Determinação de Atitude e Posição de um Quadrirrotor Utilizando Marcos Artificiais e Visão Computacional

Rafael Costa Fernandes
Programa de Pós Graduação em Engenharia Mecânica
Universidade Federal do ABC
Santo André, Brasil
costa.fernandes@ufabc.edu.br

***Resumo*—A determinação de atitude e posição de um quadrirrotor é um problema recorrente e amplamente discutido. Neste trabalho foi desenvolvido um algoritmo de determinação baseado em marcos artificiais e visão computacional. Os resultados obtidos foram comparados com as técnicas mais comuns de determinação de atitude e posição de um quadrirrotor, utilizando sensores *MEMS*.**

***Palavras-Chave*—Atitude, Posição, Quadrirrotor, Visão Computacional, Controle**

I. Introdução

A determinação precisa de atitude e posição de um quadrirrotor é um passo indispensável para um controle eficiente da plataforma, visto que todo controlador depende de informações do estado do sistema para decidir uma ação que leve ao alvo desejado, e se as informações dos estados não forem corretas, o controle será influenciado por estes erros.

Segundo [1], sensores inerciais são utilizados há décadas na indústria aeroespacial, e ultimamente são utilizadas para os mais diversos objetivos, visto que devido a avanços na tecnologia *MEMS* o preço destes sensores diminuíram exponencialmente.

Um quadrirrotor normalmente tem três sensores *MEMS* principais, um acelerômetro, que mede as acelerações do corpo, e consequentemente também o vetor gravidade, um giroscópio, que mede as velocidades angulares do corpo e um magnetômetro, que mede um vetor de norte magnético, em alguns quadrirrotores também há um sensor GPS, que mede a posição e velocidade do quadrirrotor no espaço.

Mesmo os melhores sensores *MEMS* tem erros que interferem na utilidade dos dados, os erros mais comuns são o ruído branco e o desvio de tendência, sendo que o último é o erro mais problemático [1]. Existem técnicas que reduzem o efeito do ruído do sensor na estimação do estado, como filtros de Kalman, que é um estimador de variância mínima para sistemas dinâmicos [2]. Este tipo de estimador pode se aproximar mais do valor real da medida, a partir do conhecimento a priori dos ruídos do sensor e de diversas medidas consecutivas. Um filtro estendido de Kalman pode também estimar a tendência dessas medidas.

Neste trabalho será discutido a utilização de um sistema de sensoriamento híbrido (fig. 1), que utiliza, além dos sensores *MEMS* já presentes no quadrirrotor, marcos artificiais (do inglês *artificial landmarks*) para a determinação de atitude e posição. Os dados obtidos serão comparados com as técnicas mais comuns de determinação de atitude e posição utilizando apenas os sensores a bordo.

Foi utilizada a biblioteca *OpenCV* para o tratamento e a análise das imagens obtidas na simulação em ambiente 3D. *OpenCV* é uma biblioteca de código aberto focada em visão computacional e aprendizado de máquina. *OpenCV* foi desenvolvido para fornecer uma infraestrutura para aplicações de visão computacional e acelerar a implementação de algoritmos de percepção de máquina [3].

Segundo [4], podemos dividir as tecnologias de posicionamento em duas categorias, posicionamento relativo e posicionamento absoluto. Posicionamento absoluto significa que a posição atual não depende da posição anterior, enquanto posicionamento relativo significa que a posição atual depende da posição anterior.

A determinação de atitude e posição por marco artificial depende de uma câmera montada no corpo do quadrirrotor, normalmente apontada para baixo, e pode ser considerada um tipo de tecnologia de posicionamento absoluto, visto que a medida atual não depende da medida anterior, eliminando os problemas de desvio de tendência e deriva.

O sistema de sensoriamento proposto teve bons resultados, em média uma ordem de magnitude mais preciso em comparação com os sensores convencionais, sem tendência de deriva, mantendo sua eficiência independente do tempo decorrido do sistema. Uma desvantagem do sensoriamento proposto é o tempo computacional, que é pelo menos uma ordem de magnitude maior, em comparação com os outros sensores.

II. Metodologia

Como o objetivo deste trabalho é apenas demonstrar que esta é uma técnica válida, foi escolhida a abordagem de simular completamente o sistema em ambiente computacional,

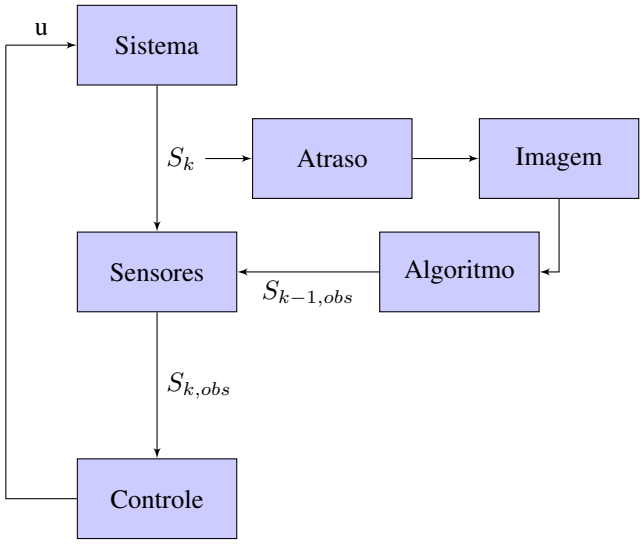

Figura 1. Diagrama Completo do Sistema Proposto

a priori evitando gastos com equipamentos e possíveis falhas catastróficas.

