ANÁLISE DE TÉCNICAS DE DETECÇÃO ATRAVÉS DE CÂMERAS EMBARCADAS EM VANTS NO PROCESSO DE DETECTAR E DESVIAR DE OUTRA AERONAVE

Vinícius Maurício de Almeida, Paulo Sergio Cugnasca
Universidade de São Paulo (USP), Escola Politécnica (Poli)

*** Corresponding author e-mail address**: viniciusmdea@gmail.com

PAPER ID: SIT226

RESUMO

O número de VANTs (Veículos Aéreos Não Tripulados), bem como o seu uso, vêm aumentando consideravelmente nos últimos anos. Porém, a sua inserção em um espaço aéreo não segregado pode se tornar uma ameaça de colisão com as demais aeronaves que compartilham o mesmo espaço aéreo. Para se evitar colisões, o processo utilizado atualmente tem o nome de s*ense and avoid*, pois consiste de duas etapas: a detecção (*sense*) de outra aeronave em rota de colisão e o desvio (*avoid*) para evitar a rota de colisão encontrada. Cada uma dessas etapas pode ser alcança de diversas maneiras, seja utilizando-se uma técnica específica de desvio, seja alterando-se o sensor ou o algoritmo de detecção associado a esse sensor. Este trabalho tem como foco a primeira etapa, analisando-se o processo de detecção de outra aeronave e utilizando-se de três diferentes combinações de sensores: a visão binocular, que consiste na utilização de duas câmeras visuais trabalhando em conjunto; a câmera de profundidade; e a câmera de profundidade em conjunto com a câmera visual. Também são analisados testes que utilizaram técnicas de fusão dos sensores descritos, os quais utilizam dois ou três sensores trabalhando em conjunto para a detecção de outra aeronave com possibilidade de colisão como o VANT. Para grupo de sensores foram analisados e testado diversos tipos de algoritmos de detecção. Os testes foram realizados por meio do AirSIM, um simulador de veículos não tripulados. As análises dos algoritmos foram feitas verificando-se a taxa de detecção e o erro da posição da aeronave calculado pelo algoritmo considerando-se a posição real da aeronave detectada. No final, chega-se a um resultado no qual detecta-se a aeronave em rota de colisão com o VANT em 100% dos casos testados, com erro médio 20 metros entre a posição detectada e a posição real da aeronave

Keywords: VANTs, UAVs, s*ense and avoid*, visão computacional, detecção.

1. INTRODUÇÃO

Devido à sua grande versatilidade, os VANTs (Veículos Aéreos Não Tripulados) vêm ganhando diversas aplicações, o que impulsiona o aumento do número destes veículos. Segundo estimativas da FAA (FAA - Federal Aviation Administration 2021) haverá um crescimento entre 9% a 44% no número de VANTs no espaço aéreo americano no período de 2020 a 2025.

Porém, a inserção desses veículos no espaço aéreo que não seja segregado pode se tornar uma ameaça às demais aeronaves que compartilham o mesmo espaço, pois permite o surgimento de cenários de risco de colisões entre elas. Para garantir que a inserção ocorra de modo seguro, é necessário garantir que esses veículos tenham capacidade de evitar colisões de forma similar ou superior a veículos com pilotos embarcados.

Pesquisas têm sido conduzidas visando à criação desta habilidade nos VANTs por meio de uma técnica chamada de *sense and avoid*, que consiste de duas etapas: a detecção da aeronave em risco de colisão (chamada de *sense*) e a criação e execução de uma rota que permita desviá-la da colisão (chamada de *avoid*). Ambas as etapas podem ser executadas de diversas maneiras, seja alterando o sensor, seja alternando o algoritmo de desvio utilizado.

Este trabalho tem como foco a etapa de detecção. Nele foram analisados três diferentes grupos de sensores para detectar a posição da aeronave em risco de colisão: a detecção através da visão binocular, que consiste em duas câmeras visuais trabalhando em conjunto; a detecção por meio de uma câmera de profundidade; e a detecção mista, que utiliza a câmera de profundidade trabalhando em conjunto com uma câmera visual. Em cada grupo de sensores foram analisados diversos algoritmos de visão computacional, comparando a precisão da posição calculada para a aeronave em rota de colisão com sua posição real, além da porcentagem de detecções corretas obtidas. Também foram feitos testes utilizando dois ou mais grupos de sensores trabalhando em conjunto, criando uma detecção for fusão de sensores.

Os testes foram realizados utilizando-se o AirSim, um simulador de veículos não tripulados. Ao final das simulações foram obtidas técnicas que detectam a aeronave em 100% das imagens nas quais ela está presente com erro de meio de 20 metros em relação a sua posição real.

2. DETECÇÃO DE OBJETOS EM RISCO DE COLISÃO POR VANTS

A detecção de conflito por *drones* é um problema que vem sendo amplamente pesquisado nos últimos anos. Existem várias maneiras realizá-la, podendo-se alterar o tipo de sensor ou de algoritmo utilizado. Em (Yu and Zhang 2015), (Lacher, Maroney, and Zeitlin 2007), (Cappello et al. 2015) é descrita uma série de sensores que podem ser utilizados para detecções por aeronaves. Esses sensores são divididos em dois diferentes tipos de detecção: a colaborativa, na qual a identificação de colisão é feita por meio de troca de informações entre as aeronaves, e a não colaborativa, na qual uma aeronave identifica objetos usando somente dados oriundos de seus próprios sensores, sem a necessidade de comunicação externa.

