Diogo Miguel Abadia Boto
Licenciado em Ciências de Engenharia Eletrotécnica e de Computadores

Avaliação da intensidade de um fogo florestal usando processamento de imagem

Dissertação para obtenção do Grau de Mestre em Engenharia Eletrotécnica e de Computadores

Orientador: Doutor André Teixeira Bento Damas Mora, Professor Auxiliar, FCT-UNL

Júri:

Presidente: Doutor Luís Augusto Bica Gomes de Oliveira, FCT/UNL

Arguente: Doutor Filipe de Carvalho Moutinho, FCT-UNL

Vogal: Doutor André Teixeira Bento Damas Mora, FCT-UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2019

Avaliação da intensidade de um fogo florestal através do processamento imagem

Copyright © Diogo Miguel Abadia Boto, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa.

A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Aos meus avós...

Agradecimentos

A conclusão de um mestrado em engenharia, é um percurso repleto de desafios, percalços, alegrias, tristezas, mas sobretudo de conquistas. Ainda que seja um percurso por vezes solitário, é um caminho repleto de pessoas que me ajudaram a crescer.

Assim, queria agradecer à Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa, a oportunidade de estudar e me formar nesta instituição. Ao meu orientador, Professor André Mora, agradeço pela prontidão com que me ajudou a encontrar um tema para a dissertação, bem como o apoio prestado ao longo do seu desenvolvimento. Aos professores que me acompanharam ao longo deste percurso, um bem-haja por todas as lições que me fizeram crescer a nível pessoal e académico.

Á minha família, obrigado pelo apoio incondicional que me deram ao longo destes anos e pela possibilidade que me deram de escolher e frequentar o curso que eu queria. Aos meus pais, muito obrigado! Sem o vosso apoio não teria sido possível a conclusão do meu percurso académico. Adoro-vos, independentemente disso!!

Aos meus colegas de curso, foi um prazer ter conhecido gente tão diversa e espetacular. Ao Duarte, ao "Gonki", ao Ivo, ao Neves, ao "Pidoca", ao Valentim, ao Teixeira, e a todos os outros – "Obrigado por esta aventura! Que a próxima aventura tenha pessoas tão fantásticas quanto vocês foram!". Ao Xico e ao Varandas – Não há como agradecer, a não ser – "Obrigado pelo jantar! Para o próximo Mestrado, pago eu!!"

Aos meus colegas de faculdade e amigos. À Tita, à Ritinha, à Beatriz, ao Tiago, à Maria, obrigado por estarem lá nos momentos mais importantes. Obrigado pelas semanas académicas, pelo traçar da capa, pelas serenatas. Esses são momentos que nunca esquecerei. Um especial obrigado ao David, pela ajuda prestada.

A todos os meus colegas e amigos – Obrigado por tudo: obrigado pelas tardes no tico e pelas noitadas no 7. Obrigado pelas festas, pelas viagens, pelas saídas, pelas alegrias e tristezas. Espero que continuem ao meu lado durante muitos anos.

Resumo

Há décadas que Portugal se tem debatido com o problema dos incêndios florestais. Estas catástrofes tiveram especial impacto nos últimos anos em que as perdas humanas e patrimoniais alcançaram recordes. Face ao aquecimento global, estas calamidades têm tendência para continuar e mais que a necessidade de prevenir, é preciso aprender a lidar melhor com o fogo. Para isso, é importante saber mais sobre os incêndios, de forma a melhor prever o seu comportamento.

Nesta dissertação, é proposto um estudo das características visuais do fogo, com o objetivo de conseguir não só detetar incêndios, mas também extrair características de propagação dos mesmos. Assim, foi desenvolvido um protótipo móvel, de baixo custo, de um sistema de visão estereoscópica, capaz de fazer a deteção de fogo e extrair características físicas da frente de fogo. Através do processamento digital de imagem, foram analisadas as imagens e aplicandos algoritmos de deteção de fogo e correspondência estereoscópica nas gamas espectrais do visível e do infravermelho próximo, de forma a obter um sistema preciso e eficiente.

Para validar o sistema de visão estereoscópica desenvolvido, foi realizado um conjunto de testes de hardware e software, e analisada a sua performance, alcance e precisão. Os algoritmos de deteção de fogo nos espectros do visível e do infravermelho próximo, apresentam, respetivamente, percentagens de erro de 0,4% e 5%. Juntando as deteções feitas em ambos os espectros, e através de correspondência estereoscópica de segmentos de fogo, foi possível aumentar a precisão do sistema, obtendo-se taxas de erro de 0,4%. Foi também analisado o algoritmo de extração de características, onde se verificou que as limitações de alcance do sistema, afetam a exatidão no cálculo de distâncias.

Palavras-chave: deteção de fogo; visão estereoscopia; incêndios; infravermelho próximo; processamento de imagem; RGB; NIR;

Abstract

For many decades, Portugal has faced the problem of wildfires. These natural disasters were especially ruinous the last few years, when life and patrimonial losses reached records. Facing the global warming, these calamities have no tendency to slow down. Now, more than the necessity of prevention, there's the need to learn how do better deal with wildfires. Therefore, it's important to know more about fires to better predict its behaviour.

This dissertation proposes a study of the visual features of fire, with the objective of not only detecting wildfire but classifying it as well. As such, it was developed a mobile prototype of a low-cost stereoscopic vision system, capable of detecting and extracting features of fire fronts. Using digital image processing, captures in the spectral bands of visible and near-infrared were analysed and the detections in both spectrums combined. This way an efficient algorithm was obtained, capable of detecting and stereo-matching fire segments in both images.

To validate the prototype of the stereoscopic vision system, a set of software and hardware tests was done, analysing its performance, working range, and precision. The fire detection algorithms, for visible and near-infrared spectrums, were able to run with a error rate of 0,4% and 5%, respectively. With the merging of the detections made by both algorithms, and using stereo-matching of fire segments, the precision of the system was increased, reducing the error rate 0,4%. In addition, the feature extraction was tested, and it was concluded that the range limitations of the system decreased the depth calculation accuracy.

Keywords: fire detection; stereoscopic vision; wildfire; near-infrared; digital image processing; RGB; NIR;

Conteúdo

1 INTRODUÇÃO 1
1.1 CONTRIBUIÇÕES 3
1.2 ESTRUTURA DO DOCUMENTO 4

2 ESTADO DA ARTE 5
2.1 IMAGEM ESTEREOSCÓPICA 5
2.1.1 Calibração de Câmara 6
2.1.2 Retificação pelo Algoritmo de Hartley 8
2.1.3 Deteção de vértices e arestas 10
2.1.4 Extrapolação de distâncias 17
2.2 DETEÇÃO DE FOGO 20
2.2.1 Deteção de fogo em RGB e YCbCr 22
2.2.2 Deteção de Fogo em YCbCr e CIELab 25
2.2.3 Deteção de fogo em CIELab com base no movimento 28
2.2.4 Deteção de fogo em NIR 32
2.2.5 Comparação da classificação nas bandas RGB e NIR 34
2.3 CLASSIFICAÇÃO DE INCÊNDIOS 36

3 TRABALHO DESENVOLVIDO 41
3.1 DESENVOLVIMENTO DO PROTÓTIPO 42
3.1.1 Hardware utilizado 43
3.1.2 Construção do protótipo 47
3.2 DESENVOLVIMENTO DO SOFTWARE 50
3.2.1 Bibliotecas utilizadas 50
3.2.2 Calibração do sistema 59

3.2.3 Recolha de Dados ... 62
3.2.4 Deteção de Fogo ... 68
3.2.5 Extração de características ... 76
3.3 INTERFACES DESENVOLVIDOS ... 79
4 RESULTADOS ... 81
4.1 LIMITAÇÕES DA VISÃO ESTEREOSCÓPICA ... 82
4.2 BASE DE DADOS PARA TESTE ... 84
4.3 RESULTADO DOS ALGORITMOS ... 85
4.3.1 Calibração e Retificação ... 85
4.3.2 Deteção na câmara RGB ... 86
4.3.3 Deteção na câmara NIR ... 88
4.3.4 Correspondência estereoscópica de segmentos ... 90
4.3.5 Extração de Características ... 92
5 CONCLUSÃO E TRABALHO FUTURO ... 95
5.1 TRABALHO FUTURO ... 97
6 REFERÊNCIAS ... 99
7 ANEXOS ... 103

Lista de Figuras

FIGURA 1.1 - ÁREA ARDIDA EM PORTUGAL NOS ANOS 2005, 2013 E 20173 2
FIGURA 2.1 - CALIBRAÇÃO DA CÂMARA 6
FIGURA 2.2 - ESQUEMA DE VISÃO ESTEREOSCÓPICA 8
FIGURA 2.4 - PAR DE IMAGENS APÓS RETIFICAÇÃO DE HARTLEY 9
FIGURA 2.3 - PAR DE IMAGENS ESTEREOSCÓPICAS 9
FIGURA 2.5 - EFEITO DO DESVIO DE UMA JANELA CENTRADA NUM VÉRTICE 10
FIGURA 2.6 - INTERPRETAÇÃO DOS DIFERENTES VALORES PRÓPRIOS FACE A DETERMINADAS CARACTERÍSTICAS NUMA IMAGEM 12
FIGURA 2.8 - ALGORITMO DE HARRIS 16
FIGURA 2.7 - ALGORITMO APRESENTADO EM (ACHARD, BIGORGNE E DEVARS, 2000), ... 16
FIGURA 2.9 - FILTROS DE LAPLACE DISCRETOS 16
FIGURA 2.10 - ESQUEMA DE VISÃO ESTEREOSCÓPICA 18
FIGURA 2.11 - IMAGEM DOS MARCADORES 19
FIGURA 2.12 - RESPOSTA ESPECTRAL DO SENSOR SONY IMX-219 20
FIGURA 2.13 – RESPOSTA ESPECTRAL DA RADIÂNCIA EM FUNÇÃO DA TEMPERATURA 21
FIGURA 2.14 – ESQUEMA DE FUNCIONAMENTO DO SISTEMA DE DETEÇÃO DE FOGO 22
FIGURA 2.15 - CURVA ROC PARA DETEÇÃO DO TH ÓTIMO 23
FIGURA 2.16 – ESQUEMA DE FUNCIONAMENTO DO ALGORITMO DE DETEÇÃO DE FOGO 25
FIGURA 2.17 - RESULTADOS DO ALGORITMO (CHEN, LUO E ALSHARIF, 2013) 27
FIGURA 2.18 – ESQUEMA DE FUNCIONAMENTO DO ALGORITMO DE DETEÇÃO DE FOGO EM *CIELAB* 28
FIGURA 2.19 - HISTOGRAMA DAS PROBABILIDADES $\boldsymbol{P}(\boldsymbol{L}*, \boldsymbol{a}*)$, $\boldsymbol{PL}*, \boldsymbol{b}*$ E $\boldsymbol{P}(\boldsymbol{a}*, \boldsymbol{b}*)$ 29
FIGURA 2.20 - CURVA ROC E RESPETIVOS VALORES $\boldsymbol{\alpha}$ 29
FIGURA 2.21 - VARIAÇÃO TEMPORAL DO NIVEL NIR 32
FIGURA 2.22 - VARIAÇÃO DE $\boldsymbol{An}$ EM FUNÇÃO DO VALOR DE N 33
FIGURA 2.23 - RESULTADO DO ALGORITMO (LEÑERO-BARDALLO *ET AL.*, 2013) 35
FIGURA 2.24 - RESULTADOS ESTATÍSTICOS DE 11 PILHAS DE LENHA 35

FIGURA 2.25 - CATEGORIAS DE INCÊNDIO FLORESTAL....36
FIGURA 2.26 - DIAGRAMA DE VAN WAGNER....37
FIGURA 2.27 - ESQUEMA DE CARACTERÍSTICAS DE UMA CHAMA....39
FIGURA 3.2 - VISÃO DIANTEIRA DA PRIMEIRA VERSÃO DO PROTÓTIPO....42
FIGURA 3.1 - FACE TRASEIRA DA PRIMEIRA VERSÃO DO PROTÓTIPO....42
FIGURA 3.3 - MICROCOMPUTADOR RASPBERRY PI 3B+....43
FIGURA 3.4 - RESPOSTA ESPECTRAL DA CÂMARA RGB....44
FIGURA 3.5 – RESPOSTA ESPECTRAL DA CÂMARA NIR....44
FIGURA 3.6 - MULTIPLEXADOR *IVPORT V2*....45
FIGURA 3.7 - ECRÃ TÁTIL *RASPBERRY PI*....46
FIGURA 3.8 – ESQUEMA DE CORTE DA ESTRUTURA....47
FIGURA 3.9 - FIXAÇÃO DA CÂMARA NO PROTÓTIPO....48
FIGURA 3.10 - FACE TRASEIRA DO PROTÓTIPO....49
FIGURA 3.11 - FACE FRONTAL DO PROTÓTIPO....49
FIGURA 3.12 – VISTA INTERIOR DO PROTÓTIPO COM A FACE FRONTAL ABERTA....49
FIGURA 3.13 - IMAGEM CAPTURADA A 1920X1080....52
FIGURA 3.14 - INTERFACE DO ALGORITMO DE CALIBRAÇÃO DA VISÃO ESTEREOSCÓPICA....61
FIGURA 3.15 - IMAGEM DE UMA FOGUEIRA A 20 METROS....62
FIGURA 3.17 - FOGUEIRA CAPTURADA A 15 METROS COM A CÂMARA NIR....63
FIGURA 3.16 – FOGUEIRA CAPTURADA A 15 METROS COM A CÂMARA RGB....63
FIGURA 3.18 - ESQUEMA DE CAPTURA DA FOGUEIRA UTILIZADO NAS RECOLHAS 2 E 3....64
FIGURA 3.19 – SECÇÃO DE UMA IMAGEM CAPTURADA A 10 METROS COM A CÂMARA NIR....65
FIGURA 3.20 - SECÇÃO DA IMAGEM CAPTURADA A 10 METROS COM A CÂMARA RGB....65
FIGURA 3.22 – IMAGEM DA FOGUEIRA, OBTIDA A 25 METROS NA TERCEIRA RECOLHA, ATRAVÉS DA CÂMARA RGB....66
FIGURA 3.21 – IMAGEM DA FOGUEIRA, OBTIDA A 25 METROS NA TERCEIRA RECOLHA, ATRAVÉS DA CÂMARA NIR....66
FIGURA 3.23 - DETEÇÃO DE FOGO COM MUITA SEGMENTAÇÃO....68
FIGURA 3.24 – ESQUEMA DE FUNCIONAMENTO DO ALGORITMO DE DETEÇÃO RGB....71
FIGURA 3.25 – CANAL VERMELHO DA IMAGEM NIR....72
FIGURA 3.26 – CANAL VERDE DA IMAGEM NIR....72
FIGURA 3.27 – CANAL AZUL DA IMAGEM NIR....72
FIGURA 3.28 - ESQUEMA DE FUNCIONAMENTO DO ALGORITMO DE DETEÇÃO NIR....73
FIGURA 3.29 – ESQUEMA DE FUNCIONAMENTO DO ALGORITMO DE CORRESPONDÊNCIA DE SEGMENTOS....75
FIGURA 3.30 - INTERFACE DE DETEÇÃO DE FOGO....79
FIGURA 4.1 - REPRESENTAÇÃO GRÁFICA DO ERRO NO CÁLCULO DA DISTÂNCIA, EM FUNÇÃO DA DIFERENÇA DE PIXÉIS DE UMA IMAGEM PARA OUTRA....82
FIGURA 4.2 - ALCANCE MÁXIMO EM FUNÇÃO DA RESOLUÇÃO E DISTÂNCIA ENTRE CÂMARAS....83
FIGURA 4.3 – IMAGEM RGB....84
FIGURA 4.4 - MÁSCARA DE DETEÇÃO RELATIVA À FIGURA 4.3....84
FIGURA 4.5 – IMAGEM RGB NÃO RETIFICADA....85
FIGURA 4.6 – IMAGEM NIR NÃO RETIFICADA....85
FIGURA 4.7 – IMAGEM NIR RETIFICADA....85
FIGURA 4.8 – IMAGEM RGB RETIFICADA....85
FIGURA 4.9 – DETEÇÃO RGB FEITA À FIGURA 4.9....86

FIGURA 4.10 – CAPTURA A 25 METROS COM A CÂMARA RGB.....86
FIGURA 4.11 – DETEÇÃO RGB FEITA À FIGURA 4.11.....86
FIGURA 4.12 – CAPTURA A 5 METROS COM A CÂMERA RGB.....86
FIGURA 4.13 – CAPTURA A 5 METROS COM A CÂMARA NIR.....88
FIGURA 4.14 - DETEÇÃO NIR FEITA À FIGURA 4.13.....88
FIGURA 4.15 – CAPTURA A 25 METROS COM A CÂMARA NIR.....89
FIGURA 4.16 – DETEÇÃO NIR FEITA À FIGURA 4.15.....89
FIGURA 4.17 – CORRESPONDÊNCIA ESTEREOSCÓPICA APLICADA À FIGURA 4:8.....90
FIGURA 4.18 – CORRESPONDÊNCIA ESTEREOSCÓPICA APLICADA À FIGURA 4:14.....90

Lista de Tabelas

TABELA 2.1 - MATRIZ DE CONFUSÃO DO CLASSIFICADOR 24
TABELA 2.2 - VALORES ÓTIMOS DA EQUAÇÃO (24) 26
TABELA 3.1 – MODOS DO SENSOR 56
TABELA 4.1 – MATRIZ DE CONFUSÃO DO ALGORITMO DE DETEÇÃO RGB 87
TABELA 4.2 - MATRIZ DE CONFUSÃO DO ALGORITMO DE DETEÇÃO NIR 89
TABELA 4.3 - MATRIZ CONFUSÃO DA DETEÇÃO ESTEREOSCÓPICA, APÓS DETEÇÃO EM CADA ESPECTRO RGB E NIR. 91
TABELA 4.4 – RESULTADOS DOS CÁLCULOS ATRAVÉS DO MENOR RETÂNGULO SEM ROTAÇÃO 92
TABELA 4.5 – RESULTADOS DOS CÁLCULOS ATRAVÉS DO CENTROIDE PESADO COM A COR 93
TABELA 4.6 - RESULTADOS DOS CÁLCULOS ATRAVÉS DO CENTROIDE 93

Siglas e Abreviaturas

⊗	–	Operador lógico *AND*
⊕	–	Operador lógico *OR*
2D/3D	–	Referente às duas e três dimensões
BD	–	"*Background difference*" – mapa de diferença do fundo da imagem
BG	–	*"Background"* - Mapa do fundo da imagem
CSI	–	*"Camera Serial Interface"* - Interface Série da Câmara
DSI	–	*"Display serial interface"* - Interface Série do *Display*
FD	–	*"Frame difference"* - Mapa da diferença binária entre *frames*
FN	–	Falsos Negativos
FP	–	Falsos Positivos
GPIO	–	"*General purpose Input/Output*" - Entradas/Saídas de uso geral
HDMI	–	*"High-definition multimedia interface"* - Interface multimédia de alta definição
IR	–	Infravermelho
MIPI	–	"*Mobile Industry Processor Interface*" – Protocolo de Interface para processadores de industria móvel
NIR	–	*"Near-infrared"* - Infravermelho próximo - referente ao tipo de câmara, imagem, gama espectral ou tipo de deteção de fogo.
RGB	–	"*Red,Green,Blue*" - Referente aos canais vermelho, verde e azul; referente ao tipo de câmara, imagem ou tipo de deteção de fogo
ROC	–	"*Receiver Operating Characteristic*" - Característica de operação do recetor
ROS	–	"*Rate of spread*" - Velocidade de propagação de fogo
SDRAM	–	"*Synchronous dynamic random-access memory*" - Memória de acesso aleatório dinâmica síncrona
SFI	–	"*Surface fire intensity*" - intensidade de superfície do fogo
USB	–	"*Universal Serial Bus*" - porta série universal
UV	–	Ultravioleta
VN	–	Verdadeiros Negativos
VP	–	Verdadeiros Positivos

1

Introdução

Desde os primórdios da Humanidade que o fogo está presente no quotidiano da sociedade. Quer seja para cozinhar, aquecimento ou como forma de produzir eletricidade, há séculos que esta reação química é utilizada nas mais variadas tarefas do dia a dia. Esta preciosa manifestação de energia tem sido moldada e aperfeiçoada às nossas necessidades, contudo a sua imprevisibilidade e "vida própria" fazem desta ferramenta uma faca de dois gumes: quer seja em acidentes ou atos premeditados, o fogo, mais concretamente os incêndios, são uma catástrofe que todos os anos fazem milhares de vítimas e levam a enormes perdas de património natural e urbano.

Todos os anos se ouve falar dos estragos provocados por esta força avassaladora, e todos os anos se reflete sobre as causas destas catástrofes, no entanto, após tantos séculos de evolução, ainda não a sabemos controlar devidamente. Os esforços para resolver este problema, têm levado ao desenvolvimento de variadas tecnologias: umas focadas no combate ao fogo (helicópteros equipados com baldes, linhas corta fogo, …); outras focadas na deteção (drones de vigilância, câmaras térmicas de longo alcance, etc).

Há décadas que Portugal se tem debatido com o problema dos incêndios florestais. Esta catástrofe teve especial impacto no ano 2017, em que arderam cerca de 520 mil hectares de floresta, dos quais um dos mais importantes patrimónios históricos e naturais, o Pinhal de Leiria. Analisando os incêndios de outubro de 2017, no espaço de dois dias foram consumidos cerca de 54 mil hectares.

Se recuarmos no tempo, verificamos outras situações de enorme gravidade (figura 1.1): em 2003 arderam 430 mil hectares, dois anos depois (2015) arderam 339 mil e em 2013 cerca de 153 mil. Associado a isto estão as perdas económicas, que no intervalo de 2005 a 2015 somaram 1 850 milhões de euros. (EFFIS, 2018) (Suspiro, 2017)

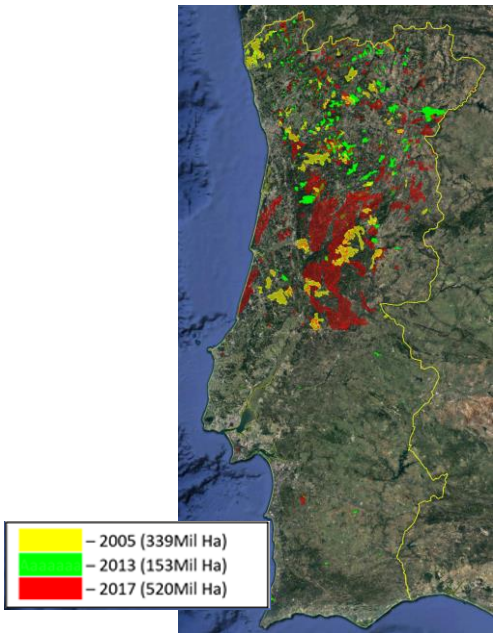

Figura 1.1 - Área ardida em Portugal nos anos 2005, 2013 e 2017

A pesar nestas avultadas perdas de património, ao longo dos anos tem-se também perdido muitas vidas: Em 2003 foram registados 20 mortos, num ano que devastou o sul de Portugal; dez anos depois, em 2013, morreram 8 bombeiros durante o combate; e no ano de 2017 mais de uma centena de pessoas morreram, muitas das quais enquanto tentavam fugir das chamas. Um dos maiores motivos para tais fatalidades reside na falta de informação e conhecimento sobre as frentes de fogo que os circundavam.

Lidar com um elemento tão poderoso e instável quanto o fogo, necessita de uma abordagem fundada. Sem a devida informação sobre as condições do fogo, tentar extingui-lo é como tentar domar um leão de olhos vendados: na falta de informação sobre as condições em que a frente se propaga, torna-se impossível criar uma estratégia de ataque, permitindo que a frente de fogo se propague e alastre.

Uma má estratégia, implica maiores riscos e maior probabilidade de propagação do fogo. Assim, esta tese foca-se em estudar e desenvolver algoritmos que possam

melhorar a informação usada pelos bombeiros no combate ao fogo. Dotados de um sistema portátil e automático, as equipas de combate ao fogo poderão, a qualquer altura, retirar mais e melhor informação sobre o fogo. Deste modo, pretende-se diminuir o número de vítimas do fogo, dando melhor informação não só aos meios de combate, mas também às populações em fuga.

Consequentemente, esta dissertação propõe-se a desenvolver um protótipo de um sistema capaz de avaliar a intensidade de um incêndio florestal, a ser integrado nas equipas de batedores que avaliam o incêndio. Ainda que as equipas de bombeiros já façam uma avaliação das frentes de fogo, esta não é a mais precisa. Como tal pretende-se também unificar a linguagem utilizada pelas equipas operacionais, e melhorar as condições no combate aos incêndios.

1.1 Contribuições

O desenvolvimento da presente dissertação permitiu o desenvolvimento de um protótipo de visão estereoscópica com o objetivo de detetar e classificar as frentes de fogo. Para isso, foi feito o estudo da visão estereoscópica, no que toca à calibração, alcance e correspondência estereoscópica. Foi também estudada a viabilidade da utilização de diferentes gamas espectrais nas câmaras da visão estereoscópica. Durante esse estudo, foi desenvolvido uma interface de calibração estereoscópica

Na área da deteção e classificação de fogo, foi feito o estudo da chama nas gamas espectrais do visível e do infravermelho próximo, a partir do qual foram desenvolvidos dois algoritmos de deteção de fogo, um para cada gama espectral. Em conjunção com esses algoritmos, foi ainda desenvolvido um algoritmo para fazer a correspondência estereoscópica entre os segmentos de fogo detetados em ambas as imagens. Em adição, ainda que os resultados não tenham sido os mais satisfatórios, foi desenvolvido um algoritmo através do qual são calculadas características do cenário de fogo, nomeadamente a distância à frente de fogo, o tamanho e a inclinação da chama.

Finalmente, os algoritmos relacionados com a deteção e classificação de fogo, foram integrados numa interface gráfica de deteção em tempo-real.

1.2 Estrutura do Documento

Este documento, encontra-se dividido em 5 capítulos: Introdução, Estado da Arte, Trabalho Desenvolvido, Resultados e Conclusão.

No Estado da Arte, é feito um estudo das tecnologias, metodologias, e algoritmos utilizados para a visão estereoscópica, deteção e classificação de fogo.

No Trabalho Desenvolvido, fala sobre o desenvolvimento do protótipo nomeadamente como foi feita a construção do protótipo físico, bem como o desenvolvimento dos algoritmos e interfaces usados na calibração, deteção e classificação.

Nos Resultados, é feita uma análise qualitativa e quantitativa dos vários algoritmos desenvolvidos ao longo da dissertação.

Nas Conclusões, é feito um resumo das elações tiradas dos resultados dos vários algoritmos, e apresentadas as conclusões sobre a qualidade e viabilidade do trabalho desenvolvido, abordando-se ainda possíveis trabalhos futuros que poderão surgir.

2

Estado da Arte

Este capítulo irá focar-se no estudo de várias tecnologias e metodologias de processamento de imagem. Para isso será feita uma síntese de trabalhos relacionados com a deteção e classificação de incêndios florestais através do processamento digital de imagem. Como tal, irão ser abordados, sequencialmente, a imagem estereoscópica, seguindo-se a deteção digital de fogo e concluindo com a classificação. Nos últimos, será feita uma análise separada aos espectros RGB e NIR de forma a melhor compreender as características de ambas as bandas.

2.1 Imagem Estereoscópica

Uma imagem estereoscópica, é uma imagem criada a partir de duas outras tiradas em diferentes pontos de vista, tendo em conta a comparação da posição relativa de objetos nas duas imagens. Para a criação dessas imagens são necessários algoritmos que façam a correspondência entre os pixéis do par de imagens, algoritmos esses que necessitam de uma grande capacidade de processamento digital de imagem. Na base da sua construção está a identificação de um mesmo objeto em ambas as imagens, de modo a corresponder os pixéis de uma imagem, aos pixéis da outra.

Em (Zou e Li, 2010) , os autores destacam que o procedimento para criar um sistema de visão estereoscópica é composto por seis partes: Captura da imagem, Calibração da Câmara, Extração de características, Correspondência Estereoscópica, Extrapolação de distâncias e *Post treatment.*

A captura de imagem, como o próprio nome indica trata-se obter imagem, através de duas câmaras. Depois através da calibração da câmara, normalmente utilizando-se uma imagem de um tabuleiro de xadrez (figura 2.1), obtêm-se as matrizes de parâmetros intrínsecos e extrínsecos da câmara. Este processo tem o objetivo de calcular matrizes que descrevam a posição da câmara e do tabuleiro no espaço, fazendo uma correspondência entre imagens 2D e 3D. Na extração de características o objetivo é tentar encontrar pixéis de interesse entre as duas imagens. Métodos conhecidos como o detetor de vértices de Harris, tentam encontrar esses tais pontos de interesse encontrando cantos nas imagens. Depois com esses dados é calculado um mapa de disparidade que permite ver as diferenças espaciais entre as duas imagens e calcular distâncias fazendo-se assim a extrapolação de distâncias. Finalmente no *Post treatment,* o objetivo é dar à visão estereoscópica a compreensão da superfície dos objetos.

