UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LEONARDO SACOMAN VON DENTZ

IMPLEMENTAÇÃO DE UMA ARQUITETURA DISTRIBUÍDA DE MICROSERVIÇOS PARA RETIFICAÇÃO DE MAPAS DE ZONA DE MANEJO COM UTILIZAÇÃO DE PROCESSAMENTO DIGITAL DE IMAGENS

TRABALHO DE CONCLUSÃO DE CURSO

MEDIANEIRA

2019

LEONARDO SACOMAN VON DENTZ

IMPLEMENTAÇÃO DE UMA ARQUITETURA DISTRIBUÍDA DE MICROSERVIÇOS PARA RETIFICAÇÃO DE MAPAS DE ZONA DE MANEJO COM UTILIZAÇÃO DE PROCESSAMENTO DIGITAL DE IMAGENS

Trabalho de conclusão de curso apresentado ao Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Bacharel em Ciência da Computação".

Orientador: Prof. Me. Ricardo Sobjak.

Co-orientador: Prof. Dr. Nelson Miguel Betzek.

MEDIANEIRA

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Coordenação do Curso de Ciência da Computação

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

IMPLEMENTAÇÃO DE UMA ARQUITETURA DISTRIBUÍDA DE MICROSERVIÇOS PARA RETIFICAÇÃO DE MAPAS DE ZONA DE MANEJO COM UTILIZAÇÃO DE PROCESSAMENTO DIGITAL DE IMAGENS

Por

LEONARDO SACOMAN VON DENTZ

Este Trabalho de conclusão de curso foi apresentado às 10:20h do dia 19 de Novembro de 2019 como requisito parcial para a obtenção do título de Bacharel no Curso de Ciência da Computação, da Universidade Tecnológica Federal do Paraná, Câmpus Medianeira. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Me. Ricardo Sobjak
UTFPR - Câmpus Medianeira

Prof. Dr. Nelson Miguel Betzek
UTFPR - Câmpus Medianeira

Prof. Dr. Everton Coimbra de Araujo
UTFPR - Câmpus Medianeira

Prof. Me. Fernando Schütz
UTFPR - Câmpus Medianeira

A folha de aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

DENTZ, Leonardo Sacoman Von. IMPLEMENTAÇÃO DE UMA ARQUITETURA DISTRIBUÍDA DE MICROSERVIÇOS PARA RETIFICAÇÃO DE MAPAS DE ZONA DE MANEJO COM UTILIZAÇÃO DE PROCESSAMENTO DIGITAL DE IMAGENS. 90 f. Trabalho de conclusão de curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2019.

A evolução das tecnologias em geoprocessamento, sistemas de posicionamento global e diversas outras tecnologias, estão possibilitando obter conhecimento mais aprofundado e tomar decisões de forma mais rápida e eficaz nas áreas de cultivo agrícola. A agricultura de precisão (AP) analisa a área agrícola de forma a levar em conta a variabilidade espacial, permitindo fazer o tratamento localizado. Zonas de Manejo (ZM) é uma forma de dividir a área agrícola em grupos homogêneos, permitindo aplicar a AP sem a necessidade de utilizar máquinas e implementos agrícolas instrumentados com equipamentos para aplicação em taxa variável e sistemas de navegação global via satélite. As ZMs são comumente geradas com algoritmos baseados em técnicas de agrupamento de dados, que podem formar grupos que não são viáveis operacionalmente a campo. Retificar estas ZMs é uma etapa importante, pois visa suavizar os delineamentos com a retirada de manchas nos mapas de ZM. Portanto, foram desenvolvidas rotinas para retificação de mapas de ZM baseadas na aplicação de filtros de processamento digital de imagem, que são executadas em uma arquitetura distribuída de microserviços Web. Pode-se testar a rotina com diferentes parâmetros, sendo que, conseguiu-se um bom desempenho de processamento quando comparado à uma rotina (BETZEK et al., 2018) considerada como referência.

Palavras-chave: Agricultura de precisão, Balanceamento de carga, Eureka

ABSTRACT

DENTZ, Leonardo Sacoman Von. IMPLEMENTATION OF A DISTRIBUTED MICROSERVICE ARCHITECTURE FOR MANAGEMENT ZONE MAP RECTIFICATION USING DIGITAL IMAGE PROCESSING. 90 f. Trabalho de conclusão de curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2019.

Evolving technologies in geoprocessing, global positioning systems and many other technologies are making it possible to gain more in-depth knowledge and make faster and more effective decisions in agricultural areas. Precision agriculture (PA) analyzes the agricultural area to take into account spatial variability, allowing for localized treatment. Management Zones (ZM) is a way of dividing the agricultural area into homogeneous groups, allowing PA to be applied without the need to use agricultural machines and implements with equipment for variable rate application and global satellite navigation systems. ZMs are commonly generated with algorithms based on data grouping techniques, which can form groups that are not operationally viable in the field. Rectifying these ZMs is an important step as it aims to smooth out the delineations by removing stains on the ZM maps. Therefore, routines for rectifying ZM maps based on the application of digital image processing filters, which are executed in a distributed microservices architecture, were developed. The routines can be tested with different parameters, good processing performance were obtained when compared to a routine (BETZEK et al., 2018) considered as reference.

Keywords: Precision Agriculture, Load Balancing, Eureka

LISTA DE FIGURAS

LISTA DE SIGLAS

AP	Agricultura de Precisão
API	Applications Programming Interface
CSV	Comma-separated values
EJB	Entreprise JavaBeans
GPS	Sistema de Posicionamento Global
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LXC	Linux Container
OSS	Open Source Software
PDI	Processamento Digital de Imagens
REST	Representational State Transfer
SI	Índice de Suavidade
ZM	Zona de Manejo

SUMÁRIO

1 INTRODUÇÃO

A evolução das tecnologias em geoprocessamento, sistemas de posicionamento global e diversas outras tecnologias, estão possibilitando à agricultura uma maneira diferenciada de se examinar a propriedade rural, deixando de ser apenas uma e tornando-se diversas propriedades dentro da mesma, porém, com atributos individuais. Esta transformação no modo de fazer agricultura está tornando o produtor rural gradativamente em um empresário rural, por gerenciar cada dia mais a linha de produção (TSCHIEDEL; FERREIRA, 2002).

Esta mudança é essencial para que se entenda que a propriedade não é homogênea, e sim, que se trate cada parte conforme as suas necessidades, fazendo assim que o produtor tenha o conhecimento de cada parte da linha de produção ou cada metro quadrado da sua propriedade.

Ao processar grandes volumes de dados espaciais, tais como mapas interpolados de atributos agrícolas ou mapas de Zonas de Manejo (ZMs), pode haver um custo de execução computacional agregado, em virtude da complexidade do algoritmo que está sendo aplicado. O algoritmo de retificação de ZMs, proposto por Betzek et al. (2018), é um desses casos. A execução pode tornar-se muito demorada em virtude do número de pixeis, do tamanho da máscara de filtragem escolhida e dos aspectos técnicos do computador que executa o procedimento. Isto pode inviabilizar a disponibilização do procedimento em um produto de software, para ser utilizado pela comunidade. A busca por estratégias para minimizar o impacto causado pela complexidade de operação do algoritmo é importante.

Neste sentido, por meio da biblioteca OpenCV e com a aplicação de filtros de processamento digital de imagens, mostra-se que a rotina de retificação de ZMs tenha seu tempo de execução otimizado. A disponibilização das novas rotinas na forma de uma Web API em microserviço, além da replicação em um ambiente distribuído, permite que seja utilizado como um componente reutilizável de software. Assim, outros desenvolvedores de software podem criar soluções de agricultura digital para uso por agricultores, pesquisadores e comunidade.

1.1 OBJETIVO GERAL

Desenvolver uma rotina computacional a fim de otimizar o processo de retificação de mapas de ZMs por meio de técnicas de processamento digital de imagem em uma arquitetura de microserviços distribuídos.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos do trabalho foram:

- Reescrever a rotina computacional para retificação de zonas de manejo em linguagem Python com auxílio da biblioteca OpenCV;
- Implementar métodos de retificação baseados nos filtros de mediana e erosão/dilatação;
- Implementar uma API em microserviço para executar o processo de retificação de zona de manejo;
- Implantar a API em uma arquitetura distribuída de microserviços;
- Avaliar o desempenho da rotina computacional e a qualidade da ZM retificada.

1.3 JUSTIFICATIVA

Observa-se no campo da agricultura uma necessidade de desenvolver ferramentas digitais que transformem operações complexas em rotinas simplificadas, favorecendo o processo decisório, reduzindo os impactos ambientais, reduzindo o uso de insumos agrícolas, auxiliando o agricultor a tomar decisões mais eficientes e consequentemente, aumentando sua produtividade.

Zonas de manejo são importantes para que o agricultor atue de maneira mais sustentável na sua área de produção agrícola, aplicando conceitos relacionados com a agricultura de precisão, a fim de auxiliar o agricultor no melhor tratamento do solo,

possivelmente na redução de custos com insumos agrícolas e no aumento da produtividade de sua propriedade.

A retificação de ZMs é uma importante etapa no processo de definição de ZMs. Diversas vezes ocorrem manchas no mapa, que inviabilizam a operacionalização a campo. Neste sentido, existem técnicas de processamento digital de imagens que podem ser utilizadas para remover essas manchas indesejadas, fazendo assim com que a ZM gerada possa ser utilizada na lavoura sem maiores problemas.

Ao trabalhar com grande volume de dados de áreas agrícolas ou mapas de ZMs, geralmente há um custo de processamento computacional agregado. Este é o caso do algoritmo de retificação de ZMs, proposto por Betzek et al. (2018), que quando utilizado com um grande volume de dados acaba se tornando lento. Portanto, é necessário otimizar esse tipo de algoritmo para obter performance na sua execução mesmo quando utilizado com grande volume de dados.

1.4 ORGANIZAÇÃO DO DOCUMENTO

Esse documento está organizado da seguinte forma: No Capítulo 2 é apresentado todo o conteúdo necessário para a compreensão do projeto. No Capítulo 3 é apresentado todo o material necessário para o desenvolvimento do projeto e todos os métodos utilizados para o desenvolvimento e avaliação da rotina. No Capítulo 4 são apresentados os resultados obtidos a partir do trabalho desenvolvido e os comparativos realizados entre a rotina desenvolvida durante o trabalho e a rotina desenvolvida por Betzek et al. (2018). No Capítulo 5 são apresentadas as conclusões do trabalho.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo são abordados os conceitos de agricultura de precisão, zonas de manejo, índice de suavidade, retificação de zonas de manejo, processamento digital de imagens e microserviços.

2.1 AGRICULTURA DE PRECISÃO

Na agricultura, a área de plantio pode se apresentar bastante heterogênea, sendo assim, é necessário manejar os insumos de maneira diferenciada ao longo da lavoura. Agricultores que possuem pequenas áreas de cultivo e trabalham de forma manual conseguem perceber este fenômeno mais facilmente. Contudo, à medida que se aumenta a quantia de terras, tal percepção se torna mais difícil e as variações de solo acabam por se tornar imperceptíveis (MOLIN et al., 2015).

Com a intenção de explorar e propor soluções para tais deficiências, é utilizada uma prática de manejo de solo conhecida como Agricultura de Precisão (AP). Segundo Molin et al. (2015) AP é uma abordagem que tem como princípio a coordenação das lavouras com um detalhamento elevado, permitindo tratá-la de maneira individual e suprindo assim as suas deficiências. De acordo com Zhang e Kovacs (2012) a AP é composta pela aplicação de técnicas geoespaciais e sensores, como imagens aéreas e GPS, para realizar a identificação de alterações na lavoura e assim utilizar diferentes abordagens para tratar os diferentes problemas.

2.2 ZONA DE MANEJO

Zona de Manejo (ZM) é uma sub-região do campo que apresenta uma série de fatores limitantes para a produção de forma homogênea, e onde é necessária uma taxa única de um insumo específico (MORAL et al., 2010). ZM é um conceito, cujo objetivo é viabilizar a AP em áreas agrícolas, sem a necessidade de utilizar máquinas e implementos agrícolas instrumentados com equipamentos para aplicação em taxa variável e sistemas de navegação global via satélite.

Após o delineamento das ZMs, o número de amostras necessárias para delimitar a variação do solo pode ser convertido a uma amostra composta por zona. Esta abordagem permite reduzir custos com análises, mantendo a mesma representatividade espacial (FERGUSON; HERGERT, 2009). A aplicação desta técnica também fará com que o uso dos nutrientes seja mais eficiente, fazendo com que o rendimento do solo seja possivelmente afetado de maneira positiva, e também com que o uso excessivo de nutrientes no solo seja reduzido (MOSHIA et al., 2014).

ZMs podem ser geradas por meio de diferentes metodologias. Essas metodologias podem ser baseadas em métodos empíricos ou métodos de análise de agrupamento de dados. Os trabalhos que utilizam abordagens baseadas no agrupamento de dados são os mais utilizados nesta área. Evidencia-se o uso do algoritmo Fuzzy C-means (VITHARANA et al., 2008; BRUBECK-HERNANDEZ et al., 2019; OLDONI et al., 2019) e K-means (BRUBECK-HERNANDEZ et al., 2019; BARBOSA et al., 2019). Gavioli et al. (2019) testou 18 métodos de agrupamento para delinear ZMs e verificou que pode-se obter ZMs com boa qualidade ao adotar outros métodos de agrupamento.

Há diferentes tipos de dados utilizados no delineamento de ZMs. Destacam-se os trabalhos desenvolvidos utilizando produtividade (MOLIN, 2002; XIANG et al., 2007; ABDUL et al., 2008; FARID et al., 2016), população de invasoras (BAZZI et al., 2007), videografia aérea (ARAÚJO et al., 2005), e a combinação de dados topográficos e de solo (FRAISSE et al., 2001; ZHA et al., 2019).

Para delinear ZMs, é necessário seguir um protocolo. Gavioli et al. (2019) utilizou um protocolo com as seguintes fases (Figura 1):

1. Processamento de dados, emprega uma variedade de técnicas para encontrar e remover pontos discrepantes do conjunto de dados;
2. Normalização dos dados, visa avaliar os dados a fim de encontrar e remover anomalias que possam atrapalhar o processo de geração das zonas de manejo;
3. Seleção de variáveis, possui papel importante para facilitar o processo de clusterização e

delimitação das zonas de manejo;

4. Interpolação dos dados, tem como objetivo aumentar exponencialmente o número de pontos cobrindo as áreas que não houve amostragem para que as zonas de manejo sejam contínuas e suaves;
5. Aplicação de métodos para delineamento de zonas de manejo, visam dividir os pontos similares de uma área agrícola em grupos ou classes;
6. Retificação das zonas de manejo, se utilizam de técnicas como a aplicação de filtros de moda, mediana, dilatação e erosão para efetuar a remoção de manchas deixadas nos mapas temáticos durante a fase de delineamento;
7. Avaliação das zonas de manejo, podem ser aplicados diversos métodos como a redução de variância, o índice de desempenho de fuzzy, entre outros para validar se as zonas de manejo geradas possuem exatidão.

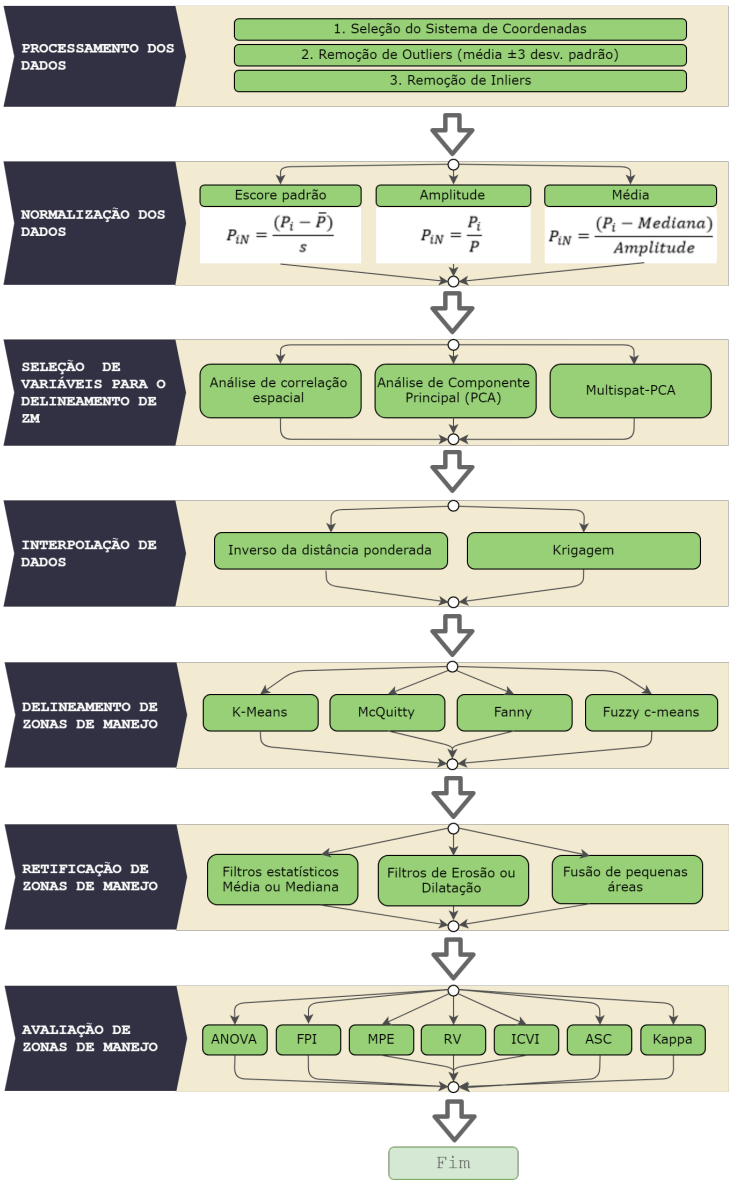

Figura 1 – Fluxograma do protocolo para delinear zonas de manejo.
Fonte: Adaptado de Gavioli et al. (2019).

2.3 ÍNDICE DE SUAVIDADE

Para avaliar a qualidade das ZMs geradas utilizando a rotina de retificação construída durante este trabalho, foi utilizado o índice de suavidade (SI, Eq.1) que calcula com que frequência existe a mudança de classes dentro da ZM gerada nas direções verticais, horizontais e

diagonais, pixel a pixel. Caso a ZM seja uma área totalmente homogênea, o índice de suavidade apresentado será de 100%, devido a inexistência da mudança de classe na ZM. Quanto mais mudanças de classe na ZM, mais próximo de zero o índice de suavidade se apresenta (GAVIOLI et al., 2019).

$$SI = 100 - (\frac{\sum_{i=1}^{k} NM_{Hi}}{4P_H} + \frac{\sum_{j=1}^{k} NM_{Vj}}{4P_V} + \frac{\sum_{l=1}^{k} NM_{DDl}}{4P_{DD}} + \frac{\sum_{m=1}^{k} NM_{DEm}}{4P_{DE}}) * 100 \quad (1)$$

em que, NM_{Hi} é o número de mudanças na linha i (horizontal); NM_{Vj} é o número de mudanças na coluna j (vertical); NM_{DDl} é o número de mudanças na diagonal direita l (DD); NM_{DEm} é o número de mudanças na diagonal esquerda m (DE); k é o número máximo de pixeis em uma linha, coluna ou diagonal; P_H é o número de possibilidade de mudanças na horizontal; P_V é o número de possibilidades de mudanças na vertical; P_{DD} é o número de possibilidades de mudanças na diagonal direita; e P_{DE} é o número de possibilidade de mudanças na diagonal esquerda (GAVIOLI et al., 2016).

2.4 RETIFICAÇÃO DE ZONAS DE MANEJO

Diversas vezes os mapas de ZM apresentam pequenas manchas dentro das classes. Isso dificulta a mecanização para a aplicação de fertilizantes e defensivos à taxa variada. Diversas soluções foram propostas para tentar resolver este problema. Lowrance (2014) desenvolveu um sistema que tenta suavizar as ZMs utilizando um método de fusão de polígonos. Xiang et al. (2007) propôs que se utilizasse uma filtragem pela maior quantidade de valores iguais entre os vizinhos (moda). Pramanik et al. (2013), propuseram a absorção das manchas por zonas maiores utilizando a atribuição de pesos para os pixeis.

Para realizar a retificação da zona de manejo é necessário um conjunto de dados (Dataset) de uma área agrícola. Este dataset é submetido à uma rotina que utiliza-se de filtros para remover as manchas contidas na ZM. Essas rotinas podem ser parametrizadas de diversas formas como: o número de pixeis utilizados pela rotina (tamanho da janela) e a disposição dos pixeis (formato da janela).

Também é importante levar em consideração o tamanho da janela de pixeis utilizada para realizar a filtragem, pois esta é responsável por identificar a quantidade de pixeis que serão considerados na filtragem (BETZEK et al., 2017).

É importante que a ZM esteja o mais suave possível para que o delineamento das classes esteja bem definido facilitando a aplicação da AP.

2.5 PROCESSAMENTO DIGITAL DE IMAGENS

Gonzalez e Woods (2009), definiram que o Processamento Digital de Imagens (PDI) engloba "processos cujas entradas e saídas são imagens e, além disso, envolve processos de extração de atributos de imagens até - e inclusive - o reconhecimento de objetos individuais."

O processamento de imagens é definido em três etapas: pré-processamento, técnicas de realce e técnicas de classificação. Meneses et al. (2012), afirmaram que, quando se deseja obter informações sobre uma imagem, geralmente é necessário que se aplique correções, que são realizadas no pré-processamento, a fim de eliminar ou reduzir os erros presentes, melhorando assim, a qualidade visual da imagem. Estas correções podem ser radiométrica, atmosférica ou geométrica.

Outra etapa do processamento de imagens se refere a técnicas de realce das imagens, geralmente este procedimento se resume na aplicação de realce de contraste, a fim de melhorar a qualidade dos elementos observados a partir das peculiaridades de suas características.

A outra etapa do processamento digital se refere ao método de classificação das imagens digitais. Tais técnicas, segundo Meneses et al. (2012), possibilitaram que o processo de classificação se tornasse mais fácil, exigindo menos esforço de quem analisa a imagem e ao mesmo tempo elimina erros de interpretação. A utilização de PDI para o delineamento de ZMs já se mostrou bastante eficaz, como no trabalho desenvolvido por Zhang e Kovacs (2012), no qual é utilizado PDI juntamente com imagens aéreas de uma região para realizar a definição das zonas de manejo.

2.6 MICROSERVIÇOS

O termo microserviços refere-se à uma arquitetura de desenvolvimento de aplicações (NAMIOT; SNEPS-SNEPPE, 2014). Microserviços podem ser considerados

aplicações de pequeno porte, criadas como serviços e que atuam em grupo (NEWMAN, 2015). São semelhantes a um componente, ou seja, podem ser melhorados ou substituídos de maneira independente, sem afetar o sistema por completo.

Diferentemente de uma estrutura monolítica, onde toda a aplicação está em um único lugar, as aplicações que utilizam a estrutura de microserviços possuem diversos serviços onde cada um desempenha sua própria função (Figura 2).

A arquitetura de microserviços deve ser leve e independente, e deve dispor de uma interface bem definida, diretamente sobre o protocolo de comunicação *Hypertext Transfer Protocol* (HTTP), para facilitar a comunicação entre serviços (NAMIOT; SNEPS-SNEPPE, 2014). Nesse modelo de arquitetura, os limites de atuação de um serviço podem ser mapeados diretamente para os limites das regras de negócio, tornando assim, fácil encontrar o código responsável pela execução de uma determinada funcionalidade (NEWMAN, 2015). Além do mais, é simples de perceber que o resultado de uma mudança em uma funcionalidade é limitado às fronteiras do serviço onde ela está.

Uma unidade deve ter apenas uma responsabilidade, seja uma classe, uma função ou um serviço. Em nenhum momento devem ter duas unidades que compartilham uma responsabilidade ou uma unidade ter mais de uma responsabilidade (RAJESH, 2016).

Uma vantagem de se usar arquitetura monolítica é a facilidade de implantação e escalonamento da aplicação, comparado a arquitetura de microserviços, onde é necessário fazer a implantação de vários serviços. Porém, microserviços possuem vantagens quando consistência e integridade dos dados são necessárias, pois para as aplicações monolíticas, essa é uma tarefa difícil. O escalonamento de sistemas monolíticos tem um custo muito alto, pois é necessário replicar todo o sistema, mesmo que apenas algumas funcionalidades sejam necessárias, bem como planejar uma estratégia de integridade dos dados entre as replicações (AMARAL, 2017).

2.6.1 Spring Cloud Netflix

Spring Cloud Netflix é um conjunto de ferramentas e serviços utilizados para criação de aplicações que utilizam a arquitetura de microserviços. Este grupo de ferramentas fazem parte do Netflix *Open Source Software* (OSS) e possibilitam a criação de aplicações usando recursos de armazenamento de dados na nuvem, mantendo o foco em escalabilidade,

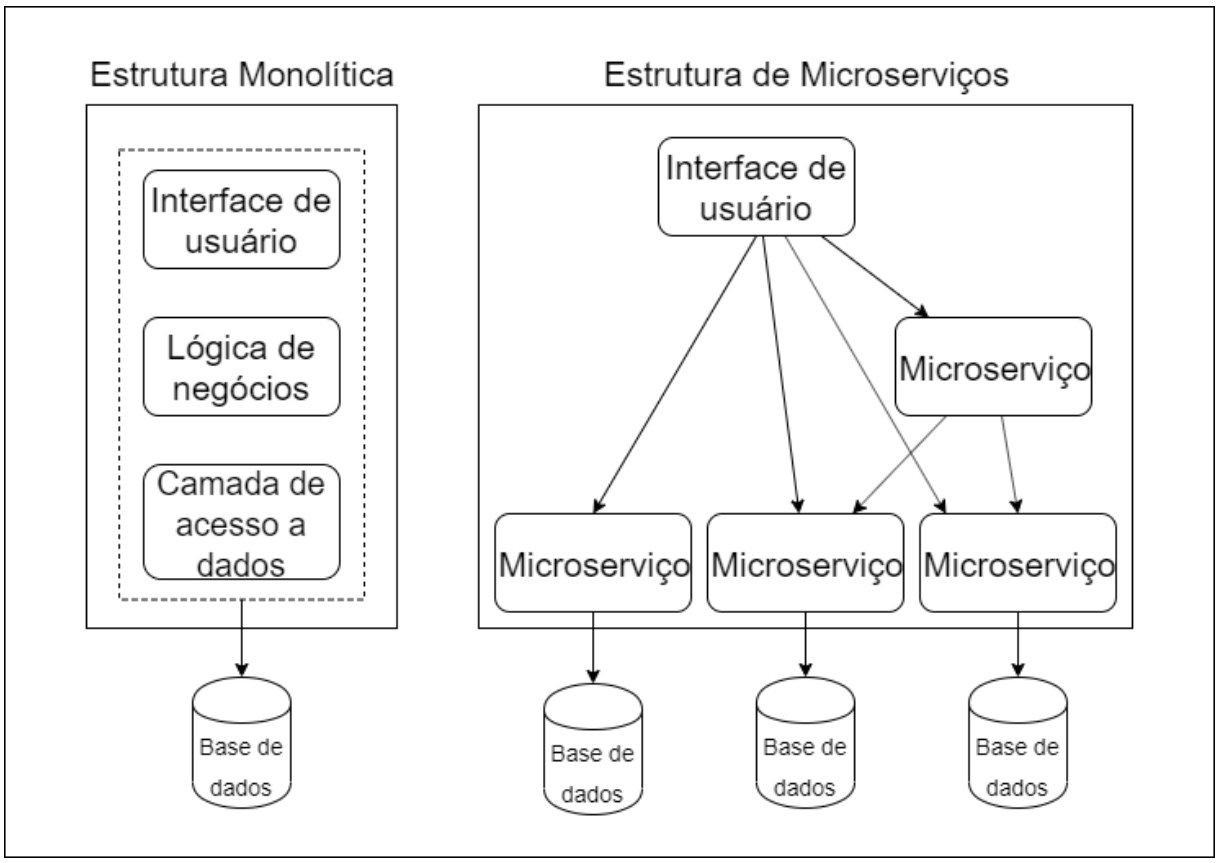

Figura 2 – Arquitetura Monolítica X Arquitetura de Microserviços.
Fonte: Adaptado de (REDHAT, 2019)

confiabilidade, desempenho, segurança e outras características. Para executar de maneira eficiente as tarefas propostas, o Spring Cloud Netflix conta com ferramentas como Eureka, Ribbon, Hystrix e Zuul (PIVOTAL, 2018).

O Eureka é um utilitário que tem os serviços de registrador e de descoberta para microserviços. Ao iniciar sua operação, o Eureka registra todos os microserviços disponíveis e suas devidas portas (Figura 3), e dispara alertas para checar o funcionamento dos mesmos. Quando é realizada uma requisição para algum dos microserviços, o Eureka é responsável por direcionar a requisição para o microserviço adequado e posteriormente, enviar a resposta recebida para o cliente (LONG, 2015).

O Ribbon é um balanceador de cargas que trabalha com a tomada de decisões, ou seja, qual instância do sistema deverá atender a próxima requisição ou até mesmo quando não existe mais instâncias disponíveis, iniciar uma nova instância de um determinado serviço para atender a demanda do sistema, e após a queda na demanda, o mesmo finaliza as instâncias que não são mais necessárias (SALERNO, 2016). O Ribbon utiliza regras pré-definidas para escolher como a instância deverá ser escolhida, mas também oferece a possibilidade de implantação de regras

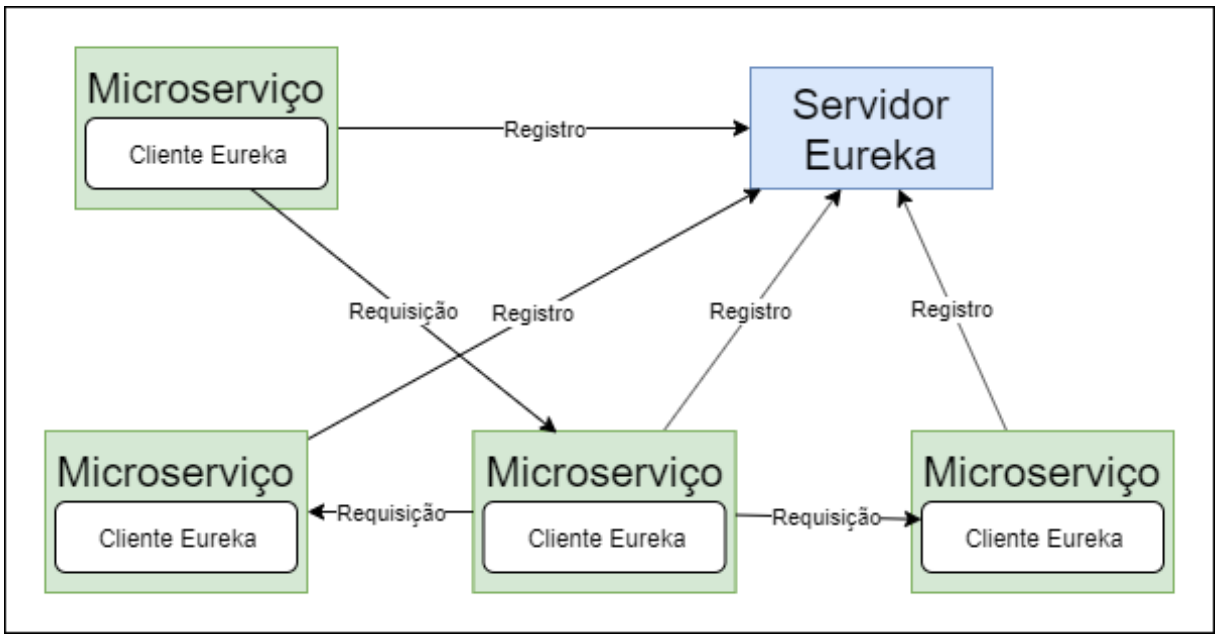

Figura 3 – Funcionamento do Eureka Server para descoberta de serviços.