2.1. Detecção colaborativa

As detecções colaborativas correspondem aos modelos mais confiáveis, pois neste tipo de detecção não é necessário identificar a posição da aeronave, apenas ler e interpretar os dados enviados por ela, os quais contêm informações sobre sua posição, direção e velocidade. Dentre esses equipamentos utilizados neste modelo de detecção estão incluídos o *Automatic Dependent Surveillance-Broadcast* (ADS-B), o *Traffic Alert and Collision Avoidance System* (TCAS) (FAA 2011), e o mais recente, *Airborne Collison Avoidance System* (ACAS) (Eurocontrol 2017), que contém uma versão para veículos não tripulados: o ACAS-Xu.

Em (Kuchar 2005) é feita uma análise de aspectos de segurança sobre a aplicação do TCAS como sistema de prevenção de colisão no VANT Global Hawk. Este sistema é capaz de identificar aeronaves equipadas com transponder por intermédio de troca de informações entre elas. Kuchar apresenta uma análise dividida em duas fases: a primeira fase, chamada de *loop* externo, utiliza-se de uma

árvore de falhas para verificar situações que levam um VANT a um cenário de colisão; a segunda fase, chamada de *loop* interno, implementa simulações com a técnica de Monte Carlo para encontrar possíveis falhas que levariam o VANT a uma colisão com outras aeronaves. Nesses testes foram encontradas taxas de falhas com valores dentro dos permitidos pela ICAO (*International Civil Aviation Organization*) e pela Eurocontrol (*European Organization for the Safety of Air Navigation*).

2.2. Detecção não colaborativa

Técnicas de detecção não colaborativa variam conforme o sensor utilizado, dentre elas a visão computacional está entre a mais estudada, por necessitar apenas de uma câmera, que é um sensor leve e barato. Dentre os trabalhos com *sense and avoid* utilizando câmeras, um dos primeiros resultados foi descrito por Ross (Ross et al. 2013), que utilizando o algoritmo de aprendizado supervisionado DAgger, o qual aprende a reconhecer a colisão e evitá-la a partir de dados de voos feitos por um piloto, possibilitou um VANT sobrevoar uma floresta, desviando-o de árvores presentes em seu caminho. Porém, seu algoritmo se limita apenas a desvio de obstáculos fixos.

Na mesma época, Pestana (Pestana et al. 2013) utilizou o algoritmo OpenTLD para reconhecer objetos em movimentos e rastreá-los. Sua técnica é utilizada com o foco em perseguição do objeto, porém sua detecção e rastreamento demonstram boa acurácia, perdendo o objeto em poucos momentos. Mas esta técnica se limita a objetos com baixas velocidades, diferentemente do ocorrido no espaço aéreo, e o seu trabalho, que tem o foco apenas no rastreamento, não inclui a detecção, sendo necessário ao usuário informar a localização inicial do objeto a ser rastreado.

Em uma pesquisa mais recente, Wu (Wu, Sui, and Wang 2017) utiliza um algoritmo para que um VANT reconheça aeronaves em movimento, tendo conseguido bons resultados, considerado o estado da arte para detecção de aeronaves. Seu algoritmo tem a detecção baseada em filtros de Kalman e duas fases distintas de escaneamento da imagem.

2.3. Fusão de sensores

Visando criação de um modelo que apresente a confiabilidade da detecção colaborativa, e a robustez da detecção não colaborativa, permitindo detectar aeronaves sem o sensor de comunicação, alguns autores propõem modelos híbridos de detecção, utilizando fusão de sensores.

Carrio propõem em (Carrio et al. 2017) uma técnica utilizando câmeras infravermelhas em conjunto com sensores ADS-B. Com essas câmeras, captam-se o calor emitido pelos objetos, e com isso é possível cobrir as falhas ocorridas nas câmeras RGB por falta de iluminação ou visibilidade. Seus testes demonstraram bons resultados em dias nublados, porém tiveram um grande número de falhas em períodos noturnos, pois as aeronaves não retinham muito o calor, atrapalhando a detecção pelas câmeras termais.

Fasano (Fasano et al. 2008) descrevem e testam um sistema de fusão de sensores, utilizando nele um radar *Ka-band* em conjunto com câmeras visuais RGB e câmeras infravermelhas, na tentativa de detectar um conflito e desviar um VANT de uma possível colisão com uma outra aeronave.

Rasamasy e Sabatini (Ramasamy, Sabatini, and Gardi 2016) descrevem um conjunto de sensores de detecção promissores para uso em VANTs, avaliando as vantagens e desvantagens de sua utilização. Entre os sensores estudados, estão o LIDAR, o MMW Radar, os sensores acústicos, as câmeras termais e óticas, além dos sensores colaborativos. É demonstrada uma modelagem de função de sensores para detecção, no qual sensores colaborativos recebem prioridade na detecção, e as detecção secundárias são feitas por meio de sensores não colaborativos, dentre eles sensores acústicos, radares, e câmeras visuais e termais, que por sua vez são complementares uma a outra.

3. DETECÃO DE OBJETOS EM VISÃO COMPUTACIONAL

Com o avanço de técnicas de visão computacional aliadas a algoritmos de *machine learning*, e devido ao baixo custo, peso e tamanho das câmeras, o número de trabalho

utilizando este sensor para detecção de objetos por VANTs vem aumentado significativamente.

