MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA CARTOGRÁFICA

EDSON TADEU DA SILVA PINTO
JOÃO ALBERTO KREITLON PEREIRA
VINÍCIUS MAGALHÃES

DESENVOLVIMENTO DE PLUGIN QGIS PARA IDENTIFICAÇÃO E CORREÇÃO DE INTERSEÇÕES ENTRE CURVAS DE NÍVEL E MASSAS D'ÁGUA

RIO DE JANEIRO
2024

EDSON TADEU DA SILVA PINTO
JOÃO ALBERTO KREITLON PEREIRA
VINÍCIUS MAGALHÃES

DESENVOLVIMENTO DE PLUGIN QGIS PARA IDENTIFICAÇÃO E CORREÇÃO DE INTERSEÇÕES ENTRE CURVAS DE NÍVEL E MASSAS D'ÁGUA

Projeto de Fim de Curso apresentado ao Curso de Graduação em Engenharia Cartográfica do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia Cartográfica.

Orientador(es): Raphael Luiz França Greco, M.Sc.

Rio de Janeiro

2024

©2024
INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP: 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

Pinto, Edson Tadeu da Silva; Pereira, João Alberto Kreitlon; Magalhães, Vinícius.

Desenvolvimento de Plugin QGIS para Identificação e Correção de Interseções entre Curvas de Nível e Massas d'Água / Edson Tadeu da Silva Pinto, João Alberto Kreitlon Pereira e Vinícius Magalhães. – Rio de Janeiro, 2024.

54 f.

Orientador(es): Raphael Luiz França Greco.

Projeto de Fim de Curso (graduação) – Instituto Militar de Engenharia, Engenharia Cartográfica, 2024.

1. Modelo Digital de Elevação. 2. QGIS. 3. Hidrografia. 4. Curvas de Nível. 5. FABDEM. i. Greco, Raphael Luiz França (orient.) ii. Título

EDSON TADEU DA SILVA PINTO
JOÃO ALBERTO KREITLON PEREIRA
VINÍCIUS MAGALHÃES

Desenvolvimento de Plugin QGIS para Identificação e Correção de Interseções entre Curvas de Nível e Massas d'Água

Projeto de Fim de Curso apresentado ao Curso de Graduação em Engenharia Cartográfica do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia Cartográfica.

Orientador(es): Raphael Luiz França Greco.

Aprovada em 25 de outubro de 2024, pela seguinte banca examinadora:

Cap **Raphael Luiz França Greco** - M.Sc. do IME

TC **Ivanildo Barbosa** - D.Sc. do IME

Maj **Felipe Ferrari** - D.E. do IME

Rio de Janeiro
2024

Ao meu filho Paulo Ricardo, seu nascimento não só transformou minha vida, como marcou o início da minha caminhada neste Instituto de excelência.

(TADEU)

À minha avó Floraci e minha madrinha Ana cujas orações me guiaram para concluir esta caminhada.

(VINICIUS)

Aos meus avós, Arthur, Regina, Antonio e Palmira, que até hoje me acompanham nessa caminhada lá de cima.

(KREITLON)

AGRADECIMENTOS

À minha esposa Gabriela, um presente especial dado por Deus, sem ela não teria conseguido. Ao Cap Raphael Luiz, pela orientação, atenção e apoio a este projeto. Ao TC Ivanildo e Maj Ferrari, integrantes da banca examinadora, pelas considerações, orientações e sugestões que permitiram realizar um refinamento teórico e metodológico ao longo do período deste projeto. Aos docentes da SE-6, que além da nobre missão de serem professores, são responsáveis pelo ambiente salutar e aconchegante da "Cartô". Aos meus amigos e colegas da Turma 2024 de Eng Cartográfica, que levarei com alegria as lembranças de nossas resenhas e sanhas ao longo da graduação.

(TADEU)

À minha mãe Regina, cujo apoio e esforço me fez chegar até aqui. Aos meus amigos da Semi-residência que foram as melhores amizades que construí neste Instituto. Aos professores Cel Cerqueira e Maj. Gabriela que me ajudaram sempre que precisei durante o período básico. Ao TC Andrade e Cap Raphael Luiz que foram exemplo de líderes na coordenação e orientação, bem como exímios professores.

(VINICIUS)

Aos meus pais, Humberto e Luiza, que sempre me apoiaram e acreditaram em mim. Às minhas queridas irmãs, Juliana e Catarina, que iluminam meu mundo desde 2001. Aos nobres colegas da Semi-residência e da "Cartô", que fizeram do IME um lugar muito melhor, e a Giulia e Iago, amizades que eu levei do colégio para a vida. Ao TC Andrade, nobre coordenador e colega de Maracanã, e ao Cap. Raphael Luiz, exemplo de liderança e orientação nesse trabalho.

(KREITLON)

"Se suas ações inspiram outros a sonharem mais, aprenderem mais, e fazerem mais, então você é um cartógrafo."

John Quincy Adams

RESUMO

Este Projeto de Fim de Curso aborda a identificação de vetores de curvas de nível representantes da altimetria da superfície física do terreno que intersectam vetores de hidrografia presentes no processo de produção cartográfica dos Centros de Geoinformação (CGEO) do Exército Brasileiro a fim de adequá-los à Norma de Especificações Técnicas para Aquisição de Dados Geoespaciais Vetoriais (ET-ADGV 3.0). Aborda-se as especificações técnicas e o padrão de exatidão cartográfica (PEC) dos Modelos Digitais de Terreno (MDT), produto onde extrai-se as curvas de nível, com destaque aos do FABDEM (*Forest and Buildings removed Copernicus DEM*), licenciado pela *Creative Commons*, compatível com a licença adotada pela DSG (Diretoria de Serviço Geográfico) para seus produtos de geoinformação. O produto desta adequação é um *plugin* para o *software* QGIS que visa auxiliar e otimizar este processo nos CGEO.

Palavras-chave: Modelo Digital de Elevação. QGIS. Hidrografia. Curvas de Nível. FABDEM.

ABSTRACT

This Final Course Project addresses the identification of contour lines vectors representing the altimetry of the physical surface of the terrain with hydrography vectors existing in the Army Geographic Database (BDGEx) present in the cartographic production of the Geoinformation Centers (CGEO) of the Brazilian Army in order to correct them in order to make them consistent with the provisions of the Technical Specifications Standard for Vector Geospatial Data Acquisition (ET-ADGV 3.0). The technical specifications and cartographic accuracy standard (PEC) of Digital Terrain Models (MDT) are discussed, a product where contour lines are extracted, with emphasis on those from FABDEM (*Forest and Buildings removed Copernicus DEM*), licensed by *Creative Commons*, compatible with the license adopted by DSG (Geographic Service Directorate) for its geoinformation products. The correction product is a *plugin* for *software* QGIS that aims to assist and optimize the CGEO cartographic production process.

Keywords: Digital Elevation Model. QGIS. Hydrography. Contour Lines. FABDEM.

LISTA DE ILUSTRAÇÕES

Figura 1 – Exemplo de um MDT e da extração de Curva de Nível a partir dele. . 13
Figura 2 – Intersecção de curva de nível com massa d'água. 14
Figura 3 – Ilustração de curva de nível com barragem. 18
Figura 4 – Intersecção de curva de nível com barragem. 18
Figura 5 – Caso de uma CN intersectante na mesma margem. 19
Figura 6 – Caso de mais de uma CN intersectante na mesma margem. 19
Figura 7 – Caso de barragens, lagos e diques. 20
Figura 8 – Caso de CN dentro de massa d'água. 20
Figura 9 – Caso de CN dentro de massa d'água. 21
Figura 10 – *Framework* do método proposto. 22
Figura 11 – Workflow com a lógica atual do código implementado 23
Figura 12 – Curvas de nível com intersecção em massa d'água. 23
Figura 13 – Workflow da etapa de pré-processamento 24
Figura 14 – Interface do *processing* de adequação de curvas de nível às massas d'água. 24
Figura 15 – Workflow da etapa de identificação de intersecções 25
Figura 16 – Vetor de flags das intersecções. 25
Figura 17 – Workflow da etapa de aplicação dos buffers 26
Figura 18 – Criação do buffer em torno da massa d'água. 26
Figura 19 – Workflow da etapa de ajuste das curvas 27
Figura 20 – Extração da linha de fronteira do buffer 27
Figura 21 – Secção das linhas de fronteira com as curvas de nível. 27
Figura 22 – Recorte das curvas de nível que sobrepõem o buffer. 28
Figura 23 – Inserção dos trechos da linha de buffer nas curvas de nível faltantes. . . 28
Figura 24 – Vetor de flag de pontas soltas. 28
Figura 25 – Exemplo de correção bem-sucedida após a utilização do código 31
Figura 26 – Exemplo de correção após a utilização do código 31
Figura 28 – Exemplo de correção mal-sucedida após a utilização do código 32
Figura 29 – Exemplo de correção mal-sucedida após a utilização do código 33
Figura 30 – Distribuição espacial das regiões analisadas 33
Figura 31 – Área analisada sobre a região de Varginha - MG 34
Figura 32 – Área analisada sobre a região de Alegrete - RS 35
Figura 33 – Área analisada sobre a região do litoral de Pernambuco 36
Figura 34 – Área analisada sobre a região de Boa Vista - RR 37

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO . . . 13
1.1 CONTEXTUALIZAÇÃO . . . 13
1.2 OBJETIVO . . . 15
1.3 JUSTIFICATIVA E MOTIVAÇÃO . . . 15
1.4 ESTRUTURA DO TRABALHO . . . 15

2 FUNDAMENTAÇÃO TEÓRICA . . . 16
2.1 MODELOS DIGITAIS DE ELEVAÇÃO E CURVAS DE NÍVEL . . . 16
2.2 NORMAS TÉCNICAS . . . 17
2.2.1 ET-ADGV . . . 17
2.3 CASOS DE INTERSECÇÃO DE CURVAS DE NÍVEL COM MASSAS D'ÁGUA 18
2.4 QGIS . . . 21
2.5 FLAGS . . . 21

3 MÉTODO E MATERIAIS . . . 22
3.1 MÉTODO . . . 22
3.1.1 VISÃO GERAL DO CÓDIGO . . . 23
3.1.2 PRÉ-PROCESSAMENTO . . . 24
3.1.3 IDENTIFICAÇÃO DE INTERSECÇÕES . . . 25
3.1.4 APLICAÇÃO DOS BUFFERS . . . 26
3.1.5 AJUSTE DAS CURVAS . . . 27
3.2 MATERIAIS . . . 29
3.2.1 MICRO-COMPUTADOR . . . 29
3.2.2 MODELOS DIGITAIS DE TERRENO . . . 29
3.2.3 VETORES DE CURVAS DE NÍVEL . . . 29
3.2.4 VETOR DE MASSAS D'ÁGUA . . . 30
3.2.5 *SOFTWARES* . . . 30

4 RESULTADOS E DISCUSSÕES . . . 31
4.1 SAÍDAS OBTIDAS . . . 31
4.1.1 UMA CURVA INTERSECTANDO EM MESMA MARGEM . . . 31
4.1.2 MAIS DE UMA CURVA INTERSECTANDO EM MESMA MARGEM . . . 31
4.1.3 BARRAGENS, LAGOS, DIQUES . . . 32
4.1.4 CURVA QUE CRUZA RIOS . . . 32
4.1.5 CURVA DENTRO DE MASSA D'ÁGUA . . . 33
4.2 REGIÕES ANALISADAS . . . 33

4.2.1 CASOS DE INTERSEÇÃO DE CURVAS DE NÍVEL COM MASSAS D'ÁGUA - MINAS GERAIS 34
4.2.2 CASOS DE INTERSEÇÃO DE CURVAS DE NÍVEL COM MASSAS D'ÁGUA - RIO GRANDE DO SUL 35
4.2.3 CASOS DE INTERSEÇÃO DE CURVAS DE NÍVEL COM MASSAS D'ÁGUA - PERNAMBUCO 36
4.2.4 CASOS DE INTERSEÇÃO DE CURVAS DE NÍVEL COM MASSAS D'ÁGUA - RORAIMA 37

5 CONCLUSÃO 39

REFERÊNCIAS 40

ANEXO A – CÓDIGO-FONTE DO COMPLEMENTO 41

1 INTRODUÇÃO

1.1 Contextualização

O mapeamento sistemático do território brasileiro prevê como um dos elementos obrigatórios de uma carta topográfica a representação do relevo por curvas de nível e pontos cotados, conforme o Norma da Especificação Técnica para Aquisição de Dados Geoespaciais Vetoriais (ET-ADGV) Versão 3.0. As curvas de nível fornecem uma representação visual da forma da superfície da terra e são essenciais para diversas aplicações, como planejamento urbano, engenharia civil e gestão ambiental (CHAGAS et al., 2010).

As curvas de nível, no processo de produção cartográfica, são extraídas a partir do Modelo Digital do Terreno (MDT) (TOUTIN; GRAY, 2000), fornecendo informações detalhadas sobre o relevo de uma determinada área, a Figura 1 mostra um exemplo dessa extração.

