Hardware-in-the-loop com o simulador de vôo Microsoft Flight Simulator

Luiz A. Cantoni Armando A. Neto

***Abstract*—A cada nova versão, simuladores de vôo *commercial-of-the-shelf* como o Microsoft Flight Simulator, X-Plane e FlightGear, conseguem reproduzir de forma mais fiel a experiência de voar. Uma tendência no desenvolvimento dos VANT's (veículos aéreos não tripulados) é a utilização do ambiente de simulação proporcionado por esse tipo de *software* para a realização de experimentos e validação de algoritmos de controle. Neste trabalho, foi explorado pela primeira vez, o ambiente do Microsoft Flight Simulator para simulações com *hardware-in-the-loop*.**

I. INTRODUÇÃO

VANT (Veículos aéreos não tripulados) também conhecidos mundialmente pela sigla UAV (*Unmanned aerial vehicles* ou *Uninhabited aerial vehicles*), podem ser definidos como aeronaves não tripuladas que voam de forma autônoma, através de um controle remoto ou de forma híbrida e que são utilizadas para executar diversas tarefas tanto militares quanto civis.

Nos últimos anos observou-se um crescimento considerável na utilização desses veículos [1]. Um dos fatores que estimulou esse crescimento foram os avanços tecnológicos ocorridos em diversas áreas como: miniaturização de componentes, sensores mais precisos, câmeras de vídeo cada vez menores e com maior poder de resolução, sistemas de GPS (*Global Position System*) de baixo custo e comunicação *wireless* [2].

Alem de preservar a vida humana (retirando o piloto da aeronave), o interesse por esse tipo de veículo está muitas vezes associado aos baixos custos de desenvolvimento e operação se comparados aos de aeronaves maiores e tripuladas. Entretanto, pelo fato de não possuir um piloto a bordo, surgem diversos desafios para fazer com que o vôo (ou pelo menos uma parte dele) seja realizado de forma autônoma. São necessários sistemas eletrônicos embarcados (como piloto automático e sensores) e algoritmos de controle e navegação que consigam manter a estabilidade e a trajetória do vôo.

Nesse sentido, um requisito fundamental para a maioria dos projetos é realizar simulações para validar estes algoritmos antes dos vôos reais. Para isso, muitos pesquisadores combinam a técnica de *hardware-in-the-loop* com simuladores COTS (*commercial-of-the-shelf*). Essa combinação permite redução significativa no tempo de desenvolvimento, pois os simuladores de vôo prevêem um ambiente que permite utilizá-los na validação de sistemas embarcados reais.

Os simuladores de vôo FlightGear e X-Plane já foram utilizados em diversos trabalhos que utilizaram *hardware-in-the-loop*, porém o Microsoft Flight Simulator ainda não havia sido explorado com esse objetivo.

II. SIMULADORES DE VÔO

Os simuladores de vôo são sistemas que tentam copiar da forma mais realista possível a experiência de voar. No contexto deste trabalho, abordam-se os simuladores COTS (*Commercial-of-the-shelf*) com foco na aviação civil, que possuem, como principal objetivo o entretenimento, mas que devido ao nível de realismo que chegaram, também são utilizados para treinamento de pilotos reais. Dentre os simuladores mais famosos pode-se citar: FlightGear [3], Microsoft Flight Simulator [4] e X-Plane [5].

Uma tendência nas pesquisas que envolvem VANT's é a utilização desse tipo de simulador para testar e validar algoritmos de controle através da técnica *hardware-in-the-loop* (HIL) [6] [7]. As principais vantagens são:

- Modelo da dinâmica de vôo já embutido no simulador;
- Ambiente 3D;
- Possibilidade de criar novas aeronaves, cenários e programas que façam o intercâmbio dos dados entre o simulador e o mundo externo;
- Possibilidade de simular variáveis como o vento e térmicas;

A. Microsoft Flight Simulator

Com mais de 25 anos de história, o Microsoft Flight Simulator é o simuladores de vôo mais famosos do mercado. O seu tipo de licença é comercial e custa por volta de R$140,00. Está na versão 10 (chamado FSX) e possui uma grande comunidade de pilotos reais e virtuais que o utilizam.

