: Autoria própria 48
Figura 4.10–Resultado obtido para o caso em que o sistema detectou corretamente as estrelas presentes no FOV, Fonte: Autoria própria 49
Figura 4.11–Resultado obtido para o caso em que o sistema detectou corretamente as estrelas presentes no FOV, Fonte: Autoria própria 49
Figura 4.12–Resultado geral de funcionamento do sistema, Fonte: (COLE; CRASSIDIS, 2006) 50

Lista de tabelas

Tabela 4.1 – Tempo de execução do algoritmo de análise de imagem, Fonte: Autoria própria 44
Tabela 4.2 – Tempo de execução da pesquisa no banco de dados, Fonte: Autoria própria 44
Tabela 4.3 – Acurácia do sistema, Fonte: Autoria própria 45

Sumário

1 Introdução . . . 5

1.1 Motivação . . . 5

1.2 Objetivos . . . 6

1.3 Estrutura . . . 7

2 Revisão bibliográfica . . . 8

2.1 Etapas de operação . . . 9

2.2 Esfera celeste . . . 10

2.3 Referencial Inercial Terrestre . . . 11

2.4 Referencial do cubesat . . . 14

2.5 Catalogo de estrelas . . . 15

2.6 Análise de imagem . . . 17

2.6.1 Distorções em imagens . . . 18

2.6.1.1 Distorção radial . . . 18

2.6.1.2 Distorção tangencial . . . 19

2.7 Detecção de estrelas . . . 19

2.7.1 Localização de círculos . . . 19

2.7.2 Transformação de coordenadas . . . 20

2.7.3 Cálculo do ângulos entre estrelas . . . 20

2.7.4 Cálculo da área antre estrelas . . . 21

2.7.5 Cálculo do momento entre estrelas . . . 22

2.8 Banco de dados . . . 22

2.8.1 Gerando o banco de dados . . . 23

2.8.1.1 Octree . . . 24

2.8.1.2 K vector . . . 25

2.8.2 Utilizando o banco de dados . . . 27

2.8.2.1 Pyramid . . . 27

3 Metodologia . . . 30

3.1 Criação do banco de dados . . . 30

3.2 Desenvolvimento de software . . . 30

3.2.1 *Test Driven Development* (TDD) . . . 30

3.2.2 *Model View Controller* (MVC) . . . 31

3.3 Simulador . . . 32

3.3.1 Recursos e características . . . 33

3.3.2 Projeção em perspectiva . . . 34

3.3.3 *Square Frustum* 34
3.3.4 Configurações de câmera 36
3.4 Arquivo de estrelas 38
3.5 Rotação 39
4 Resultados 40
4.1 Correção de camera 40
4.2 Aplicação do algoritmo k vector para pesquisa no banco de dados 43
4.3 Localização de estrelas 44
4.4 Códigos Fontes 50
Conclusão 52

Referências 54

1 Introdução

Um dos principais sistemas de um satélite é o sistema de determinação de atitude, o qual é parte do sistema de controle de atitude, que é responsável pela orientação espacial do satélite. Este sistema é de extrema importância para os satélites, pois, na maioria das suas aplicações, é requerido que as antenas ou câmeras do dispositivos apontem para regiões específicas da Terra. Como para tirar fotos da superfície terrestre e fornecer acesso de rede a estação fixa em solo, como por exemplo, é o caso do Starlink.

Durante este projeto, será desenvolvido sistema de rastreamento estrelar para cubesat, com aplicabilidade na indústria aeroespacial. O sistema será baseado em visão computacional. Pretende-se utilizar webcams convencionais e utilizar como unidade de processamento embarcados de baixo custo que executam Linux, como por exemplo a Raspberry Pi 4.

O foco desse projeto será no desenvolvimento dos algoritmos responsáveis por fazer a captura e análise das imagens, e determinar a posição angular do cubesat. Para testar o sistema, será desenvolvido uma simulação. Ela consistirá de um monitor juntamente de um software de simulação do céu estrelado a ser visualizado pelo dispositivo.

Essa simulação, será desenvolvida em linguagem Python e permitirá a rotação em todo o espaço com 360 graus de liberdade em todos os eixos, por fim será realizada uma validação final com uma webcam.

1.1 Motivação

Com o aprimoramento da eletrônica, os circuitos e sistemas presentes em satélites se tornaram menores, mais leves, mais baratos, rápidos, e com maior eficiência energética. Além disso, o desenvolvimento de uma padronização nas dimensões destes pequenos satélites possibilitou um decréscimo ainda maior de custo.

Em universidades e *StartUps* o estudo e desenvolvimento de cubesats vêm crescendo rapidamente, mesmo que os pequenos satélites apresentem limitações físicas e energéticas, o custo benefício em sua aplicabilidade é grande.

Outra limitação está na capacidade do satélite de se localizar e orientar no espaço, ou seja, controlar a sua atitude, que devido às restrições já mencionadas, costumam ser extremamente limitados ou mesmo inexistentes (DIAZ, 2006). Com isto, as possibilidades de aplicações destes cubesats tornam-se consideravelmente limitadas.

A primeira etapa para realização do controle de atitude, é identificar de forma confiável, precisa e contínua, a orientação do satélite (DIAZ, 2006). No espaço existem pontos de referência que podem ser utilizados para a determinação da atitude, como o Sol, Lua e a Terra, porém estas referências não são consistentes, já que o Sol pode estar encoberto, e a análise da superfície terrestre vista do espaço varia muito, devido a nuvens e outros fenômenos meteorológicos. Além disso, a análise de imagens complexas é custosa computacionalmente, devido às limitações de volume e energia, tornam a aplicação extremamente complicada.

Outra opção é utilizar o campo magnético da Terra, porém, a interferência eletromagnética é algo relativamente comum, uma vez que os próprios circuitos elétricos do satélite podem gerar interferências, o que inviabiliza a utilização dessa grandeza física para monitoramento de atitude de satélites.

Uma terceira opção é a utilização de uma câmera realizando a análise das estrelas, o fato do satélite estar no espaço faz com que as estrelas estejam na maioria do tempo no campo de visão do satélite (TAPPE, 2009).

Realizar o controle de atitude apenas utilizando IMU (*Inertial Measurement Unit*) é difícil, pois IMU são suscetíveis a erros de desvio de Offset, erros de Instabilidade, temperatura, são sensíveis a pancadas e vibrações (YOUNG, 2015). Desta forma, utiliza-se IMUs de alto custo, m sua maioria baseada em transdutores ópticos, como os IFOGs (*interferometric fiber optic gyroscopes*) e os RLGs (*ring laser gyros*), que são caros, e em sua maioria, grandes.

Apesar de IMUs de baixo custo não serem adequadas para identificação de atitute através da integração das velocidades angulares dos girsocópios, que devido a deriva relativamente elevada, resulta em erros de orientação consideráveis em poucos segundo, podem ser utilizados em conjunto com outras abordagens, como sensores de estrelas, para melhorar a acurácia e/ou aumentar a taxa de atualização através de fusão sensorial.

Devido a questões de custo e disponibilidade, para aplicação em cubsat dar-se preferência para dispositivos de prateleira (sensores e chips já prontos e vendidos em massa). Geralmente fazendo uso da tecnologia MEMS (*Micro Electro Mechanical Systems*), os quais são relativamente baratos, pequenos e possuem massa reduzida.

1.2 Objetivos

O objetivo deste trabalho é realizar o desenvolvimento de um sistema de rastreamento estrelar, que utiliza uma câmera comercial para detectar estrelas, aliado a um

algoritmo de rastreamento estrelar, que seja capaz de executar em um computador de bordo, com baixo custo, baixo consumo de energia e com volume mássico e peso reduzidos.

O desenvolvimento de tal sistema complexo exige a elaboração de múltiplos sistemas auxiliares, como o desenvolvimento da simulação inteiramente virtual do sistema, que é necessário para se realizar o testes dos algoritmos de rastreamento estrelar desenvolvidos. Após esta etapa, o sistema deve ser testado em simuladores com hardware real, para se verificar a viabilidade do sistema, e então o sistema é testado em um protótipo real. Devido a restrições de tempo neste trabalho é desenvolvido e testado apenas para o primeiro tipo de simulação, a simulação virtual.