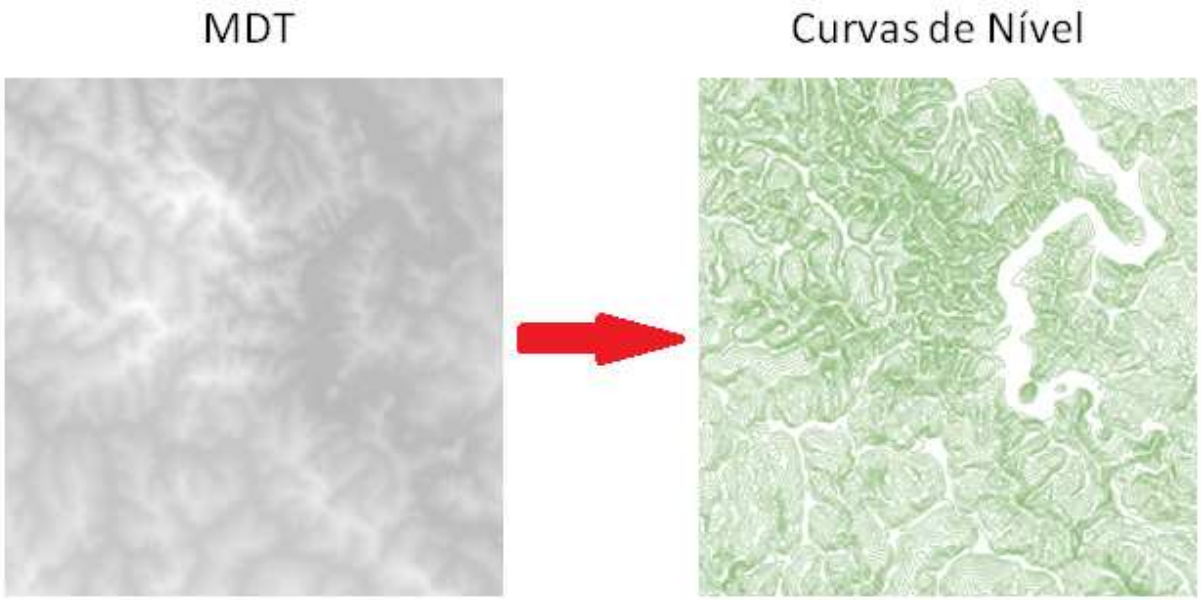


Figura 1 – Exemplo de um MDT e da extração de Curva de Nível a partir dele. Fonte: Autores (2024)

Neste contexto, a geração de um MDT pode ser realizada por imagens de radar (*radio detection and ranging*). A capacidade do radar em penetrar nuvens, vegetação densa e até mesmo operar em condições noturnas, torna-o uma escolha para aplicações de sensoriamento remoto (HASSELMANN et al., 1985).

No entanto, a ET-ADGV v3.0 prevê regras para os vetores de curvas de nível que intersectam vetores de massas d'água, tais como rios, lagos e reservatórios. Ao tratar

da altimetria dentro de massas d'água, é necessário recorrer à batimetria, uma técnica específica que leva em conta a natureza da superfície aquática (TELFORD; GELDART; SHERIFF, 1990), que neste caso seria uma curva batimétrica. Portanto, é importante garantir que as curvas de nível estejam adequadas à referida norma, a fim de manter o que ela convenciona. A Figura 2 mostra um exemplo de intersecção de curva de nível com massa d'água, com auxílio de uma imagem óptica.

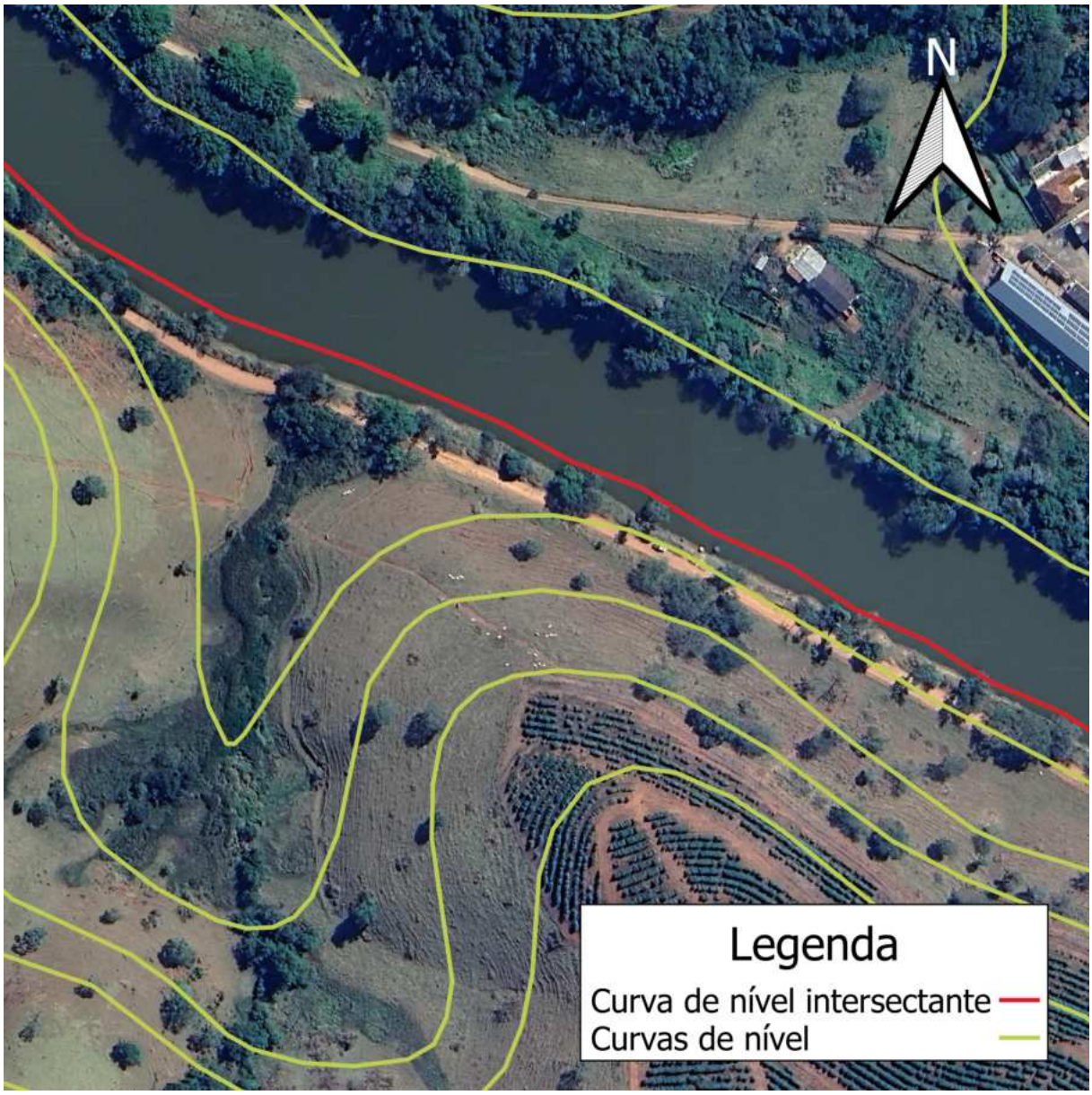


Figura 2 – Exemplo de uma intersecção de curva de nível com massa d´água. Fonte: Autores (2024)

Havendo-se a necessidade da edição vetorial destas curvas de nível para se adequar ao que a norma convenciona, tem-se o QGIS como uma ferramenta de edição vetorial de código aberto amplamente utilizada por profissionais e pesquisadores, com comunidade ativa e oficialmente utilizado pela Diretoria de Serviço Geográfico (DSG) para realizar este tipo de manipulação. Desenvolvido pela comunidade global de geoinformação, o software

QGIS oferece uma gama de funcionalidades para a análise, visualização e gestão de dados geoespaciais (GRASER et al., 2017).

1.2 Objetivo

Este projeto de fim de curso tem como objetivo implementar um complemento (*plugin*) de QGIS para adequar os vetores de curvas de nível com os vetores de massa d'água de forma que estejam coerentes com o previsto na Norma de Especificações Técnicas para Aquisição de Dados Geoespaciais Vetoriais (ET-ADGV 3.0).

1.3 Justificativa e motivação

O QGIS tem se mostrado um *software* para aquisição de dados geoespaciais, sua característica *open source* permite o desenvolvimento e aplicação de soluções que automatizam processos de edição de dados geoespaciais e vetoriais de forma a otimizar o processo de produção cartográfica nos Centros de Geoinformação (CGeo), fornecendo ferramentas úteis aos militares que realizam a aquisição destes dados.

No atual processo de produção cartográfica, a inadequação dos vetores de curvas de nível ao previsto na ET-ADGV tem se mostrado crítica, considerando que não há uma rotina que automatize essa alteração, sendo realizada manualmente pelos operadores nos CGeo. Por conta disso, torna-se uma etapa lenta e com tendência a erros grosseiros.

1.4 Estrutura do trabalho

No capítulo 2 (Fundamentação teórica) são apresentados os conceitos teóricos que norteiam o presente projeto: modelos digitais de elevação (MDE), radar, FABDEM (*Forest and Buildings removed Copernicus DEM*), conceito de curvas de nível, batimetria e as normas técnicas.

O capítulo 3 (Método e materiais) traz ao conhecimento o método empregado para o desenvolvimento do *plugin* de identificação e adequação dos vetores de curvas de nível bem como os materiais empregados ao longo do projeto, com destaque aos dados geoespaciais e vetoriais utilizados.

No capítulo 4 (Resultados) são apresentados os resultados que se esperam do desenvolvimento e implementação do complemento de QGIS para adequação dos vetores de curvas de nível às massas d'água.

No último capítulo (Conclusão) são apresentadas as considerações finais e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Para atingir o objetivo, faz-se necessário o estudo das Normas Técnicas utilizadas pela DSG a fim de verificar o que elas convencionam quanto aos dados geoespaciais e vetoriais, principalmente quanto às curvas de nível, massas d'água e MDT.

O insumo a ser utilizado como o MDT será o do FABDEM (*Forest and Buildings removed Copernicus DEM*) pois é amplamente utilizado pelos CGeo.

Também faz-se necessário o conhecimento quanto ao *plugin* DSGTools que pode ser encontrado no software QGIS e que é uma ferramenta amplamente difundida e utilizada pelos operadores nos CGeo.

2.1 Modelos Digitais de Elevação e Curvas de Nível

As curvas de nível são extraídas a partir de Modelos Digitais de Elevação (MDE), que representam numericamente a superfície da Terra e fornecem informações sobre a elevação em diferentes regiões. Existem dois tipos principais de MDE: o Modelo Digital de Superfície (MDS), que inclui objetos naturais e artificiais presentes na superfície, e o Modelo Digital de Terreno (MDT), que representa a topografia natural, excluindo tais objetos (GUTH et al., 2021).

A aquisição de MDE pode ser realizada por meio de técnicas de sensoriamento remoto, como radar, lidar e fotogrametria aérea (CHANG et al., 2004). O radar, por exemplo, oferece vantagens ao penetrar vegetação densa e operar em qualquer condição climática e hora do dia. No entanto, possui limitações como a necessidade de processamento sofisticado e ruídos na imagem (TREVETT, 2013).

O FABDEM (*Forest and Buildings removed Copernicus DEM*) é um exemplo de MDT utilizado neste projeto, obtido após a remoção de florestas e edifícios de um MDS, proporcionando uma representação mais precisa do terreno natural (HAWKER et al., 2022). Este modelo é amplamente utilizado na produção cartográfica e será a principal fonte para a extração das curvas de nível.

As curvas de nível representam a topografia ao conectar pontos de igual elevação, sendo elementos essenciais para a modelagem de terrenos em Sistemas de Informações Geográficas (SIG) (MORSE, 1969). No âmbito deste trabalho, as curvas de nível são extraídas automaticamente a partir do MDT usando ferramentas como o QGIS, que permite a criação de vetores para análise e correção de intersecções com massas d'água.

A relação entre as curvas de nível e as massas d'água é fundamental, visto que erros

nas intersecções podem comprometer a qualidade da cartografia. Neste projeto, foram identificados e resolvidos os seguintes casos de intersecção: curvas de nível intersectando a mesma margem, mais de uma curva de nível intersectando a mesma margem e curvas de nível dentro de massas d'água. No entanto, outros casos, como intersecções em barragens, lagos, diques e curvas de nível que cruzam rios, permanecem como desafios a serem abordados em trabalhos futuros.

A metodologia aplicada divide-se em três etapas: identificação das intersecções, adequação às normas técnicas e resolução de situações particulares, garantindo que o processo de correção esteja em conformidade com os padrões estabelecidos.

2.2 Normas Técnicas

A Norma da Especificação Técnica para Aquisição de Dados Geoespaciais Vetoriais (ET-ADGV) Versão 3.0 apresenta um catálogo de objetos usado para descrever a semântica da cartografia brasileira, definindo as regras de aquisição das classes de objetos alinhadas às ET-EDGV, particularmente, a classe de curva de nível e de massa d'água, que interessam a este projeto.

