



Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Instituto de Matemática e Estatística

Marcel Emanuelli Rotunno

**Toolbox Adaplin para QGIS: uma ferramenta de programação
dinâmica para a extração semiautomática de estradas**

Rio de Janeiro

2017

Marcel Emanuelli Rotunno

**Toolbox Adaplin para QGIS: uma ferramenta de programação
dinâmica para a extração semiautomática de estradas**

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Ciências Computacionais pelo Instituto de Matemática e Estatística da Universidade do Estado do Rio de Janeiro.

Orientador: Prof. Dr. Guilherme Lucio Abelha Mota

Rio de Janeiro

2017

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIRUS / BIBLIOTECA CTC-A

Rotunno, Marcel Emanuelli

Toolbox Adaplin para QGIS: uma ferramenta de programação dinâmica para a extração semiautomática de estradas / Marcel Emanuelli Rotunno - Rio de Janeiro, 2017.

97 f.

Orientador: Guilherme Lucio Abelha Mota.

Dissertação (Mestrado em ciências computacionais) - Universidade do Estado do Rio de Janeiro. Instituto de Matemática e Estatística.

1. Extração de estradas - Teses.
 2. Programação dinâmica - Teses.
 3. Métodos semiautomáticos - Teses.
 4. Sensoriamento remoto - Teses.
- I. Mota, Guilherme Lucio Abelha. II. Universidade do Estado do Rio de Janeiro. Instituto de Matemática e Estatística. III. Mestrado.

CDU 681.3.06

Autorizo, apenas para fins acadêmicos e científicos, a reprodução total ou parcial desta dissertação, desde que citada a fonte.

Assinatura

Data

Marcel Emanuelli Rotunno

**Toolbox Adaplin para QGIS: uma ferramenta de programação dinâmica para
a extração semiautomática de estradas**

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre pelo Programa de Pós-Graduação em Ciências Computacionais da Universidade do Estado do Rio de Janeiro.

Aprovado em 16 de Março de 2017.

Banca Examinadora:

Prof. Dr. Guilherme Lucio Abelha Mota (Orientador)
Instituto de Matemática e Estatística – UERJ

Prof. Dr. Gilson Alexandre Ostwald Pedro da Costa
Instituto de Matemática e Estatística - UERJ

Prof. Dr. Cláudio João Barreto dos Santos
Departamento de Engenharia Cartográfica - UERJ

Prof. Dr. Jorge Luís Nunes e Silva Brito
Departamento de Engenharia Cartográfica - UERJ

Prof. Dr. Otávio da Fonseca Martins Gomes
Centro de Tecnologia Mineral - CETEM/MCTIC

Rio de Janeiro
2017

DEDICATÓRIA

Dedico esse trabalho à minha esposa Luciana, que me acompanhou em todos momentos necessários para o cumprimento do projeto, aos meus pais, Otto e Luciah, que sempre me incentivaram e apoaram, e a meu irmão e amigo Renan, que foi constantemente solícito em ajudar.

AGRADECIMENTOS

Agradeço a meu orientador Guilherme Lucio Abelha Mota, que esteve sempre disponível para orientar, melhorar e estimular o desenvolvimento, aperfeiçoamento e conclusão do trabalho.

Aos amigos e colegas de mestrado, que me ajudaram e apoiaram no decurso desta trajetória.

Aos meus colegas de trabalho do IBGE. À Coordenação de Cartografia, nas pessoas de Patrícia Vida e Marcelo Maranhão, que muito se interessaram por este trabalho. A Darlan Nunes e Alexandre Teixeira, com quem pude conversar, aprender e trocar informação e conhecimento a respeito da dissertação. Aos meus colegas de sala Eduardo Porto Abrahão, Jamil Diuana, Viviane Diniz, Graciosa Rainha, Gerson Franca. A Paulo Santos, por sua ajuda e contribuição. A Rinaldo Menezes, por sua solicitude ao executar os experimentos da dissertação.

A todos os membros da banca da dissertação, que aperfeiçoaram o documento e ajudaram na geração da sua versão final.

RESUMO

ROTUNNO, Marcel Emanuelli. *Toolbox Adaplin para QGIS: uma ferramenta de programação dinâmica para a extração semiautomática de estradas.* 2017. 97 f. Dissertação (Mestrado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2017.

A produção de dados geográficos e a sintetização deste conhecimento em mapas são importantíssimas nos dias de hoje. Em particular, a obtenção e a atualização da base cartográfica de estradas são essenciais e possuem aplicações como cadastro urbano e planejamento dos sistemas de transporte, além do apoio para o planejamento da localização de todos estabelecimentos comerciais e residenciais no espaço geográfico. Uma estrada normalmente é caracterizada no mapa pelo seu eixo central. O procedimento mais rápido e economicamente viável para sua obtenção é o traçado do vetor usando como base imagens aéreas ou orbitais. A vantagem desta abordagem reside no fato de praticamente prescindir de trabalho de campo. Procede-se seu delineamento, normalmente de forma manual, através das ferramentas de criação e edição de vetores em ambiente de Sistemas de Informações Geográficas – SIG. Nesse contexto, o presente trabalho propõe uma ferramenta SIG semiautomática e interativa de criação de vetores que otimiza o trabalho do operador, possibilitando um menor esforço e consequentemente uma melhor produtividade em seu trabalho. Esta ferramenta foi implementada como uma extensão (*toolbox*) para o software livre QGIS. A ferramenta *toolbox* Adaplin, a cada segmento fornecido pelo usuário, agrupa um par de pontos intermediários que transformam o segmento numa polilinha. Modela-se uma função objetivo a fim de maximizar o enquadramento da polilinha à estrada presente na imagem. O procedimento de otimização modela os pontos a partir de um grafo e produz a solução por programação dinâmica. Para ajudar o operador na escolha do próximo ponto, a ferramenta gera uma previsão em tempo real do traçado da polilinha. Experimentos abrangearam a comparação da restituição manual com a realizada através da ferramenta. Foi constituída para os experimentos uma base de dados composta por 100 recortes com dimensões 4×4 km (800×800 px) obtidos de imagens orbitais Rapideye, de média resolução (distância entre pixels de 5 metros). Os recortes foram divididos em 5 classes, cada classe com 20 recortes, conforme o material de revestimento e o entorno da estrada. Para as classes de estradas pavimentadas e em leito natural cercadas por vegetação rasteira ou vegetação rasteira e floresta os experimentos mostraram uma redução do tempo de restituição que varia entre 10% e 21% e uma redução do número de cliques do mouse que varia entre 48% e 65%, sendo os erros posicionais mantidos em um nível aceitável para o mapeamento em escala 1:50.000. A ferramenta não apresentou bons resultados para a classe de estrada pavimentada cercada por solo exposto, apesar da redução do número de cliques necessários para a vetorização da estrada. A aplicação do mesmo algoritmo, porém usando a banda do infravermelho, é vislumbrada para a extração de trechos de drenagem correspondentes a fluxos de água de rios.

Palavras-chave: QGIS. Base de dados vetorial. Extensões do QGIS. Extração de Estradas.

ABSTRACT

ROTUNNO, Marcel Emanuelli. *Toolbox Adaplin for QGIS: a dynamic programming based toolbox for semi-automatic road extraction.* 2017. 97 f. Dissertação (Mestrado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2017.

The production of geographic data and the synthesis of this knowledge on maps are very important nowadays. In particular, obtaining and updating a cartographic road database are essential and have applications such as urban cadastre and planning of transport systems, support for the location of all commercial and residential establishments in the geographic space. A road normally is characterized on the map by its central axis. The fastest and most economical procedure is to trace the vector using an aerial or orbital image as the base. The advantage of this approach lies in the fact that virtually no field work is done. The vector is usually designed manually, through the creation and editing tools in an environment of Geographic Information Systems - GIS. In this context, the present work proposes a semi-automatic and interactive GIS tool for vector creation that optimizes the operator work allowing for less effort and consequently better productivity at job. This tool has been implemented as a toolbox for the open-source software QGIS. The Adaplin toolbox tool adds a pair of intermediate points for each user-supplied segment, transforming this segment into a polyline. For this task, an objective function is modeled to maximize the framing of the polyline to the road in the image. The optimization procedure models the points from a graph and produces the solution by dynamic programming. To help the operator select the next point, the tool generates a real-time preview of the polyline trace. Experiments covered the comparison of the manual restitution with the one performed through the tool. The experiments database was composed of 100 cutouts with dimensions 4×4 km (800 x 800 px) obtained from Rapideye orbital images, of medium resolution (distance between pixels of 5 meters). The cutouts were divided into 5 classes, each class with 20 cutouts, according to the coating material and the road surroundings. For the paved road and dirt road classes surrounded by undergrowth or undergrowth and forest, the experiments showed a reduction in restitution time ranging from 10% to 21% and a reduction in the number of mouse clicks ranging from 48% to 65%, with the positional errors being maintained at an acceptable level for mapping at 1:50,000 scale. The tool did not show good results for the paved road class surrounded by exposed soil, despite the reduction in the number of clicks needed to digitize the road. The application of the same algorithm, however using the infrared band, is glimpsed for the extraction of drainage stretches corresponding to rivers flows.

Keywords: QGIS. Vectorial Databases. QGIS extensions. Road extraction.

LISTA DE ILUSTRAÇÕES

Figura 1 - Segmentação por junção estatística de regiões	18
Figura 2 - Junção de objetos baseado em regras de conexão no método proposto por Dal-Poz et al. (2005)	18
Figura 3 - Método semiautomático proposto por Pandit (2009)	21
Figura 4 - Rastreador de caminho proposto por Baumgartner et al. (2002).	22
Figura 5 - Grafo de exemplo para a programação dinâmica	27
Figura 6 - Etapas do método proposto por Gruen e Li (1997)	33
Figura 7 - Grafo acíclico dirigido formado e (em vermelho) caminho ótimo de i até f	38
Figura 8 - Dois primeiros pontos inseridos pelo operador	40
Figura 9 - Grafo formado e caminho escolhido de a até b	41
Figura 10 - Diagrama de classes da classe AdaplinTool	42
Figura 11 - Exemplos das classes de recortes de imagens Rapideye presentes na base de dados	46
Figura 12 - Eixos de referência dos recortes de imagens Rapideye apresentados na Figura 11	48
Figura 13 - Resultados das extrações manual e semiautomáticas para o recorte r_5 : PAV-RAS	77
Figura 14 - Resultados das extrações manual e semiautomáticas para o recorte r_7 : PAV-FLO	78
Figura 15 - Resultados das extrações manual e semiautomáticas para o recorte r_{10} : NAT-RAS	79
Figura 16 - Resultados das extrações manual e semiautomáticas para o recorte r_3 : NAT-FLO	80
Figura 17 - Resultados das extrações manual e semiautomáticas para o recorte r_5 : PAV-SOLO	81

LISTA DE TABELAS

Tabela 1 - Disponibilidade de software e dados de teste	43
Tabela 2 - Informações técnicas sobre a constelação de satélites Rapideye	44
Tabela 3 - Descrição das classes de recortes de imagens Rapideye que compõem a base de dados	47
Tabela 4 - Quantidade de vértices da polilinha do eixo de referência por recorte .	49
Tabela 5 - Resultados para a vetorização manual da classe PAV-RAS	53
Tabela 6 - Resultados para a vetorização pelo Toolbox Adaplin da classe PAV-RAS	54
Tabela 7 - Método vitorioso em cada quesito analisado para a classe PAV-RAS .	55
Tabela 8 - Resultados para a vetorização manual da classe PAV-FLO	57
Tabela 9 - Resultados para a vetorização pelo Toolbox Adaplin da classe PAV-FLO	58
Tabela 10 - Método vitorioso em cada quesito analisado para a classe PAV-FLO .	59
Tabela 11 - Resultados para a vetorização manual da classe NAT-RAS	60
Tabela 12 - Resultados para a vetorização pelo Toolbox Adaplin da classe NAT-RAS	61
Tabela 13 - Método vitorioso em cada quesito analisado para a classe NAT-RAS .	62
Tabela 14 - Resultados para a vetorização manual da classe NAT-FLO	63
Tabela 15 - Resultados para a vetorização pelo Toolbox Adaplin da classe NAT-FLO	64
Tabela 16 - Método vitorioso em cada quesito analisado para a classe NAT-FLO .	66
Tabela 17 - Resultados para a vetorização manual da classe PAV-SOLO	67
Tabela 18 - Resultados para a vetorização pelo Toolbox Adaplin da classe PAV-SOLO	68
Tabela 19 - Método vitorioso em cada quesito analisado para a classe PAV-SOLO .	69
Tabela 20 - Resultados produzidos pelas ferramentas Erdas Easytrace e Toolbox Adaplin	70
Tabela 21 - Diferenças de parâmetros dos resultados produzidos pelas ferramentas Erdas Easytrace e Toolbox Adaplin	72
Tabela 22 - Diferenças entre médias dos parâmetros para as extração manual e via Toolbox Adaplin	75
Tabela 23 - Contagem do número de vitórias para cada classe na ordem Adaplin/-Manual	75

LISTA DE ABREVIATURAS E SIGLAS

ATOMI	<i>Automated reconstruction of Topographic Objects from aerial images using vectorized Map Information</i>
ATKIS	<i>Amtliche Topographisch-Kartographische Informationssystem</i>
ATM-R	<i>Adaptive Texture Matching for Road Extraction</i>
BKG	<i>Bundesamt für Kartographie und Geodäsie</i>
CAR	Cadastro Ambiental Rural
CCAR	Coordenação de Cartografia do IBGE
DSG	Diretoria de Serviço Geográfico do Exército
IBGE	Intituto Brasileiro de Geografia e Estatística
IPI	<i>Institut für Photogrammetrie und GeoInformation</i>
ISODATA	<i>Iterative Self Organizing Data Analysis Technique</i>
GIS	<i>Geographic Information System</i>
GNSS	<i>Global Navigation Satellite System</i>
GSD	<i>Ground Sample Distance</i>
MDT	Modelo Digital de Terreno
MMA	Ministério do Meio Ambiente
NAT-FLO	Classe estrada em leito natural cercada por vegetação rasteira e floresta
NAT-RAS	Classe estrada em leito natural cercada por vegetação rasteira
PAV-FLO	Classe estrada pavimentada cercada por vegetação rasteira e floresta
PAV-RAS	Classe estrada pavimentada cercada por vegetação rasteira
PAV-SOLO	Classe estrada pavimentada cercada por solo exposto
PEC	Padrão de Exatidão Cartográfica
SIG	Sistema de Informação Geográfica
SVM	<i>Support Vector Machine</i>
TNT	<i>Institut für Informationsverarbeitung</i>

SUMÁRIO

INTRODUÇÃO	13
1 REVISÃO BIBLIOGRÁFICA	17
1.1 Métodos automáticos	17
1.2 Métodos semiautomáticos	20
1.3 Atualização e controle da qualidade de bases cartográficas	22
2 FUNDAMENTAÇÃO TEÓRICA	25
2.1 Termos relacionados a resolução de imagens no sensoriamento remoto	25
2.2 Programação Dinâmica	25
2.2.1 Exemplo de otimização com programação dinâmica	26
2.3 A ferramenta de mapeamento QgsMapTool	28
3 MÉTODO	30
3.1 Método original de Gruen e Li	30
3.1.1 Função objetivo	33
3.2 Método empregado na ferramenta Adaplin	34
3.2.1 Modelo conceitual da estrada	35
3.2.2 Algoritmo de inserção e otimização de pontos	37
4 IMPLEMENTAÇÃO	39
4.1 A classe AdaplinTool	41
5 PROCEDIMENTO EXPERIMENTAL, RESULTADOS E ANÁLISE	43
5.1 Disponibilidade de imagens e software	43
5.2 Base de Dados	44
5.2.1 Informações sobre as imagens Rapideye	44
5.2.2 Recortes Rapideye utilizados	45
5.2.3 Dados de Referência	47
5.3 Parâmetros de análise da qualidade da extração das polilinhas . .	49
5.4 Experimentos realizados e resultados obtidos	50
5.4.1 Experimentos com os recortes da classe PAV-RAS	52
5.4.1.1 Método manual	52
5.4.1.2 Extração Toolbox Adaplin	53
5.4.1.3 Comparação dos resultados	54
5.4.2 Experimentos com a classe PAV-FLO	56
5.4.2.1 Extração manual	56
5.4.2.2 Extração Toolbox Adaplin	57
5.4.2.3 Comparação dos resultados	58

5.4.3	<u>Experimentos com a classe NAT-RAS</u>	60
5.4.3.1	Extração manual	60
5.4.3.2	Extração Toolbox Adaplin	61
5.4.3.3	Comparação dos resultados	61
5.4.4	<u>Experimentos com a classe NAT-FLO</u>	63
5.4.4.1	Extração manual	64
5.4.4.2	Extração Toolbox Adaplin	64
5.4.4.3	Comparação dos resultados	65
5.4.5	<u>Experimentos com a classe PAV-SOLO</u>	66
5.4.5.1	Extração manual	66
5.4.5.2	Extração Toolbox Adaplin	67
5.4.5.3	Comparação dos resultados	68
5.4.6	<u>Experimentos com a ferramenta Erdas Easytrace</u>	70
5.4.6.1	Comparação dos resultados	71
5.5	Discussão dos resultados	72
	CONCLUSÃO E SUGESTÕES PARA TRABALHOS FUTUROS	82
	REFERÊNCIAS	84
	APÊNDICE A - Código Python do Toolbox Adaplin	87

INTRODUÇÃO

Nas bases de dados cartográficos, normalmente, uma estrada é caracterizada pela polilinha correspondente ao seu eixo central. Uma das formas possíveis para a obtenção do eixo central é em campo, a partir do uso de equipamentos de topografia e geodésia, como receptores GNSS - *Global Navigation Satellite System* (Sistema de Navegação Global por Satélite, em português), estação total e teodolito. A extração dessas coordenadas em campo é extremamente precisa, contudo, a obtenção de uma base cartográfica completa por este meio demanda grande esforço, tempo e recursos financeiros. Uma forma mais economicamente viável para este tipo de mapeamento é seu delineamento com base em imagens de sensoriamento remoto ou mesmo ortoimagens. Estando tais imagens georreferenciadas, o eixo central é extraído intuitivamente a partir destas, reduzindo enormemente o tempo e os custos derivados da mobilização de equipes e do aluguel dos equipamentos, em relação ao que se faria em campo.

Historicamente, o traçado do vetor por sobre a imagem base é realizado de forma manual. Nesse contexto, um operador, normalmente um geógrafo ou cartógrafo, o faz através das ferramentas de criação e edição de vetores em um ambiente de Sistemas de Informações Geográficas (SIG). Como o método manual é lento e demanda bastante esforço, pesquisas científicas têm sido realizadas para propor métodos automáticos e semiautomáticos que melhorem o rendimento dos processos de extração de estradas.

Os métodos automáticos se caracterizam pela ausência de interferência humana no processo de extração da estrada. O operador é, portanto, passivo durante a extração das feições, resumindo seu trabalho à etapa de pós-edição quando é feita a correção dos possíveis erros produzidos pelo algoritmo. Por sua vez, nos métodos semiautomáticos, o operador provê informação de entrada a partir da qual o algoritmo realiza o traçado da estrada. Então, o operador tem participação direta no processo de extração, fator que tende a reduzir o volume de inconsistências. Comparando-se soluções automáticas e semiautomáticas, nota-se que a ordem de complexidade e o tamanho da entrada empregados na prática pelos métodos semiautomáticos são menores, possibilitando tempos de execução sensivelmente mais curtos.

A extração de estradas de forma plenamente automática é um ideal a ser alcançado, entretanto, há que se avaliar a possibilidade de implementação real de um método totalmente automático para que o esforço posterior de correção e edição da base cartográfica não seja maior que o de sua criação de modo inteiramente manual. Ao prescindir do auxílio humano, os métodos automáticos tendem a gerar resultados mais ruidosos. Assim sendo, para evitar o risco de inclusão de feições inconsistentes na base de dados, esta realidade prática impõe ao operador a realização de duas tarefas, a validação visual de cada uma das feições extraídas e a posterior correção das inconsistências na pós-edição.

Os métodos semiautomáticos, por tomarem proveito do conhecimento e da habilidade do operador, tendem, do ponto de vista prático, a apresentar resultados superiores aos produzidos pelos métodos automáticos. Esse fato se deve em certa medida à dificuldade intrínseca do reconhecimento de feições cartográficas em imagens, consequência da variedade na geometria e na resposta espectral, tarefa que, ainda assim, é executada com maestria pelo ser humano. Tal habilidade é sobretudo suportada pelo uso de conhecimento específico pelo profissional que opera o software restituidor. Assim, a ferramenta semiautomática pode conjugar a qualidade da visão humana em identificar os pontos a serem usados como sementes pelo algoritmo com a praticidade de se fazer a interpolação automática do traçado da estrada a medida que o resultado se adapta aos contornos expressos na imagem de sensoriamento remoto. Uma ferramenta semiautomática como essa diminui o número de pontos a serem indicados pelo operador, possibilitando, assim, a economia de esforço e consequentemente ganho de produtividade.

Por outro lado, realizar de forma inteiramente manual a restituição de estradas, apesar de fornecer resultados precisos, demanda grande concentração do operador e, por ser um processo lento, o esforço requerido se alonga no tempo. Assim, a medida que o desgaste físico e mental aumenta, o resultado da extração integralmente manual de feições cartográficas fica mais sujeito a erros. Além disso, os resultados obtidos estão atrelados à qualidade, ao treinamento, à experiência e ao nível de energia e de atenção do operador do restituidor.

Em termos práticos, o engenheiro cartógrafo precisa optar entre uma abordagem manual, automática ou semiautomática para método de extração de feições. Sendo assim, ele terá que fazer um balanço entre a facilidade e a qualidade dos resultados produzidos em cada um deles.

O Instituto Brasileiro de Geografia e Estatística (IBGE) e a Diretoria de Serviço Geográfico do Exército (DSG) são os principais órgãos de governo responsáveis pelo mapeamento cartográfico do Brasil. O IBGE, através da Coordenação de Cartografia (CCAR), possui inúmeros projetos de mapeamento do Brasil em escalas 1:1.000.000, 1:250.000 e 1:100.000. Estes projetos possuem como objetivo a formação de uma base cartográfica contínua do Brasil nessas escalas. O IBGE já disponibiliza a Base Cartográfica Contínua do Brasil em 1:1.000.000 e 1:250.000, sendo que uma nova versão de cada base é lançada de 2 em 2 anos, o que significa que estão constantemente em atualização. Em dezembro de 2016 foi lançada a primeira versão da base 1:100.000, que atualmente compreende apenas o estado de Goiás. Portanto, ainda falta o mapeamento dos demais estados para alcançar uma base nacional nesta escala.

A CCAR vem utilizando uma coleção de imagens obtidas pela constelação de satélites Rapideye para o mapeamento nessas escalas. Esta coleção de imagens ortorreferenciadas abrange todo o Brasil e foram adquiridas pelo Ministério do Meio Ambiente (MMA) visando o Cadastro Ambiental Rural (CAR), que realiza a regularização ambi-

ental de propriedades e posses rurais. O contrato entre o MMA e a empresa Santiago & Cintra Consultoria, distribuidora oficial das imagens Rapideye no Brasil, foi assinado em 2012 e prevê a disponibilização das imagens para uso de todos os órgãos públicos brasileiros. Até 2014 foram adquiridas três coleções de cobertura nacional, referentes aos anos de 2011, 2012 e 2013.

Para ilustrar a relevância no cenário nacional de ferramentas que possam auxiliar o processo de mapeamento, é preciso compreender que no Brasil ainda há algumas áreas que não possuem mapeamento oficial em escalas de maior detalhe. A existência de tais vazios é uma situação inconveniente e que persiste apesar do esforço e de projetos do IBGE e da DSG para mapeamento dessas áreas. Os chamados vazios cartográficos compreendem, por exemplo, a Amazônia, onde a vegetação é densa e, por conseguinte, imagens obtidas com base em energia eletromagnética no entorno do espectro visível não são capazes fornecer informações do terreno abaixo das copas das árvores e a textura da vegetação densa dificulta muitíssimo a identificação de pontos homólogos em imagens fotogramétricas. Nesse contexto, o uso da técnica de triangulação para a obtenção de dados tridimensionais através da fotogrametria não é a ideal para a aquisição de dados confiáveis.

A despeito das limitações anteriormente mencionadas, o panorama atual de geração e atualização das bases cartográficas do IBGE compreende a restituição manual de feições a partir de ortofotos e ortoimagens. Nesse contexto, pode-se perceber a insuficiência da mão de obra especializada disponível para fazer frente às tarefas de restituição e atualização das bases de dados no volume necessário para cobrir todo território brasileiro. Nesse mister, métodos que possam aumentar a produtividade dos operadores se fazem necessários. O método empregado nesta dissertação utiliza a técnica de programação dinâmica para obter o modelo geométrico do traçado da estrada entre os pontos sementes fornecidos. Ele possui como inspiração o método semiautomático baseado em programação dinâmica proposto por Gruen e Li (1997).

Objetivo do trabalho

O trabalho visa ao desenvolvimento de uma extensão (*toolbox*) para o software livre QGIS (QGIS Development Team, 2009) agregando a este uma ferramenta semiautomática de extração adaptativa e vetorial de estradas com base em imagens de sensoriamento remoto. A partir da marcação dos extremos de um segmento, a ferramenta, doravante chamada Toolbox Adaplin, deve gerar automaticamente o traçado do segmento vetorial, adaptando-o ao percurso da estrada em conformidade com o apresentado em uma imagem raster georreferenciada usada como entrada.