1.3 Estrutura

Este trabalho esta organizado da seguinte forma, no capitulo2 é apresentado a revisão bibliográfica, da literatura existente sobre o tema, no capitulo3 é apresentado a metodologia utilizada para o desenvolvimento do projeto, explicando-se os detalhes do processo de implementação das técnicas apresentadas na revisão bibliográfica, no capitulo4 é os resultados obtidos com a implementação dos algoritmos Por fim no capitulo4.4 é apresentado a conclusão do trabalho.

2 Revisão bibliográfica

Existem várias formas de se medir a atitude, porém, a forma mais amplamente utilizada em satélites de grande porte são os seguidores de estrelas (*star trackers*). A tecnologia de tais sistemas vêm evoluindo nos últimos anos, com a rápida melhoria dos sensores CCD (*charge-coupled device*), e tecnologias de análise de imagem. Porém, erros são inerentes a qualquer medida, por mais pequeno que um erro possa ser, pode levar a uma identificação errônea de uma estrela, resultado em um erro completo de posicionamento.

Star Trackers funcionam capturando imagens de estrelas e comparando a tabelas salvas em memória (DIAZ, 2006). Desta forma, o satélite é capaz de identificar a sua atitude.

No entanto, os sistemas comerciais possuem preço elevado, consumo de energia elevado e massa elevada para pequenos satélites, como é mostrado na Figura 2.1.

Figura 2.1 – Comparação de Star Trackers comerciais. (a) Relação entre preço e precisão, (b) Relação entre energia consumida e precisão, (c) Relação entre massa e precisão. Fonte: (DIAZ, 2006)

2.1 Etapas de operação

As etapas de operação dos Star Tracks, em geral, seguem a estrutura apresentada na Figura 2.2.

Figura 2.2 – Etapas de funcionamento de um Star Tracker. Fonte: (FIALHO, 2017)

A primeira etapa de operação, consiste na aquisição da imagem a ser utilizada, para tal é utilizado um CCD acoplado a um conjunto de lentes. A escolha desse componentes deve levar em conta uma série de parâmetros, tais como: abertura do campo visual, precisão, volume, peso e outros a variar com a missão do cubesat (CARVALHO, 2001). Neste projeto a aquisição de imagens será feita com uma *webcam*.

O algoritmo de processamento juntamente com processamento de centroide, são responsáveis por localizar cada uma das estrelas e determinar sua localização e intensidade.

O algoritmo de identificação de estrelas é responsável por fazer a relação entre

as estrelas identificadas na observação e as estrelas contidas no banco de dados. A identificação de uma estrela envolve a análise das relações entre as diversas estrelas observadas, para tal utiliza-se algoritmos como de *Planar Triangles* (COLE; CRASSIDIS, 2006).

Por fim, é determinada a atitude do cubesat, que pode ser feita utilizando ângulos de Euler ou quaternions.

Para avaliar os algoritmos desenvolvidos, foi implementado um simulador estrelar, seguindo o método demonstrado por Tappe (TAPPE, 2009). Por fim, foram realizados testes com imagens reais obtidas por *webcams* ou câmeras de celulares.

2.2 Esfera celeste

Para aplicação no rastreador estelar, as estrelas são representadas através de uma esfera celeste, na qual as estrelas são distribuídas em uma esfera ao redor do centro, que na aplicação de satélites é a própria Terra. Devido a diferença de escalas da distância das estrelas da Terra, e o tamanho da Terra, pode-se considerar que qualquer ponto na Terra e na órbita terrestre, estão exatamente no centro da esfera. O erro gerado por tal simplificação só se tornaria visível em missões em que o veículo espacial se retirasse do sistema solar, o que não é caso para Cubesats atuais (FIALHO, 2017).

Também é completamente desprezado qualquer movimento que os astros tenham em relação uns aos outros, pois estes movimentos são praticamente nulos em nossas análises. Isto se deve o fato de que, o quanto maior for a distância do observador a um objeto, menor será a variação angular para um mesmo movimento linear do objeto, como por exemplo pode-se observar que aviões aparentam estar extremamente lento para um observador na Terra, No caso do sistema estrelar, esse efeito é ainda maior, permitindo desprezar esse movimento relativo entre as estrelas sem nenhum prejuízo prático a precisão (CARVALHO, 2001).

Nesta abordagem, algumas das características da Terra são representadas na esfera, como o eixo de rotação em torno de si (rotação) e os pólos geográficos, que são nomeados respectivamente eixo e pólos celestes. Com isto, facilita-se a localização dos astro na esfera. Para isto cria-se circunferências envolvendo a esfera, concorrendo nos polos (meridianos), com circunferências perpendiculares ao eixo de rotação (paralelos) (CARVALHO, 2001).

Figura 2.3 – Codificação de coordenadas na esfera celeste. Fonte: (CARVALHO, 2001)

Como referência tem-se o paralelo central, conhecido como paralelo do Equador, o meridiano de referência é o que contém o ponto vernal . O ponto vernal é o momento em que o Sol passa o Equador de Sul para Norte, isto ocorre pois a rotação da Terra em torno do próprio eixo está inclinada em relação ao plano da trajetória elíptica da Terra em torno do Sol.Como é visto na Figura 2.4.

Figura 2.4 – Inclinação órbita terrestre. Fonte: (PORTAL. . . ,)

2.3 Referencial Inercial Terrestre

Utilizando os conceito da Esfera Celeste, cria-se o referencial inercial terrestres, em que o eixo x está alinhado com vetor radial partindo do Sol em direção a Terra (linha de Áries), que é o equinócio de inverno, o eixo z é alinhado com o eixo de rotação da Terra, e o eixo y segue a regra da mão direita, conforme as figuras 2.5 e 2.6.

Figura 2.5 – Frame inercial terrestre. Fonte: (DIAZ, 2006)

Figura 2.6 – Referencial inercial. Fonte: (CARVALHO, 2001)

As estrelas são catalogadas utilizando os pontos de referência, utilizando-se de um sistema de coordenadas polares, com duas coordenadas angulares. Um dos ângulos é definido a partir dos meridianos, que é a ascensão reta , o ponto de ares é o marco zero, a ascensão da reta, que varia de 0 a 360 graus.

A outra coordenada polar é a declinação γ, que é o ângulo entre a estrela e o paralelo do equador, variando de -90 a 90 graus. A Figura 2.8 mostra sua representação.

Figura 2.7 – Sistema equatorial de coordenadas. Fonte: (CARVALHO, 2001)

A transformação do sistema de coordenadas polares para coordenadas cartesianas é feita com base na Figura 2.8.

Figura 2.8 – Sistema de coordenadas vetorial- cartesiano e sistema equatorial de coordenadas. Fonte: (CARVALHO, 2001)

Todas as estrelas catalogadas são descritas na superfície de uma esfera unitária com centro cociente a Terra, portanto, o módulo do vetor que sai do centro da esfera até a estrela é sempre 1,

$$\left|\overrightarrow{V}\right| = 1. \tag{2.1}$$

Dessa forma, desconsidera-se o valor do raio nas equações de transformação de coordenadas; a elaboração das equações é realizada através da análise das relações trigonométricas do sistemas, o que resulta em:

$$V_x = cos(\gamma)cos(\alpha), \tag{2.2}$$

$$V_y = cos(\gamma)sin(\alpha), \tag{2.3}$$

$$V_z = sin(\gamma). \tag{2.4}$$

As equação 2.2, 2.3 e 2.4 são utilizadas para se realizar a decomposição do vetor de coordenadas da estrela, está decomposição é feita em relação ao referencial apresentado na Figura 2.8.

2.4 Referencial do cubesat

O referencial do cubesat é alinhado geometricamente com o equipamento, dessa forma, ele representa a posição do cubesat em si, como pode ser visto na Figura 2.9.

Figura 2.9 – Referência do Cubesat. Fonte: (DIAZ, 2006)

O referencial da câmera tem os eixos x e y normal ao plano da imagem, a imagem captada pela câmera se encontra a uma distância unitária do cubesat no eixo

Z, o motivo de se utilizar uma distância unitária e demais informações sobre o frame de câmera, serão tratados no capítulo de projeções em perspectiva. Neste trabalho é considerado que o referencial do cubesat é o mesmo da câmera, como é visto na Figura 2.10.

Figura 2.10 – Camera Frame. Fonte: (DIAZ, 2006)

2.5 Catalogo de estrelas

A criação do catálogo é realizada por satélites especialmente criados e lançados com este propósito, como o é o caso do NASA I/239, que compõem o banco de dados do Hipparcos and Tycho. Este banco de dados é encontrado em (Esa, 1997).