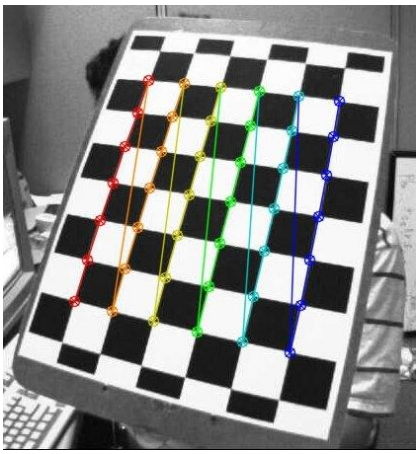

Figura 2.1 - Calibração da Câmara (OpenCV, 2019)

Para cada uma destas etapas existem diferentes métodos. Assim, serão focados os métodos de extração de características, correspondência estereoscópica e cálculo de distâncias, dada a sua importância na criação e análise de uma visão estereoscópica.

2.1.1 Calibração de Câmara

Em (Zou e Li, 2010) os autores descrevem como um sistema de visão estereoscópica implementado com auxilio das bibliotecas *OpenCV* extrapola a posições dos pontos em 3D. No sistema adotado, os autores usaram o método de Zhang Zhengyou (figura 2.1).

Este método consiste em captar imagens de um tabuleiro de xadrez com tamanho conhecido para fazer a calibração da câmara. O objetivo principal é conseguir descrever a posição das câmaras face a um dado referencial. Para isso é importante conhecer as matrizes de rotação e translação de cada uma das câmaras, respetivamente T e R. Com esses valores é então possível obter a posição de ambas as câmaras no referencial:

$$\begin{cases} \boldsymbol{P_d} = \boldsymbol{R_d P} + \boldsymbol{T_d} \\ \boldsymbol{P_e} = \boldsymbol{R_e P} + \boldsymbol{T_e} \end{cases} \quad (1)$$

Em que:

- P é o vetor posição um ponto arbitrário em comum na imagem das duas câmaras.
- P_e e P_d são respetivamente os vetores posição da câmara esquerda e direita.

Assim através da posição de um ponto arbitrário conseguimos calcular a transformação entre as duas câmaras:

$$\boldsymbol{P_1} = \boldsymbol{R^T}(\boldsymbol{P_2} - \boldsymbol{T}) \quad (2)$$

Em que:

- R e T são respetivamente as matrizes rotação e transformação entre as duas câmaras.

Ao aplicar as três equações em (1) e (2), obtemos as matrizes R e T do sistema de visão estereoscópica:

$$\begin{cases} \boldsymbol{R} = \boldsymbol{R_2 R_1 T} \\ \boldsymbol{T} = \boldsymbol{T_2} - \boldsymbol{R T_1} \end{cases} \quad (3)$$

Acontece que mesmo obtendo estes parâmetros, o sistema tem de ser retificado para o facto de as câmaras não se encontrarem alinhadas. Como tal, o objetivo é projetar ambas as imagens de forma a que as imagens estejam perfeitamente alinhadas linha a linha. Na figura 2.2 (a), podemos observar o correto alinhamento das câmaras:

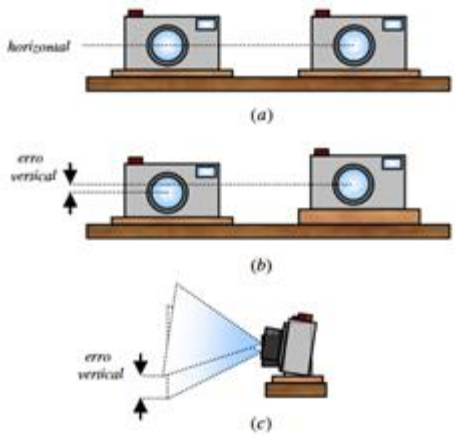

Figura 2.2 - Esquema de visão estereoscópica (Mrovlje e Vrančić, 2008)
(a) correto alinhamento; (b) e (c) câmaras desalinhadas

Os autores, identificam dois métodos de retificação muito conhecidos na literatura: o algoritmo de Hartley (Hartley, 1998) e o de Bouguet (Bouguet, 2015). É importante referir que no sistema estereoscópico os autores usam ambos os métodos. O algoritmo de Bouguet é usado quando a calibração é conhecida, pois este algoritmo usa as matrizes calculadas durante a calibração para minimizar as transformações necessárias para se passar de uma imagem para a outra. O algoritmo de Hartley é usado quando não se conhece o resultado da calibração das câmaras. Em seguida é apresentado esse algoritmo.

2.1.2 Retificação pelo Algoritmo de Hartley

O objetivo do algoritmo de Hartley (Hartley, 1998) consiste em transformar duas imagens de pontos de vista distintos, de modo a que ambas as imagens estejam alinhadas linha a linha. Ainda que no fim continue a haver disparidades ao longo de cada linha, o objetivo é minimizar a discrepância no sentido vertical e aproximar ao máximo os pixéis correspondentes das duas imagens.

Este algoritmo passa pelos seguintes 5 passos:

1. Encontrar um conjunto de pares de pontos correspondentes entre as duas imagens. O autor defende que devem ser no mínimo 7.
2. Com esses pontos calcula-se a matriz fundamental das imagens e encontram-se os epipolos de ambas as imagens. Um dos métodos utilizados é o método dos oito pontos.

3. Selecionar a matriz transformação H' que mapeia o epipolo para um ponto no infinito.
4. Encontrar a transformação H que minimize a distância calculada através do método dos mínimos quadrados.
5. Transformar ambas as imagens, esquerda e direita, respetivamente, com as matrizes H e H'.

Este é um método muito eficaz pois elimina a necessidade de calibração da câmara. Mais, o algoritmo é capaz de fazer a retificação de imagens de 1024x1024 em apenas 20 segundos. (Hartley, 1998) É importante referir, que apesar de simplificados nestes passos, este é um algoritmo complexo. Na figura 2.3 podemos observar os resultados da retificação utilizando este algoritmo na figura 2.4:

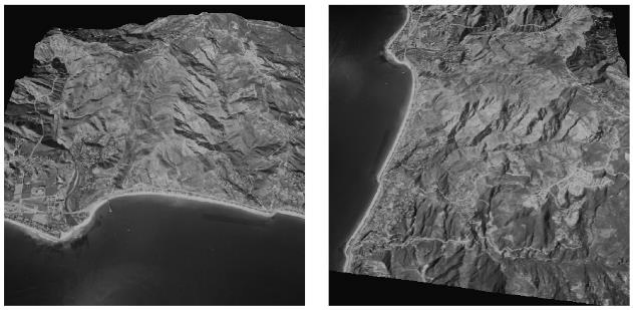

Figura 2.3 - Par de imagens estereoscópicas (Hartley, 1998)

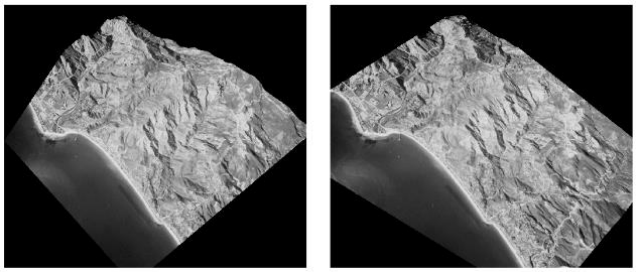

Figura 2.4 - Par de Imagens após Retificação de Hartley (Hartley, 1998)

Após a retificação das imagens, é necessário fazer-se o mapeamento entre os pixéis de ambas as imagens.

2.1.3 Deteção de vértices e arestas

Os principais métodos usados para fazer a correspondência entre os pixéis de duas imagens usam os vértices e arestas das imagens no seu algoritmo. Especialmente os vértices, são zonas da imagem com características de maior contraste em que é possível verificar a fronteira de objetos. Assim, a secção seguinte irá abordar do algoritmo de deteção de vértices de Harris (Harris e Stephens, 1988).

2.1.3.1 Deteção com o algoritmo de Harris

O algoritmo de deteção de Harris é um método matemático de encontrar arestas e vértices em imagens. Este é o método mais referido na literatura para criação de imagens estereoscópicas, visto que é um método que facilmente encontra pontos de interesse em imagens, com os quais é possível depois criar um mapa de dispersão. É também um método muito popular por ser invariante à rotação, escala, iluminação e ruído. (Derpanis, 2004)

Consideremos a seguinte imagem:

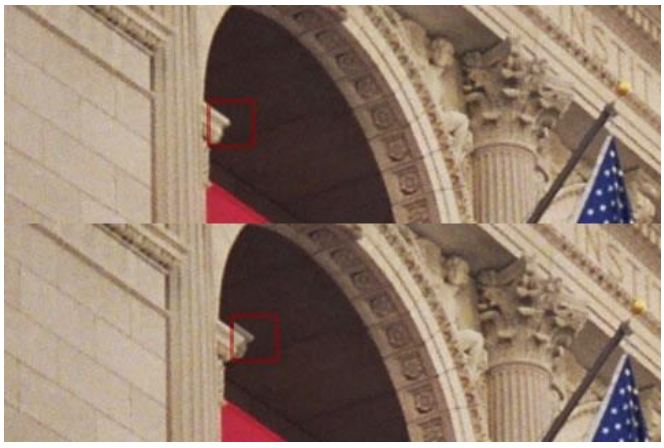

Figura 2.5 - Efeito do desvio de uma janela centrada num vértice

Nas duas imagens presentes na figura 2.5, podemos ver duas janelas (a vermelho). Um vértice, é uma zona da imagem em que ao movermos a janela temos uma grande variação na intensidade dos pixéis.

Ou seja, a função de autocorrelação vai ser alta:

$$\boldsymbol{E(x,y) = \sum_{u,v} w_{u,v}[I(u,v) - I(x+u, y+v)]^2} \quad (4)$$

Em que:

- u e v são, respetivamente, os deslocamentos horizontal e vertical da janela.
- $W_{u,v}$ é função de janela, que é igual a um dentro de uma janela retangular e a zero fora dela
- I(x,y) é a intensidade da imagem na posição (x,y)

Esta função foi proposta por Moravec em (Moravec, 1980) ,e revisitada por Chris Harris e Mike Stephens (Harris e Stephens, 1988), que verificaram alguns problemas na mesma. Nesta publicação, os autores verificaram que a resposta desta função era anisotrópica, só considerava desvios de 45 em 45º. Para corrigir este problema foi proposta a seguinte reformulação da fórmula:

$$\boldsymbol{E(x,y) = Ax^2 + 2Cxy + By^2} \quad (5)$$

Em que:

- $A=X^2 \otimes w$
- $B= Y^2 \otimes w$
- $C= (XY) \otimes w$

Verificaram também que o uso de uma janela retangular e binária (ou pertence ou não pertence à janela) levam a uma resposta com ruído. Para ultrapassar este defeito, foi introduzida uma janela Gaussiana:

$$\boldsymbol{w_{u,v} = e^{\frac{-(u^2+v^2)}{2\sigma^2}}} \quad (6)$$

Além deste problema, verificaram também que a função reagia muito a arestas. Assim, a forma final da fórmula obtida foi:

$$\boldsymbol{E(x,y) = (x,y)M(x,y)^T} \quad (7)$$

Em que $M=\begin{bmatrix} A & C \\ C & B \end{bmatrix}$

Para que esta fórmula responda apenas a vértices e não a cantos, foram introduzidas as variáveis α e β que são os valores próprios da matriz M, proporcionais à curvatura principal da função de autocorrelação. Concluíram então de que quando se tratava de vértices ambos os valores próprios atingiam valores elevados, bem como a função de autocorrelação.

Na figura 2.6 é possível perceber a que características da imagem correspondem os valores próprios:

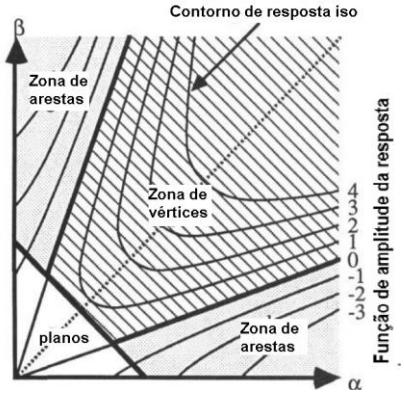

Figura 2.6 - Interpretação dos diferentes valores próprios face a determinadas características numa imagem (Harris e Stephens, 1988)

2.1.3.2 Detetor de Harris em espectro múltiplo

Uma importante alteração à formula de Moravec, (Moravec, 1980), foi introduzida por D.Firmenich, M.Brown e S.Süsstrunk. Tanto em (Moravec, 1980), como em (Harris e Stephens, 1988), estamos a falar em imagens na escala de cinza obtidas através da luminância. Assim, em (Firmenich, Brown e Süsstrunk, 2011) é introduzida também a gama espectral NIR. Deste modo a equação proposta por Moravec é ajustada ao seguinte formato:

$$\boldsymbol{E_{multispectral}}(\Delta \boldsymbol{x}) = \sum_{\boldsymbol{x_i} \in \boldsymbol{W}} \boldsymbol{w}(\boldsymbol{x_i}) \times \sum_{\boldsymbol{s}} [\boldsymbol{I}^{(\boldsymbol{s})}(\boldsymbol{x_i}) - \boldsymbol{I}^{(\boldsymbol{s})}(\boldsymbol{x_i} + \Delta \boldsymbol{x})]^{\boldsymbol{2}} \quad (8)$$

Onde $I^{(s)}(x_i)$ refere-se à intensidade no respetivo espectro *s*. Note-se ainda que, ao contrário das outras duas alternativas apresentadas, neste trabalho só é considerado o deslocamento em x.

À semelhança de (Harris e Stephens, 1988), é utilizada a série de Taylor para simplificar a expressão, deixando de parte todos os termos de segunda ordem ou superior. Assim, é proposta a seguinte equação:

$$\boldsymbol{E_{multispectral}}(\Delta \boldsymbol{x}) = \Delta \boldsymbol{x}^{\boldsymbol{T}} \times \boldsymbol{A_M} \times \Delta \boldsymbol{x} \quad (9)$$

Em que:

- $A_M = \sum_s \sum_{x_i \in W} w(x_i) \nabla I^{(s)} \nabla I^{(s)^T}$ é uma matriz 2x2 á qual se dá o nome de Matriz de autocorrelação multiespectral
- $\nabla I^{(s)}$ é o gradiente bidimensional no espectro s.

Assim, os vértices são obtidos quando ambos os valores próprios da matriz A_M atingem valores elevados.

Outro método bastante interessante, é o usado em (Achard, Bigorgne e Devars, 2000).

2.1.3.3 Detetor Sub-Pixel de vértices

Outro método bastante interessante, é o usado em (Achard, Bigorgne e Devars, 2000).

No artigo publicado, os autores fazem referência ao método desenvolvido por Harris e Stephens, baseado na matriz de autocorrelação. Esse "detetor de vértices" foi considerado em (Schmid *et al.*, 2010) como o mais robusto. Em (Achard, Bigorgne e Devars, 2000) é referido ainda que este detetor torna-se ainda melhor quando a imagem é passada primariamente por um filtro Gaussiano.

Em (Achard, Bigorgne e Devars, 2000), os autores apresentam uma alternativa mais fiável, que é facilmente adaptável a imagens de espectro múltiplo.

Nesta abordagem os autores propõem primeiro uma abordagem uni-espectral:

$$\boldsymbol{m} = \begin{cases} \frac{\boldsymbol{Ix^2\langle Iy^2\rangle + Iy^2\langle Ix^2\rangle - 2IxIy\langle IxIy\rangle}}{\boldsymbol{\langle Ix^2\rangle + \langle Iy^2\rangle}}, \boldsymbol{para\ \langle Ix^2\rangle + \langle Iy^2\rangle \geq 1} \\ \boldsymbol{Ix^2\langle Iy^2\rangle + Iy^2\langle Ix^2\rangle - 2IxIy\langle IxIy\rangle}, \boldsymbol{para\ \langle Ix^2\rangle + \langle Iy^2\rangle < 1} \end{cases} \quad (10)$$

Em que:

- **m** é a media ponderada do produto vetorial entre os vetores gradiente do pixel (Pi) em questão e dos seus vizinhos
- $\boldsymbol{Ix}$ e $\boldsymbol{Iy}$ são as derivadas parciais da imagem **I(x,y)**
- ⟨ ⟩ representa a imagem filtrada por um filtro gaussiano.

O objetivo é identificar pontos em que o valor de m seja elevado, e como tal, classificados como vértices.

2.1.3.4 Detetor sub-pixel de vértices multiespectral

Na abordagem multiespectral (Achard, Bigorgne e Devars, 2000), os autores consideram imagens multiespectrais de n bandas. Assim, estes admitem que a função imagem $f: R^2 \rightarrow R^n$, exprime a representação de uma imagem 2D nem vetor de *n* bandas de cor:

$$\boldsymbol{P(x,y)} \quad \rightarrow \quad \vec{\boldsymbol{f}} = (\boldsymbol{fr(x,y)}, \boldsymbol{fg(x,y)}, \boldsymbol{fb(x,y)} \ldots \boldsymbol{fn(x,y)}) \tag{11}$$

Para este vetor, estabelecem-se as seguintes relações[1]:

$$\boldsymbol{E} = \left(\frac{\boldsymbol{dfr}}{\boldsymbol{dx}}\right)^2 + \left(\frac{\boldsymbol{dfg}}{\boldsymbol{dx}}\right)^2 + \left(\frac{\boldsymbol{dfb}}{\boldsymbol{dx}}\right)^2 \tag{12}$$

$$\boldsymbol{G} = \left(\frac{\boldsymbol{dfr}}{\boldsymbol{dy}}\right)^2 + \left(\frac{\boldsymbol{dfg}}{\boldsymbol{dy}}\right)^2 + \left(\frac{\boldsymbol{dfb}}{\boldsymbol{dy}}\right)^2 \tag{13}$$

$$\boldsymbol{F} = \frac{\boldsymbol{dfr}}{\boldsymbol{dx}}\frac{\boldsymbol{dfr}}{\boldsymbol{dy}} + \frac{\boldsymbol{dfg}}{\boldsymbol{dx}}\frac{\boldsymbol{dfg}}{\boldsymbol{dy}} + \frac{\boldsymbol{dfb}}{\boldsymbol{dx}}\frac{\boldsymbol{dfb}}{\boldsymbol{dy}} \tag{14}$$

E a matriz $\begin{bmatrix} E & F \\ F & G \end{bmatrix}$, de onde se obtém:

$$\boldsymbol{\lambda_{+/-}} = \frac{\boldsymbol{E+G} \ \pm \sqrt{\boldsymbol{(E-G)^2+4F^2}}}{\boldsymbol{2}} \tag{15}$$

Que correspondem à discrepância:

$$\left(\boldsymbol{dx_{+/-}}, \boldsymbol{dy_{+/-}}\right) = (\boldsymbol{\cos\theta_{+/-}}, \boldsymbol{\sin\theta_{+/-}}) \tag{16}$$

Onde:

- $\theta_+ = \frac{1}{2}\tan^{-1}\frac{2F}{E-G} + k\pi$
- $\theta_- = \theta_+ \pm \frac{\pi}{2}$

Assim, alteram a expressão de m apresentada na versão uni-espectral para a versão multiespectral:

$$\boldsymbol{m} = \frac{\boldsymbol{\lambda}}{\langle\boldsymbol{\lambda}\rangle}\left[\left(\boldsymbol{1-2\,(\cos\theta)^2}\right)\langle\boldsymbol{\lambda\,(\cos\theta)^2}\rangle + \boldsymbol{(\cos\theta)^2}\,\langle\boldsymbol{\lambda}\rangle - \boldsymbol{2\sin\theta\cos\theta}\,\langle\boldsymbol{\lambda\sin\theta\cos\theta}\rangle\right] \tag{17}$$

Nos testes feitos, este detetor conseguiu melhores resultados face ao de Harris. É também em vértices obtusos que, este algoritmo obtém melhores resultados.

[1] Para melhor representação das expressões assumiu-se uma imagem com as bandas *RGB*

Figura 2.7 - Algoritmo apresentado em (Achard, Bigorgne e Devars, 2000)

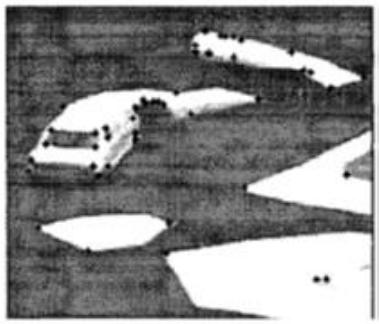

Figura 2.8 - Algoritmo de Harris

Note-se a diferença na precisão com que nas duas imagens estão identificados os vértices. Na figura 2.7 estão identificados mais vértices face à figura 2.8.

Os autores referem ainda que, para obterem os melhores resultados, o detetor sub-pixel compõe-se nos seguintes passos:

1. Usar o detetor uni-espectral na equação (10), e com base nesse detetor definir uma área de interesse em volta de um vértice detetado.
2. Usar a passagem por zero do *Laplaciano* dentro da área de interesse.
3. Encontrar a cadeia de zeros do *Laplaciano* com maior contraste.
4. Em torno dessa cadeia de pontos, calcular a medida ***m*** e encontrar o ponto com valor máximo de m. Esse ponto será o canto

Clarificando os passos 2 e 3, o que é feito é passar a imagem por um filtro de Laplace como os na figura 2.9.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Figura 2.9 - Filtros de Laplace Discretos

Estes filtros são uma representação discreta da seguinte fórmula do filtro:

$$\boldsymbol{L(x,y) = \frac{d^2 I}{dx^2} + \frac{d^2 I}{dy^2}} \quad (18)$$

Ao aplicarmos este filtro estamos a obter zonas da imagem em que a variação da intensidade da imagem varia muito bruscamente, fazendo o valor de $L(x,y)$ passar por zero. Numa imagem, arestas e vértices são pontos em que há uma variação muito grande na intensidade de cor, e como tal, pontos em que $L(x,y)$ passa por zero.

Ao obtermos essas cadeias temos pontos de interesse dos quais será calculado o ponto em que o valor de m é máximo, encontrando-se assim o vértice.

Estes algoritmos permitem-nos identificar uma lista de pontos de correspondência entre as duas imagens. O objetivo agora é encontrar algoritmos que nos façam o mapeamento entre as duas imagens 2D numa perceção 3D que nos dê informação sobre as distâncias na imagem.

2.1.4 Extrapolação de distâncias

Em (Zou e Li, 2010), os autores fazem uso do mapa de disparidade para criar a profundidade na imagem. Face a um valor de disparidade d e um ponto *2D (x,y)*, é feito o cálculo das coordenadas 3D:

$$\begin{cases} Q\begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \\ Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & f \\ 0 & 0 & -\frac{1}{T_x} & (c_x - c'_x)T_x \end{bmatrix} \end{cases} \quad (19)$$

Em que:

- As coordenadas (x, y, z) de um determinado ponto são, respetivamente, obtidas através de $\left(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W}\right)$.
- A matriz Q é a matriz de reprojeção que faz a transformação das coordenadas bidimensionais para três dimensões.
- T_x é a translação x entre as imagens direita e esquerda.
- c_x e c_y são as coordenadas do ponto de origem da imagem da câmara esquerda
- f é a distância focal da câmara esquerda
- c'_x é a abcissa c_x relativamente á imagem direita

Este algoritmo transforma duas imagens 2D numa só em 3D que além da cor de cada pixel, tem ainda a sua distância à câmara.

É de notar que como já foi referido anteriormente, este sistema assume que as imagens estão alinhadas horizontalmente. Daí na matriz de retransformação Q apenas apareça a transformação no eixo *x*.

Outra abordagem é o uso da trigonometria para o cálculo das distâncias. Foi essa a abordagem adotada em (Mrovlje e Vrančić, 2008)

Neste método, à semelhança do anterior, é importante que as câmaras estejam horizontalmente alinhadas. Assumindo isso, os autores dão o seguinte exemplo explicativo:

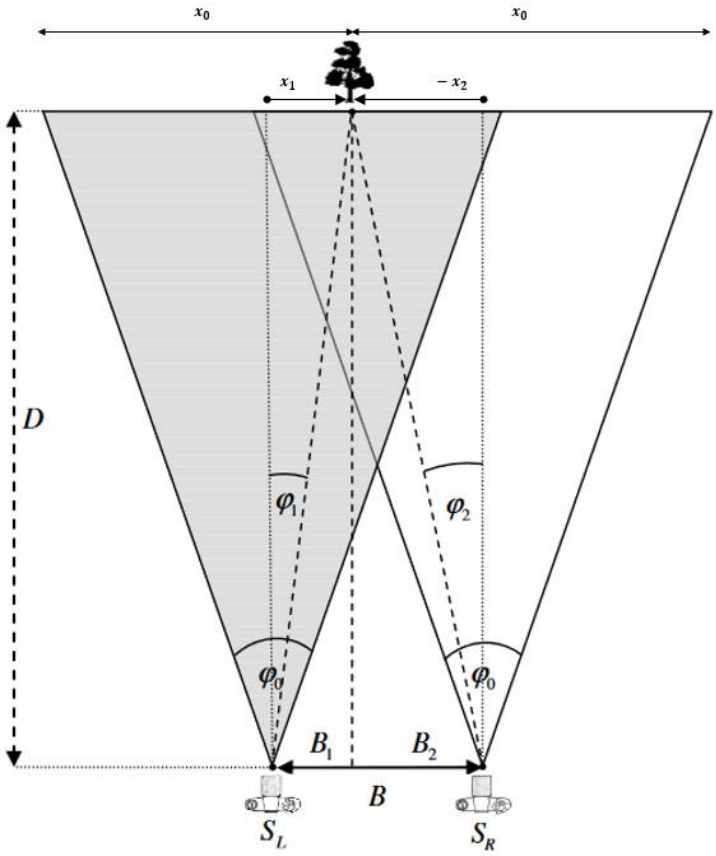

Figura 2.10 - Esquema de visão estereoscópica (Mrovlje e Vrančić, 2008)

Na figura 2.10, observa-se um sistema de visão estereoscópica a tirar foto a uma árvore. Neste sistema as câmaras S_L e S_R, esquerda e direita, estão a uma distância B uma da outra e a uma distância D da árvore, e têm um ângulo de visão horizontal φ_0.

A distância B pode ser escrita como:

$$B = B_1 + B_2 = D \tan \varphi_1 + D \tan \varphi_2 \quad (20)$$

Em que:

- φ_1 e φ_2 são respetivamente, o ângulo que o vetor câmara-árvore faz com o eixo longitudinal da lente, para a câmara esquerda e direita.

Considere-se agora a seguinte relação:

$$\frac{x_1}{\frac{x_0}{2}} = \frac{\tan \varphi_1}{\tan \frac{\varphi_0}{2}} \quad (21)$$

Daí vem que:

$$\tan \varphi_1 = \frac{x_1}{\frac{x_0}{2}} \tan \frac{\varphi_0}{2} \quad (22)$$

Esta expressão é utilizada também para a câmara direita. Aplicando à equação (20), temos que a distância à arvore é:

$$D = \frac{B x_0}{2 \tan\left(\frac{\varphi_0}{2}\right) |x_1 - x_2|} \quad (23)$$

Associada a esta distância vem ainda o erro causado pela resolução da câmara. A diferença de um pixel pode causar uma grande discrepância na imagem. Para testarem este sistema, dispuseram vários marcadores para medirem a distância (figura 2.11). Para cada marcador, foi medida a distância com as câmaras em diferentes posições e com diferentes espaçamentos (B).

Figura 2.11 - Imagem dos Marcadores (Mrovlje e Vrančić, 2008)

Face a estes testes, efetuados com marcadores às distâncias de 10, 20, 30, 40, 50 e 60 metros concluiu-se que quanto maior a distância B entre as câmaras menor foi o erro causado. Além disso, conclui-se que apesar de este ser um algoritmo sem grande complexidade, este produz resultados bons.

2.2 Deteção de Fogo

Atualmente, muitas são as técnicas baseadas em sistemas digitais que nos ajudam a detetar fogo. Especificamente, quando se fala de encontrar fogo através do processamento de imagem, há um fator muito importante a ter em conta: Em que zona do espectro será feita a captura da imagem?

