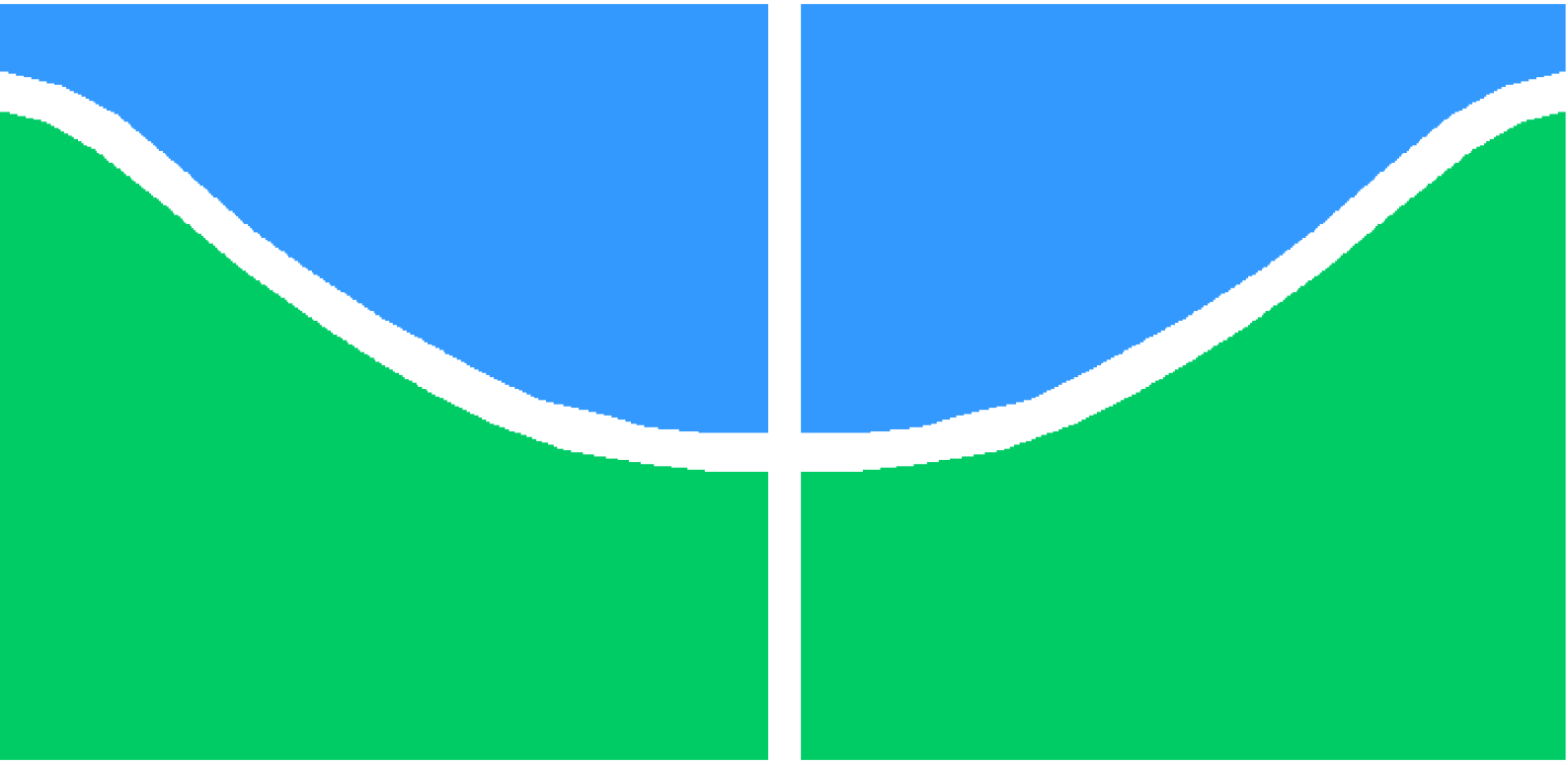

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia Eletrônica

Aplicação do algoritmo TLD para identificação e rastreio de veículos em imagens capturadas por aeronaves não-tripuladas

Autor: Filipe de Paula Carvalhêdo
Thiago Marques Siqueira

Orientador: Dr. Marcelino Monteiro de Andrade

Brasília, DF

2013

Filipe de Paula Carvalhêdo
Thiago Marques Siqueira

Aplicação do algoritmo TLD para identificação e rastreio de veículos em imagens capturadas por aeronaves não-tripuladas

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Marcelino Monteiro de Andrade

Brasília, DF

2013

Filipe de Paula Carvalhêdo
Thiago Marques Siqueira

Aplicação do algoritmo TLD para identificação e rastreio de veículos em imagens capturadas por aeronaves não-tripuladas/ Filipe de Paula Carvalhêdo Thiago Marques Siqueira. – Brasília, DF, 2013-

70 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Marcelino Monteiro de Andrade

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2013.

1. VANT. 2. TLD. I. Dr. Marcelino Monteiro de Andrade. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Aplicação do algoritmo TLD para identificação e rastreio de veículos em imagens capturadas por aeronaves não-tripuladas

CDU 02:141:005.6

Filipe de Paula Carvalhêdo
Thiago Marques Siqueira

Aplicação do algoritmo TLD para identificação e rastreio de veículos em imagens capturadas por aeronaves não-tripuladas

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Trabalho aprovado. Brasília, DF, 09 de dezembro de 2013:

Dr. Marcelino Monteiro de Andrade
Orientador

Dr. Edson Alves da Costa Júnior
Convidado 1

Dr. Fábio Macêdo Mendes
Convidado 2

Brasília, DF
2013

Resumo

Este trabalho apresenta uma aplicação do algoritmo TLD em uma plataforma aérea para auxiliar em rastreamento de automóveis. Neste trabalho foi realizado um levantamento histórico do uso de veículos aéreos não tripulados (VANTs), o surgimento de sistemas embarcados, o uso de processamento de imagem e a apresentação do algoritmo utilizado. A motivação e o desenvolvimento são apresentados de forma a deixar o leitor mais próximo do assunto que está sendo exposto, uma vez que a utilização dos componentes (multimotor e algoritmo de rastreamento) é recente e vem crescendo largamente com o uso de VANTs. O *software* OpenTLD foi aplicado para atender a solução, apresentando-se os resultados dos experimentos realizados a fim de avaliar o sistema que foi integrado.

Palavras-chaves: Veículos aéreos não tripulados. Processamento de imagens. Rastreamento de objetos. *OpenTLD*.

Abstract

This work presents an application of the algorithm TLD in an aerial platform to assist in automobile tracking. This work presents a historical survey of the use of unmanned aerial vehicles (UAVs), the emergence of embedded systems, using image processing and presentation of the algorithm used was performed. The motivation and development are presented in order to let the reader closer to the subject that is being built, since the use of components (multi-motor and tracking algorithm) is recent and growing widely with the use of UAVs. The software OpenTLD was applied to fit the solution and present the results of the experiments conducted to evaluate the system that has been integrated.

Key-words: Unmanned aerial vehicles. Image processing. Object tracking. OpenTLD.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

VANT	Veículo Aéreo não Tripulado
ARC	Aeronave Remotamente Controlada
UAV	*Unmanned Aerial Vehicles*
ANAC	Agência Nacional de Aviação Civil
TLD	*Tracking-Learning-Detection*
FPS	*Frames-Per-Second*
NTSC	*National Television System Committee*
FPV	*First Person View*
PWM	*Pulse-width modulation*
ESC	*Electronic Speed Control*

Sumário

1 Introdução

O interesse em Veículos Aéreos Não Tripulados (VANT), no Brasil conhecido como Aeronave Remotamente Contralada (ARC), tem crescido ao redor do mundo. Avanços recentes na tecnologia computacional, desenvolvimento de *software*, materiais mais leves, sistemas globais de navegação, avançados *links* de dados, sofisticados sensores e a miniaturização são os motivos do aumento de desenvolvimentos de ARCs. Atualmente, dezenas de países têm trabalho de desenvolvimento de ARC para diferentes mercados. Os Estados Unidos aparecem como um dos líderes em termos de tamanhos, tipos e sofisticações dos sistemas, voltados principalmente para aplicação militar. Outros países incluem o Japão, Coreia do Sul, Austrália, França, Inglaterra, Itália, Alemanha, além de Israel e África do Sul. O Japão se destaca com mais de 2000 ARCs aplicados em pulverização e outras aplicações na agricultura (SIMPSON, 2003).

Os ARCs podem ser divididos em duas categorias: asas fixas e asas rotativas. Essas plataformas aéreas móveis têm demonstrado, cada vez mais, uma grande utilidade para os meios de vigilância, tanto à segurança civil, como à prevenção de fogos florestais; na observação de fenômenos meteorológicos e, em especial, em ambientes de difícil acesso ao homem, entre outras aplicações.

O interesse no uso dessas aeronaves por parte comercial vem aumentando principalmente na área de imagiamento, quando equipada de uma câmera fotográfica/filmadora que transmita a imagem ao vivo através de um *link* de RF (rádio frequência). Uma aeronave não tripulada do tipo decolagem e pouso vertical (*Vertical Take-Off and Landing* — VTOL) de pequeno porte pode ser útil nas seguintes ocasiões: Fotos e filmagens panorâmicas de baixo custo para a indústria cinematográfica, mercado imobiliário (vista superior de casas, terrenos, chácaras e sítios), reportagens de telejornais, eventos esportivos, etc (AERIALS, 2013); Localização de um suspeito em recintos onde não há visada direta: dessa forma evita-se que um policial seja surpreendido pelo suspeito durante a sua procura; Inspeção de linhas de transmissão e distribuição elétrica; Monitoramento de plantações e grandes rebanhos.

1.1 Contextualização

A tecnologia cada vez mais aparece como um fator importante dentro de medidas de segurança pública. E uma das tecnologias mais utilizadas na vigilância é a câmera, por meio da qual essas imagens capturadas são enviadas a uma central de segurança responsável por monitorar atividades suspeitas. A Figura 1 mostra um desses centros de monitoramento. A Figura 2 ilustra o funcionamento de um sistema de vigilância utilizado

Figura 1 – Centro de monitoramento de câmeras (AFAUNANATAL, 2013).

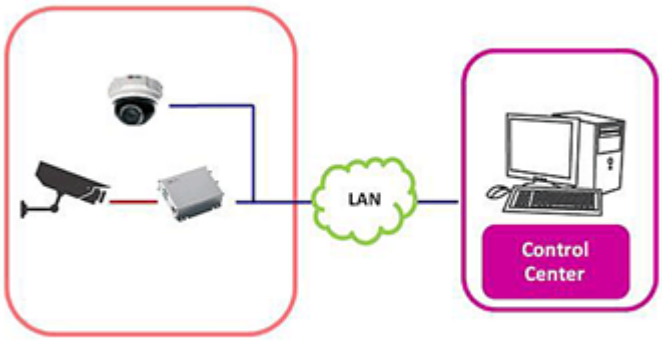

Figura 2 – Diagrama de câmeras em centros de controle (DIPOL, 2013).

em centros de segurança.

Neste estudo utilizaram-se veículos automotivos como objeto de pesquisa, pois esses possuem mobilidade para os testes estáticos e dinâmicos. O sistema (Figura 3) é constituído por uma câmera para a captura da imagem, um veículo aéreo não tripulado que possibilita maior mobilidade e agilidade, e uma plataforma computacional para o processamento de imagens.

Na parte referente ao multimotor foi feito um levantamento histórico de aplicações, tanto na área civil, como militar. Também foi realizado um estudo a respeito do funcionamento de ARCs e uma compilação das informações sobre a legislação presente

Figura 3 – Diagrama da solução proposta.

(2013) no que diz respeito a esses.

1.2 Justificativa

A principal motivação para este trabalho nasceu do interesse de desenvolver um novo sistema de rastreio de veículos, integrando processamento de imagens e ARCs. Justifica-se a escolha do tema devido ao fato de se ter observado, nos últimos anos, um maior investimento em ARCs, tanto na área militar como civil. O avanço em tecnologias VLSI (do inglês, *Very Large-Scale Integration*) e sistemas multiprocessados em um único chip (MPSoC, do inglês *Multiprocessor System-on-Chip*) (RODOLFO; CEZAR, 2009) possibilita a síntese de circuitos menores e com mais capacidade de processamento. Dessa forma, é possível aliar o processamento digital de imagens aplicando os conhecimentos adquiridos durante o curso de graduação em Engenharia Eletrônica para integrar uma solução na área de segurança.

1.3 Objetivos

1.3.1 Objetivo geral

A proposta do trabalho é apresentar um sistema para rastreamento de veículos, aplicando um algoritmo de reconhecimento de imagens com uma plataforma aérea não tripulada.

1.3.2 Objetivo específico

Os objetivos específicos são:

- Comparar o desempenho do algoritmo em diferentes plataformas de processamento;
- Realizar experimentos da aplicação do algoritmo *TLD* em diferentes plataformas aéreas.

2 Fundamentação Teórica

É ampla a literatura abordando ARCs, auxiliando a compreensão do seu funcionamento e descrição dos aparelhos. A seguir, abordar-se-á a estrutura dos ARCs, conjugada ao *software* utilizado para rastreamento de objetos (*OpenTLD*).

2.1 Aeronaves Remotamente Controladas (ARCs)

As ARCs são aeronaves que possuem a capacidade de voo com controle à distância ou autônomo, sendo normalmente aplicadas em situações repetitivas, hostis, perigosas ou de difícil acesso para aeronaves convencionais (FURTADO, 2008).

Na última década, observou-se um grande esforço para desenvolvimento e emprego de ARCs, tanto para aplicações militares como civis. As ARCs são comumente utilizados para fins militares, tais como o *Global Hawk*, *Aerosondem*, *Pr50*, todos com autonomia de voo superior a 40 minutos. Em muitos casos, esses sistemas são importantes ferramentas táticas de apoio à manutenção da soberania.

Nas aplicações militares, conforme o Departamento de Defesa Norte Americano (DoD), observa-se um acentuado investimento a partir de 2001 nos Estados Unidos da América, conforme é visto na Figura 4.

Em outro sentido, com menor estrutura e alcance mais limitado, encontram-se as mini-ARCs. Essas são menores, requerem menor infraestrutura para realização de missões e podem ser operados por pequenas equipes. A área de mini-ARCs encontra forte potencial de aplicação civil (Tabela 1). Nessa área, é possível observar a sua utilização em gerenciamento de queimadas, pesquisa ambiental, controle de poluição, segurança, monitoração de fronteira e agricultura (OLIVEIRA, 2005).