A detecção de objetos através de visão passa por duas etapas: a primeira, de localização do objeto alvo na imagem (seja ela 2D ou 3D), e a segunda, referente ao cálculo da posição deste objeto no ambiente, de modo a comparar sua posição com à do VANT. Para as duas etapas podem ser utilizados diversos algoritmos distintos.

3.1. Detecção objetos em imagem

Para a detecção de um objeto na imagem, para o algoritmo é necessário extrair o máximo de informações presentes nela, com o objetivo de se reconhecer padrões que diferenciam o objeto alvo dos demais elementos presentes na imagem. A extração de características da imagem pode ocorrer de diversas maneiras, através de informações de textura, como a utilizado no algoritmo LBP (Guo, Zhang, and Zhang 2010) por meio de cálculos do gradiente da imagem, como no algoritmo HOG ou SIFT, ou no relacionamento entre os *pixels*, como o presente nos algoritmos de redes neurais convolucionais (CNN) (Yang et al. 2011).

Após a extração das características da imagem, os algoritmos a classificam como contendo ou não o objeto alvo. Uma abordagem muito utilizada com este objetivo está presente nos algoritmos de aprendizado de máquina, como: o SVM (Russell and Norvig 2010), que calcula um hiperplano capaz de classificar os dados em classes; o *Naive Bayes*, que se utiliza de cálculos probabilístico, baseados na aplicação do algoritmo no teorema de *Bayes*; as Árvores de Decisão, que aplicam métricas matemáticas, como Entropia, para calcular as características que melhor separam as classes dos dados (Russell and Norvig 2010); o *LigthGBM* (Ke et al. 2017), que utilizam várias árvores de decisão encadeadas, para se tornar mais robusta a classificação; ou as técnicas de *deep leaning*, exemplificada pelos algoritmos Xception (Chollet 2017) e YOLO (Redmon and Farhadi 2018), que utilizam CNN para a detecção.

Estes algoritmos, após treinados, são capazes de identificar se em uma imagem está presente ou não um determinado objeto. Para se reconhecer o local exato da imagem onde ele está presente, é possível dividir a imagem em várias partes menores a serem analisadas, utilizando técnicas de janelas deslizantes. Ambos os algoritmos podem ser utilizados tanto em imagem de câmeras visuais como em imagem de câmeras de profundidade.

3.2. Cálculo da posição do objeto através da imagem

Para ser possível calcular essa distância é necessário que as imagens utilizadas sejam capazes de reconstruir o ambiente capturado em 3 dimensões. Estes dados podem ser alcançados de duas maneiras: por meio de um sensor que capte dados de profundidade, ou utilizando um conjunto de dois ou mais sensores que capturem imagem em apenas duas dimensões, que é uma técnica conhecida como visão binocular, também chamada de visão estéreo.

3.2.1. Câmeras de profundidade

Câmeras de profundidade são sensores capazes de capturar imagens em 3D. Eles podem utilizar diferentes técnicas como *lasers* ou feixes de luz. Acrescentar este tipo de sensor torna este tipo de abordagem mais cara do que as que se utilizam de câmeras tradicionais.

Seus dados frequentemente são retornados como uma matriz, similar à obtida pelas câmeras visuais, o que permite que os mesmos algoritmos possam ser utilizados neste sensor, desde de que ambos tenham etapas de treinamentos individuais, para melhor performance do algoritmo.

3.2.2. Visão binocular

A visão binocular, ocorre quando uma imagem for capturada por dois ou mais sensores, permitindo, assim, o cálculo da profundidade da imagem. Com esta técnica é possível calcular um mapa de profundidade, similar à imagem retornada pela câmera de profundidade, a partir de duas câmeras visuais simples.

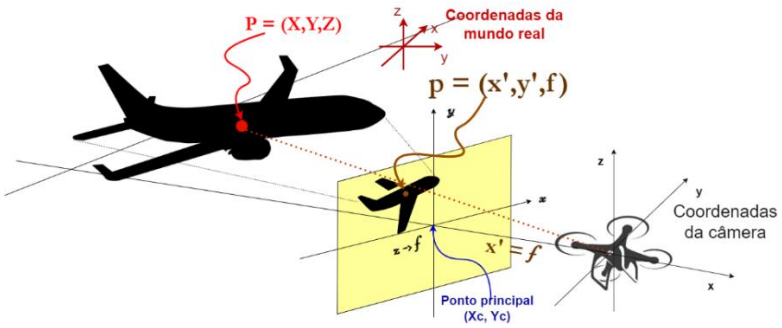

Figure 1: Formulação da imagem pelo modelo de câmera *pinhole*.

Para este cálculo, esta técnica utiliza princípios geométricos aplicados ao modelo de formação da imagem. Um dos modelos matemáticos que melhor descreve essa projeção é o modelo da câmera *pinhole*, no qual a câmera é comparada com um objeto, que permite que a luz passe apenas por um pequeno orifício, formando uma projeção do ambiente em uma imagem virtual, situada a uma distância igual a sua distância focal. (Forsyth and Ponce 2011). Neste modelo pode-se aplicar fórmulas geométricas e demonstrar, matematicamente, a formação da imagem.

Na Figura 1 apresenta-se a geometria de uma projeção de câmera, na qual o drone representa a câmera que está no centro (0, 0, 0) do seu próprio eixo de coordenadas (*x*, *y*, *z*), o objeto analisado (uma aeronave) está posicionado no ponto *P* = (*X*, *Y*, *Z*) no eixo de coordenadas do mundo real (representado em vermelho), e a imagem virtual projetada está representada na cor amarela, tendo o ponto principal (*Xc*, *Yc*) em seu o centro, e a distância *x* em relação ao *drone*, igual a distância focal da câmera *f*.

