CENTRO INTERNACIONAL DE ESTUDOS
DE DOUTORAMENTO E AVANZADOS
DA USC (CIEDUS)

TESE DE DOUTORAMENTO

FROM EFFICIENT AIRBORNE LIDAR DATA PROCESSING AND CLASSIFICATION TO 3D POINT CLOUD VISUALISATION

Jorge Martínez Sánchez

ESCOLA DE DOUTORAMENTO INTERNACIONAL DA UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
PROGRAMA DE DOUTORAMENTO EN INVESTIGACIÓN EN TECNOLOXÍAS DA INFORMACIÓN

SANTIAGO DE COMPOSTELA
2020

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

DECLARACIÓN DO AUTOR DA TESE
FROM EFFICIENT AIRBORNE LIDAR DATA PROCESSING AND CLASSIFICATION TO 3D POINT CLOUD VISUALISATION

Don Jorge Martínez Sánchez

Presento a miña tese, seguindo o procedemento adecuado ao Regulamento, e declaro que:

1. *A tese abarca os resultados da elaboración do meu traballo.*
2. *No seu caso, na tese faise referencia ás colaboracións que tivo este traballo.*
3. *A tese é a versión definitiva presentada para a súa defensa e coincide coa versión enviada en formato electrónico.*
4. *Confirmo que a tese non incorre en ningún tipo de plaxio de outros autores nin de traballos presentados por min para a obtención de outros títulos.*

En Santiago de Compostela, 30 de Outubro de 2020

Asdo. Jorge Martínez Sánchez

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

AUTORIZACIÓN DO DIRECTOR/TITOR DA TESE
FROM EFFICIENT AIRBORNE LIDAR DATA PROCESSING AND CLASSIFICATION TO 3D POINT CLOUD VISUALISATION

Don José Carlos Cabaleiro Domínguez, Profesor Titular da Área de Arquitectura e Tecnoloxía de Computadores da Universidade de Santiago de Compostela

Don David López Vilariño, Profesor Titular da Área de Electrónica da Universidade de Santiago de Compostela

INFORMAN:

*Que autorizamos a presentación desta tese, que se corresponde co traballo realizado por **Don Jorge Martínez Sánchez** baixo a nosa dirección, considerando que reúne os requisitos esixidos no Regulamento de Estudos de Doutoramento da USC, e que como directores desta non incorre nas causas de abstención establecidas na Lei 40/2015.*

En Santiago de Compostela, 30 de Outubro de 2020

Asdo. José Carlos Cabaleiro Domínguez
Director tese

Asdo. David López Vilariño
Director tese

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

A mis padres

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Computer science inverts the normal. In normal science, you're given a world, and your job is to find out the rules. In computer science, you give the computer the rules, and it creates the world.

Alan Kay

I am a big believer in believing in yourself and having self-belief.

Hafthor Björnsson

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Acknowledgments

I would like to thank my supervisors J. Carlos Cabaleiro and David López for allowing me to dive into the research world. Together with professors Francisco Fernández and Tomás Fernández, have always gave me valuable feedback and support, for which I will be forever grateful. I also want to thank Oscar García, without him the development of OLIVIA would not have been possible, and Miguel Yermo, who took over a big part of the workload during the last months, which has aid me to finish this thesis. Thanks my colleagues, Luis Roma, Rubén Laso, Silvia Rodríguez, and Sandra Buján, who helped me directly or indirectly during my research. Finally, I would like to thank the 3DGeo Research Group from Heidelberg University for their charming welcoming during my stay, and in particular to professor Bernhard Höfle for his support and valuable insights and to Katharina Anders for her outstanding assistance.

This work has received financial support from several institutions. The Xunta de Galicia with the projects GRC2014/008, R2014/041, R2014/049, R2016/037, and R2016/045. The Consellería de Cultura, Educación e Ordenación Universitaria (accreditations 2016–2019, ED431G/08, and ED431C 2018/2019 and 2019-2022 ED431G-2019/04 and reference competitive group 2019-2021). The Ministry of Science and Innovation with the project TIN2013-41129-P. The Ministry of Education, Culture and Sport, Government of Spain (Grant Number TIN2016-76373-P). The Center for the Development of Industrial Technology with the project Interconnecta: Advanced Tecnologies for large forestal incendies (colaboration between USC and INAER Galicia SL). The european network HiPEAC and the national network CAPAP-H. Babcock International Group PLC (Civil UAVs Initiative Fund of Xunta de Galicia). And the European Regional Development Fund (ERDF), which acknowledges the CiTIUS-Research Center in Intelligent Technologies of the University of Santiago de Compostela as a Research Center of the Galician University System.

The Vaihingen data set was provided by the German Society for Photogrammetry, Remote Sensing, and Geoinformation (DGPF) (Cramer, 2010). The Downtown Toronto dataset was provided by Optech Inc., First Base Solutions Inc., GeoICT Lab at York University, and ISPRS WG III/4. This work uses data services provided by the OpenTopography Facility with support from the National Science Foundation under NSF Award Numbers 1557484, 1557319, and 1557330. Babcock International and LaboraTe group (USC) provided the Alcoy and Trabada data, respectively. Some of the benchmarks has been carried out in the computation nodes provided by the Supercomputing Center of Galicia (CESGA).

30 de Outubro de 2020

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Contents

Abstract **1**

Resumo **3**

Resumen **11**

1 Introduction **19**
1.1 LiDAR Technology 20
1.2 Computational Efficiency 36
1.3 Objectives 40
1.4 Thesis Outline 41
1.5 Contribution 42

2 Segmentation **45**
2.1 Method 47
2.2 Results 51
2.3 Conclusions 59

3 Classification **61**
3.1 Method 62
3.2 Results 68
3.3 Discussion 74
3.4 Conclusions 78

4 Extraction of Building Points **79**
4.1 Method 80

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

4.2 Results . . . 88
4.3 Discussion . . . 95
4.4 Conclusions . . . 97

5 Extraction of Road Points **99**
5.1 Method . . . 101
5.2 Results . . . 112
5.3 Discussion . . . 123
5.4 Conclusions . . . 126

6 Modelling of the Road Network **129**
6.1 Method . . . 130
6.2 Discussion . . . 149
6.3 Conclusions . . . 151

7 Fast Filtering of Ground Points **153**
7.1 Method . . . 155
7.2 Results . . . 162
7.3 Embedded System Implementation . . . 169
7.4 Discussion . . . 176
7.5 Conclusions . . . 182

8 Visualisation of Point Clouds with OLIVIA **183**
8.1 OLIVIA . . . 186
8.2 Usage . . . 188
8.3 Evaluation . . . 192
8.4 Performance . . . 198
8.5 Conclusion . . . 200

Discussion **203**

Conclusions **207**

Bibliography **213**

List of Figures **243**

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

List of Tables **251**

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Abstract

The acquisition of knowledge about the world is an essential endeavour of science. However, performing on-site observations at a global scale is unfeasible, therefore remote sensing is an appealing alternative. Over the last two decades, LiDAR (Light Detection And Ranging), an active remote sensing technique, has gained significant adoption. LiDAR allows acquiring a 3D record of the target scene in the form of point cloud with high accuracy. The goal of this thesis is to develop efficient methods for the classification of LiDAR data. For this, both general-purpose methods (segmentation and classification) and application-specific methods (building and road points extraction) are proposed, which have been efficiently implemented in a middle-to-low level language with an optimal spatial indexing and multi-core parallelisation. Experiments showed that the proposed growing criteria of the region-growing segmentation improved the segmentation global score from 1.053 a 1.019 (less is better) and the overall classification accuracy by 3.57% with nearly a 10% improvement in the building class accuracy. The rule-based classification achieved without training an overall accuracy of 83.3%, with a class accuracy of 88.4%, 86.9% and 70.5%, for ground, vegetation and buildings, respectively. The building point extraction yielded correctness, completeness and quality of 92.39%, 92.64% and 86.97%, respectively, with a speedup of 2.08 over using the general-purpose classifier for this task. The road point extraction was capable of estimating the intensity threshold and showed correctness, completeness and quality of 83%, 93% and 78%, respectively, in the most complete road dataset we are aware of. The methods for road delineation and characterisation demonstrated promising experimental results. Also, the feasibility of real-time processing is explored. A method for ground filtering exploiting the scan-line acquisition pattern of the LiDAR data showed satisfactory results under visual inspection, and achieved, in a synthetic reference data, Type I and Type II errors of 0.25% and 2.71%, respectively, and a kappa value of 88.59%. The method has a high computational

performance, as it is capable of processing up to 1 million points per second in a workstation. The implementation was ported into a low-cost development board using FPGA acceleration, where experiments demonstrated that it is achievable to process up to 250,000 points per second in a higher-end reconfigurable system-on-chip, so it can cope with the data acquisition rates of the current lightweight scanners used in UAVs. Furthermore, a point cloud visualisation tool, namely OLIVIA, is presented. OLIVIA is an OpenGL-based open-source project implemented in Java, that offers an easy way to create customised visualisation and the capability of 3D stereoscopic view.

Resumo

Un dos esforzos fundamentais da ciencia é a obtención de coñecemento. Un xeito de obter información sobre o mundo que nos rodea é a través da observación. Non obstante, a observación *in situ* non é escalable e ás veces nin sequera é factible. Por este motivo, facer observacións de xeito remoto converteuse nunha alternativa para facer observacións a escala global e captar grandes cantidades de información sobre o mundo no que vivimos. A detección remota pódese definir como a adquisición de datos físicos dun obxecto sen ter contacto con el. Durante as últimas décadas, a teledetección permitiunos obter información moi valiosa sobre o medio ambiente. Nos últimos anos, unha técnica en particular atraeu o interese da comunidade científica: LiDAR, do inglés *Light Detection And Ranging*.

LiDAR é unha técnica de detección activa que permite medir a distancia entre o sensor e o obxecto de estudo. A orixe de LiDAR remóntase ás orixes do láser na década dos 50, aínda que non obtivo recoñecemento público ata o seu uso en 1971, durante a misión Apollo 15. A adopción desta tecnoloxía comezou a mediados dos anos 90, causada polos avances en tecnoloxías informáticas e a introdución de sistemas de posicionamento xeográfico como o GPS. Os datos LiDAR pódense empregar nunha ampla gama de aplicacións, o que atraeu o interese da comunidade científica e da industria durante as últimas dúas décadas. Hai diferentes estratexias de adquisición de datos LiDAR, nesta tese centrámonos na exploración aérea con láser. Nesta estratexia, o equipo LiDAR está composto principalmente por un escáner láser, un sistema GPS e un sistema inercial. O sensor emite un pulso de luz cara ao obxecto de estudo, que rebota sobre o obxecto e volve ao sensor. Dado que a velocidade da luz é constante, a distancia pódese derivar medindo o tempo que tarda o pulso en volver ao sensor. Estas distancias combínanse con GPS e sistemas inerciais para producir unha nube de puntos xeoreferenciada, onde para cada punto se rexistran as súas coordenadas 3D, a forza coa que rebota o pulso e outros datos do sensor. Debido ao movemento da aeronave durante a recollida

de datos, estes requiren un procesamento adicional para eliminar as variacións xeométricas e radiométricas derivadas. LiDAR converteuse nunha das técnicas de teledetección máis usadas debido ás súas vantaxes como a súa alta precisión, a adquisición de elevación e intensidade, a capacidade de penetrar baixo a vexetación ou a súa flexibilidade fronte ás condicións atmosféricas. A aplicación por excelencia é a xeración de modelos de terreo, aínda que as súas aplicacións son abundantes.

O obxectivo desta tese é desenvolver solucións para a clasificación de datos LiDAR. En concreto, se queren identificar as catro clases máis básicas: terreo, vexetación, edificios e estradas. Suponse que a nube de puntos LiDAR é a única fonte de datos. É previsible que o uso de fontes de datos adicionais melloren o resultado das solucións desenvolvidas. Non obstante, isto aumentaría o custo da adquisición de datos, engadiría a complexidade da integración de datos e, en moitos casos, simplemente non está dispoñible. Por iso, é interesante explorar o que se pode conseguir empregando só datos LiDAR. As solucións proporcionadas deben ter un grao de automatización, de xeito que requiran unha mínima intervención manual. Idealmente, os resultados deberían ser xeralizables, sen que o usuario precise establecer parámetros específicos. Polo tanto, prefírese investigar metodoloxías que non empreguen a aprendizaxe automática. Estas técnicas demostraron a súa eficacia, pero requiren unha fase de adestramento para a que se precisan datos de referencia: puntos etiquetados por expertos cuxa etiqueta se supón correcta. Isto ten varias implicacións. En primeiro lugar, a creación de datos de referencia ten grandes custos financeiros e de tempo, polo que xeralmente só hai unha pequena cantidade dispoñible. E segundo, dado que as nubes de puntos poden ter características moi diferentes (por exemplo, densidade de puntos, precisión vertical, rango radiométrico, etc.), é difícil garantir que se obteñan bos resultados para unha nube de puntos diferente das que proveñen os datos da referencia. Finalmente, as implementacións deben ser computacionalmente eficientes. Unha característica do procesamento de datos LiDAR é a súa elevada demanda computacional. Nunha imaxe, os píxeles dispóñense nunha malla regular e o acceso aos oito veciños dun píxel é inmediato. Non obstante, en LiDAR, os puntos da nube están distribuídos de xeito desigual, ademais de ter unha dimensión adicional. Isto fai que a busca de veciños, que é unha operación básica e recorrente en calquera algoritmo, sexa caro computacionalmente. Este custo aumenta canto maior sexa a densidade da nube de puntos. Dado que as técnicas de adquisición de datos LiDAR están en desenvolvemento continuo e a tendencia é a de obter nubes cada vez máis densas, é imprescindible que as solucións que se implementen sexan computacionalmente eficientes. En relación con este aspecto, é de

especial interese explorar a viabilidade do procesamento en tempo real.

Para cumprir os obxectivos, abordáronse varias das fases de procesamento de datos LiDAR, comezando por unha das máis básicas, a segmentación. Isto consiste en dividir a nube de puntos nun conxunto de rexións homoxéneas, nas que cada punto ten características similares. O obxectivo é separar a nube de puntos en rexións con máis significado e máis fáciles de analizar. A principal vantaxe de realizar unha segmentación é que proporciona información contextual e evita que os puntos se etiqueten de forma diferente aos seus puntos veciños, o que se denomina efecto *sal e pementa*. Un dos métodos de segmentación máis empregados é o crecemento de rexións. No capítulo 2, propoñemos un método xeral de segmentación de nube de puntos baseado no crecemento de rexións. A particularidade da nosa proposta é, en primeiro lugar, que os puntos planos e non planos están segmentados utilizando diferentes características e, en segundo lugar, que se modifica o criterio de crecemento tradicional. En primeiro lugar, calcúlanse os vectores normais de cada punto, despois tómanse como sementes os puntos ordenados por curvatura ascendente e, finalmente, realízase unha segmentación de crecemento en dúas fases das rexións. Na primeira fase tense en conta como características a proximidade espacial, a intensidade e o vector normal, mentres que na segunda fase non se ten en conta o vector normal. En canto ao criterio de crecemento, o punto analizado compárase cun punto artificial, chamado epicentro, que ten un valor intermedio entre o centroide da rexión e o último punto engadido a ela. Para avaliar os resultados, propóñense dúas estratexias. O primeiro é avaliar a calidade da segmentación en si, adaptando un método de avaliación non supervisado existente para as imaxes. O segundo é avaliar a calidade da segmentación en función do resultado da clasificación dos datos segmentados, utilizando a clasificación de referencia Vaihingen, publicada polo ISPRS. Os experimentos mostran que a inclusión do epicentro mellora tanto a calidade da segmentación como a da clasificación posterior. A puntuación de segmentación global reduciuse de 1,053 a 1,019 (menor é mellor) e a precisión global da clasificación aumentou un 3,57%, destacando un aumento na precisión da clasificación de construcións de case un 10%, o que valida a nosa proposta.

A seguinte etapa básica é a clasificación. A clasificación consiste en asignar a cada punto unha etiqueta que identifique ese punto nunha das categorías ou clases que se están a considerar. As clases máis comúns e as que se consideran nesta tese son o terreo, vexetación, edificios e estradas. O capítulo 4 presenta un método de clasificación baseado en regras que identifica as tres primeiras clases. A clasificación toma como entrada a segmentación presentada nesta tese e aplica un conxunto de regras heurísticas baseadas na elevación e na planari-

dade para etiquetar os puntos do terreo, vexetación e edificios. En primeiro lugar, extráense os puntos límites de cada rexión e despois determínase se as rexións están ou non elevadas en relación ao chan examinando as diferenzas de altura entre os puntos límites das rexións veciñas. Posteriormente, a información de planaridade da segmentación úsase para determinar se as rexións elevadas corresponden a edificios ou vexetación. Finalmente, realízase un filtrado de áreas para eliminar as zonas de vexetación planas e pequenas que poden clasificarse incorrectamente como edificios. Comparando os resultados coa clasificación de referencia de Vaihingen, lógrase unha precisión total do 83,3%, con precisións do 88,4%, 86,9% e 70,5%, respectivamente para as clases de terreo, vexetación e construción. Estes resultados están por debaixo dos resultados obtidos por métodos que utilizan a aprendizaxe automática. Non obstante, a comparación está lonxe de ser xusta. Despois de facer unha revisión da metodoloxía de avaliación, móstrase que os autores utilizan diferentes metodoloxías en termos de: os conxuntos de datos empregados, a estratexia para crear os datos de referencia ou a porcentaxe de datos de referencia con respecto ao tamaño do conxunto de datos. Ademais, moitos dos métodos fan uso de datos adicionais, como os datos LiDAR de forma de onda completa, non dispoñibles nesta tese. Para facer unha avaliación máis completa, a precisión da clasificación mídese en función da precisión na identificación de construcións nos sitios de Alcoi e unha área máis grande de Vaihingen. Debido á falta de datos de referencia, delimitáronse manualmente 66 e 227 construcións, respectivamente. As probas mostran a completitude e precisión do 100% e do 84,85% en Alcoy e do 96,46% e do 98,68% en Vaihingen, respectivamente. Non obstante, cremos que os resultados obtidos son satisfactorios. Para reducir os tempos de execución da clasificación, algunhas etapas están paralelizadas, o que representa o 64,86% do tempo de execución. Aplicando a lei de Amdahl e asumindo un factor de mellora igual ao número de núcleos físicos do equipo usado, o factor de mellora teórico máximo é 1,85. Despois da paralelización, obtense un factor de mellora de 1,81. Finalmente, discútense os tres posibles enfoques para a creación de datos de referencia, facendo fincapé na xeración automática de datos de referencia sintéticos, e particularmente en HELIOS, un simulador LiDAR creado polo Instituto de Xeografía da Universidade de Heidelberg.

Despois de cubrir os procesos máis xerais, como a segmentación e a clasificación, abordanse outras aplicacións máis específicas. O capítulo 4 trata da identificación de construcións, prescindindo dos métodos anteriores e adoptando un enfoque de procesamento máis dirixido. En primeiro lugar, os puntos non terreo identifícanse cun método de última xeración baseado nun filtrado morfolóxico progresivo. Despois, os puntos non terreo fíltranse segundo

diferentes características: altura, pendente, dimensionalidade, opacidade e morfoloxía. Os puntos restantes agrúpanse usando un método de agrupamento, únense grupos próximos e finalmente fíltranse por área. Comparando os resultados coa clasificación de referencia de Vaihingen, conséguese a precisión, integridade e calidade de 92,39%, 92,64% e 86,97%, respectivamente. Unha inspección visual nos sitios de Alcoi e Arzúa tamén mostra resultados satisfactorios. Para reducir os tempos de execución, algunhas das etapas están paralelizadas, o que representa o 83,90% do tempo total de execución. Aplicando a lei de Amdahl e asumindo un factor de mellora igual ao número de núcleos físicos do equipo usado, o factor de mellora teórico máximo é 2,70. Despois da paralelización, obtense un factor de mellora de 2,58. Isto é máis rápido que a clasificación proposta no capítulo 3, concretamente, conséguese un factor de mellora de 1,46 sobre a versión secuencial e 2,08 sobre a versión paralelizada.

Outra aplicación tratada nesta tese é a extracción de estradas. A extracción de estradas implica dúas cuestións: identificación e delimitación. Ambos temas están cubertos nos capítulos 5 e 6, respectivamente. Propuxéronse varios métodos de extracción de estradas, pero hai dúas limitacións principais. En primeiro lugar, falta automatización, xa que requírese establecer parámetros críticos, especialmente o limiar de intensidade. E segundo, descoñécese se son suficientemente xeneralizables, xa que os experimentos lévanse a cabo en poucos sitios de estudo, xeralmente un só e de natureza urbana. O capítulo 5 presenta un método para a identificación automática de puntos de estrada tanto en áreas urbanas como rurais. O método baséase nun filtro de puntos segundo varias características: intensidade, curvatura, densidade e área. Destacamos que a parte máis importante é o filtrado de intensidade, para o cal se introduce unha versión mellorada do algoritmo de equilibrio de asimetría, engadindo un filtrado de valores atípicos, estandarizando a velocidade de filtrado e proporcionándolle bidireccionalidade. O usuario só precisa establecer un parámetro, o ancho mínimo da estrada, que demostrou que non influe de xeito crítico no resultado. Os experimentos, realizados nun gran conxunto de datos composto por 10 sitios con diferentes métodos de adquisición e graos de urbanización, mostran unha integridade, exactitude e calidade de 93%, 83% e 78%, respectivamente. Os resultados son competitivos co estado da arte, só superados por métodos baseados na técnica de Campo Aleatorio Condicional, aínda que as métricas de calidade destes métodos calculáronse nun único sitio de estudo. Deste xeito, o método proposto sitúase como unha opción moi relevante para a identificación automática de estradas. Para reducir os tempos de execución, proponse unha optimización secuencial na parte máis costosa do algoritmo, para o que se consegue un factor de mellora medio de 4,56. Ademais, algunhas etapas están paralelizadas,

o que representa o 89,52% do tempo de execución. Aplicando a lei de Amdahl e asumindo un factor de mellora igual ao número de núcleos físicos do equipo usado, o factor de mellora teórico máximo é 3,04. Despois da paralelización, obtense unha aceleración de 2,32. En canto á delimitación da estrada, este segue sendo un gran desafío e a maioría dos métodos existentes presentan unha calidade mellorable. No capítulo 6 preséntase un método de modelado de estrada que é capaz de extraer a liña central, encher os ocos e estimar o ancho. Para iso, adaptáronse a LiDAR dous métodos de procesamento de imaxes, concretamente a Sinatura de Textura Angular e os Modelos de Contornos Activos. Os resultados experimentais do método proposto son prometedores. En traballos futuros, considérase necesario investigar como aumentar o grao de automatización, ademais de realizar unha avaliación exhaustiva.

Ademais, explórase a viabilidade do procesamento en tempo real no campo dos datos LiDAR. Abórdase o filtrado de puntos de terreno, xa que é a aplicación por excelencia en LiDAR e case todas as aplicacións requiren este paso previo. No capítulo 7 introdúcese un método para filtrar puntos que explota o patrón nas liñas de dixitalización onde se adquiren datos LiDAR. Supóñese que o sensor pode proporcionar os datos xeorreferenciados e tomando como entrada un conxunto de liñas de dixitalización, este método realiza unha interpolación 1D iterativa baseada en *spline* en cada liña de dixitalización, onde o número de nodos no *spline* aumenta con cada iteración, refinando a *spline*, e os nodos propaganse ás liñas adxacentes. Os puntos están marcados como terreo ou non terreo dependendo do seu residuo na *spline* final. Os experimentos realizados en sitios de probas sintéticas xerados con HELIOS mostran erros de tipo 1 (comisión) e tipo 2 (omisión) de 0,25% e 2,71%, respectivamente, e un valor kappa de 88,59%. Unha inspección visual nos sitios de Alcoy, Vahingen e Trabada tamén mostra resultados satisfactorios. Os tempos de execución son baixos e ata un millón de puntos por segundo poden procesarse nun ordenador de traballo. Isto faino atractivo tamén para o procesamento *offline*, xa que a tendencia é adquirir nubes de puntos máis densas. Para usar o método en sistemas integrados con drones, realizouse unha implementación nunha placa de desenvolvemento ZedBoard de baixo custo con aceleración FPGA. Esta implementación é capaz de procesar máis de 60.000 puntos por segundo cun baixo consumo de enerxía. Extrapolando o rendemento obtido a unha placa ultraescala de gama alta, determínase que o método é capaz de procesar aproximadamente 250.000 puntos por segundo cunha eficiencia enerxética aínda maior. Esta capacidade de procesamento permitiría procesar os datos obtidos dalgúns sensores existentes en tempo real.

Cando se desenvolven métodos para procesar datos LiDAR, é necesaria unha ferramenta

de visualización para inspeccionar as saídas dos algoritmos. Isto axuda tanto a verificar a súa precisión como a comprender como funcionan. Ademais, a actividade científica conleva á asistencia a eventos divulgativos ou incluso comerciais, co obxectivo de promover a transferencia tecnolóxica. Por iso, é de gran importancia poder amosar resultados dun xeito atractivo. Dado que non atopamos ferramentas existentes que se adapten ás nosas necesidades, desenvolvemos a nosa propia ferramenta de visualización, OLIVIA, do inglés *Open LIdar VIsualiser and Analyzer)*, introducida no capítulo 8. OLIVIA é unha ferramenta de código aberto enfocada a desenvolvedores, que ofrece funcionalidades básicas de visualización (selección de puntos/grupos, debuxo de primitivas, captura de imaxe/vídeo, espello da cámara...) e que se pode modificar facilmente para construír visualizacións personalizadas, adaptándose así ás necesidades do usuario. Ademais, permite a visualización en modo 3D estereoscópico, de xeito que se xeran dúas imaxes, unha para cada ollo, para crear unha visión 3D cun efecto de profundidade. OLIVIA está codificado en Java e usa funcións OpenGL, a API de debuxo gráfico máis soportada da industria, a través da biblioteca JOGL. A ferramenta ten tres paquetes principais: *core*, que implementa todas as funcións principais, *extended*, que implementa funcións adicionais que se comparten entre visualizacións e *visualization*, que implementa as funcionalidades específicas da visualización personalizada. No capítulo móstranse exemplos de como se pode usar para visualizar as saídas de diferentes métodos, a partir de algoritmos básicos como o cálculo normal e o cálculo da envolvente, procesos como a segmentación e clasificación e aplicacións específicas como a identificación de zonas de aterraxe. Ademais, OLIVIA empregouse ao longo da tese. As probas de rendemento mostran que OLIVIA pode usarse en ordenadores con dispositivos gráficos integrados, aínda que é preferible unha tarxeta gráfica dedicada para ver nubes de millóns de puntos. Cunha tarxeta gráfica de gama media, un Nvidia 1050TI, pódense mostrar 50 millóns de puntos a 30 fotogramas por segundo.

Os desenvolvementos desta tese materializáronse en varias contribucións. No campo científico, obtiveron catro artigos en revistas JCR e tres artigos en congresos. No campo de *software*, deron lugar a un clasificador que distingue catro clases (terreo, vexetación, edificios e estradas), un visor rexistrado como propiedade intelectual e un filtro de puntos de terreo implementado nun sistema integrado.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Resumen

Uno de los esfuerzos fundamentales de la ciencia es la obtención de conocimiento. Una forma de conseguir información sobre el mundo que nos rodea es mediante la observación. Sin embargo, la observación *in situ* no es escalable y algunas veces ni siquiera es viable. Por esta razón, hacer observaciones de forma remota se ha convertido en una alternativa para realizar observaciones a escala global y capturar ingentes cantidades de información sobre el mundo en el que vivimos. El sensado remoto puede definirse como la adquisición de datos de carácter físico de un objeto sin tener contacto con este. Durante las últimas décadas, el sensado remoto nos ha permitido obtener información muy valiosa sobre el entorno. Durante los últimos años, una técnica en particular ha suscitado el interés de la comunidad científica: LiDAR, del inglés *LIght Detection And Ranging*.

LiDAR es una técnica de sensado activa que permite medir la distancia entre el sensor y el objeto de estudio. El origen del LiDAR se remonta a los orígenes del láser en los años 50s, aunque no ganó el reconocimiento público hasta su uso en 1971, durante la misión Apollo 15. La adopción de esta tecnología comenzó a mitad de los años 90s, propiciada por los avances en las tecnologías de la computación y a la introducción de sistemas de geoposicionamiento como el GPS. Los datos LiDAR pueden usarse en un amplio rango de aplicaciones, lo que ha atraído el interés por parte de la comunidad científica y de la industria durante las dos últimas décadas. Existen diferentes estrategias de adquisición de datos LiDAR, en esta tesis nos centramos en el escaneado láser aéreo. En esta estrategia, el equipo LiDAR se compone principalmente de un escáner láser, un sistema GPS y un sistema inercial. El sensor emite un pulso de luz hacia el objeto de estudio, que rebota en el objeto y vuelve al sensor. Dado que la velocidad de la luz es constante, la distancia puede derivarse midiendo el tiempo que tarda el pulso en volver al sensor. Estas distancias se combinan con los sistemas GPS e inercial para producir una nube de puntos georeferenciada, en donde para cada punto se registran sus

coordenadas 3D, la fuerza con la que rebota el pulso y otros datos propios del sensor. Debido al movimiento de la aeronave durante la toma de datos, estos requieren de un procesado posterior para eliminar las variaciones geométricas y radiométricas derivadas. LiDAR se ha convertido en una de las técnicas de sensado remoto más usadas, debido a sus ventajas como su alta precisión, la adquisición de la elevación e intensidad, la capacidad de penetrar bajo la vegetación, o la flexibilidad frente a condiciones atmosféricas. La aplicación por excelencia es la generación de modelos del terreno, aunque sus aplicaciones son abundantes.

El objetivo de esta tesis es desarrollar soluciones para la clasificación de datos LiDAR. En concreto, se quieren identificar las cuatro clases más básicas: terreno, vegetación, construcciones y carreteras. Se asume que la nube de puntos LiDAR es la única fuente de datos. Es predecible que el uso de fuentes de datos adicionales mejoren el resultado de las soluciones desarrolladas. Sin embargo, esto incrementaría el coste de adquisición de datos, añadiría la complejidad de la integración de datos y, en muchos casos, simplemente no están disponibles. Por lo que es interesante explorar lo que se puede conseguir usando únicamente datos LiDAR. Las soluciones aportadas deben gozar de un grado de automatización, de forma que requieran la mínima intervención manual. Idealmente, los resultados deben ser generalizables, sin que se requiera el ajuste específico de parámetros por parte el usuario. Por ello, se prefiere investigar metodogías que no usen aprendizaje máquina. Estas técnicas han demostrado su efectividad, sin embargo, requieren de una fase de entrenamiento para la cual se necesitan datos de referencia: puntos etiquetados por expertos cuya etiqueta se asume ser correcta. Esto tiene varias implicaciones. Primero, la creación de datos de referencia tiene grandes costes económicos y temporales, así que, generalmente solo se dispone de una pequeña cantidad. Y segundo, dado que las nubes de puntos pueden tener características muy diferentes (p. ej. densidad de puntos, precisión vertical, rango radiométrico, etc), es difícil garantizar que se obtengan buenos resultados para una nube de puntos diferente de las que provienen los datos de entrenamiento. Finalmente, las implementaciones deben ser computacionalmente eficientes. Una característica sobre procesamiento de datos LiDAR es su gran demanda computacional. En una imagen, los píxeles están organizamos en una malla regular, y el acceso a los ocho vecinos de un píxel es inmediato. Sin embargo, en LiDAR, los puntos de la nube están distribuidos de forma irregular, además de contar con una dimensión adicional. Esto hace que la búsqueda de vecinos, la cual es una operación básica y que es recurrente en cualquier algoritmo, sea costosa computacionalmente. Este coste aumenta cuanto mayor sea la densidad de la nube de puntos. Dado las técnicas de adquisición de datos LiDAR están en

continuo desarrollo, y la tendencia es obtener cada vez nubes más densas, es imperativo que las soluciones que se implementen sean computacionalmente eficientes. En relación con este aspecto, es de especial interés explorar la viabilidad del procesamiento en tiempo real.

Para cumplir los objetivos, se han abordado varias de las etapas de procesamiento de los datos LiDAR, empezando por una de las más básicas, la segmentación. Esta consiste en dividir la nube de puntos en un conjunto de regiones homogéneas, en la que cada punto presenta caraterísticas similares. El objetivo es separar la nube de puntos en regiones con más significado y más fáciles de analizar. La principal ventaja de realizar una segmentación es que aporta información contextual, y evita que los puntos sean etiquetados de forma diferente que sus puntos vecinos, a lo que se denomina el efecto *sal y pimenta*. Uno de los métodos de segmentación más utilizados es el de crecimiento de regiones. En el capítulo 2, proponemos un método de segmentación de nubes de puntos de carácter general basado en crecimento de regiones. La particularidad de nuestra propuesta es, en primer lugar, que los puntos planos y los no planos se segmentan usando distintas características, y en segundo lugar, que se modifica el criterio de crecimiento tradicional. Primero, se estiman los vectores normales de cada punto, después se toman como semillas los puntos ordenados por curvatura de forma ascendente, y finalmente se realiza una segmentación de crecimiento de regiones en dos fases. En la primera fase se tienen en cuenta como características la proximidad espacial, la intensidad y el vector normal, mientras que en la segunda no se tiene en cuenta el vector normal. En cuanto al criterio de crecimiento, el punto analizado se compara con un punto artificial, llamado epicentro, el cual tiene un valor intermedio entre el centroide de la región y el último punto añadido a la misma. Para evaluar los resultados, se proponen dos estrategias. La primera es evaluar la calidad de la segmentación en sí, adaptando un método existente de evaluación no supervisado para imágenes. La segunda es evaluar la calidad de la segmentación en función del resultado de clasificar los datos segmentados, usando la clasificación de referencia de Vaihingen, publicada por el ISPRS. Los experimentos muestran que la inclusión del epicentro mejora tanto la calidad de la segmentación como la de la subsiguiente clasificación. La puntuación global de segmentación se redujo de 1,053 a 1,019 (menor es mejor) y la precisión total de la clasificación aumentó un 3,57%, destacando un incremento en la precisión de la clasificación de construcciones de casi un 10%, lo que valida nuestra propuesta.

La siguiente etapa básica es la clasificación. La clasificación consiste en asignar a cada punto una etiqueta que identifique a ese punto en una de las categorías, o clases, que se están considerando. Las clases más comunes, y las consideradas en esta tesis, son terreno, vege-

tación, construcciones y carreteras. En el capítulo 3 se presenta un método de clasificación basado en reglas que identifica las tres primeras clases. La clasificación toma como entrada la segmentación presentada en esta tesis, y aplica un conjunto de reglas heurísticas basadas en elevación y planaridad para etiquetar los puntos en terreno, vegetación y construcciones. Primero, se extraen los puntos frontera de cada región de puntos y después se determina si las regiones son o no elevadas respecto al terreno, examinando las diferencias de altura entre los puntos frontera de regiones vecinas. Posteriormente, se usa la información de planaridad proveniente de la segmentación para determinar si las regiones elevadas se corresponden con construcciones o con vegetación. Finalmente, se realiza un filtrado por área para eliminar áreas planas y pequeñas de vegetación que pueden ser clasificadas erróneamente como construcciones. Comparando los resultados con la clasificación de referencia de Vaihingen, se consigue una precisión total de un 83,3%, con precisiones de 88.4% 86.9% y 70.5%, para las clases de terreno, vegetación y construcciones, respectivamente. Estos resultados están por debajo de los resultados obtenidos por métodos que usan aprendizaje máquina. Sin embargo, la comparación dista de ser justa. Tras hacer una revisión de la metodología de evaluación, se muestra que los autores usan metodologías diferentes en cuanto a: los conjuntos de datos usados, la estrategia de creación de los datos de referencia o el porcentage de datos de referencia respecto al tamaño del conjunto de datos. Además, muchos de los métodos hacen uso de datos adicionales, como por ejemplo, datos LiDAR de forma onda completa, no disponibles en esta tesis. Para hacer una evaluación más completa, se mide la precisión de la clasificación en función de la precisión en la identificación de construcciones en los sitios de Alcoy y un área más grande de Vaihingen. Debido a la falta de datos de referencia, se han delineado manualmente 66 y 227 construcciones, respectivamente. Las pruebas muestran una completitud y exactitud, de 100% y 84.85% en Alcoy, y de 96.46% y 98.68% en Vaihingen, respectivamente. Con todo, creemos que los resultados obtenidos son satisfactorios. Para reducir los tiempos de ejecución de la clasificación, se paralelizan algunas etapas, que suponen un 64.86% del tiempo de ejecución. Aplicando la ley de Amdahl, y asumiendo un factor de mejora igual al número de núcleos físicos del equipo usado, el máximo factor de mejora teórico es 1.85. Tras la paralelización se obtiene un factor de mejora de 1.81. Finalmente, se discuten los tres enfoques posibles para la creación de datos de referencia, haciendo énfasis en la generación automática de datos de referencia sintéticos, y particularmente en HELIOS, un simulador LiDAR creado por el Instituto de Geografía de la Universidad de Heidelberg.

Después de tratar los procesos más generales, como la segmentación y la clasificación, se

abordan otras aplicaciones más específicas. En el capítulo 4 se trata la identificación de construcciones, prescindiendo de los métodos anteriores, y haciendo un enfoque de procesamiento más dirigido. Primero, se identifican los puntos no-terreno con un método del estado del arte basado en filtrado morfológico progresivo. Después se introduce un método para identificar los puntos terreno. Los puntos no-terreno se filtran según diferentes características: altura, pendiente, dimensionalidad, opacidad y morfología. Después, los puntos restantes se agrupan con un método de agrupamiento, los grupos cercanos se unen y finalmente se filtran por área. Comparando los resultados con la clasificación de referencia de Vaihingen, se consigue una exactitud, completitud y calidad del 92.39%, 92.64% y 86.97%, respectivamente. Una inspección visual en los sitios de Alcoy y Arzúa también muestran resultados satisfactorios. Para reducir los tiempos de ejecución, se paralelizan algunas de las etapas, que suponen un 83.90% del tiempo de ejecución total. Aplicando la ley de Amdahl, y asumiendo un factor de mejora igual al número de núcleos físicos del equipo usado, el máximo factor de mejora teórico es 2.70. Tras la paralelización se obtiene un factor de mejora de 2.58. Este es más rápido que la clasificación propuesta en el capitulo 3, concretamente se consigue un factor de mejora de 1.46 sobre la versión secuencial y 2.08 sobre la versión paralelizada.

Otra aplicación tratada en esta tesis es la extracción de carreteras. La extracción de carreteras implica dos cuestiones: identificación y delineado. Ambos temas son tratados en los capítulos 5 y 6, respectivamente. Varios métodos de extracción de carreteras han sido propuestos, sin embargo, estos presentan dos limitaciones importantes. Primero, hay una falta de automatización, ya que se requiere establecer parámetros críticos, particularmente el umbral de intensidad. Y segundo, se desconoce si son suficientemente generalizables, ya que los experimentos se realizan en pocos sitios de estudio, generalmente uno y de carácter urbano. En el capítulo 5 se presenta un método para la identificación automática de puntos de carretera tanto en zonas urbanas como rurales. El método se basa en un filtrado de puntos según varias características: intensidad, curvatura, densidad y área. Destacamos que la parte más importante es el filtrado por intensidad, para el cual se introduce una versión mejorada del algoritmo de balanceo de asimetría, añadiendo un filtrado de valores atípicos, estandarizando la velocidad del filtrado y dotándolo de bidireccionalidad. El usuario solo necesitar establecer un parámetro, el ancho mínimo de carretera, que ha demostrado no influir de forma crítica en el resultado. Los experimentos, realizados en un amplio conjunto de datos formado por 10 sitios con diferentes métodos de adquisición y grados de urbanización, muestran una completitud, exactitud y calidad de 93%, 83% y 78%, respectivamente. Los resultados son competitivos

con el estado del arte, solo superados por métodos basados en la técnica de Campo Aleatorio Condicional, aunque las métricas de calidad de estos métodos han sido calculadas en un único sitio de estudio. De esta forma, el método propuesto se posiciona como una opción muy relevante para la identificación automática de carreteras. Para reducir los tiempos de ejecución, se propone una optimización secuencial en la parte más costosa del algoritmo, para la cual se consigue un factor de mejora promedio de 4.56. Además, se paralelizan algunas etapas, que suponen un 89.52% del tiempo de ejecución. Aplicando la ley de Amdahl, y asumiendo un factor de mejora igual al número de núcleos físicos del equipo usado, el máximo factor de mejora teórico es 3.04. Tras la paralelización se obtiene una aceleración de 2.32. En cuanto al delineado de la carretera, este sigue siendo un gran desafío y la mayoría de los métodos existentes presentan una calidad mejorable. En el capítulo 6 se presenta un método de modelado de carretera, el cual es capaz de extraer la línea central, rellenar los huecos y estimar la anchura. Para ello, se han adaptado a LiDAR dos métodos procedentes de procesamiento de imagen, en concreto la Firma de Textura Angular y los Modelos de Contornos Activos. Los resultados experimentales del método propuesto son prometedores. En trabajos futuros, se considera necesario investigar en cómo aumentar el grado de automatización, además de realizar una evaluación exhaustiva.

Además, se explora la viabilidad del procesamiento en tiempo real en el ámbito de los datos LiDAR. Se aborda el filtrado de puntos de terreno, dado que es la aplicación por excelencia en LiDAR y casi la totalidad de las aplicaciones requieren de este paso previo. En el capítulo 7 se introduce un método para el filtrado de puntos que explota el patrón en líneas de escaneado en la que se adquieren los datos LiDAR. Asumiento que el sensor puede proporcionar los datos georeferenciados, y tomando como entrada un conjunto de líneas de escaneado, este método realiza una interpolación 1D iterativa basado en *spline* en cada línea de escaneado, en la que el número de nodos del *spline* aumenta en cada iteración, refinando el *spline*, y los nodos son propagados a las líneas adyacentes. Los puntos se etiquetan como terreno o no terreno dependiendo de su residuo al *spline* final. Los experimentos realizados en sitios de prueba sintéticos generados con HELIOS muestran errores de tipo 1 (comisión) y de tipo 2 (omisión) de 0.25% y 2.71%, respectivamente, y un valor kappa de 88.59%. Una inspección visual en los sitios Alcoy, Vahingen y Trabada muestra también resultados satisfactorios. Los tiempos de ejecución son bajos y es posible procesar hasta cerca de 1 millón de puntos por segundo en una computadora de trabajo. Esto lo hace atractivo también para el procesamiento *offline*, dado que la tendencia es adquirir nubes de puntos más densas.

Para usar el método en sistemas embebidos en drones, se realizó una implementación en una placa de desarrollo de bajo costo ZedBoard con aceleración FPGA. Esta implementación es capaz de procesar más de 60.000 puntos por segundo con un bajo consumo de potencia. Extrapolando el rendimiento obtenido a una placa Ultrascale de alta gama, se determina que el método es capaz de procesar aproximadamente 250.000 puntos por segundo con aún mayor eficiencia energética. Esta capacidad de procesamiento permitiría procesar en tiempo real los datos obtenidos de algunos sensores existentes.

A la hora de desarrollar métodos para el procesamiento de datos LiDAR, es necesaria una herramienta de visualización que permita inspeccionar las salidas de los algoritmos. Esto ayuda tanto a verificar su exactitud como a entender su funcionamiento. Además, la actividad científica conlleva a la asistencia eventos divulgativos o incluso de carácter comercial, con el objetivo fomentar la transferencia tecnológica. Por lo que es de gran importancia ser capaz de mostrar resultados de una forma atractiva. Debido a que no hemos encontrado herramientas existentes que se adapten a nuestras necesidades, hemos desarrollado nuestra propia herramienta de visualización, OLIVIA, del inglés (*Open LIdar VIsualiser and Analyser)*, introducida en el capítulo 8. OLIVIA es una herramienta de código abierto enfocada a desarrolladores, que proporciona funcionalidades básicas de visualización (selección de punto/grupo, dibujado de primitivas, captura de imágen/vídeo, espejado de cámara...), y que puede ser modificada de forma sencilla para construir visualizaciones personalizadas, adaptándose así a las necesidades del usuario. Además, permite la visualización en modo 3D estereoscópico, de forma que se generan dos imágenes, una para cada ojo, para crear una visión 3D con efecto de profundidad. OLIVIA está codificado en Java y usa funciones OpenGL, la API de dibujado gráfico con más soporte de la industria, a través de la librería JOGL. La herramienta tiene tres paquetes principales: *core*, que implementa todas las funcionalidades principales, *extended*, que implementa funcionalidades adicionales que son compartidas entre visualizaciones, y *visualisation*, que implementa las funcionalidades específicas de la visualización personalizada. En el capítulo se muestran ejemplos de cómo puede usarse para visualizar las salidas de diferentes métodos, desde algoritmos básicos como el cálculo de normales y cálculo de envolventes, procesos como la segmentación y clasificación, y aplicaciones específicas como la identificación de zonas de aterrizaje. Además, OLIVIA ha sido usada a lo largo de la tesis. Las pruebas de rendimiento muestran que OLIVIA puede usarse en equipos con dispositivos gráficos integrados, aunque para visualizar nubes de varios millones de puntos es preferible una tarjeta gráfica dedicada. Con una tarjeta gráfica de gama media, una Nvidia 1050TI, se

pueden visualizar 50 millones de puntos a 30 imágenes por segundo.

Los desarrollos de esta tesis se han materializado en varias contribuciones. En el ámbito científico, han dado lugar a cuatro artículos en revistas JCR y tres artículos en conferencias. En el ámbito de *software*, han dado lugar a un clasificador que distingue cuatro clases (terreno, vegetación, construcciones y carreteras), un visualizador que ha sido registrado como propiedad intelectual y un filtro de puntos terreno implementado en un sistema embebido.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Chapter 1

Introduction

The acquisition of knowledge about the world is an essential endeavour of science. In order to learn about the world around us, observations need to be made. An observation can be defined as the active collection of information from a primary source. While observations can be done by on-site, this approach is not scalable, and sometimes, is not even feasible. Because of this, remote sensing has become the alternative to perform observations at a global scale and gather immense amounts of information about the world we live on.

Remote sensing has been defined many times [36]. In this thesis, we define remote sensing as the acquisition of physical data of an object without touch or contact [148]. The process of remote sensing encompasses the concepts of *object*, *sensor data*, *extracted information* and *applications*. An *object* is any physical entity that scientists wish to examine, natural (soil, vegetation, water...) or man-made (buildings, bridges, roads...). The *sensor data* is obtained when the instrument views the physical object by recording its electromagnetic radiation. Remote sensing can be categorised into passive and active. In passive sensing, sensors measure the radiation reflected of emitted naturally from the object (e.g., a camera), while in active sensing the sensor emits its own radiation towards the object and measures the backscattered radiation (e.g., a laser). Sensor data is often abstract and might require integration with other sensors. Because of this, transformations need to be applied to sensor data to obtain *extracted information* that can be interpreted. The extracted information is then used in *applications*, which try to address specific practical problems (e.g., land-use planning).

The origins of remote sensing are tied to the beginnings of photography in the early 1800s. The first aerial image photograph was taken from a tethered balloon in 1858 by Gaspard-Félix

Tournachon. During World War I (1914-1918), the value of aerial photography for military reconnaissance and surveillance became clear, and aerial photographs were taken from aeroplanes on a routine basis. World War II (1939-1945) accelerated the adoption of the electromagnetic spectrum outside the visible region, notably the infrared and microwave regions. The first use of the *remote sensing* was in late 1950 by Evelyn Pruitt, a scientist from the U.S. Navy's Office of Naval Research, as she recognised that the term *aerial photography* was no longer accurate to describe the many ways of imagery collected using radiation outside the visible region of the spectrum. In 1972, the Landsat 1, the first Earth-orbiting satellite designed for the observation of the Earth's surface was launched, which provided enough detail to practical applications in several fields. During 1970-1980s, there were rapid advances in digital image processing. Also, the first hyperspectral sensors were introduced by scientists at the Jet Propulsion Laboratory, which could capture a high number of spectral regions. By the late 1990s, remote sensing was being applied in practice at a global scale. This was possible thanks to the launch of the first commercial satellites capable of acquiring fine resolution imagery (e.g., IKONOS), and the launch of Terra-1 by NASA, the first satellite designed to acquire global coverage to monitor Earth's changes. In the 21st century, remote sensing switched from being used in the creation specialised products for other experts to being used in the creation of products for the use of the broader public (e.g., Google Maps).

Over the last two decades, one particular remote sensing technique has attracted the attention of the scientific community. This is LiDAR (Light Detection And Ranging), an active sensing technique capable of measuring the distance between the sensor device and the target object [121]. In this thesis, we study the use of LiDAR data for several applications.

1.1 LiDAR Technology

1.1.1 History

Because LiDAR uses a laser device, its origins are tied to the origins of the laser. The historical background of the laser dates back to the mid-1950s. Some researchers working independently developed the theoretical principles of the laser: Alexander Prokhorov and Nikolay Basov at the Lebedev Physical Institute (1955), Charles Townes and Arthur Schawlow at Bell Labs (1958), and Gordon Gould at Columbia University (1958). The first use of the term "laser" was coined in 1959 by Gordon Gould, and originated as an acronym for *Light Amplification by Stimulated Emission of Radiation* [99]. The patent for the laser was credited in

1960 to Charles Townes and Arthur Schawlow. However, none of them provided a complete blueprint for building a laser. The first laser was invented in 1960 by Theodore Maiman at the Hughes Aircraft Company, which consisted of a solid-state ruby laser that emitted pulses of collimated red light [157]. The company came up with the name “colidar”, for *COherent LIght Detection And Ranging*, derived from the term ”radar”, itself an acronym for *RAdio Detection And Ranging*. A working prototype was built in 1961, and the first commercial prototype was released in late 1962 under the name of Colidar Mark II [179]. The term ”lidar” was first mentioned in 1963 as a combination of ”light” and ”radar” [202], although, there is no consensus on capitalisation. In the following years, basic development was conducted. The technology became publicly known in 1971, during the Apollo 15 mission, when astronauts used it to measure the surface height and topography of the moon from orbit [1]. During the 1970s and 1980s, LiDAR attracted researchers from the environmental sciences while also gained commercial inertia in the construction and engineering sectors [180, 140]. However, the lack of various supporting technologies prevented its full exploitation for several years. The widespread use of LiDAR began in the mid-1990s with the general advancements in computer technology and the introduction of geo-positioning systems: Global Position System (GPS), Inertial Measurement Unit (IMU), Inertial Navigation System (INS). This lead to the release of the first commercially viable airborne LiDAR systems [21], also referred to as Airborne Laser Scanning (ALS). Shortly, researchers began to study the use of LiDAR for topographic mapping [136], and in only a few years, it became the prime method for the acquisition of Digital Elevation Models (DEMs) in several countries [21]. A wide range of applications emerged in the following years and the technology was further refined. During the last years, the Full Waveform (FWF) LiDAR systems [242] have being explored. These can provide a slightly denser point cloud and better range determination, as well as additional information [159]. More recently, the use of Single Photon LiDAR (SPL) and Geiger-Mode LiDAR (GML) are being investigated. Both have the advantage of potentially higher area coverage performance, but the SPL having an increased outlier rate and a lower ranging accuracy [161], and the GML having a higher systematic range error and concerns for its effectiveness in detecting partially occluded targets, such as bare earth under vegetation [253, 222, 233].

1.1.2 Fundamentals

In this section, we describe the fundamentals of the LiDAR technology: the laser ranging principle, the different laser scanning strategies, the process of combining laser data with other sensors to produce the georeferenced point cloud, and its format.

Laser Ranging

Distance measurements of high accuracy can be obtained with a laser-based instrument, commonly referred to as laser ranger or laser rangefinder, where the distance (range) is derived from a time measurement. Two main methods exist for the precise measurement of time:

1. Time of Flight (TOF): The laser ranger emits a very short but intense pulse of light towards the object which reflects back to the sensor. The instrument measures the time interval elapsed between the pulse emitted by the sensor at point A and its return after its reflection from an object at point B (see Figure 1.1). A modulation signal in the form of a measuring wave pattern is superimposed on the carrier signal and its phase difference can be measured more precisely. The range is derived as:

$$R = c \cdot t/2 \tag{1.1}$$

where R is the range, c is the speed of light (ca. $3 \cdot 10^8$ m/s) and t is the elapsed time interval. The systems using this method are known as pulse-based or linear-mode LiDAR (LML) and they are the most widely used.

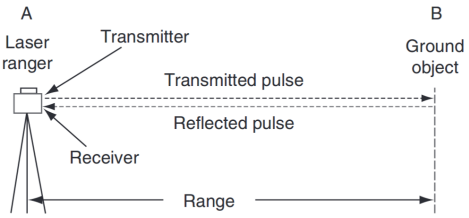


Figure 1.1: Basic operation of the laser rangefinder using the TOF method [216].

2. Phase difference: The laser transmits a continuous beam of light towards the object which reflects back to the sensor. The phase difference between the transmitted and

received versions of the sinusoidal wave patterns is measured (see Figure 1.2). The range is derived as:

$$R = (M\lambda + \Delta\lambda)/2 \tag{1.2}$$

where M is the number of wavelengths, λ is the known value of the wavelength, and $\Delta\lambda$ is the fractional part of the wavelength = $(\varphi/2\pi) \cdot \lambda$, where φ is the phase angle. The lasers used in this method are called continuous wave (CW) lasers. The systems using this method are known as phase-based LiDAR and they generally offer higher accuracy and faster acquisition speed at the cost of limited range and additional power needs.