No Brasil, existem importantes iniciativas civis de utilização das ARCs, princi-

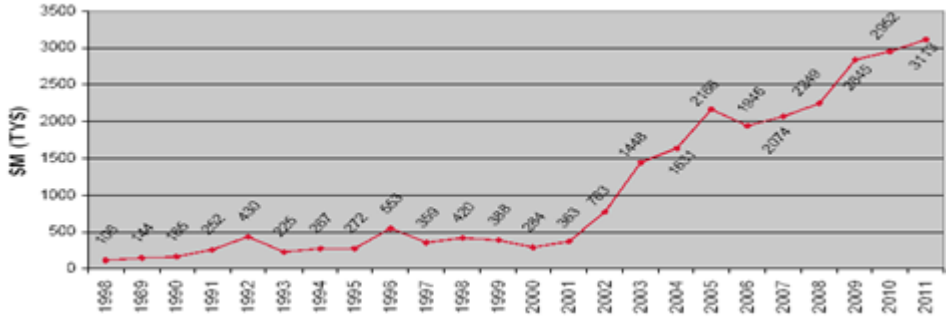

Figura 4 – Perfil dos recursos anuais do DoD para ARCs (FEDERAL, 2009)

.

Aplicação	Operador
Vigilância Policial de Áreas Urbanas	Polícias Estaduais
Vigilância de Áreas de Fronteira	Polícia Federal
Inspeção de Oleodutos e Gasodutos	Petrobras
Controle de Safras Agrícolas	Embrapa
Levantamento de Recursos Florestais e Controle de Queimadas	Ibama
Enlace de Comunicações	Empresa de Telecomunicações
Cobertura de Eventos para TV	Redes de TV

Tabela 1 – Iniciativas civis de utilização de ARCs no Brasil (OLIVEIRA, 2005)

palmente voltadas para as áreas agrícolas, gerenciamento de recursos e vigilância. Como exemplo, a Embrapa possui uma ação conjunta com a USP, denominada projeto ARARA, que busca desenvolver uma ARC para monitoramento ambiental e agrícola (RASI, 2008). A Petrobras investe nas ARCs para monitorar vazamentos e a Chesf para monitorar sua rede de transmissão de energia elétrica (FURTADO, 2008).

A capacidade de circular sem serem percebidas, guiadas remotamente a partir de informação recebida por sensores e câmeras, faz com que as ARCs sejam motivo de desconfiança. Em vários países há debate sobre ética e moral no emprego de ARCs, principalmente no que se refere a questões de privacidade.

Há temor de que a falta de transparência no uso dos veículos encubra possíveis abusos no monitoramento de áreas e pessoas, com interceptação de conversas telefônicas, fotografias e filmagens feitas de maneira irregular. Países e indústrias já estão sujeitos à espionagem, por exemplo.

As autoridades temem, ainda, o risco de colisão com aviões e obstáculos aéreos, bem como a possibilidade de que o equipamento caia sobre áreas habitadas, colocando em risco a vida de pessoas em solo. Controladas de uma cabine, as ARCs circulam sem garantia de que os operadores tenham total conhecimento da situação no ar.

A Agência Federal de Administração Aérea dos Estados Unidos (FAA), responsável pelo controle da aviação civil nos Estados Unidos, deve divulgar, ainda em 2013, normas referentes aos voos domésticos de ARCs. Vários países aguardam o documento para servir de base na criação de suas próprias leis. Atualmente, a operação civil ainda é bem controlada nos Estados Unidos, restrita à liberação de licenças individuais e proibida em regiões habitadas.

No Brasil, a Agência Nacional de Aviação Civil (ANAC) reconhece a importância do uso civil das ARCs, tanto para indústria como para a sociedade, mas afirma que: "devido aos novos desafios e características associadas ao voo remoto, são necessárias adequações na regulamentação desse tipo de aeronave para garantir níveis de segu-

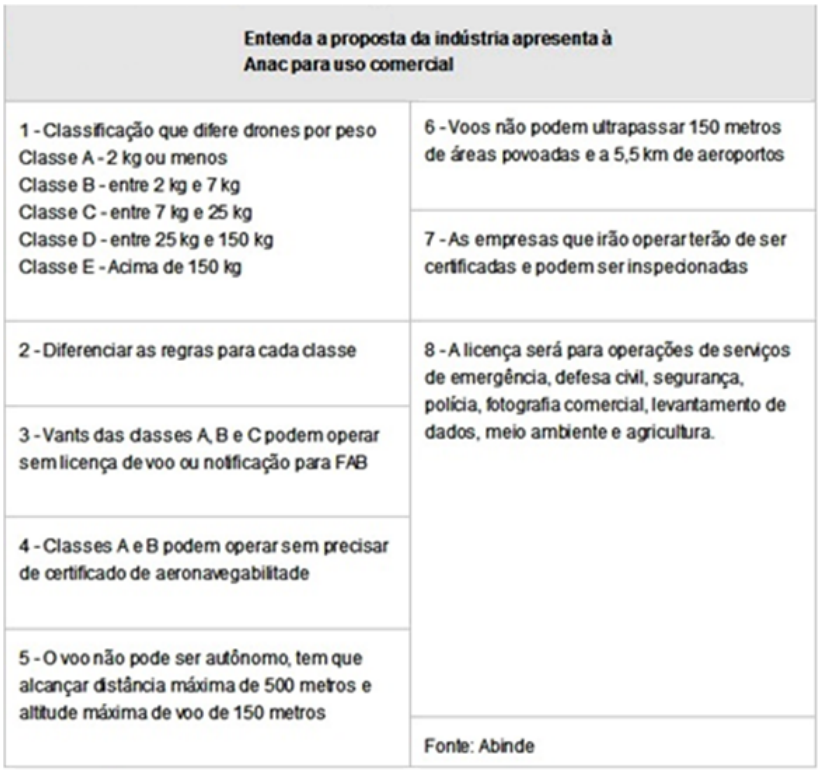

Figura 5 – Requisitos de uso de ARC propostos pela indústria à ANAC.

rança"(STOCHERO, 2013).

Tanto as normas da ANAC quanto as regras do Departamento de Controle do Espaço Aéreo (DECEA), da Aeronáutica, proíbem totalmente o voo de ARCs sobre cidades brasileiras. As demais operações precisam ser comunicadas à Aeronáutica com antecedência de 15 a 30 dias, para evitar que os veículos dividam o espaço aéreo com aviões comerciais.

Em outubro de 2012, a ANAC publicou no Diário Oficial a Instrução Suplementar (IS) 21-002, que prevê requisitos básicos para certificar os veículos. ARCs totalmente autônomas são proibidas. Interessados em obter a licença devem enviar para a agência informações sobre o modelo e o propósito da operação.

O Brasil lidera na América do Sul e também desponta no mundo, com iniciativas que envolvem o acesso de aeronaves remotamente controladas ao espaço aéreo. A Figura 5 apresenta a proposta da indústria à ANAC, para uso comercial. As ARCs podem ser divididos em três principais grupos: os aviões, helicópteros/multimotor e dirigíveis (*blimps*).

2.1.1 Aviões

Um exemplo de ARC é o *Predator Unmanned Aerial Vehicle*, (Figura 6) que foi desenvolvido pelo exército norte americano.

Figura 6 – *Predator* UAV (MURTAGH, 1983).

Figura 7 – Comparação entre dois ARCs utilizados no Brasil.

No Brasil, também há o uso de ARCs por parte da força aérea. A Figura 7 mostra o comparativos de duas ARCs, um da Força Aérea Brasileira (FAB) e outro da Polícia Federal (PF).

Uma característica em relação aos multimotores é que os aviões precisam de uma pista de pouso e decolagem, portanto não podem pousar nem decolar na vertical. Por outro lado, o avião possui mais estabilidade ao atingir a velocidade mínima de sustentação, podendo então planar, pois a força de sustentação na aeronave fica em suas asas (Figura

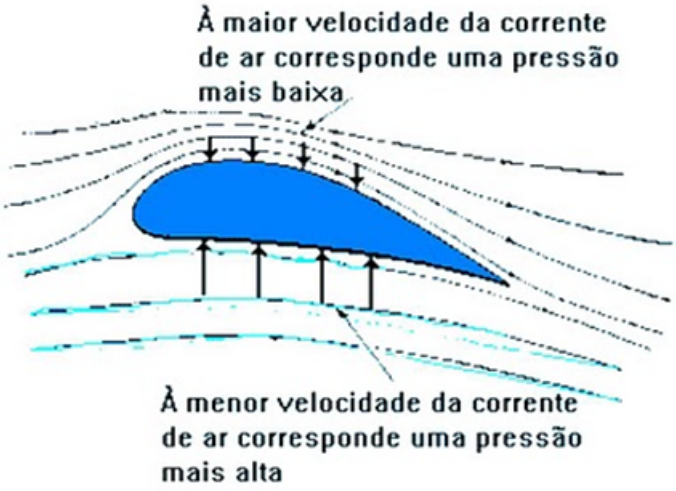

Figura 8 – Forças atuando na asa de um avião (PEREIRA, 2013).

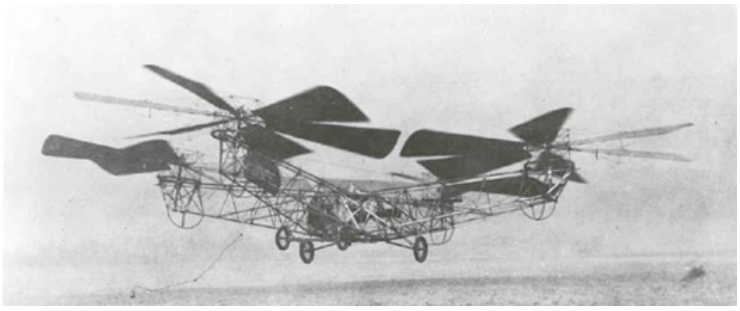

Figura 9 – De Bothezat Quadrotor, 1923. (BüCHI, 2013).

8). Devido a sua forma de locomoção e velocidade, os aviões são usados em ambientes externos (*outdoor*).

2.1.2 Helicópteros (Multimotor)

O primeiro quadrimotor surgiu em 1922 (Figura 9), foi construído para o serviço aéreo militar dos Estados Unidos por George de Bothezat, mas o projeto foi deixado de lado pela dificuldade e complexidade que o piloto tinha para controlar a aeronave. No entanto os multimotores de pequeno porte voltam a ganhar destaque, devido ao avanço da tecnologia e a estabilidade controlada de forma automática e eletrônica.

O multimotor é uma plataforma que, diferente do avião, possui sua força de sustentação nas asas. A força de sustentação é dividida em múltiplos rotores de mesma potência, de forma que o torque de reações de um cancele o do outro. Com isso, uma das virtudes do multimotor é que ele possui seis graus de liberdade, sendo eles translação ao longo nos três eixos X, Y e Z e rotação em torno destes mesmos eixos: *pitch*, *roll* e *yaw* (Figura 10).

Na Tabela 2 são mostradas as velocidade de cada motor para realizar os devidos

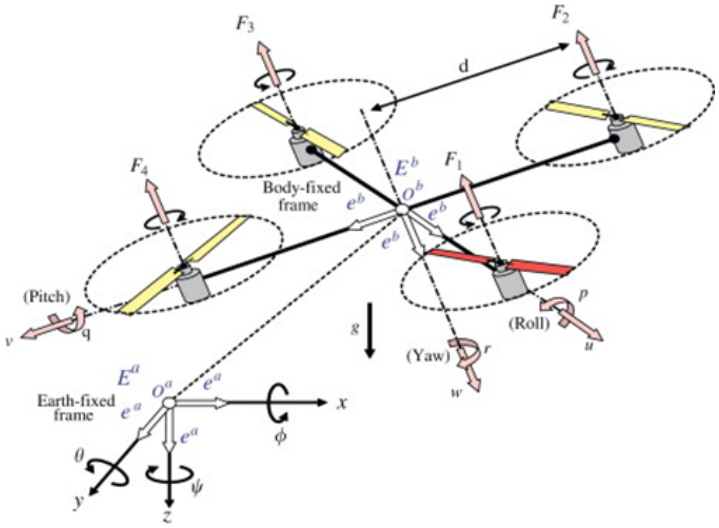

Figura 10 – Esquema físico de quadrimotor. (OZUYSAL; FUA; LEPETIT, 2007).

Movimento	Motor 1	Motor 2	Motor 3	Motor4
para cima (throttle+)	$v_1 + \Delta$v	$v_2 + \Delta$v	$v_3 + \Delta$v	$v_4 + \Delta$v
para baixo (throttle-)	v_1 - Δv	v_2 - Δv	v_3 - Δv	v_4 - Δv
para frente (pitch+)	v_1 - Δv	$v_2 + \Delta$v	v_3	v_4
para trás (pitch-)	$v_1 + \Delta$v	v_2 - Δv	v_3	v_4
para direita (roll+)	v_1	v_2	$v_3 + \Delta$v	v_4 - Δv
para esquerda (roll-)	v_1	v_2	v_3 - Δv	$v_4 + \Delta$v
horário (yaw+)	$v_1 + \Delta$v	$v_2 + \Delta$v	v_3 - Δv	v_4 - Δv
anti-horário (yaw-)	v_1 - Δv	v_2 - Δv	$v_3 + \Delta$v	$v_4 + \Delta$v

Tabela 2 – Rotação ao longo dos eixos da aeronave.

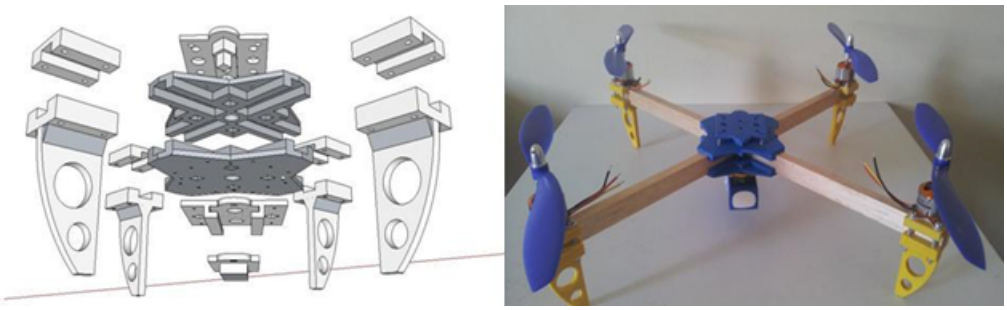

Figura 11 – Esqueleto de um quadrimotor utilizado no primeiro experimento deste trabalho.

movimentos de rotação nos eixos do quadrimotor.

