Filtragem de nuvens de pontos para geração de Modelos Digitais do Terreno

João Afonso

2014

João Miguel Nunes Lobão Dias Afonso

Nº 41781

Filtragem de nuvens de pontos para geração de Modelos Digitais do Terreno

Dissertação para obtenção do grau de Mestre em Engenharia Informática

2º Semestre, 2013/2014

Orientador: Prof. Doutor Fernando Pedro Reino Silva Birra, Prof. Auxiliar, Departamento de Informática da FCT/UNL

Co-orientador: Tenente-Coronel de Artilharia Engenheiro Geógrafo, Rui Alberto Ferreira Coelho Dias, Instituto Geográfico do Exército

Júri:

Presidente: Prof. Doutor João Costa Seco
Arguente: Prof.ª Doutora Paula Maria Ferreira de Sousa Redweik
Vogal: Prof. Doutor Pedro Reino Silva Birra

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Dezembro, 2014

João Miguel Nunes Lobão Dias Afonso

Nº 41781

Filtragem de nuvens de pontos para geração de Modelos Digitais do Terreno

Dissertação para obtenção do grau de Mestre em Engenharia Informática

2º Semestre, 2013/2014

Orientador: Prof. Doutor Fernando Pedro Reino Silva Birra, Prof. Auxiliar, Departamento de Informática da FCT/UNL

Co-orientador: Tenente-Coronel de Artilharia Engenheiro Geógrafo, Rui Alberto Ferreira Coelho Dias, Instituto Geográfico do Exército

Júri:

Presidente: Prof. Doutor João Costa Seco
Arguente: Prof.ª Doutora Paula Maria Ferreira de Sousa Redweik
Vogal: Prof. Doutor Pedro Reino Silva Birra

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Dezembro, 2014

Esta dissertação não segue as normas do Novo Acordo Ortográfico da Língua Portuguesa.

Resumo

As cartas topográficas são representações planas, generalizadas e reduzidas à escala, de zonas da superfície terrestre, contendo símbolos e informação textual para a descrição dos objectos. A forma mais comum de as produzir é por intermédio de fotografias aéreas, principalmente pela afinidade entre o conteúdo destas e aquilo que se convencionou representar na carta. O Instituto Geográfico do Exército (IGeoE) é uma entidade produtora de cartografia que provê, com informação geográfica, as Forças Armadas e a comunidade civil.

A caracterização do relevo é parte da informação existente na carta, concretizada através das curvas de nível: linhas curvas que representam uma cota pré-definida (convencionado de 10 em 10 m nas cartas à escala 1/25 000), constante em toda a sua extensão. Estas acompanham as formas do terreno e indicam a altitude do nível do solo, independentemente de se cruzarem com outros objectos à superfície terrestre (como edifícios ou arvoredo). A informação do relevo é bastante completa, abrangendo a área de toda a carta.

As curvas de nível são, por norma, restituídas, manualmente, por um operador numa estação fotogramétrica, numa tarefa compreensivelmente morosa. Uma das alternativas para a representação do relevo é por intermédio da correlação automática de fotografias aéreas, daí resultando uma nuvem de pontos cotados numa grelha regular, cada um com uma coordenada tridimensional. Alguns desses pontos contêm "ruído", visto representarem não a cota ao nível do solo, mas a cota de objectos sobre a superfície terrestre. O processo de eliminação desse "ruído", que permite corrigir a cota do topo do objecto para o solo, designa-se por filtragem.

Há diversos processos de filtragem de nuvens de pontos, embora nenhum consiga obter resultados totalmente satisfatórios, apresentando mais ou menos dificuldades em algumas zonas de terreno específicas (zonas urbanizadas ou vegetação baixa, por exemplo). Um dos caminhos apontados para auxiliar a filtragem é a utilização de outros recursos que forneçam mais informação, para além da simples coordenada tridimensional do ponto a corrigir, bem como uma mistura de algoritmos, tentando conciliar os pontos fortes de cada uma destas abordagens.

O presente trabalho desenvolveu uma metodologia automática para representar o relevo a partir de uma nuvem de pontos cotados, para ser integrada na cadeia de produção do IGeoE. A partir de uma nuvem de pontos primária, e utilizando como dados de entrada ortofotos e informação vectorial dos objectos da edição anterior e da edição de trabalho da carta (excepto relevo) da mesma região, efectua quatro filtragens: filtragem de edifícios, filtragem de áreas de arvoredo superiores a 150 m x 150 m, filtragem de áreas de arvoredo inferiores a 150 m x 150 m e árvores isoladas (envolvendo a detecção de árvores em ortofotos, ao nível do pixel, por algoritmo de aprendizagem automática), e filtragem por declives.

Palavras-chave: modelo digital do terreno, modelo digital da superfície, nuvem de pontos, filtragem, detecção de árvores em fotografias aéreas.

Abstract

Topographic maps are plane and generalized representations, reduced to scale, of earth surface zones, containing symbols and textual information to describe objects. The most common way to generate them is by aerial photographs, mainly due to the affinity of its content and what was agreed to represent in maps. The Geographic Army Institute (IGeoE) is a cartography agency that provides geographic information to the Armed Forces and the civil community.

The relief is one of the characteristics highlighted in a map, materialized in contour: curved lines representing a pre-defined height (an agreed 10 m distance between them in 1/25 000 scaled maps), constant in their whole extension. These lines circumvent the terrain features and represent the height at the ground level, regardless of crossing other objects at the earth's surface (like buildings or trees). The relief information is quite dense, scattered throughout the map.

By default, the contour is manually restored by an operator in a photogrammetric station, in an understandably consuming task. Another alternative for contour representation is by means of aerial photographs automatic correlation, which generates a point cloud in a grid, having each of those points a three-dimensional coordinate. Some of those points have associated "noise" with them, since they represent not the height at ground level, but the height of objects on the earth's surface. The process of eliminating this "noise", which allows correcting the height of the top of the object to the ground, is called filtering.

There are many filtering processes, though none of them achieve completely satisfactory outcomes in all terrain zones (like urban terrain or low vegetation). One of the already identified ways of improving filtering is the use of other resources that provide more information besides simple point coordinates, as well as a mixture of algorithms, trying to conciliate the strengths of each of the approaches.

In the current work it was developed an automatic process of relief generation via point clouds, which can eventually be integrated in the production chain of IGeoE. Given a primary point cloud, and using as input data ortophotos and vector data of the previous and current map edition (except contour) of the same region, it performs four filterings: buildings, vegetation areas greater than 150 m x 150 m, vegetation areas lesser than 150 m x 150 m (using a machine learning algorithm to detect trees, at pixel level, in ortophotos), and finally a slope filtering.

Keywords: digital terrain model, digital surface model, point clouds, filtering, tree detection from aerial imagery.

Índice

Índice de figuras

Índice de tabelas

Lista de siglas

CAD: Computer-Aided Design
DEM: Digital Elevation Modelo, Modelo Digital de Elevação
DSM: Digital Surface Model, Modelo Digital de Superfície
DTM: Digital Terrain Model, Modelo Digital do Terreno
ESRI: Environmental Systems Research Institute
GIS: Geographic Information System
GNSS: Global Navigation Satellite System
IDW: Inverse Distance Weighting
IGeoE: Instituto Geográfico do Exército
InSAR: Interferometric Synthetic Aperture Radar
ISAE: Image Station Automatic Elevation
JVM: Java Virtual Machine
LiDAR: Light Detection And Ranging
LMAS: Linear Map Accuracy Standard
MSE: Mean Squared Error
ODBC: Open Database Connectivity
RADAR: RAdio Detection And Ranging
RMSE: Root Mean Squared Error
SQL: Structured Query Language
TIN: Triangulated Irregular Network
VBTO: Vegetation, Buildings, Trees, Outliers
VTBO: Vegetation, Trees, Buildings, Outliers
WKB: Well-Known Binary
WKT: Well-Known Text

Glossário

CAD (Computer-Aided Design): designação utilizada para referir a utilização de sistemas de computadores na criação, modificação, análise ou optimização de um *design*. O *output* gerado é, normalmente, guardado em ficheiros.

Cota: distância entre um ponto à superfície terrestre e o nível médio das águas do mar, medida na vertical.

Curvas de nível: representação do relevo modelado, com base na projecção horizontal das linhas que resultam da intersecção da superfície do terreno com um conjunto de planos horizontais, equidistantes entre si.

Equidistância natural: diferença de cotas entre duas curvas de nível vizinhas. Este valor está padronizado para cada escala, sendo de 10m para a escala 1/25 000.

Escala de base: escala para a qual uma carta é elaborada de raiz. Poderão ser produzidas cartas de diferentes escalas, através de operações cartográficas, utilizando o pormenor da carta de escala de base.

Estereorrestituição: aquisição de informação tridimensional do objecto, a partir do modelo estereoscópico orientado absolutamente.

Fotogrametria: análise e processamento de informações geométricas, quantitativas, de uma foto.

Fotogrametria aérea: fotogrametria utilizando fotografias aéreas.

Modelo estereoscópico: modelo virtual do objecto, observável e mensurável tridimensionalmente. Este modelo é obtido através de um par de fotografias do mesmo objecto, orientadas, segundo as regras da estereoscopia.

Ortofoto: fotografia que sofreu alterações, por forma a eliminar distorções projectivas, devidas à inclinação do eixo óptico, e distorções perspectivas, devido ao relevo do terreno. O produto final conjuga a informação visual densa do terreno com a geometria rigorosa de escala homogénea.

Paralaxe: ângulo formado pelas duas rectas geradas a partir de dois pontos de observação diferente sobre o mesmo objecto.

Restituição fotogramétrica: extracção de informação geométrica de um objecto. Os métodos utilizados são a rectificação fotográfica e a estereorrestituição.

Shapefile: formato de ficheiros que contêm informação geoespacial em formato vectorial. É um formato aberto, desenvolvido pela *Environmental Systems Research Institute* (ESRI), para permitir interoperabilidade entre os seus *softwares* e *softwares* terceiros.

Well-Known Binary: ver "*Well-Known Text*".

Well-Known Text: linguagem de marcação de texto, para representar objectos de vectores de geometrias. Tem uma representação equivalente em *Well-Known Binary*, que não é mais que a sua tradução para um *stream* de bytes contíguo, com a intenção de clientes de ODBC, e bases de dados SQL, poderem manipular dados no mesmo *standard*.

World file: ficheiro simples, de texto, utilizado pelos Sistemas de Informação Geográfica, para georreferenciar imagens *raster*.

1 Introdução

Segundo [1, p. 103], "uma carta é uma representação plana, generalizada e reduzida à escala, de uma zona da superfície terrestre, contendo símbolos e informação textual para a descrição dos objectos", podendo ser geográfica ou temática. Além disso, contém sempre uma ou mais quadrículas coordenadas e uma escala. Nas cartas geográficas são representados os objectos existentes à superfície da terra e os pormenores e formas do relevo. Um subdomínio das cartas geográficas são as cartas topográficas, cujo objectivo é a descrição geométrica rigorosa do local, com todo o seu pormenor. A produção cartográfica é o principal campo de aplicação da fotogrametria aérea (análise e processamento de informações geométricas, dos objectos do terreno, a partir de fotografias aéreas).

Em Portugal, o IGeoE é uma entidade de referência como produtor de cartografia, nomeadamente militar, herdando mais de 80 anos de conhecimento e experiência. A sua missão é prover com informação geográfica as Forças Armadas e a comunidade civil, tendo, como produto fundamental, a carta de escala de base 1/25 000 ([2]; cada carta cobre uma área de 16x10 km, correspondente, à escala, a uma dimensão real de 64x40 cm).

A carta era considerada, até há poucos anos, o produto final da fotogrametria ([1, p. 3]). Actualmente, este paradigma foi alterado: o conjunto de dados obtidos é que se constitui como "produto final", que se pode concretizar numa grande panóplia de alternativas (cartas em formato impresso, *raster* ou vector, ortofotos, Modelos Digitais de Elevação – *Digital Elevation Model*, DEM –, outras cartas temáticas, entre outros). Esta evolução confirmou, também, que a produção para a escala base 1/25 000 e as exigências de qualidade são padrões que se deverão manter inalterados para o sucesso da missão do IGeoE.

1.1 Motivação para o tema

A cadeia de produção da carta 1/25 000 no IGeoE engloba um conjunto de etapas complexas e rigorosas, podendo ser resumidas em 3 fases bem definidas:

Figura 1.1 - Cadeia de produção da carta 1/25 000 do IGeoE

A matéria-prima da aquisição e, consequentemente, de toda a cadeia, é a fotografia aérea digital, peça indispensável na produção cartográfica. Segundo [1, p. 3], esta relevância deve-se a vários factores:

- a afinidade entre o conteúdo de uma fotografia aérea e aquilo que se convencionou representar numa carta topográfica;

- a densidade de informação contida numa fotografia;
- a possibilidade de restituir diversos tipos de informação em variadas épocas;
- a possibilidade de mecanizar o processo de produção fotogramétrica;
- a possibilidade de associar o processamento electrónico de dados e a computação gráfica à restituição fotogramétrica;
- a possibilidade de automatizar o processo de restituição, utilizando o processamento digital de imagens e um modelo analítico.

A aquisição da informação propriamente dita é efectuada por restituição estereoscópica que, em fase subsequente, é completada e validada por equipas topográficas que se deslocam ao terreno. Após a validação da informação[1], a mesma é carregada na base de dados geográfica do Instituto, mantendo-a sempre actualizada e disponível. Esta base de dados é a fonte de onde derivam os produtos cartográficos do Instituto, sendo um deles a Carta Militar, série M888, à escala 1/25 000, obtida depois de rigoroso trabalho de edição ([2]).

Embora se consiga resumir a cadeia de produção ao representado na Figura 1.1, na realidade, estão empenhados grandes recursos a nível de *hardware* e humanos, a tempo inteiro, podendo este ciclo demorar entre 3 e 6 meses (dependendo da área geográfica de trabalho). A partir da consulta dos processos das cartas 122 ([3]) e 523 ([4]) da zona do Porto e Pias, criou-se a Figura 1.2, que representa a duração da produção.

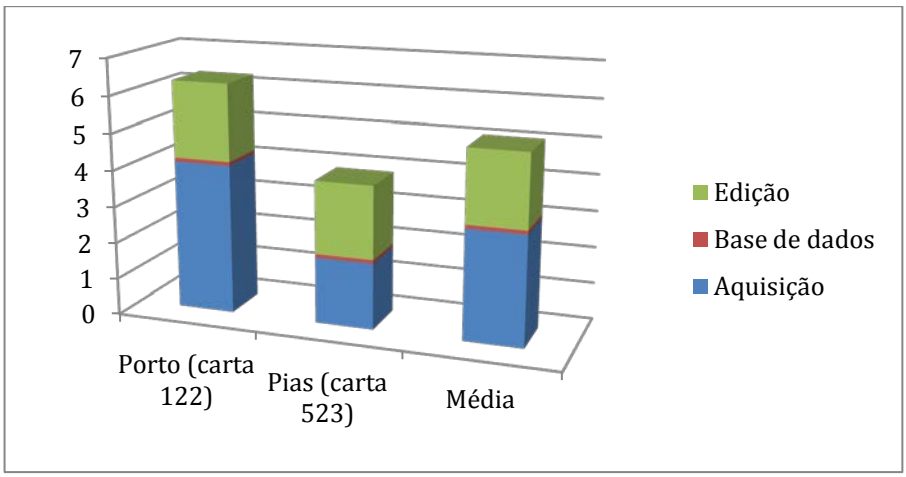


Figura 1.2 - Tempos das fases da cadeia de produção das cartas do Porto e Pias (meses)

As cartas 122 e 523 fazem parte das mais de 600 cartas, à escala 1/25 000, produzidas no IGeoE, e representam, empiricamente, as áreas mais e menos trabalhosas do território nacional. Um extracto do cartograma deste Instituto pode ser visto na Figura 1.3.

[1] Processo constituído por duas fases: a pré-validação e a validação. Na primeira, são analisadas as emendas detectadas pela Secção de Controlo de Qualidade nos vários temas (rede geodésica, toponímia, cadastro e informação vectorial), para facilitar o trabalho de validação da informação vectorial. Esta é executada na segunda fase, a validação, onde se garante a coerência semântica e morfológica da informação.

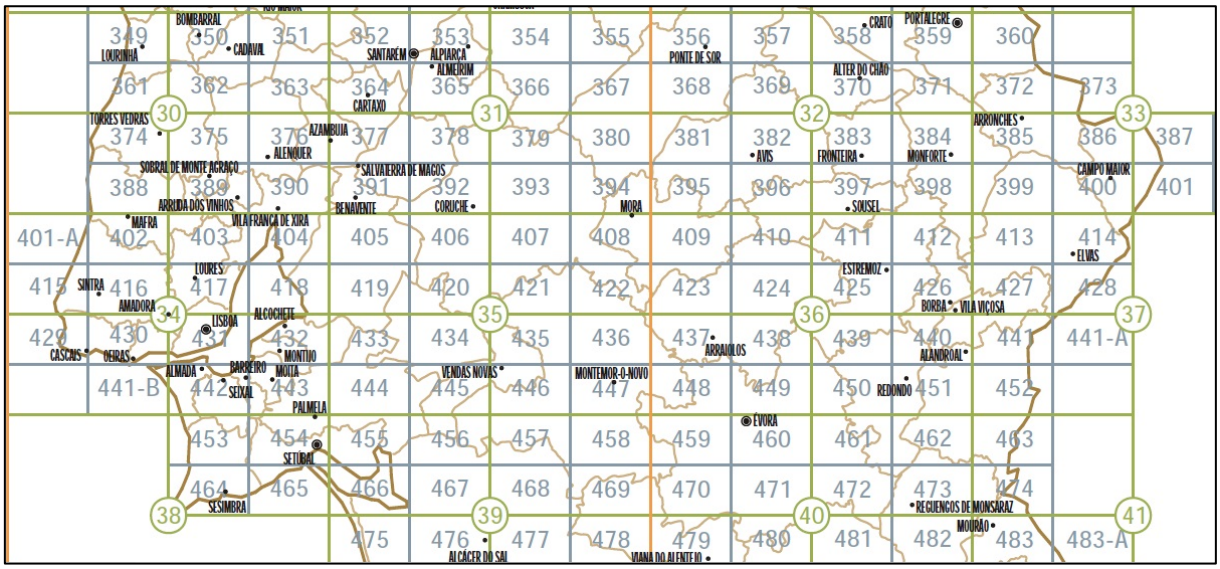


Figura 1.3 - Extracto do cartograma do IGeoE (quadriculado e numeração a azul correspondem às cartas à escala de 1/25 000, a verde à escala 1/50 000 e a laranja a 1/250 000; fonte: [2])

Conforme espelhado na Figura 1.2, a aquisição é a fase que exige mais tempo, numa razão de aproximadamente de 3/5. Verifica-se, também, que o tempo dispendido na edição é relativamente constante. Por sua vez, conforme se pode observar na Figura 1.4, a fase que mais tempo consome, durante a aquisição, é a restituição fotogramétrica.

Figura 1.4 - Tempos relativos durante a aquisição (informação coligida do documento de Gestão de Folhas do Departamento de Processamento de Dados do IGeoE)

A nível de recursos humanos e materiais, enquanto que a actualização da informação da base de dados e a edição são processos realizados por apenas uma pessoa, numa estação de trabalho ordinária (há apenas o requisito do monitor possuir bom brilho), na aquisição/restituição trabalham, em simultâneo, cerca de 20 operadores. Cada operador trabalha numa estação fotogramétrica com um preço estimado de cerca de 7 500 € (processador e placa gráfica topo de gama), equipada com um kit stéreo (monitor e óculos 3D) e um rato de restituição, cujos preços estimados são na ordem dos 650 € e 2 100 €, respectivamente (valores de 2013).

Realça-se, ainda, que a restituição fotogramétrica é um processo que não dispensa a intervenção humana, e que qualquer automatização nesta fase se traduz directamente numa redução de tempo de produção global e, por conseguinte, na redução de intervenção humana (libertando-a para outras tarefas) e de utilização de recursos (libertando-os para outras tarefas ou mesmo dispensando-os), minimizando o custo de produção – esta é a motivação para o desenvolvimento do presente tema.

1.2 Objectivo

Durante a fase da restituição, os operadores restituem todos os objectos visíveis na fotografia aérea (edifícios, redes viárias, hidrografia, vegetação…), em ambiente de estereoscopia, segundo as Normas de Aquisição da Carta Militar 1/25 000 ([5]). Essas normas definem as características que os objectos têm de possuir para ser restituídos, bem como as regras para a sua restituição. Fundamentalmente, essas características especificam os elementos de interesse militar, ignorando os restantes – nomeadamente árvores isoladas ou pequenos grupos de árvores, objectos sem relevância para a escala de trabalho.

Um dos aspectos fundamentais, numa carta topográfica, é o relevo. Na carta 1/25 000, doravante designada apenas por "Carta", aquele é representado por curvas de nível, que são os objectos de maior extensão e em maior quantidade em qualquer carta topográfica (a Figura 1.5 esquematiza o que são as curvas de nível).

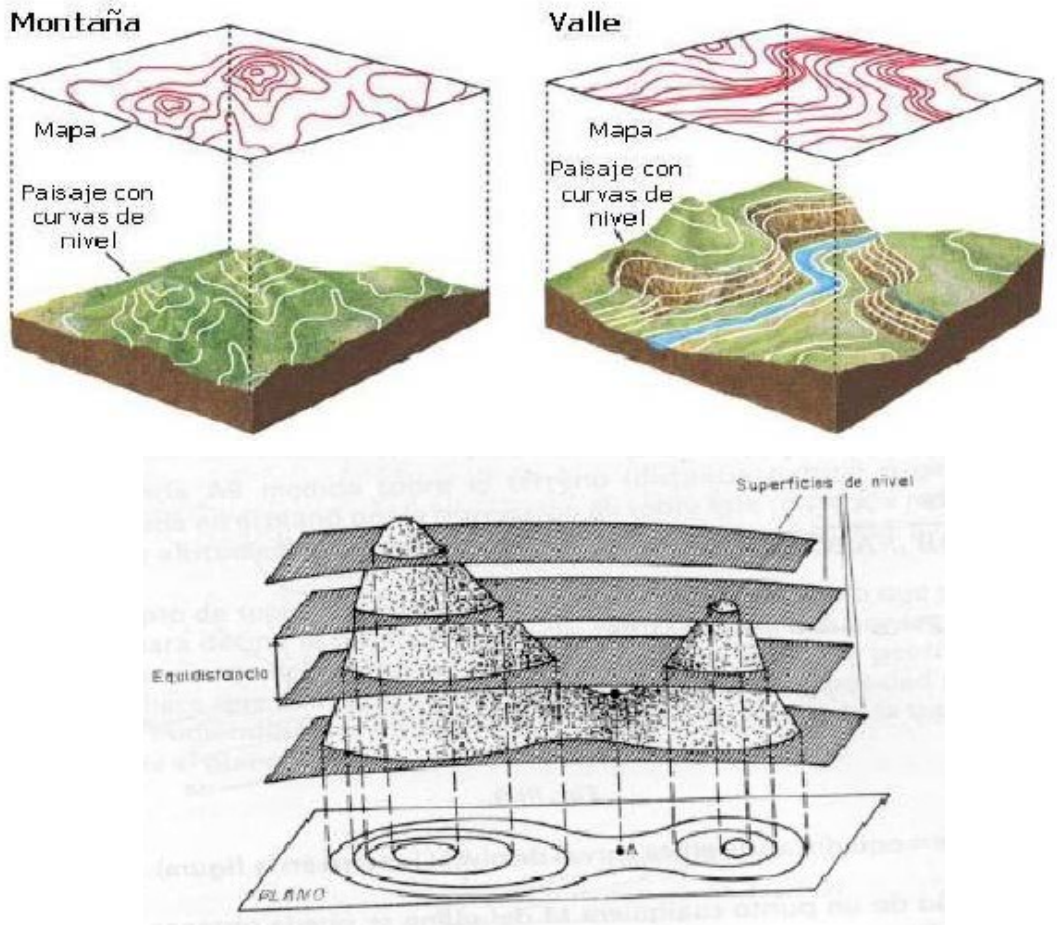


Figura 1.5 - Curvas de nível (fonte: [6]**)**

Comparando a Figura 1.6 com a Figura 1.7, na qual as curvas de nível correspondem às linhas de cor sépia, pode-se reparar que esta última, que é apenas constituída por curvas de nível, parece ter quase a mesma quantidade de informação que a primeira. As curvas de nível são, também, restituídas pelos operadores.

O ambiente de estereoscopia é gerado através de duas fotografias aéreas verticais que apresentem uma zona de sobreposição, como demonstrado na Figura 1.8 (para efeitos de estereorrestituição é exigida, normalmente, uma sobreposição longitudinal de 60%; [1, p. 5]). A partir delas, e com equipamento adequado (kit stereo – monitor e óculos 3D), é possível visualizar o terreno a 3 dimensões, embora geralmente com um exagero do relevo, devido ao facto das imagens serem obtidas de pontos bastante distantes entre si.

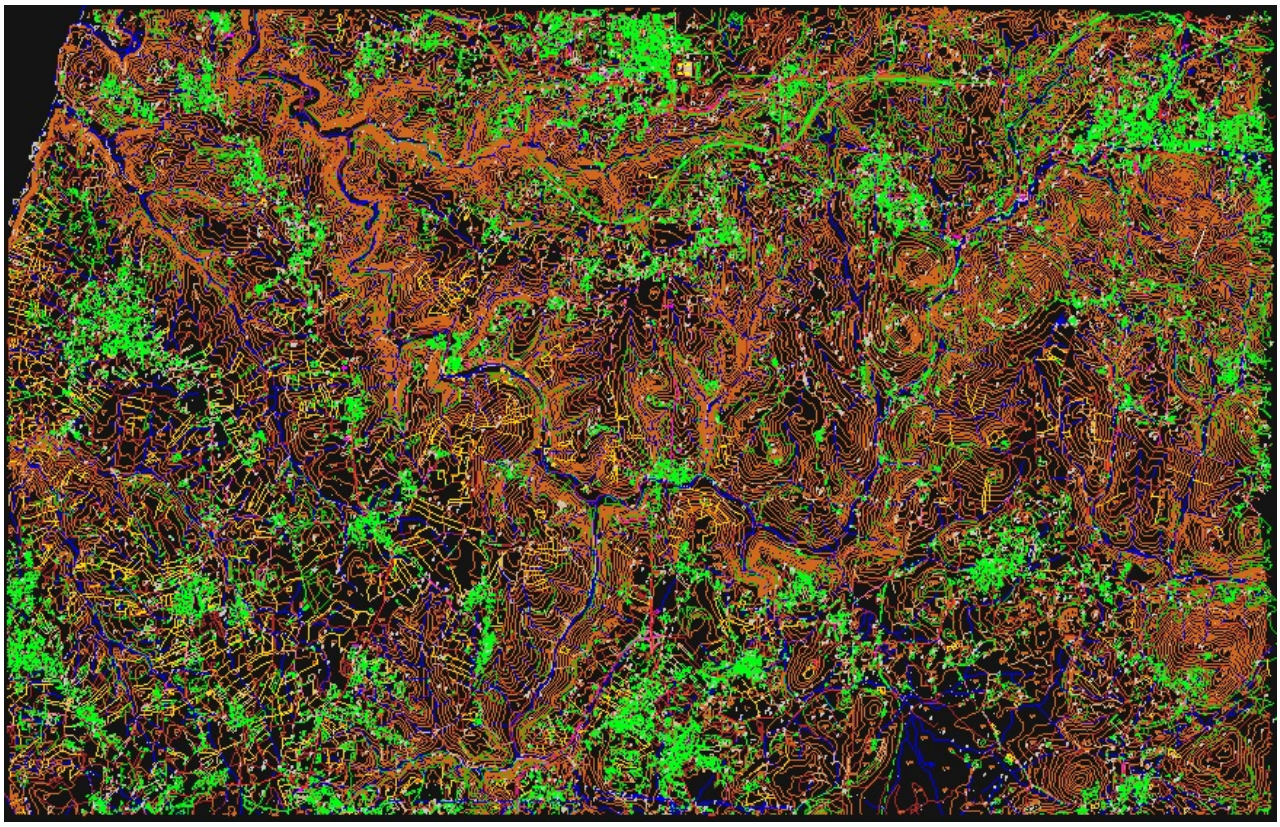


Figura 1.6 - Carta 1/25 000 nº402: informação vectorial completa

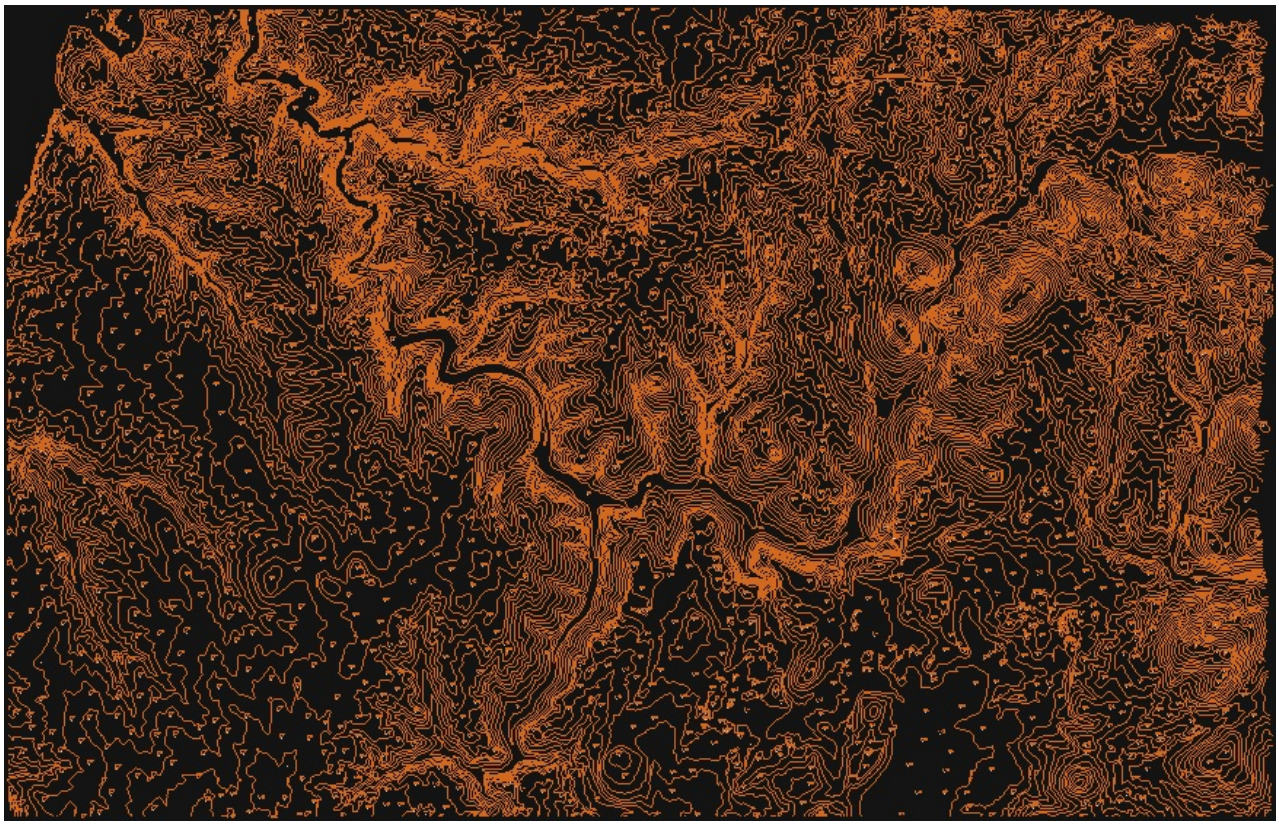


Figura 1.7 - Carta 1/25 000 nº402: informação vectorial altimétrica (curvas de nível)

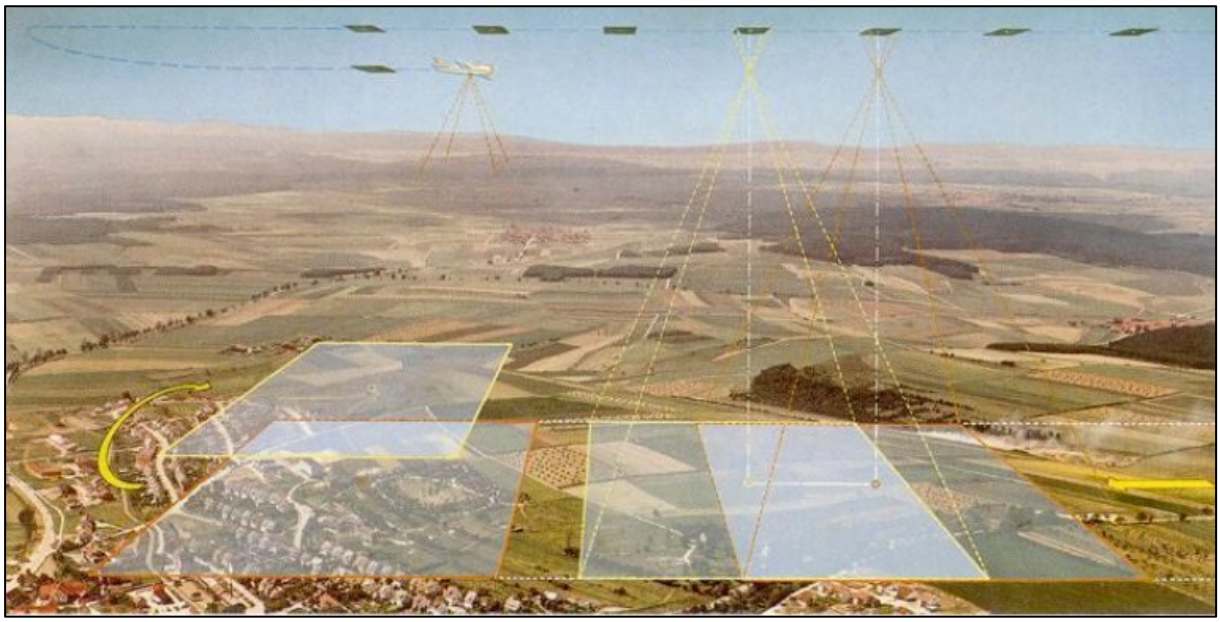


Figura 1.8 - Esquema de voo fotogramétrico, realçando sobreposição da área coberta pelas fotografias (fonte: [1])

Através de um par de fotografias consecutivas e orientadas é possível não só obter as suas coordenadas em X e Y, mas também em Z, pelo cálculo da paralaxe (ângulo formado por duas rectas geradas a partir de dois pontos de observação diferentes sobre o mesmo objecto).

