FUNDAÇÃO UNIVERSIDADE DO TOCANTINS

CURSO DE SISTEMAS DE INFORMAÇÃO

JOÃO PAULO SOUZA PAIVA

Câmera de VANT Controlada por Visão Computacional para Acompanhamento Automático de Alvos Móveis.

PALMAS/TO

2014

JOÃO PAULO SOUZA PAIVA

Câmera de VANT Controlada por Visão Computacional para Acompanhamento Automático de Alvos Móveis.

Trabalho de Conclusão de Curso ao Curso de Sistemas de Informação da Fundação Universidade do Tocantins como parte dos requisitos para obtenção do título de Bacharel em Análise de Sistemas.

Orientador: Prof. MSc. Igor Yepes

PALMAS/TO
2014

COMISSÃO EXAMINADORA

_____

Prof. MSc. Igor Yepes (orientador)

_____

Prof. MSc Silvano Malfatti (titular)

_____

Prof. MSc.Alex Coelho (titular)

Palmas, ____ de _____ de 2014.

“A mente que se abre a uma nova ideia jamais volta ao seu tamanho original.”

– Albert Einstein

DEDICATÓRIA

Dedico esse trabalho ao meu amigo Bruno Morais, amigo de todas as horas, por seu incentivo, por ter acreditado que eu era capaz, por insistir comigo, por dividir os trabalhos e contribuir com seu conhecimento, habilidades e entusiasmo.

A todos os amigos e familiares.

AGRADECIMENTOS

A minha mãe que contribuiu sobremaneira para eu chegar aonde cheguei.

A meu amigo Bruno Morais por todo o incentivo e por compartilhar de todos os momentos de caminhada durante o curso, dividindo o fardo dos dias de labuta.

Ao Professor Igor Yepes por todo o apoio, pelas contribuições durante o curso e projeto de pesquisa, e fundamental contribuição para a finalização desse trabalho.

Ao Professor Marco Sousa por incentivar-me durante o início das minhas pesquisas em visão computacional, por acreditar e investir seu tempo e esforços a meu favor.

Ao Professor Silvano Malfatti por todo o seu empenho e determinação, por tamanha contribuição com nosso aprendizado e por seu jeito insano de ser que foi motivo de muitas gargalhadas.

Agradeço ao Professor Alex Coelho, Professor Frédson Vieira Costa, Professora Stéphane Martins, Professora Arlenes Delabary Spada, Professor André Rincon, Professor Carlos Henrique e todos os demais que não foram citados aqui que direta ou indiretamente contribuíram conosco, que foram motivo de inspiração e que foram nossos companheiros durante essa fase de nossas vidas.

Por fim agradeço a Deus por ter me dado a oportunidade de conhecer essas pessoas, por ter me dado disposição, saúde e cuidado para que tudo acontecesse no tempo certo.

A todos vocês, muito obrigado.

RESUMO

Este trabalho relata o processo de desenvolvimento de um sistema de visão computacional a ser instalado em um VANT (Veículo Aéreo não Tripulado) para acompanhar e rastrear alvos móveis. A câmera acoplada ao VANT é movimentada por dois servomotores (orientação pan e tilt) que por sua vez são conectados a uma controladora compatível com o hardware livre Arduino, responsável pelo controle dos movimentos dos servos. Por meio do *openFrameworks* em conjunto com a biblioteca *OpenCV* é realizada a captura e segmentação dos objetos de interesse nas imagens e realizado o reposicionamento do suporte da câmera do VANT.

Palavras-chave: Visão Computacional; *openFrameworks*; *OpenCv*; VANT.

ABSTRACT

The following work describes the process of construction of an UAV (Unmanned Aerial Vehicle) with a system of autonomous camera, controlled by computer vision to track moving targets. The camera is moved by two servomotors (pan and tilt orientation) that are connected to an Arduino compatible open hardware, which controls the camera movements. Though openFrameworks together with OpenCV library is performing the capture and segmentation of interest objects in the images and the repositioning the camera in the UAV.

Keywords*:Computer Vision; UAV; Arduino; openFrameworks; OpenCV.*

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIAÇÕES

VANT – Veículo Aéreo Não Tripulado

UAV – *Unmanned Aerial vehicle*

RGB – *Red, Green and Blue*

HSV – *Hue, Saturation and Value*

IDE – *Interface Development Environment*

RNA – Rede Neural Artificial

DPI – Dotch per Inch

ROM – *Read Only Memory*

RAM – *Random Access Memory*

USB – *Universal Serial Bus*

AVR – *Advanced Virtual Risc*

D3 – Dangerous Dirty Dull

GPS – Global Position System

VOR – VHF Omni Directional Radio Range

VHF – Very High Frequency

OPENCV – Open Computer Vision Library

BPS – Bits por Segundo

GCC – GNU Compiler Collection

SUMÁRIO

1. INTRODUÇÃO

Veículo Aéreo Não Tripulado (VANT) é um termo genérico que identifica uma aeronave que pode voar sem tripulação, normalmente projetada para operar em situações perigosas e repetitivas em regiões consideradas hostis ou de difícil acesso. Existe uma grande diversidade de tipos de VANT's, muitos deles ganhando ênfase na esfera civil e tornando-se uma opção válida no cenário comercial atual [1].

Essas aeronaves podem ir desde um veículo em escala controlado via rádio (planadores, helicópteros, dirigíveis, aviões, entre outros) a veículos tão sofisticados como aviões em tamanho real, com seus respectivos equipamentos de navegação (GPS, VOR, Servomecanismos, entre outros).Os VANT's têm propulsão própria utilizando forças aerodinâmicas que provocam a sua sustentação e não possuem cabine de pilotagem, pois podem ser controlados a distância ou podem possuir algoritmos sofisticados de voo que não requerem a intervenção humana [2].

Há um grande mercado emergindo a partir de aplicações e serviços potenciais que podem ser oferecidos pelas aeronaves não tripuladas. Mais precisamente, VANT's podem ser aplicados em missões chamadas de D^3 (*Dangerous-Dirty-Dull*), isto é, missões identificadas como perigosas, sujas e/ou enfadonhas. Quando se considera o VANT em aplicações civis, há um grande escopo de cenários possíveis para sua utilização. Por exemplo, pesquisa ambiental remota, monitoração e certificação de poluição, gerenciamento de queimadas, segurança, monitoração de fronteiras, oceanografia, agricultura e aplicações de pesca. Em geral, todas estas aplicações podem ser divididas em quatro grandes grupos: aplicações ambientais, aplicações de segurança, aplicações de comunicação e aplicações de monitoramento [3].

Há muito a ser explorado por essa tecnologia, embora por si só sua utilização já contemple uma ampla gama de atuações. Aliar esta e a outras tecnologias que vêm crescendo nos últimos anos, como a visão computacional, e capacitar o VANT a realizar operações de forma autônoma, seja para identificar e perseguir algum alvo previamente treinado ou para rastrear focos de incêndio

para agilizar na intervenção dos bombeiros para um combate mais eficiente, no intuito de debelar as chamas.

A Visão Computacional, já largamente utilizada na automação e robótica tem sido alvo de muitas pesquisas em diferentes segmentos de atuação, podendo solucionar problemas como, por exemplo, analisar imagens de tomografias ou em modalidades competitivas de futebol de robôs, capacitando os robôs a se orientarem na área do jogo e identificarem seus adversários. A visão computacional preocupa-se em simular a visão humana em dispositivos computadorizados, capacitando-os a processar e analisar imagens digitais para tomada de decisões.

Este trabalho apresenta uma solução de uma câmera com movimentação autônoma acoplada a um VANT, para reconhecer e rastrear focos de incêndio, sob o auxílio da visão computacional e da plataforma de prototipação *Arduino* que, por sua vez, será responsável pelo controle dos servomotores acoplados no suporte da câmera (*gimbal*) e responsáveis pelos seus movimentos (*pan* e *tilt*).

1.2 ORGANIZAÇÃO DO TEXTO

Os demais capítulos desse documento estão organizados da seguinte forma: no Capítulo 2 é definida qual a motivação para o desenvolvimento deste trabalho. No Capítulo 3 estão relacionados o objetivo geral e os objetivos específicos. No Capítulo 4 têm-se um levantamento bibliográfico realizado com o intuito de formar a base teórica necessária ao desenvolvimento da pesquisa, no qual estão relacionados alguns trabalhos em conformidade com a pesquisa aqui desenvolvida; é realizado um breve estudo sobre visão computacional, imagens digitais e os modelos de cores RGB e HSV. Aborda também o processo de segmentação de imagens, o conjunto de ferramentas *openFrameworks* e a plataforma de desenvolvimento que leva o conceito de hardware livre, *Arduino*. No Capítulo 5 têm-se a descrição da metodologia adotada, do algoritmo desenvolvido no *openFrameworks* e o algoritmo e hardware do *Arduino*. No Capítulo 6 são descritos os resultados e testes realizados. Por fim, os capítulos 7 e 8 apresentam, respectivamente, as conclusões e propostas de trabalhos futuros.

2. MOTIVAÇÃO

O cerrado, vegetação predominante no território Tocantinense e de boa parte da região norte do país, possui uma característica fatídica nos períodos de estiagem, pois torna-se seco e propício a eventuais propagações de fogo. Ainda que o solo seja considerado com baixos relevos e a vegetação rasteira, arbustiva com arvores e arbustos esparsos, alguns lugares tornam-se desafiantes para equipes dos Corpos de Bombeiros no combate a incêndios florestais.

