UNIVERSIDADE ESTADUAL DE PONTA GROSSA
SETOR DE CIÊNCIAS AGRÁRIAS E DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

PEDRO HENRIQUE SOARES DE ALMEIDA

AVALIAÇÃO DE MÉTODOS DE MOSAICO DE IMAGENS APLICADOS EM IMAGENS AGRÍCOLAS OBTIDAS POR MEIO DE RPA

PONTA GROSSA

2018

PEDRO HENRIQUE SOARES DE ALMEIDA

AVALIAÇÃO DE MÉTODOS DE MOSAICO DE IMAGENS APLICADOS EM IMAGENS AGRÍCOLAS OBTIDAS POR MEIO DE RPA

Dissertação apresentada para obtenção do título de mestre na Universidade Estadual de Ponta Grossa, Área de Computação para Tecnologias em Agricultura.

Orientador: Prof. Dr. Luciano José Senger

PONTA GROSSA

2018

Ficha Catalográfica
Elaborada pelo Setor de Tratamento da Informação BICEN/UEPG

Almeida, Pedro Henrique Soares de
A447 Avaliação de métodos de mosaico de imagens aplicados em imagens agrícolas obtidas por meio de RPA/ Pedro Henrique Soares de Almeida. Ponta Grossa, 2018.
66f.

Dissertação (Mestrado em Computação Aplicada - Área de Concentração: Computação para Tecnologias em Agricultura), Universidade Estadual de Ponta Grossa.
Orientador: Prof. Dr. Luciano José Senger.

1.Mosaico de imagens. 2.Imagens agrícolas. 3.RPA. 4.Característica de baixo nível. 5.OpenCV. I.Senger, Luciano José. II. Universidade Estadual de Ponta Grossa. Mestrado em Computação Aplicada. III. T.

CDD: 006.3

TERMO DE APROVAÇÃO

Pedro Henrique Soares de Almeida

"AVALIAÇÃO DE MÉTODOS DE MOSAICO DE IMAGENS APLICADOS EM IMAGENS AGRÍCOLAS OBTIDAS POR MEIO DE RPA"

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre no Programa de Pós-Graduação em Computação Aplicada da Universidade Estadual de Ponta Grossa, pela seguinte banca examinadora:

Prof. Dr. Luciano José Senger
UEPG

Prof. Dr. Erikson Freitas de Morais
UTFPR

Profª. Drª. Alaine Margarete Guimarães
UEPG

Ponta Grossa, 15 de maio de 2018.

RESUMO

O mosaico de imagens é o alinhamento de múltiplas imagens em composições maiores que representam partes de uma cena 3D. Diversos algoritmos de mosaico de imagens foram propostos nas últimas duas décadas. Ao mesmo tempo, o advento contínuo de novos métodos de mosaico torna muito difícil escolher um algoritmo apropriado para uma finalidade específica. Este trabalho teve por objetivo avaliar métodos de mosaico baseados em característica de baixo nível utilizando imagens agrícolas obtidas por meio de Aeronave Remotamente Pilotada (RPA). Algoritmos detectores de característica de baixo nível podem ser invariantes à escala e rotação, dentre outras transformações que comumente ocorrem em imagens agrícolas obtidas por meio de RPA. O detector de cantos de Harris, detector de cantos FAST, detector de característica SIFT e detector SURF foram avaliados de acordo com o desempenho computacional e a qualidade do mosaico gerado. Para avaliar o desempenho, foram levados em consideração fatores como a média de características detectadas por imagem, o número de imagens utilizadas para compor o mosaico e o tempo de processamento (tempo de usuário ou *user time*). Para avaliar a qualidade, os mosaicos gerados pelos métodos foram utilizados para estimar a severidade da ferrugem asiática da soja e uma comparação com o *software* comercial *Pix4Dmapper* foi realizada. Em relação à qualidade, não houve diferença significativa e todos os métodos demonstraram estar no mesmo patamar. O detector SURF, dentre todos os métodos, obteve o pior desempenho utilizando, em média, apenas 33,1% das imagens de entrada para compor os mosaicos. O detector de cantos de Harris mostrou-se como a solução mais rápida, chegando a ser 7,27% mais rápido para compor o mosaico. Porém, em seu último mosaico gerado, o aproveitamento das imagens de entrada foi pobre: apenas 52%. O detector de cantos FAST obteve o melhor aproveitamento das imagens de entrada, porém, descontinuidades significativas de objetos ocorreram em seu último mosaico gerado. Além disso, obteve um tempo de processamento consideravelmente superior ao dos demais métodos, chegando a ser 6,42 vezes mais lento para compor o mosaico. O detector de característica SIFT obteve o segundo melhor tempo de processamento e o segundo melhor aproveitamento das imagens de entrada, sem apresentar problemas de descontinuidades de objetos. Portanto, mostrou-se como o método mais adequado para imagens agrícolas obtidas por meio de RPA.

Palavras-chave: Mosaico de imagens. Imagens agrícolas. RPA. Característica de baixo nível. OpenCV. Severidade da ferrugem asiática.

ABSTRACT

Image mosaicing is the alignment of multiple images into larger compositions which represent portions of a 3D scene. A number of image mosaicing algorithms have been proposed over the last two decades. At the same time, the continuous advent of new mosaicing methods in recent years makes it really difficult to choose an appropriate mosaicing algorithm for a specific purpose. This study aimed to evaluate low level feature-based mosaicing methods using agricultural images obtained by Remotely Piloted Aircraft (RPA). Low-level feature detecting algorithms can be invariant to scale and rotation, among other transformations that commonly occur in agricultural images obtained by RPA. Harris corner detector, FAST corner detector, SIFT feature detector and SURF detector were evaluated according to the computational performance and the quality of the generated mosaic. To evaluate computational performance, were taken into account factors such as the detected features average per image, the number of images used to compose the mosaic and the processing time (user time). To evaluate quality, the mosaics generated by each method were used to estimate the Asian soybean rust severity and a comparison with the commercial software *Pix4Dmapper* was performed. Regarding quality, there was no significant difference and all methods proved to be on the same level. SURF detector, among all methods, got the worst performance using, on average, only 33.1% of the input images to compose the mosaics. Harris corner detector proved to be the fastest solution, becoming 7.27% faster to compose the mosaic. However, in its final mosaic, the use of the input images was poor: only 52%. FAST corner detector had the best utilization of the input images, however, significant discontinuities of objects occurred in its final mosaic. In addition, it had a considerably longer processing time than the other methods, becoming 6.42 times slower to compose the mosaic. SIFT feature detector had the second best processing time and the second best utilization of the input images, without presenting object discontinuities problems. Therefore, presented itself as the most suitable method for agricultural images obtained by RPA.

Keywords: Image mosaicing. Agricultural images. RPA. Low-level feature. OpenCV. Asian rust severity.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AI	*Artificial Intelligence*
AKAZE	*Accelerated KAZE*
ANOVA	*Analysis of Variance*
BRIEF	*Binary Robust Independent Elementary Features*
BRISK	*Binary Robust Invariant Scalable Keypoints*
BSD	*Berkeley Software Distribution*
CRF	*Corner Response Function*
CUDA	*Compute Unified Device Architecture*
DoG	*Difference of Gaussians*
EM	Eletromagnético
FAST	*Features from Accelerated Segment Test*
FREAK	*Fast Retina Keypoint*
GCP	*Ground Control Point*
GIS	*Geographic Information System*
GLSD	*Grapevine Leaf Stripe Disease*
GNSS	*Global Navigation Satellite System*
GNU	*GNU's Not Unix*
GPU	*Graphic Processing Unit*
HDR	*High Dynamic Range*
HLB	*Huanglongbing*
IPP	*Integrated Performance Primitives*
MSD	*Maximal Self-Dissimilarity*
NCC	*Normalized Cross-Correlation*
OpenCL	*Open Computing Language*
OpenCV	*Open Source Computer Vision*
ORB	*Oriented FAST and Rotated BRIEF*
RAM	*Random Access Memory*
RANSAC	*Random Sampling Consensus*
RGB	*Red, Green, Blue*
RMSE	*Root Mean Square Error*
RPA	*Remotely Piloted Aircraft*
SIFT	*Scale-Invariant Feature Transform*
SSD	*Sum of Squared Differences*
SURF	*Speeded Up Robust Features*

SVR	*Support Vector Regression*
TBB	*Threading Building Blocks*

SUMÁRIO

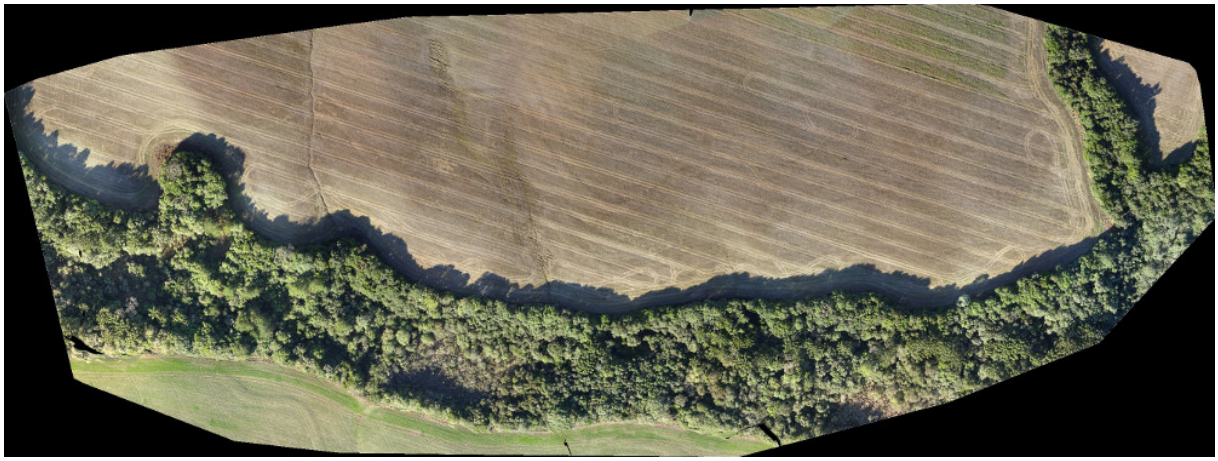
1 INTRODUÇÃO

Um dos motivos responsáveis pela proliferação de Aeronaves Remotamente Pilotadas — do inglês *Remotely Piloted Aircraft* (RPA) — comerciais é sua aplicação na agricultura. Obter frequentemente imagens aéreas de suas fazendas permite aos agricultores tomar decisões relacionadas à aplicação de fertilizantes, irrigação, aplicação de pesticidas e outras práticas agrícolas. Em uma aplicação típica, uma RPA equipada com câmeras diversas sobrevoa o campo coletando imagens georreferenciadas que são utilizadas para construir um mosaico e auxiliar na tomada de decisão agrícola (LI; ISLER, 2016).

O mosaico de imagens é o alinhamento de várias imagens em composições maiores que representam partes de uma cena 3D (CAPEL, 2004). Atualmente, essa área está ganhando interesse na comunidade de pesquisa, por sua importância científica e derivados potenciais em aplicações do mundo real (GHOSH; KAABOUCH, 2016).

Os algoritmos de construção de mosaico existentes contam com a suposição de que o mundo é plano e, portanto, duas imagens que visualizam a mesma área estão relacionadas por homografia (LI; ISLER, 2016). Para reforçar essa hipótese, na prática, as RPAs geralmente voam em altitudes elevadas. Dessa forma, a variação de profundidade, devido a objetos como árvores, se torna irrisória. A Figura 1.1 apresenta um exemplo de mosaico construído a partir de diversas imagens agrícolas obtidas por meio de RPA.

Figura 1.1 – Exemplo de mosaico construído a partir de imagens obtidas por meio de RPA



Fonte: O autor.

Como as alterações de escala e de resolução podem introduzir erros no cálculo do mosaico, o movimento da RPA é restringido para voar a uma altitude fixa com o plano da câmera essencialmente paralelo ao chão. A violação dessas premissas resulta em desalinhamento, distorção de perspectiva ou falha completa na construção do mosaico.

A construção do mosaico envolve diversas etapas de processamento de imagem: Registro, Reprojeção, *Stitching* e Mesclagem. Embora o estado da arte indique avanços nessa área de pesquisa nos últimos anos, o mosaico de imagens continua a ser um desafio devido a fatores como o Registro e a Mesclagem (GHOSH; KAABOUCH, 2016).

O Registro não é apenas uma etapa importante na criação do mosaico, mas também o fundamento do mesmo (GHOSH; KAABOUCH, 2016). Porém, devido à diversidade de imagens a serem registradas e aos diversos tipos de degradação, é improvável conceber um método universal aplicável (e que funcionará satisfatoriamente) a todas as tarefas de Registro (ZITOVÁ; FLUSSER, 2003; FEDOROV *et al.*, 2003). O mesmo aplica-se às tarefas de Mesclagem.

Ao mesmo tempo, o advento contínuo de novos métodos de mosaico torna muito difícil escolher um algoritmo de mosaico apropriado para uma finalidade específica (GHOSH; KAABOUCH, 2016). Além disso, existe uma variedade em *softwares* de processamento utilizados em mapeamento de RPA, e o mercado muda conforme a popularidade da RPA aumenta.

Pix4Dmapper e *Agisoft PhotoScan* são as duas escolhas pagas mais populares de processamento de fotogrametria[1] e imagens aéreas, com interfaces de usuário relativamente simples e manuais compreensíveis, bem como um histórico estabelecido de uso para aplicações profissionais de mapeamento aéreo. Ambos os programas são atualizados e aprimorados regularmente, à medida que a demanda por mapeamento de RPA e o mercado de *software* de fotogrametria se expandem (KAKAES *et al.*, 2015). No entanto, a fotogrametria paga é cara e pode exigir considerável poder de processamento para operar, o que deve ser considerado em orçamentos de mapeamento. No momento desta escrita (Janeiro de 2018), o preço do *Pix4Dmapper* estava fixado em US$8700 para uma licença completa e poderia ser alugado por US$3500 ao ano[2]. O custo do *Agisoft PhotoScan* estava em US$3499 para a edição profissional, enquanto uma edição padrão menos rica em recursos custava US$179[3].

Dentre as abordagens para construção do mosaico de imagens, em relação aos métodos de Registro, os algoritmos de mosaico podem ser baseados em domínio espacial ou em domínio de frequência. Dentre os de domínio espacial, destacam-se os métodos baseados em característica de baixo nível devido as suas maiores vantagens sobre os outros métodos: sua velocidade, robustez e a possibilidade de criação de imagem panorâmica de uma cena não planar com movimento de câmera irrestrito (ADEL; ELMOGY; ELBAKRY,

[1] Pode ser definida como a ciência de fazer medições a partir de fotografias.
[2] <https://cloud.pix4d.com/store/>.
[3] <http://www.agisoft.com/buy/online-store/>.

2014). Além disso, eles não exigem imagens com grandes áreas de sobreposição para criar o mosaico (GHOSH; KAABOUCH, 2016). Segundo Ghosh & Kaabouch (2016), mosaicos baseados em característica de baixo nível podem ser divididos em quatro classes: mosaicos baseados no detector de cantos de Harris, no detector de cantos *Features from Accelerated Segment Test* (FAST), no detector de característica *Scale-Invariant Feature Transform* (SIFT) e no detector *Speeded Up Robust Features* (SURF).

Algoritmos detectores de característica de baixo nível podem ser invariantes à escala e rotação, dentre outras transformações. Tais transformações são comuns em imagens agrícolas obtidas por meio de RPA, o que poderia indicar a adequabilidade dos métodos baseados em característica de baixo nível para contornar esse tipo de problema.

Uma importante área de aplicação para mosaicos criados a partir de imagens agrícolas obtidas por meio de RPA está na detecção de doenças nas culturas. A quantificação de danos é um ponto chave na definição de qualquer estratégia de controle de doenças (HIKISHIMA *et al.*, 2010). Estimativas visuais tradicionais identificam uma doença com base nos sintomas característicos da doença na planta ou sinais visíveis de um patógeno. A estimativa visual é realizada por especialistas treinados e tem sido objeto de intensa pesquisa e investigação (MAHLEIN, 2016). A fim de automatizar esse processo, técnicas de sensoriamento remoto têm sido empregadas com sucesso em estudos de detecção de doenças.

Nesse contexto, o objetivo deste trabalho foi fazer uma avaliação comparativa entre as quatro classes de métodos de mosaico baseados em característica de baixo nível supracitadas utilizando imagens agrícolas obtidas por meio de RPA. A avaliação foi feita de acordo com o desempenho computacional e a qualidade dos mosaicos gerados pelos métodos.

Para avaliar o desempenho, foram levados em consideração fatores como o número médio de características detectadas por imagem, o número de imagens utilizadas para compor o mosaico e o tempo de processamento (tempo de usuário ou *user time*). Para avaliar a qualidade, os mosaicos gerados pelos métodos foram utilizados para estimar a severidade da ferrugem asiática da soja (*Glycine max*) e uma comparação com o *software* comercial *Pix4Dmapper* foi realizada.

1.1 OBJETIVO GERAL

Fazer uma avaliação comparativa, de acordo com o desempenho computacional e a qualidade do mosaico gerado, entre quatro métodos de mosaico de imagens baseados em característica de baixo nível (mosaico baseado no detector de cantos de Harris, no detector de cantos FAST, no detector de característica SIFT e no detector SURF) utilizando imagens agrícolas obtidas por meio de RPA.

1.2 OBJETIVOS ESPECÍFICOS

- Avaliar o desempenho de acordo com o número médio de características e de *inliers* detectados por imagem, o número de imagens utilizadas para compor o mosaico e o tempo de processamento (tempo de usuário ou *user time*);
- Avaliar a qualidade de acordo com as estimativas da severidade da ferrugem asiática da soja (*Glycine max*) calculadas pelos métodos, comparando-as com a estimativa calculada pelo *software* comercial *Pix4Dmapper*.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em cinco Capítulos. O Capítulo 1 apresenta a motivação, o objetivo geral e os objetivos específicos do trabalho. O Capítulo 2 apresenta um levantamento da literatura relevante relacionada ao trabalho. O Capítulo 3 descreve o processo de aquisição do mosaico de imagens utilizando a biblioteca OpenCV e apresenta o programa computacional implementado para criação de mosaicos. O Capítulo 4 descreve como a avaliação dos métodos de mosaico de imagens foi realizada, além de apresentar os resultados obtidos e a discussão feita com trabalhos relacionados. Por fim, o Capítulo 5 apresenta as considerações finais do trabalho.

