UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

LUCAS ROTAVA

Algoritmos de tempo real para melhoramento de imagens capturadas
no espectro do infravermelho projetados para síntese em FPGA

São Paulo
2015

LUCAS ROTAVA

Algoritmos de tempo real para melhoramento de imagens capturadas
no espectro do infravermelho projetados para síntese em FPGA

Dissertação apresentada à Escola de
Engenharia de São Carlos, Universidade
de São Paulo, como parte dos requisitos
para obtenção do título de Mestre em
Ciências, Programa de Engenharia
Elétrica.

Área de Concentração: Processamento
de Sinais e Instrumentação

Orientador: Prof. Dr. Evandro Luís
Linhari Rodrigues

São Paulo
2015

Trata-se da versão corrigida da dissertação. A versão original se encontra disponível na EESC/USP que aloja o Programa de Pós-Graduação de Engenharia Elétrica.

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

R842a Rotava, Lucas
 Algoritmos de tempo real para melhoramento de
 imagens capturadas no espectro do infravermelho
 projetados para síntese em FPGA / Lucas Rotava;
 orientador Evandro Luís Linhari Rodrigues. São Carlos,
 2015.

 Dissertação (Mestrado) - Programa de Pós-Graduação
 em Engenharia Elétrica e Área de Concentração em
 Processamento de Sinais e Instrumentação -- Escola de
 Engenharia de São Carlos da Universidade de São Paulo,
 2015.

 1. Processamento de imagens. 2. Câmera térmica. 3.
 Imagem térmica. 4. FPGA. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **LUCAS ROTAVA**.

Título da dissertação: "Algoritmos de tempo real para melhoramento de imagens capturadas no espectro do infravermelho projetados para síntese em FPGA".

Data da defesa: 04/12/2015

Comissão Julgadora:

Resultado:

Prof. Associado **Evandro Luis Linhari Rodrigues**
(Orientador)
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Dr. **Marcelo Andrade da Costa Vieira**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Dr. **Celso Aparecido de França**
(Universidade Federal de São Carlos/UFSCar)

APROVADO

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica:
Prof. Associado **Luis Fernando Costa Alberto**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Paulo César Lima Segantine**

Dedico esse trabalho aos meus pais,
que aguardaram ansiosamente por mais essa etapa vencida

AGRADECIMENTOS

Ao Prof. Dr. Evandro Luís Linhari Rodrigues pela paciência, incentivo e ajuda no decorrer desses anos de trabalho.

Aos professores Marcelo Andrade da Costa Vieira e Celso França pelas dicas e sugestões para o melhor desenvolvimento do trabalho.

À minha companheira Junia pela força e pelas palavras de coragem nos momentos difíceis.

À Opto Eletrônica, e todo o pessoal de lá, pela oportunidade de trabalhar e me divertir com esse projeto.

Aos meus irmãos Jairo e Elói pelas revisões e críticas construtivas durante a fase de escrita desta dissertação.

E por último, agradeço aos meus pais. Sem o apoio deles durante a vida não seria possível nem ter começado este trabalho.

RESUMO

Rotava, L. **Algoritmos de tempo real para melhoramento de imagens capturadas no espectro do infravermelho projetados para síntese em FPGA**. 2015. Dissertação (Mestrado) – Departamento de Engenharia Elétrica e de Computação, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2015.

Este trabalho apresenta o desenvolvimento de algoritmos de processamento de imagens para câmeras térmicas, com o objetivo de sintetizá-los em FPGA. Existem diversas aplicações para imagens térmicas nas áreas médica, de segurança e industrial, por isso o conhecimento e o desenvolvimento de câmeras térmicas são de interesse para a academia e para a indústria. Por consequência, o desenvolvimento de algoritmos que tratem as imagens também representa importante papel. Os algoritmos implementados neste trabalho são: correção de não uniformidade (NUC); substituição de *pixels* defeituosos, ou *bad pixels*, (BPR); redução da resolução de cor com realce de contraste; e filtro espacial para realçar detalhes da imagem, chamado de filtro de nitidez. Os três primeiros são algoritmos importantes devido à características dos detectores e de câmeras térmicas, já o filtro de nitidez foi proposto para melhorar a visualização de objetos nas imagens. Com os algoritmos simulados em Matlab foram feitas medidas de contraste e de MTF das imagens de saída, e os resultados obtidos para os algoritmos de realce de contraste e de nitidez mostraram que eles são adições importantes ao conjunto de algoritmos básicos para câmeras térmicas, já que, para alguns casos, o realce de contraste aumentou em mais de 50% a medida de contraste da imagem, em comparação com o algoritmo anterior, e o filtro de nitidez proporcionou valores de MTF até duas vezes maiores. Os algoritmos de NUC e BPR apresentaram os resultados esperados, corrigindo a imagem recebida do detector. As imagens utilizadas eram de 640×512 *pixels* processadas em uma taxa de 30 fps, e dessa forma optou-se pelo FPGA para a síntese dos algoritmos, sendo possível realizar os processamentos paralelamente contando com a característica de alto *throughput* inerente a estes componentes. Os algoritmos implementados em FPGA apresentaram desempenho superior aos requisitos mínimos de tempo para o sistema utilizado, sendo perfeitamente capazes de processar o vídeo de entrada em tempo real.

Palavras-chave: Processamento de imagens, Câmera térmica, Imagem térmica, FPGA

ABSTRACT

Rotava, L. **Real-time infrared images enhancement algorithms developed for FPGA synthesis**. 2015. Dissertation (Master's Degree) – Department of Electrical and Computer Engineering, São Carlos School of Engineering, University of São Paulo, São Carlos, 2015.

This work presents the development of FPGA-synthesizable image processing algorithms to thermal cameras. There are plenty of applications for thermal imaging in medical, security and industrial areas, therefore, the knowledge and the development of thermal cameras are of great interest to both academia and industry. Consequently, the development of algorithms to enhance the images is also important. The implemented algorithms are: non-uniformity correction (NUC); bad pixel replacement (BPR); pixel depth reduction with contrast enhancement; and emboss spatial filter. The three first algorithms are important because of some characteristics of infrared detectors and cameras, and the emboss filter is proposed to improve the visualization of objects in the images. With the algorithms simulated in Matlab, the contrast and MTF were measured in the output images, and the results showed that the contrast enhancement and the emboss filter algorithms are important additions to the infrared camera's basic set of image processing algorithms since, for some cases, the contrast enhancement was able to improve the contrast by 50% and the emboss filter have doubled the MTF. NUC and BPR algorithms had the expected results, correcting the image from the detector. There were used images with resolution of 640×512 at 30 frames per second and, because of this, it was chosen to synthesize the algorithms in an FPGA, this way it is possible to run them in parallel, counting on the high throughput characteristic of the FPGAs. The implemented algorithms have better timing performance than the needed for the system used, being perfectly able to process the input video in real time.

Keywords: Image processing, Thermal camera, Infrared imaging, FPGA

LISTA DE FIGURAS

Figura 1 – Espectro eletromagnético, adaptado de [14]	29
Figura 2 – Janela atmosférica de absorção de radiação, adaptado de [16]	31
Figura 3 – Luz incandescente emitindo luz visível e imagem de luz incandescente no espectro infravermelho	31
Figura 4 – Comparação entre curvas de contraste por temperatura da cena para comprimentos de onda MWIR e LWIR, de [19]	34
Figura 5 – Detector de infravermelho do tipo microbolômetro [20]	35
Figura 6 – Detector de infravermelho refrigerado [21]	36
Figura 7 – Transmissividades de uma lente de Ge: (a) sem filme antirreflexo; (b) com filme antirreflexo. Adaptado de [23]	37
Figura 8 – Exemplo de responsividade de um sensor de infravermelho	38
Figura 9 – Curvas de responsividade de um sensor: (a) Sem correção; (b) Com correção de um ponto; (c) Com correção de dois pontos. Adaptado de [29]	40
Figura 10 – Equalização de histograma: (a) Imagem original; (b) Histograma original; (c) Imagem equalizada; (d) Histograma equalizado. Adaptado de [32]	43
Figura 11 – Alargamento de contraste: (a) Curvas de alargamento de contraste; (b) Imagem de entrada; (c) Imagem de saída. Adaptado de [33]	43
Figura 12 – Exemplo de redução da resolução de cor de uma imagem	44
Figura 13 – Sequência de cálculo do MTF para um sistema ideal: (a) Imagem de entrada; (b) Função de espalhamento de borda, ESF; (c) Função de espalhamento de linha, LSF; (d) Função de transferência de modulação, MTF	46
Figura 14 – Sequência de cálculo do MTF para um sistema não ideal: (a) Imagem de entrada; (b) Função de espalhamento de borda, ESF; (c) Função de espalhamento de linha, LSF; (d) Função de transferência de modulação, MTF	46
Figura 15 – Exemplo simplificado de um FPGA, adaptado de [35]	48
Figura 16 – Bloco de DSP do FPGA Cyclone IV, adaptado de [36]	48
Figura 17 – Diagrama de blocos de uma expressão booleana	50
Figura 18 – Três formas de escrever uma mesma expressão booleana em linguagem de descrição de hardware VHDL	50

Figura 19 – Divisão em subsistemas do equipamento utilizado e o fluxo de dados dentro do equipamento.....	51
Figura 20 – Relação entre sinais de sincronismo e sinais enviados pela placa de controle para o FPGA	52
Figura 21 – Diagrama de blocos da placa de processamento de vídeo.....	53
Figura 22 – Fluxo de desenvolvimento dos algoritmos	54
Figura 23 – Sequência de utilização das memórias para três frames consecutivos	57
Figura 24 – Organização da memória RAM de correção	58
Figura 25 – Algoritmo de NUC	58
Figura 26 – Pipeline do algoritmo de NUC	58
Figura 27 – Correção de um ponto – pipeline para o frame inicial	60
Figura 28 – Correção de um ponto - pipeline para os frames intermediários	60
Figura 29 – Correção de um ponto - pipeline para o frame final.....	60
Figura 30 – Conteúdo da memória de bad pixels	61
Figura 31 – Fluxograma da rotina de BPR	62
Figura 32 – Sequência para a formação e análise de histogramas	63
Figura 33 – Exemplo de histograma e os níveis de 5%, 50% e 95% do total de pixels	64
Figura 34 – Comparação entre as transformações dos algoritmos de redução de: (a) dois pontos; (b) três pontos	65
Figura 35 – Formação da matriz de vizinhança para o filtro de nitidez: (a) Relação entre matrizes de vizinhança de pixels adjacentes; (b) Disposição dos pixels da matriz de vizinhança.....	68
Figura 36 – Pipeline para formação da matriz de vizinhança para o primeiro pixel da imagem	68
Figura 37 – Pipeline para formação da matriz de vizinhança para os pixels a partir do segundo pixel da linha.....	69
Figura 38 – Pipeline para formação da matriz de vizinhança para o primeiro pixel de cada linha, exceto a primeira.....	69
Figura 39 – Imagens e curvas de responsividade simuladas: (a) Antes da correção; (b) Depois da correção de não uniformidade de dois pontos.....	71
Figura 40 – Imagens e curvas de responsividade simuladas: (a) após o temperature drift; (b) após a correção de não uniformidade de um ponto	72
Figura 41 – Comparação entre imagens: (a) sem NUC; (b) com NUC	72

Figura 42 – Comparação entre imagens: (a) sem BPR, com bad pixels em destaque; (b) com BPR	73
Figura 43 – Contraste por número de pontos do algoritmo de realce de contraste, para cada percentual de saturação de pixels avaliado.....	74
Figura 44 – Imagens e histogramas para o algoritmo de realce de contraste com dois, três e oito pontos para saturação de 5% dos pixels (a), (b) e (c), para saturação de 1% dos pixels (d), (e) e (f) e para saturação de 0,1% (g), (h) e (i), respectivamente	74
Figura 45 – Imagens e histogramas após realce de contraste com saturação de 5% dos pixels com dois, três e oito pontos. (a), (b) e (c) para cena 1, (d), (e) e (f) para cena 2 e (g), (h) e (i) para cena 3, respectivamente.....	75
Figura 46 – Cenas reais com algoritmo de dois pontos: (a), (b) e (c). E suas respectivas cenas com algoritmo de três pontos: (d), (e) e (f)	76
Figura 47 – Validação do filtro por MTF: (a) Imagem de entrada; (b) MTF com e sem a aplicação do filtro de nitidez sobre a imagem	77
Figura 48 – Avaliação subjetiva do filtro de nitidez com uma imagem sintetizada e diferentes níveis de filtragem: (a) $k = 1$; (b) $k = 6$; (c) $k = 11$; (d) $k = 15$	78
Figura 49 – Imagem e curva de MTF com corpo negro em 10 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 13$	78
Figura 50 – Avaliação do filtro por MTF com corpo negro em 10 °C: (a) Curva do valor do MTF por valor de k ; (b) Curvas de MTF para alguns valores de k	79
Figura 51 – Imagem e curva de MTF com corpo negro em 80 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 16$	79
Figura 52 – Avaliação do filtro por MTF com corpo negro em 10 °C: (a) Curva do valor do MTF por valor de k ; (b) Curvas de MTF para alguns valores de k	80
Figura 53 – Duas cenas reais processadas na câmera com diferentes níveis de filtragem: (a) e (d) $k = 1$; (b) e (e) $k = 7$; (c) e (f) $k = 15$	81
Figura 54 – Detalhe de uma placa de carro processada na câmera com duas variações do filtro de nitidez: (a) $k = 1$; (b) $k = 15$	81
Figura 55 – Imagem de um caminhão a longa distância processada na câmera com duas variações do filtro de nitidez: (a) $k = 1$; (b) $k = 10$	82
Figura 56 – Imagem e curva de MTF com corpo negro em 20 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 20$	93

Figura 57 – Imagem e curva de MTF com corpo negro em 30 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 10$	94
Figura 58 – Imagem e curva de MTF com corpo negro em 40 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 15$	94
Figura 59 – Imagem e curva de MTF com corpo negro em 50 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 11$	95
Figura 60 – Imagem e curva de MTF com corpo negro em 60 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 15$	95
Figura 61 – Imagem e curva de MTF com corpo negro em 70 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 15$	96
Figura 62 – Imagens processadas na câmera com filtro configurado com $k = 2$, (a) e (b), $k = 3$, (c) e (d), $k = 4$, (e) e (f), $k = 5$, (g) e (h), $k = 6$, (i) e (j), $k = 8$, (k) e (l), $k = 9$, (m) e (n), $k = 10$, (o) e (p), $k = 11$, (q) e (r), $k = 12$, (s) e (t), $k = 13$, (u) e (v) e $k = 14$, (w) e (x)	100

LISTA DE TABELAS

Tabela 1 – Relação entre ruídos em imagens térmicas e suas causas, adaptado de [25]	39
Tabela 2 – Comparação entre o contraste RMS local, equação 10, para o algoritmo de realce de contraste com dois, três e oito pontos para três diferentes cenas	75
Tabela 3 – Comparação entre o contraste RMS local, equação 10, para o algoritmo de realce de contraste com dois e três pontos para diferentes imagens processadas pela câmera.....	76
Tabela 4 – Relações entre o melhor MTF encontrado e o MTF sem o filtro de nitidez	80

LISTA DE ABREVIATURAS E SIGLAS

- BPR – *Bad pixel replacement* – Substituição de *pixels* defeituosos
- ESF – *Edge spread function* – Função de espalhamento de borda
- FIR – *Far infrared* – Infravermelho distante
- FPGA – *Field-programmable gate array*
- fps – *Frames per second* – *Frames* por segundo
- IR – *Infrared* – Infravermelho
- LSF – *Line spread function* – Função de espalhamento de linha
- LWIR – *Long- wavelength infrared* – Infravermelho de ondas longas
- MTF – *Modulation transfer function* – Função de transferência de modulação
- MWIR – *Medium- wavelength infrared* – Infravermelho de ondas médias
- NIR – *Near infrared* – Infravermelho próximo
- NUC – *Non uniformity correction* – Correção de não uniformidade
- QWIR – *Quantum-well infrared* – Detector do tipo *quantum-well*
- RMS – *Root mean square* – Valor eficaz
- SWIR – *Short-wavelength infrared* – Infravermelho de ondas curtas
- VHDL – *VHSIC hardware description language*
- VHSIC – *Very high speed integrated circuit*

SUMÁRIO

1	Introdução	23
1.1	Objetivos	24
1.2	Motivação	24
1.3	Contribuições	24
1.4	Organização	25
2	Fundamentação teórica	27
2.1	História	27
2.2	Aplicações das imagens térmicas	28
2.3	Radiação no infravermelho	29
2.4	Detectores de infravermelho	34
2.5	Óptica para câmeras térmicas	36
2.6	Considerações a respeito de um detector	37
2.7	Pré-processamento de imagens térmicas	39
2.8	Calibração e avaliação da imagem	45
2.9	FPGA	47
3	Materiais e Métodos	51
3.1	Materiais	51
3.2	Métodos	54
3.3	Desenvolvimento dos algoritmos	56
4	Resultados	71
4.1	Correção de não uniformidade, NUC	71
4.2	Substituição de <i>pixels</i> defeituosos, BPR	73
4.3	Contraste e redução da resolução de cor	73
4.4	Filtro de nitidez	77
5	Conclusões	83
6	Propostas para trabalhos futuros	85
	Referências	87
	Apêndices	93
	Apêndice A – Cálculo do MTF com e sem filtro de nitidez	93
	Apêndice B – Filtro de nitidez para imagens processadas na câmera	97

1 Introdução

Somente após as duas guerras mundiais, na década de 1950, que o sigilo envolvendo tecnologia termal diminuiu, tornando a tecnologia de domínio público. Até então essa era uma tecnologia quase exclusiva das forças armadas dos países envolvidos. A partir daí, como é possível ver hoje, novas formas de utilização da tecnologia foram desenvolvidas, tanto para fins militares quanto para fins civis. Existem exemplos na área médica com exames não invasivos, na área de segurança para o monitoramento noturno de ambientes, na área de construção civil para a inspeção de estruturas e até na área de saúde pública com o monitoramento de passageiros entrando em um país [1].