2.2.1 ET-ADGV

A ET-ADGV define diretrizes detalhadas para a coleta, armazenamento, processamento e distribuição de dados vetoriais, abrangendo diversos aspectos técnicos e operacionais. Ela especifica os formatos de dados, sistemas de referência, precisão posicional, classificação temática e outros atributos essenciais dos dados geoespaciais vetoriais.

Para as curvas de nível em situações próximas às massas d'água, a ET-ADGV v3.0 prevê as seguintes particularidades:

a) No encontro de Barragem, Corte ou Aterro, segue-se a regra de que a geometria dos objetos da classe *Curva Nivel* deve ser traçada de forma contínua, ou seja, sem secções nas linhas e em casos de paredões ocasionados por cortes, aterros ou barragens de estrutura rígida, cujo o ângulo destes sejam de 90º com o horizonte pode ocorrer sobreposições visuais. Conforme a Figura 3.

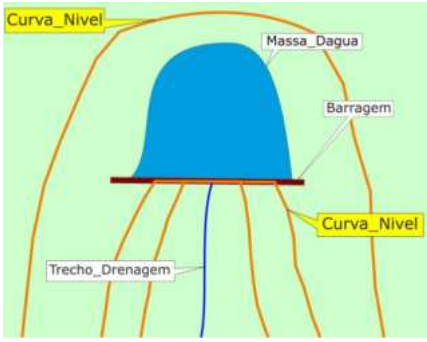


Figura 3 – Ilustração de curva de nível com barragem conforme a ET-ADGV.

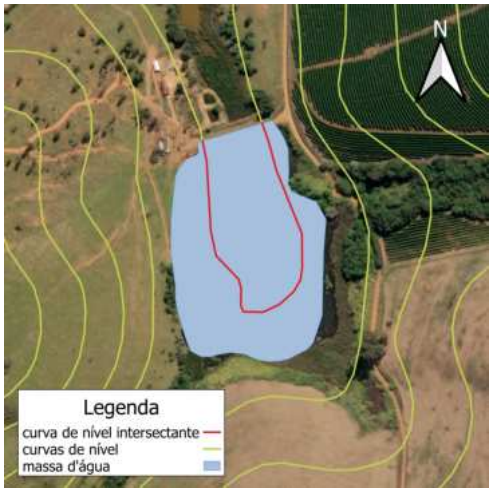


Figura 4 – Intersecção de curva de nível com barragem.

2.3 Casos de intersecção de curvas de nível com massas d'água

Realizando a investigação dos casos de intersecções encontradas, verificou-se as seguintes situações:

- Uma curva de nível intersectando mesma margem;

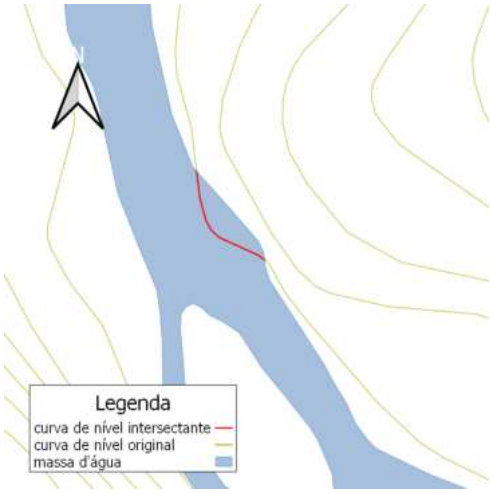


Figura 5 – Caso de uma CN intersectante na mesma margem.

- Mais de uma curva de nível intersectando mesma margem;

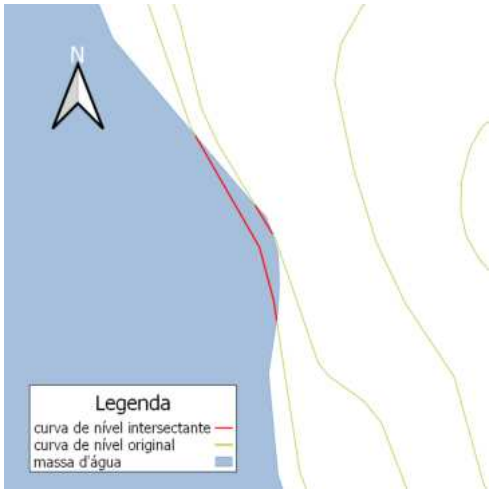


Figura 6 – Caso de mais de uma CN intersectante na mesma margem.

- Barragens, lagos e diques;

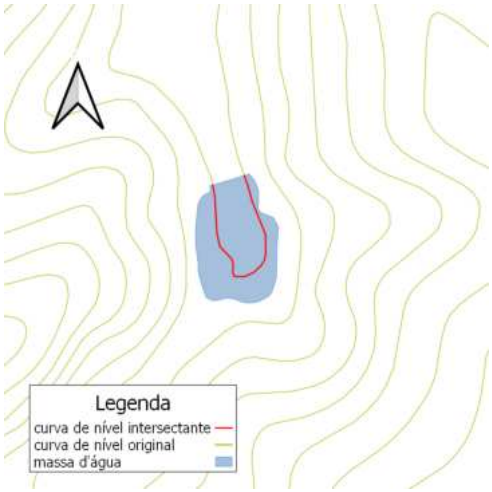


Figura 7 – Caso de barragens, lagos e diques.

- Curva de nível que cruza rios;

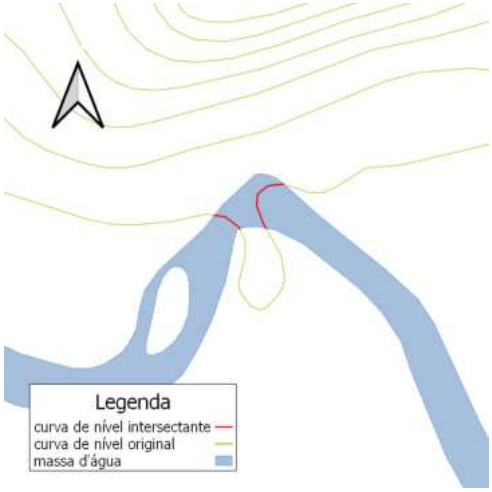


Figura 8 – Caso de CN dentro de massa d'água.

- Curva de nível dentro de massa d'água;

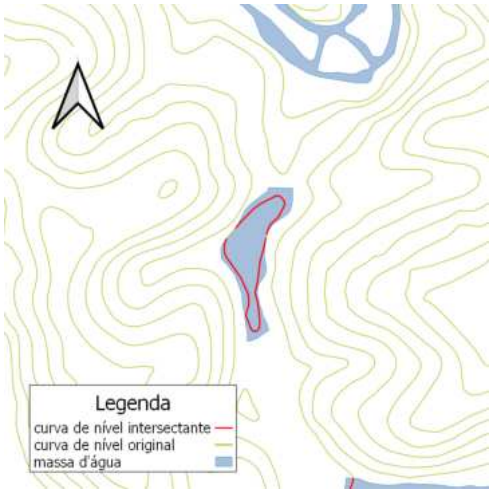


Figura 9 – Caso de CN dentro de massa d'água.

2.4 QGIS

O QGIS, sendo um *software* de código aberto, possibilita a personalização e a criação de ferramentas adicionais que atendem a necessidades específicas dos usuários, ampliando suas capacidades em diversas áreas de aplicação (GRASER et al., 2017). Um exemplo disso é o DSGTools, desenvolvido pela DSG para otimizar a produção cartográfica dos CGeo, oferecendo rotinas que seguem as Normas Técnicas da Infraestrutura Nacional de Dados Espaciais (INDE) e auxiliam na aquisição, validação e edição de dados geoespaciais (DSG, 2021). Além disso, o suporte a scripts em Python permite a criação de automações e tarefas personalizadas para análise geoespacial e processamento de dados (LAWHEAD, 2017).

2.5 Flags

Um vetor de flags é um tipo de estrutura de dados, normalmente utilizado em programação e sistemas geoespaciais, que armazena indicadores binários (geralmente "verdadeiro"ou "falso", ou "1"e "0") para sinalizar a ocorrência ou não de determinadas condições em um conjunto de dados. Cada "flag"(ou bandeira) dentro desse vetor representa uma condição ou estado específico.

No contexto de sistemas de informações geográficas (SIG) e de processamento geoespacial, um vetor de flags pode ser usado para marcar ou identificar problemas, como erros geométricos, intersecções malformadas ou dados que requerem correção. Por exemplo, em um processo de correção de intersecções de curvas de nível com massas d'água, um vetor de flags pode ser gerado para sinalizar onde há pontas soltas ou intersecções incorretas, ajudando o operador ou o software a identificar e corrigir esses problemas.

3 MÉTODO E MATERIAIS

Esta seção do trabalho tem como objetivo apresentar o método proposto para a implementação do código em linguagem Python como complemento para QGIS com a finalidade de adequar os vetores de curvas de nível com os vetores de massas d'água.

3.1 Método

A Figura 10 apresenta a esquematização geral do método proposto, que basicamente é dividido em três etapas.

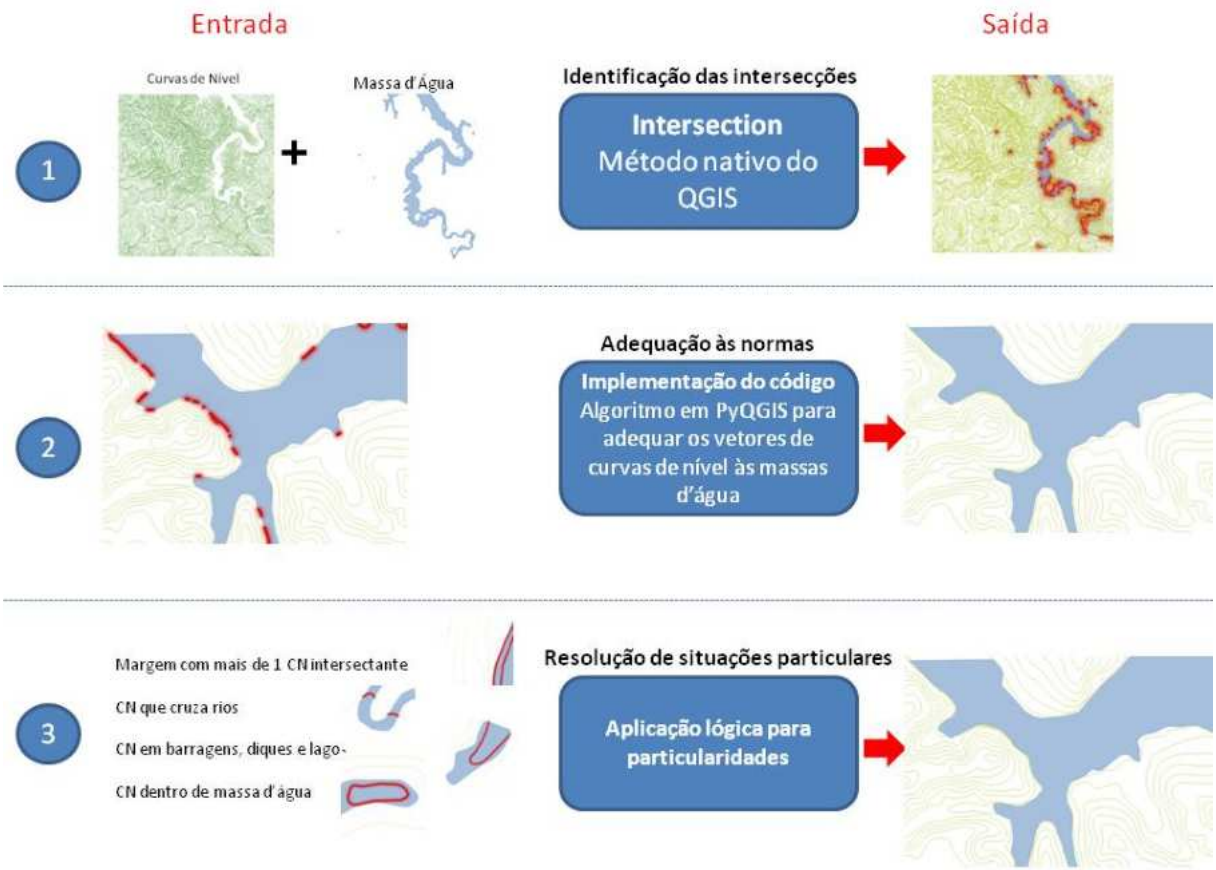


Figura 10 – Framework do método proposto. Fonte: Autores (2024)

3.1.1 Visão geral do código

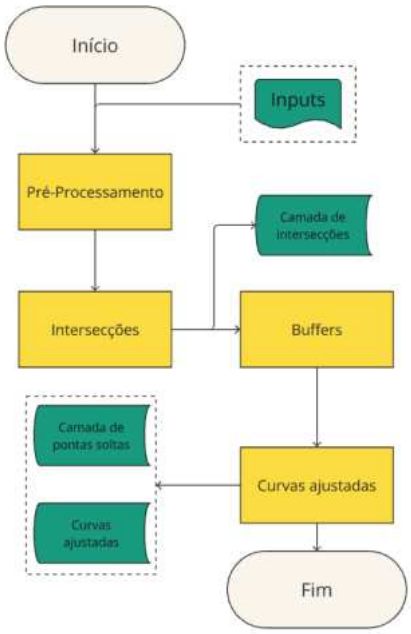


Figura 11 – Workflow com a lógica atual do código implementado

A visão geral do processo se encontra no workflow da Figura 11, no qual a lógica do código é dividida em quatro etapas, que serão detalhadas em fluxos posteriormente citados. No panorama macro, a estrutura foi dividida nas etapas de pré-processamento, identificação de intersecções, aplicação dos buffers e ajuste das curvas.