2 REVISÃO DA LITERATURA

Este Capítulo apresenta um levantamento da literatura relevante acerca dos temas abordados por este trabalho. A Seção 2.1 conceitua o processamento digital de imagens. A Seção 2.2 introduz a biblioteca de visão computacional e aprendizagem de máquina OpenCV. A Seção 2.3 apresenta informações relacionadas ao mosaico de imagens. A Seção 2.4 aborda os conceitos relacionados aos mosaicos baseados em característica de baixo nível. As Seções 2.4.1, 2.4.2, 2.4.3 e 2.4.4 descrevem cada um dos métodos escolhidos para avaliação. A Seção 2.5 aborda a utilização do sensoriamento remoto na detecção de doenças nas culturas.

2.1 PROCESSAMENTO DIGITAL DE IMAGENS

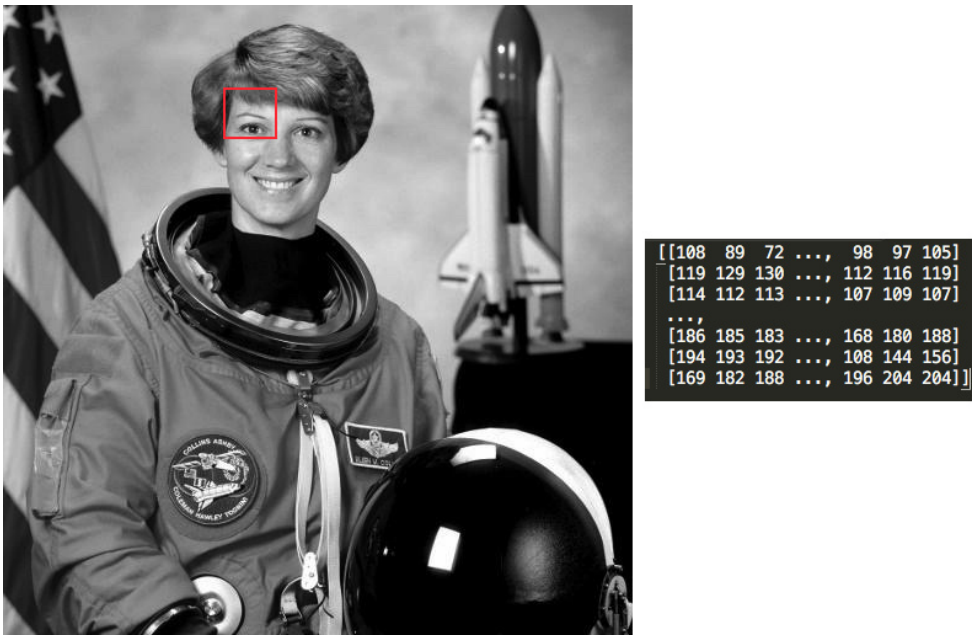
De acordo com Gonzalez & Woods (2017), uma imagem pode ser definida como uma função bidimensional, $f(x, y)$, onde x e y são coordenadas espaciais (plano), e a amplitude de f em qualquer par de coordenadas (x, y) é chamada de *intensidade* ou *nível de cinza* da imagem nesse ponto. Quando x, y e os valores de intensidade de f são quantidades finitas e discretas, pode-se chamar de *imagem digital*. O campo do processamento digital de imagens refere-se ao processamento de imagens digitais por um computador digital. Deve-se notar que uma imagem digital é composta de um número finito de elementos, cada um com localização e valor específicos. Esses elementos são chamados de elementos pictórios, elementos de imagem e pixels. Pixel é o termo mais utilizado para representar os elementos de uma imagem digital.

Uma imagem em um computador é uma matriz bidimensional de números com cada célula na matriz armazenando o(s) valor(es) de pixel(s) correspondente(s) na imagem. A Figura 2.1 é um exemplo de uma matriz de imagem.

A visão é o sentido mais avançado dos seres humanos, de forma que não é de surpreender que as imagens exerçam o papel mais importante na percepção humana. No entanto, ao contrário dos seres humanos, que são limitados à banda visual do espectro Eletromagnético (EM), os aparelhos de processamento de imagens cobrem quase todo o espectro EM, variando de ondas gama a ondas de rádio. Eles podem trabalhar com imagens geradas por fontes que os seres humanos não estão acostumados a associar com imagens, como ultrassom, microscopia eletrônica e imagens geradas por computador. Dessa forma, o processamento digital de imagens abrange um amplo e variado campo de aplicações.

Não existe um acordo geral entre os autores em relação ao ponto em que o

Figura 2.1 – Matriz de imagem (direita), como armazenada em um computador, de uma pequena porção da imagem (esquerda) no quadrado vermelho



Fonte: Kapur (2017).

processamento de imagens termina e outras áreas relacionadas, como a análise de imagens e a visão computacional, começam (GONZALEZ; WOODS, 2017). Algumas vezes, uma distinção é traçada definindo o processamento de imagens como uma disciplina na qual tanto a entrada quanto a saída de um processo são imagens. Gonzalez & Woods (2017) acreditam que essa fronteira é restritiva e, de certa forma, artificial, já que até a tarefa trivial de calcular a intensidade média de uma imagem (que resulta em um único número) não seria considerada uma operação de processamento de imagens. Por outro lado, existem campos como o da visão computacional, cuja meta é utilizar computadores para emular a visão humana, incluindo o aprendizado e a capacidade de fazer inferências e agir com base em informações visuais. Essa área representa um ramo da Inteligência Artificial — do inglês *Artificial Intelligence* (AI) — cujo objetivo é emular a inteligência humana. A área de análise de imagens está situada entre o processamento de imagens e a visão computacional.

Segundo Gonzalez & Woods (2017), não existem limites claros se for considerada uma linha contínua com o processamento de imagens em um extremo e a visão computacional no outro. No entanto, um paradigma útil seria levar em consideração três tipos de processos computacionais nessa linha contínua: processos de níveis baixo, médio e alto. Os processos de nível baixo envolvem operações primitivas, como o pré-processamento de imagens para reduzir o ruído, o realce de contraste e o aguçamento de imagens. Um processo de nível baixo é caracterizado pelo fato de tanto a entrada quanto a saída serem imagens. O

processamento de imagens de nível médio envolve tarefas como a segmentação (separação de uma imagem em regiões ou objetos), a descrição desses objetos para reduzi-los a uma forma adequada para o processamento computacional e a classificação (reconhecimento) de objetos individuais. Um processo de nível médio é caracterizado pelo fato de suas entradas, em geral, serem imagens, mas as saídas são atributos extraídos dessas imagens (isto é, bordas, contornos e a identidade de objetos individuais). Por fim, o processamento de nível alto envolve dar sentido a um conjunto de objetos reconhecidos, como na análise de imagens e, no extremo dessa linha contínua, realizar as funções cognitivas normalmente associadas à visão.

Com base no que foi descrito anteriormente, é possível ver que um ponto lógico de sobreposição entre o processamento e a análise de imagens é a área de reconhecimento de regiões ou objetos individuais em uma imagem. Dessa forma, a essência do processamento de imagens está em utilizar diferentes propriedades de uma imagem, como cor, correlação entre diferentes pixels, posicionamento de objetos e outros detalhes para extrair informações significativas, como bordas, objetos e contornos, formalmente chamados de características da imagem. Essas características podem, então, ser utilizadas em diferentes aplicações de áreas de importante valor social e econômico, como na agricultura, medicina, segurança, dentre inúmeras outras (KAPUR, 2017).

2.2 BIBLIOTECA OPENCV

Originalmente desenvolvida pela Intel[1], *Open Source Computer Vision* (OpenCV)[2] é uma biblioteca multiplataforma livre para processamento de imagens em tempo real que se tornou uma ferramenta padrão para todas as coisas relacionadas à visão computacional. A primeira versão foi lançada em 2000 sob a licença *Berkeley Software Distribution* (BSD) e, desde então, sua funcionalidade foi muito enriquecida pela comunidade científica. Em 2012, a fundação sem fins lucrativos *OpenCV.org* assumiu a tarefa de manter um site de suporte para desenvolvedores e usuários.

OpenCV está disponível para os sistemas operacionais mais populares, como GNU/Linux, Windows, Android e iOS. A primeira implementação foi na linguagem de programação `C`. No entanto, sua popularidade cresceu com sua implementação `C++` a partir da versão 2.0. Atualmente, a biblioteca também possui uma interface completa para outras linguagens de programação, como `Java` e `Python`.

[1] <https://www.intel.com/content/www/us/en/homepage.html>
[2] <https://opencv.org/>

De acordo com García *et al.* (2015), na tentativa de maximizar o desempenho das tarefas intensivas de visão computacional, OpenCV inclui suporte para:

- *Multithreading* em computadores multinúcleo utilizando *Threading Building Blocks* (TBB) — uma biblioteca de *templates* desenvolvida pela Intel;
- Um subconjunto de Primitivas de Desempenho Integradas — do inglês *Integrated Performance Primitives* (IPP) — nos processadores Intel para aumentar o desempenho. Essas primitivas estão disponíveis gratuitamente a partir da versão 3.0 beta;
- Interfaces para processamento na Unidade de Processamento Gráfico — do inglês *Graphic Processing Unit* (GPU) — utilizando *Compute Unified Device Architecture* (CUDA) e *Open Computing Language* (OpenCL).

Além disso, inclui centenas de algoritmos dentro de uma variedade de módulos. Esses módulos são categorizados em dois tipos: os módulos principais e os módulos extras. Os módulos principais são todos os módulos criados e mantidos na comunidade OpenCV, e são parte do pacote padrão fornecido pela biblioteca. Os módulos extras são adaptadores para bibliotecas de terceiros e interfaces necessárias para integrá-las em uma compilação OpenCV (TAZEHKANDI, 2018).

Alguns exemplos de módulos principais (e provavelmente os mais utilizados) são brevemente descritos a seguir (TAZEHKANDI, 2018):

- O módulo `core` contém todas as estruturas básicas, constantes e funções utilizadas por todos os outros módulos OpenCV. Por exemplo, a famosa classe OpenCV `cv::Mat` está definida nesse módulo;
- O módulo `imgproc` contém vários algoritmos diferentes para filtragem de imagens, transformação de imagens e, como o nome sugere, é utilizado para diferentes tarefas de processamento de imagens;
- O módulo `features2d` inclui classes e métodos utilizados para extração e correspondência de características de imagens;
- O módulo `video` contém algoritmos que são utilizados para tópicos como estimativa de movimento, subtração de fundo e rastreamento.

Um exemplo de módulo extra é o `xfeatures2d`. Esse módulo contém algoritmos proprietários como SURF e SIFT, ou outros algoritmos ainda em estado experimental. Os

algoritmos SURF e SIFT foram patenteados e, como tal, seu uso em aplicações comerciais pode estar sujeito a acordos de licenciamento. Essa restrição é uma das razões pelas quais esses algoritmos são encontrados no módulo extra `xfeatures2d` (LAGANIÈRE, 2017).

As aplicações para OpenCV cobrem áreas como segmentação e reconhecimento, conjuntos de ferramentas de recursos 2D e 3D, identificação de objetos, reconhecimento facial, rastreamento de movimento, reconhecimento de gestos, mosaico de imagens, imagens de Alto Alcance Dinâmico — do inglês *High Dynamic Range* (HDR), realidade aumentada, dentre outras. Além disso, para dar suporte a algumas dessas áreas de aplicação, um módulo com funções estatísticas de aprendizagem de máquina está incluído (GARCÍA *et al.*, 2015).

2.3 MOSAICO DE IMAGENS

O mosaico de imagens é o alinhamento de várias imagens em composições maiores que representam partes de uma cena 3D (CAPEL, 2004). Ele pode ser considerado um caso especial de reconstrução de cena em que as imagens são relacionadas somente por homografia planar (GHOSH *et al.*, 2012), que é uma transformação projetiva que mapeia pontos de um plano para outro plano (HARTLEY; ZISSERMAN, 2003). Essa suposição é assegurada se as imagens não apresentam efeitos de paralaxe, ou seja, quando a cena é aproximadamente plana ou quando a câmera gira exclusivamente sobre seu centro óptico (AZZARI; STEFANO; MATTOCCIA, 2008). Utilizando mosaicos, é possível ampliar o campo de visão de uma câmera, preservando a resolução original sem introduzir deformação indesejável da lente (BEVILACQUA; AZZARI, 2007). Além disso, tem havido uma variedade de novas adições às aplicações clássicas de mosaico de imagens que visam principalmente aumentar o campo de visão. A construção do mosaico está encontrando suas práticas em muitas aplicações de computação gráfica e visão computacional, tais como detecção e rastreamento de movimento, localização baseada em mosaico, aprimoramento de resolução e realidade aumentada. Além disso, a compressão de vídeo, indexação de vídeo e estabilização de imagem são algumas das áreas proeminentes onde o mosaico está criando impactos significativos (GHOSH; KAABOUCH, 2016).

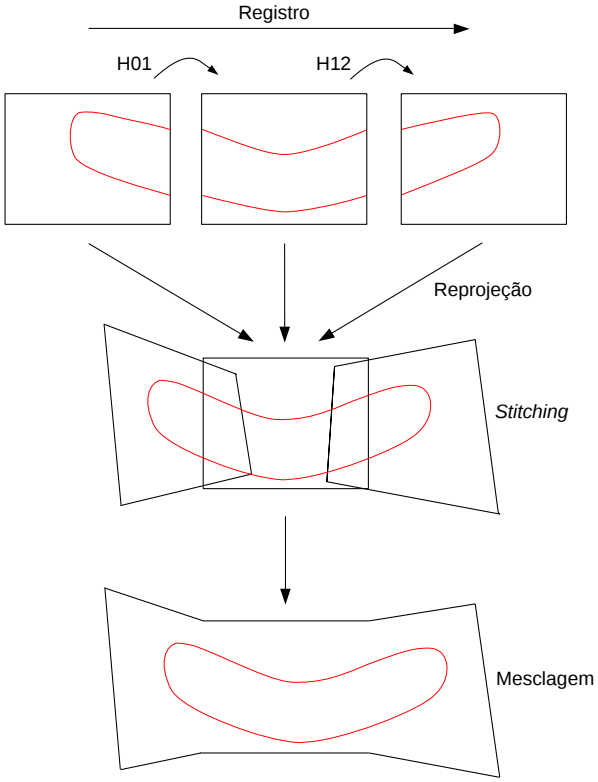
Como ilustrado na Figura 2.2, a construção do mosaico envolve diversas etapas de processamento de imagem: Registro, Reprojeção, *Stitching* e Mesclagem. Cada uma dessas etapas é brevemente descrita a seguir (CAPEL, 2004; GHOSH; KAABOUCH, 2016):

- *Registro*: refere-se ao estabelecimento da correspondência geométrica entre um par

de imagens que descrevem a mesma cena. Para registrar um conjunto de imagens, é necessário estimar as transformações geométricas que alinham as imagens de acordo com uma imagem de referência dentro desse conjunto. O conjunto pode consistir de duas ou mais imagens tiradas de uma única cena em momentos diferentes, a partir de pontos de vista diferentes, e/ou por sensores diferentes. O caso mais geral de transformação é a homografia planar de oito graus de liberdade;

- *Reprojeção*: refere-se ao alinhamento das imagens para um sistema de coordenadas comum utilizando as transformações geométricas calculadas;
- *Stitching*: o objetivo desta etapa é sobrepor as imagens alinhadas em uma composição maior, combinando valores de pixels das partes sobrepostas e retendo pixels onde não ocorre sobreposição;
- *Mesclagem*: diferenças fotométricas globais frequentemente podem ocorrer entre imagens em uma sequência, resultando na visibilidade da emenda entre elas. Esta etapa é utilizada para minimizar esse efeito e homogeneizar a aparência global do mosaico.

Figura 2.2 – Diferentes etapas do mosaico de imagens. Aqui H são as matrizes de homografia entre as imagens



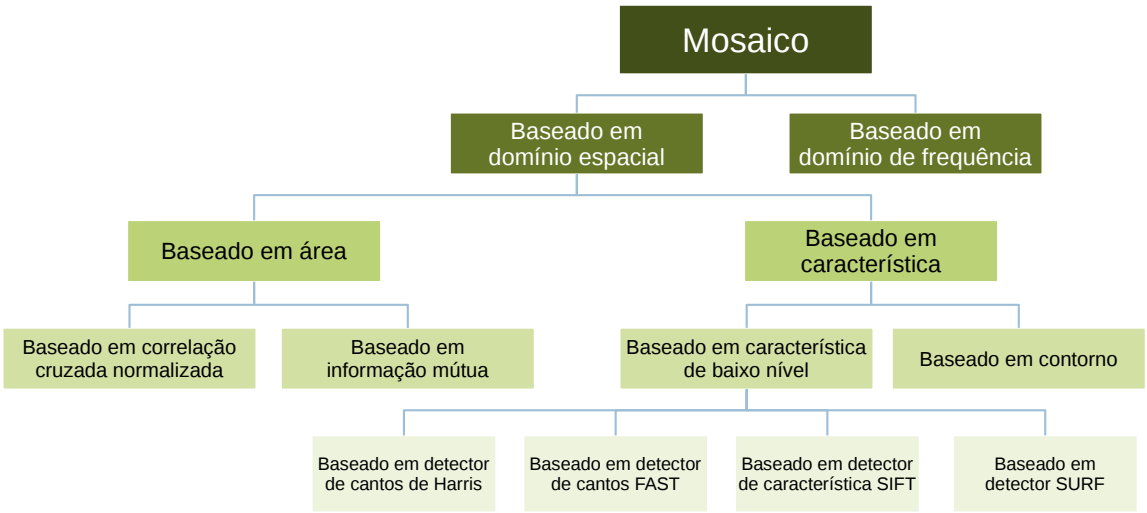
Fonte: Ghosh & Kaabouch (2016).

O Registro não é apenas uma etapa importante na construção do mosaico, mas também o fundamento do mesmo (GHOSH; KAABOUCH, 2016). Porém, devido à diversidade de imagens a serem registradas e aos diversos tipos de degradação, é improvável conceber um método universal aplicável (e que funcionará satisfatoriamente) a todas as tarefas de Registro (ZITOVÁ; FLUSSER, 2003; FEDOROV *et al.*, 2003). O mesmo aplica-se às tarefas de Mesclagem.

Assim sendo, é possível encontrar diversos métodos de mosaico de imagens na literatura. Ghosh & Kaabouch (2016) fizeram uma extensa pesquisa sobre os métodos de mosaico existentes baseados nas etapas de Registro e Mesclagem (primeira e última etapas do mosaico, respectivamente) e os classificaram em grupos.

