**UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS**

Leandro Rosendo Candido

**Construção de mosaico de imagens aéreas em
plataformas heterogêneas para aplicações agrícolas**

São Carlos

2019

Leandro Rosendo Candido

**Construção de mosaico de imagens aéreas em
plataformas heterogêneas para aplicações agrícolas**

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, para obtenção do título de Mestre em Ciências - Programa de Pós-Graduação em Engenharia Elétrica.

Área de concentração: Processamento de Sinais e Instrumentação

Orientador: Prof. Dr. Maximilian Luppe

São Carlos

2019

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

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues
Fontes da EESC/USP

C217c	<p>Candido, Leandro Rosendo</p> <p>Construção de mosaico de imagens aéreas em plataformas heterogêneas para aplicações agrícolas / Leandro Rosendo Candido; orientador Maximilian Luppe. -- São Carlos, 2019.</p> <p>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, 2019.</p> <p>1. Mosaico de imagens aéreas. 2. Computação heterogênea. 3. Agricultura de precisão. 4. Georreferenciamento direto. 5. Triangulação. 6. Bundle adjustment. 7. CUDA. 8. NVIDIA Jetson TK1. 9. IMU. 10. GPS. I. Título.</p>
-------	--

FOLHA DE JULGAMENTO

Candidato: Engenheiro **LEANDRO ROSENDO CANDIDO**.

Título da dissertação: "Construção de mosaico de imagens aéreas em plataformas heterogêneas para aplicações agrícolas".

Data da defesa: 29/03/2019.

Comissão Julgadora:

Resultado:

Prof. Dr. **Maximilian Luppe**

(Orientador)

(Escola de Engenharia de São Carlos/EESC)

Aprovado

Dr. **Lucio André de Castro Jorge**

(Empresa Brasileira de Pesquisa Agropecuária/EMBRAPA)

Aprovado

Prof. Dr. **Thiago Wellington Joazeiro de Almeida**

(Alliage S/A Equipamentos Médicos Odontológicos)

Aprovado

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica:

Prof. Associado **Ivan Nunes da Silva**

Presidente da Comissão de Pós-Graduação:

Prof. Titular **Murilo Araujo Romero**

*Dedico este trabalho primeiramente a Deus por ser essencial em minha vida,
autor de meu destino, meu guia e meu socorro na hora da angústia.*

*Ao meu pai José Aparecido Candido,
minha mãe Regina Tereza Rosendo Candido,
aos meus irmãos, Marcelo e Guilherme Rosendo Candido e
à minha amada, Verônica Soares dos Reis.*

AGRADECIMENTOS

Agradeço primeiramente à Deus que em sua infinita sabedoria colocou força em meu coração para vencer esta etapa de minha vida. A fé no Senhor, sem dúvidas, me ajudou nesta luta.

Agradeço minha mãe Regina Tereza Rosendo Candido, que encheu meu coração de amor e esperança. Também sou grato ao meu pai José Aparecido Candido, que me proporcionou a tranquilidade e o conforto que tanto precisava para vencer esta etapa. Meus irmãos Marcelo Rosendo Candido e Guilherme Rosendo Candido, por acreditarem e apoiarem meu sonho.

Verônica Soares dos Reis, meu amor, obrigado por ser tão atenciosa e por entender minha ausência em diferentes momentos.

Sou grato a todos os professores que contribuíram com a minha trajetória acadêmica, especialmente ao Prof. Dr. Maximilian Luppe, responsável pela orientação do meu projeto. Obrigado por esclarecer tantas dúvidas e ser tão atencioso e paciente.

Aos meus mentores Dra. Galina Borissevitch e Dr. Iouri Borissevitch, por exigirem de mim muito mais do que eu imaginava ser capaz de fazer. Manifesto aqui minha gratidão eterna por compartilhar a sabedoria, o tempo e a experiência que vocês têm.

E por fim, agradeço a grande fraternidade chamada "República Gato Preto", que me acolheram e dividiram experiências únicas no tempo em que estive em São Carlos.

Sem a força de vocês eu não conseguiria seguir em frente. Deus os abençoe, sempre.

"Há três caminhos para o fracasso:

Não ensinar o que sabe.

Não praticar o que ensina.

Não perguntar o que ignora "

São Beda, o Venerável

RESUMO

CANDIDO, L. R. **Construção de mosaico de imagens aéreas em plataformas heterogêneas para aplicações agrícolas.** 2019. 83p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

A agricultura de precisão tem agregado alto valor para os agricultores por causa das tecnologias que estão ligadas a ela. Sistemas que extraem informações de imagens digitais são extremamente utilizados para que o agricultor tome decisões a fim de aumentar sua produtividade. Uma das técnicas de realizar o monitoramento é a construção de um mosaico de imagens aéreas, onde são utilizadas aeronaves voando em baixa altitude. Esta técnica pode levar dezenas de horas para ser concluída, dependendo da configuração do computador que a executa. Com o intuito de reduzir o tempo nessa construção e tornar possível o embarque a essa aplicação, este trabalho apresenta uma maneira simplificada de construir o mosaico de imagens aéreas baseada na técnica de georreferenciamento direto, no qual utiliza a computação heterogênea para acelerar o desempenho. Essa abordagem é composta por apenas três técnicas que também compõem a abordagem clássica para a construção de mosaicos (*warping*, extração de características e combinação de características), além de inserir em seus cálculos os dados fornecidos pelos sensores GPS e IMU com a finalidade de direcionar e posicionar cada imagem pertencente ao conjunto que formará o mosaico. A plataforma de computação heterogênea utilizada neste trabalho é a NVIDIA Jetson TK1 escolhida pelo fato de disponibilizar de uma GPU que suporta a linguagem de programação CUDA. Utilizando esta abordagem, a falta de correção da perspectiva do conteúdo (geometria) da imagem gera um resultado inesperado, pois os dados fornecidos pela IMU, ao contrário do que se imagina, apenas servem para corrigir a posição das coordenadas do GPS registradas no momento de captura de cada imagem que compõem o mosaico. O tempo de execução da aplicação desenvolvida é satisfatório tornando possível a adoção desta abordagem.

Palavras-chave: Mosaico de imagens aéreas. Computação heterogênea. Agricultura de precisão. Georreferenciamento direto. Triangulação. Bundle adjustment. CUDA. NVIDIA Jetson TK1. IMU. GPS.

ABSTRACT

CANDIDO, L. R. **Construction of aerial imagery mosaic on platforms for agricultural applications**. 2019. 83p. Dissertation (Master's) - School of Engineering of São Carlos, University of São Paulo, São Carlos, 2019.

Accuracy agriculture has added value to farmers thanks to the new technologies that are linked to it. Systems that extract information from digital images are very useful to help farmers making decisions in order to increase their productivity. One of the techniques to perform this kind of monitoring is the construction of an aerial imagery mosaic where aircrafts flies in low altitude. This technique may take hours to be completed, depending on computer's configuration. With the purpose of reducing time in this construction, this thesis presents a simplified way to make aerial imagery mosaic based on direct georeferencing. This approach is composed by three techniques that also make up the classic approach to building mosaics (warping, extraction of characteristics and combination of characteristics), the difference is with this technique here presented is also possible to insert into the calculations the data provided by the GPS and IMU sensors with the purpose of directing and positioning each image to the belonging set to form the mosaic. The heterogeneous computing platform used in this work is the NVIDIA JetsonTK1, this platform was chosen because it offers a GPU that supports the language of CUDA programming. If the images' geometry errors weren't rectfied, using this approach, an unexpected result happens, because the data provided by IMU, contrary to what is imagined, only serve to correct the position of the GPS coordinates recorded at the moment of capture of each image that composes the mosaic. The developing time in this application is satisfactory making the adoption of this approach favorable.

Keywords: Aerial image mosaic. Heterogeneous computing. Agriculture of precision. Direct georeferencing. Triangulation. Bundle adjustment. CUDA. NVIDIA. Jetson TK1. IMU. GPS.

LISTA DE ILUSTRAÇÕES

Figura 1 – Mapa das áreas onde atuam as 100 empresas que mais estão mudando a vida no campo.	24
Figura 2 – Veículos utilizados para registrar imagens aéreas.	25
Figura 3 – Exemplo de variação espacial de uma produção agrícola. Tons vermelho e amarelo podem significar falta de nutrientes para aquela área da cultura, e tons esverdeados podem significar que a cultura está crescendo saudavelmente nessa área.	25
Figura 4 – Algoritmo de Harris. a) região plana; b) região de borda; c) canto. . . .	30
Figura 5 – Algoritmo SIFT. a) detecção extrema de escala e espaço; b) localização dos pontos chaves; c) atribuição das orientações; d) descritores dos pontos chave; e) vetor com dimensão de 128 elementos do descritor de ponto chave.	31
Figura 6 – Imagens utilizadas para gerar a Tabela 1.	33
Figura 7 – Combinações entre as características de duas imagens que possuem áreas de sobreposição.	34
Figura 8 – Aplicação do algoritmo RANSAC, onde os pontos chave indesejados que não pertencem a zona de sobreposição são removidos.	35
Figura 9 – Aplicando um filtro na imagem altera a intensidade dos pixels.	35
Figura 10 – Aplicando o (<i>warping</i>), o domínio da imagem é alterado.	36
Figura 11 – Efeito da deformação (<i>warping</i>) aplicado em duas imagens.	36
Figura 12 – Imagens separadas.	37
Figura 13 – Resultado da junção sem aplicar o <i>blending</i>	37
Figura 14 – Resultado da junção ao aplicar o <i>blending</i>	38
Figura 15 – Ilustração dos parâmetros intrínsecos.	39
Figura 16 – As retas L e R deveriam se cruzar perfeitamente no espaço, mas não é o que acontece.	42
Figura 17 – "Ilustração do resultado da aplicação do BA".	44
Figura 18 – Diferença entre a arquitetura de um CPU e de uma GPU	47
Figura 19 – "Execução de um programa CUDA"	48
Figura 20 – Arquitetura de um FPGA	51
Figura 21 – Aquisição e Processamento das Imagens Separados	53
Figura 22 – Inforce 6540 SBC	55
Figura 23 – Diagrama de Blocos da IFC6540	56
Figura 24 – Kit de Desenvolvedor NVIDIA Jetson TK1	56
Figura 25 – Parallella Board	57
Figura 26 – Diagrama de conectividade do Xilinx Zynq Dual-core ARM A9	58

Figura 27 – Raspberry Pi 3 Model B Motherboard	58
Figura 28 – Diagrama de bloco do processo para construção de um mosaico de imagens.	62
Figura 29 – Diagrama de blocos para a construção do MIA clássico, sendo que os seis graus de liberdade (ângulos de rotação fornecidos pelo sensor IMU e, a translação fornecidos pelo GPS) são conhecidos.	63
Figura 30 – Diagrama de blocos da abordagem adotada nesse trabalho.	64
Figura 31 – Resultado do MIA gerado pela aplicação desenvolvida neste trabalho utilizando o conjunto de imagens 1.	66
Figura 32 – Resultado do MIA gerado pelo "Agisoft Photoscan" utilizando o conjunto de imagens 1.	66
Figura 33 – Resultado do MIA gerado pela aplicação desenvolvida neste trabalho utilizando o conjunto de imagens 2.	67
Figura 34 – Resultado do MIA gerado pelo "Agisoft Photoscan" utilizando o conjunto de imagens 2.	67
Figura 35 – Resultado da aplicação da técnica <i>warping</i> e área que deverá permanecer (destacada pelo retângulo vermelho) após aplicação da técnica <i>Crop</i> . . .	68
Figura 36 – Imagem resultante após aplicação das técnicas <i>warping</i> e <i>crop</i>	69
Figura 37 – Comparação entre as posições de pontos chave combinados. O ponto branco nas imagens é a posição exata dos pontos chave que se combinam. .	70
Figura 38 – Mesmo após a correção das distorções causadas pelo voo, ainda há distorção no conteúdo das imagens.	71
Figura 39 – Combinações dos pontos chaves onde é possível identificar que o encaixe acontecerá na posição correspondente.	71
Figura 40 – Combinações dos pontos chaves onde é possível identificar que o encaixe não acontecerá na posição correspondente.	72
Figura 41 – Análise de como o voo foi realizado pelo VANT para identificar a origem de construção do MIA.	73
Figura 42 – Construção do MIA 1 utilizando o algoritmo desenvolvido neste trabalho: a Figura 42a foi construído pela placa Jetson TK1; a Figura 42b foi construída pelo notebook	74
Figura 43 – Construção do MIA 2 utilizando o algoritmo desenvolvido neste trabalho: a Figura 43a foi construído pela placa Jetson TK1; a Figura 43b foi construída pelo notebook	74

LISTA DE TABELAS

Tabela 1 – Análise de desempenho de detectores de características utilizando as imagens acima	33
Tabela 2 – Tabela com as principais características dos dois tipos de VANTs utilizados neste trabalho.	54
Tabela 3 – Tabela com as informações do conjunto de imagens.	54
Tabela 4 – Especificações técnicas do SoC Tegra K1	57
Tabela 5 – Tabela de comparação das plataformas apresentadas no Capítulo 3. . .	60
Tabela 6 – Tabela de comparação do tempo médio levado por cada imagem em cada processo apresentado na Figura 30.	76

LISTA DE ABREVIATURAS E SIGLAS

AP	Agricultura de Precisão
ARM	<i>Advanced RISC Machine</i>
ASIC	<i>Application Specific Integrated Circuits</i>
BA	<i>Bundle Adjustment</i>
BSD	<i>Berkeley Software Distribution</i>
CAD	<i>Computer Aided Design</i>
CH	Computação Heterogênea
CPU	<i>Central Processing Unit</i>
CUDA	<i>Compute Unified Device Architecture</i>
DEM	<i>Digital Elevation Models</i>
DSP	<i>Digital Signal Processor</i>
FPGA	<i>Field Programmable Gate Array</i>
GPGPU	Unidade de Processamento Gráfico de Propósito Geral
GPL	<i>General Public License</i>
GPS	<i>Global Positioning System</i>
GPU	<i>Graphics Processing Unit</i>
GSD	<i>Ground Sample Distance</i>
HDL	Linguagem de Descrição de Hardware
IEEE	<i>Institute of Electrical and Electronic Engineers</i>
IMU	<i>Inertial Measurement Unit</i>
IoT	<i>Internet of Things</i>
LUT	<i>Look Up Table</i>
MIA	Mosaico de Imagens Aéreas
OpenCL	<i>Open Computing Language</i>

OpenCV	<i>Open Computer Vision</i>
ORB	<i>Oriented FAST and Rotated BRIEF</i>
RANSAC	<i>Random Sample Consensus</i>
RISC	<i>Reduced Instruction Set Computer</i>
SBA	<i>Sparse Bundle Adjustment</i>
SBC	<i>Single Board Computer</i>
SFM	<i>Structure From Motion</i>
SIFT	<i>Scale Invariant Feature Transform</i>
SIMD	<i>Single Instruction Multiple Data</i>
SoC	<i>System on Chip</i>
SRAM	<i>Static Random Access Memory</i>
SURF	<i>Speed-up Robust Feature</i>
VANT	Veículo Aéreo Não Tripulado
UC	Unidade Computacional