Com os investimentos que são feitos nessa área, hoje existem desde câmeras muito compactas, leves e baratas a câmeras de grande definição e qualidade de imagem, mais caras. A construção de câmeras assim requer uma eletrônica muito sofisticada, sistema óptico bem ajustado e um processador de imagens eficiente para o tratamento da imagem formada pelo sensor. Para as câmeras compactas, é necessário que a eletrônica envolvida também seja reduzida, porém rápida, capaz de processar vídeo de 640×512 pixels com uma taxa de 30 frames por segundo. Dessa forma, considerando um *field-programmable gate array* (FPGA), que como o próprio nome diz é um *hardware* digital programável, no qual é possível implementar processamentos específicos e completamente paralelos, ele se encaixa na proposta de processador de imagens eficiente e compacto. Assim, a elaboração de algoritmos que possam ser eficientemente sintetizáveis em FPGA é um passo importante para a evolução dos equipamentos térmicos.

Devido ao processo de fabricação dos detectores de infravermelho, são necessários principalmente os algoritmos de correção de não uniformidade do detector, substituição de *bad pixels* e redução da resolução de cor da imagem. Além deles, devido às características de imagens térmicas, se faz necessário o realce do contraste da imagem. Estes quatro algoritmos são chamados de algoritmos de pré-processamento, que geralmente são parte do processamento feito em câmeras térmicas.

Apesar de poucos países serem detentores da tecnologia necessária para a construção de um elemento sensor de infravermelho, é de importância estratégica para um país ter o *know-how* na área, sendo capaz de desenvolver um sistema de imageamento térmico. E, apesar de as forças armadas não serem as únicas detentoras do conhecimento nessa área, ainda são poucas

as informações disponibilizadas publicamente. Muitas soluções são segredos industriais, de empresas sediadas principalmente nos países pioneiros da área.

1.1 Objetivos

O objetivo principal desse trabalho é propor e elaborar um conjunto de algoritmos de processamento de imagens térmicas, implementado em FPGA e capaz de processar vídeo com *frames* de 640×512 *pixels* na taxa de 30 fps.

1.2 Motivação

Como as câmeras estão cada vez menores e com maior densidade de *pixels*, é interessante realizar o processamento da imagem em *hardware*. Um FPGA é um *hardware* digital programável, o que possibilita a implementação de algoritmos de processamento de imagens paralelos, portanto se encaixa muito bem na proposta de processador de imagens compacto e eficiente. A elaboração de algoritmos que possam ser eficientemente sintetizáveis em FPGA é um passo importante no desenvolvimento dos equipamentos térmicos.

Além dos algoritmos de correção de não uniformidade e substituição de *bad pixel* [2], que são necessários em todos os sistemas de imageamento térmico, serão apresentados dois outros algoritmos, um deles é o filtro de nitidez, que tem como objetivo aprimorar a visualização de objetos em imagens térmicas, e o outro é um algoritmo de redução da resolução de cor da imagem e realce de contraste.

O estudo e o desenvolvimento dos métodos desse trabalho são parte do desenvolvimento de uma câmera térmica, e a implementação eficiente desses algoritmos permite que sejam feitas melhorias significativas na imagem exibida ao observador.

1.3 Contribuições

As contribuições desse trabalho que podem ser destacadas são:

- Apresentação de dois algoritmos para realce da imagem exibida ao observador;
- Implementação dos algoritmos de processamento de imagens em *hardware*, com desempenho elevado e baixa latência.

Esse trabalho contribui para a área pois apresenta técnicas e ideias para a construção de um sistema de aquisição de imagens no infravermelho, principalmente no que tange o

processamento digital das imagens. O domínio dessa tecnologia é importante tanto para aplicações militares, pela possibilidade de se enxergar em ambientes com visibilidade ruim, quanto para aplicações civis, como na área da saúde.

Sendo assim, para a indústria é muito interessante obter o domínio do conhecimento desta tecnologia. Já para a academia, esse trabalho complementa o conhecimento em processamento de imagens do infravermelho.

1.4 Organização

Esse trabalho é dividido em seis capítulos. O primeiro é composto da introdução e objetivos, além das contribuições propostas.

O segundo capítulo consiste da história relacionada à radiação infravermelha, sua descoberta e eventos marcantes desde então, bem como aplicações de imagens por infravermelho. Após a história, serão apresentadas características do infravermelho que são relevantes para o trabalho, pois têm influência na qualidade da imagem final. Também serão descritos brevemente os tipos de detectores existentes, os sistemas ópticos utilizados para infravermelho e algumas propriedades dos detectores, que também influenciam na qualidade da imagem e são relevantes para o desenvolvimento dos algoritmos. Depois, a forma como é feita a calibração e os testes das câmeras será explicada e, fechando a parte de infravermelho, serão apresentados os algoritmos conhecidos de processamento de imagens no espectro do infravermelho. Finalmente, uma breve descrição dos FPGA será feita.

No terceiro capítulo, primeiramente são apresentados o equipamento e as ferramentas utilizadas no desenvolvimento do trabalho. Depois o fluxo do desenvolvimento realizado é brevemente comentado. E por fim, o desenvolvimento do trabalho é apresentado, com explicações a respeito da implementação, tanto em *software* quanto em *hardware*, de cada algoritmo proposto.

Os resultados obtidos são apresentados no quarto capítulo, e as conclusões no capítulo cinco. Por último, no sexto capítulo são apresentadas propostas para trabalhos futuros.

2 Fundamentação teórica

Diz-se que a termografia nasceu nos tempos antigos, tempo dos filósofos Platão, Aristóteles, Hipócrates e Galeno, que acreditavam na relação entre calor e vida [3]. Hipócrates considerava que o aumento da temperatura do corpo humano indicava alguma doença (até hoje diagnosticamos algumas doenças com o termômetro). Em algum momento, ele espalhou lama sobre o corpo de um amigo doente e percebeu que a lama secou e rachou primeiro onde existia uma inflamação interna, que gerava calor [4].

Desde então muita coisa evoluiu, começando com a invenção dos termômetros até novos tipos de sensores e equipamentos, que são desenvolvidos e aperfeiçoados até hoje. Essa evolução só se deu graças ao descobrimento da radiação infravermelha, por Herschel em 1800 [5].

Neste capítulo será feita uma breve apresentação da radiação no espectro do infravermelho e de suas características. Também serão apresentados alguns tipos de detectores de infravermelho, aplicações e algoritmos de processamento de imagem.

Uma sucinta descrição de FPGA e das ferramentas utilizadas para o desenvolvimento de projetos digitais também será feita em seguida.

2.1 História

Como dito por Crawford em seu trabalho [5], a radiação na faixa do infravermelho foi descoberta em 1800 por Sir William Herschel, astrônomo real do rei George III da Inglaterra. Herschel, investigando a temperatura gerada pelas diferentes cores da luz do sol, ao passar a luz por um prisma descobriu que a temperatura aumentava na direção do violeta ao vermelho. E mais, a região "incolor" ao lado do vermelho era mais quente que o próprio vermelho. Na época, Herschel denominou a região de "Espectro Termométrico".

Herschel, e outros pesquisadores, realizaram diversos outros experimentos para confirmar a nova descoberta. Todos eles utilizavam sistemas ópticos baseados em vidros, de diferentes características para o infravermelho. Herschel então concluiu que os sistemas ópticos para o infravermelho estariam limitados à apenas elementos reflexivos, ou seja, espelhos. Porém, pouco tempo depois, em 1830, o pesquisador italiano Macedonio Melloni descobriu que os cristais de sal de rocha (NaCl) eram bastante transparentes ao infravermelho [6]. Esse

material foi utilizado para a construção de sistemas ópticos para o infravermelho até o advento dos cristais sintéticos, em 1930. Além disso, Melloni também foi pioneiro ao desenvolver, em 1835, um dispositivo detector de calor, conectando vários termoacopladores em série.

Em 1940 foi feita a primeira imagem térmica. Sir John Herschel, filho do descobridor do infravermelho deixou uma fina camada de óleo, com uma espécie de papel por cima, exposta a um padrão de calor. A evaporação do óleo, proporcional ao calor da área, ocorria de forma não uniforme, formando uma imagem no papel, que podia ser relacionada com o padrão de calor [7].

Mais tarde Samuel Pierpont Langley desenvolveu um dispositivo formado por duas finas camadas de platina escurecida, uma exposta à radiação e outra blindada. As duas tiras formavam uma parte de uma ponte de *Wheatstone*, que foi conectada à uma bateria e um galvanômetro. Radiação infravermelha incidia na platina exposta e a esquentava, o que mudava sua resistência, permitindo a quantificação da radiação [8]. Diz-se, que em 1880 o bolômetro de Langley foi capaz de detectar o calor de uma vaca a aproximadamente 500 m de distância.

A partir de 1900 o infravermelho passou a ser bastante pesquisado. Com o contínuo desenvolvimento dos sensores de infravermelho, após a primeira e durante a segunda guerra mundial foram desenvolvidos os sistemas passivos, que eram capazes de detectar radiação infravermelha sem que o objeto fosse iluminado por uma fonte externa de radiação [9]. Esses sistemas passivos podem captar radiação nas faixas de infravermelho distante ou infravermelho médio, sendo assim possível "enxergar na escuridão", sem ser necessária qualquer fonte de radiação infravermelha.

2.2 Aplicações das imagens térmicas

Equipamentos que geram imagens térmicas tem diversas aplicações. É possível encontrar exemplos de utilização dessa tecnologia principalmente nas áreas médica, aeroespacial, de segurança e industrial.

Na área da medicina existem trabalhos que estudam a utilização dessas câmeras para diagnóstico de câncer de mama, como no exemplo mostrado por Arora *et al* [10]. Já Diakides e Bronzino mostram outros casos em medicina [11], como utilização de imagens térmicas em cirurgias, em odontologia e até na medicina veterinária. A medicina esportiva também tem aplicações para esse tipo de imagens [12].

Aplicações de segurança e defesa são bastante conhecidas, alguns exemplos são as câmeras de segurança e os equipamentos militares de visão noturna.

Já para a área industrial existem diversos tipos de aplicações. O monitoramento de instalações elétricas e mecânicas pode ser feito com câmeras térmicas para detectar algum ponto de sobreaquecimento, vazamento de gases podem ser identificados, e é possível determinar quais componentes estão sobreaquecendo em placas eletrônicas. Esses são apenas alguns exemplos, e já existem equipamentos comerciais com esses fins.

Todas as aplicações se baseiam no fato das câmeras térmicas "enxergarem" temperatura de forma passiva e não invasiva. Na medicina, diagnosticar patologias dessa forma é muito mais cômodo para o paciente, já que ele apenas precisaria ficar parado em frente a câmera, sem incisões nem outro tipo de radiação, como os raios X. Em segurança, enxergar no escuro é uma grande vantagem, pois mesmo à noite é possível monitorar um ambiente e identificar invasores, coisa impossível sem o equipamento apropriado. E a manutenção de sistemas industriais geralmente é facilitada com a detecção de pontos com temperatura alterada, para mais quente, como em um possível curto circuito, ou mais frio, como em um vazamento de gás.

2.3 Radiação no infravermelho

A luz visível é apenas uma pequena fração do espectro eletromagnético total. Além dela, o espectro é composto por raios gama, raios X, ultravioleta, infravermelho, micro-ondas e ondas de rádio, conforme a figura 1. O espectro eletromagnético é responsável por diversas interações nossas com o mundo, como por exemplo: a luz visível nos permite ver; as ondas de rádio nos possibilitam TV e rádio; as micro-ondas são utilizadas em radares e equipamentos de micro-ondas para cozinha; e os raios X permitem que vejamos nossos órgãos internos, para diagnosticar algum problema de saúde [13].

Figura 1 – Espectro eletromagnético, adaptado de [14]

A radiação infravermelha compreende os comprimentos de onda entre $0,70\text{ }\mu\text{m}$ até 1 mm , imediatamente maiores que o comprimento da onda de luz vermelha, que fica em torno de $0,65\text{ }\mu\text{m}$, e por isso é chamado de infravermelho (a frequência do infravermelho é logo abaixo da do vermelho). Dentro do espectro infravermelho temos mais uma subdivisão, da qual cada faixa apresenta características diferentes. Uma divisão bastante utilizada para o espectro de radiação infravermelha [15] é detalhada a seguir.

De $0,75\text{ }\mu\text{m}$ até $1,4\text{ }\mu\text{m}$ está o infravermelho próximo (NIR, ou *near infrared*), que é caracterizado por ser bastante absorvido pela água. É comumente utilizado em comunicações por fibras ópticas, porque tem atenuação muito pequena em vidro de sílica (SiO_2). Os controles remotos, comuns em TVs, também utilizam NIR, tipicamente de 940 nm .

O infravermelho de ondas curtas (SWIR, ou *short-wavelength infrared*) compreende os comprimentos de onda de $1,4\text{ }\mu\text{m}$ até $3\text{ }\mu\text{m}$. Essa faixa pode ser utilizada para comunicação de longa distância e também para câmeras de segurança. As faixas NIR e SWIR são consideradas infravermelho refletido, por isso as câmeras que operam nessas faixas devem funcionar em conjunto com leds emissores de infravermelho, que iluminam a cena observada.

Dentro da faixa de $3\text{ }\mu\text{m}$ até $5\text{ }\mu\text{m}$ está o infravermelho de ondas médias (MWIR, ou *medium-wavelength infrared*). Objetos extremamente quentes, como foguetes, emitem radiação MWIR, portanto os *seekers* de mísseis guiados por temperatura são um exemplo de equipamentos que trabalham nessa faixa.

Para a observação de objetos menos quentes, como é o caso de pessoas, as câmeras geralmente trabalham na faixa de $8\text{ }\mu\text{m}$ a $15\text{ }\mu\text{m}$, o infravermelho de ondas longas (LWIR, ou *long-wavelength infrared*). Nessa faixa do espectro é possível "enxergar no escuro", com câmeras sendo capazes de gerar imagens apenas com o infravermelho emitido pelos objetos, sem a necessidade de um emissor externo. MWIR e LWIR geralmente são referidos como infravermelho termal, pois a radiação é emitida pelo objeto, devido ao seu calor interno. Por fim, existe a faixa de $15\text{ }\mu\text{m}$ até 1 mm que é chamada de infravermelho distante (FIR, ou *far infrared*) e é observada principalmente na astronomia.

Porém parte do espectro do infravermelho não é utilizável para captura de imagens. Isso acontece porque determinados comprimentos de onda são absorvidos pela atmosfera (principalmente pela água ou pelo CO_2), sendo assim inviáveis para a formação de uma boa imagem. A figura 2 mostra a taxa de transmissividade da atmosfera para diferentes tipos de radiação, é interessante notar que a região IR Refletido se refere à NIR e SWIR enquanto IR Termal compreende MWIR e LWIR. Pela figura é possível observar que as regiões NIR, SWIR,

MWIR e LWIR possuem componentes de pequena absorção atmosférica, que são as faixas utilizadas para aquisição de imagens.