Neste projeto foram usados asa fixa e multimotores. A Figura 11 mostra o quadrimotor utilizado em um dos experimentos.

Alguns dos componentes utilizados para montar um multimotor são:

- Estrutura onde ficam fixados todos os outros componentes;

- Motor *Brushless*, é um motor sem escovas de alta rotação;
- ESC, é um controlador eletrônico de velocidade dos motores;
- Controladora, é a central de processamento dos sinais dos sensores, dos sinais provenientes dos canais de rádio e controle dos ESCs;
- Hélices, conjunto de pás com um mesmo centro, que ao ser girado segundo o seu eixo causa propulsão em uma determinada direção;
- Transmissor e receptor de vídeo (FPV), responsável pela captura dos sinais de vídeo da câmera fixada no multimotor e transmissão do mesmo para o usuário em terra.

2.2 Sistemas Embarcados

Um sistema embarcado é um sistema microprocessado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla. Diferentemente de computadores de propósito geral, como o computador pessoal, um sistema embarcado realiza um conjunto de tarefas pré-definidas, geralmente com requisitos específicos. Já que o sistema é dedicado a tarefas específicas, por meior da engenharia pode-se otimizar o projeto, reduzindo tamanho, recursos computacionais e custo do produto.

Nos primeiros anos dos computadores digitais na década de 1940, eles eram dedicados a uma única tarefa. Eram, entretanto, muito grandes para serem considerados embarcados. O primeiro sistema embarcado reconhecido mundialmente foi o *Apollo Guidance Computer*, desenvolvido nos EUA por Charles Stark Draper no MIT para a NASA. O computador de guia, que operava em tempo real, era considerado o item eletrônico mais arriscado do projeto *Apollo*. No projeto desenvolvido pelo MIT foram usados circuitos integrados monolíticos, para reduzir o tamanho e peso do equipamento e aumentar a sua confiabilidade (ELETRICA, 2013).

Sistemas embarcados utilizam vários tipos de processadores: DSPs (*digital signal processors* - processadores digitais de sinais), microcontroladores, microprocessadores. Ao contrário do mercado de computadores pessoais, que é basicamente dominado pelos processadores de arquitetura x86 da Intel/AMD, sistemas embarcados utilizam amplamente as arquiteturas ARM, *PowerPC*, PIC, AVR, 8051, *Coldfire*, TMS320, *blackfin*.

Os sistemas embarcados estão mudando a forma como as pessoas vivem, trabalham, estudam, divertem e se interagem. Exemplos de tais sistemas (Figura 12) são os *smartphones*, *MP3 player*, o sistema de controle dos automóveis (computador de bordo, sistema ABS), os computadores portáteis, os fornos de microondas com controle de temperatura inteligente, as máquinas de lavar e outros eletrodomésticos.

Figura 12 – Sistema ABS e câmera digital (ABOUTAUTO, 2013).

2.3 Processamento de Imagem

Uma das primeiras aplicações de processamento de imagens remonta ao começo deste século, onde buscavam-se formas de aprimorar a qualidade de impressão de imagens digitalizadas transmitidas através do sistema *Bartlane*, que transmitia imagens por cabo submarino entre Londres e Nova Iorque. Os primeiros sistemas *Bartlane*, no início da década de 20, codificavam uma imagem em cinco níveis de intensidades distintas. Essa capacidade seria expandida, já em 1929, para 15 níveis, ao mesmo tempo em que era desenvolvido um método aprimorado de revelação de filmes por meio de feixes de luz modulados por uma fita que continha informações codificadas sobre a imagem (HUGO, 1999).

Processamento de imagem é qualquer forma de processamento de dados no qual a entrada e saída são imagens, tais como fotografias ou quadros de vídeo. O interesse em métodos de processamento de imagens digitais vem aumentando devido as principais aplicações, como: medicina, geoprocessamento, robótica e telecomunicações dentre outras. Em sua definição estão presentes a melhoria da informação de imagens e vídeos, para interpretação humana e processamento de dados de imagem para o armazenamento, transmissão e representação para a percepção de máquinas autônomas.

Os elementos de um sistema de processamento de imagens de uso genérico são mostrados na Figura 13. Esse diagrama (HUGO, 1999) permite representar desde sistemas de baixo custo até sofisticadas estações de trabalho utilizadas em aplicações que envolvem intenso uso de imagens. Ele abrange as principais operações que se pode efetuar sobre uma imagem, a saber: aquisição, armazenamento, processamento e exibição.

Aliado com o processamento de imagens digital, geralmente também tem-se a visão computacional, ou seja, um sistema computadorizado capaz de adquirir, processar e interpretar imagens correspondentes a cenas reais onde, a partir das informações, um sistema pode tomar decisões. A Figura 14 mostra esquematicamente o sistema de visão computacional.

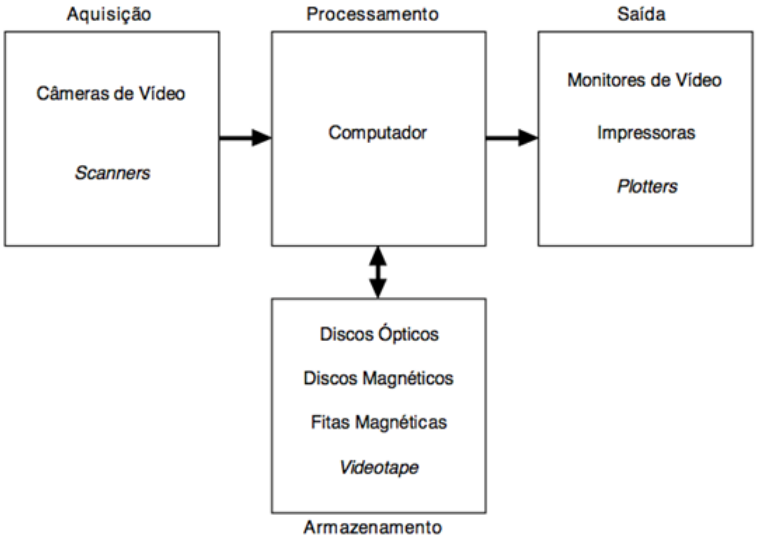

Figura 13 – Elementos de um sistema de processamento de imagens (FEDERAL, 2009).

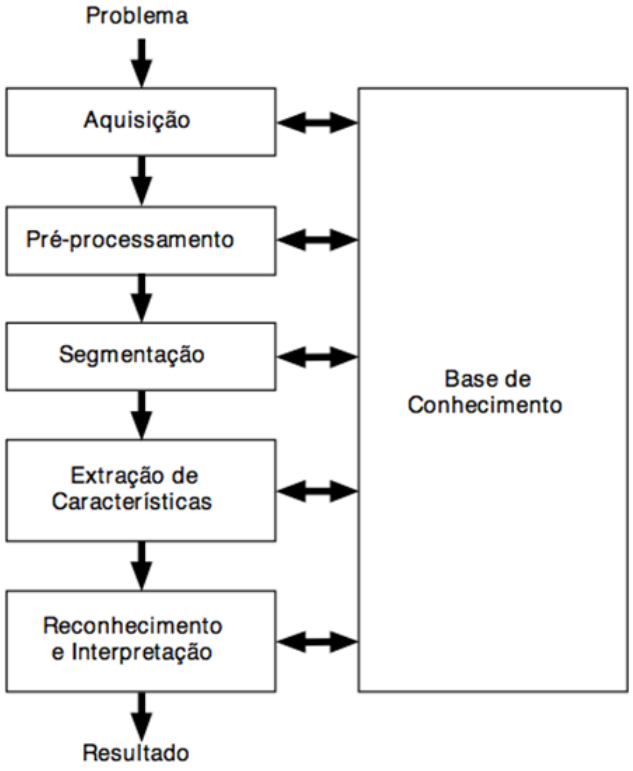

Figura 14 – Um sistema de visão computacional e suas principais etapas (HUGO, 1999).

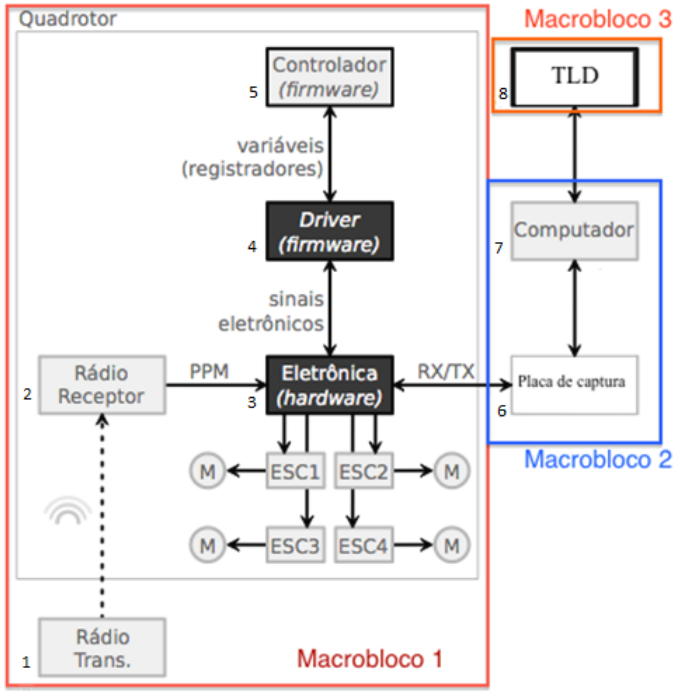

Figura 15 – Diagrama de blocos do sistema proposto.

Figura 16 – Diagrama da solução proposta.

2.4 Proposta do Sistema

Os principais blocos (Figura 15) que compõem o sistema de rastreamento de automóveis são apresentados. O sistema (Figura 16) contém uma câmera para a captura de imagens, essa é posicionada em uma ARC, também possuindo um dispositivo de transmissão para o envio das imagens capturadas. O sinal enviado pela ARC é capturado por um receptor em uma estação, para processar a informação e mostrar os resultados. Visando melhor exemplificar, o sistema proposto foi dividido em blocos.

O sistema é composto por oito blocos, mas neste trabalho o sistema foi reorganizado em três grandes blocos. Cada um desses três macroblocos são explicados nas subseções subsequentes. A primeira parte é referente à plataforma ARC, que neste pode ser um multimotor ou uma asa fixa. Esses dois representam os blocos 1, 2, 3, 4 e 5, conforme é visto o agrupamento na Figura 15. A segunda parte é referente ao sistema de captura e o dispositivo onde as informações são processadas, que são os blocos 6 e 7; e a terceira parte, algoritmo *TLD* é representada pelo bloco 8.

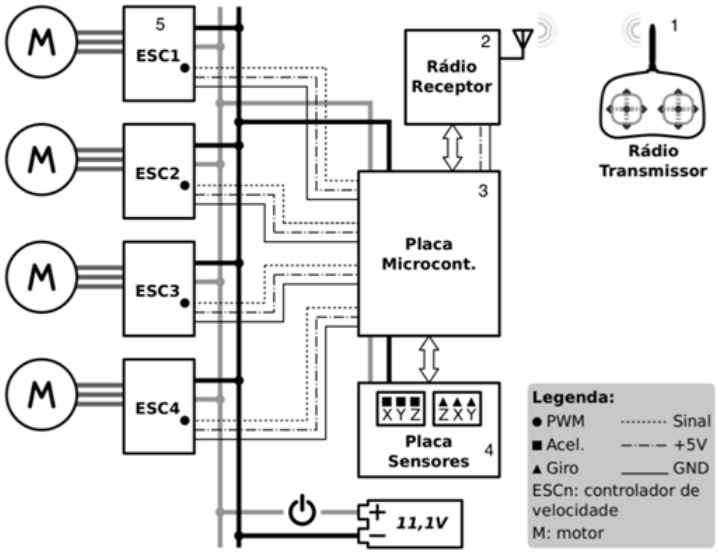

Figura 17 – Diagrama de blocos elétrico-eletrônico de uma plataforma ARC (MELO, 2010)

2.4.1 Macrobloco 1 - Plataforma ARC

Nessa subseção serão apresentados os sub-blocos que são referentes à plataforma ARC. A Figura 17 representa o diagrama de blocos elétrico-eletrônico da plataforma ARC.

O primeiro bloco que constitui a plataforma ARC é um rádio transmissor. Para se controlar uma aeronave ou embarcação remotamente, utiliza- se um *link* de RF (*Radio frequency* - Rádio Frequência) do tipo FM (*Frequency Modulation - Modulação em frequência*) composto por um rádio transmissor e um rádio receptor com portadoras em várias frequências, como 72MHz e 2,4GHz, essas frequências são liberadas pelo orgão competente (ANATEL no Brasil). Os comandos de voo são enviados serialmente, de 20ms em 20ms, e modulados por posição de pulso, isto é, no formato PPM (*Pulse Position Modulation*). O segundo bloco, que constitui uma plataforma ARC é o rádio receptor, responsável por demultiplexar o sinal e enviar para a placa controladora a informação enviada pelo rádio transmissor.

O terceiro bloco é referente à placa microcontroladora. A placa principal, ou microcontrolada, deve ser responsável por: capturar os sinais oriundos do sensores acelerômetros (X, Y e Z), giroscópios (X, Y e Z) e sistema de navegação (GPS); capturar os canais (comandos de voo *throttle*, *pitch*, *roll* e *yaw*) do rádio receptor; rodar algum algoritmo de controle para estabilização de voo; gerar os sinais de PWM para os ESCs.

Uma solução de controladora de voo é a *Ardupilot* (Figura 18). Essa microcontroladora pode ser utilizada para o uso de asas fixas ou multimotores, sendo capaz de controlar até oito motores, conforme é visto na Figura 19.

O quarto bloco é referente à instrumentação, ou seja, à placa de sensores. Nesse

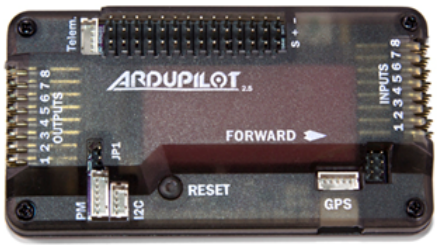

Figura 18 – Microcontroladora *Ardupilot*.