Estrutura e organização do texto

A dissertação está organizada em cinco capítulos. O capítulo 1 apresenta uma revisão bibliográfica dos métodos de extração de estradas e feições lineares. O capítulo 2 fornece os fundamentos teóricos necessários para a compreensão do método. O capítulo 3 apresenta o método semiautomático empregado para a extração de estradas, enquanto, o capítulo 4 descreve a implementação da ferramenta semiautomática de extração de estradas Toolbox Adaplin. O capítulo 5 destina-se à apresentação e à análise dos resultados experimentais. Por fim, o capítulo 6 fornece as conclusões e o apêndice A apresenta o código do Toolbox Adaplin para o QGIS, desenvolvido como parte da pesquisa apresentada na dissertação.

1 REVISÃO BIBLIOGRÁFICA

Normalmente, os métodos de extração de estradas seguem duas abordagens: automática e semiautomática. Na abordagem automática, a extração da estrada é feita sem o auxílio do operador, sendo feito um trabalho de pós-edição para corrigir as possíveis falhas. Por sua vez, na modalidade semiautomática o operador fornece informação de entrada para o algoritmo, como pontos ou regiões em uma imagem, e o algoritmo reconhece e delineia o segmento correspondente da estrada a partir dessa informação de forma autônoma.

Além dos métodos de extração, o capítulo descreve pesquisas relativas à atualização de bases cartográficas de estradas de forma automatizada ou semiautomatizada. A atualização da base cartográfica é feita baseando-se no conhecimento prévio sobre uma base já existente. Dessa forma, aproveitam-se as estradas presentes nessa base para adicionar e retirar as estradas que recentemente foram construídas ou removidas.

1.1 Métodos automáticos

Uma variedade de métodos automáticos foi proposta ao longo do tempo para extração de estradas. São descritos aqui alguns desses métodos a fim de proporcionar uma visão geral sobre as possibilidades de extração automática de estradas a partir de imagens.

O termo “segmentação” se refere, no âmbito do campo do processamento de imagens digitais, ao processo de divisão de uma imagem em grupos de pixels que guardam alguma semelhança entre si. Na prática, como resultado de uma segmentação para detecção de estradas possivelmente serão obtidos vários trechos descontínuos de estradas. Provavelmente, haverá muitos segmentos de estradas não conectados por causa de oclusões, o que atrapalha a tarefa de extração. Dessa forma, é necessário achar um meio de agrupar os diversos segmentos em uma só classe e conectá-los onde haja oclusões para que se possa extrair a malha viária.

Anil e Natarajan (2010) propõem a utilização da segmentação pela junção estatística de regiões para detecção das estradas. Este método busca detectar todos os objetos presentes na imagem, inclusive as estradas, e pertence à família de métodos baseados em crescimento de regiões. Ele foi proposto por Nock e Nielsen (2004) e pressupõe que cada imagem possui uma segmentação ótima, considerando a imagem como uma amostra dessa segmentação, e utiliza inferência estatística para alcançar tal segmentação. Posteriormente, uma limiarização é aplicada para identificação das regiões de estradas e, em seguida, um afinamento dessas regiões, com os resultados ilustrados pela Figura 1. O método usado para limiarização não é detalhado no artigo ora em questão.

Figura 1 - Segmentação por junção estatística de regiões

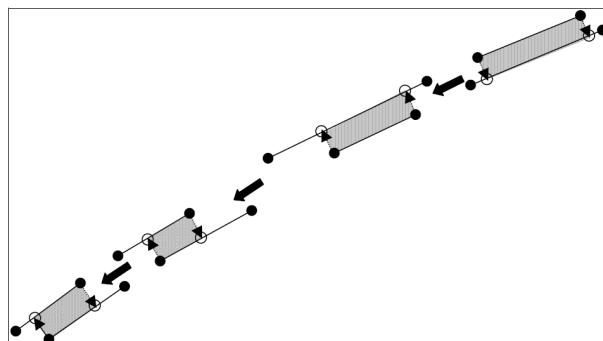


Legenda: (a) imagem original; (b) imagem segmentada.

Dal-Poz et al (2005), em vez de se basearem na segmentação por crescimento de regiões, propõem um método que funciona a partir da segmentação de contornos de Canny (1986). Após o resultado da segmentação, a vetorização é feita de forma a gerar vários polígonos. Os segmentos pertencentes a esses polígonos são comparados, caso dois segmentos próximos sejam paralelos, a região entre ambos seja homogênea e cada um tenha diferença significativa de contraste em relação ao seu redor, estes serão considerados parte de um objeto estrada. A junção dos segmentos de estrada para obtenção da estrada inteira se dá através de regras de conexão. A Figura 2 ilustra a situação da estrada sendo formada por um grupo de objetos, onde as setas indicam possíveis conexões entre os objetos a serem examinadas pelas regras de conexão.

De maneira geral, o termo classificação não-supervisionada se refere a uma classe de métodos que objetivam a divisão de um conjunto de objetos em classes, prescindindo de padrões de treinamento. Quando a classificação não-supervisionada é aplicada ao conjunto de pixels de uma imagem, esta é segmentada sem restringir os objetos espaciais

Figura 2 - Junção de objetos baseado em regras de conexão no método proposto por Dal-Poz et al. (2005)



a porções contíguas. Por exemplo, em uma imagem pode haver várias lagoas pertencentes a uma classe chamada lagoa, entretanto elas não estão conectadas.

Zhang (2006), por exemplo, propõe o uso de uma classificação não-supervisionada k-médias (MACQUEEN, 1967; STEINHAUS, 1956) com o valor empírico de 6 classes para detecção grosseira das estradas em uma imagem multiespectral. Das classes que consti-
tuem a imagem classificada, supõe-se que uma corresponde às estradas. Para a detecção automática dessa classe, calculam-se para todas as classes uma função de pertinência onde as bandas do vermelho, verde e azul têm um peso maior que a banda do infravermelho próximo. A classe que tiver o maior valor da função de pertinência é a correspondente à classe de estradas. Então, a imagem é binarizada, separando a classe de estradas das demais classes. Posteriormente, a classe estrada é refinada, seguida da decomposição dos segmentos lineares e da recomposição da rede de estradas.

O deslocamento médio (CHENG, 1995) é outra técnica de classificação não-supervisionada. A vantagem dela sobre o k-médias é que ela não possui parâmetros de entrada, sendo assim o usuário não precisa escolher o número de classes para divisão da imagem. A técnica dispõe da propriedade de preservar as mudanças abruptas de cor (COMANICIU; MEER, 2002). Long e Zhao (2005) utilizam a segmentação pelo deslocamento médio para detecção de estradas, onde posteriormente aplicam uma limiarização para separar as estradas do fundo.

Redes neurais (HAYKIN, 2008) podem ser usadas para detecção de estradas. Mokhtarzade e Zoj (2007) criaram uma rede neural composta por três camadas. A camada de saída é composta de um neurônio que aponta se o pixel central da janela pertence à classe de estradas ou não. A camada de entrada recebe os pixels em uma janela 3×3 em torno do pixel central e as distâncias euclidianas, no espaço espectral, desses pixels ao vetor que representa a coloração média de uma estrada. A quantidade de neurônios na camada intermediária é ajustável.

A morfologia matemática (FACON, 1996) estuda algoritmos sensíveis às formas dos objetos presentes na imagem. O processamento de imagens baseado em morfologia matemática transforma a forma desses objetos. As primeiras aplicações compreendiam apenas imagens binárias, mas as iniciativas neste campo se estenderam para imagens em níveis de cinza.

Zhang et al. (1999) aplicam morfologia matemática para limpeza de ruídos no processo de detecção de estradas a partir de uma ortofoto. O primeiro passo é uma classificação não-supervisionada a partir do algoritmo ISODATA (*Iterative Self Organizing Data Analysis Technique*) (BALL; HALL, 1965), que separa as estradas do fundo. Ainda restando ruído na imagem classificada, uma reconstrução morfológica é aplicada para a obtenção das componentes conexas. Então, uma análise das dimensões dos objetos presentes na imagem é feita para definir o limiar usado para filtragem de ruído. Uma abertura morfológica é aplicada para filtrar as componentes conexas que são identificadas

como ruído e manter as demais.

1.2 Métodos semiautomáticos

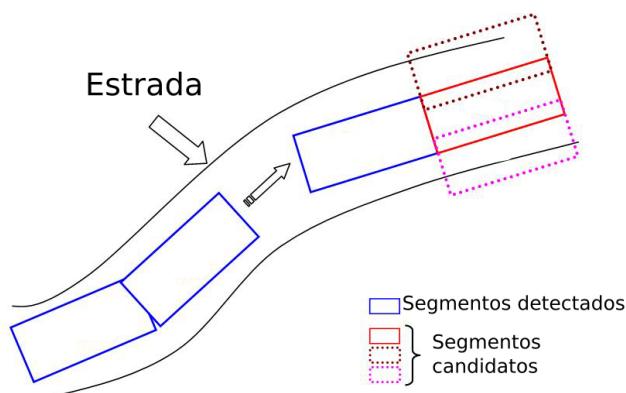
A presente seção se dedica à apresentação sucinta dos métodos semiautomáticos de extração de estradas. Tais métodos se caracterizam pelo fato de demandarem por parte do usuário informação preliminar a fim de auxiliar o procedimento de extração.

Nos métodos fundamentados na classificação supervisionada, o usuário seleciona e informa a classe de algumas áreas da imagem, criando os denominados padrões de treinamento. Em seguida, as estradas são separadas do fundo e delineadas. Song e Civco (2004) aplicaram como classificador uma Máquina de Suporte Vetorial, *Support Vector Machine – SVM*, (CORTES; VAPNIK, 1995) para separar a imagem em duas classes: estradas e não-estradas. A classificação é ruidosa, confundindo área urbana e estradas. A classe de estradas então é dividida em objetos através de um algoritmo baseado em crescimento de regiões. Para os objetos resultantes da segmentação são calculadas as propriedades de suavidade e densidade, sendo que os objetos relativos a estradas são obtidos através de limiarização com base nos valores dessas propriedades. O afinamento, procedimento que busca a minimização da largura das estradas, e a vetorização automática são os últimos passos do método.

Pandit (2009) propõe um algoritmo iterativo para extração em imagens pancromáticas IKONOS chamado “Correspondência Textural Adaptável para a Extração de Estradas” (*Adaptive Texture Matching for Road Extraction – ATM-R*) em que começa com o operador fornecendo alguns retângulos correspondentes às estradas. Então, a partir desses são criados outros retângulos adjacentes, como mostra a Figura 3, que serão candidatos a compor a estrada. Os melhores retângulos adjacentes são escolhidos para integrar a estrada com base na semelhança textural com os retângulos detectados no passo anterior. O processo continua até que os retângulos adjacentes não sejam mais semelhantes texturalmente aos retângulos detectados no passo anterior, o que é dado pela ultrapassagem de um determinado limiar na diferença textural. Este algoritmo parte do pressuposto que a estrada é uma região alongada com mudanças suaves nos níveis de cinza e direção.

Nos métodos de rastreamento de estrada o usuário provê informação sobre pontos ou direção da estrada para o algoritmo traçar a estrada. Esses métodos podem ser agrupados em duas categorias: otimizadores e rastreadores de caminho. Otimizadores são designados para achar um caminho ótimo que liga um conjunto de pontos na estrada em que são conhecidos os pontos inicial e final. Rastreadores trabalham de uma maneira distinta e buscam delinear a estrada a partir de um ponto e uma direção inicial. Segundo Amo et al. (2006), os otimizadores oferecem vantagem em relação aos rastreadores, já que

Figura 3 - Método semiautomático proposto por Pandit (2009)



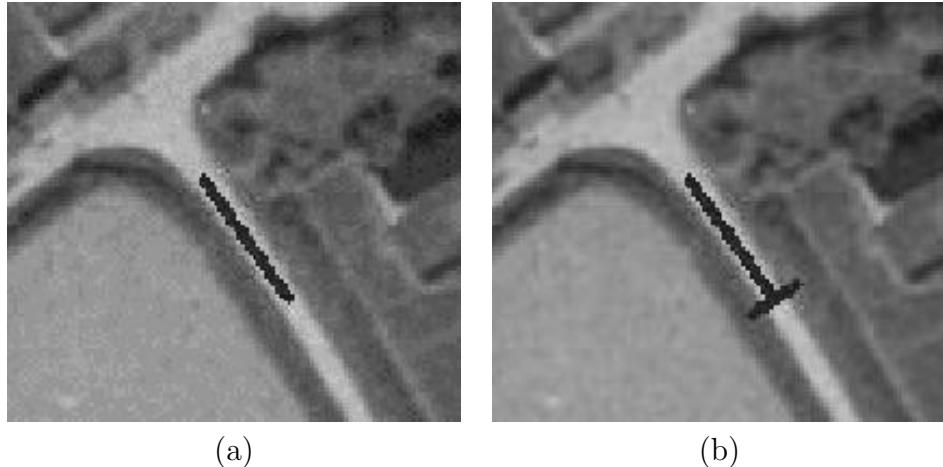
o usuário não precisa se preocupar com a possibilidade do rastreador perder a orientação da estrada.

Hu et al. (2004) propõem um otimizador de caminho em que os operadores fornecem pontos sementes de forma a prover uma aproximação grosseira para a estrada. Para cada par de segmentos que o operador fornece é traçada uma parábola de forma que a curva tenha continuidade de ordem zero - as várias parábolas possuem pontos de junção, mas as funções derivadas da junção das parábolas nesse ponto não são diferenciáveis. O algoritmo detecta as bordas da estrada e realiza um ajustamento de mínimos quadrados para calcular os parâmetros das parábolas.

Baumgartner et al. (2002) propõem uma ferramenta rastreadora de caminho baseada em um perfil da estrada. Primeiramente, o usuário provê um pequeno trecho inicial da estrada e um perfil perpendicular a esse trecho, de forma a colher uma amostra de cor referente aos níveis de cinza contidos na estrada. Perfis paralelos a este são traçados de forma a obter um perfil de referência, que é a média de todos os perfis perpendiculares ao trecho inicial. O rastreador prevê aproximadamente o próximo ponto da estrada com base na direção desse trecho, coleta perfis na vizinhança desse local e realiza uma correspondência de perfil de forma a determinar o próximo ponto da estrada com precisão. O processo continua até o rastreador parar. O rastreador para caso atinja a borda da imagem, caso ele chegue em outra estrada já extraída ou caso ele perca o rumo da estrada. Para análise do último caso, uma contagem dos perfis que apresentaram boa correlação é feita. Caso essa contagem ultrapasse um determinado limiar, o rastreador sabe que perdeu o rumo. A informação de entrada para o método está ilustrada na Figura 4.

O Toolbox Adaplin, ferramenta elaborada nesta dissertação, é inspirado no método otimizador de caminho com base em programação dinâmica (ZIVIANI, 2004) proposto por Gruen e Li (1997). O método inspirador foi escolhido por apresentar uma modelagem conceitual consistente da estrada e pelo fato de ser semiautomático, possibilitando a participação do operador no processo de extração com o consequente acréscimo na qualidade

Figura 4 - Rastreador de caminho proposto por Baumgartner et al. (2002).



Legenda: (a) trecho inicial traçado pelo operador; (b) perfil transversal da estrada.

do resultado. No método, o usuário fornece uma polilinha com a representação grosseira da estrada. A primeira ação tomada pelo otimizador de caminho será mudar cada vértice de posição caso seja necessário pesquisando nas suas vizinhanças se há um ponto que se adapte melhor às características da estrada. Uma dessas características, a curvatura da estrada, a qual o algoritmo procura minimizar, estabelece que a linha será representada por uma spline cúbica (BARTELS; BEATLY; BARSKY, 1995). Após esse procedimento dois vértices equidistantes são inseridos entre cada par de vértices da estrada traçada originalmente pelo operador. O ajuste também é feito para esses vértices e o processo continua até que seja atingida uma convergência. A cada novo vértice adicionado verifica-se a condição de colinearidade com os vértices vizinhos e a existência de erros grosseiros que geram quinas entre os segmentos da estrada. Caso alguma dessas possibilidades aconteça, o vértice é removido. No presente trabalho, o enfoque será dado a estradas em regiões rurais a partir de imagens de média resolução, compreendendo elemento de resolução no terreno entre 5 metros e 30 metros.

1.3 Atualização e controle da qualidade de bases cartográficas

Métodos de atualização de bases cartográficas se baseiam fortemente na análise da qualidade dos dados. Existe uma área de superposição entre o controle da qualidade dos dados e a atualização de dados presentes em bases cartográficas. Este fato se explica porque, se há uma necessidade de atualização, os dados disponíveis anteriormente já não satisfazem à realidade atual. Assim sendo, o panorama antigo pode ter sido modificado ou estar incompleto. No caso de uma base cartográfica de estradas, estradas antigas podem

ter sido modificadas e estradas novas podem ter sido adicionadas, fazendo com que a base precise de atualização.

Helmholz et al. (2007) escrevem sobre o trabalho feito pela *BKG* (Agência Federal Alemã de Cartografia e Geodésia) de atualização do sistema de bases cartográficas alemãs denominado *ATKIS*. O sistema *ATKIS* compreende bases cartográficas em diversas escalas, como por exemplo 1:25.000 e 1:50.000. O *BKG* tem o trabalho de verificar a qualidade e atualizar essas bases cartográficas, por isso desenvolveram um projeto chamado *WiPKA-QS* junto com o Instituto de Fotogrametria e GeoInformação (*IPI*) e o Instituto de Processamento de Informação (*TNT*) da Universidade Leibniz Hannover. O projeto *WiPKA-QS* objetiva a atualização semiautomática de objetos nos quais existem mudanças frequentes, tais como estradas, áreas industriais, terras de cultivo, pastagens e florestas. Caso um objeto passe nos testes automatizados da qualidade, ele é aceito, caso contrário ele é enviado para a validação por um operador. Para realizar tal tarefa, procedimentos automatizados são realizados usando as bases de dados existentes junto com imagens do satélite IKONOS.

No caso específico das estradas, Gerke et al. (2004) detalham melhor o processo de validação da base. Para a verificação da qualidade dos dados, uma região de interesse (*buffer*) com largura igual à largura da estrada mais o valor de 3 metros, que é a precisão almejada para a base, é gerada ao redor de cada estrada. Primeiramente um algoritmo de extração de estradas é aplicado dentro de cada *buffer*. O relacionamento topológico entre estradas e árvores é modelado porque em regiões rurais as árvores podem causar obstruções para a visualização das estradas. Dessa forma, caso as estradas extraídas pelo algoritmo correspondam às estradas originais e satisfaçam o critério topológico, elas são aprovadas na verificação, caso contrário são rejeitadas. O segundo passo consiste em avaliar as conexões entre as estradas, onde as estradas anteriormente rejeitadas são reavaliadas caso elas possuam um papel importante de conexão na rede viária.

Zhang e Baltsavias (2002) descrevem o trabalho feito na atualização e melhoria de qualidade de mapas 1 : 25.000 da Suíça referente ao projeto ATOMI (*Automated reconstruction of Topographic Objects from aerial images using vectorized Map Information*). Cada estrada possui uma região de influência delimitada considerando o erro máximo aceitável na base. Para a correta extração das estradas, múltiplas informações são usadas e combinadas. Os segmentos referentes às bordas das estradas são detectados através do algoritmo de Canny (1986) aplicado a pares estereoscópicos de imagens aéreas. Uma classificação através do método ISODATA (BALL; HALL, 1965) também é aplicada para dividir as imagens aéreas nas classes de estradas, objetos verdes, sombras, telhados escuros e telhados vermelhos. Um Modelo Digital de Terreno (MDT) também é utilizado, assim como marcas nas estradas, como setas indicando a orientação e faixas de pedestre. Então, uma série de relações é feita para a identificação das estradas. Uma estrada precisa possuir bordas paralelas, entre essas bordas precisa haver a classe estrada definida pela

classificação, ela precisa estar acima ou sobre a superfície caracterizada pelo MDT. Áreas de sombra são verificadas para analisar se causam obstruções na estrada. Setas e faixas de pedestre são usadas para confirmar o posicionamento das bordas da estrada, assim como para extrair o eixo central das estradas.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Termos relacionados a resolução de imagens no sensoriamento remoto

Os termos que levam a palavra resolução são comumente citados na literatura de sensoriamento remoto para expressar algumas características de imagens de satélite ou aéreas. A definição destes termos é importante, pois são usados intensamente ao longo da dissertação.

Resolução espacial se refere ao tamanho, expresso em termos do sistema de coordenadas do terreno, dos pixels de uma imagem. Comumente, esse tamanho diz respeito à medida do lado do pixel no terreno e é referida na literatura pelo termo *Ground Sample Distance* (GSD), que em português significa Distância de Amostragem no Terreno. Por exemplo, para uma imagem cujo pixel mede $y \times y$ metros no terreno, a resolução espacial é de y metros. Trata-se de um conceito também usado para a distinção de classes de produtos do sensoriamento remoto. Considera-se de alta resolução imagens com pixel de até 5 metros. O conceito de média resolução espacial refere-se a produtos cujo tamanho de pixel situa-se entre 5 metros e 30 metros. Por outro lado, são denominados pelo termo baixa resolução aqueles produtos com pixel maior que 30 metros. No âmbito da cartografia, a resolução espacial do dado de sensoriamento remoto limita a escala de mapeamento, pois não é possível identificar objetos menores que a resolução espacial da imagem.

A resolução espectral expressa a quantidade, distribuição e largura das faixas (ou bandas) espectrais captadas por um determinado sensor remoto. O conhecimento da resolução espectral é importante na identificação de elementos na imagem, já que alguns elementos são mais facilmente reconhecidos usando-se certas faixas espectrais. Vegetação, por exemplo, possui boa resposta espectral na região do infravermelho próximo.

Por sua vez, o termo resolução radiométrica se refere à quantidade de valores distintos que um pixel pertencente a uma banda espectral pode assumir. Quanto maior a resolução radiométrica de uma imagem, maior será sua capacidade de diferenciar variações de intensidade de energia. Por conta de tratarem-se nos dias de hoje dados de sensoriamento remoto em ambientes quase que exclusivamente digitais, esta quantidade é por vezes expressa em bits por pixel. No contexto do processamento digital de imagens, o processo de conversão da energia em um número digital é conhecido como quantização.

2.2 Programação Dinâmica

A técnica de programação dinâmica é aplicável quando é possível decompor um problema em etapas menores sucessivas. Ao percorrer estas etapas todos os resultados

parciais possíveis vão sendo armazenados e atualizados para cada etapa de forma que, quando se chega ao final, é possível saber qual a solução ótima do problema e como se chega a esta solução.

Nesta dissertação, o interesse é aplicar a programação dinâmica para achar o caminho ótimo em um grafo acíclico dirigido (THULASIRAMAN; SWAMY, 1992), dentre os diversos caminhos possíveis, desde um vértice i até um vértice j maximizando uma função objetivo. Esse processo é descrito em Horowitz e Sahni (1984). O grafo é dividido em etapas, onde k é o número de etapas e V_i designa a etapa i do grafo, sendo que o número de etapas é igual ao número de linhas do grafo. Em cada etapa V_i , há um certo estado x_k , onde x_k é uma variável que significa algum vértice do grafo que esteja na etapa k . Por exemplo, na Figura 5, há apenas o vértice 1 na etapa V_1 , portanto x_1 só poderá assumir este valor. Já na etapa V_2 , há quatro estados possíveis para x_2 .

O modelo da programação dinâmica se adequa a problemas onde as subsoluções que formam um caminho ótimo também sejam ótimas, a assim chamada subestrutura ótima do problema (ZIVIANI, 2004). Por exemplo, caso o caminho ótimo de uma cidade A até uma cidade C passe por B, então o caminho de A até B precisa ser ótimo assim como o caminho de B até C. Então, ao percorrer as etapas, os subcaminhos vão sendo otimizados e combinados até chegar na etapa final com um conjunto restrito de caminhos para comparar.

Um exemplo de resolução de um problema através de programação dinâmica será descrito a seguir.

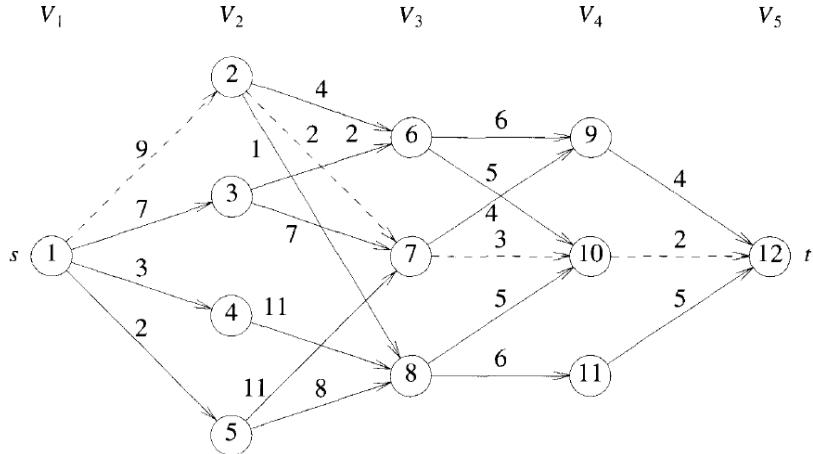
2.2.1 Exemplo de otimização com programação dinâmica

Para mostrar o problema da programação dinâmica aplicada a um grafo, parte-se de um exemplo considerado por (HOROWITZ; SAHNI, 1984) e mostrado na Figura 5. O problema se resume a ir desde o nó s até o nó t pelo caminho de custo mínimo, sendo que os custos de passar de um nó a outro são representados pelos valores associados às respectivas arestas.