SUMÁRIO

1	INTRODUÇÃO	23
1.1	Objetivo	27
1.2	Considerações Finais	27
2	CONSIDERAÇÕES TEÓRICAS	29
2.1	Introdução	29
2.2	Extração de Características	29
2.2.1	Detector de canto de Harris	29
2.2.2	<i>Scale Invariant Feature Transform</i>	30
2.2.3	<i>Speed-up Robust Feature</i>	31
2.2.4	<i>Oriented FAST and Rotated BRIEF</i>	32
2.3	Combinação de Características	34
2.4	<i>Random Sample Consensus</i> e Homografia	34
2.5	<i>Warping</i>	35
2.6	<i>Blending</i>	36
2.7	Georreferenciamento Direto (FILHO, 2017)	38
2.8	Modelo de Câmera Estenopeica (Câmera Pinhole)	38
2.8.1	Parâmetros Intrínsecos	39
2.8.2	Parâmetros Extrínsecos	39
2.8.2.1	Rotação	40
2.8.2.2	Translação	40
2.8.2.3	Matriz Extrínseca	41
2.8.3	Matriz de Projeção da Câmera	41
2.9	Triangulação	41
2.9.1	Método do ponto médio	42
2.10	<i>Bundle Adjustment</i>	43
2.11	Computação Heterogênea	44
2.11.1	ARM	45
2.11.1.1	Características da arquitetura	45
2.11.1.2	Vector Floating-Point (VFP) e NEON	46
2.11.2	GPU	47
2.11.2.1	Principais linguagens de programação para GPUs	48
2.11.2.2	CUDA	48
2.11.2.3	OpenCL	49
2.11.3	FPGA	49
2.12	Considerações Finais	51

3	MATERIAIS E MÉTODOS	53
3.1	Aquisição e Processamento das Imagens	53
3.2	O Sistema de Aquisição e o Conjunto de Imagens	53
3.3	Plataformas com Sistema Heterogêneo Embarcado	55
3.3.1	Inforce 6540 SBC (IFC6540)	55
3.3.2	NVIDIA Jetson TK1	56
3.3.3	Parallella Board	57
3.3.4	Raspberry PI	58
3.4	Considerações finais	59
4	ABORDAGEM PARA CONSTRUÇÃO DO MIA	61
4.1	Abordagem para execução deste trabalho	63
5	RESULTADOS E DISCUSSÕES	65
5.1	Resultados e comparações	65
5.2	Entendendo os processos <i>Warping</i> e <i>Crop</i>	68
5.3	Análise dos pontos chave combinados	69
5.4	Matriz intrínseca	72
5.5	O OpenCV	73
5.6	Tempo de processamento	75
6	CONCLUSÃO	77
	REFERÊNCIAS	79

1 INTRODUÇÃO

Quem, da geração atual, nunca ouviu dos pais "*Agora, você só fica no celular!*"?

Muitos exclamavam, ou ainda exclamam está questão, porque não conhecem o fascínio tecnológico que um celular possui embarcado. Em 1969 a sociedade ficou impressionada com a chegada do homem a lua, porém o que não é sabido pela maioria, é que este feito foi realizado com uma tecnologia muito inferior se comparado com a tecnologia disponível dentro de um aparelho celular que você carrega dentro do bolso da calça.

A cada dia que passa novas tecnologias surgem na sociedade e novos adeptos a ela também. Não importa a idade, sexo ou qualquer outro tipo de discriminação, todo mundo de alguma maneira utiliza algum tipo de tecnologia nos dias atuais, seja rede social (para interagir com as pessoas), *smartphone* (para comunicar e entreter), utensílios de cozinha (para melhorar e padronizar os sabores) e outras diversas aplicações que facilitam a vida das pessoas. Os novos adeptos afetam diretamente a economia do país comprando ou vendendo novas soluções tecnológicas.

Na agricultura não é diferente. Novos adeptos surgem com o propósito de encontrar soluções tecnológicas que: facilitem a vida das pessoas que trabalham no campo; aumente o lucro gerado na safra e; melhore a qualidade dos produtos oferecidos. Na economia brasileira, é indiscutível que a agricultura possui um grande impacto no mercado mundial de grãos, carnes e outros produtos derivados dos variados cultivos feitos pelo agricultor local.

Com o valor reconhecido internacionalmente, a agricultura brasileira ainda demanda por tecnologia para que seja possível controlar as diversas variáveis como clima, fertilidade do solo, pragas, qualidade das plantações, qualidade dos pastos para o gado e etc. Além dessa demanda, o país também sofre com a falta de investimentos para o desenvolvimento tecnológico nesta área, [Zafalon, Mauro \(2018\)](#). Entretanto, grandes empresas de desenvolvimento tecnológico decidiram investir no país para aumentar a produção agrícola e principalmente aumentar o lucro. Para obter lucro na atividade agrícola, é necessário comprar sabiamente insumos, ter conhecimento técnico para garantir uma boa produção pelo tempo estabelecido para o plantio e, realizar os controles já citados. É exatamente nesse ponto que as empresas de tecnologia estão entrando. Para ter como exemplo de empresas que estão investindo na agricultura nacional, em 2016 a Qualcomm e a Microsoft anunciaram novos investimentos nesta área, [Revista PEGN \(2016\)](#) e [Exame \(2016\)](#).

As grandes empresas de tecnologia podem investir altos valores no agronegócio, porém não significa que *startups* (empresas que estão em sua fase inicial ou que possuem um custo existencial muito baixo) não possuam espaço para crescer nesta área. Muitas estão

entrando nesse mercado com muita força e descobrindo, cada uma com foco e experiência diferente, diversas linhas de desenvolvimento tecnológico para aplicar no campo. Um levantamento de 2017, feito pela CBInsights e traduzido por [BeefPoint \(2017\)](#), apontou quais são as 100 *startups* que mais estão mudando a vida das pessoas que trabalham no campo (vide Figura 1).

Figura 1 – Mapa das áreas onde atuam as 100 empresas que mais estão mudando a vida no campo.

Fonte: [BeefPoint \(2017\)](#)

O trabalho apresentado aqui pertence a área de Agricultura de Precisão (AP), onde é possível utilizar aviões ou Veículos Aéreos Não Tripulados (VANTs)(vide Figura 43) para pulverizar fertilizantes, mas principalmente trabalhar como se fosse os olhos do agricultor sobre sua lavoura. Mas o que significa AP?

Essa definição é claramente apresentada por [Brasil \(2011, p. 5\)](#) como sendo:

[...] um sistema de gerenciamento agrícola baseado na variação espacial de propriedades do solo e das plantas encontradas nas lavouras e visa à otimização do lucro, sustentabilidade e proteção

Figura 2 – Veículos utilizados para registrar imagens aéreas.

Fonte: [Globo Rural \(2016\)](#) e [Senar \(2015\)](#)

do ambiente. Trata-se de um conjunto de tecnologias aplicadas para permitir um sistema de gerenciamento que considere a variabilidade espacial da produção.

Essas variações espaciais do solo podem ser captadas através de sensores de refletância (fotografias), umidade, capacitivos e etc. A Figura 3 exemplifica um mapa de variabilidade da vegetação obtido por um sensor de refletância, onde é possível analisar a variação espacial da produção em uma determinada área. Nela é possível identificar tons vermelho e amarelo, que podem significar falta de nutrientes para aquela área da cultura, e tons esverdeados, que podem significar que nessas áreas a cultura está crescendo saudavelmente.

Figura 3 – Exemplo de variação espacial de uma produção agrícola. Tons vermelho e amarelo podem significar falta de nutrientes para aquela área da cultura, e tons esverdeados podem significar que a cultura está crescendo saudavelmente nessa área.

Fonte: [AGRICULTURA... \(2017\)](#)

Com o processamento adequado das imagens é possível extrair dados que informam sobre ataque de pragas, falta de nutrientes para que a cultura cresça saudavelmente, falha na plantação, estresse hídrico e outros. Com base nos dados extraídos das imagens aéreas, o agricultor toma decisões visando aumentar a qualidade de seu produto, sua produção e seu lucro.

Para obter precisão nos dados extraídos das imagens é necessário realizar voos baixos, implicando em uma área menor a ser capturada pela câmera acoplada ao veículo aéreo. Ao realizar voos mais baixos é necessário registrar várias imagens para capturar a área total.

Após obter o conjunto de imagens que representa uma área, é necessário juntar essas imagens, de forma artística (como se fosse um quebra-cabeça), para visualizar a área como um todo, dando a impressão de que todo o conjunto foi registrado em uma única captura. A técnica que combina várias imagens aéreas com intuito de criar uma nova imagem que cobre uma determinada área é chamada de Mosaico de Imagens Aéreas (MIA). "Usando mosaicos é possível organizar inspeções de campo e fornecer um diagnóstico preciso da área de cultivo" (CANDIDO; LUPPE; JORGE, 2016, p. 22).

O MIA é de extrema importância para a AP, porém não é fácil construir um. Geralmente há um custo elevado de horas (depende do tamanho do conjunto de imagens e/ou das configurações do computador que realiza o processamento) para serem finalizados devido as técnicas de processamento digital de imagens (Extração e Combinação de Características) e de visão computacional (*Bundle Adjustment*). As técnicas citadas exigem um esforço computacional muito alto e, com isso, o impacto reflete no tempo para a construção do MIA. Com a introdução da Computação Heterogênea (CH) o grande esforço computacional para construir um MIA pode ser reduzido. Obter uma resposta rápida reflete no tempo de diagnóstico e ações efetivas a serem tomadas pelo agricultor.

O termo CH é muito utilizado para expressar o uso de diversas arquiteturas que estão ligadas ao mesmo nó computacional com a finalidade de obter um melhor desempenho nas aplicações que executarão sobre esse mesmo nó. Dentre os diversos dispositivos heterogêneos os que mais se destacam são as *Graphics Processing Units* (GPUs) e os *Field-programmable Gate Arrays* (FPGAs) (STRINGHINI; GONÇALVES; GOLDMAN, 2013).

O fato da CH ser encontrada em sistemas embarcados facilita acoplar essa tecnologia nos VANTs com a finalidade de começar a processar os dados contidos nas imagens antes que o veículo toque o solo.

Recentemente, os avanços significativos na redução do tempo no processo de construção do MIA acontece pelo fato dos processos tratarem as informações georreferenciadas que são inseridas nas imagens no momento que elas foram registradas. O conhecimento sobre as posições fornecidas pelos sensores de navegação é usado para resolver transforma-

ções que ocorreram nas imagens, assim reduzindo o custo computacional para a construção do mosaico, dizem [Moussa e El-Sheimy \(2016\)](#).

1.1 Objetivo

Este trabalho tem como objetivo apresentar uma abordagem mais simples para construção do MIA baseada na técnica de georreferenciamento direto com auxílio de uma plataforma heterogênea para realizar os cálculos de forma paralela, trazendo velocidade na tomada de decisão do agricultor. A aplicação gerada nesse trabalho juntamente com a plataforma heterogênea podem ser acopladas a um VANT e assim ter de imediato o início do processamento dos dados extraídos das imagens capturadas, afim de construir o MIA em um menor tempo.

Este trabalho está organizado da seguinte forma: Capítulo 2 apresenta a revisão literária que envolve a construção do MIA, as principais unidades computacionais e as linguagens de programação que compoem a CH. O Capítulo 3 aborda as principais plataformas heterogêneas disponíveis no mercado atual. A abordagem adotada para este trabalho é apresentada no Capítulo 4. Os resultados estão registrados no Capítulo 5 e a conclusão do trabalho desenvolvido está escrito no Capítulo 6.

1.2 Considerações Finais

A AP ainda desperta receio nos agricultores atuais (principalmente nos mais antigos), talvez por acreditarem que aplicar os conceitos envolvidos tenha um custo elevado ou pelo simples fato de não saberem qual é o seu significado e as vantagens que ela vem agregando ao longo dos anos.

É impossível parar o avanço da AP, pois as universidades estão trazendo resultados com bases científicas e o mercado esta aberto para todos os produtores. Os equipamentos envolvidos tem como principal função auxiliar o agricultor a identificar, analisar e atuar nos problemas encontrados em suas culturas ([INAMASU; BERNARDI, 2014](#)).

2 CONSIDERAÇÕES TEÓRICAS

2.1 Introdução

Os processos que envolvem a construção do Mosaico de Imagens Aéreas (MIA), seja ele da área de processamento digital de imagens ou de visão computacional, são triviais e exigem um alto esforço computacional para obter um resultado que atenda o que o agricultor busca visualizar em sua plantação.

Com base nestes fatos, este capítulo descreve os processos envolvidos na construção do MIA e os principais avanços que os pesquisadores têm conseguido nessa área, além de apresentar o embasamento teórico sobre Computação Heterogênea (CH).

2.2 Extração de Características

Adel, Elmogy e Elbakry (2014, 2015), Brown e Lowe (2007) e Joshi e Sinha (2013) relatam que a extração de características é o primeiro passo para a construção de um mosaico de imagens.

Segundo Adel, Elmogy e Elbakry (2014) este conceito refere-se aos métodos que têm como objetivo calcular e abstrair as informações de uma imagem e tomar uma decisão para todos os pontos desta imagem. As características resultantes serão um subconjunto do domínio da imagem, muitas vezes sob a forma de pontos isolados, curvas contínuas ou regiões ligadas.

Os autores também apresentam um comparativo entre as diferentes técnicas de extração de características: detector de canto de Harris, *Scale Invariant Feature Transform*, *Speed-up Robust Feature* e *Oriented FAST and Rotated BRIEF* são as quatro técnicas descritas que possuem uma maior notoriedade e são muito utilizadas e descritas em vários artigos que utilizam a extração de características. Essas quatro técnicas estão descritas nas subseções a seguir.

2.2.1 Detector de canto de Harris

É uma abordagem clássica para detecção de cantos em uma imagem que foi proposta por Harris e Stephens (1988) baseado no algoritmo de Moravec.

Para encontrar os cantos na imagem de entrada, o detector de Harris olha para a média da intensidade que é direcional. A intensidade muda em uma pequena área específica ao redor de um ponto de interesse. Essa área recebe o nome de janela. Ao deslocar a janela, se for uma região plana (Figura 4a), a janela indicará que não há mudança na intensidade em nenhuma direção. Se uma região de borda é encontrada (Figura 4b), não

haverá mudanças ao longo da direção da borda. Se um canto é encontrado (Figura 4c), haverá uma mudança significativa na intensidade em todas as direções. O algoritmo de Harris fornece uma abordagem matemática para determinar se a região é plana, borda ou canto. A equação é dada por:

$$c = \det(M) - k(\text{trace}(M))^2 \quad (2.1)$$

onde M representa a matriz de captura da intensidade da vizinhança local, k é uma constante que varia entre 0,04 e 0,15 e o determinante da matriz é dado por $\det(M) = ab$ e $\text{trace}(M) = (a + b)$, onde a e b são os autovalores da matriz.