Na Figura 12 tem-se um exemplo de caso em que há curvas a serem adequadas.

Figura 12 – Curvas de nível com intersecção em massa d'água.

3.1.2 Pré-processamento

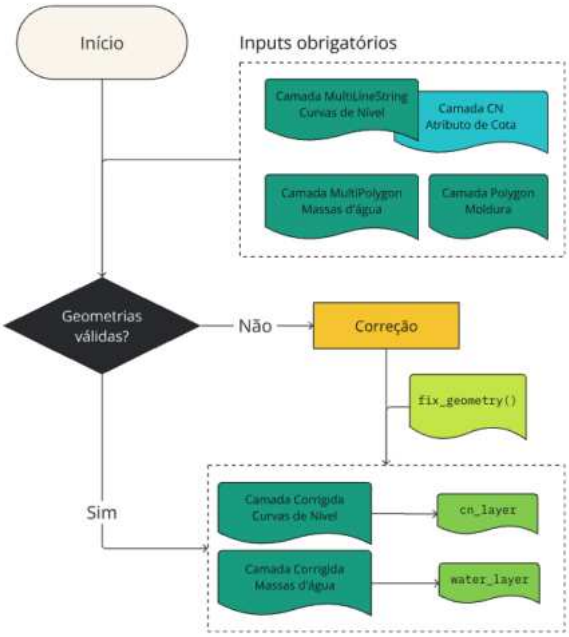


Figura 13 – Workflow da etapa de pré-processamento

Nessa etapa, conforme ilustrado na Figura 13, as camadas são fornecidas pelo usuário na tela apresentada na Figura 14. A partir daí, essas camadas passam por uma conferência de erros de geometria, e, caso for detectada alguma anomalia, são feitas as devidas correções.

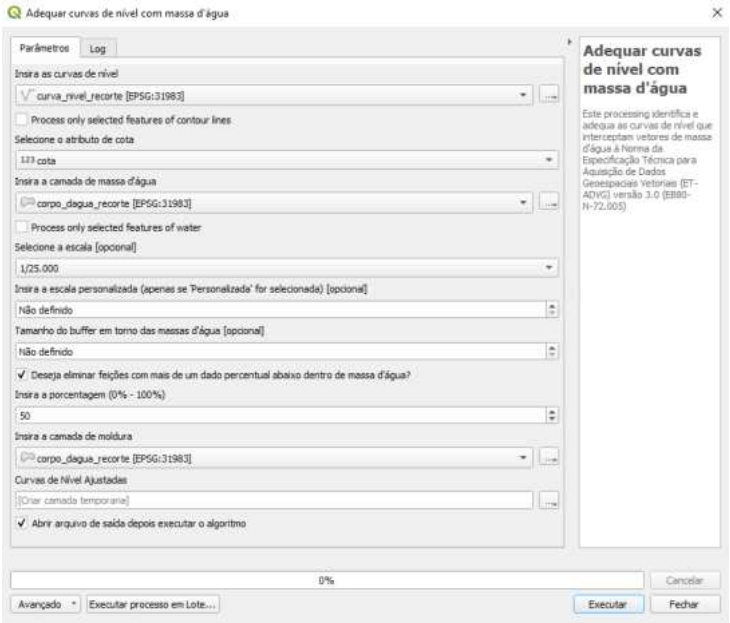


Figura 14 – Interface do *processing* de adequação de curvas de nível às massas d'água.

3.1.3 Identificação de intersecções

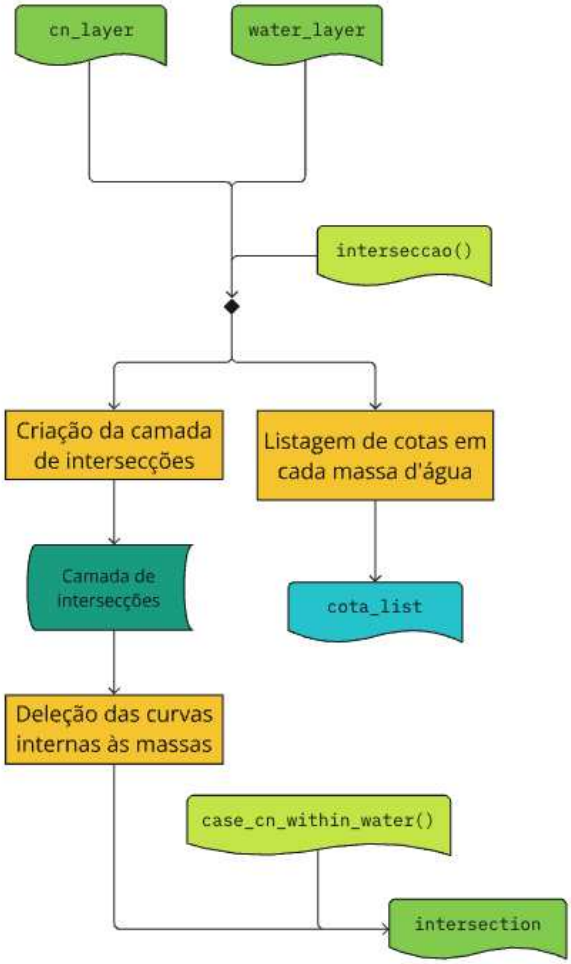


Figura 15 – Workflow da etapa de identificação de intersecções

Após o tratamento das camadas de input, é criada uma camada com as intersecções entre as curvas de nível e as massas d'água, assim como, para cada massa d'água, uma lista com todas as cotas a ela intersectantes, conforme explicitado na Figura 15. Na Figura 16, tem-se um exemplo de camada de flags das intersecções.

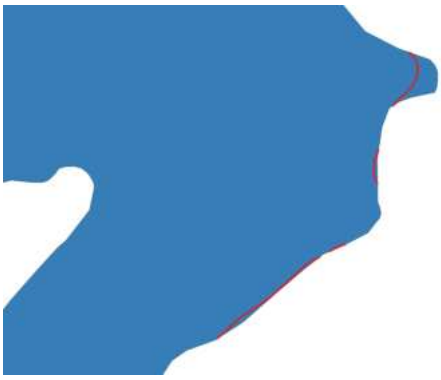


Figura 16 – Vetor de flags das intersecções.

3.1.4 Aplicação dos buffers

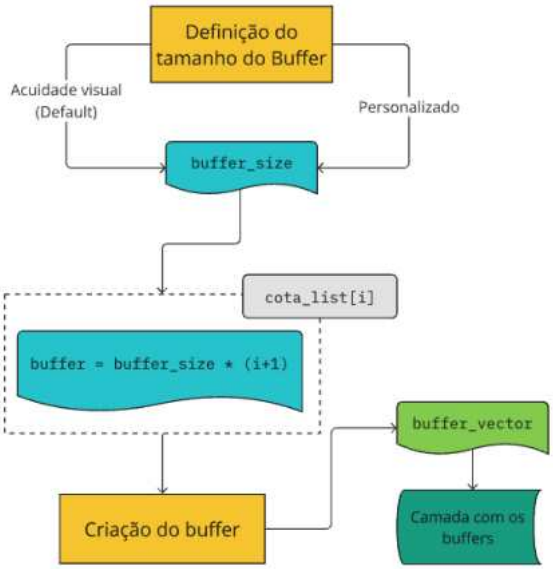


Figura 17 – Workflow da etapa de aplicação dos buffers

A partir do input do usuário a respeito do tamanho do buffer, define-se a base de cálculo para cada caso, conforme a Figura 17. Para tal, usa-se a lista gerada no passo anterior como referencial, e, para cada posição na ordenação crescente de cotas, é definido um buffer múltiplo da base de cálculo. Um exemplo de camada de buffer se encontra na Figura 18.

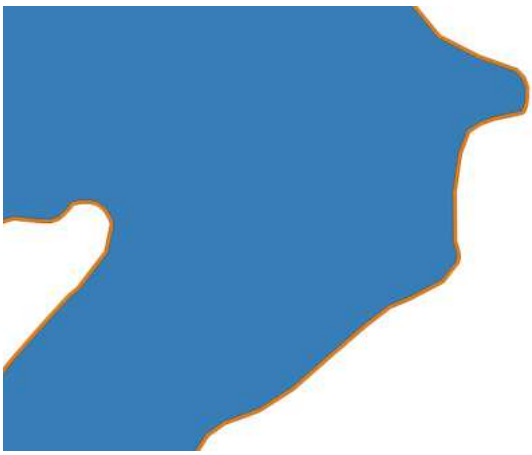


Figura 18 – Criação do buffer em torno da massa d'água.

3.1.5 Ajuste das curvas

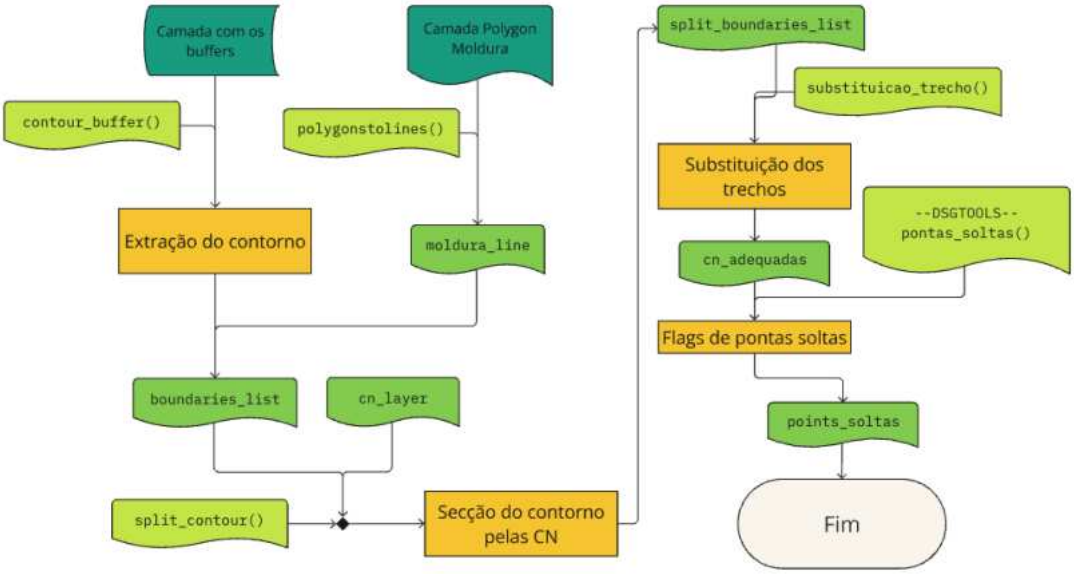


Figura 19 – Workflow da etapa de ajuste das curvas

Por fim, a Figura 19 mostra a etapa final do código. Nela, é extraído o contorno da camada com os buffers gerados no passo anterior (Figura 20), e, juntamente com a camada de moldura, gera-se uma camada com os limites dos buffers (Figura 21).

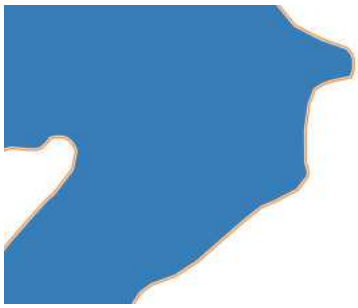


Figura 20 – Extração da linha de fronteira do buffer.

Figura 21 – Secção das linhas de fronteira com as curvas de nível.

Inserindo também as camadas de curva de nível, pode-se gerar os trechos de intersecção das curvas com seus respectivos buffers associados (Figura 22), e, por fim, substitui-se as intersecções pelo trecho de buffer (Figura 23).

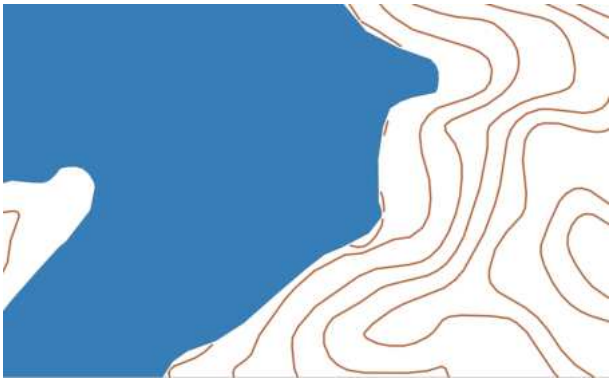


Figura 22 – Recorte das curvas de nível que sobrepõem o buffer.