As intervenções por parte dessas equipes de profissionais no intuito de debelar as chamas e realizar uma ação incisiva, exigem o emprego de uma grande logística, sejam equipamentos, material humano ou mesmo aeronaves para auxiliar nas missões. Muitas vezes, pelo fato de se possuir pouca visibilidade e condições climáticas desfavoráveis, como por exemplo, a velocidade do vento e sua direção, as vidas envolvidas podem ser colocadas em risco.

Os custos para colocar uma aeronave no ar são consideravelmente altos, sejam eles inerentes à própria aeronave, com combustível, manutenção preventiva, os profissionais que realizam a manutenção no equipamento, ou mesmo dos profissionais envolvidos na operação, piloto e tripulantes com treinamento adequado.

Diante disso, a utilização de VANTs surge como uma opção viável para auxiliar tais operações, dotado de visão computacional capaz de rastrear as chamas e trazer informações em tempo real para que, com essas informações, seja possível realizar ações mais eficazes. O custo para colocar esse equipamento em missões de inspeção é muito baixo se comparado ao que foi citado anteriormente.

Sempre que houver a necessidade de chegar a um lugar que ofereça risco ao profissional ou para agilizar a pré-visualização das condições de determinado ambiente, salvo ambientes sinistrados inacessíveis para o equipamento, um VANT poderá ser empregado e nortear os Bombeiros para que estes possuam o máximo de informações possíveis e definam a melhor estratégia, aumentando assim as chances de sucesso.

Além do exposto, há interesse em pesquisar novas funcionalidades e

aplicações para os VANTs com o intuito de favorecer os Corpos de Bombeiros e demais instituições de segurança pública, mediante pesquisas de campo direcionadas para as reais necessidades de cada um desses órgãos, buscando otimizar processos e melhor se valer dos recursos disponibilizados pelas inúmeras aplicações de técnicas de visão computacional aliada a VANTs.

3. OBJETIVOS

3.1. OBJETIVO GERAL

Construir um sistema de visão computacional capaz de analisar imagens capturadas por uma câmera, realizar o processamento e efetuar o reposicionamento do equipamento de acordo com os movimentos do alvo.

3.2 OBJETIVOS ESPECÍFICOS

- Estudar e compreender a forma de utilização da biblioteca *OpenCV*;
- Estudar e compreender o Hardware Livre *Arduino*, necessário para controle do suporte gimbal da câmera;
- Desenvolver um software capaz de identificar objetos específicos no ambiente por meio de Visão Computacional;
- Instalar e testar o sistema de acoplamento e controle da câmera do VANT;
- Desenvolver o software necessário para controlar a câmera de forma autônoma, permitindo o acompanhamento de objetos em movimento ou fixando a posição da câmera um objeto estático durante o voo;
- Realizar testes de campo como sistema desenvolvido.

4. REFERENCIAL TEÓRICO

4.1 TRABALHOS RELACIONADOS

Há muitos trabalhos relacionados a navegação autônoma de robôs com uso de visão computacional, haja vista ser essa uma área de estudos vasta e promissora. Diversos trabalhos focam no rastreamento de alvos móveis, utilizando as mais diversas técnicas de visão computacional.

Em [4] é apresentada uma solução semelhante à aqui proposta, pois é realizado o rastreamento de objetos de interesse por meio do tratamento de imagens digitais em tempo real com a movimentação de uma cabeça robótica. Seu objetivo é definir informações de localização e distância, diferente do que é tratado neste trabalho acerca da detecção de objetos por meio de cores, mas de forma semelhante, o movimento da cabeça robótica é orientado sempre para centralizar o objeto de interesse na imagem capturada pela câmera.

Utilizando técnicas para detecção de cores para verificação do bem-estar dos animais em um criadouro de suínos, tomando-se como base a temperatura corporal, dados obtidos através de uma câmera com sensor de calor, Silva [5] esboça um estudo baseado em dados comportamentais e parâmetros previamente determinados para que, diante dos resultados obtidos, possa proporcionar-lhes um ambiente confortável e melhorar a produção no criadouro.

Baseado na automação de sistemas produtivos, Rudek [6] propõe um sistema de processamento de imagens digitais e visão computacional para classificação de parafusos através da análise de padrões e segmentação por cores. Explora também uma metodologia para armazenamento, classificação e indexação de imagens em um banco de dados utilizando como referência as cores das imagens capturadas pela câmera.

Explorando uma abordagem diferenciada no processo de detecção dos objetos de interesse, Quiles [7] utiliza Redes Neurais Artificiais (RNA) para treinamento de um equipamento que realiza a segmentação por meio de padrões de cores. Desta forma, à medida que os objetos de interesse são identificados na imagem capturada, um robô móvel terá a orientação necessária para que, através

dos dados de saída da RNA, efetue o seu deslocamento em direção ao objeto segmentado.

Ross [8], discute o uso de visão computacional na identificação e localização de objetos em terra a partir de um VANT autônomo, algo semelhante ao que se propõe este trabalho em etapas ulteriores, já, de forma semelhante, em [9] é apresentada uma implementação de VANT autônomo com características de visão computacional que lhe permitem a identificação e o seguimento de objetos em movimento.

Diante do que foi apresentado, podem ser percebidos avanços no sentido de buscar maneiras variadas de empregar a visão computacional na automação de processos e, de forma semelhante, realizar a tomada de decisões a partir da detecção de objetos em imagens digitais baseando-se nas cores para sua segmentação.

4.2 VISÃO COMPUTACIONAL

A Visão Computacional tem como objetivo a aquisição, pré-processamento, segmentação, extração de características, reconhecimento e interpretação (Figura 1). Procura emular a visão humana, portanto também possui como entrada uma imagem que nesse caso é capturada por uma câmera digital, porém a saída é uma interpretação parcial, restringindo-se a apenas os objetos de interesse [4].

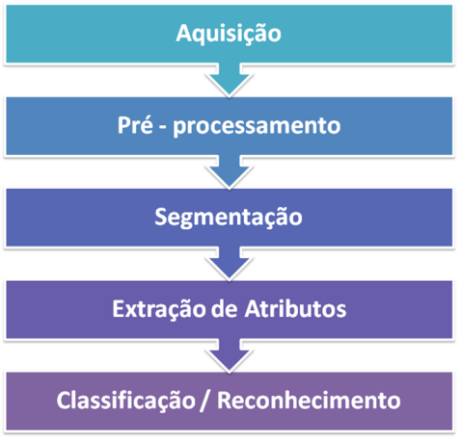

Figura 1: Etapas da Visão Computacional. Fonte: [26].

Tais técnicas são descritas da seguinte forma:

- Aquisição - Nessa etapa o hardware deve converter os sinais ópticos em sinais elétricos, sendo dotado de um sensor e posteriormente transformando o sinal analógico em sinal digital. Alguns equipamentos podem ser citados com tal capacidade, a saber: câmeras digitais, scanners, filmadoras e etc.
- Pré-Processamento - É considerada uma operação de baixo nível que consiste no melhoramento da imagem, cujo objetivo é remover possíveis ruídos, *pixels* defeituosos, melhorar a iluminação, tratar o brilho inadequado, caracteres interrompidos ou indevidamente conectados. O objetivo dessa fase é facilitar a etapa de segmentação.
- Segmentação - O objetivo é dividir a imagem em unidades significativas, ou seja, onde esse segmento de imagem será analisado para realizar a extração das características.
- Extração de Características - Nessa etapa serão utilizados algoritmos de reconhecimento de padrões cuja saída é um dado útil para posterior interpretação.
- Reconhecimento e Interpretação - Uma vez que os objetos foram segmentados e suas características são conhecidas, deve-se atribuir um devido significado ao conjunto de objetos reconhecidos.

Segundo Bradisk [23], visão computacional é a transformação de dados oriundos de uma máquina fotográfica ou câmera de vídeo em qualquer decisão ou em uma nova representação.

Shapiro [22] afirma: “A meta da visão computacional é tomar decisões sobre objetos físicos reais a partir de cenas baseadas em imagens captadas”. Ainda afirma que a fim de efetuar tomadas de decisões sobre objetos físicos reais, é quase sempre necessário construir alguma descrição ou modelo destes objetos.

Pesquisadores em visão computacional têm desenvolvido diversas técnicas para evoluir as aplicações de visão computacional, como por exemplo, efetuar a utilização de técnicas matemáticas para recuperar a forma tridimensional e aparência de objetos em imagens, rastrear um determinado objeto em uma cena,

e atém mesmo, com um sucesso moderado, encontrar o nome de todas as pessoas em uma fotografia com base em uma combinação entre rosto, roupa, cabelo utilizando-se de técnicas de detecção e reconhecimento.

Apesar de todos esses avanços, o sonho de ter um computador que interprete uma imagem no mesmo nível que um menino de dois anos (por exemplo, realizando a contagem de todos os animais em uma foto) permanece indefinido. Isso ocorre porque em visão computacional, muitas vezes é necessário fazer o reconhecimento de objetos em uma cena, mesmo com insuficiente quantidade de informações para realizar tal reconhecimento. Devemos, portanto, recorrer a técnicas baseadas em física e estatística para distinção e/ou correlação entre possível soluções e decisões [24].

Diversas e complexas técnicas de visão computacional vêm sendo estudadas para a aplicação, principalmente, na área da robótica e automação. Todavia, sua utilização tornou-se de fácil acesso para estudos graças a utilização da biblioteca *OpenCV* (*Open Source Computer Vision*), desenvolvida inicialmente pela *Intel Corporation*. A *OpenCV* implementa uma variedade de ferramentas de interpretação de imagens, indo desde de operações simples como um filtro de ruído, até operações complexas, tais como análise de movimentos, reconhecimento de padrões e reconstrução 3D. O pacote *OpenCV* está disponível gratuitamente na Internet bem como o seu manual de referência [4].