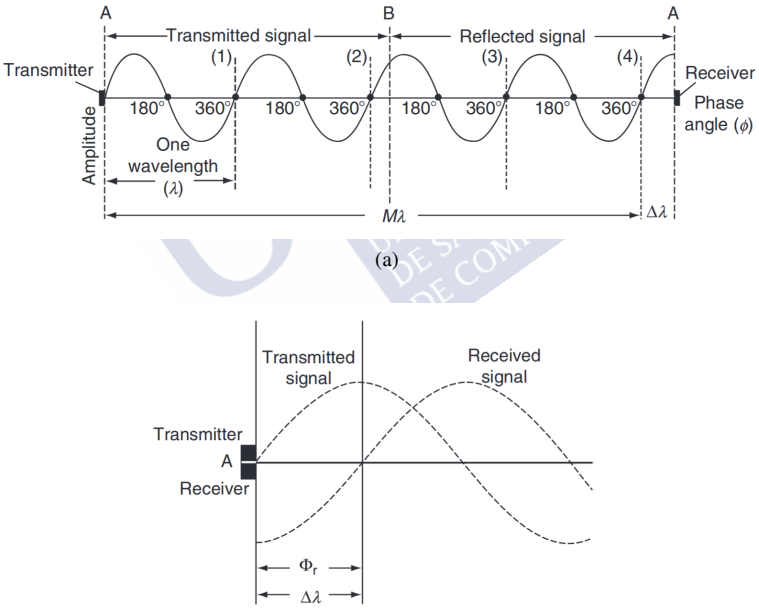


Figure 1.2: Comparison between the two signals using the Phase-based method [216].

Laser Scanning

Depending on the platform type, laser scanning can be categorised in:

- Terrestrial Laser Scanning (TLS): A stationary platform is used, typically a tripod. This method is mostly used for surveying and engineering tasks.
- Airborne Laser Scanning (ALS): A fixed-wing or rotor aircraft is used, such as an aeroplane or a helicopter. It is the preferred choice for topographic mapping of large areas.
- Mobile Laser Scanning (MLS): A ground-based vehicle is used, e.g. a car. It is mostly used in urban applications, particularly for transportation infrastructure mapping.
- Unmanned Laser Scanning (ULS): An Unmanned Aerial Vehicle (UAV) is used. It has a wide range of applications, especially, topographic mapping at a lower scale than ALS.

In moving platforms, across-track scanning is achieved by a deflection device that redirects the laser beam, while along-track scanning is achieved by the motion of the platform. In this thesis, we focus in aerial systems. Both ALS and ULS fall in this category. These systems are multi-sensor systems comprising three major components (see Figure 1.3):

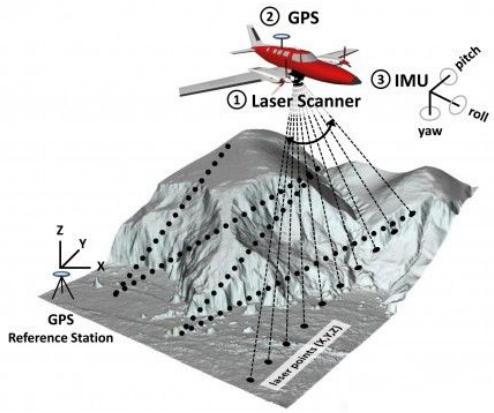


Figure 1.3: Main components of an ALS system [104].

1. A laser scanner, that is composed of the laser ranger unit and a beam deflection device. The laser ranger unit emits the light towards the deflection device. Most laser scanners emit light in the near-infrared, with typical wavelengths values of 905, 1064 and 1550 nm. In contrast, bathymetric laser scanners emit a green light of 532 nm to

penetrate the water. The deflection redirects the beam into the ground surface following a certain scanning pattern. Different deflection devices produce different patterns (see Figure 1.4). The oscillating mirror and the rotating polygon are the most common.

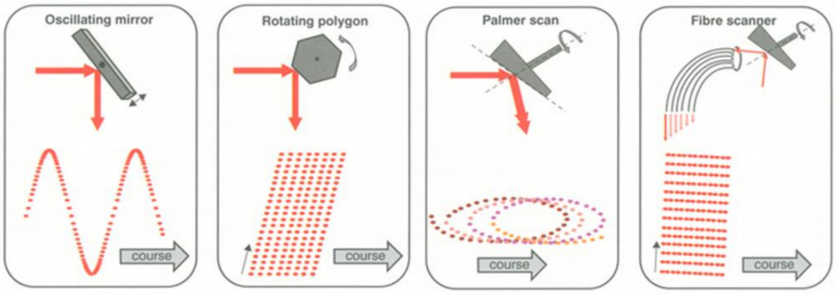


Figure 1.4: Deflection devices and resulting scanning patterns [240].

2. A Global Positioning System (GPS), or Global Navigation Satellite System (GNSS), generally running in differential mode, which is composed of a GPS receiver mounted in the aircraft and of a reference GPS ground base station.

3. An Inertial Measurement Unit (IMU), a device generally composed of three gyroscopes and three accelerometers. The gyroscopes measure the angular velocity, which describes the rotation of the aircraft (roll, pitch, yaw). The accelerometers measure to derive body acceleration. Typically, the IMU is positioned as close to the laser scanner as possible to record orientation and aircraft vibrations at its location.

Data Integration

The data from the three components must be integrated to produce the final point cloud. A typical workflow of this integration is shown in Figure 1.5.

The GPS data from the ground station and the aircraft receiver is corrected using a Differential Global Positioning System (DGPS), which allows eliminating pseudorange errors. The GPS signals from the satellites travel through the earth's atmosphere, so it is subjected to random delays that depend on the atmospheric conditions. Therefore, it is not possible to calculate the exact position at any given time, hence the name *pseudorange* instead of *range* (distance). In a differential system, the static base station has a known position, which is compared with the position obtained from the satellite signal. This difference is used to correct the

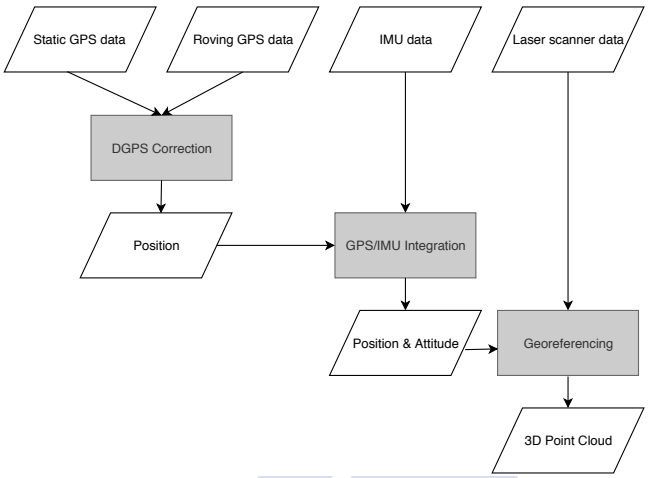


Figure 1.5: Workflow of data integration.

position of the roving receiver. The correction can be sent to the roving receiver in real-time by radio signal or it can be done after the data collection. This is only valid if both stations are close enough (< 25 km) to experiment similar atmospheric conditions [216].

The corrected GPS data is combined with the IMU data to estimate the trajectory of the aircraft. GPS/IMU is the most frequently used navigation system. GPS is widely adopted as an accurate method for providing the location. However, GPS signal is an electromagnetic signal that can be blocked, e.g., by mountains and buildings, hence GPS data is not available all the time. In contrast, IMU sensors provide position and altitude updates quicker than GPS, but they suffer from the imperfection of accelerometers and gyroscopes that cause navigation error growths over time [171], so they need to be calibrated using some type of non-inertial observations. Because of this, both types of sensors are used in a complementary way in a GPS/IMU system. The integration between GPS/IMU is done with a Kalman filter.

The Kalman filter (KF) is a set of mathematical equations that provides an efficient recursive procedure to estimate the state of a process, in a way that minimises the mean of the squared error [127, 27]. The KF is commonly used as an optimal estimator and compensator of the IMU/GPS system, that merges GPS position and velocity with IMU position, velocity and altitude to generate the aircraft trajectory. Several KF implementations exist: Linearised KF (LKF), often used with high accuracy sensors, extended KF (EKF), for better error control with medium accuracy sensors, and unscented KF (UKF), for bigger error handling with

low accuracy sensors [216]. The GPS/IMU systems can be tightly-coupled, which combines the raw data of GPS and IMU sensors in a single Kalman filter, or loosely-coupled, which simultaneously use two Kalman filters, one for each sensor, that interact occasionally. These systems are also known as centralised filter and decentralised filter, respectively. Tightly-coupled systems are preferred, as they are significantly more accurate, robust, jam-resistant, and less costly, at the cost of being more complex, inflexible, and hard to build [134, 80].

The trajectory of the aircraft is combined with the laser scanner data to derive the final point cloud. However, the GPS/IMU system, also referred to as Position Orientation System (POS), and the laser scanner has different time bases. The POS time is defined by the GPS time while the laser scanner time is defined by its internal clock, therefore, the data from both components must be synchronised. Little has been published about how commercial companies perform this synchronisation. Furthermore, the sampling rate of the laser scanner is much higher than the sampling rate of the POS, thus interpolation is required. Typically, linear interpolation is used to derive POS values for each laser pulse. After synchronisation and interpolation, data are available in the following possible arrangement [216]:

GPS time, x, y, z coordinates in WGS84, roll, pitch, yaw, range, intensity, scanning angle

These data are used to derive the geocoded 3D coordinates of each point in the process known as georeferencing. Georeferencing can be defined as the problem of transforming the laser scanner measurements into real-world coordinates. This process requires to combine data that have been acquired in three different reference frames (see Figure 1.6): The laser scanner frame (s-frame), the IMU frame (b-frame), and the mapping frame (m-frame). The method for georeferencing is well known (see Equations 1.3 and 1.4) [175, 214, 84]. The laser scanner records the distance and scanning angle in the s-frame (r_i^s). The position and orientation of the laser scanner respect to the m-frame are recorded by the navigation sensors (GPS/IMU). However, generally, navigation sensors are in different location and orientation than the laser scanner, so it is needed to apply small translations and rotations. The laser scanner coordinates are rotated to be aligned with the b-frame (R_s^b) and translated by the offset vector between the s-frame and the b-frame (a^b). The value of these transformations are constant and they are obtained during calibration. Finally, the coordinates are converted into the m-frame using the angular rotations of the aircraft ($R_b^m(t)$) and the positions from the GPS ($r_{IMU}^m(t)$).

$$r_i^m = r_{IMU}^m(t) + R_b^m(t)(R_s^b \cdot r_i^s + a^b) \tag{1.3}$$

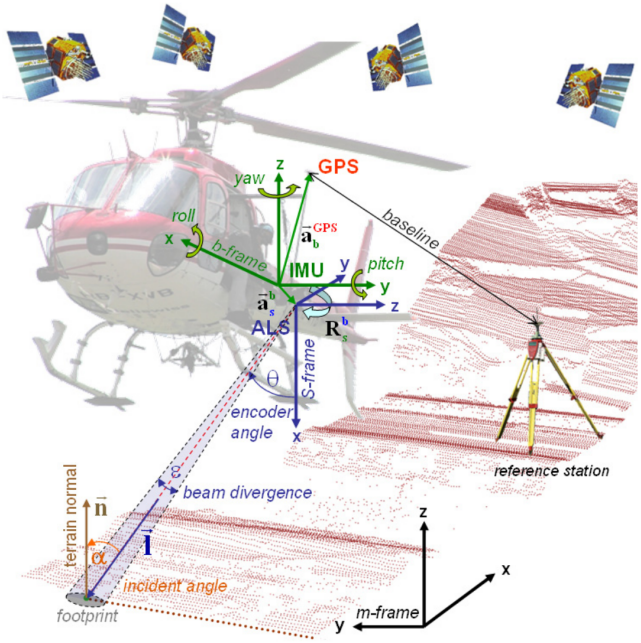


Figure 1.6: Reference frames of the ALS system [214].

where r_i^m is the coordinate vector of laser target (i) in the m-frame, t is the epoch of the laser measurement, $r_{IMU}^m(t)$ is the coordinate vector of the navigation sensor in the m-frame, $R_b^m(t)$ is the rotation from the b-frame to the m-frame, with the roll, pitch, and yaw angles, R_s^b is the rotation from the s-frame to the b-frame (boresight matrix), r_i^s is the coordinate vector of the laser target (i) in the s-frame and a^b is the coordinate vector offset between the s-frame and the b-frame (lever-arm offset). Equation 1.3 can be expanded to:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}^m = \begin{bmatrix} x \\ y \\ z \end{bmatrix}^m_{IMU} + R_b^m \left[R_s^b \begin{pmatrix} -d \cdot sin\alpha \\ 0 \\ -d \cdot cos\alpha \end{pmatrix} + \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}^b \right] \tag{1.4}$$

where d is the laser range in the s-frame and α is the scanner angle from nadir in the s-frame.

Data Format

The LAS (LASer) data format has become the standard format for storing and managing LiDAR data. LAS data format was created to provide an open format that allows different hardware and software tools to output data in a common format [74]. At the time of writing this document, the latest LAS specification was the version 1.4 R15 July 9, 2019 [227]. The format of a LAS file is shown in Table 1.1. It contains binary data consisting of a Public Header Block, any number of optional Variable Length Records (VLRs), the Point Data Records, and any number of optional Extended Variable Length Records (EVLRs). All data are in little-endian format.

Public Header Block
Variable Length Records (VLRs)
Point Data Records
Extended Variable Length Records (EVLRs)

Table 1.1: LAS 1.4 Format definition.

The Public Header Block contains generic data such as the number of points and points data bounds. The VLRs contain variable records of data including projection information, metadata, waveform information, and user application data, all limited to 65,535 bytes. The EVLRs are less limited than VLRs and can be appended to the end of a LAS file, allowing, for example, adding projection information without having to rewrite the entire file.

The start of the Point Data Records is defined in the *Offset to Point Data* field in the Public Header Block. There are ten Point Data Record Formats (PDRFs), all supported in the 1.4 specification. However, the preferred formats are 6-10 and previous PDRFs are designated as *legacy* formats. For simplicity, only the PDRF 6 is described (see Table 1.2). Later PDRFs include fields to define RGB colours (PDRF 8), near-infrared (NIR) channel (PDRF 9), and waveform data (wave packets) (PDRF 9-10). The PDRF 6 data fields are as follows:

- **X, Y, Z**: These records are used in conjunction with the scale and offset values in the Public Header Block to calculate the coordinate of each point as shown in Equation 1.5.

$$\begin{aligned} X_{coordinate} &= (X_{record} \cdot X_{scale}) + X_{offset} \\ Y_{coordinate} &= (Y_{record} \cdot Y_{scale}) + Y_{offset} \\ Z_{coordinate} &= (Z_{record} \cdot Z_{scale}) + Z_{offset} \end{aligned} \tag{1.5}$$

Field	Format	Size	Required
X	long	4 bytes	yes
Y	long	4 bytes	yes
Z	long	4 bytes	yes
Intensity	unsigned short	2 bytes	no
Return Number	4 bits (bits 0-3)	4 bits	yes
Number of Returns	4 bits (bits 4-7)	4 bits	yes
Classification Flags	4 bits (bits 0-3)	4 bits	no
Scanner Channel	2 bits (bits 4-5)	4 bits	yes
Scan Direction Flag	1 bit (bit 6)	1 bit	yes
Edge of Flight Line	1 bit (bit 7)	1 bit	yes
Classification	unsigned char	1 byte	yes
User Data	unsigned char	1 byte	no
Scan Angle	short	2 bytes	yes
Point Source ID	unsigned short	2 bytes	yes
GPS Time	double	8 bytes	yes
Minimum PDRF Size		30 bytes	

Table 1.2: PDRF 6. The fields that are bit fields are encoded into two bytes.

- **Intensity**: It is the integer representation of the pulse return magnitude. Intensity must be always normalised to 16 bit by multiplying the value by 65,536/(intensity dynamic range of the sensor) to ensure that data for different sensors can be correctly merged.
- **Return Number**: It is the pulse return number for a given pulse. A laser pulse can have many returns (commonly up to 5) that must be marked in the sequence of return.
- **Number of Returns**: It is the total number of returns for a given pulse.
- **Classification Flags**: It indicates special characteristics associated with the point. The bit definitions are shown in Table 1.3. These bits can be set in any combination.

Bit	Field Name	Description
0	Synthetic	If set, this point was created other than direct observation.
1	Key-Point	If set, this point is a model key-point not to be withheld.
2	Withheld	If set, this point should not be included in the processing.
3	Overlap	If set, this point is within the overlap region between flight strips.

Table 1.3: Classification Flags Bit Field Encoding (PDRF 6-10)

- **Scanner Channel**: It indicates the channel (scanner head) of a multi-channel system. Channel 0 is used for single scanner systems. Up to four channels are supported (0-3).
- **Scan Direction Flag**: It is the direction in which the deflection device (scanner mirror) was travelling at the time of the pulse. A value of 1 is a positive scan direction (from left to right) and a value of 0 is a negative direction (from right to left).
- **Edge of Flight Line**: It indicates when the point is at the end of a scan line (value of 1). It is the last point before the scanner mirror changes direction or its facet changes.
- **Classification**: It indicates the *class* of a point. Table 1.4 shows the standard values.

Value (Bits 0:4)	**Meaning**
0	Created, Never Classified
1	Unclassified (Unknown)
2	Ground
3	Low Vegetation
4	Medium Vegetation
5	High Vegetation
6	Building
7	Low Point (Noise)
8	*Reserved*
9	Water
10	Rail
11	Road Surface
12	*Reserved*
13	Wire - Guard (Shield)
14	Wire - Conductor (Phase)
15	Transmission Tower
16	Wire - Structure Connector
17	Bridge Deck
18	High Noise
19	Overhead Structure
20	Ignored Ground
21	Snow
22	Temporal Exclusion
23-63	*Reserved*
64-225	User Definable

Table 1.4: ASPRS Standard Point Classes (PDRF 6-10)

- **User Data**: This field may be used at the user's discretion.

- **Scan Angle**: It represents the rotational position of the laser pulse respect to the vertical dimension of the coordinate system of the data. In the data coordinate system, the 0.0 position starts at nadir and increments in a counter-clockwise rotation, as viewed from the reading of the sensor, facing in the along-track (forward) direction. Each increment (positive or negative) represents 0.006 degrees, up to 180 degrees (zenith).

- **Point Source ID**: It indicates the source from which the point originated. A source is typically defined as a grouping of temporally consistent data, such as a flight line.

- **GPS Time**: It is the time tag value at which the point was observed.

Postprocessing

The LiDAR data of an Area of Interest (AOI) are collected in flight lines or *strips* (see Figure 1.7) so that the geometry and radiometry across strips can differ. Although these discrepancies are less critical nowadays than with the first commercial systems, they are still frequent. Differences are caused both by errors in the instrumental chain of the system such as unsystematic gross errors (blunders), systematic errors and random errors (noise) and by changes during the observation, e.g., changes in the weather conditions, terrain relief or scanning angle. Because of this, a geometric adjustment and radiometric normalisation should be conducted on the LiDAR data before its delivering.

The goal of the geometric adjustment is to minimise the impact of the systematic errors in the LiDAR system, as gross errors are mostly discarded within the pre-processing of laser measurements, and random errors coming from instrument noise will always remain [76, 199]. The process consists in, first identifying the differences between adjacent strips, and then determining and applying a suitable transformation with a geometric model (see Figure 1.8). Existing methods can be categorised into data-driven and sensor system-driven. Data-driven methods generally apply translation and rotations models (e.g., six-parameter rigid-body transformation). However, errors can be attributed to the individual components of the system as well as to their integration, and some errors might be constant while others may vary over space and time [69]. Given the high sophistication of the systematic errors, the distortion of strips cannot simply be assumed as linear [156, 71]. Therefore, linear models such as translation and rotation cannot remove strip differences completely,

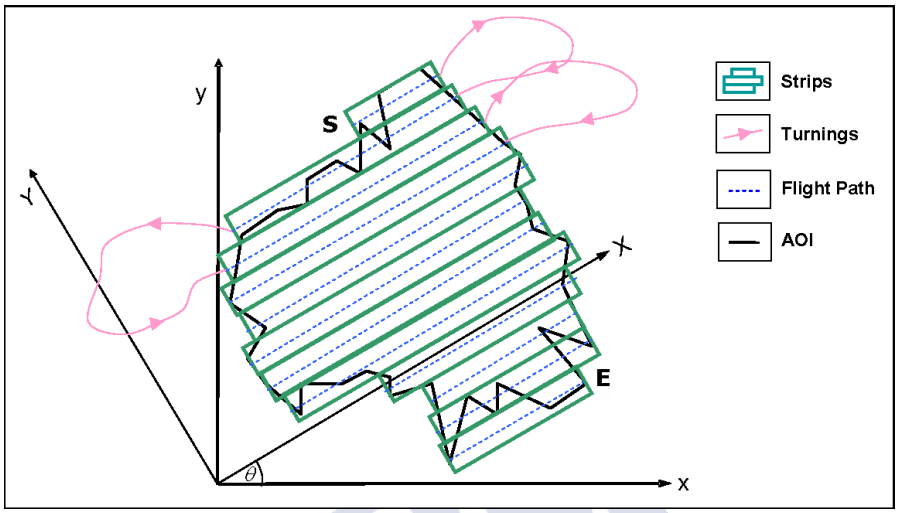


Figure 1.7: ALS Flight Planning.

and system-driven methods are required [275]. System-driven methods use LiDAR georeferencing equations as adjustment model so that they require the original sensor data (GPS, IMU, and laser scanner) or at least the trajectory and the time-tagged point clouds. These observations are usually not available to the end-user, thus the applicability of system-driven methods is limited [69]. Nonetheless, geometric differences can be minimised by following some guidelines when planning the flight, that would aid the adjustment methods. In particular, the overlap between strips must be in the 10%-30% range and one or more perpendicular strips, or *cross-strips*, must be flown [216].

The radiometric discrepancies can appear not only among different strips but within a single strip, for example, for a homogeneous surface, the intensity values recorded at nadir are usually higher than those captured along the swath edge. Therefore, radiometric correction is conducted first individually for each strip. Existing approaches can be categorised into theoretical models and empirical models [129]. Theoretical models are based on the radar (range) equation [121], and can compensate for variation in intensity caused by range, angle of incidence, atmospheric conditions, and Automatic Gain Control (AGC). The AGC is a closed-loop feedback circuit that aims to constraint the range of the recorded intensity within the radiometric resolution, e.g. [0, 255] for an 8-bit intensity data, by increasing the gain of very low intensity signals and decreasing the gain of very high-intensity signals. Generally,

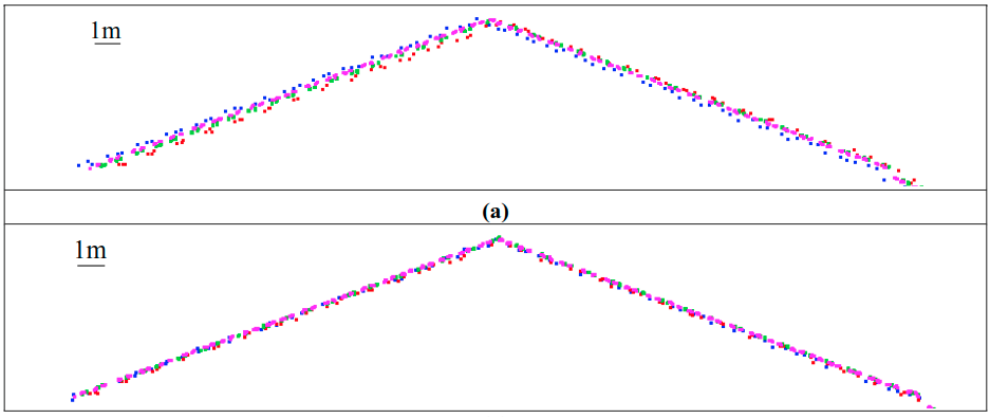


Figure 1.8: Profiles along the X direction over a building covered by strips 1 (in blue), 2 (in red), 3 (in green), and 4 (in magenta) (a) Before and (b) after the geometric adjustment [94].

only the range and angle of incidence are compensated. The main limitation of this model is that the applied radar equation is simplified so that it does not take into account non-extended targets, these are, targets smaller than the laser footprint (such as leaves and wires), in which the relationship between intensity and range is different: For extended targets, the intensity is a function of the inverse range squared ($1/R^2$) while for non-extended targets the intensity is a function of the inverse range with higher powers (e.g., $1/R^3$, $1/R^4$). Although detailed data about atmospheric conditions are generally not available, the atmospheric transmittance can be approximated by simulation models such as MODTRAN [221] or models based on the Beer-Lambert Law, in which the atmospheric transmittance is a function of range. Some systems also include an AGC module that helps to capture returns from low-reflectance surfaces, however, this can also change the recorded intensity even for within the same surface. Nevertheless, the effects of AGC cannot be perfectly modelled due to the nondisclosure of information by the sensor manufacturers [262]. Empirical models, in contrast, are based on data correlations of one or more variables (e.g., range). Therefore, these methods require homogeneous surfaces such as asphalt roads of roof planes, to be captured in multiple strips with varying settings (e.g., varying range). Data in these overlapping regions are used to estimate the constant parameters of the empirical correction function. In these methods, the sensor parameters related to the radar equation are not required to be known. After the radiometric correction of the individual strips, radiometric normalisation is conducted to reduce the dif-

ferences across strips (see Figure 1.9). However, little research has been carried out in this topic [263].

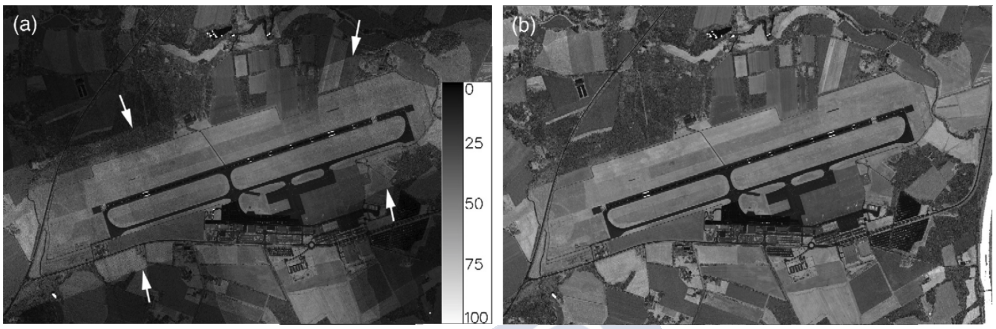


Figure 1.9: Averaged intensity rasters (gray-scaled), empty cells filled with a median filter; intensity a) before and b) after a data-driven radiometric normalisation [103].

1.1.3 Advantages

The first and most obvious advantage of LiDAR is that the data include elevation information (3D data). Furthermore, LiDAR also provides the reflectance intensity of the targets which provides good separability of ground materials with similar elevation [220]. Another relevant advantage is its ability to penetrate vegetation canopy. Although initially, LiDAR systems were single-return, most current systems are multiple-return. In multiple-return systems, the emitted beam is not completely blocked when strikes the target, and the remaining portion of the beam continues to the lower objects recording several returns (see Figure 1.10). This situation frequently occurs in forest canopies that have small gaps between branches and foliage [200]. This is particularly relevant to generate accurate Digital Terrain Models (DTMs) in densely forested areas. Another advantage is its flexibility to atmospheric conditions. LiDAR system tolerance for inclement weather conditions, such as rain, snow, fog, mist, high winds, and low cloud cover, is generally higher than that of other photogrammetric methods. Although, inclement weather conditions shall be avoided, as have been known to affect or degrade the accuracy of the LiDAR data [67, 178]. Furthermore, in contrast to passive sensors, data acquisition can be conducted regardless of sunlight conditions, and even at night.

In contrast, in photogrammetry, data acquisition is slower and it takes more time to process. Also, it is needed to establish Ground Control Points (GCP), which generally require

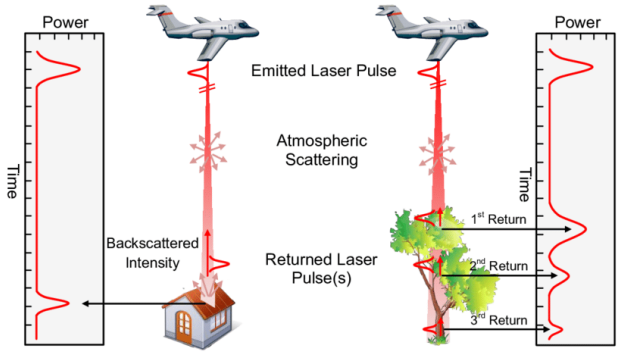


Figure 1.10: Illustration of the laser pulse return signal and the recorded signal strength [264].

traditional land surveying methods that add temporal and economic costs. Moreover, photogrammetry struggles to reveal the accurate terrain surface due to dense canopy cover [82].

Given the different characteristics of active and passive remote sensing techniques, one could argue that their combination will derive in a better description of the object space. Some authors have fused LiDAR data with aerial imagery data [264, 270], however, using several data sources increases costs and their integration is complex. Besides, in many cases, LiDAR data is the only input available. Therefore, it is interesting to explore and evaluate what can be achieved with LiDAR data only. In this thesis, we use LiDAR data exclusively and we explore the whole post-processing pipeline. We research both basic processing such as segmentation and classification, and application-specific processing such as building extraction and road point extraction. Also, we address the topic of near real-time ground filtering by exploiting the pattern of data acquisition. Furthermore, we develop an open-source visualisation tool for the examination and presentation of these and other algorithms results.

1.2 Computational Efficiency

One of the objectives of this thesis is to develop computationally efficient implementations. In this section, we describe the design choices that we made to achieve this, such as the selection of the programming language, the LiDAR data organisation and the software parallelisation.

Programming Language

To implement efficient software, a middle-to-low level language has been chosen. A low-level language is close to the machine level instruction set and provides less or no abstraction from the hardware. This has the advantage of providing a high level of control, leading to fast and memory-efficient software. However, the lack of operations to deal directly with composite objects (e.g: strings, sets, list) and built-in functions, make the software hard to write, debug and maintain, reducing programming productivity. Also, the developer must have additional knowledge of computer architecture to fully exploit the language. Thus, low-level languages are mostly used to develop performance-critical software, such as operating systems, device drivers, or applications that require direct hardware access.

Among the existing low-level languages, we have selected C. The C language is a general-purpose procedural programming language introduced in 1972 by the American computer scientist Dennis Ritchie at AT&T Bell Laboratories. C is a relatively low-level language, which means that it deals with the same sort of objects that most computers do, namely characters, numbers and memory addresses, which may be combined and moved about with the arithmetic and logical operators implemented by real machines, and provides control over the memory management with different allocation functions and pointers. C is one of the most recognisable languages, and it has been used to write the UNIX system and critical software at many organisations, including the NASA Jet Propulsion Laboratory (JPL) [105, 139].

The compiler also plays an important role in the performance of the code. We used the GNU Compiler Collection (GCC), which is the standard compiler for most projects related to GNU and Linux. GCC offers optimisation options to control various sorts of optimisations such as block reordering, function inlining or loop unrolling. Without any optimisation option, the goal of the compiler is to reduce the compilation time and to make debugging produce the expected results. Optimisation options can be turned on so that the compiler attempts to improve the code performance at the expense of compilation time and possibly the ability to debug the program [85]. From the existing three levels of optimisation, we use the second one (O2), which is the recommended level of optimisation. The highest level of optimisation (O3) enables optimisations that are expensive in terms of compile-time and memory usage, that in many cases, can slow down the program due to larger binaries and increased memory usage, and it is also associated, arguably, with causing instability, e.g. crashes or alteration of results.

Spatial Indexing

To access the data efficiently, a spatial data indexing is needed. Common structures include the R-tree [93], the kd-tree [25], and the octree [167]. While the implementations vary, their underlying principles are highly similar, as they all aim to create spatially organised structures of the data using spatial hierarchical partitioning. This allows for fast neighbours queries, which can be requested for a number of neighbours or a fixed radius.

In this work, we opt for using an octree. An octree is a tree structure in which each node has eight children, commonly referred to as octants. Conceptually, it is the generalisation of a quadtree [72], which stores two-dimensional data. The point cloud is thus recursively partitioned in the three-dimensional space until a stopping criterion is reached. A common stopping criterion is to set a maximum octant radius. However, because point density can greatly vary across point clouds, this can cause octants to hold a very different number of points which affects the reliability of the performance. A better criterion is to set a maximum number of points that an octant can hold, which is independent for the point density of the point cloud. In our case, we only allow octants to store 100 points. This organisation accelerates the lookup for 3D spatial queries significantly [176], as shown in Figure 1.11. Since spatial queries are a recurrent calculation, the overall performance increases considerably.

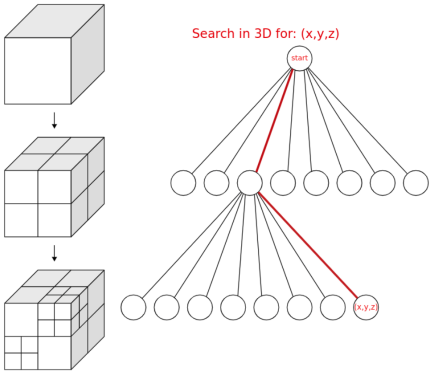


Figure 1.11: Recursive partition into octants. Left: 3D scheme. Right: 2D scheme [78].

Code Parallelisation

Since the commercialisation of the first dual-core processors in mid-2000s, multi-core processing has been gained adaption rapidly. A multi-core processor is a single integrated circuit that contains multiple Central Processing Units (CPU), more commonly known as *cores*. Nowadays, multi-core processors are present in most desktop and mobile devices, both in industry and retail. For example, the latest Intel's Skylake family desktop processors to date, Comet Lake, ranging from 2 to 10 cores [118], and the latest AMD's Zen family desktop processors, Zen 2, range from 4 to 64 cores [2]. Furthermore, many of the current processors support multi-threading, which allows the parallel execution of independent sub-processes by allocating two logical cores for each physical one. Given the number of available cores/threads in current processors, it is almost mandatory for any performance-oriented software to make of use of parallelisation techniques to fully make use of the processor processing power.

Because the software developed in this thesis is meant to be used in one single computer, we opt for using the Open Multi-Processing (OpenMP) application programming interface [188], which allows us to employ shared-memory multi-threading within the multi-core. Multi-threading is an execution model in which multiple threads exist within the context of one process, and the process resources are shared among the threads so that they execute independently to maximise the multi-core usage. Therefore, when we talk about parallelism, we refer to task parallelism, where tasks are concurrently performed by threads. OpenMP is an implementation of multi-threading and consists in a set of compiler pragmas, directives, function calls, and environment variables that explicitly instruct the compiler how and where to insert threads into the application [119]. This provides a certain level of abstraction so that the programmer can apply parallelisation strategies without requiring a deep knowledge on how to create, synchronise, and destroy threads.

Although using OpenMP eases the burden of parallelising software, some issues still need to be taken into account. First, tasks that are performed sequentially, such as iterative processes in which the results of one iteration depends on the results of the previous iteration are not suited, and sometimes impossible, to parallelise. Therefore, when selecting or designing algorithms, it is crucial to examine their tasks dependencies, and not only their time complexity. Also, this means that only some parts of the software will be parallelised so that the sequential regions of the runtime will be performed in one single thread and the parallel regions in several threads (see Figure 1.12). This can greatly limit the speedup that can be obtained for the overall program. Second, is up to the developer the select the best paral-

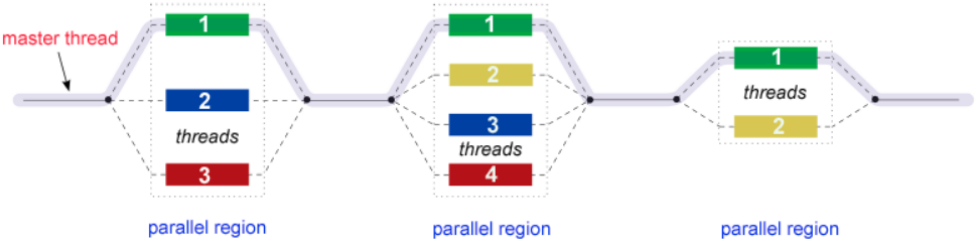


Figure 1.12: Example of multi-threading: Fork join model. Image from [28].

lelisation strategy depending on the runtime of the tasks. If the tasks to perform require the same runtime, it is evident to distribute each thread with the same amount of tasks, but if the tasks required different runtime, is better to distribute them dynamically. When processing LiDAR data, the runtime required for a task is expected to be directly related with the number of points to process, so it could be expected that point-level tasks require the same runtime, and object-level tasks require different runtime as the number of points of an object can vary. However, given the irregular distribution of the points, in practice, even point-level tasks might not have the same runtime, so that the distributing of tasks is not trivial. Third, parallelisation should be applied in specific parts of the software rather than systematically. The spawning of threads inside a parallel region has overhead, which might be bigger than the runtime required for the task if these are not computationally demanding. Also, gaining performance in parts of the program that do not represent a significant part of the overall runtime, contributes little to none to the overall speedup. Therefore, it is necessary to conduct a time profiling of the software and identify its bottlenecks in which the parallelisation efforts must be focused.

1.3 Objectives

This thesis aims to develop methods for the classification of LiDAR data. Specifically, for the identification of the four most basic classes: terrain, vegetation, buildings and roads, using LiDAR as the only data source. It could be expected that the use of additional data sources will improve the quality of the results. However, this would increase the cost of data acquisition, add the complexity of data integration, and in many cases, it is simply not available. There

are two main concerns in the development of solutions for the classification of LiDAR data:

- Design of highly automated methods. Methods should require minimal manual intervention. Ideally, the results should be generalisable, without requiring specific knowledge or parameter adjustment by the user. Therefore, it is preferred to investigate methodologies that do not rely on machine learning. These techniques have proven their effectiveness, however, they require a training phase for which reference data is needed. This has several implications. First, the creation of reference data has large financial and time costs, so generally, only a small amount is available. And second, since point clouds can have very different characteristics (e.g. point density, vertical precision, radiometric range, etc.), it is difficult to guarantee that good results will be obtained for a point cloud different from the ones that come from the training data.

- Development of computationally efficient implementations. One characteristic of LiDAR data processing is its high computational demand. In an image, the pixels are arranged in a regular mesh. However, in LiDAR data, the cloud points are unevenly distributed, as well as having an additional dimension, which adds computational complexity in comparison with image processing. Given that LiDAR data acquisition techniques are in continuous development, and the trend is to obtain increasing amounts of data, the solutions that are implemented must be computationally efficient. In this regard, the feasibility of real-time processing should also be explored.

1.4 Thesis Outline

The rest of the work presented in this thesis is organised as follows:

- Chapter 2: Describes region growing based segmentation method that introduces the concept of epicenter to allow regions to expand without losing homogeneity.

- Chapter 3: Describes a rule-based classifier that uses the aforementioned segmentation as input to label LiDAR points into three classes: ground, vegetation, and buildings.

- Chapter 4: Describes a building extraction method based on plane fitting and clustering.

- Chapter 5: Describes a road extraction method based on skewness balancing capable of identifying road points in sites with different degrees of urbanisation.

- Chapter 6: Describes a road modelling method based on road tracking, that is capable of extracting the road centreline, filling the possible gaps and deriving the road width.
- Chapter 7: Describes a scan line based ground filtering method that process the data as it is being acquired so that it can be used in near real-time environments.
- Chapter 8: Describes an open source visualisation tool implemented in Java that helps to understand, evaluate and present point cloud processing algorithms.

1.5 Contribution

The main contributions of this PhD dissertation are included in the following items:

Articles in peer reviewed journals:

- Martínez Sánchez J, Fernández Rivera F, Cabaleiro Domínguez JC, López Vilariño D, Fernández Pena T, "Automatic Extraction of Road Points from Airborne LiDAR Based on Bidirectional Skewness Balancing", *Remote Sensing*, vol. 12, no. 12:2025, 2020.
  Impact factor (JCR 2019): 4.509. Q2.
  Category: REMOTE SENSING. Rank: **9/30**.
  Impact factor (SJR 2019): 1.422. Q1.
  Category: EARTH AND PLANETARY SCIENCES. Rank: **140/1375**.
- Martínez Sánchez J, Váquez Álvarez Á, López Vilariño D, Fernández Rivera F, Cabaleiro Domínguez JC, Fernández Pena T, "Fast Ground Filtering of Airborne LiDAR Data Based on Iterative Scan-Line Spline Interpolation", *Remote Sensing*, vol. 11, no. 19, pp. 2256-2279, 2019.
  Impact factor (JCR 2019): 4.509. Q2.
  Category: REMOTE SENSING. Rank: **9/30**.
  Impact factor (SJR 2019): 1.422. Q1.
  Category: EARTH AND PLANETARY SCIENCES. Rank: **140/1375**.
- J. Martínez, O. G. Lorenzo, D. L. Vilariño, T. F. Pena, J. C. Cabaleiro and F. F. Rivera, "A Developer-Friendly Open Lidar Visualiser and Analyser for Point Clouds With 3D Stereoscopic View", *IEEE Access*, vol. 6, pp. 63813-63822, 2018.
  Impact factor (JCR 2018): 4.098. Q1.

Category: COMPUTER SCIENCE, INFORMATION SYSTEMS. Rank: **23/155**.
Impact factor (SJR 2018): 0.609. Q1.
Category: COMPUTER SCIENCE. Rank: **523/5690**.

- David L. Vilariño, José C. Cabaleiro, Jorge Martínez, Francisco F. Rivera,and Tomás F. Pena, "Graph-based approach for airborne light detection and ranging segmentation", *Journal of Applied Remote Sensing (JARS)*, vol. 11, no. 1, pp. 1-13, 2017.
  Impact factor (JCR 2017): 0.976. Q4
  Category: IMAGING SCIENCE & PHOTOGRAPHIC TECHNOLOGY. Rank: **22/27**.
  Impact factor (SJR 2017): 0.441. Q2.
  Category: EARTH AND PLANETARY SCIENCES: **567/1605**.

Articles published in international conferences:

- Yermo, M., Martínez, J., Lorenzo, O. G., Vilariño, D. L., Cabaleiro, J. C., Pena, T. F., and Rivera, F. F.: AUTOMATIC DETECTION AND CHARACTERISATION OF POWER LINES AND THEIR SURROUNDINGS USING LIDAR DATA, Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLII-2/W13, 1161–1168, 2019.

Articles published in national conferences:

- Álvaro Vázquez Álvarez, Jorge Martínez Sánchez, David López Vilariño, Francisco Fernández Rivera, José Carlos Cabaleiro y Tomás Fernández Pena, "Implementación de un algoritmo de filtrado de terreno a partir de datos LiDAR sobre SoC Zynq", in *XXX Jornadas de la Sociedad de Arquitectura y Tecnología de Computadores (SARTECO)*, pp. 577-583, September 2019.

- Francisco F. Rivera, Oscar G. Lorenzo, Jorge Martínez, Miguel Yermo, David L. Vilariño, Tomás F. Pena, and José C. Cabaleiro, "Procesamiento eficiente de nubes de puntos LiDAR aéreo para aplicaciones de caracterización del terreno", in *CivilDRON'19*, pp. 116-133, February 2019.

Intellectual property items:

- Jorge Martínez Sánchez, Óscar García Lorenzo, David López Vilariño, Tomás Fernández Pena, José Carlos Cabaleiro Domínguez, Francisco Fernández Rivera, "OLIVIA. Open LIdar VIsualiser and Analyser", Spanish Intellectual Property Registry identifier SC-249-2018, July 2018.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 2

SEGMENTATION

The segmentation is defined as the process of dividing a point cloud into a set of homogeneous regions, that is, each point inside a region present similar features. The goal is to partition the point cloud into regions that are more meaningful and easier to analyse. This procedure is often performed before classification, as it allows us to perform an object-based classification rather than a point-based classification. More details on this topic are given in the next chapter.

Performing a segmentation of the point cloud has two main advantages [166]: First, mitigates the so-called *salt and pepper* effect, in which single points are labelled differently than their surroundings, thus hindering the generation of homogeneous regions [31]. And second, integrates contextual information, so that the information derived from the segments are richer than from the individual points [252]. A good segmentation will create segments with a close relation to the real-world objects, so that segment features are often more representative for class characteristics than single point features. Other secondary advantages of using objects as base units are that the computational load of the posterior classification is reduced and the integration into a Geographic Information System (GIS) can be easier [96].

Segmentation, particularly image segmentation, has been studied in computer vision for several decades [192, 54]. This has propitiated the rapid adaptation of image segmentation methods into similar ones although using different data. Initially, these methods were adapted to close-range laser range images [106] and later adapted to LiDAR point clouds. Although the same concepts apply for images and point clouds, this adaptation is not straightforward. Consider for instance how to define the neighbourhood of a point. In an image, this is trivial, as each pixel has eight adjacent pixels, while in a point cloud, the points are irregularly dis-

tributed into the 3D space, so that the neighbourhood can be defined in several ways, such as the k-nearest neighbours or the neighbouring points inside sphere or cylinder centred in the query point (see Figure 2.1). This also increases the computational demand of the neighbour search.

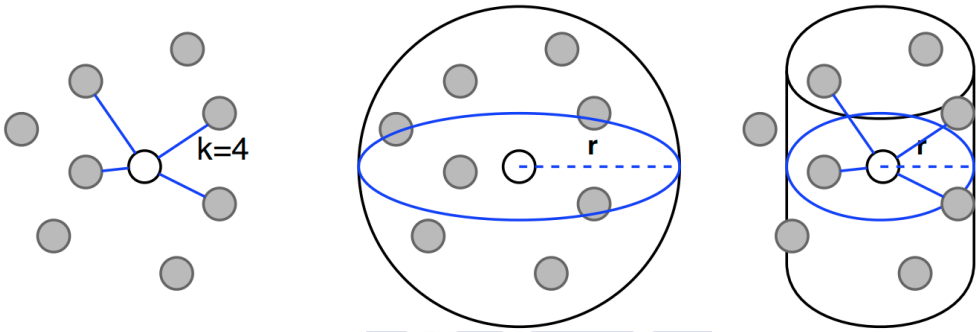


Figure 2.1: Neighbourhood models. From left to right: k-nearest, sphere, and cylinder.

Segmentation can be performed in different stages of the processing pipeline. It can be performed to filter ground points [168], to identify and characterise objects such as buildings [245] or trees [281], and to group the points before an object-based classification [218, 271, 125]. In this thesis, we focus on the later. The term of *object-based* originated from Object-based Image Analysis (OBIA). The first mentions to OBIA date back to early 2000, when some authors showed the advantages of OBIA over per-pixel methods. In point cloud analysis, however, the term Object-Based Point Cloud Analysis (OBPA) is used instead [208]. These methods are referred to as *object-oriented*, or more often as *object-based* [30].

Segmentation methods can be divided into five major categories: clustering, edge-based, model fitting, optimisation-based, and region segmentation [142, 89]. Clustering methods group the points into clusters based on certain precalculated local surface features. Common algorithms include fuzzy clustering, hierarchical clustering, K-means clustering, and mean shift. Edge-based methods consist in identifying the boundaries of the regions and grouping the points inside the boundaries. Common features for boundary identification are normal vectors, gradients, principal curvatures, and higher-order derivatives. Model fitting methods consist of fitting the point into a geometric primitives such as planes, cylinders, and spheres. The most representative algorithms in this category are RANSAC [73] and Hough Transform [59]. Optimisation-based methods formulate the segmentation as an optimisation

problem. This category includes level sets and graph-based strategies that determine the best graph cut. Region segmentation methods can be divided into seeded-region (bottom-up) and unseeded-region (top-down) methods. Furthermore, methods from different categories can be combined to balance their strengths and weaknesses, these are referred to as hybrid methods.

The region-growing segmentation follows a bottom-up strategy, so that the selected seed points, or initial regions, expand into their neighbouring points while preserving homogeneity. region-growing segmentation, therefore, consists in two steps: The selection of the seeds points and the definition of the growing criteria. The determination of the seed points can be done randomly or feature-based. Different seeds will lead to different segmentation results, which is seen as one of its main drawbacks. For the growing stage, commonly, similarity criteria are based on 3D spatial proximity, local planarity and normal vector smoothness.

Many region-growing segmentation methods have been proposed during the last decades. While initial seeded-region algorithm was introduced in late 80s [26], the first experiments with laser range images date back to early 2000 [135]. Shortly after, region-growing started being used to segment point clouds [141, 204, 237]. Generally, its application has always being related to the identification of planar surfaces, such as buildings [272, 193, 196, 208, 123], industrial machinery [198], ground surface [231] or glacier surface [101]. Also, region-growing segmentation is often performed as a previous step for classification [283, 271, 257].

In this chapter, we propose a general-purpose region-growing segmentation algorithm. A two-phase strategy allows to segment both planar and non-planar objects. Also, we introduce a new mechanism for the growing criteria, which we have called *epicentre*, to allow regions to expand without losing homogeneity.

The chapter is organised as follows. Section 2.1 introduces the two-phase region-growing segmentation method. The results are shown in Section 2.2. The conclusions are exposed in Section 2.3.

2.1 Method

The segmentation algorithm is based on well known region-growing strategies used in image processing. An overview of the proposed method is shown in Figure 2.2.

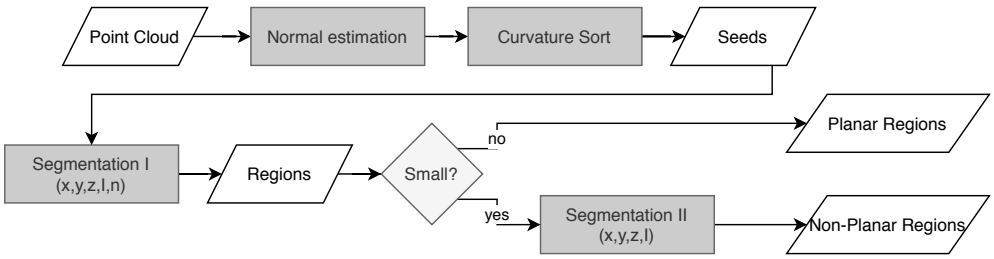


Figure 2.2: Segmentation overview.

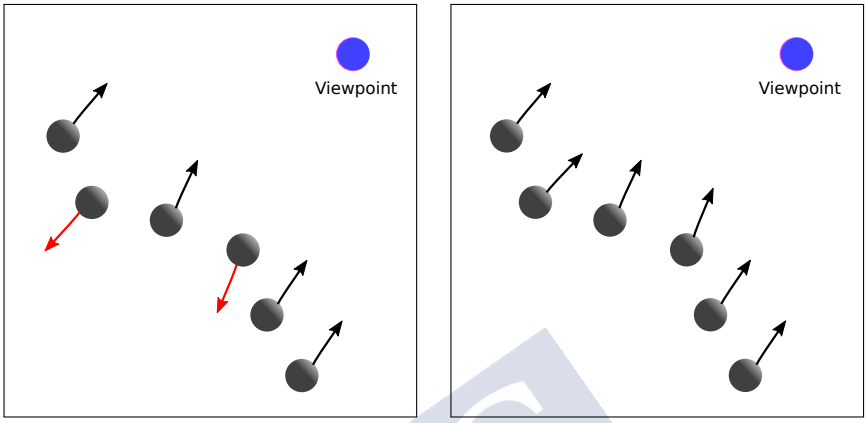
2.1.1 Normal Estimation

For estimating the normal vector of a point, first, a neighbourhood must be defined. A common neighbourhood model is a sphere with radius r. The length of r is selected based on the average point spacing (APS) of the point cloud. Experimentally, we found $r = 1.5 \cdot \text{APS}$ to be a satisfactory value. This guarantees a representative number of neighbours while avoiding too large neighbourhoods. Then the 3×3 covariance matrix of the neighbourhood is obtained. Their eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$ and eigenvectors $\{v_1, v_2, v_3\}$ can be obtained by Principal Component Analysis (PCA), so that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$, because it is symmetric positive-definite matrix. These eigenvalues describe the local geometry structure of the point set. The normal vector corresponds to the eigenvector that yields the smallest eigenvalue v_3.

One shortcoming of using PCA is that the signs of the eigenvectors are ambiguous, so that the resultant normal vectors are not consistently oriented. The determination of their orientations is not a trivial task in 3D scenes. Nevertheless, airborne LiDAR, does not generate a fully 3D scene, but 2.5D, as the points are acquired from a top-down viewpoint v. Therefore, the normal vectors must be oriented towards v. This correction is straightforward: the normal vector $\overrightarrow{n}$ of each point p has to satisfy that $\overrightarrow{n} \times (v - p) > 0$, otherwise its orientation is flipped (see Figure 2.3). Note that there is no need to know the real position of the viewpoint, it is enough with using an artificial point with an elevation higher than the maximum elevation of the point cloud.

2.1.2 Curvature Sort

The points are sorted by curvature in ascending order, so that the first seed is the point with the smallest curvature. This avoids the selection of points near the boundaries or noisy points,



(a) (b)

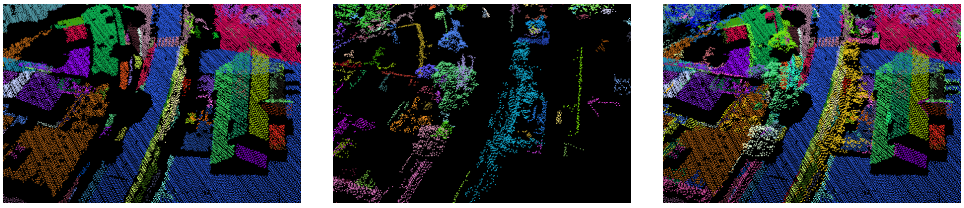
Figure 2.3: Normal vectors: (a) before and (b) after orientation correction. The wrongly oriented normal vectors (in red) are corrected using an artificial viewpoint (in blue).

thus improving the segmentation results and reducing the number of segments. The change in curvature can be derived from previously obtained eigenvalues as follows [250]:

$$C_\lambda = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \tag{2.1}$$

2.1.3 Segmentation

The segmentation consists in two region-growing phases that employ the same algorithm but with slightly different point features. On the first phase, the considered features are the spatial coordinates (x, y, z), the intensity (I), and the normal vector $(\overrightarrow{n})$. Using the normal vector causes planar objects to be grouped consistently into meaningful regions, while non-planar objects are grouped into small scattered regions due their disperse normal vectors. The points within these small regions correspond mostly with vegetation, irregular terrain, and corners of objects (e.g., roof joints). Small regions are filtered out and they are used as the input for the second region-growing phase, in which the normal vector is not considered. This way, planar and non-planar objects are segmented independently. The result of the segmentation is then the combination of the output from both procedures as shown in Figure 2.4.