Conforme ilustrado na Figura 2.3, em relação aos métodos de Registro, os algoritmos de mosaico podem ser baseados em domínio espacial ou em domínio de frequência. O mosaico de imagens baseado em domínio espacial pode ser ainda agrupado em mosaico de imagens baseado em área e baseado em característica. O mosaico de imagens baseado em área pode novamente ser subdividido em mosaico de imagens baseado em Correlação Cruzada Normalizada — do inglês *Normalized Cross-Correlation* (NCC) — e baseado em informação mútua. O mosaico de imagens baseado em característica pode novamente ser subdividido em mosaico de imagens baseado em característica de baixo nível e baseado em contorno. O mosaico baseado em característica de baixo nível pode ser dividido em quatro classes: mosaico baseado no detector de cantos de Harris, no detector de cantos FAST, no detector de característica SIFT e no detector SURF.

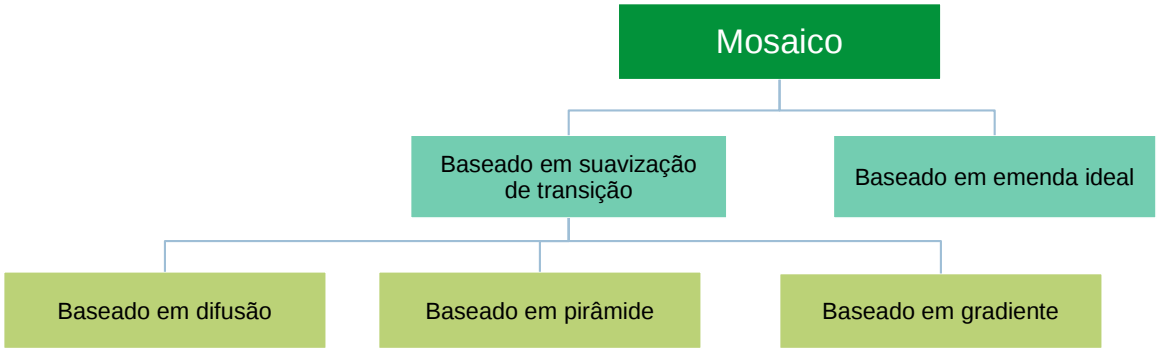
Figura 2.3 – Classificação dos mosaicos com base em Registro



Fonte: Ghosh & Kaabouch (2016).

Conforme ilustrado na Figura 2.4, em relação às técnicas de Mesclagem, os algoritmos de mosaico podem ser baseados em suavização de transição e em emenda ideal. O mosaico baseado em suavização de transição pode ser ainda agrupado em mosaico baseado em difusão, baseado em pirâmide e baseado em gradiente.

Figura 2.4 – Classificação dos mosaicos com base em Mesclagem



Fonte: Ghosh & Kaabouch (2016).

A Figura 2.5 ilustra um exemplo de mosaico construído a partir de 20 imagens. Para a etapa de Registro, um algoritmo detector de característica de baixo nível foi utilizado e, para a etapa de Mesclagem, um algoritmo baseado em difusão foi utilizado.

Figura 2.5 – Exemplo de mosaico construído a partir de 20 imagens

Fonte: Tarallo *et al.* (2013).

Dentre as abordagens para construção do mosaico de imagens, em relação aos métodos de Registro, destacam-se os métodos baseados em característica de baixo nível devido as suas maiores vantagens sobre os outros métodos: sua velocidade, robustez e a possibilidade de criação de imagem panorâmica de uma cena não planar com movimento de câmera irrestrito (ADEL; ELMOGY; ELBAKRY, 2014). Esses métodos serão descritos em maiores detalhes na próxima Seção.

2.4 MOSAICO BASEADO EM CARACTERÍSTICA DE BAIXO NÍVEL

Esses métodos não exigem imagens com grandes áreas de sobreposição para criação do mosaico (GHOSH; KAABOUCH, 2016). Essa categoria de algoritmos depende do cálculo da transformação utilizando um conjunto esparso de características de baixo nível. As características comumente utilizadas incluem borda, canto, pixel, cor, histograma, dentre outras. Independente de qual característica de baixo nível seja escolhida, ela deve ser distinta e espalhada por toda a imagem, e também deve ser eficientemente detectável em ambas as imagens (BIND, 2013).

O algoritmo detector de característica deve ser tal que o número de características comuns detectadas a partir de um conjunto de imagens seja suficientemente elevado, mesmo na presença das várias alterações geométricas e radiométricas. Além disso, o detector deve ter uma alta taxa de repetibilidade, de modo que as mesmas características sejam detectadas nas regiões de sobreposição entre um par de imagens.

Para que a correspondência de características seja possível, é necessário construir representações ricas que descrevem unicamente cada uma delas. Essas representações são feitas utilizando um descritor de característica. Um *descritor* é um vetor que descreve a vizinhança em torno de um ponto de característica em uma imagem. Diferentes métodos têm diferentes comprimentos e tipos de dados para seus descritores, mas geralmente são representados por vetores 1D ou 2D de números binários, inteiros ou pontos flutuantes (BAGGIO *et al.*, 2017; LAGANIÈRE, 2017). Executar qualquer operação em características e descritores de uma imagem geralmente é muito mais rápido e fácil do que tentar realizar o mesmo com toda a imagem, já que as características e os descritores são simplesmente um conjunto de valores numéricos que tentam descrever a imagem de uma forma ou de outra, dependendo do algoritmo utilizado para detectar características e extrair descritores (TAZEHKANDI, 2018).

O resultado produzido por qualquer algoritmo de correspondência de características contém sempre um número significativo de correspondências incorretas, que também são comumente referidas como falsas correspondências. A fim de melhorar a qualidade do conjunto de correspondências, diversas estratégias podem ser utilizadas, como as apresentadas por Vincent & Laganiere (2001).

A Figura 2.6 ilustra um exemplo de correspondências de características entre um par de imagens utilizando um algoritmo detector de característica de baixo nível. Além disso, estratégias para otimizar o conjunto de correspondências foram utilizadas. Apesar

da presença de diversas transformações entre as duas imagens, como escala, rotação e intensidade, a correspondência de características resultante foi de alta qualidade.

Figura 2.6 – Exemplo de correspondências de características entre um par de imagens com a presença de diversas transformações, como escala, rotação e intensidade

Fonte: Laganière (2017).

De acordo com Ghosh & Kaabouch (2016), os métodos mais populares de extração de característica de baixo nível utilizados na literatura de mosaico de imagens incluem: detector de cantos de Harris (HARRIS; STEPHENS, 1988), detector de cantos FAST (ROSTEN; DRUMMOND, 2006), detector de característica SIFT (LOWE, 2004) e detector SURF (BAY; TUYTELAARS; GOOL, 2006). A seguir, serão descritos os métodos de mosaico baseados nesses algoritmos.

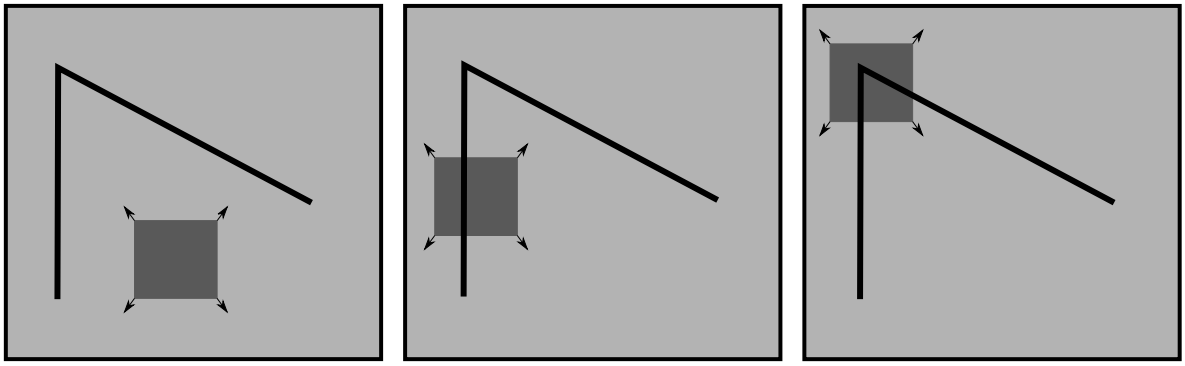
2.4.1 Mosaico Baseado no Detector de Cantos de Harris

Como o nome sugere, o detector de cantos de Harris detecta cantos nas imagens como características robustas de baixo nível. Inicialmente, uma pequena janela de detecção local é projetada na imagem (Figura 2.7). Na sequência, a variação de intensidade resultante do deslocamento dessa janela por uma pequena distância em qualquer sentido é determinada pela Equação 2.1 (JOSHI; SINHA, 2013):

$$E(u,v) = \sum_i w(x_i, y_i)[I(x_i + u, y_i + v) - I(x_i, y_i)]^2 \tag{2.1}$$

onde $w(x_i, y_i)$ é a função de janela para a janela de detecção (x_i, y_i), $I(x_i, y_i)$ é o valor de intensidade da imagem na localização de pixel (x_i, y_i), e $I(x_i + u, y_i + v)$ é a intensidade com deslocamento (u, v).

Figura 2.7 – Janela de detecção local de variação de intensidade. (a) Sem variação em todas as direções (b) Sem variação na direção da borda (c) Variação significativa em todas as direções (canto encontrado)



(a) (b) (c)

Fonte: O autor.

A textura local ao redor do pixel (x_i, y_i) é expressa como uma matriz de autocorrelação C, da seguinte forma:

$$C = \sum_i w(x_i, y_i) \begin{bmatrix} I_{x_i}^2 & I_{x_i} I_{y_i} \\ I_{x_i} I_{y_i} & I_{y_i}^2 \end{bmatrix} \tag{2.2}$$

onde I_{x_i} e I_{y_i} são a primeira derivada de $I(x_i, y_i)$. Dois autovalores grandes para a matriz C correspondem a um ponto de canto. Nesse caso, o ponto central da janela é caracterizado como tal. Para maior robustez, uma medida de *cornerness*[3] R é utilizada para eliminar os pontos de borda, conforme Equação 2.3 (GAO; JIA, 2007):

$$R = Det(C) - \alpha Tr^2(C) \tag{2.3}$$

onde $Tr(C)$ é o traço de C e α está dentro do intervalo $0{,}04 \leqslant \alpha \leqslant 0{,}06$. Pontos de canto são detectados como máximos locais de R acima de um limiar T predefinido.

Depois dos pontos de canto serem detectados em ambas as imagens, as correspondências são estabelecidas por NCC ou por outro método de Soma dos Quadrados das Diferenças — do inglês *Sum of Squared Differences* (SSD). Posteriormente, calculam-se os parâmetros de movimento geométrico e as imagens são deformadas de acordo com uma imagem de referência global a fim de criar o mosaico. Algoritmos de mosaico que utili-

[3]O objetivo dessa medida é associar uma pontuação proporcional ao quão fortemente uma região da imagem possui uma determinada estrutura, como, por exemplo, um canto.

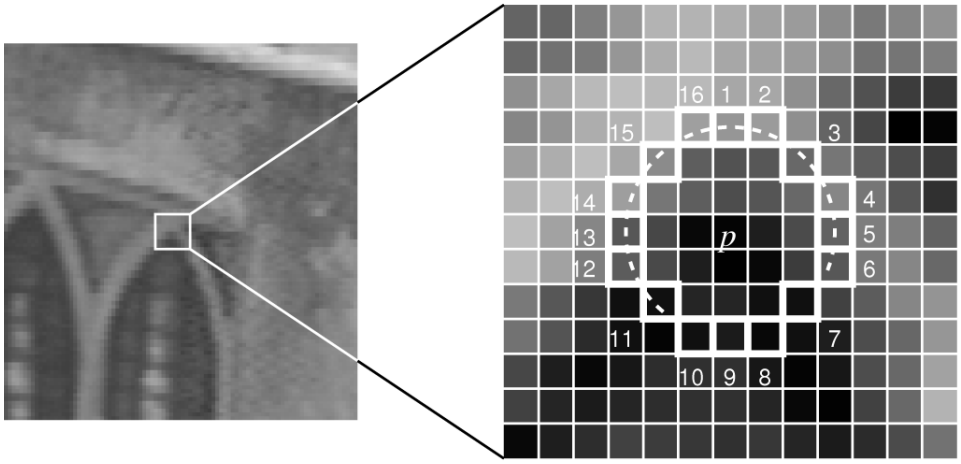
zam o detector de cantos de Harris são computacionalmente simples e acurados (GHOSH; KAABOUCH, 2016).

2.4.2 Mosaico Baseado no Detector de Cantos FAST

Assim como o detector de cantos de Harris, o detector de cantos FAST detecta cantos nas imagens. Esse algoritmo é computacionalmente mais eficiente e mais rápido do que a maioria dos outros métodos de extração de característica de baixo nível. Consequentemente, os métodos de mosaico baseados nesse algoritmo são particularmente adequados para aplicações de processamento de imagens em tempo real.

Inicialmente, um círculo de 16 pixels é considerado ao redor de cada pixel candidato a canto. O candidato é considerado um canto se existir um conjunto de n pixels contíguos no círculo que são ou todos mais claros do que a intensidade do pixel candidato mais um limiar, ou todos mais escuros do que a intensidade do pixel candidato menos um limiar, como ilustrado na Figura 2.8.

Figura 2.8 – Detecção de característica candidata para o algoritmo FAST



Fonte: Rosten & Drummond (2006).

Escolher um limiar ideal é, muitas vezes, um desafio fundamental dos algoritmos baseados no detector de cantos FAST. O número n é geralmente definido como 12. Para aumentar a velocidade do algoritmo FAST, uma função *Corner Response Function* (CRF) é utilizada. A CRF fornece a medida de *cornerness* de um ponto de canto com base nas intensidades de imagem da vizinhança local (JOSHI; SINHA, 2013). Os cantos são detectados como máximos locais para a função CRF calculada sobre toda a imagem.

Após a detecção, a correspondência de pontos de canto é realizada para cada par de imagens. Em seguida, as matrizes de homografia são calculadas e, por fim, as imagens

são projetadas em um sistema de coordenadas comum para obter o mosaico final.

2.4.3 Mosaico Baseado no Detector de Característica SIFT

O detector de característica SIFT detecta características distintas, também chamadas de "pontos-chave", nas imagens. O descritor SIFT é invariante à transformações de translação, rotação e escala no domínio da imagem e robusto para transformações de perspectiva moderadas e variações de iluminação. A operação do detector de característica SIFT é baseada em cinco etapas principais: construção do espaço de escala, detecção de extremos no espaço de escala, localização de pontos-chave, atribuição de orientação e definição de descritores aos pontos-chave (GHOSH; KAABOUCH, 2016).

Inicialmente, um espaço de escala é construído a partir da convolução repetida de uma imagem utilizando um filtro Gaussiano, com mudanças de escala e agrupamento das saídas em oitavas, conforme Equação 2.4 (LOWE, 2004):

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.4}$$

onde $*$ é o operador de convolução, $G(x, y, \sigma)$ é um filtro Gaussiano com escala variável σ e $I(x, y)$ é a imagem de entrada. Após a construção do espaço de escala, as imagens de Diferença de Gaussianas — do inglês *Difference of Gaussians* (DoG) — são calculadas a partir das imagens adjacentes de Gaussianas em cada oitava, da seguinte forma (LOWE, 2004):

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.5}$$

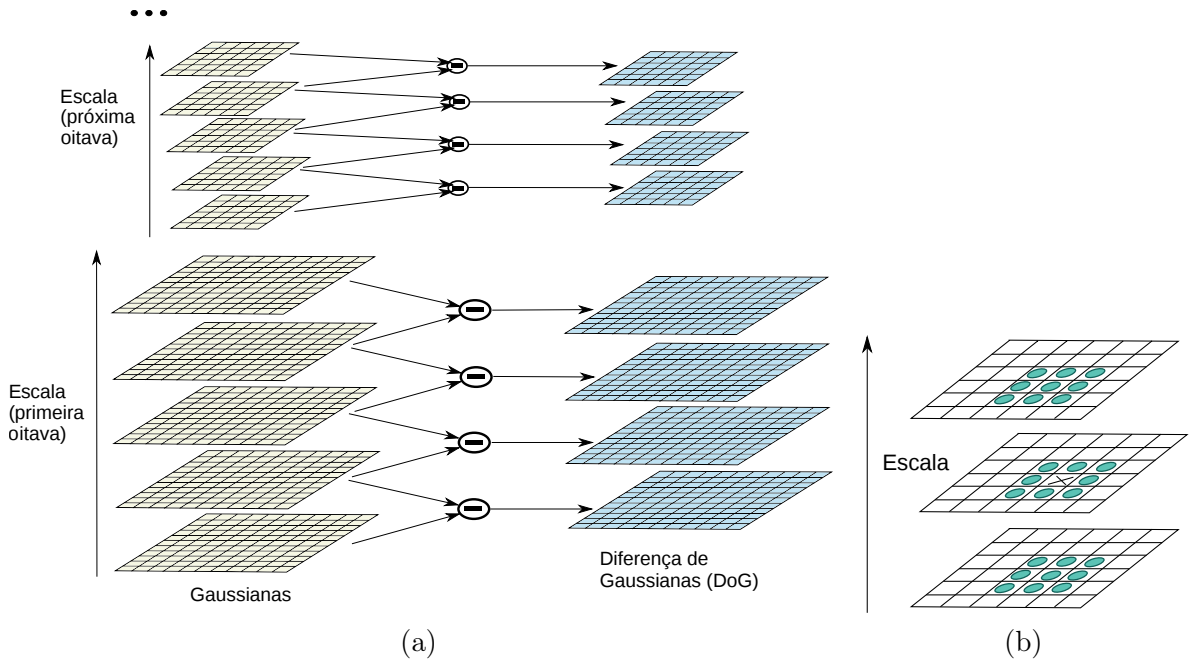
Em seguida, os pontos-chave candidatos são identificados como extremos locais das imagens de DoG em três escalas: a atual, a imediatamente acima e a imediatamente abaixo. O espaço de escala e a construção do espaço de escala de DoG, bem como a detecção de extremos no espaço de escala de DoG, são ilustrados na Figura 2.9.