4.3 IMAGENS DIGITAIS

Imagens digitais são obtidas através de equipamentos diversos dotados de sensores fotossensíveis que captam a luz, todavia, os sinais que são captados por tais equipamentos são descritos por uma função contínua, diferente do que um computador consegue processar, uma vez que lida com dados discretos, ou seja, valores representados na forma binária de “0” e “1”.

Nesse caso, a tarefa inicial é converter os estímulos luminosos, cuja representação é uma onda senoidal contínua (Figura 2), em um conjunto de valores conhecidos pelo computador, tarefa essa denominada discretização dos sinais ópticos, para então serem processados e armazenados.

O primeiro passo para se discretizar o sinal contínuo é passar pela etapa de amostragem, onde uma função f(x) recebe valores pontuais do sinal recebido e a saída da função corresponde a um valor aproximado correspondente. Esse processo de atribuição de valores é chamado de Quantização.

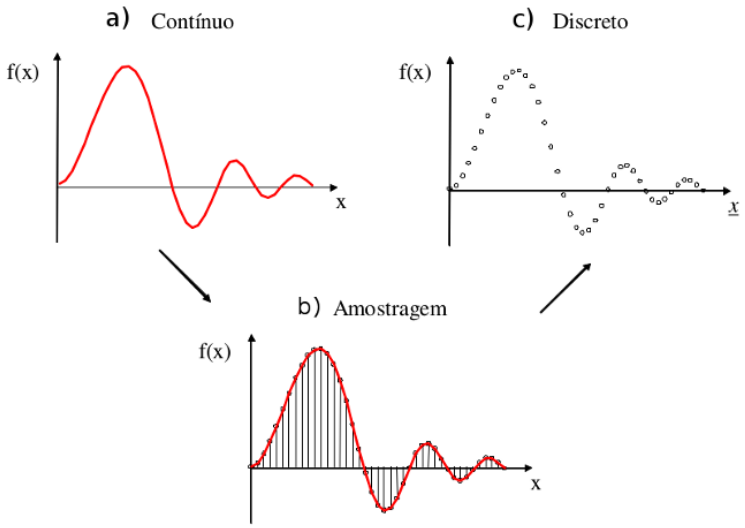

Figura 2: (a) sinal contínuo original, (b) sinal com divisão por amostragem e (c) sinal discretizado. *Fonte: [25].*

Uma vez que a imagem é digitalizada, ela assume um tamanho adimensional em *pixels*, por sua vez, o *pixel* (*Picture Element*) é o menor ponto visível em uma imagem. Em um dispositivo de armazenamento, uma imagem pode ser medida tendo como base a razão entre a quantidade de *pixels* e o seu tamanho real, o produto disso é o a resolução da imagem, que pode ser medida em pontos por polegada ou DPI (*dots per inch*), mais comumente utilizado [25].

Assim como exemplificado na Figura 3, uma imagem digital pode ser representada por uma matriz bidimensional MxN de *pixels*, cada *pixel* é dotado de um conjunto de componentes que define o espaço de cores da imagem. De acordo com o número de componentes que estão presentes em cada *pixel* é possível tratar a imagem como sendo monocromática, também denominadas imagens em escala de cinza (*grayscale image*), cuja propriedade que a define é a intensidade, ou seja, são definidos diferentes tons de cinza à medida que a intensidade aumenta (mais próxima da cor branca) ou diminui (mais próxima da

cor preta) [26].

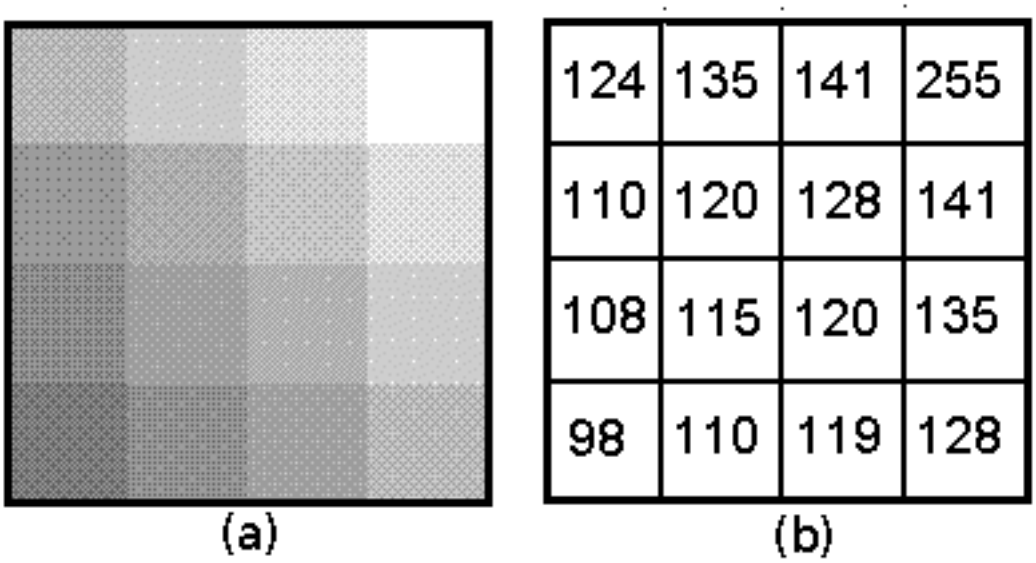

Figura 3: (a) Ilustração de uma imagem digital com diferentes tons de cinza e (b) Matriz 4x4 pixels e seus valores numéricos correspondentes aos vários níveis de cinza.Fonte: [27].

É comum em visão computacional tratar imagens monocromáticas, em que são definidos tons de cinza distintos para cada cor presente na imagem original.

4.4 MODELOS DE CORES

Em visão computacional, durante o processo de captura das imagens através de uma câmera digital, as imagens são armazenadas e tratadas com base em um modelo de cores, podendo sofrer variações para melhor se adequar ao objetivo proposto.

Um modelo de cor é uma representação tridimensional em que cada cor é representada por um ponto na coordenada de cores, sendo que cada modelo, em virtude de suas limitações, possui um universo de cores que pode representar. Não existe um modelo capaz de descrever de forma independente todos os aspectos das cores percebidos pelo olho humano, devido a isso, faz-se necessário a utilização de modelos diferentes para tal representação [17].

Dois modelos destacam-se no tratamento de imagens, são os modelos RGB (*Red, Green* e *Blue*) e HSV (*Hue, Saturation* e *Value*). O Modelo RGB está associado a maioria dos sistemas comumente utilizados para a representação de cores de forma independente, como monitores de vídeo e TV. Já o modelo HSV é

utilizado para tentar facilitar a especificação da informação da cor por parte do usuário, ou seja, onde há a necessidade de interação com o usuário na especificação da cor desejada, de forma que esta tarefa se torne mais simples.

4.4.1 MODELO RGB

Baseado na teoria tricromática de Young-Helmholtz, o modelo RGB é um sistema aditivo cuja percepção das cores por parte do olho humano está associada a estimulação dos três pigmentos visuais presentes nos cones da retina, vermelho, verde e amarelo [17]. A cor é gerada pela mistura de vários comprimentos de onda luminosa, provocando a sensação de cor quando atinge o olho. No processo aditivo, o preto é gerado pela ausência de qualquer cor, indicando que nenhuma luz é transmitida, por outro lado, o branco é a mistura de todas as cores.

Computacionalmente, para se obter uma determinada cor no modelo em questão, é usado um intervalo predeterminado que varia entre 0 e 255, cujo branco é a obtido pela combinação (255, 255, 255), o preto (0, 0, 0), o vermelho (255, 0, 0), o verde (0, 255, 0) e o azul (0, 0, 255). O sistema de cor RGB pode ser representado como um cubo, conhecido como cubo RGB, conforme a Figura 4.

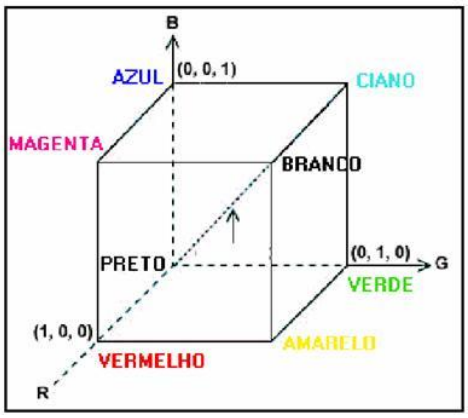

Figura 4: Cubo RGB. Fonte: [28].

Cada ponto no interior do cubo corresponde a uma cor que pode ser representada pela tripla (R,G,B) com os valores R, G e B indo de 0 a 255, ou seja,

cada *pixel* RGB possui três *bytes* de armazenamento, um *byte* para cada cor primária (vermelho, verde e azul), isso significa que nesse modelo pode-se trabalhar com mais de 16,7 milhões de cores.

Embora o sistema RGB atenda muito bem a representação de cores para monitores e TV, não é garantido que essa representação seja a mesma realizada pela percepção visual e fica difícil determinar, visualmente, se uma cor é ou não uma cor de interesse.

4.4.2 MODELO HSV/HSB

O modelo HSV (*Hue, Saturation, Value*), também conhecido por HSB (*Hue, Saturation, Brightness*), é composto por três componentes básicos, matiz, saturação e valor referente ao nível de brilho. A matiz determina a cor ou tonalidade; o brilho determina a intensidade percebida, podendo ser mais clara ou mais escura; a saturação refere-se ao aspecto de pureza da cor ou profundidade, podendo ficar esmaecida ou intensa.

A vantagem deste modelo é que para se determinar uma cor, os valores da matiz são determinados em graus, que variam entre 0 a 359. Sua percepção é mais intuitiva do que o modelo anteriormente citado, e é mais adequado para ser usado na especificação de cores em nível de interface com o usuário.