Figura 2 – Janela atmosférica de absorção de radiação, adaptado de [16]

2.3.1 Emissividade de infravermelho

Qualquer objeto com temperatura superior a 0 K emite radiação eletromagnética, e a intensidade da radiação emitida pelo objeto tem relação direta com sua temperatura [9]. Sistemas de imagem para radiação visível geralmente captam radiação refletida (um objeto verde é visto dessa cor pois reflete radiação de comprimento de onda em torno de 510 nm, ou seja, verde), porém, em alguns casos, podem captar luz visível emitida pelo objeto, como o vermelho de um objeto tão quente que fica avermelhado (lâmpada incandescente, por exemplo). Já sistemas para infravermelho geralmente captam tanto radiação refletida (que depende do ambiente) quanto radiação emitida (que depende da temperatura e do material do objeto). Na figura 3 um clássico exemplo de um objeto emitindo luz visível e uma imagem gerada por uma câmera térmica, capturando radiação infravermelha.

Figura 3 – Luz incandescente emitindo luz visível e imagem de luz incandescente no espectro infravermelho

A emissividade é a quantidade de radiação emitida por um objeto e está diretamente relacionada à temperatura desse objeto. Ela pode variar de 0, quando nenhuma parte da radiação observada é radiação emitida (superfície totalmente reflexiva), até 1, quando toda a radiação observada é emitida pelo objeto (corpo negro).

Para se chegar ao conceito prático de emissividade [17], deve ser considerado um objeto em equilíbrio térmico com o ambiente, ou seja, sua temperatura não muda com o tempo. Assim, toda a radiação direcionada ao objeto ou é refletida, ou absorvida ou simplesmente atravessa o objeto (transmitida), conforme a equação 1.

$$\%Refletida + \%Absorvida + \%Transmitida = 100\% \quad (1)$$

Considerando que o objeto seja opaco para a radiação, ele não transmite, portanto, eliminando a componente $\%Transmitida$ da equação 1, chega-se na equação 2.

$$\%Refletida + \%Absorvida = 100\% \quad (2)$$

Quando o objeto absorve radiação, ou energia, e não está em equilíbrio térmico, sua temperatura sobe. Porém, para o objeto em equilíbrio térmico, a temperatura é mantida e toda a radiação absorvida é emitida. Substituindo a componente $\%Absorvida$ pela emissividade do objeto em equilíbrio térmico as equações 3 e 4 são definidas.

$$\%Refletida + Emissividade = 100\% \quad (3)$$

Ou:

$$Emissividade = 100\% - \%Refletida \quad (4)$$

Com a equação 4 é fácil ver que a emissividade de um corpo está inversamente relacionada com sua refletividade, ou seja, quanto menos um material é reflexivo, maior a componente de radiação emitida. Se considerarmos um objeto translúcido, basta adicionar a componente de *Transmissividade* à equação. Com essas ideias, praticamente reescrevendo a equação 1, chega-se na equação da emissividade, equação 5:

$$Emissividade + Refletividade + Transmissividade = 1 \quad (5)$$

A emissividade não está apenas relacionada com o material de um objeto, pois outros fatores, como textura, também influenciam. Um objeto ideal, com emissividade igual à unidade, é chamado de corpo negro. Por isso, equipamentos de calibração de câmeras térmicas que

possuem emissividade bastante elevada também são chamados de corpo negro, e são utilizados com esse fim pois toda a radiação observada está relacionada com sua temperatura.

2.3.2 Transmissão e espalhamento atmosférico

Aplicações que utilizam radiação no infravermelho também são influenciadas pelas características da atmosfera, que tem diferentes efeitos sobre as diferentes faixas do infravermelho. Dois efeitos importantes são a absorção e o espalhamento.

Como apresentado na janela atmosférica de absorção de radiação, figura 2, o valor de transmissividade varia para os diferentes comprimentos de onda. Isso ocorre porque cada um desses comprimentos pode ser mais, ou menos, absorvido pelas moléculas presentes na atmosfera. As faixas de comprimentos de onda que não são muito afetadas são as faixas utilizadas para obtenção de imagens.

O efeito de espalhamento também reduz a quantidade de radiação que chega no elemento sensor, porém essa redução ocorre porque a direção original da radiação é alterada, como nos casos de reflexão, refração e difração da luz. Esse efeito não tem grande impacto na região do infravermelho termal, a não ser em condições de chuva ou neblina.

2.3.3 Radiação na cena observada

A radiação captada por um detector de infravermelho é resultante da soma das radiações emitida, refletida e transmitida pelo objeto. Geralmente a cena observada possui vários objetos e portanto existe a interação entre eles. Se os objetos e o fundo da cena possuem temperaturas parecidas, a radiação refletida entre eles pode diminuir o contraste. Porém, a reflexão de objetos mais quentes, ou mais frios, também pode influenciar a cena. Sharma mostra em seu trabalho [18] que para a região MWIR, 3 a 5 μm , a reflexão do sol é significativa, o que, dependendo do material de cada elemento da cena, pode melhorar o contraste. Ele também diz que o contraste em imagens térmicas é pequeno se comparado com o contraste em imagens visíveis e isso se deve, principalmente, à refletância dos materiais, que pode influenciar bastante em uma cena na qual os objetos possuem temperaturas similares.

Rogalski diz em seu trabalho [19] que o contraste é um fator importante para sistemas que captam imagens no infravermelho e apresenta uma comparação entre contrastes para diferentes comprimentos de onda, figura 4. Pode-se perceber que o contraste para comprimentos de onda menores, MWIR, é maior.

Figura 4 – Comparação entre curvas de contraste por temperatura da cena para comprimentos de onda MWIR e LWIR, de [19]

2.4 Detectores de infravermelho

Os detectores são a porta de entrada de uma câmera térmica, sendo eles que recebem a radiação infravermelha emitida pela cena e convertem essa energia em alguma unidade possível de ser medida. Um exemplo é o bolômetro, no qual o elemento sensor tem sua resistência elétrica alterada de acordo com sua temperatura, que por sua vez é alterada com a incidência de infravermelho. Existem diversos outros fenômenos que são aproveitados para a fabricação de sensores de infravermelho, são alguns exemplos: efeito termoelétrico; expansão de gás; piroeletricidade; e *quantum-well* [9].

Podemos dividir os sensores de infravermelho em duas classes, os detectores térmicos e os detectores quânticos.

Os detectores térmicos respondem à variação de sua temperatura. A incidência de radiação esquentando o sensor causando variação no sinal de saída. Exemplos desse tipo de sensor são:

- Termopares: formado de duas junções de metal, quando uma delas é aquecida, elétrons fluem entre as junções, variando a tensão entre elas;
- Bolômetros: quando aquecido, a resistência elétrica do sensor varia;
- Pirômetros: quando a temperatura aumenta a carga superficial varia, ou seja, a capacitância do sensor varia de acordo com a temperatura.

Desses, os mais utilizados para a construção de câmeras térmicas são os baseados em bolômetros. Antigamente os detectores tipo bolômetros eram bastante lentos, caros e pouco

sensíveis, mas com o avanço da tecnologia dos semicondutores eles evoluíram bastante e em 2010 respondiam por mais de 95% das câmeras térmicas [9]. Um detector de infravermelho, baseado em bolômetros, com uma matriz de sensores (320x240, por exemplo) é chamado de microbolômetro, apresentado na figura 5. Os microbolômetros são geralmente utilizados para câmeras que operam na faixa do LWIR, podendo também ser encontrado para a faixa do MWIR.

Figura 5 – Detector de infravermelho do tipo microbolômetro [20]

Os detectores quânticos respondem à taxa de fótons incidentes nos elementos sensores. O sinal de saída não depende da temperatura, mas sim da radiação incidente. Alguns exemplos:

- Fotocondutor: a condutividade do elemento varia de acordo com a incidência de radiação;
- Fotovoltaicos: são formados por junções P-N polarizadas reversamente, quando existe incidência de radiação sobre a junção, são criados elétrons-buraco que se deslocam produzindo variação de tensão e corrente.
- *Quantum-well* (QWIR): cada elemento sensor é formado por um "sanduíche" de elementos químicos, e a incidência de radiação excita os elétrons da camada central, o que muda a condutividade do sensor. Com essa abordagem é possível calibrar o sensor para determinados comprimentos de onda e ajustar a sensibilidade do sensor.

Detectores quânticos possuem boa taxa sinal-ruído e resposta rápida, porém para isso é necessário refrigerá-los criogenicamente, em torno de 70 K pois, sem a refrigeração, os elétrons são excitados também pela temperatura do detector, o que causa ruído no sinal de saída. Esse é o principal desafio na construção de detectores tipo QWIR, que, além do detector de infravermelho, necessita de um sistema de refrigeração. A figura 6 é um detector refrigerado selado dentro de uma cápsula que contém vácuo e um *cooler Sterling*, que mantém o detector na temperatura de operação ótima.

Figura 6 – Detector de infravermelho refrigerado [21]

Existem variações de detectores do tipo QWIR, que diferem entre si principalmente na seleção dos materiais que formam os sensores. Um tipo bastante utilizado é o de HgCdTe (Telureto de mercúrio e cádmio), pois esse material permite grande liberdade na construção dos detectores, podendo ser ajustado para operar em SWIR, MWIR e LWIR [9].

A construção de um detector de infravermelho envolve mais coisas além do elemento sensor (bolômetro, QWIR, etc.). Para um detector simples, ainda é necessário construir a matriz de sensores, uma eletrônica refinada para controlar essa matriz e converter o sinal de cada pixel em um sinal interpretável, a óptica do detector e um sistema de refrigeração. O detector completo pode ser chamado de plano focal.

2.5 Óptica para câmeras térmicas

Assim como as câmeras visíveis, as câmeras térmicas necessitam de um sistema óptico para focalizar a luz da cena no detector, a diferença está no material utilizado para construir as lentes. O material deve ser transparente para a radiação infravermelha (o vidro, transparente para luz visível, é extremamente reflexivo para o infravermelho). O germânio (Ge) e o seleneto de zinco (ZnSe) são materiais comumente utilizados na fabricação dessas lentes [22]. A título de exemplo, a transmissividade do germânio para o infravermelho pode ser vista na figura 7-a, para uma lente de 5 mm de espessura. É possível perceber que as maiores taxas se encontram nas faixas de 2 a 15 μm , ou seja, MWIR e LWIR.

Além do material, é interessante fazer o tratamento da lente com um filme fino. Os filmes finos são uma fina camada de algum material geralmente com o objetivo de reduzir a taxa de reflexão da radiação incidente na lente, aumentando assim a quantidade de radiação que passa para o detector. Esses são os chamados filmes antirreflexo. Na figura 7-b é possível ver a melhora na transmissividade de uma lente de Ge com filme antirreflexo (para a faixa de 8 a 12 μm).

a)

b)

Figura 7 – Transmissividades de uma lente de Ge: (a) sem filme antirreflexo; (b) com filme antirreflexo.
Adaptado de [23]

O sistema óptico que forma a imagem no detector, portanto, tem influência direta na qualidade da imagem exibida pela câmera. Algumas aberrações podem ser detectadas na imagem caso o sistema óptico não esteja bem projetado e bem montado. Aberração esférica, coma e astigmatismo são alguns exemplos.

2.6 Considerações a respeito de um detector

Nesse item serão descritas algumas características típicas de sistemas de detecção de infravermelho (detector acoplado à óptica) que têm importante impacto na qualidade da imagem gerada pelo sistema. São elas: responsividade; linearidade; e ruído.

2.6.1 Responsividade

Os primeiros detectores de infravermelho eram montados com apenas um elemento sensor e uma óptica móvel refinada que fazia a varredura da cena. Hoje são comuns detectores compostos por uma matriz de *pixels*, na qual cada pixel pode responder à mesma incidência de radiação de forma diferente, ou seja, se todos os pixels forem iluminados com a mesma quantidade de radiação, a imagem gerada será não uniforme. Isso ocorre por causa da imperfeição do processo de fabricação dos detectores [24].

Esse comportamento é facilmente observado quando se posiciona um corpo negro em frente à câmera, não calibrada. Um padrão fixo, invariante no tempo, de linhas e colunas pode ser visto na imagem de saída.

2.6.2 Linearidade

Cada *pixel* de um detector possui um gráfico de responsividade parecido com o da figura 8. Pelo gráfico é possível ver que o sensor responde de maneira consideravelmente linear, dentro de determinada faixa de radiação incidente. Fora dessa faixa a resposta não é linear, portanto é interessante calibrar o detector para que ele trabalhe sempre dentro dessa região.

Figura 8 – Exemplo de responsividade de um sensor de infravermelho

2.6.3 Ruído

Para imagens térmicas são definidos diversos tipos de ruído, que podem ser divididos em ruído espacial e ruído temporal. Ruído espacial é aquele que não varia com o tempo, varia apenas com a posição na matriz de *pixels*, e ruído temporal varia no tempo, a cada *frame* ele pode mudar. A não uniformidade, causada pela diferença na curva de responsividade dos *pixels*, é um exemplo de ruído espacial e o ruído gerado pela fonte de alimentação da câmera é um tipo de ruído temporal.

A quantidade de ruído em uma imagem é determinante para caracterizar a qualidade do sistema todo. Além disso, a caracterização do ruído presente no sistema de aquisição de imagens fornece dados importantes para calibração e aperfeiçoamento da câmera. Na tabela 1 são apresentadas possíveis causas de determinados tipos de ruído na imagem. Sabendo o tipo de ruído predominante é possível avaliar mais facilmente a causa do problema.

Tabela 1 – Relação entre ruídos em imagens térmicas e suas causas, adaptado de [25]

Tipo de ruído	Possível causa
Ruído espaço-temporal aleatório	Ruído temporal do detector
Ruído temporal de linha	Processamento da linha, 1/f, circuito de leitura
Ruído temporal de coluna	Efeitos de <i>scan</i> (<i>scanning arrays</i>)
Ruído espacial, padrão fixo bidirecional	Processamento do <i>pixel</i> , não uniformidade do detector, ruído 1/f
Ruído fixo por linha, não uniformidade entre linhas	Não uniformidade do detector
Ruído fixo por coluna, não uniformidade entre colunas	Efeitos de <i>scan</i> , não uniformidade do detector
Ruído entre <i>frames</i> consecutivos	Processamento do <i>frame</i>

Para detectores matriciais, os tipos mais significativos de ruído são os ruídos oriundos da não uniformidade do detector.

2.7 Pré-processamento de imagens térmicas

Holst e Bittencourt em seus trabalhos [25] [2] afirmam que existem alguns algoritmos básicos de processamento de imagens que devem ser aplicados às imagens captadas no infravermelho. Correção de não uniformidade, substituição de *pixels* defeituosos e ajuste de histograma são fundamentais. Além desses, podem ser aplicados outros algoritmos visando aprimorar a qualidade da imagem [25], mas esses algoritmos geralmente são de empresas privadas e apenas seus efeitos sobre as imagens são demonstrados. Um exemplo é o *Digital Detail Enhancement* [26].

2.7.1 Correção de não uniformidade - NUC, *Non Uniformity Correction*

A correção de não uniformidade visa diminuir os efeitos causados pela diferença na responsividade dos *pixels* de um detector. Os métodos mais comuns são a correção de um ponto e a correção de dois pontos [25]. Elas possuem esse nome pois o objetivo é aproximar as curvas de responsividade dos *pixels* em um ou em dois pontos, respectivamente. Assim, considerando

a correção de dois pontos, seu objetivo é igualar o sinal de saída dos *pixels* em pelo menos dois valores diferentes de radiação incidente. Na figura 9 é possível ver um exemplo do efeito das correções de um e de dois pontos nas curvas de responsividade de quatro *pixels* um detector (p_0, p_1, p_2 e p_3).

Figura 9 – Curvas de responsividade de um sensor: (a) Sem correção; (b) Com correção de um ponto; (c) Com correção de dois pontos. Adaptado de [27]

A correção de um ponto é realizada apenas somando um valor de *Offset* ao sinal de saída de cada *pixel*. Na correção de dois pontos, o sinal de cada *pixel* é primeiro multiplicado por um valor de *Ganho*, ajustando a inclinação da curva, e depois somado a um valor de *Offset*. Devem ser determinados um valor de *Ganho* e um de *Offset* para cada *pixel* do detector.