Na etapa seguinte, pontos-chave de baixo contraste e/ou localizados ao longo de bordas são descartados utilizando uma localização de ponto-chave acurada. Os pontos-chave são, então, atribuídos a uma ou mais orientações baseando-se nas direções de gradiente locais da imagem, conforme Equação 2.6 (LOWE, 2004):

$$\theta(x, y) = tan^{-1}((L(x, y+1) - L(x, y-1))/(L(x+1, y) - L(x-1, y))) \tag{2.6}$$

onde $\theta(x, y)$ representa a direção do gradiente para $L(x, y, \sigma)$. Um conjunto de histogramas de orientação é formado sobre as vizinhanças de cada ponto-chave. Por fim, um vetor nor-

Figura 2.9 – (a) Espaço de escala e construção do espaço de escala de DoG (b) Detecção de extremos no espaço de escala de DoG comparando com 26 vizinhos



Fonte: Lowe (2004).

malizado de 128 dimensões é calculado para cada ponto-chave como seu descritor (GHOSH; KAABOUCH; FEVIG, 2014).

A fim de encontrar os pontos-chave correspondentes iniciais a partir de duas imagens, o vizinho mais próximo de um ponto-chave da primeira imagem é identificado a partir de uma base de dados de pontos-chave da segunda imagem (LOWE, 2004; LIQIAN; YUEHUI, 2010). Após a correspondência inicial, o algoritmo *Random Sampling Consensus* (RANSAC) (FISCHLER; BOLLES, 1981) é utilizado para remover as falsas correspondências e calcular os parâmetros de transformação entre um par de imagens. Por fim, as imagens são deformadas utilizando os parâmetros de transformação e combinadas para gerar o mosaico. Os algoritmos de mosaico de imagens baseados no detector de característica SIFT são particularmente adequados para combinar imagens de alta resolução sob uma variedade de alterações (rotação, escala, etc.), no entanto, ao custo de um elevado tempo de processamento (GHOSH; KAABOUCH, 2016).

2.4.4 Mosaico Baseado no Detector SURF

O detector SURF é um detector de característica invariante à escala e rotação. Assim como o detector de característica SIFT, ele também é baseado na teoria do espaço de escala. No entanto, o detector SURF utiliza a matriz Hessiana de toda a imagem para

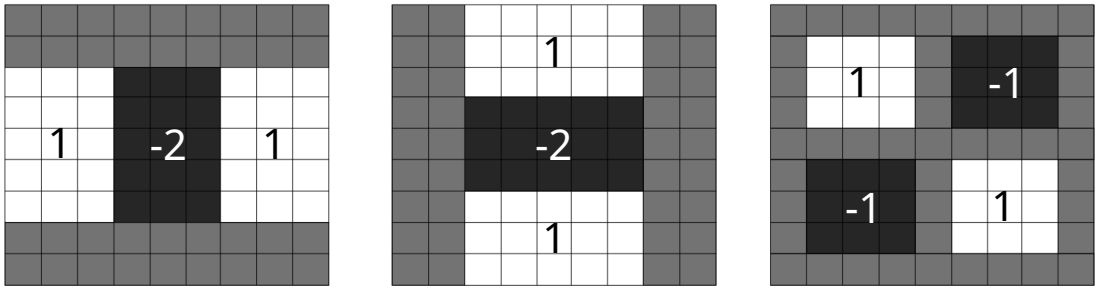
estimar máximos locais em diferentes espaços de escala (YANG *et al.*, 2011). A matriz Hessiana de uma imagem I com escala σ em qualquer ponto (x, y) é definida da seguinte forma (BAY; TUYTELAARS; GOOL, 2006):

$$H(x, y, \sigma) = \begin{pmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{pmatrix} \tag{2.7}$$

onde $L_{xx}(x, y, \sigma)$, $L_{yy}(x, y, \sigma)$ e $L_{xy}(x, y, \sigma)$ são as convoluções de I no ponto (x, y) com filtros Gaussianos de segunda ordem $\frac{\partial^2}{\partial x^2}G(x, y, \sigma)$, $\frac{\partial^2}{\partial y^2}G(x, y, \sigma)$ e $\frac{\partial^2}{\partial x \partial y}G(x, y, \sigma)$, respectivamente.

Ao calcular a matriz Hessiana em cada pixel, as operações de filtros Gaussianos são aproximadas por operações utilizando filtros de caixa, como ilustrado na Figura 2.10. A resposta em cada pixel é calculada como o determinante da matriz Hessiana. Em seguida, um limiar e uma janela de detecção de máximo local de 3×3×3 são utilizados para a supressão de não-máximos. Os máximos locais são interpolados no espaço de escala para obter os pontos-chave com seus valores de localização e escala. A fim de atribuir orientação para cada ponto-chave, respostas da *wavelet* de Haar são calculadas dentro de uma vizinhança circular ao redor de cada ponto-chave. Um vetor é formado somando todas as respostas dentro de uma janela de 60 graus. O vetor mais longo é atribuído como orientação para o ponto-chave. A fim de atribuir um vetor descritor a cada ponto-chave, uma região de vizinhança quadrada é definida ao redor do ponto-chave. Essa região é dividida em sub-regiões menores. A soma das respostas da *wavelet* de Haar de todas as sub-regiões são utilizadas para gerar um vetor descritor de 64 dimensões (BIND, 2013).

Figura 2.10 – Da esquerda para a direita: aproximação de derivadas Gaussianas parciais de segunda ordem nas direções x, y e xy



Fonte: Ghosh & Kaabouch (2016).

Depois de encontrar os pontos-chave correspondentes a partir de um par de imagens, o algoritmo RANSAC é utilizado para eliminar as falsas correspondências, bem como para calcular as matrizes de homografia. Uma vez que as matrizes de homografia são obtidas, as imagens são deformadas e combinadas para obter o mosaico final. As técnicas

de mosaico baseadas no detector SURF são mais rápidas que as técnicas baseadas no detector de característica SIFT. Entretanto, executam mal sob determinadas variações (particularmente de cor e iluminação) (GHOSH; KAABOUCH, 2016).

2.5 SENSORIAMENTO REMOTO NA DETECÇÃO DE DOENÇAS

A eficiência na tomada de decisão agrícola está relacionada à obtenção de informação rápida e acurada, especialmente no controle de pragas e doenças (TARALLO *et al.*, 2013). Obter frequentemente imagens aéreas de suas fazendas permite aos agricultores tomar decisões relacionadas à aplicação de fertilizantes, irrigação, aplicação de pesticidas e outras práticas agrícolas (LI; ISLER, 2016).

O conceito principal de sensoriamento remoto baseia-se na obtenção e coleta de dados com o uso de sensores, sem contato direto com o alvo. Os dados de sensoriamento remoto são, então, utilizados no apoio à tomada de decisão. Os critérios para o sucesso de um sistema de sensoriamento remoto devem incluir: rapidez na obtenção dos dados, confiabilidade e baixo custo. Nesse perfil, enquadram-se as RPAs comerciais. Um dos motivos responsáveis pela sua proliferação é sua utilização na agricultura. Em uma aplicação típica, uma RPA equipada com câmeras diversas sobrevoa o campo coletando imagens georreferenciadas que são utilizadas para construir um mosaico e auxiliar na tomada de decisão agrícola (LI; ISLER, 2016).

Uma importante área de aplicação para mosaicos criados a partir de imagens agrícolas obtidas por meio de RPA está na detecção de doenças nas culturas. Um exemplo é a ferrugem asiática da soja, uma doença altamente destrutiva, cuja alternativa de manejo mais utilizada é a aplicação de fungicidas (GODOY; CANTERI, 2004). Um grande desafio ao agricultor é a identificação da intensidade da doença em diferentes áreas, para evitar aplicações de fungicidas desnecessárias. Como apontado por Hikishima *et al.* (2010), a quantificação de danos é um ponto chave na definição de qualquer estratégia de controle de doenças.

Uma variável que expresse a intensidade da doença pode ser quantificada por incidência ou severidade, sendo esta última a mais apropriada para quantificar doenças foliares como ferrugens (FILHO; AMORIM, 1996). Entretanto, a quantificação da severidade em campo é um processo laborioso, feito com o uso de escalas diagramáticas, e está sujeito à visão humana, que pode gerar quantificações subjetivas da severidade. A fim de automatizar esse processo, técnicas de sensoriamento remoto têm sido empregadas com sucesso em estudos de detecção de doenças. Como exemplos, Garcia-Ruiz *et al.* (2013)

conseguiram utilizar imagens obtidas por meio de RPA para detectar a *Huanglongbing* (HLB), uma das principais doenças que atacam plantas de citros; e DI GENNARO *et al.* (2016) conseguiram utilizar imagens obtidas por meio de RPA para detectar os estágios iniciais da doença das estrias foliares da videira — do inglês *Grapevine Leaf Stripe Disease* (GLSD), mesmo antes da detecção visual.

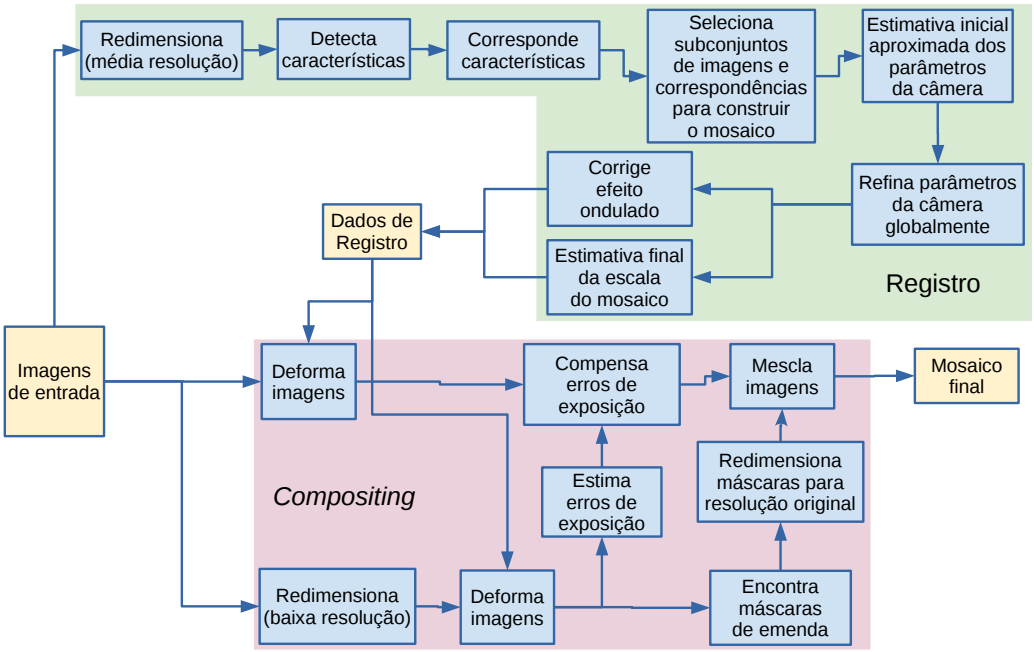
3 AQUISIÇÃO DE MOSAICO DE IMAGENS COM OPENCV

Este Capítulo descreve o processo de criação de mosaicos de imagens utilizando recursos oferecidos pela biblioteca OpenCV. A Seção 3.1 apresenta os recursos existentes e descreve como eles estão implementados internamente. A Seção 3.2 apresenta o programa computacional implementado para criação de mosaicos, apontando as diferenças em relação à implementação original da biblioteca OpenCV; e mostra, de forma resumida, seu algoritmo. A Seção 3.3 apresenta as considerações finais do Capítulo.

3.1 RECURSOS OFERECIDOS PELA BIBLIOTECA

Convenientemente, a biblioteca OpenCV (versão 3.3.1) oferece o módulo `stitching` e o submódulo `detail`, que contêm um conjunto de funções e classes que implementam um *stitcher*[1]. A Figura 3.1 ilustra o diagrama geral do módulo `stitching` implementado na classe `cv::Stitcher`. Utilizando essa classe, é possivel configurar ou pular algumas etapas, ajustando o diagrama de acordo com a necessidade. Todos os blocos de construção do diagrama estão disponíveis no espaço de nomes `cv::detail`, sendo possível combiná-los e/ou utilizá-los separadamente. O diagrama é baseado no sistema totalmente automático para criação de panorama proposto por Brown & Lowe (2007).

Figura 3.1 – Diagrama geral da implementação do módulo `stitching` na classe `cv::Stitcher`



Fonte: O autor.

[1]Pode ser entendido como um programa que cria mosaicos a partir de duas ou mais imagens.

Na pasta de exemplos do OpenCV, é possível encontrar um exemplo básico de criação de mosaicos (`[opencv_codigo_fonte]/samples/cpp/stitching.cpp`) que utiliza a classe `cv::Stitcher`. Essa classe possui diversas configurações padrão, possibilitando a criação de mosaicos de maneira simples, como ilustrado na Figura 3.2.

Figura 3.2 – Exemplo de criação de mosaico utilizando a classe `cv::Stitcher`

```
...
std::vector<cv::Mat> images; // Lê imagens de entrada
images.push_back(cv::imread("imagem1.jpg"));
images.push_back(cv::imread("imagem2.jpg"));
images.push_back(cv::imread("imagem3.jpg"));
cv::Mat mosaic; // mosaico de saída
// cria o stitcher padrão e o mosaico
cv::Stitcher stitcher = cv::Stitcher::createDefault();
cv::Stitcher::Status status = stitcher.stitch(images, mosaic);
...
```

Fonte: O autor.

Um outro exemplo, mais detalhado, que utiliza os recursos do módulo `stitching` separadamente, ou seja, sem o uso da classe `cv::Stitcher`, também pode ser encontrado (`[opencv_codigo_fonte]/samples/cpp/stitching_detailed.cpp`). Utilizar os recursos separadamente torna-se mais complexo, exigindo um conhecimento prévio da teoria acerca do processo de criação do mosaico de imagens. Entretanto, é possível melhorar a estabilidade e a qualidade na criação de mosaicos (HOWSE *et al.*, 2015).

Independente da forma escolhida para utilizar os recursos, suas implementações dentro do OpenCV são as mesmas. A seguir, está descrito como os passos mais relevantes para criação do mosaico de imagens estão implementados no algoritmo do módulo `stitching`.

3.1.1 Detecção e Correspondência de Características

Para detectar e descrever as características das imagens de entrada, três algoritmos (detectores e, ao mesmo tempo, descritores) estão disponíveis: *Accelerated KAZE* (AKAZE) (ALCANTARILLA; NUEVO; BARTOLI, 2013), *Oriented FAST and Rotated BRIEF* (ORB) (RUBLEE *et al.*, 2011) e SURF. Esses algoritmos são classes filhas da classe `cv::detail::FeaturesFinder`.

Para cada característica encontrada pelo algoritmo escolhido, as duas melhores correspondências candidatas (vizinhos mais próximos) são mantidas e algumas estratégias

são utilizadas para melhorar o conjunto de correspondências. Inicialmente, um filtro é realizado tanto nas correspondências de características da *imagemA* para a *imagemB* quanto da *imagemB* para a *imagemA*, da seguinte forma:

$$d_1 < (\sigma - t)d_2 \tag{3.1}$$

onde d_1 é a distância do vizinho mais próximo, $\sigma = 1{,}0$, t é um limiar previamente definido de acordo com o algoritmo detector de característica escolhido, e d_2 é a distância do segundo vizinho mais próximo. Caso a condição seja satisfeita, a correspondência é aceita; do contrário, ela é descartada.

Em seguida, utilizando as correspondências obtidas até o momento, o algoritmo RANSAC é utilizado para eliminar as falsas correspondências (*outliers*), bem como para calcular as matrizes de homografia. Por fim, utilizando somente as correspondências corretas (*inliers*) obtidas anteriormente, o algoritmo RANSAC é utilizado novamente para recalcular as matrizes de homografia.

3.1.2 Correspondência de Imagens

Para verificação de correspondência de imagens, o modelo probabilístico proposto por Brown & Lowe (2007) é utilizado da seguinte forma:

$$\sigma \geqslant \frac{n_i}{(\alpha + \beta n_f)} \tag{3.2}$$

onde $\sigma = 1{,}0$, n_i é o número de *inliers* obtidos antes de recalcular as matrizes de homografia, $\alpha = 8{,}0$, $\beta = 0{,}3$, e n_f é o número de correspondências de características obtidas logo após a aplicação do filtro apresentado na Equação 3.1. Caso a condição seja satisfeita, a correspondência entre o par de imagens é considerada válida; do contrário, ela é descartada.

Semelhante ao que é sugerido por Brown & Lowe (2007), o número máximo de pares por imagem pode ser limitado, de acordo com um parâmetro definido previamente. Dessa forma, ocorre uma otimização no tempo de processamento, já que não será necessário processar todos os pares possíveis para uma determinada imagem. Entretanto, quando essa opção é utilizada, as imagens de entrada, obrigatoriamente, devem ficar ordenadas, garantindo que a formação de pares ocorra em uma sequência, e não de forma aleatória.

3.1.3 *Bundle Adjustment*

Bundle adjustment é utilizado para obter todos os parâmetros da câmera simultaneamente para um determinado conjunto de correspondências entre as imagens. Elas são adicionadas a um ajustador, uma a uma, com a melhor imagem correspondente (maior número de correspondências consistentes) sendo adicionada a cada etapa. A nova imagem é inicializada com a mesma rotação e distância focal da imagem à qual melhor corresponde. Então, utiliza-se o algoritmo de Levenberg–Marquardt (LEVENBERG, 1944; MARQUARDT, 1963) para atualizar os parâmetros da câmera. O algoritmo de Levenberg–Marquardt geralmente é utilizado para resolver problemas de mínimos quadrados não lineares em problemas de ajuste de curvas (KAPUR; THAKKAR, 2015).

Este algoritmo (*bundle adjustment*) busca minimizar a soma dos quadrados dos erros de projeção. Para isso, cada característica é projetada em todas as imagens às quais corresponde, e a soma dos quadrados é minimizada em relação aos parâmetros da câmera. Se a k-ésima característica em uma imagem corresponde à l-ésima característica em outra, obtém-se o resíduo de projeção da seguinte forma (BROWN; LOWE, 2007):

$$r_{ij}^k = u_i^k - p_{ij}^k \tag{3.3}$$

onde u_i^k representa a k-ésima característica na i-ésima imagem, r_{ij}^k é o resíduo depois da projeção da k-ésima característica da j-ésima imagem na i-ésima imagem, e p_{ij}^k é a projeção da j-ésima imagem na i-ésima imagem do ponto correspondente a u_i^k.