Fonte: Adaptado de (SILVA, 2019).

customizadas (SOUZA, 2018). As regras já definidas na biblioteca são:

- Round Robin: Sempre chama a próxima instância registrada na lista de servidores;
- Response Time Weighted: Utiliza a instância com menor tempo de resposta;
- Random Load Balancing: Escolhe uma instância aleatoriamente.

Dentre os principais recursos oferecido pelo Ribbon, destacam-se:

- Funcionalidades para clouds como filtrar instâncias baseando-se em regiões;
- Integração com Service Discovery (Eureka);
- Resistência a falhas utilizando mecanismo de ping antes de realizar a requisição real, ou utilizando o padrão Circuit Breaker;
- REST Client integrado ao Load Balancer.

Hystrix é uma biblioteca desenvolvida para controlar a interação entre serviços distribuídos, gerando maior tolerância a latência e falhas. Para conseguir isso, o Hystrix segrega os pontos de comunicação entre os serviços, suprimindo os erros em cascata, e então, proporciona outras opções para finalizar a tarefa, fazendo com que a resistência a falhas do sistema aumente (CHRISTENSEN, 2012).

A resistência é o fator principal que este microserviço oferece, pelo fato de poder consumir informações para gerar alertas, e possuir um painel de controle que mostra como cada microserviço está se comportando, contendo erros, requisições, falhas e até mesmo mostrar a origem de um possível problema (CHRISTENSEN, 2012).

Zuul é o portão principal para todas as requisições feitas para os microserviços que estão sendo executados no back-end, pois o mesmo atua como um gateway. Esse software desenvolvido pela Netflix funciona como um software de monitoramento e de roteamento dinâmico, contendo dois fatores fundamentais, que são a segurança e resiliência. O mesmo consegue trabalhar com envio de requisições para diversos grupos simultaneamente. Como o Zuul centraliza grande volume de tráfego, existe uma série de filtros que este Framework utiliza para facilitar ainda mais a transmissão de informação. A primeira é a autenticação, devido ao fato de o Zuul sempre verificar se a requisição está autenticada ou se o conteúdo condiz com a assinatura do método. O monitoramento é outro fator importante, pois com ele é possível verificar todo o caminho percorrido por uma requisição. O roteamento dinâmico separa as regras de negócio em novas instâncias de acordo com a utilização e demanda. (ZUUL, 2018).

3 MATERIAL E MÉTODOS

Este Capítulo apresenta os materiais necessários para o desenvolvimento deste trabalho e quais são os métodos que serão utilizados para o desenvolvimento da rotina proposta, assim como dos testes.

3.1 ROTINA PARA RETIFICAÇÃO DE ZONA DE MANEJO

Para retificar mapas de ZMs, foi implementada uma rotina computacional baseada na aplicação de filtros de PDI. Esta rotina visa expandir as funcionalidades abordadas por Betzek et al. (2018), que implementaram rotinas de PDI diretamente no banco de dados PostgreSQL, utilizando a linguagem procedural PL/pgSQL.

Neste trabalho, a rotina de retificação de mapas foi implementada na linguagem de programação Python e utilizou a biblioteca OpenCV para executar o processamento das imagens (Figura 4).

Cada função da rotina fará a aplicação de um filtro de processamento de imagem sobre um mapa de ZM fornecido como entrada. O mapa de ZM corresponde a um conjunto de dados espaciais (vetoriais). Cada unidade do mapa representa um pixel, que tem uma posição geográfica e um valor de classe. Para retificar as ZMs, ou seja, aplicar um filtro de PDI, os dados vetoriais informados pelo usuário são transformados antecipadamente em uma imagem de formato raster. Os filtros disponíveis na aplicação para realizar a retificação dos mapas de ZM são:

- Mediana: O filtro da mediana é um filtro não linear. Neste filtro, o elemento central da janela é substituído pelo valor que corresponde a mediana de seus vizinhos (Figura 5). Para realizar o cálculo da mediana é necessário ordenar os valores em questão e escolher o valor mediano, ou seja, aquele que divide o conjunto em dois grupos de cardinalidades iguais.

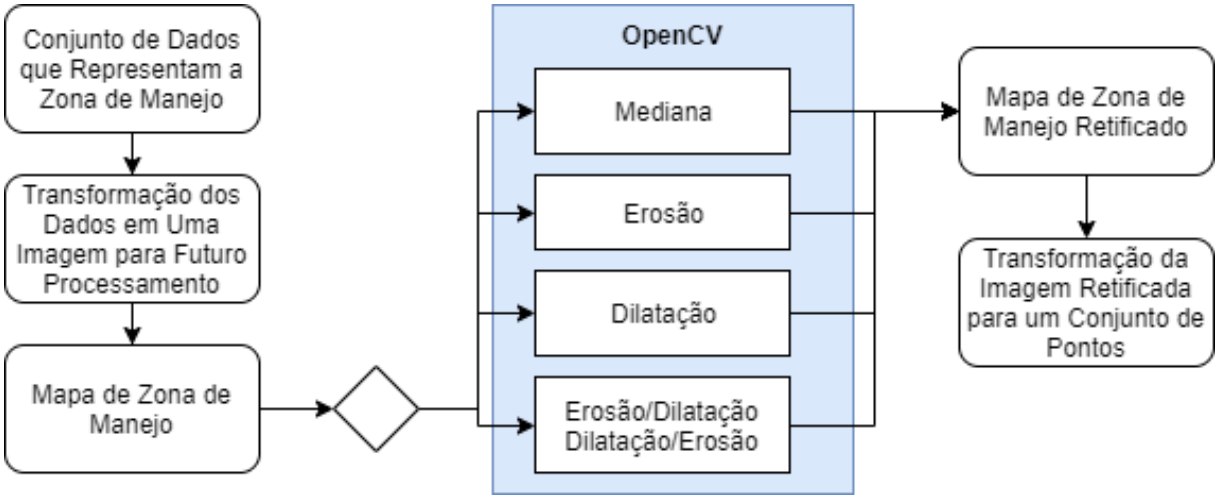

Figura 4 – Fluxo de execução da rotina de retificação de ZMs.

Fonte: Autoria Própria.

Entrada						Resultado após aplicado filtro da Mediana				
100	100	120	100	80	>>	100	100	120	100	80
95	255	120	0	115		95	110	110	110	115
110	120	110	110	80		110	120	110	100	80

Figura 5 – Exemplo de aplicação do filtro da mediana.

Fonte: Autoria Própria.

- Erosão e Dilatação: O filtro morfológico de erosão causa efeitos de erosão das partes claras da imagem (altos níveis de cinza), deixando as imagens mais escuras. O filtro morfológico de dilatação causa efeitos de dilatação das partes escuras da imagem (baixos níveis de cinza), deixando imagens mais claras. Geralmente são encadeados filtros de erosão e dilatação utilizando o mesmo elemento estruturante para obtenção de efeitos de abertura e fechamento.

  A abertura é obtida pelo encadeamento do filtro de erosão, seguido pelo de dilatação (Figura 6), provocando a quebra de istmos, e a remoção de cabos e ilhas.

  O efeito de fechamento é obtido pelo encadeamento do filtro de dilatação, seguido pelo de erosão (Figura 7), provocando a remoção de golfos e fechamento de baías.

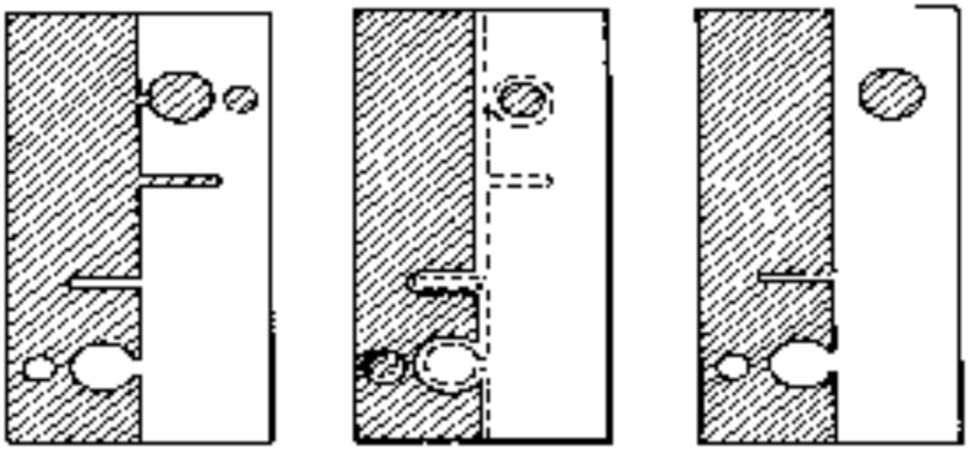

Figura 6 – Efeito do encadeamento do filtro de erosão seguido pelo de dilatação.
Fonte: (INPE, 2019).

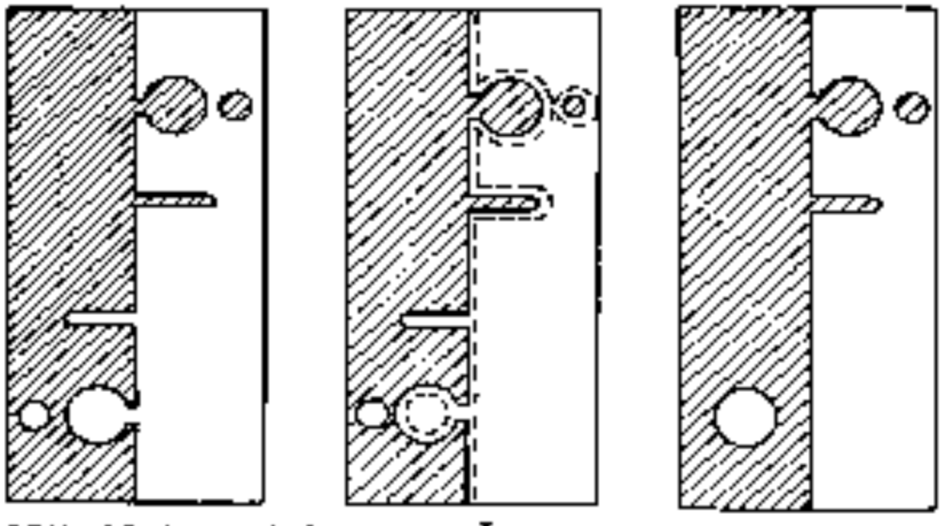

Figura 7 – Efeito do encadeamento do filtro de dilatação seguido pelo de erosão.
Fonte: (INPE, 2019).

3.2 MICROSERVIÇO PARA RETIFICAÇÃO DE ZONA DE MANEJO

A rotina para retificação de ZMs é disponibilizada na forma de uma Web API de arquitetura REST, desenvolvida em linguagem de programação NodeJS. A concepção desta API é ser um microserviço reativo, que pode ser replicado e distribuído por diversos servidores.

O microserviço é responsável por receber um conjunto de dados que representam a ZM, transformá-los em imagem raster, aplicar filtros de PDI para a remoção de manchas deixadas na ZM durante o processo de delineamento, transformar o resultado do processamento

em vetor de pontos e, por fim, retornar o resultado para a API.

Uma estrutura distribuída de microserviços foi criada utilizando as ferramentas Eureka e Http-proxy, para fazer a descoberta de serviços Web e gerenciar o balanceamento de carga das requisições, respectivamente (Figura 8).

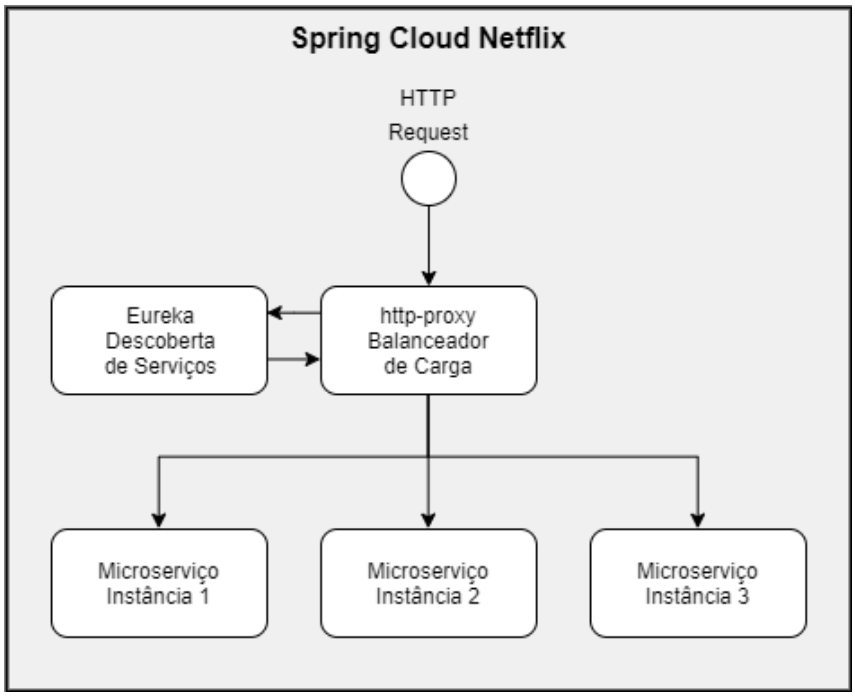

Figura 8 – Funcionamento da ferramenta Spring Cloud Netflix.

Fonte: Autoria Própria.

O microserviço foi implantado em um contêiner do Docker utilizando componentes, como “Docker Compose”, e “DockerFile”, que auxiliam na criação e replicação dos contêineres.

3.3 ESTUDO DE CASO

O estudo de caso teve como objetivo testar a solução computacional implementada, utilizando, para isto, mapas de ZMs já delineadas. Os mapas foram delineados a partir de dados experimentais de duas áreas agrícolas. A primeira (Área A) de 23,8 ha^{-1} (Figura 9), localizada no município de Serranópolis do Iguaçu/PR, na região oeste do estado do Paraná, com localização central geográfica de 25^{o}24’28” S e 54^{o}00’17” O, com altitude média de 355

m. A segunda (Área B) de 19,6 ha^{-1} (Figura 10), localizada no município de Cascavel/PR, na região oeste do estado do Paraná, com localização central geográfica de 24°57'19" S e 53°33'59" O, com altitude média de 706 m.

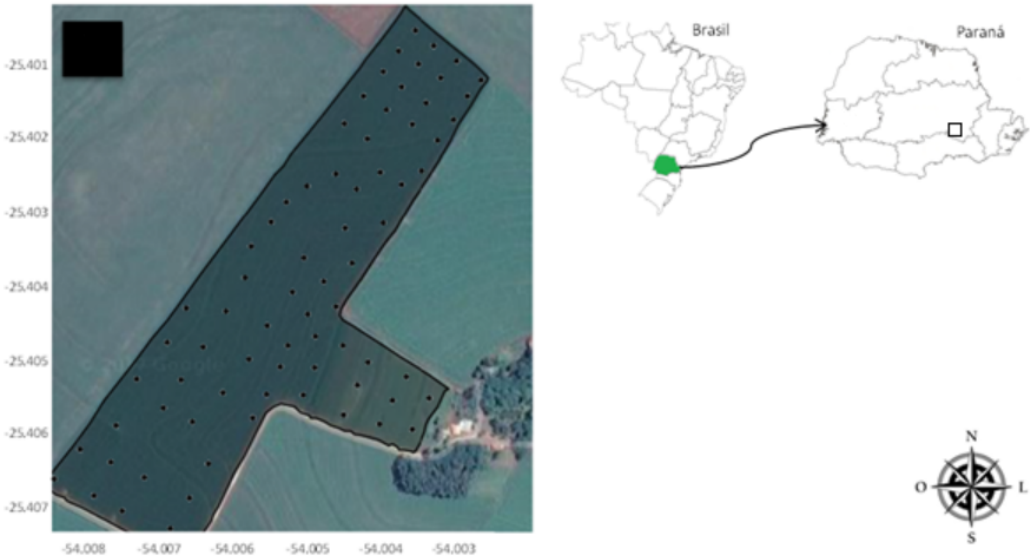

Figura 9 – Localização da área experimental de Serranópolis do Iguaçu (Área A) na região oeste do Paraná.

Fonte: (GOOGLE, INC., 2019)

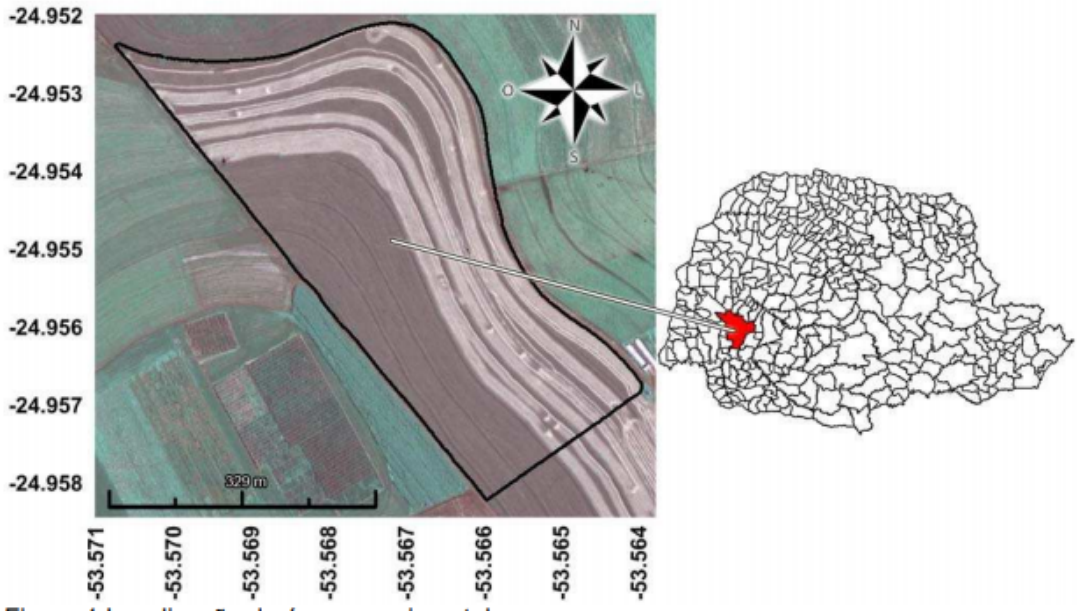

Figura 10 – Localização da área experimental de Cascavel (Área B) na região oeste do Paraná.

Fonte: (GOOGLE, INC., 2019)

Realizaram-se testes de desempenho utilizando os diferentes filtros propostos

(Mediana, erosão, dilatação, erosão/dilatação, dilatação/erosão, Figura 11) e os diferentes tamanhos de janela (Ex: 3x3, 5x5, 7x7, 9x9, 11x11 pixeis) para avaliar o tempo de execução da rotina ao retificar as ZMs com base na combinação desses parâmetros. Foi levado em consideração o tempo de execução desde o início do processo (Requisição HTTP, transformação dos dados, processamento e retorno ao usuário).

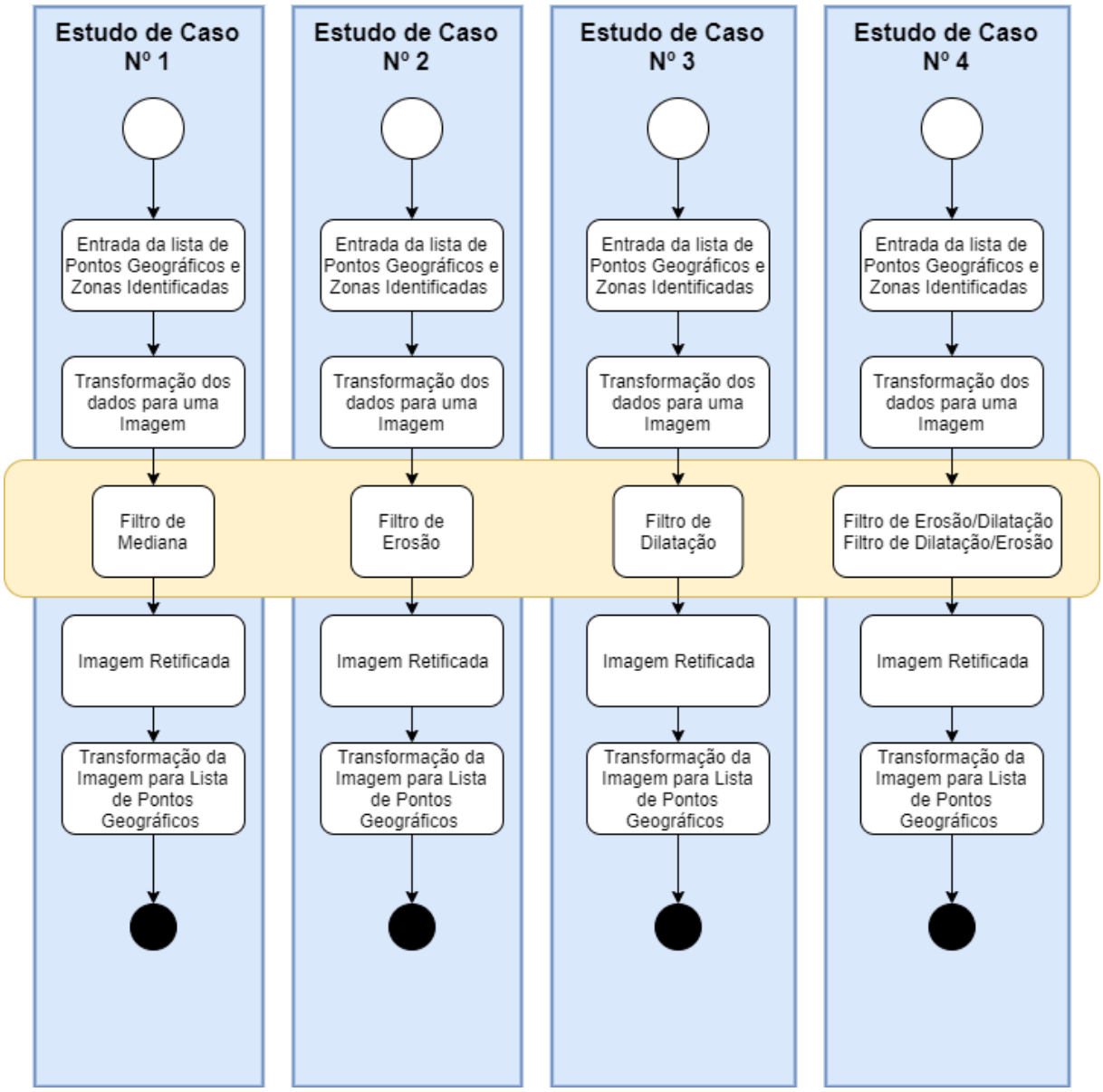

Figura 11 – Fluxograma do estudo de caso de avaliação de desempenho.
Fonte: Autoria Própria.

Para validar o desempenho da rotina de retificação de ZMs, foram realizados 30 testes consecutivos para verificar o tempo de execução e resposta da API. Durante esses testes, foi utilizado uma máquina virtual contendo 1 vCPU e 1GB de memória RAM. Os tempos foram

coletados e foi retirada a média de todos para assim ter um tempo médio de resposta para cada um dos casos.

Também foram executados testes de desempenho na rotina desenvolvida por Betzek et al. (2018) para possibilitar a execução de um comparativo de desempenho entre as rotinas.

Após a verificação dos filtros individualmente, foi realizado uma série de testes de desempenho simulando um ambiente de produção (Figura 12), onde foram enviadas diversas requisições para a API com diversos microserviços sendo executados de maneira distribuída para validar a possibilidade de utilização do microserviço em um software. Para a realização deste teste foram utilizadas 3 máquinas virtuais contendo 1 vCPU e 1GB de memória RAM cada.

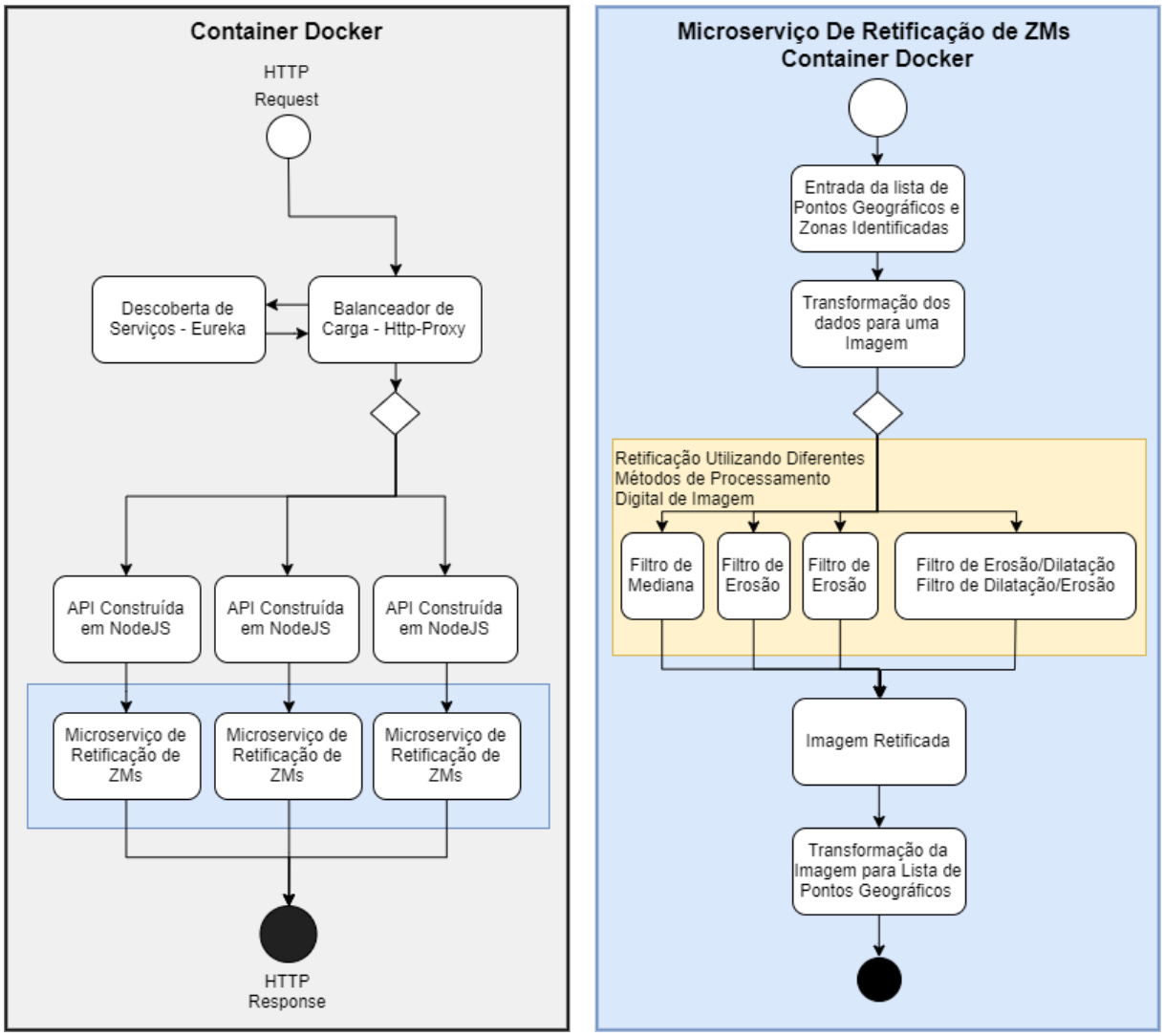

Figura 12 – Fluxograma do estudo de caso de desempenho em ambiente distribuído.
Fonte: Autoria Própria.

Foram utilizados testes de tempo de resposta (Respose Time Testing), que avaliam

o quão rápido o sistema reage para uma “Request” ou “Query” para coletar os resultados. Também foram utilizados scripts desenvolvidos na linguagem JavaScript que possibilitam o envio e recebimento de requisições HTTP. Os resultados foram tabulados e apresentados por meio de gráficos, a fim de ilustrar o desempenho dos filtros em combinação com as janelas que foram utilizadas.

4 RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados os resultados obtidos com o desenvolvimento da rotina de retificação de ZMs e com os testes de desempenho, qualidade e carga realizados.

4.1 API REST

Os dados são recebidos pela API por meio de uma requisição HTTP e devem ser enviados no formato de um objeto JSON. O conjunto de dados (dataset) pode ser enviado em diversos formatos, como: CSV, Objeto JSON ou como uma string, que são dados separados por vírgula e hífen (Código 4.1).

```
{
  method: "open",
  kernelSize: "7",
  kernelFormat: "rect",
  iterations: "2",
  datasetFormat: "csv",
  outputFormat: "image",
  dataset:   "-53.570684808,-24.952417737,1,1,1,3,1
              -53.570616162,-24.952485076,2,1,1,3,1
              -53.570616162,-24.952417737,3,1,1,3,1
              -53.570547517,-24.952619754,4,1,1,3,1"
}
```

Código 4.1 – Exemplo de JSON que deve ser enviado para a API com os dados necessários para realizar a retificação.

Após a recepção na API, os dados são transformados em objetos para facilitar a manipulação dos mesmos. Com os dados em formato de objeto, estes passam por filtros para se

obter as coordenadas geográficas distintas da ZM.