Porém, este modelo permite apenas o calcular a posição do objeto analisado em duas dimensões, pois ao se calcular a sua profundidade, chega-se a uma equação de reta. Para encontrar o valor exato da profundidade é necessário acrescentar uma segunda câmera, traçando duas equações de reta, e calculando-se o encontro delas. Este cálculo é chamado de geometria epipolar, e seu princípio está ilustrado na Figure 2.

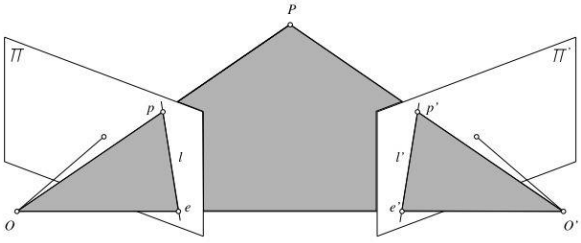

Figure 2: Geometria epipolar, em que P é a posição real do objeto, p' e p as projeções da imagem referentes às duas câmeras, e O e O' as câmeras da esquerda e da direita.

4. SIMULAÇÃO

Objetivando avaliar a eficácia, eficiência dos modelos de detecção de objetos, tornou-se necessária a realização de testes, utilizando os diferentes algoritmos descritos. Porém, para que esses testes ocorram de modo seguro, sem haver colisão entre aeronaves, eles devem ser realizados em simuladores, antes de serem implementados em VANTs reais.

4.1. Avaliação de simuladores

Visando uma simulação "a mais fidedigna possível" da realidade é necessária a escolha do simulador. Foram analisadas sete métricas essenciais que apresentam as melhores características para testes de *sense and avoid* com múltiplos sensores. São elas:

Física realista: Para que se consiga uma simulação realista de cálculos físicos do modelo da dinâmica e cinemática.

Modelagem de VANTs: Para a realização de testes que envolvam a possível colisão de aeronaves, além da dinâmica envolvida nas aeronaves e objetos é necessária uma representação física do corpo do veículo, que pode variar desde um pequeno VANT até a grandes aeronaves autônomas.

Simulação de sensores: Para testar os algoritmos é necessário que seja possível simular diferentes tipos de sensores para detecção, entre eles as câmeras visuais e de profundidade e o um sensor colaborativo, além da simulação de sensores inerciais.

Simulação do ambiente: Colisões de um VANT podem ocorrer não apenas com outras aeronaves, mas também com construções ou montanhas. Além disso, o ambiente pode ser um fator de grande impacto na qualidade de detecção.

Visualização gráfica realistas: As Câmeras têm seus algoritmos são ligados a visualização do ambiente em sua volta, sendo necessárias visualizações gráficas que simulem o cenário e o ambiente da forma mais real possível.

Código aberto: Para que o simulador possa ser utilizando por um maior número de pessoas é importante que exista a possibilidade de edição e utilização do seu código. Estas são as características desejáveis de *softwares* de código aberto (*OpenSource*).

Simulação de comunicação: Para que os testes se tornem mais realistas é necessário também simular a comunicação envolvida. Neste caso, deve-se enviar e ler dados do simulador de forma similar à encontrada em VANTs reais. Nesses veículos, essa comunicação normalmente se dá por meio de um protocolo TCP/IP, denominado de MavLink.

4.2. Estado da arte em simulação empregadas em VANTs

Dentre os simuladores pesquisados, o Gazebo (Koenig and Howard 2004) está entre o mais popular no meio robótico. Este simulador, um dos pioneiros na criação de ambientes realistas em 3D para interações com robôs, foi desenvolvido com uma representação fiel da dinâmica e da cinemática relacionadas com um robô, contando com um projeto modular que permite a criação de diversos sensores e mecanismos, como braços robóticos. Por ser um simulador de código aberto, tem grande utilização, sendo alvo de criação de vários *plug-ins* e funcionalidades incrementais. Trabalhos recentes modelaram a física e o corpo de VANT no Gazebo (Sendobry et al. 2012), permitindo simulações deste tipo de veículo. Porém, por ser um simulador antigo, o Gazebo conta com uma modelagem gráfica de certa forma ultrapassada, o que pode prejudicar testes que envolvam visão computacional.

Pesquisas mais recentes têm apresentado simuladores que se utilizam de tecnologias que apresentam gráficos modernos e realistas, como a *Unreal*, uma *engine* de *games* desenvolvida pela *Epic Games*. Ela é utilizada em diversos trabalhos de simulação e, dentre esses trabalhos, destaca-se (Savva et al. 2017) no qual é descrito o MINOS (*Multimodal Indoor Simulator for Navigation in Complex Environments*), um simulador criado para testes de navegação e inteligência artificial em robôs, utilizando técnicas multissensoriais. Porém, sem ter o foco específico em VANTs, esses veículos não estão modelados neste sensor.

O UE4Sim, desenvolvido por Mueller et al. (Müller 2017) é um outro trabalho de simulação utilizando a *Unreal Engine*. Para este simulador, os autores modelaram o corpo e a física de VANTs quadrimotores e carros, com o intuito de testar algoritmos de detecção e rastreamento por intermédio de visão computacional. Porém, este simulador conta com um número limitado de sensores, não apresentando sensores colaborativos ou inerciais.