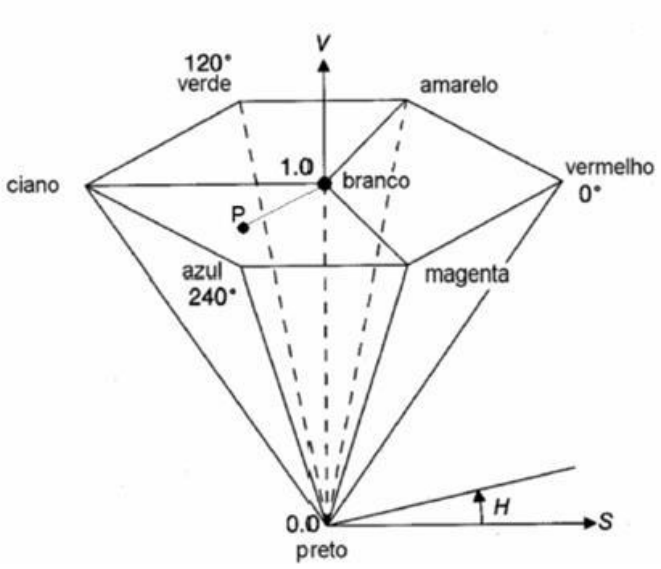

Figura 5: Hexacone HSV. Fonte [28].

O sistema HSV pode ser representado por um cone hexagonal (hexacone) ou uma pirâmide invertida de seis lados, assim como mostra a Figura 5, onde as cores correspondem às coordenadas dentro deste espaço. No topo do hexacone encontram-se todas as cores com mais brilho, à medida que posiciona-se no ponto mais fino do cone, o brilho diminui, então temos cores mais escuras [17].

4.5 SEGMENTAÇÃO

Uma etapa importante no processo de visão computacional é a etapa de segmentação, pois nesse momento são utilizadas técnicas de processamento de imagens para salientar aquilo que se deseja. Após a etapa de captura (aquisição) da imagem por meio de uma câmera digital, são empregados os devidos filtros para tirar ruídos que comprometam a detecção da área de interesse. Algumas abordagens necessitam de tratamentos mais finos, que lhes permitam capturar características específicas de algo que se procura. De acordo com Cavani [5], citando Spirkovska (1993):

> "Segmentação de imagens é aponte na visão computacional entre os subsistemas de visão de baixo nível, que inclui as operações de processamento de imagens (com redução de ruídos e extração de bordas) para melhorar a imagem de entrada, e o subsistema de alto-nível que inclui reconhecimento de objetos e interpretação da cena."

Os ruídos tendem a ser interpretados como qualquer obstáculo que venha dificultar a segmentação. Podem se originar do tipo de sensor que se está utilizando, da iluminação do ambiente, das condições climáticas, da posição da imagem relativa à câmera ou até mesmo das partes da imagem que não correspondem necessariamente a aquilo que se deseja segmentar.

Muitas vezes, devido à abordagem a ser realizada, não há a necessidade gastar tanto processamento nessa etapa (segmentação) como, por exemplo, na detecção de cores. Nesse caso, uma abordagem possível é a segmentação por crescimento de região, ou seja, os *pixels* são agrupados em regiões maiores

baseados em critérios previamente definidos. Assim, um conjunto de pontos e seus vizinhos são selecionados analisando-se medidas de similaridade para que se possa dizer que ambos fazem parte de um mesmo subgrupo. Nessa abordagem, é muito importante que sejam definidos os pontos ou sementes e um critério de parada, para limitar a área de interesse [4].

Por outro lado, algumas técnicas exigem um treinamento prévio para tornar o sistema capaz de identificar objetos em uma imagem sob as mais diversas condições, de forma que tais ruídos sejam considerados. Algumas abordagens podem utilizar técnicas com RNA's (Redes Neurais Artificiais), Lógica *Fuzzy*, Cascata de Classificadores (ver apêndice A), entre outras. Tais técnicas permitem que o sistema tenha uma taxa de falsos positivos (falhas onde o objeto segmentado não corresponde ao que se deseja) muito baixa, muito embora o preparo do sistema seja bem mais demorado, uma vez que se deve mapear os padrões de um objeto em diferentes condições de iluminação e ambientes variados, fazendo-se necessário possuir previamente um banco de imagens para realizar o treinamento.

4.6. OPENFRAMEWORKS

O *OpenFrameworks* é um *kit* de ferramentas de código aberto desenvolvido em C++. Facilita o processo de desenvolvimento de aplicações que utilizem recursos que fazem integração com o *hardware*. Ele tenta prover uma estrutura de uso geral por envolver várias bibliotecas para manipular recursos de entra e saída, tais como:

- *OpenGL* para trabalhar com interface gráfica;
- *OpenCV* para visão computacional;
- *OpenAL* para entrada e saída de áudio;
- *Assimp* para modelagem 3D;
- *FreeType*para fontes;
- *FreeImage* para manipulação de imagens, salvar, carregar;

- *Quicktime, Gstreamer* para *playback* de vídeos e captura de imagens digitais por meio de uma câmera.

O *openFrameworks* é compatível com sistemas *Android, iOs*, *Windows*, *Linux* e *OSX,* e compatível com as interfaces de desenvolvimento *XCode, Code::Blocks, Visual Studio* e *Eclipse*. É distribuído na licença *MIT* que possibilita criar aplicações comerciais ou livres, de código aberto ou não, públicas ou privadas, dando desta forma total liberdade para sua utilização. É desenvolvido em linguagem C++.

Este trabalho utiliza o Ambiente de Desenvolvimento Integrado, IDE (*Integrated Development Environment*) Code::Blocks, no ambiente operacional Linux. A Figura 6 mostra uma ilustração da IDE.

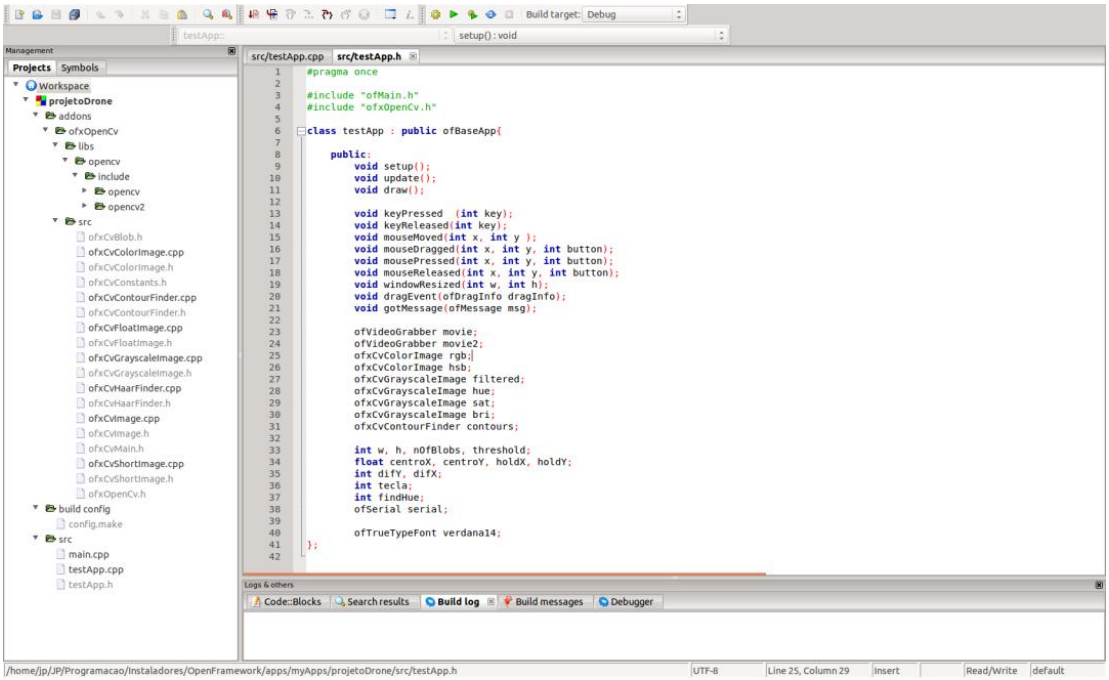

Figura 6: openFrameworks IDE. *Fonte: Autor.*

4.7. ARDUINO

O *Arduino* é uma plataforma de prototipagem eletrônica de *hardware* livre, projetada com um microcontrolador ATMel AVR de placa única, com suporte de entrada/saída embutido e com uma linguagem de programação padrão a qual tem origem em *Wiring* (essencialmente C/C++). O objetivo dessa plataforma é criar

ferramentas acessíveis, com baixo custo, flexíveis e fáceis de se usar [6].

A unidade central de processamento do *Arduino* é praticamente um computador completo, dotada de memória de acesso aleatório (RAM), memória para armazenamento da aplicação que é instalada, ou seja, somente de leitura (ROM), controle dispositivos de entrada e saída e uma unidade aritmética, tudo em um único chip [7].

A placa do *Arduino* (Figura 7) permite a conexões através de pinos, ou seja, pequenas orifícios onde são conectados os dispositivos, composta por pinos digitais e analógicos, pinos de 5 e 3,3 volts, pino de reset, pinos de neutro (GND) e referência analógica. Dependendo de como se deseja trabalhar, os pinos analógicos são configuráveis, possibilitando leituras analógicas ou entradas digitais, ainda uma porta USB e um conector para Fonte de Alimentação. Vale ressaltar que, como a quantidade de conexões do *Arduino* dependendo da versão chega ser insuficiente para algumas intervenções, deve-se fazer o uso de *protoboards* ou *protoshields*.