(a) (b) (c)

Figure 2.4: Results of (a) the first region-growing procedure, (b) the second region-growing procedure, and (c) their combination. Regions are drawn in different colours.

Growing Criterion

One of the main characteristics of our proposal is the introduction of the epicentre of a region. The epicentre is an artificial point derived from the centroid of the region and the latest point added into the region. The epicentre is used to decide if a point is added or not into a region. Only points with similar feature values to the epicentre are added. This gives regions enough freedom to expand while maintaining enough homogeneity in a balanced way.

The concept is illustrated in Figure 2.5 where a 2D space is used for simplicity. Consider a region that consists of a set of LiDAR points (in red), and a candidate point (in black), for which is yet to be decided whether or not it is going to be added to the region. A traditional approach would be to compare the feature values of the candidate point with the centroid of the region (in big red). However, this might be too restrictive. In our approach, the candidate point is compared instead with the epicentre of the region (in blue), which is an intermediate artificial point between the centroid of the region and the last point added (in purple). Note that when a region is first created, the epicentre is equal to the seed point.

The value of the epicentre for a given feature x is defined in Equation 2.2, where $R_{\bar{x}}$ is the value of feature x in the centroid, that is the mean value of feature x for the points in the region, α is the weight of $R_{\bar{x}}$, and p_x is the feature value of the last point added to the region. A value $\alpha = 1$ is equal to only use the centroid values in the growing criteria, while smaller values will encourage the region to expand. The value selection of α is discussed in Section 2.2.

$$\text{epicentre}_x = R_{\bar{x}} * \alpha + p_x * (1 - \alpha) \tag{2.2}$$

The growing criterion is then as follows: for a candidate point to be added into a region, all

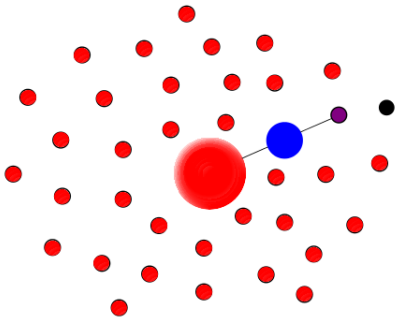


Figure 2.5: Visualisation of the concept of epicentre: region's points (in red), region's centroid (in big red), epicentre (in blue), last added point (in purple), candidate point (in black).

its feature differences with the epicentre must be smaller than the corresponding thresholds. In the first segmentation phase, the features used are the intensity and three components of the normal vector, while in the second segmentation phase, only the intensity feature is used.

Algorithm

The region-growing procedure is as follows.

1. Create a region R with the next seed point and initialise the epicentres.
2. For the current point, search its neighbours, sort them by ascending distance, and add each one into R if meets the growing criterion. Update the epicentre in each addition.
3. Get the next point in R and go to Step 2 until all points in R have been processed.
4. If in Phase 1 the area of R is lower than a threshold, filter out R and add it to the input for the Phase 2.
5. Go to Step 1 until all points in the point cloud have been processed.

2.2 Results

In this section, we present the dataset and parameters used for the experiments, and we evaluate the performance of the method both quantitatively, using available ground truth, and

qualitatively, using visual inspection. The computational performance is studied in the next chapter, as the segmentation is part of the classification framework.

2.2.1 Data and parameters

The Vaihingen dataset from the International Society for Photogrammetry and Remote Sensing (ISPRS) 3D Semantic Labelling Contest [50] was used for evaluation. This is the most widely adopted dataset in the scientific community. It was acquired in August 2008 by a Leica ALS50 system with a mean flying height of 500 m above ground and a 45° of field of view. The average strip overlap is 30% and the median point density is 6.7 p/m^2. Regions covered by only one strip have a mean point density of 4 p/m^2. Vaihingen is an urban area with the typical characteristics of an European city. The point cloud consists in 1.165.598 points, where each point has been labelled in one of the following nine classes: powerline, low vegetation, impervious surfaces, car, fence/hedge, roof, facade, shrub, and tree (see Figure 2.6).

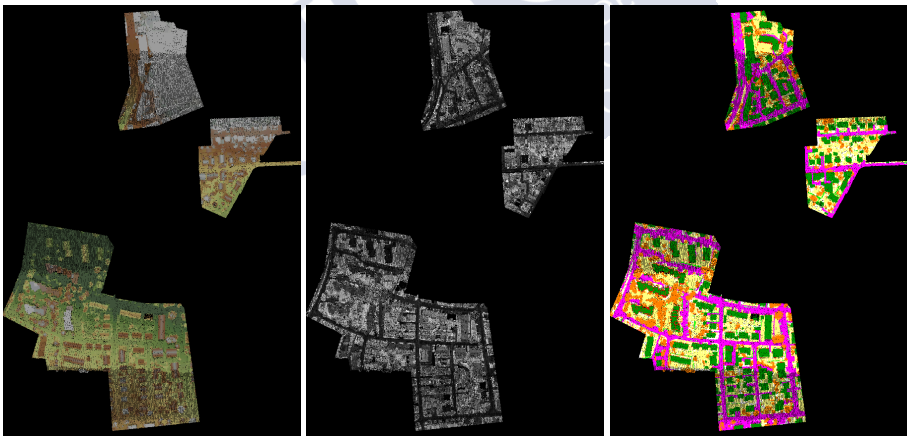


Figure 2.6: ISPRS dataset coloured by (a) height, (b) intensity, and (c) classification labels.

As other segmentation methods, it is needed to establish several parameters. In particular, this method requires five parameters. These parameters along with their suggested default values, which are the ones used in this evaluation, are shown in Table 2.1. The default values have been set experimentally, after showing satisfactory results with different datasets. Naturally, better results might be achieved with parameter tuning by the user for a particular dataset.

Parameter	Value
Sphere radius for normal estimation	$1.5 \cdot$ APS
Sphere radius for region-growing	$3 \cdot$ APS
Maximum intensity difference	7.5% of intensity range
Maximum angle difference	15^{o} for each axis
Minimum area region (non-small)	2 m^2

Table 2.1: Segmentation parameters and their default values.

Once the parameters have been set, we evaluate how the use of the epicentre influences the performance of the segmentation, and we proceed to determine the best value of α in Equation 2.2.

2.2.2 Unsupervised evaluation

Extensive research has been conducting in the evaluation of segmentation methods, but it is still difficult to assess whether one algorithm produces more accurate segmentations than another [269]. Two main approaches exist: *supervised* and *unsupervised* evaluation. In a supervised evaluation the segmentation result is compared against a manually-segmented or pre-processed reference segmentation. This is a tedious task, which is infeasible in many vision applications, and the reference segmentation could be considered subjective. In an unsupervised evaluation, objective metrics are defined to measure the segmentation quality, without the need of human assistance. For these reasons, we opt for a unsupervised evaluation.

One widely accepted definition of what constitutes a good segmentation is one in which: (i) regions are uniform and homogeneous, (ii) regions are significantly different from neighbouring regions, (iii) regions have a simple interior without many holes, and (iv) regions have boundaries that are simple, not ragged, and spatially accurate [95]. However, only the first two criteria have been realistically applied in highly texture and natural images [269], so for remote sensing images, a good segmentation should, in theory, maximise intra-segment homogeneity and inter-segment heterogeneity [124]. Therefore, unsupervised methods generally consists in calculating intra- and inter-segment measurements for each segment [64, 132, 124].

The intra-segment goodness measure used is weighted variance (WV):

$$WV = \frac{\sum_{i=1}^{n} s_i * v_i}{\sum_{i=1}^{n} s_i} \tag{2.3}$$

where v_i is the variance and s_i is the number of points of the region. This formula puts more weight in larger regions to avoid instabilities caused by smaller regions. Low values indicate that the points inside each region are similar (high intra-segment homogeneity). The inter-segment heterogeneity measure used is the global Moran's I (MI):

$$MI = \frac{n\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(y_i-\bar{y})(y_j-\bar{y})}{\sum_{i=1}^{n}(y_i-\bar{y})^2\left(\sum_{i\neq j}\sum w_{ij}\right)} \tag{2.4}$$

where n is the total number of regions, w_{ij} is a measure of the spatial proximity, y_i is the mean feature value of region R_i and $\bar{y}$ is the mean feature value of the point cloud. Each weight w_{ij} is a measure of the spatial adjacency of regions R_i and R_j. If regions R_i and R_j are adjacent, then $w_{ij} = 1$, otherwise $w_{ij} = 0$. The Moran's I is a spatial autocorrelation index that determines how, on average, a region is similar to its neighbouring regions. Small values indicate low spatial autocorrelation, this is, neighbouring regions are statistically different (high inter-segment heterogeneity). This is associated with regions showing clear boundaries.

An optimal segmentation combines both high intra-segment homogeneity and inter-segment heterogeneity, that is, small values in both weighted variance and Moran's I. In order to consider both measures equally, they are normalised into the [0,1] range using the values from all performed segmentations using the normalisation in Equation 2.5:

$$F(x) = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.5}$$

Then, both measures are combined to assign a Global Score (GS) to each segmentation.

$$GS = WV_{norm} + MI_{norm} \tag{2.6}$$

where WV_{norm} is the normalised weighted variance and MI_{norm} is the normalised Moran's I. The best segmentation is defined as the segmentation yielding the smallest GS value.

For an image, these metrics are calculated from the grey or colour pixels values. Its equivalent in LiDAR are points intensities. Besides, LiDAR introduces an additional spatial dimension. Therefore, metrics are calculated for the intensity and height features. The results are shown in Table 2.2. The best segmentation is obtained when using an epicentre with $\alpha = 0.3$. The segmentation's GS was reduced from 1.053 to 1.019 (lesser is better).

Features	Intensity					Height					Sum
Epi. α	WV	MI	WV_{norm}	MI_{norm}	GS	WV	MI	WV_{norm}	MI_{norm}	GS	GS
1.0	89.869	0.084	0.000	0.288	0.288	4.452	0.764	0.765	0.000	0.765	1.053
0.9	100.848	0.083	0.013	0.263	0.277	4.641	0.769	0.813	0.111	0.924	1.200
0.8	116.131	0.083	0.031	0.243	0.274	4.855	0.767	0.868	0.080	0.948	1.222
0.7	135.806	0.080	0.055	0.140	0.195	5.028	0.772	0.912	0.181	1.093	1.288
0.6	161.331	0.082	0.086	0.210	0.296	5.178	0.766	0.950	0.061	1.012	1.307
0.5	195.119	0.081	0.126	0.185	0.311	5.373	0.768	1.000	0.092	1.092	1.403
0.4	241.066	0.080	0.181	0.111	0.292	3.791	0.771	0.596	0.176	0.773	1.065
0.3	**301.388**	**0.078**	**0.254**	**0.045**	**0.299**	**3.042**	**0.777**	**0.405**	**0.315**	**0.720**	**1.019**
0.2	390.158	0.078	0.360	0.037	0.397	2.060	0.796	0.155	0.755	0.910	1.307
0.1	572.795	0.077	0.579	0.000	0.579	1.872	0.794	0.107	0.720	0.827	1.406
0.0	923.671	0.101	1.000	1.000	2.000	1.454	0.806	0.000	1.000	1.000	3.000

Table 2.2: Results of the unsupervised metrics.

2.2.3 Classification-derived evaluation

Segmentation goodness metrics are important to differentiate between good and bad segmentations under a theoretical point of view. However, the segmentation is generally an intermediate step of pattern recognition applications (i.e. classification). Therefore, it is also important to assess how different segmentations affect the quality of the end applications.

Each performed segmentation is used as input to a rule-based classifier (see Chapter 3), and the resulting classification labels are compared to the labels of the Vaihingen ground-truth. The classes from the classifier does not match all classes present in this ground-truth so that labels must be adapted. The classifier identifies *ground*, *vegetation*, and *buildings*, having an additional class, *unknown*, for smalls region that cannot be categorised in any of the aforementioned classes. The classifier does not include the *powerline*, *car*, nor *fence/hence* classes, so those points are not taken into account in the evaluation. Furthermore, the classifier does not distinguish among different vegetation types neither building structures, so that *shrub* and *tree* have been merged into a single *vegetation* class as well as *roof* and *facade* have been merged into a single *building* class. The label adaptation is shown in Table 2.3 and the resulting ground-truth in Figure 2.7.

Ground-truth	Classifier
powerline	NA
low vegetation	ground
impervious surfaces	ground
car	NA
fence/hedge	NA
roof	building
facade	building
shrub	vegetation
tree	vegetation

Table 2.3: Label adaptation.

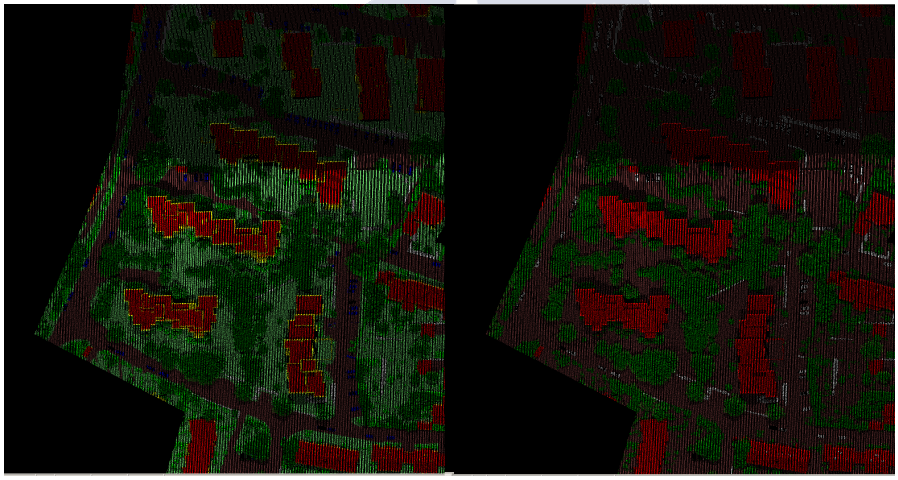


Figure 2.7: Label adaptation: Original ground-truth labels (left) and adapted labels (right).

Confusion matrices were calculated for each classification (from different segmentations) and both accuracy per class and overall accuracy was obtained. The results are shown in Table 2.4. The best classification is obtained when using an input segmentation with an epicentre with $\alpha = 0.2$. It can be observed how the use of the epicentre improves the classifications results. For the best segmentation, the overall classification accuracy is improved by 3.57%, with modest improvements in the *ground* (G) and *vegetation* (V) classes, but with a significant improvement of nearly 10% in the *building* (B) class. More noticeably, the overall accuracy

always improves, if the epicentre is used ($\alpha > 0$), regardless of the α value.

Also, note that using $\alpha = 0.3$, which obtained the best result in the previous unsupervised evaluation, obtains the second best result in this classification-derived evaluation.

	Accuracy				Accuracy improvement			
Epi. α	G	V	B	Overall	G	V	B	Overall
1.0	0.881	0.872	0.600	0.805	-	-	-	-
0.9	0.877	0.874	0.633	0.812	-0.47%	0.19%	3.34%	0.86%
0.8	0.881	0.872	0.652	0.819	-0.06%	-0.01%	5.23%	1.68%
0.7	0.891	0.874	0.666	0.828	0.94%	0.11%	6.61%	2.80%
0.6	0.893	0.871	0.671	0.829	1.18%	-0.14%	7.08%	3.03%
0.5	0.881	0.876	0.682	0.827	-0.02%	0.34%	8.25%	2.78%
0.4	0.884	0.870	0.695	0.831	0.26%	-0.26%	9.49%	3.19%
0.3	0.884	0.869	0.705	0.833	0.29%	-0.38%	10.46%	3.50%
0.2	**0.886**	**0.873**	**0.699**	**0.834**	**0.45%**	**0.10%**	**9.94%**	**3.57%**
0.1	0.880	0.873	0.707	0.833	-0.16%	0.07%	10.69%	3.42%
0.0	0.889	0.871	0.681	0.830	0.75%	0.17%	8.09%	3.08%

Table 2.4: Results of the classification-derived metrics.

2.2.4 Segmentation Goodness

Both metrics are combined to produce a single metric, namely Segmentation Goodness (SG):

$$SG = GS_{norm} + ACC_{norm} \tag{2.7}$$

where GS_{norm} is 1 minus the normalised GS score and ACC_{norm} is the normalised overall accuracy of the classification. The best segmentation is defined as the one yielding the largest SG. Table 2.5 shows the results of the metrics and their behaviour is shown in Figure 2.8. For the unsupervised metric, the goodness value is initially high when the epicentre is not used ($\alpha = 1$) and decreases slightly until increases around its peak at $\alpha = 0.3$, to later decrease abruptly, when the centroid of the region is ignored and only the features of the last point added into the region are taken into account ($\alpha = 0$). For the classification-derived metric, the accuracy is worse when the epicentre is not used ($\alpha = 1$), it increases significantly when using $\alpha \geq 0.7$, and reaches its peak at $\alpha = 0.2$, before decreasing slightly. Both metrics

Epi. α	GS_{norm}	ACC_{norm}	SG
1.0	0.983	0.000	0.983
0.9	0.909	0.240	1.149
0.8	0.898	0.470	1.368
0.7	0.864	0.784	1.648
0.6	0.855	0.850	1.705
0.5	0.806	0.780	1.587
0.4	0.977	0.895	1.872
0.3	**1.000**	**0.983**	**1.983**
0.2	0.855	1.000	1.855
0.1	0.805	0.958	1.763
0.0	0.000	0.864	0.864

Table 2.5: Normalised evaluation metrics and Segmentation Goodness.

show a small valley at $\alpha = 0.5$ and reach their peaks values so that $0.1 \leq \alpha \leq 0.4$. It can be observed clearly from both the table and the figure that the segmentation with an epicentre using $\alpha = 0.3$ yields the best results. Therefore, $\alpha = 0.3$ is set as the optimal value.

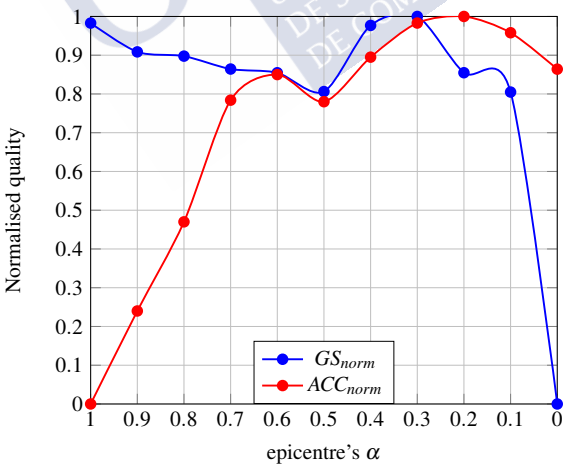


Figure 2.8: Influence of the epicentre on quality.

2.3 Conclusions

In this chapter, a region-growing method for the segmentation of LiDAR point clouds is presented. Its main feature is the introduction of the epicentre in the growing criteria. An artificial point with feature values between the centroid of the region and the last point added into the region, a trade-off that is regulated by a parameter α. The evaluation was carried out to measure the influence of using the epicentre, both from a theoretical point of view, using unsupervised segmentation metrics, and from a practical point for view, using classification-derived metrics. Experiments showed that the addition of the epicentre improves both the quality of the segmentation and the following classification. While improving the segmentation quality requires to select specifics values for α, classification accuracy always improves regardless of the value of α. The segmentation's global score was reduced by 3.23% (less is better), and the overall classification accuracy was increased by 3.57%, highlighting a building classification improvement of nearly 10%. Finally, both theoretical and practical metrics were combined into a segmentation goodness measure, and the optimal value for α was determined to be 0.3.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 3

CLASSIFICATION

Classification is the problem of assigning a class label to each LiDAR point. Depending on the goal of the task, the number of classes varies, usually between three and seven [264]. These classes may or may not correspond to those included in the LAS specification (see Section 1.1.2). Knowing the class composition of a landscape is crucial for many applications, such as change monitoring, city planning, and modelling or land mapping.

Several classification methods have been proposed in the literature (see Table 3.1). Initially, authors transformed LiDAR data into images and applied traditional pixel-based classification techniques. However, given the *salt-and-pepper* of the results, object-based classification is considered to be more appropriate. An object-based approach relies on hierarchical structures to classify the segmented objects. This approach has proven to outperform the traditional pixel-based classifiers, both in satellite images and in airborne LiDAR data, obtaining an overall accuracy over 80% [264].

Note that most of the existing classification methods make use of machine learning techniques, which require to supply training data, However, the need of labelled data introduces important implications. First, the creation of appropriate labelled data is a difficult task with high economical and temporal costs, so that only a small amount of reference data are usually available. And second, due to the heterogeneous characteristics of point cloud data, high performance on datasets different to those used for training is not guaranteed. Because of this, it is important to study which can be achieved without machine learning techniques. In this chapter, we introduce rule-based classifier that can distinguish between three classes: ground, buildings, and vegetation. The point cloud is first segmented using the method described in

the previous chapter so that the classification is performed at object-level. Next, elevation and planarity features are used as input into a set of heuristic rules to classify the segments into one of the aforementioned classes.

The chapter is organised as follows. Section 3.1 describes the rule-based classifier. Experiments are shown in Section 3.2 and discussed in Section 3.3. Conclusion are exposed in Section 3.4.

3.1 Method

An overview of the classification process is illustrated in Figure 3.1. First, the boundary points of each segment are extracted. Then, the height differences with the neighbouring segments at its boundary are examined. Next, the set of rules are applied to separate between the four classes. Finally, candidate building segments undergo an area filter.

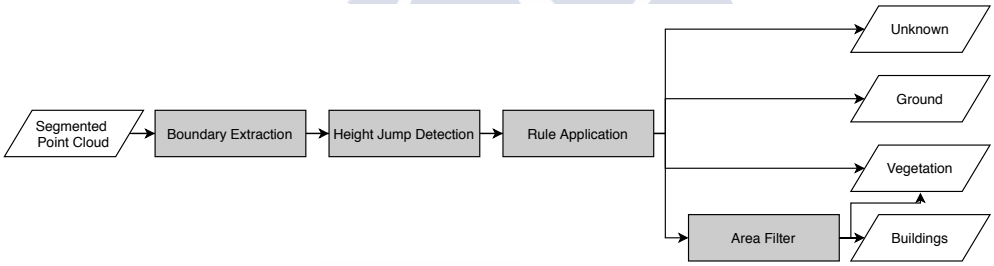


Figure 3.1: Overview of the classification strategy.

3.1.1 Boundary Extraction

To extract the boundary points, we use the Alpha Shapes algorithm [249]. An alpha shape, also known as α-shape, is a polytope that describes the shape of a point set [62]. The α-shape is uniquely determined by the point set and the parameter α, which controls the detail of the polytope. Conceptually, an α-shape is a generalisation of the convex hull. For $\alpha = \infty$ the α-shape is identical to the convex hull. When α decreases, the α-shape shrinks gradually smoothing the extracted outline. If the α value is optimal, the α-shape can extract the inner and outer outlines. However, and if α is too small, every point might be a boundary.

The alpha shapes algorithm is depicted in Figure 3.2. The principle consists on ”rolling” a circle around the point set. This circle is calculated based on two points $P_1(x_1, y_1)$ and

Table 3.1: Classification methods

Author	Category	Data	Classes	Technique	Features	Accuracy
Charaniya et al. (2004) [38]	Pixel-based	LiDAR + grey image	4:grass,trees,buildings,roads	mixture of Gaussian models	5	66-84%
Lodha et al. (2007) [150]	Pixel-based	LiDAR + grey image	4:grass,trees,buildings,roads	AdaBoost and SVM	5	94-98%
Mallet et al. (2008) [158]	Point-based	LiDAR FW	4:natural ground,vegetation, buildings,artificial ground	SVM	8	92%
Chehata et al. (2009) [39]	Pixel-based	LiDAR FW	4:natural ground,vegetation, buildings,artificial ground	Random Forest	17	94.35%
Alexander et al. (2010) [6]	Point-based	LiDAR FW	6:grass,shrub,tree,building (flat), building (pitched), road	Decision Tree	6	73-92%
Samadzadegan et al. (2010) [212]	Pixel-based	LiDAR	3:ground,tree,buildings	Multi-class SVM	10	91.66%
Niemeyer et al. (2011) [186]	Point-based	LiDAR FW	3:ground,vegetation,building	Conditional Random Fields	11	89-94%
Guo et al. (2011) [91]	Pixel-based	LiDAR FW + multispectral image	4:natural ground,vegetation, buildings,artificial ground	Random Forests	3-12	82-96%
Buján et al. (2012) [35]	Object-based	LiDAR + ortho-image	5:ground,high vegetation,low vegetation,buildings,road	Decision Tree	5	>96%
Chen and Gao (2014) [45]	Object-based	LiDAR	8:lawn,grass and crop,road and ground, shrub,tree,building,public infrastructure,water	Decision Tree	8	93.63%
Niemeyer et al. (2014) [184]	Point-based	LiDAR	7:grassland,low vegetation, tree, building (flat), building(gable),façade, road	Random Forest in Conditional Random Field	14	82.90%
Guo et al. (2015) [90]	Point-based	LiDAR	5:ground,vegetation,buildings, power-lines,pylons	JoinBoost	17	93.51%
Niemeyer et al. (2016) [185]	Point-based	LiDAR FW	6:road,natural soil,low vegetation, tree,building,car	Two-layer Conditional Random Field	12+15	81.10%
Vosselman et al. (2017) [239]	Object-based	LiDAR	7:ground,water,tree,building roof, roof structure,building facade, other	Conditional Random Field	19+12	91%
Ao et al. (2017) [10]	Point-based	LiDAR	5:terrain,tree,building, power-line,others	Back propagation neural network	11	96.97%
Yang et al. (2017) [266]	Point-based	LiDAR FW	9:powerline, low vegetation, impervious surfaces, car,fence/hedge, roof, façade,shrub,tree	Convolutional Neural Network	5	82.30%
Li et al. (2017) [143]	Point-based	LiDAR	5:open ground,covered ground, vegetation,roof,facade	Tensor-based sparse representation classification	18	82.16-91.85%
Wang et al. (2019) [243]	Pixel-based	LiDAR	4:ground,tree,building,car	Random Forest	3	87.2%
Zhao et al. (2018) [280]	Point-based	LiDAR FW	8:low vegetation, impervious surfaces, car,fence/hedge, roof, façade,shrub,tree	Multi-scale Convolutional Neural Network	3	85.20%
Yang et al. (2019) [265]	Point-based	LiDAR	8:flat roof, facade,grassland,tree, low vegetation,cars,road,gable roof	JointBoost GraphCut	16	81.6%

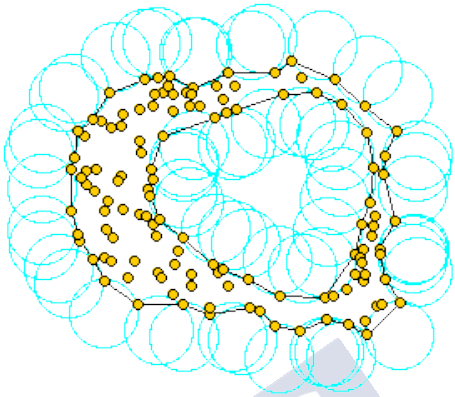


Figure 3.2: Alpha Shapes algorithm visualisation [249].

$P_2(x_2, y_2)$ and the radius α. The centre of the circle $P_0(x_0, y_0)$ is derived from the Distance Intersection Algorithm [277] shown in see Equation 3.1. If there are no points inside the circle, the P_1 and P_2 are considered boundary points, and they are connected to obtain a boundary line.

$$\begin{cases} x_0 = x_1 + \frac{1}{2}(x_2 - x_1) + H(y_2 - y_1) \\ y_0 = y_1 + \frac{1}{2}(y_2 - y_1) + H(x_1 - x_2) \end{cases} \tag{3.1}$$

where

$$H = \sqrt{\frac{\alpha^2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} - \frac{1}{4}}$$

For a point set S, the procedure is as follows:

1. Pick any point P_1 within S, and form a point set S_2 of size n with the points at a distance $D < 2\alpha$ (see Figure 3.3). Within S_2 pick any point P_2 and calculate the circle centre P_0.
2. Within S_2, calculate the distance between every point and P_0:
   - If $D_i (i = 1 \rightarrow n) \geq \alpha$, P_1 and P_2 are boundary points and P_1P_2 is a boundary segment (blue circles in the figure).
   - If $D_i < \alpha$, stop and go to Step 3 (red circles in the figure).

3. Within S_2, pick next point and repeat steps 1 and 2 until processing all points in S_2.

4. Within S, pick next point and repeat steps 1-3 until processing all points in S.

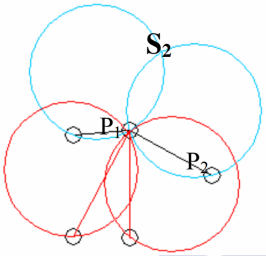
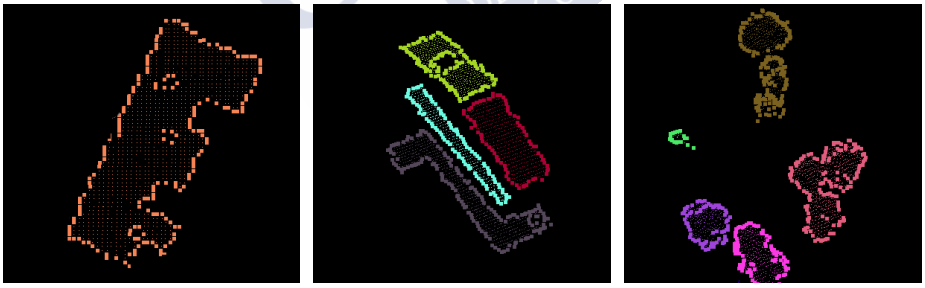


Figure 3.3: Alpha shapes algorithm judgement [249].

While for convex polygons α can be large, for concave polygons a value between the average points spacing (APS) and $2 \cdot$ APS is empathised [249]. Following this recommendation, $\alpha = 1.5 \cdot \text{APS}$ is used in our experiments, as well as other authors [97]. An example of the results of the extracted boundary points are shown in Figure 3.4.



(a) (b) (c)

Figure 3.4: Extracted boundaries for (a) single plane building, (b) a multi-plane building, and (c) trees. Points coloured by segment. Boundary points are highlighted with a bigger size.

3.1.2 Height Jump Detection

Once the boundary points are extracted, they are used to detect height differences between neighbouring segments. This allows the separation between ground and above ground seg-

ments. For point cloud split into a set of segments $S = \{S_i, S_{i+1} \dots n\}$, the process is as follows:

1. Create an octree containing only the boundary points of S. This will speed the neighbours queries in comparison with using the octree containing all points in S.

2. For each boundary point b of each segment S_i, perform the jump labelling:

   a) Calculate the mean Z of the points inside the neighbourhood of b in S_i ($\overline{Z_b}$). An infinite vertical cylinder is used as neighbourhood model (see Figure 3.5a).

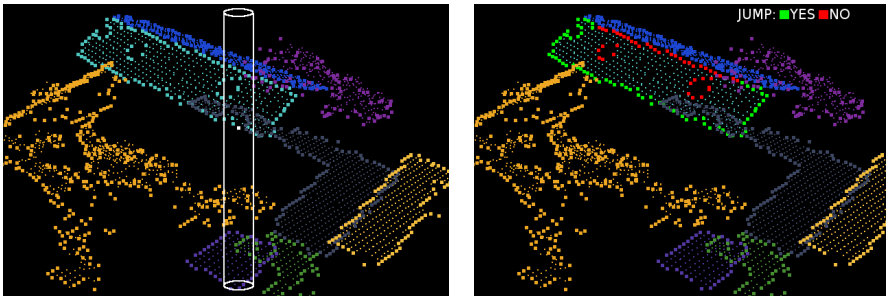
   b) Gather all boundary points inside the neighbourhood of b in $S \notin S_i$. From these points, the set of nearby segments to S_i, $N = \{N_j, N_{j+1} \dots m\}$, can be identified.

   c) For each nearby segment N_j, calculate the mean Z of the points inside the neighbourhood of b in N_j ($\overline{Z}_j$).

   d) For each nearby segment N_j, calculate the Z difference $ZD = \overline{Z_b} - \overline{Z}_j$

      - If $ZD > ZD_{max}$, label b as jump and return to Step 2.
      - Else, return to Step 2d.

3. After the jump labelling (see Figure 3.5b), each segment S_i with a percentage of boundary points labelled as jump higher than a threshold are labelled as above ground.



(a) (b)

Figure 3.5: Height Jump Detection (HJD) for a roof segment: (a) visualisation of the neighbour query for one boundary point and (b) resultant jump labels for every boundary point. Points coloured by region. Boundary points are highlighted in a bigger size. The limits of the vertical cylinder are shown only for visualisation purposes as it has infinite height.

3.1.3 Rule-based Classification

The classification consists of applying simple rules that are mainly based on height and planarity features. For the height, the percentage of boundary points labelled as jump, obtained by the previously described HJD algorithm, is used. For the planarity, it was previously obtained in the segmentation. As described in Chapter 2, the segmentation uses two phases, so that planar objects are segmented in the first phase and non-planar ones in the second phase. This way, it is straightforward to check whether a segment is planar. The flow of the rule-set application is shown in Figure 3.6. The flow is as follows: First, if the percentage of boundary points labelled as jump P is higher than a threshold P_{max}, the segment is assumed to be elevated over the ground. If the segment is elevated, planarity is examined to distinguish between *vegetation* (non-planar) and *building* (planar). In contrast, if the segment is not elevated, the number of neighbouring segments, derived by the HJD algorithm, is checked to distinguish between *ground* (non-isolated) and *unknown* (isolated).

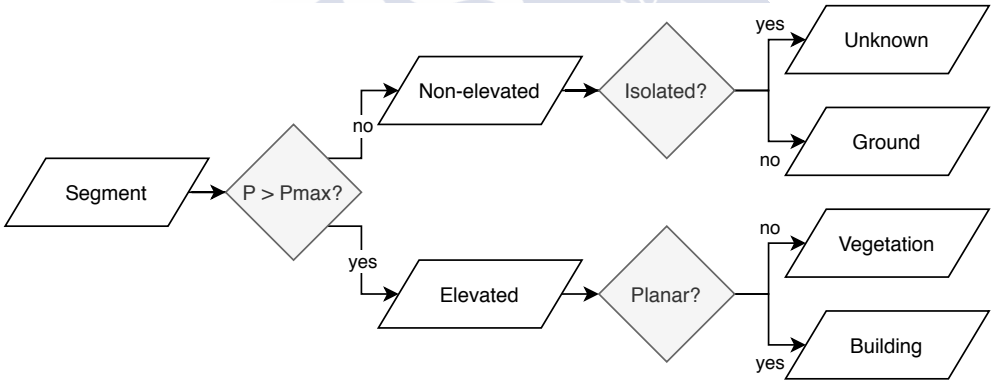


Figure 3.6: Rule-based classification flow.

3.1.4 Area Filtering

In order to remove false positives that mostly correspond to small flat vegetation surfaces, segments with an area smaller than a threshold are filtered out. The boundary points are used to derive the hull of the segments, and their area is calculated by the Shoelace method, also

known as the Gauss's area formula and the Surveyor's formula [32]:

$$\text{Area} = \frac{1}{2}\left|\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i + x_1 y_n\right| \tag{3.2}$$

3.2 Results

In Section 2.2.3 of the previous chapter, the classifier was tested in the Vaihingen dataset from the ISPRS 3D Semantic Labelling Contest [50]. Using the optimal segmentation's epicenter value ($\alpha = 0.3$), the classifier has an overall accuracy of 83.3%, with the *ground*, *vegetation* and *building* classes, yielding an accuracy of 88.4%, 86.9% and 70.5% respectively.

In this section, in order to further evaluate the performance of the classifier, we perform a quantitative evaluation of the classification based on the building extraction results, and we analyse and improve the computational performance.

3.2.1 Data and Parameters

Both urban and rural data are used. The urban data is the Vaihingen site described in the previous chapter. The rural data is the Alcoy site provided by the company Babcock International [19]. It corresponds to a Spanish city with an approximate area of 0.85×0.32 km^2 and 2.5M points, yielding a point density of 8.35 p/m^2. The site is characterised by having urbanised areas at each side, separated by a large and slopped forested area. One LiDAR strip for each site is used. The same parameters are used in both sites (see Table 3.2).

Parameter	**Value**
Cylinder radius for boundary extraction	$1.5 \cdot$ APS
Cylinder radius for jump labelling	$3.5 \cdot$ APS
Jump threshold	2 m
Jump percentage threshold	50%
Minimum area	15 m^2

Table 3.2: Classification parameters and their default values.

3.2.2 Quantitative Evaluation

As there is no available ground truth for these sites, we created it by manually labelling the buildings in the aerial images. Since Vaihingen is an urban environment with a high number of buildings, we only label and evaluate a representative part of the city of about 0.48×0.40 km^2 and 980,000 points. A total of 227 buildings were labelled in Vaihingen, and 66 in Alcoy. Small buildings with an area less than 15 m^2 are not considered in this study. An object-based evaluation is performed by comparing the labelled buildings in the aerial images to the classified building segments. Results are shown in figures 3.7 and 3.8. Note that because the date of the images and LiDAR acquisition is not the same, in Vaihingen, some buildings in the aerial images are not in the point cloud (shown in a white outline). Results show completeness of 96.46% and correctness of 98.68% in Vaihingen, and completeness of 100% and correctness of 84.85% in Alcoy. The main source of misclassifications on sudden terrain height changes, such as embankments, which can cause false positives.

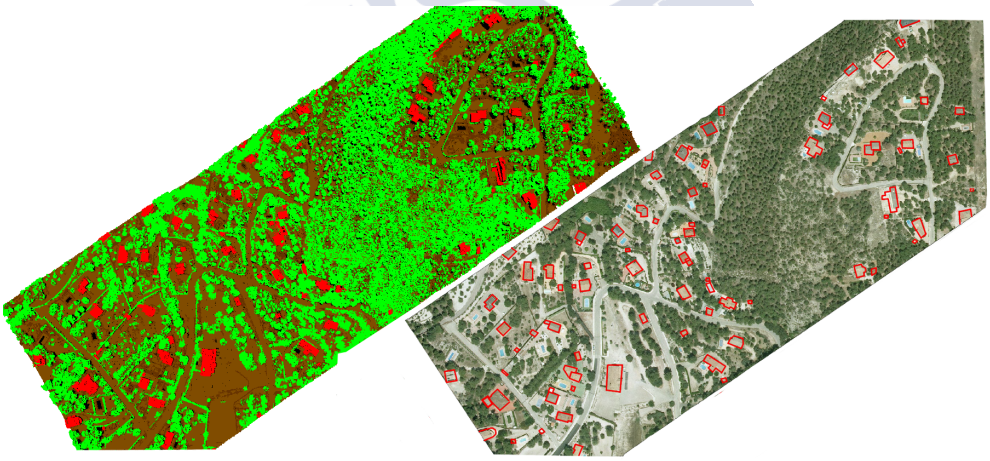
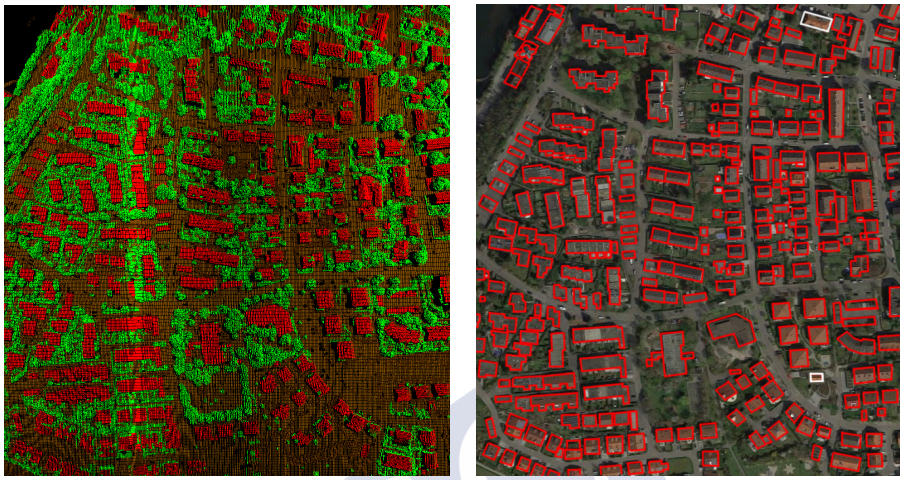


Figure 3.7: Alcoy: Classification results (left) and buildings ground truth (right.)

3.2.3 Computational Performance

The software was compiled in a Linux (Ubuntu 14.04.5 LTS) desktop computer with GCC version 4.8.4, in an Intel Core i7-4790 processor with 4 physical cores (up to 8 with Hyper-Threading) and 16 GB of RAM. This machine will be the one used in further chapters. The



(a) (b)

Figure 3.8: Vaihingen: Classification results (left) and buildings ground truth (right.)

classification framework encompasses the region-growing segmentation followed by the rule-based classification. From the computational point of view, the segmentation consists of two main stages: The estimation of the normal vector and the segmentation itself. The classification consists of three main stages: The boundary extraction, the height jump detection and the area estimation and filtering. The execution times of these stages are shown in Table 3.3. Note that some stages, such as the creation of the octree, or the application of the classification rules, are omitted, as their execution times are very low in comparison so they are not relevant for this study. Also, input and output operations are not taken into account.

To reduce the execution times, we apply parallelisation with OpenMP in all the stages but two: Segmentation and area estimation and filtering. In the case of the segmentation, it is a sequential procedure in which the current iteration depends on the results of the previous iteration, i.e. there are data dependencies. This makes it not suitable for parallelisation. This issue could be addressed by partitioning the data and segmenting the partitions independently. However, this would require a strategy for handling the overlapping areas and additional memory allocation, something we want to minimise to allow the processing of large datasets. In the case of the area estimation and filtering, the execution time is fairly low, so that the possible performance gain would be negligible.

Stage	Exec. Time (s)	Exec. Percentage	Paralellised
Normal calculation	6.8567	12.94%	✓
Segmentation	20.0168	37.77%	×
Boundary extraction	7.0531	13.31%	✓
Height jump detection	18.5206	34.94%	✓
Area estimation & filter	0.5523	1.04%	×
Overall	52.9995	100%	-

Table 3.3: Computational stages and their execution times.

The expected performance improvement can be predicted by Amdahl's Law, which defines that the theoretical speedup of the execution of a task is limited by the part of the task that is not improved. In our case, by improved we mean parallelised. The law's formula is [7]:

$$S = \frac{1}{(1-p) + \frac{p}{s}} \tag{3.3}$$

where S is the speedup of the whole task, s is the speedup of the part of the task that is improved, and p is the proportion of the execution time of the part to be improved concerning the whole task.

The parts of the program that are parallelised correspond to 61.19% of the execution time of the program. Assuming that the speedup in these parts would be equal to the number of physical cores of the tested computer, the expected theoretical speedup is given by:

$$S = \frac{1}{(1-0.6119) + \frac{0.6119}{4}} = 1.85 \tag{3.4}$$

Let's now discuss the parallelisation strategy. The normal calculation stage can only be parallelised at point-level as clusters are not available, but the boundary extraction and the height jump detection stages can be parallelised at cluster-level or point-level inside each cluster. Usually, larger granularity leads to better performance, so that cluster-level parallelisation is preferred. However, the cluster sizes can vary greatly (see Figure 3.9), causing load imbalance in this parallelisation approach. For a cluster-level parallelisation, consider for example the presence of a very large cluster. Its processing will be assigned to a single thread. It could happen that the thread is still processing the cluster while the other threads have already finished, wasting valuable computational resources. For a point-level parallelisation, in con-

trast, the large cluster will be assigned to all threads exploiting all computational resources, but in the small clusters, the time spent in the task is small and the overhead of managing the threads is significant in comparison. Because of this, we opt for a hybrid approach in which both parallelisation approaches are used so that clusters with very large sizes are parallelised at point-level and the rest of the clusters are parallelised at cluster-level. This way, the advantages of each approach are exploited. In the context of a moderate number of cores, we found experimentally that a good estimation to identify very large clusters, is to select those with a size beyond the 99th percentile.

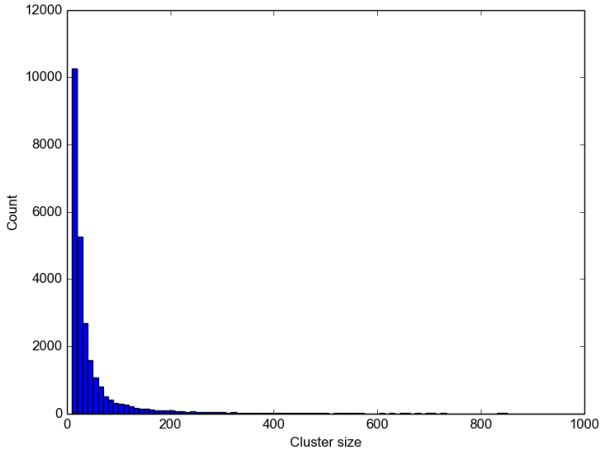


Figure 3.9: Histogram of cluster sizes. For visualisation purposes, only clusters with size in the [0,1000] range are shown. Clusters sizes in this example range from 1 to nearly 33,000.

Regarding the OpenMP loop scheduling, a *static* scheduling, in which each thread has assigned the same amount of iterations, is discarded. In a cluster-level parallelisation, clusters have different sizes, and in a point-level parallelisation, because the irregular distribution of the LiDAR points, each point would have a different amount of neighbours. Generally, this difference in the number of neighbours is not high in most of the point cloud, but it can be significant in some areas, such as in the point cloud edges, in the overlapped areas between flight scans or the presence of an abundance of non-first returns (e.g. forested areas). In both cases, the iterations of the loop have different execution times, therefore a static scheduling is not optimal. A better strategy is to use a *dynamic* scheduling, in which each thread requests

more iterations once it has processed the current ones so that iterations are assigned dynamically. This scheduling is better the higher the iterations differ in execution time. Because clusters have significantly different sizes, this approach is optimal, and it is the one used in the cluster-level parallelisation. However, there is some overhead associated with distributing the iterations dynamically. If each iteration does not have a significant execution time (i.e. high granularity), this overhead can be noticeable. Also, if the iterations do not have so different executions times, such fine-grained distribution is not needed. These two conditions apply to the point-level parallelisation. The overhead can be reduced by grouping the iterations into chunks, however, we find very challenging the selection of an appropriate chunk size for any dataset. Because of this, we opt for using a *guided* scheduling for the point-level parallelisation, which can be seen as a trade-off between the static and dynamic scheduling. Guided scheduling is similar to the dynamic scheduling, each thread requests a new chunk of iterations once it has processed the current chunk. The difference is in the size of the chunks: The size is proportional to the number of unassigned iterations divided by the number of the threads so that the size of the chunk decreases. The overhead of guided scheduling will be slightly higher per chunk than dynamic scheduling, as implies slightly more computation, but it will have less total dynamic chunks, resulting in a less overall overhead. A minimum chunk size can be set, in this thesis, we use the default value of 1.

The obtained execution times and speedups obtained after parallelisation are shown in Table 3.4. The execution time of the parallelised stages was reduced from 32.4304 s to 8.7688 s, this is a speedup of 3.70, while the execution time of the overall program was reduced from 52.9995s to 29.3379s. This is a speedup of 1.81, which is a satisfactory result, as it is very close to the theoretically achievable speedup calculated with Amdahl's Law of 1.85.

Stage	**Exec. Time (s) (Speedup)**				
Threads	**1**	**2**	**4**	**6**	**8**
Normal calculation	6.8567	4.3043 (1.59)	2.3017 (2.98)	2.1312 (3.22)	1.7457 (3.93)
Boundary extraction	7.0531	3.7397 (1.89)	2.2097 (3.19)	2.1274 (3.32)	2.0878 (3.38)
Height jump detection	18.5206	9.9711 (1.86)	6.2856 (2.95)	5.0864 (3.64)	4.9353 (3.75)
Overall	32.4304	18.0151 (1.80)	10.7970 (3.00)	9.3450 (3.47)	8.7688 (3.70)

Table 3.4: Execution times and speedups of the paralellised stages.

Also, we would like to emphasise the relevance of using an adequate spatial indexing strategy. The program was profiled with GProf [88], a performance analysis tool that calculates

the amount of time spent in each routine by statistical approximation. The tool introduces instrumentation code into the program automatically during compilation (using the -pg flag), and samples the value of the program counter at uniform intervals, allowing to infer the execution time from the distribution of the samples. As the classification framework is implemented in two executables, one for the segmentation and one the rule-based classification, profiling was done individually in each executable. Results are shown in tables 3.5 and 3.6. Tests were performed using one single thread. For simplicity, only routines with a percentage of the total execution time above 1% are shown. It can be seen how the routines related to the query of neighbours take the most time in both executables, accounting for 68.32% and 72.55% of the time for the segmentation and classification, respectively. Therefore, it is of critical relevance to use an efficient spatial indexing strategy as the octree-based indexing used in this thesis.

Time %	Cumulative Time (s)	Self Time (s)	Calls	Self Time/Call	Total Time/Call	Routine
36.55	**4.74**	**4.74**	**2500000**	**0.00**	**0.00**	**neighborsNBH**
31.77	**8.86**	**4.12**	**3843016**	**0.00**	**0.00**	**neighbors3D**
7.40	9.82	0.96	45773850	0.00	0.00	vectorAngles
4.51	10.41	0.59				cmpX
3.55	10.87	0.46	1	0.46	7.14	normalSegment
3.39	11.31	0.44	5186032	0.00	0.00	removeNode
1.85	11.55	0.24	1	0.24	1.88	secondSegment
1.54	11.75	0.20	4187811	0.00	0.00	insertPoint
1.27	11.91	0.17	86266738	0.00	0.00	onRange
1.12	12.06	0.15	22886925	0.00	0.00	onDegree
1.04	12.19	0.14	54967971	0.00	0.00	pointDist3D

Table 3.5: Segmentation GProf output. Routines for neighbour queries are highlighted.

3.3 Discussion

In this section, we compare our approach with other methods and we discuss different ways of benchmarking.

3.3.1 Comparison with Other Methods

Performance comparison of the proposed method with other existing methods is shown in Table 3.7. Ideally, our proposal should be compared only with classifiers than categorise the

Time %	Cumulative Time (s)	Self Time (s)	Calls	Self Time/Call	Total Time/Call	Routine
45.74	**9.92**	**9.92**	**21663459**	**0.00**	**0.00**	**neighbors2D**
23.12	**14.94**	**5.02**	**1483164128**	**0.00**	**0.00**	**insideCircle**
7.88	16.65	1.71	245220	0.01	0.04	alphaShapes
5.49	17.84	1.19	60648111	0.00	0.00	distInsersect
4.24	18.76	0.92	157080	0.01	0.04	AvgZSqeNeighs
3.69	**19.56**	**0.80**	**2588140**	**0.00**	**0.00**	**circleNeighbors**
2.35	20.07	0.51	10213331	0.00	0.00	avgSqZ
1.84	20.47	0.40	7607084	0.00	0.00	insertPoint

Table 3.6: Classification GProf output. Routines for neighbour queries are highlighted.

same three classes (*ground*, *vegetation* and *buildings*), however, as this will narrow the comparison, we opt to compare it with also classifiers that categorise four classes, in which the *ground* class is split between *natural soil* and *artificial soil*. It is important to note that this comparison is shown as a hint rather than a judgement, as first, the classifiers use slightly different data sources, and second, they use different datasets, therefore, accuracy values cannot be fairly compared. This issue is discussed in more detail in the next subsection. Our proposal achieves a satisfactory overall accuracy, above 80%, however, it is below the values obtained by existing methods, which are near 90%. Nevertheless, our proposal has the advantages of not requiring full-waveform LiDAR neither reference data for training.

Author	Data	Training	Classes	Accuracy	Test sites
Mallet et al. (2008) [158]	LiDAR FW	✓	4	92.00%	Biberach
Chehata et al. (2009) [39]	LiDAR FW	✓	4	94.35%	Biberach
Samadzadegan et al. (2010) [212]	LiDAR	✓	3	91.66%	Castrop-Rauxel
Niemeyer et al. (2011) [186]	LiDAR FW	✓	3	89-94%	Bonnland and Kiel
Guo et al. (2011) [91]	LiDAR	✓	4	92.00%	Biberach
Proposed method	LiDAR	×	3	83.30%	Vaihingen

Table 3.7: Performance comparison with other existing methods.

3.3.2 Classification Benchmarking

One of our main concerns within the field of LiDAR classification is the lack of a standard way to perform evaluations. This undermines the possibility to carry out a fair comparison between methods and it is hard to determine whether one method is better than others. Table 3.8 shows

how authors perform the evaluation differently. First, datasets vary among studies. Second, many of the studies created their reference data, also known as *ground truth*. This approach could be considered as biased since it is the same person whom both creates the reference data and performs the final test. Furthermore, the authors use different approaches to create reference data. One common way is to first use commercial software, such as TerraScan [226], to classify the point cloud and then correct the result manually [90, 10]. As commercial software is not always available, most authors choose to create the reference data manually, often with the aid of aerial photographs [91, 6] or field-collected data [45]. This manual labelling might also require the use of a Graphical User Interface (GUI) [38, 150] or software, e.g ArcGIS [11, 265]. In some cases, it is unclear if the authors only used the point cloud data for the manual labelling, which could be considered insufficient for the task. Even, some studies do not describe how reference data was created at all. Third, the size of the reference data varies greatly among studies. The sizes range from several points/pixels per class to the full point cloud. These inconsistencies in the evaluation also apply for other classification-related applications. For example, an exhaustive analysis of the state of the art in building extraction highlighted that the vast majority of approaches used their reference data to perform the evaluation [230]. All these issues are somehow understandable given the fact that generating reference data is a difficult and tedious task.