O banco de dados possui uma tabela principal de informações, com o formato mostrado na Figura 2.11.

Number		Descriptor: epoch J1991.25				Position: epoch J1991.25		
HIP		RA	Dec	V		α (ICRS)	δ	
		h m s	±° ′ ″	mag		deg	deg	
1	2	3	4	5	6 7	8	9	10
94301		19 11 39.93	+19 25 54.2	9.32	H	287.916 356 71	+19.431 710 10	
94302		19 11 40.52	+56 51 32.7	5.13	H	287.918 837 62	+56.859 095 94	
94303		19 11 41.74	−33 42 31.2	10.70	G	287.923 923 90	−33.708 672 65	
*94304		19 11 41.95	−81 24 46.5	7.60	H	287.924 807 49	−81.412 923 39	
*94305		19 11 41.93	+09 57 06.2	8.44	H	287.924 721 08	+09.951 709 33	
94306	H	19 11 43.52	+17 53 01.0	8.20	G	287.931 328 93	+17.883 612 16	A
94307		19 11 44.21	−47 11 16.7	9.66	H	287.934 207 13	−47.187 963 08	A
94308		19 11 45.03	+26 14 57.6	7.80	H	287.937 607 07	+26.249 326 14	A
94309		19 11 45.35	−19 11 21.6	8.15	2 H	287.938 964 02	−19.189 330 69	
94310	H	19 11 45.76	−53 19 22.0	8.90	H	287.940 663 10	−53.322 766 02	A
94311		19 11 46.01	+31 17 00.5	5.93	1 H	287.941 712 76	+31.283 463 60	
94312		19 11 46.35	−31 07 45.8	12.40	2 H	287.943 111 87	−31.129 399 09	

Figura 2.11 – Resumo de impetração da parte esquerda do catálogo Hipparcos. Fonte: (Esa, 1997)

Essa tabela possui muitas informações que não serão utilizadas, incluindo uma quantidade de estrela muito grande, portanto o banco de dados foi filtrado com apenas alguns campos de dados restantes: número de catalogação, ascensão reta (em °), declinação (em °) e magnitude, que mostrado na Figura 2.12. Além disto foi realizado uma filtragem por estrelas com magnitude cinco ou inferior, o que significa que apenas as estrelas mais brilhantes permaneceram na base de dados do sistema. Para a visualização das estrelas restantes utiliza-se o sistema equatorial de coordenadas, resultando nos plots apresentados nas figuras 2.13 e 2.14.

Figura 2.12 – Características de matriz representativa do catálogo estelar. Fonte: (CARVALHO, 2001)

Figura 2.13 – Catálogo estelar representação em 2D. Fonte: (DIAZ, 2006)

Figura 2.14 – Catálogo estelar representação em 3D. Fonte: (DIAZ, 2006)

2.6 Análise de imagem

Analise de imagem é o primeiro passo na analise de dados de nosso sistema, no caso desse trabalho a analise deve ter a capacidade de localizar a posição correta das estrelas presentes no campo de visão da câmera. Erros nesta atapa da analise pode fazer o reconhecimento da posição angular do satélite se tornar impossível.

2.6.1 Distorções em imagens

As correções de distorções nas imagens captadas devem ser realizadas. Estas distorções se dividem em dois componentes, uma sendo a radial e outra tangencial, com a componente radial sendo predominante. Além disto, existem distorções de construção das lentes, distorções de desalinhamento da lente em relação ao centro da Matriz CCD da câmera.

2.6.1.1 Distorção radial

Distorção radial é vista quando utiza-se câmera com grandes ângulos e com pequena distância focal (MALLON; WHELAN, 2004). Os efeitos da distorção radial são representados na Figura 2.15.

Figura 2.15 – Distorção radial. Fonte: (OZCAKIR, 2020)

Esta distorção pode ser corrigida através de diferentes métodos, neste trabalho utiliza-se o método matemático descrito pelo (BRADSKI, 2000). Dessa forma a distorção radial é corrigida através das equações 2.5 e 2.6.

$$X_{corrigido} = X_{original}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \tag{2.5}$$

$$Y_{corrigido} = Y_{original}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6). \tag{2.6}$$

As equações 2.5 e 2.6 são aplicadas para cada pixel da imagem, sendo que o valor de r é a distancia em pixels do centro da imagem até o pixel em questão. O formato destas equações é polinomial, pois quanto maior o valor de r maior a distorção radial. Para se medir o erro é utilizado uma imagem de referência, com medidas conhecidas, como é visto na Figura 2.15. Em que se utiliza um tabuleiro de xadrez retor para se medir a distorção radial. Tal efeito polinomial do erro pode ser observado na seção 4.1.

Com o valor de k_1, k_2 e k_3 sendo os coeficientes da correção radial, Y original e X original são os valores de pixel da imagem medidos originalmente na imagem, Y

corrigido e X corrigido são os valores de pixel da imagem medidos após a correção radial, sendo os valores que a câmera deveria obter se não houvesse distorção radial.

2.6.1.2 Distorção tangencial

Distorções tangenciais ocorrem quando a câmera e o plano da imagem não estão em paralelo (OZCAKIR, 2020). Os efeitos da distorção tangencial são representados na Figura 2.16.

Figura 2.16 – Distorção tangencial. Fonte: (OZCAKIR, 2020)

Esta distorção é corrigida através das equações 2.7 e 2.8.

$$X_{corrigido} = X_{original}(1 + p_1 r^2 + p_2 r^4 + p_3 r^6), \tag{2.7}$$

$$Y_{corrigido} = Y_{original}(1 + p_1 r^2 + p_2 r^4 + p_3 r^6). \tag{2.8}$$

Nas equações 2.7 e 2.8 r é a distância em pixels do centro da imagem até o pixel em questão, p_1, p_2 e p_3 são os coeficientes da correção tangencial. Y

Resalta-se que a ferramenta de simulação estrelar desenvolvida para este trabalho não possui distorcão tangencial, pois a câmera e o plano da imagem estão paralelos.

2.7 Detecção de estrelas

2.7.1 Localização de círculos

Os métodos de detecção de círcurlos em geral consiste nas variações do método de Hough Transform (HT), como o standard Hough Transform, o Fast Hough Transform de de Li et al.

Para aplicar um método de Hough Transform, é necessário se aplicar um filtro para detecção de bordas na imagem, que filtra apenas os pixels que estão entre na borda das estrelas detectadas, pois o HT utiliza a posição destes pixels para calcular o possível centro do círculo. Neste trabalho, utiliza-se um filtro clássico de Canny, para realizar a detecção das bordas, pois o ganho em qualidade de detecção de outros algoritmos de detecção de borda, como o método de onda contínua, que utilizam recursos computacionais maiores, e não aumenta de forma satisfatória a qualidade de detecção de bordas, como é mostrado por Kobylin (KOBYLIN; LYASHENKO, 2014)

O HT faz utilização da equação do círculo, para fazer a detecção do círculo, o método consiste em se testar todas as possilidades de círculos possíveis para cada pixel na borda de cada ponto na imagem. Uma vez feito todos os câlculos para cada um dos pixels de borda, o mêtodo seleciona o circulo com maior recorrência em diferentes pixels. Yuen demonstra sua aplicação em diferentes variações (YUEN *et al.*, 1990).

2.7.2 Transformação de coordenadas

A localização das estrelas ocorre em coordenadas cartesianas em relação ao centro do campo de visão da câmera, utilizando o referencial já demonstrado, porém é mais conveniente que a localização seja representada através de coordenadas polares, para isso é necessário se realizar a transformação de coordenadas, representadas nas equações 2.9 e 2.10.

$$\theta = \begin{cases} \arctan\left(\frac{y}{x}\right), & \text{se } x >= 0 \\ \arctan\left(\frac{y}{x}\right) + \pi, & \text{se } x < 0, \end{cases} \tag{2.9}$$

$$r = \sqrt{x^2 + y^2}. \tag{2.10}$$

A variável **r** é então transformada em uma variável ângular, através de uma relação linear. Com os coeficientes sendo calculados através de uma imagem de referência, basicamente se faz uma contagem de quantos pixels que se precisa para se percorrer 1 grau de ângulo, para se realizar a medição deste coeficiente. Na Equação 2.11, r é quantidade de pixels medida na imagem, e C é o coeficiente de transformação,

$$r_{angular} = r * C_{converção}. \tag{2.11}$$

2.7.3 Cálculo do ângulos entre estrelas

O calculo da distância entre as estrelas presentes no FOV é realizado através da Equação 2.12. Além disto, é aplicado um valor máximo para limitar o ângulo entre as

estrelas, pois isto diminui a quantidade de relações armazenadas no banco de dados. Na Figura 2.17 é demonstrado as relações entre as estrelas.