A maioria dos sensores usados na captação de imagem, funcionam numa combinação das seguintes bandas espectrais:

- Radiação no espectro visível (RGB)- situa-se entre os 390 e os 700 nanometros. Esta banda do espectro é digitalmente captada em três bandas - azul, verde e vermelho (RGB). Os sensores usados reagem de forma diferente à cor, dependendo da sua resposta espectral (figura 2.12).
- Radiação no espectro infravermelho (IR) – situa-se entre os 700 e os 14,000 nanómetros. Esta banda está subdividida em (Burnett e Wing, 2018):
    - IR próximo (NIR) – dos 700nm aos 1 400nm.
    - IR de onda curta – dos 1 400nm aos 3 000nm.
    - IR de média onda – dos 3 000nm aos 8 000nm.
    - IR de onda longa – dos 8 000nm aos 14 000nm.
- Radiação no espectro Ultravioleta (UV)- situa-se entre o 1nm e os 380nm e pode ser subdividida em:
    - UV próximo – dos 380nm aos 200nm
    - UV distante – dos 200nm aos 10nm
    - UV extremo – dos 10nm aos 1nm

Na figura 2.12, observa-se a resposta espectral de uma câmara *Raspberry Pi V.2 Noir,* que combina os partes dos espectros do visível e infravermelho próximo:

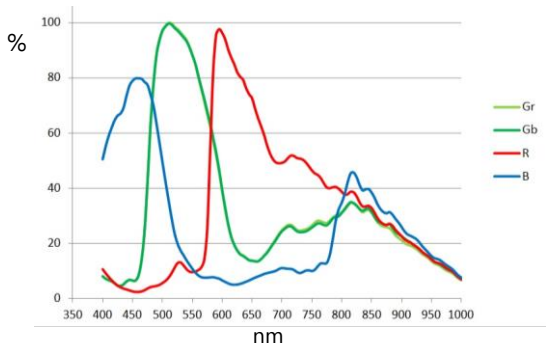

Figura 2.12 - Resposta espectral do sensor Sony IMX-219 (cortesia da Sony)

Capturar imagens no espectro visível apresenta desvantagens face ao espectro infravermelho: primeiro, porque o fogo emite mais energia no espectro IR do que do visível (figura 2.13); depois, porque o fumo e fraco contraste com fundo da imagem diminuem drasticamente a capacidade de detetar fogo. (Burnett e Wing, 2018)

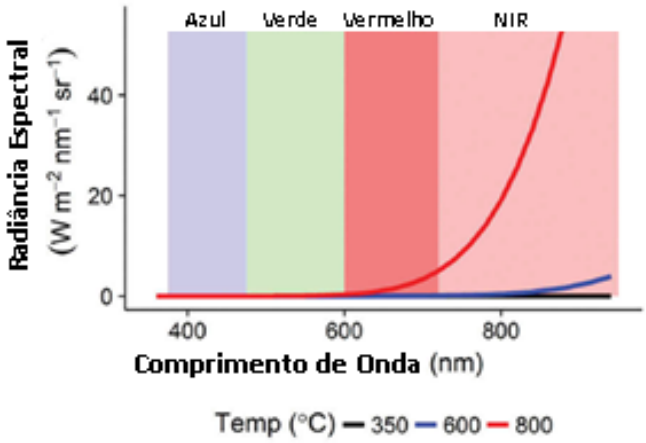

Figura 2.13 – Resposta espectral da radiância em função da temperatura (Burnett e Wing, 2018)

Por outro lado, as câmaras infravermelhas são muito mais caras do que uma simples câmara RGB. Face a isto, será feita a descrição de vários métodos utilizados na deteção de fogo nas bandas RGB e NIR, por serem as economicamente mais acessíveis.

Outro ponto importante a ter em conta, é em que espaço de cor estamos a analisar a imagem: RGB, YPbPr, YCbCr, ICtCp, YUV, CIELAB, etc. Dependendo do espaço, iremos conseguir extrair determinadas características mais facilmente.

2.2.1 Deteção de fogo em RGB e YCbCr

O método de deteção de fogo proposto em (Celik, 2010) baseia-se num sistema de decisão em árvore para determinar se um determinado pixel corresponde ou não a fogo. Nessa árvore, a imagem é analisada nos espaços de cor RGB e YCbCr.

O espaço YCbCr, está dividido nas gamas:

- Y – Luma, que é a soma ponderada dos componentes RGB em compressão gamma.
- Cb – componente azul relativamente ao verde.
- Cr – componente vermelho relativamente ao verde.

Na figura 2.14, podemos ver a árvore de decisão usada:

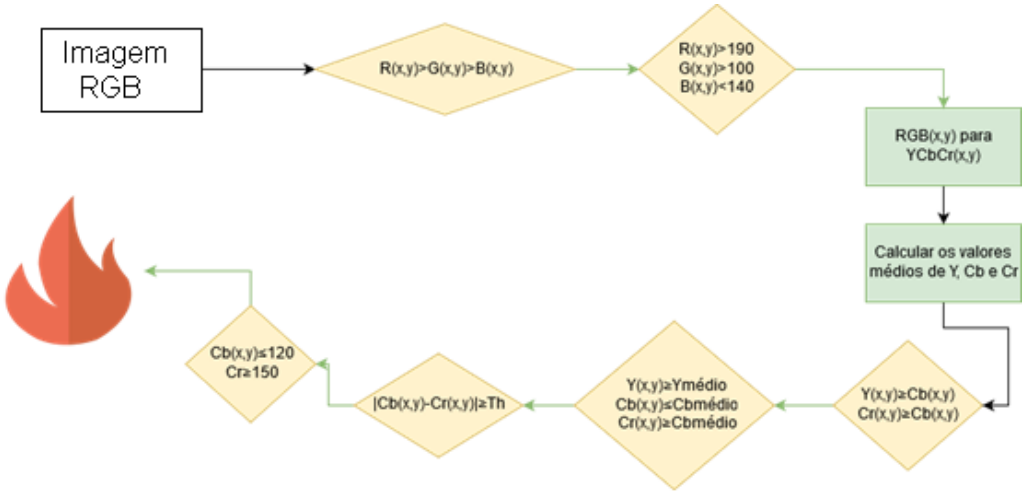

Figura 2.14 – Esquema de funcionamento do sistema de deteção de fogo

Na figura 2.14, as expressões R(x,y), G(x,y) e B(x,y) referem-se à intensidade da respetiva banda no pixel de coordenadas (x,y), no espaço de cor RGB. Do mesmo modo, Y(x,y), Cb(x,y) e Cr(x,y) refere-se à intensidade da respetiva banda no pixel de coordenadas (x,y), no espaço de cor YCbCr.

O algoritmo utilizado está dividido em dois espaços de cor: RGB e YCbCr. Assim, na primeira parte do algoritmo verifica-se, em cada pixel, se a intensidade de vermelho é superior à de verde e por sua vez a verde superior à azul – num pixel com chama, é normal que haja uma maior intensidade da banda vermelha, face às outras. Caso o pixel

passe no teste, é verificado se os valores da intensidade de cada banda se encontram enquadrados com valores espectáveis.

Após os testes em RGB, a imagem é convertida de RGB para YCbCr e calculados os seu valores médios ($Y_{médio}$,$Cb_{médio}$,$Cr_{médio}$), face a toda a imagem.

Após conversão, os testes passaram a ser feitos em YCbCr. Assim, o valor de Y e o valor de Cr tem de ser superiores ao valor de Cb, ou seja, a componente azul relativamente ao verde, deve ter valores mais baixos que Y e Cr num pixel de chama. Além disso, nos pixéis de fogo, o valor de Y e de Cr devem ser superiores à sua média, e o valor de Cb inferior.

Depois é calculado o módulo da diferença entre Cb e Cr. Este valor deve ser superior a um valor limite **Th**. O valor de **Th** é obtido através da curva de característica de operação do recetor (ROC), onde foram testados 100 valores de Th e obtidas as percentagens de correta classificação e de falsos positivos. Assim, foi obtida a seguinte curva ROC (figura 2.15)

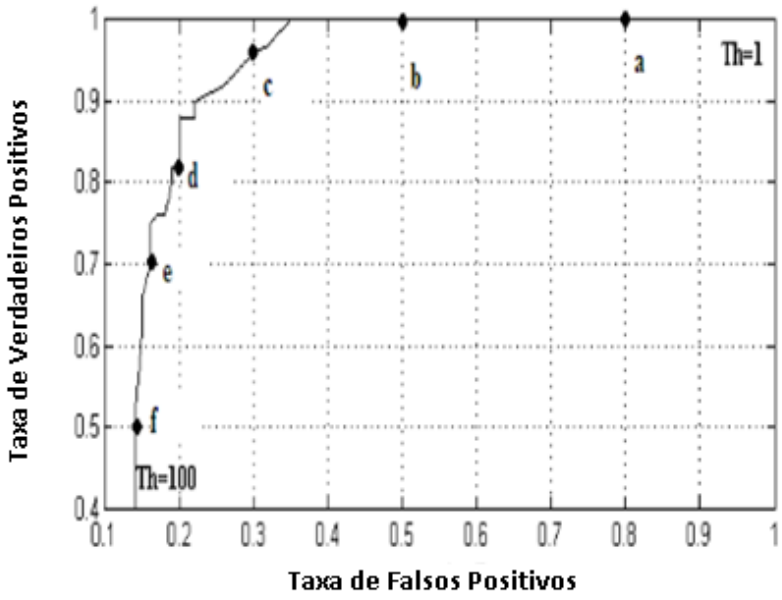

Figura 2.15 - Curva ROC para deteção do Th ótimo (Celik, 2010)

Note-se que, quanto maior a percentagem de pixéis corretamente classificados, também aumenta a percentagem de falsos negativos. Para decidir entre quais valores de Th são os melhores, é necessário balançar entre boa deteção e baixa percentagem de falsos positivos. Assim, os autores adotaram o valor de Th=70 (ponto c), o qual lhes permite obter cerca de 95% de percentagem de deteção e menos de 30% de falsos alarmes.

Finalmente, os valores de Cb e Cr são testados para ver se se enquadram com os valores espectáveis.

Os autores conseguiram obter a seguinte matriz de confusão (tabela 2.1):

Tabela 2.1 - Matriz de confusão do Classificador

		Real	
		Fogo	**Sem Fogo**
Classificado	**Fogo**	198	28
	Sem Fogo	2	172

Após esta análise, os autores concluíram que o algoritmo projetado alcançava 99% de deteções corretas com apenas 14% de falso alarme. Somando à qualidade deste algoritmo vem o facto de se tratar de um método com complexidade computacional baixa, o que o torna útil para vigilância florestal em tempo real.

2.2.2 Deteção de Fogo em YCbCr e CIELab

Em (Chen, Luo e Alsharif, 2013), os autores criaram um sistema de deteção de fogo baseado em YCbCr, que foi abordado na secção anterior, e CIELab. O respetivo fluxograma é apresentado na figura 2.16.

O espaço de cor CIELAB está dividido nas gamas:

- L* - luminosidade do pixel (L*=0 corresponde à cor preta)
- a* - localização da cor entre vermelho e verde (valores negativos aproximam-se de verde, e os positivos do vermelho)
- b* - localização da cor entre amarelo e azul (valores negativos aproximam-se do azul, e os positivos do amarelo).

Este espaço de cor é muito importante pois é um espaço de cor independente do dispositivo, que tenta simular ao máximo o olho humano.

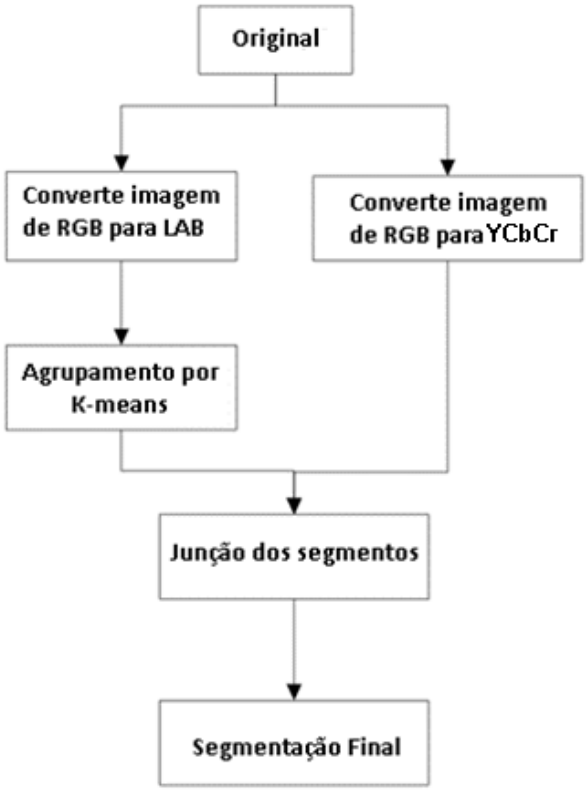

Figura 2.16 – Esquema de funcionamento do algoritmo de deteção de fogo (Chen, Luo e Alsharif, 2013)

Este sistema, faz uso de ambos os espaços para obter melhores resultados. No espaço YCbCr é criada a seguinte equação para classificação dos pixéis:

$$\boldsymbol{f(Cr,Cb)=\frac{1}{2\pi S_{Cr}S_{Cb}}e^{\frac{1}{2}\left(\left(\frac{Cr-\overline{Cr}}{S_{Cr}}\right)^2-2\frac{(Cr-\overline{Cr})(Cb-\overline{Cb})}{S_{Cr}S_{Cb}}+\left(\frac{Cb-\overline{Cb}}{S_{Cb}}\right)^2\right)}} \quad (24)$$

Em que:

- $\boldsymbol{\overline{Cr}=\frac{1}{N}\sum_{i=1}^{N}Cr_i}$ e $\boldsymbol{\overline{Cb}=\frac{1}{N}\sum_{i=1}^{N}Cb_i}$ (25)
- $\boldsymbol{S_{Cr}=\frac{1}{N}\sum_{i=1}^{N}(Cr_i-\overline{Cr})^2}$ e $\boldsymbol{S_{Cb}=\frac{1}{N}\sum_{i=1}^{N}\left(Cb_i-\overline{Cb}\right)^2}$ (26)

Com esta equação, é possível verificar se um dado pixel é ou não classificado como fogo, dando os seus valores de Cr e Cb. Assim, se $f\left(Cr(x,y),Cb(x,y)\right)>T$, classificamos como fogo. Os autores, após análise estatística obtiveram a seguinte tabela de valores (tabela 2.2):

Tabela 2.2 - Valores ótimos da equação (24)

$\overline{Cr}$	0.246431
$\overline{Cr}$	0.682332
${S_{Cr}}^2$	0.00414649
${S_{Cr}}^2$	0.0050133
T	0.00003

No espaço de cor CIELab foi adotada uma estratégia um pouco diferente. É usado o algoritmo de agrupamento *K-means* para segmentar a imagem. Esse algoritmo segue os seguintes passos:

1. Selecionar um valor de *threshold* ***T*** e um método para o cálculo da distância. Neste caso os autores usaram:
   - ***T***=0.01
   - $D=\alpha|{L^*}_1-{L^*}_2|+\beta|{a^*}_1-{a^*}_2|+\gamma|{b^*}_1-{b^*}_2|$, em que os valores de L*a*b* são referentes a dois pontos
   - $\alpha=0.2$ e $\beta=\gamma=0.4$

2. Agrupar os pixéis com base no valor de T e no fórmula da distância.
3. Voltar a percorrer a imagem onde se atribuem os pixéis aos *clusters* a menor distância.
4. Recalcular a média de cada cluster
5. Comparar a distância entre a média nova e a antiga. Se a distância nova for menor atribuir o pixel ao cluster. Senão, repetir os passos 3 e 4.
6. Para cada cluster verificar as regras: $\begin{cases} |\mu_{L^*} - \overline{L^*}| \leq S_{L^*} \\ |\mu_{a^*} - \overline{a^*}| \leq S_{a^*} \\ |\mu_{b^*} - \overline{b^*}| \leq S_{b^*} \end{cases}$, em que μ é a média do cluster, $(\overline{L^*}, \overline{a^*}, \overline{b^*}) = (0.837825\,, 0.525517, 0.735679)$ e $(S_{L^*}, S_{a^*}, S_{b^*}) = (0.1589633\,, 0.0897174\,, 0.1923818)$. Nos clusters onde esta regra se verificar, os pixéis são classificados como chama.

Combinando ambas as técnicas, em YCbCr e CIELAB, obtém-se um algoritmo com baixa percentagem de falsos negativos, mas com uma percentagem de falsos positivos alta. Na figura 2.17, verifica-se como estes dois algoritmos permitem uma boa segmentação de imagens que contenham fogo:

Figura 2.17 - Resultados do algoritmo (Chen, Luo e Alsharif, 2013) :
a) imagem original ; b) imagem após segmentação em YCbCr ; c) imagem após segmentação em CIELab ; d) imagem com combinação das segmentações em YCbCr e CIELab

2.2.3 Deteção de fogo em CIELab com base no movimento

Até este ponto, a deteção foi baseada apenas nas características de cor dos pixéis. Em (Celik, 2010), o autor explora as características de movimento do fogo para auxiliar na classificação dos pixéis.

No sistema adotado, o autor apenas usa o espaço de cor *CIELab* para a deteção. Como tal, o primeiro passo no seu algoritmo é converter a imagem nos componentes *RGB* em L*a*b*. Na figura 2.18, encontra-se o fluxograma deste algoritmo.

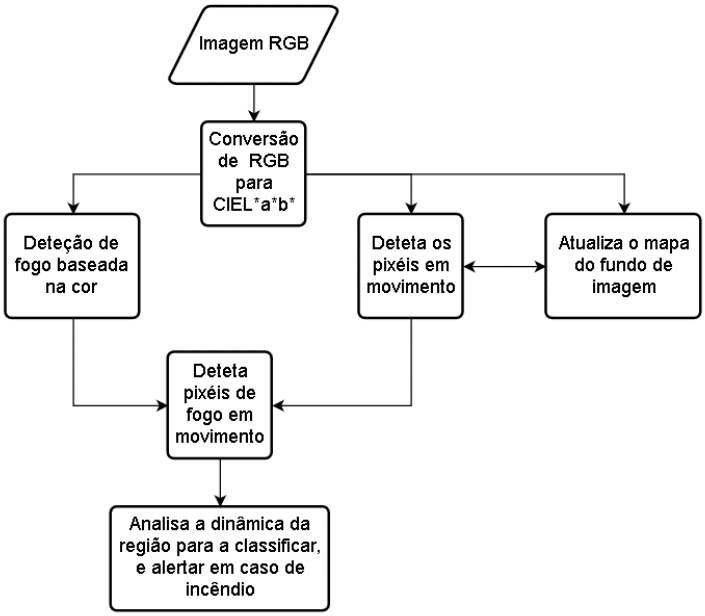

Figura 2.18 – Esquema de funcionamento do algoritmo de deteção de fogo em *CIELab* (Celik, 2010)

Após isso, à semelhança dos autores apresentados anteriormente, usa um conjunto de regras para verificar se um dado pixel pode ou não ser classificado como fogo. Para um pixel na posição (x, y) poder ser classificado como fogo, os seus componentes $L^*a^*b^*$ tem que verificar as seguintes condições, face aos valores médios das componentes:

$$\begin{cases} \boldsymbol{L^*(x,y) \geq L^*_m} \\ \boldsymbol{a^*(x,y) \geq a^*_m} \\ \boldsymbol{b^*(x,y) \geq b^*_m} \\ \boldsymbol{b^*(x,y) \geq a^*(x,y)} \end{cases} \quad (27)$$

A estas regras, o autor adiciona outra com base na análise estatística de 500 imagens, manualmente segmentadas. Nessa análise, são obtidos os seguintes histogramas que relacionam as intensidades das componentes de um pixel de fogo.

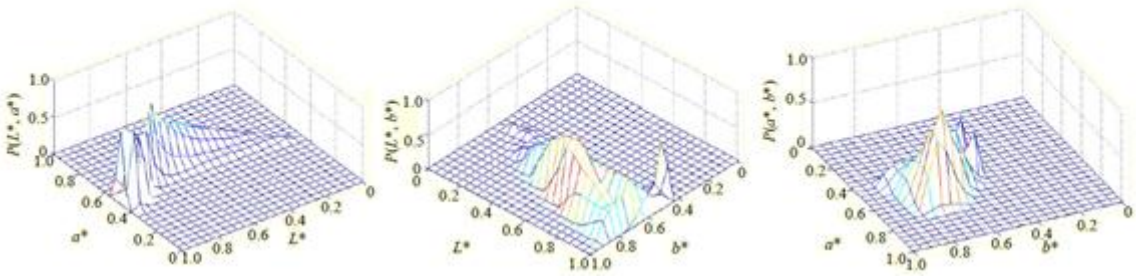

Figura 2.19 - Histograma das Probabilidades $P(L^*, a^*)$, $P(L^*, b^*)$ e $P(a^*, b^*)$ (Celik, 2010)

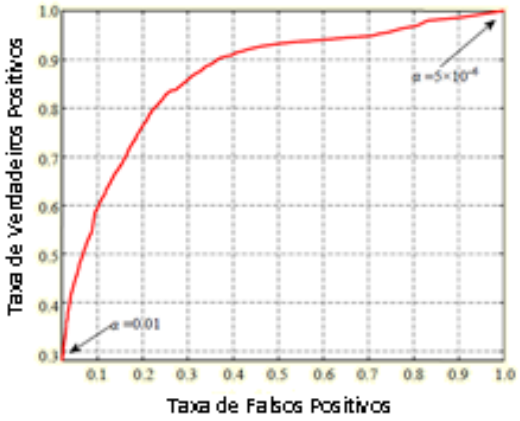

Figura 2.20 - Curva ROC e respetivos valores α (Celik, 2010)

Com base nestes histogramas, é criada uma quinta regra:

$$\boldsymbol{P(L^*, a^*)P(L^*, L^*)P(L^*, L^*) \geq \alpha} \quad (28)$$

Onde α é obtido através da análise ROC, na figura 2.20. Através do balanço entre a percentagem de classificações corretas face à probabilidade de falsos positivos, este valor pode ser obtido. Por exemplo para um valor de $\alpha = 0.00016$, tem-se 36.5% de falsos positivos face a 90% de classificações corretas.

Abordando agora a análise do movimento da chama, o autor sugere a utilização dos mapas de "*binary frame difference*" (FD) e "*background diference*" (BD), referentes à banda *L**, num dado segundo *t*. Esses mapas, são matrizes do tamanho da imagem, com valores binários.

Com esses mapas é construído o mapa "*background image*" (BG), onde são registados os pixéis que pertencem ao fundo. O cálculo desses mapas é realizado através das equações de (29) a (31):

$$\boldsymbol{FD(x,y,t)} = \begin{cases} \mathbf{1}\,, \boldsymbol{para}\ |\boldsymbol{L^*(x,y,t)} - \boldsymbol{L^*(x,y,t-1)}| \geq \boldsymbol{\tau}(|\boldsymbol{L^*(x,y,t)} - \boldsymbol{L^*(x,y,t-1)}|) \\ \mathbf{0}\,, \boldsymbol{nos\ outros\ casos} \end{cases} \quad (29)$$

$$\boldsymbol{BD(x,y,t)} = \begin{cases} \mathbf{1}\,, \boldsymbol{para}\ |\boldsymbol{L^*(x,y,t)} - \boldsymbol{BG(x,y,t-1)}| \geq \boldsymbol{\tau}(|\boldsymbol{L^*(x,y,t)} - \boldsymbol{BG(x,y,t-1)}|) \\ \mathbf{0}\,, \boldsymbol{nos\ outros\ casos} \end{cases} \quad (30)$$

$$\boldsymbol{BG(x,y,t)} = \begin{cases} \boldsymbol{L^*(x,y,t)} & , \boldsymbol{para\ SI(x,y,t) = T_{si}}) \\ \boldsymbol{BG(x,y,t-1)}\,, & \boldsymbol{nos\ outros\ casos} \end{cases} \quad (31)$$

Onde:

- $\boldsymbol{\tau(map)} = \begin{cases} \frac{\sum_x \sum_y \boldsymbol{map(x,y)}}{\boldsymbol{N}} + \sqrt{\frac{\sum_x \sum_y \left(\boldsymbol{map(x,y)} - \frac{\sum_x \sum_y \boldsymbol{map(x,y)}}{\boldsymbol{N}}\right)^2}{\boldsymbol{N}}} & , \boldsymbol{se} \geq \mathbf{10} \\ \mathbf{10} & , \boldsymbol{nos\ outros\ casos} \end{cases}$ (32)
  é o limite dinâmico para as equações (29) e (30). Estes limites são ajustados conforme a iluminação das imagens obtidas.
- $SI(x,y,t) = \begin{cases} SI^*(x,y,t-1) & , para\ FD(x,y,t) = 0 \\ 0\,, & nos\ outros\ casos \end{cases}$, é o índice de estacionariedade de um pixel, que conta há quantos *frames* é que o pixel não se move.
- T_{si} é um limite que determina quanto tempo um pixel tem de permanecer parado para ser considerado fogo. Este valor deve ser ajustado para igualar o número de *frames* processadas por segundo.

Computando o operador *OR* nos mapas *FD* e *BD*, obtém-se um mapa com os pixéis que se movem assinalados:

$$M(x,y) = FD(x,y,t) \oplus BD(x,y,t)$$

Depois desta fase, são analisadas as regiões de fogo em *frames* consecutivas. Assim, considera-se o novo mapa que assinala os pixéis em que as 5 regras, apresentadas anteriormente, se verificam *F(x,y)*.

Verificando quais pixéis são assinalados em ambos *F(x,y)* e *M(x,y)*, é criado o mapa *CF(x,y)* (equação (33)). Nesse mapa verifica-se a dimensão (*NO(t)*) das várias zonas classificadas como fogo (*O(t)*):

$$\boldsymbol{CF(x,y) = F(x,y) \otimes M(x,y)} \quad (33)$$

Assim, obtém-se o classificador final:

$$\boldsymbol{D(t) = \frac{CGO(t) - CGO(t - FPS + 1)}{FPS}} \quad (34)$$

Em que:

- *FPS* são o número de *frames* por segundo que o sistema consegue processar.
- $\boldsymbol{CGO(t)} \begin{cases} \boldsymbol{CGO(t-1)+1}, \boldsymbol{para\ NO(t) \geq NO(t-1)} \\ \boldsymbol{CGO(t-1)}, \boldsymbol{nos\ outros\ casos} \end{cases}$ (35), é um contador da atividade da presumível chama.
- Para valores de $D(t) \geq 0.4$ considera-se a zona assinalada como chama

Note-se que nesta última fase deixa-se de analisar a imagem pixel a pixel, e passa-se a analisar as várias zonas candidatas a serem classificadas como fogo. Além disso, o índice *D(t)* dá-nos informações sobre a movimentação da chama, possibilitando outras aplicações.

Outro aspeto a ter em conta, é o facto de o algoritmo assumir que o fogo se encontra em expansão, o que pode não se verificar em determinados casos.

Apesar disso, após testes feitos em 10 vídeos, que totalizaram cerca de 13 mil *frames*, foi obtida uma taxa de acerto de 99.88%. Também em testes realizados numa base de dados online o algoritmo atingiu resultados semelhantes.

2.2.4 Deteção de fogo em NIR

Como já foi referido anteriormente, as câmaras sensíveis no espectro NIR, são capazes de visualizar situações de fogo com mais clareza, sendo menos afetadas pela existência de fumo. Além disso, são câmaras com preços semelhantes às de espectro visível.

Assim, em (Leñero-Bardallo *et al.*, 2013), os autores exploram a possibilidade de utilização de uma câmara NIR para deteção de fogo. Utilizando um sensor CMOS e filtro passa-alto NIR, os autores são capazes de detetar chamas. Importante referir que os autores criaram um sensor baseado em eventos.

Segundo os resultados obtidos, as chamas provocam variações rápidas e periódicas nos valores NIR. Essas variações periódicas são fáceis de detetar através do cálculo do desvio padrão de níveis NIR anteriores. A figura 2.21 mostra a diferenças na intensidade e variação entre pixéis de chama (a azul) e os de fundo (a vermelho):

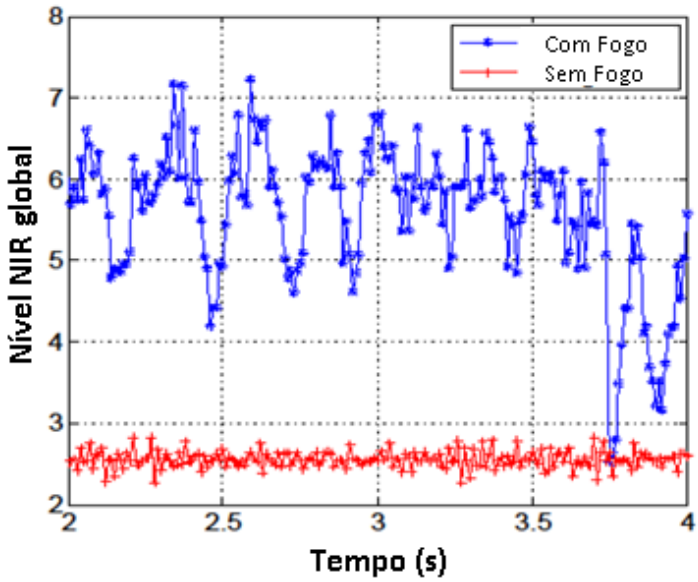

Figura 2.21 - Variação temporal do nivel NIR (Leñero-Bardallo *et al.*, 2013)

Como se pode observar, em pixéis de fogo não só a intensidade NIR é superior, como a sua variação é muito superior à dos pixéis de fundo.