Um aspecto interessante de notar no grafo é que para quaisquer dois vértices i e j em etapas distintas que possam ser conectados por um caminho, há um caminho de custo mínimo que liga esses dois vértices. A função $c_min(i, j)$ será adotada para referir-se ao custo mínimo do caminho entre os vértices i e j , enquanto $cam_min(i, j)$ será adotado para indicar o caminho que possui esse custo mínimo. Ainda, para vértices i e j adjacentes, $c(i, j)$ aponta o custo da aresta que liga i a j .

O custo mínimo entre s e t pode ser calculado levando em conta o princípio de que a solução ótima para o problema deve ser formada por subsoluções ótimas. Sendo assim,

Figura 5 - Grafo de exemplo para a programação dinâmica



Fonte: (HOROWITZ; SAHNI, 1984)

o custo mínimo entre s e t pode ser enunciado pela Equação 1.

$$c_min(s, t) = \min_{x_2 \in V_2} \{c(s, x_2) + c_min(x_2, t)\} \quad (1)$$

Outro entendimento relevante é que a decisão sobre a opção de um nó na etapa V_4 , para que o caminho tenha custo mínimo, é trivial, já que o leque de escolhas se restringe ao nó final t . Então o primeiro passo é calcular os custos mínimos desde os vértices na etapa V_3 até o vértice final t .

Para que seja possível obter o caminho de custo mínimo ao final da programação dinâmica, outra ação tomada será gravar a decisão de custo mínimo para cada vértice. Essa decisão é o vértice na etapa seguinte que leva a um caminho de custo mínimo até o vértice final t . Sua representação será feita por $d(i)$, onde i é um vértice.

$$c_min(6, t) = \min \{6 + 4, 5 + 2\} = 7 \quad d(6) = 10 \quad (2)$$

$$c_min(7, t) = \min \{4 + 4, 3 + 2\} = 5 \quad d(7) = 10 \quad (3)$$

$$c_min(8, t) = \min \{5 + 2, 6 + 5\} = 7 \quad d(8) = 10 \quad (4)$$

Com este passo concluído, é possível agora calcular os custos mínimos desde os

vértices na etapa V_2 até o vértice t reutilizando os valores calculados para a etapa V_3 , assim evitando cálculos desnecessários.

$$c_min(2, t) = \min \{4 + c_min(6, t), 2 + c_min(7, t), 7 + c_min(8, t)\} = 7 \quad d(2) = 7 \quad (5)$$

$$c_min(3, t) = \min \{2 + c_min(6, t), 7 + c_min(7, t)\} = 9 \quad d(3) = 6 \quad (6)$$

$$c_min(4, t) = \min \{11 + c_min(8, t)\} = 18 \quad d(4) = 8 \quad (7)$$

$$c_min(5, t) = \min \{11 + c_min(7, t), 8 + c_min(8, t)\} = 15 \quad d(5) = 8 \quad (8)$$

Finalmente é possível obter $c_min(s, t)$ aplicando a Equação 1.

$$\begin{aligned} c_min(s, t) &= \min \{9 + c_min(2, t), 7 + c_min(3, t), 3 + c_min(4, t), 2 + c_min(5, t)\} = 16 \\ d(s) &= 2 \end{aligned} \quad (9)$$

Para obter o caminho de custo mínimo é necessário resgatar todas as decisões $d(i)$ tomadas. Por exemplo, $d(s) = 2$, portanto o vértice 2 vem logo após o vértice inicial. Depois vem o vértice 7, pois $d(2) = 7$, e o vértice 10, pois $d(7) = 10$. Logo, o caminho de custo mínimo, $cam_min(s, t)$, é igual a $(s, 2, 7, 10, t)$, que está representado pela linha tracejada na Figura 5.

2.3 A ferramenta de mapeamento QgsMapTool

Ferramentas interativas possibilitam ao usuário clicar e interagir com o mapa presente na tela do computador de forma dinâmica. Então, ao mover o mouse ou ao clicar na tela, ações são executadas, o que permite uma interação do usuário com o aplicativo SIG. Para a criação de uma base cartográfica de estradas, uma ferramenta interativa de criação de feições é muito interessante, pois o operador já vê e edita as feições criadas, de forma a diminuir o trabalho de pós-edição.

A ferramenta semiautomática foi implementada como uma extensão do software li-

vre QGIS. Para isto, foi utilizada a biblioteca PyQGIS, que é escrita na linguagem Python e já vem acoplada ao software. Esta biblioteca permite a utilização das funcionalidades do aplicativo SIG através da linguagem Python, possibilitando a criação, edição, transformação e manipulação de vetores e *rasters*. Além dessa biblioteca, foi usada a biblioteca PyQt, que é a interface Python da biblioteca Qt de criação de interface gráfica.

Uma classe especial presente na biblioteca PyQGIS é a QgsMapTool, que define o comportamento de ações interativas do mouse e do teclado sobre o mapa na tela. As ferramentas de zoom, translação e identificação de feições, por exemplo, são criadas através de subclasses da QgsMapTool (WESTRA, 2014). A geração de ferramentas interativas customizadas envolvem a criação de subclasses da QgsMapTool, reimplementando seus métodos e possivelmente criando outros.

A classe QgsMapTool possui métodos virtuais que podem ser reescritos nas subclasses para estabelecer o comportamento da ferramenta. A seguir são apresentados os métodos virtuais da classe QgsMapTool julgados mais relevantes para a implementação do ToolBox Adaplin.

O método *activate()* é chamado toda vez que o usuário define que a respectiva extensão ou ferramenta básica do QGIS é a ferramenta de mapa em uso no programa. Conceitualmente a implementação do método confirma o estado ativo de utilização da ferramenta, mantém pressionado o botão da interface gráfica e estipula o novo cursor para o mouse. Por sua vez, o método *deactivate()* é chamado toda vez que a ferramenta designada pelo objeto é desativada, deixando de ser a ferramenta em utilização. A implementação padrão desse método sinaliza para o programa que esta não é mais a ferramenta de mapa que está sendo usada, desoprimindo o botão da interface gráfica. Quando necessário, sua reimplementação deve modificar o cursor do mouse.

O método *canvasDoubleClickEvent()* é chamado toda vez que, durante o uso da ferramenta, for dado um duplo clique com o mouse. A implementação padrão do método não faz nada. Ao reimplementá-lo em uma subclasse, é possível definir um comportamento específico que ocorrerá a partir do evento do duplo clique do mouse sobre a área útil da tela do aplicativo. O mesmo contexto se aplica aos métodos *canvasMoveEvent()*, que trata o evento de movimentação do mouse sobre a tela, *canvasPressEvent()*, que é chamado quando o usuário pressiona o botão do mouse, e *canvasReleaseEvent()*, responsável por tratar o evento de desapertar o botão do mouse.

Outros métodos virtuais importantes no presente contextos são o *keyPressEvent()*, acionado quando uma tecla é pressionada, o *keyReleaseEvent()*, chamado toda vez que uma tecla é desapertada, *setCursor()*, usado para designar um cursor específico para a ferramenta de mapa, e *wheelEvent()*, que é chamado quando o usuário usa o botão de rolagem do mouse.

3 MÉTODO

O método implementado na presente pesquisa é inspirado e adaptado a partir do método originalmente proposto por Gruen e Li (1997). Em benefício da compreensão do texto, no presente capítulo, a apresentação do método original precede a apresentação do método implementado na ferramenta Adaplin para QGIS.

3.1 Método original de Gruen e Li

O método proposto por Gruen e Li requer, inicialmente, que o usuário provenha pontos semente da polilinha, representando a estrada de maneira aproximada. A respectiva feição será extraída posteriormente através do procedimento de inserção e interpolação de pontos conforme a função objetivo, usando a técnica de programação dinâmica. A polilinha final contém n vértices, e é descrita por $P = \{p_1, p_2, p_3, \dots, p_n\}$, onde $p_i = (x_i, y_i)$ são as coordenadas de imagem do i -ésimo ponto da polilinha.

A função objetivo usada no método leva em conta características de uma estrada em uma imagem digital em níveis de cinza. Estas características estão descritas através das 6 propriedades listadas a seguir:

1. Os pixels em uma estrada possuem níveis de cinza mais altos quando comparados com os pixéis ao redor da feição. Isso se deve ao pressuposto: a estrada está localizada em áreas rurais, onde o entorno da estrada geralmente é marcado por vegetação, que produzirá níveis de cinza mais baixos que a estrada.
2. A variação do nível de cinza ao longo de uma distância curta na estrada é baixa. Esta propriedade decorre do pressuposto que o material de revestimento da estrada varia pouco em uma distância curta. Dessa forma, os níveis de cinza da estrada, decorrentes da reflectância do revestimento, provavelmente não diferirão muito.
3. Uma estrada é uma feição linear com alto nível de cinza. Esta propriedade é uma generalização das duas primeiras.
4. Uma estrada possui uma curvatura suave. Esta é uma propriedade geométrica e que reflete parâmetros empregados na construção.
5. Há um limite para a curvatura da estrada, restringindo a acentuação de sua curvatura. Esta propriedade e a anterior decorrem do pressuposto que uma estrada é feita considerando a facilidade da condução dos veículos por parte dos motoristas.
6. Não há uma mudança significativa na largura de uma estrada.

Para a modelagem matemática das propriedades citadas, Gruen e Li usam uma curva ξ contínua na imagem digital para representar a estrada e fazem as seguintes assunções acerca dessa curva e da imagem digital.

- A curva ξ é descrita por uma função $\mathbf{f}(s)$ que mapeia um arco s da curva para pontos (x, y) da imagem.
- A curva ξ possui derivadas contínuas e um vetor $\mathbf{n}(s)$ que é normal a curva em cada ponto.
- A imagem digital é representada por uma função contínua $G(x, y)$ que possui derivadas contínuas.

A primeira propriedade apresentada anteriormente implica em maximizar a função E_{p1} dada abaixo pela Equação 10,

$$E_{p1} = \int [G(\mathbf{f}(s))]^2 ds. \quad (10)$$

A segunda propriedade representa a minimização da função E_{p2} dada pela Equação 11,

$$E_{p2} = \sum_i \int_{\Delta s_i} [G(\mathbf{f}(s)) - G_m(\Delta s_i)]^2 ds, \quad (11)$$

onde Δs_i é um trecho curto ao longo da curva ξ e $G_m(\Delta s_i)$ é a média dos níveis de cinza nessa curva.

A terceira propriedade equivale à maximização da função E_{p3} , referenciada pela Equação 12:

$$E_{p3} = \int w(d(s))[G(\mathbf{f}(s)) + d(s)\mathbf{n}(s)]^2 ds, \quad (12)$$

onde, segundo Gruen e Li (1997), $d(s)$ é a distância entre a curva ξ e a feição linear próxima a ela, sendo $w(d(s))$ uma função Gaussiana de peso que decresce a medida que $d(s)$ aumenta.

A quarta propriedade consiste em minimizar a função E_g dada pela Equação 13,

$$E_g = \int |\mathbf{f}''(s)|^2 ds. \quad (13)$$

A quinta propriedade é uma restrição à curvatura da estrada, representada por C_g na Equação 14,

$$C_g = |\mathbf{f}''(s)| < T, \quad (14)$$

onde T é um limiar para a curvatura.

Finalmente, a sexta propriedade não é implementada de maneira direta, mas seu conceito está presente na modelagem matemática da terceira propriedade.

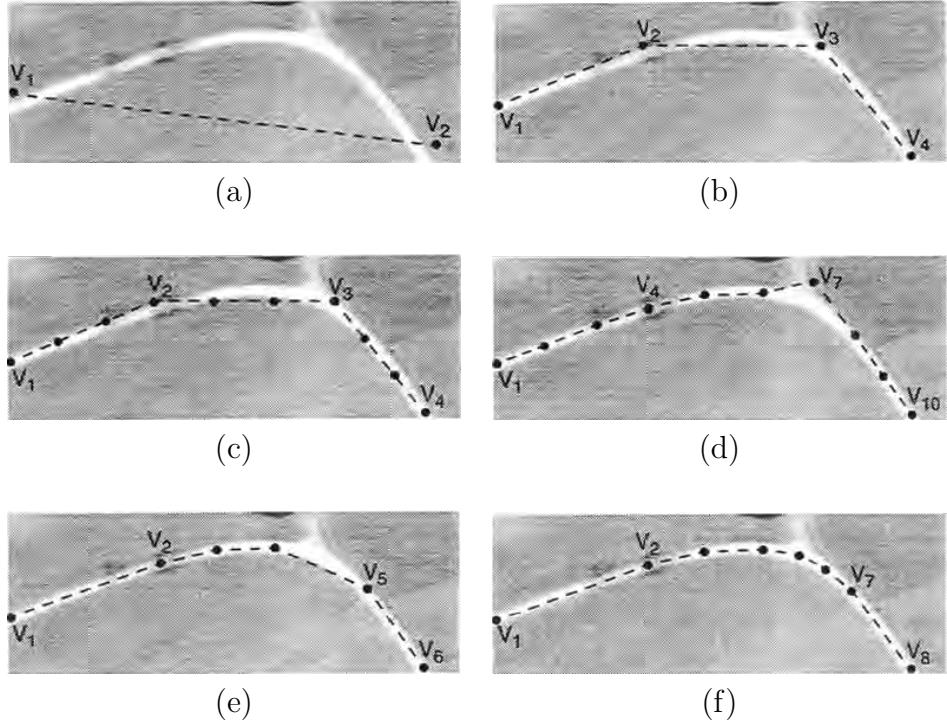
O refinamento que faz a polilinha inicial adquirir a forma da estrada é um processo iterativo e leva em conta uma combinação de procedimentos de inserção de pontos junto com movimentação destes pontos através de programação dinâmica. O procedimento que se repete é a inserção de pontos equidistantes entre cada par de pontos de um segmento e o refinamento deste segmento dando origem a novos segmentos. O número de pontos inseridos a cada iteração é dois. Há também um critério de parada que decide quando abortar a inserção de pontos porque a polilinha já se assemelha à estrada. Além disso, efetua-se a remoção de pontos colineares redundantes que não alteram a forma da estrada e de pontos que geram um zigue-zague. A Figura 6 ilustra os passos do método de inserção e refinamento dos pontos.

Através de programação dinâmica, o refinamento da polilinha, iniciado após a inserção de pontos, é realizado procurando a melhor opção nas redondezas de cada vértice em uma janela de determinada dimensão. Esta janela significa uma vizinhança de pixels na imagem, por exemplo de 5×5 , com centro em cada vértice.

Gruen e Li (1997) sugerem que é possível otimizar esta busca através de duas opções:

- Restringindo a busca a uma linha perpendicular ao polígono em cada vértice ao invés de uma janela.
- Usando uma janela de resolução variada. Por exemplo, a janela na primeira iteração pode apresentar espaçamento grande, de três pixels, entre os vértices. Na segunda iteração, o espaçamento pode ser menor, na terceira pode ser menor ainda e assim por diante, podendo inclusive chegar no nível do subpixel. A finalidade desse procedimento é gerar um grande raio de convergência na primeira iteração, pesquisando alternativas de traçado por lugares afastados da polilinha inicial, definida pelo operador, e um refinamento apurado com o uso de espaçamentos cada vez menores nas iterações subsequentes, permitindo um resultado final de alta precisão.

Figura 6 - Etapas do método proposto por Gruen e Li (1997)



Legenda: (a) ponto inicial e final marcados pelo usuário; (b) inserção de 2 pontos e refinamento desses por programação dinâmica; (c) inserção de pontos em cada segmento; (d) refinamento desses pontos; (e) remoção de pontos redundantes e erros grosseiros; (f) inserção e refinamento final.

3.1.1 Função objetivo

A função objetivo de Gruen e Li procura combinar a modelagem matemática das seis propriedades da estrada, sendo essa modelagem apresentada para o caso discreto em que uma polilinha de n vértices é descrita por $P = \{p_1, p_2, p_3, \dots, p_n\}$, onde $p_i = (x_i, y_i)$ são as coordenadas do i -ésimo ponto da polilinha.

A função objetivo procura maximizar E_{p1} e E_{p3} , enquanto minimiza E_{p2} e E_g . Há também uma restrição C_g que impõe uma limitação à curvatura da estrada, sendo que a função objetivo somente é avaliada onde a condição imposta por C_g é verdadeira. E_g e C_g são dadas no caso discreto pelas Equações 15 e 16.

$$E_g = \sum_i [2 - 2 \cos(\alpha_i - \alpha_{i+1})] / |\Delta s_i| \quad (15)$$

$$C_g = |\alpha_i - \alpha_{i+1}| < T \quad (16)$$

onde α_i é a direção estabelecida pelos pontos p_{i-1} e p_i e $|\Delta s_i|$ é a distância entre eles, enquanto α_{i+1} é a direção estabelecida pelos pontos p_i e p_{i+1} , sendo $\alpha_i - \alpha_{i+1}$ o ângulo de deflexão formado entre as direções α_i e α_{i+1} tendo T como limiar.

A função objetivo proposta por Gruen e Li é dada por E , conforme representado na Equação 17,

$$E = \sum_i [E_{p1}(p_i, p_{i+1}) - \beta E_{p2}(p_i, p_{i+1}) + \gamma E_{p3}(p_i, p_{i+1})] \times (1 + \cos(\alpha_i - \alpha_{i+1}))/|\Delta s_i|, \quad (17)$$

onde, β e γ são constantes positivas que funcionam como pesos para E_{p2} e E_{p3} .

3.2 Método empregado na ferramenta Adaplin

Apesar de, na ferramenta Adaplin, a solução proposta ser inspirada em (GRUEN; LI, 1997), os problemas aos quais os métodos se propõem a resolver diferem do ponto de vista conceitual. Por um lado, o método de Gruen e Li (1997) requer que um número razoável de pontos seja indicado ao longo de toda a feição antes que o procedimento de otimização seja invocado. Por outro lado, na ferramenta Adaplin, a partir do segundo ponto, cada novo ponto indicado pelo operador define um segmento preliminar conectando o ponto anterior ao ponto que acaba de ser medido. A partir deste momento, se inicia o processo de otimização responsável por inserir dois pontos neste segmento. O objetivo da otimização é fazer a polilinha se aproximar do contorno da estrada observado na imagem de sensoriamento remoto. Este procedimento se repete a cada novo ponto inserido e revela o caráter interativo da ferramenta Adaplin, na qual, enquanto o operador delimita pontos da estrada, a polilinha é gerada e apresentada em tempo real.

Do ponto de vista formal, a questão central, então, se resume a achar a forma da estrada entre um ponto inicial e um ponto final, fornecidos pelo usuário. Portanto, a informação de entrada do método é um segmento $\overline{IF} = (p_i, p_f)$, onde $p_i = (x_i, y_i)$ e $p_f = (x_f, y_f)$ são as coordenadas de imagem dos pontos inicial e final do segmento.

A formação da polilinha será decorrente da agregação dos resultados do método para um conjunto de segmentos obtidos através dos pontos marcados pelo usuário. O método insere dois pontos equidistantes em cada segmento. Portanto, haverá a geração de $2(n_p - 1)$ pontos ao final da construção da polilinha, sendo n_p o número de pontos inseridos pelo usuário.

A inserção dos pontos é acompanhada de um ajustamento desses usando a técnica de programação dinâmica. Esse ajustamento considera um modelo conceitual de estrada parecido com o de Gruen e Li (1997). Entretanto, é mais simples, levando em conta três das propriedades listadas na seção anterior. Além disso, objetiva trabalhar com

imagens multiespectrais, por conseguinte, a média das componentes RGB é tomada ao invés do nível de cinza de uma imagem pancromática. As subseções a seguir apresentam o modelo conceitual da estrada, seu desenvolvimento matemático, assim como o algoritmo de inserção e otimização de pontos.

3.2.1 Modelo conceitual da estrada

Para que se possa extrair uma estrada a partir de uma imagem de modo automático ou semiautomático, deve-se primeiro definir precisamente como a estrada é representada na imagem. Distingue-se entre a situação de a estrada ser representada como um elemento linear e o cenário em que a estrada aparece como uma área alongada. Imagens de média resolução (5 a 30 metros de resolução espacial) tendem a considerar a estrada como um elemento linear. Enquanto, imagens de alta resolução (resolução espacial menor que 5 metros) tendem a considerá-la como uma faixa estreita e alongada. Assim, deve-se ressaltar que tudo depende da relação da largura da estrada com a resolução espacial da imagem.

O foco do trabalho são estradas que se assemelham a linhas na imagem, o que é adequado ao contexto de estradas rurais cuja largura possa ser representada por poucos pixels da imagem. Além disso, o modelo conceitual não considera as estradas que possam ter largura menor que a resolução espacial da imagem, o que demandaria o uso de técnicas específicas para o nível do subpixel. Portanto, para o caso de estradas que possuam largura menor que a resolução espacial da imagem, a influência do entorno da estrada não pode mascarar a resposta espectral da estrada de maneira que esta fique imperceptível.

O modelo geométrico e radiométrico da estrada considera as propriedades listadas a seguir:

1. Uma estrada possui boa refletância comparada com seu redor.
2. A variação da reflectância ao longo de uma estrada é suave.
3. Uma estrada possui uma curvatura suave, facilitando a condução dos veículos pelos motoristas.

A técnica de programação dinâmica permite otimizar uma determinada função objetivo. Esta técnica pode ser aplicada de forma vantajosa em relação a outras alternativas, como a recursão, por exemplo, sempre que a abordagem recursiva levar à decomposição de um problema $O(n)$ em n subproblemas $O(n - 1)$. Nestes casos, um tratamento recursivo poderá fornecer um algoritmo $O(2^n)$ (ZIVIANI, 2004).

A abordagem de programação dinâmica empregada no presente trabalho pressupõe que as propriedades da estrada sejam modeladas matematicamente. Para fins específicos

deste trabalho, considera-se uma imagem digital formada pela média das bandas RGB da imagem original definida pela função $f(x, y)$, onde (x, y) são as coordenadas espaciais discretas e $f(x_A, y_A)$ corresponde à média RGB do pixel A de coordenadas (x_A, y_A) . Além disto, uma polilinha P é representada pela sequência de coordenadas $\{p_1, p_2, \dots, p_n\}$.

A primeira propriedade expressa que uma estrada possui refletância elevada comparada com seu redor. Para que isto seja atingido, Gruen e Li (1997) argumentam que a soma dos quadrados dos níveis de cinza dos pixels que compõem a polilinha correspondente à estrada deve ser máximo. Como aqui utiliza-se a média das componentes RGB, esta propriedade ($Prop_1$) é descrita pela equação 18,

$$Prop_1(P) = \sum_{i=1}^n f(x_i, y_i)^2, \quad (18)$$

onde, (x_i, y_i) são as coordenadas dos vértices que compõem a polilinha P e $f(x_i, y_i)$ é a média das componentes RGB da imagem em (x_i, y_i) .

A segunda propriedade diz que a variação da reflectância ao longo de uma estrada é suave. Considerando que a polilinha é formada por um conjunto de segmentos e cada segmento possui uma resposta espectral média, a diferença entre a resposta espectral de um pixel que compõe a estrada e a resposta espectral média do segmento ao qual este pixel pertence deve ser mínima. Com maior abrangência, a soma dessas diferenças ao longo da estrada deve ser mínima. Esta propriedade ($Prop_2$) é descrita pela equação 20,

$$Prop_2(P) = \sum_{j=i}^{i+1} \sum_{i=1}^{n-1} \left[f(x_j, y_j) - \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1})}{2} \right]^2, \quad (19)$$

onde (x_i, y_i) e (x_{i+1}, y_{i+1}) são as coordenadas dos vértices que compõem um segmento contido dentro da polilinha e $f(x_i, y_i)$ e $f(x_{i+1}, y_{i+1})$ são as respectivas médias RGB.

A terceira propriedade estabelece que a curvatura da estrada é suave. Como a estrada é representada por uma polilinha que é formada por segmentos, os ângulos de deflexão entre os segmentos devem ser mínimos tal que a curvatura da estrada seja minimizada. Esta propriedade ($Prop_3$) está descrita pela equação 20,

$$Prop_3(P) = \sum_{i=2}^{n-1} \frac{[1 + \cos(\Theta_i)]}{|\Delta s_i|}, \quad (20)$$

onde Θ_i é o ângulo de deflexão entre o vetor formado pelos pontos (x_{i-1}, y_{i-1}) e (x_i, y_i) e o vetor formado pelo pontos (x_i, y_i) e (x_{i+1}, y_{i+1}) . $|\Delta s_i|$ é o comprimento do vetor formado pelos pontos (x_{i-1}, y_{i-1}) e (x_i, y_i) . Esta função funciona como um fator de escala na função

objetivo, atribuindo um peso para as outras funções. Sua importância é justificada pelo fato da forma ser uma característica marcante da estrada.