Figura 2.17 – Relação entre estrelas no FOV da câmera, Fonte: Autoria própria

$$dist_{angular} = \sqrt{r^2_{estrela_1} + r^2_{estrela_2} - 2 * r_{estrela_1} * r_{estrela_2} * cos(\theta_{estrela_1} - \theta_{estrela_2})}. \quad (2.12)$$

Com r1 e r2 sendo as distâncias de cada estrela ao centro da image captada pela câmera, e θ1 e θ2 sendo os ângulos de cada estrela em relação ao eixo x do frame da imagem, visto na Figura 2.10.

2.7.4 Cálculo da área antre estrelas

Além da relação angular entre duas estrelas, pode-se calcular a área entre 3 estrelas, isto sé faz necessário pois existe uma quantidade grande de estrelas que possuem uma mesma quantidade de estrelas vizinhas a uma mesma distância angular. Portanto utiliza-se uma terceira estrela para se calcular a área entre as estrelas, que é utilizada para refinar as possíveis estrelas no campo de visão. Para esta calculo é utilizado a fórmula de Heron 2.13, que pode ser calculada através da distância angular entre as estrelas. Que já foi calculada na Equação 2.12, previamente executada pelo algoritmo.

$$area = \sqrt{p(p-a)(p-b)(p-c)}, \quad (2.13)$$

, em que **p** é a metade do perímetro do triângulo, e **a**, **b** e **c** são as distâncias entre as estrelas:

$$p = \frac{a+b+c}{2} \quad (2.14)$$

Esta divisão rm duas equações existe pois, o calculo do momento entre a estrelas também utiliza o valor **p**.

2.7.5 Cálculo do momento entre estrelas

Uma filtragem de terceiro grau é realizada para se diminuir ainda mais a quantidade de possíveis estrelas, para isso utiliza-se o momento entre as estrelas, que é calculado através da Equação 2.15,

$$momento = \frac{area(a^2 + b^2 + c^2)}{36}. \tag{2.15}$$

Pode-se utilizar o momento como uma elemento extra de filtragem pois triângulos com areas semelhantes, podem possuir momentos completamente diferentes, como demonstrado na Figura 2.18.

Figura 2.18 – Relação entre estrelas no FOV da câmera, Fonte: Autoria própria

Nota-se que esta filtragem é de custo computacional baixo, e utiliza apenas estrelas já analisadas anteriormente, portanto é uma passo extra eficaz e de baixo custo computacional.

2.8 Banco de dados

Feita a analise das relações geométricas entre as estrelas presentes no FOV, com as informações, de momentos, áreas e momentos angulares, é então inicializado a pesquisa no banco de dados interno do cubesat, para verificarão de possíveis estrelas.

Devido às limitações energéticas do sistema, é necessário que a estrutura do banco de dados seja otimizada para a resolução do problema em questão. Além disto o próprio processo de geração do banco de dados que também requer uma otimização, para que o mesmo seja gerado em um tempo aceitável. Pois, a combinação de 3 estrelas diferentes para realização do calculo da area e do momento pode gerar um problema de $O(n^3)$.

Além disto a técnicas de busca e analise de resultados também devem ser otimizadas e levar levar em consideração múltiplos fatores. Cabe ressaltar que a operação do sistema de orientação e controle do cubesat possui duas fases distintas de operação, a fase em que não há

informação previa de localização e a fase em que já há informação previa de localização. Neste trabalho é apenas abordada a fase em que não há informação previa de localização.

2.8.1 Gerando o banco de dados

Após a análise de imagem o sistema precisa compar os resultados das leituras com um banco de dados de estrelas, para que seja possível identificar a posição do cubesat, relacionando a posição da estrela observada a posição do cubesat, o banco de dados é composto por algumas partes, sendo elas:

- **Relações Áreas e Momentos angulares**: Lista ordenada da combinação de 3 estrelas. Contendo as informações de área e momento angular, juntamente com o identificador das estrelas Figura 2.19.
- **Relações de Ângulos entre estrelas**: Lista ordenada da combinação de 2 estrelas. Contendo as informações de ângulo entre as estrelas, juntamente com o identificador das estrelas Figura 2.20.
- **Relação de id e posição estrelar**: Este é o mesmo banco de dados utilizado para gerar a simulação, contendo as informações de id e posição estrelar Figura 2.21.

	A	B	C	D	E
1	area	star_1	star_2	star_3	moment
2	0.001072	0	16	1529	6.31E-07
3	0.000353	1	4	1602	4.44E-08
4	0.000604	1	3	4	2.28E-07
5	0.000563	1	4	14	2.35E-07
6	0.002684	1	3	14	1.45E-06
7	0.001177	2	8	11	2.85E-07

Figura 2.19 – Parte do arquivo de banco de dados de áreas e momentos angulares, Fonte: Autoria própria

	A	B	C
1	ang	index_s1	index_s2
2	0.023549	0	16
3	0.096387	0	1529
4	0.015597	1	4
5	0.04522	1	1602
6	0.079225	1	3
7	0.090091	1	14

Figura 2.20 – Parte do arquivo de banco de dados de ângulos entre estrelas, Fonte: Autoria própria

	A	B	C	D
1	Numero de catalc	Ascensao reta(alpha)	Declinacao (delta)	Magnitude visual(V)
2	122	0.39937928	-77.06529438	4.78
3	154	0.48996506	-6.01397169	4.37
4	301	0.93488487	-17.33597002	4.55
5	355	1.1255098	-10.50949443	4.99
6	443	1.33394073	-5.70783255	4.61
7	677	2.09653333	29.09082805	2.07
8	746	2.29204036	59.15021814	2.28
9	765	2.35224949	-45.74698836	3.88
10	910	2.81628415	-15.46732287	4.89
11	1067	3.30895828	15.18361593	2.83
12	1168	3.65045047	20.2066972	4.79

Figura 2.21 – Parte do arquivo de banco de dados de posição estrelar, Fonte: Autoria própria

2.8.1.1 Octree

Nota-se que a combinação de 3 estrelas diferentes para realização do cálculo da área e do momento pode gerar um problema de complexabilidade O(n^3), Porém, apenas estrelas próximas uma as outras que podem geram uma combinação de area valida para a analise. Nota-se que apenas estrelas dentro de um circulo ao redor da estrela central podem gerar uma combinação de area valida para a analise, como visto na Figura 2.22.

Figura 2.22 – Exemplo de Octo Tree, Fonte: (APPLE, 2020)

Para se localizar de forma eficiente as estrelas ao retor de um circulo no espaço tridimensional, é utilizado a estrutura de dados Octree, que é uma estrutura de dados que divide o espaço em 8 subespaços, como visto na Figura 2.23. Dessa forma, calcula-se apenas areas e ângulos entre estrelas que estão dentro do circulo ao redor da estrela central.

Figura 2.23 – Exemplo de Octo Tree, Fonte: (APPLE, 2020)

2.8.1.2 K vector

O método **k vector** foi desenvolvido para refinar a faixa de busca em vetores ordenados de dados (MORTARI; NETA, 2000). Para que este método seja utilizado é necessário que todos os elementos do banco de dados sejam conhecidos previamente, pois o método consiste na criação de um modelo matemático relacionando a informação a ser encontrada, com o index do banco de dados. Para isso, a análise é inciada com a ordenação crescente dos dados. Apesar de não estritamente necessário, recomenda-se que seja plotado os em um gráfico cartesiano, como na Figura 2.24. Pois a elaboração de uma equação matemática que melhor represente a curva a ser encontrada, é mais fácil de ser feita com a visualização do gráfico.

A Figura 2.24 mostra um exemplo de um gráfico de uma função linear, onde a equação matemática que melhor representa a curva é a equação da reta, porém isto não é uma regra, pois a curva pode ser uma função quadrática, cúbica, logarítmica, exponencial, etc.