Figura 7: Placa Arduino. Fonte: [7].

O ambiente de desenvolvimento integrado do *Arduino*, Figura 8, é uma interface simples, de fácil utilização que foi desenvolvida em Java, mas que na verdade roda um compilador gcc para as linguagens C e C++. A IDE se resume basicamente com o intuito de desenvolver aplicações e compilá-las instalando-as

na placa para que possam ser executada pelo microcontrolador [7].

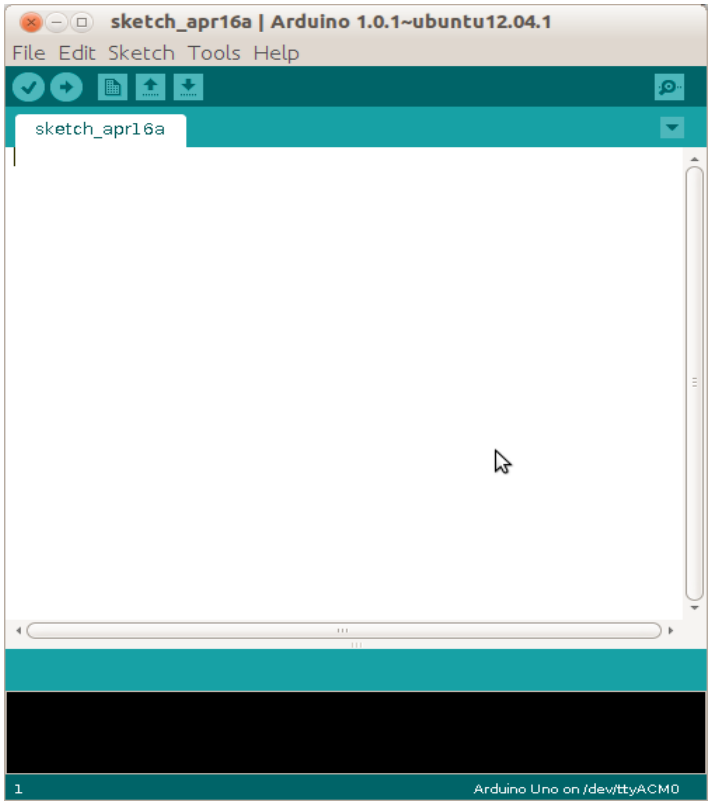

Figura 8: Arduino IDE. Fonte: Autor.

O *Arduino* IDE está disponível nas plataformas Linux, Mac e Windows.

5. MATERIAIS E MÉTODOS

5.1. ALGORITMO DESENVOLVIDO NO OPENFRAMEWORKS

O algoritmo utilizado neste trabalho foi desenvolvido utilizando o *kit* de ferramentas de código aberto *openFrameworks* juntamente com a biblioteca *OpenCV*, que primeiramente captura as imagens da câmera no padrão *RGB* e as armazena numa variável apropriada para tratamento de imagens tratamento de imagens coloridas. O sistema então converte a imagem capturada para o modelo de cores *HSV* e em seguida utilizando uma classe do *OpenCV*, *GrayScaleImage*, separa em variáveis específicas os *pixels* nos atributos que compõem o modelo de cores HSV para matiz, brilho e saturação, convertendo-os em diferentes tons de cinza, técnica está utilizada pela visão computacional para reduzir o nível de dificuldade para o processo de segmentação.

O sistema desenvolvido traz consigo a possibilidade de selecionar o valor da matiz dinamicamente, ou seja, à medida que as imagens são capturadas pela câmera e são geradas na tela do microcomputador, então o usuário poderá selecionar uma área da imagem e clicar sobre um alvo de interesse para que o sistema capture os *pixels* daquela região referentes ao valor da matiz, armazenando-os para servirem como parâmetro de busca.

Após definir a frequência dos tons de cinza para cada elemento do modelo HSV, o sistema inicia uma rotina para realizar uma varredura e selecionar dentro da imagem somente os *pixels* que correspondem à cor especificada (Figura 9), é efetuada uma binarização (preto e branco) dos valores desses *pixels* armazenando-os em uma variável cujo objetivo é servir de filtro para a classe que irá segmentar o objeto encontrado.

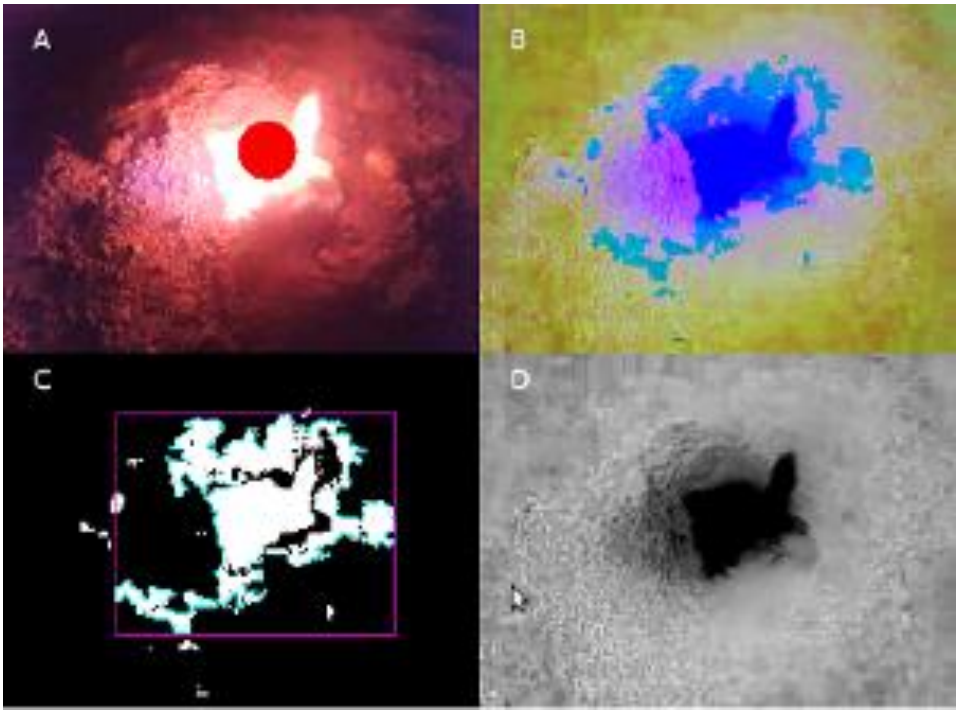

Figura 9: (A) Imagem RGB, (B) Imagem HSV, (C) Imagem binarizada com a chama sendo segmentada e (D) Imagem com tom de cinza referente ao atributo saturação do modelo HSV. Fonte: Autor.

O *OpenCV* traz nativamente classes para realizar a segmentação a partir dos tons de cinza previamente definidos, é a classe *ContourFinder*, que por meio do método *findCountours* detecta as bordas do objeto, ou seja, realizada a limiarização, cujo objetivo é converter a imagem em dois tons a partir de um dado ponto de corte, são selecionados somente os *pixels* que estiverem dentro do parâmetro que foi passado.

Uma vez que o objeto de interesse foi detectado, são realizados os seguintes procedimentos para identificar o posicionamento do objeto e definir a orientação dos movimentos vertical e horizontal da câmera, temos as seguintes equações:

- Equação para calcular o valor percentual do centro do objeto pelo eixo y no plano cartesiano, tomando como referência a altura da imagem:

$$Cp_y = \frac{\left(\frac{h_{iy}}{2}\right) \times 100}{H} \quad (1)$$

Onde Cp_y corresponde ao valor percentual do centro do objeto relativo à altura da imagem capturada, neste caso definida por H, e h_{iy} a altura do objeto detectado.

- Equação para calcular o valor percentual do centro do objeto pelo eixo x do plano cartesiano, tomando como referência a largura da imagem:

$$Cp_x = \frac{\left(w_{ix} \div 2\right) \times 100}{W} \qquad (2)$$

Onde Cp_x corresponde o valor percentual do centro do objeto relativo à altura da imagem capturada, neste caso definida por W, e w_{ix} à largura do objeto detectado.

- Calcular a porcentagem que falta para que a posição do objeto no eixo y corresponda à metade da altura da imagem:

$$d_y = 50 - Cp_y \qquad (3)$$

O valor de d_y resultante dessa equação corresponde ao movimento necessário que a câmera deverá realizar no eixo y, ou seja, movimento *pan*, para centralizar o objeto detectado na área de visualização da câmera. Caso o resultado da subtração seja negativo, isso indicará que o movimento deverá ser para baixo, caso contrário o movimento da câmera será para cima.

- Calcular quantos por cento faltam para que a posição do objeto no eixo x corresponde à metade da largura da imagem:

$$d_x = 50 - Cp_x \qquad (4)$$

O valor de d_x resultante dessa equação corresponde ao movimento

necessário que a câmera deverá realizar no eixo x, ou seja, movimento *tilt*, para centralizar o objeto detectado na área de visualização da câmera. Caso o resultado da subtração seja negativo, isso indicará que o movimento deverá ser para a esquerda, caso contrário o movimento da câmera será para a direita.

Após realizar tais cálculos, são enviados para o *Arduino* os valores de d_x e d_y, para que os servomotores sejam movimentados e reposicionados de forma que o objeto detectado seja centralizado no campo visual da câmera.

5.2. HARDWARE E O ALGORITMO DESENVOLVIDO NO ARDUINO

O equipamento proposto nesse projeto utiliza uma câmera acoplada a um *gimbal*com servomotores (ambos os equipamentos de baixo custo e fácil utilização), como pode ser observado na Figura 10, atuando em conjunto com um sistema de *hardware/software* construído com base na plataforma de prototipagem *Arduino*.