Com a combinação das 3 propriedades na função objetivo E , busca-se maximizar ($Prop_1$), minimizar ($Prop_2$) (por isso o sinal negativo), enquanto a suavidade ($Prop_3$) é maximizada. Quanto menor o Θ , maior a suavidade e consequentemente maior será o valor da função E ; quanto maior o Θ , menor a suavidade e consequentemente E será menor. A função objetivo E , a ser maximizada considerando todas polilinhas possíveis a partir de \overline{IF} , é descrita por:

$$E(P) = (Prop_1(P) - Prop_2(P)) Prop_3(P). \quad (21)$$

É importante notar que Gruen e Li colocam β e γ como constantes positivas na função objetivo para atribuir pesos às propriedades. No método empregado na ferramenta Adaplin, as constantes possuem o valor de 1. Um estudo mais específico seria necessário para encontrar os valores ótimos das constantes para uma determinada imagem.

3.2.2 Algoritmo de inserção e otimização de pontos

O algoritmo insere e otimiza 2 pontos em um segmento \overline{IF} correspondente a um par de pontos consecutivos marcados pelo operador. A polilinha final será resultado da aplicação do algoritmo para os diversos pares de pontos consecutivos assinalados pelo operador. Os pontos marcados pelo operador são mantidos fixos, sendo assim, os pontos inicial e final de um segmento são mantidos na mesma posição que o operador os colocou.

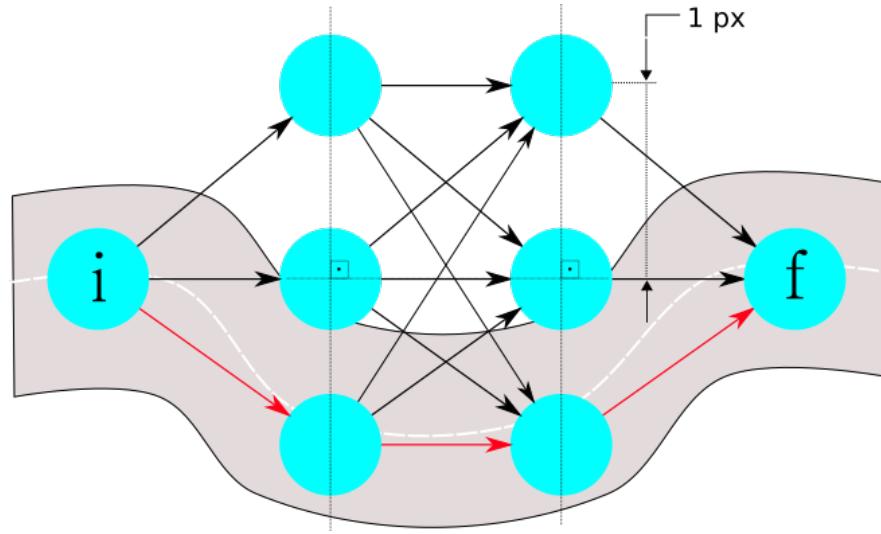
A otimização dos pontos inseridos em um segmento é feita buscando o caminho que maximiza a função objetivo em um grafo. A técnica de programação dinâmica é usada para realizar essa busca. O procedimento não é iterativo, sendo portanto realizado uma vez só, o que implica na formação de um único grafo para cada segmento.

O grafo acíclico dirigido contempla vértices divididos em etapas (ou linhas), sendo que os vértices de uma dada etapa estão situados na mesma reta perpendicular ao segmento \overline{IF} . A quantidade de vértices em uma dada etapa é arbitrária. O grafo é montado de forma que todos os vértices em uma dada etapa tenham conexão com todos os vértices da etapa seguinte.

O espaçamento entre os vértices também é arbitrário, entretanto, devem ser consideradas a largura da estrada e a resolução espacial da imagem. Por exemplo, para imagens com resolução espacial próximas à largura da estrada, acredita-se que um espaçamento em torno de um pixel garantirá que a polilinha fique situada dentro da estrada.

Supondo dois vértices i e f equivalentes aos pontos inicial e final do segmento \overline{IF} , a Figura 7 ilustra o grafo formado. Neste grafo, todos os caminhos serão testados

Figura 7 - Grafo acíclico dirigido formado e (em vermelho) caminho ótimo de i até f



com o objetivo de achar o caminho que maximiza a função objetivo descrita na subseção anterior. A título ilustrativo, na Figura 7, cada etapa intermediária, que corresponde às etapas dos pontos inseridos, contém três vértices com espaçamento de um pixel. Também a título de representação, o melhor caminho é apresentado em vermelho.

4 IMPLEMENTAÇÃO

Neste capítulo, são tratados os detalhes da implementação, no contexto da ferramenta Toolbox Adaplin, do método apresentado no capítulo anterior. A ferramenta foi implementada como uma extensão para o software livre QGIS (MENKE et al., 2016). A linguagem empregada foi a Python (LAWHEAD, 2015) com as bibliotecas PyQGIS (SHERMAN, 2014), proveniente do próprio software, e a biblioteca livre Qt (HARWANI, 2011), comumente utilizada para criação de aplicativos com interface gráfica. O Toolbox Adaplin foi implementado através da classe Adaplin que é uma subclasse derivada a partir de QgsMapTool (WESTRA, 2014).

Em seu modo normal, a ferramenta diminui o esforço do operador para delinear uma estrada. A partir do segundo ponto, cada ponto indicado forma um segmento ligando-o ao ponto anterior. Em seguida, a ferramenta insere dois pontos adicionais ao segmento, através de um processo de otimização usando programação dinâmica. Os pontos marcados pelo usuário permanecem fixos, possibilitando maior autonomia ao operador e garantindo um controle maior sobre a ferramenta. A otimização é realizada sempre sobre o último segmento marcado de forma a incorporar o resultado obtido à polilinha.

O processo de otimização que define os dois pontos adicionais se vale de um grafo direcional acíclico. Na implementação, os nós internos do grafo estão situados em duas linhas correspondentes a retas perpendiculares ao segmento definido pelos dois últimos pontos indicados pelo usuário. Tais retas perpendiculares dividem o segmento em três partes de dimensões idênticas. O número de vértices fixado para as etapas intermediárias, referentes aos pontos candidatos a serem inseridos, foi de onze, sendo um no cruzamento entre o segmento e a reta perpendicular, cinco vértices acima e cinco abaixo do segmento, conforme ilustrado pela Figura 9. Assim, os grafos utilizados possuem sempre a mesma topologia. Portanto, na camada V_1 encontra-se o nó de entrada correspondente ao penúltimo ponto indicado pelo operador. As camadas V_2 e V_3 possuem onze nós cada uma, correspondentes aos candidatos a pontos intermediários da polilinha. O espaçamento entre os vértices nessas camadas foi arbitrado em 4,50 metros, um pouco menor que a resolução espacial das imagens Rapideye usadas nos experimentos da presente dissertação. Por fim, na camada V_4 encontra-se o nó de saída, correspondente ao último ponto fornecido pelo operador. Neste tipo de grafo, todos os nós de uma camada estão ligados a todos da camada seguinte. Além disto, cada aresta do grafo possui como peso o respectivo valor da função objetivo, considerando a imagem de entrada. Uma vez estabelecido o grafo, pode-se realizar o procedimento de busca pelo caminho ótimo tendo em vista a função objetivo apresentada na Equação 21.

Também existe a opção de alternar para um modo manual. Neste modo, que é ativado pressionando-se a tecla **Ctrl**, todos os pontos da polilinha são definidos pelo

usuário. Esta alternativa foi implementada para possibilitar o delineamento da estrada nos momentos em que o procedimento de otimização for mal sucedido, como em áreas onde a estrada apresenta oclusões, a partir da existência de sombras ou árvores. Nestes casos, o modo manual garante maior autonomia ao operador.

É conveniente reforçar que os aplicativos SIG trabalham com dois tipos de sistemas de coordenadas: sistemas de coordenadas geográficas, baseados em longitude e latitude com unidade em grau decimal, e sistemas de coordenadas projetadas ou planas, baseados em eixos x e y com unidade em metro. Tanto a imagem como o vetor devem estar referenciados a um desses sistemas de coordenadas, sendo assim denominados georreferenciados. Neste sentido, a ferramenta Adaplin utiliza o espaço plano de uma projeção cartográfica ao invés do espaço imagem usado no método de Gruen e Li (1997).

Os aplicativos de SIG são capazes de sobrepor camadas de informação geográficas, vetores ou raster, em um determinado sistema de coordenadas que é usado para visualização. Alguns deles, como o QGIS, possuem a habilidade de transformação entre sistemas em tempo real, podendo os dados estar em sistemas distintos daquele usado para visualização dos dados na tela, pois o software os transforma instantaneamente para que o usuário possa compará-los com todos os dados restantes sobre um sistema de coordenadas único.

A implementação da Adaplin utiliza transformações de coordenadas entre sistemas em tempo real. Desta forma, os parâmetros para o espaçamento entre os vértices dentro do grafo e os cálculos de distâncias entre pontos na polilinha são dados em metros ao invés de pixels. Para o cálculo das componentes RGB de uma imagem em um certo ponto (x_A, y_A) , usa-se uma função do QGIS que intercepta a imagem raster nesse ponto retornando o valor do pixel nessa posição.

A Figura 8 ilustra o funcionamento da ferramenta. Nela, **a** e **b** correspondem a dois pontos consecutivos indicados pelo usuário. Note que o percurso da reta que conecta **a** e **b** destoa da estrada indicada na cor cinza.

A Figura 9 exibe o grafo formado para a realização do procedimento de otimização. Nesta implementação, cada camada intermediária possui onze nós.

Figura 8 - Dois primeiros pontos inseridos pelo operador

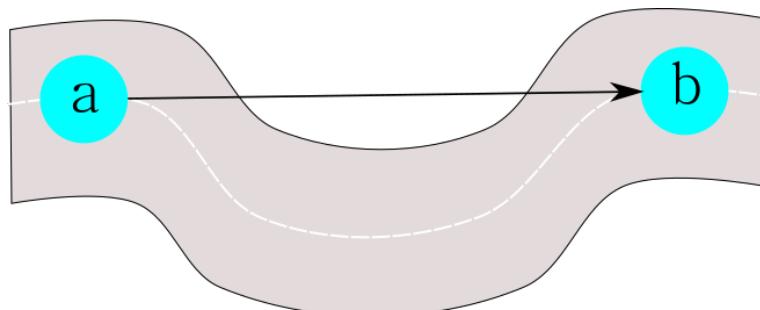
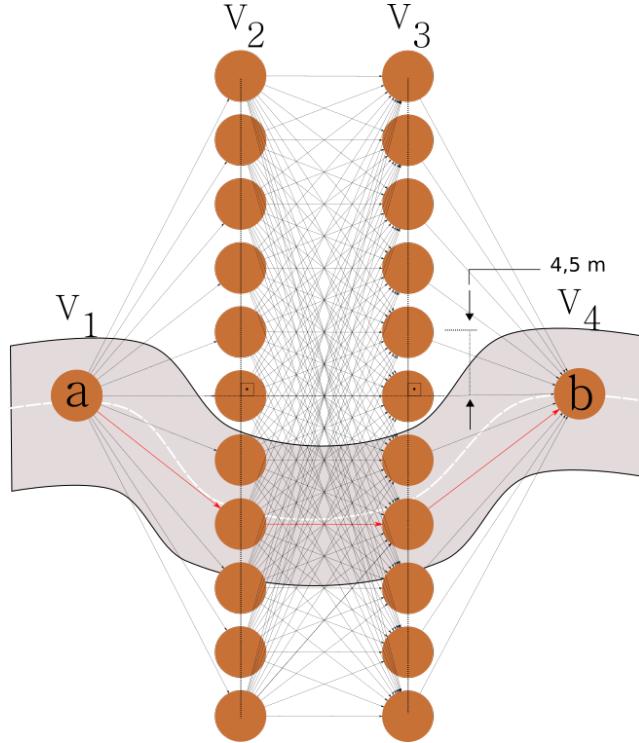


Figura 9 - Grafo formado e caminho escolhido de **a** até **b**



Aplicando-se programação dinâmica sobre este grafo, encontra-se o caminho ótimo entre **a** e **b**, indicado em vermelho na Figura 9. Este caminho será armazenado e não será mudado. O operador prossegue adicionando pontos e o mesmo procedimento é repetido. Por exemplo, sendo **c** o próximo ponto adicionado, então o caminho entre **b** e **c** será descoberto e será agregado ao caminho já conhecido entre **a** e **b**. O processo termina quando o usuário termina a polilinha.

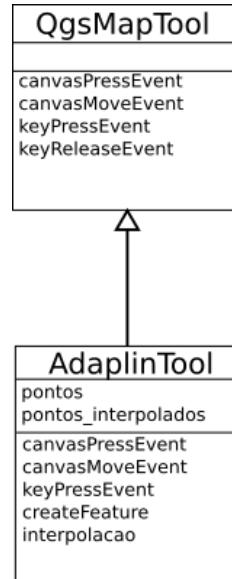
A seção a seguir apresenta a implementação da classe que modela o funcionamento prático da ferramenta.

4.1 A classe AdaplinTool

A classe AdaplinTool é uma subclasse da QgsMapTool, conforme está representado pela Figura 10. Alguns desses métodos precisam ser reescritos para prover funcionalidades específicas da extensão. Os principais métodos de classe¹ da AdaplinTool são discutidos

¹ Métodos ou funções membro, no contexto da programação orientada a objetos, são procedimentos genéricos definidos durante a definição de uma classe. A principal diferença entre uma função membro e uma função da programação estruturada reside no fato do método ter por definição acesso a todos os dados encapsulados por um objeto derivado desta classe.

Figura 10 - Diagrama de classes da classe AdaplinTool



a seguir.

O método `__init__` configura os atributos iniciais da classe. Dois atributos essenciais são as listas `pontos` e `pontosInterpolados`, que armazenam, respectivamente, os pontos marcados pelo usuário e os pontos resultantes da interpolação.

O método `canvasPressEvent` é elaborado para que um ponto seja adicionado toda vez que o botão esquerdo é pressionado. Além de adicionar o ponto, a polilinha é interpolada e atualizada. O botão direito é configurado para encerrar a feição quando pressionado. O método `createFeature` cria a feição que foi encerrada com o botão direito.

O método `canvasMoveEvent` é implementado de forma que, a cada vez que o usuário move o mouse sobre a tela do aplicativo, a otimização do caminho por programação dinâmica seja feita e a prévia da respectiva linha seja desenhada na tela. Esta funcionalidade é bastante útil para o operador, já que quando ele adiciona um novo ponto pressionando o botão esquerdo, já sabe de antemão como ficará a polilinha formada.

O método `interpolacao` recebe como entrada os pontos referentes ao último segmento. A inserção e interpolação de pontos é feita e o resultado é retornado pelo método.

O método `keyPressEvent` é configurado de forma que a ferramenta mude para o modo manual caso a tecla `Ctrl` seja pressionada. Ao pressionar outra vez a tecla `Ctrl`, a ferramenta volta ao modo semiautomático. A tecla `Esc` é usada para concluir a feição a partir do teclado, sendo uma opção ao botão direito. Para a eliminação do último segmento interpolado é utilizada a tecla `BACKSPACE`.

5 PROCEDIMENTO EXPERIMENTAL, RESULTADOS E ANÁLISE

5.1 Disponibilidade de imagens e software

As imagens de satélite disponíveis e utilizadas neste trabalho foram da constelação de satélites Rapideye (Planet Team, 2014). As mesmas foram adquiridas pelo Ministério do Meio Ambiente (MMA), visando o processo de regularização ambiental de propriedades e posses rurais através do Cadastro Ambiental Rural (CAR). Até 2014 foram adquiridas três coleções de cobertura nacional, referentes aos anos de 2011, 2012 e 2013. O contrato de aquisição foi assinado em 2012 prevendo a disponibilização de tais imagens para uso de todos os órgãos públicos brasileiros, dentre os quais o IBGE. Na presente pesquisa, as imagens foram utilizadas com a devida permissão do IBGE.

O software utilizado para construção da ferramenta foi o QGIS (QGIS Development Team, 2016) com sua biblioteca PyQGIS. Este software é um aplicativo de Sistema de Informação Geográfica – SIG e permite complementação de suas capacidades mediante a construção de extensões. No caso, a ferramenta Toolbox Adaplin, criada na presente pesquisa, foi desenvolvida como uma extensão do QGIS. Como se trata de software livre, licença GPL v2, se pôde utilizá-lo com liberdade, sem custo pela obtenção da licença de uso e com acesso pleno ao código fonte de outras extensões do QGIS.

Além de extensível, o QGIS é um software versátil. Na presente pesquisa, os programas utilizados na avaliação da exatidão das estradas extraídas pela Adaplin foram implementados em Python dentro do próprio QGIS.

A Tabela 1 sumariza os dados e software usados, cedentes, desenvolvedores e licenças.

Tabela 1 - Disponibilidade de software e dados de teste

Dado/Software	Cedente/Fornecedor	Licença
Imagens Rapideye	Instituto Brasileiro de Geografia e Estatística - IBGE	–
Aplicativo QGIS	Fundação OSGeo – Equipe de desenvolvimento do QGIS	GNU GPL v2

Fonte: o autor, 2017.

Tabela 2 - Informações técnicas sobre a constelação de satélites Rapideye

Característica	
Número de satélites	5
Lançamento	29/08/2008
Vida útil dos satélites	No mínimo 7 anos
Altitude da órbita	630 km em órbita heliosíncrona
Bandas espectrais	Azul (440-510 nm) Verde (520-590 nm) Vermelho (630-685 nm) Vermelho limítrofe (690-730 nm) Infravermelho próximo (760-850 nm)
Resolução espacial	5 m
Resolução radiométrica	16 bits por banda
Dimensões da imagem	5000 × 5000 px
Período de revisita	Diário para visada fora do nadir (<i>off-nadir</i>) 5,5 dias para visada no nadir

Fonte: (Planet Labs Team, 2016)

5.2 Base de Dados

5.2.1 Informações sobre as imagens Rapideye

As imagens Rapideye são provenientes de uma constelação de cinco satélites equipados com o mesmo sensor e que estão situados no mesmo plano orbital. Esta constelação foi lançada em agosto de 2008 desde a base de Baikonur, no Cazaquistão, e opera a uma altitude de 630 quilômetros. São cinco bandas captadas pelo sensor: azul, verde, vermelho, vermelho limítrofe e infravermelho próximo. A resolução espacial dessas bandas é de cinco metros. O nível de correção das imagens adquiridas é 3A (*RapidEye Ortho Product*), no qual são aplicadas a ortorretificação, junto com as correções radiométrica, geométrica e de terreno em uma projeção cartográfica (Planet Labs Team, 2016).

A Tabela 2 sumariza as principais características da constelação Rapideye e das imagens disponíveis. Um ponto a frisar é que as imagens são coletadas com 12 bits pelos satélites, entretanto são escalonadas e distribuídas com 16 bits.

5.2.2 Recortes Rapideye utilizados

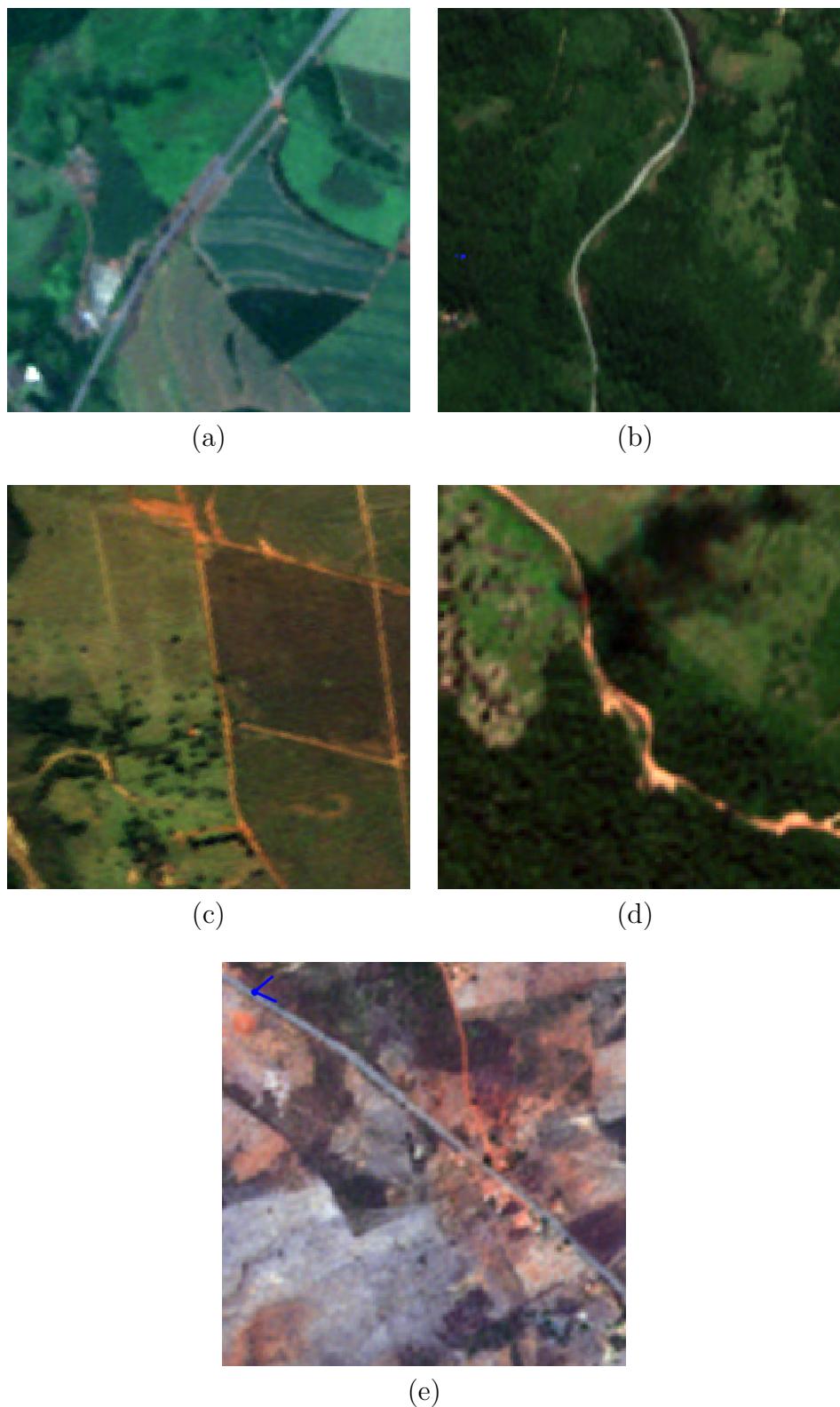
Os recortes de imagem Rapideye usados buscaram abranger certas estradas, tendo dimensões padronizadas de 4×4 km, o que equivale a 800×800 px em uma imagem Rapideye. Os recortes estão situados em área rural e foram divididos em cinco classes conforme o cenário que abrangem, cada classe com 20 recortes. A Tabela 3 enumera e descreve as classes de recortes utilizadas nos experimentos.

Uma parte dos recortes foi selecionado de forma aleatória e outra parte de forma manual. A parte aleatória consistiu em gerar uma grande quantidade de pontos aleatórios situados em trechos rodoviários oriundos da Base Cartográfica Contínua 1:250.000 fornecida pelo IBGE. Esses pontos gerados foram utilizados como localização do centro do recorte a ser extraído. Posteriormente, os recortes provenientes desse procedimento foram identificados como pertencentes a uma das classes. Como a maioria desses recortes eram referentes a estradas em leito natural, os recortes de estradas pavimentadas tiveram em grande parte que ser escolhidos manualmente de forma arbitrária.

Como há 20 recortes para cada classe, os recortes de cada classe são numerados sequencialmente de 1 a 20 e indicados por r_1, r_2, \dots, r_{20} . Sendo assim, para referir-se a um determinado recorte usa-se o número do recorte, dois pontos e a classe do recorte. Por exemplo, a notação r_7 : **PAV-RAS** se refere ao recorte r_7 da classe estradas pavimentadas cercadas por vegetação rasteira.

De todos os recortes disponíveis, foi selecionado um de cada classe para servir como base para uma análise qualitativa. O critério para a escolha dos exemplos neste subconjunto foi, exclusivamente, o aspecto visual, tendo sido selecionado um exemplo representativo para cada classe. A Figura 11 apresenta os recortes pertencentes a tal subconjunto.

Figura 11 - Exemplos das classes de recortes de imagens Rapideye presentes na base de dados



Legenda: (a) r_5 : PAV-RAS; (b) r_7 : PAV-FLO; (c) r_{10} : NAT-RAS; (d) r_3 : NAT-FLO; (e) r_5 : PAV-SOLO.

Fonte: O autor, 2017.