To address the problem of creating quality reference data, we distinguish three main approaches, in which reference data is created either by a trusted third party, by the collaboration of users (i.e. crowdsourcing) or by software (i.e. synthetic reference data). In the first case, a trusted third party or respected authority, such as a reputable surveying company, a government's cadastre department or a research organisation, ensures the production of accurate reference data. The most notable case of this approach is the Vaihingen dataset created by ISPRS 3D Semantic Labelling Contest [50], which has been introduced in Chapter 2. Since its creation in 2010, many studies have opted for using it for evaluation, as at current times it is one of the very few publicly available datasets with reference data. However, the amount of reference data is rather small, as only three areas of the dataset are provided and the point density is lower than current standards. Recently, the scientific community have become aware of these problems and more public datasets have been elaborated [284, 235], although due to their novelty, they are still to be adopted. In the second case, a set of users, which desirably are familiar with the field, make use of collaborative tools to create the reference data. The tool aids the interaction with the point cloud by proving a graphical interface supporting

Author	Test sites	Reference points (% of total points)
Charaniya et al. (2004) [38]	UCSC Campus (10 areas)	100k-150k per area (25-30%)
Lodha et al. (2007) [150]	UCSC Campus (10 areas)	100k-150k per area (25-30%)
Mallet et al. (2008) [158]	Biberach	135k (6%)
Chehata et al. (2009) [39]	Biberach	398k
Alexander et al. (2010) [6]	Bournemout	3.5k; in delineated polygons
Samadzadegan et al. (2010) [212]	Castrop-Rauxel	1.6k pixels
Niemeyer et al. (2011) [186]	Bonnland and Kiel	83k and 61k (33.3%)
Guo et al. (2011) [91]	Biberach	156k
Buján et al. (2012) [35]	Guitiriz	375 pixels; 75 per class
Chen and Gao (2014) [45]	Cambridge	500 >10 per class
Niemeyer et al. (2014) [184]	Vaihingen (3 areas)	780k (100%)
Guo et al. (2015) [90]	Hubei (2 areas)	0.5M and 5M (100%)
Niemeyer et al. (2016) [185]	Hannover	1M (48.2%)
Vosselman et al. (2017) [239]	Vaihingen (3 areas) and Rotterdam	780k and 35.6M (100% and 85.4%)
Ao et al. (2017) [10]	Santa Rosa	4.6M (100%)
Yang et al. (2017) [266]	Vaihingen (test area)	411k (100%)
Li et al. (2017) [143]	Vienna (8 areas)	3.5M (94-98%)
Zhao et al. (2018) [280]	Vaihingen (2 areas)	592k (100%)
Yang et al. (2019) [265]	Vaihingen (3 areas) and Toronto	1M and 2.7M (100%)

Table 3.8: Different methodologies to evaluate the classification accuracy.

features such as navigation and editing. An example of this approach is Supervisely's point cloud labelling tool [223]. Although crowdsourcing has been successful in related domains such as 2D labelling [189], for the time being, its adoption in point cloud labelling is still limited, which is expected, as the task of labelling 3D data needs more powerful tools and it demands more effort from users than labelling 2D. In the third case, a point cloud is generated by software simulation. This entails to supply a 3D model of the target scene and to simulate the sensing process to produce an artificial point cloud. If the 3D model contains semantic information, then reference data can be automatically generated. Since the mid-2000s, LiDAR simulators have been proposed [153, 151, 133, 247, 254], but some limitations exist: They are too application-specific (e.g. forestry focused), lack support for overhanging geometry (i.e. 2.5D geometry instead of 3D), lack support for airborne laser scanning, or do not simulate the beam divergence of the laser pulse. One simulator that addresses these issues is the Heidelberg LiDAR Operations Simulator (HELIOS) [23]. HELIOS is a software package for interactive real-time simulation and visualisation of terrestrial, mobile and airborne laser scanning surveys written in Java, that has been used in several studies [8]. We also have used HELIOS in our research, as shown in Chapter 7. At the time of writing, HELIOS is being ported into C++ with Python bindings. The advantages of software-generated reference data

are very compelling. First, reference data can be created automatically, cutting costs dramatically and providing the possibility of generating massive amounts. And second, but not less important, it removes labelling errors completely. Manual labelling is prone to human errors, and this chance of error increases as actual sensors acquire denser point clouds. Still, there are of course challenges: The creation of the scene's 3D model with semantic information is not straightforward and the simulation must be realistic for being considered for practical application. Nevertheless, we believe that this approach will attract most of the future research efforts for classification benchmarking.

3.4 Conclusions

In this chapter, a rule-based classification framework is presented. The classifier makes use of heuristic rules that are applied to the segmented objects to categorise them into three classes, *ground*, *vegetation* and *buildings*. Experiments carried out in urban and rural sites show that the method can achieve an overall accuracy above 80%. State-of-art classification methods achieve accuracy over 90%, however, they sometimes make use of additional data and often require reference data for training. Nevertheless, this comparison is not fair, as not the same benchmark has been used among the studies. The lack of a standard benchmark is a great concern in this research topic, and it is an issue still to be solved. With all that, we showed that satisfactory classification results can be achieved with LiDAR data alone, without full-waveform data neither reference data for training. Furthermore, the computational performance of the implementation was analysed and execution times were reduced by parallelisation.

CHAPTER 4

EXTRACTION OF BUILDING POINTS

In previous chapters we addressed basic processing steps, the segmentation and classification. From now on we focus on more specific applications, since narrowing down the problem domain usually is more computationally efficient. In this chapter, particularly, we address the identification of buildings. The location and features of buildings are crucial information in fields such as geographical condition monitoring, urban planning or surveying. For instance, unregistered properties are causing millionaire loses in property tax revenue to local governments. The Spanish cadastral regularisation program conducted from 2013 to 2018 found more than 3 million unregistered properties, this is 5% of the total registered properties, which leads to a loss of about 2,000 millions in revenue for the public coffers if adding the consequent revaluations, arrears and fines to the avoided taxes [56, 241]. While this program used satellite and aerial imagery, airborne LiDAR could be used instead.

Building extraction methods can be classified as supervised and unsupervised. Supervised methods include common machine learning techniques such as Support Vector Machine (SVM) [282, 271], Conditional Random Fields (CRF) [184] or Convolutional Neural Networks (CNNs) [160]. Supervised approaches have reported promising results, however, some challenges remain: many training samples are needed to precisely model the feature distribution to final target classes, which is a difficult task with high economical and temporal costs, and due to the heterogeneous characteristics of point cloud data, high quality on datasets different to those used for training is not guaranteed. Because of this, it is still relevant to explore the capabilities of unsupervised methods. In these approaches, the common strategy is to first separate between ground and non-ground points and then identify the building points within

the non-ground points. As ground filtering has been extensively studied and robust methods are available [168], the challenge resides in the identification of building points within the non-ground points. Several methods have been proposed over the years. Many approaches rely in clustering or segmentation based on plane fitting techniques [70, 63, 272, 42]. In [174] building segments are identified by examining the change in elevation obtained by Laplacian and Sobel operators. In [238, 191] methods are based on Hough Transform. In [43] a multiscale grid method is introduced using height and area thresholding and region growing. In [172] Differential Morphological Profiles (DMPs), a new algorithm for local fitting surfaces (LoFS), connected components, and contextual information are used to identify building points based on their geometric, surface, and regional attributes. In [58] an energy function is minimised by graph cuts, using the flatness, the variance of the normal vector direction, and the Gray Level Co-occurrence Matrix (GLCM) homogeneity of each grid cell to construct the data term, and the height difference to construct the smooth term. For an exhaustive review of the state of the art, we refer the reader to [230].

In this work, we present an unsupervised method for the extraction of building points, that can be categorised as plane-fitting clustering-based. An advantage of the method is its computational efficiency. The point-level processing filters out a large number of unsuitable points so that not all points in the point cloud need to be clustered. Furthermore, point-level processing can be fully parallelised in multi-core computers for additional throughput. The retained points are then used as input for the clustering stage and the object-level processing.

The rest of the chapter is organised as follows: Section 4.1 describes the building extraction method. Section 4.2 shows the results. The setting of parameters and some insight into the dimensionality filter are presented in Section 4.3. Conclusions are given in Section 4.4.

4.1 Method

An overview of the building extraction method is shown in Figure 4.1. This method requires to provide the DTM along with the point cloud as input. First, non-ground points are filtered based on point-level features and later refined with a morphological filter to identify candidate roof points. The candidate roof points are used as seeds for a plane clustering algorithm. Then, planes are merged to form the actual buildings candidates, which are filtered based on area.

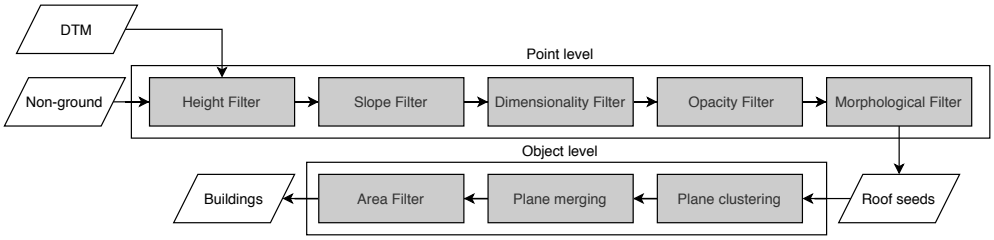


Figure 4.1: Overview of the building extraction.

4.1.1 Ground Filtering

The first step is to separate ground and non-ground points, namely ground-filtering. This is being a hot topic in the scientific community for the last two decades. Ground-filtering is directly related to the process of creating Digital Terrain Models (DTMs), which is the flagship of LiDAR-derived products. Many ground-filtering methods have been proposed based on different techniques (see Chapter 7). Although it is still challenging for one single method to obtain accurate results of every type of terrain, overall, satisfactory results have been achieved. In this work, we adopt the method proposed in [195], which is one of the top performers [274].

This method, called Simple Morphological Filter (SMRF) is based on applying image processing techniques from Mathematical Morphology. These operations have been extended in point clouds [131]. For a LiDAR point $p(x,y,z)$, the dilation of elevation z at (x,y) is defined as:

$$d_p = \max_{(x_p,y_p)\in w}(z_p) \tag{4.1}$$

where points (x_p,y_p,z_p) are the neighbours of p within a window w. This window, also referred as *structuring element* or *kernel*, can be a 1D line or 2D structure, e.g. a rectangle or a disk. The dilation output is then the maximum elevation value in the neighbourhood of p. Likewise, the erosion, which is the counterpart of dilation, is defined as:

$$d_p = \min_{(x_p,y_p)\in w}(z_p) \tag{4.2}$$

Dilation and erosion operations can be combined to produce *opening* and *closing* operations. The opening operation is achieved by performing an erosion followed by a dilation,

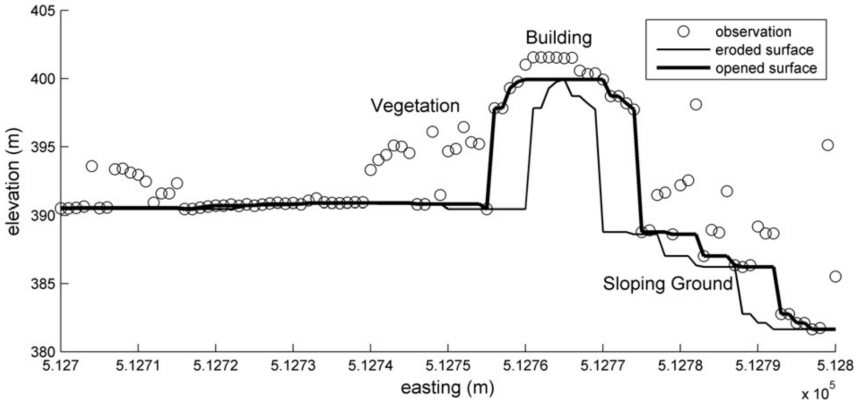


Figure 4.2: Visualisation of opening. The opening is used to remove all of the vegetation while preserving the structure of the included building as well as the sloping ground surface on the right. A ten meter linear structuring element was used. Image from [195].

while the closing operation is accomplished by carrying out a dilation followed by an erosion. The SMRF method relies on the opening operation, first, an erosion acts to snap relative high values to relative lows, and then, a dilation acts to outwardly expand relative highs (see Figure 4.2).

The SMRF method consists of four stages: Initially, the user supplies a cell size that is used to generate a grid over the data. First, a minimum surface is created by retaining the lowest point in each cell and inpainting the empty cells. Inpainting involves the replacement of the empty cell value with values calculated from other nearby cells. Second, a progressive morphological filter is performed in the minimum surface. This second stage is the heart of the algorithm. A set of window sizes is set based on a supplied maximum window size that corresponds to the largest feature to be removed. For each window size, a height threshold is derived from the current window size, the cell size, and a supplied slope threshold, and a morphological opening operation is applied to the surface using a disk-shaped structuring element. If the height difference between a new surface cell and a previous surface cell is higher than a threshold, the cell is labelled as non-bare earth. Third, the Digital Terrain Model (DTM) is created from the bare earth cells. Finally, the original LiDAR points are labelled into bare earth and non-bare earth based on their residuals to the interpolated DTM.

4.1.2 Point-level filtering

The non-ground points are used as input into a pipeline filtering based on height, slope, dimensionality, opacity and morphology. All non-ground points are used as input for the pipeline and the points passing all filters are considered as candidate building points.

Height Filter

First, points with a height below a threshold are filtered out. The height h is derived as:

$$h = z_p - \mathrm{z}_{dtm} \tag{4.3}$$

where z_p is the z value of the point and z_{dtm} the z value of the DTM at the (x,y) coordinates of the point. This height difference is sometimes referred to as normalised Digital Surface Model (nDSM), which equal to subtract the point cloud elevations (DSM) from the bare earth elevations (DTM).

Dimensionality Filter

Second, the points with a dimensionality of 3 are filtered out. The volumetric behaviour of a point, i.e., whether its neighbourhood is spread in one, two or three dimensions is obtained by the eigenvalues of the covariance matrix of its neighbourhood ($\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$). The degree of linear (a_{1D}), planar (a_{2D}), and scatter (a_{3D}) behaviours are measured with the indicators [52]:

$$a_{1D} = \frac{\sigma_1 - \sigma_2}{\sigma_3},\ a_{2D} = \frac{\sigma_2 - \sigma_3}{\sigma_3},\ a_{3D} = \frac{\sigma_3}{\sigma_3} \tag{4.4}$$

where $\sigma_i = \sqrt{\lambda_i}\ \forall i \in [1,3]$ and denotes the standard deviation along the corresponding eigenvector $\vec{v}_i$. This implies that $a_{1D}, a_{2D}, a_{3D} \in [0,1]$. The dimensionality is given by:

$$d = \arg\max[a_{iD}]\ i \in [1,3] \tag{4.5}$$

A good estimation for the radius of the neighbourhood is two times the average point spacing, $r = 2 \cdot \mathrm{APS}$, of the point cloud. This guarantees a representative number of neighbours while avoiding too large neighbourhoods, which would cause expanded transition zones at the boundaries of two or more objects with different surfaces. This neighbourhood will be used in the next filtering stages as the same rationale applies.

Slope Filter

Third, points with a slope higher than a threshold are filtered out. These points mostly correspond to building facades and fences. The slope is calculated using Equation 4.6:

$$\phi = \cos^{-1}(c) \tag{4.6}$$

where c is the third plane coefficient in its implicit form $ax + by + cz + d = 0$.

The best-fitted plane is derived by the normal vector n, obtained by PCA as described in Chapter 2, so that the plane coefficients of the plane equation are:

$$\begin{aligned} a &= n_x \\ b &= n_y \\ c &= n_z \\ d &= -(n_x * C_x + n_y * C_y + n_z * C_z) \end{aligned} \tag{4.7}$$

where C_x, C_y, and C_z are the centroids of the neighbourhood at the x, y, and z coordinates, respectively.

Opacity Filter

Then, points with an opacity lower than a threshold are filtered out. The opacity measures the ability of a surface to not being penetrated by laser pulses. This filter aims to remove most of the vegetation points, as solid surfaces like roofs yield a high opacity in contrast to vegetation. Opacity can be measured with the *Echo Ratio (ER)* [102] (see Figure 4.3a), which is a modification of the point density ratio introduced in [209] and it is derived as follows:

$$ER = n_{3D}/n_{2D} \cdot 100 \text{ with } n_{3D} \leq n_{2D} \tag{4.8}$$

where n_{3D} is the number of points inside a fixed distance measured in 3D (i.e., sphere) and n_{2D} the number of points inside the same distance measured in 2D (i.e., infinite cylinder).

However, for non-flat roofs, an increasing slope shrinks the 3D neighbourhood and decreases the value of ER. To mitigate this, the 3D search distance must be extended considering the roof slope. This is done by dividing the initial 3D distance by the cosine of the roof slope (see Figure 4.3b, hence the name *slope-adaptive Echo Ratio (sER)*. Note the relevance of the previous slope filtering: Points yielding a high slope would have caused the radius to increase

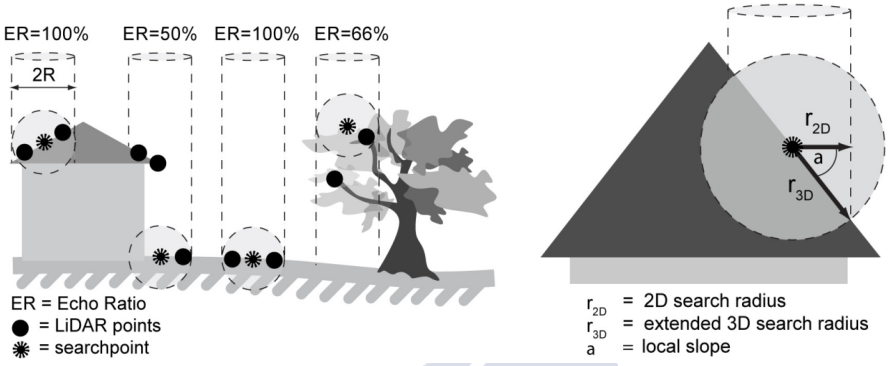


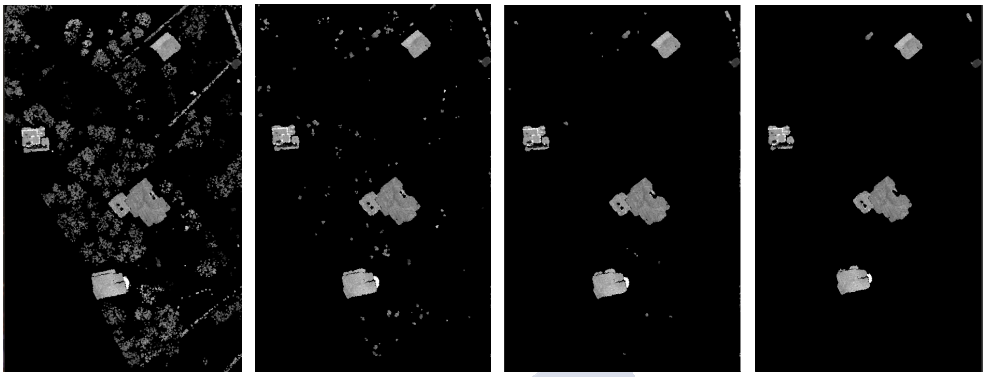
Figure 4.3: Echo ratio visualisation: (a) Basic and (b) Slope adapted versions [102].

far beyond the initial radius. To ensure that the slope adaptation is only applied to roof points, the best-fitted plane is calculated, and the standard deviation of the residuals must be under a threshold. Otherwise, the point is most probably a vegetation point so that $sER = ER$.

Morphological Filter

Mathematical Morphology [131] is a set-theoretical methodology that has been extensively used in image analysis. It provides two fundamental operations: *dilation* and *erosion*, that are employed to enlarge (dilate) or reduce (erode) features in binary images.

Based on the concept of morphological erosion, the following erosion procedure is performed: For each candidate point, these are, the points that have passed the previous filters, the ratio between the number of candidates and the number of non-candidate points $r_c \in [0,1]$ within its 2D neighbourhood is calculated. An infinite cylinder with a radius of 1 m is used as a neighbourhood. If $r_c < 0.5$, the output point is labelled as non-candidate and hence eroded. As shown in Figure 4.4, after a few erosions, which we have set experimentally as three, the points that are distributed in a spread-out fashion, along with islands of points, are removed.



(a) (b) (c) (d)

Figure 4.4: Erosion for an area of the Alcoy site: Candidate road points (a) before and after the (b) first, (c) second, and (d) third iteration of the erosion. Point coloured by intensity.

4.1.3 Object-level Filtering

The second part of the method consists in extracting the objects corresponding to the identified candidate building points. First, the points lying in the plane of each candidate point are clustered. Then, nearby planes are merged to compose the building object. Finally, small candidate buildings are filtered out to remove false positives.

Plane Clustering

The candidate points are sorted by curvature in ascending order and used as seeds for a clustering algorithm based on Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [65]. DBSCAN clusters points that are spatially close and labels as outliers points that lie in islands or are isolated. In the original DBSCAN, two parameters are needed: the minimum number of points (*minPts*) to form a cluster and the distance for the spatial search (ε). Additionally, to cluster different planes individually, points inside a cluster must also yield a plane residual under a threshold (τ). The plane clustering algorithm is as follows:

1. Sort the set of points $P = \{p_0, p_1, \cdots, p_n\}$ by curvature in ascending order.
2. Pick the next point p_i and examine its label.
    - If it is undefined go to Step 3.

- Else go to Step 2.

3. Search the 3D neighbours of p_i.
   - If the number of neighbours $<$ *minPts*, label p_i as noise and go to Step 2.
   - Else p_i is assigned to a cluster. Find the best-fitted plane for p_i and its neighbours and calculate its residuals. A point set of seeds S is created with the neighbours yielding a residual $< \tau$, and they are assigned to the same cluster. Go to Step 4.
4. Within S, pick the first point s_0 and search its 3D neighbours.
   - If the number of neighbours $\geq$ *minPts*, for each neighbour, examine its label.
     - If it is noise and the residual $< \tau$, assign it to the same cluster.
     - Else if it is undefined and the residual $< \tau$, assign it to the same cluster, add it to S and update the best-fitted plane.
   - Remove s_0 from S and return to Step 4.

An example of the clustering result is shown in Figure 4.5. It can be seen how each plane points have been grouped in different clusters. Note that only points above the height threshold (selected by the point-level filtering phase) are taken into account to prevent clusters to expand into planar surfaces lying at ground level. The minimum number of points is set to 3 as is the minimum of point needed to obtain a plane. The spatial distance is set to the double of the average point spacing as in the previous point-level filtering.

Plane Merging

To merge neighbouring planes, first, a neighbouring matrix must be established. For this, the boundary points of each plane are extracted by the Alpha Shapes algorithm described in Chapter 3. After extracting the boundary points, the neighbouring matrix is built by considering two planes as neighbours if at any of their boundary points are closer than 1 m. The neighbouring planes are then merged into one single building object.

Area Filter

As performed in the classification described in Chapter3, clusters with an area smaller than a threshold are filtered out to remove small objects, mostly false positives, that generally correspond to small flat vegetation surfaces.

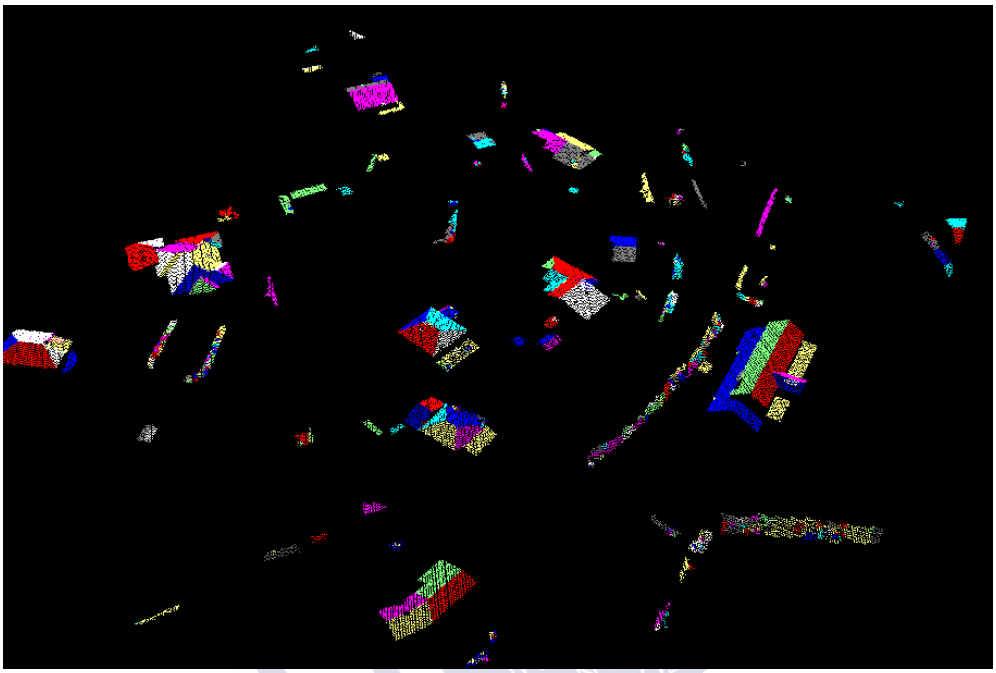


Figure 4.5: Clustering results for an area of the Alcoy site (see Section 4.2.2 for details about the data). Clusters are shown in different colours.

4.2 Results

In this section, we evaluate the performance of the method both quantitatively, using available ground truth, and qualitatively, using visual inspection. Also, we analyse the computational performance.

4.2.1 Quantitative Evaluation

The Vaihingen dataset from the International Society for Photogrammetry and Remote Sensing (ISPRS) 3D Semantic Labelling Contest [50] was used for evaluation, as in previous analysis. A detailed description is given in Chapter 2. As commented before, Vaihingen is an urban area with the typical characteristic of a European city [50]. The point cloud consists of 1,165,598 points, where point have been labelled in nine different classes (see Figure 4.6). The dataset is divided in two parts: A training dataset with 753,876 points (see Figure 4.6 left)

and a test dataset with 411,722 points (see Figure 4.6 right). The evaluation was performed both in the test data alone and in the whole data including both test data and the training data. Evaluation is meant to be carried out in the test data alone, which allows results to be comparable with other methods. However, because training is not needed for this method, we also find valuable performing the evaluation using the entire dataset as it is a bigger sample.

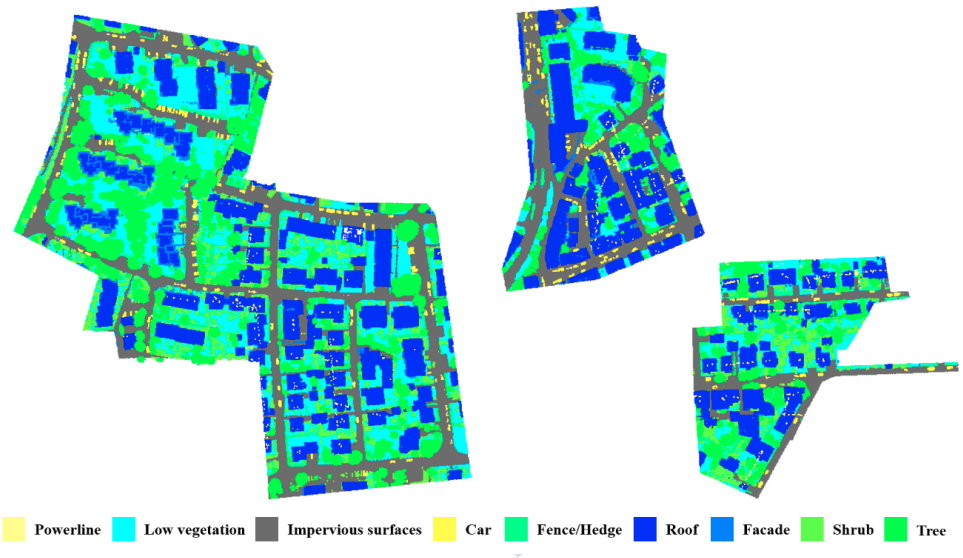


Figure 4.6: Vaihingen ground truth [251]

We use three quality metrics, namely correctness (C_r), completeness (C_p), and quality (Q), introduced in [100]. Their mathematically definitions are shown in Equation 4.9:

$$Cr = \frac{T_p}{T_p + F_p}, \quad Cp = \frac{T_p}{T_p + F_n}, \qquad Q = \frac{T_p}{T_p + F_p + F_n} \tag{4.9}$$

where T_p is the number of correctly labelled building points, F_p is the number of wrongly labelled building points, and F_n is the number of missed building points.

The default parameters (see Table 4.1) were used, except for the height threshold, that was set to 1.5 m. This is because, for the Vaihingen dataset, it has been observed by other authors that higher thresholds remove many low buildings and sheds [15].

Results are shown in Table 4.2. The method yields correctness and completeness above 90% and quality around 85% in both datasets. The error distribution is depicted in Figure 4.7.

Parameter	Value
Height threshold	2.5 m
Slope threshold	85°
sER standard deviation threshold	5 cm
sER threshold	75
DBSCAN plane residual threshold	10 cm
Minimum area	20 m^2

Table 4.1: Parameters and their default values.

Most of the false positives correspond to small vegetation surfaces with low curvature and small structures like thick walls. Also, some flat ground surfaces with a high height difference to the nearby ground are wrongly labelled as building. The false negatives mostly correspond from small building parts, that are missing because they have not being clustered due to their lack of planarity, and small sheds, that have been removed due to their small area.

Dataset	C_r	C_p	Q
Test data only	92.39%	92.64%	86.97%
Test and training data	93.76%	89.67%	84.62%

Table 4.2: Quality metrics of the quantitative evaluation.

4.2.2 Qualitative Evaluation

Visual inspection was conducted in two additional datasets to assess the performance of the method when dealing with different data. Alcoy data is a rural site with spread buildings consisting of 2.77M points with a point density of 8.65p/m^2. Arzúa data is an urban area with concentrated buildings consisting of 12.26M points with a point density of 20.35p/m^2. The default parameters were used (see Table 4.1). The results are shown in Figures 4.8 and 4.9 respectively. The figures show the extracted building points (in red) overlaid over the non-ground points coloured by intensity (in a grey range).

In the case of Alcoy, results are satisfactory, however, some limitations, highlighted in the Figure 4.8b, are present: (1) small objects belonging to buildings, such as chimneys, might not be identified; (2) man-made shrubbery can be difficult to distinguish from buildings; (3)

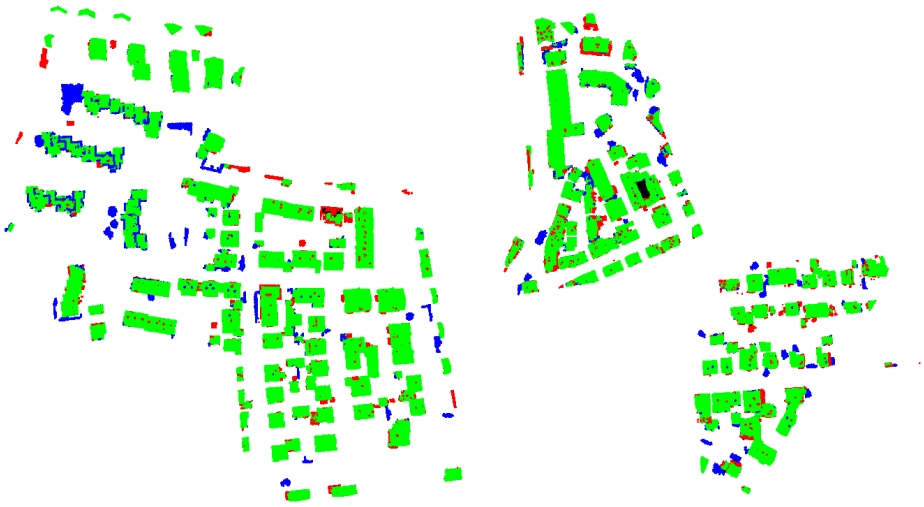
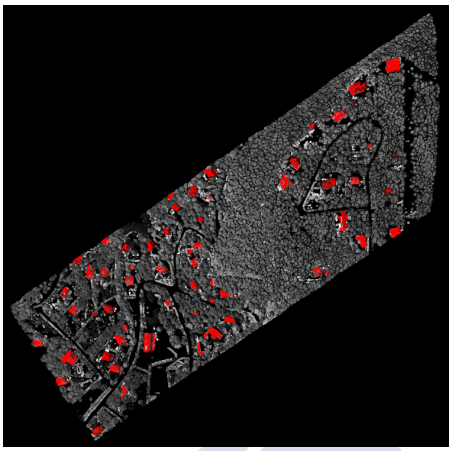


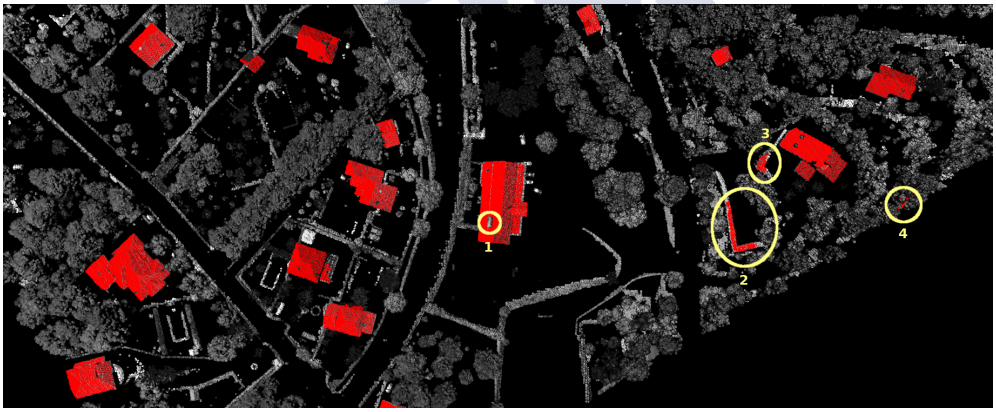
Figure 4.7: Vaihingen results: T_p in green, F_p in blue, and F_n in red.

surfaces at ground level, such as pavement, yielding a high height difference to the nearby ground can be wrongly identified as buildings; (4) the same situation that in (3) can happen with ground surfaces if these have not been removed in the initial ground filtering. Note that limitations (3) and (4) are closely related to the ground filtering, and they might be considered limitations of the ground itself rather than a limitation of the building extraction method. Despite, most buildings have been extracted successfully, even when if surrounded by vegetation (see highlights (5) and (6) in Figure 4.8c).

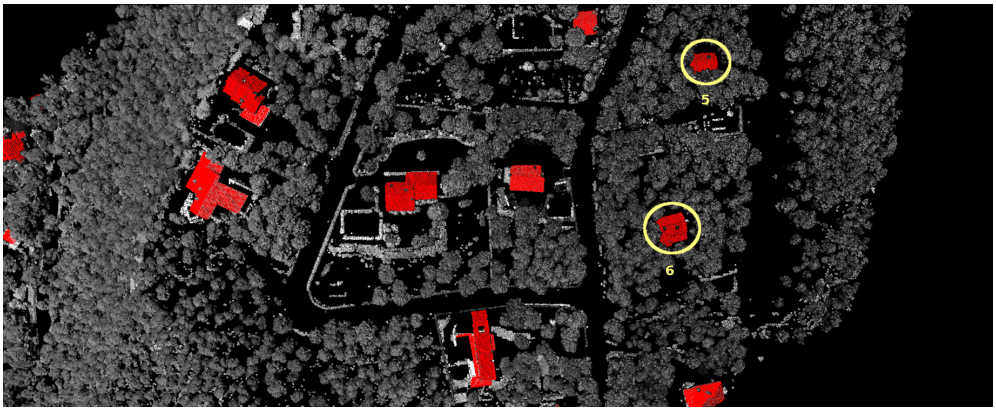
In the case of Arzúa, similar results are obtained, highlighted in the Figure 4.9b: (1) small objects belonging to buildings are missing; (2) a part of a roof surface of one large building is missing because the points have not been identified as non-ground points by the ground filter; buildings are extracted even if partially occluded by vegetation (3) and (4). However, there is a new issue related to data acquisition (see Figure 4.9c). In some areas of the site, the vertical and horizontal point spacing are significantly different, particularly, the horizontal spacing is larger. In other words, the spacing between adjacent scan lines is significantly larger than the spacing between the points within the same scan line. This means that in the neighbourhood search, the points from the adjacent scan lines are not reached so that points from the same plane are not clustered together and the clusters are too far away to join, which leads to isolated scan line-shaped clusters that will be filtered out due to their small area.



(a)

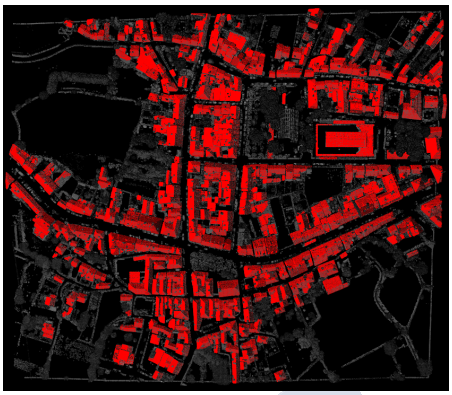


(b)

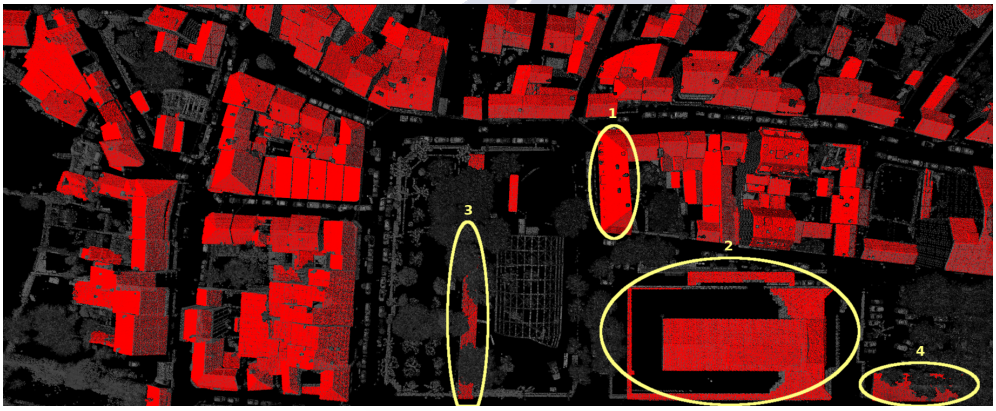


(c)

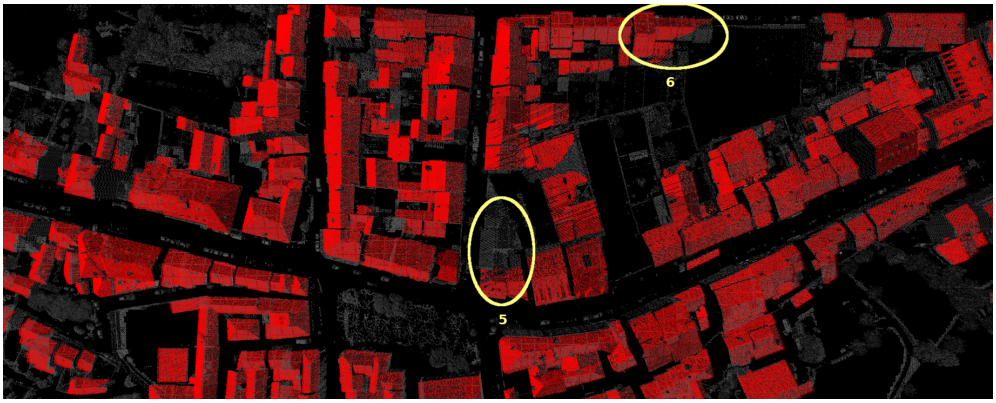
Figure 4.8



(a)



(b)



(c)

Figure 4.9

4.2.3 Computational Performance

The execution times of these stages for the Alcoy site are shown in Table 4.3. The building extraction framework encompasses the DTM generation and the building extraction itself. The DTM generation consists of five stages, being the iterative morphological filtering the core of the algorithm, which requires most of the execution time. The building extraction consists of four stages, being the point-level filtering consumes most of the time. This is expected, as it processes the most amount of points. Similar percentages are obtained when using other datasets. Therefore, these two major consuming stages were parallelised.

Procedure	**Exec. Time (s)**	**Exec. Percentage**	**Paralellised**
Min. surface generation	1.1819	3.25%	×
Outlier identification	0.0588	0.16%	×
Morphological filtering	21.4994	59.11%	✓
DTM generation	1.3849	3.81%	×
Point labelling	0.3075	0.85%	×
Point-level filtering	9.0165	24.79%	✓
Plane clustering	1.6990	4.67%	×
Plane merging	0.2563	0.70%	×
Area filtering	0.9660	2.66%	×
Overall	36.3703	100%	-

Table 4.3: Computational stages and their execution times.

The parallelised parts of the program are to 83.90% of the total execution time. Assuming that the speedup in these parts would be equal to the number of physical cores of the tested computer, which is 4, the expected theoretical speedup is given by Amdahl's Law is 2.70.

The same the OpenMP loop scheduling discussed in Chapter 3 is used. The obtained execution times and speedups obtained after parallelisation are shown in Table 4.4. The execution time of the parallelised stages was reduced from 29.9097 s to 7.6491 s, this is a speedup of 3.91, while the execution time of the overall program was reduced from 36.3703 s to 14.1097 s, that is a speedup of 2.58. Results are satisfactory, as they are very close to the predicted speedups.

Also, we can see that performing a more *adhoc* approach requires less execution time than a more general approach, as expected. The method proposed in this chapter is 1.46 times faster (using this dataset) than the classification proposed in Chapter 3 when executed with

Stage	Exec. Time (s) (Speedup)				
Threads	**1**	**2**	**4**	**6**	**8**
Morphological filtering	21.4994	11.2962 (1.90)	6.0054 (3.59)	5,6363 (3.88)	4.2473 (4.25)
Point-level filtering	8.5598	5.1200 (1.67)	3.1248 (2.74)	2.7745 (3.09)	2.5689 (3.33)
Overall	29.9097	16.5475 (1.81)	9.0667 (3.30)	8.3296 (3.59)	7.6491 (3.91)

Table 4.4: Execution times and speedups of the paralellised stages.

one single thread (36.3703 s over 52.9995 s) and 2.08 times faster (14.1097 s over 29.3379 s) when using multithreading. This should be taken into account when developing solutions that require low execution times.

4.3 Discussion

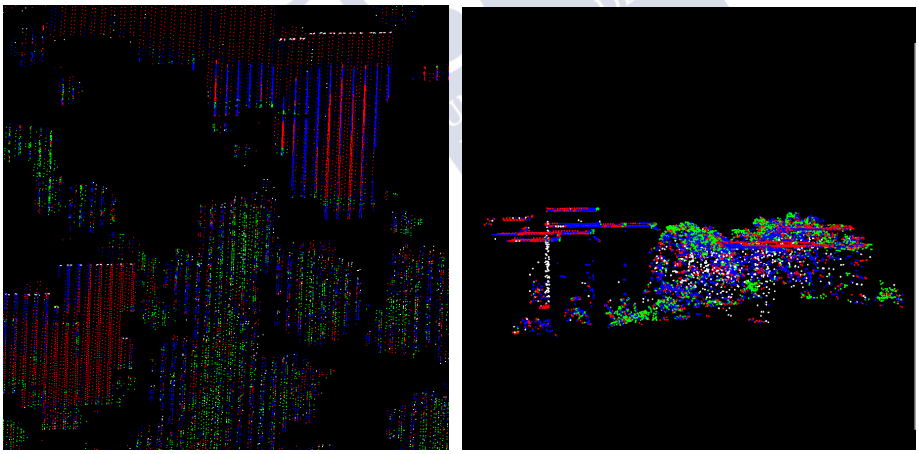
In this section we describe how the parameters are set and some insights of the dimensionality filtering.

4.3.1 Parameter Setting

This method requires six parameters than are set following different strategies, as previously shown in Table 4.1. The slope and sER thresholds have been set by experimentation, based on other studies [53, 102], and they are not mean to require tuning by the user. The sER and DBSCAN plane thresholds are derived from the data. A good estimation for the sER standard deviation of the plane residuals is the vertical accuracy and the double of the vertical accuracy for the DBSCAN plane residual threshold. In case this datum is not known, we found experimentally that a value of 10 cm is suitable for most of the cases. However, the height and minimum area thresholds are problem-specific, and although we have proposed default values, the user may need to tune them depending on the characteristic of the buildings to be extracted. Common values found in the literature are 1 m [14], 2.5 m [16, 113], 3 m [154, 43], and 3.5 m [205, 267] for the height threshold, and 5 m^2 [102], 40 m^2 [205] and 50 m^2 [174, 43] for the area threshold.

4.3.2 Dimensionality Filtering

Arguably, points with dimensionality of 1 should also be filtered out, however, some building points can yield a dimensionality of 1 due inconsistency in the scanning pattern. We have observed two scenarios when this happens. The first scenario is when the distance between scan lines is larger than usual, as shown previously in the evaluation of the Arzúa site. This large spacing causes points from the adjacent scan lines to fall outside the neighbourhood, and only points within the same scan line are taking into account so that the points inside the neighbourhood exhibit linear behaviour. The second scenario is when there is point accumulation caused by deceleration of the sensor's oscillating mirror when changing the scanning direction, as shown in Figure 4.10. This issue could be avoided if using another type of sensor mechanism without deceleration, such as rotating polygon. Because of these reasons, we opt to not filter points with dimensionality of 1.



(a) (b)

Figure 4.10: Dimensionality of the points for an area of the Vaihingen site: (a) Top view and (b) profile view. Points are coloured by dimensionality: 0 in white (isolated points), 1 in blue, 2 in red, and 3 in green.

4.4 Conclusions

In this chapter, a method for building points extraction is presented. It uses both point-level processing, to identify candidate building points, and then object-level processing to extract the final building objects. This strategy is computationally efficient and shows good results. Experiments conducted showed correctness and completeness above 90% and a quality of 85%. Visual inspection in another two datasets also showed satisfactory results. Some limitations associated with the data acquisition remain, particularly, a significant difference between the vertical and horizontal spacing of the data can lead to hinder the identification of buildings. Nevertheless, this only happens in some areas of some data, and further developments in data acquisitions will alleviate this issue.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 5

EXTRACTION OF ROAD POINTS

Precise information about the road network is crucial for a wide range of applications such as city planning, traffic management, or road monitoring. The task of road extraction addresses two issues: (1) identification and (2) delineation [57]. In this chapter, we address the first issue, while in the next chapter we address the later.

Over the last few decades, remote sensing techniques have been extensively studied to accomplish these tasks [246]. Road extraction from aerial and satellite imagery has achieved various extents of success, but it is still a difficult task due to the limitations of passive sensing. This has led many researchers to explore the advantages of using LiDAR data [197]. The recorded elevation information helps to identify roads from other impervious objects, and reflectance intensity provides good separability of ground materials with similar elevation [220]. To improve results, some authors have fused aerial imagery with LiDAR data [112, 44, 244], however, as in the work seen so far, we focus on the road point extraction based exclusively on LiDAR data.

Several authors have presented methods for identifying road points from LiDAR data during the last years. This identification is often used as input to the delineation step, also referred to as vectorisation. In [46] road points are identified by a hierarchical classification based on height, intensity, and local point density. The intensity threshold is selected manually. Then, a binary image is produced and the road is further refined using morphological operations and connected component analysis. Two sites, Fairfield and Yaronga, that are mainly urban with some rural traits were tested. In [122] ground points inside a manually selected intensity range are kept, and a constrained Delaunay Triangulated Irregular Network (TIN) is used to

group the points and remove small regions. An urban area of the city of Shashi was tested. In [279] the intensity distribution of ground points is assumed to follow a Gaussian mixture model, and an expectation maximisation algorithm is used to find the maximum likelihood solution of the model. An intensity image is produced from the ground points and each pixel is classified according to its posterior probability. A certain K value is required, so that pixels are categorised into K classes. Besides, the road class needs to be manually identified from these K classes. Three sites, Atlanta, Denver, and Oakland, mainly covering downtown and surrounding residential areas were tested. In [248] ground points inside a manually selected intensity range are kept, and the remaining noise is removed by a morphological opening operation. An urban area of Niagara was tested. In [145] ground points are segmented based on local intensity distribution histogram, and filtered by intensity gradient and area. A region of the city of Vaihingen was tested. In [115] road points are identified based on intensity, local point density, and region area constraints. A new method for the calculation of the intensity threshold based in the skewness balancing algorithm [22] is introduced, and it was tested in four urban sites, including Vaihingen. This method assumes that the distribution of intensity values always yields a positive skewness. While this is generally true for urban areas, in this thesis, we show and consider that this behaviour is not guaranteed in mixed or rural areas.

Alternatively, road centrelines can be extracted without prior extraction of the road points. In [111, 144] road centrelines are obtained by a mean-shift clustering with adaptive window size based on the intensity and surface roughness. The well known sites of Vaihingen and Toronto were used in the experiments. While these methods yield better results than previous studies, it is unclear how a mean-shift clustering will perform in less urbanised areas. A mean-shift clustering is an iterative procedure that shifts each data point to the average of data points inside a window until a stable average is reached. The idea is that ground points will converge into scattered points while road points will converge into the road centreline. This way, road centrelines can be extracted by searching linear features. However, some concerns remain: The appropriate selection of the window size, the high computational cost, and the fact that in rural sites it is common to find areas with a homogeneous intensity and surface roughness similar to road segments, that may prevent the ground points to converge into scattered points. Furthermore, because the road points are not being labelled, extracting road features, e.g. roughness or width, is not straightforward after the extraction of the centrelines.

A different approach is to extract road points with machine learning techniques such as AdaBoost [149], Support Vector Machines (SVM) [18, 164], Multiple Classifier Systems

(MCS) [211], Conditional Random Fields (CRF) [184, 183], Random Forests (RF) [68, 128], or Maximum Likelihood Classification (MLC) [234]. However, the need for training data introduces important implications. First, the creation of labelled data is a difficult task with high economical and temporal costs, so that only a small amount of reference data are often available. And second, due to the heterogeneous characteristics of point cloud data, achieving high performance on datasets different from those used for training is not guaranteed.

Two issues remain to be addressed to perform road extraction successfully. First, methods often need the manual input of parameters, being the intensity threshold a particularly critical one. Second, it is difficult to assess whether a specific method is suitable for particular data, given the fact that the datasets used in most studies are focused on urban areas and experiments are conducted in a very small number of test sites. To address both issues, in this chapter, we propose a new framework for the extraction of road points that is highly automatic, requiring only a single user non-critical parameter, which introduces an improved skewness balancing algorithm for the calculation of the intensity threshold, and we use a meaningful dataset with a variety of both urban and rural landscapes to evaluate its performance.

The rest of the chapter is organised as follows: Section 5.1 presents the framework for road point extraction. Section 5.1.1 describes the improved method to automatically estimate the intensity threshold. Experiments are presented in Section 5.2 and discussed in Section 5.3. Section 5.4 exposes the conclusions.

5.1 Method

In a preliminary stage of the road extraction process, the distinction between ground and non-ground points is required. As has been commented before, different approaches can be considered to accomplish this task. In this thesis, we use the classification method described in Chapter 3. An alternative method is also proposed in Chapter 7.

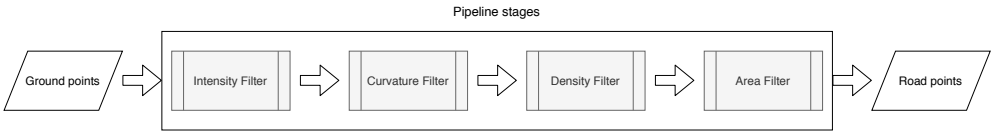


Figure 5.1: Flowchart of the pipeline filtering.

From this input, road points are separated from ground points by a pipeline filtering shown

in Figure 5.1. In each stage of the pipeline, points are evaluated to verify if they fulfil a given characteristic. Any road point is assumed to verify the following characteristics: It yields low intensity, it lies on a plane, it is surrounded mostly by road points, and it is within a set of road points with meaningful size. Only first return points with these characteristics are identified as road points. Non-first returns, which generally are present in ground areas below trees, are not considered because their inherent low intensity produces noticeable false positives in forested areas. A limited amount of road points may not fulfil all these constraints, e.g., road points within road markings yield high-intensity values and road points under vegetation will be non-first returns, but these points could be recovered in later stages. Following, the filters to evaluate if each point fulfils these characteristics are introduced in detail.

5.1.1 Stage I: Intensity Filter

The intensity can be defined as the ratio between the received versus the emitted power of the laser beam. The intensity value depends on the target characteristics (reflectance and roughness), acquisition geometry, instrumental effects, and environmental effects. It is well known that road materials, such as asphalt or concrete, have low reflectance, thus road points generally yield low-intensity values [220]. The challenge resides in selecting an appropriate intensity threshold for every dataset. Although intensity is one of the key factors to distinguish between ground and road points, little research has been conducted to investigate automatic thresholding. In [115] a skewness balancing algorithm to automatically calculate the intensity threshold is proposed. The basic assumption of the skewness balancing algorithm is that naturally measured samples will lead to a normal distribution due to the central limit theorem [61]. Originally, it was assumed that elevation values of ground points follow a normal distribution, which is disturbed by object points so that by their removal, the normally distributed ground points are obtained [22]. This concept was adopted in [115] for road extraction. In our proposal, we extend this algorithm by i) improving its robustness to intensity outliers, ii) making it suitable for non-urban areas, and iii) standardising and improving the balancing speed. A flowchart of the proposed algorithm, which we have called bidirectional skewness balancing, is shown in Figure 5.2. The main stages of this method are described in detail in the following subsections.