Após o cálculo sequencial de várias *frames*, é possível verificar quais os pixéis que podem ser classificados como fogo, através do cálculo da variação dos valores NIR na imagem.

Assim, é periodicamente calculado:

$$\boldsymbol{A[n] = \frac{\sigma_{NIR}.N}{\sum_{k=0}^{N-1} NIR[n-k]}} \quad (36)$$

Em que:

- *N* é o número de níveis NIR estimados para atualização de $A[n]$.
- σ_{NIR} é o desvio padrão dos valores NIR.

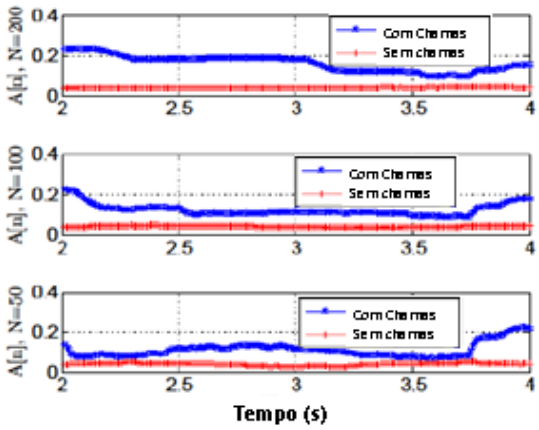

Figura 2.22 - Variação de $A[n]$ em função do valor de N (Leñero-Bardallo *et al.*, 2013)

Assim, quando $A[n] \geq 10\%$, classifica-se o pixel como chama. Este limite entre os pixéis de chama e os de fundo, foi obtido através do estudo dos valores de $A[n]$ para diferentes valores de ***N***. Na figura 2.22, pode-se ver o comportamento de $A[n]$ face a diferentes valores de ***N***. Independentemente do valor de ***N***, os pixéis de fundo assumem valores de $A[n]$ até aos 5%, face aos de fogo que, tipicamente, atingem valores três vezes superiores. Assim, assumiu-se o limite de 10% para diferenciar entre o fundo e o fogo.

Reiterando, os autores produziram um algoritmo para sensores baseados em eventos, em alternativa ao uso dos clássicos sensores baseados numa única imagem. Neste sistema, é possível a obtenção dos valores NIR em tempo real, e posterior classificação em pixéis de fogo e fundo. Essa classificação, como é referido pelos autores, é independente da iluminação da imagem.

2.2.5 Comparação da classificação nas bandas RGB e NIR

As câmaras NIR são câmaras de preço comparável ao das RGB, mas que permitem melhor reconhecimento de chamas. Por serem sensíveis na zona do espectro em que o combustível liberta energia, ajudam a captar melhor informação em situações de densidade de fumo.

Em (Burnett e Wing, 2018), os autores comparam o uso de câmaras RGB e NIR no estudo do fogo. Nos seus testes, foram usadas duas câmaras Canon S100. Para tornar uma das câmaras sensíveis à banda NIR, foi-lhe retirado o filtro passa baixo de infravermelhos, e colocado um filtro passa alto de 830nm. A câmara passa agora a operar entre os 830nm e 1000nm, continuando a ter as 3 bandas de "cor".

Para este estudo foi usado o mesmo algoritmo para classificação do fogo. O método utilizado foi o classificador de probabilidade máxima (Richards e Jia, 1999). Neste classificador, os pixéis do tipo fogo são estudados, e são criadas as estruturas m_i e $\sum i$ que são, respetivamente, o vetor de valores médios das bandas "RGB" e a matriz de autocorrelação dos pixéis de teste classificados como fogo. Estes dados são obtidos através do estudo estatístico de pixéis pertencentes a esta classe.

É ainda importante referir que se considera a existência das bandas RGB nas câmaras com gama NIR. Assim, os autores usam um conjunto de regras, associadas a equações de probabilidade que permitem fazer a classificação dos pixéis em classes, neste caso, fogo e não fogo.

Assim, para um pixel x ser classificado na respetiva classe, é calculado:

$$\boldsymbol{x \in classe_i \;\; se \;\; g_i(x) > g_i(x) \;\; para\ todo\ j \neq i\ e\ g_i(x) > T_i} \quad (37)$$

Onde:

- $\boldsymbol{g_i(x) = -ln|\sum i| - (x - m_i)^T \sum i^{-1} (x - m_i)}$ (38)
- $\boldsymbol{T_i < \frac{\left[(x-m_i)^T \sum i^{-1}(x-m_i)\right] + ln|\sum i| - 2\ \ln(p(classe_i))}{-2}}$ (39), para uma classificação aceitável

Por outras palavras, quando a probabilidade de um pixel pertencer à classe fogo é maior que a de pertencer a outras, e maior que o valor Th, este deve ser classificado com fogo.

Usando este algoritmo, os autores obtiveram os seguintes resultados:

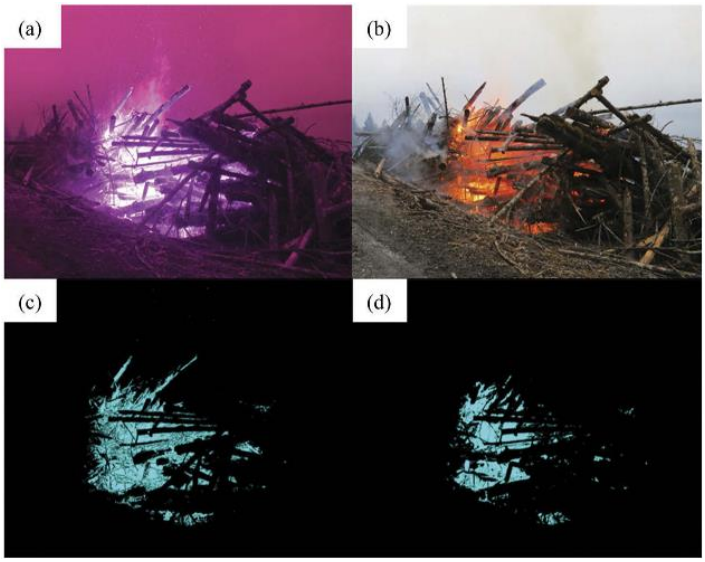

Figura 2.23 - Resultado do algoritmo (Leñero-Bardallo *et al.*, 2013): NIR - a) antes e c) depois; RGB - b) antes e c) depois

Através da figura 2.23, pode-se verificar que na gama NIR são obtidos melhores resultados que no espectro visível. Além disso, os testes foram feitos a 11 pilhas de lenha a arder, e o contraste entre os pixéis de fogo e o fundo da imagem foi superior na banda NIR (figura 2.24).

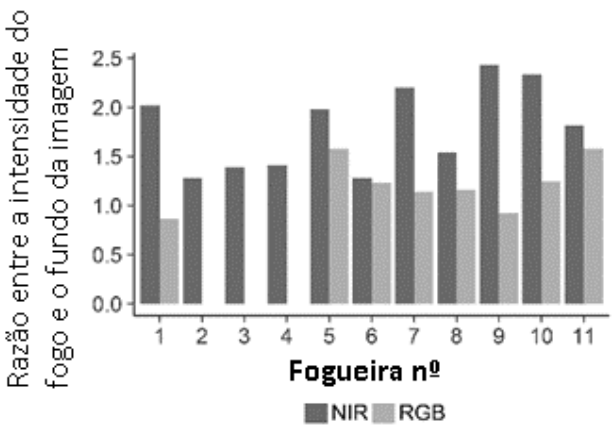

Figura 2.24 - Resultados estatísticos de 11 pilhas de lenha (Leñero-Bardallo *et al.*, 2013)

Este estudo dá boas evidências de que as câmaras no espectro NIR são mais sensíveis à presença do fogo que as de espectro visível. Facilmente se interpreta que, as câmaras NIR penetram eficientemente pelo fumo e dão outputs de alta resolução. A alternativa de baixo custo oferecida, face aos provados sensores térmicos, é não só atrativa como muito acessível(Burnett e Wing, 2018).

2.3 Classificação de Incêndios

Após detetar um fogo, é necessário avaliá-lo. Tem de existir um conjunto de parâmetros que possam ser avaliados de forma a melhor conhecer o perigo gerado por uma frente de fogo. Acontece que ao contrário de catástrofes como os sismos, ou os furacões, não existe nenhuma escala de intensidade, mundialmente aceite, para os incêndios florestais.

Por outro lado, existem categorias que nos ajudam a distinguir o tipo de incêndio florestal. Em (Alexander e Cruz, 2011) , são definidos 3 tipos gerais de fogos:

- Fogo subterrâneo – fogo de propagação lenta e sem chamas visíveis
- Fogo de superfície – fogo rasteiro com capacidade para rápida propagação (figura 2.25.A).
- Fogo de copa – fogo que atinge a copa das árvores
    - Passivo – fogo de média intensidade com capacidade para rápida propagação (figura 2.25.B).
    - Ativo – fogo de alta intensidade grande probabilidade de propagação (figura 2.25.C).

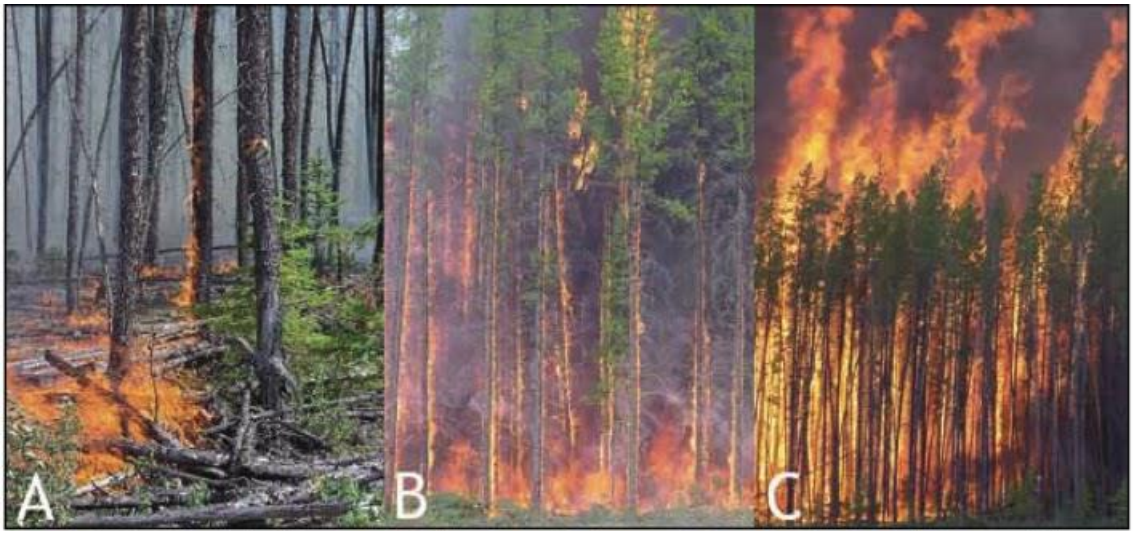

Figura 2.25 - Categorias de incêndio florestal

É importante referir que esta classificação está adequada a florestas coníferas, ainda que se verifique este tipo de incêndios noutros tipos de florestas.

Para distinção entre os tipos de fogo, Van Wagner desenhou um diagrama que ajuda na distinção entre os tipos de fogos:

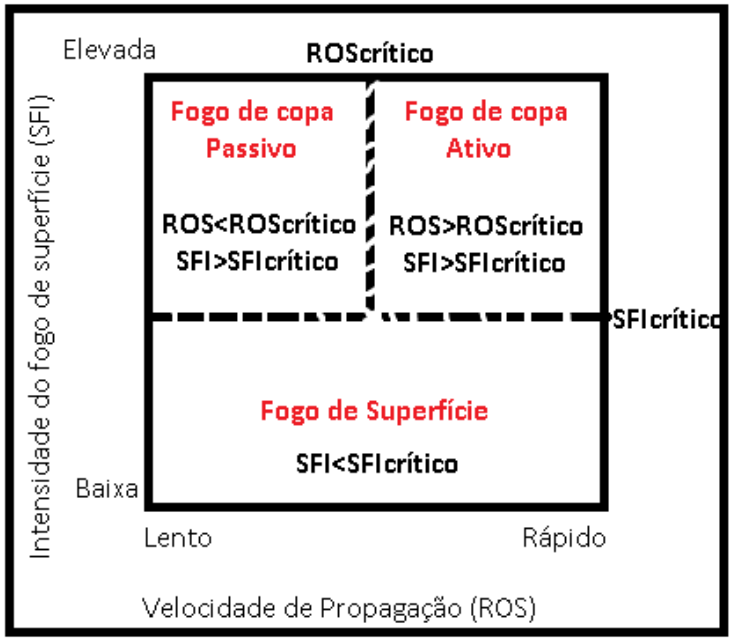

Figura 2.26 - Diagrama de Van Wagner

Este diagrama divide os três tipos de fogo apresentados na figura 2.26. Assim, quando a intensidade de fogo à superfície (SFI) é inferior a um dado limite $SFI_{crítico}$, este classifica-se como fogo de superfície. Se este limite for superado, passa-se a falar de fogo de copa. Dentro dos fogos de copa, os Passivos são aqueles cuja velocidade de propagação (ROS) é inferior a um $ROS_{crítico}$. Quando este valor é ultrapassado, está-se na presença de um fogo de copa ativo.

Em (Scott, 2006), o autor explora a criação de duas escalas para incêndios florestais: Escala de Intensidade e de Magnitude. Clarificando, a intensidade refere-se a um foco específico de incêndio, face que a magnitude trata o incêndio como um todo.

A escala de Intensidade pode ser definida como:

$$I = \log_{10}\left(\frac{H\omega_f R}{60}\right) \quad (40)$$

Em que:

- I é a intensidade do fogo, classificada em seis classes (I pode variar no intervalo $]0, 6[$)

- H é o *calor baixo de combustão*, expresso em kJ/kg
- $\boldsymbol{\omega_f}$ é a quantidade de combustível consumido na frente de fogo por área, expresso em kg/m^2
- $\boldsymbol{R}$ é a velocidade de propagação linear da frente de fogo, expressa em m/min

Esta escala, permite uma melhor compreensão do comportamento de determinadas frentes de fogo.

A escala de Magnitude é definida como:

$$\boldsymbol{M = log(\int I_B)} \quad (41)$$

Em que:

- $\boldsymbol{\int I_B = \frac{dA}{dt}.\frac{H\omega_f}{60}}$ (42) , corresponde à intensidade acumulada de todo o fogo $\boldsymbol{\frac{dA}{dt}}$ é o rácio de aumento de área a arder, expresso em $\boldsymbol{m^2/min}$

Para simplificação, o autor assume que o incêndio é uniforme e com uma forma elíptica, obtendo:

$$\boldsymbol{\frac{dA}{dt} = \sqrt{4\pi abA}} \quad (43)$$

Onde:

- $\boldsymbol{a = \frac{R_{head}}{1+\sqrt{1-L_B^{-2}}}}$ (44)
- $\boldsymbol{b = \frac{a}{L_B}}$ (45)
- $\boldsymbol{L_B = 1 + 0.155U_e}$ (46)
- U_e é a velocidade do vento em km/h
- R_{head} é a velocidade linear da frente de propagação do fogo em m/min

Nesta escala de magnitude, o valor de M varia entre]0 , 10[, permitindo obter-se 10 níveis de magnitude. Esta escala permite uma melhor avaliação da escala com que um incêndio florestal se tem espalhado e propagado.

Outra tentativa de uniformização, foi feita em (Alexander, 1982). O autor, tenta extrair o máximo de informação de observações feitas ao comportamento da frente de fogo.

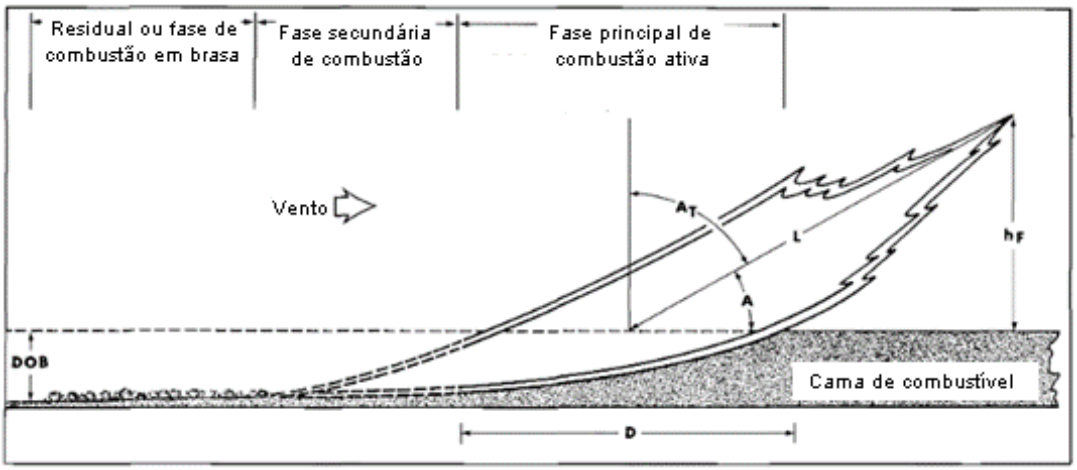

Figura 2.27 - Esquema de características de uma chama (Alexander, 1982)

Avaliando as características estruturais do fogo (figura 2.27), bem como as químicas, o autor extrai as seguintes informações:

- O tamanho da chama – $L = \frac{h_f}{\sin A} = \frac{h_f}{\cos A_t}$.
- O ritmo de combustão – $C = \frac{H\omega_f}{t_R}$, onde $t_R = \frac{D}{r}$ (r é o ritmo de propagação da chama)
- A temperatura a uma determinada altura – $\boldsymbol{T(h) = \frac{\Delta T}{3.9 I^{2/3}}\Big/ h}$ **(47)** , onde ΔT é a diferença entre a temperatura ambiente e a sentida na zona de fogo, e I é a intensidade de fogo frontal (kW/m).

O conhecimento destes dados, permitem o desenvolvimento de sistemas que melhor classifiquem e modelem o desenvolvimento e consequências de um fogo florestal.

3

Trabalho Desenvolvido

Para alcançar os objetivos da presente dissertação, dividiu-se o desenvolvimento das soluções em várias etapas. Este capítulo focar-se-á em cada uma dessas etapas, fazendo-se um estudo prévio das tecnologias a serem utilizadas, seguido de uma explicação do que foi desenvolvido.

Deste modo, o primeiro passo passa pela construção de um protótipo, análise dos recursos disponíveis e capacidade dos mesmos. Posteriormente será necessário garantir uma boa qualidade na aquisição de imagens, sendo fulcral uma boa calibração de cada câmara e da visão estereoscópica, bem como garantir consistência nas características de captura de imagem para imagem. Em seguida, o foco será a deteção e posterior extração de características como a distância ou inclinação da chama.

3.1 Desenvolvimento do Protótipo

Para o desenvolvimento desta tese foi necessário desenvolver um protótipo de um dispositivo móvel de recolha e análise de imagem que, seja não só funcional, como de fácil transporte. Este inclui o uso de diversos equipamentos e componentes eletrónicos montados de forma cuidadosa para garantir a qualidade na aquisição de dados. Nas figuras 3.1 e 3.2, encontra-se a primeira versão do protótipo criado.

Figura 3.1 - Visão traseira da primeira versão do protótipo

Figura 3.2 - Face dianteira da primeira versão do protótipo

Face à primeira versão do protótipo, teve de ser feita uma adaptação do sistema apresentado, para garantir a qualidade na captura de imagens. Este novo sistema tem como objetivo aumentar a mobilidade face ao anterior, criando uma estrutura mais estável e rígida, com fácil acesso aos vários componentes eletrónicos, bem como às várias portas *USB* e *Ethernet*.

Como o objetivo desta tese é facilitar a forma como os dados sobre a frente de fogo são recolhidos, também é importante ter em conta o custo dos equipamentos utilizados. Ao utilizar componentes de "baixo custo" é possível a criação de um protótipo que permita testar a viabilidade dos algoritmos desenvolvidos. Por outro lado, torna-se a versão funcional do sistema mais acessível e potencia-se a recolha de dados para estudo dos incêndios florestais.

3.1.1 *Hardware* utilizado

Como o objetivo desta tese é o estudo da imagem estereoscópica na deteção e classificação de frentes de fogo, foi necessário um conjunto de equipamentos de *hardware* capazes de capturar imagens, e processá-las.

Por outro lado, também foi tido em conta o custo de construção do sistema, de forma a que uma versão operacional do mesmo seja acessível às várias corporações de bombeiros.

3.1.1.1 Microcomputador RaspberryPi 3B+

Como módulo central de processamento e controlo, foi escolhido um microcomputador *RaspberryPi* versão *3B+* (figura 3.3). Este equipamento vem equipado com um processador de 64bit a 1.4 *Ghz*, com uma memória *SDRAM* de 1 *Gb*. O sistema operativo é corrido num cartão *MicroSD* instalado na placa. Para comunicação, o dispositivo tem a capacidade de comunicar em *Wifi*, tanto a 2.4 *GHz* como a 5 *GHz*, ou através de *Blutetooth* 4.2 ou BLE. É ainda possível a ligação através da porta de *Ethernet*. Além disso, esta placa vem equipada com 40 pinos *GPIO*, 1 porta *CSI* para ligação com a câmara, 1 porta *DSI* para ligação com o ecrã tátil, 1 áudio porta estéreo de 4 polos de 3.5mm, 4 portas *USB* 2.0 e uma porta *HDMI* (Raspberry Pi Foundation, 2012).

Para maximizar a performance do sistema operativo, foi escolhido um cartão de classe 10 de 32 *Gb*.

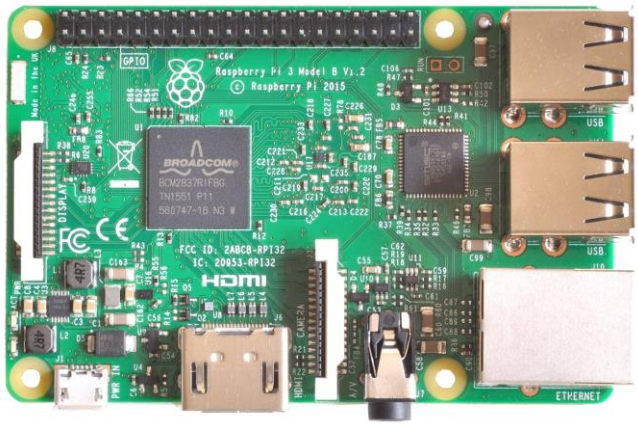

Figura 3.3 - Microcomputador Raspberry Pi 3B+

3.1.1.2 Câmaras Raspberry Pi V2

Para captura de imagem foram escolhidas duas câmaras *Raspberry Pi V2.* Estas câmaras vêm equipadas com um sensor de 8 megapixéis *Sony IMX219* capaz de fazer a captura de imagem ou gravação vídeo nos formatos 1080p30, 720p60 e VGA90. Este sensor tem uma resposta espectral compreendida entre os 400 e os 1000 nanómetros, num campo de visão com uma amplitude de 62.2 x 48.8 graus. Estas câmaras podem ser ligadas ao Raspberry, através da porta *CSI* do mesmo. Como o objetivo é o estudo de imagens estereoscópicas multiespectrais, foi escolhida uma câmara normal, e uma câmara *NoIR*.

A primeira, funciona numa gama espectral dos 400 aos 700 nanómetros, isto porque o sensor vem equipado com um filtro passa-baixo de 700 nanómetros. Esta será daqui em diante referida como câmara RGB.

No caso da câmara *NoIR*, não vem equipada com o filtro passa-baixo, funcionando na gama espectral dos 400 aos 1000 nanómetros. Para que esta câmara mais facilmente capte a radiação emitida por um incêndio, foi inserido um filtro passa-alto de 720 nanómetros, reduzindo a resposta desta câmara à gama dos 720 aos 1000 nanómetros. Esta câmara será daqui em diante referida com câmara NIR. Nas figuras 3.4 e 3.5, é possível verificar a resposta espectral de ambas as câmaras, considerando as alterações introduzidas pelos filtros.

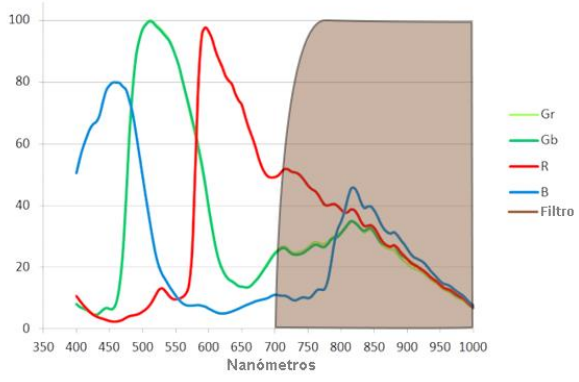

Figura 3.4 - Resposta Espectral da câmara RGB

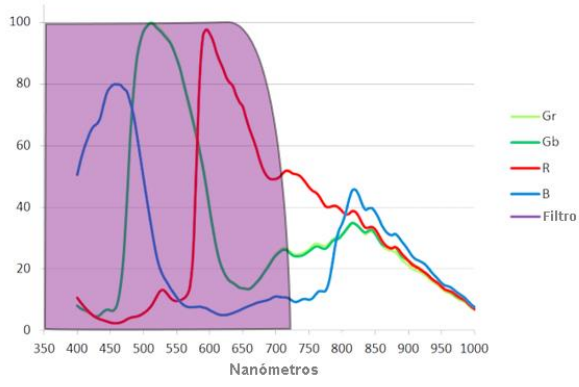

Figura 3.5 – Resposta Espectral da câmara NIR

3.1.1.3 Multiplexador IVPort Quad V2

O multiplexador utilizado foi o *IVPort Quad V2* (figura 3.6). Este multiplexador tem 4 portas *CSI*, e 4 pinos *GPIO* para controlo com o microcomputador, está desenhado para funcionar com os sensores apresentados anteriormente. O dispositivo usa o protocolo MIPI para transmitir as imagens/vídeo do multiplexador para o *Raspberry Pi*. Segundo anunciado pela marca, este dispositivo tem um atraso de 0.5 nanossegundos entre o pedido e a resposta, e demora 50ns a trocar de uma câmara para outra, não contando com atrasos do *GPIO* (Ivmech Mekatronik & İnovasyon, 2016).

Como o microcomputador utilizado tem apenas uma porta *CSI*, não é possível ligar duas câmaras em simultâneo, razão pela qual houve necessidade de incluir o multiplexador.

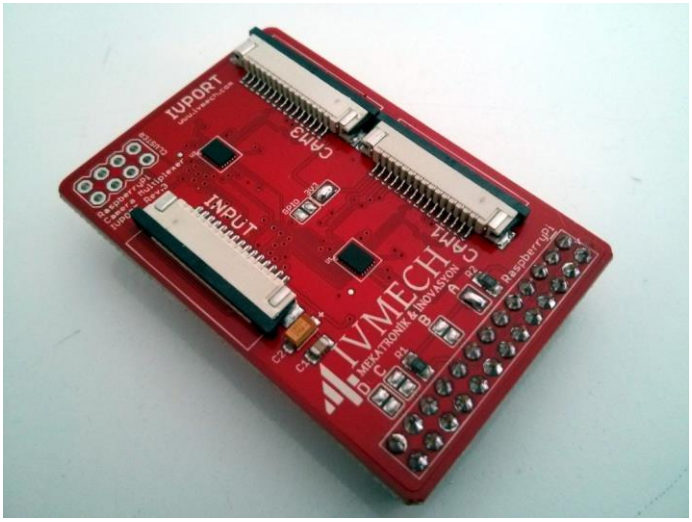

Figura 3.6 - Multiplexador *IVPort V2*

3.1.1.4 Ecrã tátil Raspberry Pi

Como parte desta dissertação passa pela aquisição de imagem, é importante garantir que o sistema está bem enquadrado com o cenário que se quer capturar. Assim, foi utilizado o ecrã tátil do *Raspberry Pi,* não só para visualização do cenário captado como para interface com o utilizador.