Para a correção de dois pontos, os valores de *Ganho* e *Offset* de cada *pixel* podem ser determinados através de um procedimento de calibração, que utiliza um corpo negro como referência. Configura-se o corpo negro para uma temperatura inferior e armazena-se o *frame* completo, chamado de *frame* frio, ou $F_f(p)$, com p sendo o índice do *pixel*. Em seguida configura-se o corpo negro para uma temperatura superior e armazena-se esse novo *frame* capturado (*frame* quente, ou $F_q(p)$). É importante que as temperaturas do corpo negro estejam dentro da região linear do detector. Então, com esses dois *frames* é possível saber a responsividade de cada *pixel* do detector para a região calibrada.

O próximo passo é calcular os valores médios dos *pixels* para cada *frame*. Na equação 6, P_f e P_q são os *pixels* médios para o *frame* frio e para o *frame* quente, respectivamente. A quantidade total de *pixels* no *frame* é indicada por N .

$$P_f = \frac{\sum F_f(p)}{N} \text{ e } P_q = \frac{\sum F_q(p)}{N} \quad (6)$$

Basta realizar alguns cálculos matemáticos para levar todos os *pixels* em F_f para o valor P_f , e os *pixels* em F_q para P_q , e determinar os valores de *Ganho* ($G(p)$) e de *Offset* ($O(p)$). Os cálculos são demonstrados nas equações 7 e 8.

$$\begin{cases} P_f = G(p) \times F_f(p) + O(p) \\ P_q = G(p) \times F_q(p) + O(p) \end{cases} \quad (7)$$

Desenvolvendo:

$$\begin{cases} G(p) = \frac{P_q - P_f}{F_q(p) - F_f(p)} \\ O(p) = P_f - G(p) \times F_f(p) \end{cases} \quad (8)$$

Já para a correção de um ponto, captura-se apenas um *frame* na temperatura desejada e calcula-se o *pixel* médio desse *frame*. Em seguida, encontra-se o valor de *Offset* ($O(p)$) que leva todos os *pixels* do *frame* ($F(p)$) para o valor do *pixel* médio (P), conforme a equação 9. Esse valor substitui o valor de *Offset* determinado durante a calibração de dois pontos.

$$O(p) = P - F(p) \quad (9)$$

Além desses dois métodos, é possível aumentar o número de pontos, dividindo a responsividade do detector em mais de uma região linear, assim, para cada *pixel*, deverá ser determinado mais de um valor de *Ganho* e *Offset*, melhorando a correção, porém aumentando a complexidade do sistema.

A forma como os valores de *Ganho* e de *Offset* são determinados também pode variar. Existem trabalhos que demonstram o uso de sistemas adaptativos [28] [29], que não necessitam do aparato de calibração (corpo negro), somente precisam de variações significativas na cena observada.

Após a calibração, com o aquecimento ou resfriamento da câmera pode ocorrer um efeito chamado de *temperature drift*. O corpo da câmera, composto do sistema óptico e das paredes próximas do detector, contribui com a quantidade de radiação incidente no detector, portanto, se ele esquenta ou esfria, as curvas de responsividade dos *pixels* podem ter ligeira variação, de maior significância na componente constante da curva.

Para corrigir esse efeito pode-se utilizar a calibração de um ponto e atualizar os valores de *Offset* da câmera. Idealmente utiliza-se o corpo negro para realizar a calibração, porém, com a câmera em utilização, o corpo negro pode não estar disponível. Uma ideia para se contornar essa situação é utilizar um material, com emissividade alta, no lugar do corpo negro. Em câmeras portáteis é comum o uso de um *shutter*, que é um pequeno pedaço de metal capaz de cobrir toda a área do detector. De tempos em tempos o *shutter* é posicionado em frente ao detector e a calibração de um ponto é realizada.

2.7.2 Substituição de *pixels* defeituosos - BPR, *Bad Pixel Replacement*

Devido ao processo de fabricação dos detectores é comum que estes apresentem alguns *pixels* que não respondem à incidência de radiação, ou eles respondem à radiação, mas de uma forma muito diferente da média. Esses *pixels* são chamados de *pixels* defeituosos, ou *bad pixels*. A rotina de BPR simplesmente visa substituir esses *pixels* por um vizinho (ou a média dos *pixels* vizinhos).

Para se determinar quais *pixels* podem ser considerados defeituosos existem três critérios: critérios do *offset*; do ruído; e da responsividade. A seguir serão brevemente explicados os três critérios.

O critério do *offset* procura por *pixels* que, para um *frame* uniforme, apresentam sinal de saída diferente da média do sinal do *frame* todo, considerando-se uma margem de tolerância. Para isso basta configurar o corpo negro para uma determinada temperatura, calcular a média do *frame* e verificar, *pixel a pixel*, se o valor do sinal é diferente da média.

O critério do ruído considera que *pixels* com nível de ruído diferente de uma região em torno da média são *pixels* defeituosos. Para determinar esses *pixels* captura-se vários *frames* uniformes. Calcula-se a média dos *frames* e então o valor médio dos *pixels* desse *frame* médio. Com esse valor e com todos os *frames* capturados calcula-se o desvio padrão de cada *pixel*. Então determina-se o valor médio dos desvios padrão e aqueles *pixels* que possuem desvio padrão diferente da região da média são considerados *pixels* defeituosos.

Para o critério de responsividade deve-se capturar um *frame* frio e um *frame* quente. Calculando a diferença, *pixel a pixel*, entre os *frames*, são determinados os valores que representam a responsividade de cada *pixel*. A seguir calcula-se a média entre os valores de responsividade e o *pixel* que possuir responsividade diferente da média, considerando uma margem de tolerância, é considerado *pixel* defeituoso.

Ao final da aplicação dos três critérios a lista dos *bad pixels* estará formada. O método de substituição mais simples é a substituição por um vizinho próximo não defeituoso. Outro método é a substituição pela média dos *pixels* vizinhos que não são defeituosos.

2.7.3 Realce de contraste

Imagens térmicas tendem a ter menos contraste que imagens visíveis, já que a radiação infravermelha observada na cena é composta das componentes emitida, refletida e transmitida pelos objetos, diferentemente da luz visível, que é basicamente composta de radiação refletida. Geralmente, algoritmos com a função de melhorar o contraste de uma imagem utilizam

informações do seu histograma. A seguir são apresentados dois algoritmos conhecidos com esse propósito.

A equalização do histograma é uma transformação não linear que visa espalhar a distribuição dos níveis por toda a faixa dinâmica do histograma, distribuindo melhor, por exemplo, os níveis de cinza de uma imagem. A figura 10 é um exemplo de equalização de histograma.

Figura 10 – Equalização de histograma: (a) Imagem original; (b) Histograma original; (c) Imagem equalizada; (d) Histograma equalizado. Adaptado de [30]

Já o algoritmo de alargamento de contraste utiliza transformações lineares para aumentar o contraste de uma imagem, com a intenção de dar mais resolução de cor para determinadas faixas do sinal de entrada. No exemplo da figura 11, a região que ganhou contraste foi a região intermediária, entre os pontos (r_1, s_1) e (r_2, s_2) .

Figura 11 – Alargamento de contraste: (a) Curvas de alargamento de contraste; (b) Imagem de entrada; (c) Imagem de saída. Adaptado de [31]

Lisani *et al* mostram em seu trabalho [32] um algoritmo de realce de contraste chamado de *Piecewise Affine Equalization* (PAE), que busca, utilizando o algoritmo de alargamento de contraste, obter um resultado semelhante, ou melhor, ao da equalização de histograma. O autor mostra que aumentando o número de subintervalos do alargamento, e consequentemente de equações lineares, obtém-se resultados semelhantes e com menor complexidade que a equalização de histograma.

2.7.4 Redução da resolução de cor

Dispositivos de exibição de imagens, como monitores e *displays*, frequentemente trabalham com resolução de cor de 8 bits, para *displays* em escala de cinza. Ou seja, a intensidade de cada *pixel* é representada por um valor de 8 bits. Já os detectores de imagens podem possuir faixa dinâmica maior, com resolução de cor maior que 8 bits.

A transição do sinal de vídeo, entre a faixa dinâmica do detector para a do *display*, pode causar perdas significativas na imagem se não for bem feita. A figura 12 é um exemplo da redução de uma imagem com *pixel* de M bits para uma imagem com *pixel* de N bits. Nesse exemplo são utilizadas funções lineares para realizar a redução da resolução de cor. Dessa forma é possível notar a semelhança dessas curvas com a da figura 11-a. A única diferença entre elas é que nesse caso a resolução de cor da imagem de entrada é diferente da resolução da imagem de saída do algoritmo, na figura 11-a as imagens de entrada e de saída possuem resolução de cor iguais.

Figura 12 – Exemplo de redução da resolução de cor de uma imagem

Durante a redução da resolução de cor é importante que o algoritmo minimize as perdas da imagem. Considerando uma imagem de entrada com resolução de cor de 12 bits, ou 4096 níveis de cinza, e a imagem de saída devendo ter 8 bits por *pixel*, ou 256 níveis, é nítida a diminuição na quantidade de cores que podem ser representadas na imagem de saída, por isso,

durante a redução, o algoritmo deve dar ênfase para determinadas regiões de cores do histograma de entrada, em detrimento de regiões menos importantes.

2.8 Calibração e avaliação da imagem

Tanto para realizar a calibração do equipamento, quanto para avaliar a imagem final da câmera, é necessário dispor de equipamentos de suporte. O principal deles é o corpo negro.

2.8.1 O corpo negro

Corpo negro, ou *blackbody*, é um equipamento que possui uma área bastante uniforme que não reflete radiação infravermelha, chamada de área de *target*. Se o corpo negro ideal é configurado para 30 °C, toda a radiação observada na área de *target* é proveniente de radiação emitida, ou seja, a radiação observada é equivalente a 30 °C, pois o corpo negro tem emissividade igual à um.

É possível utilizar diferentes tipos de *targets*. Quando toda a área é exposta, geralmente é para a realização da calibração do sistema. Para testes e avaliação do sistema são utilizadas chapas que podem sobrepor partes da área de *target*. O tipo mais comum cobre metade da área de *target*, assim metade da cena fica em temperatura ambiente e metade fica na temperatura configurada no corpo negro.

2.8.2 Avaliação da imagem por MTF

Quando se deseja conhecer a capacidade da câmera de reproduzir detalhes da cena observada, um bom teste é o da função de transferência de modulação (*modulation transfer function*, ou MTF) [25]. Esse teste se baseia no uso de um corpo negro com um *target* do tipo borda, gerando uma cena com uma transição bem definida de intensidade de radiação. A partir dessa cena, são feitos alguns cálculos matemáticos para determinar quanto a reprodução da borda captada é fiel.

Com o *target* posicionado em frente à câmera, apenas uma linha, ou a média de algumas linhas, é avaliada. Essa linha retirada da imagem forma o gráfico da função de espalhamento de borda (*edge spread function*, ou ESF), pois é a resposta do sistema para uma borda. Derivando-se a curva ESF é determinada a função de espalhamento de linha (*line spread function*, ou LSF), que traduz a capacidade do sistema de reproduzir uma linha apenas. Por fim, calculando a transformada de *Fourier* da curva LSF determina-se o MTF, que é a relação entre atenuação do sistema sobre o sinal de entrada com relação à frequência espacial desse sinal.

Considerando um sistema ideal, o cálculo do MTF é apresentado na figura 13. Nesse caso, a curva de MTF é constante em 1, o que indica que o sistema não atenua nenhuma frequência. Assim, quanto mais próximo da unidade o gráfico de MTF for, melhor é a capacidade do sistema de reproduzir detalhes, que são componentes de frequência espacial alta.

Figura 13 – Sequência de cálculo do MTF para um sistema ideal: (a) Imagem de entrada; (b) Função de espalhamento de borda, ESF; (c) Função de espalhamento de linha, LSF; (d) Função de transferência de modulação, MTF

A figura 14 é o cálculo do MTF para um sistema não ideal. Nesse caso, a curva de MTF indica que existe atenuação das frequências mais altas, com consequente redução dos detalhes da imagem. Uma forma de se indicar um valor quantitativo de MTF é determinar a frequência de corte (frequência na qual o sinal de entrada é atenuado pela metade).

Figura 14 – Sequência de cálculo do MTF para um sistema não ideal: (a) Imagem de entrada; (b) Função de espalhamento de borda, ESF; (c) Função de espalhamento de linha, LSF; (d) Função de transferência de modulação, MTF

2.8.3 Contraste RMS

Como o contraste é uma importante característica de imagens, poder calcular o valor do contraste de uma imagem é bastante interessante para se fazer avaliações de imagens e de algoritmos de processamento de imagem. Uma boa forma para se determinar o contraste é utilizando o conceito de contraste RMS (*root-mean-square*), que é o desvio padrão dos valores de intensidade da imagem [33]. Considerando uma imagem com N *pixels*, tomando a intensidade de cada *pixel* como I_n e a média entre os *pixels* como $\bar{I}$, o contraste RMS, C_{RMS} , é definido na equação 10.

$$C_{RMS} = \sqrt{\frac{\sum_{n=0}^{N-1} (I_n - \bar{I})^2}{N}} \quad (10)$$

Peli acrescenta o conceito de contraste RMS local [34], que é o cálculo do contraste RMS para várias partes menores da imagem, e a média dos valores de contraste local pode ser usada para indicar o contraste total.

2.9 FPGA

O FPGA, *Field-Programmable Gate Array*, é um circuito integrado que, após sua fabricação, pode ser configurado por um desenvolvedor. Geralmente, um *chip* de circuito integrado é fabricado para cumprir determinada tarefa, e é imutável. Os famosos *chips* da série 7400 são exemplos disso, cada um pode realizar a operação lógica para a qual foi fabricado (AND, OR, NOT, etc.), não existe a possibilidade de mudar isso. Microcontroladores são fabricados para seguir uma sequência de instruções, o *software*, porém são limitados a isso, o *hardware* não pode ser refeito. Já os FPGA possuem grandes quantidades de elementos lógicos configuráveis. Cada elemento lógico pode representar uma expressão booleana que é configurada de acordo com o projeto desenvolvido.

As duas maiores fabricantes de FPGA, Altera e Xilinx, possuem diferentes famílias de FPGA, basicamente distribuídas em três grupos: baixo custo; médio desempenho; e alto desempenho. As diferenças são basicamente com relação à quantidade de elementos lógicos, desempenho (frequência máxima de operação), periféricos e preço.

2.9.1 Arquitetura

Um FPGA é composto de, basicamente, uma matriz de elementos lógicos, na qual cada elemento é configurável e as ligações entre os elementos também é configurável. A figura 15 é um esquema simplificado da matriz de elementos lógicos, e nos detalhes estão os elementos lógicos, blocos de IO e malha de interconexão entre elementos lógicos.

Figura 15 – Exemplo simplificado de um FPGA, adaptado de [35]

Além de possuírem muitos elementos lógicos, os FPGAs modernos também possuem blocos de memória, chamada de memória *on-chip*, e blocos de processamento digital de sinal (*Digital Signal Processing*, ou DSP), que basicamente são multiplicadores. O bloco de DSP do FPGA utilizado é apresentado na figura 16.

Figura 16 – Bloco de DSP do FPGA Cyclone IV, adaptado de [36]

Por fim, é possível encontrar FPGAs que possuem outros periféricos, como *transceivers* (para PCIe, por exemplo), controladores de memória DDR3 e, mais recentemente, um processador integrado no mesmo *chip*, compartilhando memória e barramentos de dados com o FPGA.

2.9.2 Paralelismo

Se em um processador a execução das operações é sequencial de acordo com o *software*, para um FPGA é possível projetar um sistema com diversos elementos em paralelo, já que o FPGA funciona como um *hardware* personalizado. É possível projetar desde circuitos combinacionais até uma complexa arquitetura de múltiplos processadores paralelos.

Essa característica é interessante visando a aplicação em processamento de imagens para a câmera térmica, já que são diversos processamentos de imagem que devem ser feitos, e nesses componentes é possível executá-los paralelamente sobre a imagem.

2.9.3 Processamento de imagens em FPGA

Saegusa fez uma comparação entre um sistema em FPGA e processadores para o processamento de imagens [37]. A comparação se dá para três algoritmos: filtros bidimensionais; visão estéreo; e clusterização *k-means*. O trabalho conclui que a capacidade de processamento paralelo, em *hardware*, existente no FPGA é determinante para seu bom desempenho. Assim, o tamanho do FPGA e o tamanho da imagem processada são parâmetros importantes para o desempenho do processamento de imagem.