A função de erro é calculada somando os erros residuais robustos sobre todas as imagens. Para a robustez, a função de custo de Huber (HUBER, 1981) é utilizada:

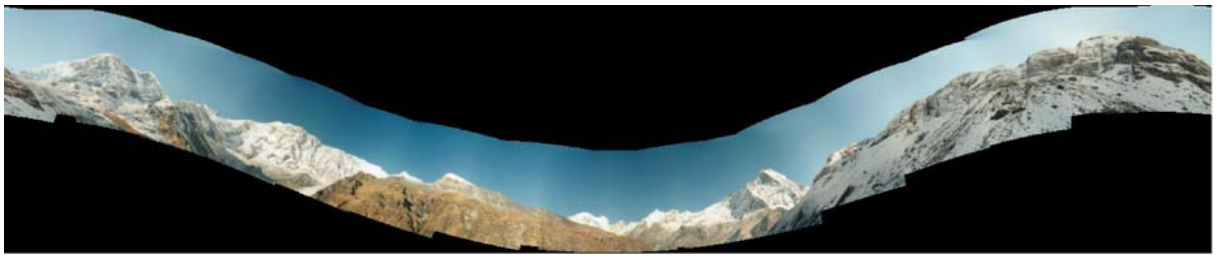
$$h(r_{ij}^k) = \begin{cases} |r_{ij}^k|^2 & ; se\ |r_{ij}^k| < \sigma \\ 2\sigma|r_{ij}^k| - \sigma^2 & ; caso\ contrário \end{cases} \tag{3.4}$$

Ao resolver isso, obtém-se uma equação não linear, que é resolvida utilizando o algoritmo de Levenberg-Marquardt, estimando-se os valores dos parâmetros da câmera.

3.1.4 Alinhamento Panorâmico Automático

Até agora, o algoritmo do módulo `stitching` foi capaz de encontrar correspondências entre as imagens e de combiná-las. No entanto, ainda existe um componente de rotação 3D desconhecido que, no caso da construção de um panorama, normalmente faz com que ele seja formado com um efeito ondulado, como ilustrado na Figura 3.3a.

Figura 3.3 – (a) Imagem panorâmica com efeito ondulado (b) Imagem panorâmica com efeito ondulado removido



(a)

(b)

Fonte: Brown & Lowe (2007).

Isso ocorre, principalmente, devido ao fato de que a câmera não teria sido perfeitamente nivelada ao obter as múltiplas imagens; e pode ser corrigido levando em consideração uma heurística sobre a forma como as pessoas tiram fotos panorâmicas. Supõe-se que é altamente improvável que a pessoa rotacione a câmera em relação ao horizonte ao tirar a foto, de modo que os vetores X (eixo horizontal) da câmera geralmente ficam no mesmo plano. Encontrando o vetor nulo da matriz de covariância dos vetores X da câmera, pode-se encontrar o vetor normal ao plano contendo o centro da câmera e o horizonte (BROWN; LOWE, 2007). Dessa forma, pode-se aplicar uma rotação nas imagens para remover efetivamente o efeito ondulado (KAPUR; THAKKAR, 2015).

3.1.5 Compensação de Ganho

Nas Seções anteriores, os parâmetros geométricos de cada câmera foram calculados. Agora, um parâmetro fotométrico é computado: o ganho geral entre as imagens.

Ganho é o parâmetro da câmera que descreve a sensibilidade da imagem à luz. Diferentes imagens poderiam ter sido obtidas em diferentes níveis de ganho, como ilustrado na Figura 3.4a. Para contornar esse tipo de problema, faz-se o uso da compensação de ganho.

Compensação de ganho refere-se à normalização do ganho nas imagens para facilitar uma imagem perfeitamente sem emendas. O método utilizado é semelhante ao que foi usado para calcular os parâmetros da câmera. A função de erro utilizada é a soma dos

Figura 3.4 – (a) Imagem sem compensação de ganho (b) Imagem com compensação de ganho

(a)

(b)

Fonte: Brown & Lowe (2007).

erros em intensidades de ganho normalizadas para todos os pixels sobrepostos (KAPUR; THAKKAR, 2015).

3.1.6 Mesclagem em Multibandas

Mesmo depois da compensação de ganho, a imagem não parece estar perfeitamente sem emendas. Assim, é necessário aplicar um algoritmo de mesclagem para combinar as imagens sem que seja notável que o mosaico tenha sido construído a partir de múltiplas imagens.

A mesclagem em multibandas (BURT; ADELSON, 1983) é classificada como um método baseado em pirâmide (Figura 2.4). Ela mescla regiões de baixa frequência sobre um amplo intervalo espacial, enquanto mescla regiões de alta frequência sobre um curto intervalo (BROWN; LOWE, 2007). Inicialmente, pesos de mesclagem (máscaras binárias) associados com cada imagem de entrada são criados localizando, no mosaico, o conjunto de pontos para o qual cada imagem é responsável. Nas regiões para as quais uma determinada imagem de entrada é a responsável, o valor 1 é atribuído à sua respectiva máscara. Nas demais regiões (para as quais outras imagens de entrada são as responsáveis), o valor 0 é atribuído.

Uma pirâmide Gaussiana é criada aplicando, iterativamente, um filtro Gaussiano na imagem de entrada com um valor diferente de desvio padrão em cada iteração. Além do filtro, uma subamostragem da imagem por um fator de dois é realizada em cada iteração. A partir da subtração da imagem original de sua versão filtrada (em cada iteração), uma pirâmide Laplaciana também é criada, com imagens que representam frequências espaciais no intervalo estabelecido pelo valor de desvio padrão do filtro Gaussiano. Isso cria uma representação reversível e supercompleta do sinal da imagem (SZELISKI, 2006). A Figura 3.5 ilustra as pirâmides Gaussiana e Laplaciana.

Figura 3.5 – (a) Pirâmide passa-baixa (Gaussiana) (b) Pirâmide passa-banda (Laplaciana)

(a) (b)

Fonte: Ghosh & Kaabouch (2016).

O mesmo processo utilizado para criar a pirâmide Gaussiana a partir da imagem de entrada original é aplicado nas máscaras binárias. Essas máscaras subamostradas e desfocadas tornam-se os pesos utilizados para realizar a mesclagem das imagens passa-banda em cada nível da pirâmide. Por fim, o mosaico final é obtido interpolando e somando todos os níveis da pirâmide (imagens passa-banda) (SZELISKI, 2006).

A Figura 3.6 é um exemplo de mesclagem em multibandas. A transição entre as imagens foi suavizada, mesmo na presença de alterações de iluminação, e, ao mesmo tempo, detalhes de alta frequência foram mantidos.

Figura 3.6 – Imagem com mesclagem em multibandas

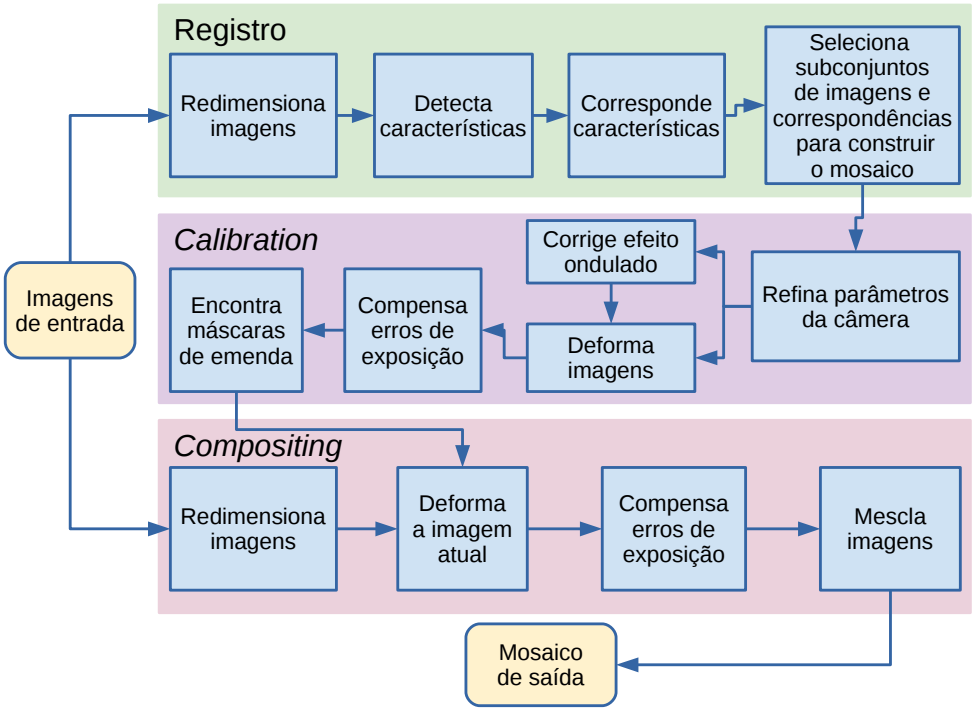
Fonte: Brown & Lowe (2007).

3.2 PROGRAMA COMPUTACIONAL IMPLEMENTADO

Devido a algumas incompatibilidades (descritas em maiores detalhes na Seção 3.2.1) entre o propósito deste trabalho e os recursos oferecidos pelo módulo `stitching`, como a ausência dos algoritmos escolhidos para avaliação e a incapacidade das estratégias implementadas em criar um conjunto de correspondências de características de qualidade para as imagens agrícolas obtidas por meio de RPA, foi necessário implementar alguns trechos de código fonte e adaptar outros, resultando em um programa para criação de mosaicos.

O programa foi implementado na linguagem de programação `C++` utilizando recursos oferecidos pela biblioteca OpenCV (versão 3.3.1), tendo como base o exemplo detalhado citado na Seção 3.1. Todo o processo foi dividido em três etapas principais: Registro, *Calibration* e *Compositing*, como o proposto por García *et al.* (2015). A Figura 3.7 apresenta o diagrama geral do programa por meio de uma representação de fluxo de dados, onde os retângulos azuis representam os processos a serem executados e as setas indicam a orientação do fluxo da informação.

Figura 3.7 – Diagrama geral do programa implementado



Fonte: O autor.

A etapa de Registro foi a única que sofreu alterações significativas em relação ao código fonte original oferecido pelo módulo `stitching`, sendo reimplementada completa-

mente. Tudo o que foi descrito entre as Seções 3.1.3 e 3.1.6 ocorre igualmente nas etapas *Calibration* e *Compositing*. Como não houve necessidade de uma nova implementação, alteração ou adaptação, o código fonte dessas etapas pôde ser reaproveitado do módulo `stitching` em sua totalidade. A seguir, a nova etapa de Registro será descrita em detalhes.

3.2.1 Etapa de Registro

De acordo com o que foi citado na Seção 3.1.1, é possível perceber que o detector de cantos de Harris, o detector de cantos FAST e o detector de característica SIFT não estavam disponíveis dentro do módulo `stitching`. Além disso, diferente dos algoritmos oferecidos nesse módulo, o detector de cantos de Harris e o detector de cantos FAST não são, além de detectores, descritores de características. Nesse caso, foi necessário utilizar um descritor de característica para ambos. O descritor escolhido foi o *Binary Robust Invariant Scalable Keypoints* (BRISK) (LEUTENEGGER; CHLI; SIEGWART, 2011), pois, durante testes realizados com diversos descritores oferecidos pela biblioteca OpenCV, ele foi o que apresentou melhor compatibilidade com os dois detectores.

Todos os detectores de características escolhidos para avaliação, bem como o descritor BRISK, estavam disponíveis dentro dos módulos `features2d` e `xfeatures2d` como classes filhas da classe `cv::Feature2D`. Devido à diferença de módulos e classes, foi necessário realizar uma implementação que utilizasse os recursos dos módulos `features2d` e `xfeatures2d` para detectar e descrever as características das imagens de entrada e que, ao mesmo tempo, fosse compatível com as classes e funções do módulo `stitching`, já que estas seriam utilizadas futuramente nas etapas *Calibration* e *Compositing*.

Inicialmente, as mesmas estratégias para melhorar o conjunto de correspondências descritas na Seção 3.1.1 foram utilizadas. Porém, durante diversos testes realizados, elas não foram capazes de criar um conjunto de qualidade para as imagens agrícolas obtidas por meio de RPA, resultando em erros durante a criação do mosaico ou em resultados pobres, nas poucas vezes em que o mosaico final foi gerado. Assim, verificou-se a necessidade de uma implementação mais robusta dessas estratégias. Apesar de algumas delas terem sido as mesmas utilizadas no módulo `stitching`, elas tiveram de ser reimplementadas para que fossem compatíveis com o código fonte oriundo dos módulos `features2d` e `xfeatures2d`.

Da mesma forma que ocorre no código original do módulo `stitching`, para cada característica encontrada pelo algoritmo escolhido, as duas melhores correspondências candidatas (vizinhos mais próximos) foram mantidas. Para melhorar o conjunto de correspondências, inicialmente, o *ratio test* (LOWE, 2004) é aplicado tanto nas correspondências

de características da *imagemA* para a *imagemB* quanto da *imagemB* para a *imagemA*, da seguinte forma:

$$\frac{d_1}{d_2} \leqslant \theta \tag{3.5}$$

onde d_1 é a distância do vizinho mais próximo, d_2 é a distância do segundo vizinho mais próximo, e $\theta = 0{,}8$. Caso a condição seja satisfeita, a correspondência é aceita; do contrário, ela é descartada. O *ratio test* elimina 90% de *outliers* enquanto descarta menos de 5% de *inliers* (LOWE, 2004).

Em seguida, a simetria entre os pontos de característica é imposta. Impor simetria significa manter somente pares nos quais cada ponto é a correspondência mais forte do outro. Isso aumenta as chances de que os pontos realmente correspondam à projeções do mesmo ponto físico na cena. Impor simetria é claramente vantajoso, pois elimina muitos *outliers* enquanto afeta somente poucos *inliers* (VINCENT; LAGANIERE, 2001). Na sequência, utilizando as correspondências obtidas até o momento, o algoritmo RANSAC é utilizado para eliminar os *outliers*, bem como para calcular as matrizes de homografia. Por fim, utilizando somente os *inliers* obtidos anteriormente, o algoritmo RANSAC é utilizado novamente para recalcular as matrizes de homografia.

Para verificação de correspondência de imagens, o mesmo modelo probabilístico apresentado na Seção 3.1.2 foi implementado.

O número máximo de pares por imagem foi limitado em seis, por padrão, e a implementação feita não exige que as imagens de entrada estejam ordenadas para que os pares sejam formados. Nesse caso, o código implementado foi mais fiel ao que é sugerido por Brown & Lowe (2007).

3.2.2 Algoritmo

A Figura 3.8 apresenta, de forma resumida, o algoritmo do programa implementado para criação de mosaicos. Em primeiro lugar, os cabeçalhos `stitching.hpp` e `detail` são incluídos (linhas 2–10) e os parâmetros mais importantes são definidos (linhas 13–26). É possível configurar o processo de criação de mosaicos com esses parâmetros. Todo o processo pode ser dividido em 10 passos importantes.

O primeiro passo lê as imagens de entrada e verifica a quantidade (linha 30). Pelo menos duas imagens são necessárias para a criação do mosaico.

O segundo passo redimensiona as imagens de entrada de acordo com o parâmetro da linha 13 e detecta as características em cada imagem utilizando o algoritmo definido

pelo parâmetro da linha 15. O redimensionamento é feito na linha 39 e a detecção de características na linha 41.

O terceiro passo corresponde as características que foram detectadas anteriormente (linha 48). O *ratio test* (linhas 51 e 52), a imposição de simetria (linha 53) e o algoritmo RANSAC (linha 55) são aplicados para melhorar o conjunto de correspondências.

O quarto passo seleciona os subconjuntos de imagens e correspondências para construir o mosaico. Primeiramente, um máximo de seis pares por imagem é mantido (linha 59). Então, de acordo com o cálculo do modelo probabilístico apresentado na Seção 3.1.2, os novos subconjuntos são selecionados (linha 62).

O quinto passo refina globalmente os parâmetros da câmera utilizando *bundle adjustment* (linha 65). O método escolhido é definido pelo parâmetro da linha 16 e executado na linha 67.

O sexto passo, definido pelo parâmetro da linha 17, é opcional. Ele corrige o efeito ondulado, caso esteja presente no mosaico final. O tipo de correção é definido pelo parâmetro da linha 18 e executado na linha 70.

O sétimo passo cria um deformador de imagem que precisa de uma projeção de mapa. A projeção utilizada é a plana, definida pelo parâmetro da linha 19. O deformador é criado na linha 72 e cada imagem é deformada (linha 77).

O oitavo passo compensa os erros de exposição criando um compensador (linha 81), que é aplicado em cada imagem deformada (linha 83).

O nono passo encontra as máscaras de emenda. Esse processo procura as melhores áreas para que as imagens que irão formar o mosaico sejam combinadas. O método utilizado é definido pelo parâmetro da linha 21.

O décimo (e último) passo compõe o mosaico final. Este passo necessita do que foi feito nos passos anteriores para configurar a composição. Quatro subpassos são realizados para calcular o mosaico final. Primeiro, cada imagem de entrada é lida (linha 86) e, se necessário, redimensionada (de acordo com o parâmetro da linha 14). Então, as imagens são deformadas com o deformador criado (linha 88). Na sequência, as imagens são compensadas por erros de exposição com o compensador criado (linha 91). Por fim, as imagens são mescladas (linha 93) utilizando a mesclagem em multibandas, definida pelo parâmetro da linha 22. O mosaico final é escrito em arquivo de acordo com o parâmetro da linha 26.

3.3 CONSIDERAÇÕES FINAIS

A biblioteca OpenCV é, normalmente, a ferramenta padrão utilizada para todas as coisas relacionadas ao processamento de imagens. Sendo assim, desde o início sua utilização foi considerada para a realização deste trabalho. Foi identificada a existência de um módulo específico para criação de mosaicos de imagens na biblioteca, o módulo `stitching`, e uma análise foi feita para verificar sua compatibilidade com o objetivo do trabalho.

De modo geral, em relação à construção do mosaico, a análise teve um resultado positivo. O módulo `stitching` apresentou uma série de recursos que possibilitavam total controle do processo de criação de mosaicos. Porém, nem todos os recursos eram compatíveis com o objetivo do trabalho. Essas incompatibilidades estavam relacionadas à etapa de Registro, que teve de ser reimplementada para adequar-se ao propósito deste trabalho, resultando em um programa computacional para criação de mosaicos a partir de imagens agrícolas obtidas por meio de RPA.