Através da sua API (*Application Programming Interface*) é possível desenvolver programas que executam dentro ou fora do simulador. Além disso, é possível inserir/alterar aviões, painéis de instrumentos, cenários, informações

meteorológicas, entre outras características. As possibilidades de customização e a qualidade gráfica podem ser apontadas como os principais motivos do sucesso do simulador.

A API nativa do FSX se chama SimConnect, através dela que pode ser desenvolvido o programa que faz o intercâmbio de dados entre o simulador e o mundo externo, permitindo assim, o desenvolvimento de um sistema HIL.

A dinâmica de vôo das aeronaves é baseada em uma abordagem paramétrica. Um conjunto de tabelas determina como o avião se comporta sob certas condições [11]. Outras API's, desenvolvida por terceiros, são a JSimConnect (livre, em Java) e a FSUIPC (comercial). Esta última está disponível desde a versão 6 do Flight Simulator.

B. *FlightGear*

Multiplataforma, livre e de código aberto, o principal objetivo desse simulador é proporcionar um ambiente para o desenvolvimento de pesquisas. Por isso, um dos pontos fortes é a possibilidade de escolher entre vários modelos de dinâmica de vôo. Os principais são: JSBSim, YASim e UIUC (baseado no LaRCsim) [11]. Essas características fazem com que o FlightGear seja bastante utilizado no meio científico.

C. *X-Plane*

Simulador de vôo multiplataforma desenvolvido pela Laminar Research e aprovado pela FAA (*Federal Aviation Administration*) para treinamento de pilotos reais. O seu tipo de licença é comercial e custa por volta de R$100,00. Está na versão 9.20 e também pode ser customizado como o seu concorrente Microsoft Flight Simulator. O X-Plane executa no Windows, Linux e MacOS. Uma das principais diferenças entre os dois simuladores é na forma como eles lidam com a dinâmica do vôo. O X-Plane possui uma abordagem geométrica. Através da estrutura da aeronave e outros parâmetros de engenharia, um processo chamado "*blade element theory*" calcula as dinâmicas do vôo [11].

III. HARDWARE-IN-THE-LOOP

Ajustar um piloto automático a partir de vôos reais pode consumir muitas horas, levar a resultados insatisfatórios e a situações perigosas. Por esse motivo, muitos pesquisadores lançam mão de técnicas como *hardware in the loop* (HIL) [6] [7] [8] [9] [10]. HIL é um método utilizado para testar e ajustar sistemas de controle sem a necessidade de levá-los a situações reais. Obviamente este método não pode substituir os experimentos reais, mas é muito útil nas fases inicias. Uma vez realizado os testes com a arquitetura HIL, o mesmo controlador pode ser conectado diretamente no veículo aéreo para uma missão real.

Em [7] foi utilizada a técnica de HIL para validar algoritmos de trajetória e altitude para um VANT que tem como missão realizar amostragens químicas do gás da coluna de erupção de vulcões. Foi utilizado o simulador de vôo X-Plane para enviar ao piloto automático real informações sobre o veículo (velocidade, altitude, atitude e posição). Já o piloto automático, envia os sinais de controle para o simulador com o objetivo de alterar as superfícies móveis do avião (*aileron*, profundor e leme) e assim, de forma autônoma, controlar a trajetória e altitude do vôo. Em alguns casos, no lugar do piloto automático real, utiliza-se um *software* (MATLAB, por exemplo), isso pode ser observado na Figura 1. Este tipo de abordagem é útil nas fases iniciais do projeto onde se deseja obter valores mais aproximados para as constantes dos controladores.