Por fim, tem-se o AirSim – *Aerial Informatics and Robotics Simulation* (Shah et al. 2017), desenvolvido pela Microsoft para teste de inteligência artificial e navegação em VANTs e em carros autônomos. Ele tem a sua essência criada com base no *Unreal Engine*, contém a modelagem de um VANT com uma ótima simulação física de dinâmica e cinemática para quadrimotores, e tem como vantagens uma API para controle de VANT por meio do protocolo MavLink (*Micro Air Vehicle Link*), similar à utilizada em VANTs reais. Permitindo que se utilize os mesmos comandos empregados em aeronaves reais, ele contém diversos tipos de sensores simulados, como câmeras visuais e de profundidade, além de sensores inerciais (acelerômetro, giroscópio, barômetro, magnetômetro e GPS).

Assim, o AirSim é o único simulador que apresenta todas as métricas analisadas, sendo,

desta forma, o escolhido para as simulações realizadas neste trabalho.

5. ARQUITETURA PROPOSTA PARA O SENSORIAMENTO

Para criação de sistema que permita múltiplos sensores trabalharem em conjunto na detecção de um objeto em risco de colisão, foi necessária a criação de uma arquitetura eficiente com múltiplos algoritmos, processando dados vindo de diferentes fontes.

Visando propor uma solução elegante para este problema, a construção do sistema se baseou em módulos seguindo os conceitos de coesão e acoplamento da engenharia de software. Coesão, segundo o princípio de responsabilidade única (E Gamma, R Helm, R Johnson 1994), refere-se à ideia de que cada módulos teve cumprir com apenas uma determinada função, enquanto o acoplamento refere-se ao relacionamento entre os módulos, tendo sempre em vista sistemas nos quais os módulos têm baixo relacionamento, comunicando-se com os demais por meio de uma interface específica.

5.1. *Observer* invertido

Porém a comunicação dos sensores com o algoritmo de detecção é um processo de fusão de algoritmos e um processo complexo, pois um dos seus requisitos é permitir uma quantidade variada de sensores. Para que a arquitetura proposta seja robusta a essa personalização, preferencialmente permitindo a troca de sensores também em tempo de execução, é necessário a criação de um *design* específico.

Na engenharia de *software* existem dois padrões de *design* clássicos que resolvem em parte este requisito, o *Strategy* e o *Observer*, porém nenhum deles resolve este requisito totalmente.

O *Strategy* (E Gamma, R Helm, R Johnson 1994) tem como principal foco encapsular um algoritmo, permitindo que o sistema defina uma família de algoritmos similares que possam ser alterados em tempo de execução, algo similar ao que ocorre no *framework* proposto neste trabalho. Porém, este padrão não permite que vários algoritmos trabalhem de forma independente ao mesmo tempo e com o mesmo conjunto de dados, nem cria uma maneira de comunicação com uma classe gerenciadora.

Por outro lado, o *Observer* (E Gamma, R Helm, R Johnson 1994) é um padrão que define uma dependência de "um para muitos" entre objetos. Nele, uma classe principal, chamada de *subject*, pode conter um número variado de objetos dependentes, chamados de *observer*. Este padrão visa facilitar a atualização do estado dos *observer*, caso ocorra uma mudança na classe *subject*. Porém, no sistema proposto, o caminho é o inverso, ou seja, existe uma dependência de "muitos para um", pois nele há somente um objeto da classe fusão de sensores, que é dependente de dados vindo de diversos objetos da classe de algoritmos de detecção.

Para resolver estes impasses neste trabalho é proposta a criação de um padrão alternativo, que unifica o *Observer* e a *Strategy*, funcionando com um padrão de dependência de "muitos para um", no qual pode-se alterar o algoritmo em tempo de execução. Nele a classe observadora é a Detecção, e esta, por sua vez, contém uma lista de classes fornecedoras de dados, que são os objetos da classe algoritmo de detecção. Os algoritmos, ao encontrarem um novo objeto em risco de colisão, enviam um sinal à classe fusão de sensores com os dados do objeto observado. Para permitir o envio deste sinal é utilizado o conceito de delegação, fazendo com que a classe de detecção tenha uma referência do objeto da classe fusão de sensores. Assim, permite-se ao sensoriamento executar a ação de atualizar o objeto de detecção, ou seja, a classe observadora delega a ação de atualização às classes fornecedoras.

A Figura 3 existe um exemplo da relação entre as classes *Fusão de sensores* e *algoritmo de detecção*, exibindo a organização do padrão proposto segundo a linguagem UML. Nela as classes descritas como interface contém apenas o padrão de comunicação, e as classes concretas herdam estes padrões, além de implementação da técnica utilizada.

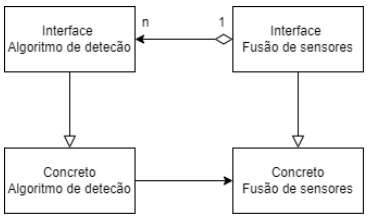

Figure 3: Diagrama de classes *Observe* Invertido.