Este ecrã de 7 polegadas, tem uma definição de 800x480 pixéis, e liga-se ao microcomputador, através de uma porta *DSI* (para transmissão de imagem) e de 4 pinos *GPIO* (para alimentação e controlo de luminosidade). Na figura 3.7, é possível observar a visão traseira do ecrã, na qual é possível visualizar o controlador que vem integrado com o mesmo (Raspberry Pi Foundation, 2015).

Figura 3.7 - Ecrã tátil *Raspberry Pi*

3.1.1.5 Fonte de Alimentação

Para alimentação do sistema, foram usadas duas alternativas: através de um transformador de 5.1V/2.5A; e através de uma bateria externa de 5V/2A, com 13200 mAh. Note-se que a bateria externa tem características abaixo das desejáveis, no entanto, e apesar do ecrã dar aviso de baixa tensão, não há défices na performance do sistema. Esta é usada para dar mobilidade ao sistema. Após vários testes, esta teve uma autonomia média de 8 horas, suficiente para os objetivos desta dissertação.

3.1.2 Construção do protótipo

Como foi referido anteriormente, a construção do protótipo é uma etapa muito importante desta dissertação, porque tem de se garantir uma boa qualidade de aquisição de imagem. Na secção 2.1.1, é explicado que para uma correta calibração de uma visão estereoscópica é necessário um alinhamento horizontal das câmaras. Consequentemente é de grande importância garantir que haja o menor erro possível no posicionamento das câmaras.

Por outro lado, é importante que o equipamento construído seja resistente, de fácil mobilidade, e que garanta um fácil acesso aos diversos componentes que o integram.

Face a estes requisitos, optou-se pela construção da caixa em acrílico, por ser um material resistente e fácil de trabalhar. Assim, utilizou-se uma placa de acrílico de 60 por 30 centímetros com uma espessura de 5 milímetros, a qual foi cortada a laser para criação das várias peças. Na figura 3.8, é possível observar os cortes feitos a uma placa de acrílico para criação das várias peças da estrutura:

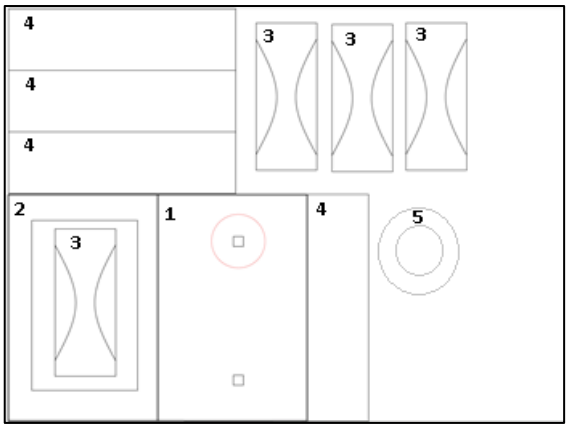

Figura 3.8 – Esquema de corte da estrutura.
1: Face frontal onde serão montadas a câmaras
2: Face traseira onde será montado o ecrã tátil
3: Pegas laterais (2 por cada lado)
4: Faces superior e inferior (2 por cada face)
5: Encaixe para segurar o filtro IR

Note-se que existe o dobro das peças para construção das faces frontais e das pegas, para dar mais espessura e consequente resistência à caixa. No caso das faces superior e inferior, só foi usada uma peça por cada, mas no caso das pegas laterais optou-se pela espessura de 1cm. Repare-se ainda que as furações não foram incluídas no corte a lazer, pois é mais fácil controlar onde serão fixados os parafusos já durante a montagem.

Um dos aspetos a ter em conta é o alinhamento das ranhuras onde serão montadas as câmaras (Figura 3.8.1). Além do correto alinhamento horizontal, é importante também garantir que ambas estão com a mesma orientação. Para tal fixou-se ambas as câmaras com porcas e parafusos, e nivelou-se a orientação das mesmas com 5 anilhas em cada parafuso, como é possível observar na figura 3.9.

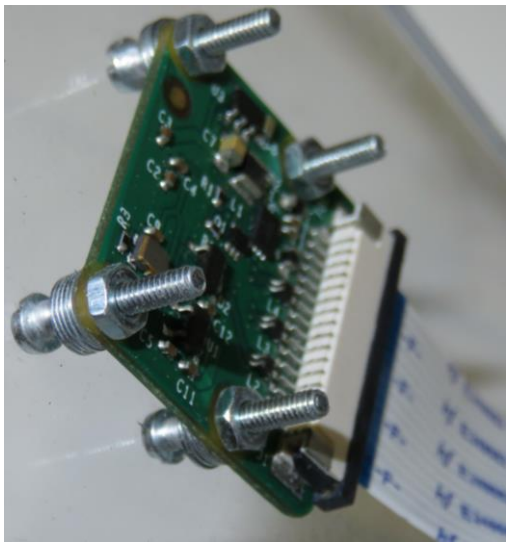

Figura 3.9 - Fixação da câmara no protótipo

Outro aspeto importante a ter em atenção é a forma como o ecrã e o filtro foram fixados. Para evitar uma grande complexidade do protótipo optou-se por fixar o ecrã e o filtro através da fricção entre o equipamento e o acrílico. Assim, os cortes feitos para a fixação destes equipamentos (figura 3.8.2 e 3.8.5), foram feitos abaixo do tamanho de cada peça, para que posteriormente através de uma lixa se possa ajustar o tamanho do encaixe ao tamanho da peça.

Teve-se ainda em consideração o fácil acesso aos vários componentes, e por isso, a face frontal está fixada ao resto do protótipo através de dobradiças, permitindo assim abrir e fechar a caixa conforme necessidade. Note-se ainda que a forma das pegas, permite a passagem de cabos do interior para o exterior da caixa.

Quanto à fixação das várias peças de acrílico, utilizou-se parafusos e cola para garantir a estabilidade da construção.

No produto final, as câmaras ficaram espaçadas de 13.3 centímetros. Nas figuras 3.10, 3.11 e 3.12, pode-se observar o aspeto final do protótipo produzido:

Figura 3.10 - Face Traseira do Protótipo

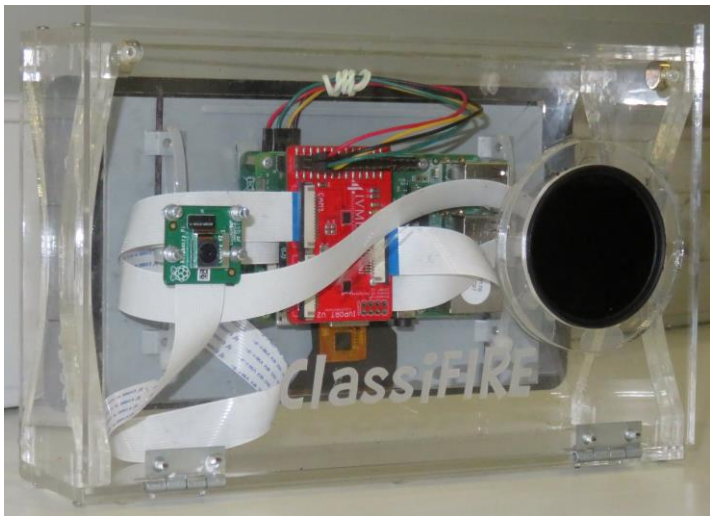

Figura 3.11 - Face Frontal do Protótipo

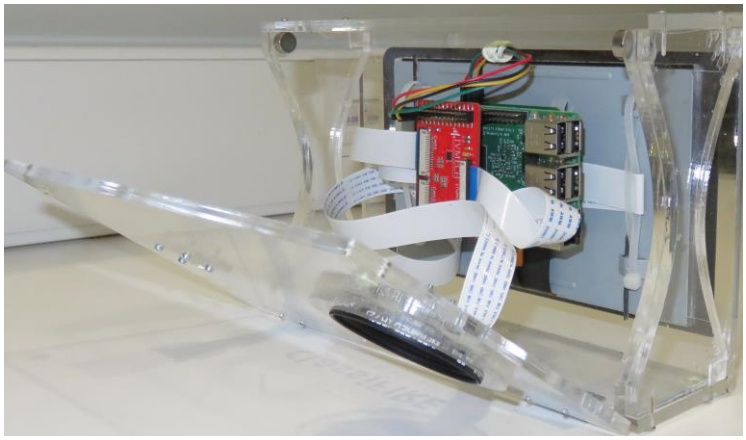

Figura 3.12 – Vista interior do protótipo com a face frontal aberta

3.2 Desenvolvimento do Software

A primeira parte do desenvolvimento dos vários algoritmos, passou pelo estudo prévio das várias bibliotecas de *software* a serem utilizadas, seja para controlo do *hardware* ou processamento de dados.

3.2.1 Bibliotecas utilizadas

No âmbito desta dissertação, faz sentido fazer a análise tanto das bibliotecas do multiplexador como das bibliotecas das câmaras. Sendo estes os meios de controlar o acesso às câmaras e a captura de imagens, é importante, estudar as suas capacidades e limitações. Assim, durante o desenvolvimento, foi necessário adaptar as várias tecnologias e técnicas utilizadas, e também de adaptar as várias tecnologias ao objetivo deste sistema, nomeadamente o as bibliotecas do multiplexador utilizado. Assim, a secção seguinte irá focar-se nas bibliotecas do multiplexador *IVPort V2.*

3.2.1.1 Bibliotecas *IVPort V2*

O uso do multiplexador foi uma tarefa desafiante. Como descrito nas bibliotecas fornecidas, "o dispositivo não traz qualquer garantia associada, nem mesmo garantia de comerciabilidade ou de adequação a qualquer propósito" (Ivmech Mekatronik & İnovasyon, 2016). Este fator, associado a uma escassa quantidade de utilizadores deste dispositivo, cerca de 500 como anunciado pela marca, tornou difícil ultrapassar certos obstáculos. Em adição, as bibliotecas do dispositivo são muito básicas e sem correção de erros que frequentemente ocorrem. Por ser um elemento chave neste sistema de visão estereoscópica é necessária uma correta análise das capacidades e defeitos, razão pela qual foi criada uma secção dedicada ao mesmo.

Como referido anteriormente, as bibliotecas fornecidas com o *IVPort V2* não são as mais adequadas, e como tal houve necessidade fazer alterações e adições ao código disponibilizado.

O código fornecido é composto por um ficheiro para funcionamento com o *hardware* e um de exemplo, nos quais são fornecidas as funções:

- *camera_open(resolution, framerate, grayScale)* – função básica, para instanciar a variável responsável pela comunicação com a câmara. Nela é possível definir

a resolução e os *frames* por segundo usado, através do tuplo de inteiros ***resolution***, e do número real ***framerate***. É ainda possível a escolha entre uma imagem a cores ou em escala de cinza, através do boleano ***grayScale*** (Falso=Cores; Verdade=Cinza).

- ***close ()*** – função básica, para quebra da ligação à câmara.
- ***change_camera(camN)*** **–** função básica, para trocar entre câmaras, onde *camN* é um número inteiro que identifica a câmara a ativar, e pode tomar os valores de 1 a 4. Neste caso concreto, usa-se o 1 para a câmara RGB e o 2 para a câmara NIR.
- ***camera_capture(fileName,**options)*** – função básica, para captura da imagem obtida pela câmara ativa no momento. A função auxilia-se da função *capture()* presente nas bibliotecas da câmara, para capturar e gravar em disco uma imagem no formato *jpeg*. Através da *string* ***fileName*** é possível especificar o prefixo do nome do ficheiro criado (Ex: para uma captura da câmara *RGB* com o nome "imagem" seria criado o ficheiro "imagem_CAM1.jpg"). Quanto à variável opcional ***options***, esta permite aceder a funcionalidades extra da função *capture()*, nomeadamente se é usada uma porta vídeo para captura rápida, ou qual o redimensionamento a ser aplicado à imagem, entre outros.
- ***camera_sequence(**options)*** – função básica para a captura sequêncial de imagens. Esta função auxilia-se da função *capture_sequence()* presente nas bibliotecas da câmara, para captura sequencial de imagens. A variável ***options*** é usada para especificar os parâmetros de entrada da função *capture_sequence.* Através desta, é necessário especificar onde serão gravadas as imagens obtidas. Opcionalmente, é possível escolher o formato de imagem, optar pelo uso da porta vídeo ou da captura em rajada entre outros.
- ***still_capture()*** – função "avançada" para captura de um par de imagens das câmaras 1 e 2. Esta função faz a captura através de comandos de sistema, gravando a imagem com o nome *"still_CAMx.jpg"*. Além disso, é usada a função *camera_change()* para trocar entre as duas câmaras.
- ***picam_capture()*** – função "avançada" para captura de um par de imagens das câmaras 1 e 2. Esta função à semelhança da anterior, usa a função *camera_change()* e captura a imagem através da função *camera_capture()* especificada anteriormente.

- ***picam_sequence()*** – função "avançada" para captura sequencial de 15 pares de imagens das câmaras 1 e 2. Esta função, também faz uso da função *camera_change()* e usa a função *camera_sequence()* especificada anteriormente. Em adição, esta função define um construtor especificamente para a criação de nomes para os ficheiros *jpeg* criados. De notar, que esta função não produz bons resultados, pois das 30 imagens recolhidas, apenas 5 a 8 pares de imagens podem ser aproveitados, pois nos outros são recolhidos pares de imagens da mesma câmara

Ainda que forneça funções para o uso básico do multiplexador, a biblioteca falha em muitas áreas: Não é possível a captura diretamente para uma variável, obrigando à leitura do disco para acesso às imagens. Além disso, não existe forma de especificar parâmetros da câmara como a sensibilidade do sensor, a velocidade do obturador, balanço de brancos, entre outros. Mesmo especificando, a biblioteca não está preparada para a necessidade de configurar as diferentes câmaras com diferentes características.

Em adição, há erros inerentes ao dispositivo, para os quais não foi possível encontrar solução. Um dos erros, que mais afetou a capacidade do sistema, foi o facto de não ser possível utilizar as câmaras na mais alta definição, com a consequência de se obter uma imagem com faixas com características de cor diferentes (figura 3.13).

Figura 3.13 - Imagem capturada a 1920x1080

Assim, uma das consequências do uso deste dispositivo foi o facto de não ser possível obter resoluções superiores a 1080x720 pixéis.

Face aos diversos problemas identificados, houve a necessidade de não só alterar como adicionar funcionalidades à biblioteca. Assim, foram feitas as seguintes alterações:

- Foi adicionada a capacidade de captura de imagens em memória, subtraindo-se a necessidade de leitura do disco para gravar as imagens em variáveis. Assim, foi adicionada a função *tempCapture(),* semelhante à função *camera_capture(), e a função tempSequence*, semelhante à função camera_sequence(). Em relação à ultima, houve ainda a necessidade de adaptação do construtor da função *picam_sequence()*, com objetivo de retornar variáveis para captura das imagens.
- Foi adicionada a capacidade de especificar as características de obtenção de imagem, para cada uma das câmaras. Assim, é possível definir o *iso*, modo de exposição, a velocidade do obturador, o modo de balanço de brancos, ganhos para balanço de brancos, a métrica para balanço, entre outros.

As alterações introduzidas, que se encontram em anexo, não só facilitam como agilizam a programação do sistema de visão estereoscópica. Assim, foi possível obter velocidades de captura de 1,3 par de *frames* por segundo a uma resolução de 1080x720 pixéis, quando testado no interpretador *Python Idle* na versão 3.5.3.

Face às características do multiplexador e das respetivas bibliotecas, tendo em conta as técnicas estudadas no capítulo 2, conclui-se que:

- Não é viável a classificação com base no movimento da chama, apresentado em 2.2.3. Com uma velocidade de captura de 1,3 fps não é possível verificar corretamente o movimento da chama. Consequentemente a criação da matriz binária dos pixéis em movimento, que é fruto da análise de *frames* consecutivos, torna-se incerta, pois de uma imagem para a outra existem alterações muito significativas.
- A captura de imagens estereoscópicas é impossível, nem mesmo no conceito de "*pseudo-estereoscopia*"[2]. Assim, as várias técnicas de correspondência estereoscópica ficam limitadas apenas às áreas da imagem em que não há alterações do motivo de uma imagem para a outra. Em casos

[2] O conceito de "*pseudo-estereoscopia*" refere-se aos casos em que não é possível a captura simultânea das imagens de ambas as câmaras, mas a diferença temporal entre capturas não é significativa para provocar gerar imagens com diferenças não associadas à estereoscopia.

específicos em que a maior parte da captura está a arder, existe muito movimento e não se torna viável a utilização as técnicas apresentadas no capítulo **2.1.3**.

- A troca constante de uma câmara para a outra torna a sequência de captura não uniforme no que toca à luminosidade e cor das fotos. Como o sensor da câmara está num permanente "liga e desliga", os sensores não conseguem estabilizar. Como consequência, obtém-se imagens com cores e iluminações diferentes. Ou seja, para detetar fogo, ou se usa um algoritmo dinâmico de deteção sensível às condições de captura, ou então uniformiza-se o máximo possível essas condições. Daí ser tão importante a funcionalidade de definir as características de captura.
- Face às limitações na resolução, e tendo em conta o protótipo construído, é possível calcular o alcance do sistema de medição de distância a ser desenvolvido. Em **2.1.4**, é apresentada uma equação para o cálculo da distância. Tendo em conta que as câmaras estão distanciadas 13.3 centímetros uma da outra, que a câmara tem um campo de visão horizontal de 62.2º, e que a resolução máxima que se consegue obter é de 1080x720, assumindo que dois objetos distam 1 pixel um do outro:

$$\boldsymbol{D = \frac{B x_0}{2 \tan\left(\frac{\varphi_0}{2}\right) |x_1 - x_2|}} = \frac{13.3 \times 1080}{2\tan\left(\frac{62.2\pi/180}{2}\right) \times 1} \approx 119\ metros$$

Tenha-se em atenção que esta distância máxima não é realista, isto porque a essa distância o erro de um pixel gera um erro de quase 60 metros. Assim, e considerando possíveis erros, a distância máxima a que se pode obter uma distância são 30 metros. Ainda assim, este valor está muito dependente de pequenos movimentos da câmara ou do cenário.

3.2.1.2 Bibliotecas *Picam*

O sistema foi equipado com duas câmaras – uma no espectro visível, com resposta espectral do sensor compreendida entre os 400 e os 700 nanómetros; outra no espectro de visível mais NIR, com uma resposta espectral dos 400 aos 1100 nanómetros. À segunda câmara, foi ainda adicionado um filtro passa alto de 700 nanómetros, reduzindo a resposta espectral dos 700 aos 1100 nanómetros. Em seguida encontram-se as principais características de captura das câmaras, (Jones, 2016):

- **Modo de exposição** – o modo de exposição pode ser configurado para vários modos padrão existentes, através do qual os valores de ganho digital e analógico das câmaras são ajustados. É importante ter em atenção (Jones, 2016)que as câmaras não devem ser ligadas com este modo em *off*, pois os valores dos ganhos, por definição, começam em valores muito baixos e só depois estabilizam nos devidos valores. Como consequência, as imagens poderão aparecer pretas.
- **Métrica de exposição** – o modo como a exposição é calculada. Este modo ajusta a forma como são calculados os valores de exposição, tendo em conta diferentes percentagens da imagem.
- ***iso*** – a sensibilidade à luz pode ser ajustada aos valores de 0 (modo automático), 100, 200, 320, 400, 500, 640, 800. Ainda é possível obter valores de sensibilidade de 1600 com a exposição em modo desporto.
- **Modo de Balanço de brancos –** este modo pode ser configurado para vários modos padrão existentes, através do qual os ganhos atribuídos aos canais vermelho e azul são definidos automaticamente, conforme as condições de luz e motivos da imagem.
- **Ganhos de balanço de brancos –** estes valores, para os canais vermelho e azul, podem ser ajustados entre 0 e 8. No entanto normalmente, estes valores estão situados entre os 0.9 e 1.9. Estes valores são responsáveis pelo ajuste da cor da imagem, dando mais predominância de um ou outro canal, face aos outros.
- **Porta vídeo –** é possível escolher entre captura através da porta de imagem ou de vídeo. O uso desta porta torna a captura mais rápida, mas com maiores flutuações de cor e luminosidade.
- **Resolução –** este valor pode ser ajustado até a um valor máximo de 2592x1944 pixéis.

- **Velocidade de captura –** este valor pode ser ajustado, até um valor máximo de 90fps.
- **Velocidade de obturação –** este valor indica, em microssegundos, o tempo durante o qual o sensor fica exposto à luz. Este valor pode ser ajustado até um máximo igual ao inverso da velocidade de captura. Por exemplo, com 20 fps o valor máximo para o tempo de obturação seria 50 000 segundos.
- **Modo do Sensor** – dependendo dos valores adotados para a resolução e velocidade de captura, o sensor ajusta-se a um determinado modo. Na tabela 3.1, pode-se ver as características de cada um deles:

Tabela 3.1 – Modos do Sensor

Modo	Resolução	Formato	Velocidade de Captura	Campo de Visão	Gravação
1	1920x1080	16:9	1-30 fps	Parcial	Vídeo
2	2592x1944	4:3	1-15 fps	Completo	Vídeo, Imagem
3	2592x1944	4:3	0.1666-1 fps	Completo	Vídeo, Imagem
4	1296x972	4:3	1-42 fps	Completo	Vídeo
5	1296x730	16:9	1-49 fps	Completo	Vídeo
6	640x480	4:3	42.1-60 fps	Completo	Vídeo
7	640x480	4:3	60.1-90 fps	Completo	Vídeo

Além destas características, é ainda possível ajustar o zoom, contraste, luminosidade, adicionar efeitos à imagem, entre outros, no entanto para diminuir a quantidade de variáveis a ter em consideração, só os anteriores serão considerados.

Em adição às funções de captura, cujas características acabaram de ser apresentadas, existe ainda a função de abertura da câmara. Sempre que se quer utilizar, é necessário criar uma instância da câmara para que se abra a conexão entre a câmara e o *Raspberry Pi*.

Surge então a questão - para usar ambas as câmaras, é necessário instanciar cada uma delas? – Seria de interesse, mas não é possível. Seria de interesse porque, ao instanciar duas câmaras, pode-se definir automaticamente as características de captura de cada câmara, e só se teria de o fazer uma vez.

Não é possível porque, apesar de a função de instanciação ***Picamera( )*** ter o parâmetro de entrada opcional ***camera_number*** , usado quando se pretende instanciar mais que uma câmara, este apenas se encontra operacional nos módulos de computação *Raspberry Pi*. Consequentemente, e como foi referido na secção anterior, as características de captura têm de ser alteradas sempre que se troca de câmara. Assim, conclui-se:

- Na escolha do tipo de porta, tem de haver um balanço entre a consistência na aquisição de dados, e o ritmo de aquisição. Por um lado, ao não usar a porta vídeo, obtém-se uma maior consistência nas imagens adquiridas. Por outro, diminui-se significativamente o ritmo de captura, não só por causa do tipo de porta, mas também pelo facto de as imagens capturadas na porta de imagem terem de ser convertidas de RGB para BGR, o que acaba por diminuir ainda mais o ritmo. Assim, foi escolhida a porta vídeo.
- Como foi necessário escolher a utilização da porta vídeo, tem-se de garantir a consistência. Por isso, não é aconselhável o uso dos modos de exposição e de balanço de brancos fora do *'off'*. Estes modos, apesar de facilitarem o trabalho de configuração das características das câmaras, são modos em que os valores de sensibilidade do sensor, velocidade do obturador e balanço de brancos são dinamicamente alterados, conforme as condições do cenário. Assim, e como foi referido na secção anterior, esses valores são fixados para diminuir variações de cor/luminosidade entre imagens.
- Quanto maior a resolução, maior o tempo de captura - mais uma vez, tem que existir um balanço entre a qualidade e a velocidade de captura. Neste caso, e como é possível verificar pela equação (23) apresentada no capítulo 2.1.4., quanto maior for a resolução, maior é a precisão de cálculo das distâncias, pois diminui-se o erro de quantização da localização de um objeto na imagem. Assim, foi escolhido o valor máximo de resolução suportado pelos vários hardwares- 1080x720 pixéis.
- A biblioteca *Picam* vem com diferentes tipos de captura: ***camera_capture, capture_continuous, capture_sequence.*** O mais simples é o primeiro em que apenas uma imagem é capturada, sendo os outros para captura de múltiplas imagens. Ainda que as últimas sejam as mais rápidas na captura de imagens sem o uso do multiplexador, quando este é usado, não só a velocidade de captura diminuir, como perde-se a fiabilidade na aquisição imagens.

3.2.1.3 Outras bibliotecas

Houve ainda a necessidade de incluir mais bibliotecas que auxiliassem no processamento de imagem. Uma das bibliotecas mais conhecidas é a biblioteca ***OpenCV*** *(Open Source Computer Vision Library)*. Esta biblioteca incluí mais de 2500 algoritmos de processamento de imagem e *machine-learning*. Esta biblioteca foi fundamental no desenvolvimento dos algoritmos implementados, não apenas por ser uma biblioteca otimizada para o processamento digital de imagem, como está repleta de funções de calibração, desenho, filtragem, realce de contornos, entre outros.

Outro recurso também utilizado, foi a biblioteca ***Numpy***. Esta biblioteca, está preparada para operar eficientemente com vetores multidimensionais. O uso dela, torna as operações entre matrizes, muito utilizada nos algoritmos implementados, mais rápida em comparação com métodos das bibliotecas *OpenCV*. A biblioteca incluí funções de criação de matrizes nulas, transposição, separação, junção, comparação entre outros.

Para criação de simples interfaces gráficas, foi utilizada a biblioteca ***Tkinter***. Esta biblioteca é o *GUI* (*Graphical User Interface*) mais utilizado na linguagem *Python*, consequência de ser uma ferramenta leve, estruturada em camadas, fácil de usar e com funcionalidades uteis. Através desta biblioteca é possível criar janelas, inserir texto, imagens, listas, botões, entre outros.

Os algoritmos implementados, foram escritos na linguagem *Python.* Esta linguagem, tem uma sintaxe simples, sem necessidade de declarar variáveis, no entanto é uma linguagem que é interpretada, em vez de compilada. O computador, quando executa um *script Python,* interpreta cada linha para obter o conjunto de ações a efetuar, em vez de o compilar. Isto faz com que o processo de execução do *script* seja mais lento em comparação com linguagens compiladas. A linguagem *C* por exemplo, é uma linguagem compilada, ou seja, o código é integralmente convertido para linguagem máquina para ser posteriormente executado. Ainda que nestas linguagens haja sempre necessidade de compilar o programa antes de o correr, as diferenças de performance são muito significativas, para se escolher a compilação à interpretação.

Deste modo, foi utilizada a biblioteca ***Cython*** que permite aplicar uma compilação *C* a uma linguagem que deveria ser interpretada. Assim, mantém-se os benefícios da utilização de uma linguagem de "médio-alto" nível, com a performance das linguagens de "médio-nível".

3.2.2 Calibração do sistema

Face ao objetivo de construir uma visão estereoscópica, é necessário não só garantir a correta calibração de ambas as câmaras, mas também a própria estereoscopia. Cada câmara tem uma distorção inerente à sua lente, que deve ser corrigida. Também a posição de cada câmara influencia a forma como a estereoscopia vê o mundo, e essa perspetiva de visão deve ser calculada. Apesar de ser um processo relativamente simples, é importante garantir uma consistência da visão estereoscópica.

3.2.2.1 Calibração das câmaras

Primeiramente, há que fazer uma calibração de cada câmara. O algoritmo desenvolvido usa as bibliotecas *OpenCV*, e é feito com base na imagem de um tabuleiro de xadrez (OpenCV, 2019).

O algoritmo pretende calcular os fatores das distorções radial e tangencial provocadas pela lente da câmara. Para tal, começa-se por capturar imagens de um tabuleiro de xadrez. É importante que o tabuleiro apareça por completo na imagem, ou caso contrário não será possível encontrar todos os vértices. Na calibração feita, foi utilizado um tabuleiro de damas (8x8 células), com cada célula a medir 3,2cm. Estes dados são importantes pois para que a função *findChessboardCorners* possa encontrar todos os vértices, é necessário informar sobre o formato e dimensões do tabuleiro. Esta função requer ainda que se informe sobre a forma que se espera encontrar, que para o caso da calibração através do tabuleiro de xadrez é uma matriz inicializada a zero. Através desta função obtêm-se a localização *(x,y)* de todos os vértices encontrados.

Naturalmente, a calibração não pode ser efetuada com apenas uma imagem, como tal tem de se capturar múltiplas imagens e compilar todos os pontos encontrados, numa estrutura. Essa estrutura será posteriormente utilizada no cálculo das matrizes de distorção e da câmara. A primeira é uma matriz que contem 5 coeficientes que informam sobre a distorção. A última contém a posição do centro e distância focal da câmara.