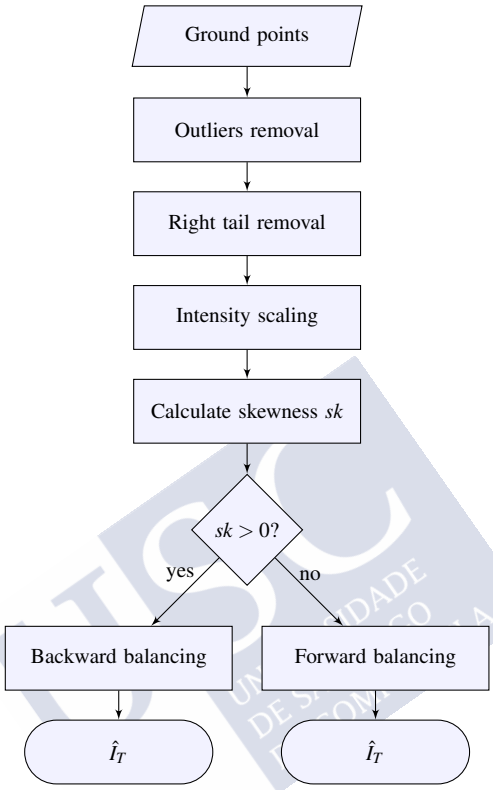


Figure 5.2: Flowchart of the bidirectional skewness balancing.

Outliers Removal

The skewness is a measure of the degree of asymmetry of a distribution. It can be used to detect whether a distribution is concentrated on the left or on the right side. This information is used to determine the direction of the skewness balancing (as explained in Section 5.1.1). Nevertheless, the skewness is very sensitive to the presence of outliers. In particular, very high-intensity values lead the skewness to be positive. A significant number of outliers can even artificially change the skewness value from negative to positive. This, coupled with the fact that the presence of intensity outliers across datasets is common in practice, makes the removal of outliers a mandatory task prior to any processing. To this end, we use the

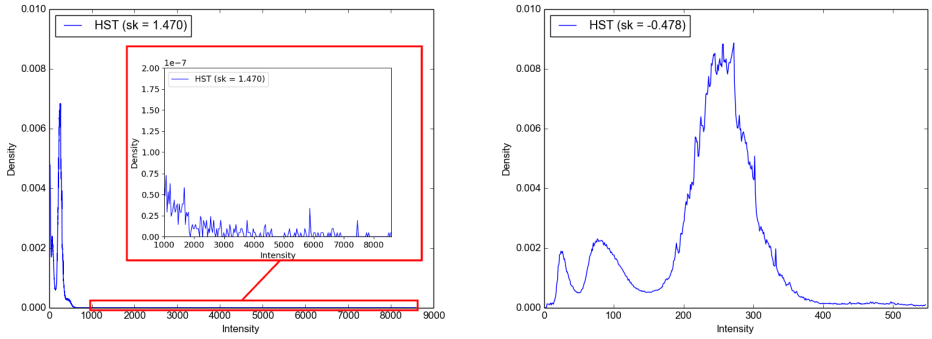
Interquartile Range (IQR) which is defined as:

$$\text{IQR} = Q_3 - Q_1 \tag{5.1}$$

where Q_3 and Q_1 are the third and first quartiles of the distribution, respectively. Then, the maximum value of the box-and-whisker plot is calculated, which is set to be the maximum intensity allowed, that is:

$$I_{max} = Q_3 + 1.5 \cdot \text{IQR} \tag{5.2}$$

All points yielding an intensity $I > I_{max}$ are considered outliers and are filtered. A clear example of the need for the removal of outliers is shown in Figure 5.3. Note that the presence of some points yielding a very high-intensity lead to a highly positive skewness (see Figure 5.3a). A positive skewness value will initiate a backward balancing which will fail to obtain an accurate intensity threshold. By removing the outliers, the actual nature of the distribution is clearly exposed, which yields a negative skewness (see Figure 5.3b). In our proposal, a negative skewness value initiates a forward balancing to calculate the appropriate intensity threshold.



(a) Before

(b) After

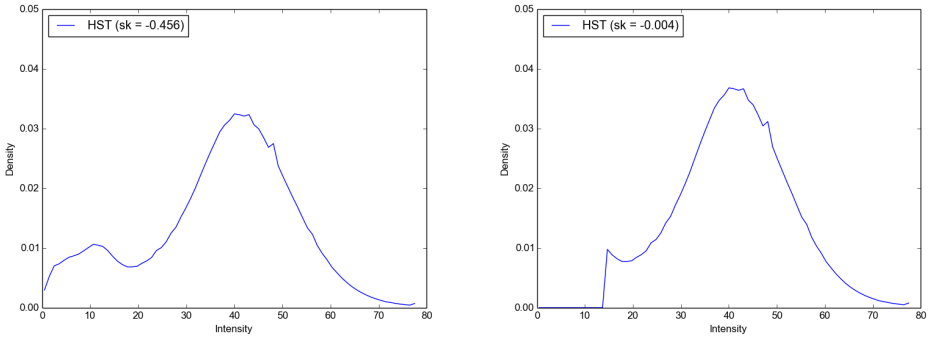
Figure 5.3: Normalised histogram (HST) of Truro data (see Section 5.2.1) (a) before and (b) after removing the outliers.

Right Tail Removal

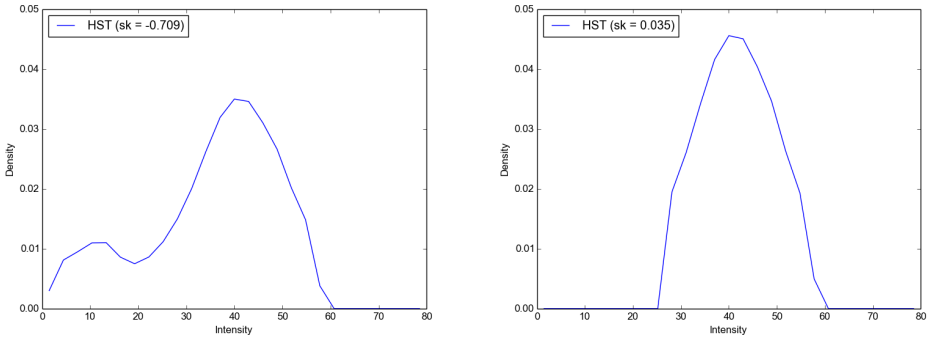
Even after the removal of outliers, the Gaussian in the histogram formed by ground points generally presents a notable right tail (see Figure 5.4a). This produces a negative impact in the algorithm and needs to be addressed. The skewness balancing stops when reaching symmetry, that is, zero skewness. This means that, in the presence of a significant right tail, an equivalent significant left tail will be preserved. This situation can be observed in Figure 5.4b, where the forward balancing stops when reaching the left tail instead of advancing until the base of the Gaussian. As a consequence, the calculated intensity threshold is systematically low. This effect is mitigated if the right tail is removed. This is performed by filtering the points with an intensity beyond a certain percentile (see Figure 5.3c). This way, the forward balancing continues until reaching the base of the Gaussian (see Figure 5.3d), resulting in a better estimation of the intensity threshold. We have determined experimentally that the 95th percentile provides consistent results in all cases. To calculate the intensity corresponding to a certain percentile, the intensity values are sorted in ascending order, and its index within the list is calculated using the nearest rank method:

$$n = \left\lceil \frac{P}{100} \cdot N \right\rceil \tag{5.3}$$

where n is the index, also known as ordinal rank, P is the considered percentile, and N is the list size.



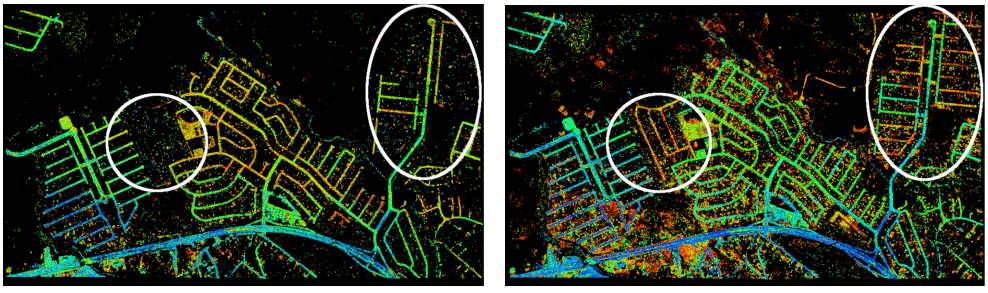
(a) (b)



(c) (d)

Figure 5.3: Effects for the Carola site. The histogram is shown (a) after removing outliers, (b) after removing values lower than the threshold obtained by the forward skewness balancing without tail removal, (c) after removing the right tail, and (d) after removing values lower than the threshold obtained by the forward skewness balancing with tail removal.

Figure 5.4 shows the improvement in the estimation of the intensity threshold when the right tail is removed. Points are coloured by their intensity values mapped into a heat map. By performing this, previously missing road segments are now detected. The new road points are the ones yielding the highest intensity (red colour). Some false positives are also added, but it is crucial to detect all road points in this stage and they can be removed in the next stages.



(a) Before (b) After

Figure 5.4: Results of the intensity filtering for Carola data (a) before and (b) after the removal of the right tail. Points are coloured by a cold-to-hot gradient of their intensity values. Some road segments enclosed in the highlighted white areas are recovered.

Intensity Scaling

In [115] the skewness balancing is performed by subtracting 1 to the intensity threshold iteratively. Nevertheless, intensity ranges can vary greatly as the intensity can be scaled into different ranges, commonly between 8 and 16 bits [74]. This means that using an absolute step size of 1 will result in different relative step sizes for point clouds with different intensity ranges. In other words, the speed of the balancing will not be the same across point clouds. For example, in the Vaihingen data, intensity values are scaled to an 8-bit range so that the relative step size is then 1/255, whereas, in the St Arnaud data, intensity values are scaled to a 16-bit range so that the step size would be 1/65535, which increases the number of iterations, reducing the computational performance significantly. There are also cases in which the intensity range after removing the outliers can be smaller than the initial bit range. In the Toronto data, the intensity values are scaled to a 8-bit range, but after removing the outliers, the remaining values are in the [0, 58] range, so the step size would be 1/58. Such large relative step size may lead to obtaining an intensity threshold that is not optimal, as the skewness balancing would have been stopped earlier if using a smaller step size. To enforce an equal relative step size, the remaining intensity values of each point cloud are scaled into the [0, 255] range and the absolute step size is always kept as 1. This assures a relative step size of 1/255 independently of the intensity range.

Balancing Speed

While using the same relative step size for every data is essential for obtaining an optimal intensity threshold, it could be done in a more computationally efficient manner. Generally, at the beginning of the skewness balancing, the skewness of the distribution is far from being zero (symmetric). Therefore, it would be more efficient to increase the balancing speed at the beginning, while gradually reducing it towards reaching a zero skewness. This can reduce the number of iterations drastically, reducing computational load significantly. To achieve this, a dynamic step size that decreases linearly is used, defined in Equation 5.4.

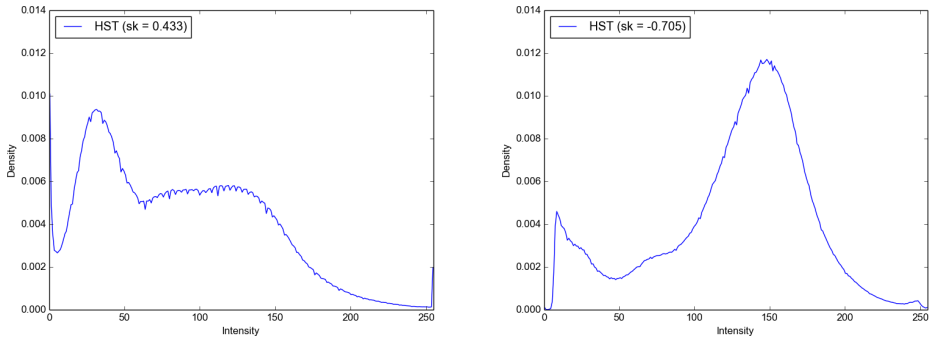
$$\text{Step} = 1 + \left(\frac{sk_i}{sk_0} \times (\text{Step}_{max} - 1) \right) \tag{5.4}$$

where sk_0 is the skewness at iteration zero, sk_i is the skewness at the current iteration so that $sk_i \leq sk_0$ and Step_{max} is the maximum step size, which has been set experimentally to 10. This way, the step size is in the [1, 10] range, the smaller the smaller the skewness.

Bidirectional Balancing

Once the intensities have been scaled, the skewness is calculated and a forward or backward balancing is initiated depending if its value is negative or positive, respectively. Let us analyse the reasons to consider both directions. In [115] only backward balancing is used, because it is assumed that intensity values of road points follow a normal distribution, which is disturbed by non-road points so that by their removal, the normally distributed road points are obtained. However, this is only true for urban areas. In such scenarios, road points are predominant over non-road points, meaning that the distribution is left-concentrated and its skewness is positive as shown in the example of Figure 5.5a. In contrast, in less urbanised or rural areas the situation is just the opposite, non-road points are predominant, so the distribution is right-concentrated and its skewness is negative as shown in the example of Figure 5.5b. In other words, the disturbance to remove is the non-predominant class, road or non-road, which changes among study sites. Thus, we reformulate the assumptions of the skewness balancing:

1. Intensity values of both road and non-road points are normally distributed (skewness = 0).
2. A predominance of road points leads to a left-concentrated distribution (skewness > 0).
3. A predominance of ground points leads to a right-concentrated distribution (skewness < 0).



(a) Vaihingen urban site

(b) Alcoy rural site

Figure 5.5: Comparison of the intensity values distribution between (a) the urban site of Vaihingen and (b) the rural site of Alcoy. The dominant point class, road or non-road, determines if the distribution is left-concentrated or right-concentrated, respectively.

While the original algorithm only works properly in cases under assumptions 1 and 2, we propose to generalise the core principle of the algorithm to make it also suitable for cases under assumption 3. The idea of the original algorithm is to filter points iteratively until reaching a zero skewness, in other words, until only the points in the Gaussian-shaped peak remain. For a distribution with positive skewness this filtering is done backwards (right-to-left), as the peak is located on the left side. For a distribution with negative skewness, we propose to perform the filtering forwards (left-to-right) as the peak is located on the right side. A flowchart of the forward balancing is shown in Figure 5.6. Therefore the direction of the balancing is dictated by the skewness value.

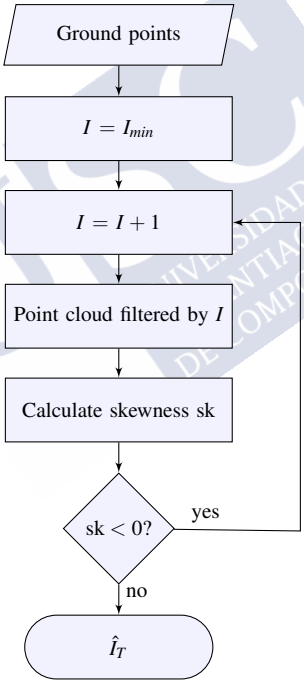


Figure 5.6: Flowchart of the forward skewness balancing.

Note that because of the intensity scaling described in the previous subsection, the obtained intensity threshold in the skewness balancing will be scaled. This can be reverted into its raw intensity value by:

$$I_T = \hat{I}_T \cdot I_{max}/255 \tag{5.5}$$

where I_T is the raw intensity threshold, $\hat{I}_T$ is the scaled intensity threshold, and I_{max} is the maximum raw intensity after the removal of the right tail.

All points with intensity $I > I_T$ are filtered out. Furthermore, points yielding a zero intensity are also filtered, as we found that they typically correspond to water surfaces.

5.1.2 Stage II: Curvature Filter

After filtering the points by intensity, their curvature is considered in the next stage of the pipeline. To analyse the curvature of a point, its neighbourhood must be identified. A common neighbourhood model is a sphere with radius r. The length of r is selected based on two factors: the average point spacing (APS) and the minimum road width (MRW). On the one hand, it is necessary to consider a significant amount of neighbouring points, an appropriate value could be $r = 2 \cdot \text{APS}$. On the other hand, if the length of r is large concerning the road width, there will be a high percentage of non-road points inside the neighbourhood influencing the curvature calculation. Ideally, a point in the centerline of the road will have only road points as neighbours, while road points near the edges will have a lower number of road points as neighbours. Considering this fact, r must not be higher than the half of the road width. The proposed value of r is:

$$r = \min(2 \cdot \text{APS}, \text{MRW}/2) \tag{5.6}$$

It is important to note that for the query of neighbours, only points in the same flight strip are considered, as points within overlapping strips can yield a significant higher curvature due to issues in the geometric adjustment.

The eigenvalues, as described in Section 2.1.1, describe the local geometry structure of the point set, from which the surface variation [194], also known as change in curvature [250], is:

$$C_\lambda = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \tag{5.7}$$

C_λ is zero for points lying on a plane, which is expected for road points. Nevertheless, because LiDAR data have inherent noise, it is not expected to achieve a zero value, therefore, a threshold needs to be set. We found experimentally that the value 0.005, used by other authors [210, 213], is suited for the whole dataset. So only points with $C_\lambda < 0.005$ are considered planar and are kept for the next filtering stage.

5.1.3 Stage III: Density Filter

Roads are connected objects, so road points must also be connected. Therefore, it is expected for a road point to be surrounded by other road points. We adopt the constraint of local point density presented in [46], which is the percentage of road points within the neighbourhood of a given point. The assumption is that a road point will have a different local point density depending on its position within the road, but always over a certain minimum. A point in the middle of the road is expected to have a local point density of 100% (Figure 5.7a), while a point in the edge of the road would have a local point density near 50% (Figure 5.7b). In an extreme case, where the point is in a corner with a sharp bend of 90° (Figure 5.7c), it will still maintain a local point density of 25%. Therefore, points yielding a local point density lower than 25% are filtered out. To define the neighbourhood, a sphere with a radius d is used. According to the previous reasoning, d must not be larger than the half minimum road width (MRW). In our case we use $d = \text{MRW}/2$.

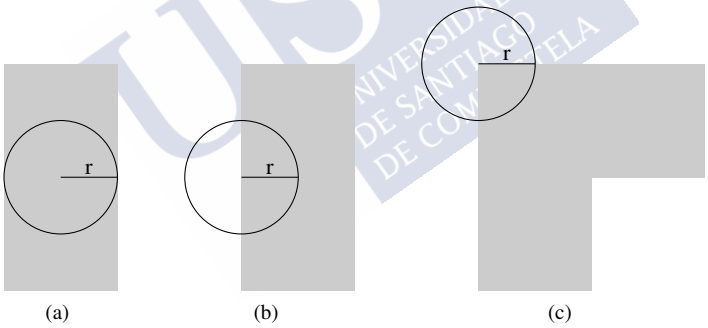


Figure 5.7: Visualisation of a circular neighbourhood with radius r for different points in the road: (a) a point in the centerline, (b) a point in the road edge, and (c) a point in a sharp bend of 90 degrees.

5.1.4 Stage IV: Area Filter

Some isolated points eventually fulfil all constraints described in the previous stages. To retain only meaningful points and remove this kind of noise, the points are grouped into clusters and only clusters with a substantial area are kept after this filter. This clustering is achieved by a simple region growing procedure, in which the seed is the first currently non-clustered point,

and the inclusion criterion is to add all neighbours inside a sphere with radius 1 m of the current point. A minimum area $A_{min} = 2 \cdot \text{MRW}^2$ is set, which can be seen as the minimum road rectangle: A rectangle with a width equal to the MRW and a length of twice the width.

5.2 Results

In this section, we present the dataset and parameters used for the experiments, and we evaluate the performance of the method both quantitatively, using manually-generated ground truth, and qualitatively, using visual inspection. Also, we analyse the computational performance.

5.2.1 Data and Parameters

Site	Urban-isation	Date	Sensor	Points (M)	Density (p/m^2)	Source
Alcoy	Medium	NA/2013	Leica ALS60	19.14	8.65	Babcock Int. [19]
Arzúa	Low	03/2018	Riegl VQ-480i	40.70	20.35	Babcock Int. [19]
Carola	Medium	07/2010	Optech Gemini	44.87	10.28	OpenTopography [92]
Logroño	Medium	09/2016	Leica ALS80	63.80	2.00	PNOA [86]
St Arnaud	Low	12/2017	Riegl LMS-Q1560	28.83	9.54	OpenTopography [224]
Toronto	High	02/2009	Optech Orion M	13.86	6.00	ISPRS [50]
Trabada	Low	11/2004	Optech 2033	29.39	4.00	LaboraTe [138]
Truro	Medium	07/2010	Optech Gemini	25.93	4.35	OpenTopography [203]
Vaihingen	High	08/2008	Leica ALS50	18.56	4.00	ISPRS [50]
Victor Harbor	Medium	09/2011	Optech Gemini	43.48	2.81	OpenTopography [232]

Table 5.1: Point clouds included the dataset.

A dataset consisting of ten point clouds is used for our experiments. This dataset has been carefully selected to consider a wide variety of scenarios. This is, to the best of our knowledge, the most complete dataset for road extraction used in the literature in the field. Regarding the acquisition of the point clouds, they were obtained from five different sources, acquired with eight different sensors, and yield point densities ranging from 2 to 20 p/m^2. The selected sites along with relevant information are shown in Table 5.1. While data from Babcock International and LaboraTe are not publicly accessible, data from the ISPRS Benchmark, OpenTopography, and PNOA (Spanish National Plan for Aerial Orthophotography), are openly available. Regarding the urbanisation level, the point clouds range from highly urbanised areas with a predominance of road over ground, such as city cores, to low urbanised

areas with a dominance of ground over road, also including moderately urbanised areas, with a more balanced ratio between road and ground. The degree of urbanisation is quantified as the percentage of road points within the ground points. As mentioned in Section 5.1.1, there is a relation between the degree of urbanisation and the skewness of the intensity values (see Figure 5.8). Because of the lack of ground-truth for this purpose, we have taken the percentage of road points extracted by our method as the degree of urbanisation. Nevertheless, a trend can be recognised, so that higher urbanisation leads to a higher skewness. The skewness value used in the figure is the one obtained after the removal of intensity outliers (see Section 5.1.1).

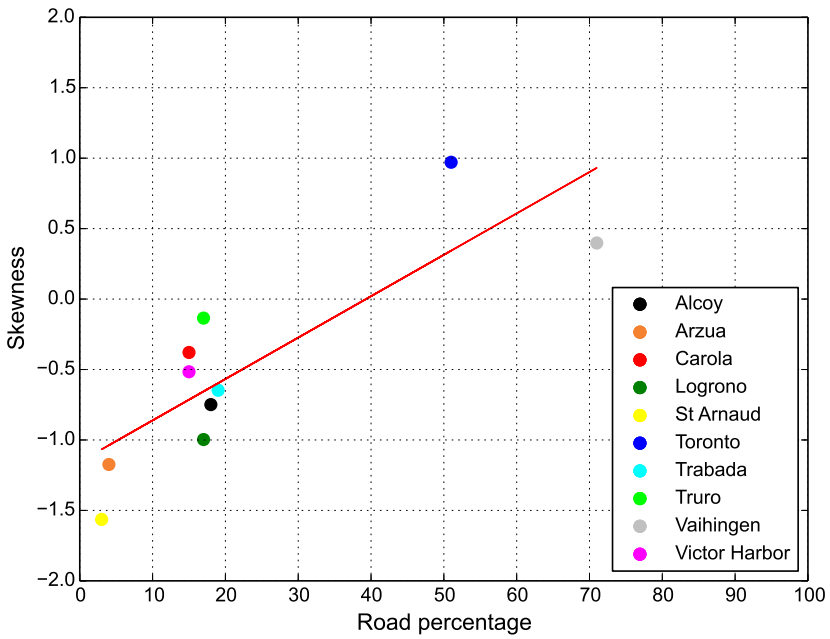


Figure 5.8: Level of urbanisation vs. skewness. The red line is the 1D polynomial fitted line. Note that the adjustment is far from perfect but it is just shown to help to visualise the trend.

We tested our method on each point cloud in the dataset, setting the minimum road width to 2 m. The selection of this parameter is not critical, and prior knowledge of the road width is not needed. A detailed explanation can be found in Section 5.3.3.

5.2.2 Qualitative Evaluation

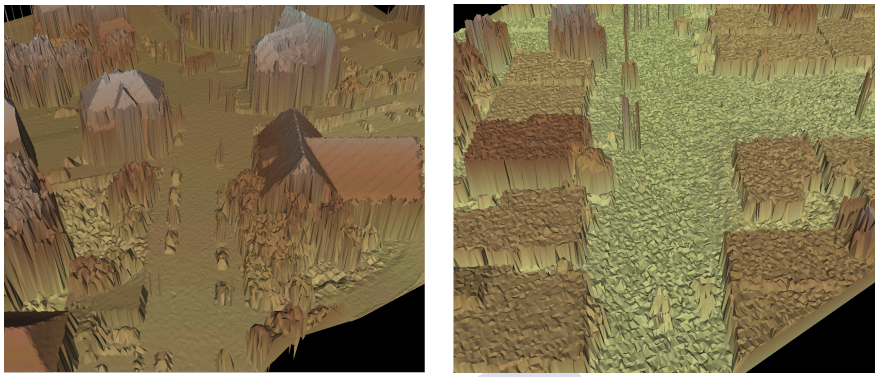
A ground-truth for each point cloud was generated by stratified random sampling with a sample size of 100 points per class, following the recommendations in [49]. These points were then manually labelled, interpreting both the point cloud and aerial images. We use three quality metrics, namely correctness (C_r), completeness (C_p), and quality (Q), already used in previous chapters, to quantitatively assess the method capability to extract road points.

The values of the three quality measures are shown in Table 5.2. To determine the suitability of the method, we refer to [165], where a minimum value of 0.60 and 0.75 for completeness and correctness, respectively, were defined for the result to become practically useful. Our method satisfies this minimum for the whole dataset. Furthermore, the authors stated that to be of real practical importance, a method probably should yield completeness and correctness values around 0.70 and 0.85, respectively. With an average completeness and correctness values of 0.93 and 0.83, respectively, our method also meets this quality standard.

Site	$\mathbf{C_p}$	$\mathbf{C_r}$	**Q**
Alcoy	0.97	0.77	0.75
Arzúa	0.81	0.82	0.69
Logroño	0.99	0.80	0.79
St Arnaud	0.99	0.88	0.87
Toronto	0.80	0.93	0.76
Trabada	0.97	0.69	0.68
Truro	0.97	0.97	0.94
Vaihingen	0.95	0.80	0.77
Victor Harbor	0.96	0.73	0.71
Min.	0.80	0.69	0.68
Avg.	**0.93**	**0.83**	**0.78**
Max.	0.99	0.97	0.94

Table 5.2: Results of the quantitative evaluation.

Note that the Carola site has not been evaluated quantitatively. This is because the points present an abnormal curvature (see Figure 5.9), which makes infeasible the task of generating ground-truth. Furthermore, the curvature filter cannot be applied in these conditions, as it will filter most of the points. Nevertheless, we tested the pipeline filtering without the curvature filter. Visual inspection demonstrates that the pipeline can still be effective even without the curvature filter stage (see Figure 5.10f).



(a) Vaihingen (b) Carola

Figure 5.9: TIN computed with FugroViewer [77] of (a) Vaihingen and (b) Carola sites. Point of view over a road. Carola data show unusual high curvature.

5.2.3 Qualitative Evaluation

The road points extracted by our method along with aerial images of the sites are shown in Figure 5.10. Visual inspection shows satisfactory results, most of the road points have been identified in every site with tolerable noise. Road points have been extracted in all point clouds, even when the percentage of road points is as small as 2.89%, as in the case of St Arnaud. To identify the limitations of the method, we analysed the two sites yielding quality values lower than 0.70, namely Arzúa and Trabada.

In the case of Arzúa, the road surface yields significantly higher intensity than in the other point clouds. This behaviour reduces the intensity difference between road and ground points hindering their distinction. This is the reason why, although the main road has been extracted, some road segments inside the city core and the eastern road segment are missing. Furthermore, a few adjacent strips show noticeably different intensity values. Because of this, there is a noticeable gap in the southwestern road segment (see Figure 5.11). Overall, a higher quality could be achieved by applying a better intensity correction.

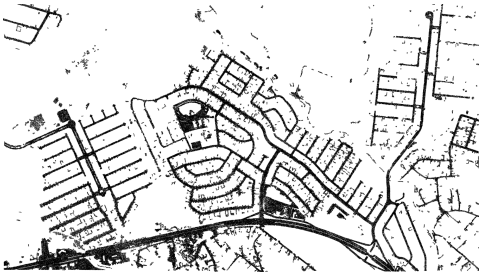
(a) Alcoy's aerial image

(b) Alcoy's extracted road points

(c) Arzúa's aerial image

(d) Arzúa's extracted road points

(e) Carola's aerial image



(f) Carola's extracted road points

 Figure 5.10: Aerial images of the tested data (left) and extracted road points (right).

(g) Logroño's aerial image

(h) Logroño's extracted road points

(i) St Arnaud's aerial image

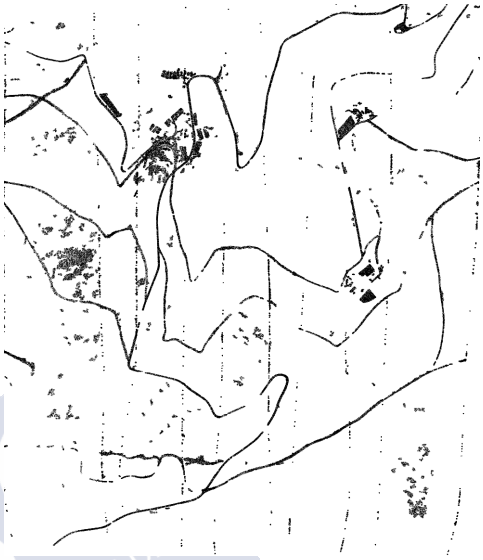
(j) St Arnaud's extracted road points

(k) Toronto's aerial image

(l) Toronto's extracted road points

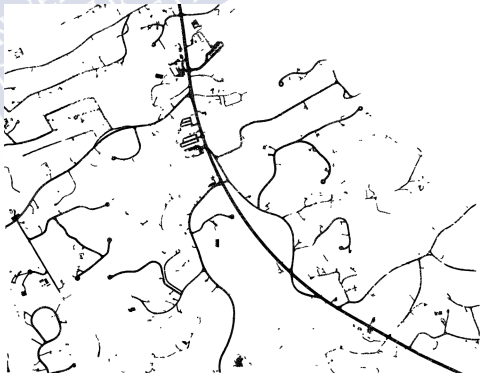
Figure 5.10: Aerial images of the tested data (left) and extracted road points (right).

(m) Trabada's aerial image



(n) Trabada's extracted road points

(o) Truro's aerial image



(p) Truro's extracted road points

Figure 5.10: Aerial images of the tested data (left) and extracted road points (right).

(q) Vaihingen's aerial image

(r) Vaihingen's extracted road points

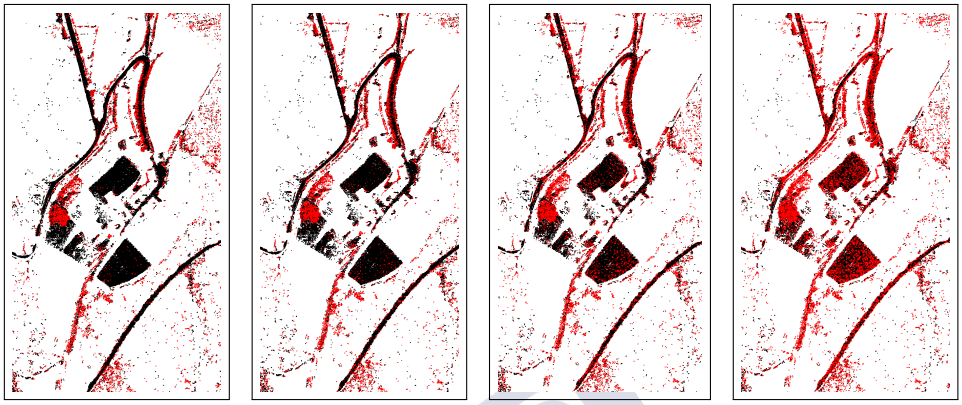
(s) Victor Harbor's aerial image

(t) Victor Harbor's extracted road points

Figure 5.10: Aerial images of the tested data (left) and extracted road points (right).

Figure 5.11: Intensity correction issues in Arzúa. The change in intensity is clearly visible at the overlap of the flight strips (highlighted with red arrows). The road segment inside this high-intensity strip cannot be identified by our method.

In the case of Trabada, there is a noticeable number of false negatives, most of them corresponding to tilled areas. These areas yield a lower intensity than the surrounding ground, and, if planar enough, they can be very hard to distinguish from road surfaces such as parking lots. Although it would be reasonable to expected tilled areas to be significantly less planar than the road surface, particularly in Trabada, this is not the case. We tried to deal with this issue by reducing the curvature threshold, but this change removes both tilled areas and road surfaces (see Figure 5.12). In this particular data, both classes yield similar curvature, and this feature cannot be used to distinguish them. Nevertheless, note that this dataset was acquired fifteen years ago, and we do not expect this behaviour to occur in more recent data.



(a) 0.005 (b) 0.004 (c) 0.003 (d) 0.002

Figure 5.12: Planar (black) and non-planar (red) points for different curvature thresholds. The scene corresponds to an area of east Trabada with tilled areas with nearby road segments.

5.2.4 Computational Performance

From the computational point of view, the method consists of four main stages: The skewness balancing, the intensity and curvature filter, the density filter and the area filter.

First, we measure the impact of using the proposed dynamic step size instead a fixed one for the skewness balancing. Execution times for both approaches are shown in Table 5.3.

Site	Fixed Exec. Time (s)	Dynamic Exec. Time (s)	Speedup
Alcoy	85.7361	18.4806	4.64
Arzúa	238.3762	63.7008	3.74
Carola	82.8657	18.9798	4.37
Logroño	349.6910	81.0136	3.01
St Arnaud	67.5577	12.7313	5.31
Toronto	2.9503	1.8043	1.64
Trabada	108.0673	23.9266	4.52
Truro	64.9537	8.3707	7.76
Vaihingen	30.5339	6.7124	4.55
Victor Harbor	172.5757	28.0137	6.16
Overall	1203.3076	263.7338	4.56

Table 5.3: Execution times of the bidirectional skewness balancing with different step sizes.

The sites whose most benefit are the most intensity-skewed, as this requires a larger number of iterations. Results are very satisfactory, yielding an overall speedup of 4.56.

The execution times of the method for the different sites are shown in Table 5.4 and the overall results are summarised in Table 5.5. Even after the performance improvement in the skewness balancing, it is still the most time consuming stage. All stages are parallelised but the area stage, as it performs a region-growing segmentation (for which the same rationale exposed in Chapter 3 applies). The loop scheduling is the same as the suggested in Chapter 3. In the case of the skewness balancing, as the core computation is inside a while loop, the parallel region is created before the while loop to avoid creating the parallel regions in each iteration, and OpenMP directives are used so that the serial tasks are run by the main thread and the parallel tasks are run by all the spawned threads.

Stage	**Balancing**	**Int. & Curv.**	**Density**	**Area**
Site	**Exec. Time (s) (Exec. Percentage)**			
Alcoy	18.4806 (55.72%)	5.8036 (17.50%)	6.2084 (18.72)%	2,6717 (8,06%)
Arzúa	63.7008 (78.44%)	4.5622 (5.62%)	8.5152 (10.49%)	4.4319 (5.46%)
Carola	18.9798 (55.34%)	11.2325 (32.75%)	1.8077 (5.27%)	2.2757 (6.64%)
Logroño	81.0136 (60.51%)	22.8498 (17.07%)	16.6724 (12.45%)	13.3379 (9.96%)
St Arnaud	12.7313 (67.38%)	2.1112 (11.17%)	2.1238 (11.24%)	1.9284 (10.21%)
Toronto	1.8043 (8.73%)	8.739 (42.26%)	6.3145 (30.54%)	3.8208 (18.48%)
Trabada	23.9266 (48.95%)	12.9904 (26.58%)	7.9118 (16.19%)	4.0461 (8.28%)
Truro	8.3707 (41.08%)	4.3671 (21.43%)	4.4913 (22.04%)	3.1458 (15.44%)
Vaihingen	6.7124 (11.46%)	22.8925 (39.09%)	20.2495 (34.58%)	8.7069 (14.87%)
Victor Harbor	28.0137 (37.10%)	22.8645 (30.28%)	13.9091 (18.42%)	10.7153 (14.19%)

Table 5.4: Computational stages and their execution times for each site.

Procedure	**Exec. Time (s)**	**Exec. Percentage**	**Paralellised**
Balancing	263.7338	50.19%	✓
Int. & Curv.	118.4128	22.54%	✓
Density	88.2036	16.79%	✓
Area	55.0804	10.48%	×
Overall	525.4307	100%	-

Table 5.5: Computational stages and their overall execution times.

The parallelised parts of the program are to 89.52% of the total execution time. Assuming

that the speedup in these parts would be equal to the number of physical cores of the tested computer, the expected theoretical speedup is given by Amdahl's Law is 3.04. The obtained execution times and speedups obtained after parallelisation are shown in Table 5.6. The execution time of the parallelised stages was reduced from 470.3502 s to 134.3503 s, this is a speedup of 2.75 while the execution time of the overall program was reduced from 525.4307 s to 176.0929 s. This is a speedup of 2.32. This result and the ones from previous chapters show that it is possible to obtain a speedup in the parallel parts of the program close to the number of physical cores and that Hyper-Threading provides modest improvements.

Stage	**Exec. Time (s) (Speedup)**				
Threads	**1**	**2**	**4**	**6**	**8**
Balancing	263.7338	165.7448 (1.59)	118.3627 (2.23)	102.3494 (2.58)	95.5096 (2.76)
Int. & Curv.	118.4128	66.5856 (1.78)	50.2015 (2.36)	49.6040 (2.39)	45.3242 (2.61)
Density	88.2036	42.7410 (2.06)	30.4205 (2.90)	31.3501 (2.81)	29.9186 (2.95)
Overall	470.3502	275.0714 (1.71)	198.9847 (2.36)	183.3035 (2.57)	170.7524 (2.75)

Table 5.6: Execution times and speedups of the paralellised stages.

5.3 Discussion

In this section, we show how important is the calculation of the intensity threshold, we analyse how the road width parameter affects the quality of the results, we highlight some limitations of the method, and we compare the obtained results with existing methods.

5.3.1 Intensity Threshold

It is worth to mention some insights about the intensity threshold calculation. Relevant information of the pipeline filtering is shown in Table 5.7. On the one hand, it is important to highlight the importance of removing the intensity outliers before calculating the skewness. The balancing direction (B_{dir}) depends on the sign of the skewness, thus its calculation is crucial. Take as examples the Carola, Truro, and Victor Harbor sites. The initial skewness (Sk_{Init}) is positive but, after removing the outliers with IQR (Sk_{IQR}), it changes to negative, which also changes the balancing direction. Without this correction, it would not be possible to calculate the optimal intensity threshold. On the other hand, the calculated intensity threshold (I_T) varies greatly across point clouds. By normalising each threshold within the [0, 255]

range ($I_{T\in[0,255]}$) to provide comparable values, we can observe that road points are expected to yield an intensity lower than half of the intensity range, after removing the outliers. This can be considered as a rule of thumb. However, the optimal intensity threshold can be significantly lower, and its value is unique for each data. Therefore, a fully automatic method, like the one presented in this thesis, is needed.

Site	$\mathbf{Sk_{Init}}$	$\mathbf{Sk_{IQR}}$	$\mathbf{Sk_{PCT}}$	$\mathbf{B_{Dir}}$	$\mathbf{I_T}$	$\mathbf{I_{T\in[0,255]}}$	**Road points (%)**
Alcoy	-0.705	-0.749	-0.947	Forward	103	108	18.24
Arzúa	-0.937	-1.174	-1.570	Forward	790	135	4.15
Carola	81.564	-0.379	-0.593	Forward	25	82	15.47
Logroño	-0.867	-0.997	-1.224	Forward	17760	119	16.80
St Arnaud	-1.542	-1.564	-1.682	Forward	26609	109	2.89
Toronto	3.327	0.971	1.013	Backward	9	38	41.00
Trabada	-0.647	-0.647	-0.873	Forward	103	103	19.35
Truro	1.470	-0.135	-0.530	Forward	154	72	17.32
Vaihingen	0.434	0.398	0.183	Backward	71	71	71.10
Victor Harbor	10.327	-0.470	-0.627	Forward	12	91	14.62

Table 5.7: Skewness balancing insights and results of the road extraction.

5.3.2 Challenging Conditions

The road extraction may struggle in the presence of water and wetland, where the intensity values can be as low as in the road surface. For instance, in Victor Harbor site, a water area in the south-east part of the site is wrongly labelled as road surface (see Figure 5.10t). Deepwater normally does not reflect the laser beam, but LiDAR points can be recorded near the coast or in the presence of floating sediment. The topic of detecting water is out of the scope of this work, so in cases of coastal areas, a water extraction procedure should be applied beforehand. A more challenging situation is the existence of wetland. In the St Arnaud site, the surrounding ground of a narrow river is wrongly labelled as road surface (see Figure 5.10j). While for this particular site the amount of wetland is relatively small, sites with large wetland areas could be an issue, although it is reasonable to expect roads to exhibit higher linear features than wetland areas.

5.3.3 Road Width Parameter

An important matter is the selection of the minimum road width (MRW) parameter. To illustrate its influence, we measure the quality obtained using different values for the Vaihingen and Truro sites (see Figure 5.13). The road width is subject to government regulations with typical values ranging from 2 m to 6 m, but for the sake of analysis, we also used larger values. Using typical values produced similar qualities, only when using large values the quality decreases significantly, especially in Truro, because the road width is typically smaller in less urbanised landscapes. Also, note that this parameter is not a constraint for the road width itself, but rather a reference for three stages of the pipeline filtering, so roads with both smaller and larger widths will also be extracted.

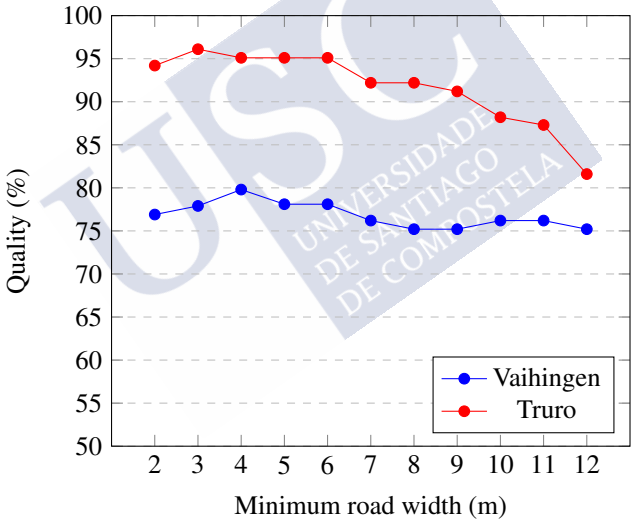


Figure 5.13: Influence of MRW parameter in the quality of the results for Vaihingen and Truro study sites.

5.3.4 Comparison with Other Methods

The completeness, correctness, and quality that recent studies have achieved are summarised in Table 5.8. Note, however, that it is difficult to compare the methods, as authors use different datasets. Furthermore, in some studies, LiDAR-derived features such as elevation and

intensity are rasterised so that processing and evaluation are carried out at pixel-level instead of point-level, which leads to a loss of information. Our method yields the highest completeness and it is the third in terms of correctness and quality, only surpassed by CRF methods [184, 183]. Nonetheless, our method offers clear advantages, first, it does not need training as CRF methods do, and second, it has been tested in a wide range of scenarios, while CRF methods have been tested only in three small regions of the urban site of Vaihingen. Notice that the lack of test sites is a prevailing issue among studies, and experiments are often carried out in one study site only, typically urban.

Author	$\mathbf{C_p}(\%)$	$\mathbf{C_r}(\%)$	$\mathbf{Q}(\%)$	No raster	Study sites
Clode et al. (2007) [46]	83.50	73.50	63.50	×	2 (Fairfield & Yeronga)
Samadzadegan et al. (2009) [211]	53.94	56.64	53.10	×	1 (Castrop-Rauxel)
Jiangui and Guang (2011) [122]	60.35	66.81	NA	✓	1 (Shashi)
Azizi et al. (2014) [18]	75.07	63.02	52.11	×	1 (Golestan)
Matkan et al. (2014) [164]	85.34	71.54	63.56	✓	1 (Rheine, 3 areas)
Niemeyer et al. (2014) [184]	87.08	**93.04**	**81.75**	✓	1 (Vaihingen, 3 areas)
Niemeyer et al. (2015) [183]	90.40	87.30	79.90	✓	1 (Vaihingen, 3 areas)
Li et al. (2015) [145]	92.94	75.50	71.41	✓	1 (Vaihingen)
Proposed method	**93.00**	83.00	78.00	✓	**10** (see Table 5.1)

Table 5.8: Quality metrics of road extraction compared to other methods.

5.4 Conclusions

In this chapter, a method for the automatic extraction of road points is proposed. The only parameter required to the user is a rough estimation of the minimum road width. Road points are identified by a pipeline filtering based on intensity, curvature, local density, and area constraints. We showed the relevance of the intensity feature, and we presented an improved algorithm for its automatic thresholding. Experiments conducted in a dataset with ten study sites showed that the method is suitable regardless of the landscape characteristics and the acquisition procedure. Road points were extracted even for sites with little presence of roads. Some limitations were also exposed, in particular, low-intensity areas such as tilled areas, wetland, and water can introduce false positives. Also, we showed that the selection of the minimum road width parameter is not critical, having little effect in the quality of the results as long the value is within a typical range. Overall, tests demonstrate the suitability of our approach as a general method for road point extraction. As the main difficulty when extracting

the road model is to first extract the road points, our contribution offers a competitive result in this confrontation.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 6

MODELLING OF THE ROAD NETWORK

Once the road points have been identified, these can be used to build the road model. Most of the emphasis in road modelling is focused on obtaining the road centrelines (i.e. delineation), although other features such as width and slope are also desirable. Well known methods for this task include the Phase Coded Disk (PCD) [46], Template Matching [110], Hough Transform [111] and skeletonisation [115], which are all based in image processing techniques [47, 34, 60, 225]. This task is still highly challenging if using LiDAR data only, as the non-road impervious surface is easily misidentified as roads [276]. Tests in the ISPRS Vaihingen dataset show that only one method achieved a quality value above 50%. Because of this, is very important to explore and develop more methods for the extraction of the road centrelines. One image processing technique whose application is still to be exploited in the LiDAR field is the Angular Texture Signature (ATS) [55]. The ATS has been used in road-related applications for IKONOS and Quickbird satellite imagery, such as road extraction [81], filtering of parking lots [273] and road tracking [146, 147]. The ATS, or some of its basic concepts, could be used for centreline extraction from the raw LiDAR data.

Another issue regarding the quality of the road model is the presence of gaps in the extracted centreline, often due to the occlusion of the road by objects such as cars or trees. Road segments may be partially or totally occluded by these objects, leading to an incomplete road model. The topic of dealing with this problem often referred to as gap filling, can be addressed with different techniques. Small occlusions might be handled by a morphological closing operation, however, larger occlusions require the use of more sophisticated techniques such as Markov Random Field (MRF) [278], hierarchical grouping [111], or Radon Transform and

spline interpolation [164]. Another image processing technique whose application remain little explored in the LiDAR field is the active contours models, where the problem is formalised as an energy minimisation one [130]. Because of the way the contours move while the energy is minimised they are commonly referred to as *snakes*. Snakes have been used extensively in image processing [260, 29], and some authors have applied them to LiDAR data for different task, such as road edges extraction [137], segmentation [13], building extraction and modelling [126, 4, 261], or tree detection and crown delineation [255]. However, snakes are not applied into the raw 3D LiDAR data but in a projected 2D image of the data instead, and its worth to conduct more research on how to apply them for filling the gaps in the road model.

In this chapter, we present a framework to construct the model of the road network, to extract the road centreline, without gaps and derive the road width. For this, we introduce a road tracking method based on an adaptation of the ATS for extracting the road centreline, a gap-filling method based on snakes that is applied into the raw LiDAR data, and a method for the estimation the road width from the centreline.

The chapter is organised as follows. Section 6.1 describes the framework for road modelling in detail and shows experimental results. Some challenges in the road tracking and gap filling are discussed in Section 6.2. Conclusions are given in Section 6.3.

6.1 Method

An overview of the method is shown in Figure 6.1. The input of the method is the road points, previously identified in Chapter 5, and its boundary points, calculated using the Alpha Shapes algorithm described in Chapter 3. First, a road tracking procedure is performed to extract the road centreline. Then, the possible missing roads are reconstructed by gap-filling procedure. Finally, the width of the road is estimated using the extracted centreline.

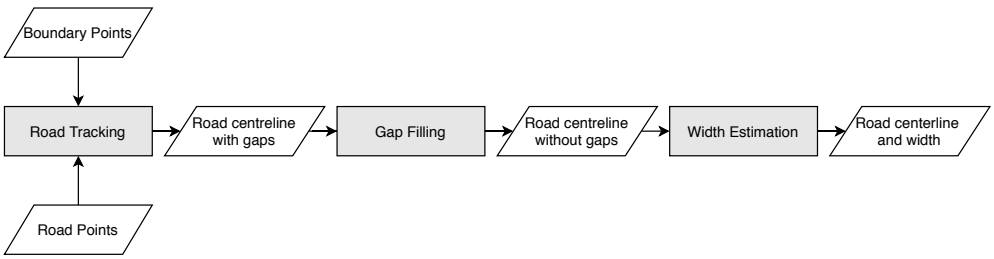


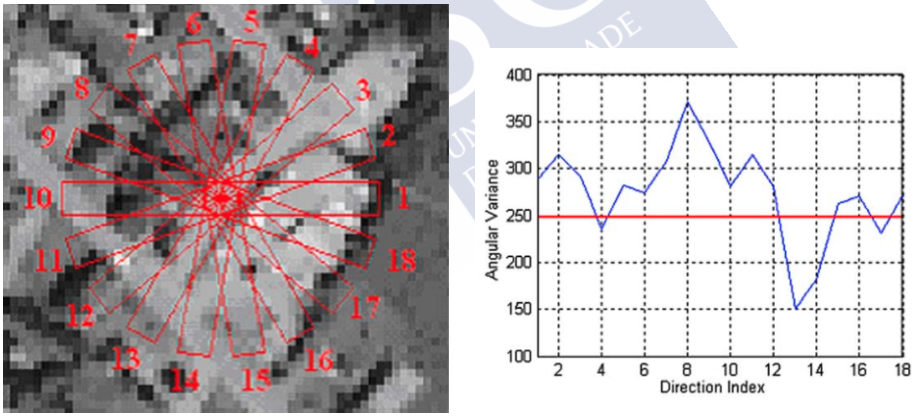
Figure 6.1: Overview of the road modelling.

6.1.1 Road Tracking

Road tracking is conducted based on the concept of the ATS. The ATS is a texture measure, initially used for road detection, that has two components: the degree of the pixel (i.e. the number of the strong local minima in the ATS) and the direction of the minimum. The idea is to build a tracker that uses this texture measure as guidance to travel through the road points.

Original ATS

The original definition of the ATS is as follows: At each pixel p of a panchromatic image, $T(\alpha, w, p)$ is defined as the variance from the mean for a rectangular set of pixels of width w around the point p whose principal axis lies at an angle of α from the horizontal. This measure is computed for a set of angles $\alpha_0, \cdots, \alpha_n$. At the point p, the ATS is defined as the set of values $T(\alpha_0, w, p), T(\alpha_1, w, p), \cdots, T(\alpha_n, w, p)$. The ATS is depicted in Figure 6.2.



(a) (b)

Figure 6.2: ATS: (a) Set of rectangular regions centred for a given pixel and (b) the graph of its ATS value. The size of the rectangular regions is 5 by 20 pixels. Image from [273].

LiDAR Adaptation

Considering that our input is the road points, we make the following adaptations to the concept of ATS to perform road tracking. First, we change the main computation. A straightforward adaptation would be to compute the variance over the intensity values of the points, instead

of using the grey values in the pixel images. However, we compute the number of road points within each rectangle instead. It is assumed that the rectangle containing the higher number of road points indicates the direction of the road. And second, we only use half the rectangle structure, in a hand fan fashion, as it is assumed that the direction of the road, although is subject to changes, cannot be backwards. The new ATS definition is then as follows: At each point p of a point cloud, $T(\alpha, w, h, p, \overrightarrow{d})$ is defined as the number of road points for a rectangular neighbourhood of width w and of height h around the point p whose principal axis lies at an angle of α from the horizontal and has an implicit direction vector $\overrightarrow{d}$. This measure is computed for a set of angles $\alpha_0, \cdots, \alpha_n$. At the point p, the ATS is defined as the set of values $T(\alpha_0, w, h, p, \overrightarrow{d}_0), T(\alpha_1, w, h, p, \overrightarrow{d}_1), \cdots, T(\alpha_n, w, h, p, \overrightarrow{d}_n)$. The new ATS is depicted in Figure 6.3. Note that for this particular application, we count the number of road points, but the new ATS definition is suitable to perform tracking of any type of points.

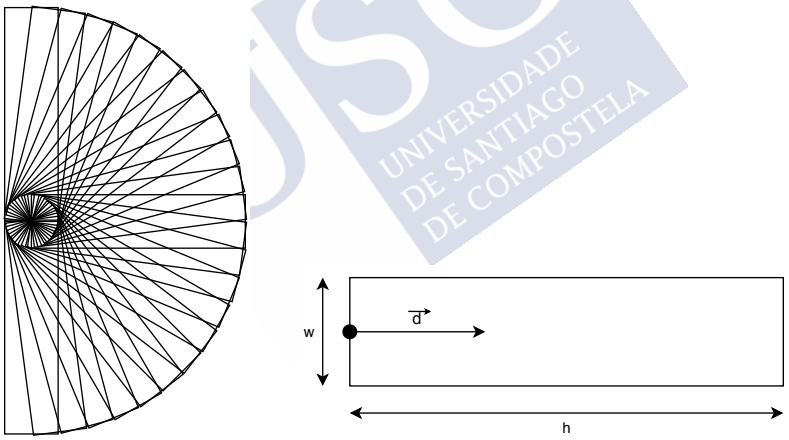


Figure 6.3: Adapted ATS: (a) Set of rectangles and (b) rectangle features.