A. Simulação da Dinâmica

As equações matemáticas que regem a dinâmica do quadrirrotor foram baseadas nas equações da referência [5]. Tais equações não são o foco deste trabalho, mas é interessante citar que contemplam:

- Arrasto Translacional e Rotacional (Eq. 1 e 3)
- Posição e atitude real no sistema corpo e inercial (Eq. 2)
- Efeito giroscópico das hélices na dinâmica do corpo (Eq. 4)
- Matriz completa de momentos de inércia (Eq. 6)

As equações finais de movimento do quatrirrotor:

$$\begin{bmatrix} f_{empuxo} \\ m_{empuxo,x} \\ m_{empuxo,y} \\ m_{empuxo,z} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

$$\begin{bmatrix} f_{xb} \\ f_{yb} \\ f_{zb} \end{bmatrix} = f_{motores} - f_{arrasto} = \\ = \begin{bmatrix} 0 \\ 0 \\ f_{empuxo} \end{bmatrix} - \rho C_d l D \begin{bmatrix} |v_x|v_x \\ |v_y|v_y \\ 2|v_z|v_z \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = R_b^I \cdot \begin{bmatrix} f_{xb} \\ f_{yb} \\ f_{zb} \end{bmatrix}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \frac{1}{M} - \begin{bmatrix} 0 \\ 0 \\ G \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} m_{arrasto,x} \\ m_{arrasto,y} \\ m_{arrasto,z} \end{bmatrix} = -\frac{1}{3} D^4 l C_d \begin{bmatrix} |w_x|w_x \\ |w_y|w_y \\ 2|w_z|w_z \end{bmatrix} \tag{3}$$

$$w_r = -w_1 + w_2 - w_3 + w_4$$

$$\begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} = \begin{bmatrix} m_{empuxo,x} \\ m_{empuxo,y} \\ m_{empuxo,z} \end{bmatrix} + \\ + \begin{bmatrix} -w_x I_R \omega_r \\ w_y I_R \omega_r \\ 0 \end{bmatrix} + \begin{bmatrix} w_{arrasto,x} \\ w_{arrasto,y} \\ w_{arrasto,z} \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} \dot{w_x} \\ \dot{w_y} \\ \dot{w_z} \end{bmatrix} = J^{-1} \left(\begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} - \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \times J \cdot \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \right) \tag{5}$$

$$\begin{bmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \begin{bmatrix} 0 & -w_x & -w_y & -w_z \\ w_x & 0 & w_z & -w_y \\ w_y & -w_z & 0 & w_x \\ w_z & w_y & -w_x & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{6}$$

Onde u_i e w_i são força e velocidade angular do respectivo motor i e ρ, C_d, l, D e R_b^I são respectivamente a densidade do ar, o coeficiente de arrasto, a espessura do braço do quadrirrotor, a distância do motor ao centro de gravidade e a matriz de rotação do sistema de coordenadas corpo para o sistema de coordenadas inercial. Substituindo as constantes físicas podemos integrar as equações 2-6 e obter todos os estados do sistema.

Para poder comparar a efetividade da solução proposta, foi desenvolvido um sistema de sensoriamento que se comporta similarmente a sensores reais. Neste sistema são modelados ruído, magnitude da deriva de tendência dos sensores. Também foi modelado neste sistema um integrador por coluna, comumente utilizado.

O resultado final é uma simulação que se comporta com semelhança de um quadrirrotor real, incluindo certas interações com o ambiente e mantendo todas as não linearidades da sua dinâmica, considerando também os erros de sensoriamento e erros de integração.

B. Simulação de Sensores e Algoritmos de Determinação Convencionais

Para cumprir o objetivo deste trabalho foi desenvolvido um algoritmo de simulação de sensores, que se comporta de maneira similar a sensores reais no corpo do quadrirrotor. Acelerômetros, giroscópios e magnetômetros tem duas características importantes: Ruído e Deriva de Tendência. Normalmente, o ruído pode ser modelado como ruído branco (ou ruído Gaussiano) e o fabricante disponibiliza a dimensão deste ruído na ficha de dados do sensor.

A simulação dos sensores é importante pois os métodos mais comuns de determinação de posição e atitude de um quadrirrotor dependem exclusivamente dos sensores a bordo do quadrirrotor e todos estes sensores estão suscetíveis a ruído e deriva de tendência (do ingles *bias drift*).