- Se os autovalores possuem valores pequenos, a intensidade da vizinhança é constante, então estamos em uma região sem cantos ou arestas;
- Se um autovalor possui um valor grande e o outro pequeno, existe uma variação de intensidade em uma direção. Isso indica uma aresta;
- Se ambos autovalores possuem valores grandes, temos variações em várias direções. Sendo assim, isso indica um possível canto na imagem;

Figura 4 – Algoritmo de Harris. a) região plana; b) região de borda; c) canto.

Fonte: Harris... (2017)

2.2.2 Scale Invariant Feature Transform

Lowe (2004) propôs um algoritmo chamado de *Scale Invariant Feature Transform* (SIFT). Esse algoritmo possui como principais características a invariância a iluminação, rotação, translação e escala da imagem. Ele é composto de quatro fases principais: detecção extrema de escala e espaço (Figura 5a), localização dos pontos chave (Figura 5b), atribuição das orientações (Figura 5c) e definição dos descritores dos pontos chave (Figura 5d).

Figura 5 – Algoritmo SIFT. a) detecção extrema de escala e espaço; b) localização dos pontos chaves; c) atribuição das orientações; d) descritores dos pontos chave; e) vetor com dimensão de 128 elementos do descritor de ponto chave.

Fonte: [Dong e Soatto \(2015\)](#)

O primeiro passo é identificar a localização e a escala dos pontos chaves usando a função que calcula a Diferença de Gaussianas (DoG)

$$DoG(x, y, \sigma) = [G(x, y, k\sigma) - G(x, y, \sigma)] * I(x, y) \quad (2.2)$$

onde: $I(x, y)$ é a imagem e G é a gaussiana dado pela fórmula:

$$G(x, y, k\sigma) = \frac{1}{2\pi k\sigma^2} e^{-(x^2+y^2)/2k\sigma^2} \quad (2.3)$$

onde: x e y são as coordenadas do pixel na imagem, σ é o desvio padrão e k é a constante de multiplicação (começando em 1).

No passo de localização dos pontos chaves, os candidatos a pontos chave são localizados e refinados, eliminando os candidatos onde eles rejeitam os pontos de baixo contraste. No passo de atribuição das orientações, a orientação do ponto chave é obtida baseada no gradiente local da imagem. No último passo, descrição dos pontos chaves, o SIFT calcula o descritor local da imagem para cada ponto chave baseado na magnitude do gradiente e orientação de cada ponto de amostra da imagem centrado no ponto chave. SIFT fornece um vetor com dimensão de 128 elementos do descritor de ponto chave (Figura 5e).

2.2.3 Speed-up Robust Feature

O algoritmo de extração de características chamado *Speed-up Robust Feature* (SURF) foi proposto por [Bay, Tuytelaars e Gool \(2006\)](#). Segundo [Oyallon e Rabin \(2015\)](#) esse algoritmo de extração de características é composto por 3 passos:

- **Análise multiescala:** segue a mesma linha de execução do algoritmo SIFT, porém ao invés de realizar uma convolução entre a imagem original e uma janela gaussiana, é feito a convolução entre a imagem original e uma janela de filtros em várias escalas resultando no que [Oyallon e Rabin \(2015\)](#) chamaram de "caixa discreta".
- **Detecção das características:** o candidato a ponto chave é a máxima local da caixa discreta, onde ele é o resultado da determinante da matriz Hessiana (equação 2.4). Esse candidato é confirmado como ponto chave se ele estiver acima de um valor limite (em inglês *threshold*).

$$H(f(a)) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(a) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(a) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1}(a) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(a) & \frac{\partial^2 f}{\partial x_2^2}(a) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_2}(a) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(a) & \frac{\partial^2 f}{\partial x_2 \partial x_n}(a) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(a) \end{pmatrix} \quad (2.4)$$

- **Descrição das características:** resulta em uma descrição dos pixels vizinhos ao ponto chave e que são invariantes as mudanças de translação, rotação e etc.

2.2.4 *Oriented FAST and Rotated BRIEF*

Oriented FAST and Rotated BRIEF (ORB) foi proposto por [Rublee et al. \(2011\)](#). O algoritmo é um descritor binário muito rápido baseado no detector FAST e no descritor de pontos chaves Binary Robust Independent Elementary Features (BRIEF). Este é um descritor de características que realiza um teste binário simples entre o pixel e uma parte da imagem suavizada. Seu desempenho é similar ao SIFT, incluindo a robustez à mudança de iluminação, borramento e distorção da imagem.

As características baseadas em extratores binários possuem várias vantagens sobre os extratores baseados em vetores (ex.: SIFT e SURF). Eles podem ser mais rápidos para calcular, mais compactos para armazenar, mais eficientes para comparar, e requerem pouca memória. Pode ser demonstrado por meio de experiências como o ORB é em duas ordens de grandeza mais rápido do que o SIFT durante a execução, bem como em muitas situações. A eficiência é testada em várias aplicações do mundo real, incluindo a detecção de objetos e rastreamento de trajeto utilizando um smartphone.

Em seus experimentos [Adel, Elmogy e Elbakry \(2014\)](#) apresentam uma tabela (1), que mostra a quantidade de características detectadas e a velocidade das técnicas de extração de características apresentadas acima.

Figura 6 – Imagens utilizadas para gerar a Tabela 1.

Fonte: Adel, Elmogy e Elbakry (2014)

Tabela 1 – Análise de desempenho de detectores de características utilizando as imagens acima

Detector	Carac. Detectadas		Tempo (ms)	
	Imagem 1	Imagem 2	Imagem 1	Imagem 2
Harris	374	452	0,04	0,06
SIFT	1279	1538	1,48	1,79
SURF	573	628	0,58	0,67
ORB	94	94	0,14	0,14

Fonte: Adel, Elmogy e Elbakry (2014).

SIFT apresentou um maior número de características extraídas para ambas imagens, porém o custo de tempo foi muito superior aos demais algoritmos. O mais rápido foi o detector de cantos de Harris ficando na terceira posição quanto a quantidade de características detectada. O algoritmo ORB foi o que detectou o menor número de características a um bom tempo e o algoritmo SURF serve como uma melhor opção caso SIFT seja dispensado por conta do tempo.

Fazendo um comparativo entre características de cada algoritmo, as características extraídas pelo algoritmo SIFT é superior ao de seus concorrentes pois elas são invariantes a escala, rotação, borrimento e iluminação. O SURF seria o próximo da classificação, perdendo para o SIFT quando o assunto é escala e borrimento (a capacidade de combinar as características para esses fatores, são menores que a do SIFT). O ORB é invariante a escala e rotação e o detector de Harris é considerado o mais fraco dentre eles, pois se a imagem for registrada em outro ângulo, perde a possibilidade de realizar a combinação das características.

Outro ponto a ser relatado é que os algoritmos SIFT e SURF possuem licença *General Public License* (GPL) sendo patenteados por seus respectivos criadores e para utilizar esses algoritmos com a finalidade de comercializar há a necessidade de pagar *royalties* para os detentores da patente. Porém para uso em pesquisas acadêmicas o uso é liberado desde que seja feito a referência correta dos trabalhos que apresentam esses algoritmos. O ORB possui uma licença *Berkeley Software Distribution* (BSD) que permite

ser comercializada desde que seja referenciado corretamente. O detector de Harris é de domínio público.

2.3 Combinação de Características

Após extrair as características de duas ou mais imagens, [Brown e Lowe \(2007\)](#) escrevem sobre a necessidade de encontrar todas combinações entre as imagens antes de realizar a junção entre elas (Figura 7). Para [Adel, Elmogy e Elbakry \(2014\)](#) "a sobreposição compartilha a mesma quantidade de características extraídas das imagens e estas informações podem ser utilizadas para estabelecer uma relação de transformação entre as imagens envolvidas no processo de construção do mosaico".

Figura 7 – Combinações entre as características de duas imagens que possuem áreas de sobreposição.

Fonte: [Brown e Lowe \(2007\)](#)

2.4 *Random Sample Consensus* e Homografia

Segundo [Adel, Elmogy e Elbakry \(2014, 2015\)](#), [Brown e Lowe \(2007\)](#) e [Joshi e Sinha \(2013\)](#), calcular a matriz de homografia é a próxima etapa na construção do mosaico de imagens. Nesse cálculo, pontos chave indesejados que não pertencem à zona de sobreposição são removidos com a ajuda do algoritmo *Random Sample Consensus* (RANSAC). Este é um método iterativo utilizado para estimar os parâmetros matemáticos de um conjunto de dados observados que contém valores atípicos. O algoritmo não é determinístico no sentido de que ele produz um resultado razoável apenas com uma certa probabilidade, esta probabilidade cresce à medida que mais iterações são permitidas. O algoritmo foi publicado pela primeira vez por [Fischler e Bolles \(1981\)](#). O algoritmo RANSAC é usado na montagem de modelos em presença de muitos *outliers* de dados disponíveis de forma robusta.

Figura 8 – Aplicação do algoritmo RANSAC, onde os pontos chave indesejados que não pertencem a zona de sobreposição são removidos.

Fonte: [Brown e Lowe \(2007\)](#)

Homografia é o mapeamento entre os dois espaços que muitas vezes são utilizados para representar a correspondência entre duas imagens da mesma cena. É amplamente útil nas situações em que várias imagens são registradas a partir de uma câmera rotativa tendo um centro de câmera fixa. Através da matriz homográfica é possível estimar os parâmetros intrínsecos (características digitais da câmera) e parâmetros extrínsecos (referências conhecidas do plano 3D) que foram utilizados para registrar as imagens aéreas.

2.5 *Warping*

[Joshi e Sinha \(2013\)](#) escrevem que o último passo é deformar e misturar todas as imagens de entrada para se obter um mosaico na saída. A deformação de imagens (*warping*) é uma transformação que mapeia todas as posições em um plano para posições em um segundo plano baseando-se na matriz homográfica.

Essa técnica não pode ser confundida com um filtro passado sobre imagem, pois este altera a intensidade dos pixels da imagem (vide Figura 9) enquanto o *warping* altera o domínio da imagem (vide Figura 10).

Figura 9 – Aplicando um filtro na imagem altera a intensidade dos pixels.

Fonte: [IMAGE... \(2018\)](#)

Figura 10 – Aplicando o (*warping*), o domínio da imagem é alterado.

Fonte: [IMAGE...](#) (2018)

Trazendo essa técnica para ser aplicada na construção do MIA, [Glasbey e Mardia \(1998, p. 155\)](#) dizem que a deformação:

[...] surge em muitos problemas de análise de imagem, quer a fim de eliminar distorções ópticas introduzidas por uma câmera ou uma perspectiva de visão particular, para registrar uma imagem com um mapa ou modelo, ou para alinhar duas ou mais imagens. [...] A correspondência entre as imagens pode ser especificada por pontos para serem trazidos para o alinhamento, por meio de medidas locais de correlações entre as imagens, ou pela coincidência de arestas.

Em outras palavras essa técnica deforma alguns pontos da imagem (no caso da construção do MIA os pontos chaves) afim de moldar sua forma para que um melhor encaixe entre as imagens sejam possíveis (vide Figura 11).

Figura 11 – Efeito da deformação (*warping*) aplicado em duas imagens.

Fonte: [Brown e Lowe \(2007\)](#)

2.6 Blending

[Brown e Lowe \(2007\)](#) relata que "cada pixel ao longo de um raio deveria ter a mesma intensidade em cada imagem, mas na prática não é isso que acontece". Após a junção, é possível ver algumas arestas das imagens devido a uma série de efeitos, tais como vinhetas (intensidade diminui em direção à borda da imagem), efeitos de paralaxe devido ao movimento indesejado do centro óptico, erros registrados incorretamente devido

a câmera, distorção radial e assim por diante (BROWN; LOWE, 2007). O *blending* possui a função de corrigir essa série de efeitos fazendo com que o MIA fique suavizado com a variação de suas cores.

Efros (2010) exemplifica de uma forma clara o funcionamento do *blending*.

Figura 12 – Imagens separadas.

Fonte: Efros (2010)

Sem aplicar essa técnica, é possível notar que ao juntar as duas imagens acima há uma mudança brusca de uma imagem para a outra, ficando visível um risco separando as duas imagens.

Figura 13 – Resultado da junção sem aplicar o *blending*.

Fonte: Efros (2010)

A fórmula para aplicação do *blending* é dada pela seguinte representação matemática:

$$I_{blend} = \alpha I_{esquerda} + (1 - \alpha) I_{direita} \quad (2.5)$$

onde: I_{blend} é a imagem resultando após a aplicação matemática, α é um coeficiente de multiplicação que varia de 0 a 1 e $I_{direita}$ e $I_{esquerda}$ representam as imagens pertencentes a posição direita e esquerda respectivamente da imagem resultante.

Aplicando a equação 2.6 do *blending* obtém-se uma junção mais suavizada das duas imagens, fazendo com que o risco que antes separava as duas imagens desapareça e ela se torne uma imagem uniforme.

Figura 14 – Resultado da junção ao aplicar o *blending*.

Fonte: Efros (2010)

Entretanto, após a construção do mosaico, aos olhos humanos é bonito ver tudo de forma uniforme, mas, quando o assunto é agricultura, aplicar o *blending* pode fazer com que haja perda significativa de informações. O fato de suavizar ("borrar") as cores de uma imagem implica em alterar o valor da intensidade do pixel para todas as bandas RGB das imagens envolvidas, fazendo com que a informação registrada com a câmera seja alterada. É preciso analisar para qual finalidade que o mosaico foi construído antes de aplicar essa técnica.

2.7 Georreferenciamento Direto (FILHO, 2017)

O georreferenciamento direto baseia-se nas medições de posicionamento e orientação do sensor (câmera). Esses valores são obtidos no momento de captura de cada imagem pelos sensores *Global Positioning System* (GPS) e *Inertial Measurement Unit* (IMU) acoplados ao VANT. Em aplicações fotogramétricas estes valores são utilizados como parâmetros para calibração do sistema (Modelo de Câmera Estenopeica).

2.8 Modelo de Câmera Estenopeica (Câmera Pinhole)

"Este modelo de câmera define um mapeamento geométrico do mundo 3D para o plano da imagem 2D, conhecido como projeção perspectiva" (SANTOS; ROCHA, 2012, p.1).

2.8.1 Parâmetros Intrínsecos

Os parâmetros intrínsecos são um "conjunto de parâmetros necessários para caracterizar a ótica, geometria e as características digitais da visualização da câmera"(TRUCCO; VERRI, 1998, p.36). Em outras palavras, esses parâmetros são utilizados para relacionar as coordenadas dos pontos da imagem com as coordenadas do sistema de referência da câmera (vide Figura 15). Para montar a matriz intrínseca, Trucco e Verri (1998) escrevem que são necessários os seguintes dados extraídos da câmera:

- distância focal em milímetros (f);
- ponto central da imagem (o_x e o_y);
- tamanho do pixel em milímetros (s_x e s_y)

Figura 15 – Ilustração dos parâmetros intrínsecos.