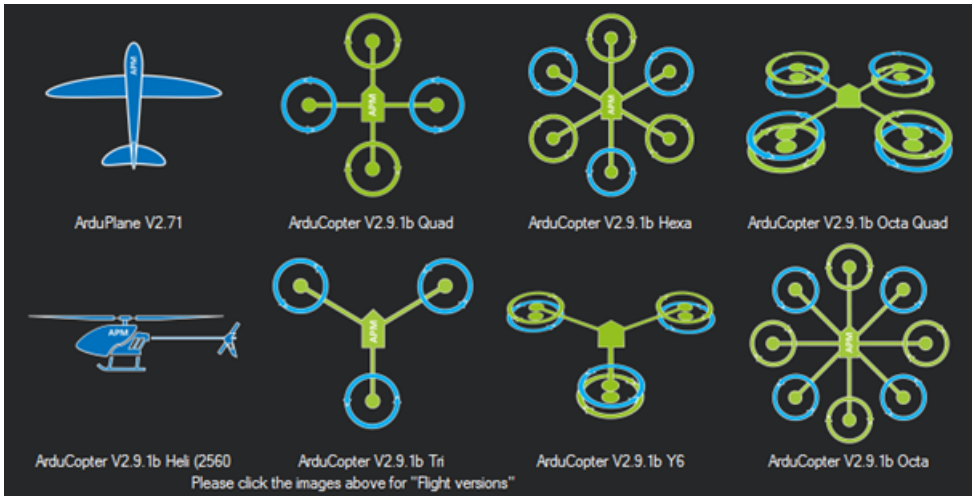

Figura 19 – Plataformas controladas pela *Ardupilot*.

bloco se encontram o acelerômetro, os giroscópios e o sistema de navegação (GPS), para proporcionar uma melhor estabilização da aeronave.

O quinto bloco é referente aos ESCs e os motores do tipo *brushless*. Diferentemente dos motores de corrente contínua (DC), os motores do tipo *brushless* não são alimentados por meio de escovas, mas sim por um circuito eletrônico. Este circuito, além de prover energia aos enrolamentos de um motor *brushless* a partir de uma fonte DC, também realiza um controle de velocidade em malha fechada. Por isso, tal circuito é chamado de Controle Eletrônico de Velocidade, ESC (*Electronic Speed Control*).

Um ESC é basicamente dividido em duas partes: uma de controle e outra de potência. A primeira é onde a controladora do ESC recebe o sinal PWM com período de 20ms, vindo da microcontroladora, por exemplo a *Ardupilot* e gera três outros, normalmente trapezoidais e defasados entre si de 120°. Esse bloco pode variar em quantidade de ESCs e motores, dependendo da plataforma a ser utilizada.

Outras plataformas de multimotores já são usadas pela polícia norte americana e do Canadá, sendo uma delas a plataforma de uso comercial o *Draganflyer X6* (Figura 20). A plataforma do *Draganflyer X6* tem sua estrutura em fibra de carbono, contém 11 sensores internos (três giroscópios, três acelerômetros, três magnetômetros, um barômetro

Figura 20 – Aeronave *Draganflyer X6*.

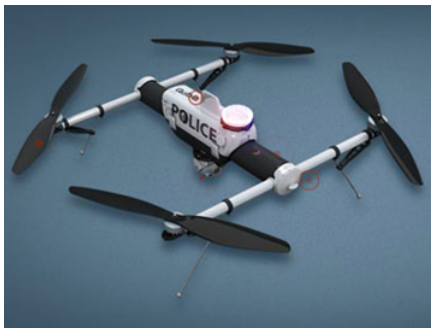

Figura 21 – Aeronave *AeroVironment Qube drone*.

e um receptor GPS), possui retorno de vídeo em tempo real.

Outra plataforma utilizada pela polícia americana é o *AeroVironment Qube drone* (Figura 21). A autonomia da aeronave é superior a 40 minutos, *link* de dados bidirecional com 1 km e criptografado e o sistema é capaz de rastreamento de movimento. O uso de outras possíveis plataformas são mostrados no Anexo A.

2.4.2 Macrobloco 2 - Sistema de Captura e Processamento das Informações

O sistema de captura é referente ao modo do envio da informação (vídeo) para uma central onde os dados serão recebidos e processados. Para o envio da informação pode-se utilizar uma placa FPV (*First Person View*), ou uma câmera com o envio de sinal *Wi-Fi*. Nesse trabalho utilizaram-se duas formas de processamento das informações, o primeiro modo foi com uma plataforma embarcada e o segundo modo utilizando um *Notebook*.

Uma possibilidade de uso de uma plataforma embarcada para o projeto é o *Raspberry Pi* (Figura 22). Esse é considerado um computador, que tem o tamanho de um cartão de crédito, desenvolvido no Reino Unido pela Fundação *Raspberry Pi*. O *hardware* é uma única placa. Seu principal objetivo é estimular o ensino de ciência da computação básica em escolas. O computador é baseado em um *chip BroadcomBCM2835*, que dispõe de um processador ARM1176JZF-Sde 700 MHz, *GPUVideoCore* IV, e 512 *Megabytes* de memória RAM (J.; LIN; YANG, 2008).

Figura 22 – Plataforma embarcada *Raspberry Pi*.

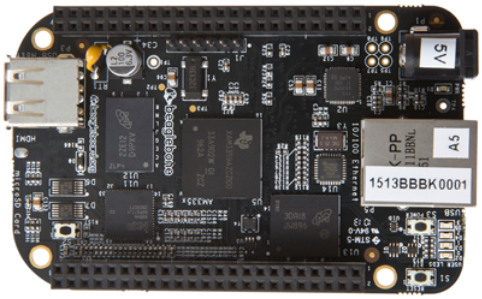

Figura 23 – Plataforma embarcada *BeagleBone Black*.

Uma segunda opção para uma plataforma embarcada é a *BeagleBone Black* (Figura 23). A *BeagleBone Black* é uma plataforma evoluida da linha de produtos *BeagleBoard*, um produto com dimensões físicas pequenas, muito poderoso e extremamente expansível que permite aos construtores, fabricantes, artistas e engenheiros a capacidade de criar projetos com a plataforma.

A *Beaglebone Black* é composta por um *ARM Cortex-A8* de 1GHz, um acelerador gráfico 3D *POWERVR SGX*, 512MB de RAM, *flash* interna de 2GB (eMMC) e entrada para cartão *MicroSD*, portas USB *host* e *device*, interface *Ethernet*, saída HDMI via um conector *microHDMI*. Na Tabela 3 são apresentadas as comparações entre as duas plataformas.

Relacionado a processamento, o *Raspberry Pi* usa um conjunto de instruções do *ARMv6*, enquanto a *BeagleBone Black* usa o conjunto de instruções *AMRv7*. Uma das vantagens da arquitetura *ARMv7* sobre o *ARMv6* é a melhoria de dempenho quando os processadores estão executando com a mesma velocidade de *clock*: o processador *ARMv7* chega a ser quase duas vezes mais rápido sobre o processador *ARMv6*. Uma vantagem da *Raspberry Pi* sobre a *BeagleBone* é em relação ao video, com o processador gráfico integrado *Videocore*. A GPU da *BeagleBone Black SGX530* já está no mercado há algum tempo enquanto a GPU da *Raspberry Pi*, a *Videocore* 4, é mais nova e apresenta um bom desempenho, podendo chegar a ser duas vezes mais rápido que a *SGX530*. A *Raspberry Pi* é capaz de decodificar video em *stream* 1080p, enquanto a *BeagleBone Black* não possui suporte a 1080p (LEONARD, 2013).

	BeagleBone Black	Raspberry Pi
Preço Base (dólar)	45	35
Processador	1GHz TI Sitara AM3359 ARM Cortex A8	700 MHz ARM1176JZFS
RAM	512 MB DDR3L a 400 MHz	512 MB SDRAM a 400 MHz
Armazenamento	2 GB on-board eMMC, MicroSD	SD
Ligações de vídeo	1 Mini-HDMI	1 HDMI, 1 Composto
Resoluções suportadas	1280×1024 (05 : 04), $1024\times$ 768 (4 : 3), 1280×720 (16 : 9), 1440×900 (16 : 10) tudo em 16 bits	Extenso de 640×350 até 1920×1200, o que inclui $1080p$
Audio	Stereo sobre HDMI	Stereo sobre HDMI, Stereo de 3,5 mm
Sistemas operacionais	Angstrom (Padrão), Ubuntu, Android, ArchLinux, Gentoo, Minix, RISC OS, outros ...	Raspbian (Recomendado), Android, ArchLinux, FreeBSD, Fedora, RISC OS, outros ...
Consumo de energia	210-460 mA a 5V sob condições variáveis	150-350 mA a 5V sob condições variáveis
GPIO	65 Pinos	8 pinos
Periféricos	1 USB Host, um cliente Mini-USB, 1 10/100 Mbps Ethernet	2 Hosts USB, uma alimentação Micro-USB, 1 10/100 Mbps Ethernet, conector da câmera RPI

Tabela 3 – Comparativo *BeagleBone Black* e *Rapberry Pi*.

2.4.3 Macrobloco 3 - Algoritmo de Rastreamento *TLD*

O algoritmo pioneiro de *Tracking-Learning-Detection* (TLD) foi proposto por Kalal (KALAL; MIKOLAJCZYK; MATAS, 2010) e foi primeiramente disponibilizado em MATLAB e utilizado para rastrear objetos em vídeos. O algoritmo usado neste trabalho foi proposto por Georg Nebehay (NEBEHAY, 2012), posteriormente implementado em C++ e publicado sobre os termos da GNU com o nome de *OpenTLD*.

O algoritmo tem como objetivo o rastreamento de objetos desconhecidos em vídeos. O objeto de interesse é definido por uma seleção manual feita pelo usuário. O TLD, simultaneamente, rastreia o objeto, aprende a sua característica e o detecta sempre que esse aparece no vídeo.

Na Figura 24 o processo de rastreamento é iniciado quando o usuário seleciona manualmente o objeto de interesse, não sendo mais requerida a sua interferência no manuseio do *software*.

O diagrama da Figura 24 ilustra o funcionamento do algoritmo TLD. Esse diagrama é dividido em seis blocos.

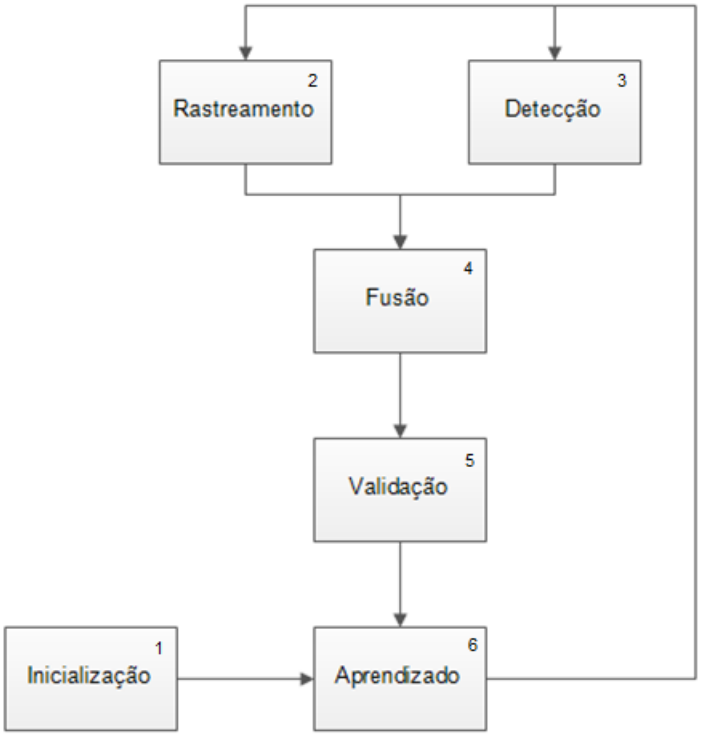

Figura 24 – Diagrama do TLD (NEBEHAY, 2012).

O bloco 1 é referente à inicialização do algoritmo, onde o usuário define o modelo inicial selecionando um contorno para o objeto a ser localizado. Essa é única etapa em que é necessária a interação do usuário com o *software*.

Os blocos 2 e 3 são diretamente responsáveis pela localização do objeto na imagem e são executados de forma paralela, sendo que os seus resultados são complementares.

O módulo de detecção, bloco 3, é predominante no algoritmo e sempre está em execução. Ele monitora toda imagem em busca do objeto de interesse, dessa forma requer um alto processamento, pois esse possui uma área de varredura do tamanho de toda a imagem. No entanto, são usados alguns artifícios para minimizar essa demanda de recursos, sendo esses uma série de testes para saber se um quadro analisado na imagem corresponde ao modelo selecionado. É importante ressaltar que o módulo de detecção somente compara o quadro em análise com modelos já existentes no banco de dados do *software*. O algoritmo da detecção é, então, dividido em quatro estágios.

O primeiro deles é uma subtração de plano de fundo com o quadro atual. Esse passo somente será validado se o tamanho do objeto formado pela subtração da imagem atual e do plano de fundo for similar ou proporcional ao tamanho do modelo inicial. Se o resultado for válido então o algoritmo leva o quadro analisado para o próximo estágio, se não ele é rejeitado e a detecção segue para o quadro seguinte. O segundo estágio é o filtro de variância que consiste em eliminar os quadros que apresentam uma baixa variância entre seus *pixels*.

O terceiro estágio usa uma técnica de classificação de possíveis resultados válidos. São selecionados alguns *pixels* aleatórios no quadro em análise e esses são comparados aos *pixels* de mesma posição das imagens presentes no banco de dados. Se metade deles apresentarem uma intensidade de brilho maior que o dessas imagens, esse quadro é, então, encaminhado para o próximo estágio, se não é descartado. O quarto e último estágio é o teste que demanda o maior processamento, pois esse faz a comparação *pixel* a *pixel* do quadro em estudo e do modelo inicial; se estes apresentarem em sua comparação um valor maior que o limiar definido nas configurações do *software*, então o resultado do módulo detector é retornado como válido, sendo esse resultado a localização do objeto procurado.