Na figura 2 podemos observar um esquema convencional de sensoriamento e determinação de atitude. Na figura observamos os estados reais $\dot{q}_k$, e $\ddot{X}_k$ sendo observados pelos sensores, resultando nos estados observados $\dot{q}_{k,obs}$, e $\ddot{X}_{k,obs}$. Estes estados observados são dependentes também do ruído presente nos sensores.

Podemos observar também que há blocos de integração no diagrama, esta operação é retroalimentada pela sua própria resposta do instante anterior.

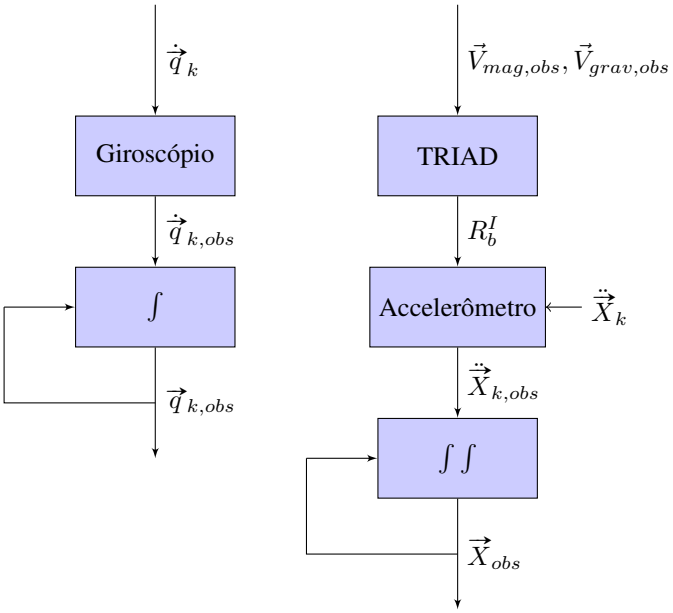

Figura 2. Diagrama de Blocos do Sensoriamento Convencional

É importante ressaltar que a deriva de tendência destes sensores é um fator determinante numa estimação correta do estado do sistema, visto que para a determinação de posição e atitude os valores obtidos dos sensores serão integrados no tempo (uma vez para o giroscópio e duas vezes para o acelerômetro), gerando um erro que se soma com ele mesmo com o passar do tempo do sistema, gerando deriva.

O desvio de tendência é uma variável mais complicada de se estimar apenas com informações dos sensores, visto que depende de diversos fatores externos, como por exemplo diferenças no fornecimento de energia do sensor, diferenças de temperatura, campos magnéticos externos etc. Para este trabalho o desvio de tendência foi iniciado aleatoriamente, com valor máximo de 1% do ruído. O valor de deriva é somado ao valor de tendência para cada passo de tempo do sistema i.e, a deriva do sensor cresce a cada intervalo em um valor constante.

O algoritmo de determinação de atitude do sistema é o TRIAD [8]. Este algoritmo fornece uma matriz de rotação do sistema de coordenadas do corpo para o sistemas de coordenadas inercial a partir das medidas do vetor gravidade e do vetor magnético. Ambos vetores são obtidos com os sensores a bordo do quadrirrotor.

A partir da matriz de rotação podemos obter as acelerações do corpo no sistema de coordenadas inercial, integrar estes valores uma vez para obter as velocidades e mais uma vez para obter as posições do sistema, conforme equações 7-9.

$$\ddot{\vec{X}}_{k,obs} = R_b^I \vec{A} \tag{7}$$

$$\dot{\vec{X}}_{k,obs} = \dot{\vec{X}}_{k-1,obs} + \ddot{\vec{X}}_{k,obs} \cdot \Delta t \tag{8}$$

$$\vec{X}_{k,obs} = \vec{X}_{k-1,obs} + \dot{\vec{X}}_{k,obs} \cdot \Delta t \tag{9}$$

Da mesma maneira podemos determinar a posição angular do sistema, na equação 10.

$$\vec{q}_{k,obs} = \vec{q}_{k-1,obs} + \dot{\vec{q}}_{k,obs} \cdot \Delta t \tag{10}$$

C. Controlador

O controle de um quadrirrotor não é uma tarefa simples. A dinâmica do sistema tem 6 graus de liberdade e 12 estados, o sistema é sub-atuado e instável. No decorrer dos anos diversas técnicas de controle foram desenvolvidas, desde controles por PID em regiões lineares do problema até controles utilizando técnicas de SDRE. O controlador utilizado neste trabalho foi desenvolvido como um dos objetivos da tese de mestrado sobre o tema, mas não cabe uma discussão aprofundada neste trabalho.

O controlador utilizado neste trabalho é um algoritmo de aprendizado de máquina e tem a liberdade de atuar diretamente nos quatro motores do quadrirrotor, em um intervalo limitado de força. O alvo deste controle é levar todos os estados a zero (posição, velocidade, ângulos de Euler e velocidade angular).