Para exemplificar esse modelo, pode-se escolher um caso no qual a detecção ocorre por meio de dois sensores, uma câmera visual, utilizando o algoritmo SVM, e uma câmera de profundidade, utilizando o algoritmo de árvores de decisão. Para a fusão dos sensores é utilizada uma média simples da posição detectada em cada algoritmo. Segundo este padrão, a classe dos dois algoritmos de detecção teria que herdar a comunicação da classe *Interface de Algoritmos de Detecção*, e assim a comunicação de ambos os algoritmos utiliza o mesmo padrão. Enquanto isso, a classe de fusão por média simples irá herdar a comunicação da classe *Interface de fusão de sensores*, comunicando-se com as demais com uma modelo da interface. Devido a todas as classes utilizarem um modelo pré-definido de comunicação, alterações pontuais em cada uma delas podem ocorrer sem a necessidade de mudanças nas demais, permitindo, por exemplo, a troca do algoritmo de detecção por câmera visual, sem exigir alterações no módulo de fusões de sensores. A Figure 4 ilustra o exemplo dado

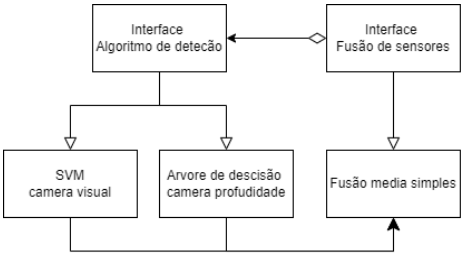

Figure 4: Exemplo de utilização do *Observer* Invertido.

6. TESTES

6.1. Cenário de teste

Os testes foram conduzidos criando-se uma simulação em um ambiente 3D no qual existem sempre apenas dois objetos: o veículo controlado, no caso um VANT quadrimotor, e o veículo em rota de colisão, um avião modelo Boeing 737.

Os cenários foram modelados de modo que ambas as aeronaves tenham uma rota retilínea e em todas as execuções gerem uma colisão frontal entre o VANT e o avião.

Para se calcular uma rota de colisão foi criada uma função que recebe quatro parâmetros: o ponto de colisão, o ângulo θ entre suas rotas, o tempo para colisão (t) e velocidade (v) da aeronave.

No total, foram gerados 9 cenários de testes distintos, nos quais se altera o ângulo θ de incidência entre a rota do VANT e a da aeronave, variando entre 40º e -40º, de 10º em 10º. A Figure 5 ilustra esse modelo.

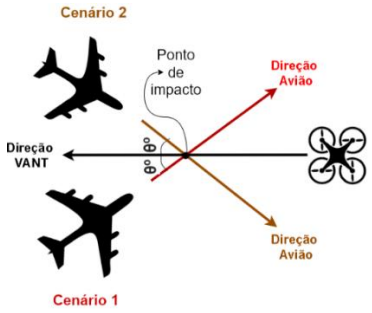

Figure 5: Ilustração de um cenário de colisão.

6.2. Algoritmos analisado

Nos testes foram analisados 3 sensores de detecção diferentes, câmeras visuais, câmeras de profundidade e o ADS-B, todos simulados através do AirSim.

Para detecção por meio do ADS-B, foram utilizados apenas cálculos geométricos, que permitem calcular a distância entre as aeronaves considerando os dados recebidos pelo sensor. Na detecção do objeto na imagem foram utilizados os mesmos algoritmos para os dados das câmeras visuais e de profundidade, porém com treinamentos diferentes. São eles: SVM, *Random Forest*, *Ligthgbm*, *Percetron* multicamadas, *Naive Bayes* e *Deep Learning* (com base no *Xception*). Porém, nos algoritmos utilizando câmeras visuais, foi utilizando a visão binocular para ser possível calcular a posição do objeto.

Após os testes das detecções individuais, os algoritmos com melhores resultados de precisão e cobertura foram testados em fusão,

e a partir disso foram criados 8 cenários diferentes de simulação, São eles:

1. Visão binocular (maior cobertura) + Câmera de profundidade (maior cobertura);
2. Visão binocular (maior precisão) + Câmera de profundidade (maior cobertura);
3. Visão binocular (maior cobertura) + Câmera de profundidade (maior precisão);
4. Visão binocular (maior precisão) + Câmera de profundidade (maior precisão);
5. Visão binocular (maior cobertura) + ADS-B;
6. Câmera de profundidade (maior cobertura) + ADS-B;
7. Visão binocular (maior cobertura) + Câmera de profundidade (maior cobertura) + Misto Câmeras + ADS-B;
8. Visão binocular (maior precisão) + Câmera de profundidade (maior precisão) + Misto Câmeras + ADS-B.

Os algoritmos de detecção utilizando apenas um sensor, e os utilizando fusão de sensores, foram todos analisados por meio das mesmas métricas.

A fusão de sensores ocorreu utilizando dois algoritmos diferentes, a média simples e a média ponderada pela precisão encontrada nos algoritmos no primeiro teste.

6.3. Métricas de análises

A avaliação foi baseada no erro entre a posição detectada da aeronave e sua posição real. A cada detecção foi calculada a distância euclidiana, em metros, entre essas duas posições (a posição detectada e a posição real). A partir dessa distância foi verificado se o objeto foi encontrado corretamente, ou não. Foi considerada uma detecção correta aquela em que a distância entre a posição real e a detectada é menor do que 100 metros, enquanto que para distâncias maiores do que 100 metros foram consideras detecções erradas. A partir deste cálculo, foram obtidos quatro métricas de avaliação para cada algoritmo, que são:

• **Média do erro:** Também denotada por μ, a média do erro é calculada por meio da média da distância entre a posição da aeronave detectada e a posição real da aeronave, em todos os *frames*. Porém, no cálculo só se leva em consideração detecções consideradas corretas, ou seja, detecções nas quais essa distância é menor do que 100 metros. Dessa forma, a média do erro indica a precisão do algoritmo.