Fonte: Campos e Câmara Neto (2007)

Tendo essas informações, eles equacionam a matriz intrínseca (equação 2.6).

$$M_{int} = \begin{bmatrix} -f/s_x & 0 & o_x \\ 0 & -f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.6)$$

2.8.2 Parâmetros Extrínsecos

Os parâmetros extrínsecos são definidos como um "conjunto de parâmetros geométricos que identificam unicamente a transformação entre a referência de uma câmera desconhecida e um frame de referência conhecido"(TRUCCO; VERRI, 1998, p.35). Em outras palavras, esses parâmetros são utilizados para descrever as transformações entre os

pontos da câmara e os pontos do mundo. Para montar a matriz extrínseca, [Trucco e Verri \(1998\)](#) escrevem que são necessárias duas matrizes:

- uma matriz 3D de translação $\mathbf{T}$, descreve a posição onde cada foto foi registrada;
- uma matriz de rotação 3x3 $\mathbf{R}$, que descreve o eixo em que a câmara estava rotacionada no momento do registro da foto;

2.8.2.1 Rotação

Os ângulos (*Roll*, *Pitch*, *Yaw*) fornecidos pelo sensor IMU são utilizados para calcular as matrizes de rotação para os eixos X, Y e Z, onde o ângulo *Roll* (α) é a rotação no eixo X, *Pitch* (β) é a rotação no eixo Y e *Yaw* (γ) é a rotação no eixo Z. As equações a seguir mostram como construir as matrizes de rotação para os eixos X, Y e Z utilizando os ângulos vindo do sensor IMU que deverão ser convertidos para radianos.

$$Rx(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix} \quad (2.7)$$

$$Ry(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \quad (2.8)$$

$$Rz(\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.9)$$

Para obter-se a matriz de rotação final, é necessário realizar a multiplicação das três equações [2.7](#), [2.8](#) e [2.9](#) como é mostrado na equação [2.10](#).

$$R = Rx(\alpha) * Ry(\beta) * Rz(\gamma) \quad (2.10)$$

2.8.2.2 Translação

A matriz de translação (equação [2.11](#)) é montada a partir das coordenadas do GPS, onde a **longitude** é o eixo **x** da imagem e a **latitude** é o eixo **y** da imagem e a **altura** (**z**) representa a transformação de escala que deverá ser aplicada na imagem.

$$T = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (2.11)$$

2.8.2.3 Matriz Extrínseca

Para montar a matriz extrínseca basta concatenar as matrizes de rotação com a de translação. Porém como há uma rotação de câmera no momento em que a foto é registrada, é necessário aplicar essa rotação na translação também. Para isso basta multiplicar isoladamente as colunas (transpostas) da matriz de rotação final pela matriz de translação para se obter as translações reais de x, y e z. Dado esses fatos, [Trucco e Verri \(1998\)](#) apresentam em seu livro a matriz extrínseca final.

$$M_{ext} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & -R_1^T * T \\ r_{21} & r_{22} & r_{23} & -R_2^T * T \\ r_{31} & r_{32} & r_{33} & -R_3^T * T \end{bmatrix} \quad (2.12)$$

2.8.3 Matriz de Projeção da Câmera

A matriz de projeção da câmera é dada pela multiplicação da matriz intrínseca pela extrínseca.

$$M_{proj} = M_{int} * M_{ext} \quad (2.13)$$

Essa matriz é muito utilizada para mapear os pontos 3D (P_w = pontos no mundo) para os pontos 2D (P_c = pontos na câmera) no plano da imagem e vice versa ([SANTOS; ROCHA, 2012](#)). A equação 2.14 mostra como os valores devem ser entrados para realizar esse tipo de mapeamento.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = M_{proj} * \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (2.14)$$

Ao obter os valores de x_c , y_c e z_c , ainda é necessário executar as seguintes equações para se obter as coordenadas do ponto 3D na imagem:

$$x_{im} = \frac{x_c}{z_c} \quad (2.15)$$

$$y_{im} = \frac{y_c}{z_c} \quad (2.16)$$

2.9 Triangulação

A triangulação é utilizada para encontrar a posição de um ponto no espaço dado sua posição em duas imagens ou mais e a matriz extrínseca. O grande desafio da triangulação é

encontrar a intersecção no espaço 3D traçando uma linha entre os pontos correspondentes das duas imagens ([HARTLEY; STURM, 1997](#)). A Figura 16 exemplifica o que [Ntawiniga \(2008\)](#) diz: "Em teoria, as duas linhas devem se cruzar perfeitamente no espaço, mas devido ao ruído e outros erros¹ não é o caso".

Figura 16 – As retas L e R deveriam se cruzar perfeitamente no espaço, mas não é o que acontece.

Fonte: O Autor

Há diversos métodos de triangulações que [Hartley e Sturm \(1997\)](#) descrevem:

- triangulação linear;
- método iterativo linear;
- método do ponto médio;
- minimizando a soma das magnitudes das distâncias;

Entretanto, é abordado neste trabalho somente o método do ponto médio, por possuir uma matemática simples e ser o método mais utilizado pelos pesquisadores.

2.9.1 Método do ponto médio

O objetivo desse método é traçar uma reta que ligue as duas retas (L e R) e o ponto triangulado estará no centro dessa reta. Todos os vetores e coordenadas utilizados para esse cálculo é tido a câmera da esquerda (L) como referência. O equacionamento das retas L e R do cálculo do ponto médio é apresentado pela equação 2.17.

$$\begin{cases} P_L = \alpha \vec{x}_L + cw \\ P_R = T + \beta R^T \vec{x}_R \end{cases} \quad (2.17)$$

¹ Esses podem ser: erro de digitalização, ruído no sensor, erro na combinação de características, etc

Onde:

α, β e $c \in \mathbb{R}$ ², w é um vetor ortogonal as retas L e R representado por $w = \vec{x}_L \times R^T \vec{x}_R$ e os dados fornecidos pelas matrizes extrínsecas são utilizados para calcular R e T.

$$R = R_R R_L^T$$

$$T = T_L - R^T T_R$$

Sendo o mesmo ponto no espaço visto por dois frames diferentes, as equações das duas retas podem ser igualadas e representadas pela equação 2.18 como é descrito por Trucco e Verri (1998).

$$\alpha \vec{x}_L - \beta R^T \vec{x}_R + c(\vec{x}_L \times R^T \vec{x}_R) = T \quad (2.18)$$

Para finalizar o cálculo basta dividir o resultado encontrado na equação 2.18 por dois (2).

2.10 Bundle Adjustment

Considere um conjunto de pontos 3D (pontos do mundo) sendo visto a partir de um conjunto de câmeras (pontos de imagens), o resultado do MIA após passar pelo *warping* não seria nada satisfatório, pois cada matriz de projeção está vindo de forma deslocada estes mesmos pontos. Esse deslocamento pode ser causado por curvas de níveis no solo, inclinação do VANT causado pelo vento forte no momento de captura da imagem. Dado essas situações, é necessário um ajuste (calibração) nas matrizes de projeção de cada câmera com a finalidade responder a seguinte questão "quais são as matrizes de projeção "ótima" que somado ao erro quadrado de reprojeção é mínimo?" (SÜNDERHAUF et al., 2006, p. 158)

O *Bundle Adjustment* (BA) é a técnica utilizada para encontrar a matriz de projeção otimizada para cada câmera, baseando-se nos pontos 3D (coordenadas do mundo) e 2D (coordenadas da imagem) (LOURAKIS; ARGYROS, 2004). Ao final da execução dessa técnica, todas as câmeras estarão alinhadas e enxergarão os mesmos pontos chaves. O BA é dado pela equação 2.19.

$$\min_{M_i P_j} \sum_i^n \sum_j^m d(M_i P_j, p_{ij})^2 \quad (2.19)$$

Onde:

² Quando α e β são iguais a zero (0) representa a origem das coordenadas das câmeras L e R (O_L e O_R). Quando forem iguais a um (1) representa o ponto no plano das imagens L e R ($\vec{x}_L$ e $\vec{x}_R$)

M_i são as matrizes de projeção de cada câmera, P_j são os pontos no espaço (mundo), p_{ij} são as coordenadas da imagem do j -ésimo ponto do mundo na i -ésima imagem e a função $d(x, y)$ é a distância Euclidiana entre x e y .

"BA é um problema de minimização não-linear que pode ser resolvido usando métodos de mínimos quadrados não-lineares iterativos como Levenberg-Marquardt" (SÜNDERHAUF et al., 2006, p.158).

A Figura 17 exemplifica o resultado da aplicação do BA onde todos os pontos chave combinados, entre 2 ou mais imagens, estão calibrados e todas as câmeras passam a visualizar o mesmo ponto no espaço 3D.

Figura 17 – "Ilustração do resultado da aplicação do BA".

Fonte: Bundle... (2012)

2.11 Computação Heterogênea

Há trinta anos, os maiores esforços para aumentar o desempenho das *Central Processing Units* (CPUs) estavam focados em como aumentar a velocidade de *clock*, porém o foco foi mudado quando os fabricantes de CPU começaram a oferecer processadores com dois cores criando assim o conceito de multi-core (SANDERS; KANDROT, 2010). Os processadores multi-core mais utilizados atualmente pelos sistemas embarcados são os *Advanced RISC Machine* (ARM).

Atualmente, os processadores ARM encontram-se nas versões ARMv7, ARMv8 e ARMv8-A, que são as arquiteturas 32-bit e 64-bit que agregam as tecnologias mais modernas, podendo envolver *chips* com mais de um núcleo, além de unidades de ponto flutuante VFP e NEON, que auxiliam em muito nas tarefas de processamento aritmético, multimídia e processamento digital de sinais (CURVELLO, 2016, p. 56).

Com o nascimento da GPU, o mundo dos *games* pode evoluir para que a parte gráfica fique muito próxima do mundo real trazendo mais emoção aos jogadores. A grande contribuição das GPUs para os jogos está focada na parte de renderização. Quando o cálculo exige um alto desempenho, a CPU solicita à GPU para que realize-os e devolva os resultados à CPU. Segundo [Ubal et al. \(2012\)](#), o cluster formado por CPU e GPU não é uma opção para computação de alto desempenho, ela é o futuro onde tanto academia e indústria têm investido fortemente em pesquisas colhendo bons resultados.

A popularização principalmente das GPUs, somado a sua facilidade em programar, promoveu uma melhora muito grande no desempenho de muitas aplicações que antes utilizavam somente a CPU ([CHE et al., 2009](#)). Com isso surgem os primeiros dispositivos heterogêneos.

Generalizando o conceito, a CH é formada pela junção de duas ou mais Unidades Computacionais (UC) diferentes com a finalidade de aumentar o desempenho e a eficiência do consumo de energia ([GAO; ZHANG, 2016](#)). Essas UC podem ser uma CPU, GPU, *Digital Signal Processor* (DSP), FPGA ou um *Application Specific Integrated Circuits* (ASIC). A grande vantagem de uma CH é repartir o processamento, que antes era feito apenas por uma UC, para as demais UC fazendo com que a aplicação ganhe em desempenho.

As principais UC utilizadas em CH embarcada são formadas por CPU e GPU, pois muitas aplicações são de propósito geral e com isso a GPU tem uma colaboração mais significativa que as demais UC. Aplicações como processamento digital de imagem, reconstrução 3D, visão computacional, reconhecimento de padrões são apenas algumas das áreas em que a CH formada por CPU e GPU teria uma grande vantagem com relação a uma aplicação sendo executada somente na CPU.

As próximas sessões descrevem brevemente sobre as UC embarcadas (CPU ARM, GPU e FPGA) que geralmente são utilizadas na CH.

2.11.1 ARM

"As CPUs ARM são conhecidas por seu baixo consumo de potência elétrica, sendo ideais para sistemas embarcados móveis"([LANGBRIDGE, 2013](#), p. xxi). Sua arquitetura é do tipo *Reduced Instruction Set Computer* (RISC) que tem como filosofia implementar um número reduzido de instruções que o processador deverá executar ([CURVELLO, 2016](#)).

2.11.1.1 Características da arquitetura

Com base no que [Forte \(2015\)](#) escreve, há alguns aspectos que não mudam conforme a evolução desses processadores, como por exemplo, o ARMv4, o ARMv5, o ARMv6 e o ARMv7:

1. Possuem 37 registradores;

- a) 30 são de propósito geral, onde 2 são reservados para ponteiro de pilha e *link register*;
 - b) e 7 de uso específico, que incluem contador de programa e registradores de estado;
2. A maioria das instruções apresenta um campo de 4 bits, em que é definido uma condição para que a operação seja executada;
3. Possui 7 modos de execução do processador;
- a) modo usuário, que não oferece nenhum tipo de privilégio ao programa em execução;
 - b) os outros 6 modos possuem diferentes opções de privilégios permitindo ao Sistema Operacional fácil adaptação a diferentes cenários e tipos de aplicações;
4. Mapeamento da memória cache é realizado de forma associativa em conjuntos;
- a) os primeiros ARMs utilizavam a memória cache de endereços lógicos (cache lógica), mas atualmente usa-se cache física;
 - b) existe um buffer de escrita entre a cache e a memória principal. Este buffer armazena dados se a memória principal estiver cheia e até que haja espaço, as escritas são suspensas;
5. A memória principal é organizada em seções (16MB e 1MB) e em páginas (64KB e 4 KB);
- a) duas tabelas são mantidas dentro da memória principal para aumentar a busca na memória (L1 para busca nas seções e L2 para busca nas páginas);
 - b) há a divisão em domínios de memória (grupos de seções e páginas) e para acessá-los é necessário permissão;
6. Possui 7 tipos de exceção para tratar interrupções;

2.11.1.2 Vector Floating-Point (VFP) e NEON

Como escrito na sessão 2.11, VFP é um coprocessador para operações aritméticas de ponto flutuante encontradas dentro dos processadores ARM1176JZF-S. "Esse coprocessador implementa os padrões IEEE para processamento aritmético binário com ponto flutuante, estabelecido pela norma IEEE 754" (CURVELLO, 2016, p. 56). Alguns exemplos de aplicações que utilizam o VFP são de controle automotivo, aplicações de imagem como escala, transformações, transformação 3D entre outras.

O Coprocessador NEON é utilizado pelos processadores ARM para acelerar aplicações multimídia e algoritmos de processamento de sinal como codificação/decodificação de

vídeo, gráficos 2D/3D, processamento de jogos, áudio e fala, processamento de imagem, telefonia e som. Essa tecnologia estende da arquitetura *Single Instruction Multiple Data* (SIMD) de 128 *bits* projetada para os processadores ARM-Cortex - A série, [ARM \(2017\)](#).

Tanto o VFP quanto o NEON podem ser considerados UC, dentro do conceito de CH, pois eles não fazem parte da arquitetura ARM, mas são encapsulados junto a ela.