O modelo de rede neural utilizado tem 20293 parâmetros variáveis, treinados seguindo uma política de otimização por proximidade (do inglês *Proximal Policy Optimization - PPO*), que é um algoritmo de aprendizado de máquina relativamente novo, e que trouxe melhorias ao algoritmo de política de otimização por região de confiança (do inglês *Trust Region Policy Optimization - TRPO*) [6].

O treinamento do controlador foi finalizado com aproximadamente 6 horas em um processador *i5-4460*, em cada passo de treino o sistema inicia em um estado aleatório entre -2.5 e 2.5 metros (ou metros por segundo) nas posições e velocidades

nos três eixos de coordenadas, entre -0.5 e 0.5 radianos (ou radianos por segundo) nas posições e velocidades angulares em ângulos de Euler. O objetivo é levar todos os 12 estados do quadrirrotor para zero, o treinamento foi considerado finalizado quando 95% das simulações de validação atingiram um erro com norma menor ou igual a 0.01 unidades.

O modelo de controle obtido deste treinamento tem um bom comportamento, diversas vezes utilizando zonas de não linearidade do modelo para uma convergência mais rápida para o alvo, demonstrando que pode ser usado com eficácia para a tarefa proposta neste trabalho.

D. Desenvolvimento do Ambiente em Três Dimensões

Para poder utilizar a ferramenta de visão computacional foi necessário simular uma câmera no corpo do quadrirrotor. Para isso foi desenvolvido um ambiente baseado em *Panda3D*, que é uma biblioteca em *Python* utilizada principalmente para jogos digitais, neste ambiente foi possível renderizar um quadrirrotor em um cenário fictício, com toda a sua dinâmica herdada das equações de movimento, com uma câmera anexada em seu corpo, apontada diretamente para baixo. Nas figuras 3 e 4 podemos observar a visão da câmera anexada ao quadrirrotor e o quadrirrotor no seu ambiente virtual, respectivamente.

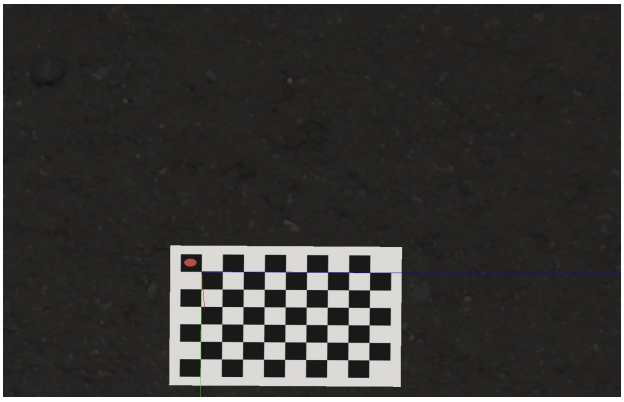

Figura 3. Visão da câmera anexada ao quadrirrotor.

Figura 4. Visão do quadrirrotor em seu ambiente virtual.

Este ambiente também foi desenvolvido para ser utilizado com outras técnicas de visão computacional, como aprendizado de máquina por imagens, aprendizado de máquina baseado em preferências humanas, técnicas SLAM, técnicas de prevenção de colisão etc. ou simplesmente para demonstração de trajetórias.

Com o ambiente renderizando uma imagem para cada passo de tempo de simulação, a renderização é recuperada da *RAM* e enviada ao *OpenCV*, no qual a imagem é tratada e utilizada no algoritmo de determinação de posição e atitude.

E. Calibração da Câmera

Para que a câmera possa ser utilizada como um sensor visual de determinação de posição e atitude a primeira etapa a ser realizada deve ser sua calibração. Só com uma câmera calibrada podemos saber a relação exata entre o tamanho de uma figura em uma imagem e seu tamanho real.

A calibração é um processo que envolve duas etapas, determinação de distorções e determinação da matriz de parâmetros intrínsecos, que tem a forma das equações 11 e 12.

$$dist_{cam} = \begin{bmatrix} k_1 & K_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{11}$$

$$mtx_{cam} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{12}$$

Para a calibração da câmera do ambiente virtual foi utilizado o algoritmo da referência [7]. Uma aproximação feita à este algoritmo é a consideração que a câmera do ambiente 3D é perfeitamente plana (sem distorção), levando todos os coeficientes da equação 11 à zero. Esta hipótese é perfeitamente plausível, visto que distorções geralmente só aparecem em câmeras que dispõem de lentes óticas, que não é o caso de uma câmera virtual.

O algoritmo de calibração tira 70 fotos do mesmo padrão xadrez utilizado no algoritmo de sensoriamento híbrido. Cada uma dessas fotos tem uma pose ligeiramente diferente da anterior. Podemos observar uma dessas poses na figura 5.

Dadas as poses é utilizado o algoritmo *findChessboardCorners* para detectar os vértices do tabuleiro de xadrez em cada uma dessas imagens. Todos os vértices obtidos das 70 imagens são utilizados na função *calibrateCamera* que retorna a matriz de parâmetros intrínsecos da câmera.