Tabela 3 - Descrição das classes de recortes de imagens Rapideye que compõem a base de dados

Classe	Nome completo	Descrição
PAV-RAS	Estrada pavimentada cercada por vegetação rasteira	Estradas de asfalto ou concreto cercadas por vegetação pouco desenvolvida
PAV-FLO	Estrada pavimentada cercada por vegetação rasteira e floresta	Estradas de asfalto ou concreto cercadas por áreas com grande densidade de árvores intercaladas por regiões de vegetação rasteira
NAT-RAS	Estrada em leito natural cercada por vegetação rasteira	Estradas de terra cercadas por vegetação pouco desenvolvida
NAT-FLO	Estrada em leito natural cercada por vegetação rasteira e floresta	Estradas de terra cercadas por áreas com grande densidade de árvores
PAV-SOLO	Estrada pavimentada cercada por solo exposto	Estradas de asfalto ou concreto cercadas por solo sem vegetação

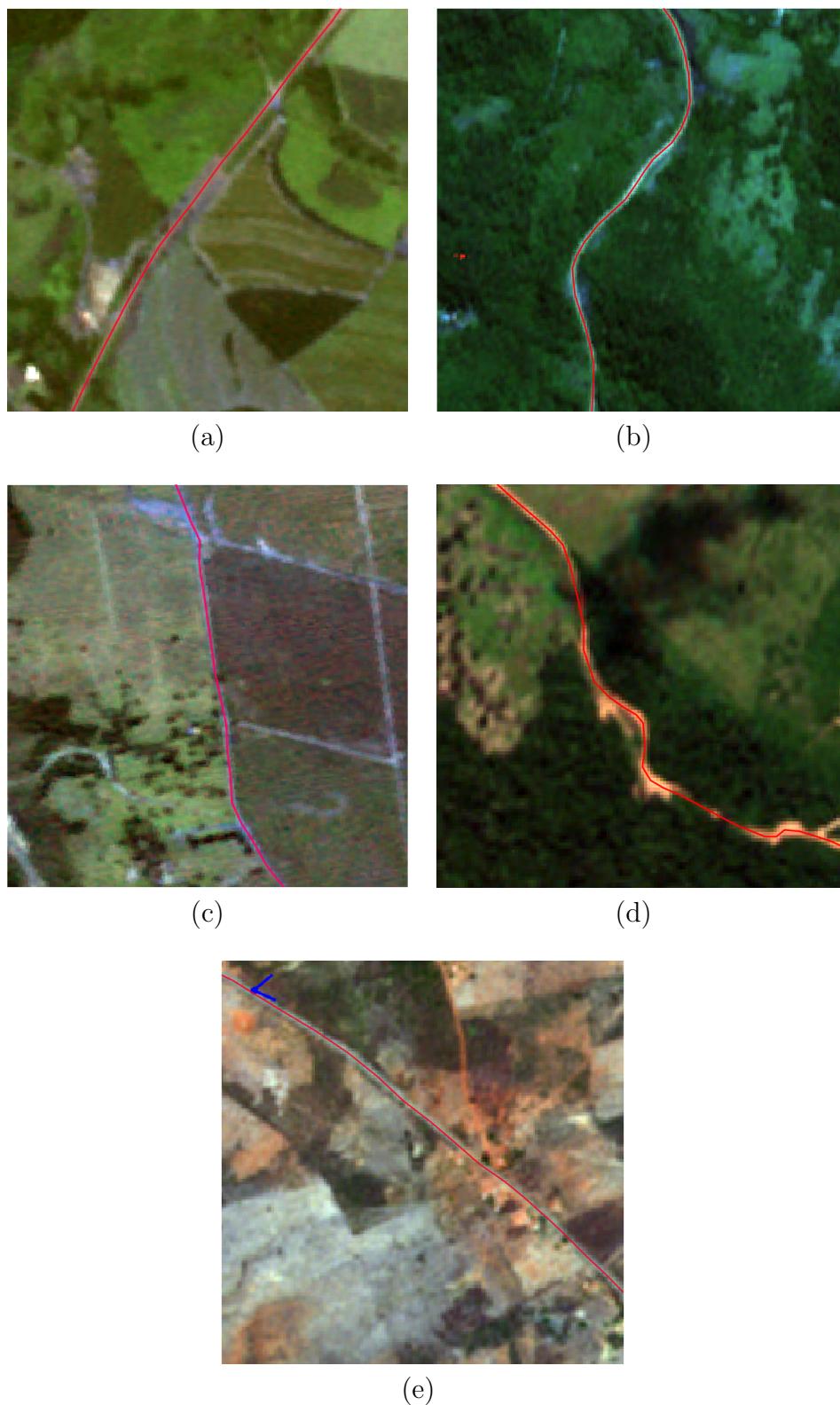
Fonte: O autor, 2017.

5.2.3 Dados de Referência

Antes da realização dos experimentos, o eixo central de cada estrada presente nos recortes foi manualmente delineado de forma cuidadosa. O zoom usado foi o maior possível a fim de permitir a estimativa precisa do eixo central da estrada. Sendo considerado como isento de erros, este eixo será tido como referência para análise da qualidade, sendo doravante denominado **eixo de referência**. A Tabela 4 apresenta o número de pontos do eixo de referência para cada recorte de uma classe da base de dados.

Por sua vez, a Figura 12 mostra o eixo de referência, desenhado em vermelho, para as amostras presentes na Figura 11.

Figura 12 - Eixos de referência dos recortes de imagens Rapideye apresentados na Figura 11



Legenda: (a) r_5 : PAV-RAS; (b) r_7 : PAV-FLO; (c) r_{10} : NAT-RAS; (d) r_3 : NAT-FLO;
(e) r_5 : PAV-SOLO.

Fonte: O autor, 2017.

Tabela 4 - Quantidade de vértices da polilinha do eixo de referência por recorte

Classe de recortes de imagens Rapideye na base de dados					
Recorte	PAV-RAS	PAV-FLO	NAT-RAS	NAT-FLO	PAV-SOLO
r_1	26	41	68	153	85
r_2	30	54	83	94	88
r_3	88	79	42	129	57
r_4	43	38	50	94	68
r_5	37	48	42	84	85
r_6	45	105	75	75	39
r_7	84	56	48	36	57
r_8	23	56	78	75	29
r_9	31	85	88	117	35
r_{10}	41	90	58	48	50
r_{11}	54	72	72	138	30
r_{12}	79	86	84	91	49
r_{13}	82	83	83	64	68
r_{14}	76	68	102	72	26
r_{15}	104	160	94	88	23
r_{16}	114	81	97	62	39
r_{17}	142	107	90	91	72
r_{18}	68	46	36	76	78
r_{19}	98	73	123	122	70
r_{20}	70	110	106	107	69

Fonte: O autor, 2017.

5.3 Parâmetros de análise da qualidade da extração das polilinhas

A análise da qualidade das polilinhas extraídas nos experimentos requer a comparação dos resultados obtidos com os respectivos eixos de referência. Faz-se necessária, portanto, uma medida da diferença entre polilinhas.

Uma métrica que pode ser usada para cálculo do afastamento entre duas linhas é a distância de Hausdorff. Dado que as duas linhas percorrem trajetos semelhantes, a distância de Hausdorff retorna o desvio máximo entre as linhas. Se as linhas são coincidentes, a distância de Hausdorff é zero. Entretanto, se elas são coincidentes em grande parte de sua extensão e possuem um pequeno pedaço com um grande afastamento, o afastamento máximo será tomado como a distância de Hausdorff.

Na sua versão discreta, a distância de Hausdorff considera uma polilinha como o conjunto de pontos que a constituem. Sendo dois conjuntos A e B referentes a duas polilinhas que representam o eixo de referência e o eixo extraído, a e b pontos pertencentes

a A e B respectivamente, a distância de Hausdorff, $d_H(A, B)$, é obtida pela Equação 22,

$$d_H(A, B) = \max \left\{ \max_{a \in A} \left\{ \min_{b \in B} d(a, b) \right\}, \max_{b \in B} \left\{ \min_{a \in A} d(b, a) \right\} \right\}, \quad (22)$$

onde, $d(a, b)$ é a distância euclidiana entre a e b . A equação 22 implica que para todo ponto $a \in A$ há um certo ponto $b \in B$ cuja distância euclidiana para a é mínima. Sendo assim, há um conjunto de distâncias mínimas de A para B , uma distância para cada ponto $a \in A$. O máximo desse conjunto de pontos é calculado. O mesmo procedimento é feito considerando as distâncias de pontos de B para A . Dessa forma, obtém-se dois máximos, e o máximo entre esses é a distância de Hausdorff.

Mozas (2008) propõe uma extensão da métrica de Hausdorff tomando a média e o desvio padrão das menores distâncias. Assim sendo, os dados são analisados de forma mais completa. A média dos deslocamentos, $\bar{d}_H(A, B)$, é representada pela Equação 23:

$$\bar{d}_H(A, B) = \max \left\{ \text{media}_{a \in A} \left\{ \min_{b \in B} d(a, b) \right\}, \text{media}_{b \in B} \left\{ \min_{a \in A} d(b, a) \right\} \right\}, \quad (23)$$

onde, a média das distâncias mínimas de A para B e de B para A são tomadas e o máximo entre elas é retornado. O cálculo do desvio padrão, $\sigma_H(A, B)$, segue procedimento análogo e está representado pela Equação 24:

$$\sigma_H(A, B) = \max \left\{ \sigma_{a \in A} \left\{ \min_{b \in B} d(a, b) \right\}, \sigma_{b \in B} \left\{ \min_{a \in A} d(b, a) \right\} \right\}. \quad (24)$$

Na presente dissertação, as medidas representadas pelas equações 22 a 24 são utilizadas para a avaliação da qualidade cartográfica do eixo extraído. Para o cálculo dessas equações, foi implementado um script Python no software QGIS. Inicialmente, o script substitui os pontos inicial e final do eixo extraído pelos pontos inicial e final do eixo de referência. Este passo possui a intenção de normalizar os limites percorridos pelas duas polilinhas, evitando a superestimação das medidas. Em seguida, considerando o adensamento como um requisito para que a forma de uma polilinha seja adequadamente aproximada pelo respectivo conjunto de pontos, o script realiza, em ambas as polilinhas em comparação, o aumento da densidade de pontos. Faz-se, no caso, para cada segmento das polilinhas, uma interpolação linear padronizada usando o intervalo de um metro.

5.4 Experimentos realizados e resultados obtidos

O conjunto principal de experimentos foi realizado com o intuito de comparar o método manual e o Toolbox Adaplin. Para tanto, ambos métodos foram empregados por um operador profissional cedido pelo IBGE na extração do eixo de cada estrada. Devido ao volume de experimentos, alguns cuidados foram tomados para assegurar a validade das

medições realizadas. Dentre as práticas empregadas, são destacadas:

- O operador foi orientado a vetorizar enquanto estivesse cômodo, de forma a não comprometer o trabalho os resultados alcançados pelo cansaço. Sob essa orientação, a produtividade média do operador foi de 20 recortes por dia e o trabalho completo foi escalonado durante duas semanas.
- O arquivo editado pelo operador ficou guardado em um banco de dados com suporte a acesso multiusuário. Isso foi feito para manter uma boa organização da atividade, possibilitando o acompanhamento do trabalho.
- Foi realizada toda a vetorização manual primeiro para depois se passar para o uso do Toolbox Adaplin. Isso foi realizado para que não houvesse uma confusão do operador no modo como funcionam as ferramentas.
- O funcionamento do Toolbox Adaplin foi explicado ao operador. Duas imagens de treino foram providenciadas para que o operador pudesse, antes de começar a vetorização, se habituar com o uso da ferramenta.
- As condições de trabalho foram verificadas de forma a certificar que o operador tivesse equipamentos em boas condições.
- Desabilitou-se o modo manual do Toolbox Adaplin durante os experimentos para que seja realizada a medição da capacidade da ferramenta em lidar com cenários complexos e desfavoráveis.

A avaliação dos resultados experimentais foi concebida a partir de três métricas principais: (1) exatidão da polilinha em comparação à de referência; (2) duração do procedimento de extração; e (3) esforço, sintetizado pelo número de pontos fornecidos pelo operador durante a restituição da feição. Nesta pesquisa, a qualidade do eixos delineados nos experimentos é representada pelos parâmetros d_H , $\overline{d_H}$ e σ_H descritos na seção anterior. Por sua vez, as métricas (2) e (3) estão ligadas à medição da complexidade do procedimento.

Em linha gerais, a avaliação realizada neste experimento visa à confirmação da hipótese a seguir.

Hipótese 1 *A ferramenta Toolbox Adaplin é capaz de reduzir a complexidade do procedimento de extração de estradas sem comprometer a exatidão das polilinhas obtidas.*

Para a extração manual, o *esforço* é igual ao número de pontos do eixo manual. No caso do eixo Adaplin, há que se considerar que a cada dois pontos marcados pelo operador, a ferramenta insere mais dois pontos sobre o segmento formado por esses dois pontos. Portanto, sendo n o número de pontos da polilinha, e n_p o esforço materializado

pelo número de pontos marcados pelo operador, é possível extrair a informação que $n = n_p + 2(n_p - 1)$, onde $(n_p - 1)$ é o total de segmentos oriundos dos n_p pontos assinalados pelo operador. Dessa equação, resulta que $n_p = (n + 2)/3$.

Ao final dos procedimentos anteriormente mencionados, foi realizado um experimento adicional. Seu intuito foi permitir a comparação dos resultados produzidos pelo Toolbox Adaplin com os fornecidos pela ferramenta semiautomática proprietária *Easy-trace* (ERDAS Inc., 2010). Trata-se de um módulo do software *Erdas Imagine* (Intergraph Corp., 2013) dedicado ao aumento da automação da extração de feições. Dentre as feições suportadas, encontra-se a classe estrada.

5.4.1 Experimentos com os recortes da classe PAV-RAS

Esta seção apresenta os parâmetros experimentais dos resultados obtidos para os recortes pertencentes à classe PAV-RAS com utilização do método manual e do Toolbox Adaplin. Uma breve comparação dos resultados para cada um dos métodos analisados é apresentada ao final.

5.4.1.1 Método manual

Os resultados da extração manual para os recortes da classe PAV-RAS estão representados pela Tabela 5. Na tabela, d_H é o deslocamento ou erro máximo, \bar{d}_H é o deslocamento ou erro médio, σ_H é o desvio padrão dos deslocamentos, a duração é dada em segundos e o esforço representa a quantidade de pontos marcados pelo operador para a vetorização da estrada presente no recorte. A média e o desvio padrão para os parâmetros de avaliação relativos a cada coluna são mostrados na parte inferior da tabela.

Tabela 5 - Resultados para a vetorização manual da classe PAV-RAS

Recorte	d_H (m)	\bar{d}_H (m)	σ_H (m)	Duração (s)	Esforço (#)
r_1	10,05	3,14	2,79	27,50	14
r_2	5,73	1,61	1,27	34,30	17
r_3	10,13	2,15	1,69	60,33	77
r_4	5,02	1,46	1,12	41,57	28
r_5	11,17	2,45	2,23	31,90	23
r_6	8,30	2,36	1,83	39,14	42
r_7	5,57	1,53	1,13	47,15	44
r_8	8,23	2,84	1,98	26,37	18
r_9	6,69	2,51	1,67	34,72	28
r_{10}	6,51	2,08	1,31	33,36	27
r_{11}	12,44	2,83	1,95	39,18	42
r_{12}	8,53	2,56	1,65	48,25	40
r_{13}	5,62	1,60	1,09	40,87	40
r_{14}	8,46	2,13	1,67	39,07	32
r_{15}	6,37	1,99	1,35	40,81	33
r_{16}	11,27	3,30	2,19	37,18	31
r_{17}	6,80	1,72	1,28	58,08	59
r_{18}	8,38	2,71	1,64	30,42	22
r_{19}	7,84	2,49	1,66	45,05	35
r_{20}	7,80	2,08	1,72	35,90	29
Média	8,05	2,28	1,66	39,56	34,05
Desvio Padrão	2,10	0,54	0,43	8,93	14,77

Fonte: O autor, 2017.

5.4.1.2 Extração Toolbox Adaplin

Os parâmetros relativos aos resultados da extração de estradas pelo Toolbox Adaplin para os recortes da classe PAV-RAS são fornecidos pela Tabela 6.

Tabela 6 - Resultados para a vetorização pelo Toolbox Adaplin da classe PAV-RAS

Recorte	d_H(m)	\bar{d}_H(m)	σ_H(m)	Duração(s)	Esforço(#)	Pontos(#)
r_1	14,37	3,90	3,27	20,95	11	31
r_2	9,14	2,76	1,85	30,25	17	49
r_3	15,64	2,71	2,44	66,28	39	115
r_4	23,26	5,75	5,58	22,95	14	40
r_5	15,48	3,60	3,29	21,77	12	34
r_6	11,64	3,15	2,32	24,00	16	46
r_7	7,63	2,28	1,54	26,97	17	49
r_8	15,92	4,64	4,56	20,56	9	25
r_9	12,86	3,18	2,41	22,75	13	37
r_{10}	6,10	1,56	1,21	29,68	15	43
r_{11}	18,41	3,29	2,78	34,94	21	61
r_{12}	17,32	3,07	2,70	52,02	29	85
r_{13}	16,59	3,83	3,39	32,23	18	52
r_{14}	13,48	2,35	2,23	24,21	16	46
r_{15}	11,61	1,93	1,62	28,04	15	43
r_{16}	16,71	2,68	2,48	31,80	17	49
r_{17}	19,45	3,06	2,92	43,41	23	67
r_{18}	12,85	2,95	2,52	24,35	13	37
r_{19}	19,49	5,24	4,07	38,02	20	58
r_{20}	18,68	4,97	3,68	33,30	20	58
Média	14,83	3,34	2,84	31,42	17,75	51,25
Desvio Padrão	4,28	1,11	1,06	11,48	6,74	20,23

Fonte: O autor, 2017.

5.4.1.3 Comparação dos resultados

Os resultados para os recortes da classe PAV-RAS serão analisados com o objetivo de saber, em termos dos parâmetros avaliados, quantas vezes ou em quantos recortes a extração pelo Toolbox Adaplin é superior à manual ou vice-versa. A Tabela 7 ilustra qual método obteve melhor resultado para cada recorte.

Para a distância d_H referente ao desvio máximo entre as polilinhas, a extração manual apresentou melhores resultados para 19 dos 20 recortes ou 95% dos recortes. Para a distância \bar{d}_H referente ao desvio médio entre as polilinhas, a extração manual teve um erro médio menor que a do *toolbox* em 17 dos 20 ou 85% dos recortes. Para o desvio padrão dos erros, referido por σ_H , houve uma menor dispersão dos erros da extração manual em 19 dos 20 recortes, o que representa 95% dos recortes.

A média de d_H para a extração manual foi de 8,1 metros para a extração manual e de 14,8 metros para a extração pelo Toolbox Adaplin, o que indica um aumento médio

Tabela 7 - Método vitorioso em cada quesito analisado para a classe PAV-RAS

Recorte	d_H	\bar{d}_H	σ_H	Duração	Esforço
r_1	m	m	m	a	a
r_2	m	m	m	a	-
r_3	m	m	m	m	a
r_4	m	m	m	a	a
r_5	m	m	m	a	a
r_6	m	m	m	a	a
r_7	m	m	m	a	a
r_8	m	m	m	a	a
r_9	m	m	m	a	a
r_{10}	a	a	a	a	a
r_{11}	m	m	m	a	a
r_{12}	m	m	m	m	a
r_{13}	m	m	m	a	a
r_{14}	m	m	m	a	a
r_{15}	m	a	m	a	a
r_{16}	m	a	m	a	a
r_{17}	m	m	m	a	a
r_{18}	m	m	m	a	a
r_{19}	m	m	m	a	a
r_{20}	m	m	m	a	a
Manual	19	17	19	2	0
Adaplin	1	3	1	18	19

Legenda: m - Manual; a - Adaplin.

Fonte: O autor, 2017.

de 6,7 metros no desvio máximo entre o eixo extraído e o eixo de referência. O parâmetro \bar{d}_H é mais representativo do erro do eixo extraído, pois toma a média dos erros ao longo da polilinha. A média de \bar{d}_H foi de 2,3 m para a extração manual e 3,3 m para a extração pelo *toolbox*, o que indica o aumento médio de 1 metro nesse parâmetro de avaliação. O método manual teve média de σ_H de 1,7 m, enquanto o método de extração pelo *toolbox* teve valor de 2,8 m para o mesmo parâmetro, o que indica um aumento médio de 1,1 m na dispersão do erro no eixo extraído.

Para o quesito duração, os resultados são diferentes. Nestes a extração pelo *Toolbox* Adaplin apresentou melhores resultados com um tempo menor em 18 dos 20 ou 90% dos recortes. No tocante ao esforço, em 19 dos 20 recortes, o *toolbox* possibilitou sua diminuição.

O somatório do tempo gasto em toda a vetorização foi de 13 minutos e 11 segundos para a extração manual e de 10 minutos e 28 segundos para a extração pelo *Toolbox*

Adaplin, o que possibilitou uma economia de tempo de 2 minutos e 43 segundos para a vetorização das estradas presentes nesse conjunto de recortes. O somatório do esforço também é menor no *toolbox*, o qual é de 681 pontos medidos para a extração manual e 355 para o *toolbox*, o que representa uma poupança de 326 e, portanto, menos energia gasta pelo operador.

5.4.2 Experimentos com a classe PAV-FLO

Esta seção apresenta os parâmetros experimentais dos resultados obtidos para os recortes pertencentes à classe PAV-FLO com a utilização do método manual e do Toolbox Adaplin. Uma breve comparação dos resultados para cada um dos métodos analisados é apresentada ao final.

5.4.2.1 Extração manual

A Tabela 8 exibe os parâmetros obtidos pelos resultados fornecidos pela extração manual para os recortes da classe PAV-FLO.

Tabela 8 - Resultados para a vetorização manual da classe PAV-FLO

Recorte	d_H (m)	\bar{d}_H (m)	σ_H (m)	Duração (s)	Esforço (#)
r_1	7,28	2,64	1,58	45,12	29
r_2	4,33	1,36	0,93	38,17	43
r_3	9,79	2,30	1,82	60,08	58
r_4	12,06	2,38	1,98	37,12	30
r_5	6,60	2,04	1,47	25,94	32
r_6	4,87	1,31	0,97	60,39	80
r_7	6,58	1,87	1,41	50,01	50
r_8	5,92	1,60	0,94	44,85	45
r_9	6,87	1,55	1,28	55,09	72
r_{10}	4,93	1,49	0,97	43,50	60
r_{11}	6,67	1,87	1,33	58,50	57
r_{12}	6,15	1,96	1,27	48,06	49
r_{13}	5,65	1,31	0,88	60,28	74
r_{14}	5,62	1,32	1,04	60,39	56
r_{15}	6,06	1,38	0,98	92,06	140
r_{16}	6,05	2,19	1,38	58,76	55
r_{17}	9,29	2,55	1,71	60,15	68
r_{18}	3,08	0,85	0,61	38,19	38
r_{19}	5,32	1,30	0,89	50,18	68
r_{20}	6,48	1,62	1,18	50,37	85
Média	6,48	1,74	1,23	51,86	59,45
Desvio Padrão	1,99	0,49	0,36	13,61	24,93

Fonte: O autor, 2017.

5.4.2.2 Extração Toolbox Adaplin

Os parâmetros dos resultados experimentais fornecidos pelo Toolbox Adaplin para os recortes da classe PAV-FLO são apresentados na Tabela 9.

Tabela 9 - Resultados para a vetorização pelo Toolbox Adaplin da classe PAV-FLO

Recorte	d_H (m)	\bar{d}_H (m)	σ_H (m)	Duração (s)	Esforço (#)	Pontos(#)
r_1	12,23	3,05	2,49	44,71	18	52
r_2	13,80	2,53	2,32	26,97	12	34
r_3	17,65	2,45	2,55	55,28	29	85
r_4	17,87	4,07	3,84	36,80	19	55
r_5	14,23	2,95	2,74	23,16	12	34
r_6	10,64	2,60	1,89	57,02	20	58
r_7	6,69	2,00	1,35	46,17	22	64
r_8	16,99	3,73	4,15	30,92	19	55
r_9	14,98	3,01	3,01	46,54	30	88
r_{10}	18,12	3,53	3,21	39,43	27	79
r_{11}	12,61	3,09	2,29	56,30	31	91
r_{12}	13,15	2,42	2,28	37,91	23	67
r_{13}	19,50	3,36	3,78	54,03	34	100
r_{14}	24,07	4,14	4,68	58,77	31	91
r_{15}	19,26	3,47	3,15	85,90	50	148
r_{16}	19,80	3,63	3,78	51,22	32	94
r_{17}	19,57	4,66	4,63	55,31	28	82
r_{18}	8,55	1,58	1,27	35,81	16	46
r_{19}	14,74	2,59	2,14	41,37	29	85
r_{20}	14,02	2,89	2,73	44,91	30	88
Média	15,42	3,09	2,91	46,43	25,60	74,80
Desv. Pad.	4,23	0,76	0,99	13,91	8,86	26,59

Fonte: O autor, 2017.

5.4.2.3 Comparação dos resultados

Como feito no caso anterior, os resultados para a classe PAV-FLO serão analisados com o objetivo de saber, para os parâmetros de análise da qualidade dados, quantas vezes ou em quantos recortes a extração com o Toolbox Adaplin é superior à manual ou vice-versa. A Tabela 10 ilustra qual método obteve melhor resultado para cada recorte.

A extração manual apresentou melhores resultados para o deslocamento máximo d_H e para o deslocamento médio \bar{d}_H em todos os recortes. Para o desvio padrão σ_H , a extração manual apresentou uma dispersão menor em 19 dos 20 ou 95% dos recortes.