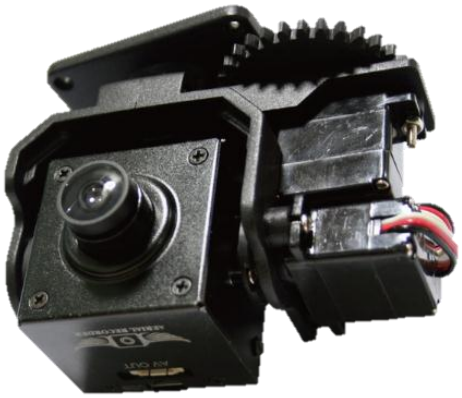

Figura 10: Câmera utilizada instalada no gimbal munido de dois servomotores controlados pelo Arduino com base nos dados das imagens capturadas. A parte superior do gimbal será fixada na parte inferior do VANT, sob o compartimento da bateria.Fonte: [30].

Para a estruturação da plataforma de visão computacional no VANT, primeiramente foi realizada a montagem do suporte da câmera que comporta dois servomotores; o primeiro liga-se à base, fixada diretamente na estrutura do VANT, para fazer a rotação horizontal da câmera (*pan*) e o segundo, conectado ao

suporte da câmera, para os movimentos na vertical (*tilt*).

A Figura 11 apresenta um esquema dos graus de liberdade da câmera montada sobre um gimbal munido de dois servomotores, detalhando as rotações dos eixos associados aos movimentos de *pan* e *tilt*.

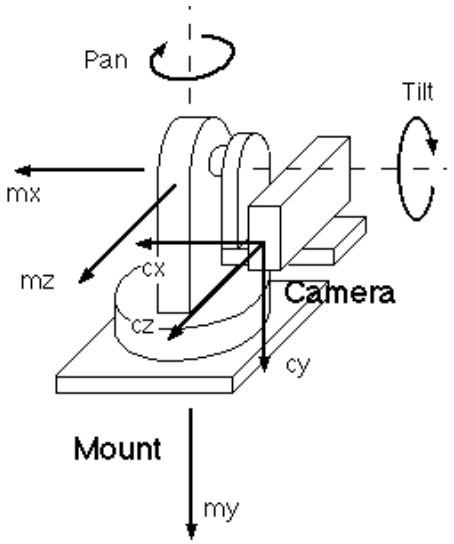

Figura 11: Visualização esquemática das rotações sobre o eixo x (tilt) e y (pan) efetuadas pelo Arduino sobre os servomotores incorporados no gimbal.Fonte: [29].

O conjunto composto pelo gimbal e pela câmera é montado na parte inferior do VANT, ficando com a área de visualização livre, sem interferência das hélices e demais componentes existentes no equipamento.

Na figura seguinte pode ser observado o protótipo do VANT no qual ainda não foi instalado o gimbal - para facilitar o desenvolvimento do sistema de visão computacional, os testes foram realizados com o equipamento separado, para posterior incorporação.

Essa separação foi necessária, uma vez que o VANT compreende uma série de subprojetos que estão sendo desenvolvidos em paralelo, abrangendo visão computacional, SLAM (*Simultaneous Localization and Mapping*) com reconstrução 3D de ambientes internos, *path planing* com uso de Inteligência Artificial, sensoriamento ambiental e desenvolvimento de sistema de controle via dispositivos móveis [13], [14], [15].

Figura 12: Protótipo de VANT em desenvolvimento, constituído por 4 motores sem escova de 750Kv, um frame de fibra de carbono, hélices de nylon-carbono, bateria com autonomia aproximada de 15 minutos, controladora de voo com magnetômetro digital, acelerômetro, giroscópio e barômetro *onboard*, módulo GPS, sistema *fail safe* e *payload* aproximado de 1Kg.Fonte: Autor.

A conexão da câmera é realizada por meio de comunicação *WiFi* utilizando o receptor que está conectado ao computador (Conjunto Transmissor/Receptor *AV WiFi*). Ambos os servos são conectados ao *Arduino* por meio de seu conjunto de pinos, possibilitando assim a comunicação entre os equipamentos.

Utilizando a classe *ofSerial,* sendo que inicialmente foi configurado com taxa de transmissão de 9600*bps e a mesma velocidade configurada no Arduino*, os dados são enviados *byte* a *byte* pela porta serial, sendo armazenados em variáveis para, em seguida, calcular o movimento adequado que os servomotores deverão realizar. De acordo com o *hardware* e após alguns testes, foi possível definir os valores relativos aos limites que os servomotores poderiam rotacionar, determinando o ângulo de rotação máximo e mínimo.

Em um primeiro momento, o algoritmo inicializa as variáveis de posicionamento da câmera para deixá-la centralizada. Ao passo que os dados são transmitidos pela porta serial para o *Arduino*, este deve verificar se os valores recebidos representam um reposicionamento, levando em consideração que variações entre -5 e 5 foram definidas como folga, ou seja, são aceitáveis não havendo necessidade de acionar os servomotores.

Esses valores de folga, definidos no algoritmo, correspondem a uma tolerância a pequenas variações na detecção do objeto de interesse no sistema de visão computacional. Após alguns teste percebeu-se, de forma empírica, que

essa variação oscilava na casa dos cinco por cento para ambas as direções nos eixos de movimentação x e y.

Caso os valores recebidos apresentem índices superiores a cinco por cento, verifica-se o sinal. Se o valor do sinal for positivo, o valor referente a posição atual do servo é incrementado, caso contrário decrementado.

As variáveis que representam o movimento dos servomotores sempre são verificadas e somente há o incremento destas se os valores forem menores que os limites previamente estabelecidos para cada eixo.

6. RESULTADOS OBTIDOS

Na tentativa de obter uma maior eficácia na detecção de chamas, o sistema trouxe o recurso de realizar a seleção das cores de forma dinâmica e tomando como base o sistema HSV, devido ao fato de ser esse modelo mais intuitivo para aplicações que requerem a intervenção do usuário, além de também possibilitar trabalhar com apenas um atributo para especificar a cor propriamente dita, no caso, a matiz.

Com o auxílio da visão computacional foi possível segmentar a região que apresenta a cor da chama, embora em condições adversas, o sistema se mostrou deficiente quando surgem no mesmo campo visual tons semelhantes aos da cor selecionada. Observando a Figura 13, onde foi selecionado o fogo, é possível perceber que o sistema não consegue fazer a distinção entre as cores do fogo com as cores referentes ao brilho do sol no muro.

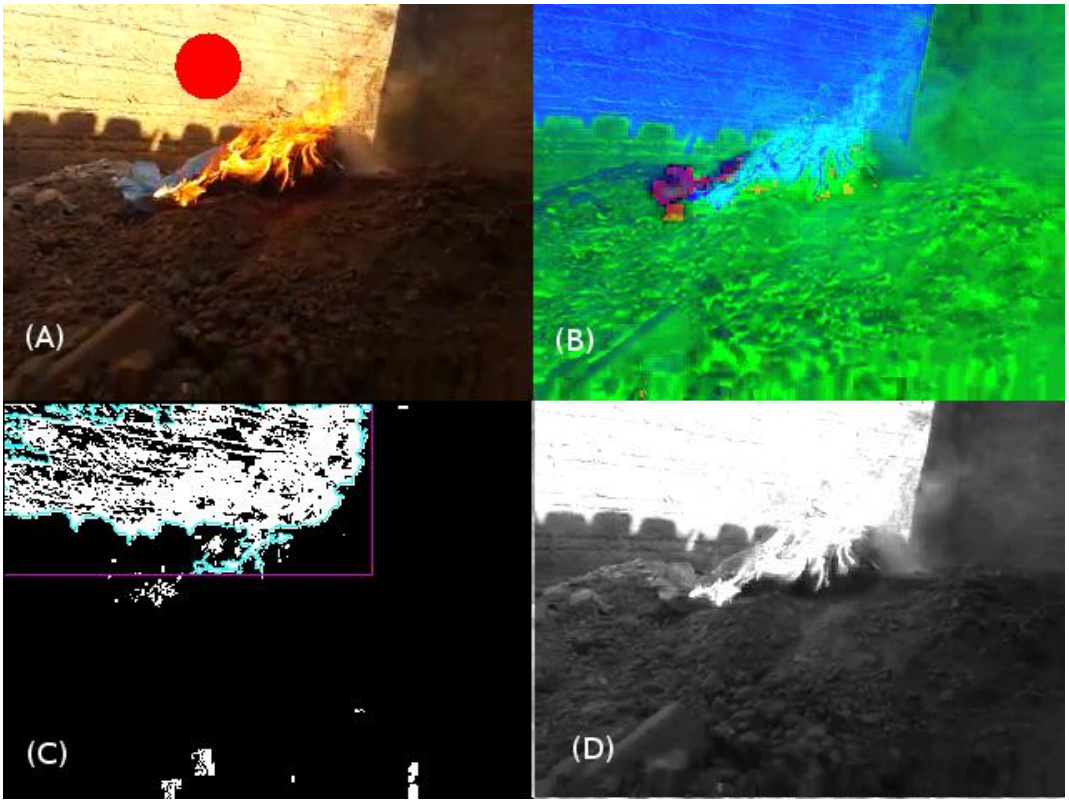

Figura 13: (A) Imagem RGB do fogo, (B) Imagem HSV, (C) Imagem binarizada detectando bordas e (D) Imagem em escala cinza. Fonte: Autor.

No entanto, o sistema comportou-se bem em termos de desempenho, uma

vez que a seleção da área na imagem pode ser feita em tempo de execução com um baixo nível de processamento. Uma vez que a imagem foi devidamente segmentada, o movimento da câmera é efetuado de forma autônoma, sempre buscando centralizar o alvo detectado na área de visualização das imagens capturadas.