• **Taxa de verdadeiro positivo:** Representam a taxa de detecções consideradas corretas considerando todas as detecções feitas pelo algoritmo, ou seja, a porcentagem de detecções no qual o erro é menor que 100 metros. Taxas altas de verdadeiros positivos significam que o algoritmo tem grande cobertura de detecção correta na maioria dos *frames*. Essa taxa é representada em termos de porcentagem.

• **Taxa de falso positivo**: Representam a porcentagem de detecções consideradas inválidas, ou seja, aquelas que apresentam distância maior do que 100 metros entre as posições real e detectada da aeronave. Taxas de falso positivo altas sinalizam que o algoritmo tem a tendência de detectar aeronaves em locais onde elas não estão. Nestes casos, houve uma detecção, porém, essas detecções estão erradas.

• **Taxa de falso negativo:** Assim como os falsos positivos, a taxa de falsos negativos também calcula uma porcentagem de erro, porém neste caso o erro se dá por não haver uma detecção onde existe uma aeronave. Dessa forma, essa taxa calcula a porcentagem de frames contendo aeronaves nos quais não foi identificada a aeronave pelo algoritmo. Algoritmos com altas taxas de falso negativo podem significar *overfit*, que ocorre quando o algoritmo está tão otimizado para um determinado conjunto de dados que não reconhece dados de mesma classe fora do conjunto de treinamento.

7. RESULTADOS

7.1. Detecções individuais

Os resultados das métricas analisadas para as detecções utilizando apenas um sensor de visão computacional estão apresentados na Tabela 1. Nela está destacado, em negrito, os valores dos resultados que apresentam maior

cobertura (%VP) e maior precisão (μ). O resultado obtido pelo uso do ADS-B está mostrado separadamente na Tabela 2.

7.2. Detecções por fusão de sensores

Para análises por fusão de sensores, foram criados 8 cenários diferentes, que variam conforme o número sensores testados e os algoritmos utilizados. Os cenários estão descritos no item 6.2.

Para os cenários foram utilizados os algoritmos de maior cobertura e com maior precisão, referentes a cada sensor. Na visão binocular, o mais preciso foi o algoritmo *Naive Bayes*, enquanto o mais com maior cobertura o *Deep Learning*, enquanto os dados de câmera de profundidade, o mais precisso o *Deep Learning*, a maior cobertura o *Random Forest*. A Tabela 3 apresenta esses resultados.

Tabela 1: Média de erro dos algoritmos de detecção entre os algoritmos de aprendizado supervisionado: os valores em negrito representam os melhores valores para um algoritmo segundo a métrica analisada

Algoritmos	***Visão Binocular***				***Câmera de Profundidade***			
Detecção	**μ*(m)***	***%VP***	***%FP***	***%FN***	**μ(*m*)**	***%VP***	***%FP***	***%FN***
SVM	*41,6*	*34%*	*64%*	*2%*	*42,5*	*93%*	*7%*	*0%*
Random Forest	*28,8*	***47%***	*53%*	*0%*	*34.3*	***100%***	*0%*	*0%*
LigthGBM	*32,6*	*31%*	*56%*	*13%*	*49,1*	*2%*	*98%*	*0%*
Naive Bayes	***28,2***	*20%*	*26%*	*54%*	*37,4*	*87%*	*13%*	*0%*
Rede Neural	*34,4*	*33%*	*64%*	*0%*	*43,2*	*97%*	*3%*	*0%*
Deep Learning	*39,23*	***50%***	*25%*	*25%*	***32,0***	*78%*	*0%*	*22%*

Tabela 2: Resultados de detecção de aeronave utilizando ADS-B

Algoritmo	**μ *(M)***	***%VP***	***%FP***	***%FN***
ADS-B	*0,69*	*99,6%*	*0%*	*0,4%*

Tabela 3: Resultados da detecção por fusão de sensores

Técnica de fusão	***Média simples***				***Média ponderada***			
COMBINAÇÃO	**μ*(m)***	***%VP***	***%FP***	***%FN***	**μ*(m)***	***%VP***	***%FP***	***%FN***
1	*34,9*	*100%*	*0%*	*0%*	*36,3*	*16%*	*82%*	*0%*
2	*23,6*	*100%*	*0%*	*0%*	*18,5*	*31%*	*70%*	*0%*
3	*51,3*	*44%*	*56%*	*0%*	*26,3*	*43%*	*57%*	*0%*
4	*48,9*	*80%*	*20%*	*0%*	*42,0*	*44%*	*56%*	*0%*
5	*30,5*	*50%*	*50%*	*0%*	*13,1*	*20%*	*80%*	*0%*
6	*18,3*	*98%*	*2%*	*0%*	*19,0*	*82%*	*18%*	*0%*
7	*18,8*	*100%*	*0%*	*0%*	*16,7*	*98%*	*2%*	*0%*
8	*16,4*	*99%*	*1%*	*0%*	*16,7*	*91%*	*9%*	*0%*

8. CONCLUSÕES

Com estes testes é possível descobrir o grau de confiança de cada de cada técnica de detecção, verificando qual pode ser a mais adequada para cada cenário, levando em conta eventuais restrições, como sensores disponíveis, capacidade de processamento ou limitações da aeronave. Nas câmeras foram alcançados resultados com taxas de precisão de 28 m em uma cobertura de 100% das imagens.