Os blocos de DSP e de memória de um FPGA também são configuráveis e podem ser acessados paralelamente, sendo assim grandes aliados no desenvolvimento de projetos que realizem processamento de imagens. A memória interna pode ser utilizada para, por exemplo, armazenar o histograma da imagem, permitindo fácil análise posterior desse histograma. Já para um filtro espacial, os blocos DSP são indispensáveis, já que eles realizam as multiplicações de forma rápida, consumindo poucos elementos lógicos.

2.9.4 Linguagens de descrição de *hardware*

Existem alguns meios de criar um projeto em FPGA, sendo os mais relevantes os diagramas de blocos e as linguagens de descrição de *hardware*. Os diagramas de blocos, também chamados de modo esquemático, contém informação visual do circuito que será implementado no FPGA. A figura 17 é o diagrama de blocos para a expressão booleana definida na equação 11, na qual a variável *sel* é utilizada para selecionar entre duas expressões booleanas de entrada: *a E b*; e *c OU d*.

$$y = (a \cdot b) \cdot \overline{sel} + (c + d) \cdot sel \quad (11)$$

Figura 17 – Diagrama de blocos de uma expressão booleana

Já as linguagens de descrição de *hardware* descrevem o circuito de forma escrita. A figura 18 é a mesma expressão booleana descrita de três formas diferentes em VHDL. VHDL significa VHSIC (*Very High Speed Integrated Circuit*) *Hardware Description Language* e é justamente utilizada para desenvolvimento de *hardware*, tanto para FPGAs quanto para circuitos integrados.

```
-- Forma 0
y <= ((a and b) and (not sel)) or ((c or d) and sel);

-- Forma 1
y <= (a and b) when (sel = '0') else (c or d);

-- Forma 2
if (sel = '0') then
    y <= (a and b);
else
    y <= (c or d);
end if;
```

Figura 18 – Três formas de escrever uma mesma expressão booleana em linguagem de descrição de *hardware* VHDL

As linguagens de descrição de *hardware* se mostram mais eficientes no desenvolvimento de módulos com maior complexidade, devido à sua capacidade de descrever o funcionamento do *hardware* (forma 2 da figura 18), não necessariamente o circuito lógico em si (forma 0 da figura 18). Cabe ao compilador traduzir o que foi descrito em *hardware* de fato.

As duas linguagens de descrição mais comuns são VHDL e Verilog. Ambas são utilizadas para gerar código sintetizável.

3 Materiais e Métodos

O estudo e o desenvolvimento dos métodos desse trabalho são parte do desenvolvimento de uma câmera térmica. Nesse capítulo, serão descritos os materiais relevantes utilizados para a construção da câmera e os métodos relacionados com o desenvolvimento dos algoritmos de processamento de imagem.

3.1 Materiais

A figura 19 é o diagrama de blocos da câmera utilizada. Os blocos principais são:

- Detector e conjunto óptico;
- Placa de processamento de vídeo;
- Placa de exibição de vídeo e *display*.

Figura 19 – Divisão em subsistemas do equipamento utilizado e o fluxo de dados dentro do equipamento

A implementação dos algoritmos propostos neste trabalho é feita no FPGA da placa de processamento de vídeo. A placa de exibição de vídeo simplesmente recebe o vídeo processado e envia no formato correto para os dispositivos de exibição.

3.1.1 Detector

O detector utilizado na câmera é um detector refrigerado, de 640×512 *pixels*. É um detector QWIR (*quantum-well*) de telureto de mercúrio e cádmio calibrado para trabalhar na faixa LWIR. O detector já vem em um encapsulamento selado com *cooler* integrado.

Além do detector, uma placa de controle fornecida pelo fabricante é responsável por ligá-lo e desligá-lo de modo seguro. A placa também faz a conversão do sinal de vídeo analógico para um sinal digital de 14 bits, de acordo com a sincronização desejada.

São dois sinais de sincronismo gerados para a placa de controle, o *clock* de *pixel* (PCLK) e o sincronismo de *frame* (FCLK). O sinal PCLK sincroniza a conversão do sinal analógico de cada *pixel*, a cada pulso do sinal, um *pixel* é convertido. Já o sinal FCLK força o detector a iniciar um novo *frame*, ou seja, quando ocorre um pulso nesse sinal, o detector inicia uma nova aquisição da radiação da cena e atualiza o sinal de saída para a nova aquisição.

A placa de controle, baseada nos sinais de sincronismo, tem como saída os sinais de sincronismo, o valor digital da saída do detector e mais um sinal que indica quando o dado digital é válido, ou seja, quando o *pixel* pertence à imagem capturada. Na figura 20 estão apresentadas as formas de onda dos sinais relevantes do detector. Os sinais sufixados com **_in** são os sinais de entrada e os com sufixo **_out**, mais DATAVALID e DATA[13:0], são os sinais de saída da placa de controle.

Figura 20 – Relação entre sinais de sincronismo e sinais enviados pela placa de controle para o FPGA

3.1.2 Placa de processamento de vídeo

A placa de processamento de vídeo é composta de um FPGA Altera Cyclone III de 55 k elementos lógicos, 328 pinos de entrada/saída, memória interna de pouco mais de 290 kbytes e 312 blocos multiplicadores. Para o armazenamento do *frame* durante o processamento, foram escolhidas três memórias RAM assíncronas de 1M x16 bits com latência de 10 ns. A latência é o tempo que, a partir do comando de leitura, o dado de saída está estável para leitura. O armazenamento não volátil é feito em uma memória *flash* de 32 Mbits. Outras duas memórias RAM são utilizadas para fazer o armazenamento dos parâmetros de correção de não uniformidade e de substituição de *bad pixel*.

A comunicação entre a placa de processamento e o computador de calibração e suporte é feita através de uma rede CAN (*controller area network*). O vídeo capturado pode ser enviado diretamente para o computador através de um barramento dedicado. Por fim, o sinal de vídeo é enviado para a placa de exibição por um barramento digital. Os sinais disponíveis nesse barramento são os sincronismos de *pixel* e de *frame* e o vídeo digital.

A figura 21 é o diagrama de blocos da placa de processamento de vídeo, mostrando os relacionamentos entre os componentes da placa. É interessante notar os acessos paralelos às memórias, que são compartilhados pelos algoritmos internos ao FPGA.

Figura 21 – Diagrama de blocos da placa de processamento de vídeo

3.1.3 Placa de exibição de vídeo

Essa placa é a *interface* entre a placa de processamento de vídeo e os dispositivos de exibição de imagem, *display* e sinal de TV, sendo possível selecionar qual o dispositivo de saída, se é só a TV, só o *display* ou ambos. Nessa placa também pode ser selecionado o padrão de cores da imagem exibida, se o que é quente aparece em branco (*white-hot*), o frio aparece branco (*white-cold*) ou escalas de cores pré-definidas (vermelho associado ao quente e azul associado ao frio, por exemplo).

Os *overlays* são adicionados nessa placa. Menus de configuração e carga da bateria são exemplos de *overlays* que podem ser adicionados na imagem.

3.1.4 Equipamento de suporte

Além da câmera, são necessários os equipamentos de suporte ao desenvolvimento. Um computador que consiga receber, de forma digital, imagens capturadas pela câmera é importante no processo de calibração e de testes. Além disso, o computador pode ser utilizado para realizar estímulos na câmera, a fim de se realizar algum teste específico.

Outro equipamento fundamental é o corpo negro, utilizado tanto durante a calibração quanto para testes e medidas de desempenho da câmera.

3.1.5 Softwares

A fase de concepção dos algoritmos de processamento é feita com o auxílio do Matlab. Assim que validados, os algoritmos podem ser implementados no FPGA. O software utilizado no desenvolvimento para FPGAs Altera é o Altera Quartus II. Durante o desenvolvimento dos algoritmos para FPGA, a simulação dos mesmos é feita no ModelSim Altera Edition, que é uma poderosa ferramenta de simulação de *hardware*.

Uma *interface* com controles, esquemas de calibração e testes da câmera está presente em um *software* desenvolvido no LabView.

3.2 Métodos

O método de desenvolvimento utilizado nesse trabalho segue o fluxo apresentado na figura 22. A seguir cada um dos passos será descrito brevemente.

Figura 22 – Fluxo de desenvolvimento dos algoritmos

3.2.1 Requisitos

Os requisitos fundamentais são os algoritmos de pré-processamento [2]. São eles: NUC; BPR; aumento de contraste; e redução da resolução de cor. NUC e BPR são importantes devido ao processo de fabricação dos detectores de infravermelho. O realce de contraste é interessante pois imagens térmicas têm contraste baixo, se comparado com imagens visíveis. Já a redução da resolução de cor deve ser feita pois o detector utilizado é de 14 bits, já o *display* tem 8 bits por *pixel*.

Além disso, julgou-se importante o desenvolvimento de um algoritmo que pudesse realçar detalhes na imagem, já que durante a redução da resolução de cor da imagem alguns detalhes podem ser perdidos.

3.2.2 Viabilidade (Matlab)

A utilização da ferramenta matemática Matlab favorece a avaliação dos algoritmos propostos para a síntese em FPGA. Os algoritmos foram escritos de uma maneira que permite a sua reprodução em VHDL, ou seja, não foram utilizadas bibliotecas específicas do Matlab e as estruturas de dados eram similares às utilizadas no VHDL. Dessa forma, como o algoritmo escrito no Matlab era bastante semelhante ao algoritmo a ser escrito em VHDL, se os resultados da avaliação fossem interessantes, a implementação em FPGA poderia ser iniciada.

3.2.3 Implementação VHDL

A implementação em VHDL para FPGAs Altera foi feita no ambiente de desenvolvimento Altera Quartus II. Esse *software* dá total apoio no desenvolvimento de sistemas para FPGA, desde o editor de texto até ferramentas de manipulação de baixo nível do sistema sintetizado.

Uma compilação completa no Quartus realiza os seguintes passos:

- *Análise e elaboração*: faz a análise da sintaxe do sistema e traduz para um esquemático (*and*, *or*, *mux*, registradores, etc.);
- *Síntese*: traduz o esquemático para os elementos lógicos da tecnologia utilizada (blocos lógicos, blocos de memória, blocos DSP, IO, etc.);
- *Fitting, placement e routing*: posiciona os elementos lógicos no FPGA e faz o roteamento entre os elementos;
- *Assembler*: gera o arquivo de configuração do FPGA;
- *Análise de timing*: verifica todos os atrasos internos do circuito e avalia se o desempenho do sistema cumpre os requisitos de tempo.

3.3 Desenvolvimento dos algoritmos

Nessa seção são descritas as etapas de desenvolvimento do sistema em FPGA da câmera térmica. Apesar de não serem partes integrantes dos algoritmos de processamento de imagens, as *interfaces* FPGA/detector e FPGA/memórias são brevemente descritas, pois elas têm influência direta nos requisitos dos algoritmos de processamento.

3.3.1 Interface com o detector

O módulo de interface com o detector é o responsável por gerar os sinais de sincronismo e por receber os sinais do detector. Para isso basta contar pulsos de *clock* para gerar os sinais de sincronismo (basicamente é um divisor de *clock*). A taxa de *frames* é 30 Hz e a frequência do *clock* de *pixel* é 20 MHz. São três saídas do detector, um *clock* de *pixel*, um barramento de vídeo digital e um sinal que indica quando o dado no barramento de vídeo é válido, que também serve como sincronismo de *frame*.

3.3.2 Interface com as memórias

O sistema conta com três memórias RAM para armazenamento dos *frames* intermediários, chamadas de memórias de *frame*, uma memória que armazena os parâmetros da correção de não uniformidade (NUC) e uma que armazena a tabela de *bad pixels*. A leitura e a escrita de um valor da memória levam no mínimo 10 ns. Além dessas memórias RAM, existe também uma *Flash* para armazenamento não volátil de parâmetros de configuração e de correção da câmera.

Das memórias de *frame*, uma é preenchida com o *frame* do detector corrigido (após a etapa de NUC). O processamento de substituição de *bad pixels* é realizado na segunda memória de *frame*. E na terceira é formado o *frame* final, com a aplicação do filtro, o contraste melhorado e a redução da resolução de cor. É dessa memória que o *frame* é lido e enviado para a placa de exibição.

A figura 23 é o esquema da utilização das memórias. Durante o instante t_0 , enquanto o *frame* n , após o NUC, é escrito na memória 0, o BPR é aplicado na memória 1, sobre o *frame* $n - 1$. E na memória 2, o *frame* $n - 2$ passa pelos algoritmos de contraste e de redução, antes de ser enviado para a placa de exibição.

No instante t_1 o *frame* $n + 1$ é escrito na memória 2, que no instante anterior continha o *frame* a ser enviado para a placa de exibição. O BPR é aplicado no *frame* n (memória 0) e o *frame* $n - 1$, na memória 1, é enviado para o *display*.

Durante o instante t_2 , o *frame* $n + 2$ é escrito na memória 1. O BPR é aplicado no *frame* $n + 1$ na memória 2 e o *frame* n , que está na memória 0, agora corrigido e sem *bad pixels*, é enviado para a placa de exibição.

O próximo *frame* repete a sequência do *frame* n , e o ciclo se fecha. Esse processo é utilizado para que o vídeo possa ser processado e exibido na taxa de 30 Hz. São 640×512 *pixels* para serem processados em 33 ms, resultando em aproximadamente 100 ns por *pixel*. Para os processamentos que são feitos diretamente no sinal de vídeo enviado pelo detector, o tempo é menor, pois como o *clock* de *pixel* é 20 MHz, cada *pixel* dura 50 ns, que é o máximo que o algoritmo de processamento pode gastar.

Figura 23 – Sequência de utilização das memórias para três *frames* consecutivos

3.3.3 Correção de não uniformidade, NUC

A calibração de dois pontos é feita no computador de suporte, determinando assim os valores de *Ganho* ($G(p)$) e de *Offset* ($O(p)$), para cada *pixel* (p), que são utilizados durante a operação da câmera. Os parâmetros da correção são armazenados na memória Flash da câmera, com tamanho de 16 bits para cada parâmetro e, durante a inicialização da câmera, esses valores

são transferidos para a memória RAM de correção, mais rápida que a *Flash*. A organização da RAM de correção é apresentada na figura 24.

Com isso, o procedimento de NUC é feito apenas com uma multiplicação pelo *Ganho* e uma adição do *Offset*, figura 25. O próprio índice do *pixel* funciona como endereçamento das memórias de correção e de armazenamento do *frame*. Para o *pixel* p , o endereço do *Ganho* correspondente é $p \times 2$ e o do *Offset* é $p \times 2 + 1$.

0	Ganho(0)
1	Offset(0)
2	Ganho(1)
3	Offset(1)
...	...
655358	Ganho(640×512-1)
655359	Offset(640×512-1)

Figura 24 – Organização da memória RAM de correção

Figura 25 – Algoritmo de NUC

Para otimizar o desempenho do algoritmo, aproveitando a capacidade de paralelismo do FPGA, quatro processos funcionam paralelamente, formando o *pipeline* apresentado na figura 26. Os quatro processos são: leitura da RAM de correção; multiplicação do *pixel* de entrada pelo seu *Ganho*; soma do valor do *Offset*; e escrita do *pixel* corrigido na memória de *frame*.

Figura 26 – Pipeline do algoritmo de NUC

O tempo de leitura dos parâmetros para cada *pixel* é, no mínimo, de 20 ns, para a memória utilizada. Já os tempos da multiplicação e da soma dependem de como elas são

implementadas. A operação de soma é convertida facilmente para elementos lógicos, portanto um ciclo de *clock* é suficiente (10 ns para um *clock* de 100 MHz), porém a multiplicação não é algo tão simples.

A Altera, fabricante do FPGA utilizado, disponibiliza uma biblioteca de módulos para os desenvolvedores. Essa biblioteca tem as grandes vantagens de já ter sido validada e de ser otimizada para as tecnologias presentes no FPGA. Dentro da biblioteca existe o módulo multiplicador, LPM_MULT. Esse módulo utiliza os blocos de DSP do FPGA de forma automática, basta configurar o número de bits de cada entrada e da saída. Além disso, é possível configurar qual a latência do multiplicador, em pulsos de *clock*. Isso é importante, pois dependendo do tamanho do multiplicador ele pode ficar lento, então adiciona-se alguns estágios de *pipeline*, que aumenta o desempenho em troca do aumento da latência.