The basic rule for setting w and h is that the width of the rectangle should be less or equal to the typical road width, and the height should be at least twice the width [273]. The angle step α_{step} determines the set size as $|ATS| = 180^\circ/\alpha_{step} + 1$.

Local Maximum Detection

One road path might split into other paths (e.g. bifurcations, junctions,...) so it is necessary to take into account not only the ATS maximum but the ATS local maxima. However, it is expected for nearby local maxima to belong to the same road path. Therefore, a Non-Maximum Suppression (NMS) strategy must be performed to extract only the local maxima that correspond to unique road paths. NMS strategies have been used broadly in computer vision tasks such as object detection. A common practice is to assume that highly overlapping detections belong to the same object and collapse them into one detection [107].

The proposed NMS strategy is based on this assumption as well, and it consists in the reduction of values within the neighbourhood of the maxima. For an ATS rectangle identified as local maximum, also known as *peak*, we set as its neighbourhood the ATS rectangles that fall within a [-22.5º, 22.5º] range (i.e. a suppression angle $\alpha_{sup} = 45°$), so that the number of neighbouring rectangles affected is $n = \alpha_{sup}/\alpha_{step}$ in total, half in the clockwise direction and half in the counterclockwise direction in respect to the ATS rectangle direction. The value of the peak is reduced by 100% (i.e. total suppression) and the values of its neighbours are reduced gradually, the further the less, until a defined minimum reduction of ω_{min}. The reduction ω for a given neighbour $i = 1, 2, \cdots, \frac{n}{2}$ is derived as shown in Equation 6.1

$$\omega_i = 100 - (i * \omega_{step}) \tag{6.1}$$

where ω_{step} is the reduction step defined as:

$$\omega_{step} = \frac{100 - \omega_{min}}{n/2} \tag{6.2}$$

Once the NMS strategy is defined, the identification of peaks is performed. For each ATS, a set of peaks P is obtained by a recursive procedure. To keep only significant peaks, subsequent peaks must yield a value above a certain percentage λ of the previous peak value. The process is as follows:

1. Find the maximum ATS rectangle ATS_{max}.
    - If $|P| = 0$, then ATS_{max} is a peak and it is added into P.
    - If $|P| > 0$, then compare ATS_{max} with the last added peak P_{last}.
        - If $ATS_{max} > \lambda P_{last}$, then ATS_{max} is a peak and it is added into P.

  - Else, then the process ends.

2. Perform the NMS at ATS_{max}.

ATS Tracking

The road tracking algorithm, that uses the new ATS and the NMS strategy, is then as follows:

1. Select a seed point p_0 and calculate the initial direction. The seed point is set to be the first boundary point. The initial direction is obtained by creating a set of rotated rectangles over 360º around p_0, instead of using the aforementioned hand fan (180º) set. The rectangle containing the highest ATS value determines the initial direction.

2. Identify all possible road directions from the current point p. Create a hand fan set of rotated rectangles over a [-90º, 90º] angle range from the initial direction of p. The road directions $D = \{\overrightarrow{d_0}, \overrightarrow{d_1}, \cdots, \overrightarrow{d}_n\}$ $n \in [0, |ATS|]$ are determined by identifying the local maxima in the ATS as explained earlier. The centreline points $C = \{c_0, c_1, \cdots, c_n\}$ are derived by these directions. The derivation of a centreline point c_i is done by shifting the ATS origin point thought $\overrightarrow{d_1}$ by an offset c_{step}. This offset control how much the road centreline is discretised.

3. Set the next centreline point. A queue Q is used to keep track of the unprocessed centreline points. The process is as follows:

   - If $|C| > 0$, then c_0 is set as the current point and the rest (if any) are added into Q. Return to step 2.
   - Else if $|C| = 0$, then,
     - If $|Q| > 0$, then the first point in Q is set as the current point. Return to step 2.
     - Else, then the process ends.
   - If the ATS value of the selected current point is significantly lower than the mean of the ATS of the previously selected centreline points ($ATS_p < \gamma\overline{ATS}$), then the process ends.

A depiction of the ATS tracking is shown in Figure 6.4 for a road segment in the Alcoy site. For the current point being processed, the ATS set centred in the point is used for identifying possible road directions and their derived road centreline points are extracted.

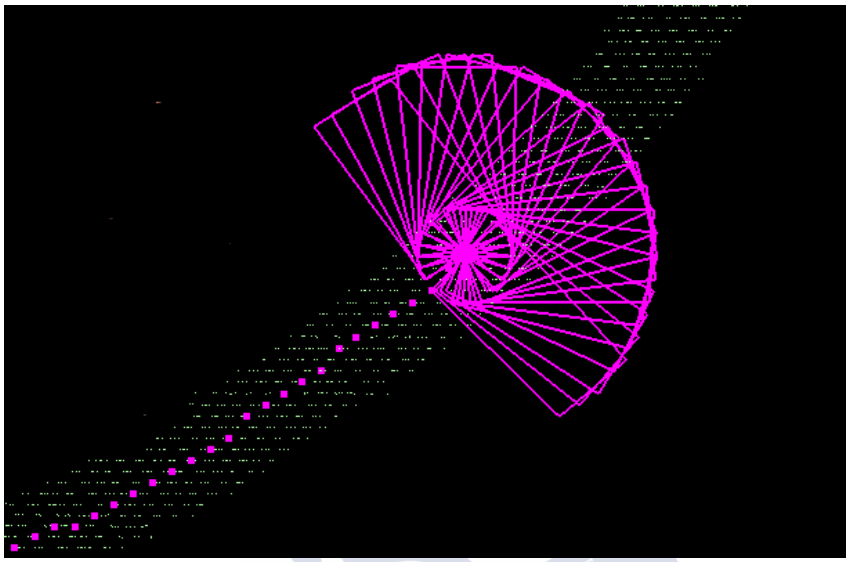


Figure 6.4: ATS road tracking: ATS rectangle set and extracted centreline points (in pink).

Experimental Results

Experiments were carried out in an area of the Alcoy and Carola sites described in previous chapter.The road points, extracted by the method described in the previous chapter, were given as input. The parameters used are shown in Table 6.1.

Parameter	**Description**	**Value**
w_{ats}	ATS rectangle width	6 m
h_{ats}	ATS rectangle height	12 m
α_{step}	ATS Angle step	7.5°
c_{step}	Centreline points offset	1 m
ω_{min}	NMS Minimum reduction	0.50
λ	NMS Next peak minimum	0.65
γ	Minimum centreline ATS relative to mean	0.35

Table 6.1: ATS tracking parameters.

The centrelines were successfully extracted from the road tracking procedure (see figures 6.5 and 6.6). Most of the road paths have been identified with two exceptions highlighted

in white. It seems that the estimated centreline is slightly displaced from the real centreline in some cases. This could be due to the influence of the point distribution in the calculation of the ATS. Also, there is an issue with large asphalt surfaces such as parking lots. In such scenarios, the ATS value is always high for any ATS rectangle, so that the identified road direction is rather random.

Figure 6.5: Extracted centreline for a road segment.

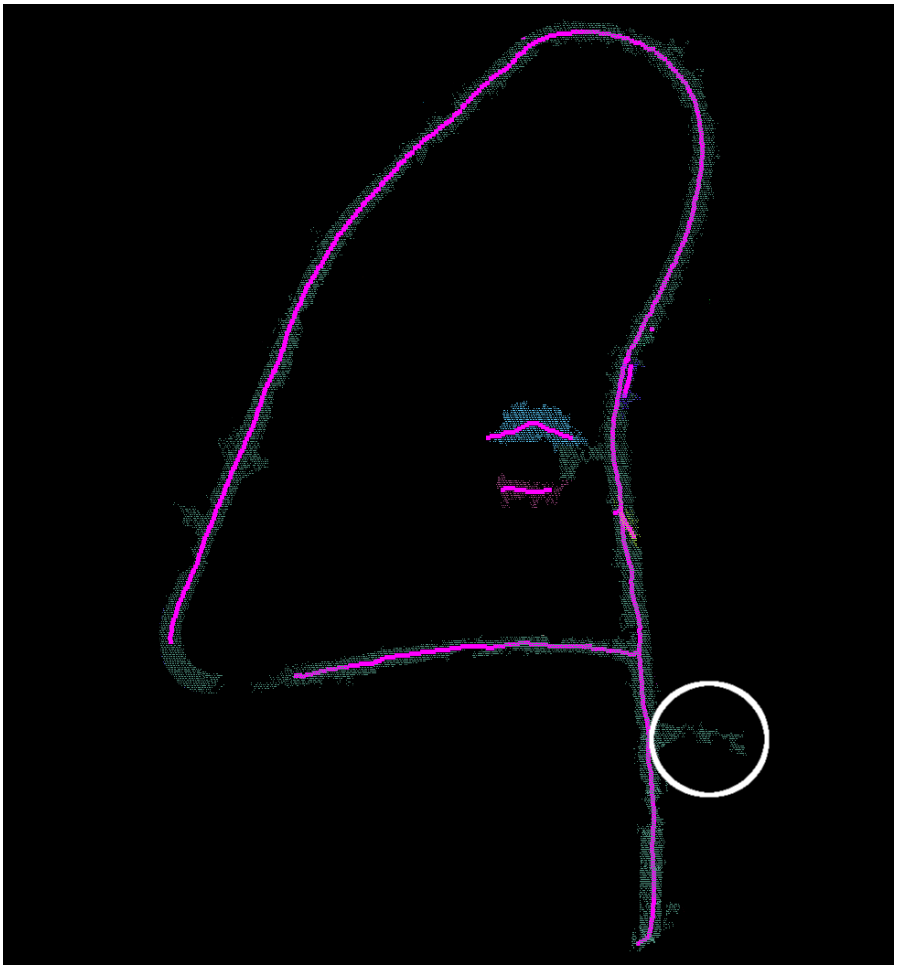


Figure 6.6: Extracted centreline for a road segment.

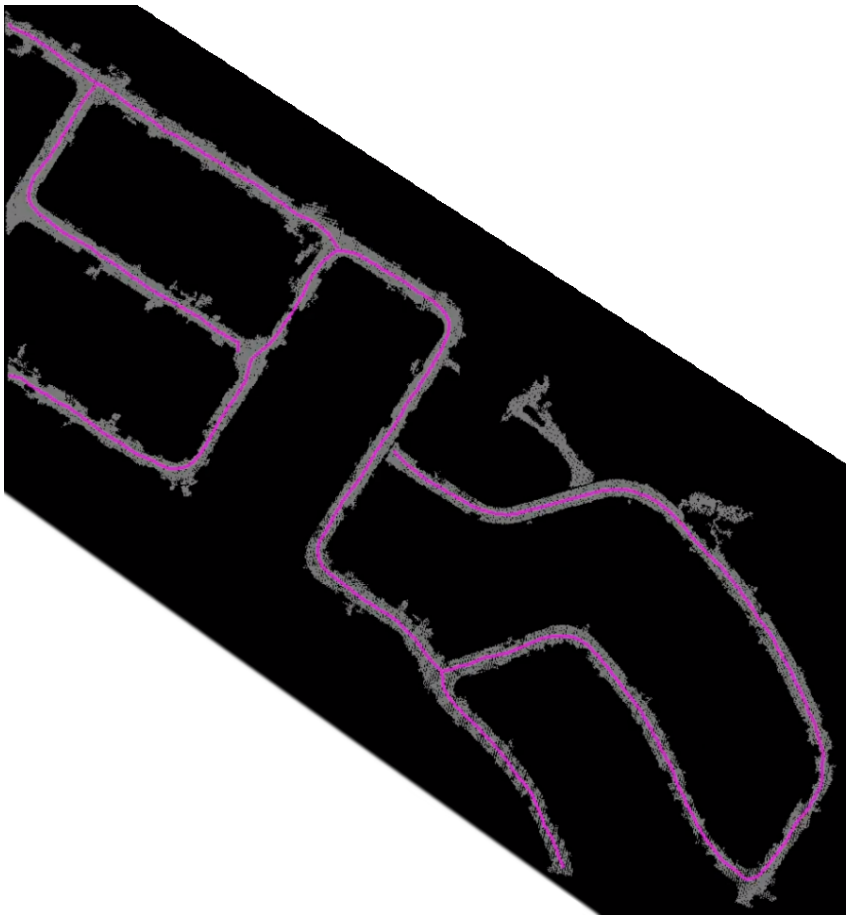


Figure 6.7: Extracted centreline for a road segment.

6.1.2 Gap Filling

The gaps in the road centreline are filled with the use of snakes. A snake is an energy-minimising spline guided by internal and external forces that move it towards certain features like lines and edges. We assume that there is a significant intensity difference between the road points and the surrounding points so that the external force can be set to be attracted to these areas with a large intensity gradient. This assumption is reasonable, given the fact that

road points yield a particularly low intensity compared to other surfaces, as previously shown in Chapter 5. The idea then is to initialise the snake at the nearest known road points from the gap, that guided by geometric constraints and the intensity gradient, will converge into the shape of the missing road segment.

Original Snakes

Let us define the traditional snakes. Their contour is represented by the parametric curve:

$$v(s) = (x(s), y(s)),\ s \in [0,1] \tag{6.3}$$

The snake moves through the image's spatial domain to minimise the energy functional:

$$E^*_{snake} = \int_0^1 E_{snake}(v(s))\,\mathrm{d}s = \int_0^1 (E_{int}(v(s)) + E_{ext}(v(s)))ds \tag{6.4}$$

where E_{int} is the internal energy and E_{ext} is the external energy. The internal energy controls the deformation of the snake by imposing geometric constraints such as elasticity and stiffness, while the external energy guides the snakes towards the image features.

The internal energy is defined as:

$$E_{int} = E_{cont} + E_{curv} \tag{6.5}$$

where the first term E_{cont} represents the continuity of the contour, making the snake act as a membrane and the second term E_{curv} represents the smoothness of the contour, making the snake act like a thin-plate. This can be expanded into the following equation 6.6:

$$E_{int} = \frac{1}{2}\left(\alpha(s)|v'(s)|^2 + \beta(s)|v''(s)|^2\right) \tag{6.6}$$

where α and β are the weights for the first-order term and second-order term that penalise the contour deformation, and $v'(s)$ and $v''(s)$ are the first and second-order derivatives of $v(s)$ respect to s, respectively. In practice, larger values of α penalise stretching, i.e. reduces elasticity, and larger values of β penalise bending, i.e. increases stiffness.

The external energy is defined as:

$$E_{ext} = E_{img} + E_{cont} \tag{6.7}$$

where first term E_{img} is the energy of the image. This term is where distinct snake implementations vary the most. The second term E_{cont} is the constricting energy, which is interactively defined by the user allowing to guide the snake. This term is often ignored as it removes automation. In this work, we also opt for not using E_{cont} in the external energy calculation.

The energy of the image and its three terms are traditionally formulated as:

$$E_{img} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term} \tag{6.8}$$

$$E_{line} = I(x,y) \tag{6.9}$$

$$E_{edge} = -\nabla|I(x,y)|^2 \tag{6.10}$$

$$E_{term} = \frac{\delta\theta}{\delta n_{\perp}} \tag{6.11}$$

where w_{line}, w_{edge} and w_{term} are the weights for the E_{line}, E_{edge} and E_{term} terms respectively, $I(x,y)$ is the intensity of the image at coordinates (x,y), ∇ is the gradient operator, θ is the gradient angle $\theta = \tan^{-1}(C_x/C_y)$ with $C(x,y) = G_\sigma(x,y) * I(x,y)$, being G_σ a 2D Gaussian function with standard deviation σ, i.e., $C(x,y)$ is the image smoothed the larger the σ the more smoothed, and $n_\perp$ is the unit vector perpendicular to the gradient direction $n_\perp = -(\sin\theta, \cos\theta)$.

A snake that minimises the energy functional of Equation 6.4 satisfies the Euler equations:

$$\begin{aligned} \alpha x'' + \beta x'''' + \frac{\partial E_{ext}}{\partial x} = 0 \\ \alpha y'' + \beta y'''' + \frac{\partial E_{ext}}{\partial y} = 0 \end{aligned} \tag{6.12}$$

A derived Euler equations can be written in matrix form as:

$$\begin{aligned} Ax + f_x(x,y) = 0 \\ Ay + f_y(x,y) = 0 \end{aligned} \tag{6.13}$$

where A is the internal force (a pentadiagonal banded matrix) and f the external force.

To find a solution, the snake is made dynamic by treating $v(s)$ as a function of time t as well as s, i.e. $v(s,t)$. Then the partial derivative of v respect to t is set equal to the left-hand side of Equation 6.12. Finally, the position of the contour at a time t is then calculated as:

$$\begin{aligned} x_t &= (A+\gamma I)^{-1}(x_{t_1} - f_x(x_{t-1}, y_{t-1})) \\ y_t &= (A+\gamma I)^{-1}(y_{t_1} - f_y(x_{t-1}, y_{t-1})) \end{aligned} \tag{6.14}$$

where γ is the step size which controls the deformation speed. The matrix $A+\gamma I$ is a pentadiagonal banded matrix, so its inverse can be calculated by LU decomposition in $\mathrm{O}(n)$ time.

This calculation is performed until the snake finds a stable position, i.e. converges. This can be identified when there is little to none difference between two consecutive iterations or it can also be imposed by setting a maximum number of iterations i_{max}, like in our case.

LiDAR Adaptation

Some adaptations must be performed to use the snakes in the LiDAR point cloud. For our road reconstruction purposes, we stick to consider only the 2D space. Taking into account the height component will, arguably, add unjustified complexity to the snakes, as height values of the resulting contour can be easily derived from the nearby road segments.

First, it is needed to redefine the intensity function $I(x,y)$. In this regard, both LiDAR points and image pixels have grey-scale intensity values, thus this adaption is straightforward: $I(x,y)$ is now defined as the intensity of a LiDAR point at coordinates (x,y).

Second, a method for calculating the partial derivatives is needed. In images, the partial derivatives can be easily obtained by convolution of a kernel over the 2D pixel grid. However, the LiDAR points are irregularly distributed, making this approach unsuitable. Because of this, we propose a method to approximate the partial derivatives by applying finite differences inside the neighbourhood of a given point. For a point p, we examine the neighbouring points $(p_0, p_1, \cdots, p_n)$ inside a sphere or radius r_δ, and the adjacent variable points (x_{i-1}, x_{i+1}) and function values (f_{i-1}, f_{i+1}), are derived as follows:

$$\begin{aligned} A &= \{p_i | x_i < x_p\},\ x_{i-1} = \frac{1}{|A|}\sum x_i[p_i \in A],\ f_{i-1} = \frac{1}{|A|}\sum f_i[p_i \in A] \\ B &= \{p_i | x_i > x_p\},\ x_{i+1} = \frac{1}{|B|}\sum x_i[p_i \in B],\ f_{i+1} = \frac{1}{|B|}\sum f_i[p_i \in B] \end{aligned} \tag{6.15}$$

Then the partial derivative is approximated by the central finite difference as:

$$\left[\frac{\delta f}{\delta x}\right]_i \approx \frac{f_{i+1} - f_{i-1}}{x_{i+1} - x_{i-1}} \tag{6.16}$$

where in the case of regularly distributed points, the denominator $x_{i+1} - x_{i-1} \approx 2\Delta x \approx r_\delta$.

A depiction of the method to calculate the partial derivative of $I(x,y)$ in respect to x at a point p is shown in Figure 6.8. Conceptually, the adjacent variable points of Equation 6.15 are the left and right centroids (C_{left}, C_{right}) of the points, respectively. The calculation in respect to y is analogous but using the upper and lower centroids instead. An example of the intensity gradient calculated from the proposed partial derivatives is shown in Figure 6.9.

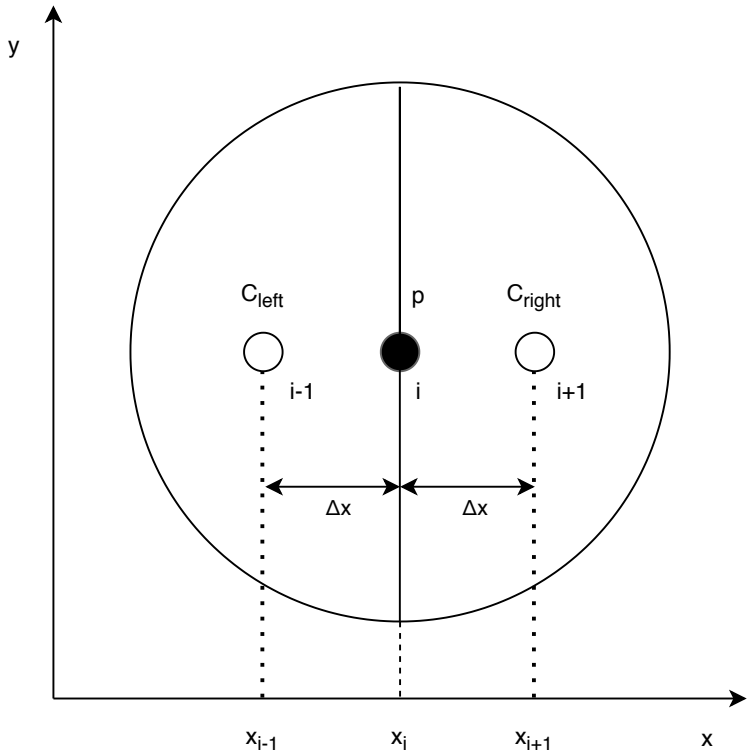


Figure 6.8: Visualisation of the finite central difference calculation. Note that for visualisation purposes, it has been assumed that the points inside the neighbourhood are regularly spaced.

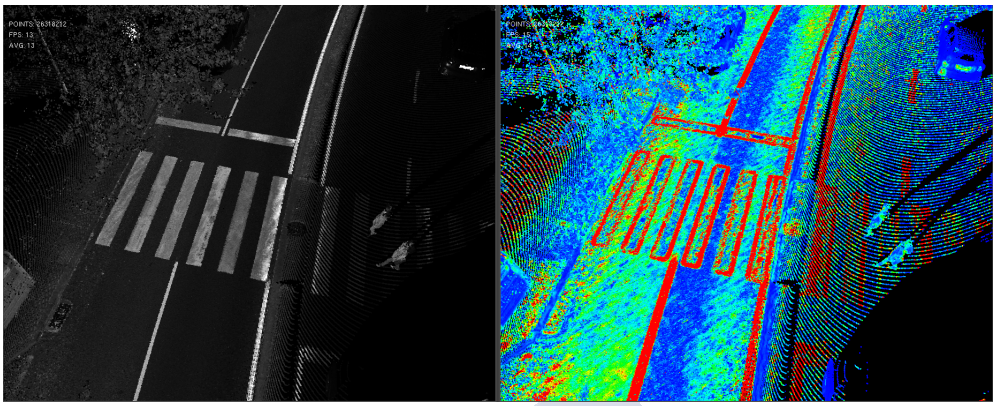


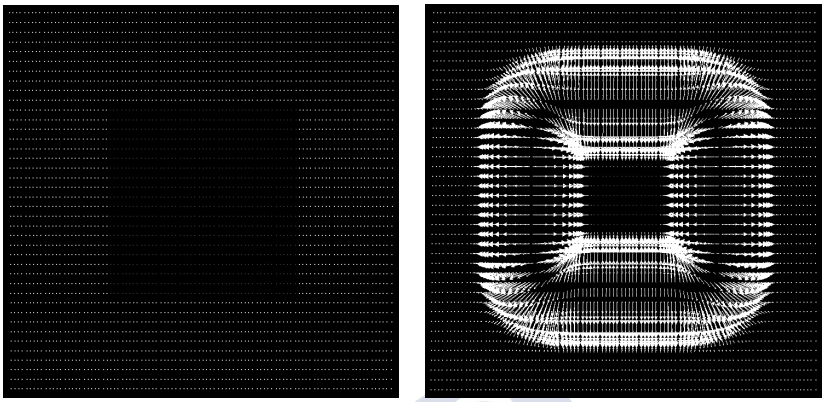
Figure 6.9: Application of the partial derivatives: (left) LiDAR data coloured by intensity and (right) magnitude of the intensity gradient coloured in a cold-to-heat heat map. It can be seen how the areas with the largest intensity gradient correspond to the surroundings of the road markings, where materials with very high intensity (painted asphalt) and very low intensity (raw asphalt) coincide.

Third, because of the aforementioned irregular distribution, it is unlikely for the contours points to move into existing points, so that it is needed to calculate the external force at coordinates where no points exist. In such cases, the external force is interpolated from the nearby exiting points. For this, we use the Inverse Distance Weighting (IDW) method [217]. For an artificial point p at coordinates (x,y), its external force f is calculated from their exiting neighbouring points $P = \{p_1, p_2, \cdots, p_n\}$ that are inside a sphere of radius r_f as:

$$f = \frac{\sum_1^n w_i f_i}{\sum_1^n w_i}, \; w_i = \frac{1}{d(p,p_i)^2} \tag{6.17}$$

where w_i is weight of p_i and $d(p,p_i)$ is the euclidean 2D distance from p to p_i. A recommended value for setting the interpolation radius r_f is between $[r_\delta, 2r_\delta]$.

A depiction of the resulting external forces from Equation 6.7 using these adaptations is shown in Figure 6.10. A synthetic small 2D point cloud was created, which consists of a square-shaped set of points with high intensity (in white) with an inner square-shaped set of points with low intensity (in grey). It can be seen how the external forces originate at the boundary of both sets of points, i.e., the area with a large intensity gradient, pointing outwards. Because the snake seeks to minimise this energy, it will be attracted to these areas.



(a) (b)

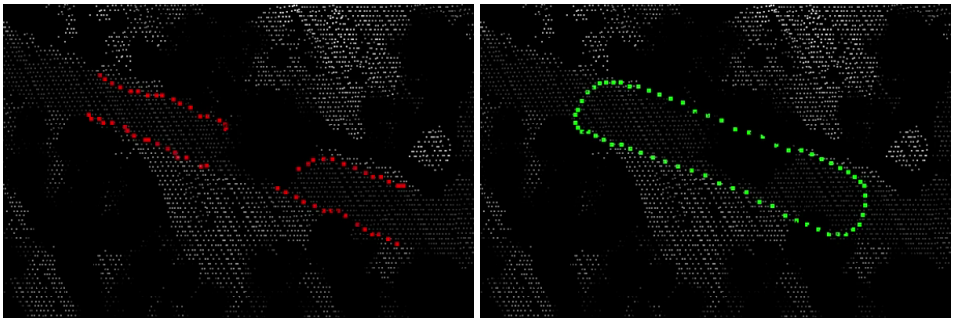
Figure 6.10: External forces: (a) synthetic data and (b) resulting external forces (white arrows).

Experimental Results

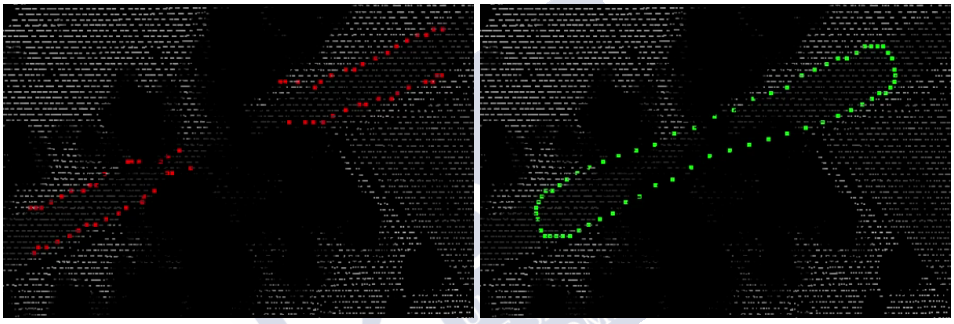
This method was tested in the Alcoy site with the parameters shown in Table 6.2. The results of the gap-filling are shown in Figures 6.10. It can be observed how the contour points adapt successfully towards the road boundary, even in the presence of large gaps (see Figure 6.10c and 6.10e), if the initial contour is rather abrupt (see Figure 6.10d), or when performed in a road curve (see Figure 6.10f).

Parameter	Description	Value
α	Stretching penalty	0.5
β	Bending penalty	0.2
w_{line}	Line attraction weight	0.14
w_{edge}	Edge attraction weight	2.0
w_{term}	Termination attraction weight	0.01
γ	Deformation speed	1
r_δ	Neighbourhood radius for partial derivatives	1 m
r_f	Neighbourhood radius for force interpolation	2 m
i_{max}	Maximum number of iterations	500

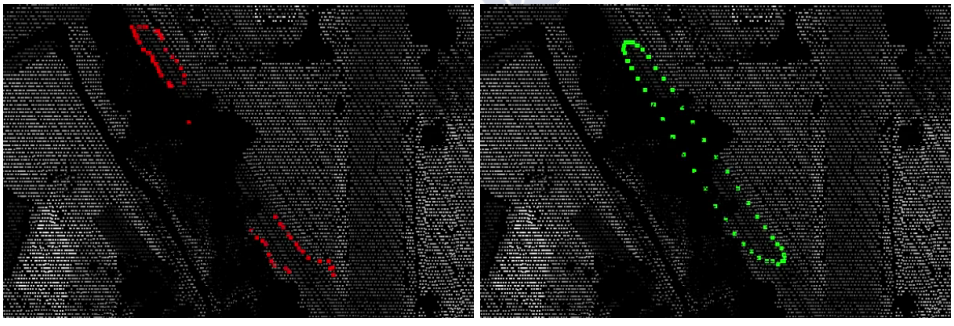
Table 6.2: Snakes parameters.



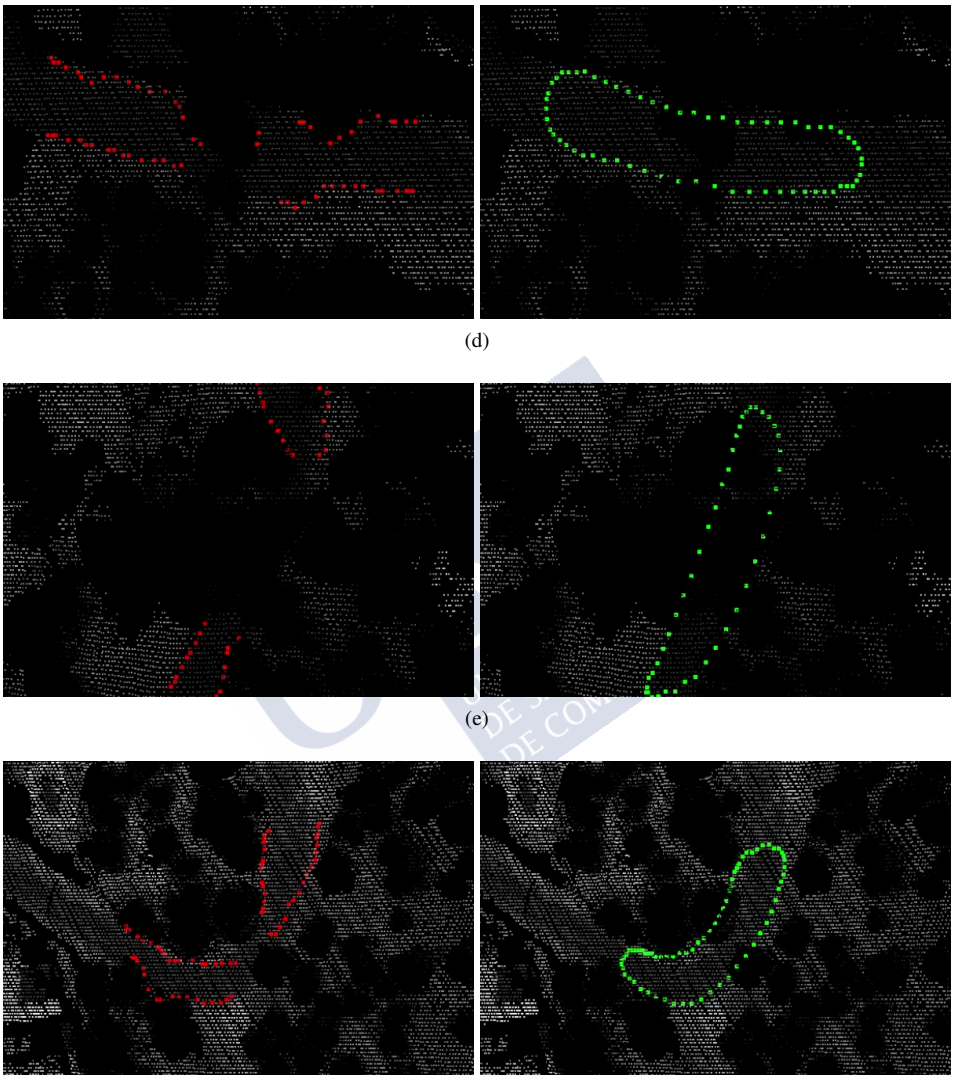
(a)



(b)



(c)



(d) (e) (f)

Figure 6.10: Snakes results for some road gaps in the Alcoy site. The LiDAR points classified as ground and road, coloured by intensity, are shown. The initial contours are shown in the left side (in red) and the final contours are shown on the right side (in green).

6.1.3 Road Width Estimation

Once the road centreline has been completely extracted, the road width can be derived geometrically from the centreline and the boundary points. A depiction of the procedure is shown in Figure 6.11. For each pair of centreline points $\{c_i, c_{i+1}\}$, the procedure is as follows:

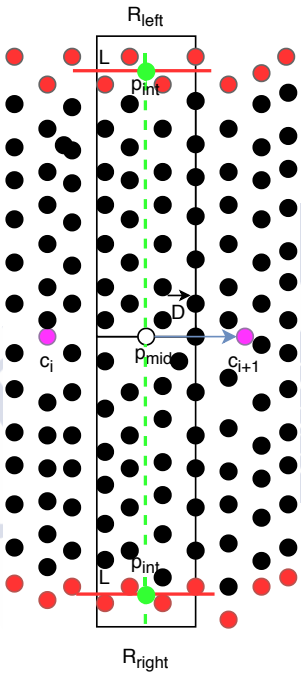


Figure 6.11: Geometric derivation of the road width. Centreline points (in violet), road points (in black), and boundary points (in red).

1. Calculate the middle point p_{mid} between $\{c_i, c_{i+1}\}$.
2. Create a pair of rectangles $\{R_{left}, R_{right}\}$ perpendicular to p_{mid} direction $\overrightarrow{D}$. Each rectangle has a height h and a width w and a direction vector $\overrightarrow{d}$, as in the ATS.
3. For each rectangle, perform the following operations:
   a) Gather the road boundary points that lie inside.
   b) Estimate the best-fitted line L of the boundary points.

c) Calculate the intersection point p_{int} between the line defined by $\overrightarrow{d}$ and L.

d) Calculate the distance d from p_{mid} to p_{int}.

4. The road width is the sum of the two d distances, one from each rectangle.

Ideally, the rectangle width should be equal to the centreline step $w = c_{step}$ so that only boundary points in between the two centreline points are considered. However, if c_{step} is small, in other words, if the discretisation of the centerline is fine-grained, a larger width could be needed, as boundary points might not be found in such narrow rectangles. For the rectangle height h, its value must be large enough to reach the road boundary, so it must be set slightly higher than the expected maximum road width (MRW), we recommend $h = 1.15 \times \text{MRW}$.

Experimental Results

Tests were done in the Alcoy site using a rectangle width and height of 1 m and 14 m respectively. The result of the width estimation is shown in Figure 6.12. For each centreline point,

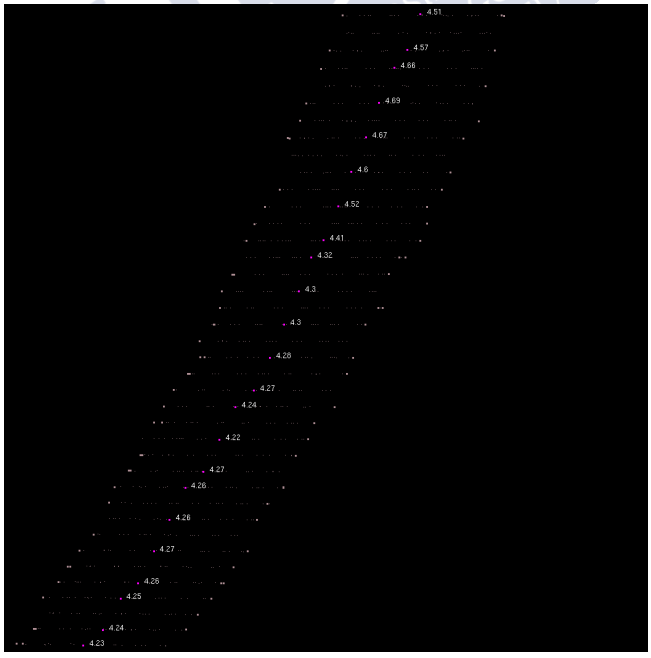


Figure 6.12: Width estimation for a road segment.

its width is obtained. The width values vary slightly. For this particular road segment, the width values are in the [4.22, 4.69] range with a mean of 4.37 m. However this variation is rather small, yielding a standard deviation of 15 cm, that is, 3.46% of the mean.

6.2 Discussion

In this section, we discuss some of the issues mentioned in the evaluation.

6.2.1 Road Tracking

We identify some points that could improve the centreline extraction. First, the issue with large asphalt areas could be addressed by stopping the tracking procedure in cases where the majority of the ATS rectangles have a large ATS value. Also, these large areas could be removed beforehand by adapting ATS-derived shape descriptors [273] into LiDAR data. A polygon can be built from the rectangle ATS values (for a 360º set), which will have circle shape if all rectangles have large ATS values, allowing to identify and filter these large areas. Also, the tracking could use some mechanism to predict the road direction based on previous iterations, e.g. using a parabola equation [146, 147], rather than relying entirely on the current iteration. Furthermore, the road width estimated in the next stage of the framework could be used to better adjust the centreline towards the middle of the road.

6.2.2 Snakes

While the rest of the framework is automatic, the use of the snakes is far for being automatic. First, the parameters of the snakes (see Table 6.2), that have been chosen experimentally, show good results in the Alcoy site, although more research is necessary to determine optimal values for any dataset in an automatic way. Second, the position of the initial contours points requires manual intervention. This initial position is critical for the correct result. A simple solution would be to initialise the contour points automatically at the boundary points of the nearby road, although it is not clear how large the initial contours should be, e.g. a snake originating far from the desired object contour may erroneously converge to some local minimum. Also, there are some factors in the algorithm whose impact needs to be studied, such as the number of initial contour points and the maximum number of iterations. Figures 6.13 and 6.14 show how these two factors, respectively, influence the final results. The same synthetic data presented in Section 6.1.2 was used. The contours are depicted in different colours: Initial

contours (in green), contours at different iterations (in blue), and final contours (in red). A reasonable option for the number of contours points would be to use as the needed to place them following the density of point cloud, however, it is unclear if a greater or lesser number would be beneficial. The optimal use of the maximum number of iterations also needs to be investigated. Furthermore, it is worth exploring other existing snakes variants, such as Gradient Vector Flow (GVF) [260], the Balloon model [48], and Geodesic Active Contours (GAC) [37].

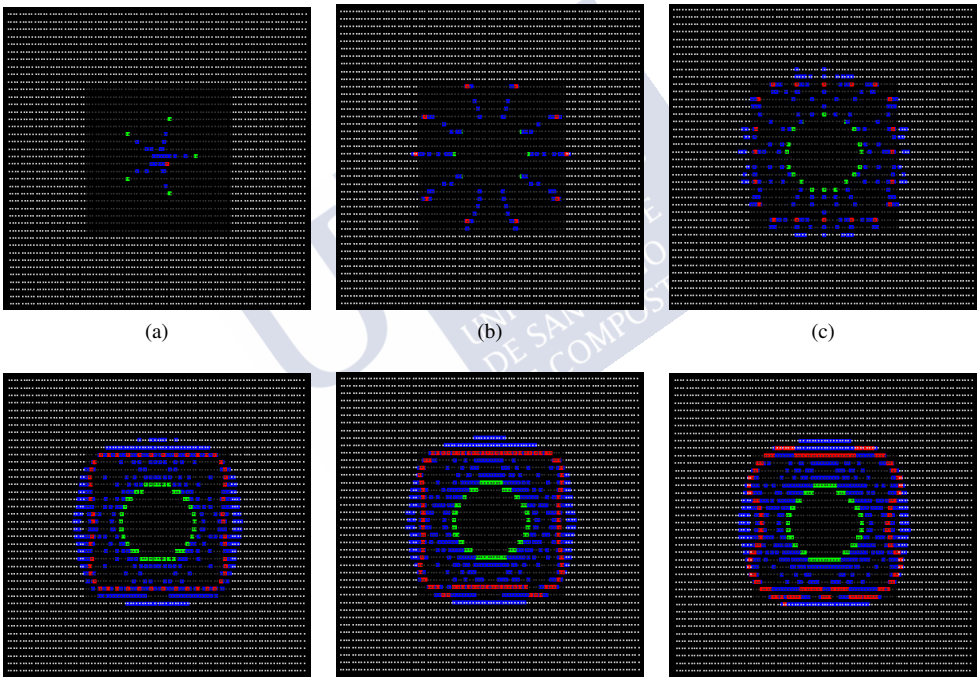


Figure 6.13: Snake results varying the number of contours to: (a) 5 (b) 10 (c) 20 (d) 50 (e) 100 (f) 150.

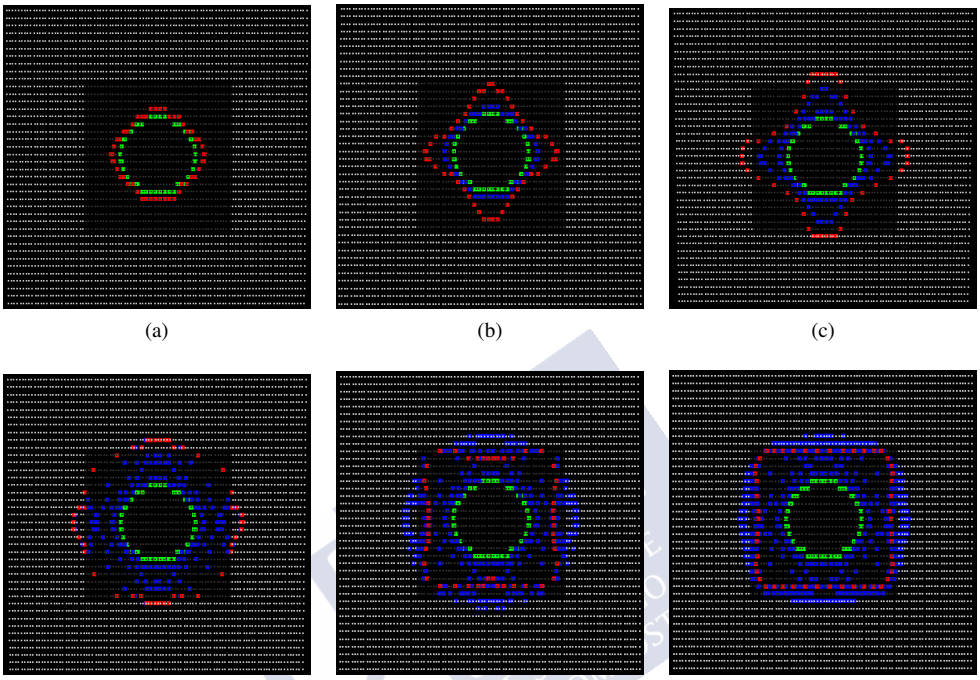


Figure 6.14: Snake results varying the maximum number of iterations to: (a) 1 (b) 2 (c) 3 (d) 5 (e) 10 (f) 50.

6.3 Conclusions

In this chapter, a framework for modelling the road network is presented. Two main techniques from image processing have been adapted into LiDAR data. Experiments show their potential to identify the road centreline, the road width, and fill the road gaps. Some limitations remain: First, large asphalt areas such as parking lots produce undefined behaviours in the road tracking, and second, the gap-filling using snakes require high manual intervention. Some ideas have been proposed to deal with the first limitation, however, most of our concerns relate to the second one. With all that, we think the obtained results are promising. Future research should focus on the application of the snakes and on finding, in an automatic way, optimal values for the framework parameters to achieve good results for any dataset.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Chapter 7

Fast Filtering of Ground Points

Until now, we have presented methods to use in an offline scenario. In this chapter, we explore the feasibility of real-time processing. For this endeavour, we consider one of the most basic stages in the processing pipeline of the LiDAR data: The ground filtering.

In the past decades, the applications of LiDAR have been extensively studied. While applications vary, nearly all of them require the separation of ground from non-ground points (ground filtering), as commented before, which is used in the process of creating DTMs. Because of this, ground filtering remains one of the most discussed topics in the literature. However, research often focuses on algorithmic performance disregarding the computational efficiency. LiDAR data acquisition technologies are on ongoing development, and the data volume is increasing consequently. Therefore, is also desirable for ground filtering methods to be computationally efficient. Especially, taking into account that a significant amount of the acquired data corresponds to the ground surface, so that ground filtering represents a considerable part of the computation in the processing pipeline of any application.

Several ground filtering methods have been proposed based on different techniques, such as segmentation and cluster filters, morphological filters, directional scanning filters and contour-based filters, triangulated irregular network (TIN)-based filters, and interpolation-based filters [168]. Although one particular method can hardly satisfy the requirements of every type of terrain, overall, satisfactory results have been achieved. State-of-the art methods achieve, approximately, an average kappa coefficient of 85% and a total error lower than 5% [17, 169, 40, 195, 173, 108, 274, 116, 41, 155]. However, using current methods in real-time scenarios may not be suitable. First, in some cases, the throughput will not be enough, as

computational efficiency is generally not prioritised. Second, they process the point-cloud scene as a whole, while in a real-time scenario, data is acquired in a scan-line fashion. In airborne LiDAR, the most common scanning devices use an oscillating mirror or a rotating polygon, so that points are acquired in scan lines perpendicular to the flight path. These scan lines can be used as input to the algorithms to address real-time processing. Few studies have used scan lines in their methods. In [215] 1-D bidirectional labelling along the scan-line profile followed by linear regression was presented. The detection of the ground level and the separation between clutter and man-made objects was addressed in [98]. In [109], a scan-line segmentation algorithm accelerated in GPU (Graphics Processing Unit) was proposed. Still, scan-line-based methods have limitations, particularly when dealing with complex scenes, because only the information in one direction is used [114]. Two-dimensional information should be considered to improve the results, but renouncing to scan-line-based processing would significantly reduce computational efficiency.

Scan-line-based methods can also be valuable for offline processing given its high computational performance. Because of this, scan-line-based methods are also being studied for other applications such as building detection [113] or plane extraction, where scan lines are used directly from the raw data [268] or are even artificially created [256].

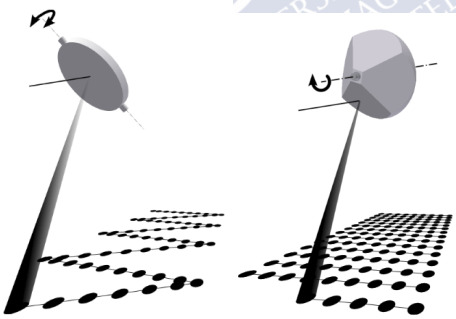
In this chapter, we propose a scan-line-based ground filtering method suitable for both real-time scenarios and fast offline processing. The method takes the scan lines as input in the same way they are acquired. A scan line can be seen as a 1-D object, where a sequence of height values is given; therefore, the processing is performed in a 1-D space, making it highly efficient. This processing consists of an iterative spline interpolation, in which, after an initial interpolation, the spline is iteratively refined. Therefore, points above a certain residual to the spline are labelled as non-ground points. The use of spline grants bending properties that are convenient when dealing with rough terrain. Nevertheless, it is challenging to identify ground accurately by processing scan lines individually, in other words, by using only the 1-D space. An alternative is to process adjacent scan lines (i.e., 2-D space). However, this requires the use of more elaborate data structures and neighbours queries, increasing the computational demand, which is not desirable for real-time applications. Because of this, a strategy of propagating the spline knots of a scan line to the next scan line is introduced, thus taking advantage of some 2-D information while preserving 1-D scan-line-based processing. This improves the reliability in complex scenes while maintaining its high throughput.

The chapter is organised as follows. The ground filtering method is detailed in Section 7.1.

Both quantitative and qualitative results, along with throughput measurements, are presented in Section 7.2. The implementation in an embedded system with field-programmable gate array (FPGA) modules is described in Section 7.3. Discussion about survey requirements, comparison with other methods, and real-time processing are presented in Section 7.4. Final conclusions are exposed in Section 7.5.

7.1 Method

The input of this method is a set of scan lines, which can be a complete flight line or pieces of it. A scan line can be defined as the set of points recorded during the scanning mechanism rotation along the field of view. Scan lines are generally acquired in a zigzag or a parallel pattern, depending on the scanning mechanism (see Figure 7.1). Consecutive scan lines during the same flight course form a flight line, which is stored in a LAS file [74] (see Figure 7.2). The stored LiDAR points follow the format established in the LAS specification, which defines the number, order, and data type of the data fields as detailed in Chapter 1.



(a) (b)

Figure 7.1: Scanning mechanisms and their resulting scan-line ground patterns [9]: Scan lines are captured perpendicularly to the flight course. The (a) oscillating mirror and (b) rotating polygon mechanisms are commonly used in airborne scenarios.

The LiDAR data must meet two requirements. First, points must be georeferenced (see Chapter 1). While this is generally granted for offline data, in the case of real-time processing, a scanner with this capability is needed. Second, individual scan lines must be identifiable. In the case of LAS data, this information is provided in the scan direction field and edge of flight line field, as it was commented before. Just one of these two fields is enough to identify scan

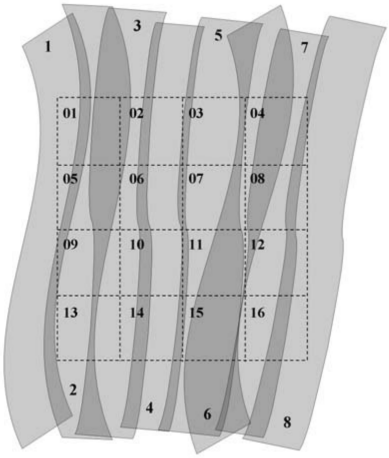


Figure 7.2: Space extend of the LAS data: A flight line (single-digit numbered) is formed by a set of consecutive scan lines. Odd- and even-numbered stripes are flown in opposite directions. The darker shading shows stripe overlap. Generally, LAS data is organised in tiles (two-digit-numbered rectangles) [79].

lines. For simplicity in the processing, the points within each scan line must follow the same scanning direction. This is the case when data have been acquired with a rotating polygon mechanism. If an oscillating mirror mechanism is used instead, it is needed to invert the point order the odd scan lines, as they are recorded in both scan directions.

A flowchart of the whole method is depicted in Figure 7.3. Note that only last-return points are taken into account, as possible previous returns are assumed to be non-ground points. The first step is to calculate the set of neighbours of each point. Then, an iterative spline interpolation is performed in each scan line sequentially. First, this is done from the first to the last scan line (forward), and then, it is done from the last to the first scan line (backwards). This iterative spline interpolation consists of three main stages: (1) the selection of the initial spline knots (seeds), (2) the iterative spline interpolation, and (3) the propagation of the final spline knots to the next scan line. Once the backward iterative spline interpolation of a scan line has been performed, the resultant spline is used to identify the ground points based on their residuals. Following, we describe each one of these steps.

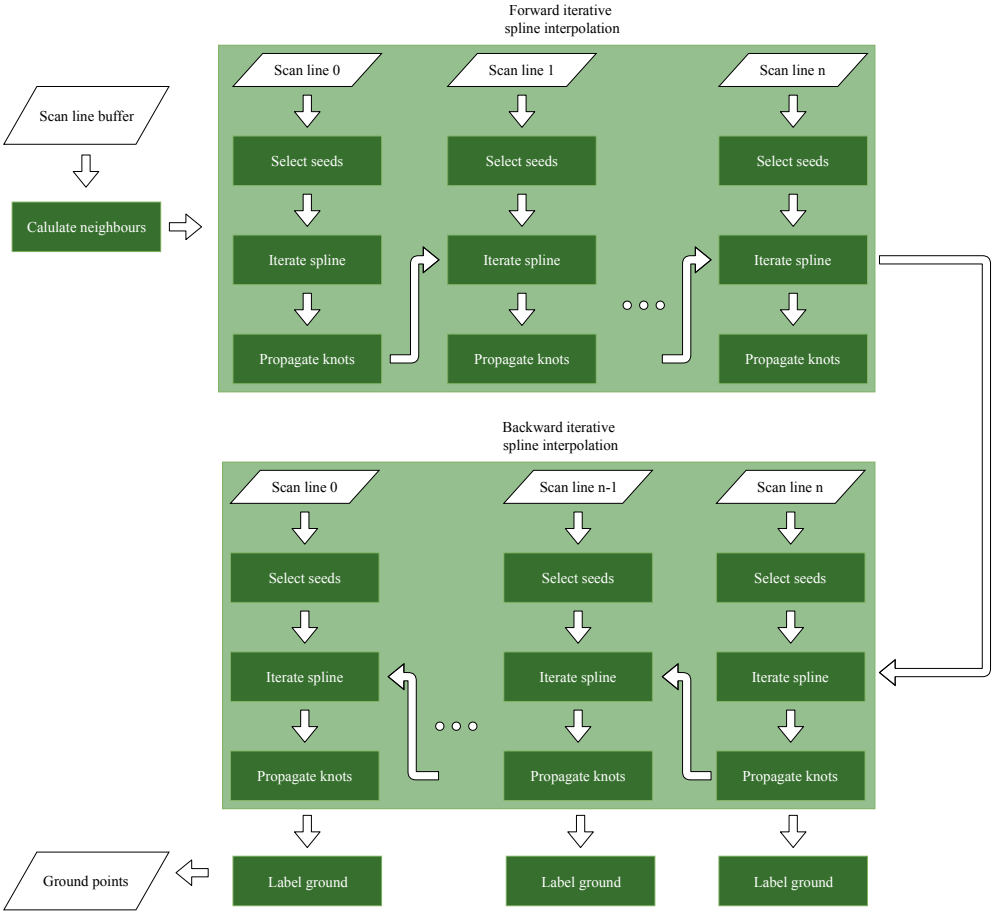


Figure 7.3: Flowchart of the iterative spline ground filtering.