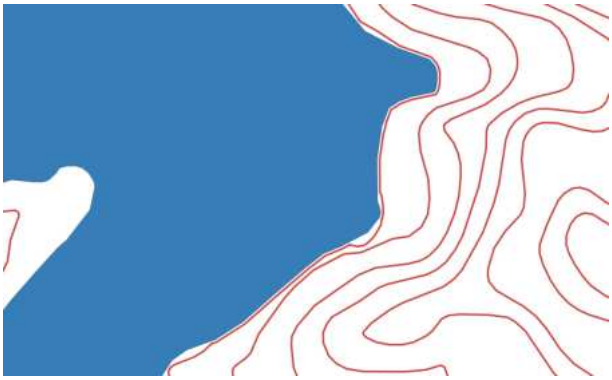


Figura 23 – Inserção dos trechos da linha de buffer nas curvas de nível faltantes.

Por fim, é gerado também uma camada com as pontas soltas das curvas adequadas através do algoritmo de identificação de pontas soltas, conforme exemplificado na Figura 24 (DSG, 2021). Esse código foi modificado para atender às necessidades específicas do processo de intersecção com hidrografia, mantendo-se a lógica principal para detectar segmentos de linha desconectados. Reconhece-se a contribuição do DSGTools na concepção original desta funcionalidade, destacando-se que a adaptação foi realizada com base nesse trabalho.

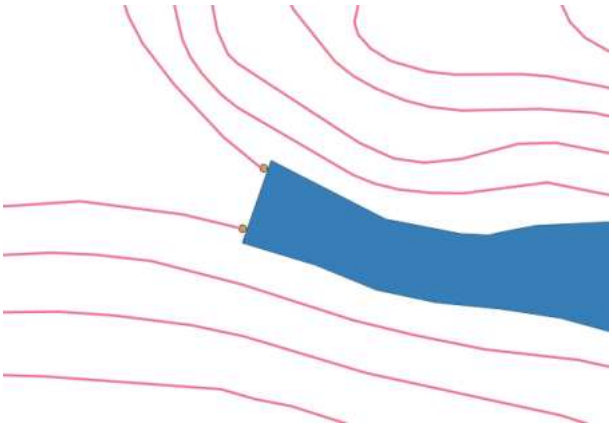


Figura 24 – Vetor de flag de pontas soltas.

3.2 Materiais

Para implementar a metodologia proposta e analisar os resultados, serão utilizados os seguintes materiais:

3.2.1 Micro-computador

Foi utilizado o micro-computador com as seguintes especificações:

- Processador Intel$^{(R)}$ Core^{TM} i7-3632QM @ 4.00 Ghz,
- RAM instalada de 8,00 GB,
- Disco rígido de 1TB,
- Placa gráfica NVIDIA GeForce,
- Sistema operacional Linux,

3.2.2 Modelos Digitais de Terreno

Os MDT utilizados possuem as seguintes especificações:

- Fonte: *Forest And Buildings removed Copernicus DEM* (FABDEM),
- Resolução Espacial: 30m,
- Resolução Radiométrica: 16 bits,
- Datum horizontal: WGS84,
- Datum vertical: REALT18,
- Formato do arquivo: GeoTiff,
- Ano de aquisição: 2021.

3.2.3 Vetores de Curvas de Nível

Os vetores de curvas de nível possuem as seguintes especificações:

- Fonte: Banco de Dados Geográficos do Exército (BDGEx),
- Datum: SIRGAS2000,
- Tipo de arquivos: Geopackage (GPKG).

3.2.4 Vetor de Massas d'água

Os vetores de massas d'água possuem as seguintes especificações:

- Fonte: Banco de Dados Geográficos do Exército (BDGEx),
- Datum: SIRGAS2000,
- Tipo de arquivos: Geopackage (GPKG).

3.2.5 *Softwares*

- Linguagem Python versão 3.11.8, IDE Visual Studio Code versão 1.88, Open-Source licença MIT (Massachusetts Institute of Technology license);
- QGIS versão 3.28.4 Firenze.

A infraestrutura necessária para o desenvolvimento do projeto de pesquisa é disponibilizada pela Seção de Engenharia Cartográfica do IME e de seus laboratórios instalados na SE/6.

4 RESULTADOS E DISCUSSÕES

Neste capítulo, serão abordados os resultados finais do trabalho.

4.1 Saídas Obtidas

4.1.1 Uma curva intersectando em mesma margem

Na Figura 25, nota-se um caso em que a adequação correta de um caso de apenas uma curva de nível intersectando uma mesma margem de massa d'água.

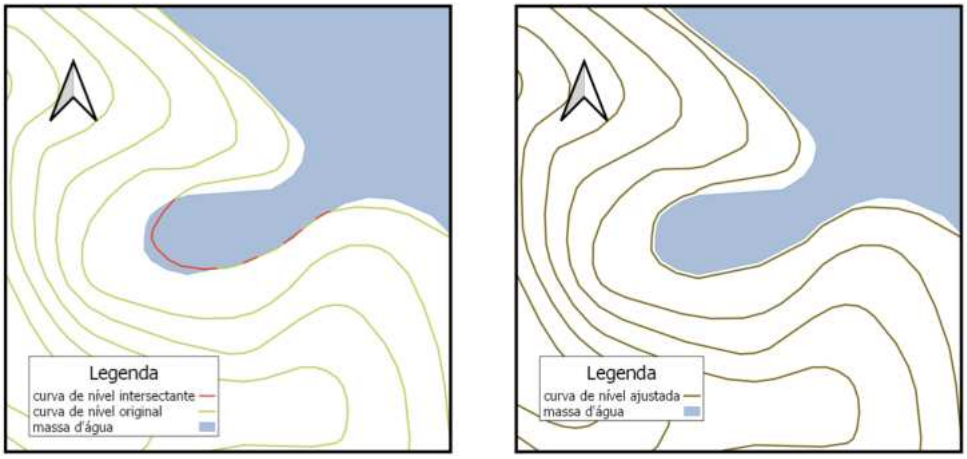


Figura 25 – Exemplo de correção bem-sucedida após a utilização do código

4.1.2 Mais de uma curva intersectando em mesma margem

Na Figura 26, nota-se o caso em que ocorreu uma adequação correta da situação em que mais de uma curva de nível intersectava uma mesma margem de massa d'água.

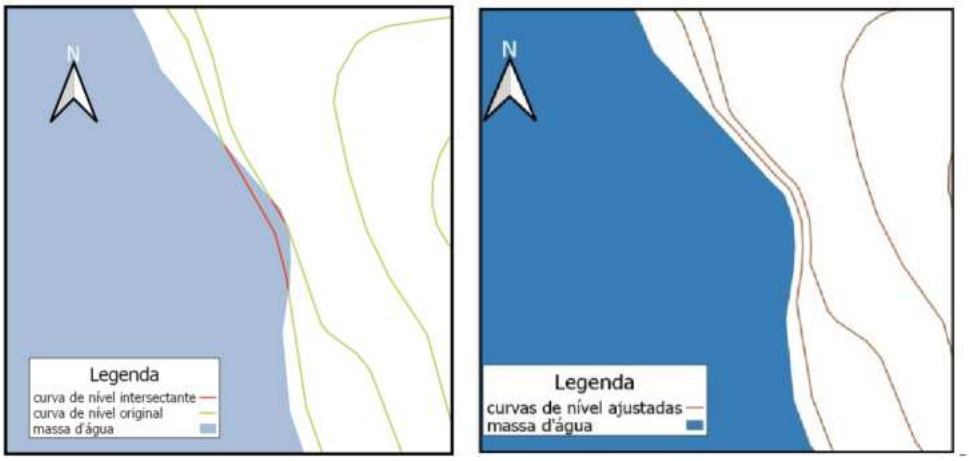
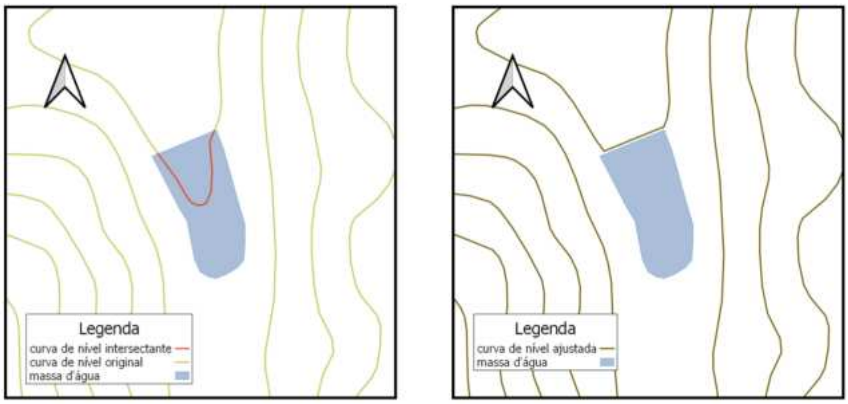


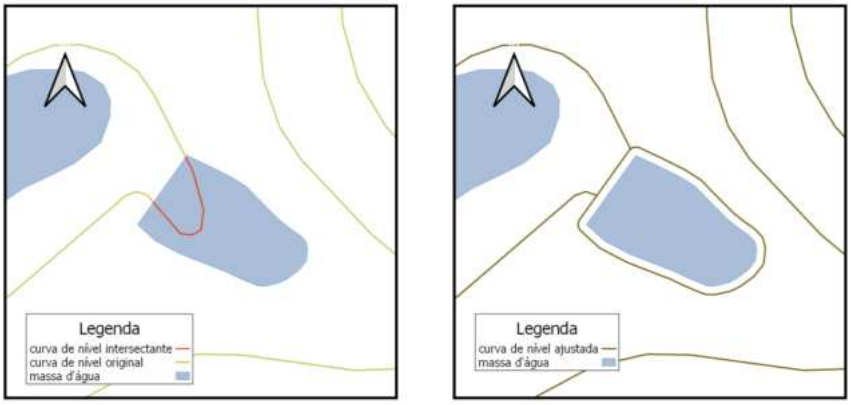
Figura 26 – Exemplo de correção após a utilização do código

4.1.3 Barragens, lagos, diques

Percebe-se na Figura 27a que houve uma adequação válida, ao passo que a Figura 27b não teve o mesmo sucesso.



(a) Exemplo de correção bem-sucedida após a utilização do código



(b) Exemplo de correção mal-sucedida após a utilização do código

4.1.4 Curva que cruza rios

No caso da Figura 28 tem-se uma situação em que a adequação não ocorreu conforme o esperado em norma.

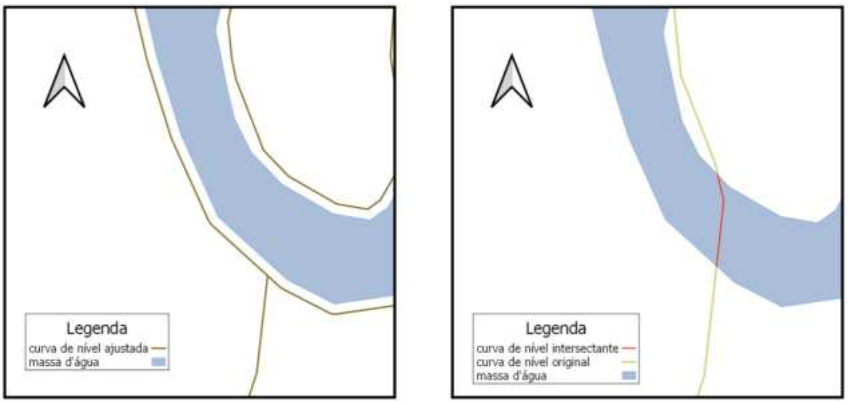


Figura 28 – Exemplo de correção mal-sucedida após a utilização do código

4.1.5 Curva dentro de massa d'água

Para tal classificação, foi observado um caso em que a adequação foi bem-sucedida (Figura 29).

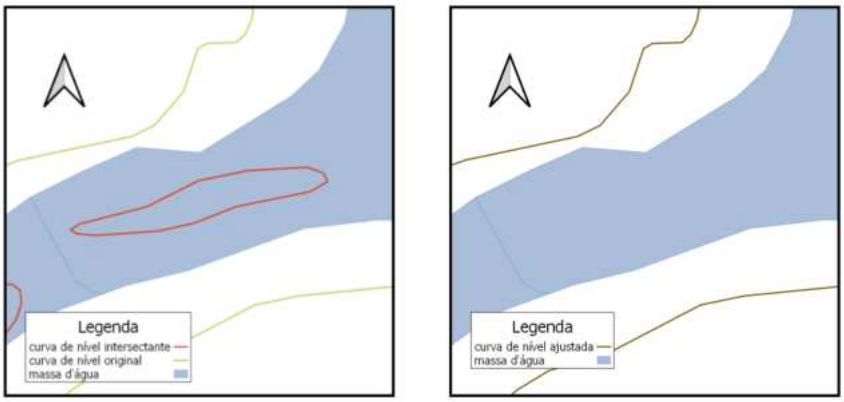


Figura 29 – Exemplo de correção bem-sucedida após a utilização do código

4.2 Regiões analisadas

Para testar o *plugin* foram analisadas quatro áreas em território brasileiro (Figura 30). Visando a pluralidade de áreas geográficas, foram realizados testes sobre a área de Alegrete - RS, com bioma de Pampas, predominantemente plana, a área litorânea da costa pernambucana, a área em região de Floresta Amazônica representada pela cidade de Boa Vista - RR, e na região Sudeste, predominantemente de Mata Atlântica e característica montanhosa representada pela região em torno da cidade de Varginha - MG. A seguir, serão abordados com detalhes os resultados obtidos em cada região e as diferentes fontes de insumos utilizados para as curvas de nível e as massas d'água.