A média dos desvios máximos representados por d_H foi 6,5 m para a extração manual e 15,4 m para a extração pelo Toolbox Adaplin, o que aponta um acréscimo médio de 8,9 m. Para o parâmetro \bar{d}_H , que computa a média dos deslocamentos entre o eixo extraído e o eixo de referência, a média dos recortes foi de 1,7 m para a extração manual e 3,1 m para a extração pelo toolbox, o que indica uma diferença de 1,4 m. A

Tabela 10 - Método vitorioso em cada quesito analisado para a classe PAV-FLO

Recorte	d_H	\bar{d}_H	σ_H	Duração	Esforço
r_1	m	m	m	a	a
r_2	m	m	m	a	a
r_3	m	m	m	a	a
r_4	m	m	m	a	a
r_5	m	m	m	a	a
r_6	m	m	m	a	a
r_7	m	m	a	a	a
r_8	m	m	m	a	a
r_9	m	m	m	a	a
r_{10}	m	m	m	a	a
r_{11}	m	m	m	a	a
r_{12}	m	m	m	a	a
r_{13}	m	m	m	a	a
r_{14}	m	m	m	a	a
r_{15}	m	m	m	a	a
r_{16}	m	m	m	a	a
r_{17}	m	m	m	a	a
r_{18}	m	m	m	a	a
r_{19}	m	m	m	a	a
r_{20}	m	m	m	a	a
Manual	20	20	19	0	0
Adaplin	0	0	1	20	20

Legenda: m - Manual; a - Adaplin.

Fonte: O autor, 2017.

média de σ_H foi 1,2 m para a extração manual e 2,9 m para a extração pelo *toolbox*, o que retrata uma diferença média de 1,7 m na dispersão dos erros nos recortes.

A extração pelo Toolbox Adaplin, por sua vez, apresentou duração e esforço menores em todos recortes.

O tempo total usado na vetorização manual foi 17 minutos e 17 segundos para a extração manual e 15 minutos e 29 segundos para a extração pelo Toolbox Adaplin, o que representa uma poupança de 1 minuto e 48 segundos. O somatório do esforço foi 1189 pontos medidos durante a extração manual e 512 na extração pelo *toolbox*, o que aponta uma economia de 677.

Tabela 11 - Resultados para a vetorização manual da classe NAT-RAS

Recorte	d_H (m)	\bar{d}_H (m)	σ_H (m)	Duração (s)	Esforço (#)
r_1	5,10	1,28	0,82	53,37	57
r_2	9,77	1,83	1,68	63,26	70
r_3	5,11	1,76	1,26	44,58	22
r_4	6,19	1,64	1,25	44,32	21
r_5	6,54	1,86	1,45	40,66	28
r_6	2,61	0,79	0,54	63,01	63
r_7	5,59	1,77	1,11	47,90	31
r_8	4,95	1,11	0,78	66,49	74
r_9	5,03	1,48	0,93	50,15	30
r_{10}	6,90	2,16	1,63	34,71	34
r_{11}	6,90	1,68	1,30	60,30	39
r_{12}	6,25	1,48	1,29	60,24	62
r_{13}	6,61	1,42	1,00	43,14	48
r_{14}	6,01	1,43	1,02	60,20	60
r_{15}	5,34	1,51	1,07	43,12	49
r_{16}	3,65	0,98	0,61	31,97	53
r_{17}	6,78	2,07	1,42	37,04	50
r_{18}	5,01	1,24	0,77	31,39	26
r_{19}	5,43	1,46	1,05	60,55	95
r_{20}	5,20	0,98	0,67	41,50	69
Média	5,75	1,50	1,08	48,90	49,05
Desvio Padrão	1,44	0,36	0,33	11,29	20,06

Fonte: O autor, 2017.

5.4.3 Experimentos com a classe NAT-RAS

Esta seção apresenta os parâmetros experimentais dos resultados obtidos para os recortes pertencentes à classe NAT-RAS com utilização do método manual e do Toolbox Adaplin. Uma breve comparação dos resultados para cada um dos métodos analisados é apresentada ao final.

5.4.3.1 Extração manual

Os parâmetros relativos aos resultados da extração manual para os recortes pertencentes à classe NAT-RAS são exibidos na Tabela 11.

Tabela 12 - Resultados para a vetorização pelo Toolbox Adaplin da classe NAT-RAS

Recorte	d_H (m)	\bar{d}_H (m)	σ_H (m)	Duração (s)	Esforço (#)	Pontos(#)
r_1	18,02	2,34	2,34	48,83	20	58
r_2	9,69	2,53	1,87	49,76	24	70
r_3	12,61	2,05	1,80	36,15	11	31
r_4	7,07	1,66	1,08	41,11	12	34
r_5	12,77	3,07	2,36	38,28	15	43
r_6	14,71	2,71	2,31	52,05	28	82
r_7	10,26	2,46	1,67	40,68	10	28
r_8	10,31	2,34	1,75	60,04	21	61
r_9	7,84	1,57	1,11	53,38	27	79
r_{10}	12,60	2,19	2,11	34,01	8	22
r_{11}	17,61	2,94	3,10	49,31	15	43
r_{12}	19,21	6,24	5,12	47,72	13	37
r_{13}	8,05	2,65	1,85	39,41	7	19
r_{14}	11,82	2,42	1,89	56,47	17	49
r_{15}	8,36	2,08	1,50	36,71	19	55
r_{16}	14,11	2,99	2,92	30,17	18	52
r_{17}	16,55	2,96	2,76	34,43	14	40
r_{18}	9,18	2,18	1,71	26,82	10	28
r_{19}	16,27	2,68	2,48	50,27	35	103
r_{20}	21,78	3,22	3,35	34,35	15	43
Média	12,94	2,66	2,25	43,00	16,95	48,85
Desv. Pad.	4,23	0,95	0,91	9,30	7,26	21,77

Fonte: O autor, 2017.

5.4.3.2 Extração Toolbox Adaplin

Os parâmetros relativos aos resultados da extração empregando o Toolbox Adaplin para os recortes pertencentes à classe NAT-RAS são exibidos na Tabela 12.

5.4.3.3 Comparação dos resultados

O mesmo procedimento de comparação feito para as classes anteriores será aplicado à classe NAT-RAS. A Tabela 13 explicita qual método obteve melhor resultado para cada recorte da classe NAT-RAS.

A extração manual exibiu melhores resultados para d_H e σ_H em 19 dos 20 ou 95% recortes, enquanto apontou desvios médios \bar{d}_H menores em todos os recortes.

A média de d_H foi 5,8 metros para a extração manual e 12,9 metros para a extração

Tabela 13 - Método vitorioso em cada quesito analisado para a classe NAT-RAS

Recorte	d_H	\bar{d}_H	σ_H	Duração	Esforço
r_1	m	m	m	a	a
r_2	a	m	m	a	a
r_3	m	m	m	a	a
r_4	m	m	a	a	a
r_5	m	m	m	a	a
r_6	m	m	m	a	a
r_7	m	m	m	a	a
r_8	m	m	m	a	a
r_9	m	m	m	m	a
r_{10}	m	m	m	a	a
r_{11}	m	m	m	a	a
r_{12}	m	m	m	a	a
r_{13}	m	m	m	a	a
r_{14}	m	m	m	a	a
r_{15}	m	m	m	a	a
r_{16}	m	m	m	a	a
r_{17}	m	m	m	a	a
r_{18}	m	m	m	a	a
r_{19}	m	m	m	a	a
r_{20}	m	m	m	a	a
Manual	19	20	19	1	0
Adaplin	1	0	1	19	20

Legenda: m - Manual; a - Adaplin.

Fonte: O autor, 2017.

pelo Toolbox Adaplin, o que assinala um aumento médio de 7,1 metros no desvio máximo entre o eixo extraído e o eixo de referência. Para o desvio médio entre o eixo extraído e o de referência representado pelo parâmetro \bar{d}_H , a média dos recortes foi 1,5 m para a extração manual e 2,7 m para a extração pelo *toolbox*, o que indica o aumento médio de 1,2 m. O método manual teve média de σ_H igual a 1,1 m, enquanto o método de extração pelo *toolbox* teve valor de 2,3 m para o mesmo parâmetro, apontando um aumento médio de 1,2 m na dispersão do erro no eixo extraído.

A extração pelo Toolbox Adaplin apresentou duração menor em 19 dos 20 recortes, enquanto exibiu esforço menor em todos os recortes.

O somatório das durações de todos os recortes da classe indica o tempo total gasto para a restituição na classe. Esse tempo foi equivalente a 16 minutos e 18 segundos para a extração manual e 14 minutos e 20 segundos para a extração pelo Toolbox Adaplin, apontando uma economia de 1 minuto e 58 segundos para a vetorização das estradas

Tabela 14 - Resultados para a vetorização manual da classe NAT-FLO

Recorte	d_H (m)	\bar{d}_H (m)	σ_H (m)	Duração (s)	Esforço (#)
r_1	4,84	1,13	0,78	86,51	137
r_2	6,20	1,91	1,46	39,90	32
r_3	7,87	1,48	1,09	60,35	81
r_4	5,20	1,18	0,98	41,42	65
r_5	4,29	1,26	0,90	60,32	80
r_6	8,14	1,63	1,28	49,64	59
r_7	5,17	1,61	1,15	27,10	23
r_8	5,25	1,86	1,17	44,92	63
r_9	4,97	1,22	0,89	56,86	75
r_{10}	6,79	2,00	1,62	28,22	30
r_{11}	6,09	1,31	0,94	86,05	111
r_{12}	4,93	1,40	0,89	44,12	55
r_{13}	4,04	1,44	0,96	23,30	30
r_{14}	11,74	1,89	2,03	49,71	48
r_{15}	3,93	1,15	0,83	54,21	65
r_{16}	10,30	2,06	1,69	44,99	39
r_{17}	11,95	1,95	1,53	60,31	71
r_{18}	4,52	1,32	0,92	22,08	51
r_{19}	9,13	1,70	1,22	74,01	91
r_{20}	3,66	1,14	0,72	59,73	91
Média	6,45	1,53	1,15	50,69	64,85
Desvio Padrão	2,57	0,32	0,35	18,40	28,90

Fonte: O autor, 2017.

presentes nos recortes da classe. O somatório do esforço é igual a 981 pontos medidos durante a extração manual e 339 para o *toolbox*, indicando uma poupança de 642 e menos energia despendida pelo operador.

5.4.4 Experimentos com a classe NAT-FLO

Esta seção apresenta os parâmetros experimentais dos resultados obtidos para os recortes pertencentes à classe NAT-FLO com utilização do método manual e do Toolbox Adaplin. Uma breve comparação dos resultados para cada um dos métodos analisados é apresentada ao final.

Tabela 15 - Resultados para a vetorização pelo Toolbox Adaplin da classe NAT-FLO

Recorte	d_H (m)	\bar{d}_H (m)	σ_H (m)	Duração (s)	Esforço (#)	Pontos(#)
r_1	14,98	2,40	1,96	76,34	58	172
r_2	13,76	2,64	2,54	34,41	22	64
r_3	10,71	2,25	1,92	50,85	31	91
r_4	19,17	2,92	3,01	36,02	20	58
r_5	22,69	3,01	3,47	54,82	30	88
r_6	12,78	2,07	1,95	41,08	24	70
r_7	12,07	1,94	1,77	22,57	10	28
r_8	10,28	3,46	2,28	42,82	25	73
r_9	18,73	2,30	2,31	51,19	37	109
r_{10}	10,51	2,08	1,95	23,34	11	31
r_{11}	15,90	2,50	2,02	64,60	38	112
r_{12}	11,31	2,26	1,93	38,67	16	46
r_{13}	6,81	1,79	1,40	19,03	9	25
r_{14}	15,30	2,95	2,68	39,23	20	58
r_{15}	14,71	3,13	2,62	41,51	22	64
r_{16}	14,61	2,43	2,12	30,80	21	61
r_{17}	11,66	2,54	2,04	58,50	28	82
r_{18}	11,77	2,39	1,63	16,04	8	22
r_{19}	13,53	2,26	1,86	57,53	40	118
r_{20}	13,63	2,74	2,44	56,34	33	97
Média	13,75	2,50	2,20	42,78	25,15	73,45
Desv. Pad.	3,57	0,43	0,49	16,04	12,34	37,02

Fonte: O autor, 2017.

5.4.4.1 Extração manual

Os parâmetros observados nos experimentos com uso do método manual para os recortes pertencentes à classe NAT-FLO são apresentados na Tabela 14.

5.4.4.2 Extração Toolbox Adaplin

Os parâmetros observados nos experimentos com uso da ferramenta Toolbox Adaplin para os recortes pertencentes à classe NAT-FLO são apresentados na Tabela 15.

5.4.4.3 Comparação dos resultados

O procedimento de comparação da classe NAT-FLO será idêntico ao realizado para as classes anteriores. A Tabela 16 ilustra qual método obteve melhor resultado para cada recorte.

A extração manual mostrou resultados melhores em todos os recortes para \bar{d}_H e σ_H , enquanto apresentou desvios máximos d_H menores em 19 dos 20 ou 95% dos recortes.

A média dos desvios máximos representados por d_H foi 6,5 m para a extração manual e 13,8 m para a extração pelo Toolbox Adaplin, o que aponta um acréscimo médio de 7,3 m no deslocamento máximo entre o eixo extraído e o eixo de referência. Para o parâmetro \bar{d}_H , que computa a média dos deslocamentos entre o eixo extraído e o eixo de referência, a média dos recortes foi equivalente a 1,5 m para a extração manual e 2,5 m para a extração pelo *toolbox*, o que indica uma diferença de 1 m. A média de σ_H foi 1,2 m para a extração manual e 2,2 m para a extração pelo *toolbox*, o que também retrata uma diferença média de 1 m na dispersão dos erros nos recortes.

A extração pelo Toolbox Adaplin apontou melhores resultados para a duração e o esforço em todos os recortes.

O tempo total despendido na vetorização manual foi 16 minutos e 54 segundos para a extração manual, enquanto o cômputo para a extração pelo Toolbox Adaplin foi de 14 minutos e 16 segundos, o que indica um racionamento de 2 minutos e 38 segundos. O somatório do esforço foi 1297 para a extração manual e 503 para a extração pelo *toolbox*, o que exprime uma economia de 794.

Tabela 16 - Método vitorioso em cada quesito analisado para a classe NAT-FLO

Recorte	d_H	\bar{d}_H	σ_H	Duração	Esforço
r_1	m	m	m	a	a
r_2	m	m	m	a	a
r_3	m	m	m	a	a
r_4	m	m	m	a	a
r_5	m	m	m	a	a
r_6	m	m	m	a	a
r_7	m	m	m	a	a
r_8	m	m	m	a	a
r_9	m	m	m	a	a
r_{10}	m	m	m	a	a
r_{11}	m	m	m	a	a
r_{12}	m	m	m	a	a
r_{13}	m	m	m	a	a
r_{14}	m	m	m	a	a
r_{15}	m	m	m	a	a
r_{16}	m	m	m	a	a
r_{17}	a	m	m	a	a
r_{18}	m	m	m	a	a
r_{19}	m	m	m	a	a
r_{20}	m	m	m	a	a
Manual	19	20	20	0	0
Adaplin	1	0	0	20	20

Legenda: m - Manual; a - Adaplin.

Fonte: O autor, 2017.

5.4.5 Experimentos com a classe PAV-SOLO

Esta seção apresenta os parâmetros experimentais dos resultados obtidos para os recortes pertencentes à classe PAV-SOLO com utilização do método manual e do Toolbox Adaplin. Uma breve comparação dos resultados para cada um dos métodos analisados é apresentada ao final.

5.4.5.1 Extração manual

Os parâmetros observados nos experimentos realizados com o método manual para os recortes pertencentes à classe PAV-SOLO são apresentados na Tabela 17.

Tabela 17 - Resultados para a vetorização manual da classe PAV-SOLO

Recorte	d_H (m)	\bar{d}_H (m)	σ_H (m)	Duração (s)	Esforço (#)
r_1	5,14	1,60	1,07	33,95	31
r_2	7,65	1,90	1,39	32,05	37
r_3	8,30	3,19	2,30	32,26	11
r_4	7,02	2,12	1,53	26,76	13
r_5	6,24	1,56	1,04	50,06	49
r_6	10,05	3,73	2,66	30,23	13
r_7	8,22	2,84	2,22	26,89	15
r_8	9,41	1,52	1,25	30,17	9
r_9	10,74	2,11	1,69	36,97	16
r_{10}	10,29	2,77	2,14	20,90	11
r_{11}	11,59	6,34	2,95	24,15	9
r_{12}	8,26	3,00	1,85	25,54	15
r_{13}	15,09	4,86	4,34	33,97	13
r_{14}	10,95	3,94	2,85	18,59	8
r_{15}	6,84	2,34	2,18	26,21	8
r_{16}	4,92	1,71	1,21	22,97	10
r_{17}	3,66	0,99	0,64	31,40	33
r_{18}	7,02	2,09	1,65	35,75	28
r_{19}	5,38	1,62	1,26	30,46	12
r_{20}	5,16	1,66	1,02	33,78	20
Média	8,10	2,59	1,86	30,15	18,05
Desvio Padrão	2,80	1,31	0,87	6,85	11,42

Fonte: O autor, 2017.

5.4.5.2 Extração Toolbox Adaplin

Os resultados alcançados pela vetorização através da ferramenta Toolbox Adaplin para os recortes da classe PAV-SOLO são apresentados na Tabela 18.

Tabela 18 - Resultados para a vetorização pelo Toolbox Adaplin da classe PAV-SOLO

Recorte	d_H (m)	\bar{d}_H (m)	σ_H (m)	Duração (s)	Esforço (#)	Pontos(#)
r_1	22,44	8,12	5,31	34,17	19	55
r_2	25,79	6,66	5,90	41,12	22	64
r_3	22,17	5,52	4,79	23,49	12	34
r_4	18,51	6,19	5,17	25,64	6	16
r_5	19,96	4,61	4,47	52,38	24	70
r_6	17,80	7,48	4,55	43,65	15	43
r_7	22,56	8,05	6,10	28,75	11	31
r_8	19,99	9,97	5,77	38,47	10	28
r_9	22,05	6,75	4,97	37,38	14	40
r_{10}	14,03	4,09	3,05	26,38	6	16
r_{11}	28,45	7,99	7,94	29,94	7	19
r_{12}	16,40	6,84	4,12	28,95	13	37
r_{13}	21,03	8,45	4,81	34,77	15	43
r_{14}	18,84	9,40	5,37	24,48	9	25
r_{15}	24,51	9,44	4,74	17,96	8	22
r_{16}	9,05	1,77	1,40	25,19	5	13
r_{17}	7,40	2,71	1,71	21,15	8	22
r_{18}	21,48	8,65	5,05	61,46	17	49
r_{19}	20,38	9,70	5,32	27,30	12	34
r_{20}	12,54	3,08	2,39	42,42	10	28
Média	19,27	6,77	4,65	33,25	12,15	34,45
Desv. Pad.	5,29	2,43	1,54	10,93	5,31	15,94

Fonte: O autor, 2017.

5.4.5.3 Comparação dos resultados

O mesmo procedimento de comparação feito para as classes anteriores será aplicado à classe PAV-SOLO. A Tabela 19 ilustra qual método obteve melhor resultado para cada recorte.

A extração manual mostrou melhores resultados para d_H , \bar{d}_H e σ_H para todos os recortes. A duração também foi menor nesta classe para 15 dos 20 ou 75% dos recortes.

Para a média do desvio máximo d_H entre o eixo extraído e o eixo de referência, a extração manual conseguiu 8,1 metros e a extração pelo Toolbox Adaplin obteve 19,3 metros, o que aponta um aumento médio de 11,2 metros. A média de \bar{d}_H apresentou resultado de 2,6 m para a extração manual e 6,8 m para a extração pelo toolbox, o que indica o acréscimo médio de 4,2 metros. A média da dispersão dos desvios σ_H entre o eixo extraído e o de referência foi 1,9 m para a extração manual e 4,7 m para a extração pelo toolbox, o que configura um aumento médio de 2,8 m.

Tabela 19 - Método vitorioso em cada quesito analisado para a classe PAV-SOLO

Recorte	d_H	\bar{d}_H	σ_H	Duração	Esforço
r_1	m	m	m	m	a
r_2	m	m	m	m	a
r_3	m	m	m	a	m
r_4	m	m	m	a	a
r_5	m	m	m	m	a
r_6	m	m	m	m	m
r_7	m	m	m	m	a
r_8	m	m	m	m	m
r_9	m	m	m	m	a
r_{10}	m	m	m	m	a
r_{11}	m	m	m	m	a
r_{12}	m	m	m	m	a
r_{13}	m	m	m	m	m
r_{14}	m	m	m	m	m
r_{15}	m	m	m	a	-
r_{16}	m	m	m	m	a
r_{17}	m	m	m	a	a
r_{18}	m	m	m	m	a
r_{19}	m	m	m	a	-
r_{20}	m	m	m	m	a
Manual	20	20	20	15	5
Adaplin	0	0	0	5	13

Legenda: m - Manual; **a** - Adaplin.

Fonte: O autor, 2017.

O único parâmetro de análise que a extração pelo Toolbox Adaplin apresentou melhores resultados foi no esforço, onde teve menos pontos marcados em 13 dos 20 ou 65% dos recortes.

O tempo integral gasto na vetorização manual foi 10 minutos e 3 segundos para a extração manual e 11 minutos e 5 segundos para a extração pelo Toolbox Adaplin, o que representa um acréscimo de 1 minuto e 2 segundos com o uso do *toolbox*. O somatório do esforço foi 361 para a extração manual e 243 para a extração pelo *toolbox*, o que aponta uma economia de 118.

Tabela 20 - Resultados produzidos pelas ferramentas Erdas Easytrace e Toolbox Adaplin

Erdas Easytrace						
Classe	Recorte	d_H (m)	$\overline{d_H}$ (m)	σ_{H} (m)	Duração(s)	Esforço (#)
PAV-RAS	r_5	17,46	3,78	3,55	24,93	8
PAV-FLO	r_7	11,33	2,63	2,01	26,12	8
NAT-RAS	r_{10}	10,88	2,22	2,03	26,45	7
NAT-FLO	r_3	13,72	1,65	1,50	42,67	21
PAV-SOLO	r_5	13,91	2,97	2,29	50,84	18

Toolbox Adaplin						
Classe	Recorte	d_H (m)	$\overline{d_H}$ (m)	σ_{H} (m)	Duração(s)	Esforço (#)
PAV-RAS	r_5	15,48	3,60	3,29	21,77	12
PAV-FLO	r_7	6,69	2,00	1,35	46,17	22
NAT-RAS	r_{10}	12,60	2,19	2,11	34,01	8
NAT-FLO	r_3	10,71	2,25	1,92	50,85	31
PAV-SOLO	r_5	19,96	4,61	4,47	52,38	24

5.4.6 Experimentos com a ferramenta Erdas Easytrace

Os experimentos com a ferramenta Erdas Easytrace (ERDAS Inc., 2010) foram feitos para os recortes dos exemplos mostrados na Figura 11. Como o Easytrace possui outras funcionalidades além da extração do eixo central de uma estrada, foi necessário especificar o modo *Centerline*. Outra opção usada foi o *Rubberband*, onde a polilinha é desenhada na tela ao mover o mouse, tal como acontece no Toolbox Adaplin. As opções avançadas foram mantidas nos valores padrão, sendo 50 para *Smoothness* (suavidade), *Straightness* (linearidade) e *Image Feature* (Recurso de imagem), All para sinalizar que serão usadas todas as bandas da imagem atribuídas à visualização RGB na tela do computador, 1 para *Generalization Tolerance (in pixels)* que é um parâmetro para simplificação

da polilinha. Para a extração do eixo central, o Easytrace requer que o usuário primeiro trace um perfil perpendicular a estrada. A Tabela 20 resume os resultados obtidos.

5.4.6.1 Comparação dos resultados

A comparação de resultados visa comparar a ferramenta *Easytrace* com o Toolbox Adaplin.

Para o recorte r_5 da classe PAV-RAS, a ferramenta *Easytrace* apresentou resultados de 17,5 m, 3,8 m, 3,6 m, 24,9s para d_H , \bar{d}_H , σ_H e duração, que se revelaram levemente piores que os do Toolbox Adaplin (15,5 m, 3,6 m, 3,3 m e 21,8s). O esforço do *Easytrace* foi 8, um pouco menor que o do Toolbox Adaplin (12).

O *Easytrace* indicou 11,3, 2,6 e 2 metros para d_H , \bar{d}_H , σ_H referentes ao recorte r_7 da classe PAV-FLO, valores maiores que 6,7, 2 e 1,4 metros do Toolbox Adaplin. Duração e esforço do *Easytrace* foram de 26,1 s e 8, menores que 46,2 s e 22 do Toolbox Adaplin.

Considerando o recorte r_{10} da classe NAT-RAS, o *Easytrace* apontou 10,9 m, 2 m, 26,5s e 7 para d_H , σ_H , duração e esforço, melhores que os resultados 12,6 m, 2,2 m, 34 s e 8 do Toolbox Adaplin. O *toolbox* e o *Easytrace* mostraram 2 metros de \bar{d}_H .

Para o r_3 da classe NAT-FLO, o *Easytrace* teve melhores resultados em termos de \bar{d}_H , σ_H , duração e esforço (1,7 m, 1,5 m, 42,7s e 21), sendo 2,3 m, 1,9 m, 50,9 s e 31 os resultados para o Toolbox Adaplin. O *toolbox* teve d_H de 10,7 m, menor que 13,7 m do *Easytrace*.