O computador utilizado para implementar e executar o programa de criação de mosaicos tinha um processador Intel Core i7-2600 3.40GHz×8, 24GB de memória RAM e executava o sistema operacional Ubuntu 16.04. Além disso, apesar do suporte existente na biblioteca OpenCV para IPP, TBB e OpenCL/CUDA, apenas IPP e TBB foram utilizadas, o que deve ser considerado ao analisar os resultados de desempenho computacional apresentados no Capítulo 4.

Figura 3.8 – Algoritmo do programa computacional implementado

```
...
#include "opencv2/stitching/detail/autocalib.hpp"
#include "opencv2/stitching/detail/blenders.hpp"
#include "opencv2/stitching/detail/timelapsers.hpp"
#include "opencv2/stitching/detail/camera.hpp"
#include "opencv2/stitching/detail/exposure_compensate.hpp"
#include "opencv2/stitching/detail/matchers.hpp"
#include "opencv2/stitching/detail/motion_estimators.hpp"
#include "opencv2/stitching/detail/seam_finders.hpp"
#include "opencv2/stitching/detail/warpers.hpp"
#include "RobustMatcher.h" // classe construída para a etapa de Registro
...
double work_megapix = 0.6;
double compose_megapix = -1;
std::string features_type = "sift"; //"sift", "surf", "fast" ou "harris"
std::string ba_cost_func = "reproj";
bool do_wave_correct = false;
cv::detail::WaveCorrectKind wave_correct = cv::detail::WAVE_CORRECT_VERT;
std::string warp_type = "plane";
int expos_comp_type = cv::detail::ExposureCompensator::GAIN_BLOCKS;
std::string seam_find_type = "gc_color";
int blend_type = cv::detail::Blender::MULTI_BAND;
float ratio = 0.8f;
int max_pairs = 6;
float conf_thresh = 1.f;
std::string result_name = "mosaico.tif";
...
int main(int argc, char* argv[]) {
...
 if (num_images < 2) { LOGLN("Precisa de mais imagens"); return -1; }
...
 RobustMatcher rmatcher; // variável do tipo da classe construída
...
 for (int i = 0; i < num_images; ++i) {
...
   work_scale = std::min(1.0, std::sqrt(work_megapix * 1e6 /
                                        full_img.size().area()));
...
   cv::resize(full_img, img, cv::Size(), work_scale, work_scale);
...
   rmatcher.computeKeyPoints(img, features[i].keypoints);
...
 } //fim for i
...
 for (int i = 0; i < num_images - 1; ++i) {
   for (int j = i + 1, num_good_pairs = 0; j < num_images; ++j) {
...
    rmatcher.robustMatch(features[i].descriptors,
                         features[j].descriptors, good_matches);
```

```
...
     ratioTest(matches12);
     ratioTest(matches21);
     symmetryTest(matches12, matches21, good_matches);
...
     rmatcher.ransacTest(good_matches, features[i].keypoints,
                                    features[j].keypoints, m_info);
...
    } //fim for i
    rmatcher.retainOnlyBestPairs(pairwise_matches,
                                    pairwise_matches_temp, num_images);
  } //fim for j
  std::vector<int> indices = cv::detail::leaveBiggestComponent(features,
                                    pairwise_matches, conf_thresh);
...
  cv::Ptr<cv::detail::BundleAdjusterBase> adjuster;
...
  if (!(*adjuster)(features, pairwise_matches, cameras)) {
    LOGLN("Ajuste dos parâmetros da câmera falhou"); return -1; }
...
  cv::detail::waveCorrect(rmats, wave_correct);
...
  cv::Ptr<cv::detail::RotationWarper> warper = warper_creator->create(
              static_cast<float>(warped_image_scale * seam_work_aspect));
...
  for (int i = 0; i < num_images; ++i) {
...
    warper->warp(masks[i], K, cameras[i].R, cv::INTER_NEAREST,
                              cv::BORDER_CONSTANT, masks_warped[i]);
  } // fim for i
...
  cv::Ptr<cv::detail::ExposureCompensator> compensator =
       cv::detail::ExposureCompensator::createDefault(expos_comp_type);
  compensator->feed(corners, images_warped, masks_warped);
...
  for (int img_idx = 0; img_idx < num_images; ++img_idx) {
    full_img = imread(img_names[img_idx]);
...
    warper->warp(img, K, cameras[img_idx].R, cv::INTER_LINEAR,
                              cv::BORDER_REFLECT, img_warped);
...
    compensator->apply(img_idx, corners[img_idx], img_warped, mask_warped);
...
    blender->feed(img_warped_s, mask_warped, corners[img_idx]);
...
  } // fim for img_idx
  return 0;
} // fim main
```

Fonte: O autor.

4 AVALIAÇÃO DOS MÉTODOS DE MOSAICO DE IMAGENS

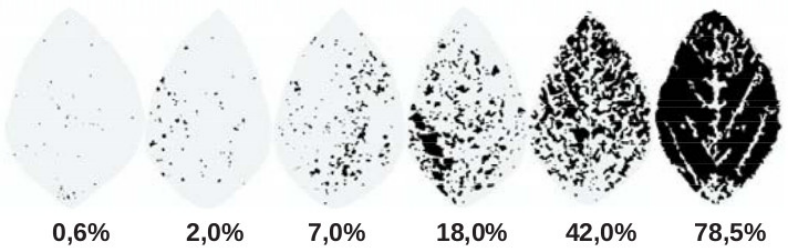
Este Capítulo apresenta como a avaliação dos métodos de mosaico de imagens foi realizada, bem como os resultados obtidos com a avaliação e a discussão feita com trabalhos relacionados. A Seção 4.1 descreve o processo de plantio da soja que, posteriormente, foi atacada pela doença conhecida como ferrugem asiática, e explica como a medição da severidade foi realizada em campo. A Seção 4.2 explica o processo de obtenção das imagens aéreas agrícolas. A Seção 4.3 descreve o processo de extração de dados dos mosaicos e a estimativa dos índices de severidade da ferrugem asiática a partir dos dados extraídos. As Seções 4.4 e 4.5 apresentam os resultados de desempenho computacional e qualidade obtidos a partir do grupo de imagens A (Seção 4.2.1), respectivamente. A Seção 4.6 apresenta os resultados de desempenho obtidos a partir do grupo de imagens B (Seção 4.2.2).

4.1 DADOS OBTIDOS EM CAMPO

A área utilizada para o plantio da soja (*Glycine max*) localiza-se na Fazenda Escola, na cidade de Ponta Grossa – Paraná – Brasil. O delineamento experimental compreendeu quatro parcelas denominadas A3, A4, B3 e B4. Cada parcela foi subdividida em quatro blocos com 11 tratamentos por bloco. A soja foi semeada nas parcelas A3 e A4 em 24/10/2016 e nas parcelas B3 e B4 em 13/12/2016. As parcelas A3 e B4 continham a cultivar *5969 Nidera* com população de 14 ou 15 plantas por metro. As parcelas A4 e B3 continham a cultivar *TMG 7262* com população de 10 plantas por metro.

A escala de Godoy, Koga & Canteri (2006) foi utilizada para quantificar a severidade da doença em campo (Figura 4.1). Sete plantas escolhidas ao acaso no centro de cada parcela foram divididas e analisadas em três partes: superior, central e inferior. O índice de severidade foi calculado a partir da média dos valores das três partes e das sete plantas.

Figura 4.1 – Escala diagramática da severidade da ferrugem da soja (*Glycine max*) (porcentagem da área foliar doente)



Fonte: Godoy, Koga & Canteri (2006).

Esses dados obtidos em campo servem como uma referência fidedigna da severidade da doença. O objetivo, agora, é realizar algum tipo de processamento nos mosaicos para que os dados obtidos a partir desse processamento aproximem-se o máximo possível dos dados fidedignos. Neste trabalho, o processamento que foi realizado está descrito na Seção 4.3.

4.2 IMAGENS OBTIDAS POR MEIO DE RPA

As imagens agrícolas foram obtidas por meio de uma RPA *eBee*[1], fabricada pela *senseFly* (Figura 4.2). Os voos foram conduzidos em uma altitude de 530 metros, entre 11h e 14h. A plataforma aérea foi equipada com uma câmera *Sony Cyber-shot DSC-WX220* de 18.2 megapixels, obtendo imagens com resolução de 15cm/pixel. A área média de sobreposição das imagens ficou em torno de 90%. Todas as imagens foram divididas em dois grupos principais: grupo de imagens A e grupo de imagens B.

Figura 4.2 – RPA *eBee*, *senseFly*

Fonte: O autor.

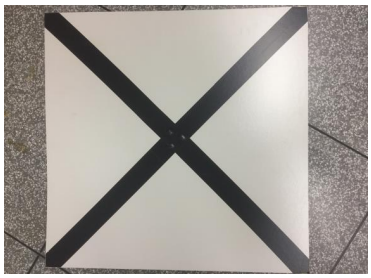
4.2.1 Grupo de Imagens A

Para obtenção deste grupo de imagens, antes de cada voo, cinco Pontos de Controle — do inglês *Ground Control Point* (GCP) — medindo 60x60cm (Figura 4.3) foram posicionados. As coordenadas geográficas do ponto central dos GCPs foram obtidas utilizando o receptor de Sistema de Navegação Global por Satélite — do inglês *Global Navigation Satellite System* (GNSS) — *Trimble Juno SC*.

Os voos foram realizados semanalmente, no dia da semana que apresentava condições climáticas favoráveis. A partir deles, três subgrupos de imagens da mesma região

[1] <https://www.sensefly.com/drones/ebee.html>

Figura 4.3 – Ponto de Controle



Fonte: O autor.

(citada na Seção 4.1) foram obtidos. O primeiro (subgrupo AA) foi composto por cinco imagens e obtido em 15/02/2017; o segundo (subgrupo AB) foi composto por seis imagens e obtido em 21/02/2017; e o terceiro (subgrupo AC) também foi composto por seis imagens e obtido em 13/03/2017.

O grupo de imagens A foi utilizado para avaliar o desempenho computacional dos métodos de mosaico e a qualidade do mosaico gerado, já que era o grupo no qual a soja estava presente nas imagens.

4.2.2 Grupo de Imagens B

Este grupo foi composto por 50 imagens e obtido em 27/06/2016 em uma região com uma aparência mais homogênea. A região localiza-se na Fazenda Paiquerê, na cidade de Piraí do Sul – Paraná – Brasil.

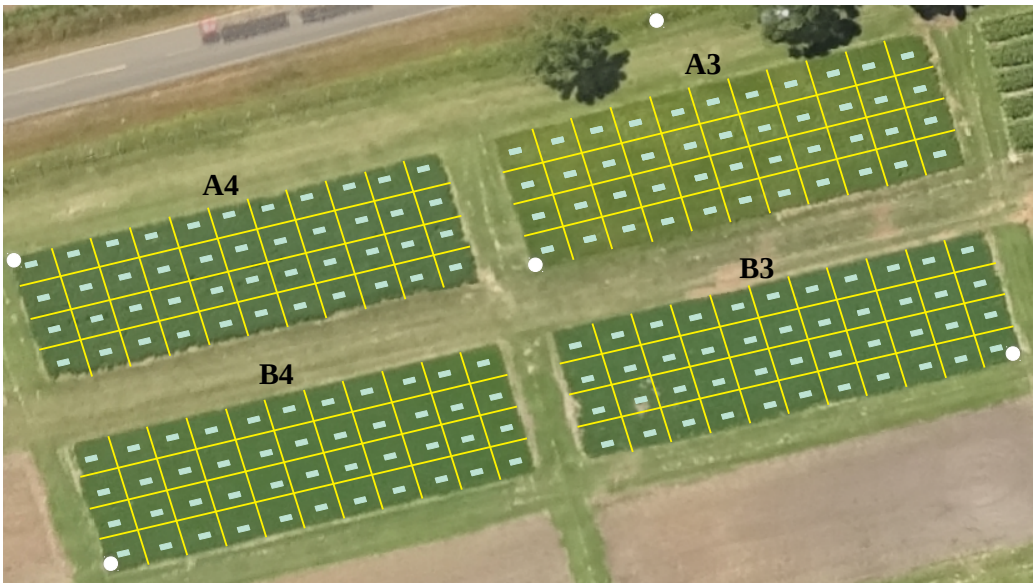
Como a soja não estava presente nas imagens deste grupo, ele foi utilizado exclusivamente para avaliar o desempenho computacional dos métodos de mosaico.

4.3 PROCESSAMENTO REALIZADO NOS MOSAICOS

O *software Quantum GIS* (versão 2.8.13) foi utilizado para corrigir o georreferenciamento de cada mosaico gerado pelo programa computacional implementado e pelo *software* comercial *Pix4Dmapper* (versão 3.1.23), baseando-se nas coordenadas dos GCPs. O *software Quantum GIS* também foi utilizado para extrair dados dos mosaicos, como ilustrado na Figura 4.4. Os círculos brancos representam os cinco GCPs; as linhas amarelas representam as divisões dos blocos dentro de cada parcela; e os retângulos verde-claros delimitam a região para extração dos dados da imagem em cada tratamento. Em cada região, 200 pontos aleatórios foram definidos, representando aproximadamente 10% de sua área total. Esses pontos foram utilizados para extrair os valores referentes às bandas *Red,*

Green, Blue (RGB).

Figura 4.4 – Representação das regiões de extração de dados nos mosaicos



Fonte: O autor.

Os dados extraídos foram divididos em duas bases. A primeira continha valores referentes às parcelas A3 e A4 e a segunda continha valores referentes às parcelas B3 e B4. Elas foram submetidas a testes estatísticos no *software* R^2 (versão 3.3.3). A fim de estimar os índices de severidade da ferrugem asiática a partir dos dados RGB extraídos (tendo como referência os dados fidedignos citados na Seção 4.1), uma análise de regressão foi realizada utilizando o algoritmo Regressão de Vetores de Suporte — do inglês *Support Vector Regression* (SVR) — (VAPNIK; GOLOWICH; SMOLA, 1997) presente no pacote *e1071* (versão 1.6.8).

O método de validação cruzada foi utilizado para validar os modelos de regressão. O conjunto de testes foi dividido em 10 partes (*folds*) e a predição foi aplicada em cada uma separadamente.

Todo o processamento descrito anteriormente está relacionado com a estimativa dos índices de severidade da ferrugem asiática da soja. Assim sendo, ele foi realizado apenas nos mosaicos gerados a partir do grupo de imagens A.

[2] <https://www.r-project.org/>

4.4 RESULTADOS DE DESEMPENHO COMPUTACIONAL A PARTIR DO GRUPO DE IMAGENS A

As Tabelas 4.1, 4.2 e 4.3 apresentam os dados de desempenho computacional obtidos com os mosaicos gerados a partir do grupo de imagens A. Os dados estão separados de acordo com o número de imagens utilizadas para compor o mosaico, a média de características detectadas em cada imagem, a média de *inliers* detectados em cada imagem, o tempo de processamento (tempo de usuário ou *user time*), em segundos (s), das etapas de Registro, *Calibration* e *Compositing*, e o tempo total, que marca o intervalo entre o momento em que o programa é inicializado até o momento em que o mosaico final é escrito em arquivo, quando o programa é finalizado.

Tabela 4.1 – Dados de desempenho computacional a partir do subgrupo de imagens AA

		SIFT	SURF	FAST	Harris
Número de imagens		5	3	5	5
Média de características		3.083	5.471	25.262	1.651
Média de *inliers*		265	207	666	73
Registro	detectar/descrever	2,74s	2,34s	3,39s	2,10s
	corresponder	3,29s	5,69s	67,05s	0,36s
Calibration		4,08s	0,44s	16,01s	3,79s
Compositing		8,67s	4,11s	7,19s	9,26s
Tempo total		21,05s	14,08s	95,38s	17,90s

Fonte: O autor.

Tabela 4.2 – Dados de desempenho computacional a partir do subgrupo de imagens AB

		SIFT	SURF	FAST	Harris
Número de imagens		6	2	6	6
Média de características		3.236	6.204	28.780	3.087
Média de *inliers*		393	364	1.043	158
Registro	detectar/descrever	3,68s	3,32s	4,59s	2,87s
	corresponder	5,56s	11,56s	139,14s	1,68s
Calibration		7,99s	0,13s	25,12s	12,88s
Compositing		8,87s	2,44s	8,84s	8,63s
Tempo total		27,93s	18,65s	179,45s	27,76s

Fonte: O autor.

Tabela 4.3 – Dados de desempenho computacional a partir do subgrupo de imagens AC

		SIFT	SURF	FAST	Harris
Número de imagens		6	6	6	5
Média de características		2.570	4.987	23.363	915
Média de *inliers*		357	598	1.487	56
Registro	detectar/descrever	3,23s	2,74s	3,79s	2,35s
	corresponder	3,57s	8,11s	94,65s	0,22s
Calibration		19,86s	18,59s	52,85s	5,20s
Compositing		7,32s	7,34s	7,43s	8,43s
Tempo total		35,69s	38,52s	160,50s	18,56s

Fonte: O autor.

Em relação à etapa de Registro, para uma melhor comparação de tempo de processamento entre os métodos, foi dividida em duas subetapas. A primeira subetapa (*detectar/descrever*) mostra o tempo que foi necessário para detectar e descrever as características de todas as imagens. A segunda subetapa (*corresponder*) mostra o tempo que foi necessário para encontrar as características correspondentes e formar os pares de imagens.

Verificou-se que, de modo geral, o número médio de características detectadas afetou diretamente o tempo de processamento da etapa de Registro, o número de imagens utilizadas para compor o mosaico afetou diretamente o tempo de processamento da etapa *Compositing*, e o número médio de *inliers* detectados, em conjunto com o número de imagens utilizadas para compor o mosaico, afetou diretamente o tempo de processamento da etapa *Calibration*.

As Tabelas 4.1 e 4.2 mostram que o detector SURF foi o único a não utilizar todas as imagens para compor o mosaico. Isso significa que a condição imposta pelo modelo probabilístico apresentado na Seção 3.1.2 não foi satisfeita para alguns pares de imagens. Devido a isso, os tempos de processamento das etapas *Calibration* e *Compositing* foram os menores dentre todos os métodos e, consequentemente, também o tempo total. Entretanto, essa vantagem de tempo não foi observada na etapa de Registro. Isso aconteceu devido ao elevado número médio de características e *inliers* detectados pelo detector SURF, ficando atrás apenas do detector de cantos FAST.

Em relação aos demais métodos (Tabelas 4.1, 4.2 e 4.3), é possível observar que o detector de cantos FAST obteve um tempo de processamento consideravelmente mais

alto para a etapa *Calibration* e para a subetapa *corresponder*. Na etapa *Calibration*, isso aconteceu devido ao elevado número de *inliers* detectados. Na subetapa *corresponder*, isso aconteceu devido à média de características detectadas, que foi cerca de 13,7 vezes maior quando comparada à menor média de características das Tabelas 4.1, 4.2 e 4.3.