Figura 30 – Distribuição espacial das regiões analisadas

4.2.1 Casos de interseção de curvas de nível com massas d'água - Minas Gerais

Avaliando a região em torno das cidades de Varginha – MG e Três Corações - MG com uma área de aproximadamente 26 mil km² e na escala de 1:25000, a cobertura de massas d'água é de 248 km², representando cerca de 1% da área analisada, conforme observa-se na Figura 31. Foram utilizados como insumos, as curvas de nível e as massas d'água fornecidas pelo 5º CGeo.

Figura 31 – Área analisada sobre a região de Varginha - MG

Foram observadas 235 intersecções entre curvas de nível e os polígonos de massas d'água, distribuídas nos seguintes casos:

Casos de Interseção	**Número de Casos**	**Porcentagem de Sucesso**
1 CN intersectante na mesma margem	69	100%
Mais de 1 CN intersectando a mesma margem	10	100%
Barragens, lagos, diques	111	86%
CN que cruza rios	4	0%
CN dentro de massa d'água	41	100%
Total	235	92%

Tabela 1 – Casos de interseção de curvas de nível com massas d'água em Minas Gerais, com porcentagem de sucesso.

4.2.2 Casos de interseção de curvas de nível com massas d'água - Rio Grande do Sul

Avaliando a região próxima à cidade de Alegrete-RS, com uma área de aproximadamente 68 km² e na escala de 1:50000, a cobertura de massas d'água é de 2,4 km², representando cerca de 3,5% da área analisada, conforme a Figura 32. Os insumos utilizados foram as curvas de nível e as massas d'água fornecidos pelo 1ºCGeo.

Figura 32 – Área analisada sobre a região de Alegrete - RS

Foram observadas 46 intersecções entre curvas de nível e massas d'água, divididas nos seguintes casos:

Casos de Interseção	**Número de Casos**	**Porcentagem de Sucesso**
1 CN intersectante na mesma margem	31	100%
Mais de 1 CN intersectando a mesma margem	0	0%
Barragens, lagos, diques	0	0%
CN que cruza rios	6	0%
CN dentro de massa d'água	9	100%
Total	46	87%

Tabela 2 – Casos de interseção de curvas de nível com massas d'água no Rio Grande do Sul, com porcentagem de sucesso.

4.2.3 Casos de interseção de curvas de nível com massas d'água - Pernambuco

Avaliando a região no litoral entre Olinda – PE e Porto de Galinhas – PE com uma área de aproximadamente 442 km² e na escala de 1:25000, a cobertura de massas d'água é de 2,25 km², excluindo-se o Oceano Atlântico, representando cerca de 0,5% da área analisada, conforme a Figura 33. As massas d'água foram fornecidas pelo 3ºCGeo, porém, as curvas de nível utilizadas foram obtidas via MDT do FABDEM.

Figura 33 – Área analisada sobre a região do litoral de Pernambuco

Foram observadas 164 intersecções entre curvas de nível e massas d'água, divididas nos seguintes casos:

Casos de Interseção	**Número de Casos**	**Porcentagem de Sucesso**
1 CN intersectante na mesma margem	83	100%
Mais de 1 CN intersectando a mesma margem	22	100%
Barragens, lagos, diques	22	61%
CN que cruza rios	16	0%
CN dentro de massa d'água	21	100%
Total	164	87%

Tabela 3 – Casos de interseção de curvas de nível com massas d'água em Pernambuco, com porcentagem de sucesso.

4.2.4 Casos de interseção de curvas de nível com massas d'água - Roraima

Avaliando a região em torno da cidade de Boa Vista-RR, com uma área de aproximadamente 775 km² e na escala de 1:25000, a cobertura de massas d'água é de 38 km², representando cerca de 4,9% da área analisada, conforme a Figura 34. As curvas de nível foram fornecidas pelo 4ºCGeo, porém, as massas d'água utilizadas foram obtidas via BDGEx.

Figura 34 – Área analisada sobre a região de Boa Vista - RR

Foram observadas 59 intersecções entre curvas de nível e polígonos de massas d'água, distribuídas nos seguintes casos:

Casos de Interseção	**Número de Casos**	**Porcentagem de Sucesso**
1 CN intersectante na mesma margem	26	100%
Mais de 1 CN intersectando a mesma margem	0	0%
Barragens, lagos, diques	2	0%
CN que cruza rios	0	0%
CN dentro de massa d'água	31	100%
Total	59	97%

Tabela 4 – Casos de interseção de curvas de nível com massas d'água em Roraima, com porcentagem de sucesso.

A tabela abaixo apresenta uma comparação entre diferentes regiões, considerando a área terrestre e a área de massas d'água, além de métricas de processamento relacionadas

ao tempo de execução, intersecções de curvas de nível com massas d'água e a quantidade de pontas soltas encontradas.

Região	Área terrestre	Área de massa d'água	Processamento	Intersecções	Pontas soltas	Fonte de CN	Fonte de hidrografia
Alegrete - RS	68km^2	2,4km^2	173s	46	0	1CGEO	1CGEO
Litoral de PE	442km^2	2,25km^2	17,3s	178	89	MDE FABDEM	3CGEO
Boa Vista - RR	774km^2	38,4km^2	5,56s	59	2	4CGEO	BDGEx
Varginha - MG	26mil km^2	24,8km^2	291s	235	56	5CGEO	5CGEO

Tabela 5 – Comparação de resultados entre diferentes regiões, em termos de área e intersecções de curvas de nível com massas d'água.

5 CONCLUSÃO

O presente trabalho teve como objetivo a correção de intersecções entre curvas de nível e massas d'água, alcançando significativos avanços na automatização de processos que antes eram realizados manualmente por operadores. O plugin desenvolvido para o QGIS foi bem-sucedido em resolver 100% dos casos de uma curva de nível intersectando a mesma margem, mais de uma curva de nível intersectando a mesma margem e curvas de nível dentro de massas d'água. Esses resultados demonstram um excelente desempenho na correção de intersecções comuns.

Por outro lado, alguns desafios permaneceram sem solução completa. Casos como intersecções em barragens, lagos e diques apresentaram uma taxa de sucesso que variou entre 0% e 86%, dependendo da região, enquanto as curvas de nível que cruzam rios não tiveram nenhuma correção bem-sucedida, com uma taxa de 0% de sucesso em todos os cenários analisados.

O tempo de processamento variou de acordo com a área de estudo, o sistema de coordenadas utilizado e a complexidade das intersecções encontradas. A região de Alegrete - RS, por exemplo, foi a única que utilizou o Sistema Geográfico de Coordenadas, o que gerou um tempo de processamento de 173 segundos. Esse aumento no tempo se deve ao fato de que, ao utilizar um sistema geográfico, o código precisa realizar cálculos adicionais para converter as coordenadas em graus para seus equivalentes em metros, o que torna o processamento mais lento. Já as demais regiões, como Boa Vista - RR e Litoral de PE, utilizam o Sistema Projetado de Coordenadas, que não requer essa conversão e, portanto, permite um processamento mais rápido.

Além disso, ao comparar a região sul de Boa Vista - RR com o Litoral de Pernambuco, observou-se que, apesar de Boa Vista possuir uma área maior (774 km^2 contra 442 km^2), o número de intersecções foi significativamente menor (59 em Boa Vista contra 178 em Pernambuco), o que levou a um tempo de processamento mais curto em Boa Vista (5,56 segundos), independentemente de sua maior extensão territorial.

A metodologia aplicada foi dividida em três etapas principais: identificação das intersecções, adequação às normas estabelecidas e resolução de casos específicos. Esse processo garantiu avanços significativos na correção automática, embora ainda restem oportunidades de aprimoramento, principalmente nos casos mais complexos. A continuidade das investigações e o refinamento do plugin poderão aumentar ainda mais sua eficácia e aplicabilidade em diversos cenários geoespaciais futuros.

REFERÊNCIAS

CHAGAS, C. S.; FILHO, E. I. F.; ROCHA, M. F.; JúNIOR, W. d. C.; NETO, N. C. S. Avaliação de modelos digitais de elevação para aplicação em um mapeamento digital de solos. *Revista Brasileira de Engenharia Agrícola e Ambiental*, Unidade Acadêmica de Engenharia Agrícola, v. 14, n. 2, p. 218–226, Feb 2010. ISSN 1415-4366. Disponível em: <https://doi.org/10.1590/S1415-43662010000200014>.

CHANG, H.-C.; GE, L.; RIZOS, C.; MILNE, T. Validation of dems derived from radar interferometry, airborne laser scanning and photogrammetry by using gps-rtk. In: IEEE. *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*. [S.l.], 2004. v. 5, p. 2815–2818.

DSG, D. do Serviço Geográfico do E. *Manual de Uso do Complemento DSGTools para o QGIS*. Brasília, Brasil, 2021.

GRASER, A.; MEARNS, B.; MANDEL, A.; FERRERO, V. O.; BRUY, A. *QGIS: Becoming a GIS power user*. [S.l.]: Packt Publishing Ltd, 2017.

GUTH, P. L.; NIEKERK, A. V.; GROHMANN, C. H.; MULLER, J.-P.; HAWKER, L.; FLORINSKY, I. V.; GESCH, D.; REUTER, H. I.; HERRERA-CRUZ, V.; RIAZANOFF, S. et al. Digital elevation models: Terminology and definitions. *Remote Sensing*, Multidisciplinary Digital Publishing Institute, v. 13, n. 18, p. 3581, 2021.

HASSELMANN, K.; RANEY, R.; PLANT, W.; ALPERS, W.; SHUCHMAN, R.; LYZENGA, D. R.; RUFENACH, C.; TUCKER, M. Theory of synthetic aperture radar ocean imaging: A marsen view. *Journal of Geophysical Research: Oceans*, Wiley Online Library, v. 90, n. C3, p. 4659–4686, 1985.

HAWKER, L.; UHE, P.; PAULO, L.; SOSA, J.; SAVAGE, J.; SAMPSON, C.; NEAL, J. A 30 m global map of elevation with forests and buildings removed. *Environmental Research Letters*, IOP Publishing, v. 17, n. 2, p. 024016, feb 2022. Disponível em: <https://dx.doi.org/10.1088/1748-9326/ac4d4f>.

LAWHEAD, J. *QGIS Python Programming Cookbook*. [S.l.]: Packt Publishing Ltd, 2017.

MORSE, S. P. Concepts of use in contour map processing. *Communications of the ACM*, ACM New York, NY, USA, v. 12, n. 3, p. 147–152, 1969.

TELFORD, W. M.; GELDART, L. P.; SHERIFF, R. E. *Applied geophysics*. [S.l.]: Cambridge university press, 1990.

TOUTIN, T.; GRAY, L. State-of-the-art of elevation extraction from satellite sar data. *ISPRS Journal of Photogrammetry and Remote Sensing*, Elsevier, v. 55, n. 1, p. 13–33, 2000.

TREVETT, J. W. *Imaging radar for resources surveys*. [S.l.]: Springer Science & Business Media, 2013.