F. Sistema de Determinação de Atitude e Posição por Marcos Artificiais

Para a determinação de atitude do quadrirrotor foi utilizado o algoritmo *findchessboardcorners*, em conjunto com o algoritmo *solvepnp* do *OpenCV*. Dada uma imagem qualquer, o primeiro algoritmo é capaz de procurar e retornar a posição de cada vértice de uma imagem com formato de um tabuleiro de xadrez, se houver um tabuleiro na imagem. Enquanto o algoritmo solvePnP pode, dado as constantes de calibração da

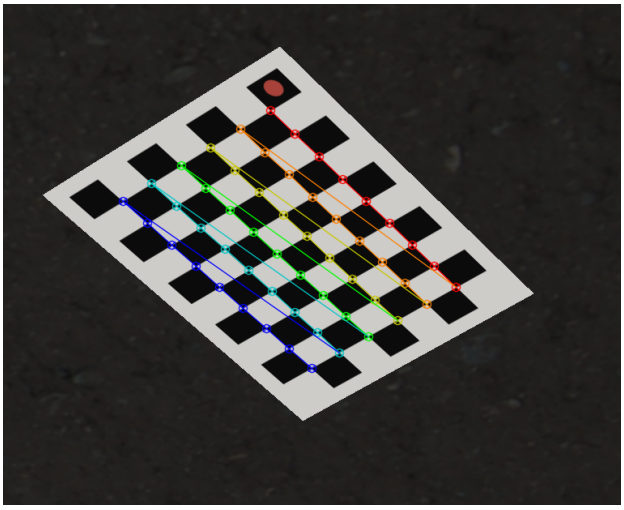

Figura 5. Exemplo de uma pose de calibração.

câmera utilizada no processo de aquisição de imagem e os vértices obtidos no algoritmo de detecção de padrão xadrez, obter a posição do padrão xadrez em coordenadas físicas, a partir do sistema de coordenadas da câmera.

O algoritmo retorna a distância e o vetor de rotação do padrão, chamados de *tvecs* e *rvecs*. Esses vetores devem ser convertidos para o sistema de coordenadas direito e o eixo Z deve ser invertido, visto que o *OpenCV* trabalha com um sistema de coordenadas esquerdo com eixo Z apontado para baixo.

Após estas conversões, temos um vetor de distância apontando da câmera ao objeto e um vetor de rotação do objeto, ambos no sistema de coordenadas da câmera. Como o objetivo é um vetor apontando do objeto à câmera e o vetor de rotação da câmera, ambos no sistema de coordenadas do objeto (que está posicionado convenientemente na origem do sistema inercial) devemos realizar algumas operações de transformação, estas podem ser observadas nas equações 13-16.

A complementação com outros sensores também se torna necessária para processadores de voo com menor poder computacional, reduzindo a frequência de análise de imagem e mantendo a frequência de controle do sistema utilizando os outros sensores a bordo, como será discutido no próximo tópico.

$$rvecs_I = \begin{bmatrix} rvecs_x \\ rvecs_y \\ -rvecs_z \end{bmatrix}, \quad \begin{bmatrix} \phi_{cam} \\ \theta_{cam} \\ \psi_{cam} \end{bmatrix} = Euler_{321}(rvecs_I) \tag{13}$$

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = - \begin{bmatrix} \phi_{cam} \\ \theta_{cam} \\ \psi_{cam} \end{bmatrix} \tag{14}$$

$$R_c^I = R_{321}(\psi_I, \theta_I, \phi_I) \tag{15}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot R_c^I \cdot \begin{bmatrix} tvecs_x \\ tvecs_y \\ tvecs_z \end{bmatrix} \tag{16}$$

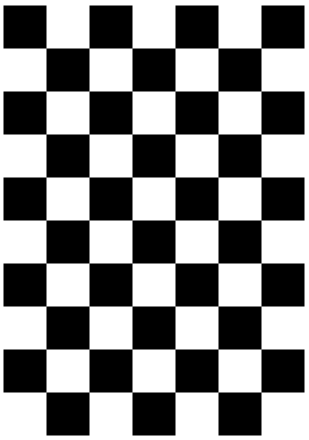

Figura 6. Marco Artificial Utilizado no Algoritmo.

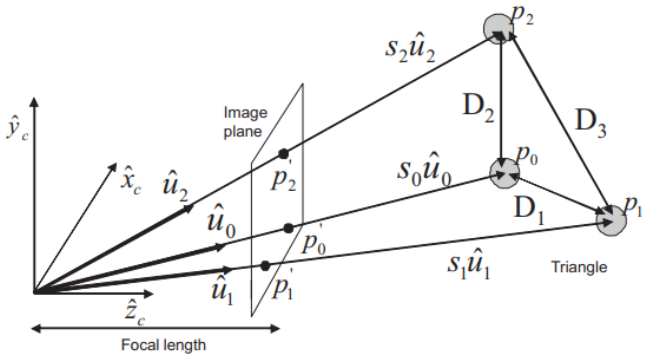

Figura 7. Problema de Ponto e Perspectiva - Fonte [9].