7.1.1 Neighbour Calculation

For each point, the closest point in the previous scan line and the closest point in the next scan line are calculated in this initial step. We further refer to these neighbours as the perpendicular neighbours. A perpendicular neighbour of a point can be calculated by brute force, computing all the distances from the point to all the points in the adjacent scan line and keeping the closest. Nevertheless, this is inefficient and severely increases the computing time. Moreover, it is reasonable to expect that a perpendicular neighbour will have a similar index within the

scan line. This assumption can be exploited to speed up the neighbour search. The idea is to start the search at the adjacent point with the same index and to continue the search backwards and forward while the distance is being reduced.

The process is depicted in Figure 7.4. For a point p_{63}, which is the point with index 63 in a scan line S, the search starts at the adjacent point p'_{63}, which is the point with the same index in the adjacent scan line S'. First, the distance between these two points, the initial distance of D_{min}, is calculated, and p'_{63} is labelled as a right neighbour. A backward search is then initiated. The distance from p_{63} to p'_{64} is larger than D_{min}, so the backward search stops. Then, the forward search starts. The distance from p_{63} to p'_{62} is smaller than D_{min}, so p'_{62} is labelled as the new right neighbour, D_{min} is set to this distance, and the search continues. The distance from p_{63} to p'_{61} is also smaller than D_{min}, so p'_{61} is labelled as the new right neighbour, D_{min} is set to this distance, and the search continues. The distance from p_{63} to p'_{60}, however, is larger than D_{min}, so that the forward search stops and p'_{61} is returned as the right neighbour.

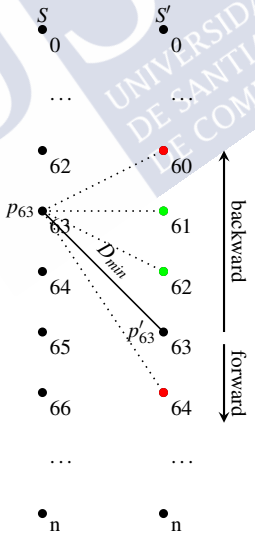


Figure 7.4: Visualisation of the right neighbour calculation for the point p_{63}. Green dots mean that the distance is lower than the previous one while red dots mean that this distance is larger.

This calculation is faster the more similar the point spacing is in both scan lines, as it leads a point and its neighbour to yield a similar index. Therefore, its speed is influenced by the

acquisition procedure to some extent (see Section 7.3.1).

7.1.2 Seeds Selection

Before the interpolation, a set of initial knots, known as seeds, are selected. The number of seeds depends on the interpolation algorithm. We use the Akima algorithm [5], which requires a minimum number of 5 seeds. For its selection, the scan line is split into 5 segments of equal length and the lowest point in each segment is selected as a seed point. Seed points are assumed to be ground points. This assumption is acceptable when the length of the scan line is reasonably large. More specifically, this is guaranteed when the length of each segment is larger than the largest non-ground object of the scene. Therefore, the length of the scan line must be considered when planning the survey. The optimal scan line length is calculated as:

$$SW = \alpha n L, \ \alpha \in [0.5, 1] \tag{7.1}$$

where α is a user-defined constant (ideally 1), n is the number of scan line segments (5 in our case), and L is the largest non-ground object size. More details are given in Section 7.4.1.

7.1.3 Iterative Spline

Based on the seeds, an initial interpolation of is performed. The Akima algorithm is a continuously differentiable sub-spline interpolation built from piecewise third-order polynomials. An important feature of this method is its locality: Function values on the interval $[x_i, x_{i+1}]$ depend only on its neighbourhood $f_{i-2}, f_{i-1}, f_i, f_{i+1}, f_{i+2}, f_{i+3}$. The advantage of this is twofold: First, it does not lead to unnatural wiggles in the resulting curve (see Figure 7.5), and second, it is computationally efficient as there is no need to solve large equation systems.

The 3-D points are projected into 1-D as follows: Its coordinates $\{x, y, z\}$ are converted into $\{x', y'\}$, where x' is the 2-D Euclidean distance to the first point of the scan line and $y' = z$. Note that x' values must be monotonically increasing, so duplicated values are removed.

The iterative interpolation consists of the execution of three stages: (1) interpolation, (2) pushdown, and (3) push-up. The spline passes through a set of knots $K = \{(x'_i, y'_i) : 0, 1 \dots n\}$. The idea is to iteratively refine the spline by adding new knots into K. The steps are as follows:

1. The interpolation of the spline is performed. In the first iteration, K consists of the seed points, and the initial spline is a rough estimation of the ground level (see Figure 7.6a).

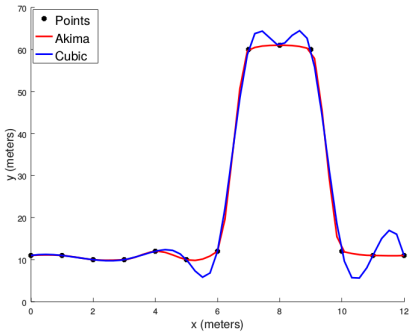


Figure 7.5: Comparison between Akima and cubic interpolations.

2. In the pushdown stage, the spline descends into the ground level by adding ground points under the spline into K (see Figure 7.6b). For each knot pair (k_i, k_{i+1}), the point with the largest residual under the spline is added into K if this residual is higher than a tolerance threshold T so that only significant residuals are considered. An adequate value for this threshold is between one and two times the vertical accuracy of the LiDAR data. In this work, $T = 0.15$ m is used. After processing all knot pairs, if K has new points, we return to Step 1; otherwise, this stage ends.

3. In the push-up stage, the spline raises by adding ground points above the spline into K (see Figure 7.6c). The idea is to start at a known ground point and to consider the next point as ground as long as the transition is smooth. This is accomplished by a 1-D bidirectional labelling algorithm. For each knot, both forward and backward labelling are conducted. The labelling process is shown in Figure 7.7. The process starts at the next point of the knot. This point has to fulfil three constraints based on height difference, slope, and step distance. Their calculation is given by Equations 7.2–7.4.

$$Z_i = z_i - z_{i-1} \tag{7.2}$$

$$S_i = \arctan\left(\frac{z_i - z_{i-1}}{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}\right), S_i \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \tag{7.3}$$

$$D_i = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}, \text{with } k \text{ as the index of the previous knot} \tag{7.4}$$

First, the pairwise height difference, Z_i, must be lower than a given threshold of Z_t. Second, the pairwise slope S_i must be lower than a given threshold S_t or the slope change (if available) must be lower than the half of S_t. Third, the 2-D distance to the previous knot, D_i, must be higher than a given threshold of D_t. This last constraint limits the number of points that will be labelled as knots, so that only meaningful knots are added, as processing all knots will increase the computational demand of the interpolation with little contribution to its accuracy. The height and slope thresholds can be tuned for particular data to obtain better results. Common values can be widely found in the literature. Our suggestion is to use as default parameters $Z_t = 0.5$ m and $S_t = 45^\circ$ or $S_t = 60^\circ$ for urban or rural sites, respectively. For the step distance threshold, we have found experimentally that $D_t = 1$ m provides the best trade-off between accuracy and throughput. If the point does not fulfil the height or slope constraints, the next point yielding a low residual is added into K and the labelling continues in that point. The process ends if a knot is reached. Finally, if K has new points, we return to Step 1; otherwise, the iterative spline interpolation finishes.

7.1.4 Knot Propagation

The knot propagation is performed after the iterative spline interpolation. Its goal is to avoid errors when ground filtering is executed on each scan line individually without knowledge of the previous filtering. For example, in high-slope terrains, few ground points might not be identifiable for a given scan line, resulting in an inaccurate spline and causing several points to be mislabelled. However, this can be mitigated if the spline of the previously filtered scan line is considered.

The process is as follows: Let K be the set of knots of a scan line S after the iterative spline interpolation and K' be the set of knots of the next scan line S' to be processed, which is initially empty. The goal is to propagate the knots of K into K'. Propagating a knot means that its perpendicular neighbour will be added into K'. To be propagated, a knot has to fulfil the same but more restrictive height, slope, and step distance constraints than in the push-up stage described previously. In particular, the height and slope thresholds used are reduced to half. Also, if a knot fulfils the height and slope constraints but the step distance is lower than D_t, the knot is ignored. However, if a knot does not fulfil the constraints but the step distance is higher than D_t, the last ignored knot is propagated. This way, the gap between propagated knots is reduced in the case of non-fulfilling knots.

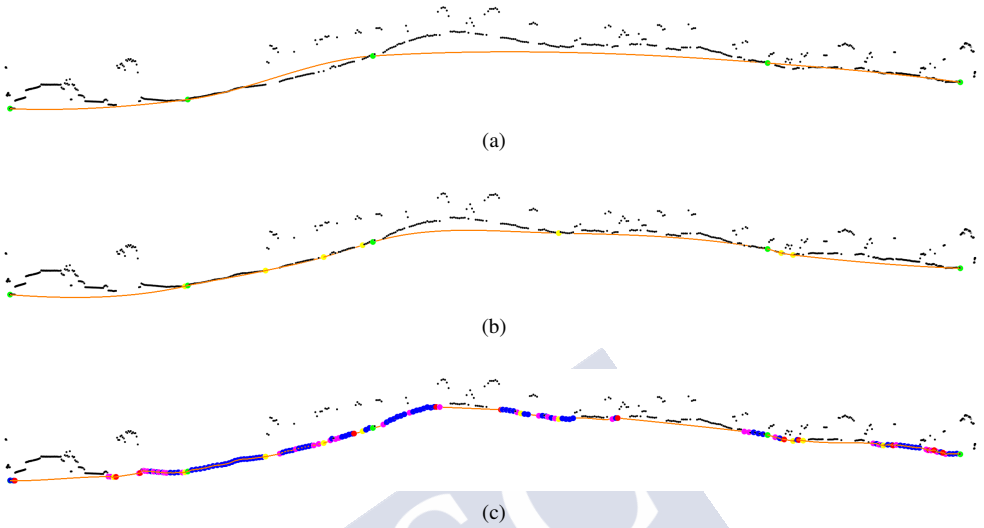


Figure 7.6: Spline (orange) of a scan line after (a) the initial interpolation, (b) the pushdown stage, and (c) the push-up stage. The knots of each stage are also shown: seed knots (green), pushdown added knots (yellow), and push-up added knots (blue, red and purple). The colours of the push-up added knots match the different knot addition processes shown in Figure 7.7: p_i (blue), p_{i-1} (red) and p_j (purple).

Therefore, at the first iteration of the iterative spline interpolation, the initial knots of a scan line are formed by the selected seed points plus the knots propagated from the previous scan line.

7.1.5 Ground Labelling

Finally, the ground points of each scan line are identified using the resulting spline after forward and backward interpolation. For each point, the residual to the spline is calculated, and if this residual is lower than the tolerance threshold T, the point is labelled as ground.

7.2 Results

In this section we perform a quantitative evaluation using synthetic ground truth and a qualitative evaluation in three additional sites. Also, we analyse the computational performance in a regular desktop computer.

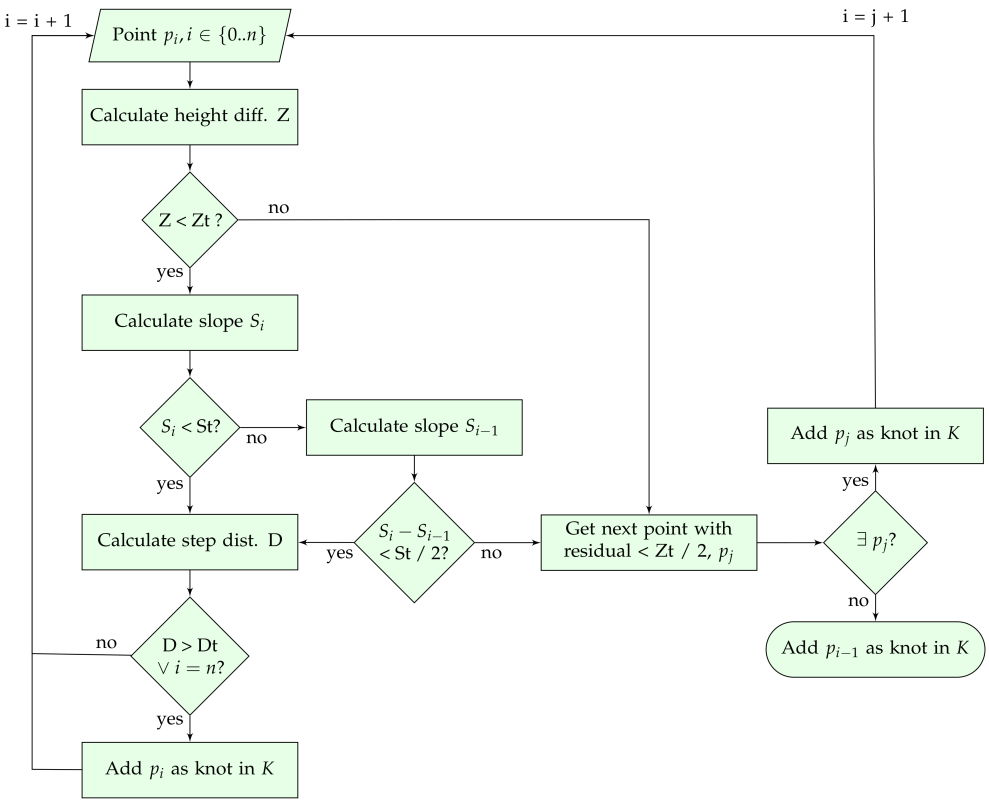


Figure 7.7: Flowchart of the push-up stage.

7.2.1 Quantitative Evaluation

The benchmark dataset provided by the ISPRS Working Group III/3 [219] is the standard dataset used to evaluate and compare ground filtering algorithms. This includes fifteen study sites with unique terrain conditions that are considered challenging to filter. Nevertheless, this dataset is not used in this study for two reasons. First, the provided LAZ files [120] do not provide information to identify the scan lines. The two related fields, scan direction and end of flight line, are set to zero. Second, even with such information available, the scan lines are particularly narrow and their length is not suitable for our method. In our approach, scan lines are initially split into 5 segments to determine the seeds for the initial spline interpolation (see Section 7.1.2). The scan line length in the ISPRS dataset is about 150 m so that each segment

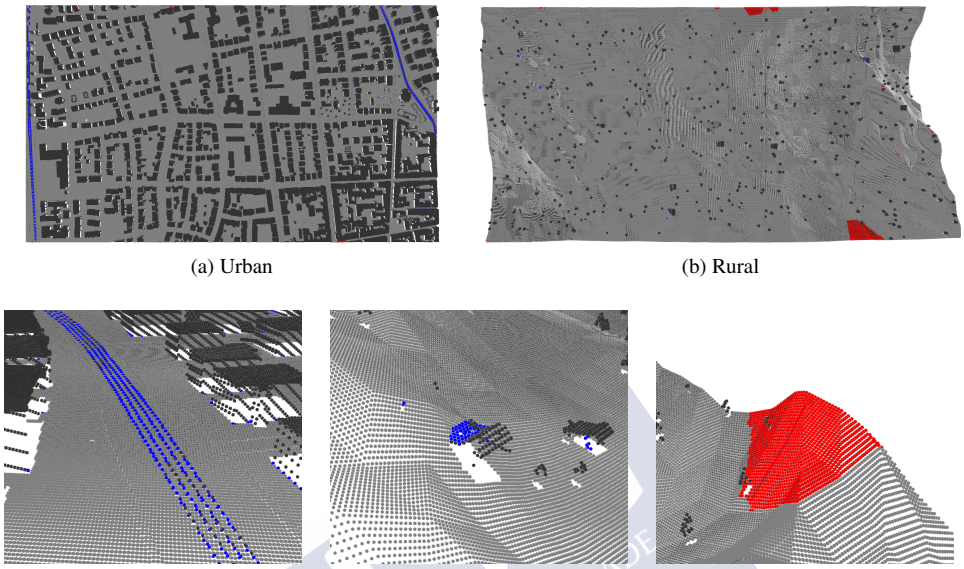
would have approximately 30 m of length, and larger objects exist in these scenes. This will cause some of the seeds to be placed on top of non-ground objects, resulting in critical errors. Therefore, we instead generate a synthetic ground truth to perform a quantitative evaluation.

Synthetic ground truth was generated using the Heidelberg LiDAR Operations Simulator (HELIOS) [23], already introduced in Chapter 3. Both urban and rural scenes were built. The urban scene was created based on OpenStreetMap (OSM) data [189]. The city of Heidelberg was chosen because of its rich OSM information. The 2-D OSM data was converted into a 3-D mesh using OSM2World [229]. The rural scene was generated by using a predefined terrain model and by randomly placing models of trees and buildings. The model's wavefront OBJ (Wavefront 3D Object File) and MTL (Material Template Library) files were modified so that the polygons corresponding to ground surfaces are identified by HELIOS. This way, when running the simulation, pulse returns from these polygons are automatically labelled as ground points, hence generating the ground truth. The urban survey consists of an airborne laser scanning one flight strip of 1 km at a flying height of 700 m and speed of 30 m/s. A Leica ALS50-II sensor was configured with a scan angle (FOV, Field of View) of 50.5° (see Section 7.4.1 for justification), a scanning frequency of 60 Hz, and a pulse rate frequency of 100 KHz. The rural survey consists of an UAV-borne laser scanning one strip of 600 m at a flying height of 300 m and speed of 20 m/s. A Riegl VUX-1UAV sensor was configured with a scan angle (FOV) of 45°, a scanning frequency of 100 Hz, and a pulse rate frequency of 100 KHz.

The default parameters proposed in Section 7.1.3 were used. Results are depicted in Figure 7.8 and the kappa coefficients and error rates are shown in Table 7.1. In the urban scene, the main error source is omission error (Type II error) present in the railway and in the base of some building facades (see Figure 7.8c). In the rural area, the main error source is commission error (Type I error) present in hilly terrains at the boundaries of the scene (see Figure 7.8e). Given its high slope, it is quite challenging to label hilly areas as ground. Also, commission error is present in some lower roofs that reach the ground (see Figure 7.8d). Overall, these results show the suitability of our method.

7.2.2 Qualitative Evaluation

Visual inspection was carried out using real LiDAR data, particularly, private data from Alcoy and Trabada, both in Spain, and public data from Vaihingen, in Germany. Information about these data is shown in Table 7.2. Alcoy is a rural site with areas with high slope, abrupt height



(a) Urban (b) Rural

(c) Urban (d) Rural (e) Rural

Figure 7.8: Error distribution in the (a) urban and (b) rural synthetic data and (c–e) sources of error. Legend: Type I error (red), Type II error (blue), ground points (light grey), and non-ground points (dark grey).

Scene	Type I (%)	Type II (%)	Total Error (%)	Kappa (%)
Urban	0.03	1.66	0.51	98.78
Rural	0.46	3.75	0.49	78.40
Mean	0.25	2.71	0.50	88.59

Table 7.1: Errors and kappa coefficients for the synthetic samples.

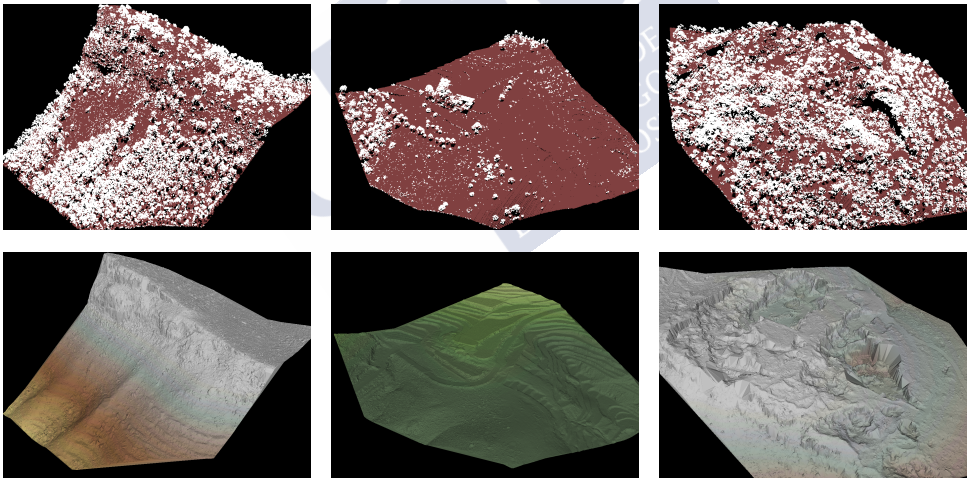
changes in the terrain, and some cliffs. Trabada is a rural area mixing well-defined lands with areas of dense vegetation. Vaihingen, as it is known, is an urban site with the features of a European city.

One flight line was processed for each site. For simplicity, only three representative areas are shown for each site. The default parameters proposed in Section 7.1.3 were used. Results are shown in Figures 7.9–7.11 and the highlighted errors are in Figure 7.12. Apart from the labelled ground points, their TIN is also shown to aid the visual inspection.

Site	Type	Date	Sensor	Density	Scan Line Length	Source
Alcoy	Rural	2011	Leica ALS60	8.65 p/m^2	282 m	Babcock Int. [19]
Trabada	Rural	2004	Optech 2033	4 p/m^2	410 m	LaboraTe [138]
Vaihingen	Urban	2008	Leica ALS50	4 p/m^2	471 m	ISPRS [50]

Table 7.2: Point clouds for the qualitative evaluation.

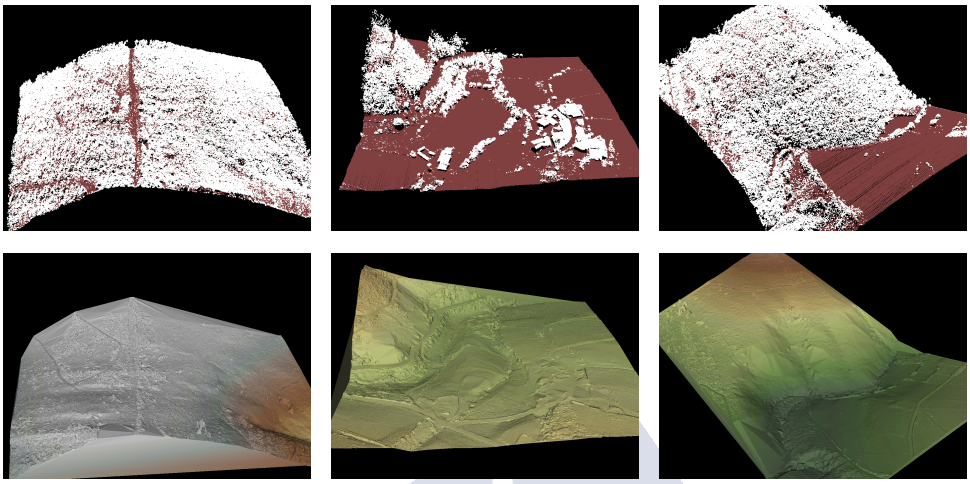
In Alcoy, there are no significant errors when processing slope terrains (see Figure 7.9a), flat areas with small but abrupt height changes (see Figure 7.9b), or irregular terrains (Figure 7.9c). Although, when dealing with irregular terrain, some ground points near the edges of the cliffs are sometimes missing, producing a less-detailed surface (see Figure 7.12a).



(a) (b) (c)

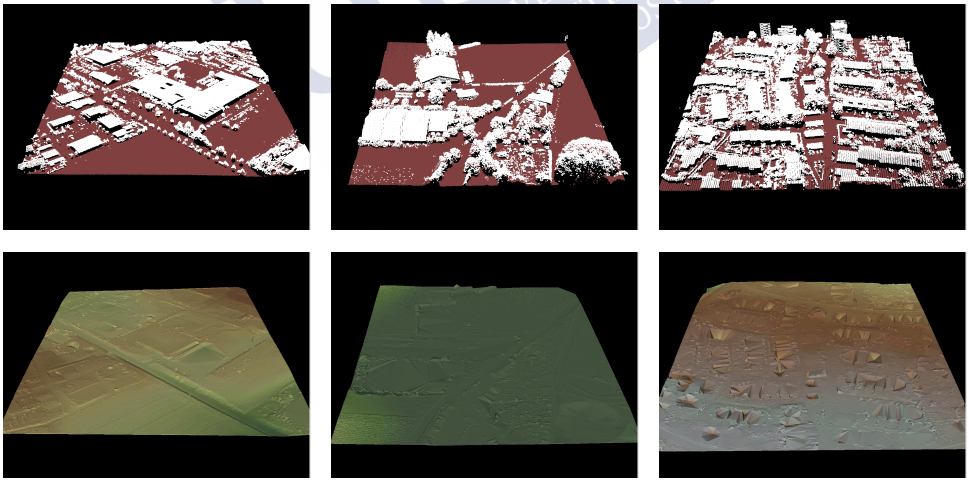
Figure 7.9: Alcoy results.

In Trabada, few points located in the lower parts of trees are mislabelled as ground, giving the ground surface a rough look. This effect is related to the slope threshold; by increasing this threshold from 45° to 60° in rural areas, the labelling of sloped terrain improves but also increases the overall roughness of the extracted surface.



(a) (b) (c)

Figure 7.10: Trabada results.



(a) (b) (c)

Figure 7.11: Vaihingen results.

In Vaihingen, there are no appreciable errors, in part, because most of the terrain is fairly

flat. Finally, we have observed that the filtering sometimes struggles in some areas near the edges of the scan lines, particularly when dealing with high sloped terrain or vegetation (see Figure 7.12c). In such areas, no ground points are labelled, leading to a less-detailed surface (see Figure 7.12b). This happens because the interpolation is not being performed at the boundaries of the scan line. More specifically, there is a set of points between the first point of the scan line and the first knot of the spline that heavily relies on the seed propagation to identify ground points, which is not always possible in these areas with high height and slope changes.

Overall, the results are satisfactory and the method can filter ground points successfully for both urban and rural sites using the default parameters.

7.2.3 Computational Performance

The execution times of the reference C implementation for the Alcoy, Trabada, and Vaihingen data samples are shown in Table 7.3. Measurements were taken on a workstation with an Intel Core i7-4790 processor, 16 GB of RAM and a Linux (Ubuntu 14.04.5 LTS) operating system. In the Alcoy and Trabada data, the algorithm can process nearly 1 million points per second. However, in the Vaihingen data, the throughput decreases significantly to 320,000 points per second. This is because of point accumulation present in the data, which slows down the neighbour search. This issue is further addressed in Section 7.3.1.

Site Points	**Alcoy** 6 million	**Trabada** 6 million	**Vaihingen** 3.77 million
Operation	**Time (s)**	**Time (s)**	**Time (s)**
Input reading	9.64	10.04	3.13
Neighbours	1.82	1.82	8.92
Ground filter	6.11	4.24	2.58
Output writing	17.37	15.79	6.54
points/s (No IO)	967,050.24	989,997.99	321,271.25
points/s (with IO)	195,093.48	188,136.22	174,510.62

Table 7.3: Execution times and throughput in the workstation with and without considering input and output (IO).

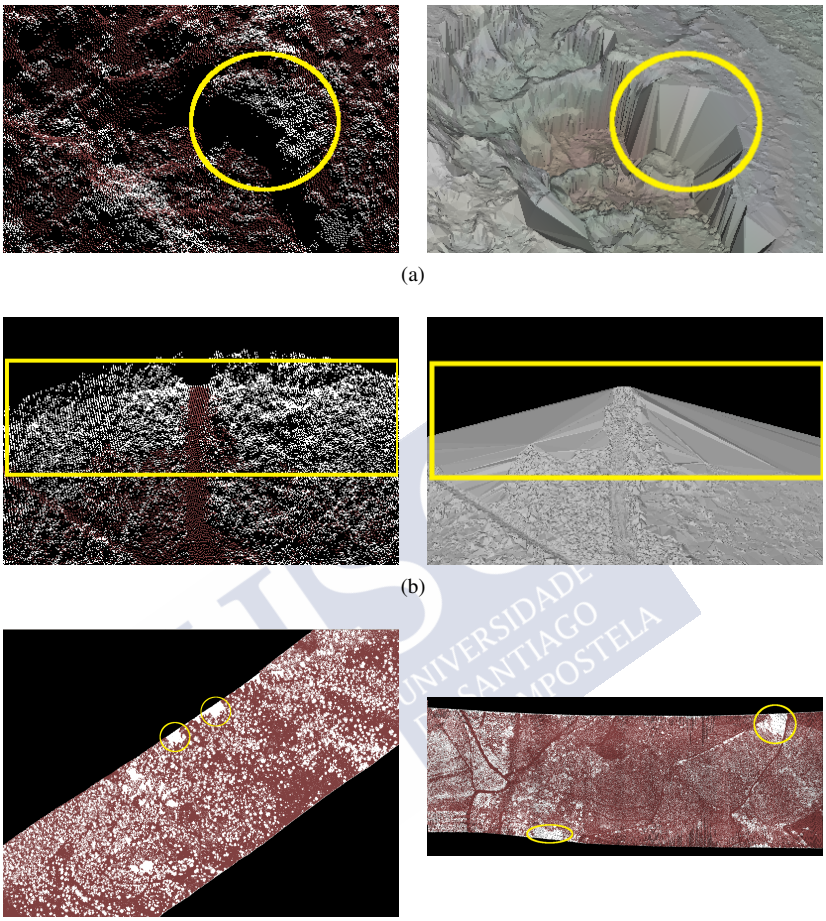


Figure 7.12: Visible errors highlighted in yellow.

7.3 Embedded System Implementation

To demonstrate the portability of the proposed method to embedded systems while maintaining its good performance, we implemented it in a Zedboard platform [51]. This is a low-cost, low-power consumption development board, and it is shipped with a Zynq-7000 System-on-Chip (SoC) from Xilinx, 512 MB of DDR3 RAM, and several I/O peripherals such as a 1-Gb Ethernet module and an SD card reader. The Zynq SoC integrates a 32-bit dual-core ARM

Cortex-A9 CPU and a low-range Z7020 FPGA [258], which allows the implementation of simple custom hardware accelerators.

In this proof of concept, we deployed two different implementations in the Zedboard. The first implementation is just a porting of the original C source code, developed for the x86-64 architecture, to the ARM v7 architecture. The results and performance comparisons for different LiDAR data are described in Section 7.3.1. The second implementation is a custom hardware accelerator on the FPGA for precomputing the perpendicular neighbours. The software-hardware system developed and the results of the evaluation test performed are described in Section 7.3.2. Power and energy consumption is shown in Section 7.3.3.

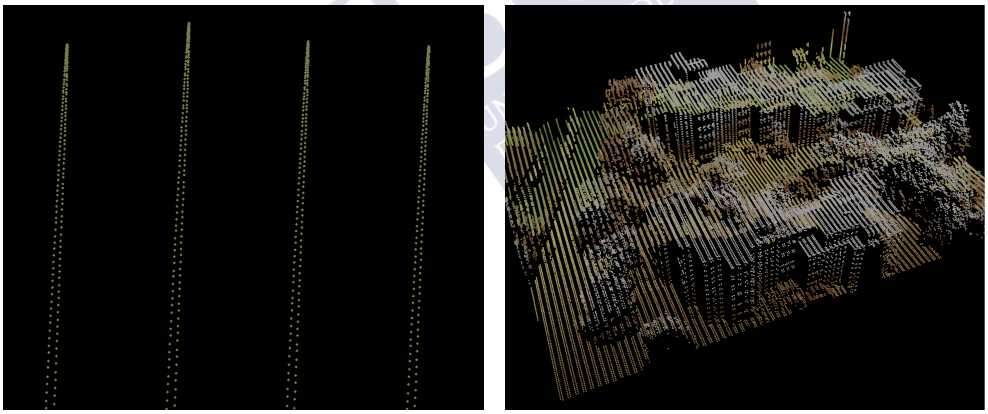
7.3.1 CPU Implementation

Since a bare-metal implementation (non-OS support) of the method in the Zynq SoC does not provide enough flexibility for future code development and enhancements, a 32-bit embedded Linux distribution (Xilinx Linux kernel 4.14.04 and an Ubuntu 16.04 root file system) was deployed in the ARM Cortex-A9. For porting the original C code to the ARM Cortex-A9, we tried to avoid any substantial code changes. The pre-compiled GNU GSL libraries [75] with hardware floating-point support for the ARM Cortex-A9 were used, and the OpenMP [188] directives presented in the original code were preserved. We show the trace results for the Alcoy, Trabada, and Vaihingen data samples in Table 7.4. The timing figures are shown for the tests run in Zedboard (only in the dual-core ARM CPU run at 667 MHz).

Site Points	**Alcoy** 6 million	**Trabada** 6 million	**Vaihingen** 3.77 million
Operation	**Time (s)**	**Time (s)**	**Time (s)**
Input reading	118.15	120.31	58.17
Neighbours	36.53	36.53	122.90
Ground filter	74.78	55.52	48.47
Output writing	159.96	142.12	88.37
points/s (No IO)	75,718.72	65,174.49	21,560.17
points/s (with IO)	18,068.89	16,923.67	11,621.92

Table 7.4: Execution times and throughput in the Zedboard for the Alcoy, Trabada, and Vaihingen data with and without considering input and output (IO).

Note that the throughput in the Vaihingen sample is significantly lower than in the other samples. This is because the calculation of neighbours is quite costly in this particular sample, as it was commented before. This is attributed to an accumulation of points at the sides of the point cloud. On the one hand, the deceleration of the sensor's oscillating mirror when changing the scanning direction increases the number of points recorded at the edges of the point cloud (see Figure 7.13a). On the other hand, the higher the scan angle, the higher the coverage of building facades, which also increases the number of points recorded (see Figure 7.13b). These accumulations of points are detrimental for the computational performance of the neighbour search, which relies on the assumption that points of two adjacent scan lines present a similar point distribution. Furthermore, it is expected for point clouds generated in real-time to yield a more irregular scan-line point distribution than fully post-processed point clouds. To increase the throughput of the algorithm in such situations, a hardware accelerator was implemented on the PL (Programmable Logic) of the Zynq SoC.



(a) (b)

Figure 7.13: Sources of point accumulation: (**a**) oscillating mirror deceleration and (**b**) building facades.

7.3.2 Programmable Logic Acceleration

The hardware architecture of the system implemented is shown in Figure 7.14. The hardware accelerator consists of two Neighbourhood Search Cores (NCOREs), each one computing

the neighbours of different scan lines in parallel. Each NCORE communicates with the on-board DDR3 RAM and the CPU via DMA (Direct Memory Access) and high-performance (HP) ports. For optimal memory bandwidth use, two 64-bit DMA Controllers (AXI-DMA in Figure 7.14) are needed for each NCORE implemented and run at the same clock speed as the NCORE. This bandwidth constraint is only required for reading data from RAM since write operations to RAM are less frequent. Both the internal frequency of the PL and the input frequency run at 100 MHz, which is the maximum that allows processing one datum per cycle with the current design. The ARM cores in the PS (Processing System) of the Zynq SoC run at 667 MHz.

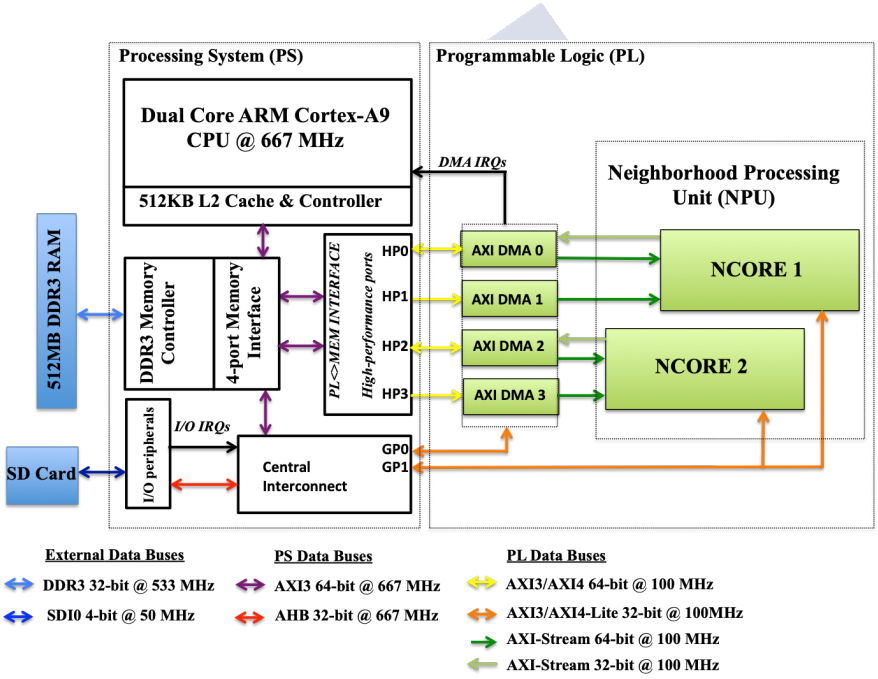


Figure 7.14: Hardware implementation in the Zynq-7000 platform.

To interact with the hardware acceleration from the software application, which runs in the Linux environment in the ARM CPU, we rely on the DMA Engine framework and Xilinx DMA device drivers. A general view of the architecture of the software-hardware system deployed is shown in Figure 7.15. In addition to the acceleration hardware, we developed a Linux kernel character driver and a DMA proxy user application. These software codes

provide the control logic for the communication of the DMA controllers built in the FPGA with the CPU via RAM.

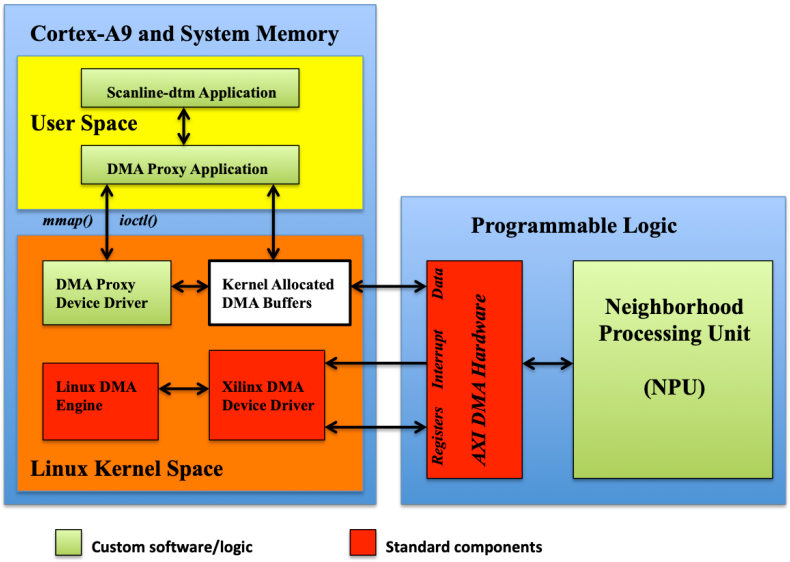


Figure 7.15: System architecture.

Finally, the architecture of the NCORE is shown in Figure 7.16. In addition to the parallelism at the scan-line level (one scan line search per NCORE), each NCORE can search concurrently for the neighbours of several consecutive points of the same scan line, issuing one search for each SPU (Stream Processing Unit) implemented in the NCORE. The structure of the SPUs is shown is Figure 7.17. Therefore, the total parallelism implemented in the hardware accelerator is given by the number of NCOREs multiplied by the number of SPUs of each NCORE. The optimal number of NCOREs is given by the ratio of the available DDR memory bandwidth concerning the NCORE throughput [1]. For our Zynq FPGA, this factor is close to two. On the other hand, we have implemented one core with 2 SPUs (NCORE 1) and the other (NCORE 2) with 3 SPUs. Therefore, the total number of SPUs (and the parallelisation factor) is 5.

The resources of the FPGA used and the total use percentage are summarised in Table 7.5.

[1]The NCORE throughput is proportional to the clock frequency and inverse to the maximum cycle interval between two consecutive data inputs

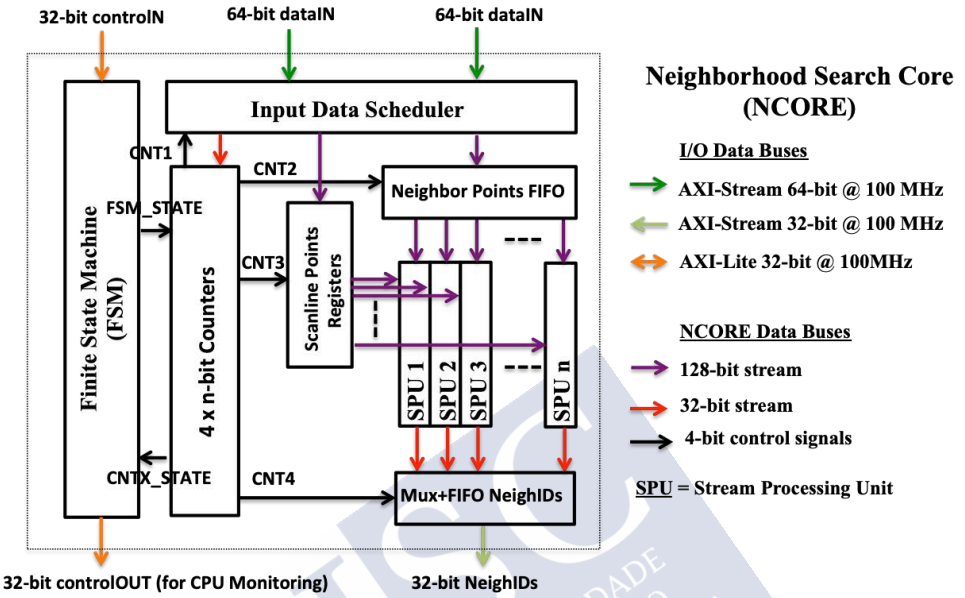


Figure 7.16: Neighborhood processing unit (Neighbourhood Search Core (NCORE)) architecture.

The FF (Flip-Flop) resources are used for implementing registers and fast data storage. The LUT (LookUp Table) is used for implementing combinational logic while the DSP48E are building blocks used in arithmetic (fixed and floating-point) operations. The performance is limited mainly by the available LUT resources, since embedding additional SPUs in a CORE does not modify the memory bandwidth usage.

Component	BRAM18K	DSP48E	FF	LUT
NCore 1	0	24	5259	7725
NCore 2	0	36	7513	11,088
HW Framework	20	0	17,484	11,532
Total HW	20	60	30,256	30,345
Available	280	220	106,400	53,200
Utilisation (%)	7	27	28	57

Table 7.5: List of field-programmable gate array (FPGA) resources used.

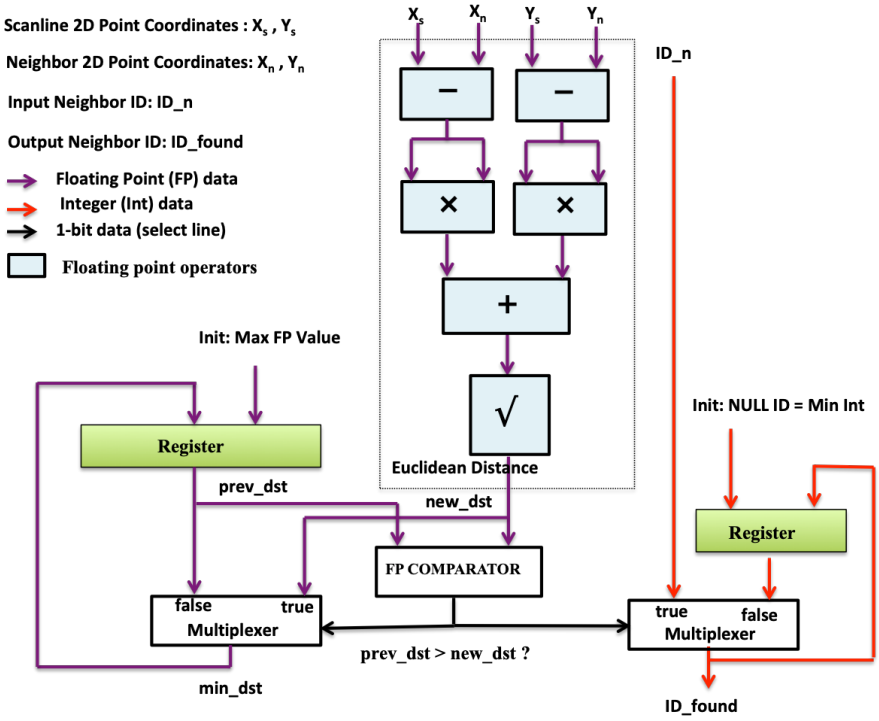


Figure 7.17: Stream Processing Unit (SPU) structure.

To provide an estimation of the achievable acceleration of the execution time of the neighbour search of the scan-line points, we run the Vaihingen sample in the Zedboard with and without using the PL hardware acceleration. The results of these tests are shown in Table 7.6. We show that the execution time of the search of neighbours can be accelerated a factor of 3.5 by implementing a (non-optimised) custom hardware accelerator.

7.3.3 Power and Energy Consumption

Low power consumption is an important design goal for UAV-borne electronic systems, as it removes the need for heat dissipation mechanisms, such as fans (less weight), and improves battery life. Two different methods have been used in order to estimate power consumption. For early estimation of the Xilinx Zynq SoC power consumption during the cycle design, we

Operation	Zedboard (PS)	Zedboard (PS + PL)
Neighbours time (s)	123	36
Total time (s)	318	233
points/s (No IO)	21,560.17	44,011.45
points/s (with IO)	11,621.92	16,220.70

Table 7.6: Hardware acceleration for the Vaihingen data.

have used the Xilinx Power Estimator (XPE). The XPE is a spreadsheet-based tool that provides detailed power and thermal information based on design information such as resource use and clock frequencies. For power consumption estimation of the Zedboard during execution, we use voltage measurements of the shunt-resistor mounted on the power supply of the Zedboard. The shunt-resistor is a 10-mΩ resistor in series with the input supply line to the board; this resistor can be used to measure the current draw of the whole board. The current draw can be used to calculate the power consumption of the board using Ohm's law and the supply voltage. As the power supply voltage is 12 V, the power consumption of the board can be obtained by measuring the voltage over the current sense resistor and by inserting the measured value into Equation 7.5.

$$P = \frac{V_{\text{measured}}}{10\ \text{m}\Omega} 12V \tag{7.5}$$

The power dissipation of the Zedboard during the tests performed with and without hardware acceleration was below 3 W. The results of these tests are summarised in Table 7.7. Energy efficiency is increased by using hardware acceleration.

System	Time (s)	Energy Cons. (J)	Energy Cons. Ratio
Zedboard (PS)	318	954	1.36
Zedboard (PS + PL)	233	699	1

Table 7.7: Power dissipation of the Zedboard for the Vaihingen data.

7.4 Discussion

In this section we analyse how using different scan-line lengths affects the quality of the result, we compare the obtained results with other existing methods, and we discuss about the

applicability of this method real-time scenarios.

7.4.1 Survey Requirements: Scan-Line Length

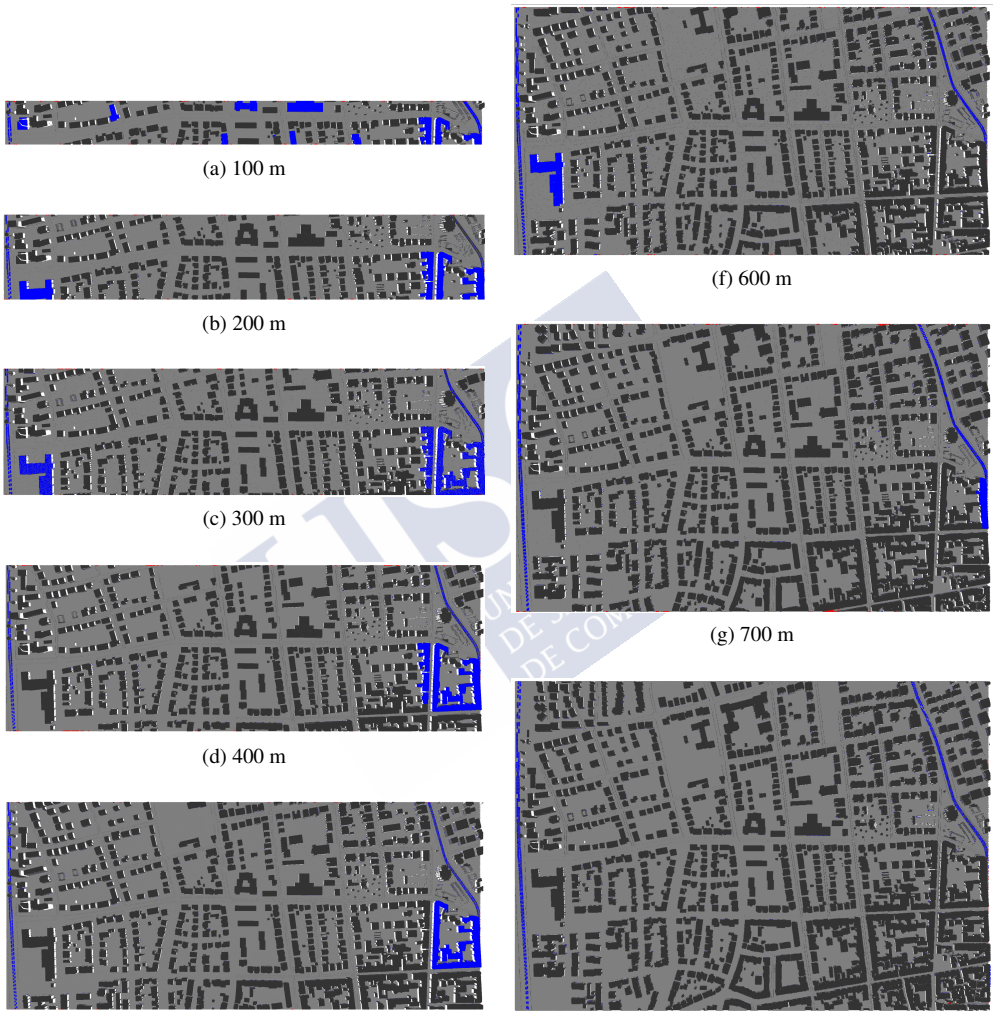
As mentioned previously, the scan lines need to have a minimum length, so that the seed points for the initial interpolation are correctly selected. To analyse the impact of the scan-line length in our method, we generated synthetic ground truth for Heidelberg urban site with different survey configurations to achieve different scan-line lengths. The scan-line length, formally known as swath width *SW*, is derived from the flying height h and the scan angle (FOV) θ, as in Equation 7.6 [20]. To consider different *SW* values, we opt to fix the flying height and to use different scan angles. The scan angle or flying height can be calculated using Equation 7.7.

$$SW = 2h\tan\left(\frac{\theta}{2}\right) \tag{7.6}$$

$$\theta = 2\arctan\left(\frac{SW}{2h}\right),\ h = \frac{SW}{2\tan(\frac{\theta}{2})} \tag{7.7}$$

The results are shown in Figure 7.18, and kappa values and total error rates are shown in Figure 7.19, respectively. The kappa values and total error rates improve the higher the scan-line length. With a scan line of 100 m, the accuracy metrics decrease significantly. Larger lengths achieve kappa values higher than 90%. The total error is under 5% when using a length as from 400 m. The main source of error is commission errors present in large buildings. Nevertheless, there are visible commission errors until the use of a length of 700 m. Also, note that some errors can appear again when using larger lengths. To understand this behaviour, we now explain the influence of the scan-line segment distribution.

Consider the case of the building placed in the left side of Figure 7.18, which has been wrongly classified using a scan-line length of 300 m, correctly classified using lengths of 400 m and 500 m, and wrongly classified again using 600 m. This is caused because, in the four cases, the building size is larger than the scan-line segment size, which can produce misclassifications depending on the distribution of the scan-line segments. The distribution is different for each scan-line length (see Figure 7.20). If a non-ground object is larger than a segment but smaller than two, there is a probability for the object to be split between two segments without fully covering any segment, thus selecting valid seed points in both segments



(a) 100 m

(b) 200 m

(c) 300 m

(d) 400 m

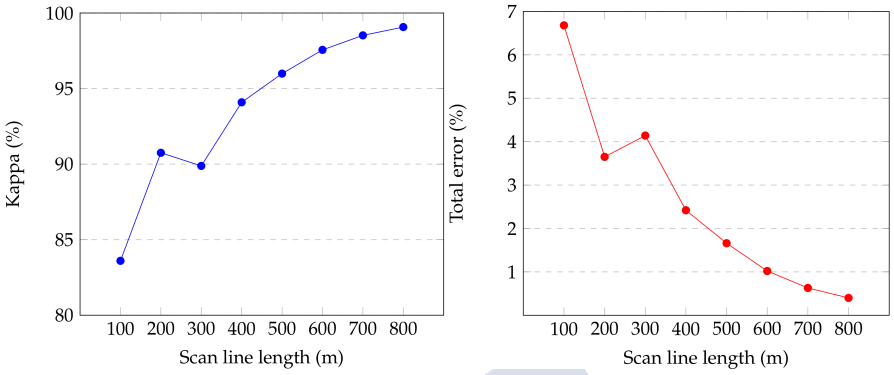
(e) 500 m

(f) 600 m

(g) 700 m

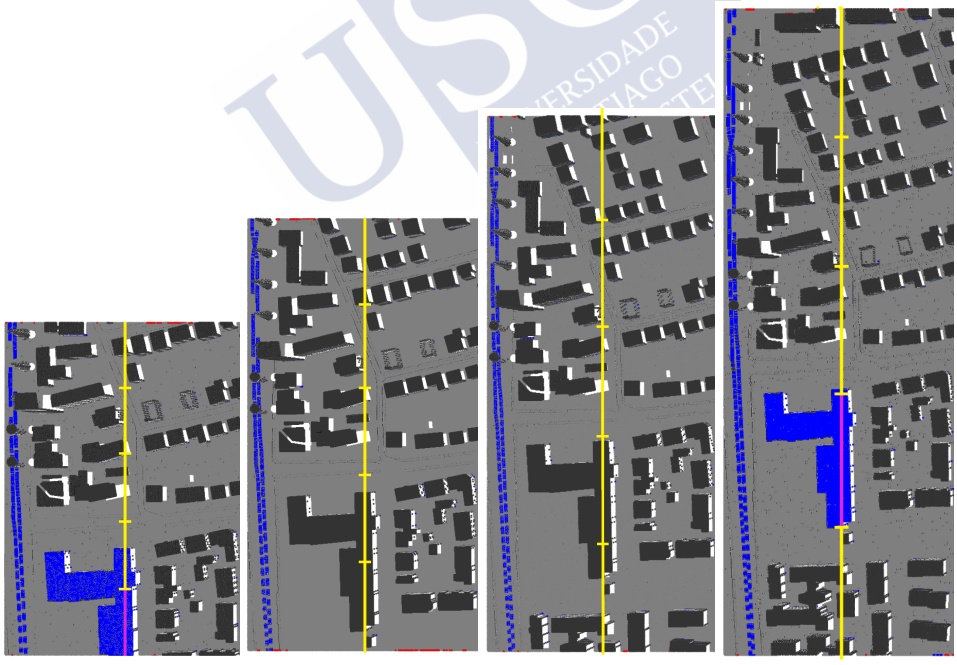
(h) 800 m

Figure 7.18: Error distribution in the synthetic Heidelberg's urban site for different scan-line lengths. Legend: Type II error (blue), ground points (light grey), and non-ground points (dark grey).