Após esse processo, as coordenadas geográficas são ordenadas de modo crescente e, então, é possível mapear as coordenadas para valores sequenciais. Com isso, é possível plotar esses dados dentro de uma matriz raster (Código 4.2).

```
1  // Popula as Matrizes com os valores corretos nas posicoes
       corretas
2  for (let lat = 0; lat < latitudesLength; lat++) {
3    for (let long = 0; long < longitudesLength; long++) {
4      for (let index = 0; index < dataArray.length; index++) {
5        const element = dataArray[index];
6        if (element.latitude == distinctLatitudes[lat] && element
             .longitude == distinctLongitudes[long]) {
7          matrixC2[lat][long] = parseInt(element.c2);
8          matrixC3[lat][long] = parseInt(element.c3);
9          matrixC4[lat][long] = parseInt(element.c4);
10         matrixC5[lat][long] = parseInt(element.c5);
11       }
12     }
13   }
14 }
```

Código 4.2 – Inserção dos dados na matriz raster

Após esse processamento dos dados, os mesmos são enviados como parâmetro para uma rotina desenvolvida utilizando a linguagem de programação Python e a biblioteca OpenCV, utilizando uma biblioteca do NodeJS, chamada de *child_proccess*. Essa biblioteca possibilita a execução de comandos dentro do sistema operacional. Ela age como uma função assíncrona e aguarda pelo retorno do comando executado.

Quando a rotina de retificação retorna os valores para a API, os mesmos são recebidos no formato de uma string. Então, faz-se necessária a realização de um outro processo que irá converter essa string para o formato adequado para retornar ao usuário.

O usuário pode escolher entre três opções de formato do retorno da requisição. Os formatos disponíveis são: CSV, Objeto JSON ou uma imagem.

4.2 LÓGICA DE RETIFICAÇÃO

A rotina de retificação foi desenvolvida utilizando a linguagem de programação Python, juntamente com a biblioteca OpenCV.

Essa rotina recebe como parâmetros o tamanho da janela (3, 5, 7, 9, 11, ...), o formato da janela (rect, elipse ou cross), o método de retificação que deve ser executado (median, open, close, e openandclose), o número de vezes que deve ser realizada a retificação, e uma string de classes separadas por vírgula e por hífen (Código 4.3).

```
1 dataRows = dataArg.split('-')
2 for row in dataRows:
3   dataColumns = row.split(',')
4   columnInt = []
5   for column in dataColumns:
6     columnInt.append(int(column))
7   formatedData.append(columnInt)
8
9 matriz = 50 * np.array(formatedData, dtype=np.uint8)
```

Código 4.3 – Transformação dos dados recebidos pela rotina em uma matriz que pode ser utilizada pelo OpenCV

Após a recepção dos parâmetros, é executado o método *numpy.array* que transforma a string de classes em uma matriz, que pode ser utilizada pelo OpenCV para a aplicação dos métodos de processamento de imagem. Os métodos que podem ser utilizados são:

- *median* que calcula o valor do ponto central da janela com base na mediana de toda a janela;
- *open* que é a combinação do efeito de erosão seguido pelo efeito de dilatação;
- *close* que é a combinação do efeito de dilatação seguido pelo efeito de erosão; e
- *openandclose* que é a combinação do efeito de erosão seguido por dois efeitos de dilatação e depois pelo de erosão novamente.

Depois de aplicado os métodos de processamento de imagem, a matriz é convertida novamente para uma string de classes. Neste momento, já com as classes retificadas, a string é devolvida para a API, utilizando o standard output do sistema.

4.3 REPLICAÇÃO E BALANCEAMENTO DE CARGA

Para tornar possível e simples a replicação do microserviço, foi desenvolvido um arquivo de configuração do Docker contendo todas as informações necessárias para executar uma nova máquina virtual, com apenas uma linha de comando.

Esse arquivo especifica todos os componentes necessários para a execução da API e da rotina.

A API trabalha em conjunto com o Netflix Eureka, que é uma aplicação que realiza a descoberta de microserviços, e com o *http-proxy* que realiza o balanceamento de carga da aplicação utilizando o algoritmo *round robin*.

Desta maneira, as novas requisições são direcionadas para os servidores que possuem mais recursos disponíveis.

4.4 TESTES DE DESEMPENHO

Ao comparar a rotina desenvolvida durante esse trabalho e a rotina desenvolvida por Betzek et al. (2018) foi realizado e com os resultados foi possível verificar uma melhora significativa de desempenho em favor da rotina desenvolvida durante este trabalho, que apresentou tempo médio de execução de 7,40 segundos enquanto a rotina desenvolvida por Betzek et al. (2018) apresentou tempo médio de 204,58 segundos (Figura 13).

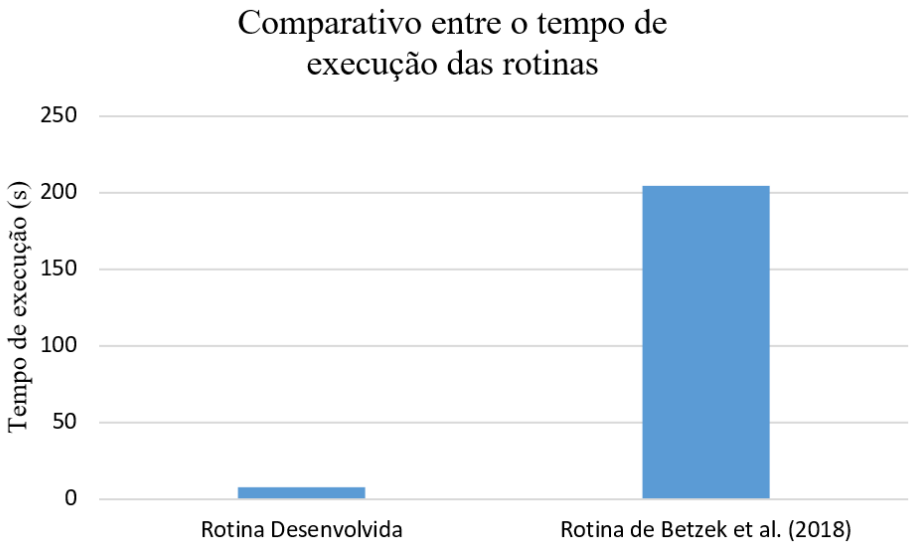

Figura 13 – Tempo médio para execução das rotinas de retificação de ZM nas áreas A e B, comparativo de desempenho entre a rotina desenvolvida e a rotina de Betzek et al. (2018).

Fonte: Autoria Própria.

4.5 TESTES DE CARGA

Foram realizadas 30 requisições consecutivas para verificar a diferença de tempo de execução. Verificou-se que com a utilização de três maquinas, o desempenho se mantém estável durante as 30 requisições. O tempo de resposta aumenta gradativamente em pequenas quantidades (Figura 14) devido as 30 requisições terem sido enviadas para o servidor simultaneamente.

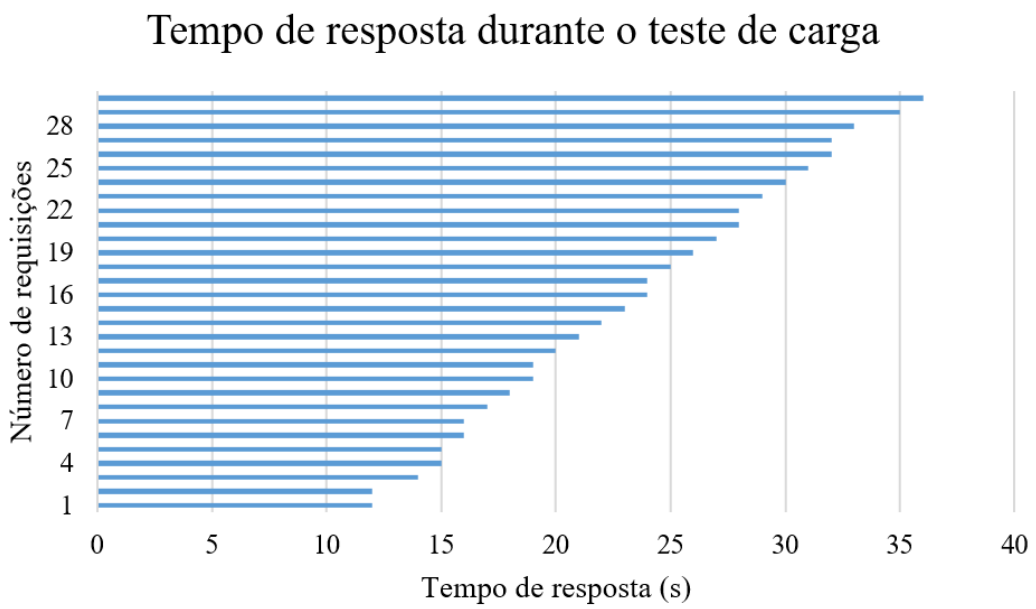

Figura 14 – Tempo de resposta de acordo com a requisição durante o teste de carga.
Fonte: Autoria Própria.

4.6 TESTES DE QUALIDADE

Para realização dos testes foram utilizadas duas ZMs, uma de 8229 pontos (Figura 15) e a outra com 3757 pontos (Figura 16). Foram realizados testes utilizando diferentes filtros e diferentes tamanhos de janelas.

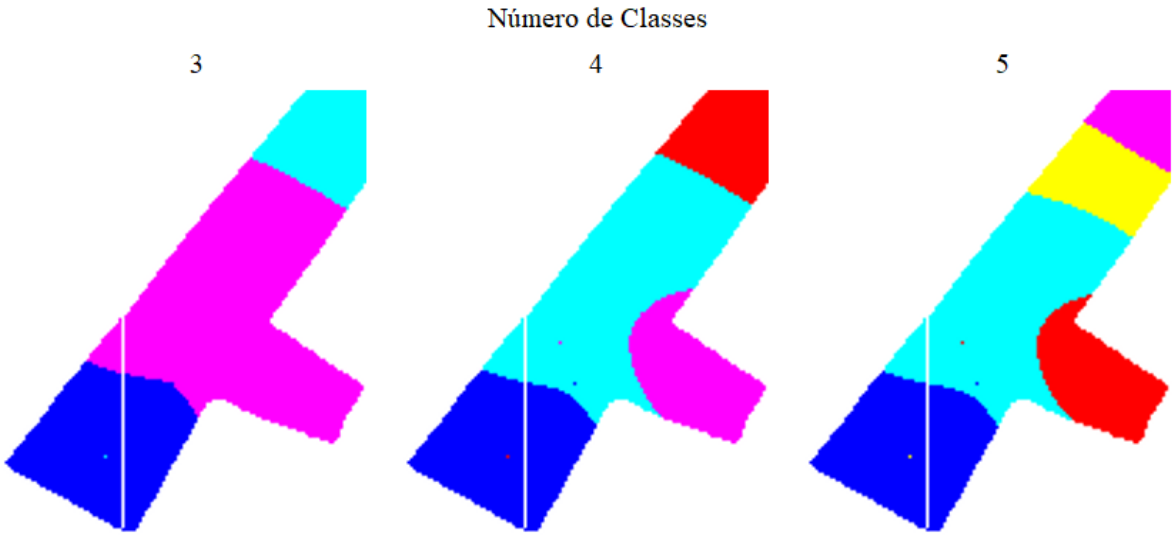

Figura 15 – Mapas de ZM utilizados como referência na área A, divididos em 3, 4 e 5 classes.
Fonte: Autoria Própria.

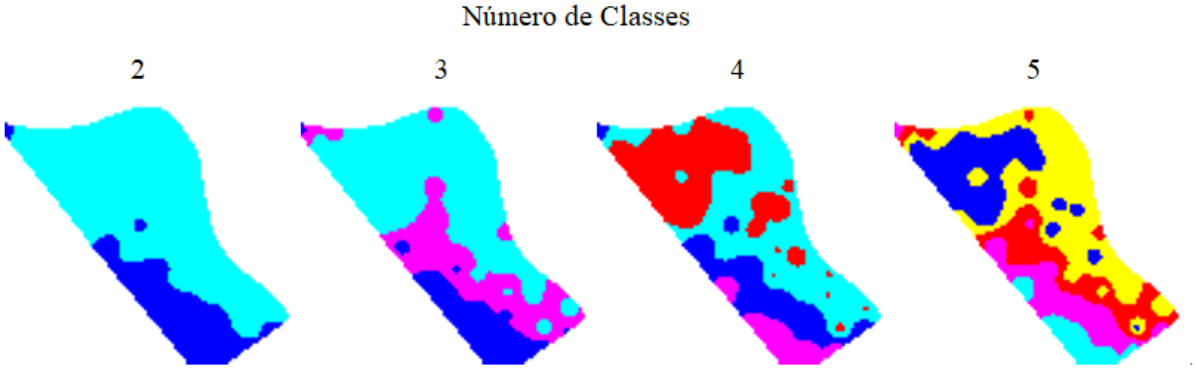

Figura 16 – Mapas de ZM utilizados como referência na área B, divididos em 2, 3, 4 e 5 classes.
Fonte: Autoria Própria.

Um dos filtros utilizados para retificar a ZM foi o filtro da mediana. Observa-se que quando utilizado o filtro de mediana para a retificação da área A (Figura 17) e da área B (Figura 18) existe uma melhora no índice de suavidade da área A (Figura 19) e também da área B (Figura 20) conforme o tamanho da janela é aumentado.

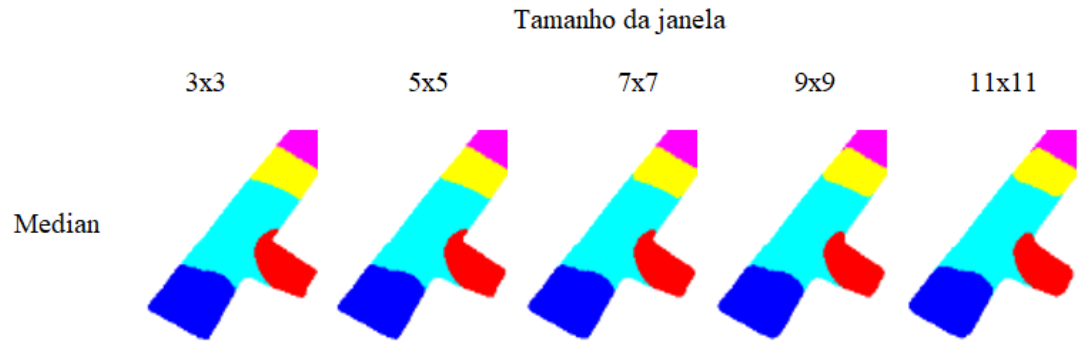

Figura 17 – Mapas da área A retificadas utilizando filtro mediana e tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11.

Fonte: Autoria Própria.

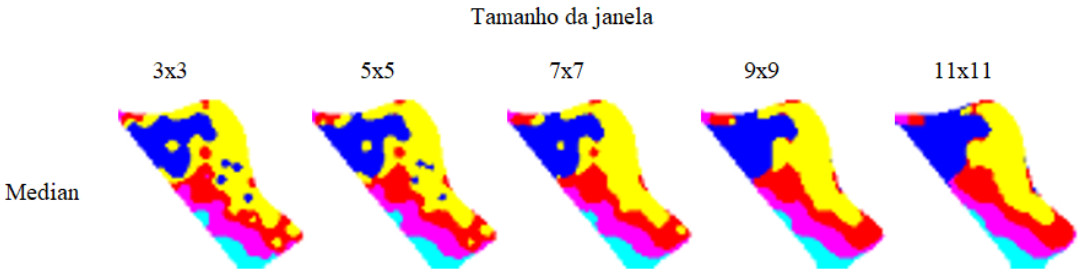

Figura 18 – Mapas da área B retificadas utilizando filtro mediana e tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11.

Fonte: Autoria Própria.

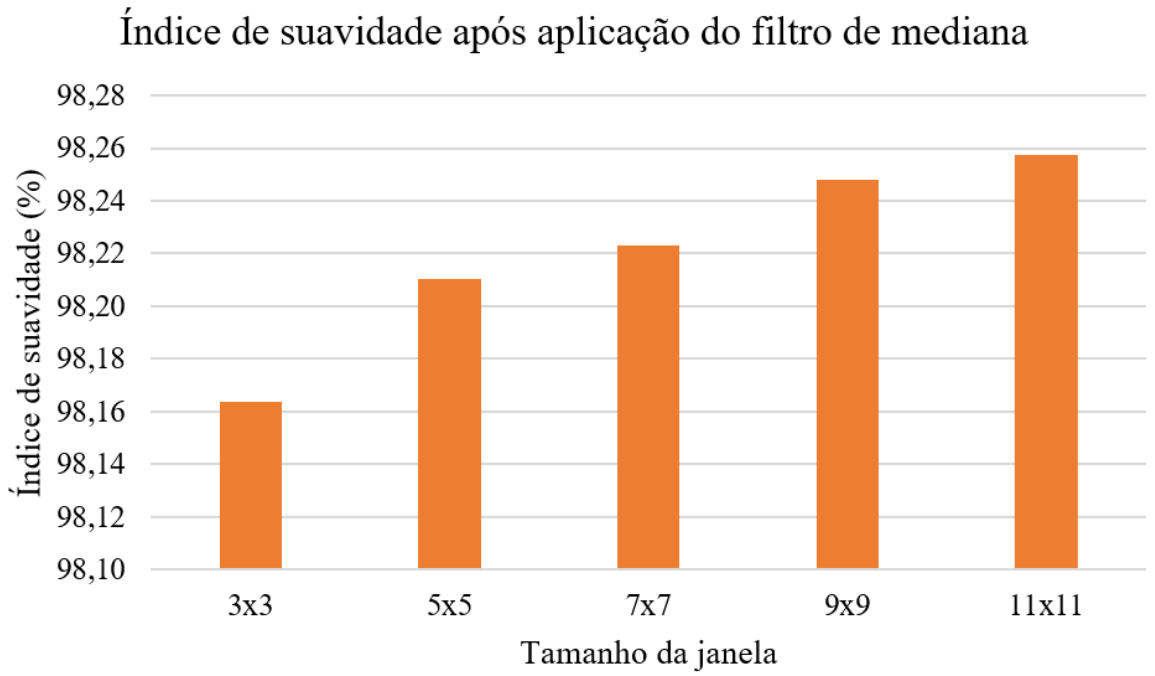

Figura 19 – Índice de suavidade do mapa da área A retificada usando o filtro da mediana.

Fonte: Autoria própria.

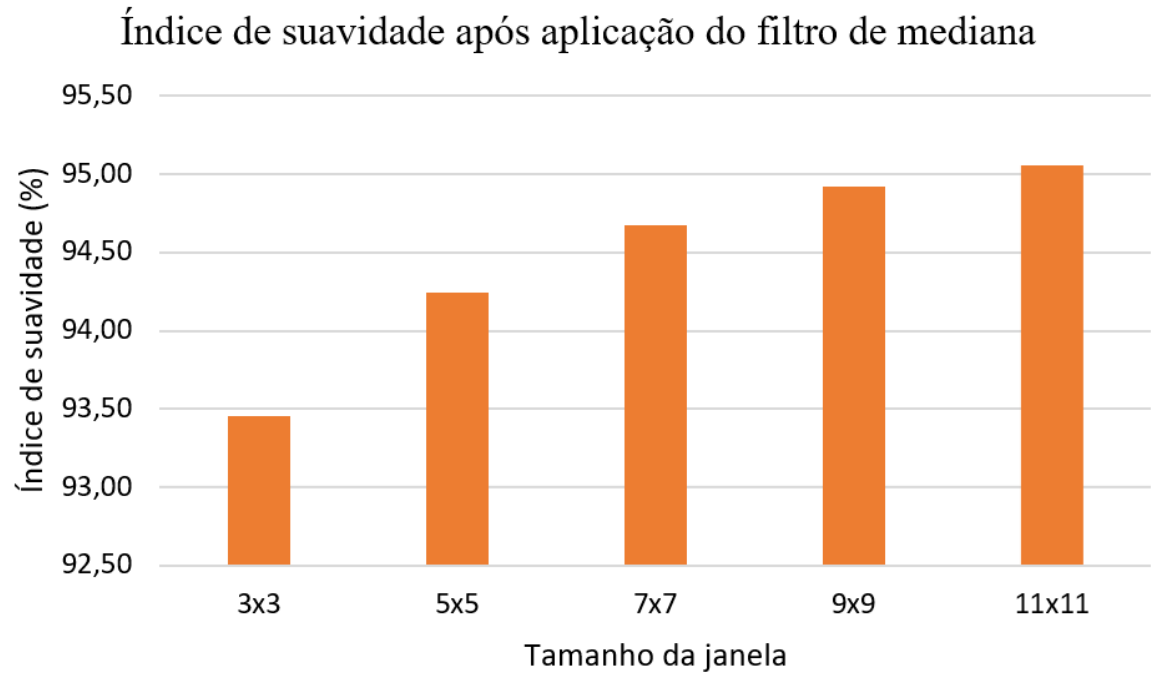

Figura 20 – Índice de suavidade do mapa da área B retificada usando o filtro da mediana.

Fonte: Autoria própria.

Também foram utilizados outros filtros como o da abertura, com janelas de 3x3, 5x5, 7x7, 9x9, e 11x11 pixeis, aplicados a área A (Figura 21) e a área B (Figura 22), foi possível verificar que, conforme o tamanho da janela e o número de iterações aumenta, a qualidade a ZM também aumenta, tanto na área A (Figura 23) quanto na área B (Figura 24), porém, caso o tamanho da janela ou o número de iterações seja muito alto, a ZM acaba sendo deformada. Isso ocorre devido a maneira de funcionamento do filtro de abertura que causa primeiramente a erosão de uma área muito grande da imagem fazendo com que áreas mais escuras da imagem sejam removidas, e então quando é aplicado a dilatação a região mais escura da imagem não exista para sofrer o efeito da dilatação.

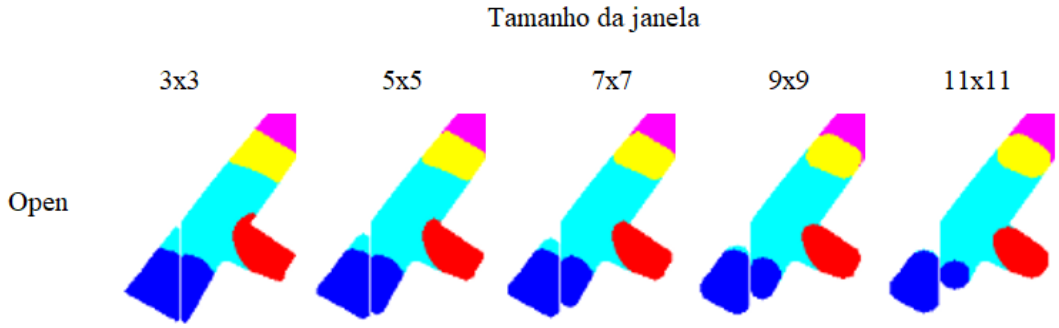

Figura 21 – Mapas da área A retificadas utilizando filtro de abertura e tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11.

Fonte: Autoria Própria.

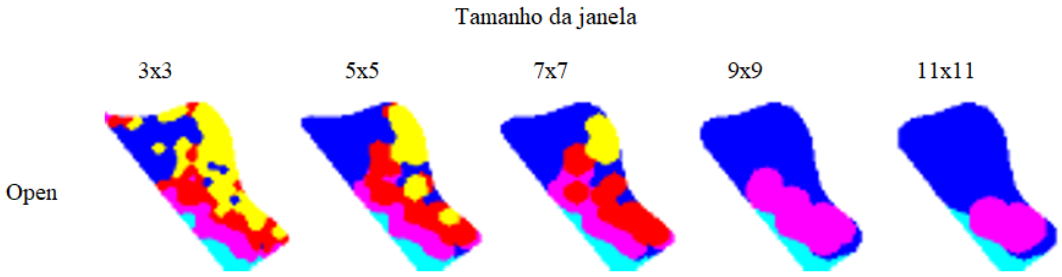

Figura 22 – Mapas da área B retificadas utilizando filtro de abertura e tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11.

Fonte: Autoria Própria.

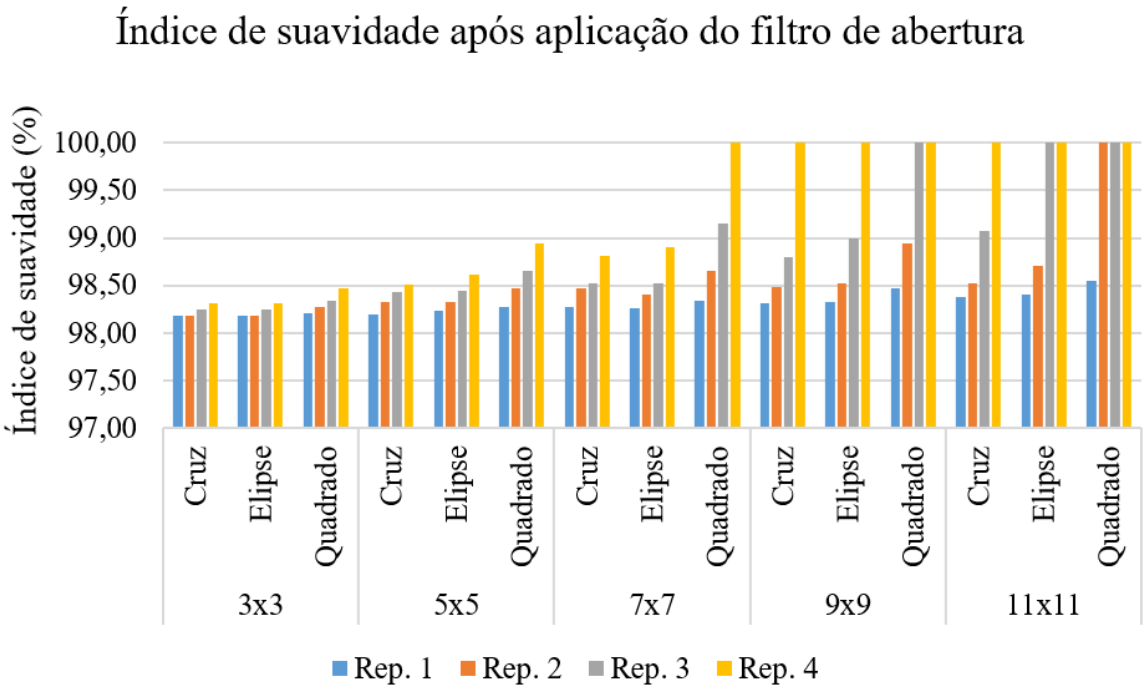

Figura 23 – Índice de suavidade do mapa da área A usando o filtro de abertura, de 1 a 4 iterações, tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

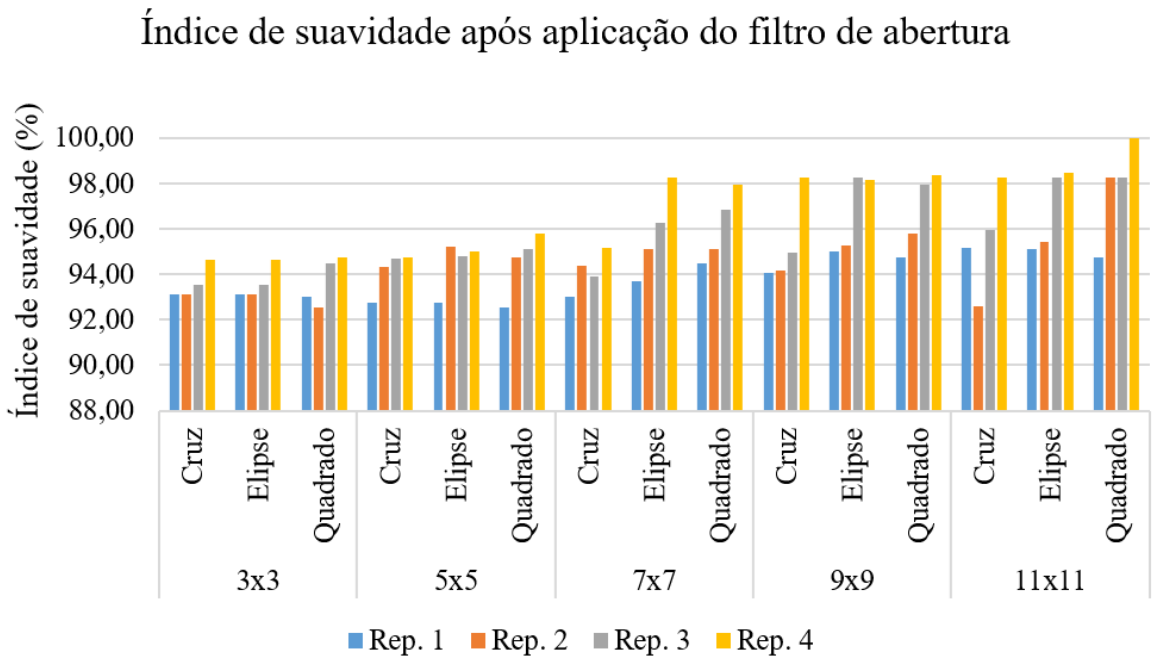

Figura 24 – Índice de suavidade do mapa da área B usando o filtro de abertura, de 1 a 4 iterações, tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

O filtro do fechamento também foi utilizado para a realização dos testes, com janelas de 3x3, 5x5, 7x7, 9x9, e 11x11 para a área A (Figura 25) e para a área B (Figura 26), foi possível verificar que o filtro do fechamento tem um resultado parecido com o filtro da abertura, onde conforme o tamanho da janela e o número de iterações, a qualidade a ZM aumenta tanto na área A (Figura 27) quanto na área B (Figura 28), porém, caso o tamanho da janela ou o número de iterações seja muito alto a ZM acaba sendo deformada. Isso ocorre devido ao funcionamento do filtro do fechamento que causa primeiramente a dilatação de uma área muito grande da imagem fazendo com que as áreas mais claras da imagem sejam removidas, e então quando é aplicada a erosão a região mais clara não existe para sofrer o efeito da dilatação.

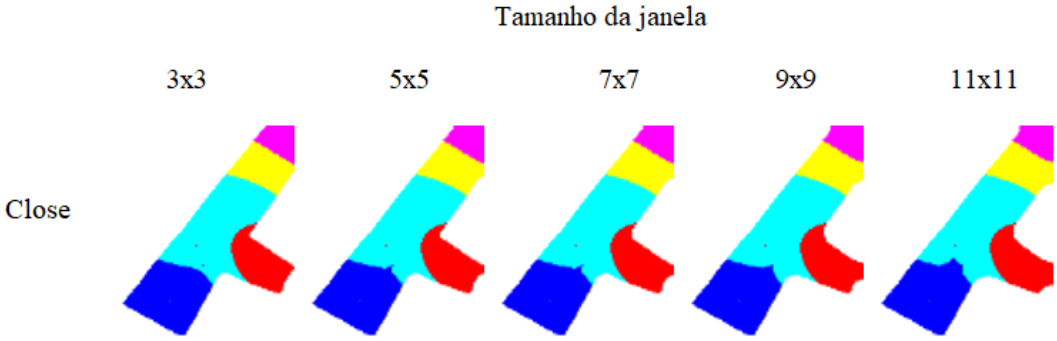

Figura 25 – Mapas da área A retificadas utilizando filtro de fechamento e tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11.

Fonte: Autoria Própria.