Também vale ressaltar que não é viável dizer que o sistema proposto pode substituir os sensores a bordo do quadrirrotor, mas sim complementa-los, visto que, mesmo podendo fornecer todas as informações necessárias para um controle do sistema, não são todos os momentos de voo que o marco artificial é captado pelo quadrirrotor. Nesses momentos os outros sensores a bordo devem fornecem as informações necessárias para o controle correto do quadrirrotor.

G. Sistema de Sensoriamento Híbrido

O sistema de sensoriamento proposto por este trabalho é um sistema híbrido, utilizando os dados dos sensores MEMS e os dados obtidos pelo algoritmo de visão computacional, executando o algoritmo de visão computacional em uma frequência menor em relação a frequência do sistema.

É possível utilizar este método, mantendo praticamente a mesma precisão do sistema de visão computacional, visto que o erro de propagação dos sensores *MEMS* embarcados só começa a ser problemático com um grande intervalo de tempo, normalmente entre 10 e 100 passos do sistema.

Pelo ponto de vista de poder computacional também é interessante iterar o algoritmo de visão computacional em uma frequência menor, visto que é um algoritmo custoso computacionalmente, necessitando em média 40 vezes mais tempo computacional por iteração em comparação com a determinação pelos sensores *MEMS*.

O algoritmo proposto atua diretamente nas equações 9 e 10, substituindo os valores propagados na integração do passo anterior, de posição $\vec{X}_{k-1,obs}$ e posição angular $\vec{q}_{k-1,obs}$, pelos valores obtidos na análise da imagem, efetivamente diminuindo o erro de propagação no passo de integração, como podemos observar no diagrama de blocos 8 e nas equações 17 e 18.

$$\vec{X}_{k,obs} = \vec{X}_{k-1,img} + \dot{\vec{X}}_{k,obs} \cdot \Delta t \tag{17}$$

$$\vec{q}_{k,obs} = \vec{q}_{k-1,img} + \dot{\vec{q}}_{k,obs} \cdot \Delta t \tag{18}$$

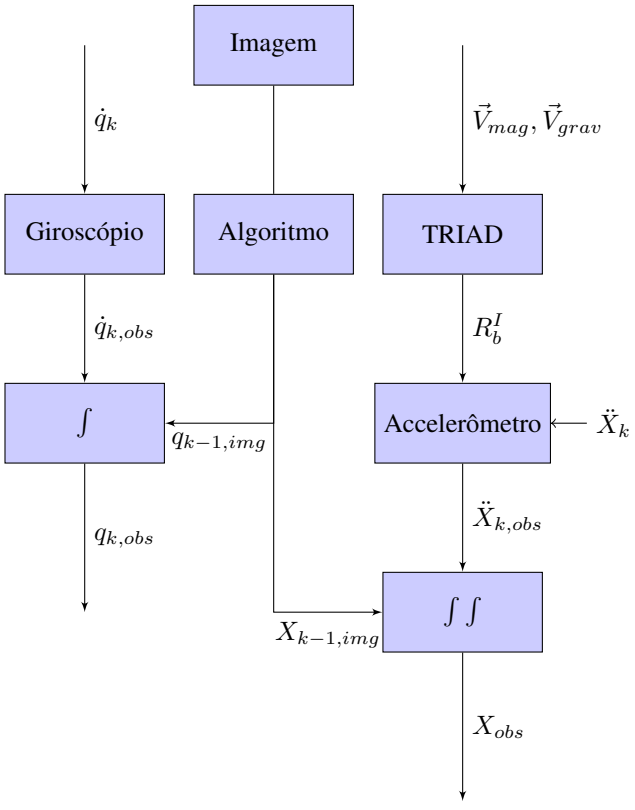

Figura 8. Diagrama de Blocos do Sensoriamento Proposto

Vale ressaltar que o diagrama de blocos é válido apenas para as iteração que há determinação de atitude por marco artificial, nas iterações do sistema que estão entre as determinações o algoritmo respeita o diagrama da figura 2 e as equações 9 e 10.

Tabela I
PARÂMETROS DO MODELO

Parâmetros Físicos Quadrirrotor Gyrofly-UFABC		
Frequência do Sistema	f_s	100Hz
Distância Motor - CG	D	$0.26m$
Espessura Braço	l	$0.05m$
Massa	m	$1.03kg$
Momento de Inércia em X	J_{xx}	$16.83 \cdot 10^{-3} kg/m^2$
Momento de Inércia em Y	J_{yy}	$16.83 \cdot 10^{-3} kg/m^2$
Momento de Inércia em Z	J_{zz}	$28.34 \cdot 10^{-3} kg/m^2$
Momento de Inércia da Hélice	I_r	$5 \cdot 10^{-5} kg/m^2$
Constante de Empuxo do Motor	K_f	$1.435 \cdot 10^{-5}$
Constante de Torque do Motor	K_m	$2.408 \cdot 10^{-7}$
Constante de Arrasto	C_d	1.1