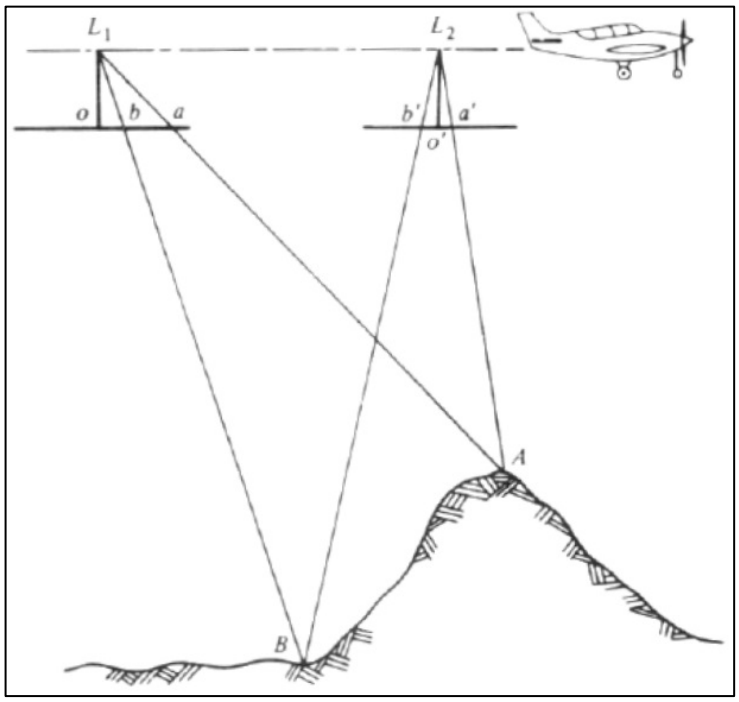


Figura 1.9 - Esquema de duas fotografias aéreas consecutivas sobre a mesma área (fonte: [7]**)**

Conforme se pode observar na Figura 1.9, as fotografias L1 e L2 capturaram os pontos A e B. Como a posição do avião é conhecida, é possível calcular o ponto de intersecção de ambas as rectas originadas na observação, quer do ponto A, quer do ponto B, através das equações de colineariedade.

A restituição das curvas de nível (na carta à escala 1/25 000, cada curva de nível é marcada de 10 em 10 m) é um pouco mais exigente e bastante morosa, porque, para além de, como já referido, haver curvas de nível em toda a extensão da carta, não há nenhuma referência, no solo, que materialize estes objectos da Carta.

Os operadores, nas suas estações de trabalho, conseguem obter a cota de cada ponto, bastando, para tal, colocar o cursor sobre o terreno, no ponto pretendido; desta forma, quando iniciam a restituição de uma curva de nível em estereoscopia, têm de acompanhar as formas do terreno com o cursor, e ir marcando vários pontos correspondentes à cota da curva de nível que estão a restituir (um conjunto de pontos de cota designa-se, genericamente, por **nuvem de pontos**). O que impede este processo de se tornar automático é que o processo de correlação de imagens calcula a cota da superfície, e por exemplo, numa casa ou arvoredo denso, a cota que se obtém é a do topo desses objectos e não do solo, o que implica observação e intervenção humanas para a restituir sob os objectos.

A minimização do tempo despendido na restituição das curvas de nível é alvo de bastantes estudos, não existindo, até ao momento, uma forma automática de o fazer com resultados satisfatórios ([8, p. 7]). A literatura relativa a este assunto aponta, como caminho futuro, uma mistura de algoritmos e recursos para se obter o resultado pretendido ([9, p. 89], [10, p. 139], [11, p. 60], [12, p. 1] ou [8, p. 7]), e passará por efectuar uma filtragem de uma nuvem de pontos cotados (neste contexto, filtragem é a designação do processo utilizado para retirar o "ruído" de um ponto de cota; o "ruído" num ponto de cota marcado sobre um edifício é a altura do próprio edifício), utilizando outra informação acessória, mas fundamental, para ajudar na decisão.

Um outro recurso de grande utilidade é a ortofoto da região a analisar (as ortofotos são fotografias aéreas rectificadas, por forma a eliminar as distorções provocadas pela inclinação do eixo óptico e pelo relevo, gerando um plano de escala rigorosa): tem a mesma informação abundante que as próprias fotografias possuem, aliada ao facto de ser uma representação de escala rigorosa. Estes factores garantem informação contextual de qualquer pormenor do terreno, com exactidão.

É intenção do presente trabalho propor uma metodologia automática para eliminar o ruído das nuvens de pontos, por forma a gerar automaticamente curvas de nível, solucionando os pontos-chave referidos até aqui:

- utilizar os objectos sobre o solo restituídos pelos operadores (mais concretamente, os edifícios e as áreas de vegetação superiores a 150 m x 150 m);
- identificar os restantes objectos sobre o solo sem interesse militar, mas que interferem na geração das curvas de nível (nomeadamente árvores isoladas e pequenos grupos de árvores);
- filtrar os pontos de cota marcados sobre todos os objectos referidos nos pontos anteriores, fazendo baixar essa cota até ao nível do solo;
- contribuir, de forma decisiva, para a redução do tempo de produção de uma carta militar.

1.3 Contribuições para o trabalho

Para a elaboração do trabalho colaboraram, de forma decisiva, diversos elementos do IGeoE e da FCT/UNL:

- Definição e delimitação do tema:

  Prof. Doutor Adriano Lopes, professor auxiliar do Departamento de Informática (DI) da FCT/UNL, Prof. Doutor Fernando Birra, Tenente-Coronel Rui Dias e Major Francisco Salvador, Oficiais do IGeoE.

- Detalhe de aprendizagem automática, especificamente algoritmos de *boosting* e *Adaboost*:

  Prof. Doutor Ludwig Krippahl, professor auxiliar do DI da FCT/UNL.

- Código inicial de alguns dos algoritmos utilizados:

  Pedro Martins, aluno nº41976 da FCT/UNL, no trabalho de Processamento de Imagem em Detecção Remota da cadeira de Programa de Introdução à Investigação Científica (2013/14).

- Acompanhamento, discussão da evolução do trabalho, dos resultados e de alternativas:

  Prof. Doutor Fernando Birra, Tenente-Coronel Paulo Domingos, Tenente-Coronel Rui Dias, Capitão António Franco e Sargento-Ajudante José Dias.

- Estações de trabalho, máquinas virtuais e restante apoio informático:

  Departamento de Informática e Gestão da Informação do IGeoE, e da sua Secção de Gestão de Informação, chefiados pelo Major Francisco Salvador e Capitão Henrique Azevedo, respectivamente.

- Disponibilização dos diferentes recursos de informação utilizados:

  Departamento de Aquisição de Dados, e da sua Secção de Fotogrametria, através do Tenente-Coronel Rui Dias, Capitão António Franco e Alferes Ana Marques.

- Verificação e validação dos resultados da metodologia:

  Sargento-Chefe Octávio Mestre.

1.4 Estrutura da dissertação

A presente dissertação está dividida em quatro capítulos essenciais:

1. O primeiro capítulo contém o enquadramento da situação a resolver, integrado na cadeia de produção do IGeoE. Define ainda os conceitos básicos e raciocínios aplicados e utilizados nos capítulos seguintes, bem como sistematiza o objectivo do trabalho.
2. O segundo capítulo é constituído pela componente teórica analisada durante a investigação, e descrição e detalhe dos recursos aplicados na metodologia proposta.
3. O terceiro capítulo pormenoriza a implementação da metodologia, materializada na aplicação *AutoDTM*. A primeira secção descreve o plano de trabalho, onde se realça o fluxo de execução geral que, para além de servir de enquadramento ao particular, descrito na secção seguinte, sugere também a forma de integração na cadeia de produção do IGeoE. A segunda secção descreve, passo a passo, toda a execução da aplicação final, desde o pré-processamento dos dados de entrada, passando por todas as filtragens. Neste ponto, e por intermédio de simbologia própria, são realçados diferentes pormenores relacionados com opções de implementação, optimizações futuras, e limitações.
4. O quarto capítulo é composto pela avaliação de resultados, descrevendo as condições utilizadas para a execução dos testes e comprovando a eficácia da metodologia proposta.
5. O quinto capítulo é composto pelas conclusões da dissertação, bem como o levantamento de eventual trabalho futuro, uma vez que, ao longo do quarto capítulo, e através de simbologia própria, já se terem realçado os diferentes aspectos que poderão ser melhorados *a posteriori.*

2 Trabalho Relacionado

2.1 Enquadramento

O DEM representa uma superfície 2D mergulhada num espaço 3D ([1]), e para a sua geração são necessárias amostras de pontos cotados. Se estas amostras contiverem a cota de objectos sobre o solo, como edifícios ou arvoredo, o modelo resultante é designado de Modelo Digital de Superfície (*Digital Surface Model*, DSM). Por outro lado, se os pontos cotados descreverem o relevo ao nível do terreno, o modelo gerado terá a designação de Modelo Digital do Terreno (*Digital Terrain Model*, DTM).

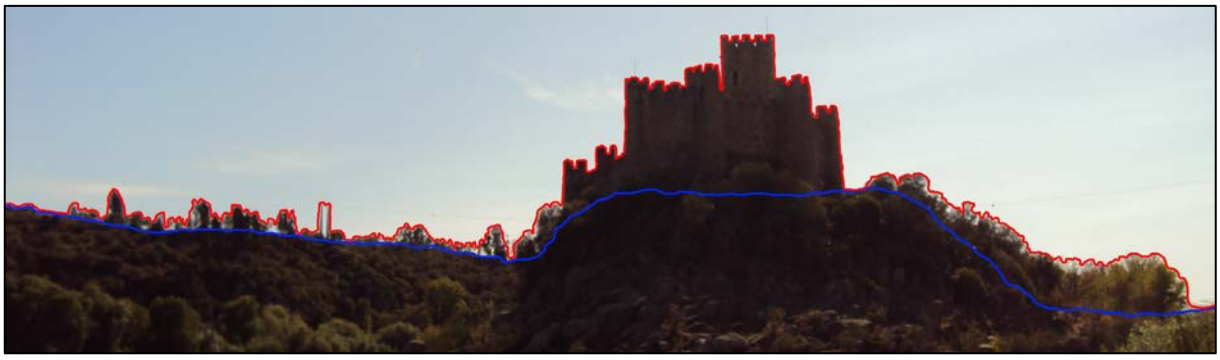


Figura 2.1 - Diferença entre DTM (linha a azul) e DSM (linha a vermelho)

Essas amostras estarão organizadas como uma grelha regular ou irregular, tomando a designação genérica de **nuvem de pontos**. A geração da superfície propriamente dita é geralmente concretizada pela união desses pontos em quadrados ou triângulos, ou seja, em *grid* ou numa *Triangular Irregular Network (TIN)*.

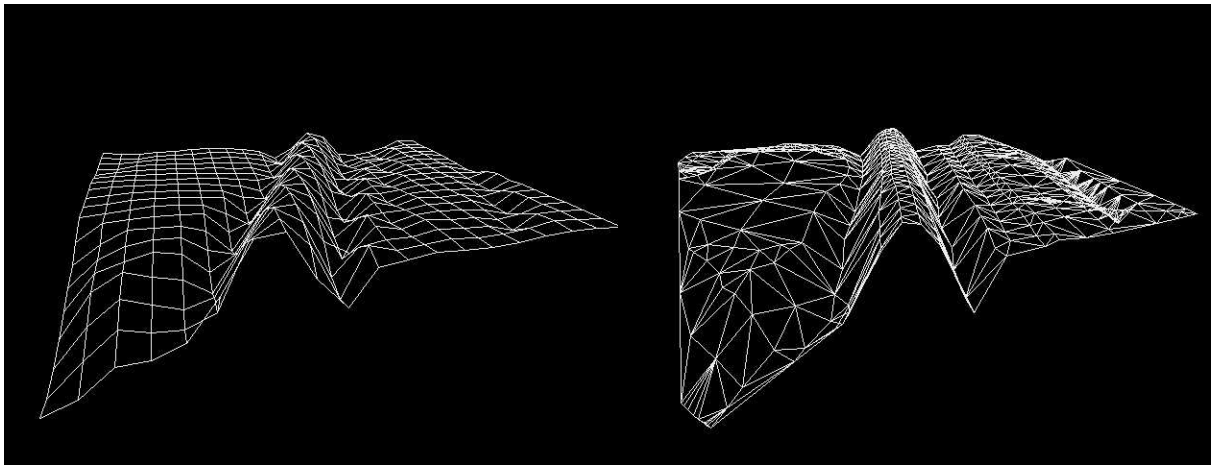


Figura 2.2 - Superfícies *grid* (à esquerda) e TIN (à direita), criadas a partir de nuvens de pontos diferentes (regular e irregular, respectivamente; fonte: [63]**)**

Qualquer uma das abordagens tem vantagens e desvantagens: as *grid* têm espaçamento fixo entre pontos, tornando-se mais fácil quer o seu armazenamento (uma matriz), quer o seu processamento (numa matriz consegue-se, com facilidade, obter os pontos vizinhos de determinado ponto), embora o terreno fique com menor caracterização (qualquer diferença no relevo inferior ao espaçamento entre pontos não vai conseguir ser caracterizada); as TIN permitem modelar e representar melhor o relevo do terreno, mas o armazenamento e processamento destes pontos/triângulos é mais complexo.

As amostras primárias podem ser obtidas pelos seguintes métodos ([1]):

- **Estereofotogrametria:** o operador humano mede as cotas no modelo estereoscópico do terreno. Dificilmente criará uma distribuição regular de pontos, mas tanto pode recolher amostras para um DSM como para um DTM;
- **Correlação automática de imagens:** realizada por processamento digital de imagem – determinação de pontos homólogos em pares estereoscópicos, após o qual é possível determinar a cota associada. Apenas recolhe amostras para DSM numa distribuição regular, e pode criar nuvem de pontos muito densas (no máximo de um ponto por *pixel* da imagem);
- **Varrimento laser (*Light Detection And Ranging*, LiDAR):** este método baseia-se num voo efectuado por um avião equipado com um *Global Navigation Satellite System* (GNSS) e com um dispositivo emissor/receptor de feixes de raios laser: à medida que o avião se desloca, este dispositivo vai emitindo impulsos laser para a superfície terrestre e capturando as reflexões, obtendo as coordenadas 3D de cada um desses pontos. Gera amostras muito densas, o que, em zonas urbanizadas, ou mesmo em arvoredo denso, permite obter pontos ao nível do solo. Recolhe amostras para DSM, mas também para DTM (com limitações, nomeadamente em zonas urbanas), numa distribuição irregular, criando nuvens de pontos extremamente densas;
- **Interferometria de radar (*Interferometric Synthetic Aperture Radar*, InSAR)**: método semelhante ao anterior, embora o dispositivo utilizado trabalhe com ondas rádio. Consegue espaçamentos de pontos de cerca de 5 m, recolhendo amostras apenas para DSM, numa distribuição irregular.

As curvas de nível podem ser adquiridas em estereoscopia por um operador, ou automaticamente a partir de um DTM. Um dos métodos de interpolação de curvas de nível consiste na determinação de pontos das cotas pretendidas, unindo-se pontos de igual cota por segmentos de recta. De seguida, a linha poligonal obtida é adoçada, obtendo-se a curva de nível pretendida ([1]).

Já foi identificado, como factor fundamental, a necessidade da existência de uma nuvem de pontos que descreva um DSM, e que se pretende transformá-lo num DTM; essa transformação é executada através de um processo de filtragem de pontos.

2.2 Filtragem

Como referido anteriormente, a amostra primária é constituída por uma nuvem de pontos com ruído, ou seja, por pontos cuja cota nem sempre corresponde à cota do solo, em virtude do seu valor de altimetria representar o topo de edifícios ou árvores, por exemplo. O processo para eliminação desse ruído é designado por **filtragem**, e o seu resultado é uma amostra secundária de pontos através da qual é possível gerar-se um DTM.

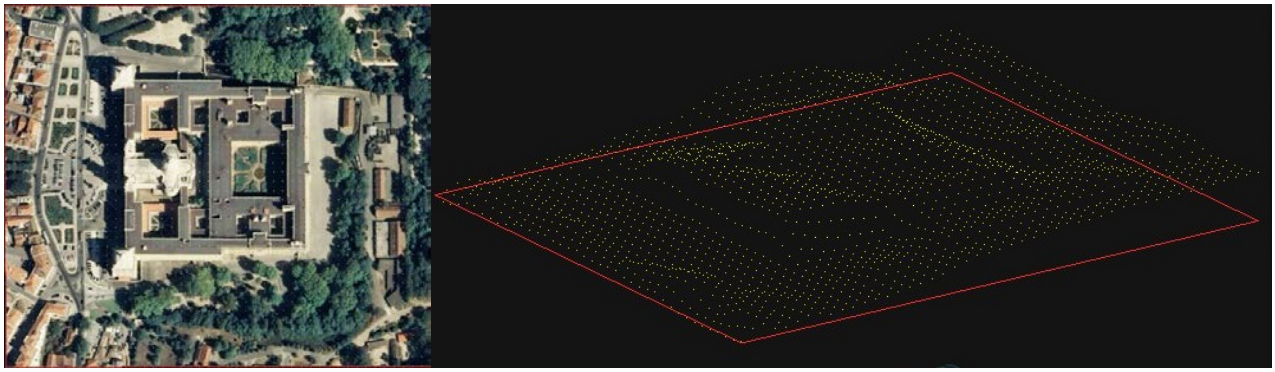


Figura 2.3 - À esquerda, extracto de ortofoto de zona sobre o convento de Mafra com nuvem regular de pontos sobreposta. À direita, a mesma nuvem de pontos da mesma zona, em projecção axonométrica

Como se pode observar na Figura 2.3, a nuvem regular de pontos (espaçamento de 10m) obtida por correlação automática de imagens sobre o convento de Mafra, contém as cotas do topo do convento. Se se aplicasse um algoritmo de filtragem nestes pontos, aqueles sobre o convento deveriam ver a sua cota alterada para a cota dos pixéis vizinhos que estão ao nível do solo, ou seja, a superfície gerada por esses pontos deveria ser plana.

As figuras Figura 2.4 e Figura 2.5 mostram um exemplo de nuvem de pontos em projecções e ampliações diferentes. Esta nuvem foi gerada por correlação automática de duas fotografias aéreas (através de [13]) e contém 194 265 pontos, espaçados entre eles por 10 metros, cobrindo uma área de dimensão aproximada 2600x7200 m.

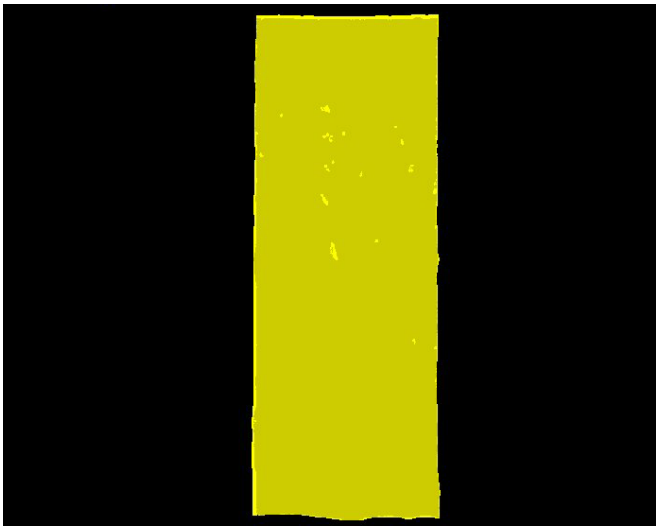


Figura 2.4 - Nuvem de pontos (fonte: [14]**)**

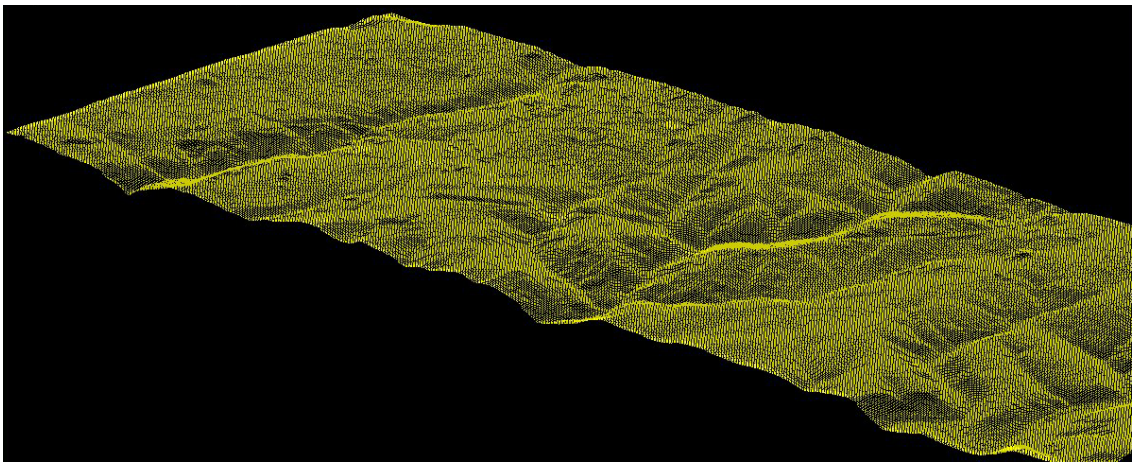


Figura 2.5 - Nuvem de pontos em projecção oblíqua (fonte: [14]**)**

Os métodos descritos infra (2.2.1) poderão ser aplicados em quaisquer nuvens de pontos, independentemente do método da obtenção da amostra primária, da distribuição e da densidade da nuvem de pontos (seja por fotogrametria ou por varrimento laser, conforme [9, p. 84]).

2.2.1 Características dos métodos de filtragem

Nos pontos seguintes descrevem-se os algoritmos de filtragem, tendo por base [8, pp. 1–2]. Este artigo tem grande importância para o presente trabalho, visto ser uma súmula do estado actual quanto a algoritmos de filtragem: não só sintetiza as suas características, como retira conclusões sustentadas das suas capacidades e limitações.

Estruturas de dados

As amostras primárias produzem nuvens de pontos irregulares ou regulares (dos métodos descritos em 2.1, apenas a correlação automática de imagens cria uma grelha regular directamente). Como se verá nos pontos seguintes, as decisões tomadas por estes algoritmos para classificar os pontos como "terreno" ou "não terreno" passam por se fazer comparação do ponto em questão com os seus pontos vizinhos.

No caso de uma nuvem regular de pontos, esta poderá ser estruturada numa matriz de *m* linhas por n colunas, em que cada elemento corresponde a um ponto. Como é regular, a distância entre os elementos/pontos é conhecida e constante. Além disso, a obtenção da vizinhança de determinado ponto torna-se trivial. Por exemplo, os oito pontos vizinhos do elemento da matriz na posição (5, 5) são (4, 4), (4, 5), (4, 6), (5, 4), (5, 6), (6, 4), (6, 5) e (6, 6). O valor de cada elemento nesta matriz é a sua cota que, juntamente com as coordenadas na matriz, formam as coordenadas 3D de cada ponto (claro que, para se obterem as coordenadas absolutas do ponto tem de se somar ao X e ao Y a coordenada real correspondente à origem da matriz, bem como a multiplicação dos índices da matriz pelo espaçamento de pontos). Uma matriz que guarda apenas um valor numérico, é uma estrutura bastante simples.

…	4	5	6
4	101	102	103
5	101	102	102
6	101	101	101

Figura 2.6 - Exemplo de matriz de armazenamento de nuvem regular de pontos (célula com valor da cota de cada ponto)

Numa nuvem irregular de pontos, a complexidade espacial é obrigatoriamente maior. Poder-se-iam guardar os dados numa matriz, mas as coordenadas de cada elemento, nessa matriz, não teriam qualquer significado útil para a computação.

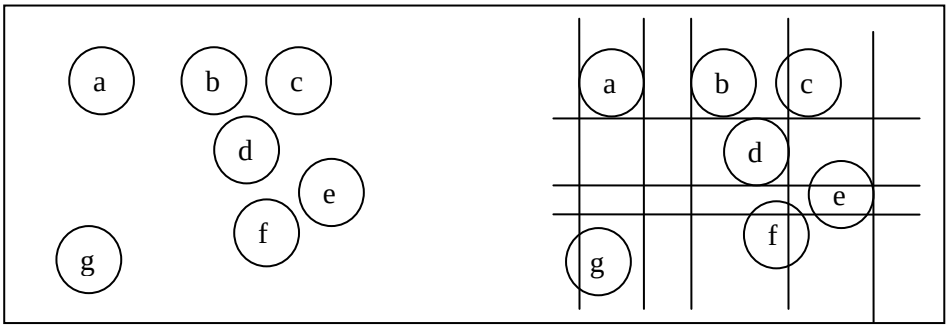


Figura 2.7 - Esquema de nuvem irregular de pontos e tentativa de os acomodar numa matriz

A Figura 2.7 representa 7 pontos de uma nuvem irregular. Tentar fazer a sua representação numa matriz é um exercício de alguma dificuldade, e qualquer vantagem que daí possa advir duvidosa. A imagem à direita tenta fazer uma acomodação dos pontos a uma matriz, criando diversas situações problemáticas, sendo uma delas a existência de células vazias. Além disso, como representam pontos de coordenadas X, Y e Z, cada elemento da matriz deveria ser composto por 3 valores numéricos, e todos estes problemas inviabilizam uma solução optimizada em matriz. Uma alternativa seria organizar os pontos num grafo, como representado na Figura 2.8.

Este seria o cenário mais viável para estruturar os pontos de espaçamento irregular num algoritmo de filtragem: cada ponto passa a "conhecer" os seus vizinhos com facilidade. No entanto, em termos de complexidade espacial, um grafo é uma estrutura mais complexa que uma matriz, e cada nó do grafo tem de conter informação das coordenadas X, Y e Z de cada ponto. Um último aspecto de relevo, e que será detalhado nos pontos seguintes, prende-se com o processamento que o algoritmo tem de efectuar quando compara um ponto com os vizinhos: como as distâncias entre pontos vizinhos

não são constantes, ter-se-ia de realizar mais operações matemáticas (pelo menos para calcular a distância entre pontos) do que no caso de uma matriz.

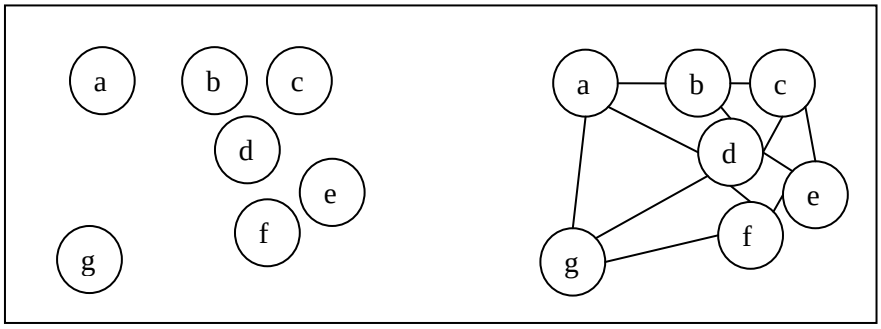


Figura 2.8 - Esquema de nuvem irregular de pontos organizada num grafo

Os parágrafos anteriores salientam as vantagens de uma estruturação de dados em matriz. Com o intuito de tirar partido dessas vantagens, nas situações em que a amostra primária é irregular, há alguns algoritmos que realizam um pré-processamento sobre a nuvem de pontos, transformando-a numa regular, por interpolação.

Testes na vizinhança dos pontos

Os filtros fazem comparação de pontos operando numa vizinhança local, classificando um ou mais pontos de cada vez como terreno ou não-terreno. Há três métodos de classificação de pontos:

- **Ponto a ponto**: nestes algoritmos, a função discriminante compara os pontos dois a dois e, caso o resultado seja superior a determinado valor, então um desses pontos é classificado como pertencente a um objecto (não-terreno);
- **Ponto a pontos**: a função discriminante avalia vários pontos vizinhos do ponto de interesse, para o classificar. Apenas o ponto de interesse é classificado;
- **Pontos a pontos**: nestes algoritmos, vários pontos são utilizados como *input* da função discriminante, e mais de um ponto recebe classificação.

Medição da descontinuidade

A maioria dos algoritmos faz, na sua função discriminante, uma medição de descontinuidade. Algumas das medições utilizadas são a diferença de cotas, declives, distância mais curta a triângulos de modelo TIN, e distância mais curta a superfícies paramétricas.

Critério de filtragem

Todos os filtros fazem uma assunção acerca dos pontos pertencentes à superfície do terreno numa vizinhança local. Esta assunção designa-se por critério de filtragem ([15, p. 853] também enumera estas características). Os pontos seguintes descrevem os diferentes conceitos, estando cada um representado na Figura 2.9, por ordem, da esquerda para a direita:

- **Por declive**: nestes algoritmos, é medido o declive ou a diferença de cotas entre pontos. Se o declive excede determinado valor, então assume-se que o ponto de maior cota é não-terreno;
- **Por bloco mínimo**: a função discriminante é representada por um plano horizontal com uma zona de *buffer*. Pontos exteriores a esse bloco são classificados como não-terreno;
- **Por superfície**: nestes algoritmos, a função discriminante é representada por uma superfície paramétrica e respectiva zona de *buffer*. Tal como o anterior, pontos fora desse *buffer* são classificados como não-terreno;
- **Por *clustering***: neste critério de filtragem criam-se *clusters* de pontos com cotas semelhantes. Os pontos que os constituem recebem a classificação de "não-terreno", se a cota desse *cluster* for superior à cota dos *clusters* vizinhos.

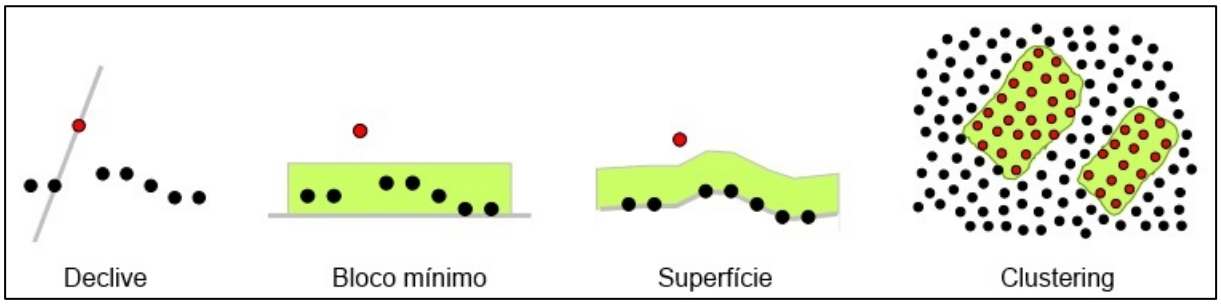


Figura 2.9 - Critério de filtragem (fonte: [8]**)**

Execução simples ou iterativa

A classificação pode ser feita apenas num varrimento, ou iterativamente. No primeiro caso, prefere-se uma menor complexidade do algoritmo, em troca de precisão e, no segundo, o inverso. À partida, num processo iterativo, consegue-se obter mais informação sobre a vizinhança de um ponto, sendo a sua classificação, desta forma, mais fiável.

Tipo de alteração: por remoção ou substituição

Após classificação de um ponto, este pode ser retirado da nuvem de pontos, ou pode ser substituído por outro com uma cota diferente. De forma geral, algoritmos que operam sobre nuvens de pontos irregulares fazem remoção de pontos, e os que operam sobre nuvens regulares fazem substituição. Neste caso, se fizessem remoção, a nuvem deixava de ser regular, invalidando os pressupostos decorrentes da utilização de uma grelha regular. Já no primeiro caso a remoção é viável, porque as operações que se realizam a partir destas nuvens, como por exemplo a geração de TIN, já executam interpolações, pelo que é desnecessário estar a repetir operações (caso fizessem remoção, a nuvem deixava de ser regular, invalidando os pressupostos decorrentes da utilização de uma grelha regular).

2.2.2 Comparação de algoritmos de filtragem

O principal objectivo de [8] foi avaliar a qualidade de algoritmos de filtragem diferentes e, para tal, foram aplicados a um conjunto de áreas de terreno com características bastante específicas. Esses algoritmos aplicaram um misto dos princípios descritos em 2.2.1, e dos testes executados realçam-se alguns pormenores que espelham as principais dificuldades:

- **Objectos muito grandes** (referido em [8, p. 7]): como muitos algoritmos analisam vizinhanças pequenas, objectos grandes que excedam essa vizinhança não são filtrados;
- **Cenários complexos** (generalidade dos algoritmos não filtrou satisfatoriamente): encontram-se principalmente em zonas urbanas com presença de árvores, edifícios de diferentes configurações, pátios interiores ou ruas paralelas e desniveladas, que fazem "confundir" os algoritmos quanto àquilo que devem assumir como "terreno";
- **Terreno não contíguo** (maiores erros nos algoritmos [16]–[18]): em ambientes urbanos, pátios ou jardins estão circundados por edifícios. Para a filtragem decidir bem nestes ambientes, a parametrização da função discriminante tem de ser ajustada;
- **Edifícios e vegetação em elevações do terreno** (maiores erros nos algoritmos [19], [20]): dificuldade na classificação, devido à diferença de cotas entre, pelo menos, uma das faces do edifício e o terreno ser mínima; a mesma lógica se aplica para a vegetação;
- **Pontes** ([21]): pontos sobre pontes devem ser classificados como não-terreno, embora a estrada, nas extremidades, deva ser classificada como terreno. A maioria dos algoritmos identifica correctamente as pontes, embora não classifique bem onde começam e onde acabam;
- **Ladeiras** (generalidade dos algoritmos não filtrou satisfatoriamente): devido ao grande declive, podem ser classificadas erradamente como não-terreno;

- **Vegetação baixa** (generalidade dos algoritmos não filtrou satisfatoriamente): devido à pouca diferença de cota, todos os pontos podem ser classificados como terreno. Para haver uma decisão correcta, a parametrização da função discriminante tem de ser ajustada.

Assim, segundo [8], as conclusões sobre algoritmos de filtragem são as seguintes:

- os algoritmos de filtragem deverão classificar determinadas estruturas como terreno e outras como não-terreno. Este raciocínio funciona correctamente para a maioria dos casos, mas não é válido para todos os pormenores do terreno. Esta falha deve-se ao facto de os filtros serem cegos, quanto ao contexto das estruturas em relação à vizinhança;
- o critério de filtragem e tamanho da vizinhança, bem como os parâmetros da função discriminante, têm de ser ajustados consoante o terreno a avaliar, para se conseguirem resultados satisfatórios;
- não é possível efectuar, satisfatoriamente, uma filtragem de uma nuvem de pontos de uma grande área, por um processo totalmente automático (também confirmado em [10, p. 150], ao ser referido que ainda não foi encontrado um algoritmo de filtragem totalmente automático, para aplicação em qualquer tipo de terreno);
- maior densidade de pontos não implica melhores resultados;
- algoritmos baseados num critério de filtragem por segmentação, ou um híbrido que também envolva este conceito, obtiveram resultados mais promissores;
- algoritmos que avaliam a medida de descontinuidade com superfícies, obtêm melhores resultados, já que, com essas superfícies, se alarga o contexto da vizinhança;
- dever-se-á utilizar informação adicional, como imagens, para extrair mais semântica do contexto dos pontos a classificar; a informação obtida simplesmente pelas coordenadas do ponto é manifestamente insuficiente;
- todos os algoritmos de filtragem têm dificuldades em três tipos de superfícies: terrenos acidentados ou com inclinação descontínua, arvoredo denso e regiões de vegetação baixa ([15, p. 853]).