ANEXO A – CÓDIGO-FONTE DO COMPLEMENTO

```
# -*- coding: utf-8 -*-
"""
/***************************************************************************
 CorrecaoCurvaNivel
                                 A QGIS plugin
 Este plugin identifica e corrige linhas de curva de nível que
intersectam vetores de massa d'água.
 Generated by Plugin Builder: http://g-sherman.github.io/Qgis-Plugin-
Builder/
                              -------------------
        begin                : 2023-11-21
        copyright            : (C) 2023 by Cap Tadeu; 1° Ten Kreitlon; 1°
Ten Vinicius
        email                : e.tadeu.eb@ime.eb.br;
joao.pereira@ime.eb.br; viniciusmagalhaes@ime.eb.br

***************************************************************************
**/

/***************************************************************************
 *
*
 *   This program is free software; you can redistribute it and/or modify
*
 *   it under the terms of the GNU General Public License as published by
*
 *   the Free Software Foundation; either version 2 of the License, or
*
 *   (at your option) any later version.
*
 *
*

***************************************************************************
**/
"""

__author__ = 'Cap Tadeu; 1° Ten Kreitlon; 1° Ten Vinicius'
__date__ = '2023-11-21'
__copyright__ = '(C) 2023 by Cap Tadeu; 1° Ten Kreitlon; 1° Ten Vinicius'

# This will get replaced with a git SHA1 when you do a git archive

__revision__ = '$Format:%H$'

import os
from code import interact
from qgis.PyQt.QtCore import QCoreApplication
from qgis.PyQt.QtGui import QIcon
from qgis.core import (QgsProcessing,
                       QgsFeature,
                       QgsFeatureSink,
                       QgsField,
                       QgsFields,
                       QgsGeometry,
```

```
                        QgsPointXY,
                        QgsProcessingAlgorithm,
                        QgsProcessingParameterBoolean,
                        QgsProcessingParameterVectorLayer,
                        QgsProcessingParameterFeatureSink,
                        QgsProcessingParameterField,
                        QgsProcessingParameterEnum,
                        QgsProcessingParameterNumber,
                        QgsProcessingException,
                        QgsProject,
                        QgsVectorLayer,
                        QgsWkbTypes,
                        QgsProcessingMultiStepFeedback,
                        QgsProcessingFeatureSourceDefinition,
                        QgsProcessingFeedback,
                        QgsFeatureRequest,
                        QgsSpatialIndex)
from qgis.gui import QgsCollapsibleGroupBox
import processing
from .pontas_soltas import pontas_soltas
from math import cos, radians

class CorrecaoCurvaNivelAlgorithm(QgsProcessingAlgorithm):

    INPUT_VECTOR = 'INPUT_VECTOR'
    INPUT_FIELD =  'INPUT_FIELD'
    INPUT_AGUA = 'INPUT_AGUA'
    INPUT_SCALE = 'INPUT_SCALE'
    CUSTOM_SCALE = 'CUSTOM_SCALE'
    BUFFER_SIZE = 'BUFFER_SIZE'
    OUTPUT = 'OUTPUT'
    SELECTED_CURVES = 'SELECTED_CURVES'
    SELECTED_WATER = 'SELECTED_WATER'
    MOLDURA = 'MOLDURA'
    REMOVE_FEATURES = 'REMOVE_FEATURES'
    PERCENTAGE = 'PERCENTAGE'

    def initAlgorithm(self, config=None):
        self.addParameter(
            QgsProcessingParameterVectorLayer(
                self.INPUT_VECTOR,
                self.tr("Insira as curvas de nível"),
                [QgsProcessing.TypeVectorLine]
            )
        )
        self.addParameter(
            QgsProcessingParameterBoolean(
                self.SELECTED_CURVES, self.tr("Process only selected
features of contour lines")
            )
        )
        self.addParameter(
            QgsProcessingParameterField(
                'INPUT_FIELD',
                self.tr('Selecione o atributo de cota'),
                type=QgsProcessingParameterField.Numeric,
                parentLayerParameterName='INPUT_VECTOR',
                defaultValue = 'cota')
```

```
            )
        self.addParameter(
            QgsProcessingParameterVectorLayer(
                self.INPUT_AGUA,
                self.tr("Insira a camada de massa d'água"),
                types=[QgsProcessing.TypeVectorPolygon]
            )
        )
        self.addParameter(
            QgsProcessingParameterBoolean(
                self.SELECTED_WATER, self.tr("Process only selected
features of water")
            )
        )

        # Parâmetro de seleção para a escala
        self.addParameter(
            QgsProcessingParameterEnum(
                self.INPUT_SCALE,
                self.tr("Selecione a escala"),
                options=['1/25.000', '1/50.000', '1/100.000',
'1/250.000', 'Personalizada'],
                defaultValue=0,
                optional=True  # Opção de entrada opcional
            )
        )

        self.addParameter(
            QgsProcessingParameterNumber(
                self.CUSTOM_SCALE,
                self.tr("Insira a escala personalizada (apenas se
'Personalizada' for selecionada)"),
                optional=True
            )
        )
        self.addParameter(
            QgsProcessingParameterNumber(
                self.BUFFER_SIZE,
                self.tr("Tamanho do buffer em torno das massas d'água"),
                optional=True
            )
        )

        self.addParameter(
            QgsProcessingParameterBoolean(
                self.REMOVE_FEATURES,
                self.tr(f"Deseja eliminar feições com mais de um dado
percentual abaixo dentro de massa d'água?"),
                defaultValue=True
            )
        )

        self.addParameter(
            QgsProcessingParameterNumber(
                self.PERCENTAGE,
                self.tr(f"Insira a porcentagem (0% - 100%)"),
                type=QgsProcessingParameterNumber.Integer,
                defaultValue=50,
```

```
                minValue=0,
                maxValue=100,
            )
        )

        self.addParameter(
            QgsProcessingParameterVectorLayer(
                self.MOLDURA,
                self.tr("Insira a camada de moldura"),
                types=[QgsProcessing.TypeVectorPolygon]
            )
        )
        self.addParameter(
            QgsProcessingParameterFeatureSink(
                self.OUTPUT,
                self.tr('Curvas de Nível Ajustadas')
            )
        )

    def processAlgorithm(self, parameters, context, feedback):
        """
        Executa o processamento principal do algoritmo.
        """
        curvas = self.parameterAsVectorLayer(parameters,
self.INPUT_VECTOR, context)
        cota_field = self.parameterAsFields(parameters,'INPUT_FIELD',
context )
        massas = self.parameterAsVectorLayer(parameters, self.INPUT_AGUA,
context)
        scale_option = self.parameterAsEnum(parameters, self.INPUT_SCALE,
context)
        custom_scale = self.parameterAsDouble(parameters,
self.CUSTOM_SCALE, context)
        buffer_size = self.parameterAsDouble(parameters,
self.BUFFER_SIZE, context)
        onlySelectedCN = self.parameterAsBool(parameters,
self.SELECTED_CURVES, context)
        onlySelectedWater = self.parameterAsBool(parameters,
self.SELECTED_WATER, context)
        moldura = self.parameterAsVectorLayer(parameters, self.MOLDURA,
context)
        remove_features = self.parameterAsBool(parameters,
self.REMOVE_FEATURES, context)
        percentage = self.parameterAsInt(parameters, self.PERCENTAGE,
context)

        # Criação de uma camada de CN somente com as feições selecionadas
de CN
        if onlySelectedCN == False: inputLayerCN = curvas
        else:
            inputLayerCN = self.layer_features_selected(curvas)

        # Criação de uma camada de água somente com as feições
selecionadas de água
        if onlySelectedWater == False: inputLayerWater = massas
        else:
            inputLayerWater = self.layer_features_selected(massas)
```

```
        # A variável scale refere-se ao denominador de escala
        if scale_option == 4:  # Personalizada
            if custom_scale <= 0:
                raise QgsProcessingException(self.tr("A escala
personalizada deve ser um valor positivo."))
            scale = custom_scale
        else:
            scales = [25000, 50000, 100000, 250000]
            scale = scales[scale_option]

        (sink, dest_id) = self.parameterAsSink(parameters,
                                               self.OUTPUT,
                                                context,
                                                curvas.fields(),
                                                QgsWkbTypes.LineString,
                                                curvas.sourceCrs())

        feedback.setProgressText('Procurando e corrigindo geometrias
inválidas nas camadas de vetores...')
        # Correção de geometrias inválidas nas camadas de vetores
        curves = self.fix_geometry(inputLayerCN)
        water = self.fix_geometry(inputLayerWater)

        total = 100.0 / water.featureCount() if water.featureCount() else
0

        moldura_line = processing.run("native:polygonstolines",
                                {'INPUT': moldura,
                                'OUTPUT': 'TEMPORARY_OUTPUT'})['OUTPUT']

        # 1) Obtenção da intersecção das curvas de nível com as massas
d'água
        feedback.setProgressText('\nProcurando e identificando as
intersecções entre as camadas de vetores...')
        intersection = self.interseccao(parameters, context, curves,
water)

        # 2) Eliminação de feições dentro de massas d'água
        feedback.setProgressText('\nDeletando curvas de nível dentro de
massas d\'água...')
        curves = self.case_cn_within_water(curves, water,
remove_features, percentage)

        cn_adequadas = curves

        for current, massa_feat in enumerate(water.getFeatures()):

            if feedback.isCanceled():
                break

            massa_geom = massa_feat.geometry()
            cotas_list = list()

            for line in intersection.getFeatures():
                geom_inter = line.geometry()

                if geom_inter.intersects(massa_geom):
```

```
                    cota = line[cota_field[0]] #Aqui ele pega a string
'cota'
                    cotas_list.append(cota)
                    feedback.pushInfo(f'A feição {line} possui a cota
{cota} e está dentro da massa {massa_feat}.')

            cotas_list = sorted(set(cotas_list)) #Elementos únicos (sem
cotas duplicadas) e crescentemente ordenadas

            if len(cotas_list) > 0:

                # 3) Inserção de n camada de buffer em torno da massa
d'água
                feedback.setProgressText(f'\nAplicando {len(cotas_list)}
buffers ao redor da massa d\'água {massa_feat.id()}...')
                buffers_list = self.list_buffers(parameters, context,
buffer_size, curvas, massas, massa_feat, massa_geom, cotas_list, scale)

                # 4) Criação de uma camada de linhas no contorno do
buffer
                feedback.setProgressText(f'\nExtraindo o contorno dos
buffers da massa d\'água {massa_feat.id()}...')
                boundaries_list = self.contour_buffer(parameters,
context, buffers_list, moldura)
                feedback.pushInfo(f'A massa {massa_feat} possui
{len(boundaries_list)} contorno(s) de buffer com as seguintes camadas
{boundaries_list}.')

                # 5) Secção da linha de contorno
                feedback.setProgressText(f'\nSeccionando o contorno dos
buffers da massa d\'água {massa_feat.id()} com curvas de nível...')
                split_boundaries_list = self.split_contour(parameters,
context, feedback, boundaries_list, curves, moldura_line, cotas_list,
cota_field[0])
                feedback.pushInfo(f'A massa {massa_feat} possui
{len(split_boundaries_list)} contorno(s) de buffer seccionados pelas CN
com as seguintes camadas {split_boundaries_list}.')
                #O ordenamento na lista, segue a lógica de ligar o
primeiro elemento com a menor cota

                #feedback.pushInfo(f'\nO alcance da lista de buffer é
{len(buffers_list)} de cota é {len(cotas_list)} e da split_boundaries
{len(split_boundaries_list)}.')
                for b in range (0,len(buffers_list)):
                    elevacao = cotas_list[b]
                    # 6) Criação das CN cortada pelo buffer
                    feedback.setProgressText(f'\nCortando as curvas de
nível que intersectam o buffer da massa d\'água {massa_feat.id()}...')
                    buffer = buffers_list[b]

                    #feedback.pushInfo(f'\n\nA camada de CN a ser
cortadas pelo buffer é {cn_adequadas}.')
                    cn_cortadas = self.cut_cn(parameters, context,
feedback, cn_adequadas, buffer, cota_field[0], elevacao)
                    feedback.pushInfo(f'O recorte das CN {cn_adequadas}
pelo buffer {buffers_list[b]} de cota {cotas_list[b]}.')

                    #cn_cortadas = self.fix_geometry(cn_cortadas)
```

```
                    #feedback.pushInfo(f'\n\nA camada de CN cortadas é
{cn_cortadas}.')

                    # 7) Substituição de trechos das curvas de nível
                    feedback.setProgressText(f"\nIniciando a conexão das
curvas de nível que foram cortadas...")
                    trechos_substitutos = split_boundaries_list[b]
                    cn_adequadas = self.substituicao_trecho(parameters,
context, feedback, trechos_substitutos, cn_cortadas, cota_field[0],
elevacao)

                    #cn_adequadas = self.fix_geometry(cn_adequadas)
                    #feedback.pushInfo(f'\n\nA camada de CN adequadas é
{cn_adequadas}.')

            feedback.setProgress(int(current * total))

        points_soltas = pontas_soltas(self, parameters, context,
feedback, cn_adequadas, moldura)
        # Adicionar a nova camada ao projeto
        QgsProject.instance().addMapLayer(points_soltas)

        # Adição das feições na camada output
        for feat in cn_adequadas.getFeatures():
            sink.addFeature(feat, QgsFeatureSink.FastInsert)

        return {self.OUTPUT: dest_id}

    def interseccao(self, parameter, context, cn_layer, water_layer):

        intersection = processing.run("native:intersection",
                                      {'INPUT':cn_layer,
                                       'OVERLAY':water_layer,

'OUTPUT':'TEMPORARY_OUTPUT'})['OUTPUT']

        intersection = processing.run("native:multiparttosingleparts",
                                      {'INPUT': intersection,