Posteriormente a câmera será acoplada a um VANT para uso em missões de inspeção e reconhecimento pelo Corpo de Bombeiros e órgãos da Defesa Civil do Tocantins.

O sistema apresentou funcionamento estável e resposta satisfatória no movimento da câmera. A centralização do alvo em movimento também se mostrou bastante precisa, permanecendo focado mesmo com o alvo em movimento aleatório, não deixando escapar da área de visualização dentro dos limites dos servomotores.

O tempo de resposta do reposicionamento ficou abaixo do esperado devido à limitação dos servomotores – se o alvo se movimenta em alta velocidade, a câmera não consegue acompanhar. Levando em conta que a câmera estará acoplada ao VANT, em uma altitude média de 50m a 100m, dificilmente um alvo em movimento em terra não poderá ser seguido, uma vez que a área de abrangência da câmera aumenta significativamente.

Em testes realizados em terra com a câmera a distâncias superiores a dez metros, não houve dificuldade no seguimento dos objetos detectados, mas seria interessante a substituição dos servomotores por motores com resposta mais rápida.

É importante ressaltar que o trabalho aqui desenvolvido também resultou algumas publicações em eventos nacionais e internacionais, a saber:

- Resumo estendido submetido e aceito no VII Seminário de Informática e Tecnologia 2013, sediado na cidade Bandeiras-PR, pela Universidade Estadual do Norte do Paraná – UENP, sob o título: Câmera de VANT Controlada por Visão Computacional para Acompanhamento Automático de Alvos Móveis [31].
- Resumo estendido submetido e aceito no *Computer On The Beach 2014,* sediado na cidade de Florianópolis –SC, pela Universidade do Vale do Itajaí, sob o título: Uso de Visão Computacional para Controle Autônomo de

Câmera Embarcada em Veículo Aéreo não Tripulado [32].

- Artigo Científico submetido e aceito no INTERCON 2014, sediado na cidade de Arequipa – Peru, pela Universidade Católica San Pablo, sob o título: Câmera de VANT Controlada a Partir da Detecção Automática de Objetos Utilizando Visão Computacional [33].

Por último, após a publicação do artigo no INTERCON 2014, o mesmo foi selecionado para submissão na revista DYNA, onde foi realizado formalmente um convite para uma versão atualizada do tema. Como solicitado, o artigo foi submetido para aprovação e encontra-se aguardando as devidas considerações e avaliação por parte da comissão organizadora do evento.

DYNA é uma revista de grande prestígio internacional, publicada pela Faculdade de Minas, Universidade Nacional da Colômbia, Campus Medelín dês de 1933. É qualificada pela CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) como QUALIS B2 de acordo com o nível de qualidade dos artigos científicos publicados.

Concomitantemente, diante das eventuais possibilidades e por atender diretamente as necessidades da atividade com a solução proposta neste trabalho, a Defesa Civil Estadual se propôs a adquirir um VANT para auxiliar nas pesquisas e que possa ser utilizado para o controle no combate às queimadas no Tocantins.

7. CONCLUSÃO

O presente trabalho teve como objetivo apresentar uma solução para rastreamento automático de objetos de interesse em uma imagem digital utilizando como referência a extração das características das cores dos *pixels*.

A biblioteca *OpenCV* é uma grande aliada e, por ser de código aberto, torna-se um recurso acessível para quem queira aprofunda-se nos estudos voltados para processamento de imagens e visão computacional – ainda há muito a ser explorado nessa área. O arcabouço de funcionalidades que a biblioteca da *Intel* possui é imenso, permitindo aos pesquisadores direcionarem seus esforços para diferentes contextos de aplicabilidade e obter resultados inovadores neste segmento de pesquisa.

A utilização de VANTs tem crescido muito nos últimos anos e por ser um equipamento acessível, de baixo custo, podendo ser baseado em tecnologia de *hardware* livre como o *Arduino*, já muito utilizado nos meios acadêmicos para estudos voltados para a automação e robótica, torna-se viável explorar diversas funcionalidades como, por exemplo, o emprego da visão computacional.

Nesse trabalho, primou-se pela utilização de ferramentas exclusivamente *opensource* (código aberto), desde o sistema operacional até o hardware utilizado, que leva o conceito de *hardware* livre. As ferramentas estão disponíveis para download nos respectivos sites dos seus desenvolvedores e o *hardware* é de baixo custo, sendo disponibilizado por diversos revendedores na internet, tornando fácil e financeiramente acessível a sua aquisição.

Para a solução do que foi proposto nesse trabalho, a utilização do *openFrameworks* foi fundamental, tanto por se tratar de código aberto e estar disponível para a plataforma Linux, quanto pelo fato de ser possível fazer a integração entre *Arduino*, Câmera e *OpenCV*, graças a utilização das várias bibliotecas desenvolvidas por meio de um trabalho colaborativo de diversos desenvolvedores engajados no projeto.

8. TRABALHOS FUTUROS

Uma proposta para pesquisas futuras é focar a segmentação para formas específicas, treinar o equipamento para identificar padrões em uma imagem, cujo contexto vá além do padrão de cores.

Diversas técnicas podem ser tratadas para tal finalidade, uma delas, cuja pesquisa já foi iniciada, mas encontra-se em vias de testes e aquisição de um banco de imagens satisfatório, é conhecida como treinamento de cascata de classificadores. O apêndice A relata o que vem a ser a cascata de classificadores e a forma que o treinamento é realizado com o auxílio de ferramentas nativas ao pacote *OpenCV*.

Outra possibilidade de trabalhos futuros é explorar a utilização da visão computacional para a análise biométrica para utilizar os VANTs em uma espécie de sistema vigilante, para isso será necessário que o treinamento da cascata de classificadores seja finalizado e obtenha resultados satisfatórios para rastrear a fisionomia – detalhes dos rostos humanos.

Outra questão a ser sanada para trabalhos futuros é a aquisição de um equipamento que permita a captura de imagens com maior qualidade, uma vez que imagens com maior resolução trazem informações indispensáveis no que tange a análise biométrica.

9. REFERENCIAS

[1] V. H. Furtado, R. A. V. Gimenes, J. B. Camargo Jr e J. R. De Almeida Jr. "Aspectos de Segurança na Integração de Veículos Aéreos não Tripulados (VANT) no Espaço Aéreo Brasileiro," Simpósio de Pesquisa em Transporte Aéreo, 2008, Rio de Janeiro. Anais do VII SITRAER. Rio de Janeiro: E-apers, 2008.

[2] J. D. A de Sousa. "*DevelopmentofUnmannedAerial Four-Rotor Vehicle,*" Dissertação de Mestrado Integrado em Engenharia Electrotécnica e de Computadores Major Automação. Faculdade de Engenharia da Universidade do Porto, Portugal: 2011.

[3] E. Pastor, J. Lopes, e P. Royo. "*UAV Payload and Mission Hardware/Software Architecture*," IEEE A&E Systems Magazine.June 2007.

[4] M. Morengoni e D. Stringhini. "Tutorial: Introdução à Visão Computacional usando OpenCV," Universidade Presbiteriana de Mackenzie, Faculdade de Computação e Informática e Pós Graduação de Engenharia Elétrica. 2009.

[5] F. A. Cavani. "Análise de cenas pomares de laranjeiras através de segmentação de imagens e reconhecimento de padrões," Dissertação (Mestrado) ¨C Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2007.

[6] M. Banzi. "*Getting started with Arduino,"* O´Reilly & Associates, 2007.

[7] ERUS - Equipe de Robótica da UFES. "Minicurso Arduino," 2012. Disponível em: http://www.inf.ufes.br/~erus/arquivos/, Acessado em: 14 de Abril de 2014.

[8] E. T. F. Dias, H. V. Neto e J. F. C. Nunes. "Localização Visual de Objetos Utilizando uma Cabeça Robótica com Visão Estéreo." In Anais do XIV Seminário de Iniciação Científica e Tecnológica da UTFPR, Universidade Tecnológica Federal do Paraná, Pato Branco, Brazil, 2009.

[9] W. T. Silva, D. J. Moura, I. de A. Naas, A. S. Mendes e K. A. O. De Lima. "Estimativa do Bem-Estar de Leitões Utilizando a Visão Computacional." Faculdade de Engenharia Agrícola da Universidade de Campinas. Revista Brasileira de Agroinformática. Campinas-PR. 2004.

[10] M. Rudek, L. dos S. Coelho e O. Conciglieri Jr. "Visão Computacional Aplicada a Sistemas Produtivos: Fundamentos e Estudo de Caso." Pontífica Universidade Católica do Paraná - PUCPR/LAS/CCET. Curitiba ¨C PR. 2001.

[11] M. G. Quiles e R. A. F. Romero. "Um Sistema de Visão Computacional Baseado em Cores Aplicado ao Controle de um Robô Móvel." Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo. IV Congresso Brasileiro de Computação ¨C CBComp - São Paulo - SP. 2004.

[12] J. A. Ross. "*Computer vision and target localization: algorithms for autonomous unmanned aerial vehicles*," Dissertação de Mestrado, Departamento de Engenharia Aeroespacial, Universidade do Estado da Pennsylvania, 2008.

[13] B. Ludington, E. Johnson e G. Vachtsevanos. "*Vision-Based Navigation and Target Tracking for Unmanned Aerial Vehicles*," IEEE Robotics & Automation Magazine, ed. Set. 2006, p.63-71.

[14] W. O. Teixeira, M. G. Soares e I. YEPES. "Estudo de Sensores de Temperatura e Umidade de Baixo Custo para Aplicação em Veículo Aéreo não Tripulado." In: Computer on The Beach, 2014, Florianópolis. Computer on The Beach 2014. Florianópolis/SC: UNIVALI, 2014.