No caso apresentado na Figura 2.24, pode-se utilizar, uma relação linear obtida através do método dos mínimos quadrados pelas equações 2.16, 2.17 e 2.18,

$$\beta = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{2.16}$$

$$\alpha = \bar{y} - \beta * \bar{x}, \tag{2.17}$$

$$y = \alpha + \beta * x. \tag{2.18}$$

Para explicar o funcionamento do método, será utilizado o exemplo da Figura 2.25. Neste exemplo, objetivo é descobrir o dia em que um numero qualquer de novos casos de covid-19

Figura 2.24 – Exemplo da construção de um K vector. Fonte: (MORTARI; NETA, 2000)

ocorreram, dessa forma se utiliza um método de regressão que melhor se encaixa na curva, em teremos como entrda **x** o numero de casos e como saída **y** o dia.

Figura 2.25 – Casos Covid-19. Fonte: (VALENTE, 2020)

Seja a curva vermelha presente na Figura 2.26 a regressão que melhor satisfez aproximação da curva da Figura 2.25, está modelagem matemática apresenta de erros em sua aproximação, como pode se observar no desvio da curva vermelha em relação a curva azul. Portando além do valor da regressão, é necessário calcular o erro da regressão 2.27, para que então possa se refinar a pesquisa em apenas uma parte dos set de dados. Quando então algoritmo de busca binária é utilizado para finalizar a busca.

Figura 2.26 – Curva de regressão, Fonte: Autoria Própria

Figura 2.27 – Possíveis dias para um certo numero de casos, Fonte: Autoria Própria

Dessa forma o método consegue diminuir o tempo de pesquisa, ja que um vetor de tamanho **n**, com todas os valores conhecidos pode ter seu tempo busca reduzido do habitual O(n*log(n)) para O(2*error*log(2*error)), diminuí-se o tamanho do vetor de pesquisa a 2 vezes o erro máximo da linearização, dessa forma os algorítimos de busca binária só precisam executar a pesquisa neste vetor selecionado. Neste trabalho, considero-se que o erro associado a aproximação é, o maior erro entre todos o valor calculado para todos o pontos da curva.

2.8.2 Utilizando o banco de dados

Após o sistema de analise de imagem, medir os ângulos, areas e momentos triangulares entre as estrelas presentes na imagem, o sistema aplica um algótico de seleção de possibilidades, onde é selecionado as estrelas que possuem os ângulos e momentos triangulares mais próximos dos valores esperados.

2.8.2.1 Pyramid

Neste trabalho é utilizado o algorítimo de Pyramid, o algoritmo funciona da seguinte forma, primeiro é selecionado 3 estrelas, α, βe γ, com os valores de ângulos juntamente com o erro máximo da medição, é aplicado ao k-vector, que gera 3 listas diferentes possibilidade.

Com estas listas filtradas é crias-se uma lista com todas as possíveis combinações de 3 estrelas, então é analisado o valor de area e momento triangular das estrelas no FOV, filtrado possíveis combinações que não se encaixam nos valores esperados.

Caso a analise resulte em mais de uma combinação possível, é realizado a analise de um segundo triangulo, o qual possui pelo menos uma estrela em comum com o primeiro triangulo, então o segundo triangulo é analisado da mesma forma que o primeiro. Então os resultados dos dois triângulos são comparados, para selecionar apenas as possibilidades que estão presentes nos dois resultados.

Pode se visual o processo nas figuras **??** e **??** e **??**.

Figura 2.28 – Pyramid, Fonte: (FIALHO, 2017)

Figura 2.29 – Exemplo de Pyramid, Fonte: (FIALHO, 2017)

Figura 2.30 – Exemplo de Pyramid, Fonte: Autoria Própria

No exemplo da Figura 2.30, a estrela que faz parte dos dois triângulos pode ser 13, 2 segundo a analise do triangulo amarelo, segundo a outra analise de pode-ser 13, 4 portando está estrela deve ser a 13.

3 Metodologia

Este trabalho teve como fogo o desenvolvimento pratico dos algoritmos apresentados na seção 2, de forma que o leitor possa executar os algoritmos e verificar o seu funcionamento, por conta própria. Para que isto os dois programas executáveis foram desenvolvidos,e testados em ambiente Windows 11, pois é o ambiente mais utilizado mundialmente. Cabe salientar que todos os algoritmos foram construídos pensando-se um ambiente embarcado linux, e este são perfeitamente executáveis em tal ambiente. Porém os algoritmos foram entregados a interfaces gráficas para facilitar a execução e a visualização dos resultados.

3.1 Criação do banco de dados

A criação dos bancos de dados foi feita utilizando notebooks juypter, que faz o import dos dados e dos algoritmos implementados, esta escolha se deu por ser uma ferramenta de fácil utilização e que permite a visualização dos dados e dos resultados de forma mais intuitiva. Além disto todo os arquivos de dados estão em formato csv ou json para facilitar a leitura e a manipulação dos dados, por seres humanos, cabe ressaltar que aplicação real de dais bancos de dados se da em arquivos binários, por limitações de espaço e de velocidade de leitura de dados.

3.2 Desenvolvimento de software

3.2.1 *Test Driven Development* (TDD)

O Desenvolvimento Orientado a Testes, consiste em desenvolver testes separados para cada parte de um software como é o caso das funções, os testes devem garantir que o código funcione e cumpra a tarefa desejada.

Para aplicá-lo, primeiro cria-se o teste, dessa forma garante-se que o que durante o desenvolvimento do código o resultado inicialmente esperado seja obtido, dessa forma o teste falha na primeira vez que é executado, já que este está testando o recurso que ainda não existe, em seguida é desenvolvido o recurso de forma a fazer o teste passar, então repete-se este ciclo função por função.

Caso seja localizado um *bug* no software, primeiramente o desenvolvedor deve implementar um teste que consiga replicar o erro em específico, só então a realizar a correção, dessa forma se garante a correta implementação da solução, assim como o novo software deve manter-se passando nos testes anteriores. Algumas das vantagem de tal método são:

- Facilidade em focar em problemas específicos de desenvolvimento;

- Códigos são fáceis de refatorar;
- Facilidade em corrigir bugs, mesmo durante o desenvolvimento;
- Capacidade em garantir do que está funcionando e como está funcionando;
- Dividir de forma mais clara cada parte do código;
- Descobrir falhas de forma prematura durante o processo de desenvolvimento;
- Maior organização de código;

Como este projeto é desenvolvido em python, é utilizado o framework *pytest* que realiza e execução de testes em cada parte do código separadamente, neste trabalho, apenas os módulos do software com as implementações matemáticas, foram testados por tal metodologia. Os códigos relacionados a interfaces gráficas não foram desenvolvidos dessa forma, isto ocorreu pois, apesar de ser possível a utilização TDD, os testes para tails componentes de software são de criação lenta e difícil, optando-se então por apenas realizar *checks* visuais.

3.2.2 *Model View Controller* (MVC)

MVC é um arquitetura de software desenvolvida para otimizar o processo de desenvolvimento de interfaces gráficas, com foco em diminuir o tempo de manutenção do código, facilitando a depuração de elementos visuais do sistema, também facilitando a interação de múltiplos desenvolvedores e proporciona uma maior clareza de código exigindo um baixo nível de acoplamento entre as partes.

Para isso, o MVC se divide em 3 componentes principais, o *model*, o *controller* e o *view*. As regras de negócio estão contidas no model assim como as informações necessárias para execução do programa o *controller* realizar a interface entre o *model* e *view*, que contém as especificações de como os usuários interagem com o sistema.

O *view* é responsável por decide como apresentar as informações, por exemplo se uma lista de dados sera apresentada em forma de tabela ou gráfico, ou inputs de informações, que pode ser linhas de textos, botoes, *check box* etc.

Quando o usuário interage com este *viwer*, é gerado um evento que é tratado pelo controlador, que se preciso for irá formatar os dados gerados pela interface, então o controlador transfere o dado para o model, que executa a aplicação em si, como por exemplo realizando equações matemáticas, carregando arquivos e demais lógicas de negócio.

O *model* também é responsável por armazenar as informações sobre o estado de execução do sistema.

No simulador estrelar aqui desenvolvido utiliza-se 3 *views* separados, para gerenciar diferentes janelas de usuário, com um único modelo que é responsável por armazenar a lógica de negócios, que consiste na execução dos códigos de simulação.