2.3 Informação geográfica produzida no IGeoE

As incapacidades dos algoritmos de filtragem puros estão bem identificadas e, como consequência, também as pistas para possíveis soluções, com vista a colmatá-las. Uma das conclusões enumeradas em 2.2.2 é a falta de semântica numa nuvem de pontos em relação a cada ponto e ao seu contexto, e procurou-se desenvolver este aspecto nesta dissertação.

O IGeoE, como já referido, é uma entidade produtora de informação geográfica e produz uma grande panóplia de produtos diferentes. Deste facto também deriva que possui um histórico de todo o trabalho que efectua, e estes dois aspectos poderão contribuir para colmatar as deficiências da filtragem.

Neste Instituto já houve uma tentativa preliminar de geração de curvas de nível por processos automáticos, que aplicou o raciocínio do parágrafo anterior ([22]), embora o resultado obtido não tenha sido satisfatório (quer em termos do DTM, quer no processo em si, visto ter havido bastante interacção do operador). No entanto, os princípios em que se baseou esta tentativa são importantes, na medida em que utilizou vários dos recursos ao dispor.

2.3.1 Ortofotos

Segundo [1], "a fotografia aérea apresenta distorções projectivas (devidas à inclinação do eixo óptico) e perspectivas (devidas ao relevo do terreno), que fazem com que a escala na área da fotografia varie com uma componente regular e outra irregular". A componente regular, como já referido, é originada por ser uma projecção central e não ortogonal: "a semelhança geométrica entre uma carta e uma

fotografia aérea da mesma escala e do mesmo terreno só existe, se a foto for rigorosamente vertical e o terreno for plano e horizontal". A geração de uma ortofoto tem em vista a eliminação das distorções originadas pela projecção central da fotografia aérea e pelo relevo, transformando-a numa projecção ortogonal, obtendo-se a informação visual densa do terreno com a geometria rigorosa de escala homogénea.

A sobreposição de uma nuvem de pontos sobre uma ortofoto da mesma região, ambas orientadas, permite obter, com precisão, a informação radiométrica do *pixel* sob cada um dos pontos, bem como o seu contexto.

A produção de uma ortofoto é feita, automaticamente, a partir das fotografias aéreas, das características técnicas das câmaras fotográficas aéreas, e do DTM ou DSM respectivo (DTM da edição anterior, ou DSM da edição de trabalho). Este pormenor é relevante, em virtude de permitir a utilização deste *output* como apoio à contextualização num algoritmo de filtragem (as ortofotos são produzidas durante a fase da aquisição em paralelo com as restantes actividades, não estando dependentes delas).

2.3.2 Informação das edições anteriores e de trabalho

Todo o território português está cartografado pelo IGeoE, pelo que é possível utilizar informação de edições anteriores, como apoio à actualização da carta da edição de trabalho. Esta é uma situação privilegiada e única e, a possibilidade de se poder recorrer a toda a informação vectorial da carta da edição anterior, permite condicionar e esclarecer positivamente um algoritmo de filtragem.

Os objectos que os operadores restituem estão perfeitamente discriminados protocolarmente, correspondendo a mais de 250 componentes diferentes, organizadas em cerca de 50 *layers*. Essas componentes estão divididas em vários temas, nomeadamente ([2]):

- Altimetria;
- Hidrografia;
- Rede viária principal;
- Toponímia;
- Rede ferroviária;
- Limites de vegetação;
- Caminhos;
- Construções.

Com esta informação da edição anterior conseguem-se fazer comparações com o que se encontra na edição de trabalho. Por hipótese, se um algoritmo de filtragem por declives classifica um dado ponto como não-terreno, poderá fazer-se uma análise subsequente, para verificar se esse ponto, na edição anterior, pertencia a algum objecto do tipo edifício e, desta forma, confirmá-lo como não-terreno (ou verificar que correspondia a uma ladeira e, embora o declive seja grande demais, o ponto deva ser classificado como terreno).

Um outro aspecto relevante é o facto de também ser possível utilizar-se toda a informação vectorial da edição de trabalho. Durante a restituição, não há qualquer precedência dos objectos a restituir, ou seja, e a título de exemplo, os operadores tanto podem restituir a vegetação, altimetria e rede viária, como qualquer outra combinação (passando-se o mesmo com os restantes temas). Assim, a altimetria pode ser restituída em último lugar, podendo fazer-se uso da restituição dos restantes objectos, para auxiliar esse processo. Eventualmente poderia haver prejuízo no paralelismo das restituições dos vários operadores, mas como se propõe que esta metodologia se aplique em último lugar, tal não acontece.

Em 2.2.2 enumerou-se que as maiores dificuldades da filtragem eram em objectos muito grandes, ambientes urbanos, edifícios e vegetação em elevações do terreno, pontes, vegetação baixa e ladeiras. Se se considerar as categorias supra enunciadas, percebe-se que estes objectos críticos são todos restituídos, individualmente, pelos operadores durante a fase de aquisição.

Como segundo [10, p. 139] os algoritmos de filtragem poderão melhorar os resultados obtidos, se houver uma identificação, *a priori*, de edifícios, estando de antemão na posse de um conjunto ainda maior de informação, mais refinado poderá ser o resultado da filtragem.

Há ainda uma particularidade da restituição, praticada no IGeoE, relacionada com o arvoredo: como a missão principal deste Instituto é a produção de cartografia militar, nem todas as manchas verdes são relevantes para este fim. Pelas normas internas de produção, apenas são restituídas zonas de arvoredo de área igual ou superior a 150 m x 150 m ([5]), fazendo com que pequenos grupos de árvores e árvores isoladas não sejam cadastradas.

A informação vectorial da edição anterior e da de trabalho, abre um leque de oportunidades para optimização de um algoritmo de filtragem: para além de garantir uma classificação inequívoca dos pontos localizados sobre qualquer objecto restituído, permite que se parametrize a função discriminante apenas para os objectos não restituídos, nomeadamente pequenos grupos de árvores ou árvores isoladas.

2.4 Detecção de árvores

Em 2.3.1 salientou-se a relevância das ortofotos, e em 2.3.2 também se constatou que os objectos de maior dificuldade, quanto à sua detecção, são pequenos grupos de árvores e árvores isoladas. Caso se consiga fazer esta detecção, então este problema é reduzido à identificação de quais pontos estão sobre estas novas áreas identificadas.

2.4.1 Por identificação de formas em imagens

Uma das abordagens ponderadas para se concretizar esta ideia foi por meio da identificação de formas nas ortofotos, através de algoritmos de segmentação. Esta metodologia tem algum grau de aprofundamento (tal como referido em [23, p. 2], que por sua vez cita quatro outros artigos), nomeadamente na extracção de objectos construídos pelo homem, como estradas, cruzamentos ou edifícios (formas bem definidas).

As copas das árvores nas ortofotos são tendencialmente circulares. Como é possível observar na Figura 2.10, que representa um pomar na região de Mafra (Carta nº 402), praticamente todas as árvores que se observam têm a mesma forma. É importante salientar que todas as árvores são da mesma espécie, embora com idades diferentes, e daí terem a mesma forma circular, mas tamanhos diferentes.

Figura 2.10 - Copas de árvores de forma circular (ortofoto)

No entanto, noutros extractos da mesma ortofoto (Figura 2.11), consegue-se observar outras copas de árvores de formas irregulares (na ortofoto superior esquerda é possível identificar que as ramagens das duas árvores maiores tornam a sua forma irregular; na ortofoto inferior direita consegue-se também identificar ciprestes ao longo do caminho de terra batida, cuja forma é ovalizada) e, inclusivamente, de outras cores.

Figura 2.11 - Diferentes copas de árvores extraídas da mesma ortofoto

O problema de uma identificação, por *matching* de formas, adensa-se pelo facto de existirem, nas ortofotos, outros objectos com as mesmas formas circulares (que à partida é o padrão de uma copa de árvore) e que não são árvores, conforme se pode verificar na Figura 2.12.

Figura 2.12 - Realce de outros objectos circulares que não copas de árvores (ortofoto)

Quer a piscina no canto superior direito (embora sendo de outra cor), quer os silos e a rotunda no canto inferior esquerdo, seriam com certeza erradamente classificadas como árvores, caso se pesquisasse exclusivamente por objectos de forma circular.

Por estes exemplos conclui-se que a identificação de formas, em imagens para detectar árvores, iria gerar muitos falsos positivos e falsos negativos. Embora esta abordagem não faça parte da metodologia final proposta, o seu estudo foi útil para solidificar a opção da pesquisa de solução, através de algoritmos de aprendizagem automática.

2.4.2 Por aprendizagem automática

A detecção de formas em imagem, descrita na subsecção anterior, revelou não ser o suficiente para identificar os objectos sobre o terreno, mas indiciou que, juntamente com outros aspectos, nomeadamente com a cor dos pixéis, permitiria obter conclusões mais sólidas. Este raciocínio canalizou a solução para um algoritmo que pondere diferentes factores, alguns com maior preponderância que outros. Um algoritmo de aprendizagem automática encaixa nestes requisitos.

Após pesquisa bibliográfica que versa esta temática (como [24], [25], [26] ou [27]), verificou-se que a maioria estava vocacionada para a detecção de copas de árvores em imagens de escalas pequenas e com objectivos diversos, mas principalmente para permitir distinção entre as espécies de árvores detectadas. No entanto, houve um documento ([23], datado de 2010) que se realçou, quer pela sua percentagem de precisão (superior a 90%), quer por se enquadrar exactamente no problema que se pretende resolver (identificação de árvores/conjuntos de árvores em fotografias aéreas), que será analisado de seguida. Além disso, conforme constatado em [23, p. 2], a metodologia proposta visa detecção de árvores em imagens de grande escala, ao contrário da maioria dos estudos precedentes.

Descrição geral

O algoritmo descrito em [23], opera em duas fases: uma primeira, de classificação de cada *pixel* como árvore ou não-árvore, e posterior segmentação de regiões árvore ou não-árvore. Na segunda fase, tenta-se "identificar" as copas das árvores nessas áreas, com base em modelos. Para a presente dissertação só a primeira fase é relevante, já que apenas se pretende associar pontos, da nuvem de pontos, a áreas de árvore ou não-árvore.

Selecção de *features*

Como este classificador funciona ao nível do *pixel*, as *features* a utilizar são também a este nível, já que conseguem capturar as propriedades mais distintas das árvores. Cada amostra é constituída por 27 *features*, que se descrevem de seguida, sendo ressalvado que o algoritmo pode ser aperfeiçoado, caso se acrescentem mais *features* discriminativas (em [23, p. 2] sugere-se dados LiDAR, ou seja, características de altimetria):

- **Cor**: constatou-se, conforme já enunciado no início deste sub-capítulo, que esta é a característica mais distinta na detecção de árvores. Converteu-se a fotografia aérea de RGB para o espaço de cores CIE L*a*b* ([28]) bem como para um outro espaço de cores (*illumination-invariant*, [29]), ambos caracterizando as cores de forma mais semelhante a como o olho humano as vê. Neste contexto, a particularidade mais relevante do CIE L*a*b* é fazer distar as cores, no seu referencial, da mesma forma que a percepção do olho humano. A Figura 2.13 mostra uma comparação de duas paletas de cores RGB (em cima) e CIE L*a*b* (em baixo), do vermelho para o amarelo, interpoladas à mesma distância entre elas em cada espaço de cores. Realça-se que, no RGB, as duas primeiras cores parecem muito semelhantes, enquanto, no CIE L*a*b*, a diferença parece equilibradamente gradual.

Figura 2.13 - Paleta de cores RGB (em cima) e CIE L*a*b* (em baixo) na interpolação do vermelho ao amarelo, com o mesmo espaçamento entre cores (fonte: [30]**)**

Já a particularidade do *illumination-invariant* é permitir que se façam operações, sobre o espaço de cores, independentemente da iluminação. Destes dois espaços de cores são obtidas 6 *features* (3 correspondendo ao CIE L*a*b* e 3 ao *illumination-invariant*);

- **Textura**: o padrão de textura formado pelas folhagens das árvores permite diferenciar se um *pixel* pertence a uma árvore ou a objectos de cores semelhantes (relvados, por exemplo). Utilizando o canal L* (brilho) da fotografia, aplicaram-se 18 filtros de Gabor, destinados a realçar arestas, com 6 orientações e 3 valores de sigma diferentes. O resultado da aplicação destes filtros é a geração de mais 18 *features* diferentes para cada *pixel;*
- **Entropia**: a entropia mede a desordem/imprevisibilidade de um sistema e, neste caso específico, calculou-se a entropia do brilho (canal L*) na vizinhança de cada *pixel*. Devido às sombras aleatórias existentes nas folhagens das árvores, é esperado que exista maior entropia nesses pixéis do que noutros correspondentes a objectos construídos pelo homem (como estradas ou edifícios), ou mesmo a outro tipo de vegetação (como relvados). Calculou-se a entropia de cada *pixel* em janelas de 5x5, 9x9 e 17x17, gerando mais 3 *features*.

Classificador

O algoritmo utilizado para classificar as amostras foi o *Adaboost* ([31]), um algoritmo de aprendizagem automática supervisionada, de classificação, do tipo *boost*, cuja ideia geral consiste na utilização de vários classificadores fracos, cada um com um peso relativo, para gerar um classificador consistente.

Classificador fraco

Um classificador fraco é uma função que, dada uma amostra, retorna uma classificação, e cuja probabilidade acerto é pouco mais de 50%, caso contrário daria resultados totalmente aleatórios. Na implementação da presente dissertação, cada classificador é constituído por uma *feature*, um valor de *threshold* para separar as duas classes, e o sinal que indica se, acima desse *threshold*, estão os casos positivos ou negativos. Durante o treino, deve ser encontrado o valor de *threshold* que melhor separa as duas classes.

Adaboost

O *adaboost* é um classificador "forte", que utiliza uma combinação linear de classificadores fracos, para produzir resultados. A modalidade de implementação foi por *stumping*, ou seja, pode ser representado por uma árvore de decisão de um nível com N nós (tantos quanto os classificadores fracos definidos).

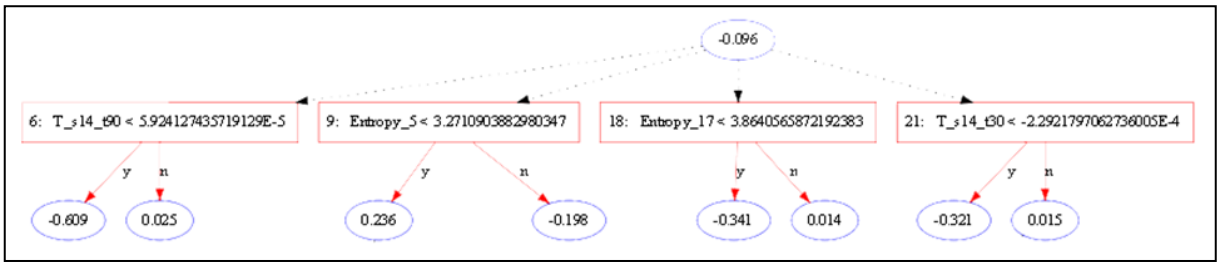


Figura 2.14 - Exemplo de árvore de decisão com *stumps*

Para além do conjunto de amostras, o único valor deste algoritmo, controlado pelo utilizador, é a definição do número de classificadores fracos pretendidos (utilizou-se sempre 200, conforme sugerido no *paper* [23]).

Execução da aprendizagem:

- Define pesos iguais para cada amostra do conjunto de treino
- Itera pelo número de classificadores fracos pretendidos (200). Por cada iteração:
    - Itera pelas 27 *features*. Por cada *feature*:
        - Calcula o *threshold* que melhor divide as amostras nas suas classes (árvore, ou não-árvore), e calcula o erro desse corte (o cálculo do erro tem em conta o peso de cada amostra).
    - Escolhe qual das 27 *features* (e respectivo *threshold*) gerou menor erro, definindo assim o classificador fraco.
    - Classifica cada uma das amostras, utilizando esse classificador fraco, ajustando o peso de cada uma (itera todas as amostras, e verifica quais foram mal classificadas; nestas, aumenta o seu peso, inversamente ao erro, para forçar o algoritmo, nas iterações seguintes, a classificá-las correctamente)

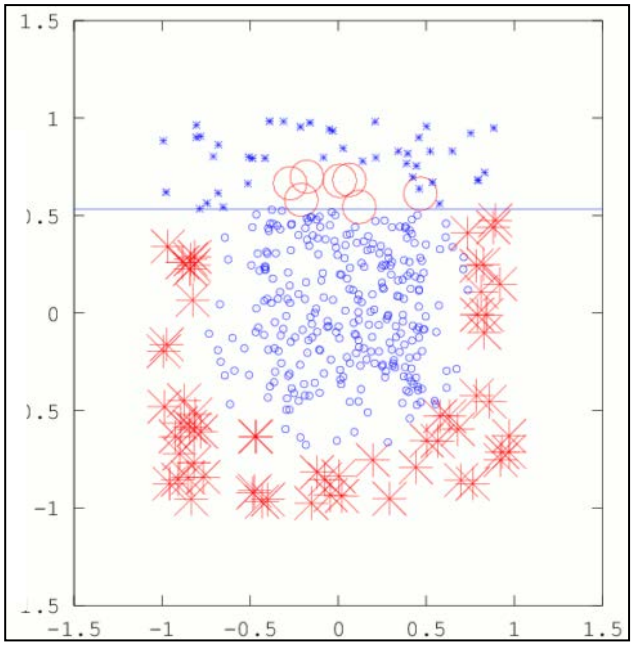


Figura 2.15 - Exemplo de classificação das amostras por um classificador fraco (círculos e cruzes vermelhas são amostras mal classificadas em cada uma das classes; fonte: [32]**)**

A linha que representa o *threshold*, num classificador, é a que melhor separa ambas as classes. A distância dos diferentes pontos a essa linha é designada por margem (as amostras bem classificadas contribuem positivamente, enquanto que as mal contribuem negativamente), e é o valor que define quão bom é esse classificador (quanto maior a margem, melhor é a separação das amostras pelas

classes, e menor a probabilidade de erro). A Figura 2.15 mostra um *threshold* representado pela linha azul horizontal, que separa as duas classes, onde as amostras a azul estão bem classificadas, e as vermelhas mal.

Após a aprendizagem feita, o modelo resultante é utilizado para avaliar outras amostras. Cada amostra é avaliada por todos os *stumps* da árvore de decisão resultante. Revendo a Figura 2.14, e utilizando-a como exemplo, os nós rectangulares possuem a descrição da *feature* e o respectivo valor the *threshold*, e os nós circulares contêm o valor da margem. Por cada *stump* é verificada a condição e acumulado o valor da margem parcial. Quanto maior for esse valor acumulado absoluto (soma de todas as margens parciais), mais certeza há que essa amostra pertence a uma classe ou outra (se o valor for positivo pertence a uma das classes, caso seja negativo corresponde à outra).

Refinamento do classificador

Uma das desvantagens da classificação pixel a pixel, é que a previsão de cada um é feita independentemente dos restantes, não sendo considerada, nessa altura, a consistência entre pixéis adjacentes. Isto origina uma imagem com bastante ruído, como se pode observar na Figura 2.16: as duas imagens da primeira linha correspondem às fotografias aéreas que se avaliaram, e as imagens da segunda linha correspondem ao resultado da avaliação pelo *adaboost* e respectivo refinamento. Nas imagens da segunda linha, os pixéis brancos correspondem às áreas identificadas como árvore, e os pixéis pretos às áreas identificadas como não-árvore; pode-se observar a presença de grande ruído pela existência de vários pixéis brancos isolados, por exemplo, sobre o campo de futebol, ou sobre a zona de água.

O refinamento da imagem é formalizado como um problema de minimização de energia, tendo como base a metodologia proposta em [33].

Embora este algoritmo de minimização de energia possa ser utilizado para segmentar várias classes, a sua definição e aplicação é apenas para duas: "*foreground*" e "*background*", associadas a "*source*" e "*sink*".

Cada pixel é associado a uma das classes, e o problema é ajustar essas ligações aos nós terminais, de tal forma que a soma das suas valorizações, menos as penalizações, seja máxima.

A função de energia utilizada é a definida em [23]:

$$E = \sum_{p \in X} D\ (p, L(p)) + \sum_{(p,q) \in N} V(L(p), L(q))$$

O primeiro termo (*data term*), D, mede a correspondência entre a classificação do pixel e a probabilidade, definidos pelo *AdaBoost* (função de penalização, que relaciona a classificação do pixel com a sua probabilidade). O segundo termo (*smoothness term*), V, mede a suavização da classificação (função que encoraja coerência espacial ao penalizar descontinuidades entre pixéis vizinhos). Estes termos são definidos da seguinte forma:

$$D\ \big(p, L(p)\big) = \begin{cases} \log\big(1 - P_{árvore}(p)\big) & se\ L(p) = árvore \\ \log\big(P_{árvore}(p)\big) & caso\ contrário \end{cases}$$

$$V\big(L(p), L(q)\big) = \begin{cases} 0 & se\ L(p) = L(q) \\ \beta & caso\ contrário \end{cases}$$

X é o conjunto de todos os píxeis, L(p) ∈ {árvore, não-árvore} (classificação de cada pixel *p* dada pelo sinal, durante *AdaBoost*), N o conjunto de pixéis vizinhos de *p* (8 ligações aos vizinhos), $P_{árvore}$ a probabilidade normalizada de ser árvore ou não-árvore, e β definido empiricamente como 1.

A sua implementação é num grafo pesado, onde cada pixel corresponde a um nó. Além disso, neste grafo, há dois nós "especiais", terminais, aos quais correspondem as duas classes a segmentar: o "*foreground*" e "*background*", ou "*source*" e "*sink*".

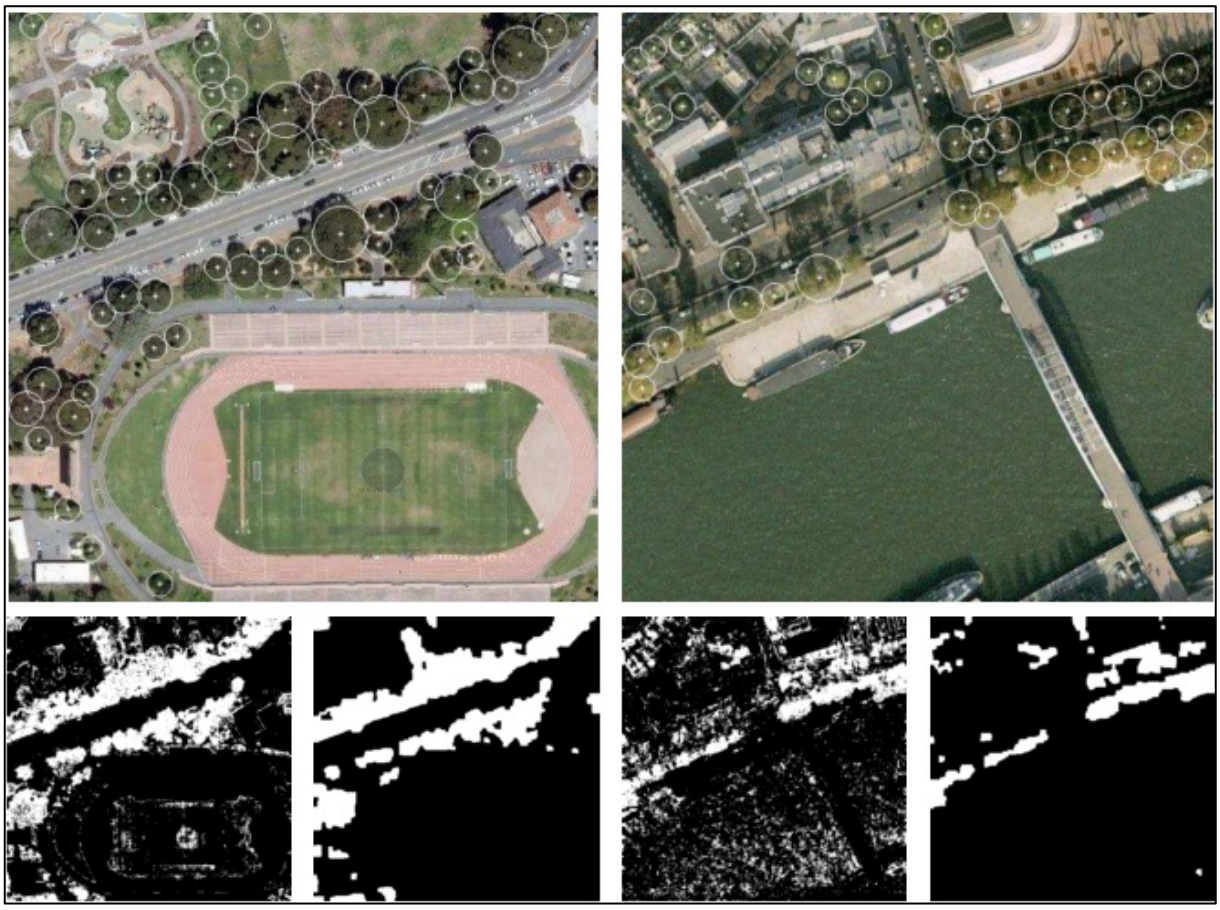


Figura 2.16 - Refinamento do classificador (fonte: [23, p. 4]**)**

A preparação do grafo passa por associar cada nó/pixel ao nó terminal correspondente, tendo em conta a classificação dos pixéis obtida pelo *AdaBoost*. O custo dessa associação corresponde ao "data term" da função de energia, ou seja, o custo da associação do nó *p* ao seu terminal é $\log(1 - P_{\text{árvore}}(p))$ se L(p) for árvore, ou $\log(P_{\text{árvore}}(p))$ caso contrário.

Depois de conectados aos terminais, é necessário criar as ligações entre nós vizinhos, representando o segundo termo da função de energia (V, *smoothness term*). Assim, cada nó é conectado aos seus 8 vizinhos com um peso de 0 se L(p)=L(q), ou 1 caso contrário.

Descrição do algoritmo de refinamento

Durante o algoritmo são mantidas duas "árvores" não sobrepostas, S e T, com as raízes na "*source*" e "*sink*". Os nós pertencentes a S ou T podem ser "activos" ou "passivos", consoante estão na fronteira ou no interior.

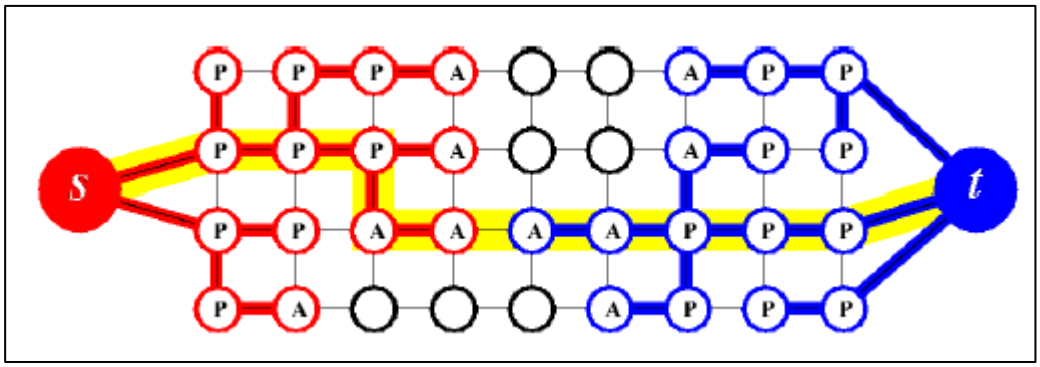


Figura 2.17 - Esquema de grafo pesado representando o algoritmo *min-cut/max-flow* (fonte: [33]**)**

Os nós activos permitem às árvores crescerem, ao adquirir novos filhos ao conjunto de nós livres. Quando um nó activo encontra um nó da outra árvore, então foi encontrado um "caminho de aumento".

Iterativamente, o algoritmo executa as seguintes fases:

- Fase de crescimento

  Durante esta fase dá-se a expansão das árvores. Os nós adquiridos tornam-se activos e, quando todos os vizinhos de dado nó forem explorados, esse nó passa a passivo. Esta fase termina quando um nó activo encontra um nó vizinho que pertence a outra árvore.

- Fase de aumento

  Alguns dos nós das árvores S e T podem-se tornar órfãos, ou seja, as arestas que os ligavam aos seus pais já não são válidas (tornaram-se saturadas). Na verdade, durante esta fase pode-se dar a quebra das árvores em várias florestas. O "*source*" o "*sink*" continuam a ser as raízes de 2 das árvores, enquanto que os órfãos formam raízes das restantes árvores.

- Fase de adopção

  A finalidade desta fase é restaurar as árvores isoladas, pertencentes/associadas a S e T, com as raízes "*source*" e "*sink*". Nesta fase tenta-se encontrar um nó pai para cada órfão. O novo pai deve pertencer ao mesmo conjunto do órfão (S ou T). Além disso, deverá ser ligado por uma aresta não saturada. Se não se conseguir encontrar um pai, remove-se o órfão do conjunto S ou T, tornando-se um nó livre (não pertencendo a nenhuma das árvores), e todos os seus filhos também são definidos como órfãos. Esta fase termina quando deixa de haver órfãos, e as árvores S e T são restauradas.

O algoritmo termina, quando as árvores S e T deixam de poder crescer (sem nós activos), e as árvores são separadas por arestas saturadas. Isto implica que o *maximum flow* foi atingido, e o *minimum cut* corresponde ao Si=S e Ti=T (Si e Ti são os conjuntos S e T no início da iteração).

3 Implementação

3.1 Plano de trabalho

Após a descrição das diferentes abordagens possíveis no capítulo anterior, e seguindo o raciocínio de alguns dos autores ([9, p. 89], [10, p. 139], [11, p. 60], [12, p. 1] ou [8, p. 7]), a solução proposta para o problema passa pela criação de uma aplicação que execute e faça uso de diferentes algoritmos e recursos, para eliminação do "ruído" existente num DSM, por forma a obter o DTM desejado. Este "ruído" é gerado por três tipos de objectos sobre o solo: edifícios, vegetação (áreas superiores a 150 m x 150 m), e pequenos grupos de árvores ou árvores isoladas.

Assim, pretende-se obter uma nuvem de pontos filtrada, da qual se pode gerar o DTM de uma região, utilizando como dados de entrada:

- **Amostra primária de nuvem de pontos (correspondente ao DSM)**
  Irá sofrer sucessivas filtragens, para se obter a amostra final correspondente ao DTM.
- **Informação vectorial da edição de trabalho (excepto altimetria)**
  Permite identificar todos os pontos da amostra primária sob os objectos restituídos, nomeadamente edifícios e vegetação. Os pontos sobre edifícios serão interpolados, para os nivelar pela cota do solo.
- **Informação vectorial da edição anterior (altimetria)**
  Serve para estimar a cota dos pontos sob vegetação, nas áreas de vegetação obtidas pelo método descrito no ponto anterior. Nas fotografias aéreas, este tipo de vegetação esconde as características do solo (existência de linhas de água, por exemplo). Na impossibilidade de captura dessa informação por fotografia, optou-se por manter o terreno conforme a edição anterior (haverá hipótese de acumulação de erros, mas aceitáveis, por não existir alternativa nesta fase da aquisição).
- **Ortofotos (obtidas pelas fotografias aéreas)**
  Possibilitam a aplicação de algoritmo de aprendizagem automática, para detecção de árvores isoladas e pequenos grupos de árvores. Pontos da amostra primária sob essas áreas serão substituídos por outros interpolados.

3.1.1 Metodologia

Nos organigramas das figuras 3.1 e 3.2 estão representados os fluxos de informação e correspondente processamento em diferentes fases, desde quando se recebe a "matéria-prima" até à obtenção da amostra secundária de nuvem de pontos, pronta a ser utilizada para geração de DTM.

O fluxo geral (Figura 3.1), devido a transcender o âmbito do presente estudo, está sucintamente descrito na subsecção 3.1.2. O fluxo do processamento (Figura 3.2), a essência de todo o estudo, está dissecado ao longo da secção 3.2, e poderá ser ainda completado com o código fonte, e/ou Javadoc, da aplicação final. Para salientar alguns pormenores com relevância, bem como facilitar a consulta futura, nesses pontos serão usadas as seguintes notações:

Caixas de texto, com um símbolo de cronómetro, conterão aspectos identificados como permitindo optimização futura, quer estejam relacionados com a metodologia, quer com a aplicação propriamente dita.
Caixas de texto, com um símbolo de bifurcação, conterão decisões que tiveram de ser tomadas com base no conhecimento e contexto actual, e que poderão não ser as mais adequadas, considerando evoluções futuras de *software* e da cadeia de produção do IGeoE.

Caixas de texto, com um símbolo de exclamação, conterão avisos, principalmente devidos e relativos a limitações do *software* ou *hardware* utilizado, ou a imposições resultantes da cadeia de produção do IGeoE.