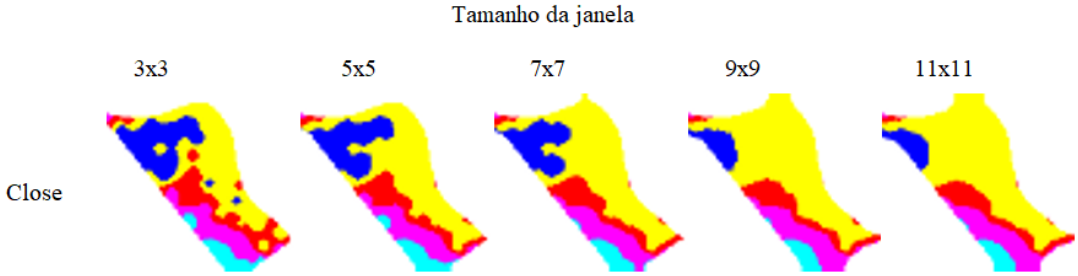

Figura 26 – Mapas da área B retificadas utilizando filtro de fechamento e tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11.

Fonte: Autoria Própria.

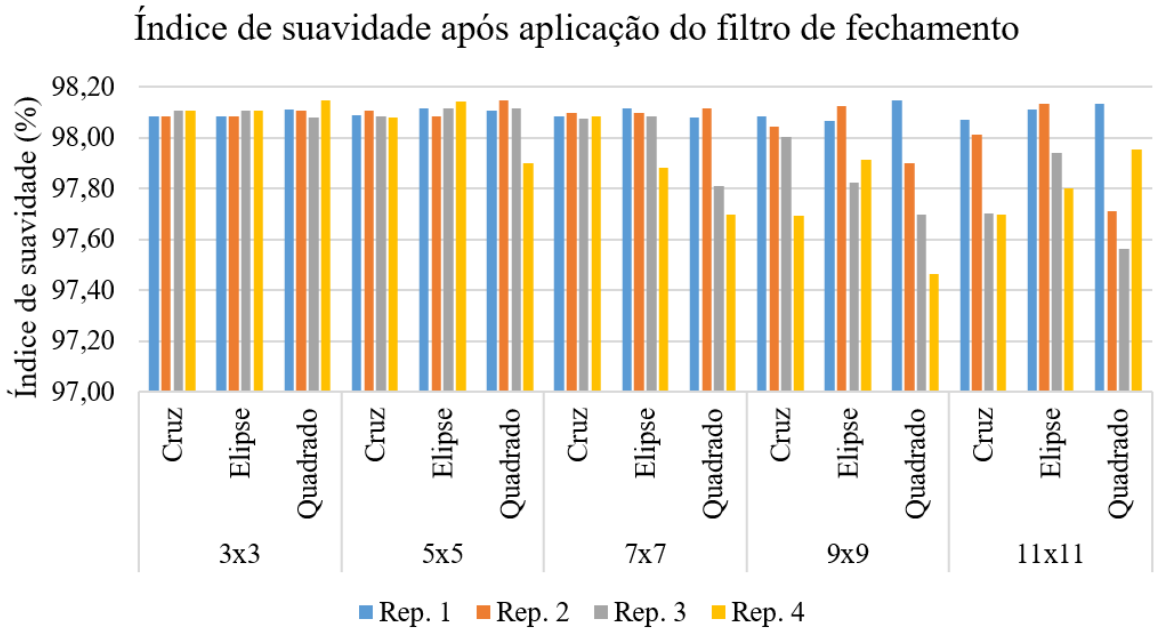

Figura 27 – Índice de suavidade do mapa da área A usando o filtro de fechamento, de 1 a 4 iterações, tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

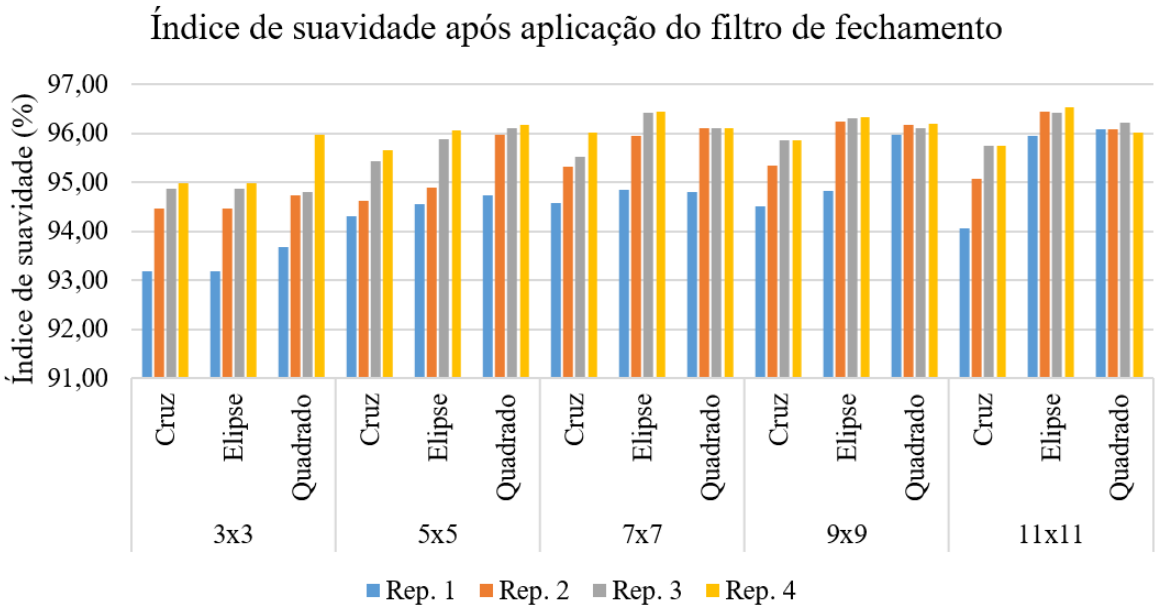

Figura 28 – Índice de suavidade do mapa da área B usando o filtro de fechamento, de 1 a 4 iterações, tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

Por fim, foram utilizados os filtros de abertura e fechamento em conjunto na área A

(Figura 29) e na área B (Figura 30) e novamente houve melhora conforme o tamanho da janela e o número de iterações aumenta. Tanto na área A (Figura 31) quanto na área B, (Figura 32) houve essa melhora, porém, caso o tamanho da janela ou o número de iterações seja muito alto a ZM também fica deformada.

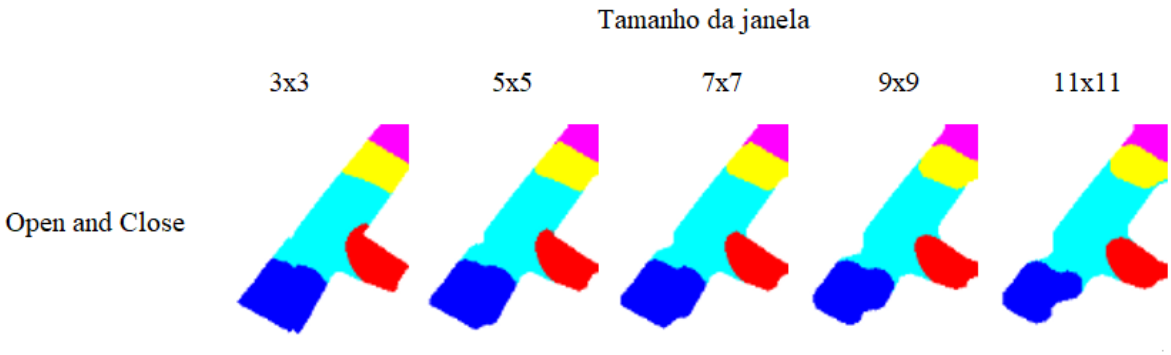

Figura 29 – Mapas da área A retificadas utilizando filtro de abertura e fechamento e tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11.

Fonte: Autoria Própria.

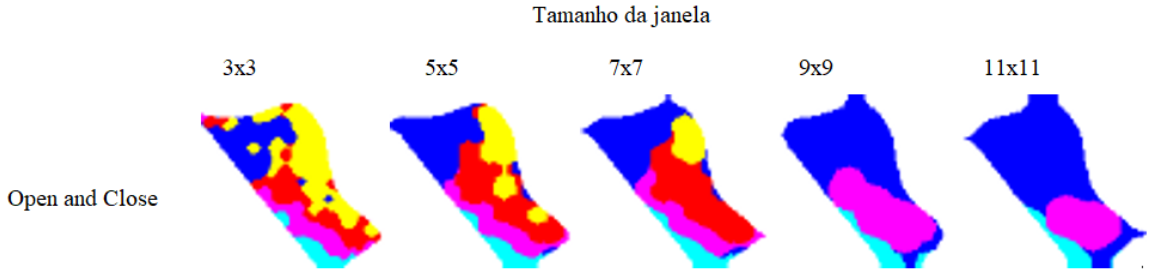

Figura 30 – Mapas da área B retificadas utilizando filtro de abertura e fechamento e tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11.

Fonte: Autoria Própria.

Os testes mostraram que a retificação das ZMs utilizando os filtros propostos melhoram a qualidade da ZM quando utilizado o SI para avaliar a ZM, fortalecendo as pesquisas realizadas por Betzek et al. (2017) e de Lowrance (2014).

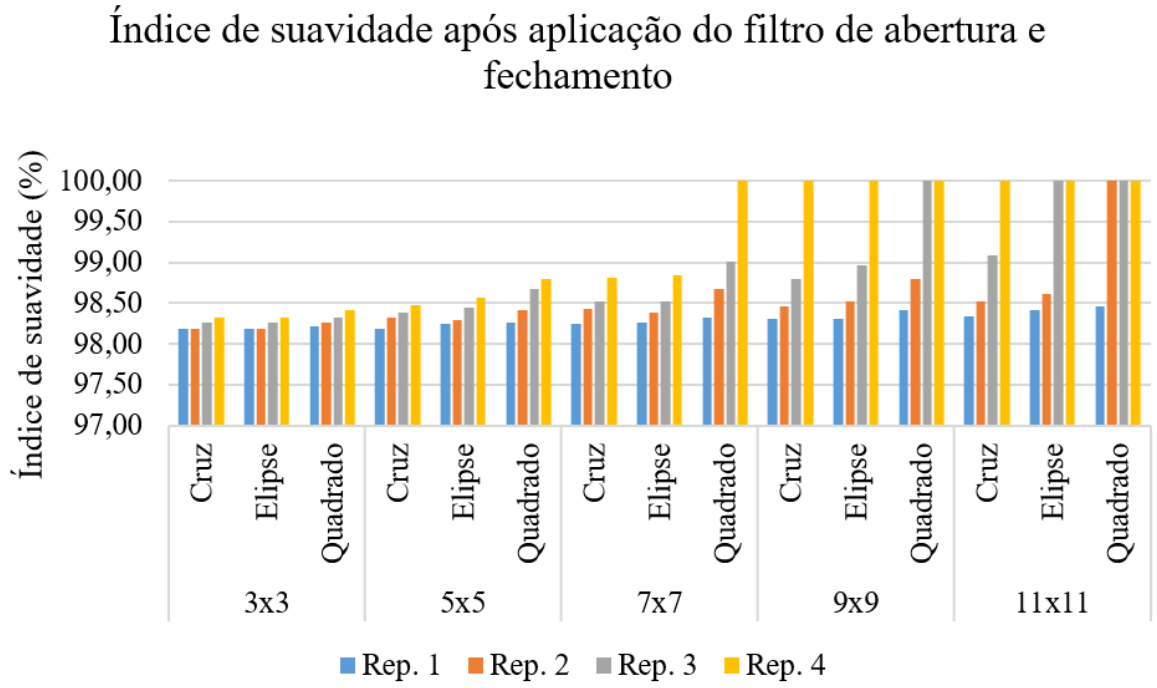

Figura 31 – Índice de suavidade do mapa da área A usando o filtro de abertura seguido pelo de fechamento, de 1 a 4 iterações, tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

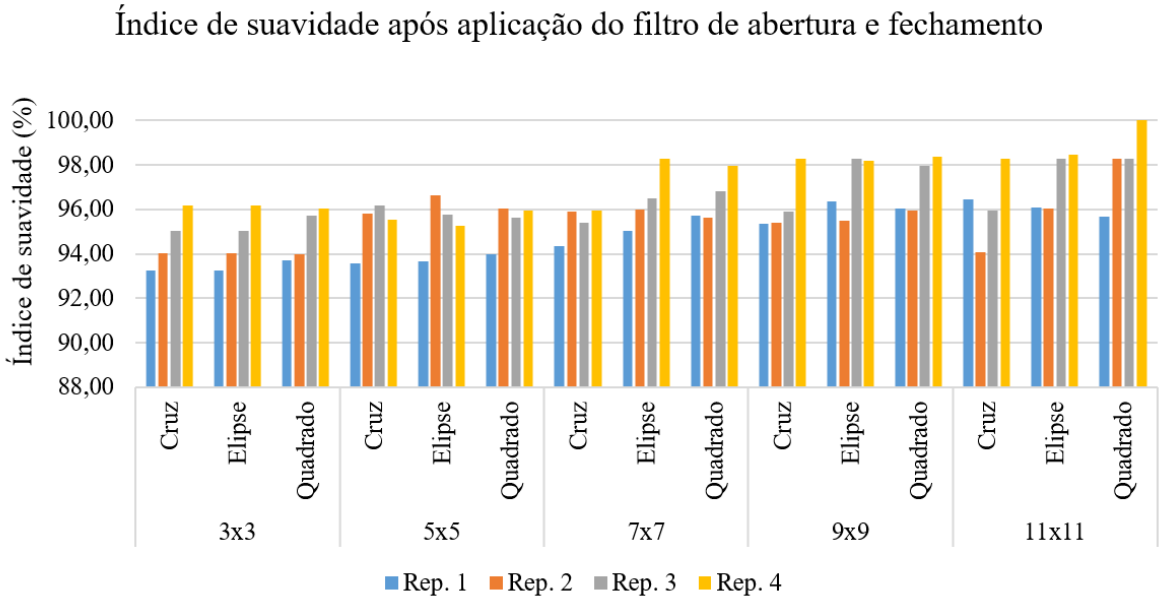

Figura 32 – Índice de suavidade do mapa da área B usando o filtro de abertura seguido pelo de fechamento, de 1 a 4 iterações, tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

5 CONCLUSÕES

Neste trabalho foi apresentado uma abordagem diferenciada para a realização da retificação de ZMs. Essa abordagem mostrou-se muito eficaz quanto ao tempo de execução quando comparado com a abordagem apresentada por Betzek et al. (2018).

Seu principal objetivo foi construir um microserviço escalável e com alta performance, utilizando tecnologias como OpenCV e o Netflix OSS, bem como realizar um estudo comparativo entre a rotina construída durante o desenvolvimento deste trabalho e a rotina de retificação existente desenvolvida por Betzek et al. (2018). Para que caso a rotina se mostrasse eficaz, seja possível a utilização deste microserviço em conjunto com outras aplicações que visam melhorar a utilização das terras na lavoura.

Com os resultados obtidos, foi possível verificar que a utilização do Python juntamente com o OpenCV para desenvolver a aplicação para realizar a retificação da ZM, torna o microserviço muito mais eficiente, a implementação dos filtros de mediana, erosão e dilatação foi realizada e foi possível verificar que se utilizado de maneira correta, estes filtros podem melhorar a qualidade da ZM.

Verificou-se também que dependendo do tamanho da janela e do número de iterações, a suavidade da ZM atinge um valor de 100%, porém a ZM perde as suas características tornando a ZM inutilizável.

Com a utilização do Netflix OSS e do *Http-proxy*, foi possível criar uma estrutura distribuída e altamente escalável, possibilitando assim, a utilização desse microserviço em aplicações comerciais.

6 TRABALHOS FUTUROS

Neste trabalho não foi avaliada a qualidade do mapa no aspecto estatístico quando sobreposto com uma variável alvo, como a produtividade da cultura. Como sugestão, recomenda-se fazer avaliações de estatística descritiva dentro dos grupos da ZM e verificar a redução da variância após a retificação.

Não foram realizadas também tratativas para quando a ZM perde suas características após a aplicação de algum filtro. Como sugestão, recomenda-se desenvolver uma rotina que aplique uma máscara sobre a ZM retificada a fim de evitar deformações na ZM.

Filtros como *2D Convolution*, *Gaussian Blur*, *Bilateral Filter* também podem ser utilizados e gerar bons resultados. Como sugestão, recomenda-se desenvolver uma rotina que utilize esses filtros para verificar de que maneira a qualidade da ZM é afetada.

ANEXO A – ROTINA DE RETIFICAÇÃO DE ZONAS DE MANEJO

```
import numpy as np
import sys
import cv2

formatedData = []
identifier = sys.argv[1]
method = sys.argv[2]
kernelSize = sys.argv[3]
kernelFormat = sys.argv[4]
iterations = sys.argv[5]
dataArg = sys.argv[6]

dataRows = dataArg.split('-')
for row in dataRows:
  dataColumns = row.split(',')
  columnInt = []
  for column in dataColumns:
    columnInt.append(int(column))
  formatedData.append(columnInt)

data2 = 50 * np.array(formatedData, dtype=np.uint8)

if kernelFormat == 'rect':
  kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (int(
      kernelSize), int(kernelSize)))
elif kernelFormat == 'ellipse':
  kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (int(
      kernelSize), int(kernelSize)))
elif kernelFormat == 'cross':
  kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (int(
      kernelSize), int(kernelSize)))
```

```
29
30 if method == 'median':
31   result = cv2.medianBlur(data2, int(kernelSize))
32 elif method == 'open':
33   resulta = cv2.erode(data2, kernel, iterations = int(iterations)
       )
34   result = cv2.dilate(resulta, kernel, iterations = int(
       iterations))
35 elif method == 'close':
36   resulta = cv2.dilate(data2, kernel, iterations = int(iterations
       ))
37   result = cv2.erode(resulta, kernel, iterations = int(iterations
       ))
38 elif method == 'openandclose':
39   resulta = cv2.erode(data2, kernel, iterations = int(iterations)
       )
40   resultb = cv2.dilate(resulta, kernel, iterations = int(
       iterations))
41   resultc = cv2.dilate(resultb, kernel, iterations = int(
       iterations))
42   result = cv2.erode(resultc, kernel, iterations = int(iterations
       ))
43
44 retorno = list(np.asarray(result))
45
46 retornoString = ''
47 for x in retorno:
48   xarray = list(np.asarray(x))
49   retornoString = retornoString + str(xarray)
50
51 sys.stdout.write(retornoString)
```

ANEXO B – API REST RESPONSÁVEL POR TRATAR OS DADOS

```
const { to, ReE, ReS, SendCSV, SendImage } = require('../services/util.service');
const { ParseObject, ParseString, ParseCSV } = require('../services/input.service');
const os = require('os');
var path = require('path');

const Matrix = require('node-matrix');
var exec = require('child_process').exec;

module.exports.retify = async (req, res) => {
  const distinct = (value, index, self) => {
    return self.indexOf(value) === index;
  };

  const matrixToString = mat => {
    let result = '';
    for (let i = 0; i < mat.numRows; i++) {
      let position = i.toString();
      const element = mat[position];
      result += element;
      result += i == mat.numRows - 1 ? '' : '-';
    }
    return result;
  };

  method = req.body.method;
  kernelSize = req.body.kernelSize;
  kernelFormat = req.body.kernelFormat;
  iterations = req.body.iterations;
```

```
29  data = req.body.dataset;
30  datasetFormat = req.body.datasetFormat;
31  outputFormat = req.body.outputFormat;
32  dataArray = [];
33
34  if (datasetFormat == 'object') {
35    dataArray = ParseObject(data);
36  } else if (datasetFormat == 'string') {
37    dataArray = ParseString(data);
38  } else {
39    dataArray = ParseCSV(data);
40  }
41
42  let latitudes = [];
43  let longitudes = [];
44  dataArray.forEach(element => {
45    latitudes.push(element.latitude);
46    longitudes.push(element.longitude);
47  });
48
49  let distinctLatitudes = latitudes.filter(distinct);
50  let distinctLongitudes = longitudes.filter(distinct);
51
52  distinctLatitudes = distinctLatitudes.sort((a, b) => a - b);
53  distinctLongitudes = distinctLongitudes.sort((a, b) => a - b);
54
55  const latitudesLength = distinctLatitudes.length;
56  const longitudesLength = distinctLongitudes.length;
57
58  console.log(latitudesLength);
59  console.log(longitudesLength);
60
61  let matrixC2 = Matrix({
62    rows: latitudesLength,
63    columns: longitudesLength
64  });
65
66  // Popula as Matrizes com os valores corretos nas posicoes
```

```
    corretas
for (let lat = 0; lat < latitudesLength; lat++) {
  for (let long = 0; long < longitudesLength; long++) {
    for (let index = 0; index < dataArray.length; index++) {
      const element = dataArray[index];
      if (element.latitude == distinctLatitudes[lat] && element
          .longitude == distinctLongitudes[long]) {
        matrixC2[lat][long] = parseInt(element.c2);
      }
    }
  }
}

// processa os dados que retornam do script de retificacao
const processReturnDataFromScript = values => {
  let matrixRetorno = Matrix({
    rows: latitudesLength,
    columns: longitudesLength
  });
  rows = values.split('][');
  for (let i = 0; i < rows.length; i++) {
    rows[i] = rows[i].replace('[', '');
    rows[i] = rows[i].replace(']', '');
    cols = rows[i].split(',');
    for (let j = 0; j < cols.length; j++) {
      matrixRetorno[i][j] = parseInt(cols[j]);
    }
  }
  return matrixRetorno;
};

const execCommand = (id, method, kernelSize, kernelFormat,
    iterations, stringData) => {
  return new Promise(resolve => {
    const cmd = 'sudo python3 retify.py ' + id + ' ' + method.
        toString() + ' ' + kernelSize.toString() + ' ' +
        kernelFormat.toString() + ' ' + iterations.toString() + '
         ' + stringData;
```

```
99
100       exec( cmd, { cwd: __dirname },
101         (err, stdout, stderr) => {
102           resolve(stdout);
103         }
104       );
105     });
106   };
107
108   const [out2, out3, out4, out5] = await Promise.all([
109     execCommand('c2', method, kernelSize, kernelFormat,
          iterations, matrixToString(matrixC2))
110   ]);
111
112   matrixRetornoC2 = processReturnDataFromScript(out2);
113
114   // popula o objeto com as informa  es das ZMs retificadas
115   for (let lat = 0; lat < latitudesLength; lat++) {
116     for (let long = 0; long < longitudesLength; long++) {
117       for (let index = 0; index < dataArray.length; index++) {
118         const element = dataArray[index];
119         if (element.latitude == distinctLatitudes[lat] && element
              .longitude == distinctLongitudes[long]) {
120           dataArray[index].c2 = Math.round(matrixRetornoC2[lat][
                long] / 50);
121         }
122       }
123     }
124   }
125
126   if (outputFormat == 'image') {
127     return SendImage(res, path.join(__dirname, '../public/', '
          retificadac2.png'));
128   } else if (outputFormat == 'object') {
129     return ReS(res, dataArray, 200);
130   } else {
131     return SendCSV(res, dataArray);
132   }
```

```
133 };
```

ANEXO C – IMAGENS DOS RESULTADOS OBTIDOS

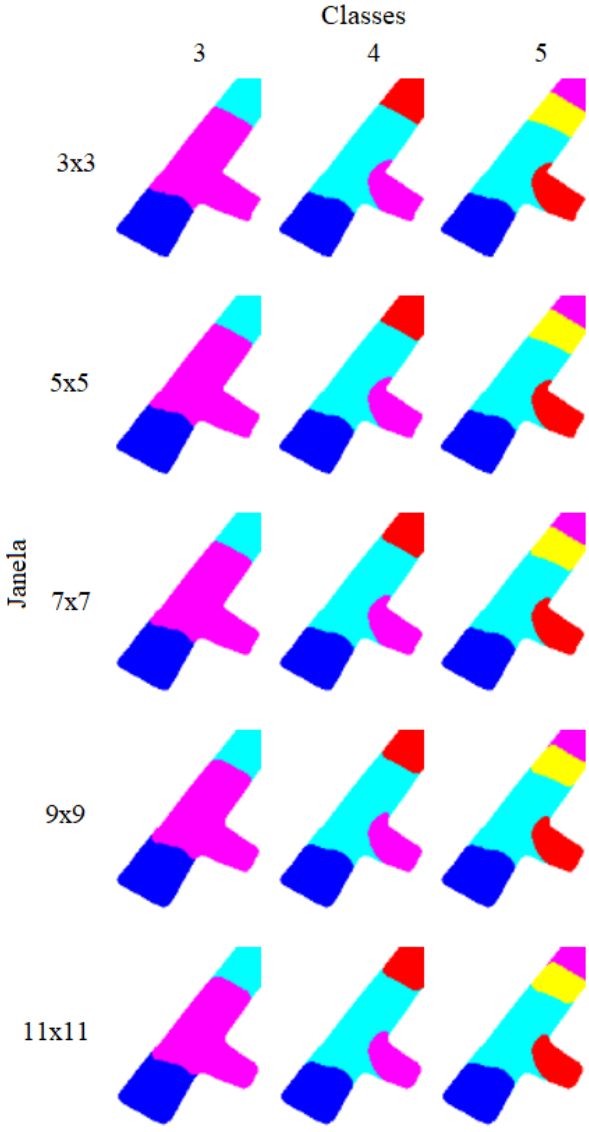

Figura 33 – Mapas da área A retificadas utilizando filtro mediana e tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11 nas divisões em 3, 4 e 5 classes.

Fonte: Autoria própria.

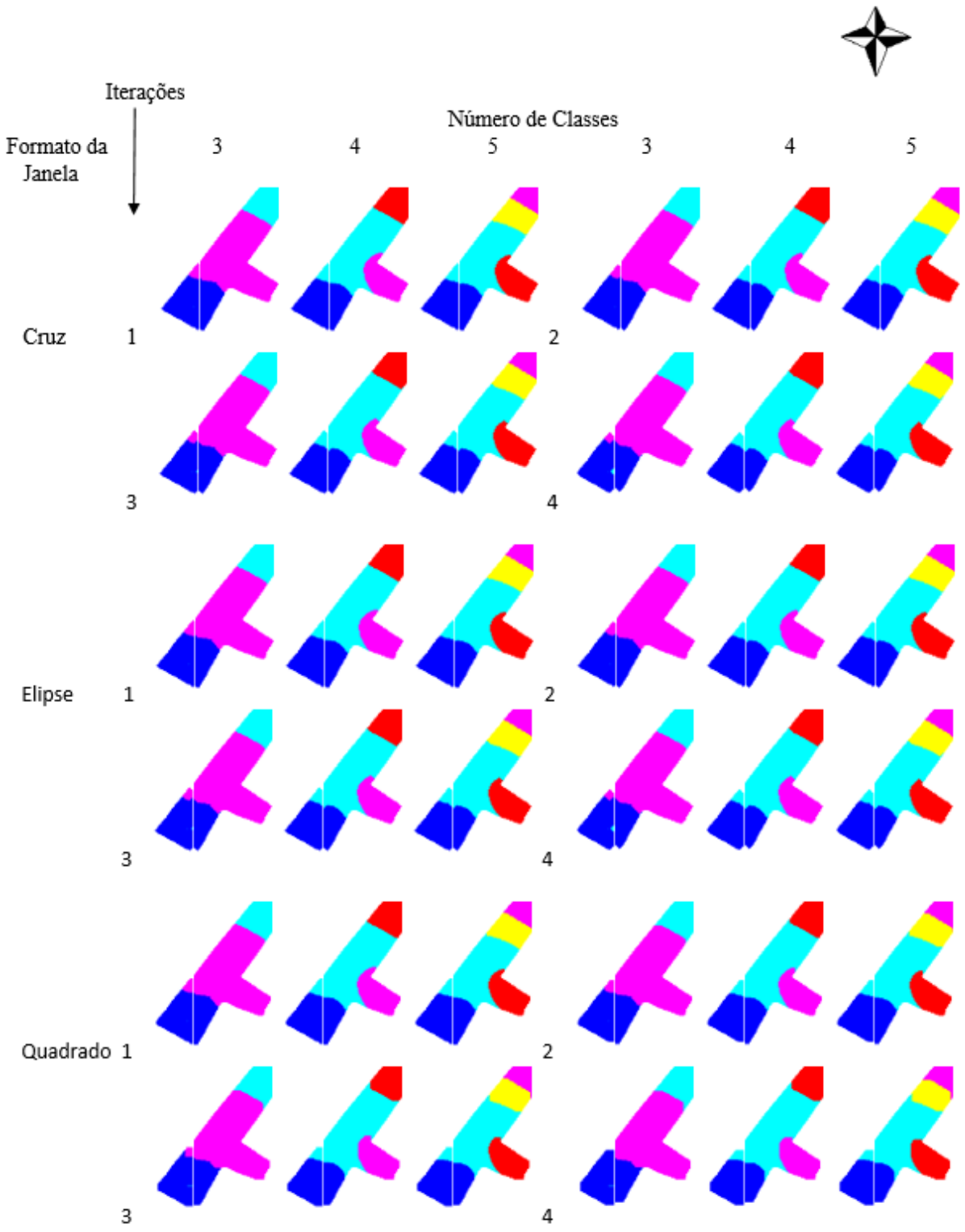

Figura 34 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de abertura, de 1 a 4 iterações, tamanho de janela 3x3, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