3.2.2.2 Calibração estereoscópica

Após calculadas as matrizes de distorção e da câmara, para cada uma das câmaras, *RGB* e *NIR,* há que efetuar agora a calibração da estereoscopia. As estruturas criadas com a posição de todos os vértices de todas as imagens, de cada câmara, vão ser usados no seguinte passo, bem como as matrizes de ambas as câmaras. Através da função ***stereoCalibrate***, vão ser calculadas 4 matrizes que informarão sobre as características da visão, e de cada câmara. As matrizes calculadas são:

- Matriz Rotação – informa sobre a rotação existente entre os sistemas de coordenadas, da primeira para a segunda câmara.
- Matriz Translação – informa sobre a translação existente entre os dois sistemas de coordenadas.
- Matrizes Essencial e Fundamental– informam sobre a relação entre pontos correspondentes na visão estereoscópica.

Estas matrizes são gravadas em disco para fazer a retificação das imagens, sem necessidade de haver uma calibração cada vez que se usa a o sistema.

3.2.2.3 Algoritmo de calibração implementado

O desenvolvimento do presente algoritmo pretendeu agilizar a forma como a calibração da câmara é feita. Nos diversos algoritmos existentes *online*, a calibração e a aquisição das imagens para calibração, são feitas em algoritmos separados. Como consequência, não é possível verificar, imediatamente, se as imagens recolhidas são aceitáveis para a calibração. Mais, como a calibração deve ser feita com cerca de 20 imagens, é possível que não se consiga obter a quantidade desejada, obrigando a uma nova recolha de imagens.

Assim, para colmatar essa necessidade, este algoritmo junta a aquisição com a calibração. Com auxílio de uma interface gráfica, que se pode observar na figura 3.14, foi implementado o seguinte algoritmo:

1. Captura-se o tabuleiro de xadrez através das duas câmaras. O utilizador é avisado, através de uma contagem regressiva a partir de 3, do momento em que as imagens serão capturadas

2. Faz-se a deteção dos vértices em cada imagem. Se todos os vértices, forem encontrados em ambas as câmaras, apresentam-se as imagens com os vértices detetados (figura 3:14). Caso contrário é apresentado um aviso informando em que câmara(s) ocorreram erros, e volta-se ao passo 1.

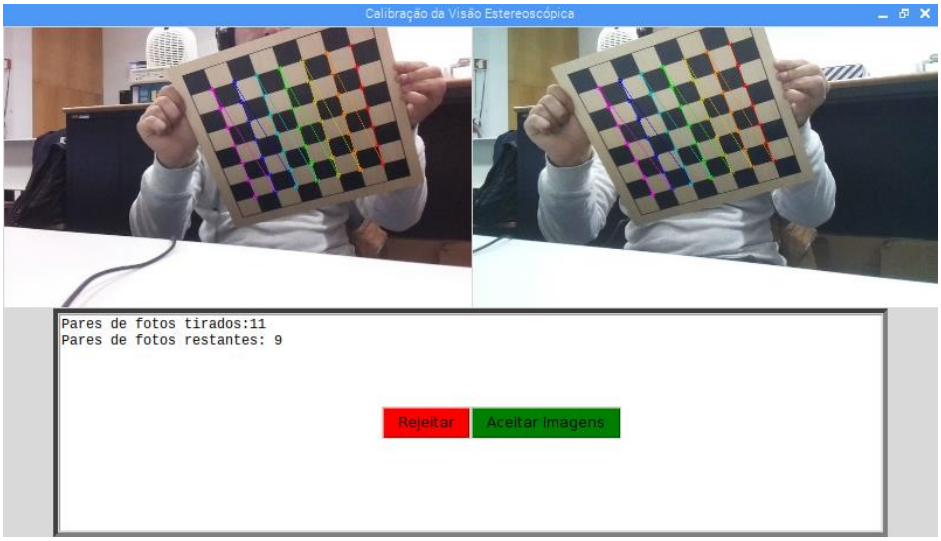

Figura 3.14 - Interface do algoritmo de calibração da visão estereoscópica

3. O utilizador verifica se os vértices foram corretamente calculados, e aceita/rejeita o par de imagens. Quando um par é aceite, a localização dos vértices em cada imagem é adicionada à respetiva estrutura. Quando o par é rejeitado, volta-se ao passo 1.
4. Quando os 20 pares de imagens forem obtidos, e respetivos vértices calculados, é feita a calibração de cada câmara, e obtidas as respetivas matrizes de distorção e da câmara.
5. É feita a calibração da visão estereoscópica, e obtidas as Matrizes de rotação, de translação, fundamental e essencial do sistema de visão estereoscópica.
6. São guardadas as anteriores matrizes em disco, para uso futuro.

Através das matrizes calculadas, é então feita a devida retificação das imagens. Esta retificação é importante para garantir o correto alinhamento dos vários objetos da imagem, em ambas as câmaras. Para efetuar essa correção, foram usadas as funções *stereoRectify(), initUndistortRectifyMap() e remap()* das bibliotecas *OpenCV*.

3.2.3 Recolha de Dados

Sendo este um sistema único, que implementa a estereoscopia em imagens de gamas espectrais diferentes, não existem repositórios de imagens online que nos auxiliem no seu desenvolvimento. O propósito da presente dissertação é aplicar a estereoscopia na deteção de fogos florestais. No entanto, esta foi desenvolvida durante os meses frios, na ausência de fogos florestais no território nacional, o que impossibilitou a captura de imagens de incêndios em floresta.

Em alternativa, foram realizadas três recolhas separadas, em que se simulou um pequeno incêndio através de uma fogueira. Deste modo, foi possível testar diferentes condições de fogo, do cenário e de captura da câmara.

No que toca à câmara NIR, não existem muitos problemas na recolha de imagens. A radiação existente no espectro infravermelho próximo vem essencialmente da radiação solar e das chamas. Por isso existe um bom contraste entre o fundo e as chamas. Mais, as imagens da câmara NIR podem ser reduzidas a uma gama só, isto porque não existem diferenças significativas entre as gamas vermelha, verde e azul. No que toca à câmara RGB, existe pouco contraste entre a chama e o fundo da imagem. A própria radiação solar introduz ruído na imagem, tornando o fogo presente nela pouco percetível. Na imagem seguinte, verifica-se a dificuldade em percecionar a chama, quando esta era perfeitamente vista a olho nu.

Figura 3.15 - Imagem de uma fogueira a 20 metros

A primeira recolha foi efetuada com uma fogueira com 3 metros de largura, e 0,9 metros de altura. As câmaras foram configuradas nos modos automáticos, e as capturas feitas através da porta vídeo, a uma resolução de 640x480 e uma frequência de 0,6 pares de imagens por segundo. Durante a recolha, foram capturadas imagens às distâncias de 10, 12, 15, 17, 20, 21, 25 e 29 metros A resolução foi escolhida tão baixa, porque após contacto com a empresa que produz o multiplexador, foi aconselhado que se usasse a câmara nesta resolução para aumentar a frequência de captura. Deste teste resultaram dados inconsistentes nas condições de captura, onde não só não era percetível a chama, como os balanços de brancos feitos automaticamente pela câmara, resultaram em imagens esverdeadas. O resultado destes testes pode-se verificar nas figuras 3.16 e 3.17, onde se consegue visualizar a chama na câmara NIR e não na RGB.

Figura 3.16 – Fogueira capturada a 15 metros com a câmara NIR

Figura 3.17 - Fogueira capturada a 15 metros com a câmara RGB

A segunda recolha foi efetuada com uma fogueira de 2 metros por 1,3, com o objetivo de que as chamas ficassem mais altas, e mais visíveis. Nesta recolha não foram assumidos nenhum dos modos automáticos: a sensibilidade do sensor foi configurada a 100 para ambas as câmaras; a velocidade de obturação foi fixada nos 100 nanossegundos para a câmara NIR e 200 nanossegundos para a câmara RGB, e o balanço de brancos a 1,8 para o vermelho e 1,3 para o azul.

Nesta segunda e na terceira recolha, fez-se uma limpeza do terreno à volta da fogueira para reduzir a quantidade de verde existente na imagem, e reduzindo a quantidade de interferência na cor da chama. Também foram colocados vários pontos de gravação às distâncias de 25, 20, 15,10, 7,5 e 5 metros. Estes pontos de gravação foram ajustados com um nível, de forma a garantir o máximo alinhamento das câmaras. Na figura 3.18, pode-se observar a forma como foram montados os vários postos, face à localização da fogueira. Note-se que além dos apresentados na figura, existem ainda mais três postos a 5, 7,5 e 10 metros de distância.

Figura 3.18 - Esquema de captura da fogueira utilizado nas recolhas 2 e 3

Na segunda recolha de dados, já foi possível adquirir imagens com uma resolução de 1080x720 pixéis, a uma frequência de 1,3 pares de imagem por segundo. As imagens obtidas, apresentam grande consistência, fruto da configuração manual das câmaras, mas continuam a não apresentar a chama facilmente.

Figura 3.19 – Secção de uma imagem capturada a 10 metros com a câmara NIR

Figura 3.20 - Secção da imagem capturada a 10 metros com a câmara RGB

Nas figuras 3.19 e 3.20, encontra-se um par de imagens capturadas a 10 metros. Estas imagens não estão com o tamanho original, pois foram recortadas para mais fácil visualização da fogueira. Mais uma vez, existe muito pouco contraste entre a chama e o resto da imagem, o que torna a deteção fragmentada em pequenas zonas detetadas, em vez de uma zona única de deteção. Por outro lado, também se verificou que as imagens obtidas, estão pouco nítidas, consequência de um tempo de obturação baixo.

Chega-se à conclusão que, uma fogueira feita apenas com troncos de madeira, não tem energia suficiente para fazer a sua chama destacar-se face ao resto da imagem. O objetivo é obter imagens em que o fogo seja semelhante àqueles em filmagens de incêndios florestais. Por outro lado, foi possível obter uma consistência na captura, muito maior face à primeira recolha, mas há que melhorar a nitidez das imagens.

O problema agora está em arranjar forma de tornar a chama visível, sem necessidade de se efetuar capturas à noite.

A solução encontrada na terceira recolha de dados, foi utilizar petróleo para aumentar a energia existente na fogueira e destacar a chama face ao fundo da imagem. Também, para resolver o problema de nitidez das imagens, foi necessário diminuir a velocidade de obturação.

Assim, a terceira recolha foi efetuada com uma fogueira de 1,6 metros de comprimento por 1,5 de altura, com o objetivo de aumentar ainda mais o tamanho das chamas. Não foram assumidos nenhum dos modos automáticos: a sensibilidade do sensor foi configurada a 100 para ambas as câmaras; a velocidade de obturação foi fixada nos 400

nanossegundos para a câmara NIR e 500 nanossegundos para a câmara RGB, e o balanço de brancos a 1,8 para o vermelho e 0,9 para o azul. Á semelhança da anterior recolha, as imagens foram capturadas através da porta vídeo, a uma frequência de captura de 1,3 pares de imagens por segundo, e resolução de 1080x720 pixéis.

As figuras 3.21 e 3.22 mostram como a adição de petróleo resolveu os problemas de percetibilidade da chama. Repare-se que em comparação com as imagens apresentadas anteriormente, obtém-se muito maior nitidez da chama, mesmo a distâncias superiores, e sem recortes na imagem.

Figura 3.21 – Imagem da fogueira, obtida a 25 metros na terceira recolha, através da câmara NIR

Figura 3.22 – Imagem da fogueira, obtida a 25 metros na terceira recolha, através da câmara RGB

Estes são os dados que foram utilizados para o desenvolvimento dos algoritmos de deteção e correspondência estereoscópica de segmentos. É de notar que, apesar da qualidade muito superior destes dados face aos recolhidos anteriormente, continuam a existir defeitos impossíveis de corrigir, nomeadamente diferenças do cenário. Nas figuras 3.21 e 3.22, pode-se verificar diferenças significativas na forma das chamas de ambas as imagens. Este defeito é consequência da impossibilidade de se criar um sistema verdadeiramente estereoscópico com o hardware disponível.

3.2.4 Deteção de Fogo

Na deteção de fogo, foi necessário adaptar algoritmos estudados no capítulo 2.2. Como foi referido anteriormente, qualquer algoritmo que tenha por base o movimento da chama, não é viável para o sistema em utilização, pois o par de imagens é captado com frações de segundo de diferença entre as câmaras. Como consequência a chama aparecerá com diferenças significativas entre as imagens obtidas através das câmaras RGB e NIR. Por outro lado, métodos de deteção baseados em análise probabilística das características da chama requerem a construção de grandes bancos de dados, devidamente classificados. Este facto é um entrave à construção de um algoritmo do gênero, devido à dificuldade de capturar fogos durante os meses frios, e de classificar centenas de imagens manualmente.

Assim, a deteção de fogo teve de ser baseada numa árvore de decisão que verifica várias condições de cor características de um pixel com chama. Como base do desenvolvimento do algoritmo de deteção em RGB, foram testadas várias características de cor estudadas em 2.2.1 e 2.2.2. A secção seguinte foca-se no desenvolvimento desse algoritmo.

3.2.4.1 Deteção RGB

Como referido, foram testados os algoritmos estudados no capítulo 2. Estes algoritmos têm uma particularidade que os torna menos bons para o objetivo de deteção de fogo - são algoritmos que pretendem não só detetar como evitar ao máximo que haja falsos positivos, que podem levar a um alerta de incêndio desnecessário. Como consequência desse tipo de algoritmo são geradas imagens extremamente segmentadas, como se pode verificar na figura 3.23.

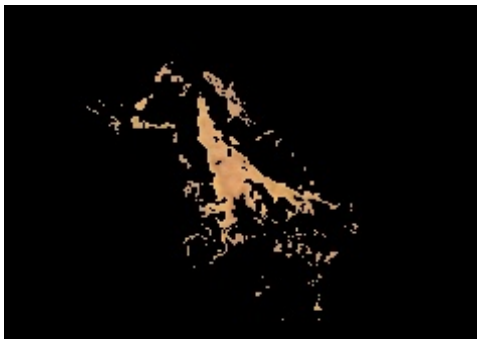

Figura 3.23 - Deteção de fogo com muita segmentação

Este tipo de classificação pode ser indicado para sistemas a operar numa única gama espectral, visto que garante uma taxa de falsos positivos baixa, quando existe informação limitada na gama. No entanto, como o algoritmo de deteção proposto nesta dissertação está sustentado por uma deteção no espectro NIR, e pretende fazer a deteção de zonas de interesse para correspondência estereoscópica, não existe problemas em se obter taxas de falsos positivos um pouco mais alta, e garantir taxas de verdadeiros positivos muito altas.

Em adição, estes algoritmos, apesar de não terem uma complexidade muito alta, demoram muito tempo a serem interpretados e executados em *Python*, diminuindo as capacidades de tempo-real de um sistema com uma captura de imagens lenta.

Assim, para resolver este problema, foi adaptado o algoritmo estudado em 2.2.1, com o objetivo de diminuir a segmentação, aglomerar zonas de deteção contiguas que correspondam ao mesmo foco de fogo e diminuir o tempo de processamento. Após testes feitos a este algoritmo, verificou-se que existiam regras que limitavam em demasia a quantidade de pixéis classificados como fogo, assumindo características da chama muito restritas. Um dos testes que mais aumenta a quantidade de falsos negativos foi assunção de que os valores de vermelho, verde e azul da chama se situam, respetivamente, dentro da condição $R > 190 \ \wedge \ \ G > 100 \ \ \wedge \ \ B < 140$. Esta regra assume condições de captura da imagem muito específicas, e produz uma taxa muito elevada de falsos negativos. Por outro lado, com a condição $R > G > B$ consegue-se boa segmentação da imagem. No entanto, esta condição deixa ainda muitos segmentos, de cores perto dos cinza e branco, nos falsos positivos.

O algoritmo proposto é constituído pelas seguintes regras:

1. $\frac{|R-B|+|R-G|+|G-B|}{3} > 10$ – Esta regra tem por objetivo diminuir a quantidade de tons de cinza classificados como fogo. Os tons de cinza são obtidos quando a variação média entre a intensidade dos canais é mais próxima de zero.
2. $R > G \ \wedge \ R > B$ – Esta regra tem por objetivo encontrar as zonas vermelhas da imagem. A característica mais importante do fogo é a sua cor avermelhada, que se obtém quando o canal vermelho de um pixel é superior aos canais verde e azul.

3. $Tamanho\ do\ Segmento > 70$ – Esta regra tem por objetivo eliminar pequenos segmentos que não são relevantes para a correspondência estereoscópica.

A partir destas regras foi construído o algoritmo. Na deteção a imagem começa por ser dividida nas suas três gamas, no espaço de cor RGB. Depois, é aplicada a primeira regra, criando-se uma máscara com o resultado. Essa máscara é então erodida em 2 iterações e posteriormente dilatada em 2 iterações. Com este processo eliminam-se pontos isolados, reduzindo a quantidade de segmentos na imagem. Depois, a máscara é mais uma vez dilatada e posteriormente erodida (note-se a ordem inversa) , em 4 iterações. O objetivo deste processo é preencher buracos nos segmentos, e diminuir a irregularidade da fronteira do segmento, aumentando a área classificada ao incluir as irregularidades da fronteira. Este processo também tem por objetivo obter formas mais regulares e mais semelhantes à forma da chama obtida na câmara NIR, e tentar melhorar a extração de características. Posteriormente, é aplicada a máscara à imagem original, e aplicada a segunda regra. A máscara obtida após essa condição, é então dilatada e erodida em 7 e 2 iterações, respetivamente, com o objetivo de unir segmentos contíguos que pertencem à mesma chama. Na máscara obtida, serão obtidos todos os contornos dos segmentos presentes. Finalmente aplica-se a terceira regra aos segmentos encontrados, e aplicada a máscara obtida à imagem original. Na figura 3.24, encontra-se esquematizado o funcionamento do algoritmo.

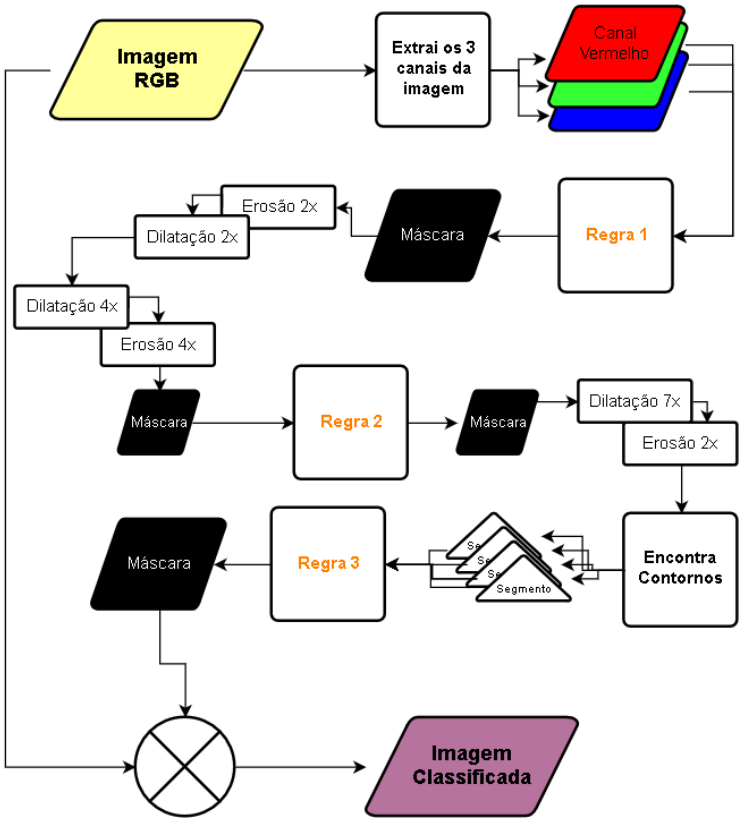

Figura 3.24 – Esquema de funcionamento do algoritmo de Deteção RGB

3.2.4.2 Deteção no espectro infravermelho próximo

A deteção no espectro infravermelho próximo baseia-se na radiação infravermelha emitida pela combustão da madeira. No capítulo 2.2.4, verificou-se que, numa imagem, a intensidade de cor é consideravelmente mais alta no fogo que no fundo da mesma. Mais, a variação da intensidade dos pixéis ao longo do tempo é mais alta na presença de fogo que no fundo da imagem. Deste modo, o método usado requer que se faça uma análise do comportamento da intensidade da chama ao longo do tempo. No entanto, o sistema utilizado não tem capacidade de fazer capturas a uma frequência que permita verificar a variação de intensidade de cor na chama.

Por outro lado, o facto de se ter de fazer um processamento da intensidade de cor da imagem prévio à deteção torna o algoritmo demasiado complexo para as necessidades do sistema que se está a implementar. Como esta deteção vai ser posteriormente comparada com a deteção no espectro visível, é apenas necessário que se garanta que a taxa de falsos negativos seja baixa.

Durante o estudo das imagens capturadas pela câmara NIR, verificou-se que a imagem tem tons rosa. Estes tons da imagem são resultado da inferior sensibilidade do canal verde do sensor, face aos canais vermelho e azul, como se pode observar na figura 3.5. Verificou-se ainda que a utilização dos 3 canais do sensor não é vantajosa, pois os canais são muito semelhantes entre si. Como tal, foi escolhido apenas um canal para deteção da imagem. Para a escolha do canal a usar foi tido em conta o contraste existente entre a chama e o fundo da imagem. Como é possível verificar pelas figuras 3.25, 3.26 e 3.27, ambos os canais vermelho e azul apresentam um contraste inferior face ao canal verde. Nesse canal, apesar de a generalidade da imagem ter uma luminosidade inferior, na chama a intensidade é sempre superior a 200.

Figura 3.25 – Canal vermelho da imagem NIR

Figura 3.26 – Canal verde da imagem NIR

Figura 3.27 – Canal azul da imagem NIR

Assim, foi escolhido apenas o canal verde da imagem NIR para o desenvolvimento do algoritmo. A deteção, baseada no canal verde, consiste em 3 regras, que estabelecem condições em que se pode considerar um pixel como chama:

1. $G > 200$ – esta regra tem como objetivo encontrar todos os focos de luz existentes na imagem. Desta condição resulta uma imagem em que tanto os pixel de fogo como zonas de alta luminosidade, como o céu, são classificados como incêndio.
2. $Tamanho\ do\ Segmento > 70$ – Esta regra tem como objetivo eliminar pequenos pontos dispersos de dimensões reduzidas, por não terem significado face à dimensão total da imagem.
3. $\frac{\sum G}{Tamanho\ do\ Segmento} > 200$ – Esta regra tem como objetivo garantir a qualidade de deteção em segmentos após dilatação, verificando se a média de cor do segmento é superior a 200. Em certos casos, ao fazer dilatação em multiplas iterações obtem-se um segmento com maior area de pixéis de fundo que de fogo. Ao usar esta condição, garante-se que o segmento gerado continua a ser maioritariamente fogo.

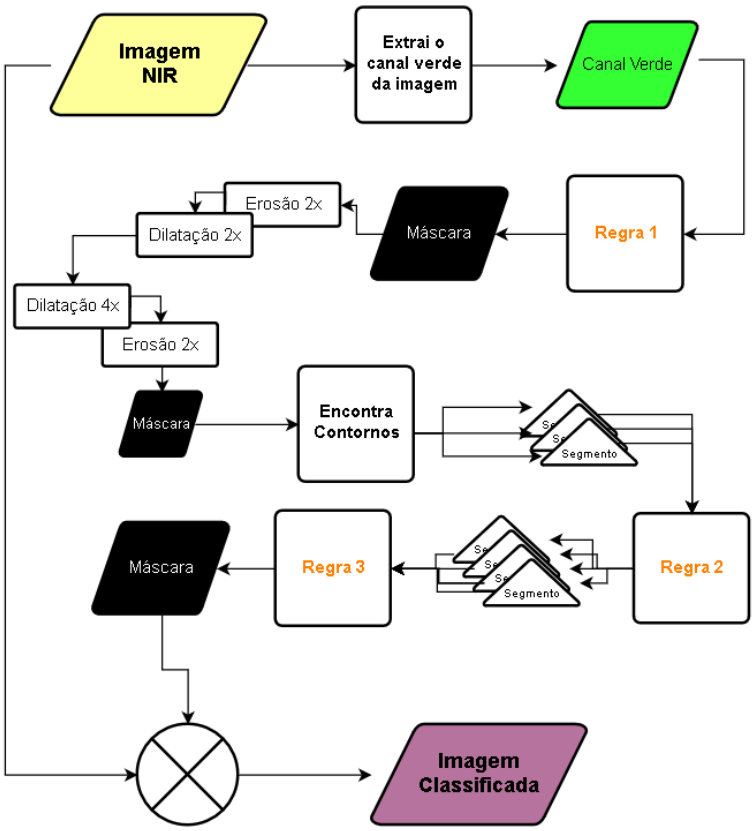

Figura 3.28 - Esquema de Funcionamento do algoritmo de deteção NIR

A partir destas regras foi construído o algoritmo. Na figura 3.28, é possível observar o funcionamento do algoritmo implementado.

Na deteção, começa-se por extrair o canal verde da imagem NIR. A partir da condição da primeira regra a imagem é transformada numa máscara de pixéis detetados. Essa máscara é então erodida em 2 iterações e posteriormente dilatada em 2 iterações. Com este processo eliminam-se pontos isolados, reduzindo a quantidade de segmentos na imagem. Depois, a máscara é mais uma vez dilatada e posteriormente erodida (note-se a ordem inversa), em 4 e 2 iterações. O objetivo deste processo é preencher buracos nos segmentos, e agrupar segmentos contíguos que pertençam à mesma chama, enquanto diminui a irregularidade da fronteira do segmento. Na máscara obtida, serão obtidos todos os contornos dos segmentos presentes. Apenas os segmentos que respeitem as regras 2 e 3 serão incluídos na máscara final e retornados pelo algoritmo.

3.2.4.3 Correspondência estereoscópica de segmentos

Apesar da qualidade de funcionamento dos algoritmos criados para deteção de fogo, continuam a existir taxas de falsos positivos consideráveis. Especificamente na deteção no espectro NIR o algoritmo não consegue distinguir entre o céu junto ao sol e a fogueira. No contexto deste sistema, não existe necessidade dessa distinção nas imagens NIR, visto que essa distinção é claramente aplicada pelo algoritmo para as imagens RGB. Como tal, foi criado um algoritmo capaz de juntar as classificações feitas em ambos os algoritmos, de forma a eliminar erros que ainda existam. Em adição, também é de interesse perceber a correspondência entre os segmentos das duas imagens.

Este algoritmo tem então como objetivo aperfeiçoar a deteção de fogo, e compilar numa estrutura todos os pares de segmentos correspondentes, se existirem. Um dos requisitos mais fulcrais deste algoritmo é uma boa calibração. O algoritmo estuda o tamanho e posição dos segmentos para fazer correspondência entre as duas imagens. Como tal, tem de se garantir que as imagens se encontram retificadas antes de proceder à execução deste algoritmo.

Assim, após retificação das imagens e deteção nos respetivos espectros, são encontrados os contornos de todos os segmentos existentes nas imagens classificadas. Para cada segmento da imagem RGB, é feita uma comparação da sua área com a dos segmentos NIR e escolhidos os 3 melhores (no máximo).

Posteriormente, do conjunto de melhores segmentos NIR, é escolhido o segmento cujo centro se encontra a menor distância vertical do centro do segmento RGB. Se o rácio entre as áreas, $\frac{areaRGB}{areaNIR}$, estiver entre 0.5 e 2, e a distância vertical entre os centros for menor que 80, então ambos os segmentos RGB e NIR serão adicionados à lista de pares estereoscópicos. Note-se que, após um segmento NIR ser correspondido com um segmento RGB, esse segmento não poderá ser correspondido com outro RGB. Como resultado deste algoritmo, é retornada uma lista de pares estereoscópicos de *contornos* que irá ser posteriormente utilizada na extração da profundidade. Assim, cada vez que um segmento NIR é adicionado à lista de pares estereoscópicos este é eliminado da lista de segmentos NIR existentes. Na figura 3.29 seguinte pode-se observar o funcionamento do algoritmo implementado.