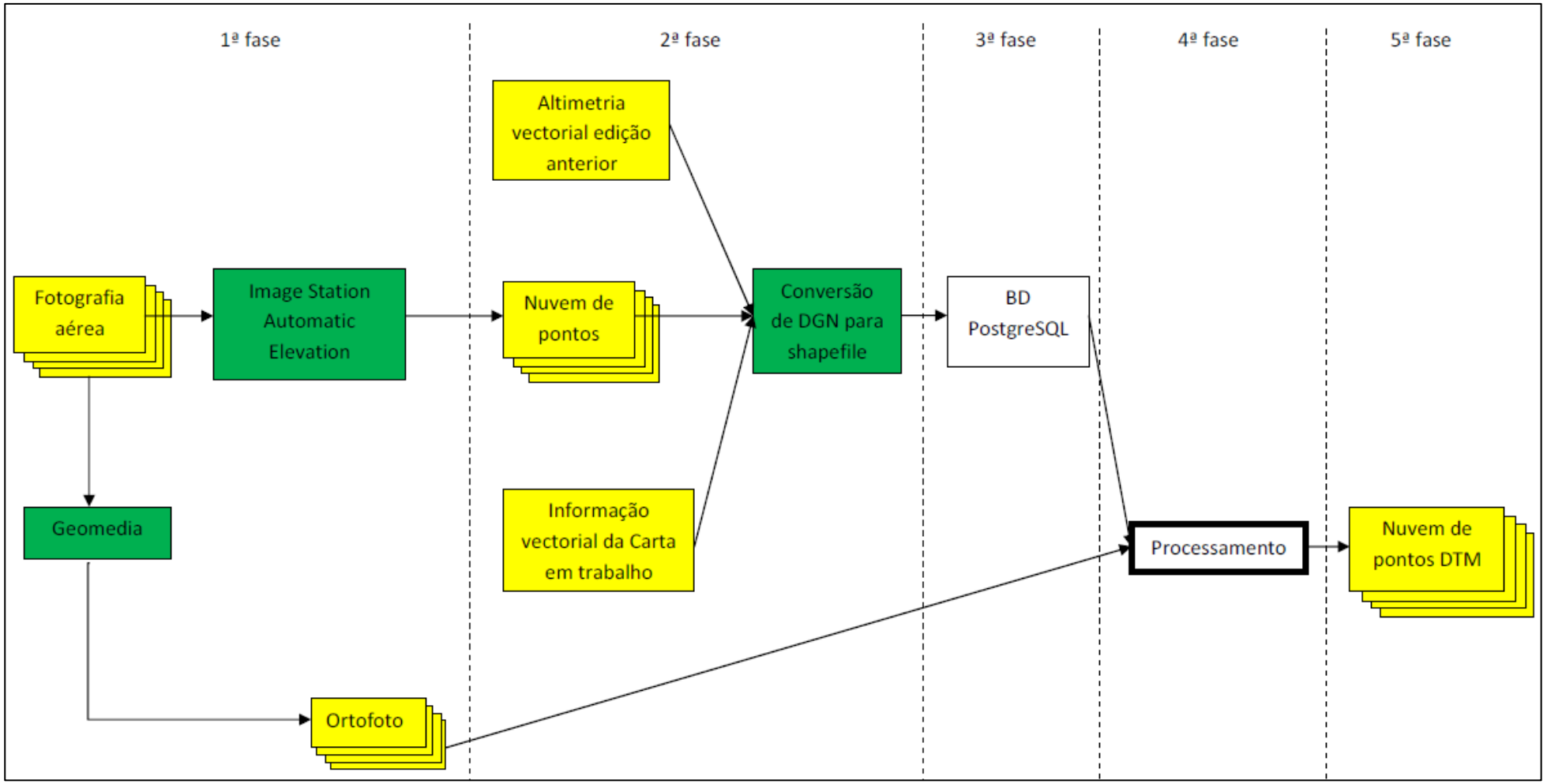


Figura 3.1 - Fluxo geral

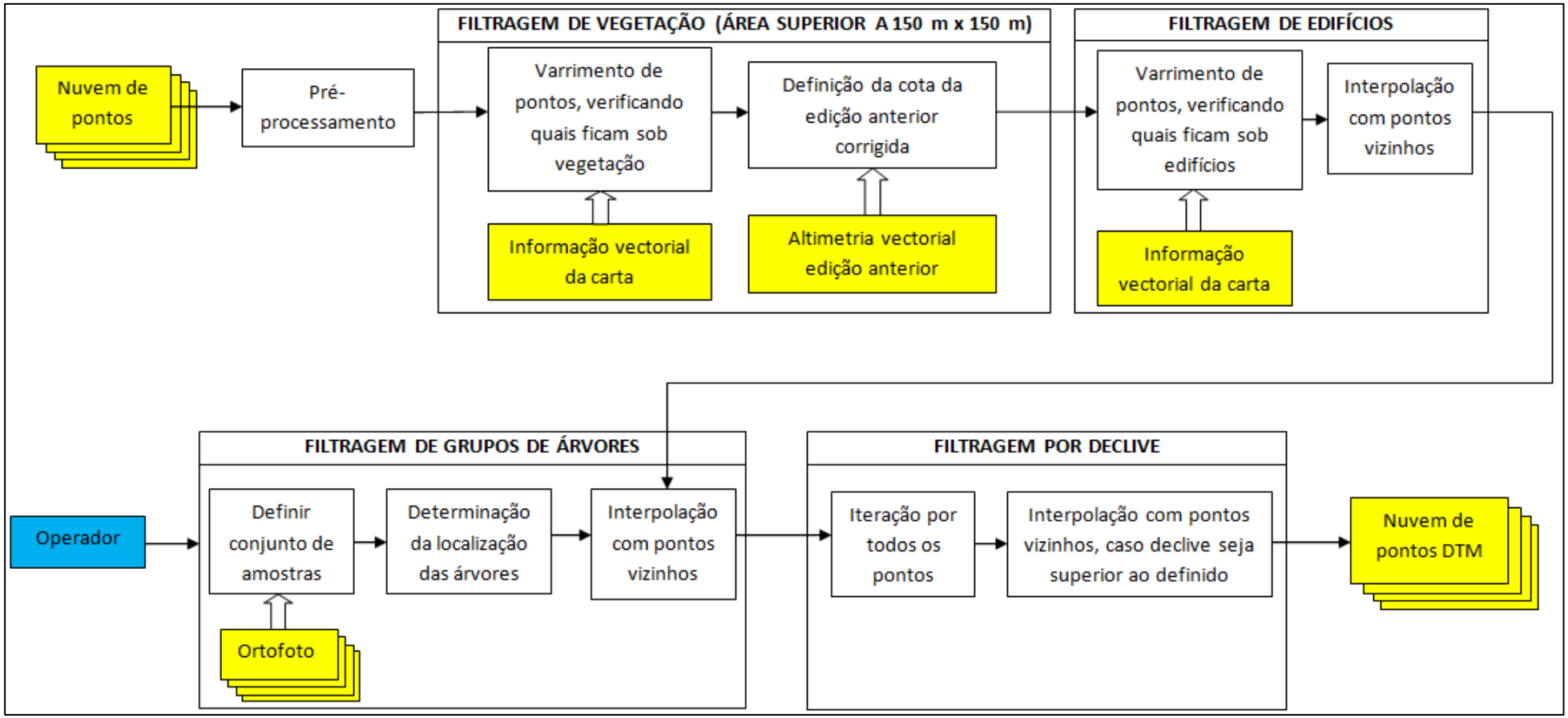


Figura 3.2 - Fluxo específico do processamento

3.1.2 Fluxo geral

Primeira fase – preparação dos dados de entrada

A Figura 3.1 tem realçado, a cor amarela, os diferentes "produtos" do fluxo. Conforme já referido, o elemento base de todo este processo são as fotografias aéreas. Estas sofrem, logo à partida, dois refinamentos: o primeiro, já descrito, é a sua transformação em nuvem de pontos por correlação automática de imagens, através do Image Station Automatic Elevation (ISAE, [13]). Estas nuvens de pontos constituem-se como a amostra primária.

De salientar que a execução do ISAE deverá ser feita com o menor adoçamento possível, caso contrário ele irá compensar eventuais grandes diferenças de cota, adicionando mais erros sobre a cota dos pontos.

A segunda transformação operada é na geração de ortofotos: estas são criadas automaticamente, utilizando o *plugin* OrtoPro da aplicação Geomedia ([34]). Tendo como dados de entrada as fotografias aéreas correspondentes à área de uma carta topográfica, os parâmetros da aerotriangulação e o DTM da edição anterior da carta[2], a aplicação transforma a projecção central das fotografias aéreas numa projecção ortogonal. As ortofotos resultantes são uma representação de escala rigorosa, podendo ser utilizadas para calcular distâncias com precisão.

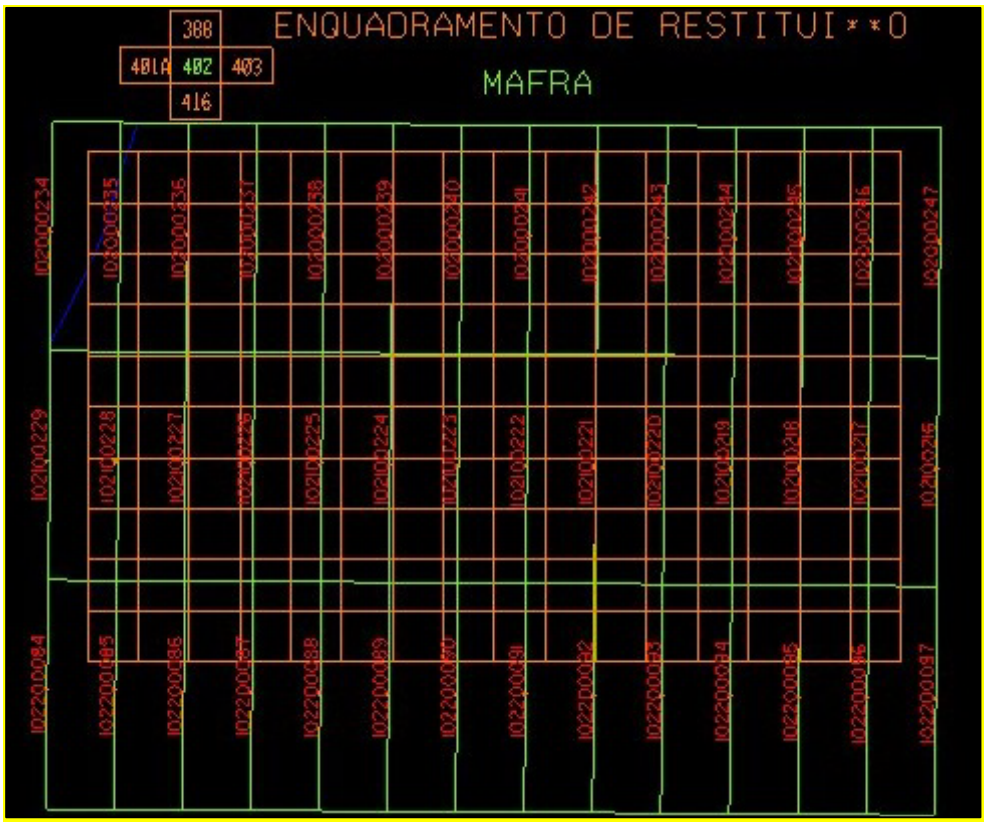


Figura 3.3 - Enquadramento das fotografias aéreas (quadriculado verde com numeração por cada fotografia) sobre a área da carta topográfica nº 402 (quadriculado a laranja; cada quadrado destes representa 1 km de lado)

A carta que serviu de base para a maioria dos testes foi a 402, da zona de Mafra, e a Figura 3.3 mostra o enquadramento das respectivas fotografias aéreas da edição mais recente. Para cobrir a área

[2] Dever-se-ia utilizar o DTM gerado com a informação das fotografias a ortorrectificar, mas face à necessidade das ortofotos na Cadeia de Produção em fases anteriores à produção do novo DTM, e devido à exactidão do DTM da edição anterior, a utilização deste último permite a criação das ortofotos dentro dos padrões de qualidade exigidos.

da carta (16x10km), foram necessárias 39 fotografias, e para gerar a altimetria pela metodologia proposta, seria necessário a geração de 36 nuvens de pontos.

Segunda fase – uniformização dos dados de entrada em ficheiros DGN, e conversão para *shapefile*

A segunda fase do fluxo refere ainda mais dois produtos: a informação vectorial da edição de trabalho, bem como a altimetria da edição anterior. Como o *software* utilizado na produção no IGeoE pertence à *Integraph*™ e à *Bentley Systems*™ (nomeadamente o ISAE [13] e o Microstation [14]), e fruto de como essa cadeia está implementada, as informações vectoriais são guardadas em ficheiros de extensão DGN (formato de ficheiros CAD suportados pelos programas da *Bentley Systems*™ e *Intergraph*™ [35]). Portanto, estes dois produtos constituem-se, na prática, como 2 ficheiros (um por produto).

Durante o desenvolvimento da aplicação final, os testes foram executados sobre a nuvem de pontos gerada por correlação das fotografias 102000240 e 102000241, por conterem grande diversidade de características do terreno (tais como zona de edifícios "densa", arvoredo denso, arvoredo esparso e árvores isoladas, campos de futebol, edifícios de grandes áreas – convento de Mafra –, entre outras).

Considerando que a criação das nuvens de pontos também gerou um ficheiro DGN (o ISAE [13] gera um ficheiro DGN por cada par de fotografias aéreas correlacionadas), no final desta fase existirão 3 ficheiros DGN com a nuvem de pontos, informação vectorial da edição anterior e da edição de trabalho, e ficheiros TIF, juntamente com os seus ficheiros TFW (ficheiros *world*, [36]), com as ortofotos.

Os ficheiros DGN contêm informação geoespacial, nomeadamente pontos e polígonos. No entanto, como é um formato proprietário, fazer a sua leitura e manipulação não é trivial (embora a sua especificação já seja pública, para permitir, precisamente, interoperabilidade entre aplicações). Há duas possibilidades para solucionar este problema:

- utilizar ferramentas de *software* existente para fazer a conversão do DGN para *shapefile* (formato aberto, de mais fácil manipulação), nomeadamente do ArcGIS ([37]);
- com base na especificação dos ficheiros DGN ([38]), fazer um conversor que os permita abrir e guardar noutro formato mais conveniente.

Embora não esteja propriamente no âmbito desta tese, uma das formas de isolar melhor todo o processo e metodologia proposta, e acelerar o desempenho, é a criação de um conversor de ficheiro DGN para *shapefile*. A especificação detalhada em [38] permite criar o conversor em qualquer linguagem e integrá-lo na aplicação final.

Uma outra hipótese considerada foi a conversão dos ficheiros DGN para TXT, utilizando o Microstation ([14]). No entanto, como esta ferramenta apenas converte pontos (o TXT resultante é constituído por tantas linhas quantos os pontos, cada linha com coordenada X, Y e Z), esta solução só era viável para o DGN que contém a nuvem de pontos, persistindo o problema da conversão dos outros dois DGN com a informação vectorial.

Por restrições de tempo, e por esta conversão sair do âmbito do trabalho, a opção recaiu sobre a conversão de DGN para *shapefile* utilizando o ArcGIS.

Terceira fase – importação dos ficheiros DGN para base de dados

Uma vez na posse dos *shapefiles*, é necessário considerar a forma de os manipular. Há bibliotecas que o fazem, nomeadamente o GeoTools™ ([39]), embora a solução mais apropriada passe por importar a informação destes ficheiros para uma base de dados geográfica. Este raciocínio é suportado por dois motivos:

- é intenção do comando do IGeoE fazer a migração para bases de dados geográficas ([40]), bem como utilização gradual de ferramentas *open-source*/livres;

- a existência de uma base de dados o*pen-source*, PostgreSQL ([41]), com um *plugin* dedicado e optimizado para tratamento de informação geográfica (PostGIS, [42]).

Tendo em conta o primeiro ponto, realça-se que as conversões DGN-*shapefile* e *shapefile*-base de dados são acções que poderão ser optimizadas, como referido, por um conversor próprio para o efeito, mas também, no futuro, poderão nem ser necessárias, visto prever-se que os dados geográficos já façam parte de uma base de dados geográfica.

Esta fase consiste, portanto, na importação dos *shapefile* para base de dados PostgreSQL/PostGIS (a instalação do PostGIS inclui um *shapefile importer*) para a sua posterior manipulação.

Quarta fase – processamento

Esta fase está esquematizada no diagrama de fluxos específico (Figura 3.2), e é pormenorizada na secção 3.2.

Quinta fase – obtenção da amostra secundária correspondente ao DTM e validação

Após o processamento, todos os pontos da nuvem de pontos resultante deverão ter a cota correspondente à do nível do solo, e o resultado deverá ser avaliado. Segundo [8, p. 4] e [43, p. 36], baseado em [15, p. 839], há três tipos de métodos para verificar a qualidade do DTM gerado: a inspecção visual em estereoscopia, o levantamento de amostras aleatórias de pontos de controlo na área de estudo, e a comparação com áreas de testes classificadas manualmente. Tendo em conta que, no IGeoE, os operadores acumulam a experiência de anos de restituição fotogramétrica, e que serão eles próprios a eventualmente utilizar o produto resultante do presente estudo, a forma de avaliação mais indicada é a inspecção visual em estereoscopia.

Assim, para validar a amostra secundária, a nuvem de pontos resultante foi exportada para um ficheiro *shapefile* (utilizando o *shapefile importer* do PostGIS, que também permite fazer a exportação), sendo novamente convertida, através do ArcGIS ([37]), para ficheiro DGN, para poder ser aberta como nova *layer*, no Microstation ([14]), para validação em estereoscopia por um operador. Salienta-se que este procedimento final já não faz parte do processamento proposto na presente dissertação.

Durante o desenvolvimento da aplicação foi utilizada uma outra forma de avaliação de resultados (confirmada, no final, pelo método acima descrito). Visto já existir informação vectorial da altimetria da carta 402, restituída pelo método tradicional (operador em estereoscopia), foi feita conversão das curvas de nível e pontos de cota em nuvem de pontos, com as mesmas características da nuvem de pontos secundária produzida pela aplicação final (grelha regular, com o mesmo espaçamento entre pontos), e compararam-se ambas, ponto a ponto. Desta forma, foi possível calcular, por exemplo, o erro quadrático médio, que serviu como indicador quantitativo do desempenho da aplicação.

Esta forma de validação adiciona erros, nomeadamente porque as curvas de nível, num vector finalizado, já foram adoçadas. Ainda assim, assumindo como aceitável, empiricamente, um erro de até 3 m, constituiu-se como indicador valioso.

3.1.3 Condições

Embora a cadeia de produção do IGeoE se apoie, quase exclusivamente, em ficheiros DGN, é intenção do comando daquele Instituto fazer a migração para bases de dados geográficas ([40]), bem como utilização gradual de ferramentas *open-source/*livres. Assim, quer a aplicação final, quer as bibliotecas por esta utilizada, quer as restantes aplicações utilizadas no fluxo de produção, deverão ser *open-source/*livres.

3.1.4 Pressupostos

Para a elaboração da aplicação foi necessário assumir determinados pressupostos, caso contrário não se poderia chegar a conclusões elucidativas. São os seguintes:

- existência de informação vectorial, ou de DTM, de uma edição anterior;
- nuvem de pontos de amostra primária fiel e precisa;

- informação vectorial da edição de trabalho (todos os objectos com excepção da altimetria) correcta e exacta;
- existência de ortofotos orientadas e referenciadas, através de um ficheiro "*world*", fiéis e precisas;
- todos os *inputs* (informação vectorial de edição anterior e corrente, DTM de edição anterior, ortofoto) estão georreferenciados e no mesmo sistema de coordenadas.

3.1.5 Nível pretendido

A avaliação do *AutoDTM* foi feita através das normas definidas em [44] e [45], com base no cálculo da Raiz do Erro Médio Quadrático (*Root Mean Square Error*, RMSE) e do Linear Map Accuracy Standard (LMAS).

Para as primeiras normas, são critérios de validação, em altimetria e em cartas à escala 1/10 000:

- 90% dos pontos não pode ter desvios superiores a 3 metros
- Valor de RMSE inferior ou igual a 1.8 metros.

As segundas normas exigem que se faça a avaliação com, pelo menos 167 pontos, e, em altimetria, definem uma classificação crescente, de 0 a 4. Para além do número mínimo de pontos, os critérios de avaliação , em altimetria e em cartas à escala 1/25 000, são os seguintes:

Tabela 3.1 - Avaliação segundo as normas descritas em [45] (tabela adaptada de [45])

Classificação	**Valor de LMAS**
	Escala: 1/25 000
0	2,5 m
1	5 m
2	10 m
3	Piores que a classificação 2
4	Não determinado

Assim, pretendeu-se que o *AutoDTM* produzisse resultados que obedecessem aos critérios de validação de [44], e que, no máximo, obtivesse uma classificação de 1 na Tabela 3.1.

3.1.6 Ferramentas utilizadas

Como referido em 3.1.3, quer a aplicação, quer as bibliotecas, quer as restantes aplicações que contribuem para a execução da metodologia proposta, deverão ser *open-source/*livres. O quadro seguinte resume as alternativas ponderadas e as opções tidas nesse âmbito:

Tabela 3.2 - Ferramentas utilizadas

Fase/descrição	**Ferramenta/biblioteca**	**Observações**
1ª fase		
Criação de ortofotos	Geomedia	Transcende o presente trabalho; executado em *software* proprietário, há alternativas *open-source/*livres.
Geração de nuvens de pontos	Image Station Automatic Elevation	Transcende o presente trabalho; embora seja executado em *software* proprietário, há alternativas *open-source/*livres.

2ª fase		
Criação de vector, edição anterior e corrente	Microstation v8 (formato DGN)	Transcende o presente trabalho; todas as alternativas em estereoscopia são em *software* proprietário; necessita *plugin* da Intergraph para se utilizar em estereoscopia.
Criação de DTM da edição anterior	ArcGIS v10.1	Transcende o presente trabalho; embora seja executado em *software* proprietário, há alternativas *open-source/*livres.
Conversão de DGN para *shapefile* (formato aberto)	ArcGIS v10.1 QuantumGIS v2.0.1 ([46])	Transcende o presente trabalho; há alternativas em *software* proprietário, como o ArcGIS, ou *open-source*, como o Quantum GIS.
3ª fase		
Importação para base de dados geoespacial	PostgreSQL v9.3 PostGIS v2.1.0 PostGIS *shapefile loader exporter*	PostgreSQL e o plugin PostGIS são *open-source*.
4ª fase		
Aplicação final, "*AutoDTM*"	Java (v6u25)	Linguagem de programação Java é *open-source*. Opção de desenvolvimento recaiu em Java por rapidez e facilidade de implementação, por ter grande panóplia de bibliotecas à disposição, e por haver indícios de ser linguagem que terá grande durabilidade temporal ([47])
Ligação Java - PostgreSQL	JDBC v4 ([48])	Biblioteca *open-source/*livre disponibilizada pelo próprio PostgreSQL.
Leitura e manipulação de imagens	OpenCV v2.4.8 ([49])	*Open-source/*livre. Há outras alternativas *open-source*/livres, mas o OpenCV dá garantia de segurança e estabilidade por ser amplamente utilizada em produção; disponibiliza bastantes ferramentas de manipulação de imagens, nomeadamente na conversão entre espaços de cores; tratamento de imagens por matrizes, sendo trivial manipular cada canal individualmente. Em termos de optimização, pelo menos na fase da filtragem de árvores, por ser biblioteca desenvolvida em C/C++, haveria vantagem em que a aplicação final também fosse em C/C++.
Adaboost	Weka v3.6.10 ([50]) JBoost v2.4 ([51]) OpenCV v2.4.8	Todas as ferramentas listadas disponibilizam alguma implementação de Adaboost, mas, como será detalhado na subsecção 3.2.3, aquela que apresentou melhores resultados foi o Weka.

4ª fase (continuação)		
Carregamento de *rasters* em base de dados	raster2pgsql.exe ([52])	Executável que converte imagens *raster* em comandos SQL, que poderão ser utilizados para carregar uma base de dados.

Das diferentes ferramentas apresentadas, o PostGIS merece algum destaque nesta altura, em virtude de disponibilizar uma base sólida e standardizada para todas as manipulações sobre objectos geoespaciais, essencial desde a primeira fase do fluxo do processamento (pré-processamento).

PostGIS ([42]**)**

O PostGIS é um *plugin* para o PostgreSQL, que dá propriedades geoespaciais a bases de dados, permitindo que todas as manipulações sobre elas sejam feitas por intermédio de *queries* SQL.

Para tornar uma base de dados do PostgreSQL geoespacial, com o *plugin* PostGIS, é necessário:

1. Criar a base de dados que se pretende tornar geoespacial
   Exemplo de comando SQL: “CREATE DATABASE gis”;
2. Criar a extensão “postgis”
   Exemplo de comando SQL: “CREATE EXTENSION postgis”;
   Após a criação desta extensão, a base de dados passa a ter uma tabela que contém uma vasta listagem de sistemas de referência de coordenadas, e mais de 1000 funções disponíveis (a listagem da maioria delas, que se constitui como bom auxílio para o desenvolvimento, pode ser encontrada em [53]) para manipulação de dados geoespaciais (ver Figura 3.4). O cerne do PostGIS está nestas funções.

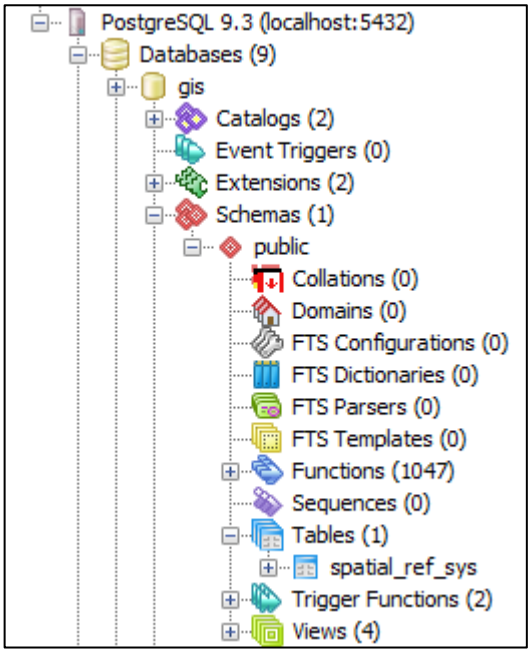


Figura 3.4 - Aspecto de base de dados geoespacial

Após esta preparação da base de dados, pode-se criar qualquer tabela geoespacial. Estas tabelas têm de conter uma coluna do tipo *geometry*, que define o objecto geoespacial, podendo conter também quaisquer outras colunas (nomeadamente, e utilizando o *standard* do PostGIS, uma coluna *gid*, que corresponde ao *id* da geometria; exemplo de query SQL para criação de tabela: “CREATE TABLE data (gid SERIAL, geom GEOMETRY)”).

A coluna *geometry* pode guardar objectos geoespaciais de diferentes tipos (*point*, *polygon*, *linestring*, entre outros), com a sua definição a obedecer ao *standard well-known text* (WKT, [54]). A

inserção de objectos poderá ser feita por ***query* SQL** (utilizando, por exemplo, a função "ST_geomfromtext", na qual um dos argumentos é a descrição da geometria através do já referido WKT), através de um **cliente gráfico** (por exemplo, o QuantumGIS permite abrir *layers* PostGIS – tabelas geoespaciais –, onde podem ser criados, graficamente, quaisquer objectos; assim que se guarde a *layer*, essas alterações são introduzidas na base de dados, e cada objecto terá a sua representação num tuplo da base de dados), ou através do **importador/exportador de ficheiros *shapefile*** (aplicação que faz parte da instalação do PostGIS).

Parte da eficiência da manipulação de dados geoespaciais, em bases de dados, prende-se com a indexação da coluna *geometry*: na realidade, o que é armazenado nessa coluna não é a *string* em formato WKT, mas o seu equivalente em *Well-Known Binary* (WKB).

3.2 Execução

Nesta secção é desenvolvido o fluxo do processamento, que descreve, com pormenor, a implementação da aplicação final *AutoDTM*

A execução do *AutoDTM* é controlada através de um ficheiro de configurações, com diversas variáveis disponíveis para gestão dos mais variados aspectos da aplicação. O anexo 7.1 é um exemplo, completo, do conteúdo desse ficheiro. Outra alternativa de implementação seria iniciar o *AutoDTM* com tantos argumentos quantas as variáveis, mas, dada a grande quantidade destas variáveis, essa opção tornar-se-ia impraticável e muito susceptível a erros.

Executando o *AutoDTM* sem qualquer argumento, este procurará as definições no ficheiro "configuration.autoDTM". Para permitir a execução em *batch*, com várias configurações diferentes, a aplicação também está preparada para receber como argumento o caminho para o ficheiro de configurações. Assim, é possível criar diferentes ficheiros de configuração, para controlo do *AutoDTM*, e a sua chamada para execução diferirá apenas no argumento que contém o *path* para os diferentes ficheiros de configuração.

Desta forma, antes de se dar início à execução, é necessário ajustar adequadamente as configurações, preparando a base de dados, directorias e ficheiros de *input*, de acordo com essas configurações. A Figura 3.5 e a Figura 3.6 são exemplos do conteúdo de uma base de dados e da directoria de ficheiros de entrada.

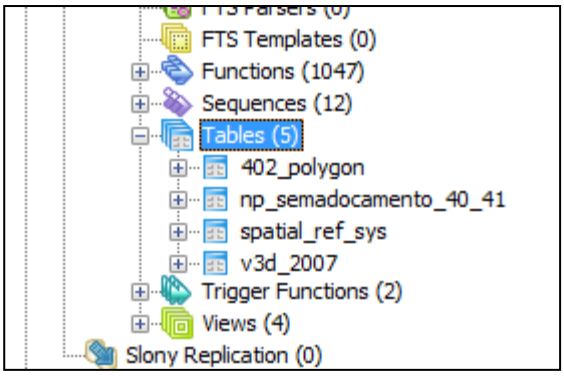


Figura 3.5 - Listagem das tabelas necessárias à execução do *AutoDTM* (exemplo; extraído do PHPAdmin)

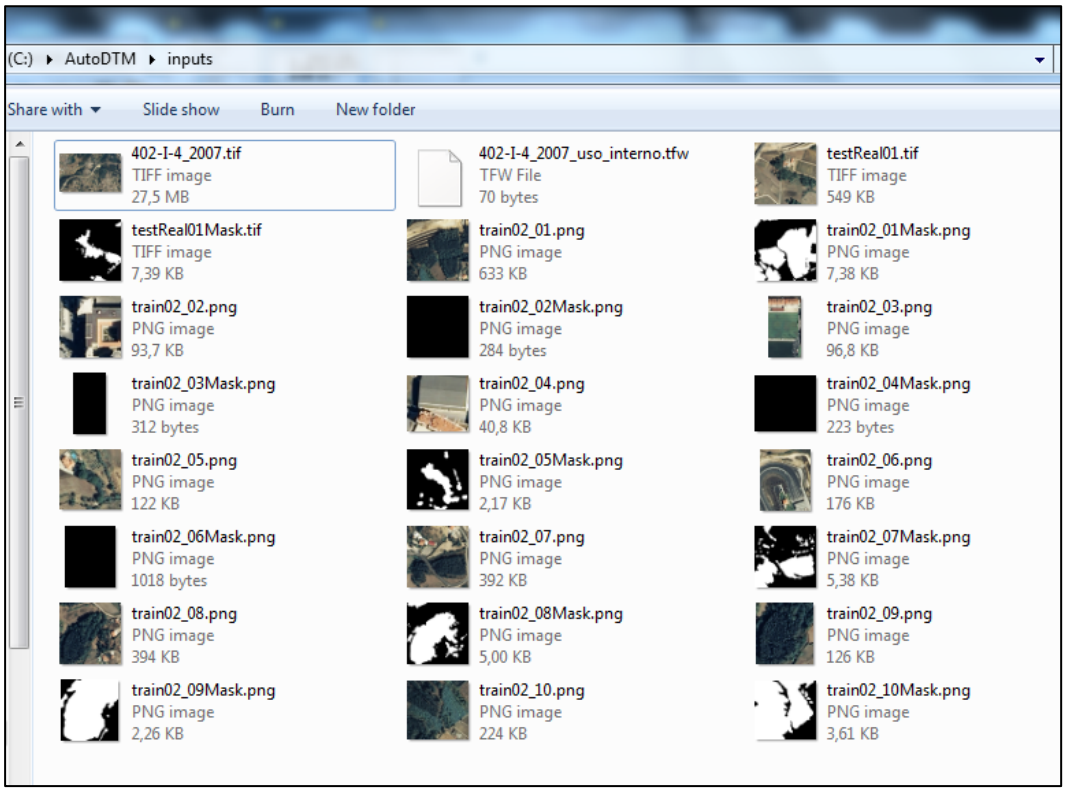


Figura 3.6 - Listagem dos ficheiros de *input* necessários à execução do *AutoDTM* (exemplo)

3.2.1 Pré-processamento

Na fase do pré-processamento, o *AutoDTM* vai executar uma série de operações sobre estes dados de entrada, nomeadamente:

1. Clonagem da tabela do vector da edição de trabalho
   Na Figura 3.5, a tabela do vector da edição de trabalho corresponde à tabela "402_polygon". Executa-se esta clonagem, porque têm de se efectuar alterações aos polígonos, evitando, assim, alterar a tabela inicial.
2. Execução de regras de validação sobre as geometrias do clone da edição de trabalho
   Na realização dos testes durante o desenvolvimento, detectaram-se algumas "irregularidades" nos polígonos dos vectores da edição de trabalho. Não significa que, por exemplo, tenham sido mal restituídos, mas apenas que a sua configuração não é válida para a execução de algumas das funções do PostGIS.

   Assim, esta validação ajusta os polígonos por forma a não gerar erros por existência daquelas irregularidades. Há, portanto, a possibilidade de existência de outras irregularidades não detectadas naquele vector, sendo necessário alterar o código fonte do *AutoDTM* para as corrigir.

   São utilizadas as seguintes validações e correcções: remoção de polígonos com apenas 3 pontos (o PostGIS gera erros se fizer leitura de polígonos com 3 ou menos pontos, pelo que a forma de o evitar foi eliminá-los), e a transformação de polígonos, cujas arestas se intersectam, em polígonos sem intersecções neles próprios.

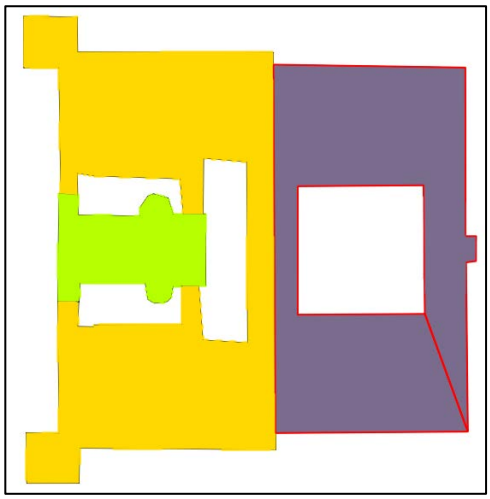


Figura 3.7 - Auto-intersecção de polígono (polígono que se auto-intersecta, à direita, está realçado a vermelho)

3. Adicionamento de uma zona de *buffer* a todos os polígonos da tabela de polígonos clonada
   Se o ficheiro de configurações assim o indicar (variável *polygonBufferSizeToAttenuateDesynchronizationBetweenPolygonsAndCloudPoints* maior que 0), adiciona uma zona de *buffer* à volta de todos os polígonos, ou dito de outra forma, efectua a dilatação do polígono. A necessidade deste *buffer* prende-se com as diferentes fontes de dados (nuvens de pontos, vector e ortofotos) não estarem perfeitamente sincronizadas (na Figura 3.8 está patente esta situação: o ponto rosa tem a cota do topo do edifício, mas está no exterior do polígono).

Figura 3.8 - Desfasamento entre nuvem de pontos (ponto rosa), polígonos (verde) e ortofoto

O acrescento deste *buffer* gera duas situações distintas:

- pontos sobre o terreno, e com a cota errada, passam a ser intersectados pelos polígonos;
- pontos sobre o terreno, mas com a cota correcta, passam também a ser intersectados pelos polígonos.