O bloco 2 refere-se ao módulo de rastreamento, esse é responsável pela localização do objeto quando há movimentação do mesmo na imagem. O seu funcionamento sempre depende de uma entrada anterior sendo proveniente ou da seleção do modelo inicial ou de um resultado válido do módulo de detecção. Ou seja, quando um resultado de detecção é validado o módulo de rastreamento é reiniciado com esse resultado.

Os blocos 4 e 5 são referentes à fusão e à validação dos resultados dos blocos 3 e 4. Esses blocos lidam com a questão de como combinar as saídas de ambos os métodos de localização dentro de um único resultado. Um método de comparação de modelos é aplicado nos resultados do detector e do rastreador. Se o primeiro apresentar um resultado com confiança maior que o segundo, então a resposta do detector é atribuída ao resultado final. Isso corresponde a uma reinicialização do rastreador. Se esse último produziu um resultado e esse não foi reinicializado pelo detector, ou é porque há mais de uma detecção ou há exatamente uma detecção que é menos confiável que o objeto rastreado. Nesse caso, o resultado do rastreador é atribuído ao resultado final. Nos outros casos, o resultado final continua vazio, o que sugere que o objeto não está visível no quadro atual.

O bloco 6 é referente ao módulo de aprendizagem do algoritmo. Esse é dividido em dois bancos de dados, chamados de banco positivo e banco negativo. O banco positivo armazena as imagens que apresentam um alto grau de confiança de ser o objeto procurado e são utilizadas como modelos de comparação para o módulo de detecção. Por sua vez, o banco negativo armazena modelos que já se mostraram ter uma boa similaridade com o modelo inicial, mas que, por comparação entre os módulos, se mostrou ser um resultado errôneo e é, então, armazenado para evitar futuros enganos. O resultado da validação, bloco 5, é incumbido de alimentar esses bancos e funciona da seguinte maneira: somente se o resultado final for validado que esse é armazenado em um dos bancos apresentados.

Esse resultado final é validado se estiver de acordo com dois requisitos: ambos assumem que o rastreador não foi reinicializado, ou seja, esse está gerando algum resultado de rastreamento. O primeiro requisito será válido e, portanto, armazenado no banco positivo se o rastreador produzir um resultado com valor de confiança maior que um limiar positivo determinado na configuração do software. Por outro lado o segundo requisito será

armazenado no banco negativo se o rastreador produzir um resultado com valor de confiança maior que um limiar negativo determinado na configuração do software. O resultado da inicialização, bloco 1, é sempre considerado como um resultado válido e é armazenado no banco positivo. Os bancos têm uma capacidade limitada de modelos e, portanto, os modelos mais antigos são constantemente atualizados para novos modelos, para evitar que haja muitas comparações e uma consequente queda de desempenho do algoritmo.

Na tese de mestrado de Georg Nebehay (NEBEHAY, 2012), é detalhado o funcionamento do algoritmo *TDL*.

3 Materiais e Métodos

Neste capítulo são apresentados o conjunto de materiais e métodos adotados no trabalho. Uma sequência de cinco experimentos foi desenvolvida, abordando aeronaves de asa fixa e rotatória para aplicação em diferentes platarformas visando a indentificação de pontencial de uso do sistema proposto. Configurações de diferentes soluções tecnológicas e metodológicas foram aplicadas, com variação no sistema de captura e identificação/rastreamento.

Para a identificação do potêncial de uso, decidiu-se por realizar quatro experimentos com diferentes plataformas aéreas aumentando-se o grau de complexidade. No primeiro experimento, é usado um quadrimotor, para realizar um voo em um campo aberto utilizando apenas um carro e mantendo a câmera estática. No segundo experimento, utilizou-se um hexamotor que matenteve-se fixo em uma rodovia com intuito de analisar o comportamento do sistema com a exposição de diferentes modelos de carros. Para o terceiro experimento, foi utilizado um octamotor com um estabilizador de imagem (*Gimbal*) em um local de testes de aeromodelos. O aparelho foi usado de forma dinâmica, ou seja, o aparelho teve sua posição alterada durante o experimento e foi utilizado somente um carro para o rastreamento, com a diferença de que é forçada uma perda de captura do objeto para testar o sistema. Para o quarto experimento, foi usada uma plataforma de asa fixa em uma rodovia com o objetivo de testar o rastreamento em carros seguindo na velocidade da via tendo a exposição de diferentes modelos de carros na filmagem.

A aplicação do algoritmo TLD é baseado em um arquivo de configuração que é dividido em três partes: aquisição, detecção e rastreamento (Tabelas 10, 11 e 12). Essas configurações e tabelas estão disponíveis no anexo B.

3.0.4 Análise de Desempenho do *OpenTLD* em Diferentes *Hardwares* de Processamento

Na Tabela 4, são apresentados os materiais utilizados para a realização do experimento de comparação de desempenho entre três diferentes *hardwares*.

Neste experimento, pretende-se avaliar se é possivel utilizar uma plataforma embarcada para realizar o processamento do *software OpenTLD* em tempo real. Para embarcar o *software OpenTLD* nas plataformas, primeiro gerou-se a imagem da distribuição. Após gerar a imagem é necessário grava-lá em um cartão de memória, o qual será lido pela plataforma. Com o cartão de memória conectado na placa, liga-se e efetua-se a inicialização do sistema operacional. Com o sistema carregado é necessário baixar as bibliotecas do *OpenCV* apresentados no anexo B.

Macrobloco 1	
Controladora	YS-X4
Motor	4112-320KV *Turnigy Multistar* 22 *Pole Brushless Multi-Rotor Motor*
Estrutura (*Frame*)	5mm 3K *Carbon* 1050 *Wheelbase Octa-Copter Frame Set (Star Shape)*
Macrobloco 2	
Camêra	Panasonic DMC-GH2 com estabilizador de imagem *Gimbal Cinestar Clone*
Modo de aquisição	*Offline*
Placa de aquisição	Não usado
Hardware de processamento	1. *Raspberry Pi* com sistema operacional *Linux raspbian kernel 3.6* (Figura 25); 2. *BeagleBone Black* com sistema operacional *Linux ubuntu kernel 3.8* (Figura 25); 3. *Notebook* com processador *Core 2 Duo* P7550 e memória RAM 4GB DDR2 com sistema operacional *Linux Mint kernel 3.8*;
Macrobloco 3	
Algoritmo	*OpenTLD*
Biblioteca	*OpenCV*

Tabela 4 – Tabela dos materiais para a realização do experimento de análise de desempenho entre os dispositivos de processamento.

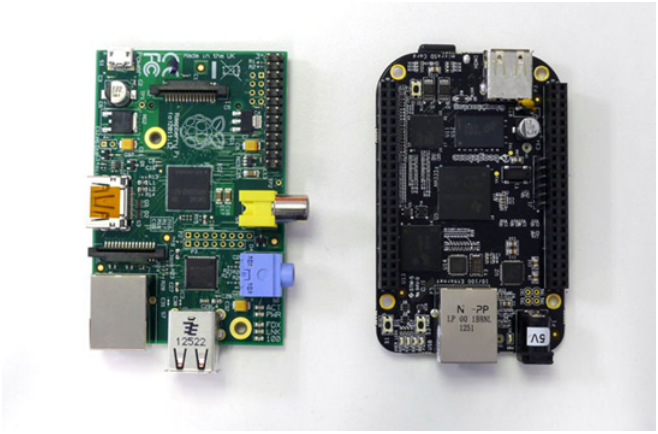

Figura 25 – *Rapberry Pi* (esquerda) e *BeagleBone Black* (Direita) (SMITH, 2011).

Macrobloco 1	
Controladora	Multiwii and Megapirate AIO Flight Controller (ATmega 2560)
Motor	A2212-15 Brushless Outrunner 930KV
Estrutura (*Frame*)	Estrutura de um quadrimotor feita em impressora 3D com material ABS tendo algumas partes confeccionadas com madeira balsa (Figura 26)
Macrobloco 2	
Camêra	Câmera SONY CCD 1/3inch
Modo de aquisição	*Online*
Placa de aquisição	Usb 2.0 Easycap Dc60+ 27
Hardware de processamento	*Notebook* com processador *Core 2 Duo* P7550 e memória RAM 4GB DDR2 com sistema operacional *Linux Mint kernel 3.8*;
Macrobloco 3	
Algoritmo	*OpenTLD*
Biblioteca	*OpenCV*

Tabela 5 – Tabela dos materiais para a realização do primeiro experimento com ARC.

Após a instalação do *software*, foram feitos testes nas seguintes plataformas: *Raspberry Pi*, *BeagleBone* e *Notebook*. Para esse teste foi usado um mesmo arquivo de vídeo gerado no teste do octamotor (Subseçao 3.0.7).

Para comparação de desempenho entre as plataformas, gerou-se um arquivo contendo os valores de FPS (Quadros Por Segundo) de cada uma das plataformas, para posterior confecção de um gráfico. Foi realizado um teste executando o vídeo no *software OpenTLD*, selecionando um modelo incial, para conhecer os reais valores de FPS quando o *software* estiver rastreando.

3.0.5 Primeiro Experimento com Plataforma Aérea

A Tabela 5 apresenta os materiais utilizados no primeiro experimento de análise do comportamento do *software OpenTLD* integrado com um ARC do tipo quadrimotor.

No primeiro experimento escolheu-se um campo aberto. O campo utilizado no experimento é mostrado na Figura 28.

A realização do experimento se deu ligando a placa de captura ao computador, depois o quadrimotor foi mantido em uma posição fixa no ar sempre focando o automóvel. O método de aquisição do *software* foi alterado para ser usado como uma entrada em tempo real como é explicado no Anexo B. Iniciado o *software OpenTLD*, bastou selecionar o modelo do carro a ser localizado(Figura 29).

O modelo escolhido foi uma imagem do carro na lateral. A Figura 29 mostra a

Figura 26 – Quadrimotor com a câmera e o receptor de vídeo ligado a um *display* de LCD.

Figura 27 – Placa de captura de video *EasyCap*.

Figura 28 – Campo aberto.

Figura 29 – Seleção de objeto inicial feita pelo usuário.

Macrobloco 1	
Controladora	NAZA *Flight Controller*
Motor	Turnigy 2217 20turn 860KV 22A Outrunner
Estrutura (*Frame*)	Estrutura de um hexamotor Turnigy Talon de fibra de carbono- 625mm (Figura 30)
Macrobloco 2	
Camêra	Câmera Gopro silver hero 3 (Figura 31)
Modo de aquisição	*Offline*
Placa de aquisição	Não usado
Hardware de processamento	*Notebook* com processador *Core 2 Duo* P7550 e memória RAM 4GB DDR2 com sistema operacional *Linux Mint kernel 3.8*;
Macrobloco 3	
Algoritmo	*OpenTLD*
Biblioteca	*OpenCV*

Tabela 6 – Tabela dos materiais para a realização do segundo experimento com ARC.

primeira detecção após escolher o modelo.

Nesse experimento visa-se testar o reconhecimento de um modelo de carro sem interferência de outros modelos de carros nas imagens e também avaliar o comportamento do algoritmo com *link* de vídeo em tempo real, ou seja aplicar o *software* de rastreamento no instante em que a imagem é capturada, considerando também as interferências de sinal de transmissão.

3.0.6 Segundo Experimento com Plataforma Aérea

A Tabela 6 apresenta os materiais utilizados no segundo experimento de análise do comportamento do *software OpenTLD* integrado com um ARC do tipo hexamotor.

Figura 30 – Hexamotor com a câmera em cima do aparelho.

Figura 31 – Câmera *Gopro silver hero* 3.

Nesse segundo experimento foi escolhida uma área ao lado de uma rodovia por onde trafegavam diversos automóveis (Figura 32), sendo então possível avaliar o comportamento do sistema com vários carros.

Na Figura 32, a câmera do aparelho foi instalada acima dos motores e, portanto dois deles irão necessariamente aparecer na imagem formada. Com esse experimento, objetiva-se avaliar o comportamento do algoritmo quando houver vários modelos de carros disponíveis na imagem.

Diferente do primeiro experimento, este não foi feito em tempo real então o método de aquisição do *software* foi alterado para tal como é explicado no Anexo B. Primeiramente, foi realizada a gravação do teste e depois o vídeo foi usado para fazer o rastreamento. Com o vídeo disponível, foram selecionados três modelos iniciais (Figura 33).

Figura 32 – Campo ao lado de uma rodovia.

Figura 33 – Modelos iniciais selecionados para o segundo experimento.

Figura 34 – Aparelho octocóptero com *gimbal*.

3.0.7 Terceiro Experimento com Plataforma Aérea

A Tabela 7 apresenta os materiais utilizados no terceiro experimento de análise do comportamento do *software OpenTLD* integrado com um ARC do tipo octamotor.

Neste experimento, o teste foi feito em um campo destinado a aeromodelismo,

Macrobloco 1	
Controladora	YS-X4
Motor	4112-320KV *Turnigy Multistar* 22 *Pole Brushless Multi-Rotor Motor*
Estrutura (*Frame*)	5mm 3K *Carbon* 1050 *Wheelbase Octa-Copter Frame Set (Star Shape)* (Figura 34)
Macrobloco 2	
Camêra	Panasonic DMC-GH2 com estabilizador de imagem *Gimbal Cinestar Clone* (Figura 35)
Modo de aquisição	*Offline*
Placa de aquisição	Não usada
Hardware de processamento	*Notebook* com processador *Core 2 Duo* P7550 e memória RAM 4GB DDR2 com sistema operacional *Linux Mint kernel 3.8*;
Macrobloco 3	
Algoritmo	*OpenTLD*
Biblioteca	*OpenCV*

Tabela 7 – Tabela dos materiais para a realização do terceiro experimento com ARC.

Figura 35 – Câmera Panasonic DMC-GH2.

conforme mostrado na Figura 36. Com esse teste, pretende-se avaliar como o algoritmo se comporta caso a plataforma voadora esteja em movimento, tendo a visão superior de um carro e algumas obstruções do objeto, como postes de iluminação e também a perda do objeto na imagem e a recuperação do mesmo.

Assim como no segundo experimento, esse teste foi feito utilizando o vídeo gravado pela câmera de forma *offline*, ou seja, não foi feito em tempo real. Com o vídeo disponível, foi selecionado o modelo inicial.

3.0.8 Quarto Experimento com Plataforma Aérea

A Tabela 8 apresenta os materiais utilizados no quarto experimento de análise do comportamento do *software OpenTLD* integrado com um ARC do tipo asa fixa.