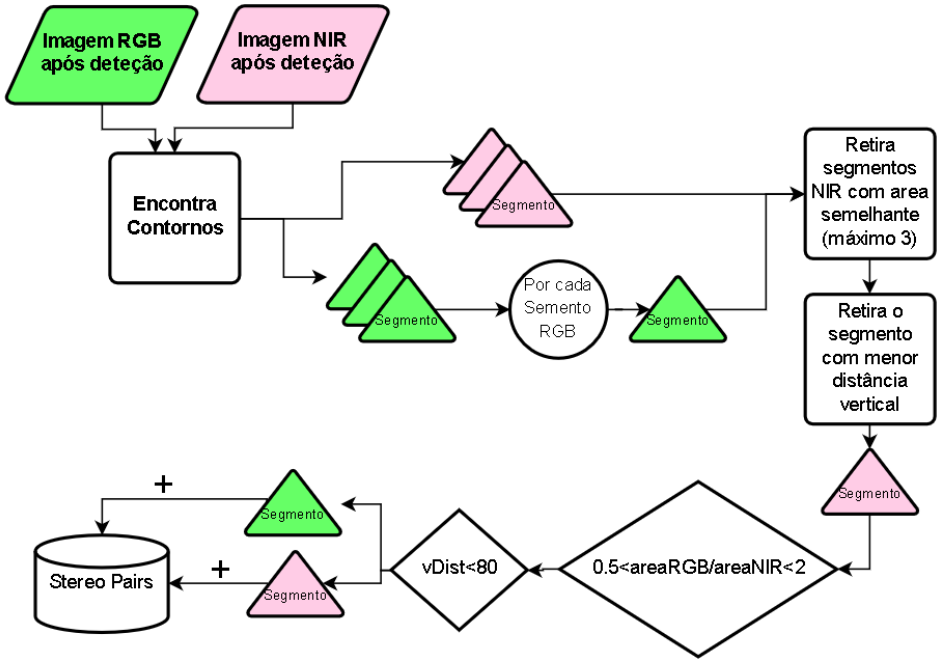

Figura 3.29 – Esquema de funcionamento do algoritmo de correspondência de segmentos

3.2.5 Extração de características

Quando se tem como objetivo extrair características de uma imagem estereoscópica, mais que fazer uma correspondência de segmentos é necessário fazer correspondências entre os pixéis de uma imagem e outra.

No algoritmo apresentado no capítulo 3.2.4.3, já é feita uma correspondência de uma zona de pixéis de uma imagem à zona correspondente da outra. No entanto, para que haja uma verdadeira correspondência estereoscópica, é necessário fazer a correspondência entre pixéis e não entre segmentos das imagens.

Os métodos estudados na bibliografia, ainda que interessantes nas suas condições normais de funcionamento, não são capazes de fazer a correspondência estereoscópica entre imagens captadas em diferentes espectros luminosos.

Na tentativa de testar os algoritmos existentes nas bibliotecas *OpenCV*, foi utilizado o algoritmo apresentado em (Samartzidis, 2017). Neste algoritmo, o autor faz uma correspondência estereoscópica tentando fazer corresponder na imagem direita, pixéis característicos encontrados na imagem esquerda. A partir dessas correspondências, é criado um mapa de disparidade.

Ao tentar aplicar este algoritmo não foi possível a obtenção de bons resultados porque as imagens captadas encontram-se em espectros diferentes. Seria como se cada olho visualizasse o mundo em gamas de cor completamente diferentes – não seria uma tarefa trivial corresponder objetos. Assim, houve necessidade de encontrar outros pontos de referência que nos permitam calcular a disparidade entre as duas imagens.

Deste modo, os pontos de referência usados foram:

- Centroide do segmento
- Centroide pesado com a intensidade de cor do segmento
- Posição e dimensão do menor retângulo (sem rotação) que contém o segmento
- Centroide da base da chama. Considerou-se a base da chama como contendo 1/3 da altura.
- Posição e dimensão do menor retângulo (sem rotação) que contém a base da chama, calculada a 1/3 da altura.

Com base nestes pontos referência, é calculada a disparidade entre as imagens. É de notar que estes pontos referência são muito dependentes da forma da chama, que varia da imagem RGB para a NIR. Assim, para evitar disparidades irrealistas, se a diferença entre a posição vertical dos pontos referência for superior a 15 pixéis, não é aceite essa disparidade. Com base nas 5 disparidades calculadas, a partir dos pontos referência descritos, é calculada a disparidade média, ***Disp***.

Foi também programado o cálculo necessário para cálculo da distância de captura. Com base no método trigonométrico descrito no capítulo 2.1.4., foi não só calculada a distância como o tamanho da frente de fogo.

Assim, a para o cálculo da distância à fogueira foi simplificada a equação (23):

$$D = \frac{x_0 B}{2\tan\frac{\varphi_0}{2} \times Disp} = \frac{1080 pixeis \times 0{,}133\, metros}{2\tan\frac{60°}{2} \times Disp}$$

Daí:

$$\begin{cases} factor = \frac{1080 \times 0{,}133}{2\ \tan\frac{60}{2}} \\ D(Disp) = \frac{factor}{Disp} \end{cases} \qquad (48)$$

O valor do ângulo de visão da câmara foi assumido a 60º, ao invés de 62,2%. Foi feita esta aproximação para tentar compensar a perda de campo de visão após retificação.

Para o cálculo da largura da frente de fogo (l_{fogo}), assumiu-se uma relação de proporcionalidade direta entre a largura total observada na imagem (em pixéis e metros, tendo em conta a distância de captura) e a largura parcial do fogo. A partir da relação trigonométrica entre a distância à fogueira (***D***), a largura em metros observada na imagem (l_0) e o ângulo de visão horizontal da câmara (φ_0) foi possível obter a expressão:

$$l_0 = 2D\tan\frac{\varphi_0}{2} \qquad (49)$$

Da qual, aplicando a relação de proporcionalidade direta, vem que:

$$l_{fogo} = \frac{x_{fogo} \times 2D\tan\frac{\varphi_0}{2}}{x_0}$$

Substituindo *D*, tem-se a largura da frente de fogo em função da disparidade:

$$\boldsymbol{l_{fogo}(Disp) = \frac{x_{fogo} \times B}{Disp}} \qquad (50)$$

Em que:

- B é a distância entre câmaras
- x_{fogo} é a largura, em pixéis, da fogueira

Em adição, foi ainda calculado o ângulo de inclinação da chama. Para isso tomou-se como referência o centroide de base da chama de coordenadas ($x_{centroide}$, $y_{centroide}$), e o ponto de extremidade do topo da chama, de coordenadas (x_{topo} , y_{topo}). Assim, tem-se que o ângulo de inclinação da chama em graus, θ_{fogo}, é:

$$\boldsymbol{\theta_{fogo} = \tan^{-1}\left(\frac{y_{topo} - y_{centroide}}{x_{topo} - x_{centroide}}\right) \times \frac{180}{\pi}} \qquad (51)$$

3.3 Interfaces desenvolvidos

Para complementar os algoritmos desenvolvidos, foram criadas duas interfaces.

A primeira serve como interface de captura da visão de captura, no qual existem dois botões para iniciar e parar a gravação de pares de imagens. Ao mesmo tempo vão sendo apresentadas as imagens capturadas.

A segunda junta os algoritmos de deteção, correspondência estereoscópica de segmentos e extração de características numa interface para análise das características de uma frente de fogo.

Na figura 3.30, observa-se o algoritmo, onde se encontram as deteções de fogo e correspondência de segmentos aplicadas à captura feita pelo sistema. No caso da figura, a fogueira tinha 2 metros de largura estava a ser captada a 10 metros.

Figura 3.30 - Interface de Deteção de Fogo

4

Resultados

No capítulo anterior, foi explanado o funcionamento dos vários algoritmos de deteção e extração de características de fogo. Apesar de ter sido possível verificar a qualidade dos algoritmos ao longo do seu desenvolvimento, a avaliação da qualidade do algoritmo é muito subjetiva. Como a avaliação é feita visualmente, e com base em dezenas de imagens, é impossível manter contagem na quantidade de boas e más classificações. Mais, dentro das classificações incorretas é importante verificar as circunstâncias em que tal acontece, percebendo assim a relevância dos erros obtidos face ao objetivo do sistema implementado. Assim, houve necessidade de criar uma base de dados para teste, e respetivo método de teste.

Deste modo, este capítulo foca-se na análise dos resultados obtidos, bem como avaliar a qualidade e limitações do sistema desenvolvido.

4.1 Limitações da visão estereoscópica

A visão estereoscópica, apesar das grandes vantagens que traz, tem algumas limitações que condicionam a sua construção e desenvolvimento. Dependendo do alcance pretendido, existem requisitos mínimos a ser respeitados na implementação da estereoscopia.

Após estudo da equação (23), apresentada no capítulo 2.1.4, observa-se que o alcance máximo é alcançado quando a disparidade entre imagens é de 11 pixéis. Para disparidades inferiores, verifica-se que o erro de distância causado pela diferença de 1 pixel é superior a 10%. Na figura 4.1, pode-se observar como o erro de 1 pixel pode afetar o cálculo da distância: à medida que a distância aumenta, também o erro na distância, devido à quantização, aumenta.

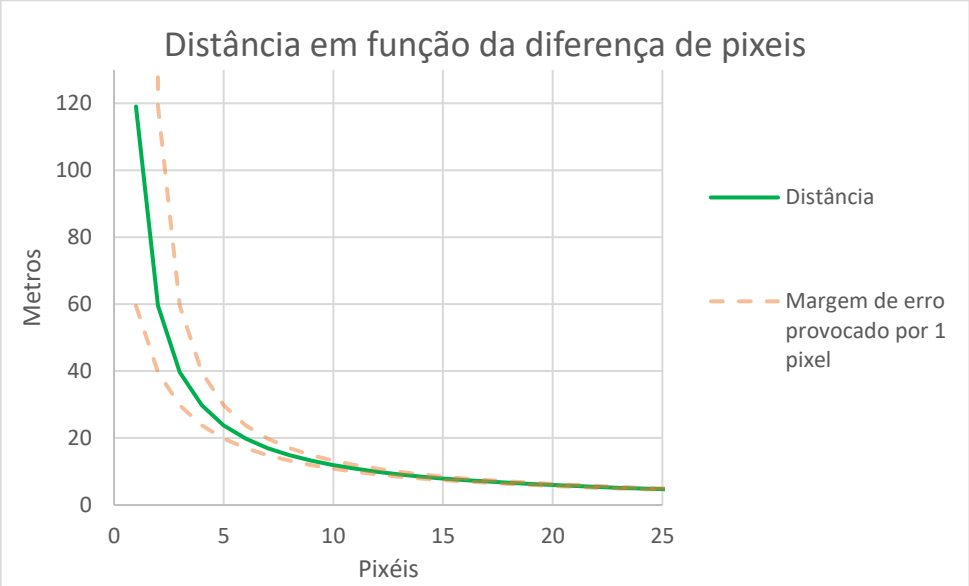

Figura 4.1 - Representação gráfica do erro no cálculo da distância, em função da diferença de pixéis de uma imagem para outra

Por outro lado, ao assumir um ângulo médio de visão horizontal de 60 graus, verifica-se que para se obter alcances superiores a 1,5 Km tem de existir o correto balanço entre a distância entre as câmaras e a resolução. Por um lado, escolhendo uma resolução elevada, aumenta-se o custo do sistema. Por outro, escolhendo uma distância elevada entre câmaras, limita-se a mobilidade do sistema.

Na figura 4.2, pode-se observar a variação do alcance em função da resolução e da distância entre câmaras. Propositadamente, foram destacados alcances a rondar 1,5Km. Deste modo, verifica-se que para se conseguir esse alcance, as câmaras têm de estar à distância mínima de 4,7 metros, e a resolução horizontal superior a 2000 pixéis.

Em adição, mesmo considerando os requisitos mínimos de 5 metros e 940 pixéis, verifica-se a inviabilidade do sistema.

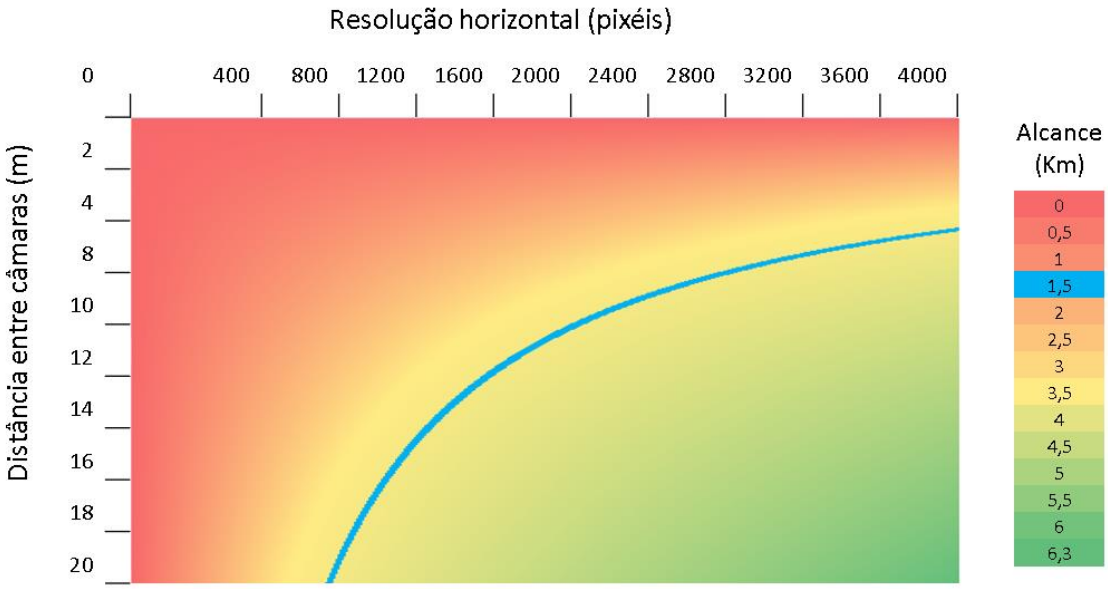

Figura 4.2 - Alcance máximo em função da resolução e distância entre câmaras

Por um lado, não existe mobilidade num equipamento com mais de 5 metros de largura. Por outro, mesmo considerando resoluções de 4000 pixéis, não há garantia de que se consiga ver os detalhes na imagem com a devida nitidez. Assim, a implementação de um sistema estereoscópico com grande alcance implica uma qualidade de captura a longa distância, e por isso, é importante o uso de lentes telescópicas. Apesar de reduzirem o alcance mínimo, estas lentes são capazes de captar com nitidez características de objetos a mais de 1Km.

4.2 Base de dados para teste

A base de dados criada usou os dados recolhidos e apresentados no capítulo 3.2.3. Do conjunto de dados foram escolhidas 194 imagens RGB e NIR, captadas às distâncias de 5, 7,5, 10, 15, 20 e 25 metros.

Através de um programa de edição de imagem, foram criadas, manualmente, máscaras de classificação. Essas máscaras são imagens binárias, em que o fogo aparece a branco e o resto da imagem a preto. Nas figuras 4.3 e 4.4 pode-se ver uma imagem capturada pela câmara RGB e respetiva máscara de deteção.

Figura 4.3 – Imagem RGB

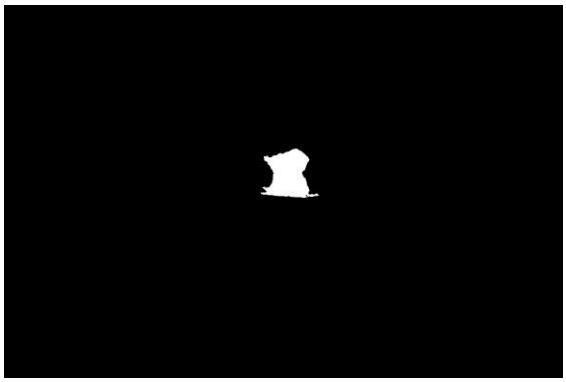

Figura 4.4 - Máscara de Deteção relativa à figura 4.3

Ainda que o método para criação destas máscaras tenha alguma subjetividade, a avaliação gerada com base nestes dados é muito mais objetiva, e permite retirar conclusões uteis. Esta base de dados foi construída para avaliação da qualidade de deteção.

Para o caso da extração de características, separaram-se as imagens em várias pastas, cada uma contendo as capturas a cada uma das distâncias. Através do nome da pasta é possível retirar a informação da distância à qual foi capturada a imagem, e a distância horizontal entre os segmentos das imagens RGB e NIR. Por exemplo, para o caso das capturas feitas a 5 metros, a distância horizontal espectável seria de 25 pixéis, como tal o nome da pasta seria “p25_m5”.

4.3 Resultado dos algoritmos

4.3.1 Calibração e Retificação

A calibração é um dos passos mais importantes neste sistema de deteção estereoscópico. Através do algoritmo de calibração, é possível retirar as matrizes de distorção e de foco de cada câmara. Mais, para o contexto de visão estereoscópica este algoritmo retorna ainda as matrizes de Fundamental, Essencial, de Rotação e de Translação da estereoscopia.

Nas figuras de 4.5 a 4.8, pode-se observar o resultado da retificação aplicada a duas imagens, após a calibração completa da imagem:

Figura 4.5 – Imagem NIR não retificada

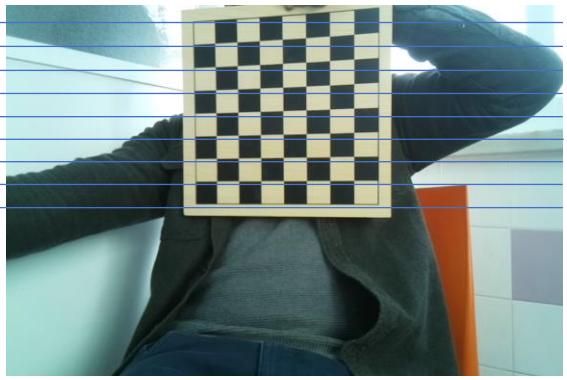

Figura 4.6 – Imagem RGB não retificada

Figura 4.7 – Imagem NIR retificada

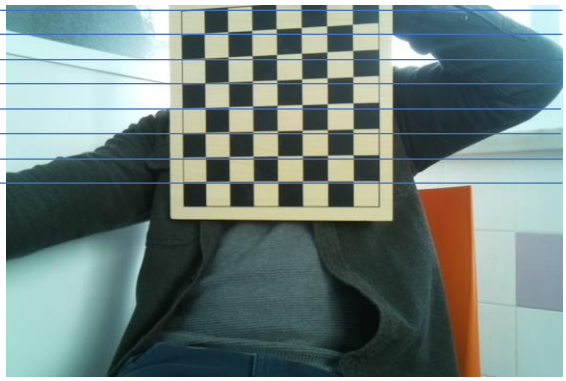

Figura 4.8 – Imagem RGB retificada

Repare-se que, nas figuras 4.5 a 4.8, foram sobrepostas linhas horizontais para ajudar a compreender o efeito do algoritmo de retificação em ambas imagens. Ao observar o primeiro par de imagens (4.5 e 4.6) é possível verificar que existem discrepâncias verticais entre a posição das células do tabuleiro nas duas imagens, facto que tem de ser corrigido antes de aplicação da profundidade à visão estereoscópica.

Por outro lado, também se pode verificar uma certa curvatura nas retas que limitam o tabuleiro, resultado da distorção radial e tangencial das lentes das câmaras.

Em contraste, nas figuras 4.7 e 4.8 não só a distorção é menor, como o desvio vertical entre câmaras é quase indetetável. Como se pode ver, os cantos esquerdos das células da primeira coluna na figura 4.7 estão corretamente alinhados com os respetivos cantos na figura 4.8.

4.3.2 Deteção na câmara RGB

Após a execução do algoritmo de deteção de fogo no espectro visível, verifica-se a transformação da imagem original, para uma imagem apenas com as manchas de fogo detetadas, com todo o fundo a preto e o fogo com a cor original. Nas figuras 4:9 a 4:12, pode-se observar o resultado da segmentação usando este algoritmo, com capturas a 5 e 25 metros.

Figura 4.9 – Captura a 25 metros com a câmara RGB

Figura 4.10 – Deteção RGB feita à figura 4.9

Figura 4.11 – Captura a 5 metros com a câmera RGB

Figura 4.12 – Deteção RGB feita à figura 4.11

As figuras anteriores (4.9 a 4.12), permitem verificar e identificar erros e suas causas. Ao analisar as figuras 4.10 e 4.12, verifica-se que são incluídas zonas contiguas à chama, consequência do algoritmo criado. Ao adicionar estas zonas à classificação, aumenta-se a zona onde é possível fazer correspondência estereoscópica. Por outro lado, estas também ajudam a tornar os segmentos RGB mais semelhantes aos da deteção NIR. Repare-se ainda que apesar de continuarem a existir segmentos contíguos pertencentes à mesma chama, a quantidade desses segmentos diminuiu significativamente, quer por se terem anexado ao segmento da chama principal, quer por terem sido eliminados por tamanho insuficiente.

Apesar de a base de dados de teste não ter elementos vermelhos no cenário, caso existissem, estes seriam classificados como fogo. Ainda que seja um erro relevante, não tem importância no contexto geral do sistema desenvolvido. Um objeto vermelho ainda que seja detetado pela câmara RGB, não irá ser detetado pela câmara NIR. Deste modo, ainda que existam falsos positivos, estes são descartados pela correspondência estereoscópica.

Para ter uma avaliação mais objetiva, foi criado um algoritmo que permite fazer a contagem de verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN). Esse algoritmo é também usado para teste da deteção em NIR e estereoscópica. Para isso são feitas várias subtrações entre a máscara detetada e a máscara de fogo real, e contados os pontos pertencentes a cada tipo de classificação. Neste teste foram testadas 97 imagens RGB de 1080x720 pixéis. O resultado desta análise encontra-se na seguinte matriz confusão (tabela 4.1):

Tabela 4.1 – Matriz de confusão do algoritmo de deteção RGB

		Detetado	
		Positivo	**Negativo**
Real	**Positivo**	747776 (99,14%) VP	262050 (0.35%) FN
	Negativo	49077 (0,07%) FP	338464 (0,44%) VN

Pela análise da matriz de confusão, verifica-se uma taxa alta de falsos negativos. Estes casos são obtidos quando, a chama está sobreposta com fundos de cores vivas que criam ruído na cor da mesma, dificultando a correta deteção.

Foi possível calcular a percentagem de erro do classificador:

$$\%erro = \frac{FP + FN}{VP + VN + FP + FN} \times 100 = 0{,}4125\%$$

Verifica-se assim uma boa qualidade de deteção. Mais, este classificador foi capaz de classificar, corretamente, 97 de 97 imagens do banco de dados de teste, como contendo fogo.

4.3.3 Deteção na câmara NIR

Após a execução do algoritmo de deteção no espectro infravermelho próximo, verifica-se a transformação da imagem original, para uma imagem apenas com as manchas de fogo detetadas, com todo o fundo a preto e o fogo com a intensidade de cor original. Nas figuras 4.13 a 4.16, pode-se observar o resultado da segmentação usando este algoritmo, com capturas a 5 e 25 metros.

Figura 4.13 – Captura a 5 metros com a câmara NIR

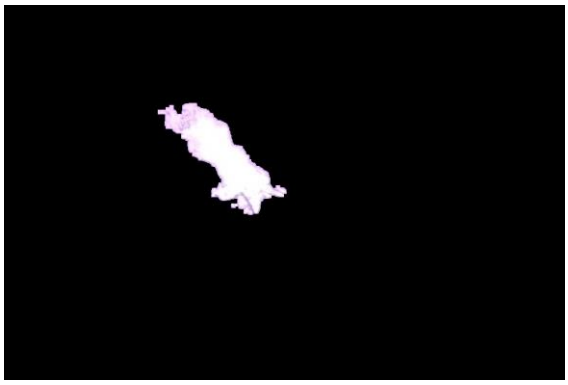

Figura 4.14 - Deteção NIR feita à figura 4.13

Figura 4.15 – Captura a 25 metros com a câmara NIR

Figura 4.16 – Deteção NIR feita à figura 4.15

As figuras 4.13 a 4.16, permitem verificar a qualidade do classificador e a identificar erros e suas causas. Ao analisar as figuras 4.13 e 4.14, verifica-se que o algoritmo falha em detetar extremidades da chama, isto porque, nas extremidades a chama não emite tanta radiação quanto no seu núcleo. Ainda assim pode-se verificar que a zona de maior intensidade da fogueira é corretamente detetada. Olhando para as figuras 4.15 e 4.16, nota-se ainda que o algoritmo tende a classificar partes do céu como fogo. Em termos de radiação o algoritmo não tem capacidade para diferenciar a chama do céu, isto porque ambas são zonas com alta emissão de radiação infravermelha.

Como foi referido na secção anterior, com auxílio da base de dados para teste criada, foram avaliados os pixéis em que se encontram verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN). Neste teste foram testadas 97 imagens NIR de 1080x720 pixéis. O resultado desta análise encontra-se na seguinte matriz de confusão (tabela 4.2):

Tabela 4.2 - Matriz de confusão do algoritmo de deteção NIR

		Detetado	
		Positivo	Negativo
Real	Positivo	71053665 (94,20%) VP	219917 (0,29%) FN
	Negativo	3558449 (4,72%) FP	595169 (0,79%) VN

Pela análise da matriz confusão, e pelo comportamento de deteção apresentado nas imagens 69 a 72, verifica-se uma taxa alta de falsas deteções. A maioria, falsos positivos, são provocados pela intensidade de radiação infravermelha vinda do céu, como foi explicado anteriormente.

Foi possível calcular a percentagem de erro do classificador:

$$\%erro = \frac{FP + FN}{VP + VN + FP + FN} \times 100 = 5{,}0093\%$$

Este classificador apresenta uma taxa de erro de 5,0093%. Verifica-se assim uma grande diminuição da qualidade de deteção, em comparação com o respetivo algoritmo da câmara RGB. Ainda assim, como a maioria dos erros existentes vêm de falsos positivos, pode-se afirmar que este é um bom algoritmo de deteção, pois esses erros são facilmente corrigidos pela correspondência estereoscópica. Mais, este classificador foi capaz de classificar, corretamente, 97 de 97 imagens do banco de dados de teste, como contendo fogo.

4.3.4 Correspondência estereoscópica de segmentos

Este algoritmo faz uma intersecção entre a classificação RGB e NIR. Após a execução do algoritmo de correspondência estereoscópica, serão apenas apresentados os segmentos que existem em ambas as imagens. Assim, verifica-se a diminuição de segmentos detetados face às deteções anteriormente feitas. Nas figuras 4.17 e 4.18, pode-se verificar as diferenças existentes após correspondência estereoscópica, face às deteções primárias que se observou nas figuras 4.10 e 4.16.

Figura 4.17 – Correspondência estereoscópica aplicada à figura 4.10

Figura 4.18 – Correspondência estereoscópica aplicada à figura 4:16

Ao observar as figuras 4.17 e 4.18, verifica-se que a correspondência estereoscópica produz bons resultados, quando aplicada em conjunto com as deteções nos respetivos espectros. Na figura 4.18, em comparação com a figura 4.16, todos os segmentos que se encontravam no céu foram excluídos da classificação, mantendo apenas o segmento respetivo ao fogo detetado na figura 4.10.

Assim, após inclusão deste algoritmo obteve-se melhorias bastante significativas na segmentação das manchas de fogo.

A aplicação deste algoritmo, permitiu reduzir a percentagem de erro da deteção RGB de 0,4125% para 0,4030%. Também em NIR, foi possível reduzir a percentagem de erro dos de 5,0093% para os 0,4473%.

Assim, ao juntar os algoritmos de deteção com o de correspondência estereoscópica, foi possível obter a seguinte matriz confusão (tabela 4.3), onde se encontram separados os valores para as imagens RGB e NIR, através das cores laranja e verde respetivamente:

Tabela 4.3 - Matriz confusão da deteção estereoscópica, após deteção em cada espectro RGB e NIR

		Detetado			
		Positivo		Negativo	
		RGB	NIR	RGB	NIR
Real	Positivo	74681141 (99,01%) VP	74368661 (98,60%)	223598 (0,30%)	22107 (0,03%) FN
	Negativo	80394 (0,11%) FP	348686 (0,46%)	442067 (0,58%)	687746 (0,91%) VN

Como se pode observar, comparando as tabelas 4.1, 4.2 e 4.3, o algoritmo de correspondência estereoscópica de segmentos ajuda a melhorar a precisão das deteções feitas dentro de cada espectro. Aplicando a mesma fórmula de erro a todo o algoritmo obteve-se para a conjunção dos 3 algoritmos, a percentagem de erro de 0,4473%. Em adição, o algoritmo foi capaz detetar o fogo presente em 97 de 97 imagens de uma fogueira.

4.3.5 Extração de Características

O processo de extração de características baseia-se em encontrar pontos referência correspondentes nas duas imagens. O facto de as duas câmaras captarem as imagens em gamas espectrais distintas diminui a capacidade de fazer a correspondência estereoscópica de pixéis. Enquanto na imagem RGB é possível verificar-se muitas nuances na cor da chama, na imagem NIR a chama encontra-se geralmente sobre-exposta. Assim, é praticamente impossível encontrar pontos semelhantes entre as duas imagens. Também, apesar de as máscaras de deteção apresentarem formas muito semelhantes, a área de chama apresentada pela deteção NIR é superior à da RGB, consequência da diferente sensibilidade que as câmaras têm à radiação infravermelha.