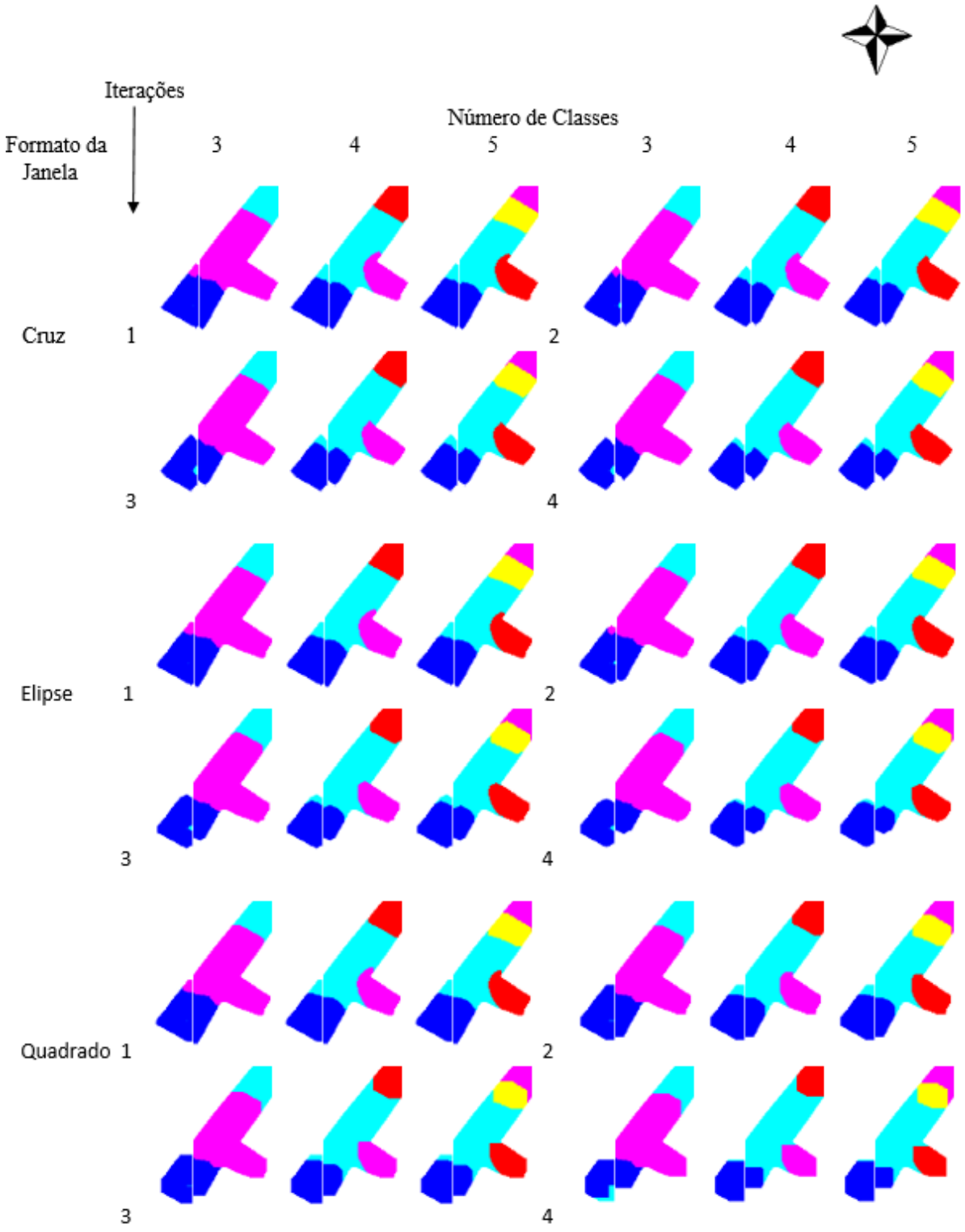

Figura 35 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de abertura, de 1 a 4 iterações, tamanho de janela 5x5, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

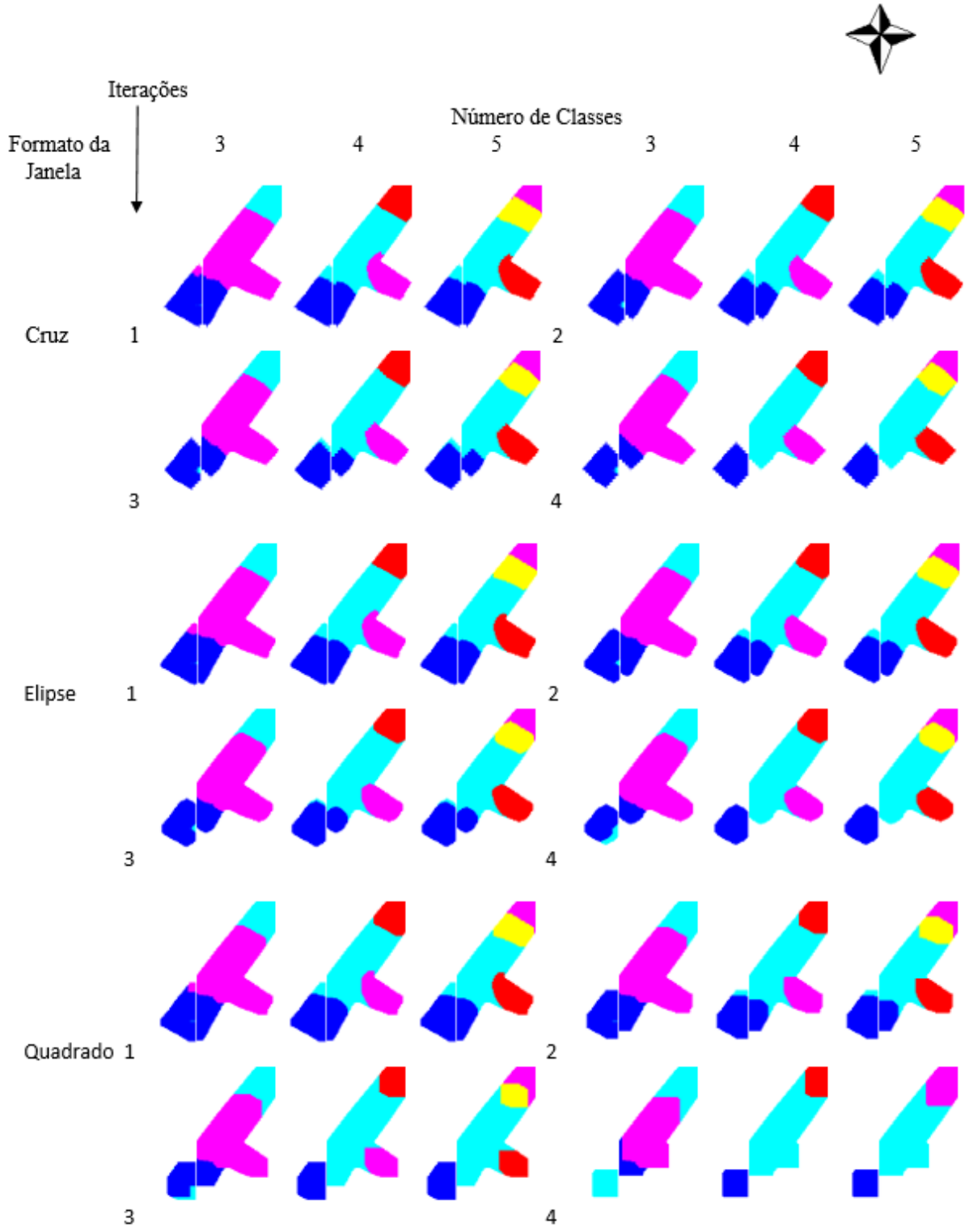

Figura 36 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de abertura, de 1 a 4 iterações, tamanho de janela 7x7, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

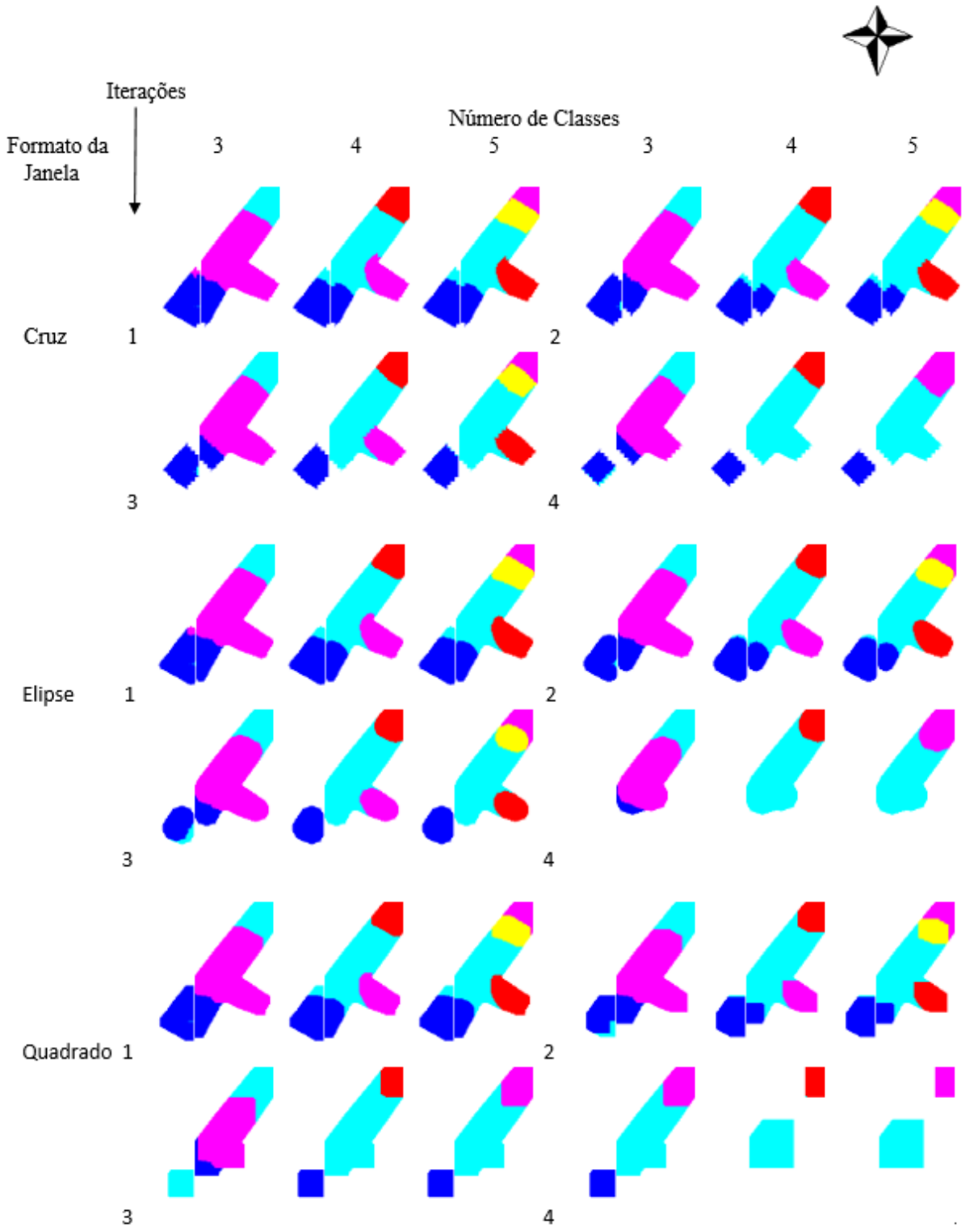

Figura 37 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de abertura, de 1 a 4 iterações, tamanho de janela 9x9, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

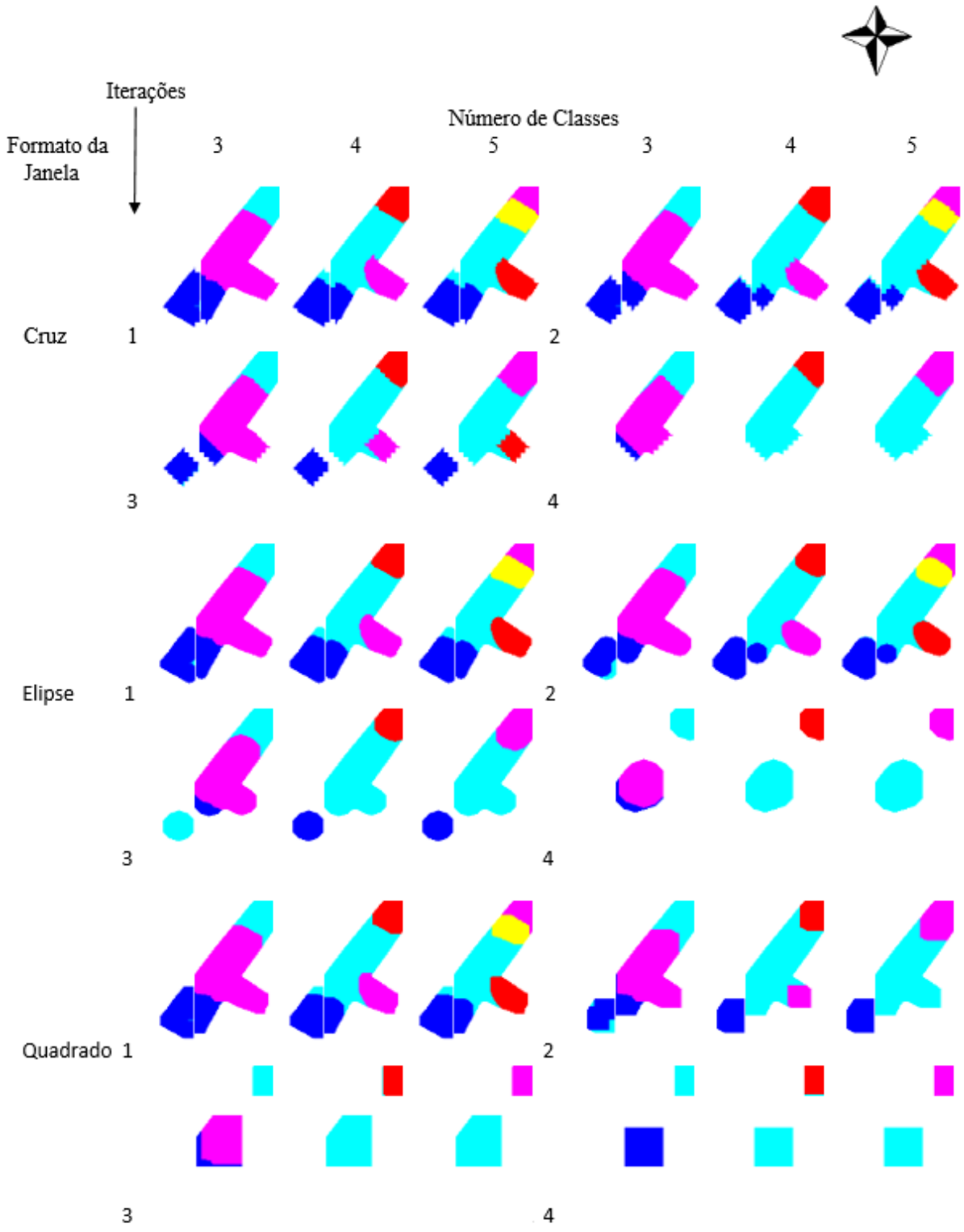

Figura 38 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de abertura, de 1 a 4 iterações, tamanho de janela 11x11, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

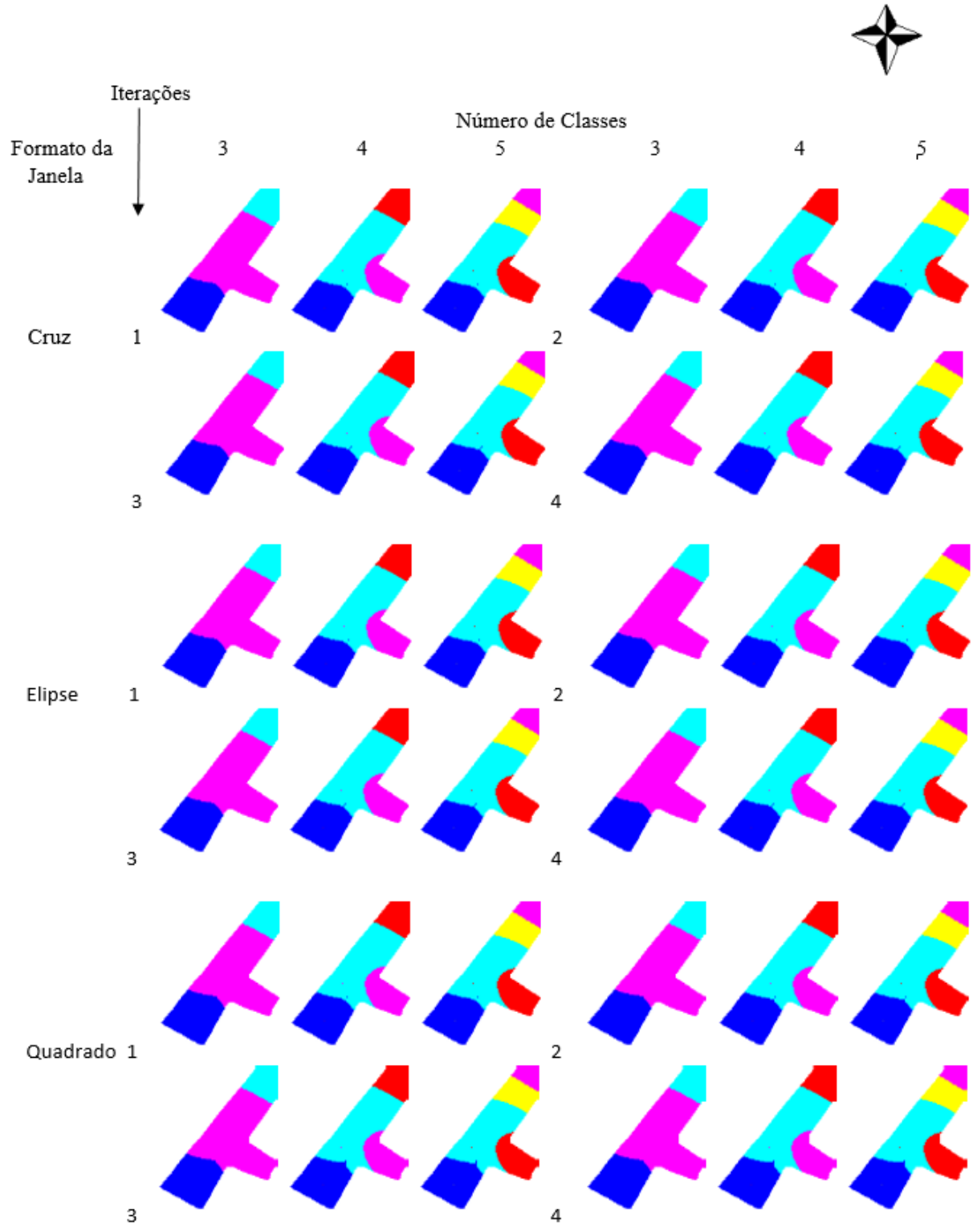

Figura 39 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de fechamento, de 1 a 4 iterações, tamanho de janela 3x3, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

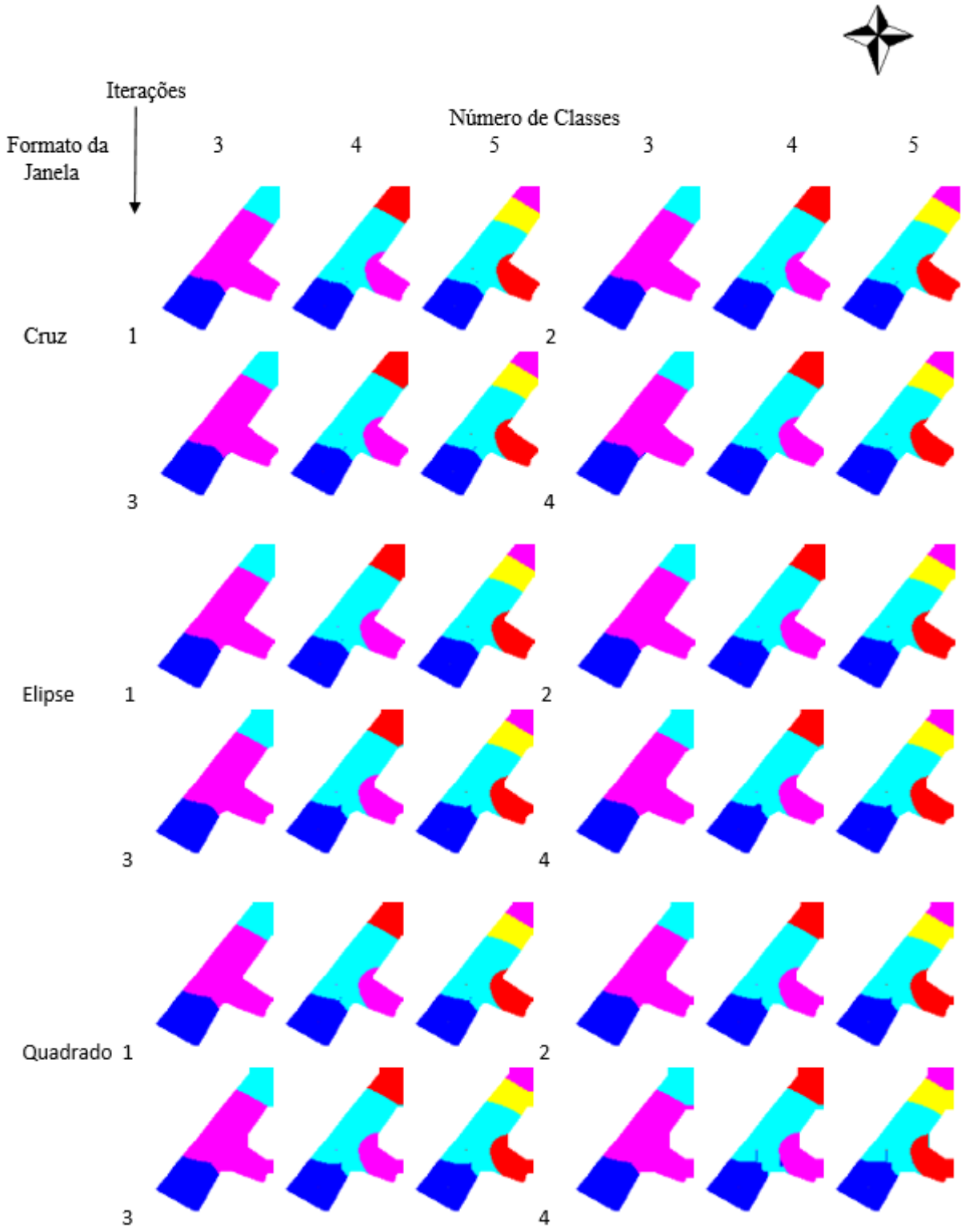

Figura 40 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de fechamento, de 1 a 4 iterações, tamanho de janela 5x5, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

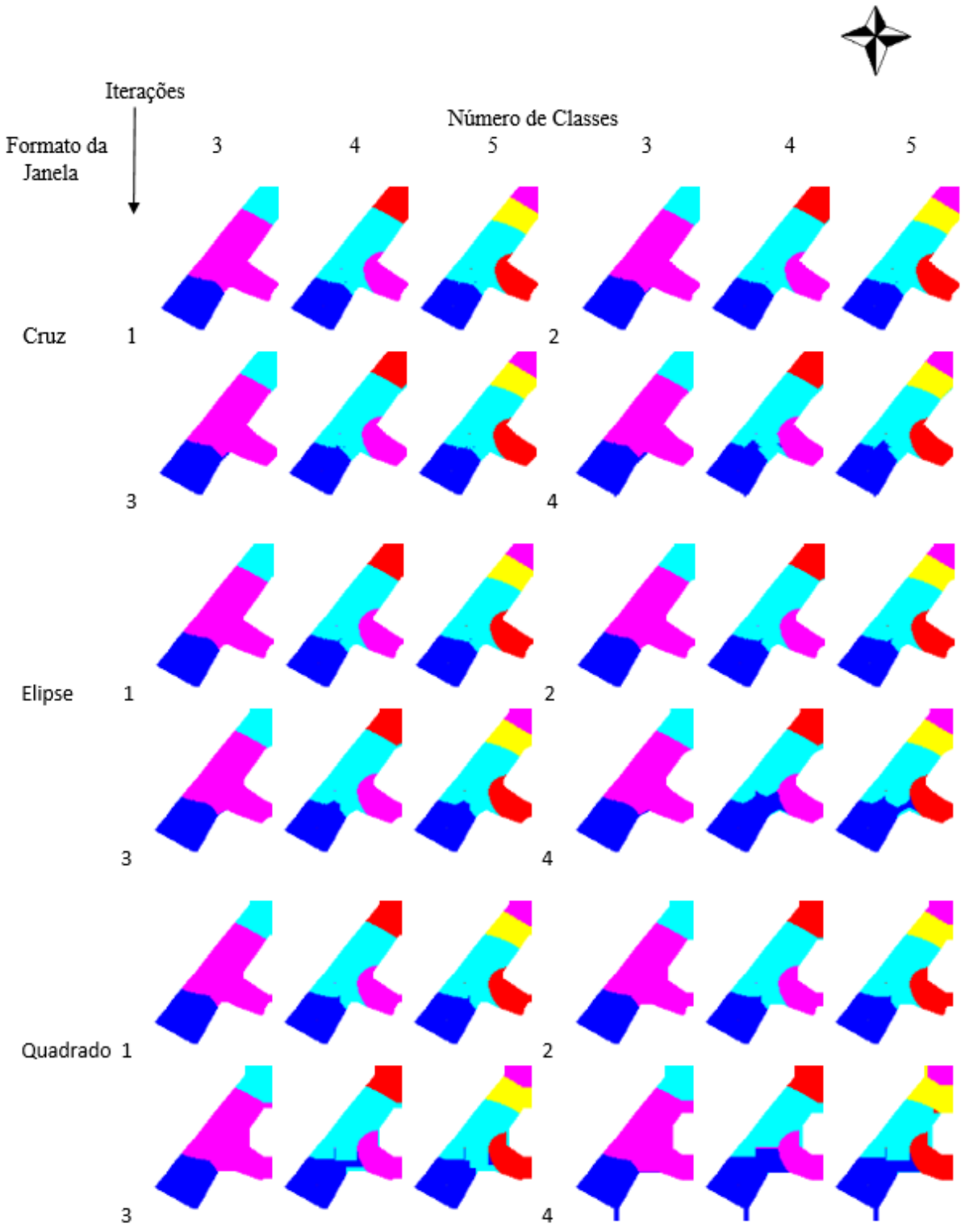

Figura 41 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de fechamento, de 1 a 4 iterações, tamanho de janela 7x7, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

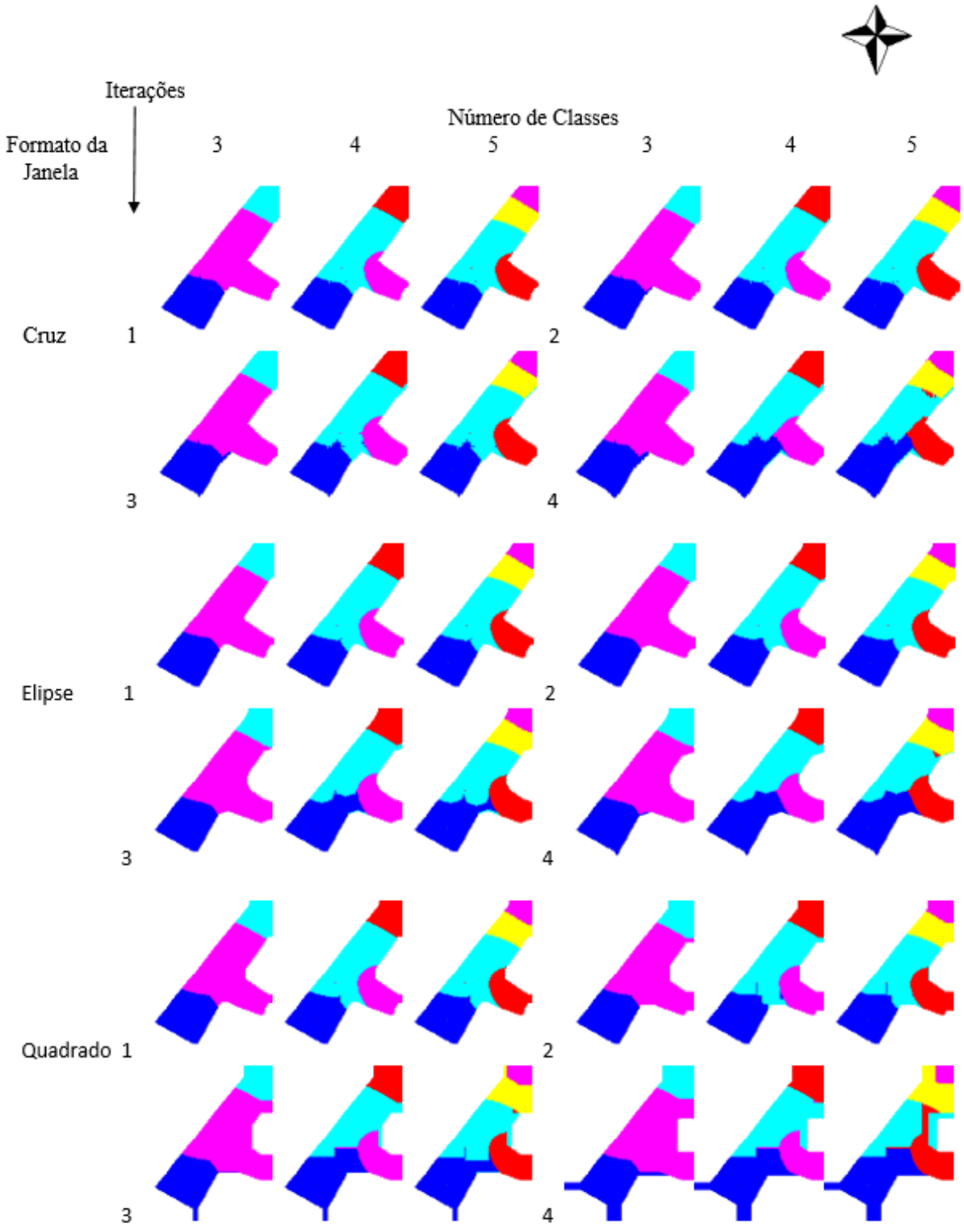

Figura 42 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de fechamento, de 1 a 4 iterações, tamanho de janela 9x9, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

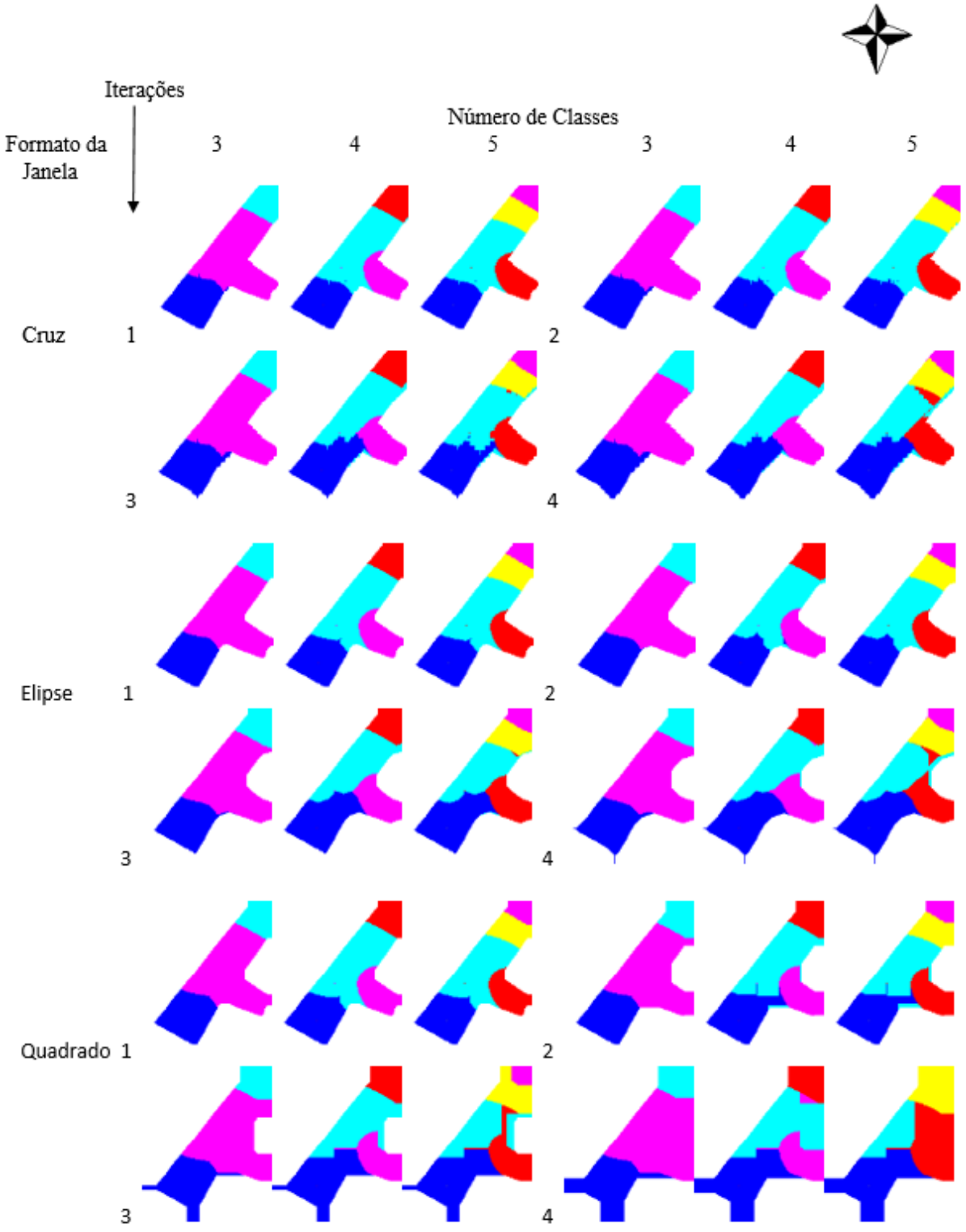

Figura 43 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de fechamento, de 1 a 4 iterações, tamanho de janela 11x11, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