[15] B. Morais, A. A. C. Silva, O. K. O Camargo e I. Yepes. "Protótipo de Veículo Aéreo Não Tripulado para Monitoramento Ambiental e Apoio a Missões USAR." In: Computer on The Beach, 2014, Florianópolis. Computer on The Beach 2014. Florianópolis /SC: UNIVALI, 2014.

[16] C. S. Morais e A. Coelho. "Desenvolvimento de Sistema de Controle para Veículo Aéreo Não Tripulado (VANT) por Meio de Dispositivos Móveis." In proceeding of: Computer on The Beach 2014. Florianópolis/SC: UNIVALI, 2014.

[17] RODRIGO ALVES, Dorirley. Avaliação dos Modelos de Cores RGB e HSV na Segmentação de Curvas de Nível em Cartas Topográficas Coloridas.Pontifícia Universidade Católica de Minas Gerais ¨C PUC Minas, Belo Horizonte - MG, 2010.

[18] Paul Viola, Michael Jones. *Rapid Object Detection using a Boosted Cascade of Simple Features*. Conference on Computer Vision and Pattern Recognition (CVPR), 2001, pp. 511-518.

[19] Alexander Kuranov, Rainer Lienhart, and VadimPisarevsky. *An Empirical Analysis of Boosting Algorithms for Rapid Objects With an Extended Set of Haar-like Features*. Intel Technical Report MRL-TR, 2002.

[20] Mehner, Robin. Train Your Own OpenCVHaar Classifier. Conding Robin. 2013.

[21] Gomes Moreira, Gustavo Costa. Reconhecedor de Objetos em Vídeos Digitais para Aplicações Interativas. Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio. Rio de Janeiro - RJ. 2008.

[22] Shapiro, Linda; STOCKMAN George. Computer Vision. 1aed. Prentice Hall, 2001.

[23] Bradisk, Gary; KAEHLER, Adrian. Learning OpenCV ¨C Computer Vision WithOpenCV Library. 1aed. O´Reilly Media, 2008.

[24] Szelisk, Richard. Computer Vision:Algorithms and Applications. New York:Springer-Verlag. 2010.

[25] Scuri, Antônio Escño. Fundamentos da Imagem Digital. Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio. Rio de Janeiro – RJ. 2002.

[26] Oliveira Nascimento, Débora Natália. Classificação ADABOOST para Detecção e Contagem Automática de Plaquetas. Escola Politécnica de Pernambuco – Universidade de Pernambuco. Recife – PE. 2001.

[27] Almeida, Antônio Bittencourt. Usando o Computador para Processamento de Imagens Médicas. Volume 01, Número 6. 1998. Disponível em: http://www.informaticamedica.org.br/informaticamedica/n0106/imagens.htm, Acessado em: 12 de agosto de 2014.

[28] CEPSRM – Centro Estadual de Pesquisas Em Sensoriamento Remoto. Página Dinâmica para Aprendizado do Sensoriamento Remoto. Disponível em:http://www.ufrgs.br/engcart/PDASR/formcor.html, Acessado em: 05 de agosto de 2014.

[29] Sourceforge. MissionPackage. 2009. Disponível em: http://usarsim.sourceforge.net/wiki/index.php/7._Mission_package, Acessado em: 20 de março de 2014.

[30] RCGroups.Pan-Tilt with servo for Boscam HD19 Explorer Full HD 1080p FPV Camera. 2013. Disponível em: http://www.rcgroups.com/forums/showthread.php?t=1872540, Acessado em: 20 de março de 2014.

[31] Paiva, João Paulo; Yepes, Igor. Câmera de VANT Controlada Por Visão Computacional Para Acompanhamento Automático de Alvos Móveis. VII Seminário de Informática e Tecnologia – Universidade Estadual do Norte do Paraná – UENP. Bandeirantes - PR. 2013.

[32] Paiva, João Paulo; Yepes, Igor. Uso de Visão Computacional para Controle Autônomo de Câmera Embarcada em Veículo Aéreo não Tripulado. *Computer On The Beach* – Universidade do Vale do Itajaí – UNIVALI. Florianópolis – SC. 2014.

[33] Paiva, João Paulo et al. Câmera de VANT Controlada a Partir da Detecção Automática de Objetos Utilizando Visão Computacional. INTERCON 2014. Universidade Católica San Pablo. Arequipa - Peru. 2014.

APÊNDICE A :Treinamento de Cascata de Classificadores

Dentre as várias ferramentas que o *OpenCV* provê, no pacote de instalação estão algumas funções e programas para realizar o treinamento da cascata de classificadores que possibilitam realizar a detecção de qualquer objeto que se queira. Essa técnica foi proposta por Viola e Jones (2001) como uma solução genérica para a detecção de objetos em tempo real, possuindo uma taxa pequena de falsos positivos [21].

O conceito de funcionamento da cascata de classificadores, se baseia em procurar as principais características obtidas através de um descritor previamente treinado por meio do algoritmo de aprendizado presente no pacote do *OpenCV*. A grande vantagem dessa técnica é que a análise é realizada segmentando-se a imagem em blocos com formatos retangulares, também conhecidos como características *Haar*. Assim, o objeto alvo será detectado na medida em que tais características estiverem presentes, tornando o processo muito mais rápido do que a análise baseada estritamente em *pixels*, uma vez que o número de características a serem analisadas é bem inferior, reduzindo o custo de processamento [21].

Duas funções são encontradas no pacote *OpenCV* para realizar o treinamento: *OpenCV-TrainCascade* e *OpenCV-HaarTraininng*. A diferença entre a duas é que a primeira é uma evolução da segunda, sendo escrita na linguagem C++ em conformidade com a API 2.0 do *OpenCV*. A nova versão também possibilita trabalhar com *multi-threading*, tornando-se muito mais rápida do que *OpenCV-HaarTraining*. Após o treinamento, as duas funções geram arquivos de saída diferentes, embora *OpenCV-TrainCascade* suporte ambos os formatos, podendo salvar a saída no formato da versão anterior.

O treinamento é realizado com base em um conjunto de imagens que contenham somente o objeto a ser rastreado pelo sistema, também chamadas de imagens positivas, e outro conjunto de imagens chamadas de imagens negativas, composta por imagens aleatórias com diferentes planos de fundos sem que contenham o objeto de interesse. Essa é uma etapa anterior ao treinamento, que corresponde à seleção do que se deseja detectar a partir das imagens que serão carregadas para as funções supracitadas. Isso tudo é realizado com o auxílio de

outra função da API do *OpenCV* chamada de *OpenCV-CreateSamples*.

O *OpenCV-CreateSamples* prepara o conjunto de imagens (amostras) de forma que a saída desta função gera um arquivo na extensão “*.vec” contendo as imagens selecionadas. Esse arquivo é suportado por ambas as funções de treinamento da cascata de classificadores mencionadas anteriormente. Kuronav et. al. (2002) apud Robin [20] recomenda o uso de amostras com resoluções de 20x20 *pixels*, pois nessa resolução é possível atingir uma maior taxa de acertos. Além disso, deve-se levar em consideração que à medida que a resolução aumenta, dependendo do equipamento utilizado, torna-se inviável realizar o processamento em virtude do custo de memória e processamento, exigindo assim um computador mais robusto para efetuar o treinamento [20].

Foram realizados testes de treinamento com 258 (duzentos e cinquenta e oito) imagens positivas, ou seja, somente imagens contendo o objeto de interesse (nesse caso o fogo) e 501 (quinhentas e uma) imagens negativas, imagens aleatórias que não contenham o objeto a ser detectado. De acordo com Robin [20], foram gerados dois arquivos com a lista de imagens positivas e negativas, para dar subsídio para a função que irá criar as amostras cuja saída é um conjunto de arquivos na extensão “.vec”, que posteriormente foi carregado para a função *OpenCV-haartraininng*.

O treinamento mostrou-se demasiadamente dispendioso em termos de processamento, com tempo de duração prolongado até a sua finalização e geração do arquivo no formato XML, produto este resultado do treinamento que servirá para que a função do *openFrameworks*, *ofxCvHaarFinder*, tenha condições de rastrear o objeto previamente treinado. Foram feitos testes com imagens em tamanhos superiores a vinte *pixels* de largura e altura, mas sempre que eram executados, erros de alocação de memória eram retornados, provando que imagens superiores a 20x20 *pixels* não são apropriadas para o treinamento do *haartraining*.

Feito o treinamento, o resultado não se mostrou satisfatório em virtude da complexidade do objeto a ser detectado, uma vez que o objetivo proposto é a detecção de chamas, demonstrando que a quantidade de amostras é insuficiente. Há também que se rever a qualidade de dessas amostras, uma vez que tais

imagens devem conter somente o objeto que se deseja, o que dificulta um pouco no caso do fogo, devendo ser tratada imagem por imagem.

Como pode ser observado na Figura 14, a partir da cascata de classificadores que foi gerada após o treinamento, o programa não consegue segmentar somente o objeto alvo, demais áreas da imagem são selecionadas gerando uma série de falsos positivos, uma vez que somente o fogo deveria ser segmentado.

Figura 14: Fogo sendo rastreado a partir da Cascata de Classificadores. Fonte: Autor

Diante do que foi verificado, o treinamento da cascata de classificadores é bastante trabalhoso no que se refere ao tempo gasto para se realizar todas as etapas, que compreendem a aquisição de um banco de imagens que contenha um número significativo de amostras positivas e negativas, levando-se em consideração a qualidade das imagens positivas, e o tempo de processamento propriamente dito que o programa leva até a geração do produto do treinamento, ou seja, o arquivo no formato XML contendo a cascata de classificadores.