A primeira situação é aquela que se pretende corrigir, e a segunda é indesejável. No entanto, como as intersecções entre polígonos e nuvens de pontos servem para referenciar quais pontos devem ser interpolados, o que esta situação vai provocar é que se interpolem pontos, de cota correcta, com pontos vizinhos que também pertencem ao terreno: executar uma interpolação, nestas condições, fará variar de forma irrelevante a cota do ponto a

interpolar. Ainda assim, na eventualidade da geração de *outliers*, a última filtragem do processamento também os detectará e corrigirá.

Nos diferentes testes executados, um *buffer* de 10 metros (correspondente ao espaçamento entre os pontos da nuvem de pontos) provocou reduções significativas na RMSE (corrigiu os erros pelo desfasamento das fontes, não tendo gerado *outliers*). Embora empiricamente o valor que se deve utilizar deva ser igual ao espaçamento entre pontos da nuvem de pontos, será possível obter resultados diferentes, eventualmente melhores, ajustando caso a caso.

4. Criação e preenchimento da tabela da nuvem de pontos secundária (final)
   A tabela com o resultado final tem como atributos as colunas *gid* e *geom*, correspondentes ao *id* e à geometria propriamente dita, e também os atributos booleanos *isBuilding*, *isVegetation*, *isTree* e *filtered*, para controlo da execução. Há um último atributo, *elevation*, utilizado durante o desenvolvimento apenas para *debugging*, e mantido, na versão final, para mais facilmente verificar e detectar anomalias.

   Após a sua criação, todos os pontos da nuvem de pontos primária (na Figura 3.5, esta tabela é a denominada por "np_semadocamento_40_41") serão copiados e inseridos nesta nova tabela, com todos os atributos booleanos a receberem o valor *false*.

	gid integer	geom geometry(PointZM)	filtered boolean	isbuilding boolean	isvegetation boolean	istree boolean	elevation numeric
1	1	01010000C0000000	f	f	f	f	29.291
2	2	01010000C0000000	f	f	f	f	29.723
3	3	01010000C0000000	f	f	f	f	31.951
4	4	01010000C0000000	f	f	f	f	33.014
5	5	01010000C0000000	f	f	f	f	32.564
6	6	01010000C0000000	f	f	f	f	32.734
7	7	01010000C0000000	f	f	f	f	31.468
8	8	01010000C0000000	f	f	f	f	33.04
9	9	01010000C0000000	f	f	f	f	33.32
10	10	01010000C0000000	f	f	f	f	34.817

Figura 3.9 - Extracto de tabela da nuvem de pontos secundária, após a sua criação e preenchimento

Tal como sucede na tabela do vector da edição de trabalho, a nuvem de pontos primária é clonada numa nova tabela, para todas as manipulações de dados serem feitas nesta, em vez de na original.

5. Remoção de pontos duplicados
   O ISAE, aquando da criação da nuvem de pontos durante a correlação automática de imagens, produz alguns pontos duplicados (pontos com *gid* diferentes, mas com as mesmas coordenadas 3D). Este facto cria dois problemas: para além de afectar a eficiência da aplicação, ainda que de forma marginal, dificulta a leitura de pontos da tabela para, por exemplo, preenchimento de uma matriz. Por estes motivos, nesta fase, são apagados os pontos duplicados.

	gid integer	st_x double precision	st_y double precision	st_z double precision	gid integer	st_x double precision	st_y double precision	st_z double precision
1	38577	94810	216170	111.138	38578	94810	216170	111.138
2	41584	94780	216280	120.278	41583	94780	216280	120.278
3	44589	94750	216390	133.136	44590	94750	216390	133.136
4	121954	94770	219220	161.834	121953	94770	219220	161.834
5	128466	94800	219460	150.355	128467	94800	219460	150.355

Figura 3.10 - Pontos, da nuvem de pontos, com coordenadas duplicadas, gerados pelo ISAE

6. Catalogação de pontos *isBuilding* e *isVegetation*

   O vector da edição de trabalho contém todos os objectos restituídos pelos operadores, com excepção da altimetria, estando divididos por diferentes *layers*, segundo as Normas Técnicas da Aquisição de Dados ([5]) do IGeoE. Do conjunto total, para o *AutoDTM*, são apenas relevantes os seguintes polígonos:

   - Edifícios – correspondem aos polígonos das *layers* 16, 17, 18 e 19.
   - Vegetação – corresponde aos polígonos da *layer* 41.

   Essas *layers*, para além dos polígonos definidores das áreas dos edifícios e vegetação, contêm ainda outros polígonos, designados por células, que se constituem apenas como símbolos e grafismo para enriquecimento da informação das cartas. Uma das formas de os diferenciar é através de um outro atributo, denominado *entity*, que, para os polígonos propriamente ditos, tem como valor "*Closed Shape*" e, para células, tem valor "*Cell*". A Figura 3.11 mostra um desses exemplos: uma casa em ruínas, com o respectivo símbolo gráfico sobre a mesma. Ambos os polígonos (o que representa rigorosamente a casa, e os 6 círculos que se constituem como símbolo de ruínas) têm o valor 16 como atributo *layer*, mas a *entity* difere.

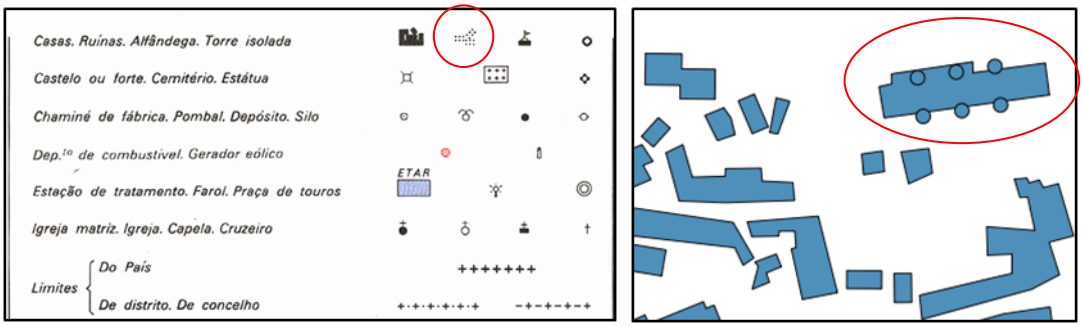


Figura 3.11 - Realce de célula (símbolo de ruínas) e edifício num vector e na legenda da carta

Numa primeira fase o *AutoDTM* liga-se à base de dados, e através de *queries* SQL, faz a leitura de todos os edifícios e vegetação. Essa leitura é filtrada pelo atributo *layer* com os valores referidos nos pontos supra, e também pelo atributo *entity*, com os valores '*Closed Shape*' (com esta restrição, as células deixam de ser consideradas), criando um objecto para cada polígono. Para este procedimento, necessita que a tabela do vector tenha um atributo *layer*, bem como um atributo *entity*. Como esses atributos existem na tabela, porque foram importados do *shapefile* (por intermédio de conversão do DGN original), e como é pouco expectável que esta organização de dados, no IGeoE, se altere, esta informação está *hardcoded* no *AutoDTM*. Portanto se, no futuro, ela se alterar, é necessário alterar o código fonte.

Na fase seguinte, através de *queries* SQL (utilizando a função "ST_Contains", [53], do PostGIS), cada um daqueles polígonos é intersectado com a nuvem de pontos secundária, catalogando cada um dos pontos intersectados como *isBuilding* ou *isVegetation*, consoante o polígono seja um edifício ou vegetação.

Como uma nuvem de pontos corresponde a uma pequena área total da carta (no caso da 402, 1/36), para optimizar a execução, evitando a leitura de todos os edifícios e vegetações de todo o vector, o *AutoDTM*, através das coordenadas da nuvem de pontos (obtidas também por *query* SQL), faz apenas a leitura dos polígonos nessa área.

Após estas operações, o *AutoDTM* está pronto a realizar as filtragens propriamente ditas.

3.2.2 Filtragem de vegetação (área superior a 150 m x 150 m)

Conforme referido anteriormente, a vegetação é uma das características do terreno que necessita ser removida, para se obter a correcta cota do solo. As zonas de vegetação com área superior a 150 m x

150 m são restituídas pelos operadores e, devido à sua dimensão, o terreno sob elas pode conter características que não são capturadas nas fotografias aéreas, nomeadamente linhas de água.

Como não há nenhum recurso à disposição para verificar e garantir a configuração desse terreno, a confirmação é efectuada durante o processo de produção da nova edição. Assim, a abordagem que melhor se adequa é manter o terreno conforme a edição anterior (a alternativa era interpolar esses pontos, o que ignoraria qualquer forma do terreno já identificada anteriormente).

Para o concretizar, pode-se utilizar, como fonte de informação, um de dois recursos: vector da edição anterior ("v3d"), ou o DTM (ficheiro TIF) da edição anterior.

O *AutoDTM*, nesta filtragem, executa as seguintes acções:

- Prepara a forma de obtenção de informação (seja pelo vector "v3d", seja pelo DTM, descritas infra).
- Obtém, da base de dados, todos os pontos catalogados como *isVegetation* e com o valor *filtered* como *false*, e transforma-os em objectos Java
- Itera cada um dos pontos a filtrar. Por cada ponto:
    - Obtém a cota na edição anterior (vector ou DTM), através das coordenadas 2D desse ponto.
    - Substitui, na nuvem de pontos secundária, a cota anterior pela nova, e ajusta o atributo *filtered* para *true*.

Os DTM, no presente, são gerados utilizando o ArcGIS. Dando como *input* a altimetria (curvas de nível e pontos cotados) e a hidrografia (linhas de água, cursos de água, bacias, etc), é gerado um TIN, que, por sua vez, é transformado em DTM (*raster* em escala de cinzentos, cujo valor do pixel corresponderá à cota). A execução é regulamentada por [55], onde, resumidamente, se define a hidrografia como *hard lines* e a altimetria como *soft lines*: as *soft lines*, quando se cruzam com *hard lines*, alteram a sua forma, criando concavidades/convexidades (por exemplo, nos casos em que linhas de água se cruzam com curvas de nível).

Embora o DTM da edição anterior contenha, em teoria, mais informação que o vector, na prática já sofreu várias alterações e adoçamentos, o que geraria acumulação de erros. De qualquer forma, apenas se utilizou o vector da altimetria da edição anterior, devido ao DTM da edição anterior estar em formato obsoleto. Ainda assim, através do ficheiro de configurações, é possível definir qual metodologia se pretende utilizar.

Todas as filtragens da presente metodologia, com excepção desta, executam interpolações de pontos utilizando pontos vizinhos, onde é necessário garantir que estes pertencem ao solo. Como esta filtragem é a única que não o faz, deverá ocorrer em primeiro lugar para, assim, disponibilizar mais pontos pertencentes ao solo às restantes filtragens.

Utilizando, por base, o vector de altimetria da edição anterior

Nesta modalidade, o *AutoDTM* gera um TIN a partir do vector de altimetria da edição anterior, para, a partir daí, poder obter a cota de cada ponto, dadas umas coordenadas 2D.

Para optimizar a execução e não ter de criar o TIN de todo o vector, antes de o produzir verifica qual a área ocupada pela nuvem de pontos, adiciona-lhe um *buffer* de 200m (definido empiricamente, e *hard coded*) e através da função "ST_Intersects" do PostGIS, apenas utiliza as curvas de nível dessa área.

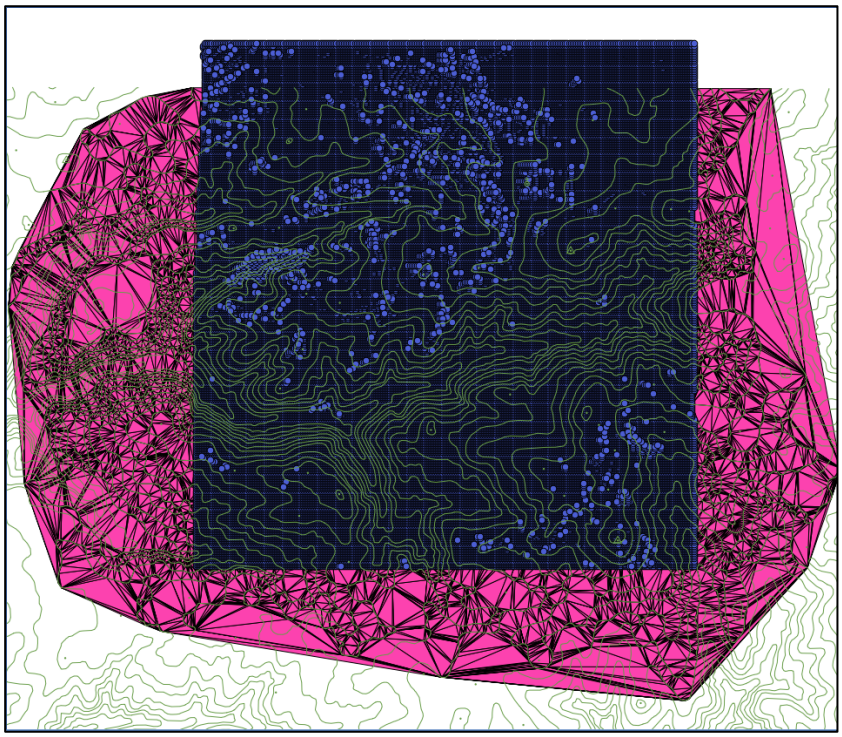


Figura 3.12 - TIN gerado a partir de curvas de nível na área da nuvem de pontos

Cada curva de nível é constituída por diversos pontos, que são utilizados para a geração do TIN, executada através da função "ST_DelaunayTriangles[3]", do PostGIS, e cada um dos triângulos produzidos é introduzido numa nova tabela (um polígono triangular, 3D, por cada triângulo gerado).

Finalmente, os pontos, previamente catalogados como *isVegetation*, são obtidos da base de dados, transformados em objectos Java, iterados, e é feita a verificação de a qual triângulo esse ponto pertence (em coordenadas 2D). Identificado o triângulo respectivo, por intermédio da equação do plano, é calculada a cota e, na nuvem de pontos secundária, a coordenada Z desse ponto é ajustada para o novo valor.

Utilizando, por base, o DTM da edição anterior

Na modalidade do DTM anterior, é necessário definir, no ficheiro de configurações, qual a localização do ficheiro *raster* desse DTM. Como se espera que o DTM tenha sido gerado pelo ArcGIS, juntamente com o *raster* (ficheiro TIF), há também mais dois ficheiros com informação do DTM criado: ambos com o mesmo nome do *raster*, mas um com extensão XML e outro com extensão AUX.XML.

O ficheiro AUX.XML contém informação sobre as bandas utilizadas. O *raster* é um ficheiro em escala de cinzentos, onde o valor de cada pixel corresponde a uma cota. Esses pixeis podem ter um valor entre um mínimo e um máximo (cada *raster* tem mínimos e máximos diferentes), definido nesse ficheiro AUX.XML. Esse ficheiro também tem a correspondência entre o valor mínimo e a cota mínima, e o valor máximo e a cota máxima (o valor do pixel é em vírgula flutuante, a cota é em metros).

O ficheiro XML contém informação correspondente a um *world file*: coordenada real do ponto superior esquerdo do *raster*, bem como o tamanho do pixel nas direcções X e Y.

[3] A Triangulação de Delaunay é efectuada num conjunto de pontos, onde nenhum deles está dentro da circunferência formada por qualquer triângulo (excepto os 3 pontos que determinam o triângulo).

O conteúdo destes ficheiros é lido pelo *AutoDTM* utilizando XPath. É usada esta ferramenta e não outra, por dois motivos: para além de fazer parte da biblioteca *standard* do Java, as *queries* necessárias para obter os valores pretendidos são triviais.

Com a informação destes ficheiros auxiliares, o *AutoDTM* abre o *raster* utilizando o OpenCV e, a partir de duas coordenadas reais, converte-as para coordenadas de imagem, faz a leitura do valor do pixel, e converte-o na cota respectiva. Assim, iterando os pontos catalogados como *isVegetation*, obtém a cota do DTM pelo método acabado de descrever, actualizando a cota de cada um desses pontos, na nuvem de pontos secundária.

Uma nota final prende-se com a constituição do *world file.* Como se pode verificar em [36], para além das coordenadas do canto superior esquerdo e do tamanho do pixel nos eixos X e Y, faz também parte destes ficheiros a eventual rotação do *raster* sobre os eixos X e Y. Na cadeia de produção nunca se utilizam imagens com rotação, e, portanto, o *AutoDTM,* neste passo, assume rotação 0.

3.2.3 Filtragem de árvores/grupos de árvores

As árvores e pequenos grupos de árvores são, na realidade, o principal problema de todo o processo: são características de terreno que não são restituídas, e não podem ser ignoradas, devido à sua altura ser relevante e, por conseguinte, causar ruído e erro na geração de DTM.

Esta filtragem reproduz grande parte do algoritmo descrito em [23], e é executada pelo *AutoDTM* em diversas fases, controláveis a partir do ficheiro de configurações (este aspecto é relevante, em virtude de permitir que se execute selectivamente cada uma das fases; por exemplo, é possível mandar executar o *AutoDTM* apenas para treinar o algoritmo de detecção, não fazendo qualquer tipo de avaliação ou carregamento de dados na base de dados).

De forma geral, o *AutoDTM* faz a detecção de árvores numa ortofoto, e produz uma imagem a preto e branco, onde as áreas a branco correspondem a árvores, e as áreas a preto a tudo o que não seja árvore. Essas áreas a branco são, seguidamente, convertidas em polígonos, estes são carregados na base de dados, e os pontos da nuvem de pontos secundária, que são intersectados por esses polígonos, são interpolados.

Treino e teste

O *adaboost,* no *AutoDTM*, é treinado através de uma ou mais imagens de treino e respectivas máscaras (construídas manualmente), com todos esses ficheiros a estarem definidos no ficheiro de configurações.

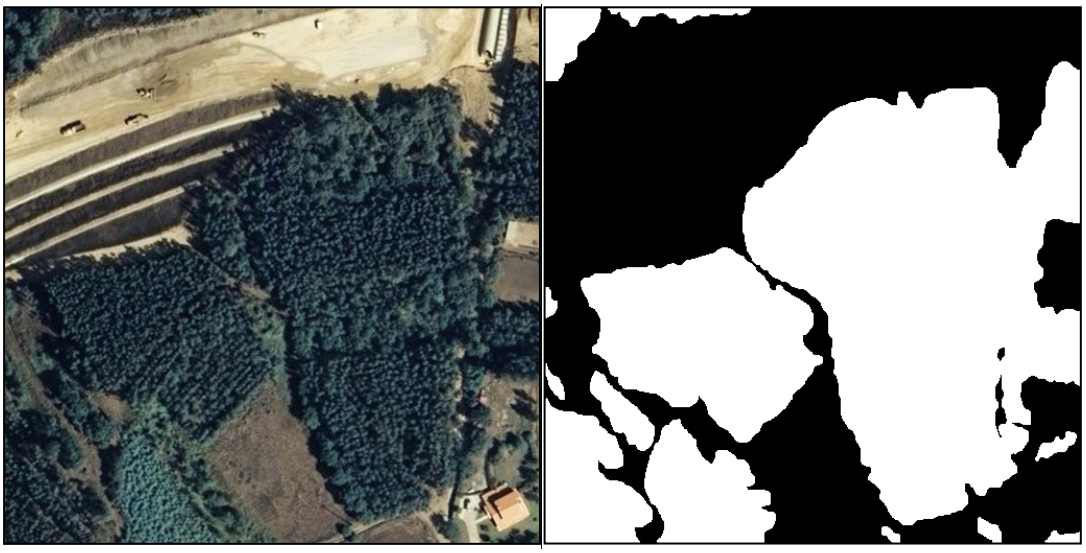


Figura 3.13 - Imagem de treino e respectiva máscara

Segundo [23], basta treinar com 1% do conjunto total, para se obter eficiência superior a 90%. Durante o desenvolvimento do *AutoDTM* a relação treino/conjunto total foi mais elevada (imagem de

treino de, aproximadamente, 500x500, para avaliar três ortofotos de cerca de 8000x5000), obtendo-se uma eficiência semelhante (cerca de 90%). A Figura 3.13 mostra, à esquerda, o extracto de ortofoto utilizado para treino, e à direita a respectiva máscara criada manualmente (a branco, áreas de árvores).

O *AutoDTM* realiza as seguintes acções para efectuar o treino e construir um modelo:

1. Abre a imagem de treino e a máscara, utilizando o OpenCV.
    - Por causa do elevado tamanho das ortofotos, implementou-se a possibilidade de fazer o tratamento das imagens por *tiles*, armazenando-as em disco. Assim, por controlo de variáveis no ficheiro de configurações, pode-se definir o tamanho máximo de cada *tile*. Como algumas *features* necessitam de informação de pixéis vizinhos, para garantir que os pixéis nas margens destas *tiles* possuem toda a informação necessária, pode-se também definir o valor de um *buffer* em redor da *tile* (pelo menos 8 pixeis, visto que as *features* de textura vão buscar informação até essa distância). De realçar que, caso no futuro se modifiquem as *features*, este *buffer* deverá ser ajustado de acordo, de forma a haver uma melhor gestão da memória.
    - O *adaboost* não permite incremento do treino em fases posteriores, excepto se se guardar todas as amostras entre incrementos. Isto acontece porque o algoritmo dá pesos às amostras; sem elas, treinos incrementais não fazem sentido. Por este motivo, todas as amostras e respectivas *features* têm de estar em memória durante o treino. Considerando que há 27 *features*, cada *feature* utiliza um *double*, e se cada *double* ocupar 8 *bytes*, cada amostra ocupa 216 bytes em memória. Treinando com imagens com um tamanho equivalente a, por exemplo, 3000x2000, significa que em memória estarão perto de 2GB (apenas ocupados pelo conjunto das amostras). No entanto, como referido em [56], o máximo de *heap* de uma Java Virtual Machine (JVM) num sistema operativo de 32 bits é de cerca de 1.6GB. Portanto, tendo em conta o tamanho do conjunto de treino, pode haver necessidade deste ser realizado em sistema operativo de 64 bits com a JVM.

    - Devido a este último ponto, embora o mecanismo de *tiles* tenha como finalidade fazer melhor gestão de recursos, no caso do treino, não é mesmo possível evitar que todas as amostras estejam em memória.
    - No conjunto de treino, cada amostra tem associada uma classe. Esta classificação é obtida pela máscara: pixel a branco é árvore, pixel a preto é não-árvore.
2. Itera as imagens de treino, em cada uma itera as *tiles* (se assim tiver sido definido), e por *tile* itera pixel a pixel. Por cada pixel:
    a. Gera cada uma das 27 *features*, construindo a amostra.
        1) 3 *features* do CIEL*a*b*
        O OpenCV disponibiliza função de conversão de sRGB, formato das ortofotos, para este sistema de cores.
        2) 3 *features* do *illumination-invariant*
        Este sistema de cores teve de ser implementado com base em [29]. Primeiro, converte-se de sRGB para CIE XYZ, através do OpenCV, visto no documento de referência ser esse o espaço de cores das imagens de *input*. De seguida, operam-se as operações de matrizes descritas no mesmo documento, obtendo-se os três canais do *illumination-invariant*.
        3) 18 *features* da textura

Em [23] utilizaram-se filtros Gaussianos como *kernel* para efectuar as convoluções, embora sugiram que os mesmos resultados poderiam ser alcançados com outros tipos de *kernel*, tal como o filtro de Gabor. Foi este que foi usado por ser de mais fácil implementação.

Mantiveram-se os valores das variáveis defindos em [23], nomeadamente o *sigma* com valores 1, $\sqrt{2}$ e 2 (tamanho do *kernel*) e o *theta* com valores 0, 30, 60, 90, 120 e 150 (rotação do *kernel*), e às restantes variáveis (*phi*, *gamma* e *bandwidth*) foram atribuídos os valores definidos em [57] (os mais adequados para se obterem características de textura).

A implementação do *kernel* foi feita a partir do código da mesma função em Matlab ([58]), adaptada para Java no *AutoDTM*. A convolução para geração das imagens de textura foi efectuada, por sua vez, por função do OpenCV.

4) 3 *features* da entropia

Estas *features* calculam a entropia do brilho (canal L* do espaço de cores CIE L*a*b*) em *kernels* de tamanho 5x5, 9x9 e 17x17. No documento de referência é sugerida uma alternativa de optimização, que passa por calcular três vezes a entropia em janelas de 5x5, mas com a imagem de tamanho normal, reduzida a metade e a quatro vezes. Efectivamente torna o processamento mais célere, mas à custa de valores diferentes (embora o *paper* refira que não é diferença relevante).

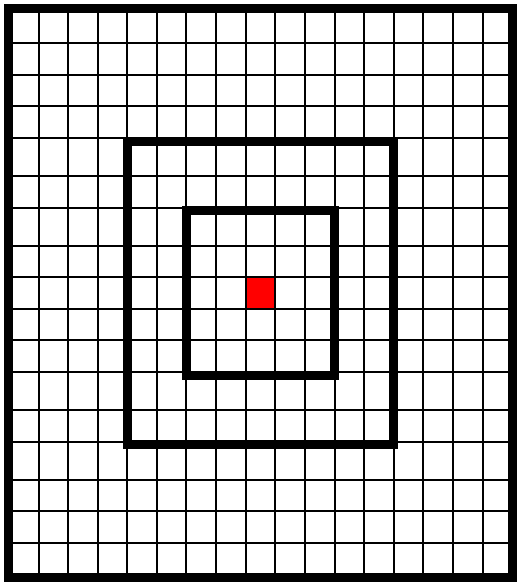


Figura 3.14 - Esquema dos 3 *kernel* para o cálculo da entropia

A geração destas *features*, no *AutoDTM*, foi implementada de forma ainda mais optimizada, obtendo os mesmos resultados que o cálculo individual das 3 janelas. A optimização passa por, por pixel, ler sempre uma janela de 17x17 pixeis, distribuindo-os por 3 *sets*, correspondendo a cada um dos tamanhos de janela, apenas numa iteração.

b. Define a classe da amostra. Para o fazer, procura, na máscara, o pixel análogo ao que está a ser analisado. Se esse pixel for branco, classifica a amostra como "árvore", caso contrário classifica-a como "não-árvore".

3. Depois do conjunto de treino estar completo, treina o classificador, dando-lhe como *inputs* esse conjunto, bem como a definição do número de iterações a executar, ou seja, o número de classificadores fracos (com base em [23], deverão ser definidas 200 iterações, confirmados pelos testes realizados: com mais iterações não afecta o resultado da classificação e, com menos iterações, reduz a percentagem de acerto).
   Testaram-se diferentes bibliotecas para a execução do *adaboost*, nomeadamente o OpenCV, o JBoost e o Weka. A opção recaiu no Weka, pelos seguintes motivos:
   - O OpenCV não permite definir as amostras dos conjuntos de treino e teste (dá-se apenas um conjunto de amostras, e a própria função, aleatoriamente, escolhe as amostras que pertencerão a um e outro conjunto). No caso do *AutoDTM*, este pormenor é necessário, porque se pretende "enviesar" o treino do *adaboost*, forçando-o a treinar com pequenas imagens que correspondam a determinados pormenores do terreno, como árvores de diferentes cores, piscinas ou edifícios (se as amostras fossem obtidas de um conjunto aleatório, estes pormenores poderiam não ser treinados). Além disso, não produz, por si, qualquer estatística, nem disponibiliza informação suficiente para se poder gerá-la.
   - Embora o JBoost disponibilize execução em múltiplas *threads* (o que aumentaria bastante o desempenho, já que a aprendizagem é um processo demorado) e bons *scripts* para produção de *outputs*, houve algumas execuções em que o algoritmo foi interrompido inesperadamente, ou dava resultados que não eram os expectáveis. Aparentemente, é uma biblioteca estável, já vai na versão 2.0, mas o seu comportamento não corresponde (pelo menos no que diz respeito ao *adaboost*, já que é uma biblioteca que implementa variados algoritmos de *boosting*).

> Há ainda uma alternativa que permite optimização do desempenho, possibilitando inclusivamente paralelização, que é fazer a própria implementação do *adaboost* (adaptando, por exemplo, o código fonte do Weka)

4. Depois de treinado, o modelo é serializado em formato Java (embora a própria biblioteca disponibilize métodos de serialização, esta foi implementada no próprio *AutoDTM*) e guardado como classe Java (gerada automaticamente pelo Weka).
5. Efectua o teste do modelo, utilizando a orto de teste e respectiva máscara.
   O teste do modelo é executado pela mesma rotina que efectua o treino: abre a orto de teste e a máscara pelo OpenCV, também cria as *tiles*, se assim tiver sido definido, gera as *features* de cada pixel, cria as amostras do conjunto de teste, e define a classe de cada amostra com base na máscara. Este é o único *input* dado para testar o modelo.
6. Regista no ficheiro de log, e no *stream standard* de *output*, as estatísticas relativas ao teste.
   Neste ponto, realça-se a seguinte situação: a percentagem de acerto é a que corresponde ao modelo, mas não é a percentagem final, em virtude de ainda ter de se realizar o refinamento da imagem. Pelos testes efectuados, o refinamento melhorou a percentagem até mais cerca de 3%. Assim, juntamente com as estatísticas do modelo geradas pelo Weka, aparecerá também uma mensagem no ficheiro de *log* a informar acerca deste pormenor.
7. Se tiver sido definido no ficheiro de configurações, executa *cross validation*.

> Para a *cross validation*, está *hard coded* que serão formados 5 conjuntos de teste. Para modificar o número de conjuntos, é necessário alterar o código fonte.

Por controlo de variáveis no ficheiro de configurações, é possível dar indicação ao *AutoDTM*, para guardar as imagens geradas no cálculo das diferentes *features*.

Avaliação

Para executar a avaliação, é necessário definir, nas configurações, qual o ficheiro que contém o modelo serializado, permitindo, assim, a execução do treino e da avaliação de forma independente. A execução expectável do *AutoDTM*, em relação à detecção de árvores, é mesmo essa: efectuar o treino do modelo numa das execuções, utilizando-o para avaliar ortofotos em múltiplas execuções.

A execução da avaliação é semelhante ao treino e teste, com alguns pormenores diferentes:

1. Abre a ortofoto de avaliação, utilizando o OpenCV.
   Uma das diferenças para o treino e teste está na abertura da ortofoto de avaliação. Aqui, ao contrário do treino, o mecanismo de *tiles* pode fazer efectivamente uma melhor utilização de recursos, visto não ser necessário ter todas as amostras em memória. Se as variáveis do ficheiro de configuração estiverem devidamente definidas, só é necessário que o sistema operativo disponha de memória suficiente para abrir a ortofoto de uma vez só, visto que, de seguida, se procede à geração de *tiles.*
2. Itera cada uma das *tiles* (se assim tiver sido definido), e por *tile* itera pixel a pixel. Por cada pixel gera as 27 *features*, cria as amostras, e constrói o conjunto de avaliação.
3. Efectua a avaliação
   Cria o objecto Java do modelo a partir do ficheiro serializado, avalia o conjunto de avaliação, e produz uma imagem de output ao estilo das máscaras atrás referidas (imagem a preto e branco, onde pixéis brancos correspondem a "árvore" e os pretos a "não-árvore"). Se estiver a avaliar diferentes *tiles*, no final desta avaliação, o conjunto de amostras é "descartado", libertando a memória ocupada.

   Durante a avaliação, o *AutoDTM* vai preenchendo uma matriz do mesmo tamanho da ortofoto, com a probabilidade calculada para o pixel. Esta matriz é utilizada, mais tarde, durante a execução do refinamento da imagem (algoritmo *min-cut/max-flow*).
4. Se foram utilizadas *tiles*, estas são iteradas e as imagens produzidas na avaliação são novamente unidas, criando uma única máscara final (do mesmo tamanho da ortofoto avaliada).
5. Executa o refinamento da imagem final
   Para a execução *min-cut/max-flow* é necessário dar, como argumentos, a matriz de probabilidades gerada durante a avaliação, bem como a matriz que corresponde à imagem avaliada.

   Dos testes efectuados, o refinamento melhorou as imagens avaliadas em cerca de 3%. A forma de verificação foi por comparação pixel a pixel da imagem final, já refinada, produzida pelo *AutoDTM*, com uma máscara da mesma região, mas gerada manualmente.

Durante o desenvolvimento, utilizou-se um mecanismo para ajudar a verificar, visualmente, o desempenho da detecção de árvores, que se designou por coloração do output. Se esta variável estiver definida no ficheiro de configurações, a máscara produzida, na avaliação, vem em escala de cinzentos: o preto significa "não-árvore", e os 4 níveis de cinzentos, juntamente com o branco, significam "árvore", utilizando a matriz de probabilidades construída durante a avaliação (entre 50 e 60% é um tom mais escuro, dos 60 aos 70% é ligeiramente mais claro, até à cor branca, entre 90 e 100%).

De realçar que a coloração do output deverá ser utilizada apenas para *debugging*, visto que a transformação de polígonos, para a base de dados, apenas considera os pixéis efectivamente brancos (é possível, claro, que qualquer tom de cinzento seja assumido como árvore também, mas, como este mecanismo tem pouca utilidade para além de verificar o funcionamento do algoritmo, não foi implementado).

Transformação em polígonos

O produto da avaliação é uma imagem, constituída apenas por pixeis pretos ou brancos, onde os brancos significam "árvore". São as áreas com esses pixéis que interessa obter e manipular.

Para fazer o tratamento da informação contida na imagem, com os mesmos métodos das restantes filtragens, é necessário que essa informação seja importada para a base de dados. O PostGIS disponibiliza ferramentas para o fazer, nomeadamente o ficheiro "raster2pgsql.exe", que converte um *raster* em inserções SQL, para serem executadas numa base de dados geoespacial.

O "raster2pgsql" é executado a partir do *AutoDTM*, que cria um novo processo para essa finalidade. Pode receber vários argumentos, mas o que o *AutoDTM* utiliza, *hard coded*, é apenas o "-R", que dá indicação que se registe o *raster* como ficheiro do sistema, em vez de fazer o *upload* de todo o *raster* (só os metadados e o caminho para o ficheiro é que efectivamente são guardados na base de dados; o tratamento do *raster* será mais lento, mas, como só se pretende obter as áreas brancas da imagem, não há vantagens em duplicar informação). O *raster* é importado para uma nova tabela, criada dinamicamente pelo *AutoDTM*.

Um dos possíveis argumentos é a definição do sistema de coordenadas do *raster*. No entanto, o *AutoDTM*, por opção de implementação, não faz qualquer alteração.

Devido a este *raster*, que se pretende importar para a base de dados, ser gerado pelo próprio *AutoDTM*, e não ter definido, portanto, qualquer *datum*, caso se se fizesse a importação directa, seria assumido pelo PostGIS que a sua georreferenciação seria na origem do referencial.