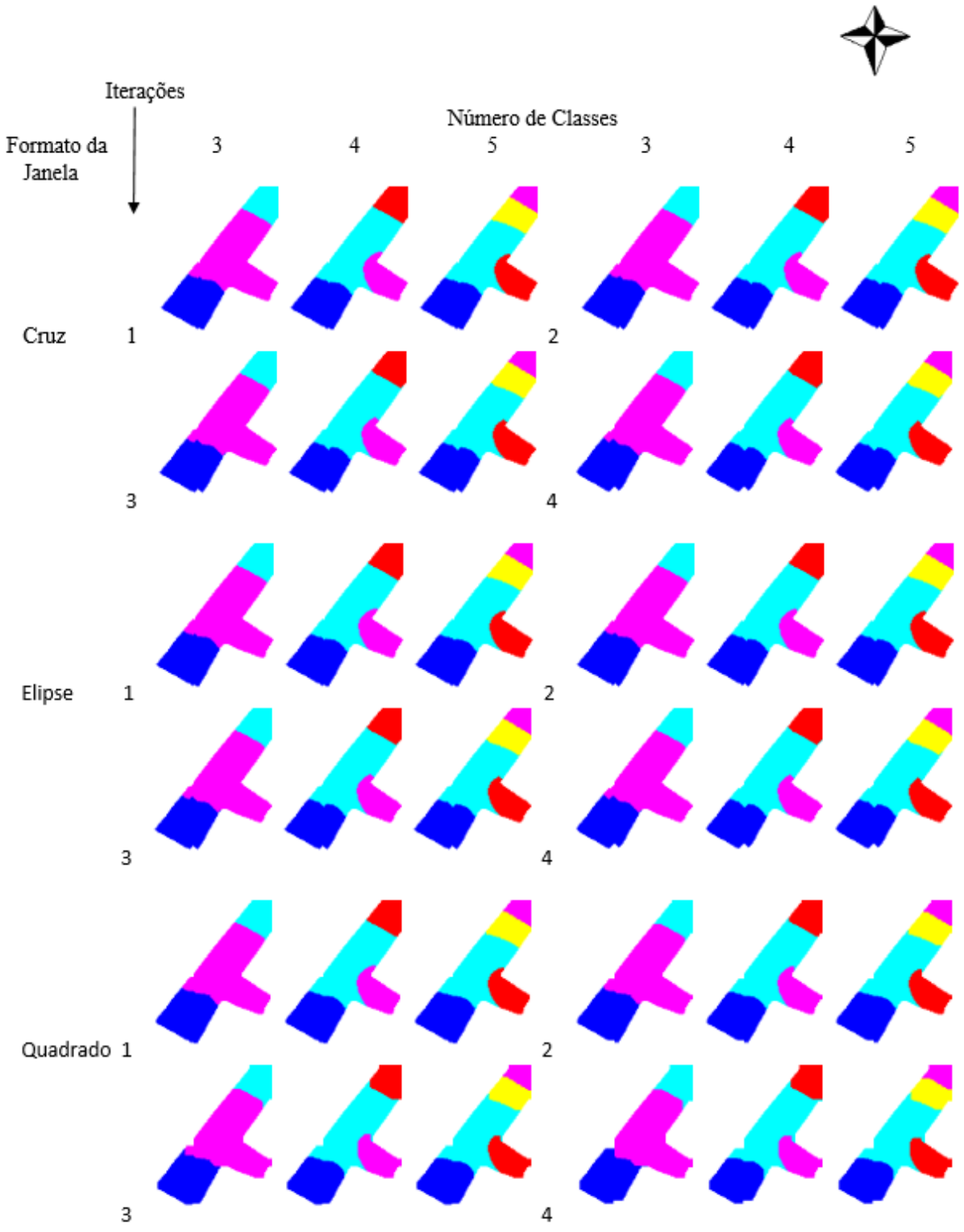

Figura 44 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de abertura seguido pelo fechamento, de 1 a 4 iterações, tamanho de janela 3x3, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

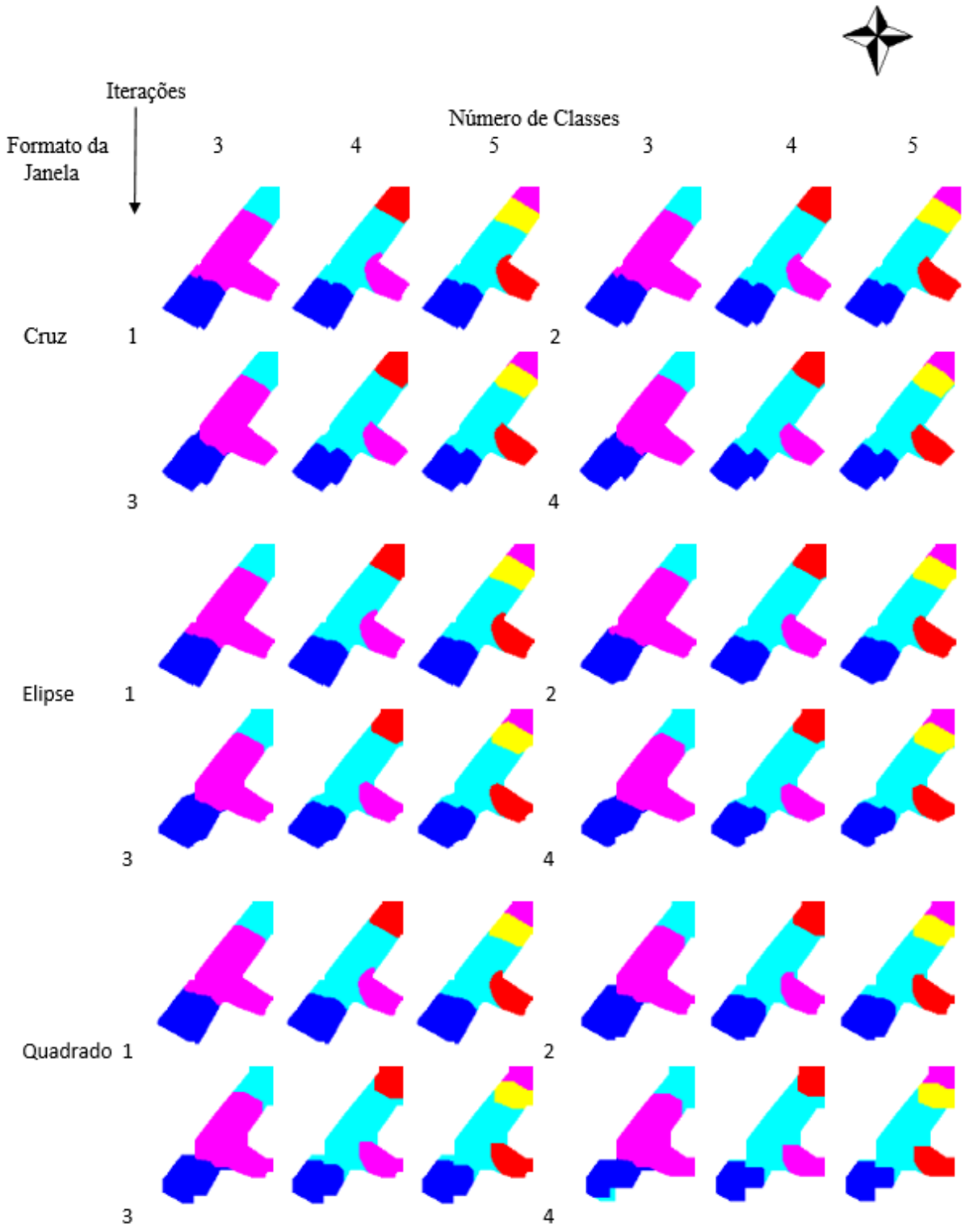

Figura 45 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de abertura seguido pelo fechamento, de 1 a 4 iterações, tamanho de janela 5x5, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

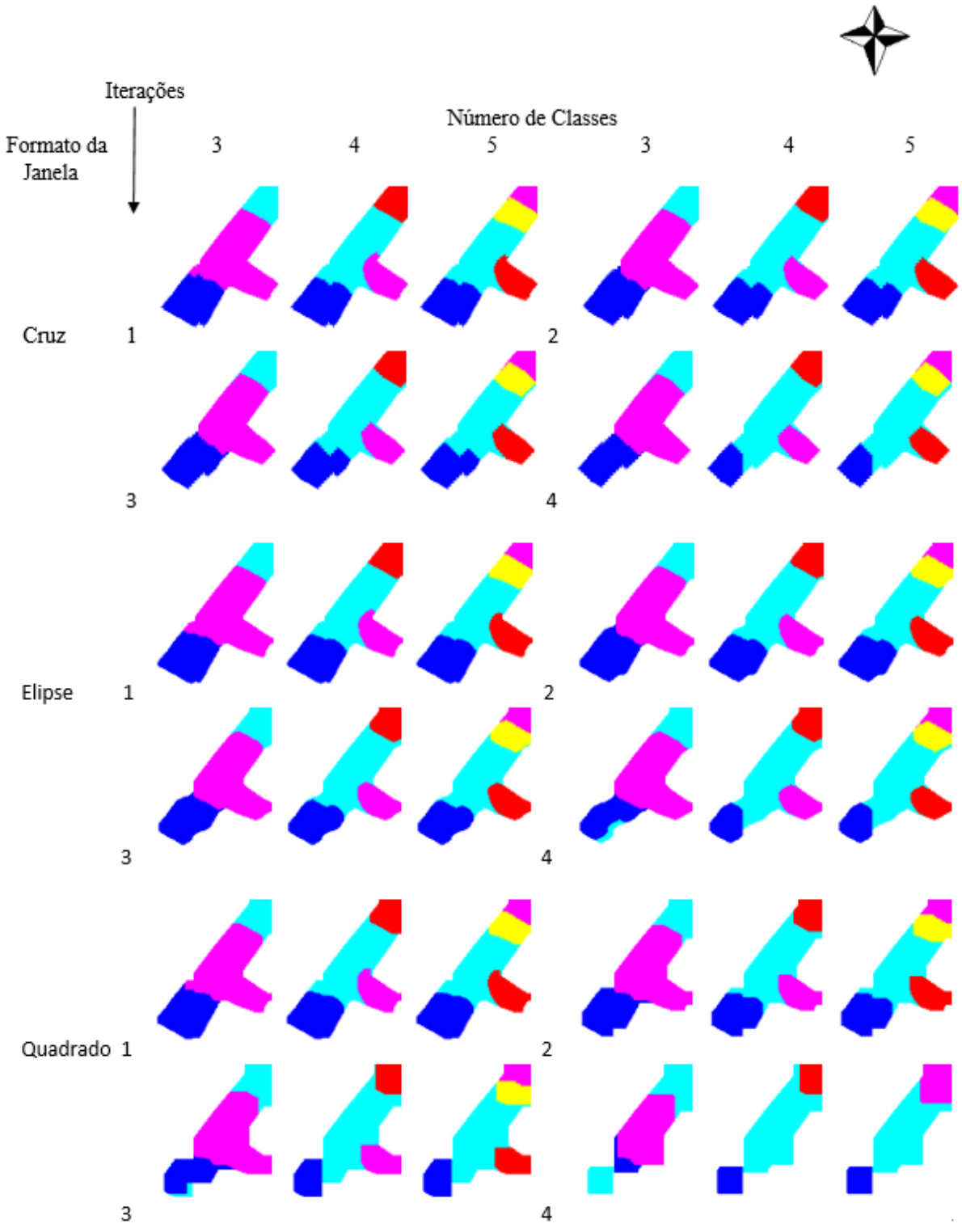

Figura 46 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de abertura seguido pelo fechamento, de 1 a 4 iterações, tamanho de janela 7x7, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

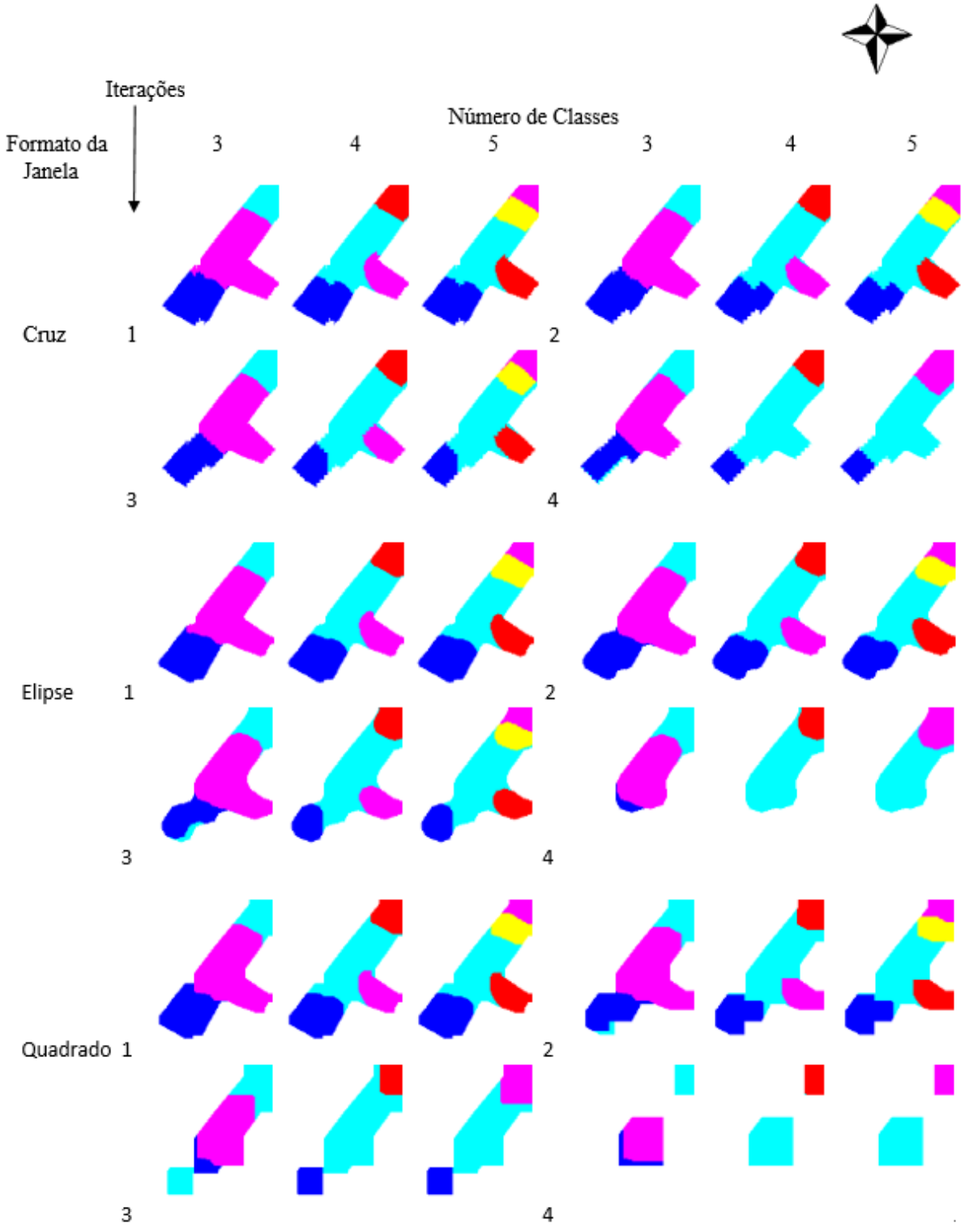

Figura 47 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de abertura seguido pelo fechamento, de 1 a 4 iterações, tamanho de janela 9x9, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

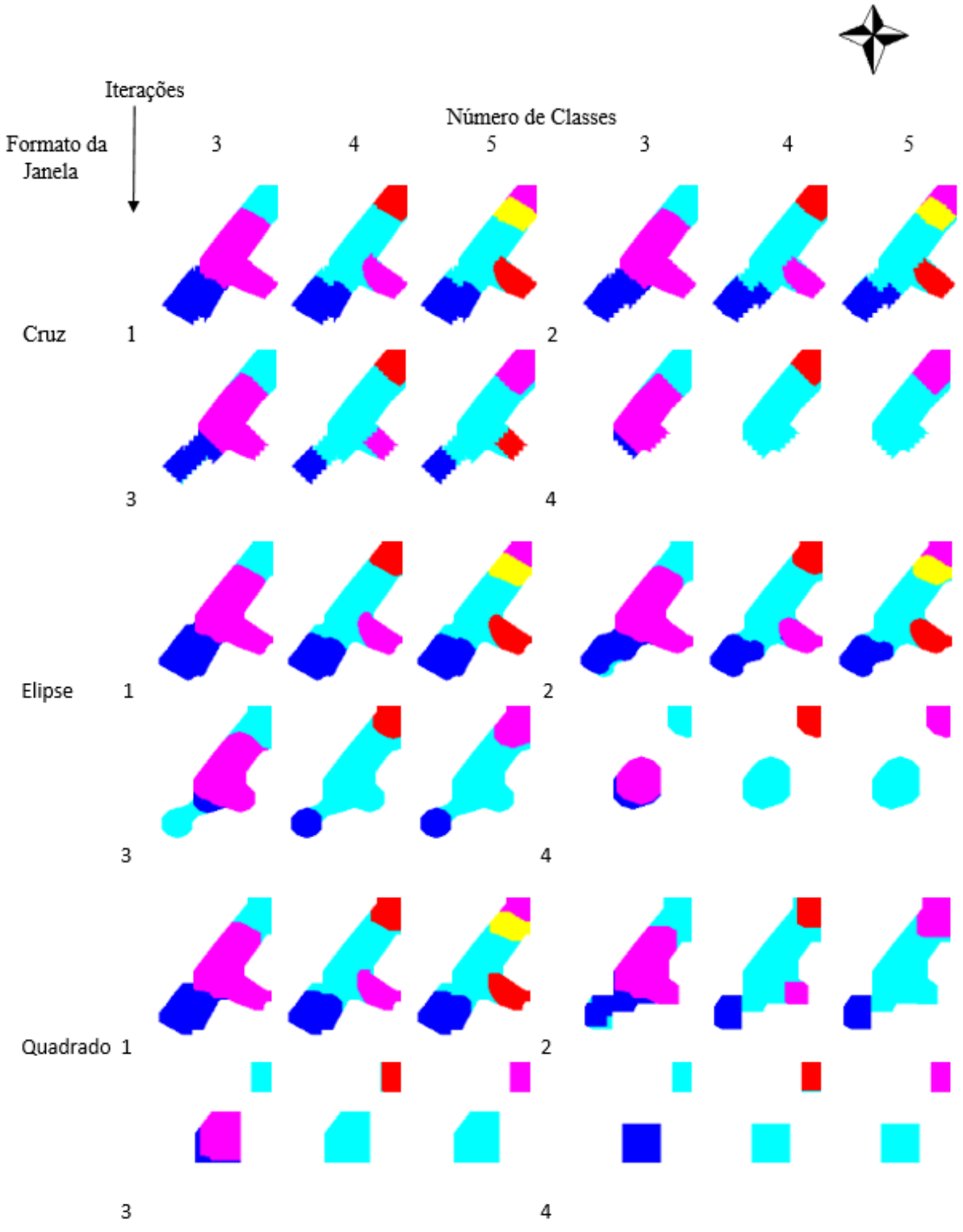

Figura 48 – Mapas da área A, divididos de 3 a 5 classes, retificados utilizando filtro de abertura seguido pelo fechamento, de 1 a 4 iterações, tamanho de janela 11x11, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

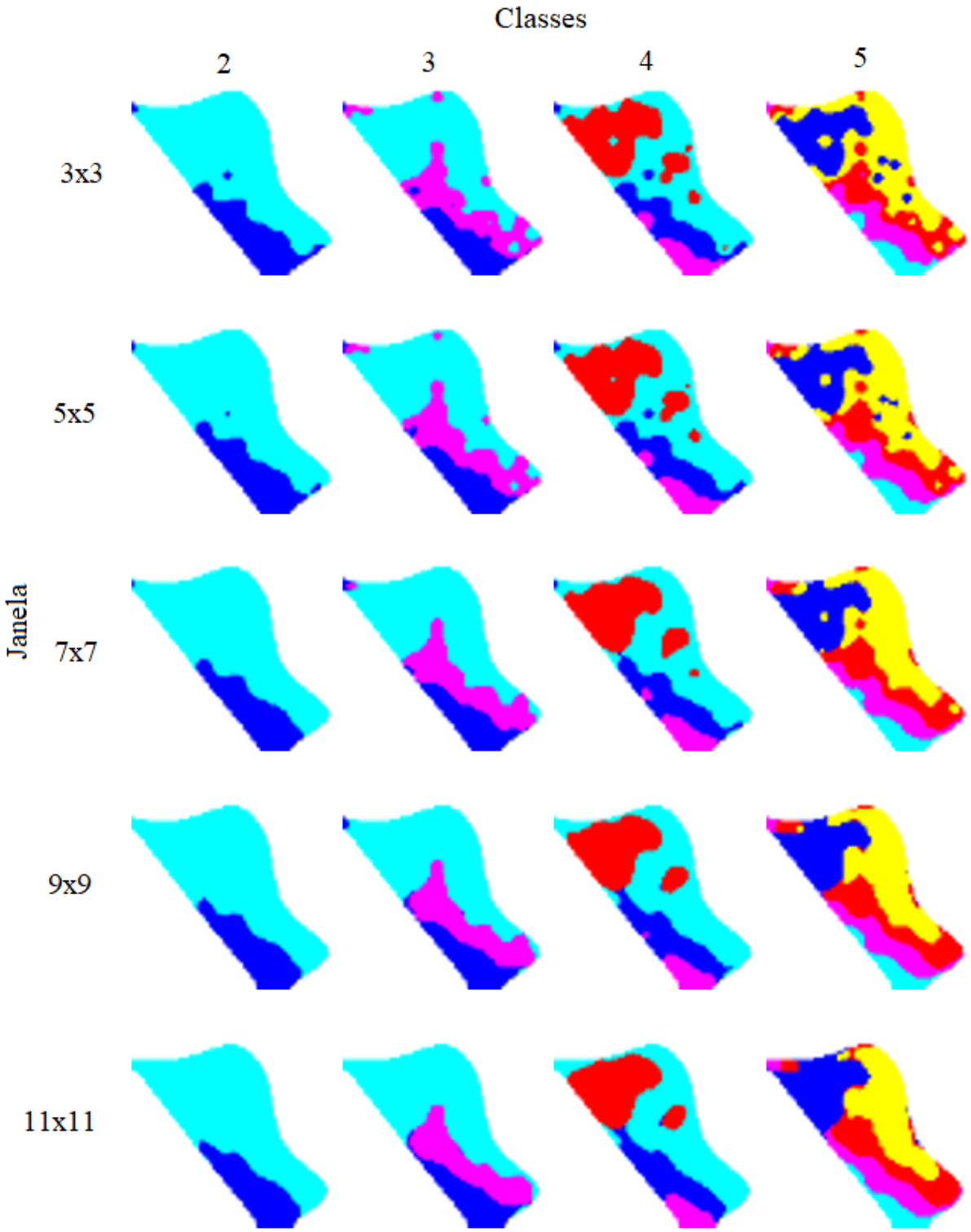

Figura 49 – Mapas da área B retificadas utilizando filtro mediana e tamanhos de janela 3x3, 5x5, 7x7, 9x9, 11x11 nas divisões em 2, 3, 4 e 5 classes.

Fonte: Autoria própria.

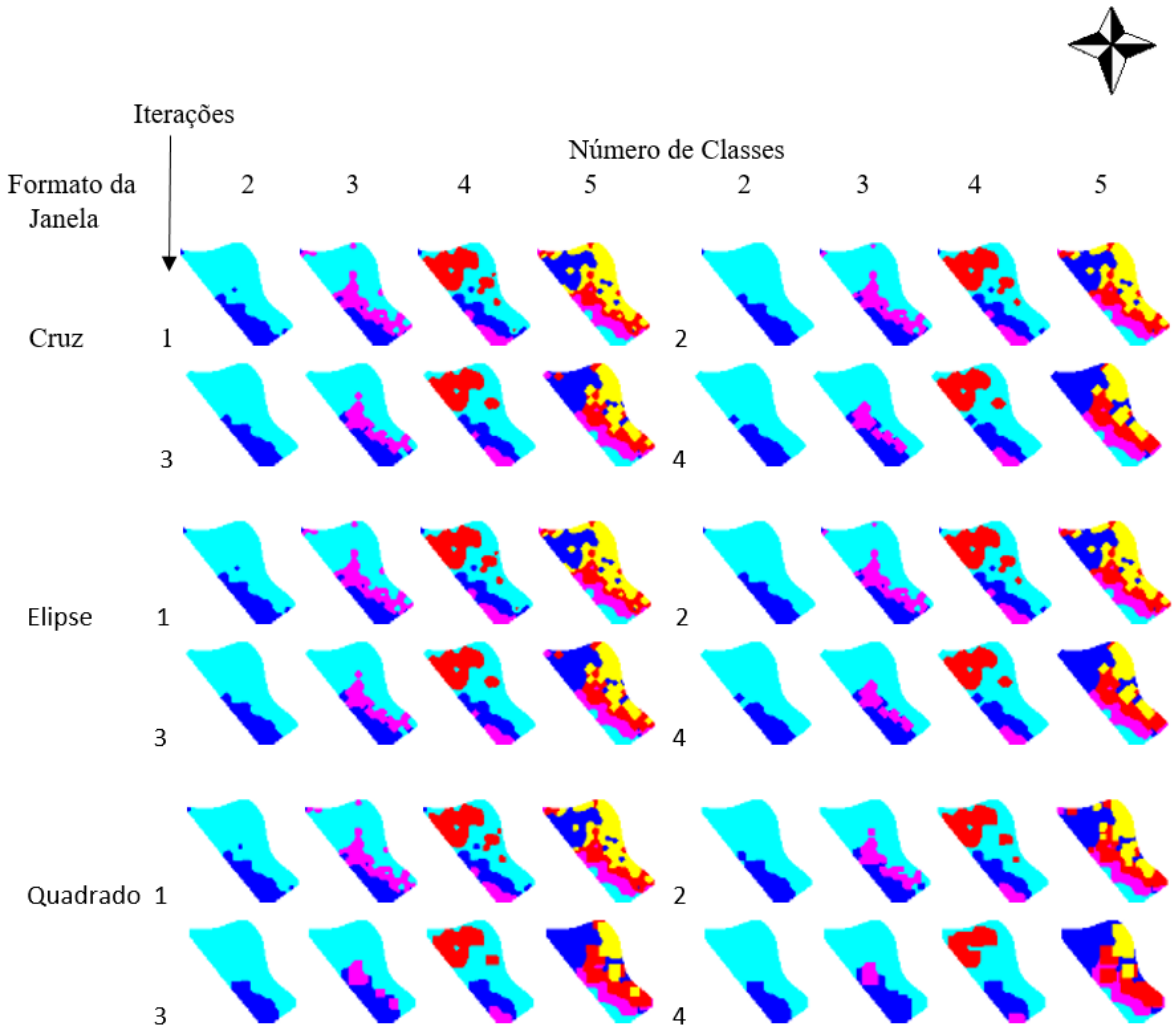

Figura 50 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de abertura, de 1 a 4 iterações, tamanho de janela 3x3, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