Entretanto, o tempo médio de processamento da subetapa *detectar/descrever* foi apenas 1,6 vezes acima da média de tempo mais rápida das Tabelas 4.1, 4.2 e 4.3. Isso mostra que o detector de cantos FAST é mais rápido que os outros métodos para detectar características, o que corrobora com os resultados obtidos por Adel, Elmogy & Elbakry (2014), que compararam o tempo de processamento entre diversas técnicas de extração de características de imagem com foco em mosaico em tempo real. O detector de cantos FAST foi o mais rápido, seguido pelo detector de cantos de Harris.

A Tabela 4.3 mostra que, dessa vez, o detector SURF utilizou todas as imagens para compor o mosaico, fazendo com que a vantagem de tempo apresentada nas Tabelas 4.1 e 4.2 desaparecesse. O tempo total foi ainda superior ao do detector de característica SIFT. Entretanto, isso aconteceu devido ao elevado número médio de características e *inliers* detectados. Karami, Prasad & Shehata (2015) compararam diferentes técnicas de correspondência de imagens contra diferentes tipos de transformações e deformações, com um número similar de características detectadas por imagem. O detector SURF foi quase três vezes mais rápido que o detector de característica SIFT.

As Figuras 4.5, 4.6, e 4.7 apresentam os mosaicos gerados pelos métodos a partir do grupo de imagens A. O objetivo dessas Figuras é, simplesmente, mostrar a semelhança visual entre os mosaicos gerados pelos métodos e pelo *software* comercial *Pix4Dmapper*. De modo geral, a principal diferença está na área de abrangência, seja pela não utilização de todas as imagens na composição do mosaico ou, no caso do *software* comercial *Pix4Dmapper*, por uma restrição de área de interesse, o que faz com que ele descarte as extremidades do mosaico. Entretanto, é possível observar na Figura 4.5 que algumas "manchas escuras" causaram diferença visual significativa entre os mosaicos. Trata-se de um caso isolado, pois, na data de obtenção do subgrupo de imagens AA, algumas nuvens estavam sobre a região, resultando em sombras (manchas escuras) nas imagens. Como diferentes fatores podem influenciar na aparência final do mosaico (número de imagens utilizadas, detecção de regiões de sobreposição entre as imagens, algoritmo de mesclagem, etc.), é previsto, nesse caso, que a aparência dos mosaicos possa ter diferenças significativas. Os mosaicos das Figuras 4.5a, 4.5c e 4.5d podem ser utilizados como referência, já que não houve diferença visual significativa entre eles. No caso do mosaico da Figura 4.5b, a diferença ocorreu porque apenas três (de um total de cinco) imagens foram utilizadas para compor o

mosaico, resultando na detecção de diferentes áreas de sobreposição entre as imagens e em uma diferente influência do algoritmo de mesclagem. No caso do mosaico da Figura 4.5e, a diferença ocorreu por se tratar de um algoritmo completamente diferente (*software* comercial), tendo seu próprio modo de encontrar regiões de sobreposição entre as imagens e de realizar a mesclagem, resultando ou não na presença de regiões com sombra.

4.5 RESULTADOS DE QUALIDADE A PARTIR DO GRUPO DE IMAGENS A

As Tabelas 4.4 e 4.5 apresentam os resultados da estimativa da severidade da ferrugem asiática obtidos pelos métodos a partir dos mosaicos criados com o grupo de imagens A. O Coeficiente de Correlação (R) e a Raiz do Erro Quadrático Médio — do inglês *Root Mean Square Error* (RMSE) — são apresentados, de acordo com o subgrupo de imagens. Na data de obtenção do subgrupo de imagens AC, a soja das parcelas A3 e A4 já havia sido colhida. Nesse caso, não foi possível utilizar esse subgrupo para estimar os índices de severidade da ferrugem asiática para essas parcelas.

Tabela 4.4 – Dados de qualidade - estimativa da ferrugem asiática para as parcelas A3 e A4

	AA		AB	
	R	RMSE	R	RMSE
SIFT	0,83	6,10	0,84	5,89
SURF	0,86	5,56	0,86	5,54
FAST	0,83	6,09	0,85	5,76
Harris	0,83	5,98	0,85	5,67
Pix4Dmapper	0,85	5,71	0,88	5,17

Fonte: O autor.

Tabela 4.5 – Dados de qualidade - estimativa da ferrugem asiática para as parcelas B3 e B4

	AA		AB		AC	
	R	RMSE	R	RMSE	R	RMSE
SIFT	0,80	4,03	0,78	4,94	0,75	11,68
SURF	0,69	4,59	0,80	4,83	0,76	11,46
FAST	0,80	3,90	0,79	4,86	0,80	10,65
Harris	0,81	3,91	0,77	5,11	0,77	11,35
Pix4Dmapper	0,82	3,74	0,81	4,69	0,79	10,80

Fonte: O autor.

De modo geral, é possível observar que todos os valores das duas Tabelas foram bastante semelhantes. Entretanto, nos dados do subgrupo de imagens AA (Tabela 4.5), o R calculado para o detector SURF foi razoavelmente menor do que os demais. Isso aconteceu, especificamente nesse caso, devido ao baixo nível de detalhes (baixa resolução) do mosaico gerado, especialmente na região das parcelas B3 e B4. O nível de detalhes está associado à resolução espacial, que pode afetar diretamente os resultados obtidos a partir dos dados extraídos da imagem.

Garcia-Ruiz *et al.* (2013) conduziram um estudo comparando o impacto da resolução espacial das imagens na detecção da doença HLB, que ataca plantas de citros. As resoluções de 50cm/pixel e 5,45cm/pixel foram comparadas. Os autores verificaram que o conjunto de dados baseado nas imagens de maior resolução produziram uma melhor acurácia na classificação (67-85%) e menos falsos negativos (7-32%) do que o conjunto de dados correspondente baseado nas imagens com menor resolução (61-74% e 28-45%, respectivamente).

Apesar do baixo R calculado para o detector SURF na Tabela 4.5, uma Análise de Variância — do inglês *Analysis of Variance* (ANOVA) — de fator único com nível de significância de 5% foi realizada tanto com os valores de R quanto com os valores de RMSE. Em ambos os casos, não houve diferença significativa. Isso indica que a qualidade final dos mosaicos gerados pelos métodos avaliados e pelo *software* comercial *Pix4Dmapper* estava no mesmo patamar.

4.6 RESULTADOS DE DESEMPENHO COMPUTACIONAL A PARTIR DO GRUPO DE IMAGENS B

A Figura 4.8 mostra os mosaicos gerados pelos métodos a partir do grupo de imagens B. A Tabela 4.6 apresenta os dados de desempenho obtidos pelos métodos a partir desses mosaicos.

É possível observar que nenhum dos métodos utilizou o número total de imagens para compor o mosaico. O detector de cantos FAST obteve o melhor aproveitamento, com 48 imagens. Apesar disso, o alinhamento das imagens no mosaico final não foi o ideal, ocasionando descontinuidades de objetos, como apontado na Figura 4.8c. O detector de característica SIFT obteve o segundo melhor aproveitamento, com 41 imagens. O detector de cantos de Harris utilizou 26 imagens e o detector SURF obteve o pior aproveitamento, com apenas 3 imagens.

Além do fato de que a área utilizada tinha uma aparência mais homogênea, o que

Tabela 4.6 – Dados de desempenho computacional a partir do grupo de imagens B

		SIFT	SURF	FAST	Harris
Número de imagens		41	3	48	26
Média de características		4.556	6.198	32.021	6.401
Média de *inliers*		56	53	177	37
Registro	detectar/descrever	26,63s	24,30s	34,07s	18,40s
	corresponder	806,24s	867,51s	10.599s	563,88s
Calibration		1.559s	0,36s	8.494s	353,79s
Compositing		46,96s	3,13s	60,58s	15,64s
Tempo total		2.444s	896,56s	19.194s	953,09s

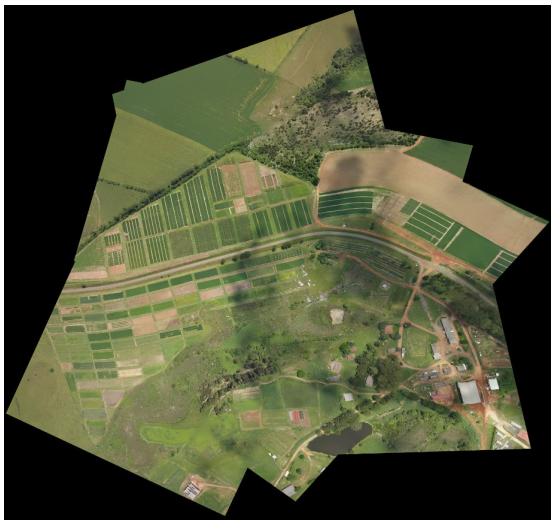
Fonte: O autor.

poderia dificultar a correspondência de características e, consequentemente, a formação de pares de imagens, segundo Ghosh & Kaabouch (2016), isso pode ser explicado devido ao desempenho pobre que o detector SURF tem sob determinadas transformações, como de cor e de iluminação, o que é comum em imagens agrícolas obtidas por meio de RPA; e o detector de cantos de Harris é bom apenas para mudanças moderadas em escala e rotação, situações que são facilmente superadas, especialmente em imagens obtidas por meio de RPA.

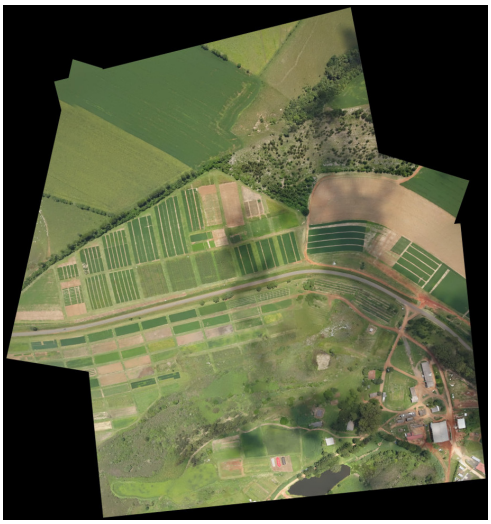
Por outro lado, pelo menos em relação às imagens agrícolas obtidas por meio de RPA, esses resultados, somados aos resultados pobres obtidos pelo detector SURF nas Tabelas 4.1 e 4.2, vão contra o que é afirmado por Ghosh & Kaabouch (2016), que dizem que métodos baseados em característica de baixo nível não exigem imagens com grandes áreas de sobreposição para criar o mosaico. Como mencionado na Seção 4.2, a área média de sobreposição ficou em torno de 90% e, ainda assim, os métodos nem sempre foram capazes de utilizar todas as imagens para compor os mosaicos.

Para superar os problemas de correspondência de características e de formação de pares de imagens encontrados durante a criação do mosaico, as estratégias para melhorar o conjunto de correspondências e o modelo probabilístico para formação de pares de imagens sempre podem ser ajustados. Entretanto, estratégias muito simples e um modelo probabilístico inadequado podem levar à incorretas correspondências, gerando resultados pobres ou falha completa durante a criação do mosaico.

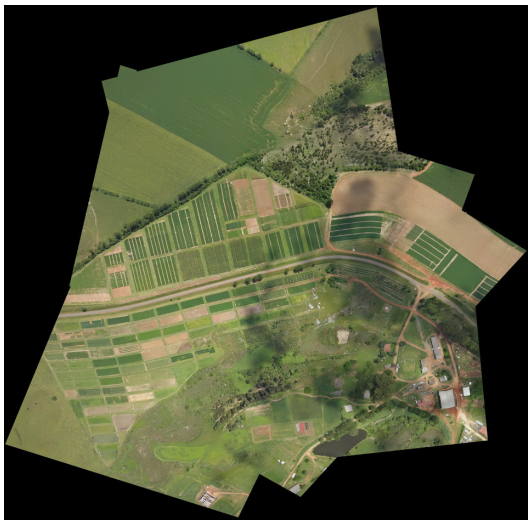
Figura 4.5 – Mosaicos criados a partir do subgrupo de imagens AA



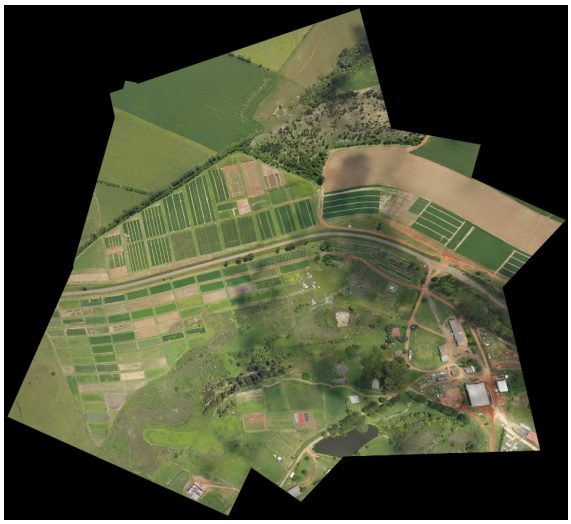
(a) SIFT



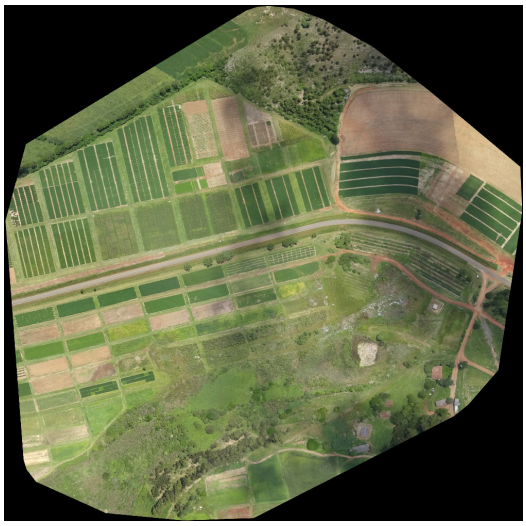
(b) SURF



(c) FAST



(d) Harris

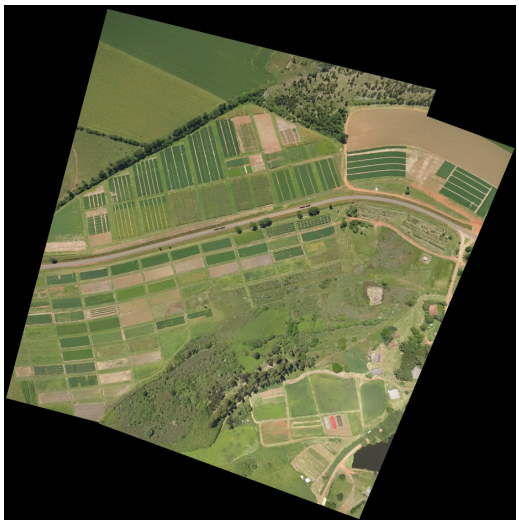


(e) Pix4Dmapper

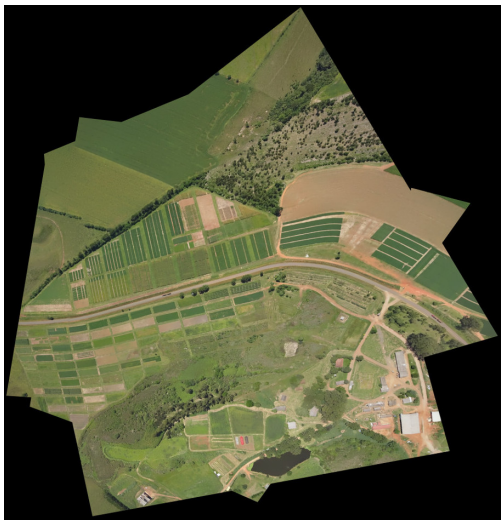
Fonte: O autor.

Figura 4.6 – Mosaicos criados a partir do subgrupo de imagens AB

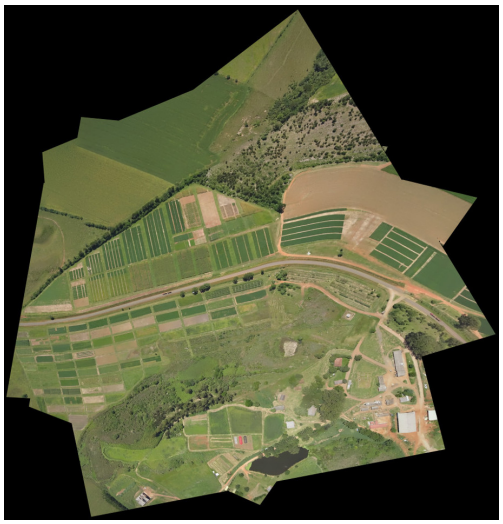
(a) SIFT



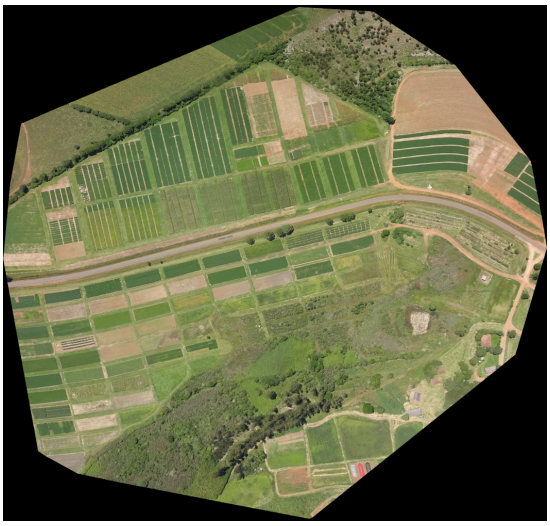
(b) SURF



(c) FAST



(d) Harris

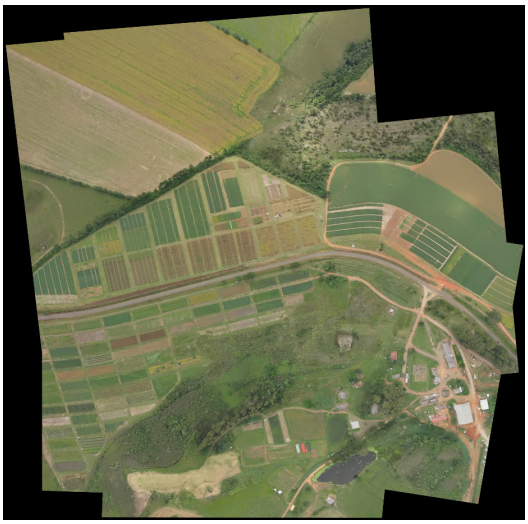


(e) Pix4Dmapper

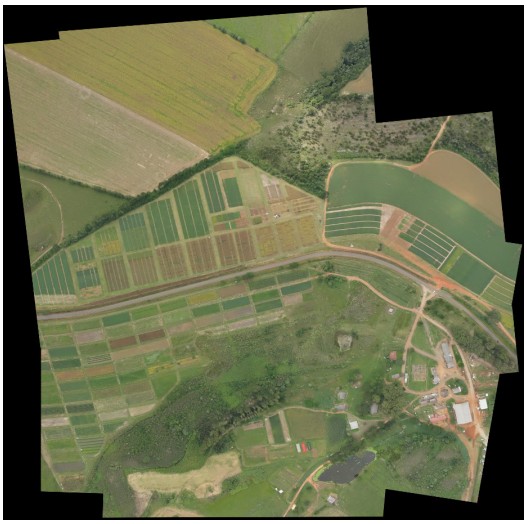
Fonte: O autor.