Para o recorte r_5 da classe PAV-SOLO, o *Easytrace* apresentou 13,9 m, 3 m, 2,3 m, 50,8s e 18 para d_H , \bar{d}_H , σ_H , duração e esforço, todos melhores que o do Toolbox Adaplin (20 m, 4,6 m, 4,5 m, 52,4s e 24).

A tabela 21 demonstra as diferenças dos parâmetros produzidos por *Easytrace* e Adaplin.

Um fator limitante ao uso massivo do Erdas Imagine é o alto custo da sua licença, no qual uma unidade custa na ordem de R\$ 30.000,00. O QGIS e o Toolbox Adaplin, por outro lado, podem ser instalados em uma quantidade ilimitada de computadores sem ônus.

Tabela 21 - Diferenças de parâmetros dos resultados produzidos pelas ferramentas Erdas Easytrace e Toolbox Adaplin

Classe	Recorte	Δd_H (m)	$\Delta \bar{d}_H$ (m)	$\Delta \sigma_H$ (m)	Δ Duração(s)	Δ Esforço (#)
PAV-RAS	r_5	2	0,2	0,6	3,1	-4
PAV-FLO	r_7	4,6	0,6	0,6	-20,1	-14
NAT-RAS	r_{10}	-1,7	0	-0,2	-7,5	-1
NAT-FLO	r_3	3	-0,6	-0,4	-8,2	-10
PAV-SOLO	r_5	-6,1	-1,6	-2,2	-1,6	-6

Legenda: + - Diferença favorável a Adaplin; - - Diferença favorável ao Easytrace.

Fonte: O autor, 2017.

5.5 Discussão dos resultados

Para a classe PAV-RAS, a duração média por recorte da extração manual ficou em 39,6 segundos, enquanto a da extração pelo Toolbox Adaplin ficou em 31,4 segundos, representando uma redução de 20,6% do tempo gasto por recorte. A estrada teve uma boa separação do entorno na maioria dos recortes, o que facilitou o traçado da estrada pela ferramenta. Em termos de exatidão, a extração manual apresentou melhores resultados. Ela apresentou 8,1 metros como a média dos desvios máximos, descritos por d_H , e 2,3 metros como a média dos desvios, descritos por \bar{d}_H . A extração com o Toolbox Adaplin teve média dos desvios máximos equivalente a 14,8 metros e média dos desvios equivalente a 3,3 metros. Os desvios máximos podem surgir, por exemplo, de objetos nas margens da estrada e que possuem brilho maior que esta, atraindo portanto os pontos inseridos pela ferramenta. As médias dos desvios de ambos modos de extração ficaram menores do que a resolução do pixel da imagem Rapideye, que é 5 metros. O esforço médio por recorte da extração manual foi de 34,1, enquanto o da extração pelo Toolbox Adaplin foi de 17,8, o que revela uma redução de 47,9% no esforço e sugere um maior conforto do operador ao realizar o traçado da estrada. A Figura 13 ilustra os resultados obtidos pelo método manual, e pelas ferramentas Adaplin e Easytrace para o recorte r_5 : PAV-RAS.

A vetorização pelo Toolbox Adaplin também proporcionou economia de tempo na classe PAV-FLO, embora esta economia tenha sido menor que na classe de estradas

pavimentadas cercadas apenas por vegetação rasteira. Este fato é percebido pela média da duração por recorte, que é de 51,9 segundos para a extração manual e de 46,4 segundos para a extração pelo *toolbox*, o que configura uma poupança de 10,5% do tempo. A média dos desvios por recorte foi de 1,74 metro para a extração manual e de 3,1 metros para a extração pelo *toolbox*. Ambos modos de extração tiveram melhores resultados referentes à média dos desvios do que os respectivos resultados na classe PAV-RAS. Isto pode ser devido ao bom contraste apresentado entre a estrada pavimentada e a floresta. Entretanto, a média por recorte dos desvios máximos foi de 15,4 metros para extração pelo *toolbox*, enquanto foi de 6,5 metros na extração manual. Um evento que atrapalha o *toolbox* é a oclusão da via por árvores ou sombras de árvores, o que pode causar o mau funcionamento da ferramenta, fazendo com que ela tente uma trajetória que se desvia das sombras ou árvores e que não corresponde ao eixo real da estrada. O esforço médio foi de 59,5 para o modo manual e 25,6 para o modo semiautomático, o que é uma redução de 56,9%. A Figura 14 ilustra o resultado das extrações manual e semiautomáticas do Adaplin e do Easytrace comparadas com o eixo de referência para o recorte r_7 : PAV-FLO.

Para a classe NAT-RAS houve uma poupança de tempo em relação à extração manual com o uso do Toolbox Adaplin. A duração média por recorte da extração manual ficou em 48,9 segundos, enquanto o da extração pelo Toolbox Adaplin ficou em 43 segundos, o que sugere uma economia de 12,1% do tempo. Foi notado que a invasão de estradas por vegetação, que a presença de objetos claros nas margens das estradas e que a presença de outras estradas atrapalharam o resultado da extração gerando ruído no resultado. Entretanto as médias dos desvios médios e máximos foram boas em relação às outras classes. A média dos desvios ficou em 1,5 metro para a extração manual e 2,7 metros para a extração pelo *toolbox*, enquanto a média dos desvios máximos ficou em 5,8 metros para a extração manual e 12,9 metros para a extração pelo *toolbox*. A ferramenta traz conforto ao usuário, pois o esforço médio por recorte foi de 49,1 para a extração manual e 17 para a extração pelo Toolbox Adaplin, representando uma redução de 65,4% no esforço. A Figura 15 ilustra o resultado das extrações manual e semiautomáticas do Adaplin e do Easytrace comparadas para o recorte r_{10} : NAT-RAS.

A duração média por recorte medida para a classe NAT-FLO foi de 50,7 segundos para a extração manual e de 42,8 segundos para a extração pelo Toolbox Adaplin, o que representa uma poupança de 15,6% do tempo. Alguns fatores dificultaram a extração através do Toolbox Adaplin, como a obstrução da estrada por árvores e a presença de objetos claros nas margens da estrada, como casas e estradas adjacentes. A média dos desvios médios foi 1,5 metro para a extração manual e de 2,5 metros para a extração pelo *toolbox*, enquanto a média dos desvios máximos foi de 6,5 metros para a extração manual e de 13,8 metros para a extração pelo *toolbox*. O esforço médio por recorte foi de 64,9 para a extração manual e de 25,2 para a extração pelo Toolbox Adaplin, o que mostra uma redução de 61,2% dos pontos marcados pelo operador. A Figura 16 ilustra o resultado

das extrações manual e semiautomáticas do Adaplin e do Easytrace comparadas com o eixo de referência para o recorte r_3 : NAT-FLO.

De maneira geral, os resultados foram ruins para a classe PAV-SOLO. O eixo traçado pela ferramenta não adere ao eixo central, oscilando em torno dele. O solo exposto tende a atrair os pontos inseridos pela ferramenta na maior parte das vezes. A duração média por recorte foi de 30,2 segundos para a extração manual e de 33,3 segundos para a extração pelo Toolbox Adaplin, o que significa um acréscimo de 10,3% no tempo de extração. A média dos desvios foi de 2,6 metros para a extração manual e de 6,8 metros para a extração pelo *toolbox*, enquanto a média dos desvios máximos foi de 8,1 metros para a extração manual e de 19,3 metros para a extração pelo *toolbox*. O esforço médio foi de 12,2 para a extração pelo *toolbox* e de 18,1 para a extração manual. A Figura 17 ilustra o resultado das extrações manual e semiautomáticas do Adaplin e do Easytrace comparadas com o eixo de referência para o recorte r_5 : PAV-SOLO.

A Tabela 22 ilustra as diferenças entre as médias de d_H , \bar{d}_H , σ_H , duração e esforço resultantes das extrações manual e via *toolbox* para todas as classes. É conveniente reparar que as diferenças entre as médias de \bar{d}_H estão dentro de um píxel da imagem Rapideye, cuja resolução espacial é de 5 metros. Por outro lado, a Tabela 23 mostra a contagem do número de vitórias para cada classe dos métodos via Toolbox Adaplin e manual.

O Decreto Lei nº 89.817 de 20 de junho de 1984 estabelece o Padrão de Exatidão Cartográfica (PEC) para a produção de cartografia no Brasil. Segundo o PEC, as cartas podem ser classificadas quanto a sua exatidão em A, B e C. O PEC se divide em planimétrico e altimétrico, portanto, a qualidade da planimetria e da altimetria devem ser avaliados separadamente. Para classificar a carta, 90% dos pontos testados no terreno não podem apresentar erro superior ao PEC e o desvio padrão desse conjunto de erros não deve ultrapassar 60,8% do PEC.

O PEC Classe A planimétrico é de 0,5 mm na escala da carta, sendo de 0,3 mm o desvio padrão correspondente. Por exemplo, o PEC planimétrico de uma escala 1:100.000 é de 50 metros e o desvio padrão correspondente é de 30 metros, enquanto o PEC de uma escala 1:50.000 é de 25 metros e seu desvio padrão correspondente é de 15 metros. Já o PEC de uma escala 1:25.000 é de 12,5 metros e o desvio padrão compatível é de 7,5 metros.

O decreto se refere a feições pontuais, entretanto nesta dissertação trabalha-se com feições lineares. Para aplicar a classificação conforme o decreto, o desvio máximo d_H será tomado como se fosse o erro pontual mencionado para que se faça o cálculo para o pior caso. Consequentemente, também será calculado o desvio padrão desse erro.

Agregando os resultados de d_H nas tabelas 6, 9, 12, 15 e 18 de avaliação do Toolbox Adaplin nas diversas classes, o desvio padrão desse conjunto de dados é 4,8 metros, sendo portanto menor que 15 metros do desvio padrão para a carta classe A da escala 1:50.000. Também é notado que apenas 2 recortes ultrapassaram o erro de 25 metros definido

Tabela 22 - Diferenças entre médias dos parâmetros para as extração manual e via Toolbox Adaplin

Classe	Δd_H (m)	$\Delta \bar{d}_H$ (m)	$\Delta \sigma_H$ (m)	Δ Duração(s)	Δ Esforço (#)
PAV-RAS	-6,7	-1	-1,1	8,2	16,3
PAV-FLO	-8,9	-1,4	-1,7	5,5	33,9
NAT-RAS	-7,1	-1,2	-1,2	5,9	32,1
NAT-FLO	-7,3	-1	-1	4,9	39,7
PAV-SOLO	-11,2	-4,2	-2,8	-3,1	5,9

Legenda: + - Diferença favorável a Adaplin; - - Diferença favorável ao manual.

Fonte: O autor, 2017.

Tabela 23 - Contagem do número de vitórias para cada classe na ordem Adaplin/Manual

Classe	d_H (m)	\bar{d}_H (m)	σ_H (m)	Duração(s)	Esforço (#)
PAV-RAS	1/19	3/17	1/19	18/2	19/0
PAV-FLO	0/20	0/20	1/19	20/0	20/0
NAT-RAS	1/19	0/20	1/19	19/1	20/0
NAT-FLO	1/19	0/20	0/20	20/0	20/0
PAV-SOLO	0/20	0/20	0/20	5/15	13/5

Fonte: O autor, 2017.

pelo PEC e esse fato aconteceu para a classe PAV-SOLO, o que corresponde a 2% dos 100 recortes analisados. Considerando as imagens com um georreferenciamento de boa exatidão, esses cálculos sugerem o uso do Toolbox Adaplin com imagens Rapideye para o mapeamento, considerando as escalas tradicionais, em até 1:50.000 com PEC Classe A. Entretanto, cautela é necessária para a classe PAV-SOLO, sendo preferível usar a extração manual nesta situação. Como alternativa para estes casos, a ferramenta Toolbox Adaplin permite, com o uso da tecla **Ctrl**, que o operador alterne entre os processos manual e semiautomático. Trata-se de uma alternativa interessante e com a qual, com algum treinamento, o operador poderá se valer das vantagens de cada abordagem em tempo real em função do contexto específico do local sob análise.

Figura 13 - Resultados das extrações manual e semiautomáticas para o recorte r_5 : PAV-RAS



Figura 14 - Resultados das extrações manual e semiautomáticas para o recorte r_7 : PAV-FLO

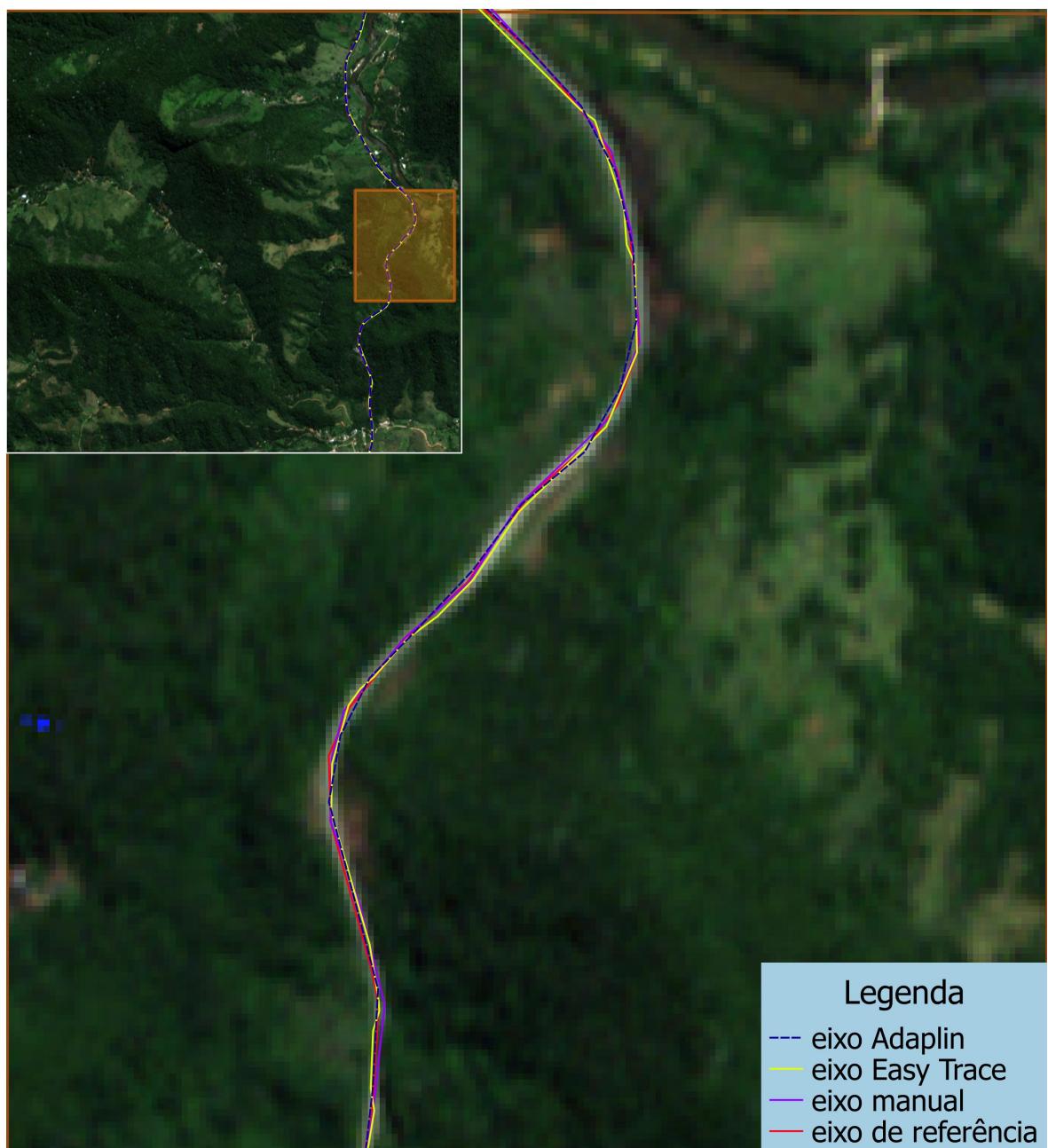


Figura 15 - Resultados das extrações manual e semiautomáticas para o recorte r_{10} : NAT-RAS

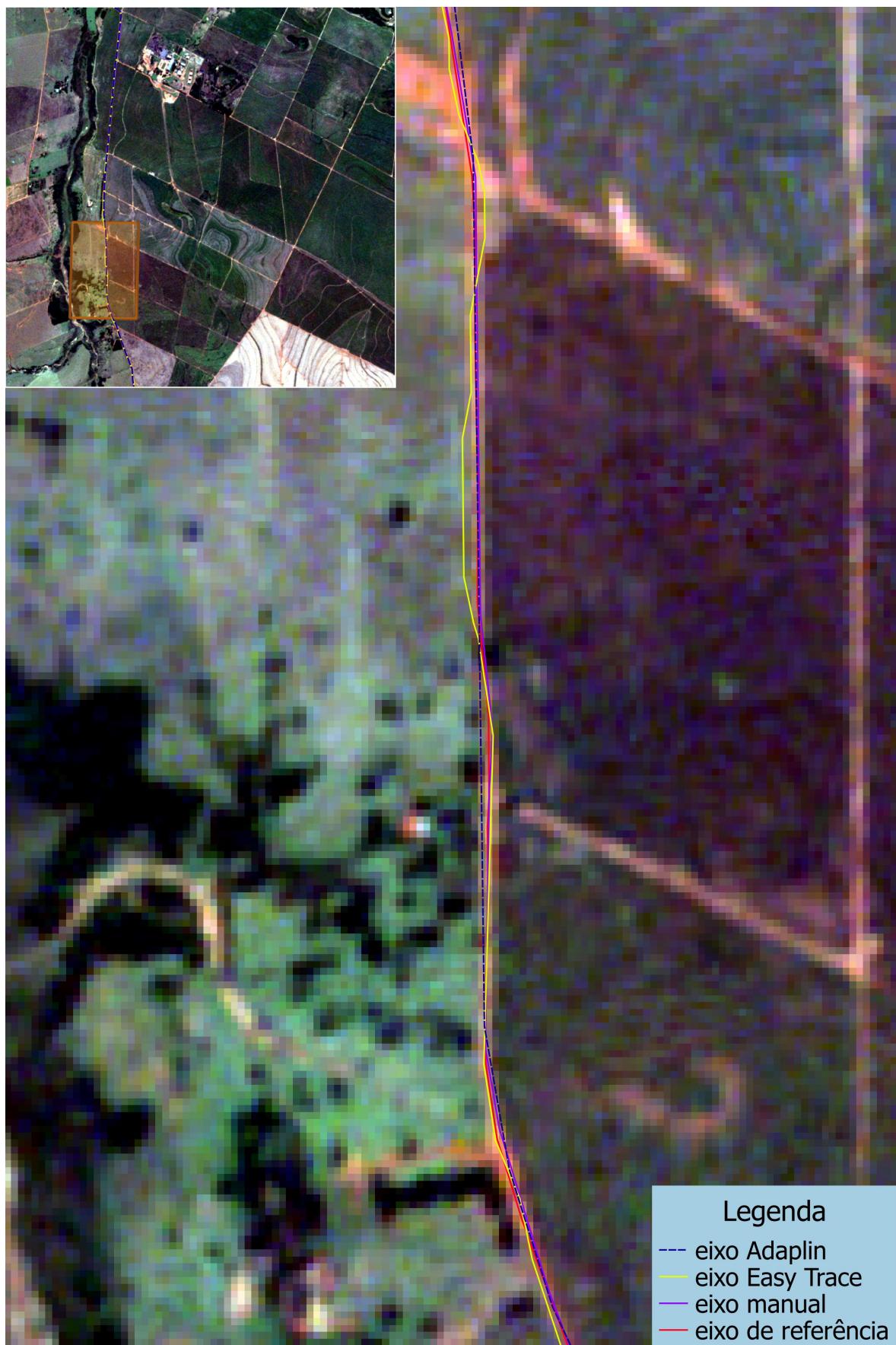


Figura 16 - Resultados das extrações manual e semiautomáticas para o recorte r_3 : NAT-FLO



Figura 17 - Resultados das extrações manual e semiautomáticas para o recorte r_5 : PAV-SOLO



CONCLUSÃO E SUGESTÕES PARA TRABALHOS FUTUROS

O trabalho desenvolveu uma ferramenta interativa para vetorização semiautomática de estradas em um ambiente de Sistemas de Informações Geográficas, visando otimizar a vetorização manual. Ela foi implementada como uma extensão (*toolbox*) do software livre QGIS. A extensão mostrou um bom potencial como alternativa ao método manual e a ferramentas proprietárias, como o Erdas Easytrace.

A ferramenta desenvolvida se aplica a estradas que possuem largura de poucos pixels na imagem, assemelhando-se assim a uma linha. A medida que a resolução espacial da imagem aumenta, as estradas tendem a se assemelhar mais a uma área alongada que a uma linha. Apesar de isso não ter sido explorado consistentemente na presente pesquisa, é de se esperar que, para imagens de maior resolução, o eixo obtido tenha um maior deslocamento em relação ao verdadeiro eixo central da estrada, impactando a exatidão do resultado produzido pela ferramenta.

Nos experimentos realizados, embora a extração manual tenha apresentado resultados mais exatos, o Toolbox Adaplin evidenciou uma melhoria do tempo da restituição em quase todas as classes, com exceção da classe PAV-SOLO (estrada pavimentada cercada por solo exposto). Essa exceção se deve ao comportamento da ferramenta de maximizar a média das componentes RGB da imagem, o que tende a atrair os pontos inseridos pela ferramenta para o solo exposto em detrimento do asfalto. Para as demais classes, a ferramenta também proporcionou um maior conforto para o operador, diminuindo consistentemente o esforço necessário para a vetorização.

Outras limitações da ferramenta são lidar com estradas adjacentes e outros objetos luminosos como casas, que ficam localizadas nas margens da estrada, com obstruções da estrada por árvores e com a invasão da estrada por vegetação nos casos de estradas em leito natural. Para superar essas limitações, nessas situações, é recomendado utilizar o modo manual da ferramenta, que é ativado pressionando a tecla **Ctrl**. Trata-se de uma alternativa interessante e que possibilita que o operador alterne entre os processos manual e semiautomático da ferramenta. Assim, com algum treinamento, o usuário poderá valer-se das vantagens de cada abordagem em tempo real em função do contexto específico do local sob análise.

Levando em conta que as imagens possuem boa exatidão no georreferenciamento, os cômputos feitos na presente dissertação indicam a viabilidade da aplicação do Toolbox Adaplin com imagens Rapideye para o mapeamento em escala até 1:50.000 conforme Padrão de Exatidão Cartográfica (PEC) classe A. Ressalta-se o cuidado a ser tomado para a classe PAV-SOLO, sendo preferível a extração manual nesse caso.

O Toolbox Adaplin desenvolvido na presente dissertação será disponibilizado como extensão do QGIS, de forma que os usuários da comunidade de geotecnologias livres

possam instalá-lo e utilizá-lo livremente nos termos da licença GNU GPL v2. O código fonte também será providenciado de forma que os desenvolvedores possam melhorar e contribuir para o desenvolvimento da ferramenta.

A ferramenta atualmente acrescenta dois pontos para cada segmento traçado pelo usuário. Como passo futuro sugere-se a inserção de uma quantidade variável de pontos em função do comprimento do segmento. Outros passos são a pesquisa de uma forma de agregar à ferramenta a capacidade de extração do eixo central, permitindo que trabalhe com estradas que possuam vários pixels de largura, e o teste de outras combinações de bandas, além das componentes RGB, para a extração mais exata possível em cenários diversos. É interessante também o estudo de outros critérios de escolha do traçado, como por exemplo atribuindo um peso maior ao gradiente dos pixels e a consideração da distância euclidiana entre os valores dos pixels ao invés da diferença entre médias RGB. A ferramenta também pode ser testada para a extração de trechos de drenagem usando a banda do infravermelho.