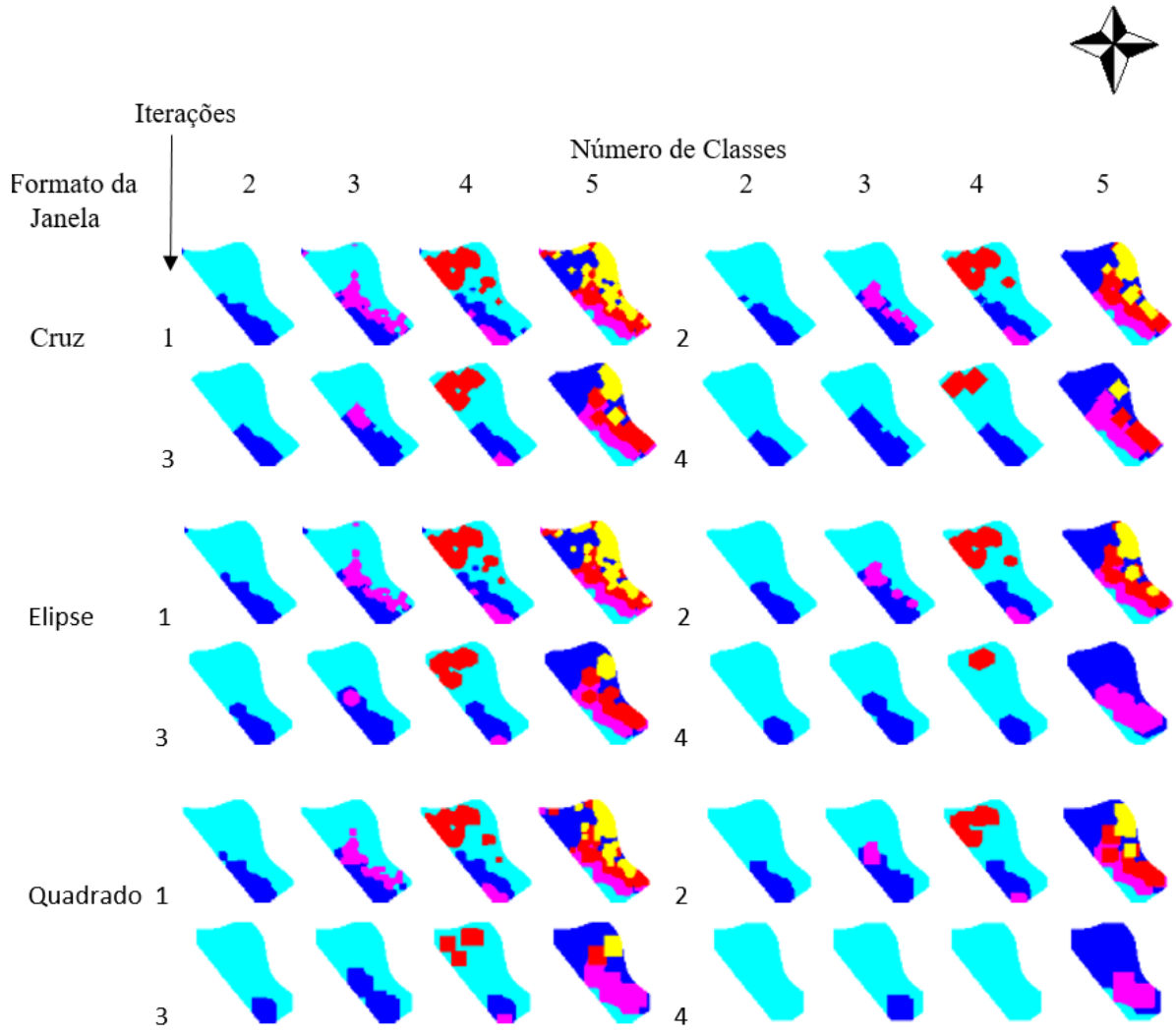

Figura 51 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de abertura, de 1 a 4 iterações, tamanho de janela 5x5, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

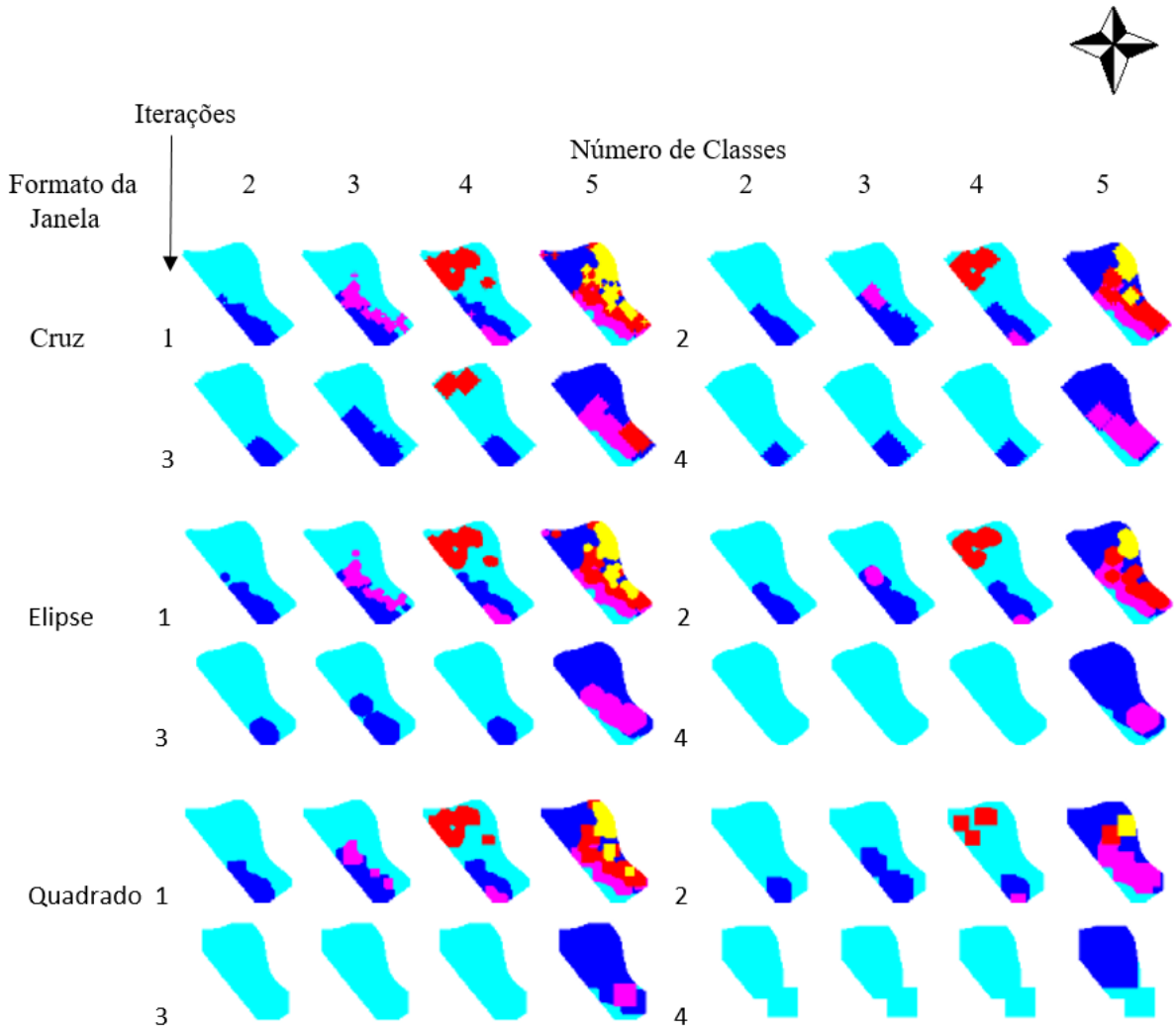

Figura 52 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de abertura, de 1 a 4 iterações, tamanho de janela 7x7, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

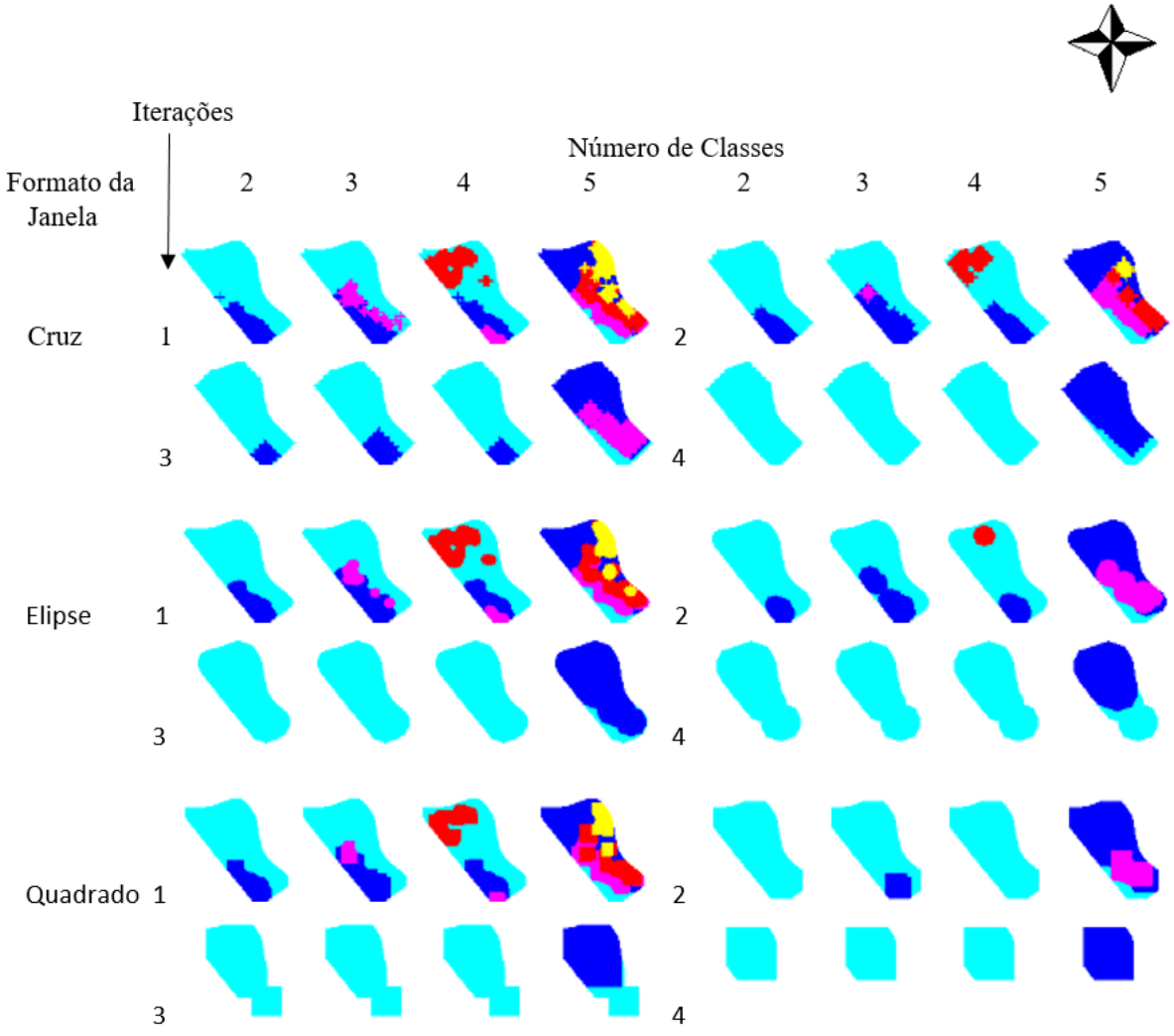

Figura 53 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de abertura, de 1 a 4 iterações, tamanho de janela 9x9, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

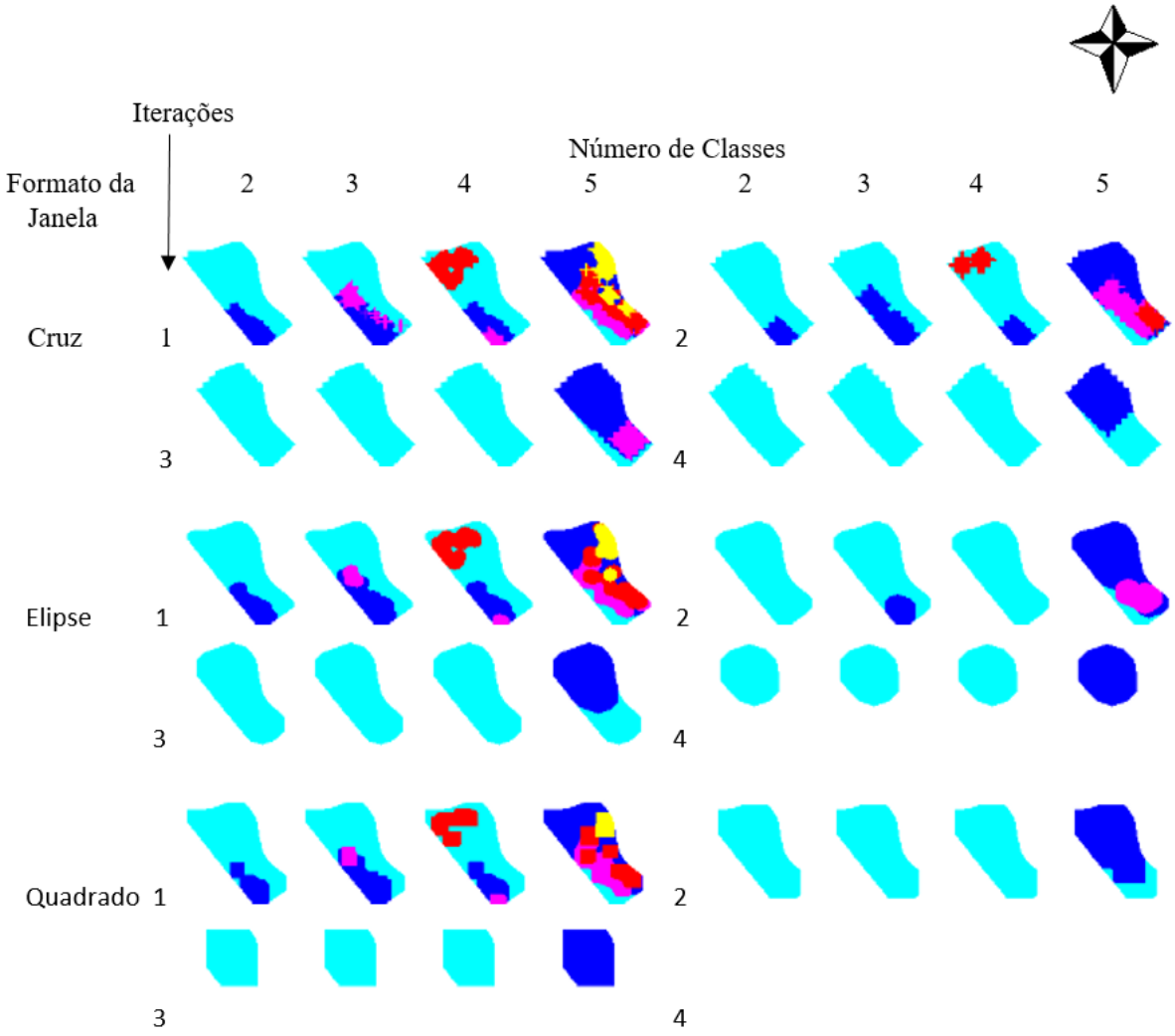

Figura 54 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de abertura, de 1 a 4 iterações, tamanho de janela 11x11, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

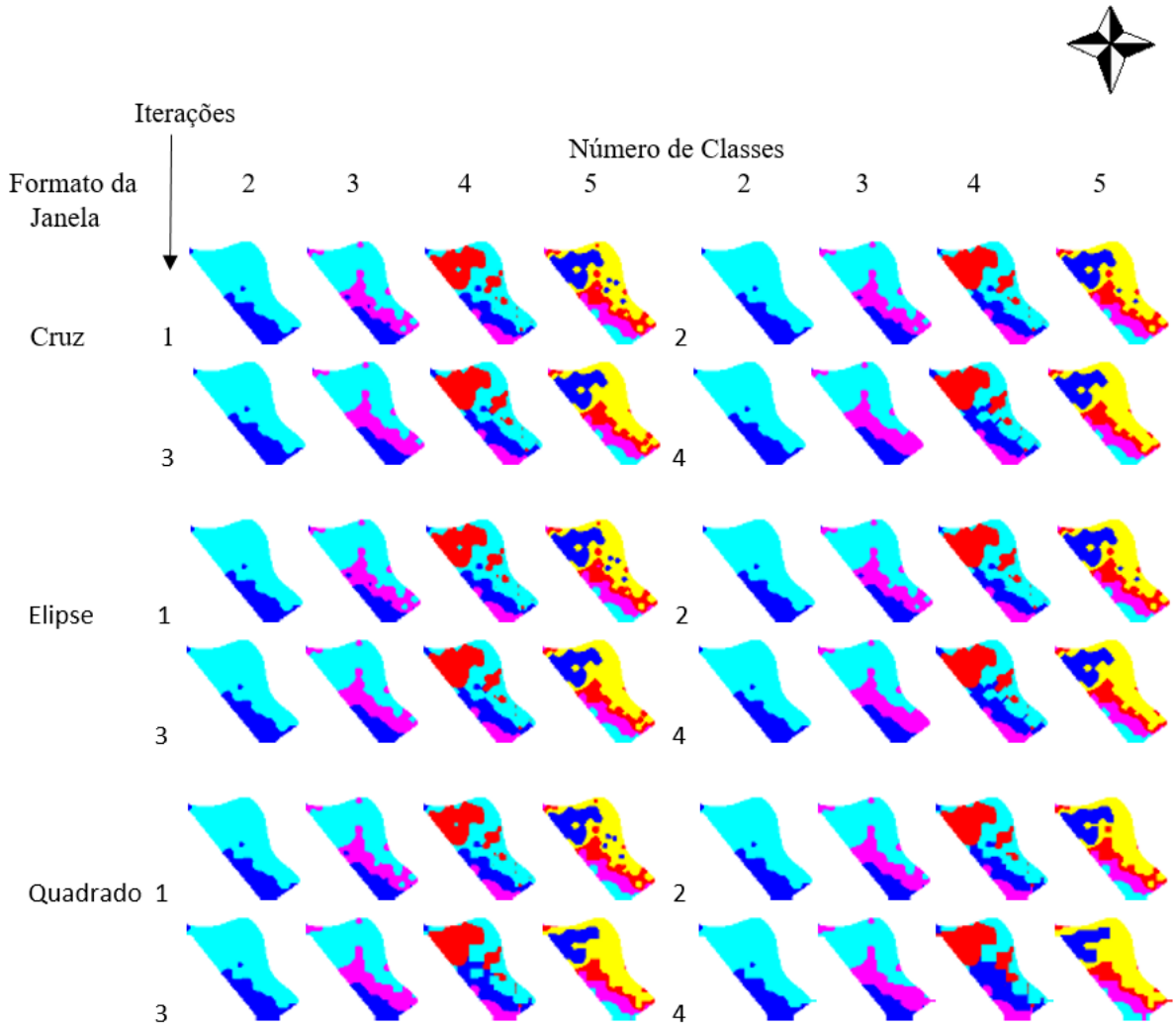

Figura 55 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de fechamento, de 1 a 4 iterações, tamanho de janela 3x3, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

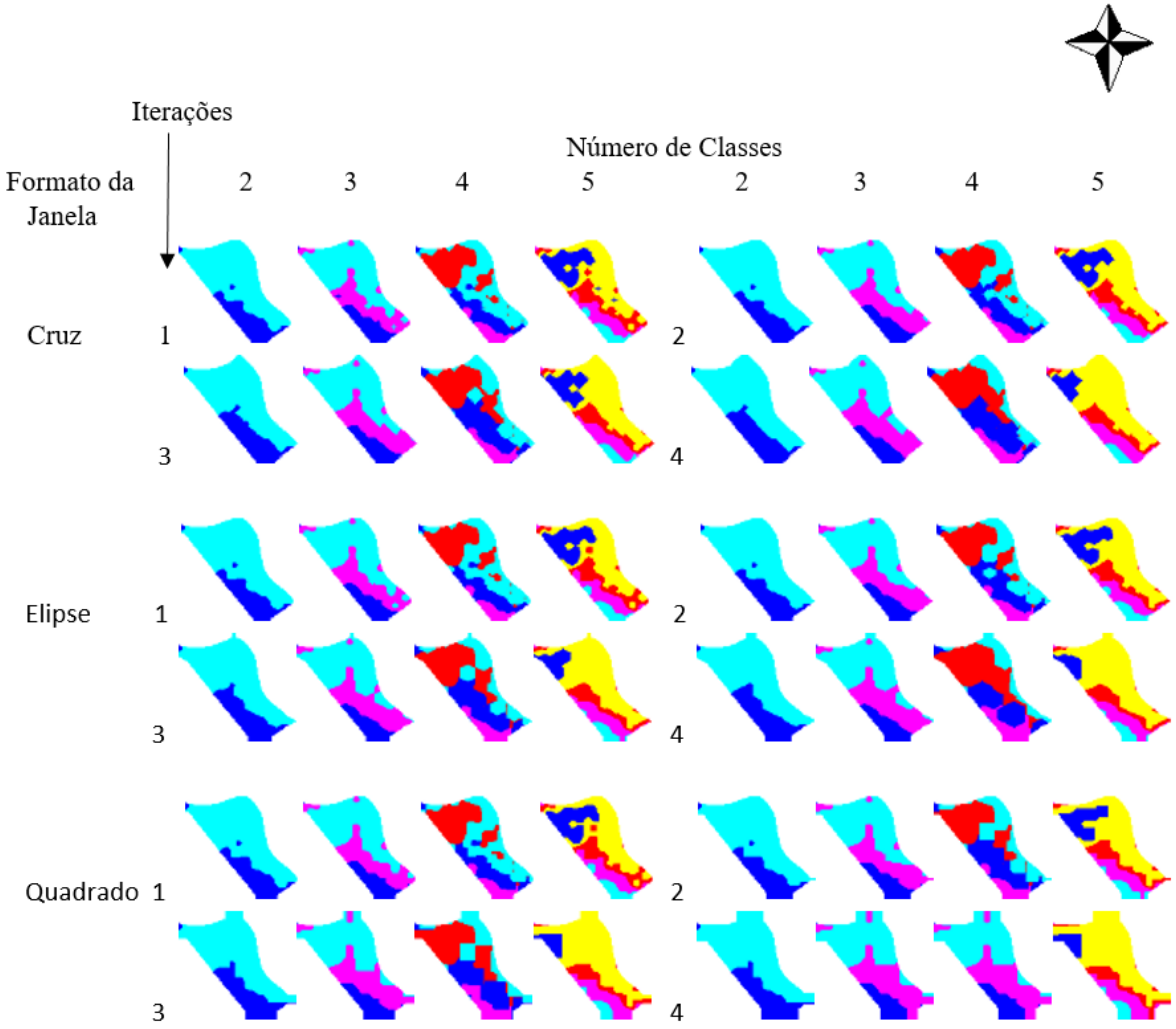

Figura 56 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de fechamento, de 1 a 4 iterações, tamanho de janela 5x5, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

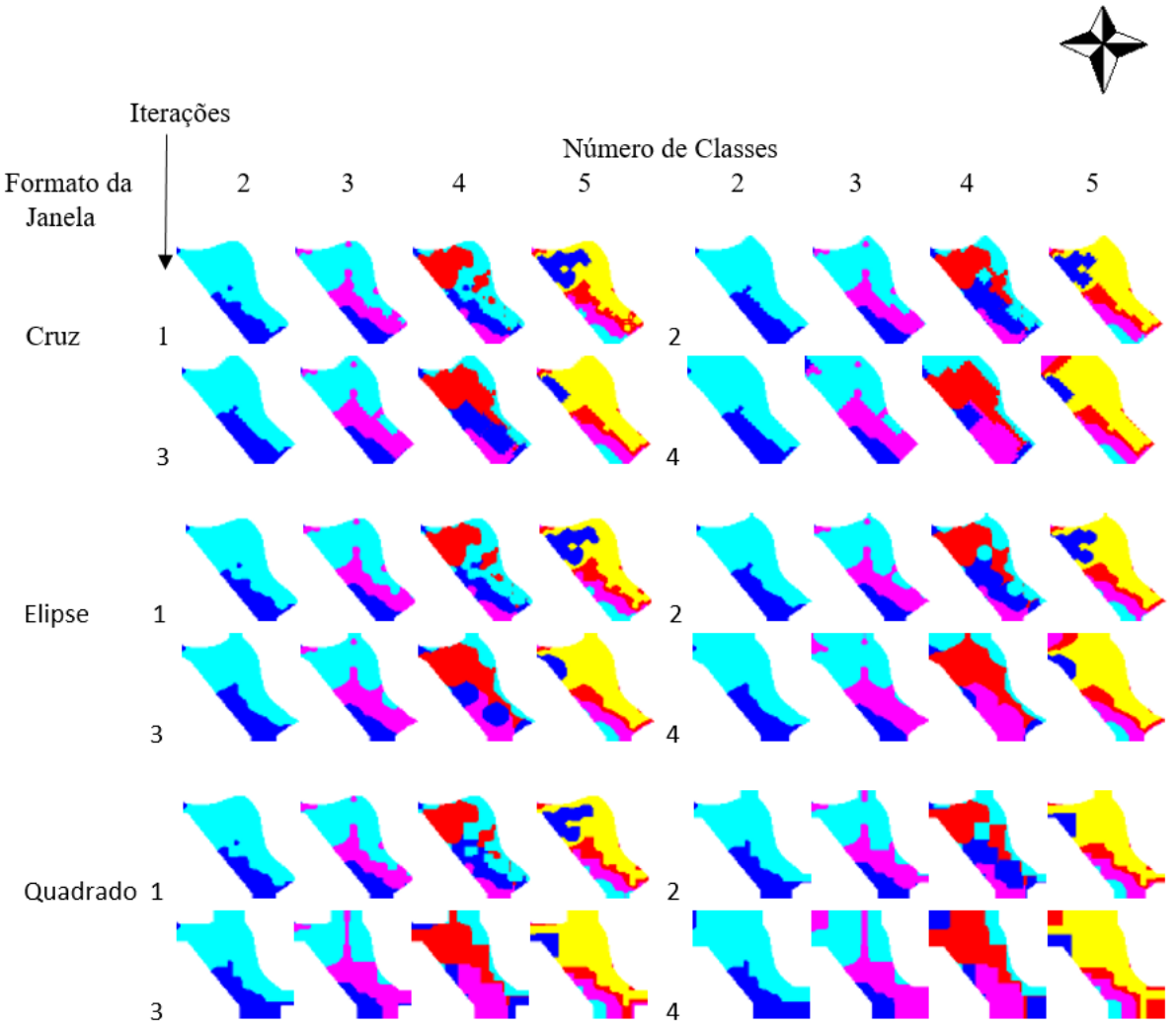

Figura 57 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de fechamento, de 1 a 4 iterações, tamanho de janela 7x7, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

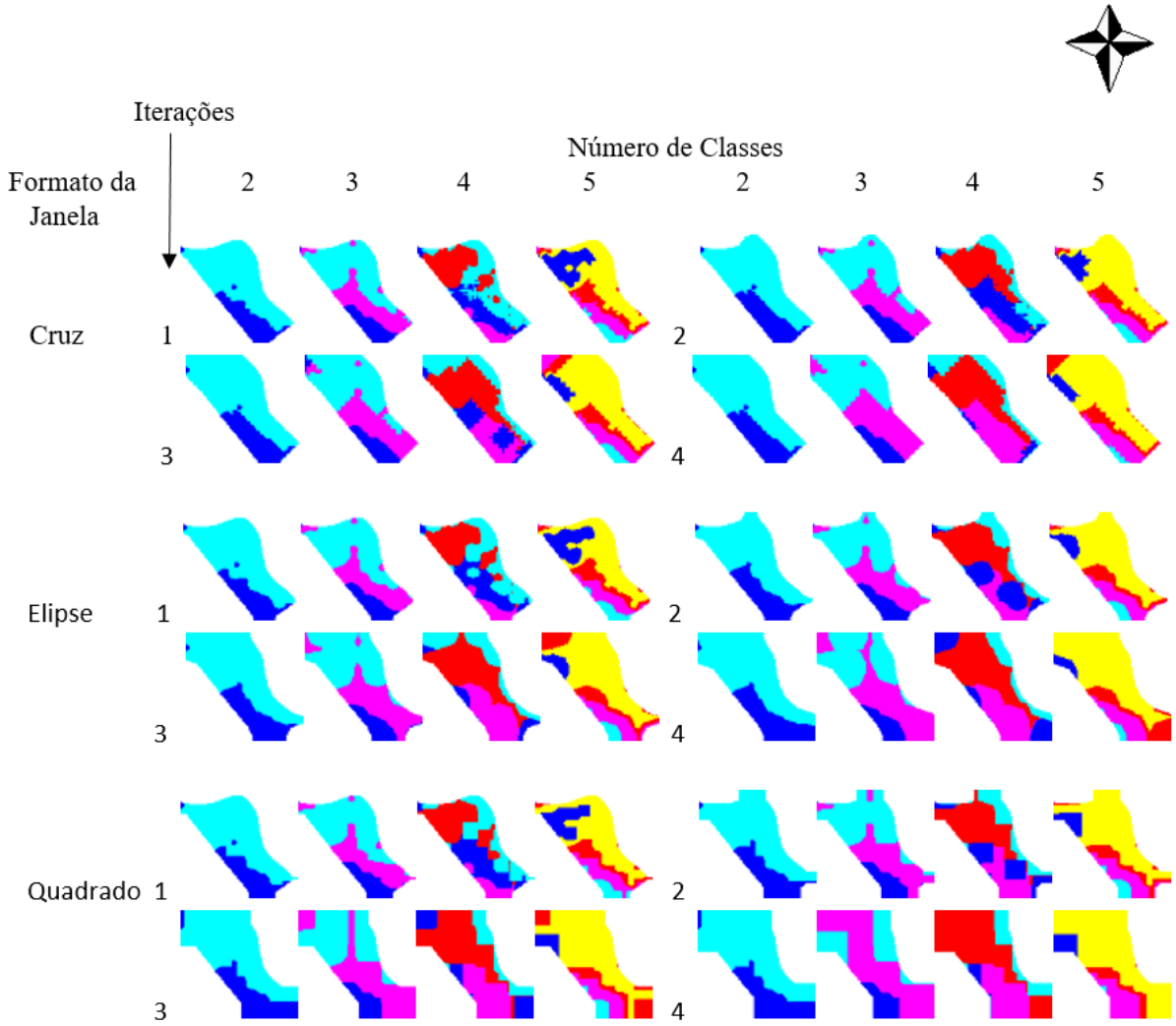

Figura 58 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de fechamento, de 1 a 4 iterações, tamanho de janela 9x9, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