Para evitar esta assunção errada, e visto que este *raster* é uma máscara da ortofoto, e que cada ortofoto tem associado um *world file* que contém toda a informação necessária para georreferenciação, antes de executar o "raster2pgsql", o *AutoDTM* faz uma cópia desse ficheiro de *world* e utiliza-a na importação (se houver um *world file*, com o mesmo nome do *raster*, mas extensão tfw, o "raster2pgsql" faz a sua leitura para georreferenciar o *raster*). A outra alternativa seria guardar o *raster* como geotiff, embutindo, nos metadados, a informação da georreferenciação.

Depois do *raster* estar na base de dados, a identificação e exportação dos polígonos formados pelos pixéis brancos é executada através da função do PostGIS "ST_DumpAsPolygons", juntamente com a condição "WHERE val=255", onde a variável "val" é o valor do pixel. O *AutoDTM* executa esta função, inserindo os resultados numa nova tabela, cujo nome é definido no ficheiro de configurações, que conterá todos os polígonos formados pelos pixéis brancos. A estes polígonos não é acrescentado qualquer *buffer* (este mecanismo serve para corrigir o desfasamento entre *inputs* de diferentes fontes, mas, nesta situação, não é relevante, visto as áreas a branco terem sido calculadas a partir da própria ortofoto).

Caso esta tabela final, para onde os polígonos vão ser exportados, já exista, o comportamento do *AutoDTM* é incrementar a referida tabela: desta forma, é possível aumentar o seu conteúdo em diferentes execuções.

Catalogação dos pontos

O passo seguinte é a catalogação dos pontos da nuvem de pontos secundária como *isTree*. Tal como acontece na filtragem de edifícios, o que o *AutoDTM* faz é carregar, da base de dados, todos os polígonos gerados a partir do *raster*, e iterá-los. Por cada polígono, faz uma intersecção com a nuvem de pontos secundária, e os pontos resultantes serão catalogados como *isTree*. Para optimizar o desempenho desta operação, apenas são carregados os polígonos que estiverem na mesma área da nuvem de pontos (em vez de se carregarem todos os polígonos).

Interpolação dos pontos

A Tabela 3.3 foi baseada em [59], e sistematiza a classificação dos métodos de interpolação.

Tabela 3.3 - Classificação dos métodos de interpolação

Característica	Detalhe	Descrição
Extensão espacial	Global	Usam todos os pontos da amostra
	Local	Usam um pequeno conjunto de pontos vizinhos
Ajustamento	Exacto	Valor estimado é idêntico ao valor observado nessa localização
	Aproximado	Valor estimado é deferente do valor observado nessa localização; utilizado principalmente para evitar picos abruptos
Modelo	Determinístico	Pontos que estão mais próximos assemelham-se mais ao ponto a interpolar do que os distantes
	Probabilístico	Analisa-se autocorrelação espacial para modelar a variabilidade espacial; usam modelos de probabilidade para realizar as estimativas

Pela perspectiva pretendida para o *AutoDTM:*

- quanto à extensão global, como se pretende interpolar apenas pequenas regiões e se têm pontos de apoio suficientes para o fazer, os métodos de interpolação a utilizar deverão ser **locais**;
- quanto ao ajustamento, deseja-se uma interpolação **exacta**, devido à nuvem de pontos secundária ter por finalidade servir para gerar curvas de nível (as ferramentas que realizam este processo já executam uma serie de operações de adoçamento, o que faria acumular ainda mais erros);
- quanto ao modelo, devido à grande quantidade de pontos de apoio, e à reduzida distância aos pontos a interpolar, a opção mais adequada é **determinístico**.

Alguns dos exemplos de algoritmos locais, exactos e determinísticos, são *splines* e o *Inverse Distance Weighting* (IDW). Algoritmos que implementam *splines* não serão os mais adequados, porque suavizam a curva, e em superfícies rugosas, sabendo que se pretende rigor na interpolação, esse não é o método mais adequado.

O IDW é um método de implementação simples, amplamente utilizado, e que efectivamente representa a semelhança de pontos por proximidade espacial. No entanto, apenas considera a distância entre os pontos de apoio e o ponto a interpolar, descurando a localização dos primeiros. Este aspecto pode ser combatido, se se garantir que os pontos de apoio estão a 360º em redor do ponto a interpolar. Utiliza apenas a função do inverso da distância como peso para o cálculo da cota.

Outro dos métodos analisados foi o *kriging* ([60]): é local, exacto e probabilístico, e ponderou-se a sua utilização nos casos em que, eventualmente, possam não existir pontos de apoio suficientes para a interpolação. Utiliza o variograma como função de interpolação, permitindo, com diversos variogramas adequados à região do ponto a interpolar, ter em consideração a localização dos pontos de apoio. Contém mais alguns pormenores que o tornam um método complexo, nomeadamente definir uma distância a partir da qual não considera os pontos de apoio, mas caso use um variograma semelhante à função do inverso da distância, produz resultados iguais ao IDW.

Após análise dos métodos mais relevantes e que, à partida, melhor se adequam ao pretendido no *AutoDTM*, optou-se por se implementar, apenas, o IDW.

Esta decisão teve por base os seguintes aspectos:

- "quando os dados são abundantes, todos os métodos de interpolação produzem valores semelhantes" ([60]);
- "a precisão do *Inverse Distance Weighting* (IDW) e do *Kriging* é quase a mesma" ([61]);

- considerando uma nuvem de pontos com espaçamento de 10 metros, num *buffer* de 10 metros, em redor dos edifícios, há uma média de quase 6 pontos de apoio por quadrante, ou seja, pode-se assumir que há suficientes pontos de apoio, adjacentes e a 360º, em redor dos pontos/áreas a interpolar (num *buffer* de 30 metros esse valor sobe para quase 21 pontos; testes executados com o vector da carta 402 e com a nuvem de pontos 40-41, localizada no centro de Mafra).

A execução da interpolação é a seguinte:

- Carrega todos os polígonos gerados pelo algoritmo de detecção de árvores, transforma-os em objectos Java e itera-os. Por cada polígono:
  - Intersecta-o com a nuvem de pontos secundária, obtendo os pontos a filtrar.
  - Obtém os pontos de apoio (pontos que não estejam catalogados como *isTree*, *isVegetation* ou *isBuilding*, ou que estejam catalogados como *isFiltered*) em redor do edifício, numa zona de *buffer* cujo tamanho é definido no ficheiro de configurações. O tamanho desta zona de *buffer*, se devidamente ajustado, poderá melhorar resultados.
  - Itera os pontos a filtrar. Por cada ponto:
    - Distribui os pontos de apoio em quadrantes, guardando-os em quatro listas diferentes, e ordenando cada uma delas pela proximidade ao ponto a filtrar.
    - Verifica que existe, em cada lista, o valor mínimo de pontos por quadrante definido no ficheiro de configurações. O *AutoDTM*, para efectuar interpolações, utiliza o IDW, e impõe que exista, no mínimo, 1 ponto de apoio por quadrante (para este método dar resultados eficazes, é necessário que os pontos de apoio estejam dispersos a 360º em redor do ponto a interpolar). Não existindo, tal facto é registado no output do *AutoDTM*, não interpola o ponto, e inicia o tratamento do ponto seguinte. É natural que haja pontos que não consigam ser interpolados, nomeadamente aqueles que estão no limite da nuvem de pontos.
    - Interpola o ponto pelo IDW. Retira, de cada uma das listas atrás referida, a mesma quantidade de pontos de apoio (valor este que é definido no ficheiro de configurações), e utiliza-os na interpolação. Ainda que marginalmente, a quantidade de pontos utilizados na interpolação afecta a interpolação, e é um valor que, devidamente ajustado, poderá melhorar resultados.
    - Ajusta o atributo *filtered*, do ponto interpolado, para *true*.

3.2.4 Filtragem de edifícios

Esta filtragem utiliza a informação do vector da edição de trabalho, nomeadamente os polígonos que correspondem a edifícios (*layers* 16, 17, 18 e 19), intersecta-os com a nuvem de pontos secundária, e os pontos resultantes dessa intersecção são interpolados com os pontos vizinhos (pontos de apoio).

Tal como noutras fases da execução do *AutoDTM*, para evitar fazer a leitura de todos os edifícios presentes no vector, são verificadas as dimensões e a localização da nuvem de pontos a filtrar, e apenas os polígonos nessa área são efectivamente carregados.

A execução é bastante semelhante à fase da interpolação de pontos da filtragem de árvores, descrita em 3.2.3 (inclusivamente utiliza os mesmos métodos), e consiste em:

- Transforma os edifícios da base de dados em objectos Java.
- Itera os edifícios. Por cada edifício:

- Intersecta com nuvem de pontos secundária, obtendo os pontos a filtrar.
- Obtém os pontos de apoio (possibilidade de optimização, descrita na interpolação de pontos da subsecção 3.2.3).
- Itera pelos pontos a filtrar e, por cada ponto:
  - Distribui pontos de apoio por quadrantes.
  - Verifica se existe número mínimo de pontos por quadrante, utilizando o mesmo número de pontos de cada quadrante como pontos de apoio para efectuar a interpolação.
  - Efectua a interpolação, utilizando os pontos de apoio definidos no ponto anterior.
  - Ajusta o atributo *filtered* para *true*.

À semelhança da filtragem de vegetação de área superior a 150 m x 150 m, a filtragem de edifícios podia ser executada de forma a ir buscar à base de dados todos os pontos catalogados como *isBuilding*, em vez de ir buscar os polígonos e iterá-los – o que teria efeitos no desempenho da filtragem.

No entanto, como se pode observar na Figura 3.15, se a filtragem fosse feita ponto a ponto, aquele que está selecionado só conseguiria ter pontos de apoio no segundo quadrante, caso a zona de *buffer* de pesquisa de pontos de apoio tivesse sido definida a mais de 50 m (o que faria perder desempenho, para além de 50 m solucionarem este caso específico, mas não solucionar todos os possíveis casos). Foi por este motivo que se optou por utilizar os pontos de apoio a uma distância fixa de *buffer* em redor do edifício, em vez de o fazer ponto a ponto.

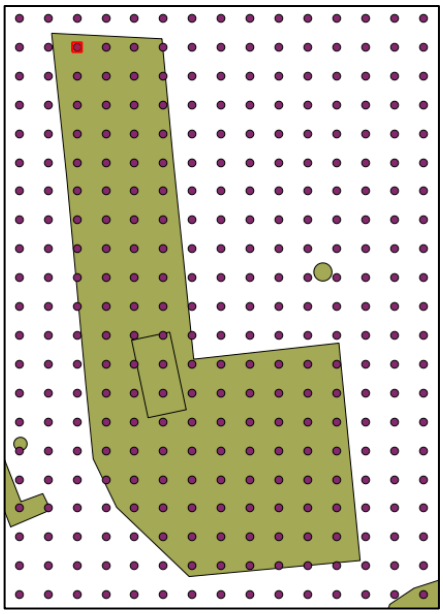


Figura 3.15 - Realce do motivo pelo qual a filtragem de edifícios deve ser executada polígono a polígono, em vez de ponto a ponto (nuvem de pontos com espaçamento de 10m)

3.2.5 Filtragem por declive

Esta filtragem tem duas finalidades: corrigir *outliers*, e corrigir o efeito do desfasamento de polígonos e ortofotos com nuvem de pontos.

Os *outiers* podem ser fruto de alguma anomalia originada nas filtragens anteriores, bem como podem surgir logo desde a correlação automática de imagens (se, por exemplo, algum dos pontos da nuvem primária coincidiu com um poste de alta tensão).

A segunda fonte de *outliers* é um desfasamento mínimo de coordenadas, gerado pela utilização de informação de diferentes origens: a ortofoto é gerada automaticamente, sofrendo alterações por forma

a permitir fazer medições mais exactas sobre ela; os polígonos são restituídos, em estereoscopia, por um operador; as nuvens de pontos são geradas, automaticamente, a partir das fotografias aéreas.

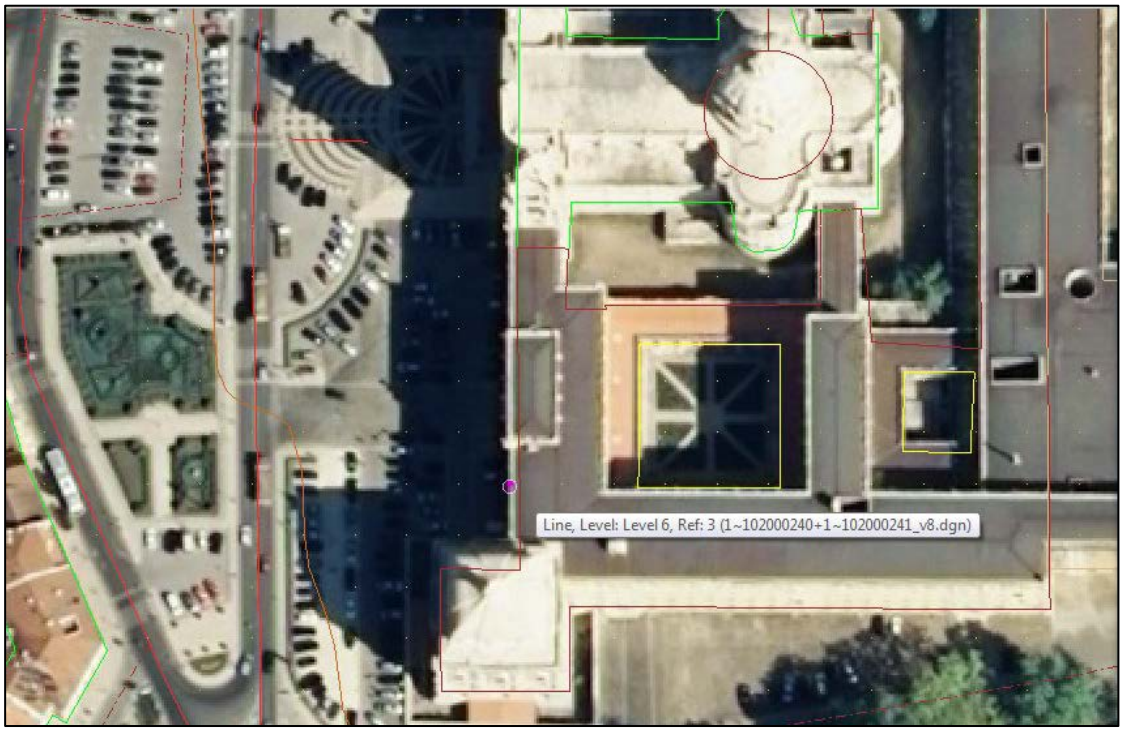


Figura 3.16 - Desfasamento entre nuvem de pontos, polígonos e ortofoto

A Figura 3.16, que mostra um extracto da zona Sudoeste do convento de Mafra, revela estes desfasamentos: os pontos a amarelo correspondem à nuvem de pontos; o polígono do vector que representa o convento, a vermelho menos vivo; e a própria ortofoto[4]. O ponto selecionado (círculo com centro cor-de-rosa) está localizado fora do polígono (e mesmo da ortofoto), mas tem a mesma cota que os terraços do convento (cerca de 257 metros). Como o convento de Mafra tem orientação Este-Oeste, todos os pontos daquela coluna, entre os dois torreões, são catalogados pelo *AutoDTM* como pertencentes ao terreno, mas todos têm a cota sobre-elevada. O ponto imediatamente a Oeste daquele que está seleccionado, no entanto, já tem a cota correspondente ao solo.

Estes desfasamentos também podem ser corrigidos com o aumento da zona de *buffer* em redor dos polígonos, controlado por variável no ficheiro de configurações. Para o *AutoDTM*, a eventualidade das zonas de *buffer* de diferentes polígonos se intersectarem é irrelevante, visto que apenas interpola os pontos uma vez (excepto a filtragem por declives).

Esta filtragem obedece aos conceitos teóricos descritos na subsecção 2.2.1:

- a estrutura de dados utilizada é uma nuvem regular de pontos;
- os testes na vizinhança são efectuados ponto a pontos (outro método poderá dar resultados diferentes);
- a medição da descontinuidade/critério de filtragem é por declive (outro método poderá dar resultados diferentes);
- a iteração dos pontos é simples (efectua apenas um varrimento; outro método poderá dar resultados diferentes);
- o ajuste dos pontos interpolados é feito por substituição (outro método poderá dar resultados diferentes).

[4] Parte deste problema é devido ao facto de a ortorrectificação ser efectuada com base na cota terreno do DTM, e na realidade o topo do edifício está alguns metros acima (confrontar com *true ortophotos*, [62]).

A execução, pelo *AutoDTM*, é a seguinte:

- Lê todos os pontos da nuvem de pontos secundária, da base de dados, e cria objectos Java correspondentes.
- Organiza os pontos numa matriz rectangular.
- Itera os pontos da matriz, da esquerda para a direita, e de cima para baixo. Por cada ponto:
  - Calcula o declive com cada um dos 6 ou 8 pontos vizinhos, através da tangente (ver explicação infra sobre a motivação dos 6 ou 8 pontos).
  - Efectua interpolação se nalgum dos pontos vizinhos esse declive superar aquele definido no ficheiro de configurações (o ajuste deste ângulo pode melhorar resultados; o método de interpolação utilizado é o IDW, descrito em 3.2.3). De realçar que, embora no ficheiro de configurações o valor introduzido seja positivo, o que, na verdade, é comparado, é o negativo desse valor com o declive negativo dos pontos a comparar (dito de outra forma, só se executa interpolação, se o ponto central tiver maior cota que algum dos vizinhos).
  - Altera a cota do ponto interpolado, na nuvem de pontos secundária, para a nova cota, e define o valor *filtered* para *true*.

Para evitar o efeito do desfasamento entre nuvem, ortofoto e vector, para além do já referido aumento da zona de *buffer* por cada polígono, também se pode fazê-lo, através da manipulação da variável do ficheiro de configurações, que define uma interpolação com 8 ou 6 pontos de apoio, o que pode alterar os resultados.

230	257	233
231	257	234
231	257	232

Interpolação a 8: 238,125

Interpolação a 6: 231,833

Figura 3.17 - Diferenças na interpolação com 6 ou com 8 pontos de apoio

A Figura 3.17 representa a situação hipotética da filtragem de *outliers* do ponto realçado na Figura 3.16, assumindo que se utilizou 0 metros como zona de *buffer* em redor dos polígonos, e que a filtragem dos edifícios ajustou correctamente os pontos sobre o convento de Mafra (excepto a faixa daqueles pontos centrais, visto terem a cota do topo do edifício, mas estarem localizados fora do polígono). Como este efeito de desfasamento gera, de forma geral, linhas isoladas de cota mais elevada, dos 9 pontos disponíveis, 3 deles pertencerão a essa linha (no exemplo é linha vertical, mas poderá ocorrer com outra configuração, e tendendo a ter sempre 3 pontos sobre-elevados). É por este motivo que se se utilizarem 8 pontos, a cota é apenas reduzida "ligeiramente" (2 dos pontos prejudicam a interpolação e, neste exemplo, a diferença entre ambas gera uma diferença de mais de 6 metros).

Por outro lado, em zonas de terreno que não tenham qualquer *outlier*, a interpolação a 6 pontos continua a garantir resultados eficientes, visto utilizar sempre pontos a quase 360º, em redor do ponto central. Realça-se que o *AutoDTM* organiza os pontos de apoio por cota crescente, escolhendo os 6 primeiros pontos (ou todos) para a interpolação (na prática, utiliza sempre os pontos de cota mais reduzida).

O *AutoDTM* está preparado para receber nuvens de pontos regulares, de qualquer espaçamento, sendo apenas necessário alterar a variável de configuração respectiva. Na verdade, exceptuando esta última filtragem por declives, também poderá filtrar nuvens de pontos irregulares, como por exemplo, aquelas geradas por dispositivos LiDAR (a variável de configuração do espaçamento apenas é utilizada nesta filtragem). Esta impossibilidade está relacionada com o carregamento dos dados da base de dados para uma matriz. Embora saia do âmbito da tese, para filtrar nuvens de pontos irregulares utilizando o *AutoDTM*, ou não se faz a filtragem por declives, ou, antes dessa filtragem, externamente, interpola-se esses pontos, por forma a gerar uma nuvem regular.

3.2.6 Considerações finais sobre o processamento

Numa fase inicial, foi ponderada a hipótese de paralelização das filtragens de vegetação, árvores e edifícios, mas, mais tarde, verificou-se que a ordem de execução é relevante, produzindo, inclusivamente, resultados diferentes.

Como todo o processamento é demorado, obter-se-ão vantagens se se efectuar paralelização, mas esta optimização terá de ser feita em cada uma das filtragens. Em alternativa, dividir a nuvem de pontos primária em várias nuvens mais pequenas, e processá-las em separado, também optimiza consideravelmente a execução.

Além disso, por também se ter identificado que a detecção de árvores poderia gerar áreas bastante superiores a 150 m x 150 m (que aumentaria a probabilidade de erro, ao efectuar a interpolação com pontos de apoio muito distantes), a própria detecção de árvores foi dividida em duas fases (uma constituída pelo treino, avaliação, identificação e catalogação, e a segunda pela interpolação), para permitir que se efectue a catalogação e a interpolação em alturas diferentes.

Ordem de execução das filtragens

Na ordem de execução das filtragens, deve-se ter em mente que filtragens antecedentes disponibilizam às subsequentes pontos filtrados, no terreno, ou seja, estas últimas terão à disposição mais pontos de apoio, permitindo que efectuem uma melhor filtragem.

Assim, primeiro executa-se a filtragem de vegetação de áreas superiores a 150 m x 150 m, porque os pontos a filtrar não necessitam de qualquer informação dos pontos vizinhos (nomeadamente, para fazer interpolação).

De seguida, pode-se efectuar a filtragem de pequenos grupos de árvores, ou a filtragem de edifícios, cuja ordenação é controlada pelo ficheiro de configurações.

Finaliza-se com a filtragem de *outliers*, porque a intenção é detectar e corrigir eventuais anomalias existentes em todos os pontos, inclusivamente os já filtrados.

Catalogação de pontos como *isVegetation*, na detecção de árvores, se as áreas forem superiores a 150 m x 150 m

Durante a execução de testes, verificou-se que o algoritmo de detecção de árvores poderia gerar áreas superiores a 150 m x 150 m. Nalguns casos particulares onde esta situação ocorreu, a interpolação de pontos gerou maior erro, em virtude de se utilizarem pontos de apoio bastante distantes uns dos outros.

Assim, seguindo o mesmo raciocínio da filtragem de áreas grandes de vegetação (áreas superiores a 150 m x 150 m poderão ocultar configurações de terreno particulares), o *AutoDTM* disponibiliza a hipótese de, durante a detecção de árvores, os pontos intersectados serem catalogados como *isTree* ou *isVegetation*, consoante a intersecção seja feita por objectos com áreas inferiores ou superiores a 150 m x 150 m, respectivamente. Esta área é calculada, durante a execução, através da função do *PostGIS* "ST_Area".

O descrito no parágrafo anterior foi a motivação para se dividir a detecção de árvores, como já referido, em dois momentos: treino, avaliação, identificação e catalogação, e interpolação. Desta forma, se tiver sido definido, no ficheiro de configurações, que se devem catalogar pontos intersectados por áreas superiores a 150 m x 150 m como *isVegetation*, na detecção de árvores, a primeira fase desta detecção é executada após o pré-processamento, para que a filtragem de vegetação inclua esses pontos. Já a segunda fase, a interpolação, ocorre na ordem definida no ficheiro de configurações (antes ou depois da filtragem de edifícios).

Dos testes efectuados, concluiu-se que a catalogação de pontos como *isTree* ou *isVegetation*, na detecção de árvores, produz os melhores resultados, embora, em determinadas zonas de terreno específicas, possa fazer aumentar o erro (por exemplo, em zonas em que haja áreas de vegetação que sejam pouco maiores que os 150 m x 150 m).

Outras optimizações

Nesta subsecção serão listadas outras modificações que se poderão fazer ao *AutoDTM* para, eventualmente, optimizar a sua execução. Como estas alternativas não foram suficientemente testadas, ou sequer implementadas, não foi possível tirar conclusões sólidas:

- geração de outros índices aquando da inserção da informação no PostgreSQL;
- optimização das queries feitas à base de dados;
- execução das queries das filtragens directamente na base de dados, nem que seja pela criação de funções auxiliares, evitando o passo intermédio da transformação dos objectos da base de dados em objectos Java (com excepção da filtragem de *outliers*, devido às vantagens de trabalhar tamanha quantidade de dados em matriz);
- optimização do código propriamente dito: há métodos que são repetidos "desnecessariamente", por opção de implementação, para fasear e delimitar melhor as filtragens.

4 Avaliação dos resultados

Este capítulo contém a avaliação dos resultados produzidos pelo *AutoDTM*. A primeira secção descreve brevemente os ficheiros de *output* que a aplicação pode produzir, nomeadamente o ficheiro de *log*. A segunda secção detalha o comportamento do *adaboost*, uma das partes fundamentais da dissertação, que permitiu refinar o classificador para se poder executar os testes completos descritos na secção seguinte. A última secção avalia os resultados, obtidos pelo *AutoDTM*, no processo de filtragem, completo, de duas nuvens de pontos. Com a primeira nuvem de pontos pretendeu-se concluir quais as melhores variáveis de configuração e, com a segunda, fazer toda a execução em *pipeline*, para se analisar o comportamento numa situação o mais semelhante possível à eventual integração na cadeia de produção do IGeoE.

A avaliação dos resultados, na segunda nuvem de pontos, e conforme já referido, foi feita pelo cálculo da RMSE e do LMAS (regulamentados em [44] e [45], respectivamente). Esta avaliação foi executada como se de um produto do IGeoE se tratasse, tendo sido utilizada, inclusive, a mesma ferramenta de avaliação (folha de Excel disponibilizada pela NATO, para o cálculo do LMAS, ajustada para também calcular o RMSE).

Os testes de "produção" foram executados em duas máquinas diferentes (a primeira, numa máquina física, a segunda, numa virtual; em termos de *software*, a diferença residiu na instalação de versões a 32 ou 64 bits):

- Máquina 1:
  - Windows 7, 32 bits
  - Processador Intel Pentium Dual Core
  - 2GB RAM
- Máquina 2
  - Windows 7, 64 bits
  - Processador Intel Xeon 2.4GHz, 6 cores
  - 8GB RAM

A máquina 2 foi utilizada apenas para o treino do classificador, devido às restrições de RAM já referidas anteriormente.

4.1 Ficheiro de *log*

O ficheiro de *log*, contendo a informação da execução, pode ir de poucos *bytes* até *gigabytes*. Poderão também ser produzidos outros *outputs* na execução do *AutoDTM*, nomeadamente ficheiros de imagem gerados durante a detecção de árvores. A quantidade de informação é gerida no ficheiro de configurações, ajustando as variáveis finais relativas a *debug*. A configuração conforme representada na secção 7.1, cria um ficheiro de log de cerca de 1MB, e conterá informação suficiente para verificar o correcto funcionamento da execução. Os testes utilizaram as definições de *debugging* contidas no referido anexo, já que, se fosse com mais detalhe, atrasaria consideravelmente a execução.

4.2 Classificador

O treino do classificador é um dos aspectos sensíveis da aplicação, e de difícil ajuste. Se, por um lado, os conjuntos de teste e treino devem ser constituídos aleatoriamente a partir do conjunto de amostras total, por outro lado, dada a natureza deste problema específico, é fundamental que o classificador treine aspectos característicos do terreno (árvores, estradas, casas, piscinas, relvados,

etc). Este é, também, aparentemente, o raciocínio aplicado em [23]: utilizou-se 1% das imagens totais para treino, ou seja, treinaram-se fotos inteiras, com contexto (100 *tiles* de 512x512 pixeis).

Para o *AutoDTM* treinou-se com ortofotos inteiras, com conjunto aleatório de amostras, e com pequenas imagens criteriosamente escolhidas, sendo esta terceira hipótese a que deu melhores resultados. O anexo 7.2 contém o conjunto da melhor combinação de imagens encontrada, adequada, pelo menos, ao terreno da zona de Mafra. Apenas se fizeram testes a 6 ortofotos, nesta região, avaliadas por inspecção visual, mas tudo indica que terá o mesmo desempenho em áreas mais vastas de Portugal.

O treino e teste do classificador foram sendo refinados ao longo do trabalho, até se ter atingido o valor de precisão anunciado em [23] (valores superiores a 90%). Esta precisão foi medida de duas formas:

- *Output* do *AutoDTM*
  Fornecendo as imagens de treino e de teste ao *AutoDTM*, este produz no ficheiro de log o resultado do teste, com a imagem de treino fornecida, onde o *output* relativo ao classificador é gerado pela biblioteca *Weka*. Estes resultados, no entanto, são subvalorizados, visto não contemplarem a acção do algoritmo de refinamento.
- *Output* da comparação com máscara manual
  Para avaliar a precisão final do algoritmo, bem como a do algoritmo de refinamento, foi necessário fazer testes diferentes. Assim, para as imagens de teste criaram-se máscaras, manualmente, do que era vegetação. A precisão foi, então, calculada por comparação, pixel a pixel, entre a máscara manual e a produzida pelo *AutoDTM*.

Tabela 4.1 - Precisões obtidas durante o teste do classificador

Inputs	**Precisão do *AutoDTM/Weka* (sem refinamento)**	**Precisão da comparação com máscara manual (sem refinamento)**	**Precisão da comparação com máscara manual (com refinamento)**
Imagem de teste de 512x512; classificador treinado com 10 imagens (imagens no anexo 7.2)	85,8 %	---	---
Imagem de teste de 2627x2337 (primeira imagem do anexo 7.3); classificador treinado com 1 imagem (primeira imagem no anexo 7.2)	---	88,12 %	90,82 %
Imagem de teste de 2627x2337; classificador treinado com 10 imagens (anexo 7.2)	---	90,92 %	90,99 %

Ainda sobre o *Adaboost*, é importante verificar a influência que os 27 atributos têm no classificador, e comparar com o referido em [23] (esta influência foi calculada iterando sobre os 200 classificadores fracos, somando os valores de margem de cada um por atributo).

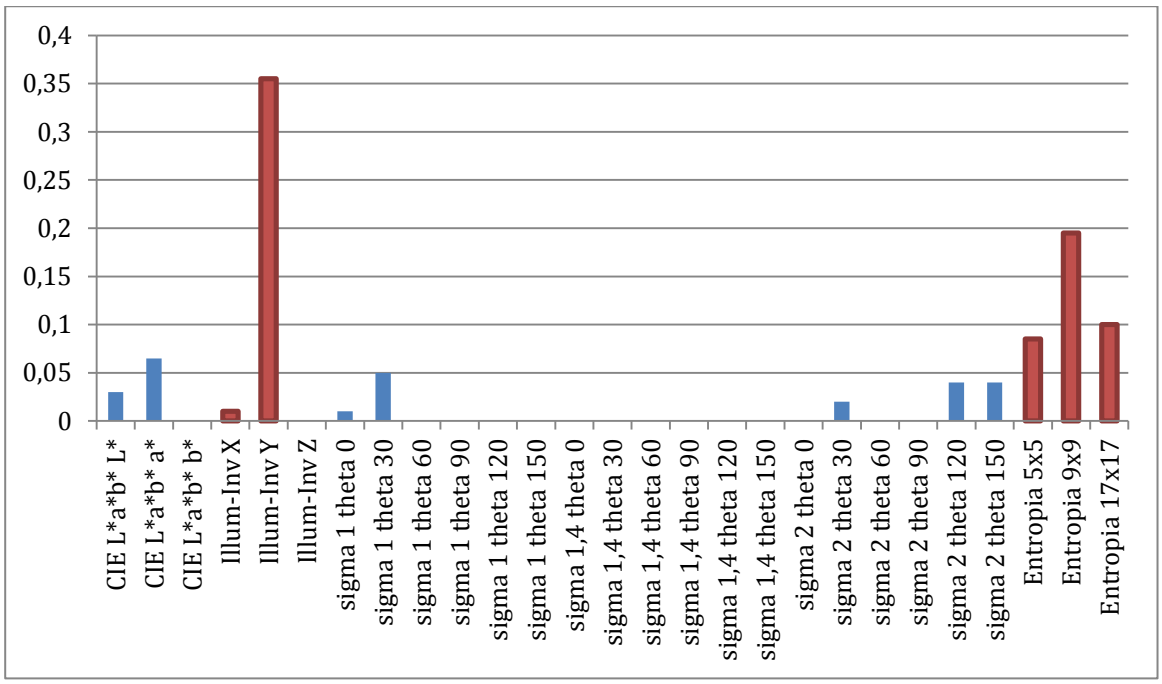


Figura 4.1- Influência dos atributos no classificador

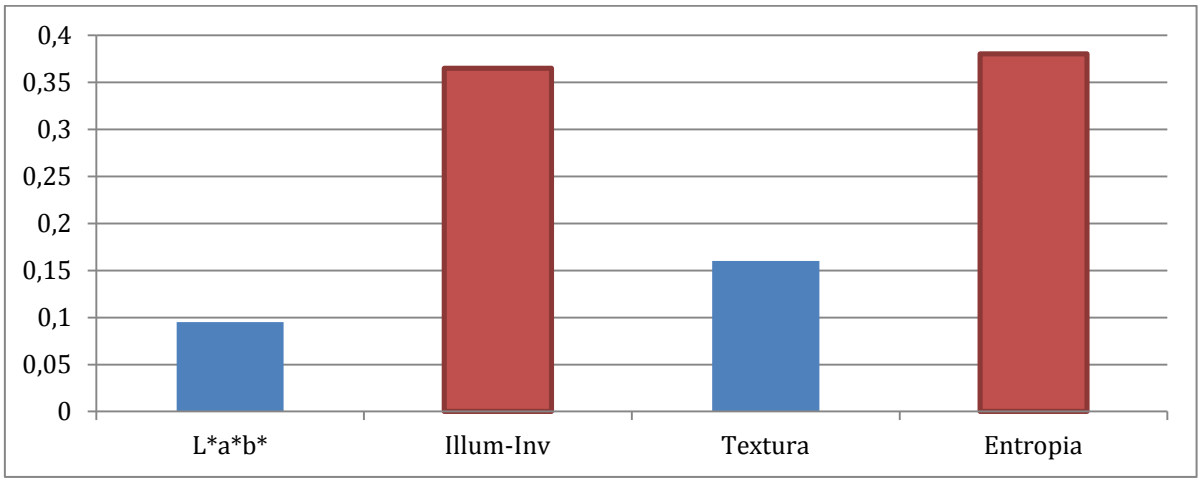


Figura 4.2 - Influência dos atributos, agrupados, no classificador

Em [23], todos os atributos contribuíram para o classificador forte, ao contrário deste caso, em que apenas contribuíram 12 (principalmente a textura que, dos 18, apenas 5 tiveram influência diferente de 0). Factores que terão contribuído para esta diferença:

- utilização de filtros de Gabor, em vez de Gauss, no cálculo da textura;
- forma de cálculo da entropia: [23] sugere uma forma simplificada, enquanto que o *AutoDTM* calcula-a, analisando exactamente as janelas de 5x5, 9x9 e 17x17 pixeis;
- as imagens utilizadas em [23] são fotografias aéreas com o tamanho do pixel variável e entre 0,5 m e 1 m, enquanto o *AutoDTM* utilizou ortofotos, e com tamanho do píxel de 0,5 m;
- a quantidade de amostras de treino utilizadas em [23] foi cerca de 26 vezes superior ao utilizado com o *AutoDTM* (26 214 400 e 955 668 amostras, respectivamente).