Neste experimento, o teste foi feito com uma asa voadora em uma rodovia. Com esse teste, pretende-se avaliar como o algoritmo se comporta se a plataforma voadora

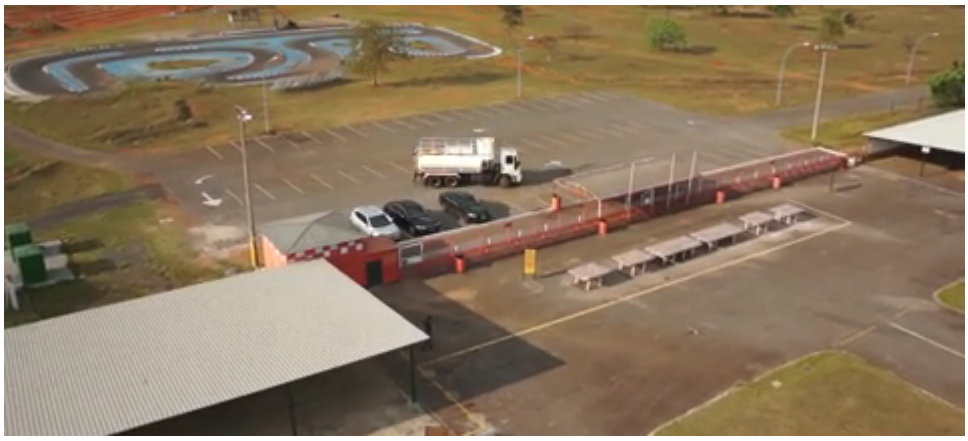

Figura 36 – Campo de Aeromodelismo.

Figura 37 – Seleção modelo inicial.

Macrobloco 1	
Controladora	*Ardupilot*
Motor	1530 *Brushless Inrunner Motor* 3800KV
Estrutura (*Frame*)	Asa fixa tipo Zagi (Figura 38)
Macrobloco 2	
Camêra	Câmera SONY CCD 1/3inch
Modo de aquisição	*Offline*
Placa de aquisição	
Hardware de processamento	*Notebook* com processador *Core 2 Duo* P7550 e memória RAM 4GB DDR2 com sistema operacional *Linux Mint kernel 3.8*;
Macrobloco 3	
Algoritmo	*OpenTLD*
Biblioteca	*OpenCV*

Tabela 8 – Tabela dos materiais para a realização do quarto experimento com ARC.

Figura 38 – Aparelho Asa Fixa.

Figura 39 – Seleção modelo inicial para o experimento 4.

estiver em movimento com uma velocidade próxima à da via, tendo a visão superior de um carro e também a perda do objeto na imagem e a recuperação do mesmo.

Esse teste foi feito utilizando o vídeo gravado pela câmera de forma *offline*, ou seja, não foi feito em tempo real. Com o vídeo disponível, foi selecionado o modelo inicial.

4 Resultados e Discussão

Nos resultados foram apresentados conjuntos de imagens que são fragmentos dos vídeos capturados pelos respectivos aparelhos (quadrimotor, hexamotor, octamotor e asa fixa) segundo o seguinte critério: momentos de transição do reconhecimento para o não reconhecimento do objeto selecionado e vice-versa e transição do reconhecimento ou não reconhecimento para o aprendizado e vice-versa.

4.1 Análise de Desempenho do *OpenTLD* em Diferentes *Hardwares* de Processamento

Neste experimento, o principal objetivo foi avaliar a performance do *software OpenTLD* nas plataformas: *Raspberry Pi*, *Beaglebone* e *Notebook*.

No gráfico da Figura 40 são mostrados os valores de FPS do *software* quando selecionado o modelo inicial, isto é, executando o rastreamento. Inicialmente todas as três plataformas têm suas médias com valores mais altos , pois o modelo inicial ainda não foi selecionado. Próximo ao quadro 100, pode ser percebida uma queda de desempenho nos três dispositivos, pois nesse instante, foi selecionado o modelo inicial e, portanto, foi aplicado o algoritmo de rastreamento. O *Raspberry* apresentou uma média de 2 FPS, o *Beaglebone*, uma média de 10 FPS e por fim o *notebook* executou a uma média 45 FPS. De acordo com (COMMITTEE, 1953) o número mínimo de FPS requerido para o padrão NTSC é 25. Nesse teste, somente o *notebook* teve sua média superior ao padrão estabelecido e, portanto, das três plataformas é o único que apresentou desempenho suficiente

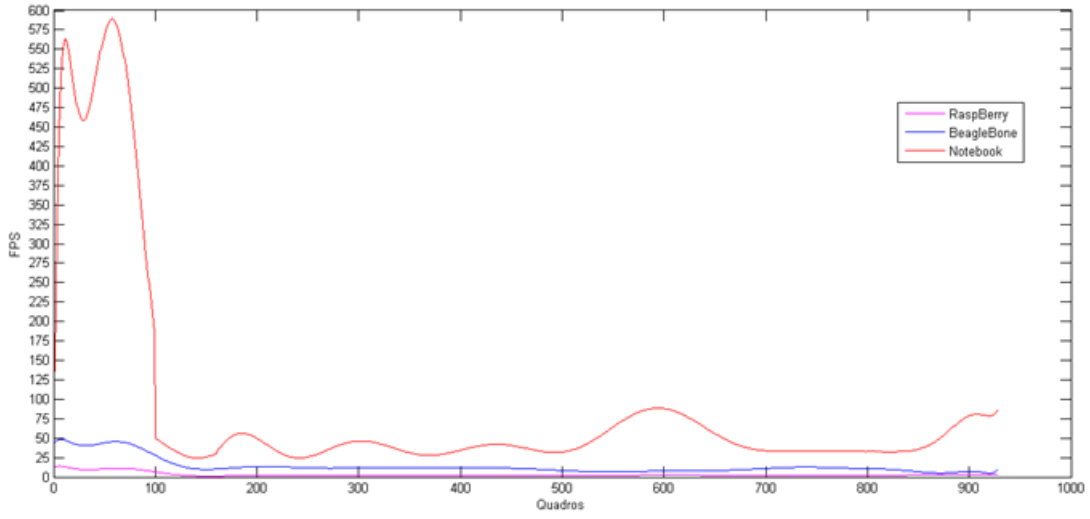

Figura 40 – Gráfico de desempenho das três plataformas com rastreamento.

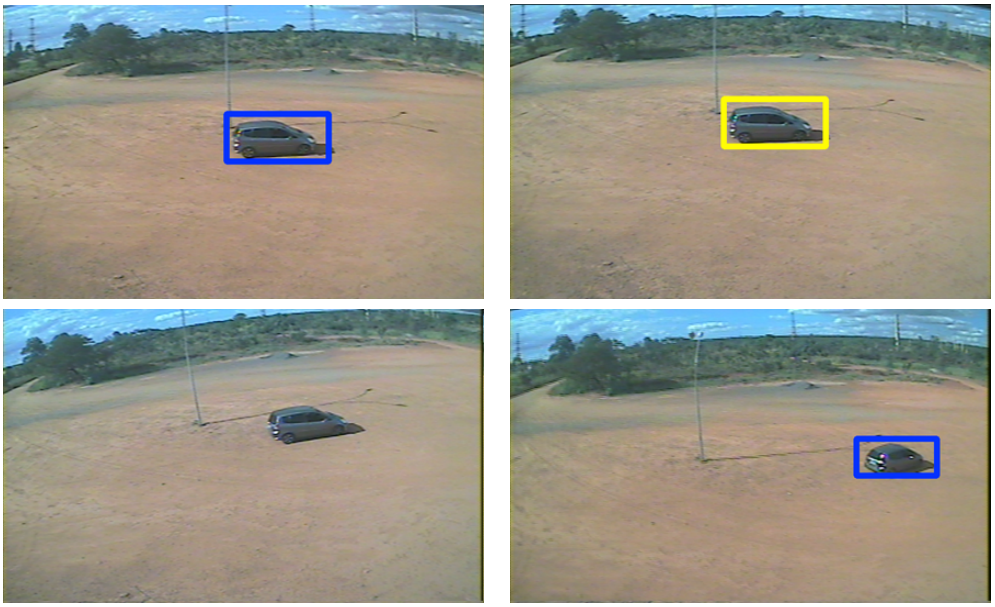

Figura 41 – Resultados do primeiro experimento.

para executar o *software OpenTLD* de forma a obter resultados em tempo real.

Um possível motivo para que as plataformas embarcadas não tenham apresentado resultados aceitáveis é relacionado à otimização do código com os *hardwares* de tais dispositivos. Analisando o código do *software*, foi possível constatar que esse somente usa o processador principal do sistema (CPU) e, possivelmente, se o *software* for otimizado para efetuar os cálculos no processador de vídeo (GPU), esse poderia apresentar melhores desempenhos em todas as plataformas apresentadas.

4.2 Primeiro Experimento com Plataforma Aérea

O experimento a seguir foi realizado utilizando um quadrimotor. A Figura 41 apresenta o comportamento do sistema durante o primeiro experimento. Após selecionar o modelo, o automóvel começa a se deslocar: nesse primeiro momento, o algoritmo começa a aprender o modelo que foi selecionado pelo usuário. Quando a caixa de seleção é azul, significa que o algoritmo mantém o rastreamento do objeto; quando está em amarelo, significa que está ocorrendo uma aprendizagem e, portanto, uma nova imagem está sendo adicionada ao banco de modelos válidos.

Neste teste, o trajeto do automóvel foi realizar cinco voltas em torno de um poste (Figura 41). Sendo assim, após a seleção de um modelo inicial do objeto, foi possível para o algoritmo ter acesso a vários ângulos de visão do automóvel. Os módulos de rastreamento, detecção e aprendizagem são os módulos apresentados na Subseção 2.4.3.

Na primeira volta, o algoritmo somente localizou o automóvel na visão lateral,

Figura 42 – Ruído na recepção do sinal de vídeo.

sendo essa a mesma visão selecionada no modelo inicial. Após ter seu ângulo de visão alterado, não foi possível fazer a localização do mesmo. Isso ocorreu pois não foi possível para algoritmo atualizar o banco de dados durante a mudança do ângulo de visão, ou seja, somente essa volta não foi suficiente para o módulo rastreador gerar resultados com confiança para a ocorrência do aprendizado. Com aumento de exposição do automóvel durante algumas voltas foi possível ao módulo rastreador gerar resultados com ângulos diferentes de forma a possibilitar o aprendizado desses novos ângulos.

A detecção também foi melhorando com passar do tempo, pois com novos modelos armazenados no banco pelo módulo de aprendizagem, o objeto pode ser localizado na imagem com uma maior frequência, tornando a localização do automóvel mais robusta ao longo do trajeto.

Com o uso do modo de aquisição da imagem em tempo real, foi possível avaliar algumas possíveis dificuldades como ruído de transmissão e qualidade na captação da imagem. As dificuldades encontradas durante a realização do experimento correspondem ao ruído na imagem Figura 42. Outra limitação encontrada foi na transmissão/recebimento do sinal (Figura 43), pois interferências no sinal promoviam imagens duplicadas, gerando imagens com "fantasma" e pouca nitidez. Por consequência, ocorre a perda da localização do objeto na imagem, como é visto na Figura 43. Pouco tempo depois do início do teste, percebeu-se que esse problema ocorria devido ao aumento da força do vento e também quando ocorriam vibrações nos cabos que ligavam o receptor na placa de captura. Para resolver esse problema, fixaram-se as partes móveis do receptor e os cabos de ligação.

Uma terceira dificuldade possivelmente se encontra na decodificação da imagem para a formação da mesma, pois ocorre um "escorregamento" da imagem quando, ao formar a imagem entrelaçando as linhas pelas colunas, ocorre um erro que desloca as linhas ou as colunas, causando esse efeito de "escorregamento" na imagem, como são

Figura 43 – Problemas na recepção do sinal de vídeo.

Figura 44 – Imagem apresenta efeito de “escorregamento”.

vistas na Figura 44 linhas horizontais bem evidenciadas.

A quarta dificuldade encontrada foi um erro na quantização. Ao transformar o sinal analógico para digital, realizado pela placa de captura, tal efeito foi percebido, porque, ao se comparar a imagem analógica que é mostrada no *display* de LCD com a imagem no computador, na primeira não aparecia esse efeito de linhas pretas na horizontal no carro. A Figura 45 mostra a imagem com erros de quantização.

Mesmo com as dificuldades apresentadas e a não estabilização da câmera, o sistema conseguiu detectar, rastrear e aprender com o modelo pré-determinado, desempenhando a função de rastreio em 62% do tempo de realização do experimento.

Figura 45 – Imagem apresenta problemas na quantização digital da imagem.

4.3 Segundo Experimento com Plataforma Aérea

Neste experimento foi utilizado um hexamotor e foram selecionados três modelos, em cada um deles, foi definido uma seleção de carro diferente para realizar o rastreamento/detecção.

Após selecionar o primeiro modelo, o algoritmo iniciou a detecção do objeto (Figura 46), o qual foi rastreado com sucesso durante certo momento, pois somente o carro selecionado foi localizado dentre inúmeros outros. Quando a imagem se aproximou do motor do equipamento, houve um aprendizado equivocado por parte do algoritmo na sobreposição do motor com o carro. No último quadro, pode-se observar que o automóvel inicialmente selecionado se deslocou, mas o sistema manteve-se focado em um dos motores.

Tal fato ocorreu pois o módulo rastreador do *OpenTLD* foi aprendendo um modelo errôneo à medida que o automóvel selecionado era sobreposto aos poucos pelo motor. Dessa forma, as imagens da sobreposição do motor e do carro foram sendo adicionadas ao banco de alta confiança. Assim a imagem do motor passou a ser parte do modelo usado pelo algoritmo. Como o motor estará sempre disponível na imagem, pois este é fixo em relação à camera, então somente o módulo de rastreamento terá efeito na localização do objeto.

Neste teste, a localização do objeto foi afetada pela constante aparição do motor na imagem, pois quando a imagem do motor se sobrepôs à imagem do objeto em constante localização, o módulo de rastreamento falhou na consideração dessa sobreposição como um resultado válido. O algoritmo ainda pode se recuperar dessa falha, se o detector reinciar o rastreamento: isso pode acontecer se o modelo procurado puder ser enquadrado novamente na imagem por um período suficiente para o módulo detector o localizar.