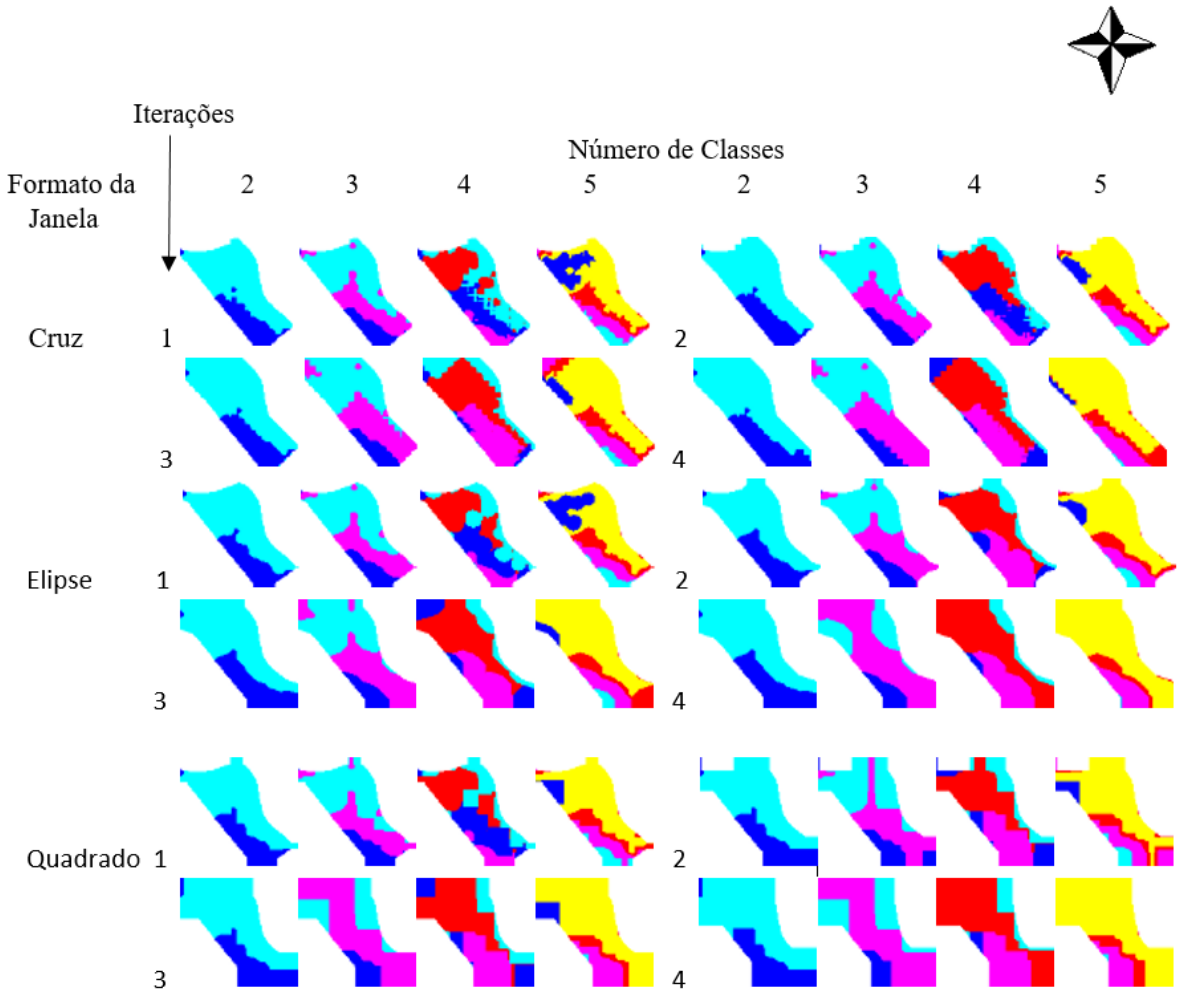

Figura 59 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de fechamento, de 1 a 4 iterações, tamanho de janela 11x11, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

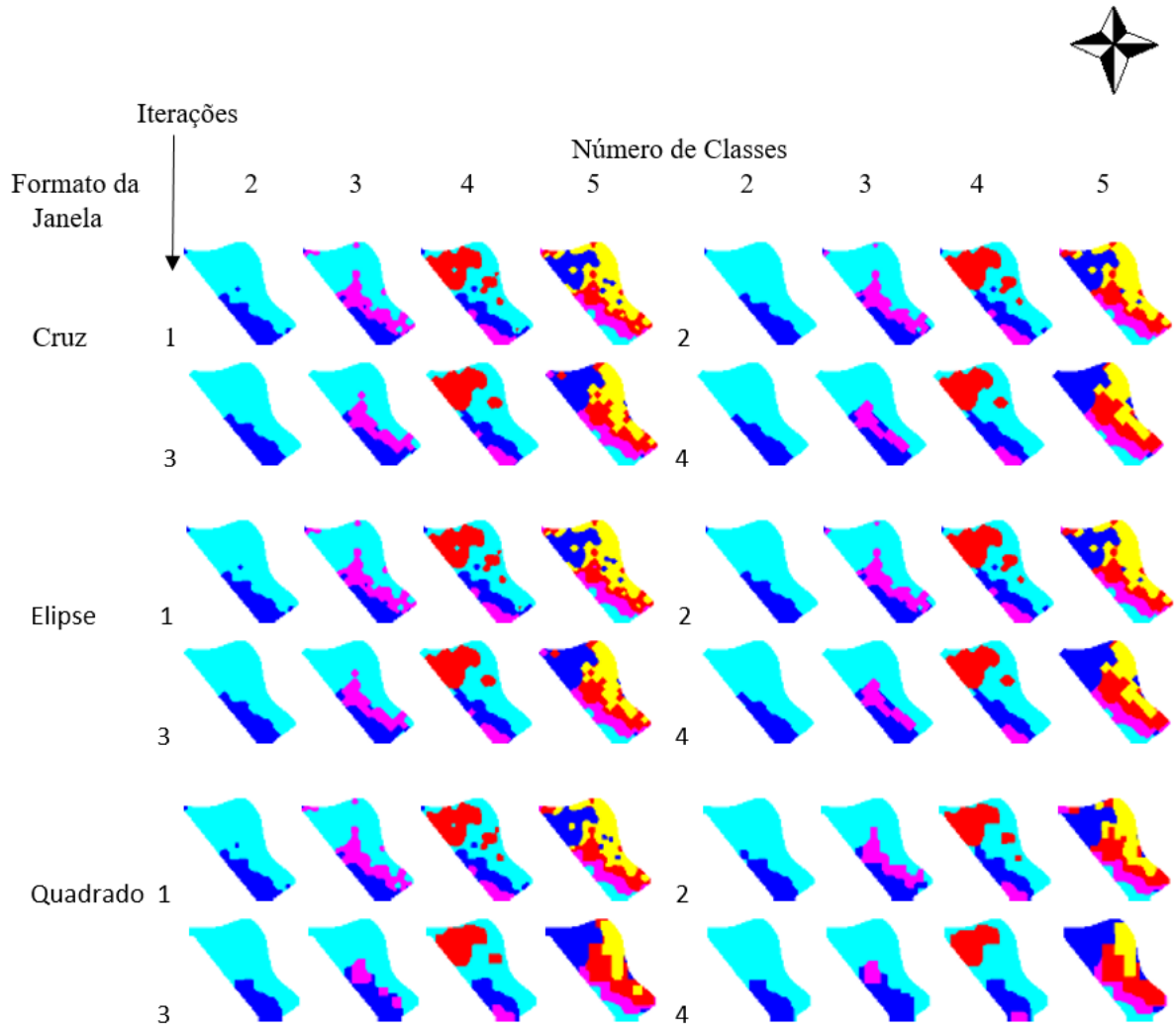

Figura 60 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de abertura seguido pelo fechamento, de 1 a 4 iterações, tamanho de janela 3x3, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

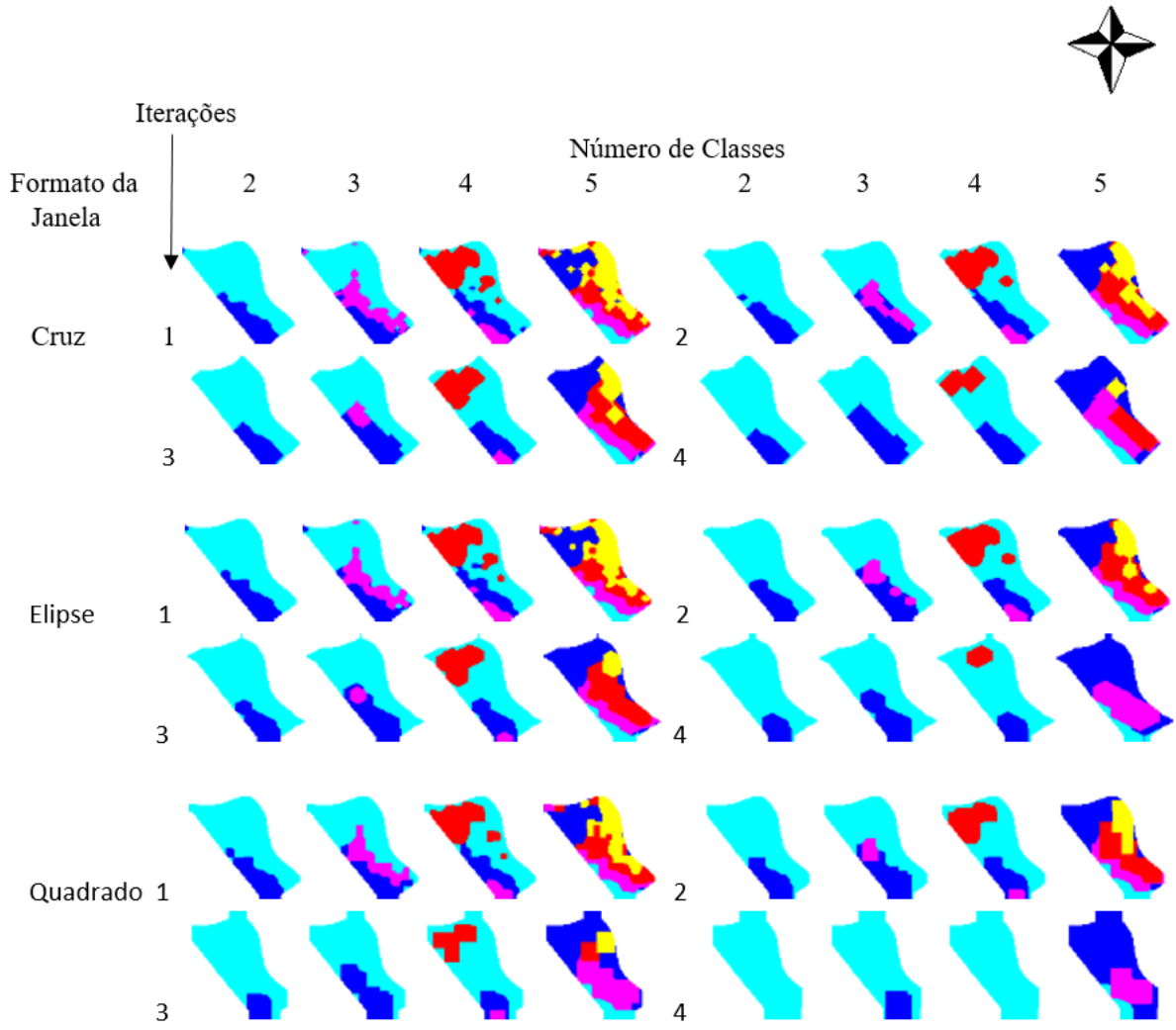

Figura 61 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de abertura seguido pelo fechamento, de 1 a 4 iterações, tamanho de janela 5x5, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

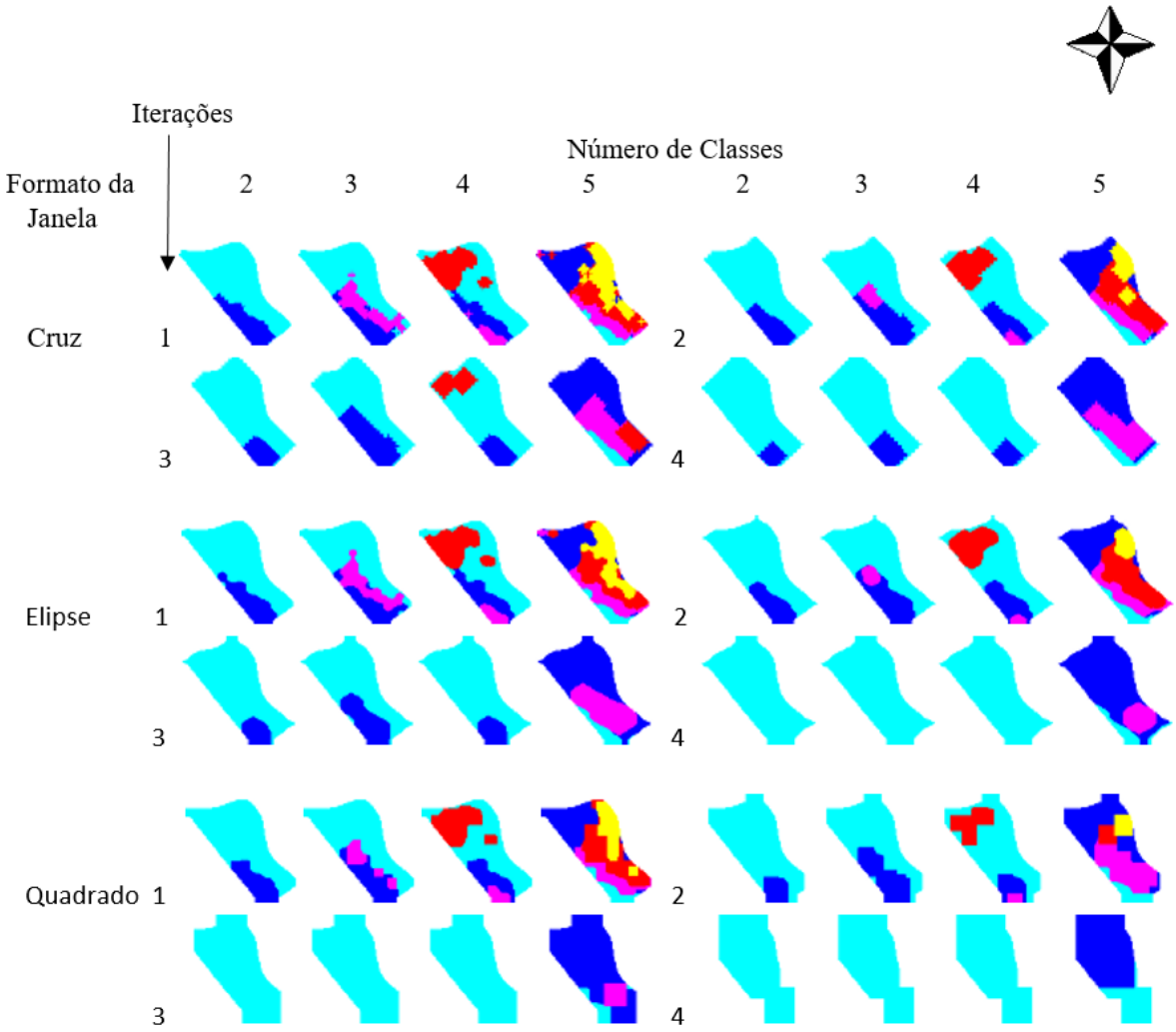

Figura 62 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de abertura seguido pelo fechamento, de 1 a 4 iterações, tamanho de janela 7x7, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

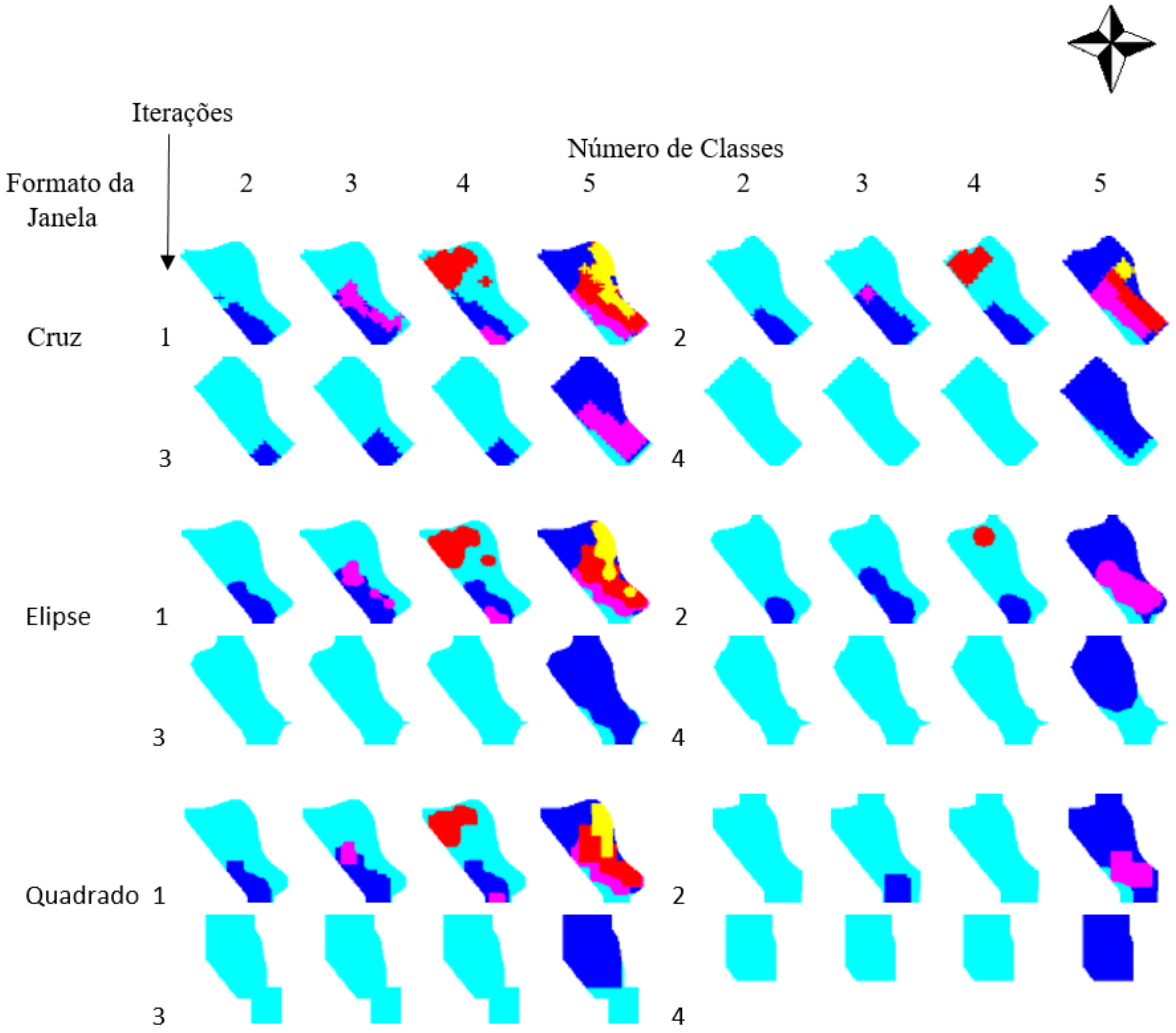

Figura 63 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de abertura seguido pelo fechamento, de 1 a 4 iterações, tamanho de janela 9x9, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

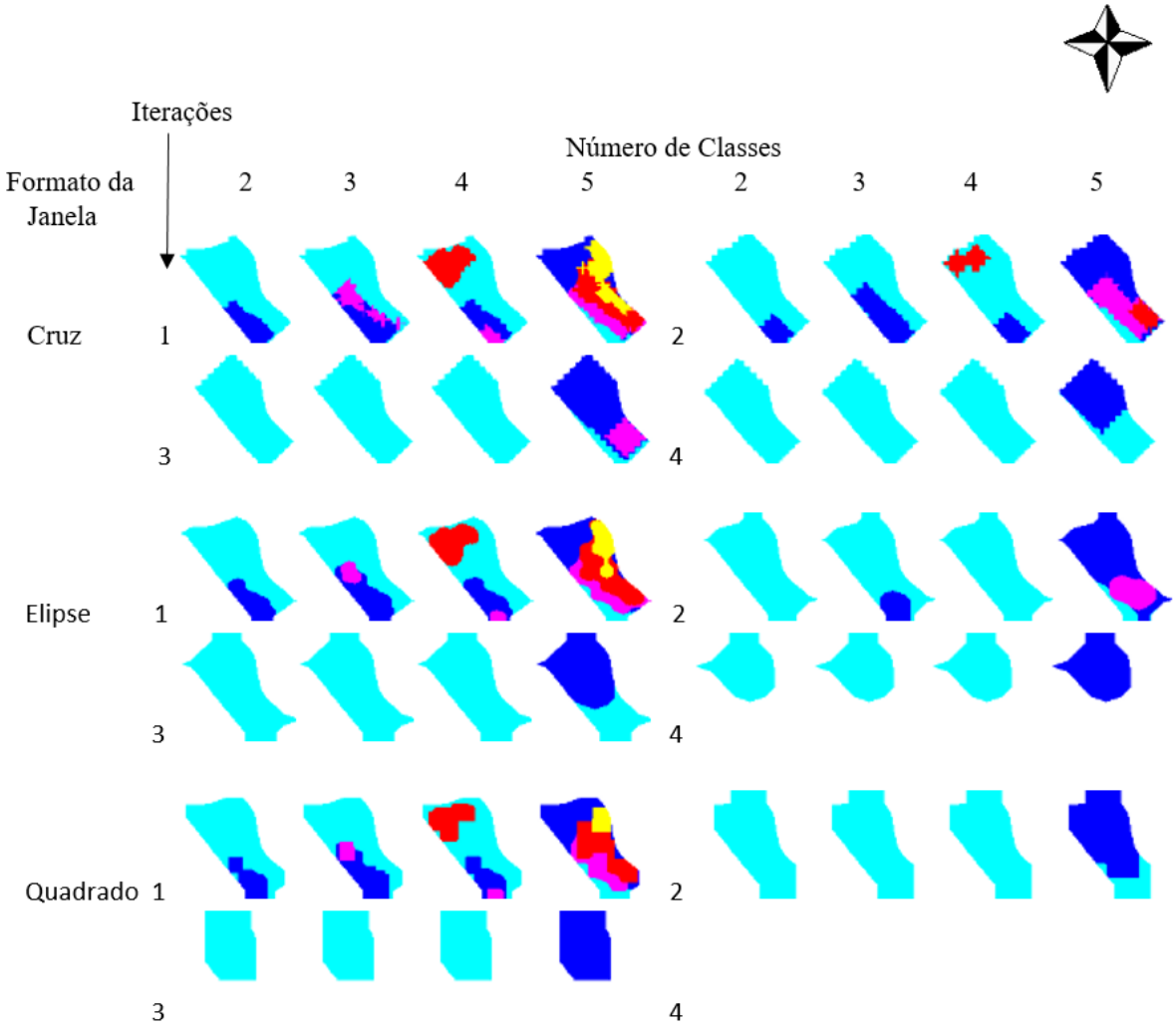

Figura 64 – Mapas da área B, divididos de 2 a 5 classes, retificados utilizando filtro de abertura seguido pelo fechamento, de 1 a 4 iterações, tamanho de janela 11x11, formatos de janela cruz, elipse e quadrado.

Fonte: Autoria própria.

REFERÊNCIAS

ABDUL, R. A.; GOH, K.; HEOH, T.; OSUMANU, H. A. et al. Transforming spatio-temporal yield maps to classified management zone maps for efficient management of oil palm. **American Journal of Applied Sciences**, Science Publications, v. 5, n. 10, p. 1392–1396, 2008.

AMARAL, O. Arquitetura de micro serviços: uma comparação com sistemas monolíticos. Universidade Federal da Paraíba, 2017.

ARAÚJO, J. de; VETTORAZZI, C.; MOLIN, J. et al. Grain crops yield estimate and management zones delineation through multispectral aerial videography. **Acta Scientiarum-Agronomy**, Universidade Estadual de Maringá, v. 27, n. 3, p. 437–447, 2005.

BARBOSA, D. P.; BOTTEGA, E. L.; VALENTE, D. S. M.; SANTOS, N. T.; GUIMARÃES, W. D.; FERREIRA, M. de P. Influence geometric anisotropy in management zones delineation. **Revista Ciência Agronômica**, v. 50, n. 4, p. 543–551, 2019.

BAZZI, C.; SOUZA, E.; RODRIGUES, S.; NÓBREGA, L.; OPAZO, M.; SANTOS, D.; KONOPATZKI, M.; SUSZEK, G. Definição de unidades de manejo para controle de plantas invasoras. **Avances en Ingeniería Rural**, v. 2009, p. 835–842, 2007.

BETZEK, N. M. et al. Módulos computacionais de análise geoestatística e retificação de zonas de manejo. Universidade Estadual do Oeste do Paraná, 2017.

BETZEK, N. M.; SOUZA, E. G. de; BAZZI, C. L.; SCHENATTO, K.; GAVIOLI, A. Rectification methods for optimization of management zones. **Computers and electronics in agriculture**, Elsevier, v. 146, p. 1–11, 2018.

BRUBECK-HERNANDEZ, F.; VLADIMIROVA, T.; POOLEY, M.; THOMPSON, R.; KNIGHT, B. Zone management in precision agriculture using satellite imagery. In: IEEE. **2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)**. [S.l.], 2019. p. 65–71.

CHRISTENSEN, B. **Introducing Hystrix for Resilience Engineering**. 2012. Disponível em: <https://medium.com/netflix-techblog/introducing-hystrix-for-resilience-engineering-13531c1ab362>. Acesso em: 01 de Julho de 2019.

FARID, H.; BAKHSH, A.; AHMAD, N.; AHMAD, A.; MAHMOOD-KHAN, Z. Delineating site-specific management zones for precision agriculture. **The Journal of Agricultural Science**, Cambridge University Press, v. 154, n. 2, p. 273–286, 2016.

FERGUSON, R. B.; HERGERT, G. W. Soil sampling for precision agriculture. **Extension, EC**, v. 154, 2009.

FRAISSE, C.; SUDDUTH, K.; KITCHEN, N. Delineation of site-specific management zones by unsupervised classification of topographic attributes and soil electrical conductivity. **Transactions of the ASAE**, American Society of Agricultural and Biological Engineers, v. 44, n. 1, p. 155, 2001.

GAVIOLI, A.; SOUZA, E. G. de; BAZZI, C. L.; GUEDES, L. P. C.; SCHENATTO, K. Optimization of management zone delineation by using spatial principal components. **Computers and Electronics in Agriculture**, Elsevier, v. 127, p. 302–310, 2016.

GAVIOLI, A.; SOUZA, E. G. de; BAZZI, C. L.; SCHENATTO, K.; BETZEK, N. M. Identification of management zones in precision agriculture: An evaluation of alternative cluster analysis methods. **Biosystems engineering**, Elsevier, v. 181, p. 86–102, 2019.

GONZALEZ, R. C.; WOODS, R. C. **Processamento digital de imagens .** [S.l.]: Pearson Educación, 2009.

GOOGLE, INC. **Google Maps**. 2019. Disponível em: <https://www.google.com/maps>. Acesso em: 13 de Dezembro de 2019.

INPE. **Teoria : Processamento de Imagens**. 2019. Disponível em: <http://www.dpi.inpe.br/spring/teoria/filtrage/filtragem.htm>. Acesso em: 22 de Agosto de 2019.

LONG, J. **Microservice Registration an Discovery with Spring Clud and Netflix's Eureka**. 2015. Disponível em: <https://spring.io/blog/2015/01/20/microservice-registration-and-discovery-with-spring-cloud-and-netflix-s-eureka/>. Acesso em: 01 de Julho de 2019.

LOWRANCE, C. Open source hardware and software in agriculture: an autonomous sap flow measurement wireless network a user friendly management zone delineation tool. University of Georgia, 2014.

MENESES, P. R.; ALMEIDA, T. d. et al. Introdução ao processamento de imagens de sensoriamento remoto. **Universidade de Brasília, Brasília**, 2012.

MOLIN, J. P. Definição de unidades de manejo a partir de mapas de produtividade. **Engenharia Agrícola**, v. 22, n. 1, p. 83–92, 2002.

MOLIN, J. P.; AMARAL, L. R. do; COLAÇO, A. **Agricultura de precisão**. [S.l.]: Oficina de Textos, 2015.

MORAL, F.; TERRÓN, J.; SILVA, J. M. D. Delineation of management zones using mobile measurements of soil apparent electrical conductivity and multivariate geostatistical techniques. **Soil and Tillage Research**, Elsevier, v. 106, n. 2, p. 335–343, 2010.

MOSHIA, M.; KHOSLA, R.; LONGCHAMPS, L.; REICH, R.; DAVIS, J.; WESTFALL, D. Precision manure management across site-specific management zones: grain yield and economic analysis. **Agronomy Journal**, The American Society of Agronomy, Inc., v. 106, n. 6, p. 2146–2156, 2014.

NAMIOT, D.; SNEPS-SNEPPE, M. On micro-services architecture. **International Journal of Open Information Technologies**, v. 2, n. 9, p. 24–27, 2014.

NEWMAN, S. **Building microservices: designing fine-grained systems**. [S.l.]: "O'Reilly Media, Inc.", 2015.

OLDONI, H.; TERRA, V. S. S.; TIMM, L. C.; JÚNIOR, C. R.; MONTEIRO, A. B. Delineation of management zones in a peach orchard using multivariate and geostatistical analyses. **Soil and Tillage Research**, Elsevier, v. 191, p. 1–10, 2019.

PIVOTAL, S. **Spring Cloud Netflix**. 2018. Disponível em: <https://spring.io/projects/spring-cloud-netflixoverview>. Acesso em: 01 de Julho de 2019.

PRAMANIK, S.; PRUSTY, S.; BHATTACHARJEE, D.; BHUNRE, P. K. A region-to-pixel based multi-sensor image fusion. **Procedia Technology**, Elsevier, v. 10, p. 654–662, 2013.

RAJESH, R. **Spring microservices**. [S.l.]: Packt Publishing Ltd, 2016.

REDHAT. **O que são os microsserviços?** 2019. Disponível em: <https://www.redhat.com/pt-br/topics/microservices/what-are-microservices>. Acesso em: 22 de Agosto de 2019.

SALERNO, R. **Spring Cloud Netflix: Load Balancer with Ribbon/Feign**. 2016. Disponível em: <https://dzone.com/articles/spring-cloud-netflix-load-balancer-with-ribbonfeig>. Acesso em: 01 de Julho de 2019.

SILVA, R. C. da. **Construindo microservices auto-curáveis com Spring Cloud e Netflix OSS**. 2019. Disponível em: <https://www.infoq.com/br/presentations/construindo-microservices-auto-curaveis-com-spring-cloud/>. Acesso em: 22 de Agosto de 2019.

SOUZA, J. P. **Balanceamento de carga em microsserviços com Spring Cloud Netflix**. 2018. Disponível em: <http://www.matera.com/blog/post/balanceamento-de-carga-em-microsservicos-com-spring-cloud-netflix>. Acesso em: 03 de Julho de 2019.

TSCHIEDEL, M.; FERREIRA, M. F. Introdução à agricultura de precisão: conceitos e vantagens. **Ciência Rural**, Universidade Federal de Santa Maria, v. 32, n. 1, p. 159–163, 2002.

VITHARANA, U. W.; MEIRVENNE, M. V.; SIMPSON, D.; COCKX, L.; BAERDEMAEKER, J. D. Key soil and topographic properties to delineate potential management classes for precision agriculture in the european loess area. **Geoderma**, Elsevier, v. 143, n. 1-2, p. 206–215, 2008.

XIANG, L.; YU-CHUN, P.; ZHONG-QIANG, G.; CHIN-JIANG, Z. Delineation and scale effect of precision agriculture management zones using yield monitor data over four years. Agricultural Sciences in China, 2007.

ZHA, H.; CAMMARANO, D.; WILSON, L.; LI, Y.; BATCHELOR, W.; MIAO, Y. Combining crop modelling and remote sensing to create yield maps for management zone delineation in small scale farming systems. In: **Precision agriculture'19**. [S.l.]: Wageningen Academic Publishers, 2019. p. 671–674.

ZHANG, C.; KOVACS, J. M. The application of small unmanned aerial systems for precision agriculture: a review. **Precision agriculture**, Springer, v. 13, n. 6, p. 693–712, 2012.

ZUUL. **Zuul**. 2018. Disponível em: <https://github.com/Netflix/zuul/wiki>. Acesso em: 01 de Julho de 2019.