Figura 4.7 – Mosaicos criados a partir do subgrupo de imagens AC

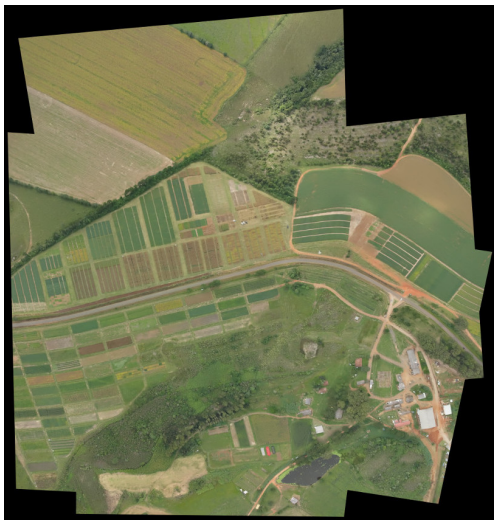
(a) SIFT



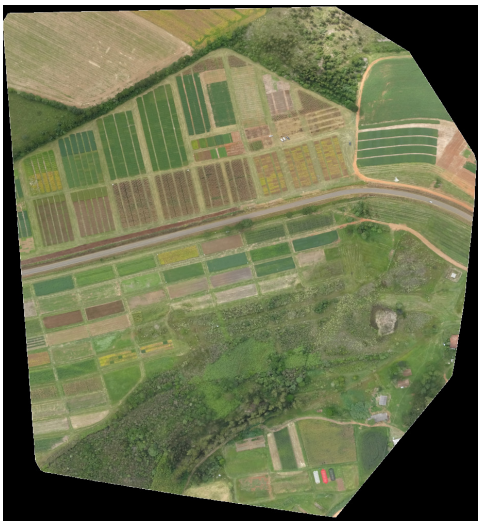
(b) SURF



(c) FAST



(d) Harris



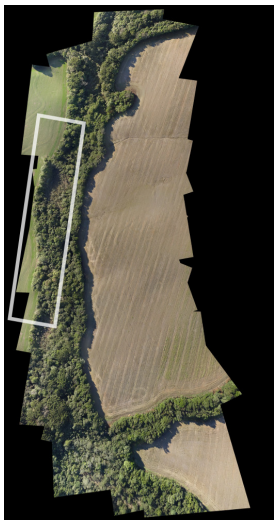
(e) Pix4Dmapper

Fonte: O autor.

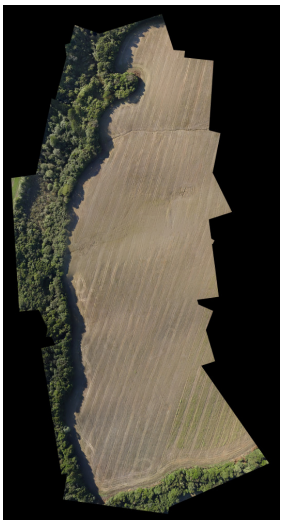
Figura 4.8 – Mosaicos criados a partir do grupo de imagens B

(a) SIFT

(b) SURF



(c) FAST



(d) Harris

(e) Pix4Dmapper

Fonte: O autor.

5 CONCLUSÃO

O mosaico de imagens demonstra sua relevância encontrando suas aplicações em diversas áreas; algumas delas sendo de importante valor social e econômico, como a medicina e a agricultura.

A área de pesquisa que envolve o mosaico de imagens permanece com estudos atuais, indicando a pertinência do tema. Ela é ampla e possui diversos métodos que podem ser utilizados para as mais variadas aplicações. Assim, são necessárias avaliações e comparações entre métodos para uma escolha apropriada à uma finalidade específica.

Este trabalho apresentou uma avaliação comparativa de quatro métodos de mosaico de imagens baseados em característica de baixo nível utilizando imagens agrícolas obtidas por meio de RPA. A avaliação foi feita de acordo com o desempenho computacional e a qualidade do mosaico gerado. Para avaliar o desempenho, foram levados em consideração o número médio de características e de *inliers* detectados por imagem, o número de imagens utilizadas para compor o mosaico e o tempo de processamento (tempo de usuário ou *user time*). Para avaliar a qualidade, os mosaicos gerados por cada método foram utilizados para estimar a severidade da ferrugem asiática da soja (*Glycine max*) e uma comparação com o *software* comercial *Pix4Dmapper* foi realizada.

A comparação feita entre as estimativas da severidade da ferrugem asiática da soja calculadas pelo *software* comercial *Pix4Dmapper* e pelos demais métodos demonstrou que a qualidade dos mosaicos gerados estava no mesmo patamar. Apesar dos valores não terem sido iguais, a ANOVA foi aplicada e os resultados não apresentaram diferenças significativas.

O detector SURF obteve o pior desempenho dentre todos os métodos, não sendo capaz de corresponder a maioria dos pares de imagens em três dos quatro grupos de imagens testados. Nesses casos, uma média de apenas 33,1% das imagens de entrada foi utilizada para compor o mosaico final. Assim, apresentou-se como uma solução inadequada para imagens agrícolas obtidas por meio de RPA.

O detector de cantos de Harris mostrou-se como a solução mais rápida dentre todos os métodos, podendo ser confiável para pequenos grupos de imagens agrícolas de regiões com uma aparência mais heterogênea, levando em conta o bom desempenho apresentado nas Tabelas 4.1, 4.2 e 4.3. Quando comparado à segunda solução mais rápida, ele foi, em média, 2,3 vezes mais rápido para detectar as características e corresponder os pares de imagens. Além disso, foi 7,27% mais rápido para compor o mosaico final nos casos em

que todas as imagens foram utilizadas (Tabelas 4.1 e 4.2). Porém, em seu último mosaico gerado, o aproveitamento das imagens de entrada foi pobre, ficando apenas em 52%.

O detector de cantos FAST, devido ao elevado número médio de características e *inliers* que detectou, obteve um melhor aproveitamento do número de imagens na composição dos mosaicos. Entretanto, seu tempo de processamento foi consideravelmente superior, chegando a ser, nos casos em que todas as imagens foram utilizadas para compor o mosaico, 6,42 vezes mais lento que a solução mais lenta dentre os demais métodos (Tabela 4.2). Além disso, descontinuidades de objetos significativas ocorreram em seu último mosaico gerado. Assim, apresentou-se como uma solução inadequada para imagens agrícolas de regiões com uma aparência mais homogênea.

O detector de característica SIFT mostrou-se como o método mais adequado. Obteve o segundo melhor tempo de processamento e o segundo melhor aproveitamento das imagens de entrada para compor os mosaicos, utilizando 82% delas no pior dos casos (Tabela 4.6). Além disso, não houveram problemas de descontinuidades de objetos. Como apontado por Ghosh & Kaabouch (2016), o detector de característica SIFT é eficiente para imagens de alta resolução e oferece invariância à diversas transformações. Portanto, provou ser uma solução confiável para imagens agrícolas obtidas por meio de RPA.

Levando em conta as considerações anteriores, este trabalho pode servir como base para outros estudos avaliativos com diferentes detectores e descritores de características, como ORB, *Fast Retina Keypoint* (FREAK) (ALAHI; ORTIZ; VANDERGHEYNST, 2012), *Maximal Self-Dissimilarity* (MSD) (TOMBARI; STEFANO, 2015), etc. Outras estratégias para melhorar o conjunto de correspondências de características podem ser avaliadas ou, até mesmo, somadas às estratégias que foram utilizadas neste trabalho, visando a obtenção de um conjunto de maior qualidade, o que é essencial em situações em que as imagens possuem diversos tipos de transformações. Além disso, pode-se tentar manipular individualmente o desempenho computacional dos métodos para melhor adequá-los ao tipo de imagens utilizado neste trabalho. Com isso, pode ser possível chegar a uma solução livre especificamente adequada e voltada para a criação de mosaicos a partir de imagens agrícolas obtidas por meio de RPA.

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.

ALAHI, A.; ORTIZ, R.; VANDERGHEYNST, P. Freak: Fast retina keypoint. In: **2012 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2012. p. 510–517. ISSN 1063-6919.

ALCANTARILLA, P. F.; NUEVO, J.; BARTOLI, A. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In: **British Machine Vision Conference (BMVC)**. [S.l.: s.n.], 2013.

AZZARI, P.; STEFANO, L. D.; MATTOCCIA, S. An evaluation methodology for image mosaicing algorithms. In: **Advanced Concepts for Intelligent Vision Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 89–100. ISBN 978-3-540-88458-3.

BAGGIO, D. L. *et al.* **Mastering OpenCV 3**. 2. ed. [S.l.]: Packt Publishing Ltd, 2017. ISBN 9781786467171.

BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. In: **Computer Vision – ECCV 2006**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 404–417. ISBN 978-3-540-33833-8.

BEVILACQUA, A.; AZZARI, P. A fast and reliable image mosaicing technique with application to wide area motion detection. In: **Image Analysis and Recognition**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 501–512. ISBN 978-3-540-74260-9.

BIND, V. S. **Robust Techniques for Feature-based Image Mosaicing**. Tese (Doutorado) — National Institute of Technology Rourkela, 2013.

BROWN, M.; LOWE, D. G. Automatic panoramic image stitching using invariant features. **International Journal of Computer Vision**, v. 74, n. 1, p. 59–73, Aug 2007. ISSN 1573-1405.

BURT, P. J.; ADELSON, E. H. A multiresolution spline with application to image mosaics. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 2, n. 4, p. 217–236, out. 1983. ISSN 0730-0301.

CAPEL, D. **Image Mosaicing and Super-resolution**. [S.l.]: Springer-Verlag London, 2004. (Distinguished Dissertations). ISBN 978-1-4471-1049-1 978-0-85729-384-8.

DI GENNARO, S. F. *et al.* Unmanned aerial vehicle (uav)-based remote sensing to monitor grapevine leaf stripe disease within a vineyard affected by esca complex. **Phytopathologia Mediterranea**, v. 55, n. 2, 2016. ISSN 1593-2095.

FEDOROV, D. V. *et al.* Automatic registration and mosaicking system for remotely sensed imagery. In: **Image and Signal Processing for Remote Sensing VIII.** [S.l.: s.n.], 2003. v. 4885, p. 444–451.

FILHO, A. B.; AMORIM, L. **Doenças de plantas tropicais: epidemiologia e controle econômico**. [S.l.]: Ed. Agronômica Ceres, 1996.

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.

GAO, G.; JIA, K. A new image mosaics algorithm based on feature points matching. In: **Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)**. [S.l.: s.n.], 2007. p. 471–471.

GARCÍA, G. B. *et al.* **Learning Image Processing with OpenCV**. [S.l.]: Packt Publishing Ltd, 2015. ISBN 9781783287659.

GARCIA-RUIZ, F. *et al.* Comparison of two aerial imaging platforms for identification of huanglongbing-infected citrus trees. **Computers and Electronics in Agriculture**, v. 91, p. 106–115, 2013. ISSN 0168-1699.

GHOSH, D.; KAABOUCH, N. A survey on image mosaicing techniques. **Journal of Visual Communication and Image Representation**, v. 34, p. 1–11, 2016. ISSN 1047-3203.

GHOSH, D.; KAABOUCH, N.; FEVIG, R. A. Robust spatial-domain based super-resolution mosaicing of cubesat video frames: Algorithm and evaluation. **Computer and Information Science**, v. 7, n. 2, p. 68, 2014.

GHOSH, D. *et al.* Quantitative evaluation of image mosaicing in multiple scene categories. In: **2012 IEEE International Conference on Electro/Information Technology**. [S.l.: s.n.], 2012. p. 1–6. ISSN 2154-0357.

GODOY, C. A. V.; CANTERI, M. G. Efeitos protetor, curativo e erradicante de fungicidas no controle da ferrugem da soja causada por Phakopsora pachyrhizi, em casa de vegetação. **Fitopatologia Brasileira**, scielo, v. 29, p. 97–101, 02 2004. ISSN 0100-4158.

GODOY, C. A. V.; KOGA, L. J.; CANTERI, M. G. Diagrammatic scale for assessment of soybean rust severity. **Fitopatologia Brasileira**, scielo, v. 31, p. 63–68, 02 2006. ISSN 0100-4158.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 4. ed. New York, NY: Pearson, 2017. ISBN 9780133356724.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: **Fourth Alvey Vision Conference**. Manchester, UK: [s.n.], 1988. p. 147–151.

HARTLEY, R.; ZISSERMAN, A. **Multiple View Geometry in Computer Vision**. 2. ed. New York, NY, USA: Cambridge University Press, 2003. ISBN 0521540518.

HIKISHIMA, M. *et al.* Quantificação de danos e relações entre severidade, medidas de refletância e produtividade no patossistema ferrugem asiática da soja. **Tropical Plant Pathology**, scielo, v. 35, p. 96–103, 04 2010. ISSN 1982-5676.

HOWSE, J. *et al.* **OpenCV 3 Blueprints**. [S.l.]: Packt Publishing Ltd, 2015. ISBN 9781784399757.

HUBER, P. J. **Robust Statistics**. New York: Wiley, 1981. (Wiley Series in Probability and Statistics). ISBN 9780471418054.

JOSHI, H.; SINHA, M. K. A survey on image mosaicing techniques. **International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)**, v. 2, n. 2, p. 365–369, 2013.

KAKAES, K. *et al.* **Drones and aerial observation: New technologies for property rights, human rights, and global development**. [S.l.]: New America, 2015.

KAPUR, S. **Computer Vision with Python 3**. [S.l.]: Packt Publishing Ltd, 2017. ISBN 9781788299763.

KAPUR, S.; THAKKAR, N. **Mastering OpenCV Android Application Programming**. [S.l.]: Packt Publishing Ltd, 2015. ISBN 9781783988211.

KARAMI, E.; PRASAD, S.; SHEHATA, M. Image matching using sift, surf, brief and orb: Performance comparison for distorted images. **Proceedings of Newfoundland Electrical and Computer Engineering Conference**, St. John's, Canada, nov. 2015.

LAGANIÈRE, R. **OpenCV 3 Computer Vision Application Programming Cookbook**. 3. ed. [S.l.]: Packt Publishing Ltd, 2017. ISBN 9781786469717.

LEUTENEGGER, S.; CHLI, M.; SIEGWART, R. Y. Brisk: Binary robust invariant scalable keypoints. In: **2011 International Conference on Computer Vision**. [S.l.: s.n.], 2011. p. 2548–2555. ISSN 1550-5499.

LEVENBERG, K. A method for the solution of certain non-linear problems in least squares. **Quarterly of Applied Mathematics**, Brown University, v. 2, n. 2, p. 164–168, 1944. ISSN 0033569X, 15524485.

LI, Z.; ISLER, V. Large scale image mosaic construction for agricultural applications. **IEEE Robotics and Automation Letters**, v. 1, n. 1, p. 295–302, Jan 2016. ISSN 2377-3766.

LIQIAN, D.; YUEHUI, J. Moon landform images fusion and mosaic based on sift method. In: **2010 International Conference on Computer and Information Application**. [S.l.: s.n.], 2010. p. 29–32.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. **International Journal of Computer Vision**, v. 60, n. 2, p. 91–110, Nov 2004. ISSN 1573-1405.

MAHLEIN, A.-K. Plant disease detection by imaging sensors – parallels and specific demands for precision agriculture and plant phenotyping. **Plant Disease**, v. 100, n. 2, p. 241–251, 2016.

MARQUARDT, D. W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. **SIAM Journal on Applied Mathematics**, SIAM, v. 11, n. 2, p. 431–441, 1963.

ROSTEN, E.; DRUMMOND, T. Machine learning for high-speed corner detection. In: **Computer Vision – ECCV 2006**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 430–443. ISBN 978-3-540-33833-8.

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.

SZELISKI, R. Image alignment and stitching: A tutorial. **Foundations and Trends in Computer Graphics and Vision**, Now Publishers Inc., v. 2, n. 1, p. 1–104, 2006.

TARALLO, A. d. S. *et al.* Parallel processing applied to image mosaic generation. In: Universidade Federal Fluminense (UFF). **IX Workshop de Visão Computacional (WVC 2013)**. [S.l.], 2013.

TAZEHKANDI, A. A. **Computer Vision with OpenCV 3 and QT5**. [S.l.]: Packt Publishing Ltd, 2018. ISBN 9781788472395.

TOMBARI, F.; STEFANO, L. D. Interest points via maximal self-dissimilarities. In: **Computer Vision – ACCV 2014**. Cham: Springer International Publishing, 2015. p. 586–600. ISBN 978-3-319-16808-1.

VAPNIK, V.; GOLOWICH, S. E.; SMOLA, A. J. Support vector method for function approximation, regression estimation and signal processing. In: **Advances in Neural Information Processing Systems 9**. [S.l.]: MIT Press, 1997. p. 281–287.

VINCENT, E.; LAGANIERE, R. Matching feature points in stereo pairs: A comparative study of some matching strategies. **Machine Graphics and Vision**, v. 10, n. 3, p. 237–259, 2001.

YANG, L. *et al.* A research of feature-based image mosaic algorithm. In: **2011 4th International Congress on Image and Signal Processing**. [S.l.: s.n.], 2011. v. 2, p. 846–849.

ZITOVÁ, B.; FLUSSER, J. Image registration methods: a survey. **Image and Vision Computing**, v. 21, n. 11, p. 977–1000, 2003. ISSN 0262-8856.