2.11.2 GPU

"A GPU é um processador de múltiplos núcleos altamente paralelo, *multithreaded*, com uma alta capacidade de computar e alta memória de largura de banda"([SOOKNANAN; JOSHI, 2016](#), p. 1261). Ao contrário das CPUs que executam as aplicações de forma sequencial, as GPUs são muito utilizadas para executar as aplicações paralelamente diminuindo o tempo de resposta das saídas. As GPUs são otimizadas para executarem um número muito alto de *threads* e as CPUs são otimizadas para um bom desempenho em códigos executados sequencialmente, diz [Sooknanan e Joshi \(2016\)](#). A diferença do design das duas arquiteturas é apresentada pela Figura 18.

Figura 18 – Diferença entre a arquitetura de um CPU e de uma GPU

Fonte: [Introduction...](#) (2017)

As GPUs geralmente são utilizadas pelos sistemas embarcados como uma Unidade de Processamento Gráfico de Propósito Geral (GPGPU). Esse nome é dado para a GPU quando ela não é utilizada com a exclusividade na renderização gráfica, mas também quando utilizada para processamento de imagens, cálculos matemáticos entre outros.

Existem diversos fabricantes de GPUs, porém Intel, NVIDIA e ATI destacam-se sobre os outros fabricantes. "A Intel, produz processadores gráficos mais direcionados para aplicações de baixo custo integradas à placas-mãe, enquanto que a NVIDIA e a ATI atuam nos demais segmentos"([MACHADO, 2012](#)).

2.11.2.1 Principais linguagens de programação para GPUs

Duas linguagens de programação possuem o maior foco quando se trata em programar GPUs, são elas *Compute Unified Device Architecture* (CUDA) e *Open Computing Language* (OpenCL). Esta vem ganhando maior notoriedade entre a comunidade por ser uma linguagem de programação aberta, porém CUDA possui maior desempenho.

2.11.2.2 CUDA

"CUDA é uma plataforma de computação paralela e modelo de programação inventado pela NVIDIA. Permite aumentos dramáticos no desempenho de computação aproveitando a potência da GPU"(NVIDIA..., 2016, p. 1).

Os programas desenvolvidos em CUDA são divididos em partes que podem ser executadas tanto no *host* (CPU) quanto em um dispositivo como a GPU. Quem realiza a separação do que será executado no *host* e no dispositivo é o compilador NVIDIA C. O código que executará no *host* segue o padrão de codificação ANSI C e o que executará na GPU é escrito usando ANSI C também, porém com algumas palavras chaves (como *kernel*) que são utilizadas para informar que determinada função deverá ser processada de forma paralela pela GPU (KIRK; WEN-MEI, 2010). A Figura 19 exemplifica a execução de um programa escrito utilizando a linguagem CUDA.

Figura 19 – "Execução de um programa CUDA"

Fonte: Kirk e Wen-Mei (2010)

Kirk e Wen-Mei (2010, p. 42) explica o funcionamento da Figura 19 da seguinte maneira:

"A execução do programa começa no *host* (CPU). Quando uma função *kernel* é invocada a execução é movida para um dispositivo (GPU), onde um número grande de *threads* são geradas para tirar vantagem do paralelismo. Todas as *threads* são criadas por um *kernel* e durante uma invocação são chamadas coletivamente de *grid*. A Figura 19 mostra a execução de duas *grids* de *threads*. [...]"

Quando todas as *threads* de um *kernel* completam sua execução, o *grid* correspondente termina e a execução continua no *host* até outro *kernel* ser invocado".

2.11.2.3 OpenCL

Munshi et al. (2011, p. 3) diz que o "OpenCL é uma estrutura padrão da indústria para a programação de computadores compostos por uma combinação de CPUs, GPUs e outros processadores". Utilizando o OpenCL é possível criar programas totalmente independentes do fabricante e do dispositivo e assim acelerar aplicações em diversas plataformas de *hardware* (GASTER et al., 2012). O OpenCL atualmente é mantido pelo grupo Khronos ³.

O código que é executado no dispositivo OpenCL (GPU, FPGA, etc.) é escrito na linguagem OpenCL C. "OpenCL C é uma versão restrita da linguagem C99 com extensões apropriadas para a execução de código paralelo de dados em uma variedade de dispositivos heterogêneos"(GASTER et al., 2012, p. 15).

Munshi et al. (2011) explica as especificações do OpenCL que são divididas em 4 módulos:

- **Plataforma:** uma descrição de alto nível do sistema heterogêneo;
- **Execução:** uma representação abstrata de como os fluxos de instruções executam na plataforma heterogênea;
- **Memória:** a coleção de regiões de memória dentro OpenCL e como eles interagem durante uma computação OpenCL;
- **Programação:** as abstrações de alto nível que um programador usa ao projetar algoritmos para implementar um aplicativo;

2.11.3 FPGA

Um FPGA é um circuito integrado desenvolvido para que o projetista de *hardware* consiga reprogramar o dispositivo após sua fabricação.

"O termo '*field programmable*' refere-se à capacidade de alterar as operações do dispositivo 'no campo', enquanto que '*gate array*' é uma referência a arquitetura interna básica que torna possível esta reprogramação após a fabricação"(PELLERIN; THIBAUT, 2005, p. 2).

Pellerin e Thibault (2005) escrevem que todos os FPGAs compartilham das mesmas características básicas:

³ <https://www.khronos.org>

1. **Elementos Lógicos (Células Lógicas):** os FPGAs baseiam-se em matrizes de elementos lógicos digitais (AND, OR, NOT). "Para usar tal dispositivo, problemas de lógica digital devem ser decompostos em um arranjo de circuitos lógicos menores que podem ser mapeados para um ou mais destes elementos lógicos"(PELLERIN; THIBAUT, 2005);
2. **Lookup tables:** a estrutura de cada elemento lógico muda conforme o fabricante e a família do qual os FPGAs pertencem, como por exemplo fabricante Altera família Cyclone e fabricante Xilinx família Virtex. Os elementos lógicos são compostos de pelo menos um registrador programável (Flip-Flop) e algumas formas de entrada que são implementadas em formas de *look up table* (LUT) de n entradas. "Uma LUT de n entradas pode ser considerada como uma pequena memória de 2^n por 1. Ao escrever corretamente o conteúdo da memória, podemos usar a LUT para implementar qualquer função combinacional de n entradas"(CHU, 2011, p. 12).
3. **Recursos de Memória:** A maioria dos FPGAs atuais possuem uma memória incorporada ao chip (uma Static Random Access Memory (SRAM)). O acesso a memória é feito de forma hierárquica. Para acessar a memória local, que está dentro de cada elemento lógico, é preciso combinar os blocos de memória globalmente compartilhados;
4. **Recursos de Roteamento (Interconexões Programáveis):** o roteamento é a chave para a flexibilidade dos FPGAs, pois permite conectar os elementos lógicos em função das necessidades do projeto que está sendo desenvolvido. A velocidade e o tempo de propagação dos sinais nas chaves programáveis são afetadas pela resistência, capacitância e tecnologia de fabricação. Volatilidade e capacidade de reprogramação são fatores a serem avaliados na escolha de um FPGA. Existem 3 tecnologias para se programar as chaves de roteamento: SRAM, Antifuse e Gate Flutuante (OLIVEIRA; AGUIAR; FONTANINI, 2017);
5. **Entrada/Saída Configurável:**

As aplicações baseadas em FPGA têm uma grande variedade de requisitos de interface de nível de sistema e, por isso, as FPGAs têm muitas características de I/Os programáveis. Os pinos de uma FPGA podem ser configurados como TTL, CMOS, PCI e muito mais, permitindo as FPGAs serem interfaceadas e convertidas para muitas tecnologias de circuitos diferentes. A maioria dos FPGAs também possuem I/Os de alta velocidade dedicados para clocks e resets globais, e muitos FPGAs incluem PLLs e esquemas de gerenciamento de clock, permitindo que projetos com vários domínios de clock independentes sejam criados e gerenciados (PELLERIN; THIBAUT, 2005, p.3).

A Figura 20 apresenta a arquitetura de um FPGA ilustrando as características descritas acima.

Figura 20 – Arquitetura de um FPGA

Fonte: [INTRO...](#) (2014)

Usualmente os FPGAs são programados utilizando uma linguagem de descrição de *hardware* (HDL) (como VHDL ou Verilog), OpenCL ou combinando diagramas de bloco de funções pré-definidas. Porém hoje em dia já é possível configurar o comportamento do FPGA utilizando as linguagens tradicionais como C, C++, Java e Python. Utilizando qualquer uma dessas linguagens o resultado final é um arquivo chamado *netlist* que realiza o mapeamento para a FPGA escolhida usando um software de roteamento do fabricante da FPGA escolhida ([PELLERIN; THIBAUT, 2005](#)).

2.12 Considerações Finais

Tanto as técnicas de processamento digital de imagens (principalmente extração de características) e de visão computacional (especificamente triangulação e *bundle Adjustment*) envolvidas no processo de construção do mosaico de imagens aéreas exigem um alto poder de processamento pelo fato de utilizarem cálculos pesados e recursivos.

Para este trabalho são utilizadas as técnicas *warping* para manipular a perspectiva das imagens que formam o mosaico de imagens aéreas, modelo de câmera pinhole para posicionar as imagens para serem encaixadas e extração e combinação de características para refinar o encaixe. As demais técnicas não são aplicadas neste trabalho, pois é sugerida uma abordagem mais simplificada para a construção do mosaico, que será apresentada no Capítulo 4.

Visando diminuir esse esforço computacional exigido pelas técnicas citadas acima, com a arquitetura do FPGA, seria muito mais rápido do que com a GPU em casos que o processamento seja implementado diretamente no *hardware*, transformando-o em um dispositivo ASIC, pois este possui uma quantidade maior de elementos lógicos. Caso a intenção seja implementar um processador dentro do FPGA para realizar as tarefas citadas antes, ele perderia em velocidade para a GPU, pois a frequência que o FPGA trabalha é baixa (ficando na casa dos MHz), enquanto a GPU trabalha com uma frequência mais alta (ficando na casa dos GHz).

Ao utilizar a GPU em dispositivos embarcados, ela vem acompanhada de uma CPU (geralmente um ARM) para realizar toda a parte de processamento sequencial e assim, ajudar o sistema operacional a gerenciar as tarefas. O FPGA também possui uma CPU integrada, porém esta pode estar implementada ou não utilizando os elementos lógicos disponibilizados pela FPGA, ou seja um processador implementado em *hardware*.

Ao desenvolver uma aplicação em CUDA qualquer dispositivo gráfico NVIDIA, seja ele embarcado ou não, a aplicação poderá ser compilada sem a necessidade de realizar alterações no código. Caso a aplicação escolhida seja desenvolvida em OpenCL, há a necessidade de verificar se as bibliotecas estão disponíveis para a arquitetura e dispositivo alvos.

3 MATERIAIS E MÉTODOS

No presente Capítulo será descrito como foi realizado a aquisição e o processamento das imagens, o conjunto de dados que compõem as imagens utilizadas neste trabalho e uma breve descrição sobre as principais plataformas heterogêneas disponíveis no mercado atual e que podem se encaixar na execução deste trabalho.

3.1 Aquisição e Processamento das Imagens

Todo o processo, partindo do registro das imagens até o MIA, está separado em duas partes principais: o sistema de aquisição (utilizando um VANT) e o sistema de processamento (utilizando uma plataforma CH). Não foi possível integrar os dois sistemas porque isso exigiria alterações no VANT e este pertence a um terceiro. O processamento das imagens foi iniciado somente após o término do voo. A Figura 21 ilustra os dois processos realizado para construir o MIA, onde o segundo processo foi realizado por uma plataforma heterogênea e por um notebook.

Figura 21 – Aquisição e Processamento das Imagens Separados

Fonte: O Autor

3.2 O Sistema de Aquisição e o Conjunto de Imagens

As imagens utilizadas neste trabalho foram capturadas utilizando dois tipos diferentes de VANTs: DJI Phantom 4¹ (utilizado para construir o MIA 1) e o DJI Phantom 4 Pro² (utilizado para construir o MIA 2). As principais características desses dois VANTs são descritas pela tabela 2.

¹ <https://www.dji.com/br/phantom-4/info>

² <https://www.dji.com/br/phantom-4-pro/info>

Tabela 2 – Tabela com as principais características dos dois tipos de VANTs utilizados neste trabalho.

	<i>DJI Phantom 4</i>	<i>DJI Phantom 4 Pro</i>
Aeronave		
<i>Sist. Posicionamento por Satélite</i>	GPS/GLONASS	
<i>Faixa de Precisão Hover</i>	Vertical: ± 0,1 m (Visão) e ± 0,5 m (GPS) Horizontal: ± 0,3 m (Visão) e ± 1,5 m (GPS)	
Controle Remoto		
<i>Frequência</i>	2,400 a 2,483 GHz	2,400 a 2,483 GHz e 5,725 a 5,825 GHz
<i>Distância de Alcance</i>	3,5 a 5,0 km	2,0 a 7,0 km
<i>Bateria</i>	6000 mAh LiPo 2S	
Gimbal		
<i>Estabilização</i>	3 eixos (pitch, roll, yaw)	
<i>Faixa Controlável</i>	-90° a +30° em pitch	
<i>Vibração Angular</i>	± 0,02°	
Câmera		
<i>Modelo</i>	FC330	FC6310
<i>Sensor</i>	1/2.3"CMOS	1"CMOS
<i>Lente</i>	FOV 90° 20 mm	FOV 84° 24 mm
<i>Pixels</i>	12.4 M	20 M
<i>Resolução</i>	4000 x 3000	5472 x 3648

Fonte: O Autor

As informações que compõem cada conjunto de imagem para formar os MIAs 1 e 2 estão descritas na tabela 3.

Tabela 3 – Tabela com as informações do conjunto de imagens.

	MIA 1	MIA 2
<i>Altitude Relativa do Voo (m)</i>	100,0	120,0
<i>Altitude de Voo Nível do Mar (m)</i>	663,49	939,45
<i>Coordenada Central do MIA</i>	22°32'59.4"S, 46°56'48.5"W	21° 57' 14.1"S, 47° 51' 13.1"W
<i>Quantidade de Imagens Registradas</i>	291	144

Fonte: O Autor

3.3 Plataformas com Sistema Heterogêneo Embarcado

3.3.1 Inforce 6540 SBC (IFC6540)

A IFC6540 (Figura 22) é uma *Single Board Computer* (SBC) fabricada pela Inforce... (2016) e tem como principal processador o Snapdragon 805 (APQ8084) fabricado pela Qualcomm ³. Este processador é um dispositivo da CH, pois ele possui encapsulado em um mesmo chip algumas unidades computacionais como:

- Quad core Krait 450 CPU executando em 2.7GHz
- Adreno 420 GPU @4.8 Giga Pixels/segundos
- Hexagon DSP v50 @600MHz
- Dual Image Signal Processors (ISPs) @1.2GP/s

Figura 22 – Inforce 6540 SBC

Fonte: Inforce... (2016)

A IFC6540 também possui outros componentes fundamentais para a SBC funcionar como um computador. A Figura 23 apresenta todos os componentes eletrônicos que esta SBC contém e que o usuário final pode utilizar.