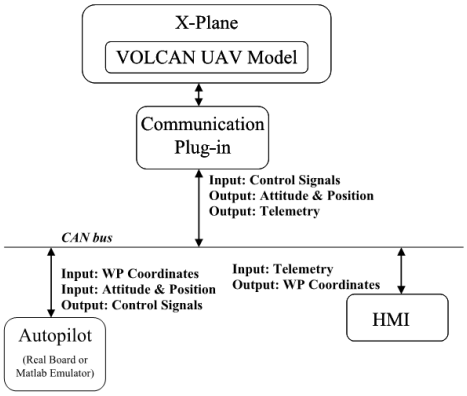


Figura 1 – Arquitetura HIL utilizada em [7]

IV. METODOLOGIA

O trabalho foi dividido em três partes:

- Estudo da API SimConnect;
- Implementação dos algoritmos de controle para altitude e orientação;
- Experimentos utilizando o FlightGear e o Microsoft Flight Simulator;

A. *API SimConnect*

Nas versões anteriores do Microsoft Flight Simulator (FS2004, FS2002, FS2000, FS98 e FS95) é utilizado um mecanismo baseado em IPC (*Inter-process communication*) para recuperar/alterar dados no simulador. Basicamente, os módulos são DLL's (*Dynamic-link library*) que obedecem a uma interface para executar dentro do espaço de memória do simulador. Para facilitar esse processo, foram desenvolvidas (por terceiro e comercial) as bibliotecas FSUIPC e WideFS que tem como objetivo principal facilitar o desenvolvimento de customizações para o simulador e permitir a execução dos módulos criados em outra máquina, que não seja a que está

executando o simulador. A WideFS se comunica através de TCP/IP.

Já na versão X, a Microsoft desenvolveu uma nova API para acesso ao simulador, chamada SimConnect. Uma das mudanças foi na forma de acesso ao simulador, a comunicação é realizada através de TCP/IP.

Na figura 2 é exibida a arquitetura da SimConnect.

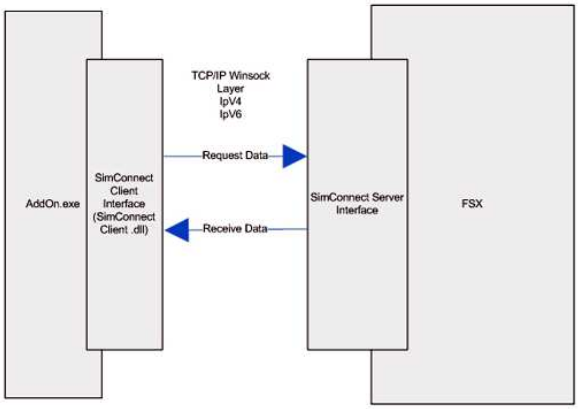


Figura 2 – Arquitetura da API SimConnect

Diversos dados podem ser obtidos do simulador e alguns deles podem ser alterados. No contexto desse trabalho, os principais parâmetros foram:

- Razão de subida, Velocidade, Altitude, Orientação;
- Posição do Aileron, Profundor (o controle do leme foi feito de forma automática pelo próprio simulador);
- Ângulo de Rolagem e Arfagem;

A API do FlightGear é mais direta e funciona através de pacotes UDP/IP. Basicamente, são trocados pacotes baseados em uma estrutura que o simulador disponibiliza através de duas classes em C++.

B. *Taxa de aquisição dos dados*

Geralmente, quando se projeta um controlador, é necessária uma taxa mínima de aquisição de dados para que ele consiga fazer o controle de forma adequada. Nos experimentos realizados com o FSX, uma taxa média de 46 Hz foi obtida (limite em 50 Hz). Entretanto, observou-se que a taxa possui uma variação significativa, com desvio padrão em 4,6.

Na API é possível configurar para receber os dados a cada *frame* de simulação, sendo ele renderizado ou não. Entretanto, percebeu-se que a taxa de atualização do modelo da dinâmica de vôo está diretamente ligada com a taxa de atualização da interface gráfica. Essa característica foi observada, pois ao diminuir o *framerate* da renderização da interface gráfica para 10 Hz, o controlador não conseguiu manter o avião na trajetória.