Considerando que há atributos com influência a 0, que não contribuem para a classificação, estes poderão ser removidos. Como referido em 2.4.2 e na Tabela 4.2, a variação de atributos no classificador provoca alterações espaciais e temporais exponenciais, e portanto, em termos de eficiência, o *AutoDTM* teria melhor *performance*. No entanto, estes resultados não foram constantes:

alguns dos valores da textura que, neste exemplo, tiveram peso nulo, contribuíram com algum peso noutros treinos. Por não ter havido oportunidade para se verificar se algum dos atributos se matinha sempre nulo, optou-se por manter os 27 propostos inicialmente. Pelo mesmo motivo, também não foram acrescentados outros atributos que, eventualmente, melhorariam a eficácia do *AutoDTM*. Alguns exemplos de atributos que poderiam contribuir para tal, e cuja obtenção e cálculo são de possível realização no IGeoE, são o valor do pixel de imagens em infravermelho, e a cota.

Seguidamente, enumeram-se os aspectos a considerar, aquando da constituição do conjunto de treino, que se concluiu após todos os testes efectuados:

- as amostras devem ser obtidas a partir de imagens completas, mesmo que bastante pequenas (utilizou-se, por exemplo, uma imagem com resolução de 144x131), em vez de serem obtidas, aleatoriamente, a partir de um grande conjunto;
- as imagens devem conter diferentes tipos de características do terreno: o *Adaboost* precisa aprender o que é árvore, mas também o que não é árvore. Portanto, deverão fazer parte deste conjunto não só imagens das árvores predominantes da zona, mas também imagens de piscina, edifícios de telhados de cores diferentes, estrada de alcatrão, estrada de terra batida, pontes, rios, etc;
- a detecção de árvores, no *AutoDTM,* está enquadrada num fluxo de execução que envolve outras operações e filtragens. Assim, dado que se irão executar interpolações em fases subsequentes, pode ser proveitosa a geração de falsos positivos:
    - há maior probabilidade de todas as árvores serem capturadas;
    - aos pontos sobre edifícios ou vegetação densa que tiverem sido erradamente capturados, poderá acontecer uma de duas situações: ou já foram interpolados, e, neste caso, não sofrem alteração, ou serão interpolados, tal como deveriam (embora numa filtragem indevida);
    - se um ponto no terreno for identificado como ponto de árvore, o que acontecerá é esse ponto ser interpolado com pontos circundantes pertencentes ao terreno, gerando alterações desprezáveis.

  Isto significa que, um treino menos rigoroso, poderá, eventualmente, produzir os mesmos ou até melhores resultados (a Tabela 4.1 mostra precisões semelhantes num teste executado com um conjunto de 260 000 amostras e noutro quase 5 vezes superior);
- o número de amostras positivas será semelhante ao número de amostras negativas (sensivelmente 50%).

A Tabela 4.2 mostra o tempo despendido no treino e teste do classificador *Adaboost*, numa das suas execuções (aquela que produziu melhores resultados, utilizando várias imagens que totalizavam 955 668 pixeis). Os valores, em si, servem para se ter uma estimativa da execução geral, mas a informação mais relevante é que o treino e teste do classificador pode demorar quase tanto como uma execução completa do *AutoDTM*.

Tabela 4.2 – Tempos de execução do treino e teste do *Adaboost*

Fase	Tempo despendido (aproximado)
Treino e teste do classificador, sem *cross-validation*	1 h 30 m
Treino e teste do classificador, com *cross-validation*	8 h
Execução completa do *AutoDTM* (pré-processamento, filtragem de vegetação, filtragem de árvores – excepto treino do classificador –, filtragem de edifícios, filtragem por declives)	9 h

Avaliou-se a média dos tempos de processamento das diferentes fases, resumidos na Tabela 4.3. Esta apresenta resultados relativos, e o intuito é ter-se noção de quais as fases que consomem mais tempo.

Tabela 4.3 – Tempos relativos de execução das várias filtragens, em ordem decrescente

Fase/filtragem	Tempo relativo de processamento
Filtragem de árvores (excepto avaliação da ortofoto)	56 % (44 %)*
Filtragem de edifícios	20 %
Avaliação da ortofoto	19 %
Pré-processamento	2,9 %
Filtragem da vegetação	2 % (14 %)*
Filtragem por declives	0,1 %

* quando a filtragem de árvores cataloga pontos como *isTree* ou *isVegetation*

Estes tempos relativos foram obtidos com a filtragem de árvores a catalogar todos os pontos como *isTree*. Caso esta filtragem catalogasse como *isTree* ou *isVegetation*, faria alterar a quantidade de pontos a interpolar, quer nessa própria filtragem, quer na filtragem da vegetação (esta variação está representada nos valores da tabela marcados com um asterisco).

As restantes variáveis de configuração também fazem alterar marginalmente estes tempos relativos, mas a ordem de grandeza mantém-se bastante semelhante.

Em nenhuma destas fases está incluído o treino e teste do *Adaboost*, visto estes resultados terem sido obtidos em produção, com o classificador já devidamente treinado.

4.3 Nuvens de pontos

Executaram-se uma série de testes em duas nuvens de pontos diferentes, a primeira das quais tinha cerca de 20% do tamanho da segunda.

Tabela 4.4 - Características das nuvens de pontos testadas

Nuvens de pontos testadas	Características
Extracto da nuvem de pontos das fotografias 1021000240 e 1021000241 (NP 40-41)	• Quantidade de pontos: 36 427 • Área: 1,5 x 2,5 km
Nuvem de pontos das fotografias 1021000223 e 1021000224 (NP 23-24)	• Quantidade de pontos: 184 869 • Área: 7 x 3 km

Ambas tiveram, em comum, os seguintes *inputs*:

- Área do vector da edição de trabalho: 16 x 10 km.
- Quantidade de objectos no vector da edição de trabalho: 24 877.
- Área do vector (v3d, só curvas de nível e pontos de cota) da edição anterior: 16 x 10 km.
- Quantidade de objectos no vector da edição anterior: 4 060.

Diferiram, no entanto, nas ortofotos utilizadas:

- NP 40-41: uma ortofoto (402-I-4), de área 16 x 10 km.

- NP 23-24: três ortofotos (402-III-1, 402-III-2 e 402-IV-2), cada uma com área 16 x 10 km.

4.3.1 Nuvem de pontos NP 40-41 (extracto)

Os testes efectuados na NP 40-41 foram exaustivos, para se conseguir perceber e sistematizar o efeito da alteração das variáveis de controlo do *AutoDTM*.

A validação de nuvem de pontos secundária foi efectuada da seguinte forma: a edição mais recente da Carta 402 está completa, pelo que o respectivo vector de altimetria (v3d) já existe, elaborado pela forma tradicional. Este vector foi convertido em nuvem de pontos, sendo esta nuvem considerada a prova, o produto final a obter. Assim, as nuvens de pontos secundárias produzidas pelo *AutoDTM* são comparadas, ponto a ponto, com esta nuvem, para cálculo do MSE e RMSE. A utilização desta prova e, por conseguinte, deste método de avaliação, não é o ideal, porque acumula alguns erros:

- a nuvem de pontos de prova é produzida a partir das curvas de nível de um vector, curvas essas que já sofreram adoçamento;
- não se conseguiu fazer coincidir as coordenadas X e Y dos pontos da nuvem de pontos de prova com as coordenadas X e Y das nuvens de pontos a avaliar. Este desvio anda na ordem dos 4 metros.

No entanto, permite validar todos os pontos da nuvem de pontos, ao contrário de, por exemplo, uma validação em estereoscopia. Assim, pesando as vantagens e desvantagens, assumiu-se esta forma de validação.

Validou-se, desde logo, a nuvem de pontos primária com a prova, para se registar o erro inicial e conseguir perceber a evolução das filtragens, tendo-se obtido, aproximadamente, 6,029 metros de RMSE. A segunda validação a ser executada foi entre a altimetria da edição anterior e da prova, que resultou num RMSE de, aproximadamente, 3,374. Não se aproxima mais de 0 devido aos dois pontos realçados anteriormente, bem como à evolução do *software* e *hardware* utilizados na restituição e, obviamente, às alterações do terreno durante o tempo que mediou entre as última duas elaborações da Carta. A utilidade deste valor é permitir ter noção de um resultado final a obter e que, assim, deverá rondar os 3 metros de RMSE.

As variáveis de controlo testadas e que influenciam a execução do *AutoDTM* foram as seguintes:

1. *Buffer* ao redor dos edifícios.
   - Variável: *polygonBufferSizeToAttenuateDesynchronizationBetweenPolygonsAndCloudPoints.*
   - Influencia a filtragem de edifícios e de *outliers*.
   - Valores utilizados: 0, 10 e 20.
2. Declive.
   - Variável: *slopeThreshold.*
   - Influencia a filtragem de *outliers*.
   - Valores utilizados: 15, 20, 30
3. Número de pontos de apoio na interpolação da filtragem de *outliers*.
   - Variável: *outlierInterpolationUsingLowest6Neighbours*.
   - Influencia a filtragem de *outliers*.
   - Valores utilizados: 6 ou 8.
4. Catalogação de pontos durante a detecção de árvores.
   - Variável: *defineTreeAreasBiggerThan150m2AsVegetation*.
   - Influencia as filtragens de vegetação e detecção de árvores.
   - Valores utilizados: catalogação de pontos em áreas superiores a 150 m x 150 m como *isVegetation*, ou todos como *isTree*.
5. Ordem das filtragens de árvores e de edifícios.

- Variável: *treeFilteringFirstThanBuildingFiltering*.
- Influencia as filtragens de edifícios e de detecção de árvores.
- Valores utilizados: execução, em primeiro lugar, da filtragem da detecção de árvores e, depois, a da detecção de árvores (*Vegetation*, *Trees*, *Buildings*, *Outliers*, VTBO), ou vice-versa (VBTO).

Foram executados 72 testes (combinação das variáveis supra), que permitiram tirar algumas conclusões. Convém salientar que, conforme espelhado na Tabela 4.5, na NP 40-41, a área de vegetação e árvores é quase o dobro da área de edifícios. Na análise de resultados far-se-á extrapolação para terreno cuja área de edifícios seja superior à da vegetação, embora não tenham sido feitos testes com esta configuração do terreno.

Tabela 4.5- Quantidade de pontos sobre vegetação, árvores e edifícios

	Detecção de árvores: todos pontos catalogados como *isTree*	
	VTBO	**VBTO**
Vegetação	654	654
Árvores	8988	8978
Edifícios	5195	5205
	Detecção de árvores: pontos catalogados como *isTree* ou *isVegetation*	
Vegetação	9398	9398
Árvores	244	243
Edifícios	5195	5196

Em virtude das diferentes fontes de informação, e da ordem pela qual as filtragens são executadas, há valores apresentados na Tabela 4.5 que podem parecer incoerentes. Por exemplo, a diferença de pontos sobre edifícios deve-se ao facto da detecção de árvores ter criado falsos positivos, e da própria filtragem de edifícios ser executada antes, ou depois, da das árvores. Assim, como cada ponto poderá ser catalogado com mais do que uma etiqueta, é filtrado pela primeira filtragem que se execute.

As figuras que se seguem analisam o comportamento das 5 variáveis de controlo acima enumeradas. Nessas 5 figuras, o eixo das ordenadas representa o RMSE, enquanto o das abcissas representa os testes executados (cada vértice das linhas de cor corresponde a um dos testes).

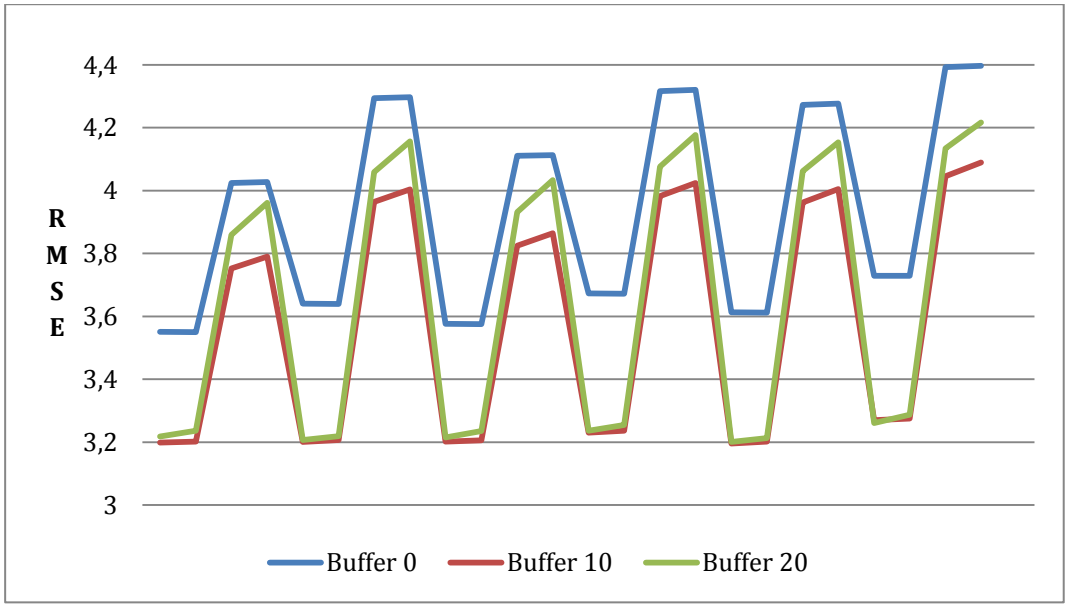


Figura 4.3 - Variação do RMSE com diferentes valores de *buffer* (0, 10 e 20 m)

A Figura 4.3 mostra o comportamento da variação do tamanho do *buffer*, revelando que, com 0 metros, gera bastantes erros, e que o melhor valor é, em média, 10 metros.

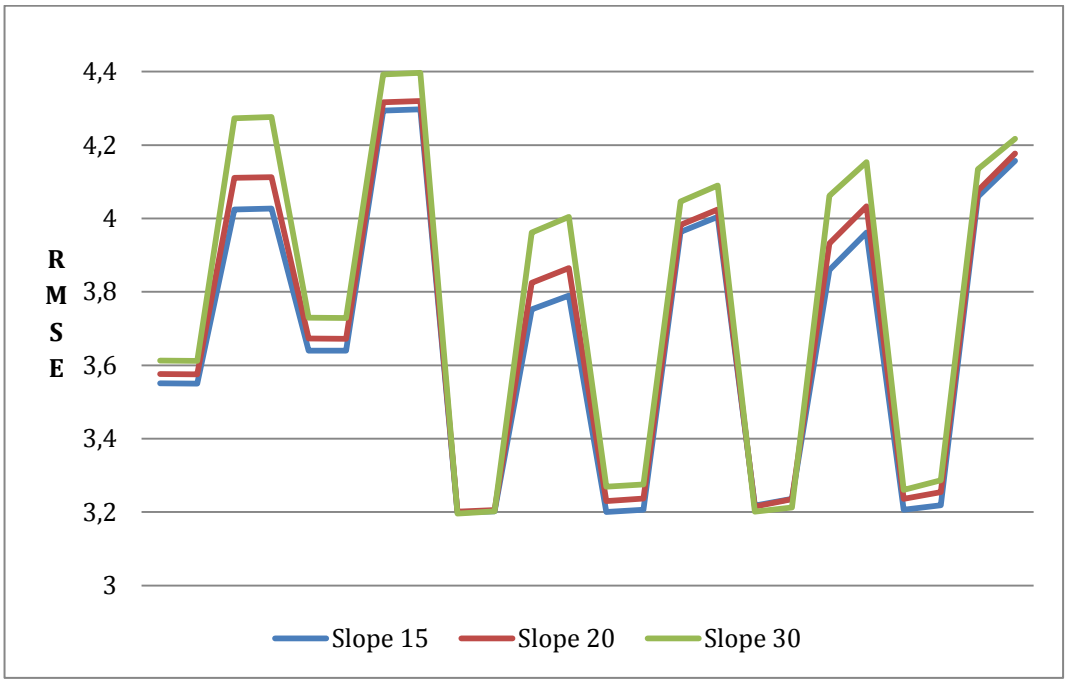


Figura 4.4 - Variação da RMSE com diferentes valores de declive (15, 20 e 30º)

A análise do declive, na Figura 4.4, não permite retirar conclusões muito óbvias, embora a maioria dos mínimos seja quando há declives de 15 ou de 30 metros.

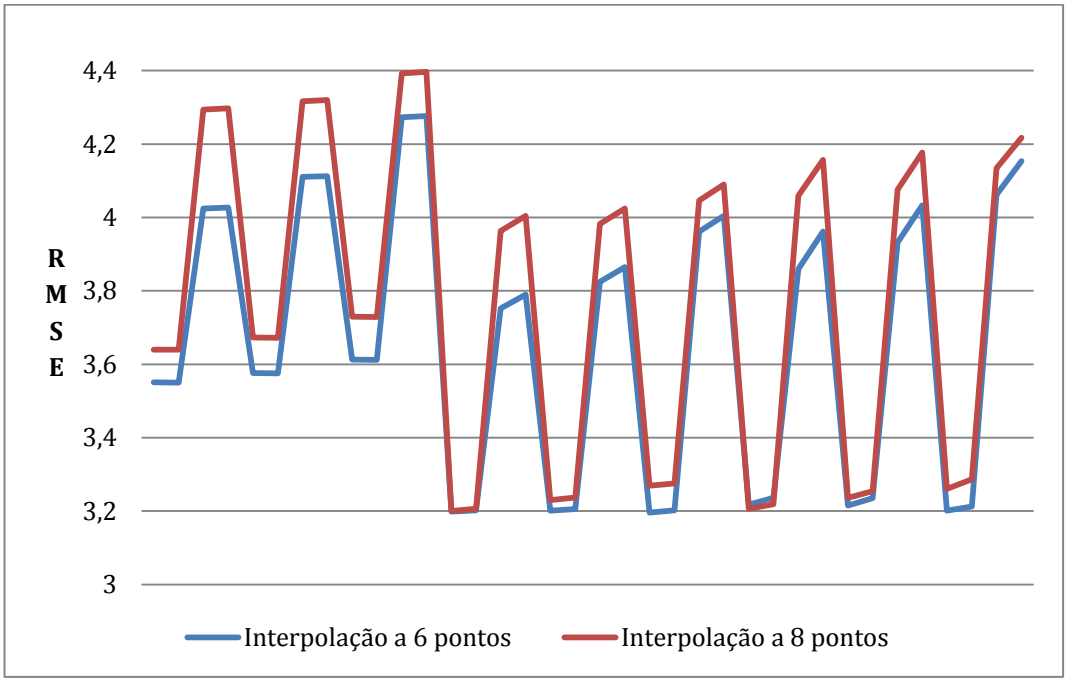


Figura 4.5 - Variação da RMSE com diferentes valores de pontos de apoio utilizados para interpolação (6 e 8)

A Figura 4.5 revela que há tendência para diminuição do RMSE quando a interpolação de *outliers* é executada a 6 pontos. Ainda assim, não é totalmente esclarecedora quanto aos mínimos absolutos.

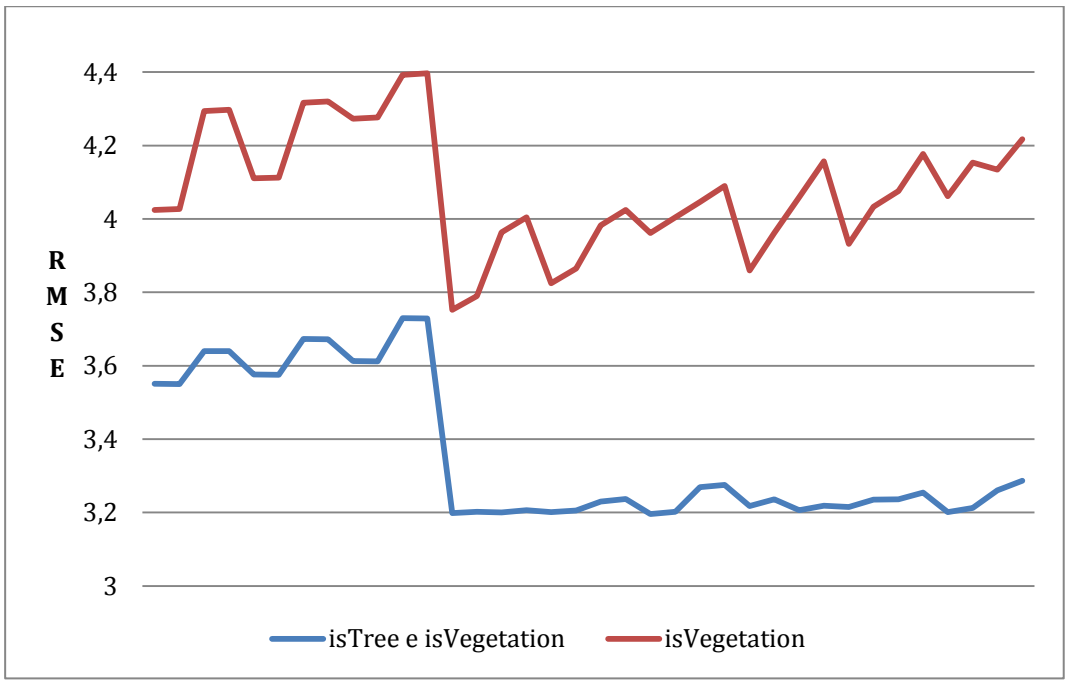


Figura 4.6 - Variação da RMSE com diferentes valores na interpolação de árvores (catalogação de pontos como *isTree* e *isVegetation*, ou apenas com *isTree*)

A catalogação de pontos na detecção de árvores como *isTree* e *isVegetation* apresenta vantagens claras, conforme representado na Figura 4.6. Esta figura permite, também, confirmar a ideia de que, onde há garantia que o terreno não se alterou, é preferível manter o que existe do antecedente (como há mais pontos catalogados como *isVegetation*, estes são substituídos pelas cotas da altimetria de edição anterior).

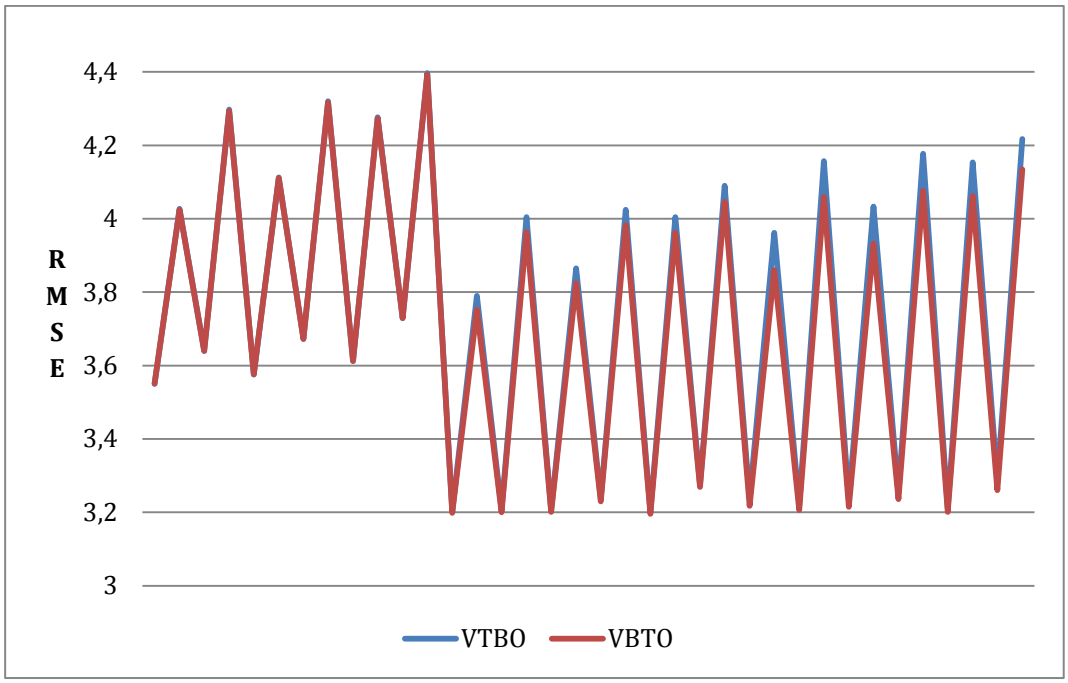


Figura 4.7 - Variação da RMSE com diferentes valores na ordem de interpolação (VTBO e VBTO)

A evolução última variável de controlo, a ordem das filtragens, está representada na Figura 4.7, não sendo possível, no entanto, tirar grandes conclusões.

Em suma, a análise individual das variáveis fez realçar um *buffer* de 10 metros, declive de 15 ou 30 graus e interpolação dos *outliers* utilizando 6 pontos, como sendo as que mais contribuem para a redução do RMSE.

Para se tentar concluir quais os melhores valores para a catalogação de pontos na detecção de árvores, bem como a ordem de execução das filtragens, foi necessário efectuar outras análises. Assim, passou-se a foco para os 24 melhores testes, correspondentes aos mínimos das figuras supra. A Tabela 4.6 lista-os em ordem crescente de RMSE, onde as células a cinzento representam as variáveis utilizadas nesses testes (não se colocaram colunas para *buffer* 0 e catalogação de pontos, porque seriam colunas totalmente a branco ou a preto). Pela observação desta tabela, consegue-se também deduzir que a execução da filtragem de árvores antes da de edifícios (VTBO), produz melhores resultados.

Tabela 4.6 - RMSE dos 24 melhores testes, e variáveis a analisar

RMSE	VBTO	VTBO	*Buffer 10*	*Buffer 20*	*Slope 15*	*Slope 20*	*Slope 30*	Int. 6	Int. 8
3,195673		■	■				■	■	
3,198404		■	■		■			■	
3,200701		■	■		■				■
3,201118		■		■			■	■	
3,201579		■	■			■		■	
3,201776	■		■		■			■	
3,201872	■		■				■	■	
3,205677	■		■			■		■	
3,206572		■		■	■				■
3,206611	■		■		■				■
3,212916	■			■			■	■	
3,21501		■		■		■		■	
3,218104		■		■	■			■	
3,218779	■			■	■				■
3,230151		■	■			■			■
3,235472	■			■		■		■	
3,235965		■		■		■			■
3,235975	■			■	■			■	
3,236583	■		■			■			■
3,254123	■			■		■			■
3,260409		■		■			■		■
3,269513		■	■				■		■
3,275267	■		■				■		■
3,286881	■			■			■		■
SOMA	12	12	12	12	8	8	8	12	12

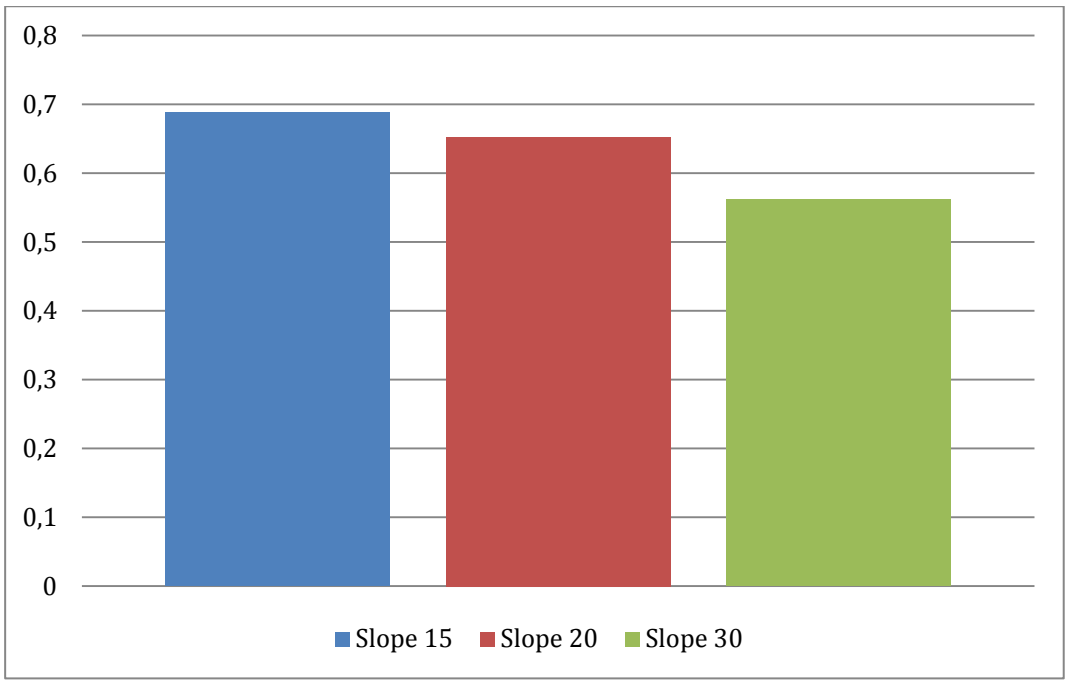


Figura 4.8 - Média da redução da RMSE, na filtragem de *outliers*, utilizando diferentes declives

A Figura 4.8 traz outra perspectiva quanto ao valor do declive: quanto mais baixo, maior a redução da RMSE (embora contrarie o melhor teste da Tabela 4.6). A observação desta figura, no entanto, levanta outro problema que não se teve oportunidade de verificar: qual o valor que maximiza a redução de RMSE.

Uma das validações que também se fez para permitir verificar o desempenho das filtragens de vegetação e edifícios, foi verificar a média do RMSE de ambas, em todos os 72 testes, nas seguintes circunstâncias: para a filtragem de vegetação, apenas se considerou os testes cuja detecção de árvores não catalogou pontos como *isVegetation*, e para a filtragem de edifícios, apenas se considerou os testes cujo *buffer* tivesse valor de 10 metros. Pretendia-se que estes valores fossem na ordem dos 3 metros, o que se conseguiu obter, conforme se pode observar na Tabela 4.7.

Tabela 4.7 - RMSE médio nas filtragens de vegetação e edifícios com as melhores variáveis de configuração

	Filtragem de vegetação	**Filtragem de edifícios**
Média de RMSE	1,1636801	2,9164372

Outro pormenor que esta tabela revela é o reduzido erro na filtragem da vegetação. Tal como já referido anteriormente, quando se vai buscar informação do antecedente, o erro é mais reduzido. É relevante é garantir que o terreno, nessas zonas, não sofreu alterações.

Para concluir a avaliação do *AutoDTM* sobre a NP 40-41, resume-se, de seguida, quais as configurações que, estatisticamente, produziram melhores resultados:

- *buffer* de 10 metros;
- declive de 15 graus (embora a melhor execução na Tabela 4.6 tenha sido com declive de 30 graus);
- interpolação a 6 pontos;
- filtragem de detecção de árvores a catalogar pontos como *isTree* e *isVegetation*;
- filtragem de árvores isoladas e pequenos grupos de árvores primeiro que a de edifícios (VTBO).

4.3.2 Nuvem de pontos NP 23-24

Os testes efectuados na NP 23-24 foram selectivos, utilizando o conhecimento adquirido com os resultados da NP 40-41. Essas execuções assemelharam-se ao que eventualmente poderá ser o desempenho do *AutoDTM*, se eventualmente for integrado na cadeia de produção do IGeoE. Assim, utilizaram-se combinações referidas na subsecção anterior, sintetizadas na Tabela 4.8.

Tabela 4.8 - Parâmetros utilizados para validação da NP 23-24

	Buffer	**Declive**	**Interpolação**	**Catalogação de pontos na detecção de árvores**	**Ordem das filtragens**
1ª execução	10	15	6	*isTree* e *isVegetation*	VTBO
2ª execução	10	30	6	*isTree* e *isVegetation*	VTBO

A validação da NP 23-24 foi efectuada por um operador em estereoscopia, por amostragem, o que significa que os erros calculados têm o máximo de precisão possível (não considerando realizar-se medições no campo). O inconveniente deste método é que apenas permite validar uma quantidade muito reduzida de pontos: neste caso, foram validados aproximadamente 0,1 % da NP 23-24, ou seja, 180 pontos, escolhidos aleatoriamente (esquema na Figura 4.9).

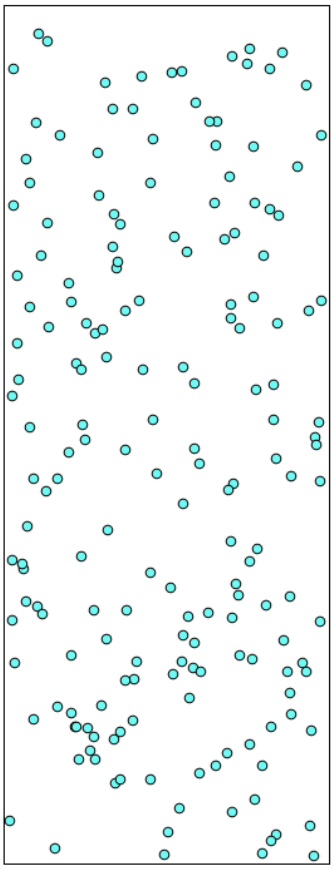


Figura 4.9 - Pontos, escolhidos aleatoriamente, utilizados para validação da NP 23-24

Embora esta quantidade de pontos seja, aparentemente, reduzida, cumpre com as normas definidas em [45].

A Tabela 4.9 resume o RMSE obtido nesta nuvem de pontos, utilizando os parâmetros referidos na Tabela 4.8. Para se ter melhor noção da relação da optimização, o RMSE da nuvem de pontos primária com a prova era de 2,1856.

Tabela 4.9 - RMSE obtida na filtragem da NP 23-24 (prova constituída por pontos restituídos em estereoscopia)

	RMSE	**Percentagem de pontos com erro inferior a 3 metros**
1ª execução	1,8281084	92%
2ª execução	1,8622381	92%

Nas mesmas execuções, os valores de LMAS obtidos foram os constantes na Tabela 4.10.

Tabela 4.10 - LMAS obtido na filtragem da NP 23-24 (prova constituída por pontos restituídos em estereoscopia)

	LMAS
1ª execução	2,99859653
2ª execução	3,05616185

Observando as tabelas 4.9 e 4.10, a variação entre ambas execuções não é significativa, pelo que não se torna óbvio qual o melhor valor do declive (se 15 ou 30 graus), embora, o declive de 15 graus tenha produzido os melhores resultados.