3.3 Simulador

O objetivo da simulação é permitir testes em ambiente controlado, de forma a se averiguar a precisão, acurácia e velocidade de processamento da posição por parte do Star Tracker. Para isto, a simulação implementa modelos matemáticos que representem o sistema real o mais fielmente possível.

O programa foi desenvolvido de forma a permitir ao leitor testar diferentes cenários de forma rápida e intuitiva, podendo assim carregar diferentes arquivos como figuração de câmera, diferentes arquivos de dados de estrelas. Além disto, o simulador permite a visualização tanto em 3D como em 2D, para que o usuário possa ter uma visão mais clara do que está acontecendo. A visualização em 2D permite se comparar os resultados com os resultados obtidos em outros trabalhos.

Atualmente, o simulador apenas salva o frame desejado em um arquivo de imagem, porém, o sistema foi construído a permitir a visualização em um segundo monitor do simulação em tempo real. Isto permitirá possíveis fases de simulações com hardware real, como pode ser visto na Figura 3.1 e na Figura 3.2.

Figura 3.1 – Simulação em projetor, Fonte (TAPPE, 2009)

Figura 3.2 – Meu do simulador, Fonte (CARVALHO, 2001)

3.3.1 Recursos e características

GUI (*Graphical User Interface*): A simulação é controlada através de uma GUI criada para facilitar o controle e a execução dos testes, também é utilizada para facilitar a visualização e a correção de possíveis erros e *bugs* no software e no desenvolvimento do trabalho em geral. A implementação dos módulos de funcionamento do software segue conceitos dos TDD (*Test driven development*), juntamente da arquitetura de software MVC (*model view controller*), para organizar o funcionamento da interface em si (FITZPATRICK, 2022).

Gerador de campo de visão: Esta é a função principal do software de simulação, que é gerar uma janela com o campo de visão que teoricamente seria visto pela câmera, a imagem gerada é monocromática com as estrelas projetadas sobre a superfície plana do monitor. Neste trabalho as estrelas simuladas são apenas círculos brancos em um fundo preto, porém são estrelas retiradas de uma base de dados reais.

Carregamento de arquivos: Para facilitar a mudança de parâmetros durante os testes e simulações , o software carrega todas as informações inerentes a configuração e execução dos testes, através de caixas de diálogo que pedem a seleção dos arquivos de simulação. Para uma maior modularidade, cada informação está dividida em arquivos diferentes, com arquivos separados para configuração de câmera, base de dados de estrelas e sequência de movimentos a ser realizada.

Elementos auxiliares de visualização: Além das funcionalidades básicas, o software implementa recursos de configuração de aspectos visuais. Apesar de não serem estritamente necessários são de extrema importância para análises menos rigorosas das informações, como por exemplo, o plot 3D, a projeção cartográfica em 2D. Com opções de visualização da posição das câmeras e demais auxílios.

Controles manuais e automáticos: O software possui controles manuais de rotação, que permitem a rotação da simulação em todos os 3 eixos de liberdade, através de con-

figurações na GUI, além disto o usuário pode preparar uma sequência de movimentos e suas respectivas durações previamente.

Salvar frames: O usuário pode salvar frames de imagem em arquivos png para facilitar a utilização posterior da posição.

3.3.2 Projeção em perspectiva

A projeção descreve matemática como representar um ponto com 3 graus de liberdade, no espaço bidimensional do monitor e do frame da câmera. Essa transformação pode ser realizada através de matrizes, que levam em consideração elementos como o ângulo focal da lente, a resolução e proporção da imagem.

3.3.3 *Square Frustum*

Frustum retangular é um espaço que contém a linha de visão de um visualizador como se este possuísse um ângulo de visão retangular, que é caso de câmeras com CCD retangulares, para melhor entender este conceito recomenda-se a observação da Figura 3.3.

Figura 3.3 – Visualização Frustum rectangular, Fonte: (FARLEX, 2019)

A existência de um plano ao fundo na cena (*far clip*), se deve ao equacionamento que leva os pontos presentes na cena 3D para o frame 2D da câmara, com adição do *far clip* o custo de processamento requerido pelo programa diminui, desta forma sendo necessário equacionamento do frame da câmera, apesar deste conceito não existir em cameras reais.

Neste trabalho, convencionou-se que o ângulo de visão da câmera esteja apontado para o positivo do eixo x, como é mostrado na Figura 3.4.

Figura 3.4 – Visualização da câmera na posição inicial

As setas coloridas presentes na imagem representam as coordenadas do frame da câmera, que seguem um esquema de cores usual para esse tipo de aplicação, na Figura 3.5 é demonstrado as relações X,Y e Z com as cores:

Figura 3.5 – Coordenadas do frame da camera. Fonte: (FARLEX, 2019)

Com isso, se desenvolveu a seguinte equação matricial da câmera, seguindo os passos mostrados por Brendan Galea em seu video (GALEA, 2021):

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} \frac{f+n}{f-n} & 0 & 0 & \frac{-2fn}{f-n} \\ 0 & \frac{h}{w*tan(\frac{\theta}{2})} & 0 & 0 \\ 0 & 0 & \frac{1}{tan(\frac{\theta}{2})} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{3.1}$$

Em que, **f** é a distância do *far plane* ao ponto (0,0,0), que neste caso é sempre unitário já que a estrelas estão presentes em um círculo unitário, Θ é o ângulo de abertura da câmera, **w** é a quantidade de pixels na horizontal camera, **h** é quatidade e pixel na vertical da câmera por fim **n** é a distância do near plane, que neste caso é o coseno de $\frac{\Theta}{2}$.

Nota-se que a matriz de transformação da câmera é uma matriz 4X4 isso ocorre pois as equações de transformação são realizadas através de coordenadas homogêneas e não coordenadas cartesianas tradicionais.

Além da matriz, existe uma equação de *clipping* que determina se uma estrela deve ou não aparecer no frame. Devido a simplicidade da simulação implementada nesse trabalho, essa equação se tornou apenas uma verificação se a estrela se encontrava na metade da esfera celeste em que x é positivo.

O resultado dessa simulação pode ser visto na Figura 3.6.

Figura 3.6 – Camera frame at position,declination 0º, ascensão 0º, roll 0º. Fonte: Autoria própria

3.3.4 Configurações de câmera

No simulador, o leitor pode carregar um arquivo de câmera, que é um arquivo json, com as seguintes informações:

- **ang**: Abertura da lente, nas figuras **??** e **??** ve-se o impacto dessa variável;
- **w**: Quantidade de pixels que a câmera possui de altura;
- **h**: Quantidade de pixels que a câmera possui de largura;

Neste trabalho, a altura e a largura em pixel são apenas utilizadas para calcular a proporção do frame gerado, com o angulo de abertura da lente, responsável por controlar o tamanho geral do FOV da câmera.

Figura 3.7 – Camera ang 30

Figura 3.8 – Camera ang 20

3.4 Arquivo de estrelas

A estrutura desse arquivo ja foi explicada na seção 2.5. Nas figuras 3.9 e 3.10 são apresentados os resultados do carregamento do catalogo NASA I/239, filtrado para estrelas com magnitude menor que 5.

Figura 3.9 – Simulação em 3D, Fonte: Autoria própria

Figura 3.10 – Simulação em 2D, Fonte: Autoria própria

Os resultados foram muito semelhantes aos obtidos por (DIAZ, 2006), fato que corrobora a correta leitura dos arquivos de dados.

3.5 Rotação

As operações de rotação são realizadas através de quaternions, através de um quatérnion normalizado qR que em conjunto de seu conjugado é capaz de rotacionar um ponto qualquer, com isso, é possível realizar a rotação de uma condenada A para um condenada B, pela equação 3.2 que é apresentada por Ben-Ari em seu artigo (BEN-ARI, 2014).

$$q_b = q_R q_a q_R \tag{3.2}$$

4 Resultados

4.1 Correção de camera

Nesta seção apresenta-se os resultados dos trabalhos realizado a cerca da correção de efeitos de camera. Apesar do trabalho utilizar apenas simulações, esta simulação também apresenta distorções. Como observa-se na figura 4.1 as estrelas mais distantes do centro do frame da camera estão distante do plano da imagem, o que gera o e efeito de distorção radial.

Figura 4.1 – Visualização do erro espacial de distorção radial. Fonte: Autoria própria

Para realizara corre da correção no simulador, foi utilizado uma base da dados de calibração da camera, que consiste em estrelas com relação angular conhecida, neste caso estrelas com 5 graus de separação angular entre si, o que resulta no frame Visto na figura 4.2.