A única forma de manter uma taxa de atualização constante é utilizar uma configuração de computador com maior capacidade de processamento (vídeo principalmente). Nos experimentos foi utilizada a placa de vídeo (*on-board*) Intel 965 *Express Chipset Family*. Obviamente, não é a placa ideal para rodar um simulador como o FSX. Mas é preciso notar que, em simulações com *hardware-in-the-loop*, a qualidade gráfica não é tão significativa quanto a fidelidade da simulação do vôo.

Esta característica do FSX é prejudicial, pois além de não refletir a realidade da taxa enviada por *hardware* embarcado real, exige máquinas relativamente poderosas para executar a simulação.

C. *Algoritmos de Controle*

Foram utilizados dois algoritmos de controle (altitude e orientação) para avaliar o FSX. As implementações, retiradas de [6], seguem o projeto do piloto automático do FlightGear que, por sua vez, é baseado em controladores PID em cascata. O mesmo tipo de controlador foi implementado em [7] e pode ser observado nas figuras abaixo.

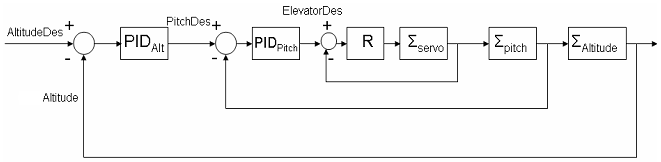


Figura 3 –Controlador PID para altitude (retirado de [7])

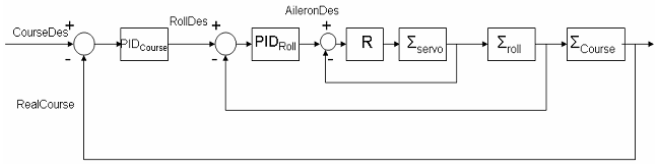


Figura 4 –Controlador PID para orientação (retirado de [7])

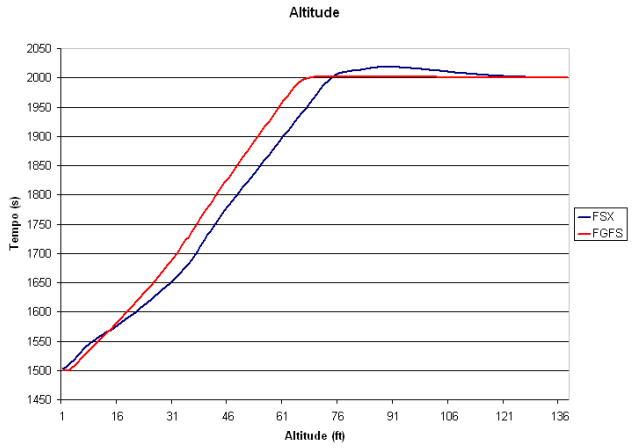
V. EXPERIMENTOS

A seguinte metodologia foi definida para os experimentos:

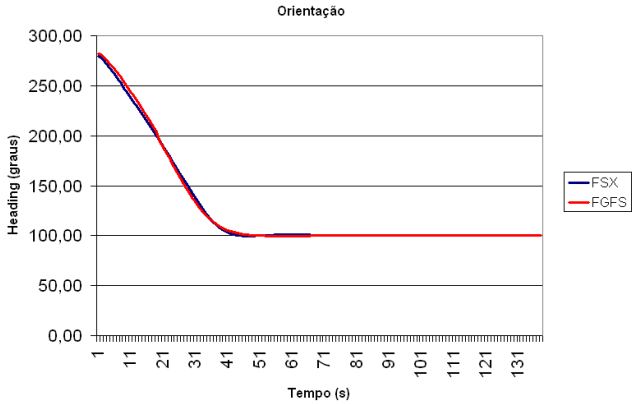
- **Avião**: Piper J3-Cub;
- **Velocidade inicial**: 70 nós;
- **Altitude inicial**: 1500 pés;
- **Altitude desejada**: 2000 pés;
- **Orientação inicial**: 280 graus;
- **Orientação desejada:** 100 graus;
- **Tempo de execução:** 140 segundos;