O *AutoDTM* não teve desvios superiores a 3 metros em mais de 90% dos pontos, obteve valores de RMSE ligeiramente superiores a 1,8 metros (1,828 m e 1,862 m), e obteve valores de LMAS na ordem dos 3 metros (2,998 m e 3,056 m). Quanto às primeiras normas de avaliação ([44]), não cumpre, por 0,02 metros, o segundo critério de aceitação, mas considerando que essas normas são para cartas à escala 1/10 000, e a escala utilizada pelo *AutoDTM* foi de 1/25 000, acaba por cumprir com todos os critérios. Quanto às segundas normas ([45]), obtém uma classificação de 1, que, embora não sendo o valor máximo, mas devido à grande proximidade (0,5 metros de diferença), abre boas perspectivas futuras.

Para se poder ter uma melhor noção dos resultados obtidos, os valores do controlo de qualidade de altimetria (RMSE), com restituição pelos métodos tradicionais, no IGeoE, dos blocos do Porto, Lisboa, Padrela e Santarém, foram 1,04 m, 1,17 m, 1,21 m e 1,86 m, respectivamente.

5 Conclusões e trabalho futuro

O problema que se pretendeu minimizar, ou eliminar, na presente dissertação, foi a intervenção humana na geração dos DTM. Todas as automatizações experimentadas, até então, eram pouco eficazes, não cumprindo os padrões mínimos exigidos, e portanto careciam sempre de verificação e correcção humanas. Estas correcções permitiram concluir que os erros ocorriam, sobretudo, em áreas de árvores isoladas ou de pequenos conjuntos de árvores, e, na realidade, tornou-se o desafio principal da tese. A opção, estudo e implementação do método de detecção de árvores, ocupou mais de metade do tempo disponível. Quanto às restantes áreas geradoras de discrepância entre DSM e DTM, os edifícios e grandes áreas de arvoredo, foram problemas mais simples de solucionar, visto já ter sido identificado, noutros estudos, que um dos caminhos para a obtenção de bons resultados era pela identificação *a priori* destas áreas, sendo que no IGeoE é realizada esta restituição.

A referência utilizada para a detecção das árvores ([23]) é um estudo relativamente recente, de 2010, que afirmava eficácia de mais de 90%, e cuja execução poderia ser reutilizada quase na sua totalidade (apenas a terceira fase, a localização individual de cada árvore, era supérflua para a presente tese). Assim, para o *AutoDTM* definiu-se como objectivo, para a detecção das árvores, a obtenção de uma eficácia de mais de 90%, tendo-se conseguido atingir este nível. No entanto, durante o desenvolvimento, registaram-se vários factores que, provavelmente, permitirão melhorar esta eficácia, destacados ao longo do capítulo 3.2 por intermédio de caixas de texto com simbologia própria. Por outro lado, considerando que o refinamento melhora marginalmente esta eficácia, e que até pode ser benéfico a criação de falsos positivos, a implementação do refinamento poderá estar apenas a tornar mais complexo o *AutoDTM*. A grande vantagem do refinamento é que, como gera agregados de pixéis conexos e elimina pixéis isolados, diminui a quantidade de polígonos resultantes, simplificando, desta forma, a filtragem.

Acerca das restantes filtragens, nomeadamente a dos edifícios e áreas de arvoredo, atingiu-se parcialmente o objectivo proposto: quanto às normas [45], o valor de LMAS obtido definiu uma classificação de 1, e quanto às normas [44], que definem um máximo admissível de 1,8 m de RMSE, obteve-se um valor ligeiramente superior (1,828 m na melhor execução). No entanto, esse valor máximo foi definido para cartas à escala 1/10 000, sendo, desta forma, um valor rigoroso para cartas à escala 1/25 000 (como no caso do *AutoDTM*).

Além disso, a sua execução é morosa, devido essencialmente à leitura de dados da base de dados, transformação em objectos Java, processamento, e gravação de dados na base de dados. Se, para a detecção de árvores, é necessária a existência de processamento externo à base de dados, para estas filtragens não é o caso: era possível efectuar o processamento executado pelo *AutoDTM* directamente no PostgreSQL, através de, por exemplo, funções.

A filtragem por declive, para detecção e correcção de *outliers*, aplicou a teoria base dos algoritmos de filtragem de pontos, desenvolvida ao longo de anos, e ajustada a este problema específico. Revendo, novamente, a quantidade de pontos das nuvens na Tabela 4.4, e o tempo relativo de execução na Tabela 4.3, verifica-se que o tempo de execução é marginal, quando comparado com as restantes, mas tendo feito reduzir o RMSE, em média, 0,4 m (na NP 40-41 representou 15% da redução do RMSE geral).

O *AutoDTM*, na filtragem da NP 40-41 com as variáveis de configuração optimizadas, obteve uma RMSE de 3,19567 metros. No entanto, não se deve tirar conclusões baseadas neste valor, devido às diversas imprecisões associadas à forma de validação, enumeradas em 4.3. Já a filtragem da NP 23-24 obteve uma RMSE final muito perto de 1,8 metros, em ambas as execuções, com a prova a ter sido obtida em estereoscopia. Embora tendo sido utilizada uma reduzida quantidade de pontos (cerca de 0,1%), estes resultados foram bastante satisfatórios, quase cumprindo com os critérios mais rigorosos de avaliação.

Os pontos a explorar e aperfeiçoar, no futuro, com vista a melhorar o *AutoDTM*, são os seguintes:

- criação de um manual de utilização;
- criação de um conversor de ficheiros DGN para *shapefile;*
- à semelhança da detecção de árvores, em que o *AutoDTM* divide as imagens em *tiles* e processa-as individualmente por questões de falta de memória, fazer a divisão da nuvem de pontos em conjuntos mais pequenos. O objectivo aqui, no entanto, é para permitir paralelização;
- na filtragem de vegetação, assim que existam DTM efectuados com *software* actual, comparar as diferenças de RMSE quando se utilizam estes, ou o vector da edição anterior;
- criar classe/objecto que execute o *adaboost*, permitindo paralelização;
- explorar outras features para a caracterização de pixeis, para a execução da detecção de árvores. Por exemplo, utilização de imagens de infra-vermelhos, ou a cota;
- exportar as máscaras geradas pela detecção de árvores como ficheiros geotiff, contendo a sua georreferenciação nos cabeçalhos, em vez de exportar como TIF ou PNG;
- não utilizar o algoritmo do refinamento da imagem após a detecção de árvores;
- tendo em conta que poderá e deverá haver, no IGeoE, um classificador ajustado a diferentes zonas de terreno (zonas de areal, zonas de vegetação queimada, etc), seria conveniente que o *AutoDTM* , dada uma *pool* de classificadores, conseguisse definir qual o melhor para o terreno que se quer avaliar. Uma possibilidade para o concretizar pode passar por um operador fazer manualmente uma máscara de árvores/não-arvores, numa pequena zona de terreno das novas fotografias aéreas, permitindo que o *AutoDTM* execute tantos testes quantos os classificadores, e escolha o mais adequado;
- utilizar outro método de interpolação, que não por declives, na filtragem de *outliers;*
- verificar qual o melhor declive, considerando que os testados foram de 30, 20 e 15 graus, e que se obteve gradualmente melhores resultados;
- verificar variação de RMSE utilizando nuvens de pontos de espaçamento mais reduzido;
- preparar o *AutoDTM* para filtrar nuvens de pontos irregulares, como é o caso daquelas geradas com informação recolhida por dispositivos LiDAR;
- à partida, as nuvens de pontos produzidas pelo *AutoDTM* serão transformadas numa TIN (DTM), para posterior geração de curvas de nível. Considerando isto, poder-se-á explorar a remoção de pontos, em vez de se efectuar sempre interpolação;
- geração de outros índices aquando da inserção da informação no PostgreSQL;
- optimização das *queries* feitas à base de dados;
- optimização do código propriamente dito: há métodos que são repetidos "desnecessariamente", por opção de implementação, para fasear e delimitar melhor as filtragens;
- efectuar todas as filtragens por *queries* SQL, optimizando a utilização da base de dados e o *AutoDTM*, executando apenas a detecção de árvores e a filtragem de outliers externamente;
- o *AutoDTM* ter sido programado em Java, provou ser um *bottleneck*. A *Java Virtual Machine* limita a utilização da RAM, tendo gerado grandes restrições no tamanho dos conjuntos de amostras na detecção de árvores. Esta limitação, por sua vez, também restringe a utilização das bibliotecas associadas, nomeadamente o OpenCV: a abertura de imagens grandes, como as ortofotos, apenas foi possível em formato PNG (e não no TIF original), e a execução de algumas das funções esgotava a RAM disponível, devido à necessidade de utilização de matrizes temporárias do mesmo tamanho que as originais. Assim, a eventual rescrita noutra linguagem deverá ser considerada. Uma possibilidade é em C++, que também daria completa compatibilidade com o OpenCV. Com outra

linguagem, eventualmente também é necessário ponderar outra biblioteca de tratamento de imagens.

De todos os pontos supra, há alguns que se julga serem prioritários: considerando que o processamento demora muito tempo, dever-se-á focar na paralelização das várias fases do *AutoDTM* (não só a execução do *adaboost*, mas também cada uma das filtragens); a redução do número de *features* por pixel, na detecção de árvores, também reduzirá o processamento espacial e temporalmente; a rescrita noutra linguagem, nomeadamente em C++, permitirá melhor gestão da memória, contribuindo também para uma redução temporal da execução do AutoDTM.

Por fim, os objectivos propostos para a presente dissertação foram atingidos. Quanto à adaptação da metodologia na cadeia de produção do IGeoE, carecerá, com certeza, de mais testes de produção e da implementação de algumas das optimizações propostas, mas tendo já sido trilhado a maior parte do caminho.

6 Bibliografia

[1] P. Redweik, “Fotogrametria aérea.” Departamento de Engenharia Geográfica, Geofísica e Energia da Faculdade de Ciências da Universidade de Lisboa, Lisboa, 2007.

[2] “Instituto Geográfico do Exército.” [Online]. Available: http://www.igeoe.pt.

[3] “Processo da carta 122 (Porto), Protocolo.” IGeoE, Lisboa, 2013.

[4] “Processo da carta 523 (Pias), Protocolo.” IGeoE, Lisboa, 2011.

[5] I. G. do Exército, “Normas técnicas.” Lisboa, 2008.

[6] “Práctica Topografía V - Interpolación Curvas de Nivel.” [Online]. Available: http://pt.scribd.com/doc/118760504/Interpolacion-Curvas-de-Nivel. [Accessed: 28-Aug-2014].

[7] L. Batista, “Aerofotogrametria no contexto do processo cartográfico.” .

[8] G. Sithole and G. Vosselman, “Comparison of filtering algorithms,” in *3-D reconstruction from airborne laserscanner and InSAR data*, 2003.

[9] E. P. Baltsavias, “A comparison between photogrammetry and laser scanning,” *International Archives of Photogrammetry and Remote Sensing*, Zurich, pp. 83–94, 1999.

[10] G. Vosselman, H.-G. Maas, and et. al, “Airborne and terrestrial laser scanning,” in *Airborne and terrestrial laser scanning*, First., G. Vosselman and H.-G. Maas, Eds. Dunbeath, Caithness KW6 6EY, Scotland, UK: Whittles Publishing, 2010, pp. 135–167.

[11] C. Briese, N. Pfeifer, and P. Dorninger, “Applications of the robust interpolation for DTM determination,” *International Archives of Photogrammetry and Remote Sensing*, Graz, Austria, pp. 55–61, 2002.

[12] G. Vosselman, “Cartographic Feature Extraction,” Delft, The Netherlands, 1998.

[13] “ImageStation Automatic Elevation.” Intergraph.

[14] Bentley, “Microstation V8 XM Edition.” 2006.

[15] X. Meng, N. Currit, and K. Zhao, “Ground Filtering Algorithms for Airborne LiDAR Data : A Review of Critical Issues,” *Remote Sens.*, vol. 2, no. 3, pp. 833–860, 2010.

[16] M. Roggero, “Airborne laser scanning: clustering in raw data,” in *International Archives of Photogrammetry and Remote Sensing2*, 2001, pp. 227–232.

[17] G. Vosselman, “Slope based filtering of laser altimetry data,” in *International Archives of Photogrammetry and Remote Sensing2*, 2000, pp. 935–942.

[18] G. Sithole, “Filtering of laser altimetry data using a slope adaptative filter,” in *International Archives of Photogrammetry and Remote Sensing*, 2001, pp. 203–210.

[19] M. Elmqvist, “Ground estimation of laser radar data using active shape models,” Stockholm, 2001.

[20] M. Brovelli, M. Cannata, and U. Longini, “Managing and processing LiDAR data within GRASS,” in *GRASS Users Conference 2002*, 2002.

[21] P. Axelsson, “DEM generation from laser scanning data using adaptative TIN models,” in *International Archives of Photogrammetry and Remote Sensing*, 2000, pp. 110–117.

[22] R. Dias and A. Marques, “Geração automática de curvas de nível para a carta militar, série M888, escala 1:25 000, uma abordagem possível,” *Boletim do Instituto Geográfico do Exército, n^{o}74*, Lisboa, pp. 36–39, Nov-2012.

[23] L. Yang, X. We, E. Praun, and X. Ma, “Tree detection from aerial imagery,” in *CCS ’10: Proceedings of the 17th ACM Conference on Computer and Communications Security*, 2010.

[24] M. Hirschmugl, M. Ofner, J. Raggam, and M. Schardt, “Single tree detection in very high resolution remote sensing data,” vol. 110, pp. 533–544, 2007.

[25] D. G. Leckie, A. Gougeon, N. Walsworth, and D. Paradine, “Stand delineation and composition estimation using semi-automated individual tree crown analysis,” vol. 85, pp. 355–369, 2003.

[26] D. A. Pouliot, D. J. King, F. W. Bell, and D. G. Pitt, “Automated tree crown detection and delineation in high-resolution digital camera imagery of coniferous forest regeneration,” vol. 82, pp. 322–334, 2002.

[27] M. Wulder, K. O. Niemann, and D. G. Goodenough, “Local Maximum Filtering for the Extraction of Tree Locations and Basal Area from High Spatial Resolution Imagery,” vol. 4257, no. 00, 1999.

[28] “Lab color space.” [Online]. Available: http://en.wikipedia.org/wiki/Lab_color_space#CIELAB.

[29] H. Y. Chong, S. J. Gortler, and T. Zickler, “A Perception-based Color Space for Illumination-invariant Image Processing,” *ACM SIGGRAPH 2008 Pap.*, vol. 27, no. 3, 2008.

[30] “Playing with CIE Lab Colors in R.” [Online]. Available: http://vis4.net/blog/posts/playing-with-cie-lab-colors-in-r.

[31] Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,” *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[32] L. Krippahl, “Aprendizagem Automática, aula 4.” Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, Lisboa, 2013.

[33] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, Sep. 2004.

[34] “Geomedia.” Intergraph.

[35] “DGN.” [Online]. Available: http://en.wikipedia.org/wiki/DGN.

[36] “World file.” [Online]. Available: http://en.wikipedia.org/wiki/World_file. [Accessed: 10-Aug-2014].

[37] “ArcGIS.” ESRI, 2013.

[38] “Intergraph Standard File Formats.” [Online]. Available: http://dgnlib.maptools.org/dgn.html.

[39] “GeoTools.” [Online]. Available: http://geotools.org.

[40] IGeoE, “Plano de Actividades, Secção de Edição.” IGeoE, 2014.

[41] “PostgreSQL.” [Online]. Available: http://www.postgresql.org.

[42] “PostGIS.” [Online]. Available: http://postgis.net.

[43] P. M. R. de C. Dias, “Elaboração de uma Carta de Aterros da Cidade de Lisboa por LiDAR e Fotogrametria Aérea,” Universidade de Lisboa, Faculdade de Ciências, 2013.

[44] I. G. Português, “Normas técnicas da cartografia à escala 1:10000.” 2009.

[45] NATO, “STANAG 2215 IGEO - Evaluation of land maps, aeronautical charts and digital topographic data,” vol. 2215, no. Edition 6. 2002.

[46] “QGIS: a free and open source geographic information system.” [Online]. Available: http://www.qgis.org/en/site/. [Accessed: 06-Aug-2014].

[47] “Programming language popularity.” [Online]. Available: http://langpop.com/. [Accessed: 06-Aug-2014].

[48] “PostgreSQL JDBC driver.” [Online]. Available: http://jdbc.postgresql.org/. [Accessed: 08-Aug-2014].

[49] “OpenCV.” [Online]. Available: http://opencv.org/. [Accessed: 06-Aug-2014].

[50] “Weka 3: data mining software in Java.” [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/. [Accessed: 06-Aug-2014].

[51] “JBoost.” [Online]. Available: http://jboost.sourceforge.net/. [Accessed: 06-Aug-2014].

[52] “raster2pgsql.exe.” [Online]. Available: http://postgis.net/docs/using_raster_dataman.html. [Accessed: 06-Aug-2014].

[53] “PostGIS 2.0 Cheatsheet.” [Online]. Available: http://www.postgis.us/downloads/postgis20_cheatsheet.html. [Accessed: 08-Aug-2014].

[54] “Well-known text.” [Online]. Available: http://en.wikipedia.org/wiki/Well-known_text. [Accessed: 10-Aug-2014].

[55] IGeoE, “Instrução de Trabalho IT - SSIG_03 - Criação de Modelo Digital de Terreno (MDT) 1. O.” 2013.

[56] “FAQ about the Java HotSpot VM - 32bits JVM heap.” [Online]. Available: http://www.oracle.com/technetwork/java/hotspotfaq-138619.html#gc_heap_32bit. [Accessed: 11-Aug-2014].

[57] “Gabor filter for image processing and computer vision.” [Online]. Available: http://matlabserver.cs.rug.nl/edgedetectionweb/web/edgedetection_params.html. [Accessed: 11-Aug-2014].

[58] “Matlab gaborkernel code.” [Online]. Available: http://matlabserver.cs.rug.nl/edgedetectionweb/gaborkernel2d.m. [Accessed: 11-Aug-2014].

[59] C. M. Catita, “Análise Espacial da Informação Geográfica - métodos de interpolação determinísticos.” Departamento de Engenharia Geográfica, Geofísica e Energia da Faculdade de Ciências da Universidade de Lisboa, Lisboa, 2011.

[60] C. M. Catita, “Análise Espacial da Informação Geográfica - métodos de interpolação probabilísticos.” Departamento de Engenharia Geográfica, Geofísica e Energia da Faculdade de Ciências da Universidade de Lisboa, Lisboa, 2012.

[61] A. Setianto and T. Triandini, “Comparison of Kriging and Inverse Distance Weighted (IDW) interpolation methods in lineament extraction and analysis,” *J. Southeast Asian Appl. Geol.*, vol. 5, no. 1, pp. 21–29, 2013.

[62] L. Haskell and R. O’Donnell, “Stand staright up: a true ortho perspective on downtown Denver,” 2001. [Online]. Available: http://proceedings.esri.com/library/userconf/proc01/professional/papers/pap622/p622.htm. [Accessed: 02-Sep-2014].

[63] “Minimizing subjectivity in digital orthopho imagery.” [Online]. Available: http://proceedings.esri.com/library/userconf/proc00/professional/papers/pap306/p306.htm.

7 Anexos

7.1 Exemplo de ficheiro de configurações do *AutoDTM*

```
//Auto DTM configuration file

//Execution log file
logFile=C:\\AutoDTM\\outputs\\consoleLog.txt

//Postgresql/PostGIS database connection
host=localhost
port=5432
databaseName=gis
username=geodata
password=geodata

//Table name, in postgres, of the current vector data. Only polygons will be used
currentVectorTableName=402_polygon

//Table name, in postgres, of the previous vector data (altimetry, v3d).
previousV3DVectorTableName=v3d_2007

//Table name, in postgres, of the point cloud. Only points will be used.
inputPointCloudTableName=np_semadocamento_40_41

//Table name, in postgres, where output will be saved
outputPointCloudTableName=point_cloud_final

//Table name, in postgres, where refined prediction raster will be saved. Will only
be created/used if adaboost evaluation is performed. If this table already exists,
eventual polygons detected from Adaboost evaluation will be appended to same table.
This permits to increase such table with different inputs and in different runtimes
outputRefinedPredictionRasterTableName=refinedPredictionRaster

//Point spacing. All work done with 10m spacing...
pointSpacing=10

//Threshold value for outlier filtering execute interpolation, in grades. Slopes
higher than this grade trigger IDW interpolation
slopeThreshold=45

//For executing IDW to interpolate points over polygons, it's needed to figure out
neighbour terrain points. These points will be searched in a buffer around polygon
of polygonBufferSizeForInterpolation size. To attain good results with IDW, it must
```

be used at least one point per quadrant. In the tests done, with a buffer of 10m there are almost 6 points average per quadrant. With 30m, almost 21 points average per quadrant. This value must exist, but should be updated only for efficiency (if a buffer too big, will force AutoDTM to spend more time grabbing all points in such big area, but in truth will only use a maximum of "polygonMaximumPointsPerQuadrantUsedForInterpolation" points - the nearest ones of the point to interpolate)

polygonBufferSizeForInterpolation=100

//If in polygon interpolation there's an absurd number of useful points per quadrant, there's no gain in precision. Therefore a cap should be defined for how many points per quadrant, in maximum, will be used for interpolation.

polygonMaximumPointsPerQuadrantUsedForInterpolation=4

//Due mostly to the automatic image correlation software execution, generated point clouds are not "perfectly" synchronized with polygons, and even with ortophotos. A common workaround is adding a small buffer around each polygon, to counter this effect. This buffer will be added to all polygons (buildings) during the pre-processing phase

polygonBufferSizeToAttenuateDesynchronizationBetweenPolygonsAndCloudPoints=0

//This variable will also contribute to the out-of-synch effect between polygons and point cloud. The idea is simple: the de-synchronization will leave "lines" of high heights. This means that when slope is triggered, some of the neighbour pixels may have incorrect height as well. Considering this, this alternate method will search for the lowest 6 members and will use them to interpolate (if the error is due to a de-synchronization line, it will influentiate 3 out of 9 points, including the center one). With this tag commented, or set to off, will use all 8 neighbours

outlierInterpolationUsingLowest6Neighbours=on

//After tree detection, the points intersected by white areas can all be tagged as "isTree", or to avoid interpolation of big areas (and to follow the vegetation rationale), can tag points as "isTree" if in areas<=150m^2, and as "isVegetation" areas>150m^2

defineTreeAreasBiggerThan150m2AsVegetation=on

//Performing tree filtering or building filtering first may generate different results, as the first filtering will provide more "terrain" points for the second one

treeFilteringFirstThanBuildingFiltering=on

//Path to the ortophoto file (tif) of the cloud point region. Use double slash for directory separator. Can be of any size; autoDTM will create and evaluate tiles no bigger than wekaTileSize x wekaTileSize pixel's size, gathering them in the end

ortoFile=C:\\AutoDTM\\inputs\\402-I-4_2007.png

//Path to the ortophoto world file (tfw), that contains scale, skew and NW corner values - used to georeferenciate output mask

ortoWorldFile=C:\\AutoDTM\\inputs\\402-I-4_2007_uso_interno.tfw

//Path to the output directory. In that directory will be saved output files, such as images or Adaboost algorithm support files. Use double slash for directory separator. Should end with double slash.

outputPath=C:\\AutoDTM\\outputs

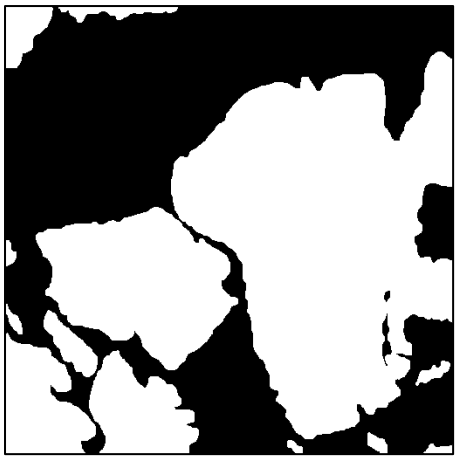
```
//To perform outliers filtering (last filtering) or not
outliersFiltering=on

//To perform buildings filtering or not
buildingFiltering=on

//To perform vegetation filtering or not
vegetationFiltering=on

//To perform vegetation filtering, AutoDTM needs to convert previous v3d vector into TIN. As it's a slow process, and as the resulting TIN layer can support the filtering of multiple point clouds, use this flag to control this layer generation. The name of this layer, in PostgreSQL, is the variable “previousV3DVectorTableName” plus "_TIN"
generateTINFromV3D=on

//An alternative to generate TIN from v3d vector, is to use directly the previous MDT. mdtFile variable should point to an MDT file generated by Arcgis, [name].tif, and in same directory should exist a [name].tif.xml file and a [name].tif.aux.xml filewith MDT details. AutoDTM parses that XML looking after the following values (XPath queries, first items):
//- //nativeExtBox/westBL - upper left corner X coordinate; [name].tif.xml
//- //nativeExtBox/northBL - upper left corner Y coordinate; [name].tif.xml
//- //axisDimension[@type='002']/dimResol/value - pixel size, X direction; [name].tif.xml
//- //axisDimension[@type='001']/dimResol/value - pixel size, Y direction; [name].tif.xml
//- //Histograms/HistItem/HistMin - minimum value of MDT pixel (minimum "real" height); [name].tif.aux.xml
//- //Histograms/HistItem/HistMax - maximum value of MDT pixel (maximum "real" height); [name].tif.aux.xml
//This option is mutually exclusive with TIN generation! If generateTINFromV3D is set to on, AutoDTM will ignore mdtFile variable and will use TIN
mdtFile=C:\\AutoDTM\\inputs\\mdt_corrente\\402.tif

//Flags to control tree detection from aerial imagery. To train and evaluate, both flags should be on. If is intended to evaluate an orto, set treeDetectionTrainAdaboost to off, treeDetectionEvaluateOrto to on, provide a valid input serialized tree and comment output serialized tree. If is intended to just train Adaboost model, just set treeDetectionTrainAdaboost to on. To upload refined prediction image into PostgreSQL, so white areas can be transformed into polygons, set treeDetectionUploadRefinedPredictionToPostgreSQLRaster to on
treeDetectionTrainAdaboost=off
treeDetectionEvaluateOrto=off
treeDetectionUploadRefinedPredictionToPostgreSQLRaster=on
treeDetectionInterpolation=on

//OpenCV library definitions. Path to file without dll extension, relative to executing file directory. In execution, AutoDTM will first check the OS bits, and then load the proper library
openCVLibrary32bits=lib/opencv_java249_32bits
```

```
openCVLibrary64bits=lib/opencv_java249_64bits

//Path to PostgreSQL raster2pgsql.exe file
raster2pgsqlPath=C:\\Program Files\\PostgreSQL\\9.3\\bin\\raster2pgsql.exe

//Path to the ortophotos and mask files to be used for AdaBoost training and testing. Masks should be filled with black and white pixels: black meaning non-tree pixels, white meaning tree pixels. It can be trained with multiple ortophotos. To do so, repeat as many trainOrtoFile/trainMaskFile tags as ortophotos you have (first trainOrtoFile will use first trainMaskFile, and so on). Each orto/mask size should be of about 500x500 pixels; higher sizes can throw “out of memory” exceptions; directly related with features quantity as well
trainOrtoFile=C:\\AutoDTM\\inputs\\train02_01.png
trainMaskFile=C:\\AutoDTM\\inputs\\train02_01Mask.png
trainOrtoFile=C:\\AutoDTM\\inputs\\train02_02.png
trainMaskFile=C:\\AutoDTM\\inputs\\train02_02Mask.png
trainOrtoFile=C:\\AutoDTM\\inputs\\train02_03.png
trainMaskFile=C:\\AutoDTM\\inputs\\train02_03Mask.png
trainOrtoFile=C:\\AutoDTM\\inputs\\train02_04.png
trainMaskFile=C:\\AutoDTM\\inputs\\train02_04Mask.png
trainOrtoFile=C:\\AutoDTM\\inputs\\train02_05.png
trainMaskFile=C:\\AutoDTM\\inputs\\train02_05Mask.png
trainOrtoFile=C:\\AutoDTM\\inputs\\train02_06.png
trainMaskFile=C:\\AutoDTM\\inputs\\train02_06Mask.png
trainOrtoFile=C:\\AutoDTM\\inputs\\train02_07.png
trainMaskFile=C:\\AutoDTM\\inputs\\train02_07Mask.png
trainOrtoFile=C:\\AutoDTM\\inputs\\train02_08.png
trainMaskFile=C:\\AutoDTM\\inputs\\train02_08Mask.png
trainOrtoFile=C:\\AutoDTM\\inputs\\train02_09.png
trainMaskFile=C:\\AutoDTM\\inputs\\train02_09Mask.png
trainOrtoFile=C:\\AutoDTM\\inputs\\train02_10.png
trainMaskFile=C:\\AutoDTM\\inputs\\train02_10Mask.png
testOrtoFile=C:\\AutoDTM\\inputs\\testReal01.tif
testMaskFile=C:\\AutoDTM\\inputs\\testReal01Mask.tif

//Prefix of output files generated during training/testing/evaluation. Generated files will be saved in outputPath
prefixEvaluatedOutput=evaluated_
prefixTrainOutput=train_
prefixTestOutput=test_
prefixEntropyImageFiles=entropy_
prefixTextureImageFiles=gaborFilter_

//Image file name to save conversion to CIEL*a*b* space color. Must have an image file extension such as PNG or TIF
CIELabImageOutputFile=CIELab.png
```

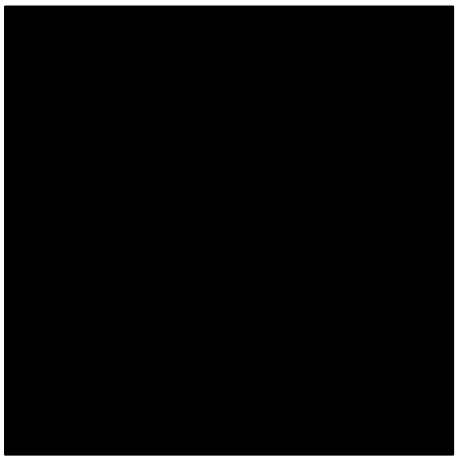
```
//Image file name to save channel L of CIEL*a*b* space color. Must have an image file extension such as PNG or TIF
CIELab_LChannelImageOutputFile=CIELab_LChannel.png

//Image file name to save conversion to illumination-invariant space color. Must have an image file extension such as PNG or TIF
IlluminationInvariantImageOutputFile=IlluminationInvariant.png

//Image file to save Adaboost evaluation result. Must have an image file extension such as PNG or TIF
predictionImageFile=prediction.tif

//Image file to save refined Adaboost evaluation result (after execution of min-cut/max-flow denoising algorythm). Must be a file with extension tif!
refinedPredictionImageFile=refinedPrediction.tif

//Weka (AdaBoost) execution control variables
//number of iterations
wekaIterations=200
//debug mode; comment, if no debug wanted
wekaDebugMode=-D
//"colorize" output: shows non-trees in black, and trees in a scale of gray to white considering the predicted probability. In white, only pixels with 90% probability or above. Just use it for debug purposes, as output needs to be a black or white image, for post processing in Postgre/PostGIS
wekaColorizeOutput=off
//input serialized model to be used for evaluation only
wekaInputSerializedModelFilePath=C:\\AutoDTM\\outputs\\wekaModel.serial.weka
//output serialized/java model only created after training
wekaOutputSerializedModelFile=wekaModel.serial.weka
wekaOutputJavaModelFile=wekaModel.java.weka
wekaPerformCrossValidation=off
//Tile settings.
//In order to evaluate an entire ortophoto, autoDTM is ready to accept big images that will then be cropped into tiles. Each tile will be a rectangle no bigger than tileSize side, plus a buffer of tileBuffer size. That buffer is an overlap over adjacent tiles, and must exist due to the nature of some of the features (for instance, texture evaluates pixel windows of up to 8 adjacent pixels). After evaluation, buffers will be discarded and the tiles, now with tileSize sizes, will be gathered into the final image. If it's not needed to crop input images, simply comment both wekaTileSize and wekaTileBuffer lines.
wekaTileSize=512
wekaTileBuffer=20
//If a big ortophoto still generate out of memory exceptions, set wekaSerializeTiles to on. If on, each evaluated tile is saved in disk and not kept in memory, and will only be loaded when generating the final prediction picture. Obviously, it will be much more time consuming.
wekaSerializeTiles=on

//JVM SYSTEM PROPERTIES; All values should be "on" or "off".
```

```
//General debug for building and tree filterings.
autoDTM.debug=on

//Object listing debug: show detailed info about building or tree polygons being loaded/manipulated from database.
autoDTM.debugObjectsListing=off

//General debug for tree detection.
autoDTM.treeDetection.debug=on

//Save partial/support tree detection images in output directory.
autoDTM.treeDetection.saveResultsInImage=off

//Show detailed info such as the features/samples being generated from the training ortophoto, and the probability of each point belonging to tree or non-tree
autoDTM.treeDetection.debugPointDetails=off
```

7.2 Imagens, e respectivas máscaras, utilizadas no treino do classificador *Adaboost*

512x512

200x200

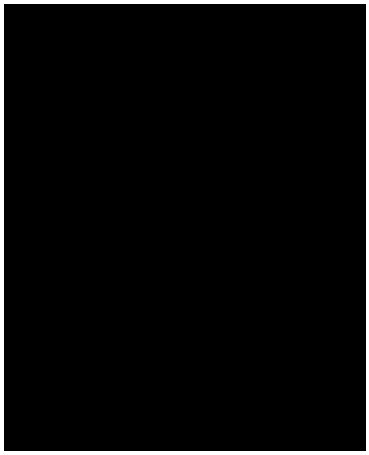

164x301

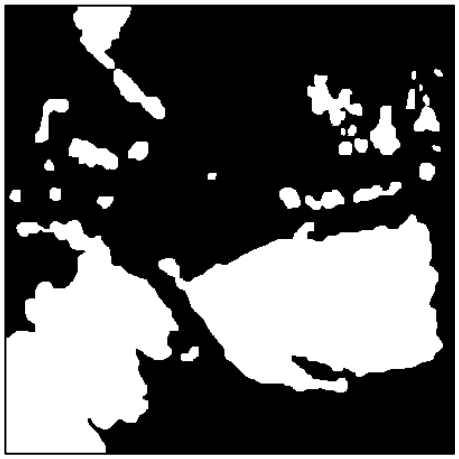


144x131



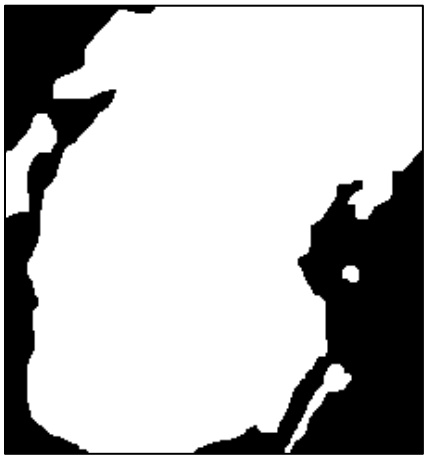
223x219



241x229



399x399



399x399



225x241



309x297

7.3 Avaliação e refinamento de imagem com resolução de 2627x2337 pixeis