Figura 4.2 – Visualização do frame da camera com estrelas conhecidas. Fonte: Autoria própria

Os dados da figura 4.2 são colocados no algoritmo de detecção de centroides para encontrar a distancia medida das estrelas até o centro da imagem, além disto é calculado a média das distancias medidas de estrelas com mesma distancia até o centro da imagem, para reduzir o erro de medição. Com isto podemos comparar ao valor esperado caso a camera não apresentasse distorção radial, neste trabalho considera-se que a medida real é o valor medido da estrela da estrela mais proxima ao centro da imagem. Com isto pode-se comparar o valor medido com o valor esperado, que é visto na figura 4.3.

Figura 4.3 – Visualização do erro espacial de distorção radial. Fonte: Autoria própria

Como o esperado o erro segue uma reta polinomial, como explicado na seção 2.6.1.1, com os representados mostrados na figura 4.4. Com o polinômio de ajuste resultante possuindo as constantes 4.1,4.2,4.3,

$$K1 = -9.6415774825692e - 09 \tag{4.1}$$

$$K2 = -2.0788174598076455e - 15, \tag{4.2}$$

$$K3 = 2.910928744425903e - 22. \tag{4.3}$$

Aplicação das correções resulta na figura 4.4, que mostra que a correção foi bem sucedida, pois o erro que estava em até 120 pixel agora está em até 1 pixel.

Figura 4.4 – Resultado da correção radial. Fonte: Autoria própria

4.2 Aplicação do algoritmo k vector para pesquisa no banco de dados

A plicação do algoritmo k vector para criação de um banco de dados mais eficiente, foi realizada com a lista de ângulos entre estrelas, na Figura 4.5 é mostrado o resultado para a aplicação utilizando um polinômio de grau 2.

Originalmente o vetor possui mais de 400 mil elementos, com a utilização do algoritmo k vector, o vetor de pesquisa é reduzido a 2 * o erro máximo, que de neste caso se torna um vetor menor de 4 mil elementos. O que é uma redução para menos de 1% do tamanho original.

Figura 4.5 – Resultado da aplicação do algoritmo k vector para o banco de dados de ângulos. Fonte: Autoria própria

4.3 Localização de estrelas

Os resultados obtidos em simulação, indicam que o sistema tem potencial para ser utilizado em cubesats reais como foi proposto inicialmente, pois o algoritmo se mostra rápido e robusto, conseguindo indentificar as estrelas presentes no FOV(*Field of View*) na maioria dos testes. Na tabela 4.1 é possível observar o tempo em que o algoritmo leva para executar, a análise da imagem em si, o que envolve a aplicação do filtro do borda de Canny, a utilização da transformada de Hough e a detecção de círculos, e o calculo das relações angulares e geométricas entre as estrelas.

A tabela 4.2 mostra o tempo de execução da busca no banco de dados, resultando que esta aplicação realiza a busca para o caso em que o cubesat não possui nenhuma informação anterior sobre a sua posição.

Cabe ressaltar que a terminação final da atitude do cubesat ainda exige a implementação de um algoritmo que relacione a estrela visualizada com a sua posição no espaço.

Roll (dec º)	**Acensão (dec º)**	**Declinação (dec º)**	**Tempo analise de imagem (s)**
0	0	0	1.760
191	170	79	0.770
11	31	44	0.903
92	105	50	–
264	159	-63	6.393

Tabela 4.1 – Tempo de execução do algoritmo de análise de imagem, Fonte: Autoria própria

Roll (dec º)	**Acensão (dec º)**	**declinação (dec º)**	**Tempo de busca (s)**
0	0	0	0.362
191	170	79	3.457
11	31	44	1.132
92	105	50	–
264	159	-63	2.116

Tabela 4.2 – Tempo de execução da pesquisa no banco de dados, Fonte: Autoria própria

A linha em que o tempo de análise de imagem é --"indica que o sistema não consegui determinar quais estrelas estavam sendo analisada no FOV. Junto a análise de tempo de execução, é realizada a analise de acurácia do sistema, verificando se o sistema detectou corretamente as estrelas presentes no FOV.

A analise é realizada da seguinte forma, primeiramente gera-se um frame aleatório, através do *software* de simulação, após isto, o frame é analisado pelo *software* do cubesat que gera uma lista de IDs das possíveis estrelas encontradas no FOV. Com está informação é então criado um novo *set* de estrelas contendo apenas as estrelas localizadas pelo *software* do cubesat.

Este novo *set* refinado de estrelas é carregado no *software* de simulação e é então gerado um novo frame, quando então é comparado se o novo frame é compatível com o frame

gerado anteriormente. A Tabela 4.3 mostra os resultados obtidos, pode-se ter 3 resultados distintos:

- **Correto**: o sistema detectou corretamente as estrelas presentes no FOV;
- **Incorreto**: o sistema detectou incorretamente as estrelas presentes no FOV;
- **Inconclusivo**: o sistema não detectou todas as estrelas presentes no FOV.

Roll (dec º)	**Acensão (dec º)**	**declinação (dec º)**	**Resultado**
0	0	0	Correto
191	170	79	Correto
11	31	44	Incorreto
92	105	50	Inconclusivo
264	159	-63	Correto

Tabela 4.3 – Acurácia do sistema, Fonte: Autoria própria

Para reforçar os resultados obtidos as figuras 4.6, 4.7, 4.8, 4.9, 4.10 e 4.11 mostram os resultados obtidos para os casos em que o sistema detectou incorretamente e corretamente as estrelas presentes no FOV.

Para se considerar que o sistema detectou corretamente ou não as estrelas presentes no FOV, é realizado a construção de um *sub-set* apenas com as estrelas detectadas, se o novo frame gerado a partir do *sub-set* possuir as estrelas selecionadas no mesmo ponto do frame anterior, então o teste é considerado como correto, caso contrário é considerado como incorreto. Teste inconclusivo é quando o sistema não retorna nenhuma estrela em especifico como solução.

Figura 4.6 – Resultado obtido para o caso em que o sistema detectou incorretamente as estrelas presentes no FOV, Fonte: Autoria própria

Figura 4.7 – Resultado obtido para o caso em que o sistema detectou incorretamente as estrelas presentes no FOV, Fonte: Autoria própria

Figura 4.8 – Resultado obtido para o caso em que o sistema detectou incorretamente as estrelas presentes no FOV, Fonte: Autoria própria

Figura 4.9 – Resultado obtido para o caso em que o sistema detectou corretamente as estrelas presentes no FOV, Fonte: Autoria própria

Figura 4.10 – Resultado obtido para o caso em que o sistema detectou corretamente as estrelas presentes no FOV, Fonte: Autoria própria

Figura 4.11 – Resultado obtido para o caso em que o sistema detectou corretamente as estrelas presentes no FOV, Fonte: Autoria própria

O resultado obtido para o caso em que o sistema detectou incorretamente as estrelas presentes no FOV, pode ser explicado pelo fato por erros na correção das distorções da imagem, assim como pequenos erros na localização dos centroides das estrelas. Isto é decorrentes de erros no algorítimo de detecção de bordas aliado ao Hough Transform.

Porém, tal problema já era esperado, pois trabalhos como o de (COLE; CRASSIDIS, 2006), já haviam demonstrado que start trackers possuem erros em algumas situações, como pode ser visto na Figura 4.12, porem estes erros são corrigidos em um momento posterior com a analise do proximo frame.

Figura 4.12 – Resultado geral de funcionamento do sistema, Fonte: (COLE; CRASSIDIS, 2006)

4.4 Códigos Fontes

Os trabalhos desenvolvidos neste projeto estão disponíveis em 3 repositórios no GitHub, sendo eles:

- ***star tracker cubesat***: Contém a implementação dos algorítimos que rodam diretamente no Cubesat, juntamente dos códigos da interface utilizada para testar estes algorítimos, disponível em: (PARREIRA, 2022b);
- ***star tracker simulator***: Contém a implementação do simulador implementado em Python, juntamente do executável desta simulação, disponível em: (PARREIRA, 2022d);

- ***star tracker documentation***: Contém a documentação do projeto, incluindo o código fonte em latex, do presente documento, assim como outros dados auxiliares, como os dados NASA I/239, e demais documentos disponível em: (PARREIRA, 2022a).