A escrita teórica da RAM leva 10 ns. Esse valor aumenta para 20 ns no caso de um *clock* máximo de 100 MHz, devido à sinalização necessária para realizar a escrita. Para atingir o mínimo permitido pela memória seria necessário *clock* de 200 MHz, o que é virtualmente impraticável para o FPGA utilizado. Então, o tempo total que o algoritmo demora por *pixel* é o maior tempo dentre as etapas do *pipeline*, ou seja, entre os 20 ns do acesso à RAM e o tempo de latência configurado no bloco multiplicador utilizado.

3.3.4 Correção de um ponto

Um algoritmo de correção de um ponto é importante para a câmera, devido ao efeito de *temperature drift*. Para isso, o sistema faz a média de alguns *frames*, calcula o *pixel* médio do *frame* médio e então calcula os novos valores de *Offset* para o algoritmo de NUC.

Quando entra em modo de correção, a câmera imediatamente congela a imagem. Assim, das três memórias de *frame*, uma possui a última imagem exibida, que é o *frame* enviado para a placa de exibição, e as outras duas são utilizadas em paralelo, como se fossem uma memória x32, para os cálculos da correção. É necessário utilizar palavras de 32 bits, pois um *pixel* já é de 16 bits, então durante a soma de dois *frames* pode acontecer estouro de variável. Com a memória x32 seria possível fazer a média de até 65536 (2^{16}) *frames*. Quanto mais *frames*, menos ruído aleatório, porém o tempo gasto com a correção aumenta. Para este trabalho foram utilizados 128 *frames* no cálculo dessa média.

A correção é dividida em cinco partes: *frame* inicial; *frames* intermediários; *frame* final; cálculo do *pixel* médio; e atualização do *Offset*. Para as três etapas iniciais, os dados utilizados vêm diretamente do detector, portanto, o tempo do processamento de cada *pixel* deve ser menor que o período do *clock* de *pixel* do detector.

Durante o *frame* inicial, é escrito na memória de *frame* o valor do *pixel* multiplicado pelo seu *Ganho* correspondente, pois é considerado que apenas o valor do *Offset* sofreu alteração, figura 27. O tempo gasto nessa operação é o maior entre a multiplicação e 20 ns, que é o tempo de escrita da memória.

Figura 27 – Correção de um ponto – *pipeline* para o *frame* inicial

Durante os *frames* intermediários o valor do *pixel* multiplicado pelo *Ganho* é somado com o conteúdo atual da memória. O resultado é então escrito na memória, sobrescrevendo o valor anterior. Aqui o tempo da operação é o maior entre a multiplicação e o conjunto leitura/escrita da memória. Com um *clock* de 100 MHz o tempo de processamento total é de 30 ns para leitura/escrita. A sequência é ilustrada figura 28.

Figura 28 – Correção de um ponto - *pipeline* para os *frames* intermediários

Já no *frame* final, o *pixel* multiplicado pelo *Ganho* também é somado com o valor atual da memória, porém dessa vez, o resultado é dividido pelo número de *frames* utilizados para o cálculo da média. É o resultado da divisão que é armazenado na memória. Esse processo é ilustrado na figura 29. Como o tempo utilizado para calcular a média é nulo, pois como a média é feita com 128 *frames*, que é potência de dois, a divisão é somente um *shift* de bits para a direita. Assim, o tempo gasto é igual ao dos *frames* intermediários.

Figura 29 – Correção de um ponto - *pipeline* para o *frame* final

Os próximos passos não precisam ser sincronizados com o detector, visto que só são utilizados dados que estão na memória, assim, a preocupação com desempenho não é fundamental.

Com o *frame* médio armazenado na memória, inicia-se o cálculo do *pixel* médio. Cada *pixel* é lido da memória e somado no acumulador. Ao final do *frame* o valor do acumulador é dividido pela quantidade de *pixels*, 640×512 , ou 327680. De forma a diminuir a divisão, esse valor foi quebrado em 2^{16} e 5, assim basta fazer um *shift* de 16 bits para a direita e dividir por

5. Da mesma forma que o multiplicador o divisor é fornecido pela Altera, e também é possível determinar a quantidade de estágios de *pipeline*. Geralmente o divisor tem mais estágios que o multiplicador, pois ele não utiliza blocos específicos do FPGA, apenas elementos lógicos, o que aumenta sua complexidade.

Com o valor do *pixel* médio determinado, são calculados os novos valores de *Offset*. Para isso, é feita a leitura do valor de cada *pixel* do *frame* médio e é encontrada a diferença entre eles e o *pixel* médio, que é o valor do *Offset*. Um *Offset* positivo acontece quando o *pixel* médio é maior que o *pixel* do *frame* médio. Calculado o parâmetro, ele é escrito na memória de correção, sobrescrevendo o valor anterior. E então a câmera volta a exibir imagens.

3.3.5 Substituição de *pixels* defeituosos, BPR

A substituição dos *pixels* defeituosos também é feita em duas etapas. A primeira é feita em laboratório com auxílio do corpo negro e do computador. Nessa etapa os *bad pixels* são identificados e o *pixel* que irá substituí-lo é determinado. O *pixel* substituto é um vizinho adjacente não defeituoso. Então, os índices do *bad pixel* em conjunto com do seu substituto são armazenados na memória Flash da câmera. Durante a inicialização, o conteúdo da Flash é copiado para a memória de *bad pixels*, para ser utilizado durante a operação. Na figura 30 é possível ver a organização da memória de *bad pixels*, com uma lista contendo n *pixels*.

0	<i>Bad pixel</i> 0 LSB
1	<i>Bad pixel</i> 0 MSB
2	<i>Bad pixel</i> 1 LSB
3	<i>Bad pixel</i> 1 MSB
...	...
$(n-1) \times 2$	<i>Bad pixel</i> $n-1$ LSB
$(n-1) \times 2 + 1$	<i>Bad pixel</i> $n-1$ MSB
$(n-1) \times 2 + 2$	Término de lista

Figura 30 – Conteúdo da memória de *bad pixels*

A segunda etapa da substituição dos *bad pixels* é feita em tempo real durante a operação da câmera. Esse algoritmo não é sincronizado com o *clock* de *pixel* do detector, somente com o sincronismo de *frame*, ou seja, o processamento de um *bad pixel* pode ser mais lento que o *clock* de *pixel*, porém o processamento de toda a lista de *bad pixels* de um *frame* deve estar pronto até que o outro comece, 33 ms depois de iniciado.

Imediatamente no início do *frame* a memória de *bad pixel* é lida. Ao ler uma palavra, é verificado se é um término de lista, se sim, espera o próximo *frame*, se não, lê a próxima palavra. Com as duas palavras, os índices do *bad pixel* e seu substituto são determinados. Então, é feita a leitura do valor do *pixel* substituto e a escrita desse valor na posição do *bad pixel*. Como a quantidade de *bad pixels* é bastante inferior à quantidade total de *pixels* da imagem, não existe muita preocupação com o desempenho do algoritmo, portanto a implementação é uma simples máquina de estados, ilustrada no fluxograma da figura 31.

Figura 31 – Fluxograma da rotina de BPR

Com o *clock* de processamento de 100 MHz são 20 ns para leitura dos índices, 10 ns para leitura do valor do *pixel* substituto e 20 ns para sobrescrita do *bad pixel*. Adicionando um pulso de *clock* entre as etapas, para interpretação das informações, resulta em 80 ns por *bad pixel*. Para o *frame* que dura 33 ms, tem tempo suficiente para realizar a substituição de todos os *pixels* do *frame*.

3.3.6 Histograma, contraste e redução da resolução de cor

O contraste de uma imagem tem relação com a variação das intensidades dos *pixels* que a compõem. Quanto maior a variação, maior o contraste, e isso se traduz no histograma

dessa imagem. Se o histograma possui grande quantidade de *pixels* concentrada em uma pequena faixa, a imagem possui baixo contraste. Portanto, o primeiro passo para melhorar o contraste da imagem é extrair seu histograma. Com o histograma formado é possível aplicar algumas transformações para que o contraste seja melhorado.

Uma imagem inteira, após a correção de não uniformidade (que é uma multiplicação e uma soma), ocupa $640 \times 512 \times 16$ bits em memória, sendo necessário utilizar memória externa para armazená-la. Já o histograma completo teria 2^{16} entradas com valor podendo chegar em até 640×512 , o que resultaria em $65536 \times 640 \times 512$ bits.

São utilizados dois histogramas, enquanto um é formado o outro é analisado. A figura 32 é a representação da sequência de formação e de análise dos histogramas. Para o *frame* N, o histograma H0 é formado e o histograma H1 é analisado, para o próximo *frame*, N+1, o histograma H0 é analisado e o histograma H1 é formado, e o ciclo se repete.

Frame N	Frame N+1	Frame N+2
Forma histograma N (H0)	Forma histograma N+1 (H1)	Forma histograma N+2 (H0)
	Analisa histograma N (H0)	Analisa histograma N+1 (H1)
		Analisa histograma N+2 (H0)

Figura 32 – Sequência para a formação e análise de histogramas

Neste trabalho o histograma foi reduzido para 2^{15} entradas de 19 bits cada (19 bits comportam todos os *pixels* da imagem), permitindo armazenar o total de *pixels* da imagem mas ocupando 622592 bits em memória, cerca de um oitavo do tamanho da imagem. Assim é possível utilizar a memória *on-chip* do FPGA para armazenar os dois histogramas.

Aproveitando que se trata de uma memória, a construção do histograma é feita utilizando o valor do nível do *pixel* como endereço da memória e o conteúdo da memória é o valor acumulado de *pixels* nesse nível. Cada *pixel* tem 16 bits de largura, e o histograma utilizado tem entrada de 15 bits, ou seja, cada dois níveis consecutivos são armazenados na mesma entrada. Isso é feito removendo-se o bit menos significativo do valor de nível do *pixel* e utilizando o valor restante como endereço para o histograma.

A formação do histograma é feita durante a etapa de correção da imagem, NUC, portanto é sincronizada com o detector. Após o *pixel* ser corrigido, além de ser escrito na memória de *frame*, ele é enviado para o processo que monta o histograma. Ao receber o valor do *pixel*, o processo faz a leitura da respectiva entrada do histograma, incrementa uma unidade e sobrescreve com o valor incrementado. Se o valor lido for o valor máximo do histograma, 2^{19} , então não incrementa, mantém o valor máximo para não estourar a variável.

A memória interna é bastante rápida, um pulso de *clock* para leitura e um para escrita, portanto, para o *clock* de 100 MHz, são consumidos 20 ns para atualizar uma entrada do histograma.

Com o histograma formado, na versão citada por Bittencourt [2], o algoritmo faz a varredura do histograma em busca dos níveis inicial e final do histograma. Para determinar esses níveis, o algoritmo utiliza um valor limite predeterminado, o primeiro e o último nível que possuem quantidade de *pixels* acima desse limite são considerados os níveis inicial e final. Esse método não se mostrou robusto, pois alguma componente de ruído no histograma poderia deslocar significativamente o nível inicial ou o final. Dessa forma, neste trabalho são propostas duas novas versões desse algoritmo, também baseadas em níveis encontrados no histograma, com menor sensibilidade à essas condições.

A primeira versão utiliza um método semelhante ao da função '*imadjust()*' do Matlab. Inicialmente é definida qual a porcentagem de *pixels* que será saturada, para baixo e para cima. Então o processo percorre o histograma acumulando os *pixels* dos níveis inferiores, até superar a porcentagem definida, nesse momento ele marca o nível corrente como o nível inicial. O algoritmo continua percorrendo o histograma até encontrar a quantidade de *pixels* suficiente para saturar a porcentagem definida para cima e então marca como nível final.

Para a segunda versão desse algoritmo, além de serem procurados os níveis inicial e final, são procurados um ou mais níveis intermediários, também relacionados com uma porcentagem definida para cada nível. A figura 33 é um exemplo de histograma com três níveis determinados: 5%; 50%; e 95% dos *pixels* da imagem.

Figura 33 – Exemplo de histograma e os níveis de 5%, 50% e 95% do total de *pixels*

Em ambas as versões do algoritmo, busca-se mapear os níveis encontrados no histograma de entrada para níveis da imagem de saída. Para isso, os níveis encontrados durante

a análise do histograma são mapeados para valores predefinidos, e os níveis intermediários são mapeados linearmente, para formar a imagem de saída. Na figura 34 são apresentados exemplos das transformações feitas pelos algoritmos de dois e de três pontos.

Figura 34 – Comparação entre as transformações dos algoritmos de redução de: (a) dois pontos; (b) três pontos

Definindo-se K níveis (L_k) para o realce de contraste, após a análise do histograma, são encontrados K níveis. Então, também devem ser definidos K valores de *targets* (T_k) que representam os valores de saída, para os quais os níveis encontrados no histograma de entrada são mapeados. Os dois grupos, níveis e *targets*, estão definidos na equação 12.

$$\begin{cases} \text{níveis} = [L_0 \ L_1 \ \dots \ L_{K-1}] \\ \text{targets} = [T_0 \ T_1 \ \dots \ T_{K-1}] \end{cases} \quad (12)$$

Com esses dois conjuntos, são definidas equações para a transformação da imagem de entrada para a saída. Considerando $F_t(n)$ como *frame* transformado, $F_o(n)$ como *frame* original, e g_k (*ganho*) e o_k (*offset*) como os parâmetros da transformação k de realce de contraste, obtém-se a equação 13.

$$\begin{cases} F_t(n) = T_0; & F_o(n) < L_0 \\ F_t(n) = F_o(n) \times g_k - o_k; & L_i < F_o(n) < L_{n+1}, 0 \leq n \leq N-1 \\ F_t(n) = T_{n-1}; & F_o(n) \geq L_{K-1} \end{cases} \quad (13)$$

Dessa forma, os *pixels* com nível menor L_0 são forçados para o valor T_0 , os com nível maior que L_{K-1} são levados para o valor T_{K-1} , e os que possuem níveis entre L_k e L_{k+1} obedecem à transformação linear com *ganho* g_k e *offset* o_k .

Para determinar os valores de *ganho* e *offset* das transformações, basta utilizar os valores de *targets* predefinidos e desenvolver a equação 13, resultando na equação 14.

$$\begin{cases} g_k = \frac{T_{k+1} - T_k}{L_{k+1} - L_k} \\ o_k = L_k \times g_k \end{cases} \quad (14)$$

Se, por exemplo, for escolhida uma transformação com apenas dois pontos, saturando 1% para baixo e para cima, mapeados para 0 e 255, mínimo e máximo de uma escala de 8 bits respectivamente, as equações 13 e 14 são desenvolvidas para as equações 15 e 16, nas quais $L_{1\%}$ e $L_{99\%}$ são os níveis inicial e final encontrados, $T_{1\%}$ e $T_{99\%}$ são os *targets* para os esses níveis e g e o são o *ganho* e o *offset* para a transformação dos níveis intermediários. Na equação 16, $T_{1\%}$ e $T_{99\%}$ foram substituídos por 0 e 255, respectivamente.

$$\begin{cases} F_t(n) = T_{1\%}; & F_o(n) < L_{1\%} \\ F_t(n) = F_o(n) \times g - o; & L_{1\%} < F_o(n) < L_{99\%} \\ F_t(n) = T_{99\%}; & F_o(n) \geq L_{99\%} \end{cases} \quad (15)$$

$$\begin{cases} g = \frac{255}{L_{99\%} - L_{1\%}} \\ o = L_{1\%} \times g \end{cases} \quad (16)$$

Para esse algoritmo também são utilizados os módulos multiplicador e divisor da Altera. O tempo necessário para encontrar os pontos inicial e final e calcular os valores de *ganho* e *offset* não pode ser maior que o período de um *frame*, 33 ms. Com o *clock* de 100 MHz, a leitura completa do histograma leva em torno de 0,3 ms ($2^{15} * 10 \text{ ns}$), que é muito inferior ao período do *frame*. O tempo restante é utilizado para realizar os cálculos para encontrar os valores de *ganho* e de *offset* do *frame*.

3.3.7 Filtro de nitidez

Depois da redução da resolução de cor, é possível que pequenos detalhes da imagem sejam perdidos. Por isso, pode ser interessante tratar a imagem com um filtro espacial de realce antes de se fazer a redução da resolução de cor, assim, detalhes realçados pelo filtro não são perdidos após a redução.

Como os filtros de realce também podem aumentar bastante o ruído da imagem, foi desenvolvida uma variação parametrizada do filtro, para que o observador possa ajustar o nível da filtragem conforme a imagem observada.