A arquitetura proposta também se mostrou eficiente para testes com diversos sensores, algo que pode ser implemento em veículos reais, assim protegendo o sistema de eventuais falhas, pois a arquitetura é capaz de funcionar com um número variável de sensores.

Para uma melhoria na análise do sistema pode-se, futuramente, testar novos sensores e

algoritmos, além de criar uma segunda etapa com técnicas de desvio do objeto em colisão, completando a etapa de *sense and avoid*.

9. REFERÊNCIAS

Cappello, Francesco, Subramanian Ramasamy, Roberto Sabatini, and Jing Liu. 2015. "Low-Cost Sensors Based Multi-Sensor Data Fusion Techniques for RPAS Navigation and Guidance." *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015* 714–22.

Carrio, Adrian, Yucong Lin, Srikanth Saripalli, and Pascual Campoy. 2017. "Obstacle Detection System for Small UAVs Using ADS-B and Thermal Imaging." *Journal of Intelligent & Robotic Systems* (May 2014).

Chollet, François. 2017. "Xception: Deep Learning with Depthwise Separable Convolutions." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 1251–58.

E Gamma, R Helm, R Johnson, J. Vlissides. 1994. *Design Patterns: Abstraction and Reuse of Object-Oriented Design*. Berlin: Springer.

Eurocontrol. 2017. "ACAS Guide Airborne Collision Avoidance." (December).

FAA - Federal Aviation Administration. 2011. "Introduction to TCAS II - Version 7.1." 1–50.

FAA - Federal Aviation Administration. 2021. "FAA Aerospace Forecast."

Fasano, Giancarmine, Domenico Accardo, Antonio Moccia, Ciro Carbone, Umberto Ciniglio, Federico Corraro, and Salvatore Luongo. 2008. "Multi-Sensor-Based Fully Autonomous Non-Cooperative Collision Avoidance System for Unmanned Air Vehicles." *Journal of Aerospace Computing, Information, and Communication* 5(10):338–60.

Forsyth, David and Jean Ponce. 2011. *Computer Vision: A Modern Approach*. Vol. 59.

Guo, Zhenhua, Lei Zhang, and David Zhang. 2010. "A Completed Modeling of Local Binary Pattern Operator for Texture Classification." 19(6):1657–63.

Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-yan Liu. 2017. "LightGBM : A Highly Efficient Gradient Boosting Decision Tree." (Nips):1–9.

Koenig, N. and A. Howard. 2004. "Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator." *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)* 3:2149–54.

Kuchar, J. 2005. "Safety Analysis Methodology for Unmanned Aerial Vehicle (UAV) Collision Avoidance Systems." *6 Th USA / Europe Seminar on Air Traffic Management Research and Development, Baltimore, MD, June* 27:30.

Lacher, Andrew R., David R. Maroney, and a D. Zeitlin. 2007. "Unmanned Aircraft Collision Avoidance–Technology Assessment and Evaluation Methods." *7th Air Traffic Managemtent Research & Development Seminar* 1–10.

Müller, Thomas. 2017. "Robust Drone Detection for Day/Night Counter-UAV with Static VIS and SWIR Cameras." 10190(0):1019018.

Pestana, Jesús, José Luis Sanchez-Lopez, Pascual Campoy, and Srikanth Saripalli. 2013. "Vision Based GPS-Denied Object Tracking and Following for Unmanned Aerial Vehicles." *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2013*.

Ramasamy, Subramanian, Roberto Sabatini, and Alessandro Gardi. 2014. "Avionics Sensor Fusion for Small Size Unmanned Aircraft Sense-and-Avoid." *2014 IEEE International Workshop on Metrology for Aerospace, MetroAeroSpace 2014 - Proceedings* (November):271–76.

Ramasamy, Subramanian, Roberto Sabatini, and Alessandro Gardi. 2016. "A Unified Approach to Separation Assurance and Collision Avoidance for Flight Management Systems." *AIAA/IEEE Digital Avionics Systems Conference - Proceedings* 2016-Decem:1–8.

Redmon, Joseph and Ali Farhadi. 2018. "YOLOv3: An Incremental Improvement."

Ross, Stephane, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J. Andrew Bagnell, and Martial Hebert. 2013. "Learning Monocular Reactive UAV Control in Cluttered Natural Environments." *Proceedings - IEEE International Conference on Robotics and Automation* 1765–72.

Russell, Stuart and Peter Norvig. 2010. *Artificial Intelligence A Modern Approach*. Pearson Education.

Savva, Manolis, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. 2017. "MINOS: Multimodal Indoor Simulator for Navigation in Complex Environments." 1–14.

Sendobry, Alexander, Heidelberger Printers, Stefan Kohlbrecher, Uwe Klingauf, and Oskar Von Stryk. 2012. "Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo." 7628(November 2012).

Shah, Shital, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. 2017. "Aerial Informatics and Robotics Platform." 1–17.

Wu, Yuanwei, Yao Sui, and Guanghui Wang. 2017. "Vision-Based Real-Time Aerial Object Localization and Tracking for UAV Sensing System." 1–9.

Yang, Hanxuan, Ling Shao, Feng Zheng, Liang Wang, Zhan Song, and Chain Monte. 2011. "Recent Advances and Trends in Visual Tracking : A Review." *Neurocomputing* 74(18):3823–31.

Yu, Xiang and Youmin Zhang. 2015. "Sense and Avoid Technologies with Applications to Unmanned Aircraft Systems: Review and Prospects." *Progress in Aerospace Sciences* 74:152–66.