Além disto também foi disponibilizado um vídeo demonstrando o funcionamento do sistema, disponível em: (PARREIRA, 2022c).

Conclusão

Nesta seção são sumarizados os principais resultados obtidos com esse projeto, juntamente com as principais conclusões e perspectivas futuras, e pontos a serem melhorados, o que envolve melhorias e adições ao Star Tracker em si, mas também envolvem a integração do sistema aqui desenvolvido com outros outros componentes do sistema de orientação espacial do cubesat.

Além disso, alguns erros de arquitetura do sistema cometidos pelo autor, não permitiram a analise com grandes conjuntos de dados, com diferentes frames, porém o sistema se mostra promissor.

Trabalhos futuros

- **Melhorias na implementação do algoritmo de correção de distorções**: Apesar da correção da distorção de pixel parecer aceitável, pelos resultados apresentados na seção anterior, a incidência de erros na resposta final de detecção de estrelas, indica que esta etapas pode conter erros. Uma das possíveis soluções, é a utilização da biblioteca open CV para realizar esta etapa, pois ela já possui algorítimos para resolução de distorções. Optou-se inicialmente não utilizada por conter uma serie de recursos matemáticos ainda não completamente entendidos pelo autor.

- **Utilização do triângulos esféricos**: Neste trabalho utiliza-se triângulos planares, porém a utilização de triângulos esféricos aumenta a precisão do sistema, como é explicado por Cole (COLE; CRASSIDIS, 2004).

- **Refatoração da estrutura de testes**: A atual estrutura de testes é pensada para ser de fácil compressão, e permitir o teste de diferentes senários pelo usuário. Porém o ideal séria uma estrutura de testes que permitisse a realização de testes em massa, para que seja feita uma analise estatista como é realizada por (COLE; CRASSIDIS, 2004), o que não é possível com a estrutura atual. Isto não se deve a limitações de algorítimo, pois o algoritmo é capaz de realizar a detecção em tempo pequeno, assim como pode ser executado em múltiplos processos em paralelo, com a limitação sendo um erro estrutural de organização de testes. Portanto, a refatoração destes códigos se faz necessária para um melhor desenvolvimento de etapas futuras.

Perspectivas Futuras

- **Desenvolvimento de um simulador para teste de hardware**: Esta etapa já foi descrita na Seção 3.3.
- **Construção física de um sistema de câmera**: A construção de um sistema físico, para deste mais fieis a realidade também se faz necessária, nota-se que ao longo deste trabalho todos os códigos foram testado em Raspberry pi 4, o que facilita muito no desenvolvimento de etapas futuras.
- **Integração com filtro de Kalman**: Com o protótipo físico em mãos pode-se iniciar a integração com outros sistema de medição para se refinar a precisão.

Referências

APPLE, D. *GKOctree*. 2020. Urlhttps://developer.apple.com/documentation/gameplaykit/gkoctree. Citado 3 vezes nas páginas 3, 24 e 25.

BEN-ARI, M. A tutorial on euler angles and quaternions. *Weizmann Institute of Science: Rehovot, Israel*, 2014. Citado na página 39.

BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. Citado na página 18.

CARVALHO, G. B. *Levantamento de Técnicas de Identificação de Estrelas e Desenvolvimento de um Ambiente de Simulação e Testes para Análise de seus Desempenhos em Aplicações Espaciais*. Dissertação (Mestrado) — Master Thesis,(INPE-8307-TDI/766), 2001. Citado 9 vezes nas páginas 3, 4, 9, 10, 11, 12, 13, 16 e 33.

COLE, C.; CRASSIDIS, J. Fast star pattern recognition using spherical triangles. v. 3, 08 2004. Citado na página 52.

COLE, C.; CRASSIDIS, J. Fast star-pattern recognition using planar triangles. *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, v. 29, 01 2006. Citado 3 vezes nas páginas 4, 10 e 50.

DIAZ, K. D. *PERFORMANCE ANALYSIS OF A FIXED POINT STAR TRACKER ALGORITHM FOR USE ONBOARD A PICOSATELLITE*. Dissertação (Mestrado) — California Polytechnic State University San Luis Obispo, 2006. Citado 9 vezes nas páginas 3, 5, 6, 8, 12, 14, 15, 17 e 39.

Esa, . VizieR Online Data Catalog: The Hipparcos and Tycho Catalogues (ESA 1997). *VizieR Online Data Catalog*, p. I/239, fev. 1997. Citado 3 vezes nas páginas 3, 15 e 16.

FARLEX. *View Frustum*. Farlex, 2019. Disponível em: <https://encyclopedia2.thefreedictionary.com/View+frustum>. Citado 3 vezes nas páginas 4, 34 e 35.

FIALHO, M. A. A. *Improved star identification algorithms and techniques for monochrome and color star trackers*. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais São José dos Campos, Brazil, 2017. Citado 5 vezes nas páginas 3, 4, 9, 10 e 28.

FITZPATRICK, M. *Create GUI Applications Book*. 5th. ed. [S.l.]: Martin Fitzpatrick, 2022. Citado na página 33.

GALEA, B. *The Math behind (most) 3D games - Perspective Projection*. 2021. <https://www.youtube.com/watch?v=U0_ONQQ5ZNM&list=WL&index=8&t=479s>. Citado na página 35.

KOBYLIN, O.; LYASHENKO, V. Comparison of standard image edge detection techniques and of method based on wavelet transform. IJAR, 2014. Citado na página 20.

MALLON, J.; WHELAN, P. Precise radial un-distortion of images. In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*. [S.l.: s.n.], 2004. v. 1, p. 18–21 Vol.1. Citado na página 18.

MORTARI, D.; NETA, B. k-vector range searching technique. v. 105, 01 2000. Citado 3 vezes nas páginas 3, 25 e 26.

OZCAKIR, E. *Camera Calibration with OpenCV*. 2020. Disponível em: <https://medium.com/@elifozcakiir/camera-calibration-with-opencv-9fb104fdf879#:~:text=Tangential%20distortion%20occurs%20when%20the,straight%20lines%20to%20appear%20curved.&text=*%20x%20*%20y%5D-,x%2C%20y%20%E2%80%94%20undistorted%20pixels%20that%20are%20in%20image%20coordinate%20system,distortion%20coefficients%20of%20the%20lens.> Citado 3 vezes nas páginas 3, 18 e 19.

PARREIRA, H. *Star Tracker Cocumentation*. 2022. Disponível em: <https://github.com/HebertWP/star-tracker-documentation>. Citado na página 51.

PARREIRA, H. *star tracker cubesat*. 2022. Disponível em: <https://github.com/HebertWP/star-tracker-cubesat>. Citado na página 50.

PARREIRA, H. *Star Tracker exaplantion*. 2022. Disponível em: <https://www.youtube.com>. Citado na página 51.

PARREIRA, h. *Star Tracker Simulator*. 2022. Disponível em: <https://github.com/HebertWP/star-tracker-simulator>. Citado na página 50.

PORTAL do Astronomo,. <http://portaldoastronomo.org/2021/03/equinocio-vernal-ou-para-qual-lado-e-para-cima/>. Accessed: 2022-08-29. Citado 2 vezes nas páginas 3 e 11.

TAPPE, J. A. *Development of star tracker system for accurate estimation of spacecraft attitude*. Monterey, California. Naval Postgraduate School, 2009. Disponível em: <http://hdl.handle.net/10945/4335>. Citado 4 vezes nas páginas 4, 6, 10 e 32.

VALENTE, J. *https://agenciabrasil.ebc.com.br/saude/noticia/2020-07/curva-casos-confirmados-covid-19-volta-a-subir-no-brasil*. 2020. Urlhttps://agenciabrasil.ebc.com.br/saude/noticia/2020-07/curva-casos-confirmados-covid-19-volta-a-subir-no-brasil. Citado 2 vezes nas páginas 4 e 26.

YOUNG, K. *Characterization Tests of IMUs for Small Satellite Implementation*. Tese (Doutorado) — San Jose State University, 2015. Citado na página 6.

YUEN, H.; PRINCEN, J.; ILLINGWORTH, J.; KITTLER, J. Comparative study of hough transform methods for circle finding. *Image and Vision Computing*, v. 8, n. 1, p. 71–77, 1990. ISSN 0262-8856. Disponível em: <https://www.sciencedirect.com/science/article/pii/026288569090059E>. Citado na página 20.