'OUTPUT':'TEMPORARY_OUTPUT'})['OUTPUT']

        intersection.setName("Intersecções")

        QgsProject.instance().addMapLayer(intersection)

        return intersection

    def list_buffers(self, parameters, context, buffer_tamanho, cn_layer,
water_layer, water_feat, water_geom, cotas_lista, escala):

        buffers = list()
        if buffer_tamanho:

            #Caso seja um buffer personalizado, ele verifica se o sistema
é geográfico e converte no equivalente a graus
            if cn_layer.crs().isGeographic():
                extent = cn_layer.extent()
```

```
                centroid_lat = (extent.yMinimum() + extent.yMaximum()) /
2 #Pega a latitude do centroide do projeto
                buffer_size_graus = buffer_tamanho / (111320 *
cos(radians(centroid_lat)))

                for i in range (0, len(cotas_lista)):
                    buffer = water_geom.buffer(buffer_size_graus*(i+1),
5)
                    crs = water_layer.sourceCrs()
                    buffer_vector =
QgsVectorLayer(f"{QgsWkbTypes.displayString(water_layer.wkbType())}?crs={
crs.authid()}", "buffer", "memory")

buffer_vector.dataProvider().addAttributes(water_layer.fields())
                    buffer_vector.updateFields()
                    buffer_feat = QgsFeature()
                    buffer_feat.setGeometry(buffer)
                    buffer_feat.setAttributes(water_feat.attributes())

buffer_vector.dataProvider().addFeatures([buffer_feat])
                    buffer_vector.updateExtents()

                    #QgsProject.instance().addMapLayer(buffer_vector)
                    buffers.append(buffer_vector) #Lista de buffers em
camada


            else:
                for i in range (0, len(cotas_lista)):
                    buffer = water_geom.buffer(buffer_tamanho*(i+1), 5)
                    crs = water_layer.sourceCrs()
                    buffer_vector =
QgsVectorLayer(f"{QgsWkbTypes.displayString(water_layer.wkbType())}?crs={
crs.authid()}", "buffer", "memory")

buffer_vector.dataProvider().addAttributes(water_layer.fields())
                    buffer_vector.updateFields()
                    buffer_feat = QgsFeature()
                    buffer_feat.setGeometry(buffer)
                    buffer_feat.setAttributes(water_feat.attributes())

buffer_vector.dataProvider().addFeatures([buffer_feat])
                    buffer_vector.updateExtents()
                    #QgsProject.instance().addMapLayer(buffer_vector)
                    buffers.append(buffer_vector) #Lista de buffers em
camada

        else:
            lim_aquidade = 0.0002 # O limite da aquidade visual é de
0,2mm (0,0002m)
            if cn_layer.crs().isGeographic():
                extent = cn_layer.extent()
                centroid_lat = (extent.yMinimum() + extent.yMaximum()) /
2 #Pega a latitude do centroide do projeto
                buffer_size_meters = lim_aquidade * escala
                buffer_size = buffer_size_meters / (111320 *
cos(radians(centroid_lat)))
```

```
                for i in range (0, len(cotas_lista)):
                # Aplicar buffer nas coordenadas geográficas (latitude e
longitude)
                    buffer = water_geom.buffer(buffer_size*(i+1), 5)
                    crs = water_layer.sourceCrs()
                    buffer_vector =
QgsVectorLayer(f"{QgsWkbTypes.displayString(water_layer.wkbType())}?crs={
crs.authid()}", "buffer", "memory")

buffer_vector.dataProvider().addAttributes(water_layer.fields())
                    buffer_vector.updateFields()
                    buffer_feat = QgsFeature()
                    buffer_feat.setGeometry(buffer)
                    buffer_feat.setAttributes(water_feat.attributes())

buffer_vector.dataProvider().addFeatures([buffer_feat])
                    buffer_vector.updateExtents()
                    #QgsProject.instance().addMapLayer(buffer_vector)
                    buffers.append(buffer_vector) #Lista de buffers em
camada

            else:

                buffer_size = lim_aquidade * escala
                for i in range (0, len(cotas_lista)):
                    buffer = water_geom.buffer(buffer_size*(i+1), 5)
                    crs = water_layer.sourceCrs()
                    buffer_vector =
QgsVectorLayer(f"{QgsWkbTypes.displayString(water_layer.wkbType())}?crs={
crs.authid()}", "buffer", "memory")

buffer_vector.dataProvider().addAttributes(water_layer.fields())
                    buffer_vector.updateFields()
                    buffer_feat = QgsFeature()
                    buffer_feat.setGeometry(buffer)
                    buffer_feat.setAttributes(water_feat.attributes())

buffer_vector.dataProvider().addFeatures([buffer_feat])
                    buffer_vector.updateExtents()
                    #QgsProject.instance().addMapLayer(buffer_vector)
                    buffers.append(buffer_vector) #Lista de buffers em
camada

        return buffers

    def contour_buffer(self, parameters, context, buffers_lista, mold):

        boundaries = list()

        for buffer in buffers_lista:
            boundary = processing.run("native:polygonstolines",
                                      {'INPUT': buffer,
                                      'OUTPUT':
'TEMPORARY_OUTPUT'})['OUTPUT']

            boundary = processing.run("native:clip",
                                          {'INPUT':boundary,
                                          'OVERLAY':mold,
```

```
'OUTPUT':'TEMPORARY_OUTPUT'})['OUTPUT']
            #QgsProject.instance().addMapLayer(boundary)
            boundaries.append(boundary)

        return boundaries

    def split_contour(self, parameters, context, feedback, boundaries,
cn_layer, mold_line, cotas_lista, cota_field):

        split_boundaries = list()
        for f in range (0, len(boundaries)):
            # Filtrar feições da moldura_line com base no atributo
desejado (exemplo: 'category' == 1)
            atributo = cota_field #nome do atributo
            valor_cota = cotas_lista[f] #A lista de cotas crescente está
par a par com o buffer

            # Criar uma nova camada temporária com as feições filtradas
            features_filtradas = [feat for feat in cn_layer.getFeatures()
if feat[atributo] == valor_cota]

            crs = cn_layer.sourceCrs()
            mem_layer =
QgsVectorLayer(f"{QgsWkbTypes.displayString(cn_layer.wkbType())}?crs={crs
.authid()}", "feições_selecionadas", "memory")
            mem_layer.dataProvider().addAttributes(cn_layer.fields())
            mem_layer.updateFields()
            mem_layer.dataProvider().addFeatures(features_filtradas)
            mem_layer.updateExtents()

            # Executar o algoritmo de divisão com a camada filtrada
            split_boundary = processing.run("native:splitwithlines",
                                            {'INPUT': boundaries[f],
                                            'LINES': mem_layer,
                                            'OUTPUT':
'TEMPORARY_OUTPUT'})['OUTPUT']

            split_boundary = processing.run("native:splitwithlines",
                                            {'INPUT': split_boundary,
                                            'LINES': mold_line,
                                            'OUTPUT':
'TEMPORARY_OUTPUT'})['OUTPUT'] #Secção da linha pela moldura

            # Adicionar a camada de contorno seccionada ao projeto
            #QgsProject.instance().addMapLayer(split_boundary)
            split_boundaries.append(split_boundary)

        return split_boundaries

    def cut_cn(self, parameters, context, feedback, cn, buffer_layer,
cota_field, cota):

        # Atributo e valor da cota a ser filtrado
        atributo = cota_field #atributo 'cota'
        valor_cota = cota #valor de cota filtrado
```

```
        # Criar uma nova camada de memória para armazenar as feições
modificadas
        crs = cn.sourceCrs()
        mem_layer =
QgsVectorLayer(f"{QgsWkbTypes.displayString(cn.wkbType())}?crs={crs.authi
d()}", f"CN_cortada_na_cota_{valor_cota}", "memory")
        mem_layer.dataProvider().addAttributes(cn.fields())
        mem_layer.updateFields()

        trechos = list()
        # Iterar sobre as feições da camada de linhas
        for line_feat in cn.getFeatures():
            line_geom = line_feat.geometry()

            # Verificar se a feição atende ao critério de cota
            if line_feat[atributo] == valor_cota:
                # Aplicação da diferença (difference) com cada geometria
da camada de buffer
                for buffer_feat in buffer_layer.getFeatures():
                    buffer_geom = buffer_feat.geometry()

                    # Aplicar a diferença
                    diff_geom = line_geom.difference(buffer_geom)

                    # Criação de uma nova feição para armazenar a
geometria resultante
                    new_feature = QgsFeature()
                    new_feature.setGeometry(diff_geom)
                    new_feature.setAttributes(line_feat.attributes())  #
Manter os atributos da linha original

                    # Adicionar a feição resultante à camada de memória
                    trechos.append(new_feature)

            else:
                # Se não for a feição com cota desejada, adicionar a
feição original
                trechos.append(line_feat)

        # Atualizar a extensão da nova camada
        mem_layer.dataProvider().addFeatures(trechos)
        mem_layer.updateExtents()

        cn = processing.run("native:multiparttosingleparts",
                        {'INPUT': mem_layer,
                        'OUTPUT':'TEMPORARY_OUTPUT'})['OUTPUT']

        # Adicionar a nova camada ao projeto
        #QgsProject.instance().addMapLayer(cn)

        return cn

    def substituicao_trecho(self, parameters, context, feedback,
split_boundary, cn_cortadas_layer, cota_field, cota):

        trechos_feat = list()
        for line_con in split_boundary.getFeatures():
            line_con_geom = line_con.geometry()
```

```
            line_con_geom_buffer = line_con.geometry().buffer(1, 5)
            bbox = line_con_geom_buffer.boundingBox()

            for line in cn_cortadas_layer.getFeatures(bbox):
                line_geom = line.geometry()

                if line_con_geom_buffer.intersects(line_geom):

                    for line_2 in cn_cortadas_layer.getFeatures(bbox):
                        line_2_geom = line_2.geometry()

                        if line_con_geom_buffer.intersects(line_2_geom):
                            if line.id() != line_2.id() and line.id() >
line_2.id() and line[cota_field] == cota and line_2[cota_field] == cota:
                                #feedback.pushInfo(f'A linha de conexão
{line_con} conecta as linhas {line.id()} de cota {line[cota_field]} e
{line_2.id()} de cota {line_2[cota_field]}.')
                                new_feature =
QgsFeature(cn_cortadas_layer.fields())
                                new_feature.setGeometry(line_con_geom)

new_feature.setAttributes(line.attributes())
                                trechos_feat.append(new_feature)

        cn_cortadas_layer.dataProvider().addFeatures(trechos_feat)
        cn_cortadas_layer.updateExtents()
        #QgsProject.instance().addMapLayer(cn_cortadas_layer)

        return cn_cortadas_layer

    def layer_features_selected(self, layer):

        crs = layer.sourceCrs()
        outputLayer =
QgsVectorLayer(f"{QgsWkbTypes.displayString(layer.wkbType())}?crs={crs.au
thid()}",
                                    "feições_selecionadas",
                                    "memory")
        outputLayer.dataProvider().addAttributes(layer.fields())
        outputLayer.updateFields()
        outputFeat = layer.selectedFeatures()
        outputLayer.dataProvider().addFeatures(outputFeat)
        outputLayer.updateExtents()

        return outputLayer

    def fix_geometry (self, layer):

        outputLayer = processing.run("native:fixgeometries",
                                     {'INPUT':layer,

'OUTPUT':'TEMPORARY_OUTPUT'})['OUTPUT']

        return outputLayer

    def case_cn_within_water(self, layer_cn, layer_water,
remove_features, percentage):
        layer_cn.startEditing()
```

```
        for linha_feature in layer_cn.getFeatures():
            geom_linha = linha_feature.geometry()
            comprimento_total = geom_linha.length()
            bbox = geom_linha.boundingBox()

            comprimento_interseccao = 0
            for poligono_feature in layer_water.getFeatures(bbox):
                geom_poligono = poligono_feature.geometry()
                interseccao = geom_linha.intersection(geom_poligono)

                if interseccao:
                    comprimento_interseccao += interseccao.length()

            # Verificação se a condição de eliminação é atendida
            if remove_features and comprimento_interseccao > (percentage
/ 100 * comprimento_total):
                layer_cn.deleteFeature(linha_feature.id())

        layer_cn.commitChanges()

        return layer_cn


    def name(self):
        """
        Returns the algorithm name, used for identifying the algorithm.
This
        string should be fixed for the algorithm, and must not be
localised.
        The name should be unique within each provider. Names should
contain
        lowercase alphanumeric characters only and no spaces or other
        formatting characters.
        """
        return "Adequar curvas de nível com massa d'água"

    def displayName(self):
        """
        Returns the translated algorithm name, which should be used for
any
        user-visible display of the algorithm name.
        """
        return self.tr(self.name())

    def group(self):
        """
        Returns the name of the group this algorithm belongs to. This
string
        should be localised.
        """
        return self.tr("Adequar curvas de nível com massa d'água")

    def tr(self, string):
        return QCoreApplication.translate('Processing', string)

    def createInstance(self):
        return CorrecaoCurvaNivelAlgorithm()
```

```
    def shortHelpString(self):
        return self.tr("Este processing identifica e adequa as curvas de
nível que interceptam vetores de massa d'água à Norma da Especificação
Técnica para Aquisição de Dados Geoespaciais Vetoriais (ET-ADVG) versão
3.0 (EB80-N-72.005)")

    def icon(self):
        return QIcon(os.path.join(os.path.dirname(__file__),
'cn_agua.png'))
```