A derivação do filtro proposto $f(k)$ é demonstrada na equação 17, com k sendo o parâmetro do filtro. Basicamente, um filtro da média foi somado com um filtro passa altas

parametrizado para que, quando o valor de k for 0, o filtro resultante seja exatamente o filtro da média, equação 18. Quando o valor de k for 1, o filtro é desabilitado, pois o resultado da transformação é exatamente o *pixel* central da matriz de vizinhança, equação 19. Já quando o valor de k for maior que 1, o filtro da média é anulado e o resultado é um filtro de realce de detalhes. Quanto maior o valor de k , maior é a amplificação dos detalhes.

$$f(k) = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} + \frac{k}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} = \frac{1}{9} \times \begin{bmatrix} 1-k & 1-k & 1-k \\ 1-k & 8k+1 & 1-k \\ 1-k & 1-k & 1-k \end{bmatrix} \quad (17)$$

$$f(0) = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (18)$$

$$f(1) = \frac{1}{9} \times \begin{bmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (19)$$

A implementação em FPGA desse algoritmo tem complexidade maior que a dos demais, pois se trata principalmente de nove multiplicações e uma divisão. Além disso, para realizar a convolução da imagem pelo filtro proposto, os nove *pixels* que compõem a matriz de vizinhança devem ser lidos sequencialmente da memória, e como a leitura de uma entrada da memória leva no mínimo 10 ns, a leitura dos nove valores requer pelo menos 90 ns. Soma-se a esse tempo mais um ciclo para registrar o valor lido e o tempo total chega a 100 ns. O tempo máximo de processamento por *pixel* é de aproximadamente 100 ns, portanto é interessante diminuir o tempo do processamento, para que sobre alguma margem para futuras modificações, ou até para aumentar a taxa de *frames* da câmera.

Foi observado que, na maior parte das situações durante a convolução, as matrizes de vizinhança para dois *pixels* vizinhos possuem seis valores em comum. É possível ver na figura 35-a as relações entre matrizes de vizinhança em algumas situações. O sinal representado por *neighborhood* é a matriz de vizinhança do *pixel* corrente, o sinal *last_pixel* contém os seis valores que serão utilizados para formar a matriz de vizinhança do próximo *pixel* da linha, e *last_line* contém seis valores da matriz do primeiro *pixel* dessa linha que serão utilizados na matriz de vizinhança do primeiro *pixel* da próxima linha. Dessa forma, exceto para o primeiro *pixel*, passa a ser necessário realizar apenas três leituras da memória de *frame* para compor a matriz de vizinhança do *pixel*, reduzindo em 60 ns o tempo do processamento. Na figura 35-b é apresentada a disposição dos *pixels* da matriz de vizinhança.

Figura 35 – Formação da matriz de vizinhança para o filtro de nitidez: (a) Relação entre matrizes de vizinhança de *pixels* adjacentes; (b) Disposição dos *pixels* da matriz de vizinhança

Para o primeiro *pixel* da imagem, todos os nove *pixels* são lidos da memória e armazenados no sinal *neighborhood*. Ao mesmo tempo, são formadas as matrizes *last_pixel* e *last_line*, armazenando somente os seis *pixels* comuns entre as matrizes de vizinhança. O fluxo dos dados para a formação das matrizes do primeiro *pixel* da imagem é apresentado na figura 36.

Figura 36 – Pipeline para formação da matriz de vizinhança para o primeiro *pixel* da imagem

Para os demais *pixels* da linha, para qualquer linha da imagem, o fluxo muda. Inicialmente, a matriz *last_pixel* é copiada na matriz *neighborhood*, e então são feitas as leituras

dos três *pixels* que faltam na matriz *neighborhood*, que são os da direita (M3, M6 e M9). A figura 37 é a representação desse fluxo de dados, é importante notar que, paralelamente à formação da matriz *neighborhood*, a matriz *last_pixel* é atualizada para o próximo *pixel* da imagem.

Figura 37 – Pipeline para formação da matriz de vizinhança para os *pixels* a partir do segundo *pixel* da linha

Para o primeiro *pixel* das demais linhas, exceto a primeira, é utilizada a matriz *last_line* para formar a matriz *neighborhood*. Inicialmente, a matriz *last_line* é copiada para a *neighborhood*. Então os três *pixels* que faltam, M7, M8 e M9, são lidos da memória de *frame* para completar a matriz *neighborhood*. Ao mesmo tempo que a matriz *neighborhood* é formada, ambas as matrizes *last_pixel* e *last_line* são atualizadas para o próximo *pixel* e para a próxima linha, respectivamente. Na figura 38 é possível ver como são atualizadas essas três matrizes.

Figura 38 – Pipeline para formação da matriz de vizinhança para o primeiro *pixel* de cada linha, exceto a primeira

Com a matriz de vizinhança formada, o *pixel* filtrado, p_f , é determinado multiplicando cada elemento da matriz de vizinhança, $neighborhood(n)$, por seu correspondente no filtro

proposto, $filtro(n)$, somando todos os resultados das multiplicações e dividindo pelo número de *pixels* da matriz, como mostrado na equação 20.

$$p_f = \frac{\sum_0^8 neighborhood(n) \times filtro(n)}{9} \quad (20)$$

As operações de multiplicação são feitas paralelamente e precisam de três ciclos de *clock*, totalizando 30 ns, para completarem. A soma de todos os valores leva cinco ciclos, 50 ns. Já o divisor precisa de nove ciclos para terminar o cálculo, 90 ns. Como essas operações são feitas em *pipeline*, elas não diminuem o *throughput* do sistema, apenas aumentam a latência do processamento.

4 Resultados

Para determinar os resultados dos algoritmos foram utilizadas imagens geradas no Matlab e imagens reais. Inicialmente, foram utilizadas as imagens simuladas para verificar o funcionamento dos algoritmos em simulação. Depois que os algoritmos foram verificados e implementados, foram capturadas as imagens reais, processadas na câmera.

4.1 Correção de não uniformidade, NUC

Para verificar o algoritmo de NUC, a não uniformidade do detector foi simulada em Matlab, gerando aleatoriamente as curvas de responsividade de cada *pixel* do detector. Então, com dois *frames* de calibração (*Hot* e *Cold*) foram calculados os parâmetros da correção. As imagens, e as curvas de alguns *pixels*, antes e depois da correção são mostradas na figura 39. É possível notar que as curvas após a correção estão muito mais agrupadas, apesar de algumas curvas não estarem perfeitas. Isso ocorre pois a responsividade desses *pixels* não é linear em toda faixa de calibração (temperaturas entre *Hot* e *Cold*).

Figura 39 – Imagens e curvas de responsividade simuladas: (a) Antes da correção; (b) Depois da correção de não uniformidade de dois pontos

Também foi simulado o efeito de *temperature drift* do detector, quando cada *pixel* recebe uma quantidade uniforme de radiação a mais ou a menos devido à mudança de temperatura de operação da câmera. A figura 40 é uma comparação entre as curvas de responsividade de alguns *pixels* logo após o *temperature drift* e após a correção de um ponto.

Figura 40 – Imagens e curvas de responsividade simuladas: (a) após o *temperature drift*; (b) após a correção de não uniformidade de um ponto

Resultado do algoritmo já implementado em FPGA, a figura 41 é uma comparação entre uma imagem real sem e outra com correção. É fácil ver que na imagem sem correção existe uma grande componente de ruído vertical. Esse ruído não é temporal, portanto se trata de ruído por não uniformidade do detector, e por isso a correção é importante.

Figura 41 – Comparação entre imagens: (a) sem NUC; (b) com NUC

4.2 Substituição de *pixels* defeituosos, BPR

O algoritmo de BPR não foi implementado em Matlab. Como é um algoritmo bastante simples, ele foi implementado diretamente em FPGA, e seu bom desempenho depende fundamentalmente da lista de *bad pixels* fornecida. Essa lista de *bad pixels* geralmente é fornecida pelo fabricante do detector. A figura 42 é uma comparação entre uma imagem sem BPR e outra com. É possível ver nos detalhes alguns *bad pixels* que são substituídos durante o processamento.

Figura 42 – Comparação entre imagens: (a) sem BPR, com *bad pixels* em destaque; (b) com BPR

4.3 Contraste e redução da resolução de cor

O algoritmo de realce de contraste foi avaliado variando a quantidade de pontos utilizados no algoritmo, de dois até dezesseis, e utilizando uma imagem do corpo negro. A avaliação foi feita calculando-se o contraste RMS local das imagens geradas e, além da quantidade de pontos, variou-se a quantidade de *pixels* saturados pelo algoritmo em 5%, 1% e 0,1%. A figura 43 é um gráfico que reúne os valores de contraste para cada variação desse algoritmo. No gráfico é possível ver que o contraste é maior quando se aumenta o número de pontos do algoritmo, porém ele possui o comportamento assintótico e, a partir de oito pontos, varia pouco.

Na figura 44 são apresentadas as imagens e histogramas resultantes do algoritmo de realce de contraste com saturações de 5%, 1% e 0,1% para dois, três e oito pontos cada. Os números de pontos foram escolhidos para ilustrar a diferença entre o algoritmo com dois, três e com oito pontos, a partir do qual o contraste começa a se estabilizar. É possível ver que, quando se aumenta o número de pontos, o histograma fica melhor distribuído por toda a faixa,

tendo um efeito semelhante ao da equalização de histograma. Também é fácil notar que para saturações de 1% e 0,1%, o algoritmo com dois pontos não se comporta bem.

Figura 43 – Contraste por número de pontos do algoritmo de realce de contraste, para cada percentual de saturação de *pixels* avaliado

Figura 44 – Imagens e histogramas para o algoritmo de realce de contraste com dois, três e oito pontos para saturação de 5% dos *pixels* (a), (b) e (c), para saturação de 1% dos *pixels* (d), (e) e (f) e para saturação de 0,1% (g), (h) e (i), respectivamente

Após isso, a porcentagem de saturação foi fixada em 5% e foram processadas diferentes imagens com dois, três e oito pontos. As imagens e histogramas resultantes são apresentados na figura 45 e na tabela 2 são apresentados os respectivos valores de contraste RMS. Os aumentos de contraste detectados entre os algoritmos de três e de dois pontos e os algoritmos de oito e de dois pontos também são apresentados na tabela.

Figura 45 – Imagens e histogramas após realce de contraste com saturação de 5% dos *pixels* com dois, três e oito pontos. (a), (b) e (c) para cena 1, (d), (e) e (f) para cena 2 e (g), (h) e (i) para cena 3, respectivamente

Tabela 2 – Comparação entre o contraste RMS local, equação 10, para o algoritmo de realce de contraste com dois, três e oito pontos para três diferentes cenas

	Contraste 2 pontos	Contraste 3 pontos	Aumento 2-3	Contraste 8 pontos	Aumento 2-8
Cena 1	0,0759	0,0885	16,5%	0,1197	57,7%
Cena 2	0,0956	0,1159	21,3%	0,1438	50,4%
Cena 3	0,0664	0,0953	43,5%	0,1194	79,8%

Na figura 46 é apresentada a diferença entre os algoritmos com dois e com três pontos para cenas reais, observadas e processadas pela câmera. Como as imagens foram processadas na própria câmera, elas não são idênticas, pois durante o tempo que a imagem era transferida da câmera para o computador podem ter ocorrido pequenas mudanças na cena observada, por efeito do vento e da movimentação do sol principalmente. Por esse motivo, as comparações de contraste nesses casos não são determinantes, já que a comparação é feita entre imagens ligeiramente diferentes.

Figura 46 – Cenas reais com algoritmo de dois pontos: (a), (b) e (c). E suas respectivas cenas com algoritmo de três pontos: (d), (e) e (f)

Na tabela 3 são apresentados os valores calculados de contraste RMS local para ambos os algoritmos e a relação entre eles, para as imagens da figura 46. Nas imagens que possuem áreas com temperaturas altas, ou maior faixa de temperaturas, é possível perceber a melhora no contraste quando é utilizado o algoritmo de três pontos (figura 46-a e figura 46-b), fato observado também pelo aumento de 26% e 36%, respectivamente, no valor do contraste RMS das imagens.

Tabela 3 – Comparação entre o contraste RMS local, equação 10, para o algoritmo de realce de contraste com dois e três pontos para diferentes imagens processadas pela câmera

	2 pontos	3 pontos	Aumento
Cena 1	0,0849	0,1073	26,4%
Cena 2	0,0717	0,0977	36,3%
Cena 3	0,0910	0,0962	5,7%

A porcentagem de saturação utilizada na implementação foi a de 5%, dessa forma ambos os algoritmos puderam ser implementados juntamente na câmera, pois como visto anteriormente, na figura 44, o algoritmo de dois pontos não se comporta bem com saturações menores. Já o algoritmo de três pontos foi utilizado pois, pelas análises anteriores, para saturação de 5% ele apresentou resultados considerados interessantes (melhora no contraste), com menor complexidade que os algoritmos com mais pontos, dessa maneira o algoritmo de oito pontos não foi implementado em FPGA.

4.4 Filtro de nitidez

Para validar a ideia do filtro de nitidez, foi criada uma imagem sintética com uma borda não ideal e o filtro proposto foi aplicado sobre a imagem. Então, foi feito o cálculo do MTF para ambas as imagens, original e filtrada. A figura 47 é o comparativo entre os resultados obtidos, e nela é possível ver que com o filtro existe uma melhora no MTF da imagem, já que a frequência de corte aumentou.

Figura 47 – Validação do filtro por MTF: (a) Imagem de entrada; (b) MTF com e sem a aplicação do filtro de nitidez sobre a imagem

Um segundo método de avaliação utiliza uma imagem digital sintética com quatro quadrados pretos nos cantos e um branco no centro. Dentro dos quadrados existem linhas com cores ligeiramente diferentes da cor do quadrado. O restante da imagem é de uma cor intermediária, cinza. A figura 48 é a comparação entre quatro níveis diferentes de filtragem ($k = 1$, $k = 6$, $k = 11$ e $k = 15$). Percebe-se que quanto maior o nível de filtragem mais as linhas internas dos quadrados ficam visíveis, resultando em um aumento na capacidade de se observar os detalhes da imagem. O algoritmo de realce de contraste e de redução de resolução de cor utilizado foi o de três pontos com saturação de 5% para cima e para baixo.

Figura 48 – Avaliação subjetiva do filtro de nitidez com uma imagem sintetizada e diferentes níveis de filtragem:
(a) $k = 1$; (b) $k = 6$; (c) $k = 11$; (d) $k = 15$

Posteriormente, foi feita a avaliação do filtro com imagens reais da câmera, utilizando o corpo negro configurado em diferentes temperaturas. Após a filtragem, com o parâmetro sendo variado de um até vinte, foram realizados o realce de contraste e a redução da resolução de cor. Com as imagens resultantes foram calculados o MTF do sistema.

Na figura 49 são mostradas as imagens processadas e as curvas de MTF para dois casos com o corpo negro configurado em 10 °C. O primeiro caso é com o filtro desativado ($k = 1$) e o segundo é o melhor caso encontrado ($k = 13$). A figura 50-a é a curva dos valores de MTF (frequência de corte) por k , e a figura 50-b são as curvas de MTF para alguns valores do parâmetro ($k = 1, 4, 7, 10, 13, 16$ e 19).

Figura 49 – Imagem e curva de MTF com corpo negro em 10 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 13$

Figura 50 – Avaliação do filtro por MTF com corpo negro em 10 °C: (a) Curva do valor do MTF por valor de k ; (b) Curvas de MTF para alguns valores de k

Com esses resultados é possível ver que com o aumento do valor do parâmetro podem ser inseridos artefatos na imagem, nas áreas de baixa frequência espacial, porém o MTF do sistema também aumenta (aumento na frequência de corte do sistema), característica importante para a detecção de objetos na imagem. As figuras 51 e 52 são as mesmas análises para o corpo negro configurado em 80 °C.

Figura 51 – Imagem e curva de MTF com corpo negro em 80 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 16$

Figura 52 – Avaliação do filtro por MTF com corpo negro em 10 °C: (a) Curva do valor do MTF por valor de k ; (b) Curvas de MTF para alguns valores de k

Além das duas condições apresentadas, a mesma verificação foi feita para outras seis temperaturas, presentes no apêndice A. Na tabela 4 estão as relações, para cada condição, entre o melhor MTF encontrado variando-se o parâmetro k e o MTF quando o filtro não é aplicado. Pela tabela é possível ver que em todos os casos houve melhora no MTF, sendo que o maior aumento, de 104%, ocorreu no maior delta de temperatura, com o corpo negro configurado em 80 °C, e o menor aumento, de 61%, ocorreu quando o corpo negro foi configurado em 30 °C, o que era muito próximo da temperatura ambiente de 28 °C, fazendo com que a borda não tivesse muita qualidade.