No controle de altitude, conforme demonstrado na figura 3,

houve uma diferença entre as execuções do FSX e do FlightGear. Porém, apesar do *overshoot*, o avião foi estabilizado nos 2000 pés. Essa diferença entre os simuladores era esperada e pode ser atribuída às características da implementação do modelo da dinâmica de vôo de cada simulador, ou seja, os ganhos do controlador foram otimizados para a dinâmica de vôo do FlightGear. Para corrigir essa divergência é necessário alterar os ganhos, adaptando-os ao FSX.



No controle de trajetória (figura 4) não houve *overshoot* por parte do FSX, entretanto a diferença também aconteceu e pode ser atribuída ao mesmo fator.



VI. CONCLUSÃO

Este trabalho apresentou pela primeira vez uma simulação *hardware-in-the-loop* utilizando o simulador de vôo Microsoft Flight Simulator versão X (FSX). Os resultados foram satisfatórios e indicam que o simulador pode ser utilizado na validação de algoritmos de controle para VANT's. A API SimConnect, apesar de complexa, permitiu recuperar/alterar todas as variáveis necessárias para a realização do trabalho.

Entretanto, o FlightGear é um simulador mais preparado para esse tipo de simulação e deve ser escolhido, devido as seguintes características:

- API simples e direta;
- Não exige tanto processamento de vídeo;
- Possibilidade de escolher diferentes modelos da dinâmica de vôo;
- Gratuito e de código aberto;

REFERENCES

[1] K.P. Valavanis, "Introduction" in *Unmanned Aerial Vehicles: State of the Art and the Road to Autonomy*, Vol. 33, Intelligent Systems, Control, and Automation: Science and Engineering, K.P. Valavanis, Ed. Springer: 2007, pp. 23-34.

[2] A. Ryan et al., "An overview of emerging results in cooperative UAV control," *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, 2004, pp. 602-607 Vol.1.

[3] "FlightGear Flight Simulator"; http://www.flightgear.org/. [Acessado Outubro 10, 2008].

[4] "Flight Simulator X"; http://www.microsoft.com/games/flightsimulatorX/. [Acessado Outubro 10, 2008]

[5] "X-Plane"; http://www.x-plane.com. [Acessado Outubro 10, 2008]

[6] A.A. Neto e M.F.M. Campos, "Implementação de um sistema Hardware-in-the-loop para veículos aéreos autônomos não-tripulados," *Congresso Brasileiro de Automática*, Juiz de Fora: 2008.

[7] G. Astuti et al., "Hardware in the Loop Tuning for a Volcanic Gas Sampling UAV" in *Unmanned Aerial Vehicles: State of the Art and the Road to Autonomy*, Vol. 33, Intelligent Systems, Control, and Automation: Science and Engineering, K.P. Valavanis, Ed. Springer, 2007, pp. 485-505.

[8] W. Adiprawita, A. Suwandi Ahmad, e J. Sembiring, "Hardware in the Loop Simulation for Simple Low Cost Autonomous UAV (Unmanned Aerial Vehicle) Autopilot System Research and Development," *International Conference on Electrical Engineering and Informatics (ICEEI2007)*, 2007.

[9] J. How, E. King, e Y. Kuwata, "Flight Demonstrations of Cooperative Control for UAV Teams," *Proceedings of the 2004 AIAA Unmanned Unlimited Technical Conference, Workshop and Exhibit*, 2004.

[10] D. Jung e P. Tsiotras, "Modeling and Hardware-in-the-Loop Simulation for a Small Unmanned Aerial Vehicle," *AIAA Infotech at Aerospace*, Rohnert Park, CA: 2007.

[11] R. Gimenes et al., "Using Flight Simulation Environments with Agent-Controlled UAVs," *Autonomous Robot Systems and Competitions: Proceedings of the 8th Conference*, Aveiro, Portugal: 2008.