Tabela II
PARÂMETROS DOS SENSORES

Sensor	**Parâmetros do Modelo**	
	Desvio Padrão	Desvio de Tendência
Acelerômetro	$0.1m/s^2$	$0.001m/s^2$
Giroscópio	$35mrad/s$	$0.3mrad/s$
Magnetômetro	$15mG$	$0.15mG$
GPS - Posição	$1.71m$	$0m$
GPS - Velocidade	$0.5m/s$	$0m/s$

H. Parâmetros do Modelo

Os parâmetros físicos do sistema e dos sensores podem ser observados nas tabelas I e II. A câmera acoplada ao corpo do quadrirrotor foi configurada com tamanho de sensor de 36 por 24 milímetros e comprimento focal de 45 milímetros. O mundo foi cuidadosamente criado para estar na escala correta em comparação com o quadrirrotor. Cada quadrado do padrão xadrez utilizado no algoritmo tem arestas de 0.1 metros, com a área interna do padrão correspondendo a uma folha de papel A2.

A renderização 3D da cena foi realizada em resolução de $1920x1080$, com um filtro de textura anisotrópico de $16x$ e o modo de qualidade de textura *slow*. Estas configurações foram escolhidas para manter uma boa fidelidade das texturas do mundo, principalmente o padrão xadrez. Resoluções menores ou filtros menos agressivos deixam as texturas com aspecto de névoa, dificultando muito o funcionamento correto do algoritmo.

III. RESULTADOS

Definimos erro como a diferença absoluta entre o estado real do sistema e o estado observado. É possível observar nas tabelas III e IV o comportamento deste erro pelo sistema de sensoriamento convencional e o sistema híbrido. Para o cálculo da média destes error foram utilizados 10.000 amostras, retiradas de aproximadamente seis episódios de simulação.

Tabela III
MÉDIA DE ERRO EM VOO PAIRADO DE 30 SEGUNDOS

Tipo de Sensoriamento	Média e Variância de Erro	
	Posição $(10^{-1}m)$	***Quaternion*** (10^{-3})
Híbrido	0.431 $\pm$ 0.008	4.791 $\pm$ 0.038
Convencional	4.698 $\pm$ 0.376	6.675 $\pm$ 0.072

Tabela IV
MÉDIA DE ERRO EM VOO DE ESTADO INICIAL ALEATÓRIO PARA NULO

Tipo de Sensoriamento	Média e variância de erro	
	Posição $(10^{-1}m)$	***Quaternion*** (10^{-3})
Híbrido	0.462 $\pm$ 0.029	3.300 $\pm$ 0.002
Convencional	2.908 $\pm$ 0.124	5.001 $\pm$ 0.003

Na tabela III, o sistema foi iniciado em posição zero, com único objetivo de se manter nesta posição, enquanto na tabela IV o sistema foi iniciado com estados aleatórios, devendo convergir a zero e normalmente sem contato visual inicial com o marco artificial.

Nas figuras 9-11 podemos observar, no decorrer do tempo, o erro de um dos casos com estado inicial aleatório. Os casos apresentados entre Híbrido e Convencional não iniciaram no mesmo estado, devido ao caráter aleatório da simulação, mas ilustram claramente a deriva no sistema de sensores convencional, em comparação com o sistema de sensoriamento proposto.

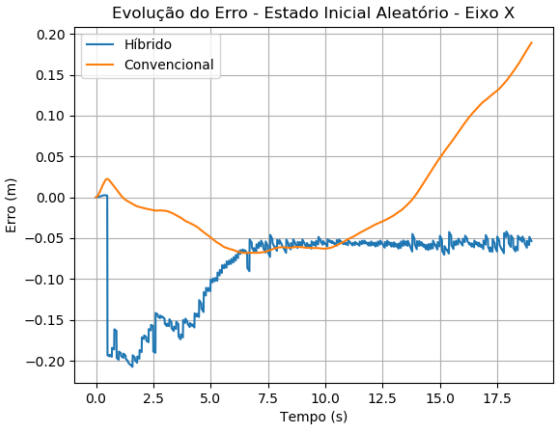

Figura 9. Desenvolvimento no tempo do erro no eixo X.

Os resultados obtidos com o sistema híbrido foram melhores, em comparação com os sensores *MEMS*. A média de erro foi 8.4 vezes menor para o sistema híbrido, com precisão absoluta na média de 4.5 centímetros, enquanto o sistema convencional teve precisão absoluta na média de 38 centímetros. Para a posição angular a diferença foi menor, mas ainda positiva para o sistema híbrido, com precisão média 45% melhor em relação ao sistema convencional.