Figura 46 – Resultados do primeiro modelo selecionado.

Figura 47 – Resultados do segundo modelo selecionado.

O segundo modelo detectado Figura 47 apresentou melhores resultados se comparado ao primeiro modelo, pois mesmo esse sofrendo uma pequena obstrução do braço do hexamotor, o algoritmo pôde continuar a rastrear a maior parte do objeto durante toda a passagem do carro sem que houvesse a detecção de outros carros presentes nas imagens.

Isso se deve ao fato de que, nesse segundo teste, o carro passou pela região do braço do aparelho. Essa não é suficiente para obstruir o automóvel rastreado e, portanto, mesmo que por um momento seja reconhecida como parte do objeto, ainda é possível para o algoritmo usar o módulo de detecção para localizar o carro quando esse está se distanciando do braço. Dessa forma, o rastreador é reiniciado e tem como nova entrada a imagem localizada pelo detector.

Neste experimento, a localização do automóvel não foi influenciada pela constante aparição do motor na imagem devido à sobreposição do braço do motor com o carro, pois o resultado da sobreposição não gerou um valor de confiança suficiente para que ocorresse o seu aprendizado.

Figura 48 – Resultados do terceiro modelo selecionado.

Na terceira seleção Figura 48, assim como no primeiro, pôde-se observar a ocorrência da detecção equivocada do motor do hexamotor ao invés do automóvel. Tal fato ocorreu pelo mesmo motivo já apresentado no primeiro modelo. A câmera acima do aparelho, tendo a obstrução dos motores, mostrou que não é o posicionamento ideal para o correto funcionamento do algoritmo, pois dois dos três modelos selecionados apresentaram falhas em seus rastreamentos.

4.4 Terceiro Experimento com Plataforma Aérea

Neste teste utilizou-se um octamotor, com um estabilizador de imagem, em movimento, para adquirir as imagens de um carro também em movimento (Figura 49). O modelo inicial foi selecionado com a traseira do carro vista de cima, tendo o ângulo de visão alterado posteriormente de forma a circular sobre as laterais do objeto, para aumentar a confiabilidade do modelo armazenado.

Na segunda imagem, é possível perceber que o sistema perde o objeto e somente consegue reconhecê-lo novamente quando o objeto volta a ser observado pela traseira. Em um segundo momento, quinta figura, a detecção do carro não sofreu influência de um poste, que obstruiu a imagem daquele. Em uma terceira fase dessa análise, o carro foi propositalmente desenquadrado da imagem e logo após enquadrado novamente, e esse foi encontrado pelo algoritmo com sucesso.

O módulo de rastreamento se mostrou eficaz nesse teste, pois quando incializado com o modelo do automóvel localizou este enquando permaneceu enquadrado na imagem. E mesmo quando o automóvel foi sobreposto por um poste (terceira imagem da primeira coluna da Figura 49), o módulo de rastreamento não gerou resultados sufcientemente válidos para que ocorresse a aprendizagem. O módulo de detecção também mostrou-se eficaz, pois esse foi capaz de reiniciar o módulo de rastreamento com o modelo correto

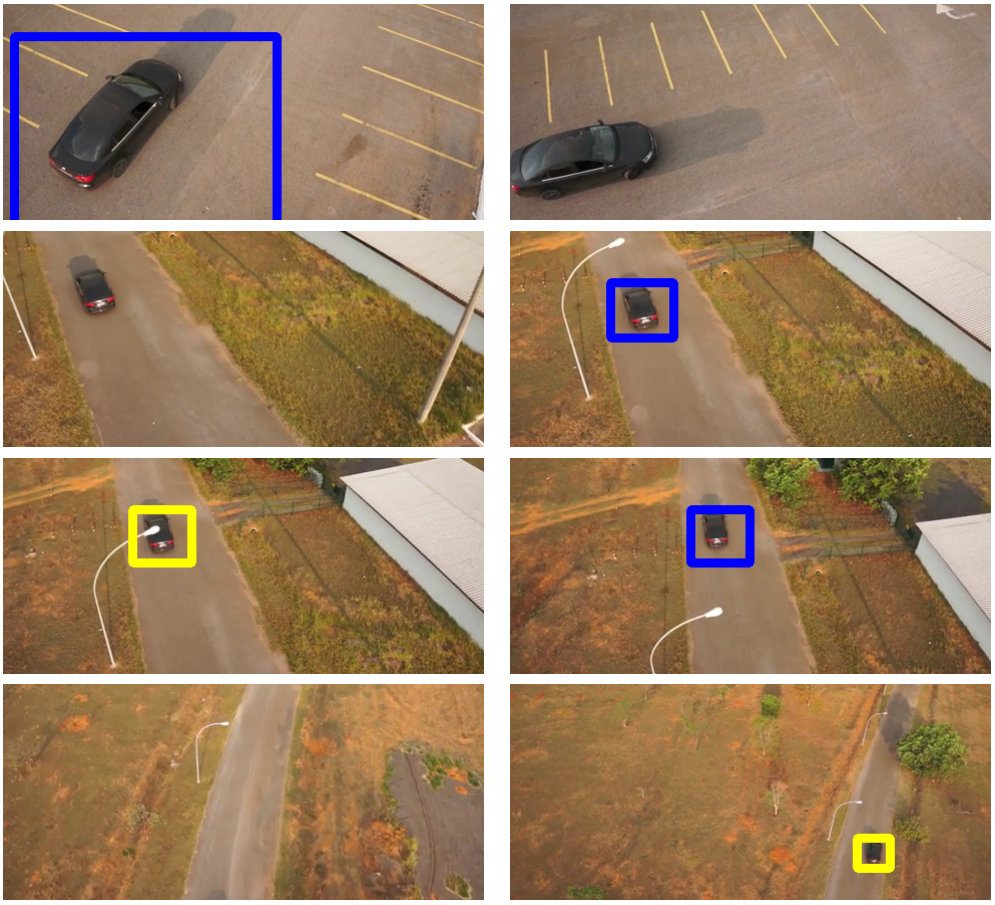

Figura 49 – Resultados do terceiro experimento.

quando o automóvel foi desenquadrado da imagem, retornando posteriormente.

Então, para apenas um automóvel, o algoritmo de detecção se mostrou eficiente na recuperação do modelo previamente selecionado, apresentando resultados válidos para os módulos de rastreamento e detecção. Utilizando equipamentos profissionais para aquisição da imagem, estabilização e o próprio equipamento para o voo, possibilitou a obtenção de imagens com melhor qualidade de definição e de estabilização. Tendo esses aparatos à disposição, foi possível testar a detecção de um carro com o aparelho voador em movimento. A detecção foi bem sucedida enquanto carro se manteve enquadrado pela câmera, de forma que pode-se avaliar o comportamento do algoritmo com mudança de plano de fundo. Ainda foi possível avaliar o comportamento do *software* com o desenquadramento do objeto rastreado, e posterior enquadramento, tendo esse se comportado de forma positiva, pois o automóvel foi detectado com sucesso quando era novamente enquadrado.

4.5 Quarto Experimento com Plataforma Aérea

Diferente de todos os testes anteriormente realizados, neste experimento foi utilizada uma asa fixa, proporcionando uma análise para acompanhamento de automóveis em

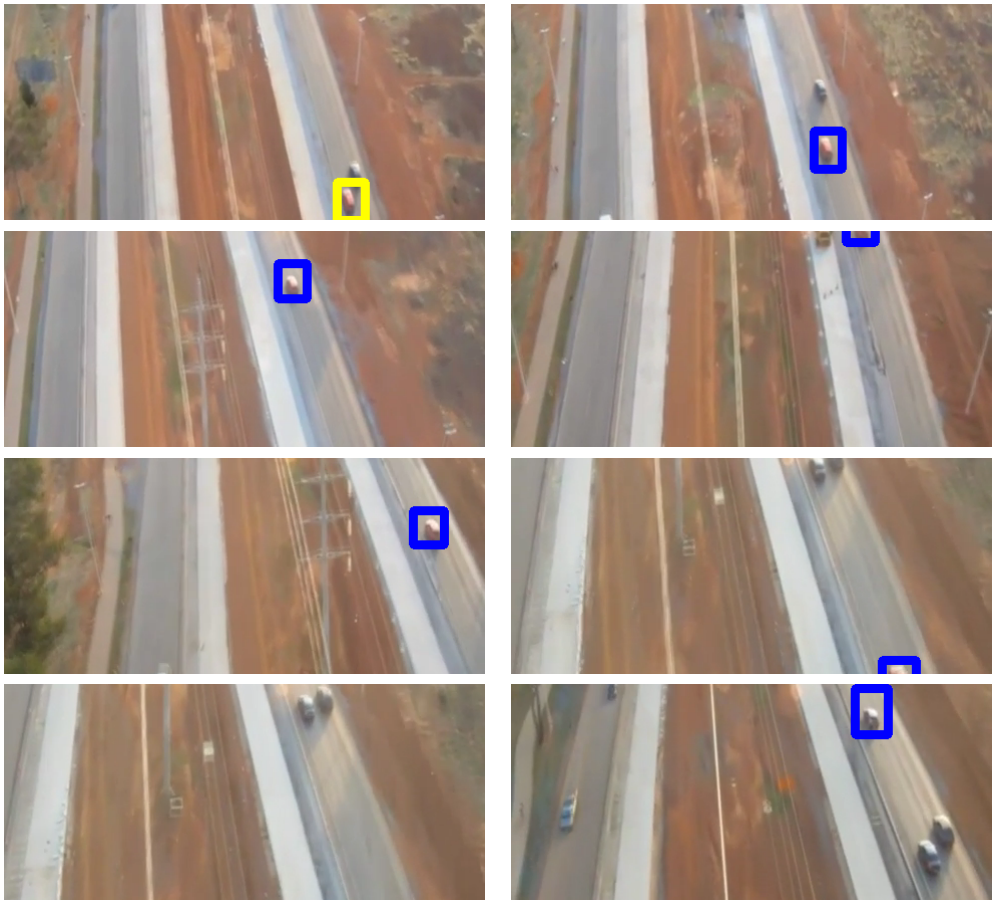

Figura 50 – Resultados do quarto experimento.

maiores velocidades e longas distâncias. Nas primeiras imagens da Figura 50, pode ser observado que o modelo selecionado é localizado com facilidade, mesmo havendo vários veículos próximos a este. Isso mostra que o módulo de rastreamento do *OpenTLD* funcionou de forma eficiente, pois não houve a perda do objeto selecionado enquanto esse se manteve enquadrado pela câmera.

Por outro lado, o módulo de detecção não se mostrou confiável neste teste, pois quando o veículo foi desenquadrado nas duas últimas imagens, o algoritmo de detecção reconheceu outro carro presente na rodovia como sendo o modelo previamente escolhido.

Isso se deve ao fato de a imagem que foi selecionada como modelo possuir uma deficiência de detalhes que o caracterizaria de forma única dentre os outros carros. Sendo assim, para o módulo de detecção, todos esses carros são candidatos a serem possíveis modelos do objeto procurado e, portanto, algum será considerado pelo detector como possível resultado. O grande problema apresentado neste teste foi a qualidade da imagem, pois não foi possível para módulo detector distinguir dentre os possíveis automóveis o que foi previamente selecionado.

Com esse experimento observou-se o comportamento do *software OpenTLD*, tendo

uma movimentação do plano de fundo a uma maior velocidade. Para atingir a velocidade da via foi utilizada uma asa fixa, pois essa tem a capacidade de atingir velocidades próximas dos automóveis. Um segundo ponto analisado foi a presença de diferentes automóveis na rodovia. Com isso pode-se avaliar a capacidade de diferenciação entre o modelo selecionado e outros veículos presentes na imagem. O algoritmo apresentou um ótimo desempenho no rastreamento do modelo selecionado enquanto esse se manteve enquadrado na imagem, porém ao desenquadrar o objeto, o algoritmo apresentou falhas na detecção, pois rastreou outro automóvel na rodovia. Esse problema ocorreu por conta da baixa fidelidade do modelo devido à altura que dificultou o melhor detalhamento da imagem capturada. Portanto, é necessária uma maior proximidade com o objeto desejado, para que o nível de detalhes seja o mínimo necessário para o rastreamento.

5 Conclusão e Trabalhos Futuros

5.1 Conclusão

O presente trabalho remeteu-se à aplicação do algoritmo de rastreamento em plataformas voadoras não tripuladas. Também foram levantados os componentes que compõem o sistema proposto visando à integração deste ao *software OpenTLD*. Neste trabalho utilizou-se veículos automotivos como objetos de pesquisa, pois esses possuem mobilidade paras os testes estáticos e dinâmicos.

Inicialmente, cogitou-se a possibilidade de se utilizar uma plataforma embarcada para o processamento de imagens do sistema, porém, após a realização de testes do *software* com duas plataformas populares, não se obtiveram resultados que atendessem o requisito mínimo estipulado pelo padrão NTSC para uma imediata implementação do algoritmo com esses dispositivos. Porém, computadores portáteis com configurações semelhantes ao usado neste trabalho podem ser imediatamente usados como ambiente de processamento para o *software OpenTLD* de forma a gerar resultados em tempo real.

Foram realizados experimentos variando a plataforma ARC e a forma de aquisição de imagens. A partir dos experimentos realizados com os multimotores, foi possível avaliar a aplicação do algoritmo nessas diferentes plataformas. Foi possível avaliar o comportamento do sistema operando com captura de imagens em tempo real, onde as maiores dificuldades encontradas foram no sistema de transmissão de vídeo e no dispositivo de captura utilizado no computador. Ainda foi possível avaliar o comportamento do algoritmo com diferentes modelos de automóveis.

Com este trabalho foi possível concluir que o *software* ainda precisa de ajustes para melhor se adequar ao sistema proposto, como enviar somente o resultado da caixa de seleção do objeto localizado em vez da imagem completa o que ocasionaria um aumento de desempenho do processamento do *software*.

Para detecção de somente um automóvel, o sistema já se apresenta confiável, tanto em cenários em que o aparelho voador esteja estático, quanto dinâmico. O módulo de rastreamento se mostrou bem eficiente na localização dos veículos procurados enquanto esses não eram obstruidos ou desanquadrados, no entanto o módulo de detecção mostrou dificuldades na localização em alguns cenários onde o detalhamento do objeto não era suficiente. Pode-se concluir também que, se o objeto não for desenquadrado, esse será localizado sem perdas do rastreamento, podendo ser usado em casos de perseguição, escolta ou monitoramento de determinados veículos.