Deste modo, a extração de características é muito imprecisa. Nos pontos de referência baseados na base da chama, não foi possível obter disparidades admissíveis, isto porque a diferença entre a localização vertical dos pontos foi sempre superior a 15 pixéis.

Já nos pontos de referência baseados no menor retângulo sem rotação, e nos centroides (normal e baseado na intensidade de cor), foi possível obter disparidades que, na maioria das vezes, a diferença vertical não ultrapassou os 15 pixéis. Para estes pontos de referência, foi testado o erro da disparidade e da distância, e anotada a quantidade de vezes que a distância vertical superou os 15 pixéis. Nas tabelas 4.4, 4.5 e 4.6, encontra-se o resultado desse teste, para os 3 cálculos de pontos referência, dividido por distâncias.

Tabela 4.4 – Resultados dos cálculos através do menor retângulo sem rotação

Distância (m)	Total Imagens	Imagens Analisadas	Erro absoluto Disparidade (pixéis)	Erro absoluto Distância (m)	Desvio Padrão (pixéis)
5	16	0	-	-	-
7,5	16	10	13,6	38,3	4,0
10	17	17	3,2	5,2	3,2
15	17	17	5,4	19,2	3,0
20	15	14	3,9	7,4	3,4
25	15	15	2,9	31,4	2,0

Tabela 4.5 – Resultados dos cálculos através do centroide pesado com a cor

Distância (m)	Total Imagens	Imagens Analisadas	Erro absoluto Disparidade (pixéis)	Erro absoluto Distância (m)	Desvio Padrão (pixéis)
5	16	5	72,7	3,2	39,4
7,5	16	16	11,0	114,2	3,4
10	17	17	3,9	5,4	4,4
15	17	17	3,6	6,3	2,1
20	15	17	7,2	17,5	16,1
25	15	15	3,4	18,8	2,1

Tabela 4.6 - Resultados dos cálculos através do centroide

Distância (m)	Total Imagens	Imagens Analisadas	Erro absoluto Disparidade (pixéis)	Erro absoluto Distância (m)	Desvio Padrão (pixéis)
5	16	0	-	-	-
7,5	16	5	14,5	8,3	3,8
10	17	17	4,0	6,0	4,4
15	17	17	3,6	6,8	2,2
20	15	14	3,1	19,8	2,6
25	15	15	3,5	16,6	2,1

Observando o erro absoluto da disparidade, nas tabelas 4.4, 4.5 e 4.6, verifica-se que quanto maior a distância a que se está do fogo, menor é o erro da disparidade. À medida que se aumenta a distância à chama, menor ela aparece na imagem, e menor é o espaço para erro, justificando-se assim a relação entre o erro de disparidade e a distância. Pela análise das tabelas 4.4 a 4.6, conclui-se que apesar da média de erro de disparidade ser baixa, existem muitas flutuações no cálculo da disparidade. Isto leva a que haja um erro maior no cálculo da distância. Também no cálculo da largura da frente de fogo, os valores dependem da disparidade, tornando-o muito impreciso e incorreto. Como se pode concluir, os métodos utilizados não obtiveram resultados bons com margem de erro baixas, pois o erro no cálculo da distância é da ordem de grandeza da distância real. No entanto, a partir do desvio padrão, verifica-se que o erro existente não se afasta em demasia da média. Assim, se o sistema estereoscópico tivesse uma distância entre câmaras maior seria possível uma deteção mais precisa e exata.

No caso da extração do ângulo de inclinação, foram obtidos bons resultados em situações em que a inclinação da chama é bem notória, como é o caso da figura 4.13. Nos casos em que a chama tem um topo irregular, não foi possível calcular os ângulos corretos.

5

Conclusão e Trabalho Futuro

Os sistemas digitais de processamento de imagem são uma grande ajuda na automação e melhoramento de sistemas de alerta atualmente em vigor. Nos casos estudados, os sistemas estereoscópicos, quando aplicados a situações de incêndios, requerem condições de montagem e funcionamento especiais. Especificamente, para garantir a qualidade dos dados recolhidos é importante uma correta calibração do sistema estereoscópico. O sistema não só tem de ter as câmaras horizontalmente alinhadas, como também deve ter guardadas as matrizes que informam o software quanto à posição relativa de cada câmara, entre câmaras e no espaço. Por outro lado, estes sistemas têm muitas limitações no alcance.

Como observado no capítulo 4.1, o uso de sistemas estereoscópicos em situações de longo alcance requer condições mínimas para o correto funcionamento. Quer seja devido a erros de quantização, falta de nitidez da imagem ou limitações de alcance, é necessário fazer um correto dimensionamento das características do sistema. Assim, considerando o uso de lentes sem ampliação, para se obter alcances de 1,5km tem de se usar câmaras de alta resolução, com grande sensibilidade, separadas de pelo menos 4,7 metros.

A partir da recolha de dados e desenvolvimento dos algoritmos de deteção, verificou-se que as condições de captura de imagem condicionam os algoritmos de deteção. Mais, a própria dimensão do fogo pode influenciar o resultado ao afetar a quantidade de radiação que entra no sensor. Deste modo, os algoritmos desenvolvidos

apresentaram bons resultados para as condições de captura testadas. Nessas situações, os algoritmos de deteção e correspondência estereoscópica de segmentos de fogo são bastante precisos.

Na deteção de fogo em RGB, por ser uma deteção muito baseada na cor, existem erros inerentes ao tipo de algoritmo, árvore de decisão. Em imagens com objetos de tom vermelho, o sistema aumenta a taxa de falsos positivos. Em contrapartida, imagens com fogo são classificadas com taxas de falsos negativos de 0,35%. Como resultado, este algoritmo obteve uma taxa de erro de 0,41%.

Na deteção de fogo em NIR, por ser uma deteção muito baseada na luminosidade da chama, quando existem zonas luminosas na imagem, estas serão classificadas como fogo. Em contraste, quando se está na presença de fogo, o algoritmo gera 0,29% de falsos negativos. Como resultado, este algoritmo obteve uma taxa de erro de 5,01%.

No caso da correspondência estereoscópica de segmentos de fogo, como os algoritmos de deteção foram desenvolvidos de modo a se obter baixas taxas de falsos negativos, é feita uma boa filtragem dos segmentos detetados previamente. Ao combinar as características do fogo nos espectros visível e infravermelho próximo, é possível corrigir casos em que a percentagem de falsas deteções é muito alta. Eliminando segmentos sem correspondência, foi possível reduzir a percentagem de erro das deteções nas imagens RGB e NIR, para 0,40% e 0,45% respetivamente. Deste modo, limita-se a deteção das imagens às zonas vermelhas de alta intensidade de radiação IR. Como resultado, este algoritmo obteve uma taxa de erro de 0,45%.

Na extração de características, como efeito das particularidades do sistema estereoscópico, não foi possível obter a mesma precisão. Devido à característica do erro em função da distância de captura, os algoritmos não têm precisão suficiente para uma correta avaliação da frente de fogo. Também, devido ao uso de diferentes gamas espectrais nas câmaras, a semelhança existente entre imagens RGB e NIR é limitada. Como consequência, os algoritmos usados na correspondência estereoscópica de pixéis não são capazes detetar corretamente pontos correspondentes em ambas as imagens. Ainda assim, no algoritmo de cálculo da disparidade entre os segmentos de fogo, foi possível obter erros absolutos baixos face à resolução da imagem, permitindo fazer uma estimativa da distância de captura e dimensões do fogo. Essa estimativa, dependendo da distância ao fogo, está sujeita a erros da ordem de grandeza do valor real. Como resultado, este

algoritmo é capaz de fazer a deteção do ângulo de inclinação da chama corretamente, no entanto no cálculo de distâncias é muito impreciso.

Em suma, o uso de diferentes gamas espectrais, produz deteções de fogo muito boas, que minimizam a quantidade de erros. No entanto, esse uso traz dificuldades na correspondência estereoscópica de imagens. Ainda que o algoritmo de correspondência de segmentos limite a imagem a zonas de alta correspondência de pixéis, as diferenças espectrais existentes não permitem uma correta correspondência.

Concluindo, a visão estereoscópica multiespectral é uma ferramenta de grande utilidade para extração de características da dimensão e forma de uma frente de fogo. Assim, esta ferramenta, quando aplicada à classificação de fogo, tem potencial para melhorar o combate aos incêndios, e aumentar o conhecimento sobre os mesmos.

5.1 Trabalho Futuro

O trabalho desenvolvido permitiu melhorias na deteção de fogo, contudo existem ainda aspetos que podem ser melhorados.

Para se garantir um sistema de deteção fiável é necessário uma maior quantidade e variedade de imagens para teste. Aumentar a base de dados permite uma análise mais aprofundada das características do fogo. Deste modo, possibilita-se também o uso de técnicas de *machine-learning*, que necessitam de extensos bancos de dados. Face à expansão da base de dados, poderá ser necessário ajustar os algoritmos de deteção e correspondência de segmentos de fogo.

Para melhorar a precisão no cálculo das características do fogo aconselha-se o uso de lentes telescópicas. O uso deste tipo de lentes, ainda que aumente a distância mínima de captura, permite aumentar o alcance do sistema, mantendo a qualidade da imagem a longa distância e diminuindo a distância a que as câmaras devem estar.

Para potenciar a correspondência estereoscópica de pixéis será necessário aumentar a frequência de captura. Como tal, sugere-se o uso de dois microcomputadores *Raspberry Pi* para eliminar o atraso que o uso do multiplexador introduz. Assim, aumenta-se a capacidade de processamento do sistema e possibilita-se a criação de uma estereoscopia completa, enquanto se diminui o custo do sistema.

Também para melhoria da correspondência estereoscópica, será necessário aumentar a semelhança existente entre imagens de diferentes espectros. Como tal, deverá ser desenvolvido um algoritmo que faça um ajuste dinâmico das características de captura da câmera, com o objetivo de aumentar a nitidez nos pormenores do fogo em cada imagem. Deste modo, prevê-se que seja possível a correta correspondência estereoscópica de pixéis.

Finalmente, através de um sistema devidamente equipado com as características e elementos descritos acima, deverá ser feito um estudo das características dos incêndios abordadas no capítulo 2.3. Alargando a informação existente, poderá ser alargado o estudo existente sobre o desenvolvimento de um fogo florestal, através do processamento digital de imagem. Mais, através de sistemas de geolocalização aliados à estereoscopia, e em conjunto com dispositivos de comunicação de longo alcance, pretende-se uma maior qualidade de informação sobre os avanços das frentes de fogo, e que esta circule com maior facilidade, Assim, pretende-se melhorar a capacidade de antecipar os movimentos das frentes de fogo, e consequentemente a capacidade e a eficiência com que os bombeiros dão resposta aos avanços das mesmas.

6

Referências

Achard, C., Bigorgne, E. e Devars, J. (2000) «A sub-pixel and multispectral corner detector», Proceedings-International Conference on Pattern Recognition, 15(3), pp. 959–962. doi: 10.1109/ICPR.2000.903704.

Alexander, M. E. (1982) «Calculating and interpreting forest fire intensities», Canadian Journal of Botany, 60(4), pp. 349–357. doi: 10.1139/b82-048.

Alexander, M. E. e Cruz, M. G. (2011) «Crown fire dynamics in conifer forests», Synthesis of knowledge of extreme fire behavior, I(C). Disponível em: http://cafiresci.org/s/jfsp_extreme-fire-behavior_9-2011.pdf#page=119.

Bouguet, J.-Y. B., 2015. Camera Calibration Toolbox for Matlab.[Online] Disponível em: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html [Acedido em Novembro 2018].

Burnett, J. D. e Wing, M. G. (2018) «A low-cost near-infrared digital camera for fire detection and monitoring», International Journal of Remote Sensing. Taylor & Francis, 39(3), pp. 741–753. doi: 10.1080/01431161.2017.1385109.

Celik, T. (2010) «Fast and efficient method for fire detection using image processing», ETRI Journal, 32(6), pp. 881–890. doi: 10.4218/etrij.10.0109.0695.

Chen, R., Luo, Y. e Alsharif, M. R. (2013) «Forest fire detection algorithm based on digital image», Journal of Software, 8(8), pp. 1897–1905. doi: 10.4304/jsw.8.8.1897-1905.

Derpanis, K. G. (2004) «The harris corner detector», York University, (March), pp. 2–3.

EFFIS, 2018. Área ardida em Portugal. [Online]
Disponível em: http://effis.jrc.ec.europa.eu/
[Acedido em Agosto 2018].

Firmenich, D., Brown, M. e Süsstrunk, S. (2011) «MULTISPECTRAL INTEREST POINTS FOR RGB-NIR IMAGE REGISTRATION D . Firmenich , M . Brown , S . S ¨ usstrunk School of Computer and Communication Sciences , Ecole Polytechnique F ´ erale de Lausanne email : damien.firmenich@epfl.ch», Ieee International Conference On Image Processing, pp. 185–188.

Harris, C. e Stephens, M. (1988) «A Combined Corner and Edge Detector», Procedings of the Alvey Vision Conference 1988, p. 23.1-23.6. doi: 10.5244/C.2.23.

Hartley, R. I. (1998) «Theory and Practice of Projective», International Journal of Computer Vision, 35(2), pp. 115–127. doi: 10.1023/A:1008115206617.

Ivmech Mekatronik & İnovasyon, 2016. IVPort V2 Raspberry Pi Camera Module V2 Multiplexer. [Online]
Disponível em: http://www.ivmech.com/magaza/en/development-modules-c-4/ivport-v2-raspberry-pi-camera-module-v2-multiplexer-p-107
[Acedido em Setembro 2018].

Ivmech Mekatronik & İnovasyon, 2016. Raspberry Pi Camera Module V2 Multiplexer. [Online]
Disponível em: https://github.com/ivmech/ivport-v2/blob/master/ivport.py
[Acedido em Outubro 2018].

Jones, D., 2016. *Picamera.* [Online]
Disponível em: https://picamera.readthedocs.io/en/release-1.13/
[Acedido em Fevereiro 2019].

Leñero-Bardallo, J. A. et al. (2013) «Fire detection with a frame-less vision sensor working in the NIR band», VII Internationl Conference on Forest Fire Research, 4(4), pp. 23–30.

Moravec, H. P. (1980) «Obstacle avoidance and navigation in the real world by a seeing robot rover.», tech. report CMU-RI-TR-80-03, p. 175. Disponível em: https://www.ri.cmu.edu/publication_view.html?pub_id=22.

Mrovlje, J. e Vran, D. (2008) «Distance measuring based on stereoscopic pictures», 9th International PhD Workshop on Systems and Control: Young Generation Viewpoint, 2(October), pp. 1–6.

OpenCV, 2019. Camera Calibration and 3D Reconstruction. [Online]
Disponível em: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html
[Acedido em Outubro 2018].

Raspberry Pi Foundation, 2012. Raspberry Pi 3 Model B+. [Online]
Disponível em: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
[Acedido em Janeiro 2019].

Raspberry Pi Foundation, 2015. Raspberry Pi Touch Display. [Online]
Disponível em: https://www.raspberrypi.org/products/raspberry-pi-touch-display/
[Acedido em Janeiro 2019].

Richards, J. A. e Jia, X. (1999) Remote sensing digital image analysis. 4th ed. Canberra, Australia: Springer. doi: 10.1007/978-3-642-88087-2.

Schmid, C. et al. (2010) «Evaluation of Interest Point Detector». doi: 10.1023/A:1008199403446.

Samartzidis, T., 2017. Open CV Stereo. [Online]
Disponível em: http://timosam.com/python_opencv_depthimage
[Acedido em Outubro 2018].

Scott, J. H. (2006) «Off the Richter: Magnitude and intensity scales for wildland fire», Proceedings of the 3rd international fire ecology and management congress.

Suspiro, A., 2017. Incêndios. [Online]
Disponível em: https://observador.pt/2017/10/18/2017-tera-sido-o-pior-ano-em-area-ardida-sistema-europeu-aponta-para-mais-de-500-mil-hectares/
[Acedido em Agosto 2018].

Zou, L. e Li, Y. (2010) «A method of stereo vision matching based on OpenCV», ICALIP 2010 - 2010 International Conference on Audio, Language and Image Processing, Proceedings, pp. 185–190. doi: 10.1109/ICALIP.2010.5684978.

7

Anexos

Alterações realizadas ao Código da biblioteca do multiplexador IVPort:

```
1. #!/usr/bin/env python
2. #
3. # This file is part of Ivport.
4. # Copyright (C) 2016 Ivmech Mechatronics Ltd. <bilgi@ivmech.com>
5. #
6. # Ivport is free software: you can redistribute it and/or modify
7. # it under the terms of the GNU General Public License as published
   by
8. # the Free Software Foundation, either version 3 of the License, or
9. # (at your option) any later version.
10.     #
11.     # Ivport is distributed in the hope that it will be useful,
12.     # but WITHOUT ANY WARRANTY; without even the implied warranty
   of
13.     # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
14.     # GNU General Public License for more details.
15.     #
16.     # You should have received a copy of the GNU General Public Li-
   cense
17.     # along with this program.  If not, see <http://www.gnu.org/li-
   censes/>.
18.
19.     #title           :ivportNEW.py edited from ivport.py
20.     #description     :ivportNEW.py is module for capturing ivport
   camera multiplexer
21.     #author          :Caner Durmusoglu (Edited by Diogo Boto)
22.     #date            :20160514         (Edited in 2019)
23.     #version         :0.1
24.     #usage           :import ivport
25.     #notes           :
26.     #python_version  :2.7
27.     #==============================================================
   ===============
```

```
28.     import numpy as np
29.     import sys
30.     import cv2
31.     import time
32.     from picamera.array import PiRGBArray
33.
34.     try:
35.         import IIC
36.         import RPi.GPIO as gp
37.         gp.setwarnings(False)
38.         gp.setmode(gp.BOARD)
39.     except:
40.         print ("There are no IIC.py and RPi.GPIO module.")
41.         print ("install RPi.GPIO: sudo apt-get install python-
    rpi.gpio")
42.         sys.exit(0)
43.
44.     try:
45.         import picamera
46.     except:
47.         print ("There are no picamera module or directory.")
48.         sys.exit(0)
49.
50.     TYPE_QUAD = 0
51.     TYPE_QUAD2 = 1
52.     TYPE_DUAL = 2
53.     TYPE_DUAL2 = 3
54.
55.     class IVPort():
56.         IVJP = {'A': (11, 12), 'C': (21, 22), 'B': (15, 16), 'D':
    (23, 24)}
57.
58.         def __init__(self, iv_type=TYPE_DUAL2, iv_jumper=1):
59.
60.             self.fPin = self.f1Pin = self.f2Pin = self.ePin = 0
61.             self.ivport_type = iv_type
62.             self.is_camera_v2 = self.ivport_type in (TYPE_DUAL2,
    TYPE_QUAD2)
63.             self.is_dual = self.ivport_type in (TYPE_DUAL2,
    TYPE_DUAL)
64.             self.ivport_jumper = iv_jumper
65.             if not self.is_dual: self.ivport_jumper = 'A'
66.             self.camera = 1
67.             self.is_opened = False
68.             self.configured1=False
69.             self.configured2=False
70.
71.             if self.is_camera_v2:
72.                 self.iviic = IIC.IIC(addr=(0x70), bus_enable
    =(0x01))
73.
74.             self.link_gpio()
75.
76.         def link_gpio(self):
77.             if self.is_dual:
78.                 self.fPin = self.DIVJP[self.ivport_jumper]
79.                 gp.setup(self.fPin, gp.OUT)
80.             else:
```

```
81.                 self.f1Pin, self.f2Pin =
   self.IVJP[self.ivport_jumper]
82.                 self.ePin = 7
83.                 gp.setup(self.f1Pin, gp.OUT)
84.                 gp.setup(self.f2Pin, gp.OUT)
85.                 gp.setup(self.ePin, gp.OUT)
86.
87.         # ivport camera change
88.         def camera_change(self, camera=1):
89.             if self.is_dual:
90.                 if camera == 1:
91.                     if self.is_camera_v2: self.iviic.write_con-
   trol_register((0x01))
92.                     gp.output(self.fPin, False)
93.                 elif camera == 2:
94.                     if self.is_camera_v2: self.iviic.write_con-
   trol_register((0x02))
95.                     gp.output(self.fPin, True)
96.                 else:
97.                     print ("Ivport type is DUAL.")
98.                     print ("There isnt camera: %d" % camera)
99.                     self.close()
100.                    sys.exit(0)
101.            else:
102.                if camera == 1:
103.                    if self.is_camera_v2: self.iviic.write_con-
   trol_register((0x01))
104.                    gp.output(self.ePin, False)
105.                    gp.output(self.f1Pin, False)
106.                    gp.output(self.f2Pin, True)
107.                    if self.configured1 is True:
108.                        self.picam.iso=self.iso_C1
109.                        self.picam.awb_gains=self.awb_gains_C1
110.                        self.picam.shutter_speed=self.shut-
   ter_speed_C1
111.                        self.picam.meter_mode=self.meter_mode_C1
112.
113.                elif camera == 2:
114.                    if self.is_camera_v2: self.iviic.write_con-
   trol_register((0x02))
115.                    gp.output(self.ePin, True)
116.                    gp.output(self.f1Pin, False)
117.                    gp.output(self.f2Pin, True)
118.                    if self.configured2 is True:
119.                        self.picam.iso=self.iso_C2
120.                        self.picam.awb_gains=self.awb_gains_C2
121.                        self.picam.shutter_speed=self.shut-
   ter_speed_C2
122.                        self.picam.meter_mode=self.meter_mode_C2
123.
124.                elif camera == 3:
125.                    if self.is_camera_v2: self.iviic.write_con-
   trol_register((0x04))
126.                    gp.output(self.ePin, False)
127.                    gp.output(self.f1Pin, True)
128.                    gp.output(self.f2Pin, False)
129.                elif camera == 4:
```

```
130.                     if self.is_camera_v2: self.iviic.write_con-
   trol_register((0x08))
131.                     gp.output(self.ePin, True)
132.                     gp.output(self.f1Pin, True)
133.                     gp.output(self.f2Pin, False)
134.                 else:
135.                     print ("Ivport type is QUAD.")
136.                     print ("Cluster feature hasnt been implemented
   yet.")
137.                     print ("There isnt camera: %d" % camera)
138.                     self.close()
139.                     sys.exit(0)
140.             self.camera = camera
141.
142.         # picamera initialize
143.         # Camera V2
144.         # capture_sequence and start_recording require "cam-
   era_v2=True"
145.         # standart capture function doesnt require "camera_v2=True"
146.         def camera_open(self, camera_v2=True, resolution=None,
   framerate=None, grayscale=False):
147.
148.             if self.is_opened: return
149.             self.picam = picamera.PiCamera(camera_v2=camera_v2,
   resolution=resolution, framerate=framerate)
150.             self.rawCapturePair=[]
151.             self.rawCapturePair.append(PiRGBAr-
   ray(self.picam,size=resolution))
152.             self.rawCapturePair.append(PiRGBAr-
   ray(self.picam,size=resolution))
153.
154.             self.iso_C1=0
155.             self.awb_gains_C1=0,0
156.             self.exposure_mode_C1='auto'
157.             self.shutter_speed_C1=1000000/framerate
158.             self.awb_mode_C1='auto'
159.             self.meter_mode_C1='average'
160.
161.             self.iso_C2=0
162.             self.awb_gains_C2=0,0
163.             self.exposure_mode_C2='auto'
164.             self.shutter_speed_C2=1000000/framerate
165.             self.awb_mode_C2='auto'
166.             self.meter_mode_C2='average'
167.             self.configured1=False
168.             self.configured2=False
169.
170.             if grayscale: self.picam.color_effects = (128, 128)
171.             self.is_opened = True
172.
173.         def camera_setup(self, cameraNumber=1, iso=None,shut-
   ter_speed=None,awb_gains=None,meter_mode=None):
174.             if cameraNumber is 1:
175.                 self.configured1=True
176.                 if iso is None:
177.                     self.exposure_mode_C1='auto'
178.                 else:
179.                     self.iso_C1=iso
```

```
180.                     self.exposure_mode_C1='off'
181.                 if shutter_speed is None:
182.                     self.shutter_speed_C1=1000000/self.picam.fram-
    erate
183.                 else:
184.                     self.shutter_speed_C1=shutter_speed
185.                 if awb_gains is None:
186.                     self.awb_mode_C1='auto'
187.                 else:
188.                     self.awb_mode_C1='off'
189.                     self.awb_gains_C1=awb_gains
190.                 if meter_mode is None:
191.                     self.meter_mode_C1='average'
192.                 else:
193.                     self.meter_mode_C1=meter_mode
194.             else:
195.                 if cameraNumber is 2:
196.                     self.configured2=True
197.                     if iso is None:
198.                         self.exposure_mode_C2='auto'
199.                     else:
200.                         self.iso_C2=iso
201.                     if shutter_speed is None:
202.                         self.shut-
    ter_speed_C2=1000000/self.picam.framerate
203.                     else:
204.                         self.shutter_speed_C2=shutter_speed
205.                     if awb_gains is None:
206.                         self.awb_mode_C2='auto'
207.                     else:
208.                         self.awb_gains_C2=awb_gains
209.                     if meter_mode is None:
210.                         self.meter_mode_C2='average'
211.                     else:
212.                         self.meter_mode_C2=meter_mode
213.
214.         # picamera capture
215.         def camera_capture(self, filename, **options):
216.             if self.is_opened:
217.                 self.picam.capture(filename + "_CAM" +
    str(self.camera) + '.jpg', **options)
218.             else:
219.                 print ("Camera is not opened.")
220.
221.         def tempCapture(self,**options):
222.             if self.is_opened:
223.                 print("Camera "+str(self.camera)+":
    iso="+str(self.picam.iso))
224.                 self.picam.capture(self.rawCapturePair[self.camera-
    1],format='bgr',**options)
225.                 img=self.rawCapturePair[self.camera-1].array
226.                 self.rawCapturePair[self.camera-1].truncate(0)
227.
228.
229.                 return img
230.             else:
231.                 print("Camera is not opened.")
232.                 return
```

```
233.
234.         def close(self):
235.             self.camera_change(1)
236.             if self.is_opened: self.picam.close()
```

Código de para captura de imagens usado em conjunto com o código anterior:

```
1. from picamera.array import PiRGBArray
2. from picamera import PiCamera
3. import picamera
4. import time
5. import cv2
6. import cv2 as cv
7. import os
8. import ivportNEW as ivport
9.
10.     try:
11.             iv = ivport.IVPort(ivport.TYPE_QUAD2, iv_jumper='A')
12.
13.             iv.camera_open(camera_v2=True, resolution=(1080, 720),
    framerate=10)
14.             AWB_GAINS=1.8,0.8
15.             ISO_RGB=500
16.             ISO_NIR=360
17.             METER_MODE='backlit'
18.             SHUTTER_NIR=50000
19.             SHUTTER_RGB=50000
20.             iv.camera_setup(cameraNumber=1,iso=ISO_RGB,me-
    ter_mode=METER_MODE,shutter_speed=SHUTTER_RGB,awb_gains=AWB_GAINS)
21.             print("abriu 1")
22.             iv.camera_setup(cameraNumber=2,iso=ISO_NIR,me-
    ter_mode=METER_MODE,shutter_speed=SHUTTER_NIR,awb_gains=AWB_GAINS)
23.             print("abriu 2")
24.             osError=cv2.imread("OSerror.png")
25.     except(OSError):
26.             print("OSError while opening camera")
27.
28.     def stereoPair():
29.             try:
30.
31.                     iv.camera_change(1)
32.                     time.sleep(0.1)
33.                     imageR=iv.tempCapture(use_video_port=True)
34.                     iv.camera_change(2)
35.                     time.sleep(0.1)
36.                     imageL=iv.tempCapture(use_video_port=True)
37.
38.                     return True,imageL,imageR
39.             except (OSError):
40.                     return False,osError,osError
41.
42.     def leftCam(name):
43.         iv.camera_change(2)
44.         time.sleep(0.1)
```

```
45.         iv.camera_capture(name, use_video_port=True)
46.         fileName2=name+"_CAM2.jpg"
47.         imageL=cv2.imread(fileName2)
48.         return imageL
49.
50.     def rightCam(name):
51.         iv.camera_change(1)
52.         time.sleep(0.1)
53.         iv.camera_capture(name, use_video_port=True)
54.         fileName1=name+"_CAM1.jpg"
55.         imageR=cv2.imread(fileName1)
56.         return imageR
```