Na tabela V podemos comparar o tempo computacional do sensoriamento convencional e o sensoriamento híbrido. O

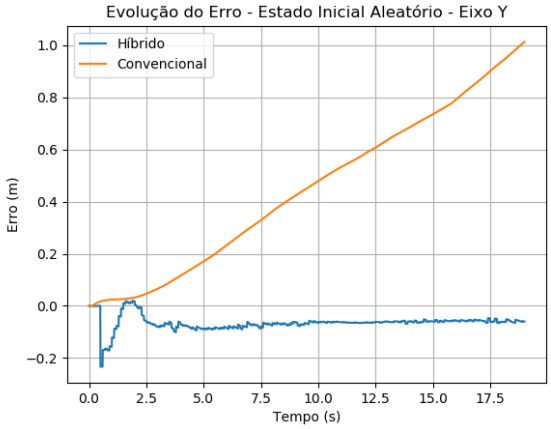

Figura 10. Desenvolvimento no tempo do erro no eixo Y.

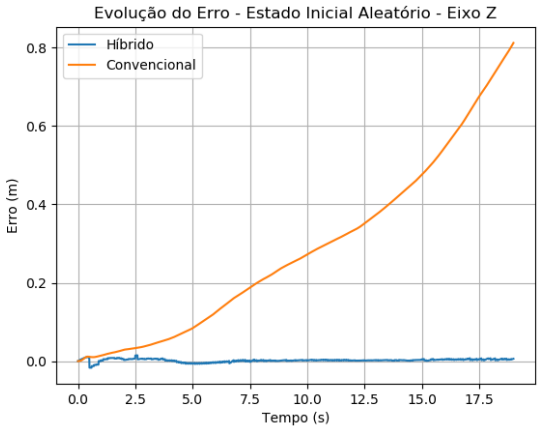

Figura 11. Desenvolvimento no tempo do erro no eixo Z.

tempo computacional no sensoriamento híbrido, na média de 10.000 amostras, ficou três vezes maior no voo pairado e sete vezes maior no voo com estado inicial aleatório.

Tabela V
MÉDIA DE TEMPO COMPUTACIONAL POR ITERAÇÃO

Tipo de Voo	Tempo Computacional por Iteração		
	Det. Imagem	*MEMS*	Híbrido
Voo Pairado	22.9 ms	1.16 ms	3.45 ms
Estado Aleatório	69.6 ms	1.16 ms	8.12 ms

Uma das causas do tempo maior para o voo com estado inicial aleatório está no algoritmo de detecção do marco. Quando o marco não está visível na cena o algoritmo escaneia todo o domínio da imagem antes de resolver que não há marco na cena. Se há marco na cena, ele escaneia até o início do marco, que normalmente está no meio da imagem, reduzindo o tempo computacional pela metade, normalmente.

Para o tempo computacional do sistema híbrido é considerado que o algoritmo de determinação de atitude e posição só

ocorre a cada dez passos de tempo do sistema, logo o tempo dessa etapa é diluído nos outros nove passos que só ocorre determinação pelos sensores *MEMS*.

O tempo efetivo de uma iteração do sensoriamento por determinação de imagem é cerca de quarenta vezes maior que o sensoriamento convencional, como podemos observar na primeira coluna da tabela V. O uso do sistema híbrido proposto reduz consideravelmente o tempo computacional, mantendo em média um tempo computacional de 5.8 milissegundos.

Um problema do sistema híbrido é a dinâmica do algoritmo. São nove iterações que em somadas demoram 10.44 milissegundos e uma iteração que em média demora 47,41 milissegundos. Esta disparidade de tempo entre iterações não é interessante, podendo efetivamente atrasar o controle e a obtenção de dados dos sensores, logo é interessante que o sensoriamento proposto ocorra em outro núcleo computacional, em paralelo, evitando possíveis atrasos na dinâmica do algoritmo principal de controle e obtenção de dados.

REFERÊNCIAS

[1] G. Farid. “Analysis, modeling and compensation of bias drift in MEMS inertial sensors.” 2009 4th International Conference on Recent Advances in Space Technologies. IEEE, 2009.

[2] L. Qiang, et al. “Kalman filter and its application.” 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS). IEEE, 2015.

[3] B. Gary and A. Kaehler. “OpenCV.” Dr. Dobb’s journal of software tools 3 (2000).

[4] L. Hugh and G. Pang. “Accelerometer for mobile robot positioning.” IEEE Transactions on Industry Applications 37.3 (2001): 812-819.

[5] O. Allan, “Modelagem dinâmica e controle de um veículo aéreo não tripulado do tipo quadrirrotor.” 2019. Dissertação (mestrado) - Universidade Federal do ABC, Programa de Pós-Graduação em Engenharia Mecânica, 2019. Disponível em: https://bit.ly/2T9sie0 Acesso em: 16 mai. 2020.

[6] S. J., et al. “Proximal policy optimization algorithms. arXiv 2017.” arXiv preprint arXiv:1707.06347.

[7] Documentação OpenCV. “Camera Calibration´´ Disponível em: https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html Acesso em: 18 mai. 2020.

[8] B. Harold D. “A passive system for determining the attitude of a satellite.” AIAA journal 2.7 (1964): 1350-1351.

[9] M. Marco, et al. “A Traffic Sign Detector For Mobile Robot Localization.” Disponível em https://bit.ly/3fTpqLT Acesso em: 16 mai. 2020.