Tabela 4 – Relações entre o melhor MTF encontrado e o MTF sem o filtro de nitidez

Temperatura da chapa	MTF sem filtro (MTF ₀)	Melhor MTF (MTF _M)	Relação MTF _M /MTF ₀
10 °C	6,6	12,3; $k=13$	1,86
20 °C	7,2	12,4; $k=21$	1,72
30 °C	7,6	12,2; $k=10$	1,61
40 °C	7,1	13,7; $k=15$	1,93
50 °C	7,3	13,6; $k=11$	1,86
60 °C	6,8	13,6; $k=15$	2,00
70 °C	6,8	13,6; $k=15$	2,00
80 °C	6,7	13,7; $k=16$	2,04

Com o algoritmo implementado em FPGA, foi feito um comparativo entre diferentes níveis de filtragem para duas cenas diferentes, três deles são apresentados na figura 53 ($k = 1$, $k = 7$ e $k = 15$), os demais são apresentados no apêndice B. A primeira cena é um conjunto de telhados, postes e fios e a segunda é a traseira de um carro. Com o aumento do nível de

filtragem, é possível ver alguns detalhes a mais nas imagens, como os números da placa do carro na segunda cena, detalhe apresentado na figura 54.

Figura 53 – Duas cenas reais processadas na câmera com diferentes níveis de filtragem: (a) e (d) $k = 1$; (b) e (e) $k = 7$; (c) e (f) $k = 15$

Figura 54 – Detalhe de uma placa de carro processada na câmera com duas variações do filtro de nitidez: (a) $k = 1$; (b) $k = 15$

A figura 55 é uma cena de um caminhão a longa distância, também processada no FPGA. Quando o filtro é configurado para não realçar detalhes (figura 55-a, $k = 1$) é difícil reconhecer o caminhão na cena, porém com o filtro ativado (figura 55-b, $k = 10$) pode-se notar o formato do caminhão com mais facilidade.

Figura 55 – Imagem de um caminhão a longa distância processada na câmera com duas variações do filtro de nitidez:
(a) $k = 1$; (b) $k = 10$

5 Conclusões

Os algoritmos básicos, NUC e BPR [2] [25], foram implementados em FPGA e possuem desempenho melhor que o necessário para o sistema proposto, em termos de tempo de processamento. Para o NUC, o processamento de cada *pixel* leva 20 ns, valor menor que a duração de cada *pixel* de entrada, em torno de 50 ns. Já o BPR, que não precisa ser aplicado diretamente no sinal de entrada, tem a capacidade de processar cada *pixel* defeituoso em 80 ns, portanto, considerando 5% de *pixels* defeituosos na imagem, 16384 *pixels*, o algoritmo demora cerca de 1,3 ms, tempo inferior ao do *frame* completo, 33 ms

A variação do algoritmo de realce de contraste e de redução da resolução de cor se mostrou mais robusta que a versão que utiliza apenas dois pontos do histograma de entrada. Algoritmos com mais de dois pontos aproveitam melhor a faixa dinâmica da imagem de saída, melhorando o contraste das imagens apresentadas ao observador. Para alguns casos foram observados aumentos maiores que 50% no valor do contraste RMS local em comparação com o algoritmo de dois pontos. Para a implementação em FPGA do algoritmo com três pontos também foi observada melhora no contraste, em comparação com o algoritmo com dois pontos.

A implementação do algoritmo em FPGA foi dividida em três etapas: formação do histograma da imagem de entrada; análise do histograma; e transformação dos *pixels* de entrada. O histograma de entrada é formado ao mesmo tempo que a imagem é recebida do detector. Durante o próximo *frame*, este histograma é analisado para se determinar os níveis do algoritmo de realce de contraste e se calcular os parâmetros da transformação dos *pixels*. Já a transformação dos *pixels* consiste basicamente de uma multiplicação e uma soma. Dessa forma, todo o algoritmo de realce de contraste tem uma latência de três *frames*, sendo compatível com o sistema de três memórias utilizado, que permite essa latência.

Os resultados do filtro espacial mostram que o algoritmo é ser interessante para a aplicação já que, através de avaliações objetivas, foi confirmado que o filtro proposto melhora a qualidade da imagem em termos de reprodução de detalhes, podendo aumentar o MTF em até 100% para alguns casos. O custo disso é que, com o aumento do valor do parâmetro do filtro (k), a relação sinal-ruído diminui e, em alguns casos são adicionados artefatos na imagem, principalmente bordas, que não existem na imagem original. Dessa forma, utilizar o filtro com o parâmetro permite ao observador selecionar a melhor configuração.

Para a implementação do filtro em *hardware* foi necessário aproveitar as características do FPGA, relacionadas a paralelismo, e elaborar um *pipeline*, que satisfizesse as restrições de tempo do sistema, responsável por realizar a convolução do filtro proposto sobre a imagem. Com essa implementação, o algoritmo requer 100 ns para o primeiro *pixel* da imagem e 40 ns para os demais *pixels*, totalizando aproximadamente 13,1 ms para um *frame* completo.

Ambos os algoritmos propostos e implementados nesse trabalho aprimoram a qualidade da imagem, em comparação à imagem gerada pelos algoritmos apresentados no trabalho de Bittencourt [2].

6 Propostas para trabalhos futuros

Para imagens de entrada que possuam faixa dinâmica muito grande, pode ser interessante realizar o algoritmo de realce de contraste e redução da resolução de cor em subdivisões da imagem, localmente. Assim, o resultado é otimizado para aquela parte apenas, não sofrendo influência do restante da imagem. Além da complexidade, um desafio dessa abordagem é nas intersecções das subdivisões processadas, pois cada subdivisão terá seu próprio conjunto de transformações, provavelmente diferente da subdivisão adjacente, o que tornaria a intersecção visível na imagem de saída, degradando-a.

Outra variação do algoritmo de realce de contraste e redução da resolução de cor pode ser feita tornando a definição dos níveis e *targets* do algoritmo adaptativa. Isso pode ser interessante, pois para diferentes cenas é possível que diferentes definições funcionem melhor. Assim, se for verificado que isso ocorre, é interessante desenvolver um método que, com as características extraídas da imagem de entrada, determine o melhor conjunto de parâmetros do algoritmo.

O mesmo pode ser feito para o filtro de nitidez, com o desenvolvimento de um método adaptativo para se determinar o valor ótimo do parâmetro k do filtro. Além de características da imagem de entrada, esse método também pode levar em consideração o contraste da imagem da saída, e tentar maximizá-lo.

Referências

- [1] NGUYEN, et al. **Comparison of 3 infrared thermal detection systems and self-report for mass fever screening**. Emerg Infect Dis, 2010. doi: 10.3201/eid1611.100703
- [2] BITTENCOURT, T. M. G. **Pré processamento digital de imagens obtidas na faixa espectral do infravermelho distante (dissertação de mestrado)**. São Carlos: EESC-USP, 2012.
- [3] BRIOSCHI, M. L. **A História da Termografia**. São Carlos: IFSC-USP, 2008.
- [4] NEWMAN. **Infrared Imaging Thermography**, 13 Legal Aspects of Med. Prac, 1985.
- [5] CRAWFORD, D. **The King's Astronomer, William Herschel**. New York: J. Messner, 1968.
- [6] FORBES J. D. **A Review of the Progress of Mathematical and Physical Science between the Years of 1775 and 1850**, Edinburgh: Adam and Black, 1858. Disponível em: <http://todayinsci.com/M/Melloni_Macedonio/MelloniMacedonio-RadiantHeat.htm> Acesso em 06/11/2015. doi: 10.5962/bhl.title.23065
- [7] BUTTMANN, G. **The Shadow of the Telescope**. New York: Charles Scribner's Sons, 1970. doi: 10.1063/1.3022884.
- [8] WALCOTT, C. D. **Biographical Memoir of Samuel Pierpont Langley**. Washington: National Academy of Sciences, p. 247-268, 1912.
- [9] ROGALSKI, A. **History of infrared detectors**. Opto-Electronics Review, v. 20, n. 3, p 279-308, September 2012. doi: 10.2478/s11772-012-0037-7

- [10] ARORA N. et al. Effectiveness of a noninvasive digital infrared thermal imaging system in the detection of breast cancer: *Am. J. Surg.* v. 196, n. 4, p. 523-526, 2008. doi: 10.1016/S1043-321X(09)79326-9
- [11] DIAKIDES, N. A.; BRONZINO, J. D. **Medical Infrared Imaging**. Boca Raton, FL: CRC Press. 2007. doi: 10.1201/9781420008340
- [12] HILDEBRANDT, C.; RASCHNER, C.; AMMER, K. **An Overview of Recent Application of Medical Infrared Thermography in Sports Medicine in Austria**. *Sensors* 10, n. 5, p. 4700-4715, 2010. doi: 10.3390/s100504700.
- [13] SITKO M. L. **Reflectance Spectroscopy - Introduction to the ALTA Spectrometer**. Disponível em <<http://homepages.uc.edu/~sitkoml/ReflectanceSpectroscopy.pdf>> Acesso em 06/11/2015.
- [14] VILLATE, J. E. **Física 2 - Eletricidade e Magnetismo**. Faculdade de Engenharia - Universidade do Porto – Portugal. Disponível em <<http://www.villate.org/doc/fisica2/>>. Acesso em 06/11/2015.
- [15] D'AMICO A. et al. **Volatile Compounds Detection by IR Acousto-Optic Detectors**. Unexploded Ordnance Detection and Mitigation, NATO Science for Peace and Security Series B: Physics and Biophysics, p. 21-59, 2009. doi: 10.1007/978-1-4020-9253-4_2.
- [16] GLOBUS, A. **Towards an Early Profitable PowerSat**. Space Manufacturing 14: Critical Technologies for Space Settlement. NASA Ames Research Center: Space Studies Institute, 2010.
- [17] DELTAT, **Infrared Basics - Infrared Energy, Emissivity, Reflection & Transmission**. Disponível em: <<http://www.deltat.com/pdf/Infrared%20Energy,%20Emissivity,%20Reflection%20%26%20Transmission.pdf>>. Acesso em 06/11/2015.

- [18] SHARMA, U. **Infra Red Detectors**. M. Tech Credit Seminar Report, Electronic Systems Group, EE Dept, IIT Bombay, submitted Oct 2004.
- [19] ROGALSKI, A. **Infrared detectors, status and trends**. Progress in Quantum Electronics, v. 27, issue 2-3, p. 59-210, 2003. doi: 10.1016/S0079-6727(02)00024-1
- [20] ULIS. **Infrared Detectors**. Disponível em: < <http://www.ulis-ir.com/index.php?infrared-detector=products>>. Acesso em 06/11/2015.
- [21] SOFRADIR. **How night vision works**. Disponível em: <<http://www.hownightvisionworks.com/>>. Acesso em 06/11/2015.
- [22] KOKORINA, V. F., **Glasses for infrared optics**, New York: CRC Press, 1996.
- [23] THORLABS. **Germanium Windows**. Disponível em: <http://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=3980>. Acesso em 26/02/2014.
- [24] MILTON, A. F.; BARONE F. B.; KRUEER M. R.; **Influence of nonuniformity on infrared focal plane array performance**. Optical Engineering, v. 24, n. 5, p. 855-862, 1985. doi: 10.1117/12.7973588.
- [25] HOLST, G. C. **Testing and Evaluation of Infrared Imaging Systems**. Florida: JCD; Bellingham: SPIE, 3rd ed., 2008.
- [26] FLIR. **Digital Detail Enhancement (DDE)**. Technical Note.
- [27] KIM, S., **Two-point correction and minimum filter-based nonuniformity correction for scan-based aerial infrared cameras**. *Opt. Eng.* v. 51, n. 10, 106401, 2012. doi: 10.1117/1.OE.51.10.106401.

- [28] TENDERO Y.; GILLES J. **ADMIRE: a locally adaptive single-image, non-uniformity correction and denoising algorithm: application to uncooled IR camera**. Proc. SPIE 8353, Infrared Technology and Applications XXXVIII, 83531O (May 1, 2012); doi:10.1117/12.912966.
- [29] TORRES S. N. et al. **Adaptive Scene-Based Nonuniformity Correction Method for Infrared-Focal Plane Arrays**. Proc. SPIE 5076, Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XIV, p.130-139, 2003. doi: 10.1117/12.487217.
- [30] MARQUES FILHO, O.; VIEIRA NETO, H. **Processamento Digital de Imagens**. 1. ed. Rio de Janeiro: Brasport, 200p, 1999.
- [31] GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 976p, 2002.
- [32] LISANI, J. L.; PETRO, A. B.; SBERT, C. **Color and Contrast Enhancement by Controlled Piecewise Affine Histogram Equalization**. Image Processing Online, p. 243-265, 2012. doi: 10.5201/ipol.2012.lps-pae.
- [33] PELI, E. **Contrast in Complex Images**, Journal of the Optical Society of America A, v. 7, n. 10, p. 2032–2040, 1990. doi: 10.1364/JOSAA.7.002032.
- [34] BEX P.J, MAKOUS, W. **Spatial frequency, phase, and the contrast of natural images**. Journal of the Optical Society of America A, v. 19, n. 6, p. 1096-1106, 2002. doi: 10.1364/JOSAA.19.001096.
- [35] NATIONAL INSTRUMENTS. **Controle baseado em FPGA: Milhões de transistores ao seu comando (FAQ)**. Disponível em: <http://www.ni.com/white-paper/3357/pt/>. Acesso em 12/02/2014.

[36] ALTERA, **Embedded Multipliers in Cyclone IV Devices**. Disponível em https://www.altera.com/en_US/pdfs/literature/hb/cyclone-iv/cyiv-51004.pdf. Acesso em 06/11/2015.

[37] SAEGUSA, T.; MARUYAMA, T.; YAMAGUCHI, Y. **How fast is an FPGA in image processing?** International Conference on Field Programmable Logic and Applications, p. 77 - 82, 2008. doi: 10.1109/FPL.2008.4629911.

Apêndices

Apêndice A – Cálculo do MTF com e sem filtro de nitidez

Para realizar a avaliação do filtro de nitidez o corpo negro foi configurado em diferentes temperaturas, então os MTF das imagens sem e com filtro foram calculados e comparados. Na seção de resultados do filtro foram apresentados os casos do corpo negro em 10 °C e em 80 °C, figuras 49 e 51 respectivamente. A seguir são apresentados os demais casos utilizados na avaliação do filtro: 20 °C na figura 56; 30 °C na figura 57; 40 °C na figura 58; 50 °C na figura 59; 60 °C na figura 60; 70 °C na figura 61.

Figura 56 – Imagem e curva de MTF com corpo negro em 20 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 20$

Figura 57 – Imagem e curva de MTF com corpo negro em 30 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 10$

Figura 58 – Imagem e curva de MTF com corpo negro em 40 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 15$

Figura 59 – Imagem e curva de MTF com corpo negro em 50 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 11$

Figura 60 – Imagem e curva de MTF com corpo negro em 60 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 15$

Figura 61 – Imagem e curva de MTF com corpo negro em 70 °C para dois casos: (a) sem filtro, $k = 1$; (b) melhor MTF, após filtro de nitidez com $k = 15$

Apêndice B – Filtro de nitidez para imagens processadas na câmera

Foram processadas na câmera duas cenas reais, com o parâmetro de filtragem, k , variando entre 1 e 15. Na figura 62 são apresentados os resultados para $k = 2$, $k = 3$, $k = 4$, $k = 5$, $k = 6$, $k = 8$, $k = 9$, $k = 10$, $k = 11$, $k = 12$, $k = 13$ e $k = 14$.

a)

b)

c)

d)

e)

f)

g)

h)

i)

j)

k)

l)

m)

n)

o)

p)

q)

r)

s)

t)

u)

v)

Figura 62 – Imagens processadas na câmera com filtro configurado com $k = 2$, (a) e (b), $k = 3$, (c) e (d), $k = 4$, (e) e (f), $k = 5$, (g) e (h), $k = 6$, (i) e (j), $k = 8$, (k) e (l), $k = 9$, (m) e (n), $k = 10$, (o) e (p), $k = 11$, (q) e (r), $k = 12$, (s) e (t), $k = 13$, (u) e (v) e $k = 14$, (w) e (x)