Ainda foi possível confeccionar uma apostila de configuração do *software OpenTLD*

em português, que consta em anexo, para facilitar a reprodução dos testes feitos neste trabalho e também outros testes que possam auxiliar no uso desse *software* em outros trabalhos.

5.2 Trabalhos Futuros

Para trabalhos futuros, ainda é preciso determinar o nível mínimo de detalhes que a imagem capturada deve possuir, para que, se o modelo selecionado for desenquadrado e enquadrado novamente dentre inúmeros outros objetos semelhantes, esse ainda possa ser detectado sem que haja equívocos no rastreamento de falsos objetos. Além disso, pretende-se aperfeiçoar o *software OpenTLD*, para executar em plataformas embarcadas de forma eficiente e que possa ser usada em tempo real, atendendo às necessidades de rastreamento.

Referências

ABOUTAUTO. *Auto Electrical Services*. 2013. Disponível em: <http://www.aboutauto.com.au/?page_id=68>. Citado 2 vezes nas páginas 9 e 28.

AERIALS. *Perspectives Aerials*. 2013. Acesso em 10 de julho. Disponível em: <http://www.perspectiveaerials.com/>. Citado na página 17.

AFAUNANATAL. *FiscalizacaoPosturasGuaruja*. 2013. Acesso em 06 de julho de 2013. Disponível em: <http://afaunanatal.files.wordpress.com/2011/12/post.gif>. Citado 2 vezes nas páginas 9 e 18.

BüCHI, R. *Quadcopter*. 2013. Acesso em 28 de julho. Disponível em: <http://en.wikipedia.org/wiki/Quadcopter>. Citado 2 vezes nas páginas 9 e 25.

COMMITTEE, N. T. S. *Some supplementary references cited in the Reports, and the Petition for adoption of transmission standards for color television before the Federal Communications Commission*. 1953. Citado na página 49.

DIPOL. *TV and SAT TV, CCTV, WLAN*. 2013. Acesso em 06 de julho. Disponível em: <http://www.dipol.ie/newsletter2/inf_dipo_2011_10.html>. Citado 2 vezes nas páginas 9 e 18.

ELETRICA. *Ênfase em Engenharia de Sistemas Eletrônicos Embarcados*. 2013. Disponível em: <http://www.eletrica.ufpr.br/graduacao/noturno/embarcados.html>. Citado na página 27.

FEDERAL, G. D. D. *Instrumentação de Aeronaves não Tripuladas para o Gerenciamento de Queimadas no Cerrado*. 2009. Citado 3 vezes nas páginas 9, 21 e 29.

FURTADO, V. H. *Aspectos de Segurança na Integração de Veículos Aéreos Não Tripulados (vant) no Espaço Aéreo Brasileiro*. 2008. 506-517 p. SITRAER. Citado 2 vezes nas páginas 21 e 22.

HUGO, M. F. O. V. N. *Processamento Digital de Imagens*. Rio de Janeiro: [s.n.], 1999. Citado 3 vezes nas páginas 9, 28 e 29.

J., D. R.; LIN, L. R.-S.; YANG, M.-H. Incremental learning for robust visual tracking. In: . [S.l.: s.n.], 2008. International Journal of Computer Vision. Citado na página 33.

KALAL, Z.; MIKOLAJCZYK, K.; MATAS, J. Forward-backward error: Automatic detection of tracking failures. In: . [S.l.: s.n.], 2010. In International Conference on Pattern Recognition. Citado na página 35.

LEONARD, M. *Raspberry Pi or BeagleBone*. 2013. Disponível em: <http://www.michaelhleonard.com/raspberry-pi-or-beaglebone-black/>. Citado na página 34.

MELO, A. S. de. *Implementação de um quadrotor como plataforma de desenvolvimento para algoritmos de controle*. 2010. Citado 2 vezes nas páginas 9 e 31.

MURTAGH, F. *A survey of recent advances in hierarchical clustering algorithms.* 1983. The Computer Journal. Citado 2 vezes nas páginas 9 e 24.

NEBEHAY, G. Robust object tracking based on tracking-learning-detection. In: . [S.l.: s.n.], 2012. Citado 4 vezes nas páginas 9, 35, 36 e 38.

OLIVEIRA, F. A. Veículos aéreos não tripulados. In: . [S.l.: s.n.], 2005. Seminários Temáticos para a 3ª Conferência Nacional de C,T&I. Citado 3 vezes nas páginas 11, 21 e 22.

OZUYSAL, M.; FUA, P.; LEPETIT, V. Fast keypoint recognition in ten lines of code. In: . CA, USA: [s.n.], 2007. In IEEE Conference on Computer Vision and Pattern Recognition, Los Alamitos. Citado 2 vezes nas páginas 9 e 26.

PEREIRA, R. T. *A Física do Cotidiano - Porque um avião voa?* 2013. Acesso em 28 de julho. Disponível em: <http://curtindoaciencia.blogspot.com.br/2011/09/fisica-do-cotidiano-porque-um-aviao-voa.html>. Citado 2 vezes nas páginas 9 e 25.

RASI, J. R. *Desenvolvimento de um Veículo Aéreo Não Tripulado para Aplicação em Pulverização Agrícola.* 2008. RS. Citado na página 22.

RODOLFO, W. R.; CEZAR. *Desenvolvimento de um ambiente de execução de aplicações embarcadas para a plataforma multiprocessada Hemps.* 2009. Acesso em 02 de abril de 2013. Disponível em: <http://curtindoaciencia.blogspot.com.br/2011/09/fisica-do-cotidiano-porque-um-aviao-voa.html>. Citado na página 19.

SIMPSON, A. *Development of an unmanned aerial vehicle for low-cost remote sensing and aerial photography.* 2003. University of Kentucky. Citado na página 17.

SMITH, T. *Review BeagleBone Black.* 2011. Disponível em: <http://www.theregister.co.uk/2013/06/11/review_beagleboard_beaglebone_black/?page=1>. Citado 2 vezes nas páginas 9 e 40.

STOCHERO, T. *Polêmicos e revolucionários, mais de 200 'drones' voam no país sem regra.* 2013. Acesso em 25 de maio. Disponível em: <http://g1.globo.com/brasil/noticia/2013/03/polemicos-e-revolucionarios-mais-de-200-drones-voam-no-brasil-sem-regra.html>. Citado na página 23.

Anexos

ANEXO A – Custos do Sistema

A.0.1 Quadrimotor

Nessa secão são analisados alguns preços de *kits* que podem ser utilizados no sistema proposto. Na Tabela 9 são mostrados o preços de cada componente utilizados para montar o *kit*, que seria o quadrimotor, com a câmera e o sistema embarcado.

ITEM	PREÇO
Hélices para o quadmotor	R$5,60
Motor A2212-15 Brushless Outrunner 930KV	R$112,00
Turnigy Multistar 30 Amp Multi-rotor Brushless ESC 2-4S	R$89,20
Vídeo camera 1/30inch SONY CCD	R$34,00
Flight Controller com FTDI (ATmega 2560)	R$98,00
Boscam 5.8GHz 200mW FPV Wireless AV Tx e Rx Set	R$116,00
Raspberry Pi 512MB	R$50,00
Total dos itens:	R$504,80
Total dos itens + Importação + frete:	R$888,45

Tabela 9 – Lista de preços dos componentes utilizados no *kit* quadrimotor (21/10/2013).

A.0.2 Octamotor

A seguir são mostrados uma lista de itens e o preço final para montar um octamotor como o que foi usado em um dos experimentos desse trabalho utilizando um estabilizador de imagens.

- 01 (um) Frame Octocóptero;
- 08 (oito) Motores;
- 08 (oito) Hélices de Carbono 15,5×5;
- 01 (uma) Controladora YS-X4;
- 01 (um) Gimbal Cinestar Clone 3 Eixos;
- 01 (um) Placa Hover Fly Gimbal (ou similar);
- 01 (um) sistema de rádio 900MHz para Telemetria;
- 01 (um) Conversor HDMI DKI;

- 01 (um) FPV Transmissor 5.8GHz 600mW;
- 01 (um) Receptor 5.8GHz de FPV;
- 01 (um) Rádio Controle Futaba para controle do octamotor;
- 01 (um) Rádio Controle Turnigy para controle do *gimbal*;
- 04 (quatro) Baterias 4S 6000mAh Nanotech;
- 02 (duas) Baterias 3S;
- 01 (um) Carregador iCharger 206B + Fonte;
- 01 (um) Tripé + Monitor LCD 7";

O custo é de R$ 26.650,00 (21/10/2013). Este investimento contempla todo o orçamento acima, bem como despesas de treinamento, para o uso da plataforma.

ANEXO B – Configuração e instalação do software OpenTLD

B.1 Configuração

A aplicação do algoritmo *OpenTLD* é baseado em um arquivo de configuração que é dividido em três partes: aquisição , detecção e rastreamento (Tabelas 10, 11 e 12). Na aquisição são definidas as opções básicas de configuração, onde é estabelecida a forma de entrada para o programa. Os métodos de entrada podem ser “CAM”, “IMGS”, “VID” e “LIVESIM”. A forma de entrada “CAM” é usada para uma entrada de um dispositivo de captura disponível ao sistema operacional como *webcams* e dispositivos de captura de vídeos. O método “IMGS” é usado para fazer o rastreamento em um grupo de quadros de imagens disponíveis. “VID” é usado para fazer o rastreamento em arquivos de vídeo. A última opção “LIVESIM” é usada para *streams* de vídeo que geralmente são passados pela rede interna ou pela internet. Somente os métodos de entrada “CAM” e “VID” foram utilizados nos testes aplicados neste trabalho.

Tanto na detecção como no rastreamento os parâmetros não foram alterados para os testes realizados, e não foram usados modelos prévios para os testes.

Parâmetros	Descrição
method = "CAM";	Tipos de entrada: CAM, IMGS, VID, LIVESIM
imgPath = "diretório/arquivo.avi";	É requerido quando as entradas são IMGS, LIVESIM e VID.
camNo = 2;	Quando selecionado entrada tipo CAM, deve ser especificado o número do dispositivo.
startFrame = 0;	Se a entrada for do tipo VID, o número do quadro inicial a ser executado pode ser informado nesse parâmetro.
lastFrame = 0;	Também para o parâmetro VID, pode ser informado o quadro que deve ser finalizada a execução.
fps=24.0;	Taxa de quadros/Segundo a ser analisado no vídeo

Tabela 10 – Parâmetros de configuração de aquisição.

Parâmetros	Descrição
useProportionalShift = true;	Define as janelas a serem escaneadas por uma porcentagem de dimensão de janela (especificado no parâmetro proportionalShift).
proportionalShift = 0.1;	Porcentagem da janela.
minScale = -10;	Número de escalas menor que o tamanho do objeto inicial.
maxScale = 10;	Número de escalas maior que o tamanho do objeto inicial.
numFeatures = 10;	Número de recursos usados no conjunto de calssificadores.
numTrees = 10;	Número de arvores usadas no conjunto de classificadores.
minSize = 25;	Tamanho mínimo de uma janela a ser escaneada.
thetaP = 0.65;	Limiar da classe positiva.
thetaN = 0.5;	Limiar da classe negativa.
varianceFilterEnabled = true;	Habilita o uso de filtro de variância
ensembleClassifierEnabled = true;	Habilita o conjunto de classificadores
nnClassifierEnabled = true;	Habilita número de classificadores.

Tabela 11 – Parâmetros de configuração de detecção.

B.2 Instalação

Nesta seção serão apresentados os passos para instalação do *software OpenTLD* e suas dependências. Para instalar as bibliotecas *OpenCV* utilizou-se os seguintes passos:

1. Abrir o terminal;
2. Tornar super usuário da distribuição, digitando-se:

   #`sudo su`
3. Instalar a biblioteca OpenCV:

   #`apt-get install libopencv-dev`

O próximo passo foi instalar uma biblioteca que contém os arquivos *headers* e bibliotecas estáticas para compilar aplicações em *OpenCV* que possui uma interface gráfica, para isso realizou-se o seguinte comando no terminal:

#`apt-get install libhighgui-dev`

O *OpenCV* está instalado e pronto para usar. Mas ainda é necessária a instalação do binário do *cmake* para criar o *Makefile*, para isso digita-se no terminal:

#`apt-get install cmake`

Parâmetros	Descrição
trackerEnabled = true;	Habilita o uso do rastreador.
loadModel = true;	Se habilitado um modelo inicial pode ser especificado pelo parâmetro “modelPath”.
modelPath = "model.mod";	Diretório onde se encontra o modelo desejado.
initialBoundingBox = [100, 100, 100, 100];	A caixa de seleção inicial pode ser definida por esse parâmetro.
selectManually = false;	Se verdadeiro o usuário deve definir a caixa de seleção inicial do objeto no decorrer da execução.
threshold = 0.5;	Limiar para determinar resultados positivos.
learningEnabled = true;	Habilita o aprendizado enquanto ocorre o processamento.
trajectory = 20;	Especifica o número dos últimos quadros que são considerados pela trajetória. 0 desabilita a trajetória.
showOutput = true;	Se habilitado cria uma janela que mostra os resultados.
showNotConfident = true;	Mostra a caixa de seleção mesmo que a confiança esteja baixa.
showColorImage = false;	Mostra imagens coloridas ao invés de tons de cinza.
showForeground = false;	Mostra o primeiro plano.
saveOutput = true;	Se verdadeiro disponibiliza os quadros de saída no diretório especificado em “saveDir”.
saveDir = "Imagens/";	Diretório onde serão salvos os quadros.
printResults = "resultsFile.txt";	Gera um arquivo com os resultados.
printTiming = "timingFile.txt";	Gera um arquivo com os tempos dos resultados.
alternating = false;	Se for habilitado, o detector é desabilitado enquanto o rastreador estiver rodando.
exportModelAfterRun = false	Se habilitado o modelo criado atualmente será exportado para o diretório especificado em “modelExportFile”.
modelExportFile="model.mod";	Diretório onde será salvo o modelo atual.

Tabela 12 – Parâmetros de configuração de rastreamento.

Após configurar o ambiente basta realizar a compilação do código para gerar o binário.

#`cmake`

Terminada a compilação, para executar o programa digita-se o seguinte comando no terminal:

$`./opentld arquivodeconfiguracao.cfg`