³ <https://www.qualcomm.com/>

Figura 23 – Diagrama de Blocos da IFC6540

Fonte: [Inforce...](#) (2016)

3.3.2 NVIDIA Jetson TK1

O Kit de Desenvolvedor NVIDIA Jetson TK1 (Figura 24) é fabricado pela [Nvidia...](#) (2017). Esse kit é baseado no *System on Chip* (SoC) NVIDIA Tegra® K1 e conta com a mesma GPU que os supercomputadores encontrados ao redor do mundo utilizam, a NVIDIA Kepler™. As especificações do SoC Tegra K1 é apresentada na Tabela 4.

Figura 24 – Kit de Desenvolvedor NVIDIA Jetson TK1

Fonte: [Nvidia...](#) (2017)

Tabela 4 – Especificações técnicas do SoC Tegra K1

Tegra K1	
<i>CPU</i>	
Núcleos de CPU e arquitetura	ARM Cortex-A15 "r3" de quatro núcleos NVIDIA 4-Plus-1™
Velocidade máxima do clock	2,3 GHz
<i>GPU</i>	
Arquitetura NVIDIA® Kepler™	192 núcleos NVIDIA CUDA®
<i>Memória</i>	
Tipo de Memória	DDR3L e LPDDR3
Capacidade máxima de memória	8 GB (com extensão de endereço de 40 bits)
<i>Tela</i>	
LCD	3840x2160
HDMI	4K (UltraHD, 4096x2160)
<i>Produto</i>	
Tipo / Tamanho	23x23 FCBGA, 16x16 S-FCCSP, 15x15 FC PoP
Processo	28 nm

Fonte: [Nvidia...](#) (2017).

3.3.3 Parallella Board

A Parallella Board (Figura 25) é um computador de alto desempenho baseado no chip multi-core Epiphany 16-core CPU da [The Parallella...](#) (2014). Esta plataforma tem como processador principal Xilinx Zynq Dual-core ARM A9 que permite executar diversas distribuições populares do Linux, incluindo o Ubuntu.

Figura 25 – Parallella Board

Fonte: [The Parallella...](#) (2014)

A Figura 26 apresenta o diagrama de blocos de conectividade do processador Xilinx Zynq Dual-core ARM A9.

Figura 26 – Diagrama de conectividade do Xilinx Zynq Dual-core ARM A9

Fonte: [The Parallella...](#) (2014)

3.3.4 Raspberry PI

A plataforma Raspberry Pi (Figura 27) é desenvolvida e mantida pela [Raspberry...](#) (2009). O seu atual modelo no mercado, o Raspberry Pi 3 Model B Motherboard, é baseado no processador BCM2837 64bit ARMv8 Cortex-A53 Quad-Core fabricado pela Broadcom.

Figura 27 – Raspberry Pi 3 Model B Motherboard

Fonte: [Raspberry...](#) (2009)

Outras especificações técnicas da Raspberry Pi 3 Model B Motherboard são:

- A 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot (now push-pull rather than push-push)
- VideoCore IV 3D graphics core

3.4 Considerações finais

A tabela 5 tem a função de descrever as principais características das plataformas citadas neste Capítulo para que facilite a comparação objetivando escolher a melhor plataforma para compilar e executar a aplicação desenvolvida neste trabalho.

Tabela 5 – Tabela de comparação das plataformas apresentadas no Capítulo 3.

	IFC6540	Jetson TK1	Parallella	Raspberry PI
CPU	Quad core Krait 450	Quad core ARM Cortex-A15	Xilinx Zynq Dual-core ARM A9	Quad core ARMv8
S.O	Linaro / Android	Ubuntu	qualquer distribuição popular Linux	Raspbian
GPU	Adreno 420	Kepler 192 core	Não	Não
CUDA	Não	Sim	Não	Não
OpenCL	Disponível somente para Android	Não	Sim	Disponível para o dispositivo VideoCore
FPGA	Não	Não	Sim	Não
DSP	Hexagon DSP v50	Não	80 partes programáveis para DSP	Não
Memória	2 GB DDR3	2 GB DDR3L	1 GB DDR3	1 GB

Fonte: O Autor

Todas as plataformas evidenciadas na tabela acima apresentam vantagens e desvantagens para a aplicação, como por exemplo a Parallella Board teria grande vantagem sobre as concorrentes por ser um FPGA e a implementação deve ser feita em hardware, porém para realizar um processamento em uma CPU, perde um pouco de vantagem devido a baixa frequência de operação de um FPGA. Das quatro plataformas apresentadas a Raspberry PI possui a menor vantagem sobre as demais, pois ela não possui um FPGA e também não possui GPU, contendo apenas um dispositivo dedicado para processar vídeos.

A Jetson TK1 e a IFC6540 possuem GPUs, porém apenas a Jetson TK1 oferece suporte a linguagem CUDA. Em contrapartida a IFC6540 oferece suporte para OpenCL utilizando Android como sistema operacional e a Jetson não possui esse suporte. Como a aplicação foi desenvolvida para operar em distribuições Linux utilizando CH, a Jetson TK1 foi utilizada para executar a aplicação deste trabalho. Parte da aplicação é executada na CPU (processamento sequencial utilizando a linguagem C++) e parte executada na GPU (processamento paralelo utilizando a linguagem CUDA).

4 ABORDAGEM PARA CONSTRUÇÃO DO MIA

Ao ler o Capítulo 2, é possível tomar conhecimento dos conceitos que devem ser aplicados para construir um mosaico de imagens.

Ao buscar por estudos que discorram sobre o assunto apresentado, retornou as abordagens propostas por Adel, Elmogy e Elbakry (2014, 2015), ambas publicações recentes, que apresentam uma maneira de tornar o processo de construção de mosaico de imagens mais rápido utilizando o método de extração de características chamado ORB, pois na maioria das publicações utilizam SIFT ou SURF. O ORB é muito utilizado para sistemas embarcados devido a sua velocidade em realizar os cálculos e ao baixo consumo de memória.

Joshi e Sinha (2013), também em um trabalho recente, mostram o processo todo para obter-se um mosaico de imagens. São seis passos:

1. Entrada de imagens;
2. Extração de características;
3. Registro de imagem (Combinação de Características);
4. Homografia usando RANSAC;
5. *Warping* e *Blending*;
6. Imagem de saída

A Figura 28 ilustra o processo para construção do MIA descrito por eles.

Ao estudar um pouco mais sobre o assunto, nota-se que os passos 1 e 2 descritos por Joshi e Sinha (2013) estão ligados diretamente e por isso eles tendem a ficar no mesmo bloco do diagrama. O passo 5 é separado, tendo cada um o seu bloco no diagrama, e então, é esboçado um novo diagrama para representar o fluxo dos métodos que envolve a construção do mosaico (vide Figura 28).

O projeto proposto por Brown e Lowe (2007) apresenta uma nova maneira totalmente automática para construir imagens panorâmicas utilizando o extrator de características SIFT. O projeto mostra de forma descritiva todos os cálculos utilizados para estimar os parâmetros intrínsecos e extrínsecos da câmera a partir da matriz homografica. Estes parâmetros são estimados a partir de uma técnica chamada *Bundle Adjustment* e depois eles são utilizados no passo de *warping*. A biblioteca de visão computacional chamada *Open Computer Vision* (OpenCV) mantém uma versão do software implementado por Brown e Lowe (2007) (chamado de *Stitcher*¹ na versão 3.1 do OpenCV), porém só é possível

¹ <http://docs.opencv.org/3.1.0/d2/d8d/classcv_1_1Stitcher.html>

Figura 28 – Diagrama de bloco do processo para construção de um mosaico de imagens.

Fonte: O Autor

escolher entre SURF ou ORB como extrator de característica.

Estimar os parâmetros da câmera utilizando as características das imagens é uma das opções para se construir um MIA, porém quando as imagens são georreferenciadas e com o conhecimento das técnicas descritas nas sessões 2.8, 2.9 e 2.10 é possível substituir o bloco "RANSAC e Homografia" da Figura 28, pois não há a necessidade de estimar os parâmetros de câmera já que são conhecidos os seis graus de liberdade (ângulos de rotação fornecidos pelo sensor IMU e a translação fornecidos pelo GPS) e acrescentar duas técnicas (blocos) novas, a Triangulação e o BA para refinar o encaixe entre as imagens que formam o MIA.

Com os seis graus de liberdade conhecidos, o diagrama de blocos fica melhor representado pela Figura 29.

Figura 29 – Diagrama de blocos para a construção do MIA clássico, sendo que os seis graus de liberdade (ângulos de rotação fornecidos pelo sensor IMU e, a translação fornecidos pelo GPS) são conhecidos.

Fonte: O Autor

4.1 Abordagem para execução deste trabalho

O desenvolvimento da aplicação não segue a abordagem clássica (seguindo os passos descritos pelos autores Joshi e Sinha (2013), (Figura 28).) e também não segue a abordagem apresentada ao final da sessão anterior, no qual os seis graus de liberdade são conhecidos (Figura 29).

Neste trabalho é sugerido uma abordagem simplificada para a construção do MIA baseada na técnica de georreferenciamento direto (2.7). Todo o processo de calibração de câmera (que inclui triangulação e *bundle adjustment*) não foi aplicado na geração do MIA final com o intuito de ganhar tempo ao processar cada imagem. Os passos para a construção do MIA adotados neste trabalho são: montagem do frame (imagem de resolução maior que receberá as imagens que constituirão o MIA), *warping*, *crop*² (corte), extração e combinação das características (ORB), cálculo do modelo de câmera *pinhole* (para realizar

² O algoritmo desenvolvido para realizar o corte das imagens foi baseado em uma dúvida publicada no StackOverFlow há 4 anos e 8 meses e está disponível em: <https://stackoverflow.com/questions/24507593/how-to-draw-inscribed-rectangle-in-fish-eye-corrected-image-using-opencv>.

ajustes nas coordenadas GPS da imagem), cálculo da derivada da distância [em pixels] entre os pontos chaves das imagens que se combinam com a finalidade de que fiquem na mesma posição ao serem colados ao frame e o último passo é a colagem da imagem ao frame na posição calculada. Esse processo é repetido para todas as imagens do conjunto que pertence ao MIA final. A Figura 30 esboça o novo diagrama de blocos para a abordagem deste trabalho.

Figura 30 – Diagrama de blocos da abordagem adotada nesse trabalho.

Fonte: O Autor

5 RESULTADOS E DISCUSSÕES

Este capítulo tem como objetivo apresentar o resultado alcançado e os principais desafios superados na execução deste trabalho. Para ter como base de comparação, os MIAs apresentados neste trabalho são comparados aos MIAs gerado por um software comercial chamado "Agisoft Photoscan"¹ (versão 1.2.5.2614) executado em linux (Ubuntu). Este é muito utilizado para gerar ortomosaicos, modelos digitas de superfícies, nuvens de pontos entre outros. Esse software comercial segue o processo clássico para construção do MIA (apresentado na Figura 29) gerando o resultado final após algumas horas ou até dezenas de horas dependendo da configuração de *hardware* do computador que executa esse *software*, da quantidade de imagens que irá compor o MIA e dos parâmetros passado para ele.

5.1 Resultados e comparações

As Figuras 31 e 33 são os resultados gerados pela aplicação desenvolvida neste trabalho e as Figuras 32 e 34 são os MIAs gerados pelo "Agisoft Photoscan" que foram utilizados como *benchmark*.

No primeiro conjunto de imagens, que forma a Figura 31 (vide MIA1 na tabela 3), a execução das técnicas de extração e combinação de características se manteve íntegro, onde todas a imagens possuíam pontos chaves combinados com a próxima imagem que iria ser inserida ao frame. Os encaixes das imagens não são precisos porque ao contrário do que se pensava no início deste trabalho, os ângulos de rotação *yaw*, *pitch* e *roll* servem apenas para corrigir as coordenadas geográficas (Latitude e Longitude) e não corrige as perspectivas do conteúdo de cada imagem, em outras palavras, há uma distorção geométrica no MIA resultante desta aplicação que pode ser causada pelo deslocamento do VANT, precisão de GPS, entre outros fatores realcionados ao voo. Porém ao aplicar a técnica *warping* sobre esses ângulos é possível deixar próximo a posição que a imagem irá se encaixar.

Para o segundo conjunto de imagens, que forma a Figura 33 (vide MIA2 na tabela 3), o algoritmo se perde ao realizar a comparação dos pontos chaves em um determinado momento. Isso será discutido na Sessão 5.3. Devido a esse fato, o MIA que representa o conjunto de imagens 2 foi construído com uma quantidade menor de imagens, porém os mesmos erros visuais apresentados na construção do MIA 1 podem ser constatados no MIA 2.

O resultado gerado pelo *software* comercial está com as cores mais equalizadas porque um de seus processos é a aplicação da técnica *blanding* (vide 2.6), como indicado

¹ <<http://www.agisoft.com/>>

no diagrama da Figura 29.

Figura 31 – Resultado do MIA gerado pela aplicação desenvolvida neste trabalho utilizando o conjunto de imagens 1.

Fonte: O Autor

Figura 32 – Resultado do MIA gerado pelo "Agisoft Photoscan" utilizando o conjunto de imagens 1.

Fonte: O Autor

Figura 33 – Resultado do MIA gerado pela aplicação desenvolvida neste trabalho utilizando o conjunto de imagens 2.

Fonte: O Autor

Figura 34 – Resultado do MIA gerado pelo "Agisoft Photoscan" utilizando o conjunto de imagens 2.

Fonte: O Autor

5.2 Entendendo os processos *Warping* e *Crop*

A aplicação desenvolvida para esse trabalho utiliza a técnica *Warping* (também utilizada nas abordagens apresentada por Joshi e Sinha (2013) e na apresentada no capítulo anterior, vide Figura 29). Esta técnica manipula a perspectiva das imagens baseando em três graus de liberdade que são adquiridos através do metadado das imagens registrado no momento de aquisição pelo sensor da câmera. Esses dados são os ângulos *yaw*, *pitch* e *roll* (rotação nos eixos z, y e x respectivamente) que informam os ângulos que a câmera estava posicionada no momento em que a imagem foi registrada.

Uma imagem pertencente ao conjunto de imagens 1 (vide MIA1 na tabela 3) foi selecionada para demonstrar as transformações que ocorrem ao aplicar o *Warping*. A imagem possui como graus de liberdade os valores: *yaw* -170,5°, *pitch* -90,0° e *roll* 0,0°;

Aplicando a técnica *Warping*² o resultado obtido é uma imagem rotacionada conforme os ângulos informados e com uma grande área preenchida com pixels de intensidade preta e de resolução igual a 7331 x 7331 (Figura 35). A resolução e o tamanho da área preenchida com pixels de intensidade preta ao final da execução da técnica muda conforme os graus de liberdade.