REFERÊNCIAS

- AMO, Miriam; MARTINEZ, Fernando; TORRE, Margarita. Road extraction from aerial images using a region competition algorithm. *IEEE transactions on image processing*, v. 15, n. 5, p. 1192–1201, 2006.
- ANIL, PN; NATARAJAN, S. Automatic road extraction from high resolution imagery based on statistical region merging and skeletonization. *International Journal of Engineering Science and Technology*, v. 2, n. 3, p. 165–171, 2010.
- BALL, Geoffrey H.; HALL, David J. *ISODATA, a novel method of data analysis and pattern classification*. [S.l.], 1965.
- BARTELS, Richard H.; BEATLY, John C.; BARSKY, Brian A. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. 1. ed. San Mateo, CA: Morgan Kaufmann, 1995. (The Morgan Kaufmann Series in Computer Graphics). ISBN 978-1558604001.
- BAUMGARTNER, A.; HINZ, S.; WIEDEMANN, C. Efficient methods and interfaces for road tracking. In: *International Archives of Photogrammetry and Remote Sensing*. [S.l.: s.n.], 2002.
- CANNY, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8, n. 6, p. 679–698, Nov 1986. ISSN 0162-8828.
- CHENG, Yizong. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, IEEE Computer Society, Washington, DC, USA, v. 17, n. 8, p. 790–799, ago. 1995. ISSN 0162-8828. Disponível em: <<http://dx.doi.org/10.1109/34.400568>>.
- COMANICIU, Dorin; MEER, Peter. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE, v. 24, n. 5, p. 603–619, 2002.
- CORTES, Corinna; VAPNIK, Vladimir. Support-vector networks. *Machine Learning*, v. 20, n. 3, p. 273–297, 1995. ISSN 1573-0565. Disponível em: <<http://dx.doi.org/10.1007/BF00994018>>.
- DAL-POZ, Aluir P.; VALE, Giovane M. Do; ZANIN, Rodrigo B. Automatic extraction of road seeds from high-resolution aerial images. *Anais da Academia Brasileira de Ciências*, scielo, v. 77, p. 509 – 520, 09 2005. ISSN 0001-3765. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0001-37652005000300011&nrm=iso>.
- ERDAS Inc. *IMAGINE EasytraceTM Users Guide*. 1. ed. Norcross, GA: ERDAS, Inc., 2010. Disponível em: <ftp://jetty.ecn.purdue.edu/jshan/ERDAS_Library/2011help/hardcopy/Easytrace.pdf>.
- FACON, Jacques. *Morfologia Matemática: Teoria e Exemplos*. 1. ed. Curitiba: Editora Universitária Champagnat da Pontifícia Universidade Católica do Paraná., 1996.
- GERKE, Markus et al. Graph-supported verification of road databases. *ISPRS Journal of Photogrammetry and Remote Sensing*, Elsevier, v. 58, n. 3, p. 152–165, 2004.

- GRUEN, Armin; LI, Haihong. Semi-automatic linear feature extraction by dynamic programming and lsb-snakes. *Photogrammetric engineering and remote sensing*, [Falls Church, Va.] American Society of Photogrammetry., v. 63, n. 8, p. 985–994, 1997.
- HARWANI, B. M. *Introduction to Python Programming and Developing GUI Applications with PyQt*. 1. ed. Boston, MA: Cengage Learning PTR, 2011. ISBN 978-1-4354-6097-9.
- HAYKIN, Simon. *Neural Networks and Learning Machines*. 3. ed. New York: Pearson, 2008. ISBN 978-0131471399.
- HELMHOLZ, Petra et al. Semi-automatic verification of geodata for quality management and updating of gis. *IntArchPhRS*, v. 36, n. 4, p. 9–13, ago. 2007. Disponível em: <http://www.ipi.uni-hannover.de/uploads/tkpublikationen/helmholz_urumchi.pdf>.
- HOROWITZ, Ellis; SAHNI, Sartaj. *Fundamentals of computer algorithms*. [S.l.]: Galgotia Publications, 1984.
- HU, Xiangyun; ZHANG, Zuxun; TAO, C. Vincent. A robust method for semi-automatic extraction of road centerlines using a piecewise parabolic model and least square template matching. *Photogrammetric Engineering & Remote Sensing*, v. 70, n. 12, p. 1393–1398, 2004. ISSN 0099-1112. Disponível em: <<http://www.ingentaconnect.com/content/asprs/pers/2004/00000070/00000012/art00005>>.
- Intergraph Corp. *Erdas IMAGINE 2013: product features and comparisons*. 1. ed. Norcross, GA: Intergraph Corp., 2013. Disponível em: <http://www.hexagon-solutions.com.cn/Libraries/Tech_Docs/ERDAS_IMAGINE_2013_Product_Description.sflb.pdf>.
- LAWHEAD, Joel. *QGIS Python Programming Cookbook*. 1. ed. Birmingham, UK: Packt Publishing Ltd., 2015. ISBN 978-1-78398-498-5.
- LONG, Hui; ZHAO, Zhongming. Urban road extraction from high-resolution optical satellite images. *International Journal of Remote Sensing*, Taylor & Francis, v. 26, n. 22, p. 4907–4921, 2005.
- MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967. p. 281–297. Disponível em: <<http://projecteuclid.org/euclid.bsmsp/1200512992>>.
- MENKE, Kurt et al. *Mastering QGIS*. 2. ed. Birmingham, UK: Packt Publishing Ltd., 2016. ISBN 978-1-78439-868-2.
- MOKHTARZADE, Mehdi; ZOEJ, MJ Valadan. Road detection from high-resolution satellite images using artificial neural networks. *International journal of applied earth observation and geoinformation*, Elsevier, v. 9, n. 1, p. 32–40, 2007. ISSN 0303-2434.
- MOZAS-CALVACHE, Antonio-Tomás. Control de calidad posicional en cartografía por elementos lineales. Jaén: Universidad de Jaén, 2008.
- NOCK, Richard; NIELSEN, Frank. Statistical region merging. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, IEEE Computer Society, Los Alamitos, CA, USA, v. 26, n. 11, p. 1452–1458, 2004. ISSN 0162-8828.

- PANDIT, Vinay. *Automatic road extraction from high resolution satellite imagery*. Dissertaçāo (Mestrado) — International Institute of Information Technology Hyderabad, 2009.
- Planet Labs Team. *Rapideye imagery product specifications*. San Francisco, CA, 2016. 50 p. Disponível em: <<https://www.planet.com/products/satellite-imagery/files/160625-RapidEyeImage-Product-Specifications.pdf>>.
- Planet Team. *Planet Application Program Interface: In Space for Life on Earth*. San Francisco, CA: Planet, 2014. Disponível em: <<https://api.planet.com>>.
- QGIS Development Team. *QGIS Geographic Information System*. [S.l.], 2009. Disponível em: <<http://qgis.osgeo.org>>.
- _____. *QGIS Geographic Information System Developers Manual*. on-line. Open Source Geospatial Foundation Project, 2016. Disponível em: <<https://www.qgis.org/en/site/getinvolved/development/qgisdevelopersguide/index.html>>.
- SHERMAN, Gary. *The PyQGIS Programmer's Guide - Extending QGIS with Python*. 1. ed. Chugiak, AK: Locate Press LLC, 2014. ISBN 978-0989421720.
- SONG, Mingjun; CIVCO, Daniel. Road extraction using svm and image segmentation. *Photogrammetric Engineering & Remote Sensing*, American Society for Photogrammetry and Remote Sensing, v. 70, n. 12, p. 1365–1371, 2004.
- STEINHAUS, H. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, v. 1, p. 801–804, 1956.
- THULASIRAMAN, K.; SWAMY, M. N. S. *Graphs: Theory and Algorithms*. 1. ed. New York: John Wiley & Sons, Inc., 1992. ISBN 9781118033104.
- WESTRA, Erik. *Building Mapping Applications with QGIS*. 1. ed. Birmingham, UK: Packt Publishing Ltd., 2014. ISBN 978-1-78398-466-4.
- ZHANG, Chunsun; BALTSAVIAS, Emmanuel. Improving cartographic road databases by image analysis. In: *International Archives of Photogrammetry and Remote Sensing 34, Part 3A*. [S.l.: s.n.], 2002. p. 400–405.
- ZHANG, Chunsun; MURAI, Shunji; BALTSAVIAS, Emmanuel. Road network detection by mathematical morphology. In: CITESEER. *PROC. ISPRS WORKSHOP “3D GEOSPATIAL DATA PRODUCTION: MEETING APPLICATION REQUIREMENTS*. [S.l.], 1999.
- ZHANG, Qiaoping. *Automated Road Network Extraction from High Spatial Resolution Multi-Spectral Imagery*. Tese (PhD Thesis) — SCHULICH School of Engineering, University of Calgary, Alberta, Calgary, Canada, April 2006. Disponível em: <http://www.ucalgary.ca/engo_webdocs/IC/06.20240_QZhang.pdf>.
- ZIVIANI, Nivio. *Projeto de Algoritmos com Implementações em Pascal e C*. 2. ed. São Paulo: Pioneira Thomson Learning, 2004. ISBN 85-221-0390-9.

APÊNDICE A - Código Python do Toolbox Adaplin

```

1  class Adaplin(QgsMapTool):
2      # Atributos iniciais da classe
3      def __init__(self, iface, camada_raster):
4          # Camada raster é armazenada como atributo da classe
5          self.camada_raster = camada_raster
6
7          self.iface = iface
8          self.canvas = self.iface.mapCanvas()
9          QgsMapTool.__init__(self, self.canvas)
10
11         self.rb = QgsRubberBand(self.canvas, Qgs.Polygon)
12         self.type = Qgs.Polygon
13
14         # self.points é a lista de pontos marcados pelo operador
15         # self.pontos_interpolados é a lista de pontos que
16         # agrupa o resultado da interpolação
17         self.points = []
18         self.pontos_interpolados = []
19
20         # self.mCtrl define se ferramenta está no modo manual
21         self.mCtrl = False
22
23         self.cursor = QCursor(Qt.CrossCursor)
24
25     # Pressionamento de um botão do mouse
26     def canvasPressEvent(self, event):
27         # Posições x e y (em pixels) do cursor na tela do QGIS
28         x = event.pos().x()
29         y = event.pos().y()
30
31         # Se botão esquerdo do mouse for pressionado
32         if event.button() == Qt.LeftButton:
33             startingPoint = QPoint(x,y)
34
35             # Atração para vértices pertos (Snap)
36             # conforme configurações do QGIS
37             snapper = QgsMapCanvasSnapper(self.canvas)
38             (retval, result) = snapper.snapToCurrentLayer (startingPoint,
39             QgsSnapper.SnapToVertex)
40
41             if result <> []:
42                 point = QgsPoint( result[0].snappedVertex )

```

```

43         (retval, result) = snapper.snapToBackgroundLayers(
44             startingPoint)
45         if result <> []:
46             point = QgsPoint( result[0].snappedVertex )
47         else:
48             point = self.canvas.getCoordinateTransform().toMapCoordinates(
49                 event.pos().x(), event.pos().y())
50
51             # Ponto marcado é adicionado à sua respectiva lista
52             self.points.append(point)
53
54             # Adiciona apenas o ponto marcado com modo manual ativado
55             # Senão os pontos interpolados são adicionados também
56             if self.mCtrl:
57                 self.pontos_interpolados.append(point)
58             else:
59                 pontos_recentes, grafo, retas_perpendiculares, result =
60                 self.interpolacao(self.points[-2:])
61                 self.pontos_interpolados = self.pontos_interpolados +
62                 pontos_recentes[1:]
63
64                 # Se o botão direito for pressionado a linha é concluída
65                 else:
66                     if len(self.points) >= 2:
67                         self.createFeature(self.pontos_interpolados)
68
69                     self.resetPoints()
70                     self.resetRubberBand()
71                     self.canvas.refresh()
72
73
74             # Reinicia listas para próxima feição
75             def resetPoints(self):
76                 self.points = []
77                 self.pontos_interpolados = []
78
79             # Cria feição
80             def createFeature(self, pontos_interpolados):
81                 layer = self.canvas.currentLayer()
82                 provider = layer.dataProvider()
83                 fields = layer.pendingFields()
84                 f = QgsFeature(fields)
85
86                 coords = pontos_interpolados
87
88                 if self.canvas.mapRenderer().hasCrsTransformEnabled() and layer.crs()
89                 () != self.canvas.mapRenderer().destinationCrs():
90                     coords_tmp = coords[:]

```

```

85     coords = []
86     for point in coords_tmp:
87         transformedPoint = self.canvas.mapRenderer().mapToLayerCoordinates( layer , point )
88         coords.append(transformedPoint)
89
90     if self.isPolygon == True:
91         g = QgsGeometry().fromPolygon([coords])
92     else:
93         g = QgsGeometry().fromPolyline(coords)
94     f.setGeometry(g)
95
96     for field in fields.toList():
97         ix = fields.indexFromName( field.name() )
98         f[ field.name() ] = provider.defaultValue(ix)
99
100    layer.beginEditCommand("Feature_added")
101
102    settings = QSettings()
103
104    disable_attributes = settings.value( "/qgis/digitizing/
105 disable_enter_attribute_values_dialog", False, type=bool)
106
107    if disable_attributes:
108        layer.addFeature(f)
109        layer.endEditCommand()
110
111    else:
112        dlg = self iface.getFeatureForm(layer, f)
113        if QGis.QGIS_VERSION_INT >= 20400:
114            dlg.setIsAddDialog( True )
115            if dlg.exec_():
116                if QGis.QGIS_VERSION_INT < 20400:
117                    layer.addFeature(f)
118                    layer.endEditCommand()
119
120    # Resposta ao movimento do mouse
121    def canvasMoveEvent(self, event):
122        # Cor e espessura da linha
123        color = QColor(255,0,0,100)
124        self.rb.setColor(color)
125        self.rb.setWidth(3)
126
127        x = event.pos().x()
128        y = event.pos().y()
129
130        # Ações semelhantes ao pressionamento

```

```

130     # do botão esquerdo do mouse
131     startingPoint = QPoint(x,y)
132     snapper = QgsMapCanvasSnapper( self.canvas )
133
134     (retval,result) = snapper.snapToCurrentLayer( startingPoint ,
135         QgsSnapper.SnapToVertex)
135     if result <> []:
136         point = QgsPoint( result[0].snappedVertex )
137     else:
138         (retval,result) = snapper.snapToBackgroundLayers( startingPoint )
139         if result <> []:
140             point = QgsPoint( result[0].snappedVertex )
141         else:
142             point = self.canvas.getCoordinateTransform() .
143             toMapCoordinates( event.pos().x() , event.pos().y() );
144
144     pontos_marcados = list( self.points )
145     pontos_interpolados = list( self.pontos_interpolados )
146     pontos_marcados.append( point )
147
148     if self.mCtrl:
149         pontos_interpolados.append( point )
150     else:
151         pontos_recentes , grafo , pontos_perpendiculares , result = self.
152         interpolacao ( pontos_marcados[-2::] )
152         pontos_interpolados = pontos_interpolados + pontos_recentes[1:]
153
154     self.setRubberBandPoints( pontos_interpolados )
155
156 def showSettingsWarning( self ):
157     pass
158
159 # Chamado quando a ferramenta é ativada
160 def activate( self ):
161     self.canvas.setCursor( self.cursor )
162
163     mc = self.canvas
164     layer = mc.currentLayer()
165     self.type = layer.geometryType()
166     self.isPolygon = False
167     if self.type == QGis.Polygon:
168         self.isPolygon = True
169
170 # Reinicia desenho da linha na tela
171 def resetRubberBand( self ):
172     self.rb.reset( self.type )
173

```

```

174 # Estabelece pontos para o desenho
175 # da linha na tela
176 def setRubberBandPoints(self ,points):
177     self.resetRubberBand()
178     for point in points:
179         update = point is points[-1]
180         self.rb.addPoint( point , update )
181
182 def isZoomTool( self ):
183     return False
184
185 def isTransient( self ):
186     return False
187
188 def isEditTool( self ):
189     return True
190
191 # Faz o trabalho de interpolação de um segmento
192 def interpolacao(self , points):
193     # Grafo criado é uma lista de listas
194     # Cada lista é uma etapa
195     grafo = []
196
197     npoints = len(points)
198
199     # Caso do ponto inicial em que uma lista de
200     # um ponto é passada como argumento
201     if npoints == 1: return [QgsPoint(1,1) , points[0]] , grafo , [] , None
202
203     # Adiciona pontos perpendiculares ao segmento
204     # em etapas para formação do grafo
205     for i in range(0 , npoints-1):
206         try:
207             retas_perpendiculares = self.calcula_retas(points[i] ,
208                 points[i+1])
209         except:
210             return points , grafo , [] , []
211
212         grafo.append([points[i]])
213         grafo.extend(retas_perpendiculares)
214
215         grafo.append([points[-1]])
216
217         # Acha melhor caminho no grafo
218         result , p = self.acha_caminho(grafo , points)
219
220         return result , grafo , retas_perpendiculares , result

```

```

220
221 # Calcula pontos pertencentes às retas das etapas
222 def calcula_retas(self, p1, p2):
223     # Quantidade de pontos acima e abaixo do segmento
224     pontos_acima_e_abaixo = 5
225
226     # Espaçamento entre os pontos de uma etapa
227     resolucao_y = 4.5
228
229     dx = p2.x() - p1.x()
230     dy = p2.y() - p1.y()
231
232     dist = sqrt(dx*dx + dy*dy)
233
234     dx /= dist
235     dy /= dist
236
237     x_p1 = p1.x()
238     x_p2 = p2.x()
239     y_p1 = p1.y()
240     y_p2 = p2.y()
241
242     # n é a quantidade de segmentos originada
243     # da inserção de q pontos
244     n = 3.0
245     q = 2
246
247     # Coordenadas x e y dos pontos equidistantes
248     # inseridos sobre o segmento
249     coord_x_sobre = []
250     coord_y_sobre = []
251
252     for i in range(q, 0, -1):
253         coord_x_sobre.append((i/(n))*x_p1 + (1. - i/n)*x_p2)
254         coord_y_sobre.append((i/(n))*y_p1 + (1. - i/n)*y_p2)
255
256     # retas_perpendiculares é uma lista de listas
257     # Cada lista corresponde aos pontos de uma etapa
258     retas_perpendiculares = []
259
260     for i in range(len(coord_x_sobre)):
261         x_acima = [(coord_x_sobre[i] - resolucao_y*j*dy) for j in range
262                     (1, pontos_acima_e_abaixo)]
263         y_acima = [(coord_y_sobre[i] + resolucao_y*j*dx) for j in range
264                     (1, pontos_acima_e_abaixo)]

```

```

264         x_abixo = [(coord_x_sobre[i] + resolucao_y*j*dy) for j in
265             range(1, pontos_acima_e_abixo)]
266         y_abixo = [(coord_y_sobre[i] - resolucao_y*j*dx) for j in
267             range(1, pontos_acima_e_abixo)]
268
269         lista_de_x = [coord_x_sobre[i]] + x_acima + x_abixo
270         lista_de_y = [coord_y_sobre[i]] + y_acima + y_abixo
271
272         lista_pontos_perpendiculares = [QgsPoint(lista_de_x[j],
273             lista_de_y[j]) for j in range(len(lista_de_x))]
274
275         retas_perpendiculares.append(lista_pontos_perpendiculares)
276
277     return retas_perpendiculares
278
279 # Propriedade 1 da função objetivo
280 def Prop1(self, no1, no2, no3):
281     no1_media = self.rgb_media(self.camada_raster, no1)
282     no2_media = self.rgb_media(self.camada_raster, no2)
283     no3_media = self.rgb_media(self.camada_raster, no3)
284
285     return no1_media*no1_media + no2_media*no2_media + no3_media*
286         no3_media
287
288 # Propriedade 2 da função objetivo
289 def Prop2(self, no1, no2, no3):
290     no1no2 = self.pointsDist(no1, no2)
291     no2no3 = self.pointsDist(no2, no3)
292
293     nc_no1 = self.rgb_media(self.camada_raster, no1)
294     nc_no2 = self.rgb_media(self.camada_raster, no2)
295     nc_no3 = self.rgb_media(self.camada_raster, no3)
296
297     mediaNo1No2 = (nc_no1 + nc_no2)/2.
298     mediaNo2No3 = (nc_no2 + nc_no3)/2.
299
300     somatorio_Prop2 = ((nc_no1 - mediaNo1No2)*(nc_no1 - mediaNo1No2) +
301         (nc_no2 - mediaNo1No2)*(nc_no2 - mediaNo1No2)) + ((nc_no2 - mediaNo2No3) *
302             (nc_no2 - mediaNo2No3)) + (nc_no3 - mediaNo2No3)*(nc_no3 - mediaNo2No3))
303
304     return somatorio_Prop2
305
306 # Propriedade 3 da função objetivo
307 def Prop3(self, no1, no2, no3):
308     no1no2 = self.pointsDist(no1, no2)
309     no2no3 = self.pointsDist(no2, no3)
310     no1no3 = self.pointsDist(no1, no3)

```

```

305     distTotal = no1no2 + no2no3
306
307     cosseno_interno = ( (no1no2*no1no2 + no2no3*no2no3 - no1no3*no1no3)
308     / (2*no1no2*no2no3) )
309     cosseno_deflexao = -1*cosseno_interno
310
311     return (1 + cosseno_deflexao) /no1no2
312
313 # Média das bandas RGB
314 def rgb_media(self , raster , ponto):
315     provedor = raster.dataProvider()
316
317     mapCanvasSrs = self iface.mapCanvas().mapRenderer().destinationCrs
318     ()
319
320     rasterSrs = raster.crs()
321     srsTransform = QgsCoordinateTransform(mapCanvasSrs , rasterSrs)
322
323     ponto_transform = srsTransform.transform(ponto)
324     ponto_valor = provedor.identify(ponto_transform , QgsRaster.
325 IdentifyFormatValue).results()
326
327     if len(ponto_valor) > 3:
328         ponto_media = sum([ponto_valor[i] for i in [1 , 2 , 3]]) /3.
329     else:
330         ponto_media = sum([ponto_valor[i] for i in ponto_valor]) /float(
331             len(ponto_valor))
332
333     return ponto_media
334
335 # Acha melhor caminho no grafo
336 def acha_caminho(self , grafo , points):
337     # Listas de dicionários são usadas para
338     # armazenar os custos em cada etapa
339     lista_dic = []
340     lista_dic.append({})
341
342     no_inicial = grafo [0][0]
343
344     for no in grafo [1]:
345         lista_dic [0][(no_inicial , no)] = {MAX: 0}
346
347         for i in range(len(grafo)):
348             g = grafo [i]
349
350             try:
351                 g_k_mais_dois = grafo [i+2]

```

```

348     except:
349         break
350
351     g_k_mais_um = grafo[ i+1]
352     lista_dic.append( {} )
353
354     for j in range( len(g_k_mais_um) ):
355         for k in range( len(g_k_mais_dois) ):
356             no_k_mais_um = g_k_mais_um[ j ]
357             no_k_mais_dois = g_k_mais_dois[ k ]
358
359             lista_dic[ i+1 ][( no_k_mais_um , no_k_mais_dois )] = {}
360
361             for z in range( len(g) ):
362                 no_atual = g[ z ]
363
364             try:
365                 res_parcial = lista_dic[ i ][( no_atual ,
366                 no_k_mais_um )][ MAX ] + ( self.Prop1( no_atual , no_k_mais_um ,
367                 no_k_mais_dois ) - self.Prop2( no_atual , no_k_mais_um , no_k_mais_dois ) ) *
368                 self.Prop3( no_atual , no_k_mais_um , no_k_mais_dois )
369             except:
370                 return points , points
371
372             lista_dic[ i+1 ][( no_k_mais_um , no_k_mais_dois )][
373                 no_atual ] = res_parcial
374
375             lista_dic[ i+1 ][( no_k_mais_um , no_k_mais_dois )][ MAX ] =
376             max([ lista_dic[ i+1 ][( no_k_mais_um , no_k_mais_dois )][ el ] for el in
377                   lista_dic[ i+1 ][( no_k_mais_um , no_k_mais_dois )] ])
378
379             lista_dic_inv = lista_dic[ -1:0:-1 ]
380             prima = lista_dic_inv[ 0 ]
381
382             ultimos , f_acum , no_escolhido = max([( chave , valor [ MAX ] , self .
383               no_maximo( valor ) ) for chave , valor in prima . iteritems () ] , key = lambda
384               par_facum: par_facum[ 1 ])
385             no_anterior , no_posterior = ultimos
386
387             caminho = []
388             caminho += ( no_posterior , no_anterior , no_escolhido )
389
390             for elemento in lista_dic_inv[ 1 :: ]:
391                 no_posterior , no_anterior = no_anterior , no_escolhido

```

```

386         no_escolhido = self.no_maximo(elemento[ (no_anterior ,
387                                         no_posterior) ])
388         caminho.append( no_escolhido )
389
390         caminho.reverse()
391
392         return caminho, points
393
394     # Usado na identificação da
395     # melhor escolha em cada etapa
396     def no_maximo(self , dic):
397         for elem in dic:
398             if elem == MAX:
399                 continue
400             if dic [elem] == dic [MAX]:
401                 return elem
402
403     # Distância euclidiana entre 2 pontos
404     def pointsDist(self , a , b):
405         dx = a.x() - b.x()
406         dy = a.y() - b.y()
407         return sqrt( dx*dx + dy*dy )
408
409     # Define modo manual ao pressionar a tecla Ctrl ,
410     # Esc como tecla opcional para concluir feição
411     # e Backspace para apagar último ponto
412     def keyPressEvent(self , event):
413         if event.key () == Qt.Key_Control:
414             if self.mCtrl is True:
415                 self.mCtrl = False
416             else:
417                 self.mCtrl = True
418         if event.key () == Qt.Key_Escape:
419             self.createFeature( self.pontos_interpolados )
420             self.resetPoints()
421             self.resetRubberBand()
422             self.canvas.refresh()
423         if event.key () == Qt.Key_Backspace:
424             self.removeLastPoint()
425
426     # Método usado para apagar último ponto
427     def removeLastPoint( self ):
428         if len( self.points ) == 0:
429             QMessageBox.information( self iface.mainWindow() , Aviso , Nenhum
430                                     ponto marcado ainda)
431         elif len( self.points ) == 1:
432             self.points.pop()

```

```
431     self.pontos_interpolados.pop()
432 else:
433     penultimo_elemento_marcado = self.points[-2]
434     self.pontos_interpolados = self.pontos_interpolados[0:self.
435     pontos_interpolados.index(penultimo_elemento_marcado)+1]
        self.points.pop()
```