Figura 35 – Resultado da aplicação da técnica *warping* e área que deverá permanecer (destacada pelo retângulo vermelho) após aplicação da técnica *Crop*.

Fonte: O Autor

² Antes de aplicar a técnica *Warping* é necessário levar em consideração que o ângulo *pitch* deverá possuir um ponto de partida de -90° para os casos onde é utilizado o dispositivo gimbal (utilizado para estabilizar a câmera durante o voo), pelo fato de a câmera no VANT ter que ficar apontando sempre para baixo. Com isso é necessário somar o valor deste ângulo (obtido pelo metadado da imagem) a 90. Para as imagens registradas sem o auxílio do gimbal, o valor fornecido pelo metadado para o ângulo *pitch* é o que deve ser utilizado.

Seguindo o diagrama de blocos apresentado no capítulo anterior (Figura 30), a imagem resultante é cortada para que ela fique somente com a área de imagem. Em outras palavras, é removido parte da imagem juntamente com a área de intensidade preta. A Figura 35 mostra a área que deverá permanecer ao realizar o corte da imagem.

Após a aplicação desta técnica o resultado é uma imagem (Figura 36) sem os excessos causados pela técnica anterior. A resolução da imagem resultante é de 3648 x 2433 pixels. É muito importante enfatizar que a resolução final da imagem irá mudar conforme os valores dos graus de liberdade.

Figura 36 – Imagem resultante após aplicação das técnicas *warping* e *crop*.

Fonte: O Autor

5.3 Análise dos pontos chave combinados

Após os processos de *warping* e *crop* é possível extrair as características das imagens.

Na construção do MIA é de extrema importância a utilização de um algoritmo a que seja leve e rápido para extração das características quando realizar o processamento em um sistema embarcado, pois este possui um desempenho inferior se comparado ao desempenho de um *Desktop*. Por esse motivo o algoritmo escolhido para realizar essa função e que possui desempenho melhor para sistemas embarcados a um bom tempo é o algoritmo ORB.

Após realizar a combinação entre os pontos chave de duas imagens, nota-se que há uma pequena divergência com relação a suas posições e que os pontos correspondentes estão próximos, porém não no mesmo local (vide Figura 37).

Figura 37 – Comparação entre as posições de pontos chave combinados. O ponto branco nas imagens é a posição exata dos pontos chave que se combinam.

Fonte: O Autor

Há dois fatores que podem estar relacionados a este pequeno erro:

- **perspectivas diferentes:** o VANT registra as imagens em diferentes ângulos e posições fazendo com que haja uma diferença no conteúdo (geometria) da imagem mesmo que elas retratem a mesma cena. Ex.: na Figura 37, a imagem 2 está um pouco mais esticada horizontalmente se comparada a imagem 1.
- **precisão do algoritmo ORB:** apesar do algoritmo ORB ser o mais rápido e o mais indicado para utilização em sistemas embarcados, ele possui o grau de confiança de seus pontos chaves menor se comparado aos algoritmos SIFT e SURF. Tendo um grau menor de confiança, ao realizar a combinação dos pontos chaves torna-se visível o pequeno erro.

Ao analisar a colagem das imagens ao frame, verifica que mesmo após a correção das distorções causadas pelo voo, utilizando a técnica *warping* e os metadados de rotação (*yaw*, *pitch* e *roll*), ainda há distorções no conteúdo da imagem (vide Figura 38). Por mais que as combinações ocorreram corretamente, seria necessário aplicar as técnicas de triangulação e de *bundle adjustment* para corrigir os ângulos da rotação e a perspectiva do conteúdo das imagens para um encaixe preciso;

Figura 38 – Mesmo após a correção das distorções causadas pelo voo, ainda há distorção no conteúdo das imagens.

Fonte: O Autor

Analizando visualmente as combinações dos pontos chave, em algumas imagens são encontradas dificuldades para encontrar uma combinação entre eles. Na maioria das vezes é possível realizar essas combinações de forma fácil, como apresentado na Figura 39 na qual é possível identificar visualmente que o encaixe acontecerá na posição correspondente ao escolher qualquer uma das combinações. Porém há imagens que a combinação foi realizada com os pontos chaves incorretos e isso faz com que o algoritmo se perca, realizando a colagem da imagem na posição errada (vide Figura 40).

Figura 39 – Combinações dos pontos chaves onde é possível identificar que o encaixe acontecerá na posição correspondente.

Fonte: O Autor

Figura 40 – Combinações dos pontos chaves onde é possível identificar que o encaixe não acontecerá na posição correspondente.

Fonte: O Autor

Isso levanta a discussão se o algoritmo de extração de características ORB é aconselhável para aplicações em imagens agrícolas, pois em muitas imagens acontece esse processo de combinar os pontos chave erroneamente.

5.4 Matriz intrínseca

Essa fórmula matemática (equação 2.6), utilizada para transformar as coordenadas de referência da câmera para os pontos (*pixels*) na imagem, é a que mais apresenta dificuldades para generalizar os seus parâmetros para os diversos tipos de câmeras. O simples fato de estar contido o tamanho do sensor (s_x e s_y) e a distância focal (f) na fórmula modifica todo o sistema, pois ao criar um novo frame para abrigar o MIA resultante esses valores são alterados, porque o frame tem que visualizar o resultado final como uma única imagem. Então ao utilizar câmeras diferentes para construir o MIA, será necessário reconstruir (recalcular) essa matriz para que imagem seja posicionada ao seu destino no frame.

Para o conjunto que formou o MIA 1 (Figura 31) a estrutura criada para receber os parâmetros que compoem a matriz intrínseca ficou com os seguintes valores:

```
intrinsic.focalLength = 15000.0f;
intrinsic.sx = 2.0f;
intrinsic.sy = conf.getGSD() * 1.5f;
intrinsic.ox = imgFinal.size().width - (2 * conf.getImgWidth());
intrinsic.oy = conf.getImgHeigh();
```

Para o conjunto que formou o MIA 2 (Figura 33) a estrutura criada para receber os parâmetros que compoem a matriz intrínseca ficou com os seguintes valores:

```
float aspectRatio = conf.getCameraSensorHeigh() / conf.getCameraSensorWidth();

intrinsic.focalLength = 100.0f / conf.getGSD() * 3.1415;
intrinsic.sx = conf.getCameraSensorWidth() / 1000;
```

```

intrinsic.sy = -conf.getCameraSensorHeigth() / 1000 * (1 + aspectRatio);
intrinsic.ox = conf.getImgWidth();
intrinsic.oy = conf.getImgHeigth();

```

A distância focal é calculada para saber quantos pixels representa 1 (um) metro na imagem. O valor do numerando irá depender da unidade que se encontra o *Ground Sample Distance* (GSD³) que é o denominador. Entretanto ao calcular a distância focal do MIA 2 foi preciso multiplicar essa conversão por π para que a imagem se encaixe mais próximo de seu destino no frame.

Outro ponto importante a ser ressaltado na matriz intrínseca é a origem do MIA (*ox* e *oy*) que mudará de acordo com o voo realizado pelo VANT. Este indicará se o MIA deverá ter origem no centro, mais a direita ou mais a esquerda e mais ao topo ou mais em baixo no frame. É preciso analisar essas informações antes de realizar a construção, e ela pode ser feita com base nas coordenadas geográficas contidas no metadado de cada imagem. A Figura 41 mostra os voos realizados para registrar os conjuntos de imagens que compoem os MIAs 1 e 2.

Figura 41 – Análise de como o voo foi realizado pelo VANT para identificar a origem de construção do MIA.

Fonte: o Autor

5.5 O OpenCV

A placa Jetson TK1 da Nvidia já vem com uma versão de OpenCV chamada de "OpenCV4Tegra"⁴. Entretanto, essa versão de OpenCV é customizada e enxuta, no qual não possui todas as funções (ex.: *warping*, ORB, ...) do OpenCV se comparada com a versão original. Dado esses fatos, foram instalados duas versões do OpenCV a **4.0.0**

³ O GSD é uma unidade de terreno (dado em metros, centímetros, milímetros, ...) representado por um pixel.

⁴ <http://developer.download.nvidia.com/embedded/L4T/r23_Release_v1.0/OpenCV4Tegra-2.4.12.3-README.txt>

(versão mais recente) e a **3.1.0**. A versão **4.0.0** foi instalada em um notebook⁵ com GPU da NVidia e produziu os resultados apresentados nas Figuras 31 e 33. A versão **3.1.0** foi instalada na placa Jetson TK1 (versões acima desta não foi possível compilar diretamente na placa), pois não foi possível utilizar o OpenCV4Tegra. Os resultados gerados pela versão 3.1.0 (executado na placa) diverge dos resultados gerados pela versão 4.0.0 (executado no notebook).

Os resultados gerados a partir das execuções na placa foram:

Figura 42 – Construção do MIA 1 utilizando o algoritmo desenvolvido neste trabalho: a Figura 42a foi construído pela placa Jetson TK1; a Figura 42b foi construída pelo notebook

Fonte: O Autor

Figura 43 – Construção do MIA 2 utilizando o algoritmo desenvolvido neste trabalho: a Figura 43a foi construído pela placa Jetson TK1; a Figura 43b foi construída pelo notebook

Fonte: O Autor

⁵ Vide configuração do notebook em: <<https://www.dell.com/pt-br/shop/cty/pdp/spd/inspiron-15-7567-laptop>> (Dell Inspiron 15 7567)

Utilizando a versão 3.1.0 do OpenCV na placa, o processo de extração e combinação de características se perde e realiza combinações erroneamente como pode ser visto nas Figuras 42a e 43a e o encaixe fica ainda mais impreciso.

5.6 Tempo de processamento

Analizando a qualidade dos MIAs criados tanto pelo "Agisoft Photoscan" quanto pela aplicação desenvolvida neste trabalho, é indiscutível a superioridade do resultado gerado pelo *software* comercial. Com a intenção de mensurar o desempenho da aplicação e comparar com o desempenho do *software* comercial, os dois MIAs foram construídos por ambos. Como a aplicação para o MIA 2 consegue processar uma quantidade menor de imagens antes de se perder, para realizar o teste de desempenho (tempo), foi igualado a quantidade de imagens para ambos os conjuntos.

O *software* comercial, sendo executado no mesmo notebook com a versão do OpenCV 4.0.0, levou **3 horas e 35 minutos** para finalizar todo o processo e gerar o MIA 1 que as 291 imagens apresentam. O passo a passo para gerar o MIA pelo "Agisoft Photoscan" é descrito abaixo:

- Alinhar as fotos (*Align Photos*)
- Gerar Nuvem Densa de Pontos (*Build Dense Cloud*)
- Gerar as Malhas (*Build Mesh*)
- Gerar Texturas (*Build Texture*)
- Gerar Modelo Digital de Elevação (*Build Digital Elevation Models (DEM)*)
- Gerar Ortomosaico (*Build Orthomosaic*)

Analizando os itens acima, nota-se que o *software* entrega muito mais coisas do que somente o MIA.

Para o MIA 2, seguindo os mesmos passos e com os mesmos entregáveis, levou **37 minutos** para as 88 imagens.

Voltando os olhares para a aplicação desenvolvida neste trabalho, o desempenho foi medido tanto no notebook quanto na placa Jetson TK1. A tabela 6 apresenta a comparação do tempo médio levado por cada imagem em cada processo apresentado na Figura 30 e o tempo total para gerar o MIA final.

Tabela 6 – Tabela de comparação do tempo médio levado por cada imagem em cada processo apresentado na Figura 30.

Unidade	Medição	Notebook		Jetson TK1	
		OpenCV 4.0.0		OpenCV 3.1.0	
		Conjunto 1	Conjunto 2	Conjunto 1	Conjunto 2
ms	Carrega Metadado e Configuração	200,637	11,226	1.170,020	118,392
	Abre a Imagem e Warping	199,166	303,136	3.282,810	5.048,730
	Crop	403,834	1.576,140	2.182,450	9.051,640
	Extração e Combinação de Características	246,913	169,795	2.324,250	1.500,560
	Cola no Frame	3,823	4,387	10,718	13,226
	Tempo Médio por Imagem	1.054,373	2.064,684	8.970,248	15.732,548
s	Tempo Total MIA	252,130	181,569	2.285,820	1.379,030
mm:ss		04:12,13	03:01,57	38:05,82	22:59,03

Fonte: O Autor

O tempo de execução da aplicação para construção do MIA 1 (291 imagens de dimensão 4000x3000 pixels) é bom para o método escolhido, porém o resultado não é satisfatório com relação ao conteúdo do MIA executado tanto no notebook quanto na Jetson TK1. Para a construção do MIA 2 (88 imagens de dimensão 5472x3648 pixels) o desempenho não é bom quando executado na placa Jetson TK1. O tempo de construção da aplicação, executada na placa, é apenas **1.61** vezes menor que o tempo levado pelo "Agisoft Photoscan".

6 CONCLUSÃO

Este trabalho apresentou uma maneira de construir um Mosaico de Imagens Aéreas (MIA) baseada a técnica de georreferenciamento direto (vide Sessão 2.7), que é uma forma mais simples do que a abordagem clássica apresentada no artigo de Joshi e Sinha (2013).

A implementação do software para execução deste trabalho baseou-se nas técnicas apresentadas na Figura 30 que são: *Warping*, *Crop* (recorte das imagens), Extração e Combinação de Características utilizando o algoritmo ORB, Cálculo do modelo de Câmera Pinhole, Cálculo da Derivada da Distância e Colagem da Imagem ao Frame. Esta aplicação foi executada em dois cenários diferentes: notebook¹ e na placa Jetson TK1. Este possuía a versão do OpenCV 3.1.0 (versão mais recente que foi possível compilar na Jetson TK1) e aquele a versão do OpenCV foi a 4.0.0.

Os MIAs gerados pela aplicação desenvolvida neste trabalho (Figuras 42 e 43 têm como principal falha a perspectiva. Mesmo utilizando os ângulos capturados no momento do registro da imagem, a perspectiva do conteúdo (geometria) da imagem não é corrigida pela técnica *warping* utilizando somente esses valores (vide Figura 38). Esses ângulos servem para corrigir as coordenadas GPS que foram registradas no momento de captura de cada imagem e posicioná-las próximas ao encaixe. Para realizar as correções de perspectivas no conteúdo das imagens é necessário aplicar as técnicas: Triangulação e *Bundle Adjustment*, apresentadas nas Sessões 2.9 e 2.10 respectivamente.

Mesmo com as perspectivas diferentes, ao realizar a combinação das características extraídas com a técnica ORB (técnica apresentada na Sessão 2.2.4) nota-se que o posicionamento do ponto chave fica deslocado nas imagens que se encaixam (vide Figura 37). Em alguns casos, a combinação dos pontos chave é realizada erroneamente gerando um problema maior no encaixe da imagem ao frame (vide Figura 40). Isso faz com que a aplicação se perca e cole a imagem na posição errada (vide Figuras 42a e 43a).

Outro ponto de bastante relevância é a matriz intrínseca (vide Sessão 2.8.1). O simples fato de criar um novo frame para receber as imagens que irão compor o MIA, já causa uma alteração nos parâmetros desta matriz, pois a distância focal (f), pontos principais (ox e oy) e as dimensões do sensor (sx e sy) serão diferentes dos valores informados pelo fabricante da câmera utilizada para capturar as imagens. Para os pontos principais é necessário analisar previamente o percurso realizado pelo VANT para identificar onde iniciou a captura das imagens.

Apesar das pequenas falhas apresentadas no conteúdo (geometria) dos MIAs

¹ Vide configuração do notebook em: <<https://www.dell.com/pt-br/shop/cty/pdp/spd/inspiron-15-7567-laptop>> (Dell Inspiron 15 7567)

resultantes da aplicação desenvolvida neste trabalho, o tempo para construção dos MIAs são menores se comparados com o tempo levado pelo *software* comercial, utilizado para comparação. O ganho de tempo pode chegar até 51 vezes utilizando esta aplicação para construção do MIA tranando-se um grande potencial para ser embarcada em um VANT e o processo ser realizado em tempo real.

REFERÊNCIAS

ADEL, E.; ELMOGY, M.; ELBAKRY, H. Real time image mosaicing system based on feature extraction techniques. In: **2014 9th International Conference on Computer Engineering Systems (ICCES)**. [S.l.: s.n.], 2014. p. 339–345.

_____. Image stitching system based on orb feature-based technique and compensation blending. **Image**, Citeseer, v. 6, n. 9, 2015.

AGRICULTURA de precisão. I: Wikipédia: a enciclopédia livre, 2017. Disponível em: <https://pt.wikipedia.org/wiki/Agricultura_de_precis%C3%A3o>. Acesso em: 24 fev. 2018.

ALIADOS a Drones, Softwares são direcionados para a agricultura de precisão. Senar, 2015. Disponível em: <<http://www.senar.org.br/agricultura-precisao/aliados-a-drones-softwares-sao-direcionados-para-agricultura-de-precisao/>>. Acesso em: 25 out. 2018.

ARM. **NEON**. [s.n.], 2017. Disponível em: <<https://www.arm.com/products/processors/technologies/neon.php>>. Acesso em: 29 mar. 2018.

BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. In: _____. **Computer Vision – ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 404–417. ISBN 978-3-540-33833-8. Disponível em: <http://dx.doi.org/10.1007/11744023_32>. Acesso em: 24 fev. 2018.

BRASIL. Ministério da Agricultura, Pecuária e Abastecimento. Secretaria de Desenvolvimento Agropecuário e Cooperativismo. **Agricultura de Precisão**. 2.. ed. Brasília: Mapa/ACS, 2011. (Boletim técnico).

BROWN, M.; LOWE, D. G. Automatic panoramic image stitching using invariant features. **International Journal of Computer Vision**, v. 74, n. 1, p. 59–73, 2007. ISSN 1573-1405. Disponível em: <<http://dx.doi.org/10.1007/s11263-006-0002-3>>. Acesso em: 24 fev. 2018.

CAMPOS, M. F. M.; Câmara Neto, V. F. d. **DCC884: visão Computacional - calibração de câmeras**. Belo Horizonte: UFMG, 2007. Power Point.

CANDIDO, L. R.; LUPPE, M.; JORGE, L. A. d. C. **Aerial Image Mosaics Construction Using Heterogeneous Computing for Agricultural Applications**. Lisbon, Portugal: BIONATURE 2016 the Seventh International Conference on Bioenvironment, Biodiversity and Renewable Energies : ENVIROSENS 2016 : the International Symposium on Remote Sensing for Climate and Earth Monitoring : June 26-30, 2016, Lisbon, Portugal, 2016. 22–23 p. ISBN 9781612084893. Disponível em: <https://www.thinkmind.org/index.php?view=article{&}articleid=bionature{__}2016{__}2>. Acesso em: 24 fev. 2018.

CHE, S. et al. Rodinia: A benchmark suite for heterogeneous computing. In: **2009 IEEE International Symposium on Workload Characterization (IISWC)**. [S.l.: s.n.], 2009. p. 44–54.

CHU, P. P. **FPGA prototyping by VHDL examples: Xilinx Spartan-3 version**. [S.l.]: John Wiley & Sons, 2011.

CORNER detection: Harris corner and Shi-Tomasi corner detection. Harris..., 2017. Disponível em: <<http://prog3.com/sbdlm/blog/xw20084898/article/details/21180729>>. Acesso em: 17 jan. 2018.

CURVELLO, A. M. de L. **Proposta para aceleração de desempenho de algoritmos de visão computacional em sistemas embarcados**. 2016. Dissertação (Mestrado em Processamento de Sinais de Instrumentação) — Escola de Engenharia de São Carlos, University of São Paulo, São Carlos, 2016. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/18/18152/tde-03082016-164741/>>. Acesso em: 2018-01-17.

DONG, J.; SOATTO, S. Domain-size pooling in local descriptors: Dsp-sift. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 5097–5106. ISSN 1063-6919.

EFROS, A. **Image blending: 15-463 – computational photography**. [S.l.]: [S.l.:s.n.], 2010. Power Point.

EMBRAER entrega cinco primeiros aviões Ipanema 203. Globo Rural, 2016. Disponível em: <<https://revistagloborural.globo.com/Noticias/Agricultura/noticia/2016/01/embraer-entrega-cinco-primeiros-avioes-ipanema-203.html>>. Acesso em: 25 out. 2018.

EMBRAPA e Qualcomm se unem para levar drones para a fazenda. Revista Pequenas Empresas e Grandes Negócios, 2016. Disponível em: <<https://revistapegn.globo.com/Noticias/noticia/2016/06/embrapa-e-qualcomm-se-unem-para-levar-drones-para-fazenda.html>>. Acesso em: 18 out. 2018.

FILHO, L. E. Calibração de um sistema fotogramétrico multi-câmeras para uso em aplicações de mapeamento com georreferenciamento direto. 2017.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. **Commun. ACM**, ACM, New York, NY, USA, v. 24, n. 6, p. 381–395, jun. 1981. ISSN 0001-0782. Disponível em: <<http://doi.acm.org.ez67.periodicos.capes.gov.br/10.1145/358669.358692>>. Acesso em: 24 fev. 2018.

FORTE, C. H. V. **Processadores ARM: visão geral e aplicações**. São José do Rio Preto: Grupo de Sistemas Paralelos e Distribuídos(GSPD), Departamento de Ciência da Computação e Estatística, Universidade Estadual Paulista "Júlio de Mesquita Filho"(UNESP), 2015. Disponível em: <https://semac.sjrp.unesp.br/~aleardo/cursos/arqcomp/Semin_ARM.pdf>. Acesso em: 25 jan. 2018.

GAO, Y.; ZHANG, P. A survey of homogeneous and heterogeneous system architectures in high performance computing. In: **2016 IEEE International Conference on Smart Cloud (SmartCloud)**. [S.l.: s.n.], 2016. p. 170–175.

GASTER, B. et al. **Heterogeneous Computing with OpenCL: Revised OpenCL 1**. [S.l.]: Newnes, 2012.

- GLASBEY, C. A.; MARDIA, K. V. A review of image-warping methods. **Journal of Applied Statistics**, v. 25, n. 2, p. 155–171, 1998. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/02664769823151>. Acesso em: 24 fev. 2018.
- HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: **In Proc. of Fourth Alvey Vision Conference**. [S.l.: s.n.], 1988. p. 147–151.
- HARTLEY, R. I.; STURM, P. Triangulation. **Computer Vision and Image Understanding**, v. 68, n. 2, p. 146 – 157, 1997. ISSN 1077-3142. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1077314297905476>. Acesso em: 24 fev. 2018.
- IMAGE Warping: CSE 399b, spring 07 computer vision. [s.n.], 2018. Disponível em: <http://www.seas.upenn.edu/~cse399b/Lectures/CSE399b-07-warping.pdf>. Acesso em: 26 fev. 2018.
- INAMASU, R. Y.; BERNARDI, A. C. de C. Agricultura de precisão. **BERNARDI, ACC et al (editores técnicos). Agricultura de precisão: resultados de um novo olhar. Brasília: Embrapa**, 2014.
- INFORCE 6540TM Single Board Computer (SBC). Inforce..., 2016. Disponível em: <http://www.inforcecomputing.com/products/single-board-computers-sbc/qualcomm-snapdragon-805-inforce-6540-sbc>. Acesso em: 17 jan. 2018.
- INTRO to programmable logic & FPGAs – april 27. 2014. [s.n.], 2014. Disponível em: <https://pumpingstationone.org/tag/fpga/>. Acesso em: 24 fev. 2018.
- INTRODUCTION to CUDA programming. [s.n.], 2017. Disponível em: <https://sites.google.com/site/computationvisualization/programming/cuda/article1>. Acesso em: 24 fev. 2018.
- JOSHI, H.; SINHA, M. K. A survey on image mosaicing techniques. **international Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume**, v. 2, 2013.
- KIRK, D. B.; WEN-MEI, W. H. **Programming massively parallel processors: a hands-on approach**. [S.l.]: Morgan Kaufmann, 2010.
- LANGBRIDGE, J. A. **Professional embedded ARM development**. [S.l.]: John Wiley & Sons, 2013.
- LOURAKIS, M.; ARGYROS, A. **The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm**. [S.l.], 2004.
- LOWE, D. G. Distinctive image features from scale-invariant keypoints. **International Journal of Computer Vision**, v. 60, n. 2, p. 91–110, 2004. ISSN 1573-1405. Disponível em: <http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94>. Acesso em: 24 fev. 2018.
- MACHADO, J. **O que é GPU?** TecMundo, 2012. Disponível em: <https://www.tecmundo.com.br/hardware/1127-o-que-e-gpu-.htm>. Acesso em: 25 jan. 2018.

MAPA do mercado de tecnologia agrícola: mais de 100 startups potencializam o futuro da produção rural e do agronegócio. Equipe BeefPoint, 2017. Disponível em: <<https://www.beefpoint.com.br/mapa-do-mercado-de-tecnologia-agricola-mais-de-100-startups-potencializam-o-futuro-da-producao-rural-e>>. Acesso em: 18 out. 2018.

MODERNIZAÇÃO da agricultura esbarra em escassez de investimento. Folha de S. Paulo, 2018. Disponível em: <<https://www1.folha.uol.com.br/seminariosfolha/2018/03/modernizacao-da-agricultura-esbarra-em-escassez-de-investimento.shtml>>. Acesso em: 27 nov. 2018.

MONSANTO e Microsoft investem em startups no Brasil. Exame, 2016. Disponível em: <<https://exame.abril.com.br/pme/monsanto-e-microsoft-investem-em-startups-no-brasil/>>. Acesso em: 18 out. 2018.

MOUSSA, A.; EL-SHEIMY, N. A fast approach for stitching of aerial images. **ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences**, p. 769–774, 2016.

MUNSHI, A. et al. **OpenCL programming guide**. [S.l.]: Pearson Education, 2011.

NTAWINIGA, F. **CHAPTER 6: 3D Reconstruction**. Université Laval, 2008. Disponível em: <<http://archimede.bibl.ulaval.ca/archimede/fichiers/25229/ch06.html>>. Acesso em: 01 feb. 2018.

NVIDIA Cuda installation guide for Linux: installation and verification on Linux systems. SU-05247-001_v.8.8, 2016. Disponível em: <http://docs.nvidia.com/cuda/pdf/CUDA_Installation_Guide_Linux.pdf>. Acesso em: 23 jan. 2018.

NVIDIA Jetson TK1. Nvidia..., 2017. Disponível em: <<http://www.nvidia.com.br/object/jetson-tk1-embedded-dev-kit-br.html>>. Acesso em: 18 jan. 2018.

OLIVEIRA, C. A. de; AGUIAR, J. A. de; FONTANINI, M. G. S. **Dispositivos lógicos programáveis**. Revisado por Marcelo Wendling. Guaratinguetá: Instituto Técnico Industrial de Guaratinguetá “Professor Carlos Augusto Patrício Amorim”/UNESP, 2017. Disponível em: <<http://www2.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/logica-programavel.pdf>>. Acesso em: 23 jan. 2018.

OYALLON, E.; RABIN, J. An analysis of the surf method. **Image Processing On Line**, v. 5, p. 176–218, 2015.

PELLERIN, D.; THIBAUT, S. **Practical fpga programming in c**. [S.l.]: Prentice Hall Press, 2005.

RASPBERRY Pi 3 Model B. Raspberry..., 2009. Disponível em: <<https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>>. Acesso em: 19 jan. 2018.

RUBLEE, E. et al. Orb: An efficient alternative to sift or surf. In: **2011 International Conference on Computer Vision**. [S.l.: s.n.], 2011. p. 2564–2571. ISSN 1550-5499.

SANDERS, J.; KANDROT, E. **CUDA by example: an introduction to general-purpose GPU programming**. [S.l.]: Addison-Wesley Professional, 2010.

SANTOS, M. C. dos; ROCHA, A. Revisão de conceitos em projeção, homografia, calibração de câmera, geometria epipolar, mapas de profundidade e varredura de planos. 2012.

SOOKNANAN, D. J.; JOSHI, A. Gpu computing using cuda in the deployment of smart grids. In: **2016 SAI Computing Conference (SAI)**. [S.l.: s.n.], 2016. p. 1260–1266.

STRINGHINI, D.; GONÇALVES, R. A.; GOLDMAN, A. Introdução à computação heterogênea. cap. 7. 2013. Disponível em: <<https://pt.scribd.com/document/143655726/Jai-Computacao-Heterogenea-Sbc>>. Acesso em: 24 fev. 2018.

SÜNDERHAUF, N. et al. Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle. In: _____. **Autonome Mobile Systeme 2005: 19. Fachgespräch Stuttgart, 8./9. Dezember 2005**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 157–163. ISBN 978-3-540-30292-6. Disponível em: <http://dx.doi.org/10.1007/3-540-30292-1_20>. Acesso em: 24 fev. 2018.

THE Bundle Adjustment (BA) problem. Bundle..., 2012. Disponível em: <<https://mappingignorance.org/2012/12/17/to-filter-or-not-to-filter-thats-the-question-in-3d-scene-reconstruction/jlb1/>>. Acesso em: 24 mar. 2018.

THE Parallella Board. The Parallella..., 2014. Disponível em: <<https://www.parallella.org/>>. Acesso em: 18 jan. 2018.

TRUCCO, E.; VERRI, A. **Introductory techniques for 3-D computer vision**. [S.l.]: Prentice Hall Englewood Cliffs, 1998. v. 201.

UBAL, R. et al. Multi2sim: A simulation framework for cpu-gpu computing. In: **Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques**. New York, NY, USA: ACM, 2012. (PACT '12), p. 335–344. ISBN 978-1-4503-1182-3. Disponível em: <<http://doi.acm.org.ez67.periodicos.capes.gov.br/10.1145/2370816.2370865>>. Acesso em: 24 fev. 2018.