(a) Kappa (b) Total error

Figure 7.19: Kappa and total error in the Heidelberg's site for different scan-line lengths.



(a) 300 m (b) 400 m (c) 500 m (d) 600 m

Figure 7.20: Segments distribution (yellow) of a scan line for different scan-line lengths. Scan-line segments fully covered by the building are highlighted in purple.

(as in Figure 7.20b,c). However, if the non-ground object covers a full segment, then the selected seed point will be a non-ground point instead of a ground point, which will incorrectly shift the spline upwards, causing the misclassification of the object (see Figure 7.20a,d). The parameter α of Equation 7.1 is used to set this probability. Also, it is important to mention that the survey was done so that most non-ground are aligned with the scan-line direction, which is the worst-case situation.

Let us analyse an appropriate scan-line length for this site using Equation 7.1. The largest non-ground object is a building with an approximate largest size of 165 m. The ideal length is calculated by setting $\alpha = 1$, which results in 815 m. This would guarantee the correct selection of seeds independently from both the alignment of the objects with the scan line and the distribution of the scan-line segments. However, using such lengths may be impractical in some situations, particularly for UAV-borne laser scanning. In such cases, a smaller α can be used. Experimentally, we found that $\alpha = 0.80$ offers the best trade-off between accuracy and length. In this scene, this is a length of 660 m, which derives a scan angle of 50.5° with the used survey configuration.

7.4.2 Comparison with other Methods

Several authors have presented new methods for ground filtering during the last years. The total error and kappa coefficients they achieved are summarised in Table 7.8.

Note that our experiments have been conducted in a dataset different from ISPRS because, as previously explained in Section 7.2.1, the ISPRS benchmark is not suitable for scan-line-based processing. In fact, to the best of our knowledge, none of the scan-line-based methods has been tested with this benchmark. Our method yields the lowest total error rate, however, this can be partly attributed to the use of a synthetic dataset and lower accuracy values are expected when dealing with real data, as its complexity is higher. Nevertheless, tests with real data show that there is clear evidence that the proposed method yields similar accuracy values to existing methods. This is achieved without the need to train samples or to set complex thresholds and with high throughput. Regarding the embedded system implementation, we are not aware of any other method tested in such a platform to be compared with.

Author	Mean Total Error (%)	Mean Kappa (%)	Scan Line Based	Dataset
Axelsson (2000) [17]	4.82	84.19	×	ISPRS
Meng et al. (2009) [169]	NA	79.90	×	ISPRS
Pingel et al. (2013) [195]	2.97	90.02	×	ISPRS
Chen et al. (2013) [40]	4.11	86.27	×	ISPRS
Mongus et al. (2014) [173]	2.74	NA	×	ISPRS
Hu et al. (2014) [108]	2.85	90.29	×	ISPRS
Zhang et al. (2016) [274]	4.39	83.86	×	ISPRS
Hui et al. (2016) [116]	5.33	81.72	×	ISPRS
Chen et al. (2017) [41]	3.03	89.44	×	ISPRS
Ma and Li (2019) [155]	6.22	80.19	×	ISPRS
Shan and Sampath (2005) [215]	2.61	93.26	✓	Own
Hebel and Stilla (2008) [98]	NA	NA	✓	Own
Hu et al. (2013) [109]	8.24	NA	✓	Own
Proposed method	0.50	88.59	✓	Own

Table 7.8: Mean total error and kappa coefficients compared to other methods.

7.4.3 Real-time Processing Capability

For practical application, the proof of concept implemented on the Zedboard evaluation board should be prototyped on a high-end Zynq chip. We take as a reference the Zynq Ultrascale Zcu104 [259], which has a larger bandwidth (×4 of the Zedboard), a faster CPU (a 64-bit quad-core ARM Cortex-A53 at 1.5 GHz), and more PL resources that will allow increasing the number of NCORES to four with eight SPUs per NCORE, improving the hardware acceleration. Therefore, we expect an Ultrascale implementation to have a throughput four times faster than the Zedboard implementation. The Zedboard has an average throughput in the three samples of 61,600 points per second, so we expect to reach a throughput of nearly 250,000 points per second in the Ultrascale without any modification in the source code. The user must be aware of this throughput when selecting the sensor and its configuration, as real-time processing is accomplished if this throughput is higher than the data acquisition rate of the scanner. Therefore, real-time requirements are established by the LiDAR sensor. Some examples of current laser scanners that could be used to achieve real-time processing are the Ibeo Lux HD (19,000 points/s) [117] or the Riegl miniVUX-1UAV (100,000 points/s) [201]. This is assuming a setup tuned to achieve the maximum number of measures per second, al-

though a less demanding configuration could be used. In such a case, this implementation could cope with other scanners such as the Velodyne VLP-16 (300,000 points/s) [236]. Nevertheless, the algorithm requires the data to be georeferenced, which will certainly add some overhead in the processing. Moreover, although the power consumption estimation of the Ultrascale (9 W) is greater than the Zedboard (3 W), the energy consumption is 33% lower; therefore, the Ultrascale implementation is more energy efficient.

7.5 Conclusions

In this chapter, an efficient method for ground filtering based on scan-line processing is presented. Its main feature is the capability to be used in real-time scenarios. The core of the processing consists of a novel 1-D iterative spline interpolation algorithm. Two-dimensional information is considered while maintaining a fast 1-D processing approach thanks to a procedure to propagate the spline's knots. This strategy achieves a high throughput, which allows for fast ground filtering. Experiments showed that the method can perform ground filtering effectively in both urban and rural sites. Ground filtering is still challenging in areas with large non-ground objects and hilly regions. One relevant element to consider when using this method is the scan-line length, formally known as swath width, of the input point cloud. Guidelines for selecting an appropriate value when planning the survey have been provided, and its influence has been studied. Furthermore, an embedded system implementation we developed can process as many points as some current lightweight scanners can acquire with low-energy consumption. Overall, we conclude that this method accomplished real-time requirements for ground filtering.

CHAPTER 8

VISUALISATION OF POINT CLOUDS WITH OLIVIA

Point cloud processing methods often consist of multi-step frameworks with complex algorithms and the correctness of each step needs to be evaluated to ensure the overall robustness. To aid the understanding, evaluation, and presentation of these steps, visualisation is highly demanded. Several tools exist to visualise point clouds, but it can be difficult to choose the appropriate one for each particular purpose. When developing processing methods, displaying very specific drawings are frequently needed. Closed source projects do not offer flexibility because the source code cannot be adapted to develop specific drawings. Alternatively, in open source projects, attempting to create such custom visualisations can be difficult and time-consuming because of the broad scope of the projects or the lack of guidance.

The visualisation of point clouds can be easily done with several existing tools, but displaying application-specific content may not be straightforward. Many of the tools are not open source thus they cannot be modified to add additional functionalities, because of this, only open-source projects are considered in this summary. Three of the most popular tools were selected. An overview of them and the proposed tool, Olivia, is shown in Table 8.1.

ParaView [3] is an open-source, multi-platform data analysis and visualisation application. The project started in 2000 as a collaborative effort between Kitware Inc. and Los Alamos National Laboratory. Its main strength is the ability to analyse extremely large datasets using distributed memory computing resources. It allows loading, display, filter, query, and animate data. Its built-in features include subsampling, cropping, time-varying data, streaming and

Name	Release	Language	License
ParaView	2002	C++	BSD-3
CloudCompare	2009	C++	GPL-2
Point Cloud Library	2011	C++	BSD-3
Olivia	2018	Java	GPL-3

Table 8.1: Popular open source visualisation tools and Olivia.

parallel processing, etc. To grant more processing capabilities, the ParaView-PCL Plugin can be installed to provide access to the algorithms and Visualisation Toolkit (VTK) filters of the Point Cloud Library (PCL). Other features include Python integration, batching processing, and virtual reality visualisation. A JavaScript library is also provided to support interactive visualisation in a web browser. The use of this tool is described in detail in the 251-page ParaView Guide (Community Edition), as the number of functionalities in this tools is impressive. It is also possible to write custom applications, although it can be overwhelming to deal with such a big project.

CloudCompare [83] is a 3D point cloud and triangular mesh editing and processing software. The project started in 2003 as a collaboration between Telecom ParisTech and the R&D division of EDF and it has been released in the public domain around 2009. It was originally designed to perform a direct comparison between dense 3D point clouds, making it suitable for change detection and analysis. Its built-in features include many processing algorithms (registration, resampling, colour/normal vectors/scalar fields management, statistics computation, etc.) as well as display enhancement tools (colour ramps, colour and normal vectors, calibrated pictures, OpenGL shaders...). This tool is oriented to processing and lacks an easy way of creating custom visualisations.

Point Cloud Library (PCL) [207] is a large scale, open project for 2D/3D image and point cloud processing. The project started in 2010 at the robotics research lab Willow Garage being first officially released in May 2011. It contains numerous state-of-the-art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting, and segmentation. The library also offers a visualisation module providing methods for rendering and setting visual properties, drawing basic 3D shapes, histograms, range images, and geometry and colour handlers, which is handy for analysing the algorithms within the library. This tool is also oriented to processing, but future developments of this visualisation module should be taken into account. Also, the C++ language presents a higher barrier to entry than

Java.

Summarising, these tools have been developed for several years and they provide robust and broad functionalities, although we observe two disadvantages. First, most of them are oriented to point cloud processing instead to visualisation. Second, the learning curve to use, modify or improve these tools is high.

To deal with these issues, we introduce a new open-source stereoscopic 3D visualisation tool for LiDAR point clouds. This tool, named Olivia [163], can be easily extended and tailored to the needs of any particular point cloud processing application. It could also be adapted to other point cloud sensor sources if needed with some additional effort. The implementation, architecture and functionalities of the tool are described in this chapter. By performing a visual inspection of the results, valuable feedback can be gathered to refine the algorithms used in these use cases. We have developed Olivia focusing on the following features:

- Designed for visualisation (no processing) purposes.
- Ease to quickly develop customised visualisations.
- High portability across systems.

Olivia does not compete with the tools described above, in fact, initially, our tool offers fewer functionalities in comparison. The goal is to build an open-source tool which can be easily adopted by the user with minimum friction to create custom visualisations. All the basic functionalities expected from a visualisation tool such as point cloud rendering, geometry drawing, and point/cluster selection are already provided. Some additional functionalities are implemented as well, such as stereoscopic 3D or neighbours display. Users can build, in a short time frame, custom visualisations and take advantage of these embedded functionalities. Also, thanks to its high portability it can be used in almost any computer. We aim to help researchers and users alike to understand, evaluate, and show point cloud processing algorithms quickly and easyly requiring only basic Java knowledge.

The rest of the chapter is organised as follows: Section 8.1 introduces the Olivia architecture. Section 8.2 describes its main functionalities. In Section 8.3, four use cases that take advantage of our tool are described. In Section 8.4 the performance in terms of rendering speed and memory usage is analysed. Finally, conclusions are presented in Section 8.5.

8.1 OLIVIA

In this section, we describe how the tool has been implemented, how the Java packages are organised and show the graphical user interface.

8.1.1 Implementation

Olivia is written in the Java programming language for two main reasons. First, Java is one of the most popular programming languages. It has a higher popularity rank than many other languages (see TIOBE [228], RedMonk [187], and IEEE Spectrum [182] ratings) and higher rank than C++, the language used in the aforementioned tools. Second, Java is well known for its great portability, the source code is compiled to byte code that can be executed in any platform with the use of the Java Virtual Machine (JVM).

This software uses the Open Graphics Library (OpenGL), the industry's most widely used, supported, and best-documented API for rendering 2D and 3D graphics. To be able to use OpenGL from Java, the wrapper library Java OpenGL (JOGL) [87] is used, which implements Java bindings for OpenGL providing full access to the API, as well as almost all vendor extensions (OpenCL, OpenMAX, and OpenAL) through the use of the Java Native Interface (JNI). JOGL integrates with the Java own windowing system, the Abstract Window Toolkit (AWT), Swing and SWT widget sets, and with custom windowing toolkits using the NativeWindow API, as well as providing its own Native Windowing Toolkit (NEWT). The points are stored into Vertex Buffer Objects (VBOs), which allows vertex array data to be stored in video device memory for faster rendering. An interleaved vertex format is used to improve memory locality for each vertex.

Furthermore, the wrapper library JavaCV [12] is also used to access to some functionalities provided by the OpenCV [33] and FFmpeg [24] libraries. Finally, Substance [66] is used for the look and feel of the graphical user interface.

8.1.2 Package structure

Olivia is split into three main packages: core, extended and visualisation as shown in Figure 8.1. The core package implements all the basic functionalities, the extended package implements functionalities that are shared across visualisations, and the visualisation package implements all the visualisation-specific functionalities. To create a custom visualisation, it is necessary to implement an own visualisation package.

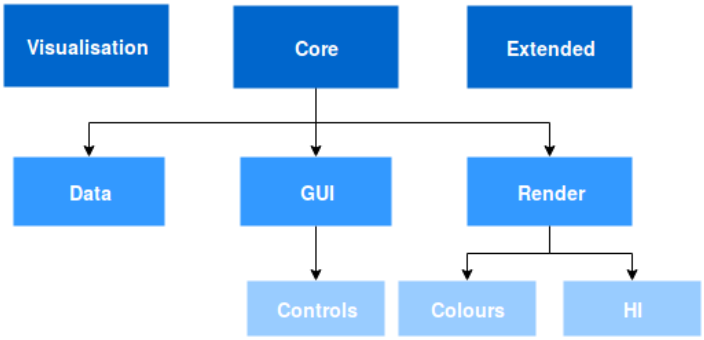


Figure 8.1: Package structure of Olivia.

- **Core**: This package is the backbone of the application and implements the main functionalities that are common to any type of visualisation. It acts as a layer of abstraction providing data structures and methods, from which users can benefit making little to none modifications. The package stores the main class of the application, which parses the arguments and manages the creation of visualisations. This package is split into the following sub packages.
  - Data: Specifies the characteristics of the data. There are three structures available: point, point array, and cluster. A point is understood as a 3D point. These points are encapsulated in a point array, which is a structure that provides access to the rendering methods. This point array can be encapsulated into a cluster, a structure with an identifier and some additional information such as the sum of the coordinates of the points.
  - GUI: Defines the graphical user interface, this is the graphical components and its controls. One of these components is reserved for the visualisations, which can be modified by the visualisation package.
  - Render: Provides all the methods related to the rendering. This package also manages the plotting of OpenGL primitives (e.g., lines, triangles, quads, etc.), the movement of the camera and the screen capture (image or video). Particularly two features are controlled by this package:
    * Colours: Manages the colours of the points.

    - HI: Manages the human interaction through the mouse and keyboard. The selection of points is handled using a Ray-Casting technique.

- **Visualisation**: This package contains the visualisation-specific logic. A visualisation needs to implement three main tasks: reading the data, displaying the data, and creating the visualisation panel. These methods are easy to implement by extending the methods inherited from the core package. Three visualisation types are incorporated to serve as guidance for developing tailored visualisations: basic, segmenter, and classifier.

- **Extended**: This package holds structures that are common to some visualisations, but not common to all, thus they are not included in the core package. In this manner, the duplication of code is minimised.

8.1.3 Graphical user interface

An overview of the interface is illustrated in Figure 8.2. The layout of the tool is composed of a menu bar holding the main options and three panels. The control panel, labelled 1 in the figure, holds some quick options (camera speed, point size, line width, centre at point), the information about the selected point and cluster, and the console output. This console gives feedback about the execution of the tool, for displaying errors and aiding troubleshooting. The render panel labelled 2 in the figure, is where the rendering is shown. Several visualisations can be displayed in the render panel, which are resized automatically. The visualisation panel labelled 3 in the figure, is set free for visualisation-specific controls.

8.2 Usage

In this section, we describe the main functionalities of the tool, how to download and install the software, and how to create custom visualisations.

8.2.1 Functionalities

Currently, Olivia main functionalities are the following.

- **Point selection**: Points can be selected using the left mouse click showing information about the point in the control panel. Additional information can be displayed in the visualisation panel.

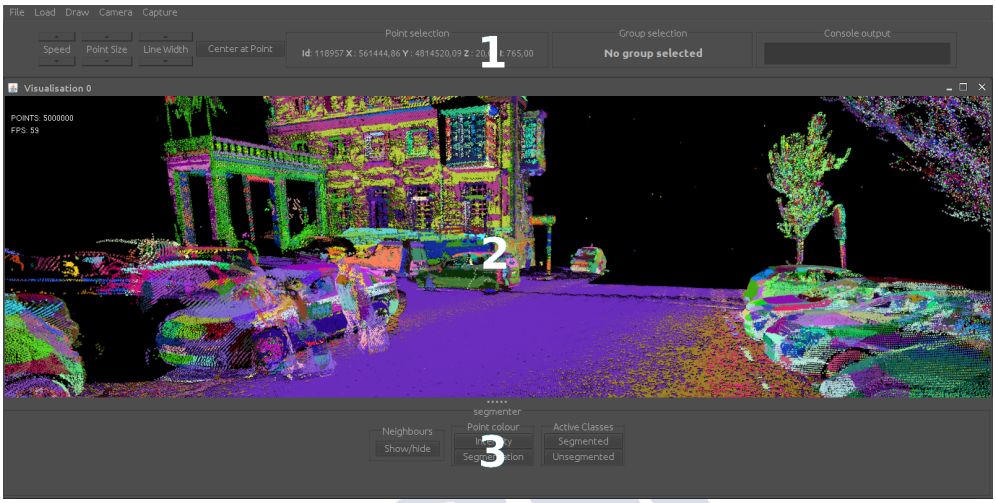


Figure 8.2: Screen capture of a segmentation visualisation showing the panel composition: control panel (1), render panel (2), and visualisation panel (3).

- **Cluster selection**: Clusters can be selected using the mouse wheel click, showing information about the cluster in the control panel. When the cluster is selected, only that cluster will be shown to give a clear view of it. This is particularly useful for analysing segmented point clouds when isolating a specific cluster makes the task easier. Note that this functionality is related to clustering methods.

- **Neighbours visualisation**: The neighbours of each point can be displayed on any visualisation if a neighbours file is available. Currently, Olivia supports the format used in the Fast Library for Approximate Nearest Neighbours (FLANN) [177]. Almost every point cloud processing method needs to compute the neighbourhood of the points, so its evaluation is a key issue. If this option is activated, the neighbourhood of the selected point will be shown, displaying a connection from that point to its neighbouring points as well as the distance or weight in each connection.

- **Drawing of OpenGL primitives**: Different representation of the point streams can be plotted. This allows drawing primitives such as lines, triangles, and quads on top of the point cloud. This is useful for displaying the output of different algorithms. For example, the hull of a set of points or the normal vector of each point can be rendered

overlaying the point cloud.

- **Camera state**: It is possible to save and load the camera state, namely, its position and rotations. This facilitates the comparison of point clouds across different executions of the algorithm under analysis.

- **Camera mirroring**: The camera of the visualisations can be synchronised, meaning that all visualisations will move simultaneously. This helps compare the same point cloud in different visualisations. For example, the user can run two different segmentation algorithms for the same point cloud, load each segmentation in one visualisation panel, and then activate the camera mirroring. This way the position of the point clouds will be the same regardless of the point cloud which is being moved, making the visual comparison much easier.

- **Stereoscopic 3D view**: It is possible to render the scene twice, producing one image for the left eye and another image for the right eye of the viewer. These images are combined when using 3D glasses to give the perception of depth. This feature is extremely useful for both in-depth analysis of the point cloud and demonstration purposes as shown in Figure 8.3. Note that because the scene is rendered twice, the computational cost doubles, which can be a performance issue for less powerful computers.

- **Screen capture**: Both image and video can be captured. Only the render screen is captured, hiding the other panels and taking into account the current resolution. If the stereoscopic 3D is enabled, two videos (left and right) are recorded to build the 3D video later.

8.2.2 Getting the tool

Both the source code and the executable are available in the official repository: `https://github.com/citiususc/olivia`. The application can be automatically built using the project management and build automation tool Apache Maven [170]. Installation and use instructions are provided in the repository. Because of being an open-source project, we hope to bring the community together, so developers can contribute to its development.

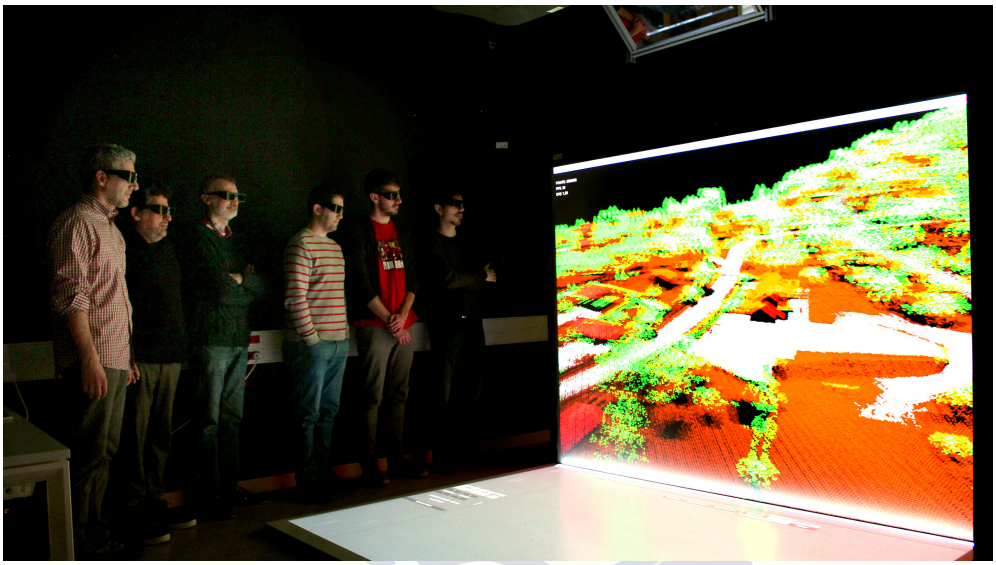


Figure 8.3: Demonstrating classification results using the 3D stereoscopic mode in the Airborne LiDAR Solutions stand. II Edition of Inside the Lab, a Technological Demonstration Day at CiTIUS.

8.2.3 Creating custom visualisations

The main advantage of Olivia is its simplicity to add new tailored visualisations. Developers only need to be familiarised with the Java programming language, although some OpenGL knowledge may be required for more complex visualisations. Adding a new visualisation is performed by adding a new package extending the default visualisation package. Then, three simple elements must be implemented: 1) reading the data, 2) displaying the data, and 3) creating the visualisation panel. These elements are easy to implement by using inherited functionalities provided by the tool. Displaying the point cloud can be performed by simply calling the drawing method of the point array structure holding the points. Source code for three different visualisations is included to provide examples on how to add custom visualisations and make the development easier. If there is the need for drawing functionalities not supported by Olivia, these should be implemented in Java classes within the custom visualisation package. Only if these functionalities benefit all types of visualisation will be added in the core package.

8.3 Evaluation

In this section, the usefulness of the tool in four use cases is shown, ranging from common point cloud processing scenarios to specific applications.

8.3.1 Visualising neighbours

The most common step for point cloud processing is the calculation of neighbours. Several neighbourhoods can be chosen for this task: kth Nearest Neighbours (kNN), spatial neighbours (voxel, sphere, cylinder...), feature neighbours, etc. The output of these methods is used as input to nearly every subsequent processing step, thus the query of neighbours is a critical task of any framework and it must be evaluated carefully. In this example, the calculation using a voxelised neighbourhood is inspected. For this, first, an octree is created to speed up the search. The vertices of the octree nodes are saved in a file and then loaded as quads primitives, as shown in the left image of Figure 8.4. With a glance we can have some awareness about the depth of the octree and how it is partitioned. This can also be done for only the nodes containing points as shown in the right image of the figure. In this example, the stopping criterion relates to the number of points: a node will continue splitting until the number of points inside is lower than a certain threshold. This overview of the leaf nodes can help to select the appropriate threshold.

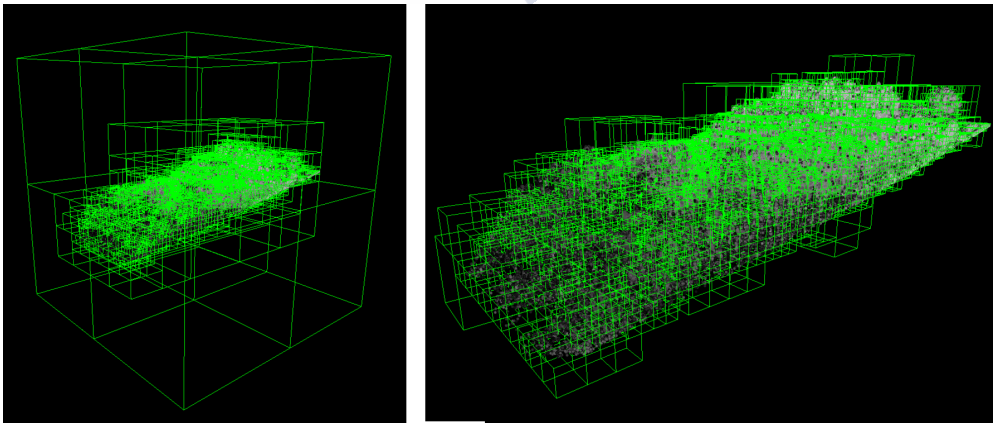


Figure 8.4: Octree nodes showed in green: All (left), only leafs (right).

Afterwards, the result of the computed neighbourhood is examined. For this, we provide

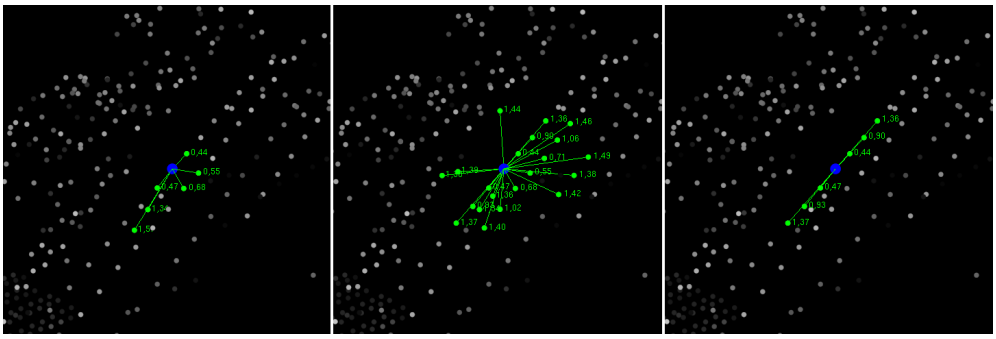


Figure 8.5: Roof point (in blue) and its neighbours (in green) for different methods: Triangular Irregular Network (left), sphere (center), and orientated cylinder (right).

the capability of displaying the neighbours of each point. In Figure 8.5 the neighbours of a conflictive building point (in blue) are shown, using three different neighbourhoods. This point at the boundary of the building shares neighbours with both the building itself (right to the point) and a nearby tree (left to the point). This situation can cause problems for further processing, e.g., estimating the normal vector of the point. Visualising the neighbours can help to verify that the selected neighbourhood model is appropriate. The feature of displaying the neighbours of a selected point is inherited by any type of visualisation if a file describing the neighbours is available.

8.3.2 Visualising segmentations

Another common step in point cloud processing is segmentation. In this section, our procedure to evaluate the segmentation proposed in [162] is described. As described in Chapter 2, this method performs two region-growing phases: the first one using the normal vector to extract only planar patches and the second one without it in the over-segmented clusters. There are some stages in this method which must be evaluated. First, the density of the point cloud can be estimated by computing the hull. To visualise it, the points of the hull have to be stored in a file. Then, the resultant hull could be plotted to visually verify that the calculation is correct. Once the point density is estimated, the size of the neighbourhood needed to compute normal vectors is chosen accordingly. We can also perform a visual inspection of the normal vectors. As mentioned before, the value of the normal vectors needs to be stored in a file to visualise them afterwards. In Figure 8.6, the visualisation of the normal vectors is shown. For

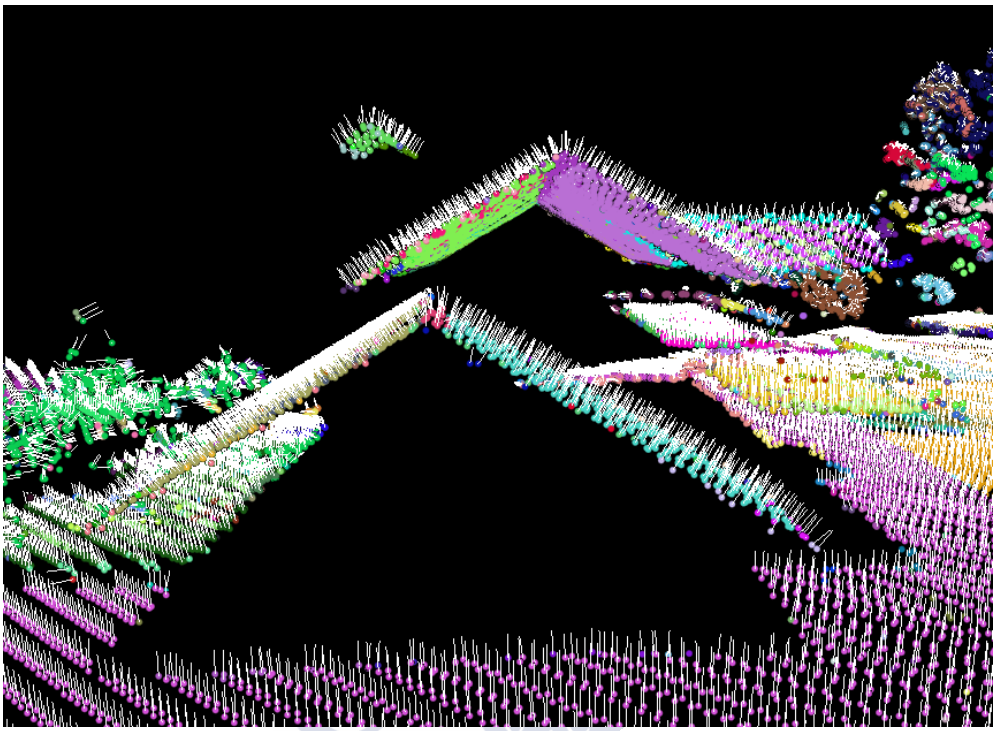


Figure 8.6: Normal vectors (white lines). Points coloured by cluster.

the end-user, it is important to see that sometimes the computation produces tricky situations, such as in boundaries among objects and areas with irregular point density.

After verifying that the previous steps are correct, the segmentation result itself is analysed. Both points and clusters can be selected with the mouse to provide information such as an identifier, coordinates, intensity, etc. Furthermore, if a cluster is selected only points of that specific cluster are displayed. This provides a clear overview of the cluster, allowing to quickly recognise its shape and boundaries. Some road segments can easily be identified in this particular segmentation, thus giving the idea that the algorithm is well suited for road classification. If the road clusters are selected, it can be quickly identified that they also include some asphalt of entries to houses nearby. This can be a problem depending on the application. Hints like this can be easily highlighted using the cluster selection feature.

Finally, different segmentation results can be visually compared. As shown in Chapter 2,

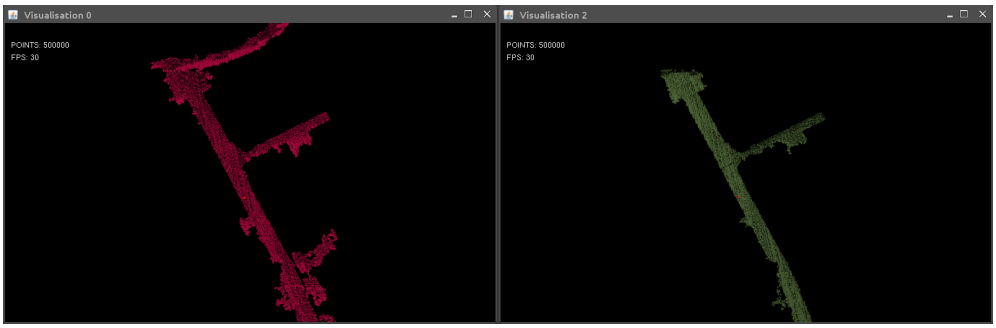


Figure 8.7: Usage of both selection and camera mirroring. Comparing the same cluster changing the intensity threshold in the segmentation process.

it is still difficult to assess whether one segmentation is more accurate than others, thus visual inspection is still important to perform the evaluation. The comparison can be performed among different segmentation methods or within the same method among different parameters. In this example, we address the later. First, two segmentations are loaded, each one in one window and the camera mirroring option is activated. This will facilitate the task by synchronising the camera movement among the segmentations. Furthermore, the selection can also be synchronised, greatly simplifying the comparison of single clusters between segmentations. For example, in Figure 8.7 the effects of modifying the intensity threshold parameter can be observed. The default value is set to 7.5% of the intensity range (right). If the threshold is changed to 15% (left), we see that the cluster can to include also points from the top road segment, but also includes more points near the boundaries of the road which belong to the ground. Olivia helps the user to identify and evaluate this kind of trade-offs.

8.3.3 Visualising Classifications and Beyond

As other visualisation tools, the results of classification can be displayed. Each class has its colour, and it is possible to enable or disable the drawing or each class. For example, all classes could be disabled except the ground class, to have a clear view of the terrain. The classification labels must be the defined in the LAS Specification [74].

One of the applications derived from the classification is the estimation of building footprints. Different algorithms can be used to compute the hull of the buildings. In a previous comparative of these methods [206] we used Olivia to visualise how they behave in different

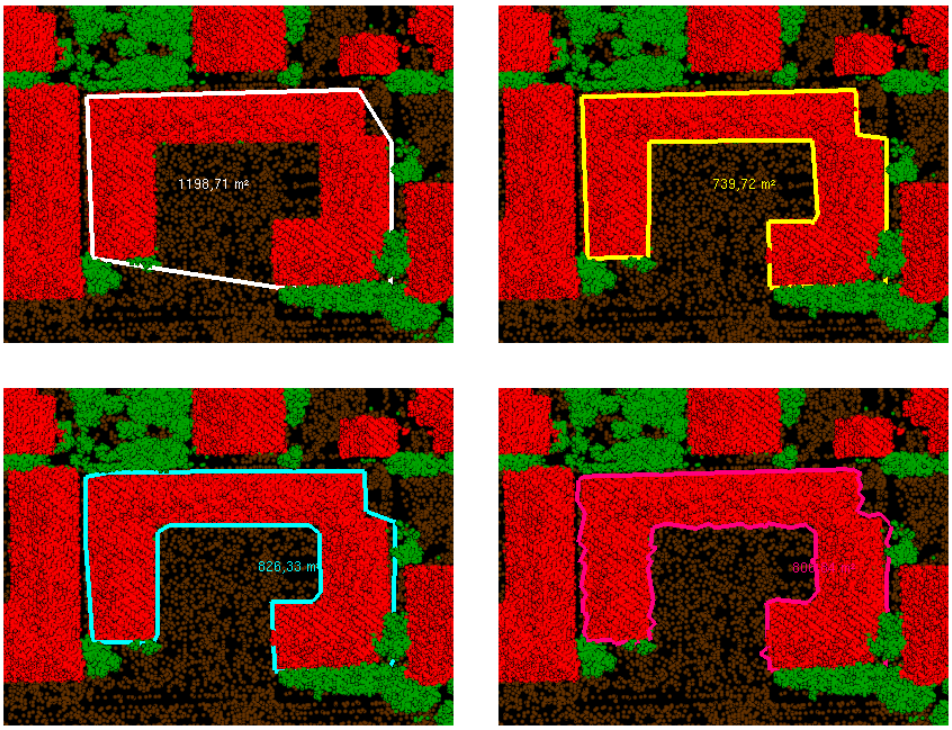


Figure 8.8: Computed hulls and areas using different algorithms; from left-to-right to top-to-bottom: convex, concave, exterior, voronoi.

scenarios. To this end, both the calculated hulls and areas for each algorithm can be plotted, as shown in Figure 8.8. This allows us to quickly identify the shortcomings of the algorithms for different building shapes.

8.3.4 Visualising Applications: Adaptation to a landing site detection

More specific applications will also benefit from the functionalities provided by Olivia. In this section, we present how the tool is used to analyse a Landing Site Detection Algorithm (LSDA) previously developed in [152]. In short, the algorithm classifies the suitability of the points for a helicopter to land. This classification takes into account the features for both the terrain (planarity, slope, etc.) and the helicopter (skid size, rotor size...). There are several

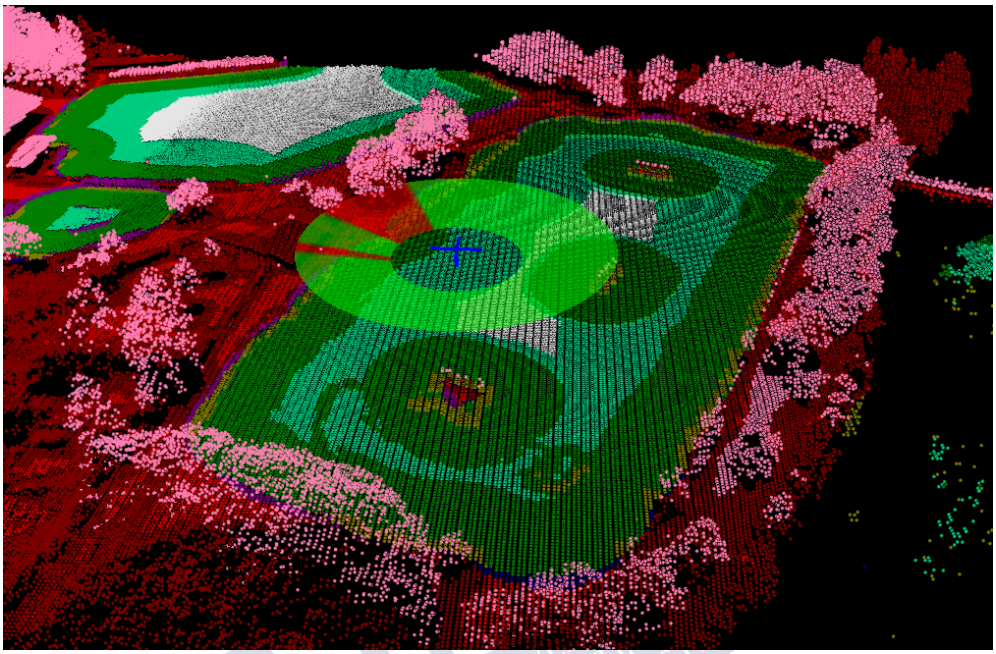


Figure 8.9: Result of the LSDA, the suitability for landing is represented in different colours. The approximation cone is shown for the selected point.

classification labels, which are colour coded for visualisation, from worse to better: unsuitable (red and light red), risky (blue), suitable (yellow, light green, green, and white).

The result of executing the algorithm in a particular case is shown Figure 8.9. While in the white areas, the helicopter can perform an approximation from any angle, in the rest of suitable areas, obstacles may invalidate some approximation angles. Thus it is necessary to plot from where the helicopter can come in those cases. For this, the approximation cone calculated by the algorithm is stored in a file, and then, OpenGL methods are used to plot our customised approximation cone (see Figure 8.10). The cone shows in green the angles from where the helicopter can approximate, and in red the angles discarded for the presence of obstacles. Its drawing is triggered by the selection of a point. Olivia allows us to short the time and effort needed to evaluate the results.

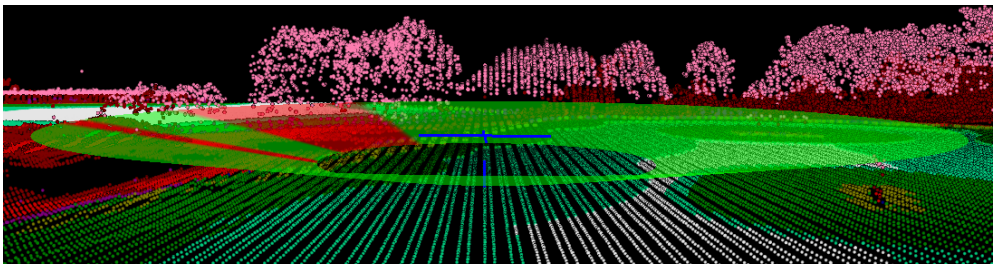


Figure 8.10: A detailed view of the approximation cone: landing position (blue bars), valid approximation angles (in green) and invalid approximation angles (in red).

8.4 Performance

Rendering speed and memory footprint are analysed to measure the performance of Olivia and to give an idea of the hardware requirements for a particular case of study. These measurements should be taken as lower limits, as further developments could improve performance considerably. A MLS point cloud of a street from Narón (Spain) is selected for the benchmark given its high density. The point cloud is split into five samples of different sizes ranging from 1 to 75 million points. Benchmarking was carried out in a Linux (Ubuntu 14.04.5 LTS) desktop computer with an Intel Core i7-4790 processor and 16 GB of RAM. Each sample is inspected using the basic visualisation mode.

The average frames per second (FPS) is used as a metric to measure rendering speed. This value is provided by a built-in method within the JOGL library. Three graphic devices were used to represent different ranges: an integrated Intel HD4600 for low-range, a Nvidia GTX 1050 Ti for mid-range and a Nvidia GTX 1080 Ti for high-range. Results are shown in Figure 8.11. While the integrated graphic struggles to achieve a smooth rendering (30 FPS) with 10 million points, both dedicated graphics achieve fluid renderings until 50 million points. For higher amounts of points, only the high-range graphic card still offers a smooth rendering.

The memory footprint is gathered for the host (RAM) and device (VRAM) memory. Furthermore, for the host memory, both allocated and used memory are measured. The host stores the full point data and Java-related objects, while the device only stores the coordinates and colours of the points. We use the NetBeans [181] Java profiling tool and the Nvidia X Server to measure host and device memory, respectively. Results are shown in Figure 8.12. The

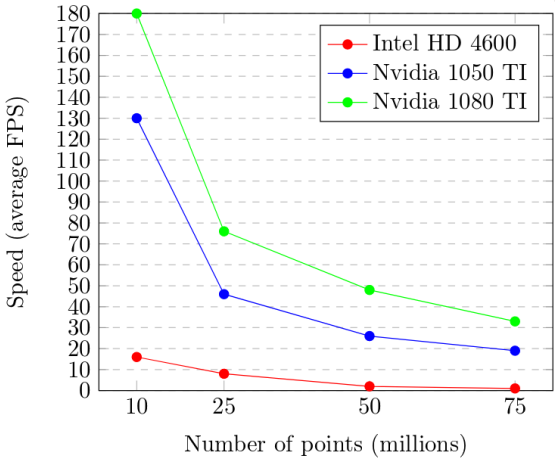


Figure 8.11: Rendering speed of Olivia for different graphic devices.

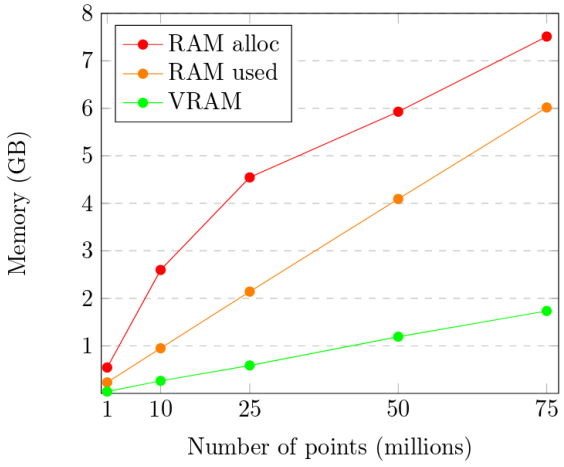


Figure 8.12: Memory usage of Olivia for host and device memory.

memory consumption for both host and device scales linearly. Note that for the host memory, Java allocates an excess of memory if plenty of memory still available, which can mislead the real memory usage.

As expected, the hardware requirements directly depend on the number of points to be displayed, hence they will vary from one user to another. Nevertheless, because is common

to work with points clouds of a few tens of millions of points, we recommend a system with at least 8 GB of RAM and a dedicated graphic device with 2 GB of VRAM.

8.5 Conclusion

LiDAR point cloud processing is a hot topic, and visualisation is important to understand, evaluate, and present these related research results. Given the broad number of possible applications, it is difficult to find a visualisation tool that meets the requirements of a specific application. Only an open-source tool can be flexible enough. Most of the currently available tools are not open source, and those which are open source may require a noticeable learning curve giving its wide scope. Because of this, we present a tool that provides the backbone for any type of visualisation, that can be easily extended and tailored to allow the creation of custom visualisations, quickly and with minimum effort. Researchers and users alike will benefit from such a tool.

The proposed tool, named Olivia, is written in Java, one of the most common languages, which allows high portability and clear package design. To perform the rendering it uses OpenGL, the most well-known API for 3D graphics. The tool offers out of the box several functionalities such as point cloud rendering, point/cluster selection, OpenGL primitives drawing or 3D stereoscopic view. Building custom visualisations becomes an easy task thanks to the packaging structure and the inherited functionalities of Olivia. Users can extend and modify the tool to fit their needs, giving high flexibility. We want to highlight the utility of the 3D stereoscopic view, not only it is a very engaging way to demonstrate results to the public but also is very well suited for error identification.

Through this chapter, we show how to use the tool in different processing scenarios: neighbourhood calculation, segmentation, classification, and landing site detection. Likewise, several images shown in this thesis have been captured with Olivia. Using this tool, the results can be inspected and valuable information can be obtained to validate and refine them. A brief benchmark shows that Olivia does not need high hardware requirements and a regular desktop computer is suitable to visualise tens of millions of points.

In the near future, we pretend to improve the control over the OpenGL primitives that can be drawn overlaying the point cloud, by adding the functionality to perform translations, rotations, and animations. We also want to add the capability of using heat maps and colours gradients. Other future work on this topic may be to add more visualisation modules, adapt

Olivia to virtual reality environments or improve the performance by implementing advanced rendering techniques.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Discussion

In this thesis, we addressed several steps of LiDAR data processing in order to classify the point cloud data into four of the most common classes: ground, vegetation, building and road. For this, both general-purpose methods and application-specific methods have been proposed. In this chapter, the main findings are presented, along with the limitations of the developed methods, their practical implications and recommendations for further investigation.

In a first approach, a general-purpose classification method was proposed, which consists of a region-growing segmentation followed by a rule-based classification.

It is important to note the relevance of the segmentation, as its result is the input for the rule-based classification so that it will heavily influence the classification result. Because of this, an improved version of the traditional region-growing algorithm was proposed. The quality of the method was evaluated using an adaptation of unsupervised metrics used for segmentation of images. This adaptation is subject to further research, particularly, it is interesting to study the implications of considering point features (e.g. height), that are not available in image data, in the global segmentation score. Experiments showed that the global segmentation score was reduced from 1.053 to 1.019 (lower is better). Furthermore, the segmentation was evaluated in regard to the rule-based classification results. Experiments showed of the overall accuracy increased by 3.57%, highlighting an increase in the building classification accuracy of almost 10%. Both evaluations demonstrate the superiority of the proposed segmentation method and make evident the need to perform a good segmentation.

After the segmentation, a rule-based classification was performed. This classification showed an overall accuracy of 83.3%, with an accuracy of 88.4%, 86.9% and 70.5%, for the ground, vegetation and building classes, respectively. While results are about 10% lower than the state of the art methods, this method does not require training. However, note that the comparison with other existing methods is far from being fair, as the accuracy metrics re-

ported by the authors have been taken in different datasets. Some initiatives try to address this issue, although the research community still has to work towards providing more comparable results by agreeing on using the same dataset and evaluation methodology. Nevertheless, the proposed classification yields an overall accuracy above 83%, is visually appealing in both rural and urban data and requires minimal parameter tuning. Therefore, we find the proposed classification to be a reliable option to perform a general-purpose classification. Future research could be focused on recognising more classes such as water or power lines.

In a second approach, application-specific methods were proposed, for the identification of the building, road and ground, exploring real-time processing for the later.

Alternatively, a plane-fitting clustering-based method is proposed to identify the building class. This method achieves 92.39% correctness and 92.64% completeness. The results are not superior to the ones achieved by the general-purpose classifier (98.68% correctness and 96.46% completeness), however, the execution time is severely reduced, being this approach twice as fast. Therefore, when developing a solution, it is important to consider the trade-offs between using a general-purpose or an application-specific approach. While a general-purpose could potentially open the way for new applications, a tailored solution can achieve comparable results for a certain application in much lower time. One concern is that quality metrics reported by this method and the rule-based classification are only available for the Vaihingen dataset, as it is the only publicly available and adopted dataset with reference data. Ideally, the methods should be tested in more datasets to provide more robust quality metrics.

The extraction of the road network is another task that has been addressed as a specific application. The extraction of the road network consists of two steps: Identification and delineation; both described in this thesis. The proposed identification method shows completeness, correctness and quality of 93%, 83% and 78%. The results are competitive with the state of the art, only surpassed by methods based on Conditional Random Fields, although note that the quality metrics of these methods have been calculated in a single study site. Therefore, our proposal is arguably more robust, as it has been tested in 10 study sites. As we expected, tests have shown that the main feature to distinguish road from ground points is the intensity, which is significantly lower in road points. The advantage of our proposal is the capability to automatically estimate the optimal intensity threshold for each study site. Some limitations still need to be addressed, such as the presence of water or flat surfaces with low intensity. Nevertheless, with these results, the proposed method makes practical the identification of road points in a fully automatic way. Furthermore, a framework for modelling the road was

proposed, which is able to extract the centreline, fill in the gaps and estimate its width. The proposed method makes use of image processing that has been adapted to LiDAR data. The task of delineating the road is still highly challenging and existing methods yield a low quality. While the proposed method shows promising results, there is still room for improvement. Currently, it needs significant user intervention so that future research must focus on increasing the degree of automation, in addition to carrying out a thorough evaluation.

To explore the feasibility of real-time processing, the most demanded LiDAR process was selected: ground filtering. To achieve high computational throughput, a scan-line based method was proposed. Experiments performed on synthetic test sites show Type 1 (commission) and Type 2 (omission) errors of 0.25% and 2.71%, respectively, and a kappa value of 88.59%. These results are very competitive with the state of the art methods. However, these results should be taken only as an approximation. It is expected that these results will decrease when using real data. Unfortunately, the reference data used in the literature are not suitable for scan-line processing so that real data could not be used in the quantitative evaluation. Nevertheless, visual inspection at the Alcoy, Vahingen and Trabada sites also shows satisfactory results. Execution times are low, and up to about 1 million points per second can be processed on a regular workstation. This makes it attractive for *offline* processing as well, as the trend is to acquire denser point clouds. To use the method in UAV systems, it was implemented into a low-cost SoC development board with FPGA acceleration. This implementation is capable of processing more than 60,000 points per second with low power consumption. The estimate of the expected performance on a higher-end development board shows that the method is capable of processing nearly 250,000 points per second with even greater energy efficiency. This computational throughput allows performing ground filtering in real-time, as it is higher than the data acquisition rate of some existing sensors.

For the development of the processing tasks introduced in this thesis, it is often required the visual analysis of experimental results. Since we have not found existing tools that suit our needs, we have developed our own visualisation tool. OLIVIA is an open-source tool focused on developers, that provides basic visualisation functionalities and that can be easily modified to build customised visualisations. Thanks to this tool, it has been easier to evaluate and show the work presented in this thesis. Another main advantage is the support for stereoscopic 3D view, which has been key for engaging with the audience in public demonstrations. The performance tests showed that OLIVIA can be used on a regular workstation. However, a dedicated graphics card is preferable for displaying point clouds of several millions of points.

For reference, with a mid-range graphic card, 50 million points can be smoothly displayed at 30 frames per second. Still, there is room to increase computational performance. Future research should focus on upgrading the tool to the latest OpenGL specification and make use of its new features.

All the methods presented in this thesis have been implemented efficiently. Thanks to the use of a mid-to-low level language, optimal spatial indexing and parallelisation, the implementations have low execution times. It is important to highlight the importance of the spatial indexing. As was said before, the neighbour's search is one recurrent calculation in most of the presented methods. It has been shown that a significant part of the execution time is spent on this task. Specifically, in the segmentation and classification, this task spends 68.32% and 72.55% of the time, respectively. Therefore, choosing the appropriate spatial indexing strategy is critical for computational performance. Another important element is the selection of the parallelisation strategy. As clusters can greatly vary in size, two different strategies have been used depending on their size. Regular-size clusters are parallelised at cluster-level with dynamic scheduling, while large-size clusters are parallelised at point-level with guided scheduling. In future work, more research should be conducted on how to select an optimal threshold to distinguish between regular and large clusters. Thanks to this strategy, the speedups achieved in this work have been close to the theoretical maximum speedup estimated with Amdahl's Law: 1.81 of 1.85 for the classification, 2.58 of 2.70 for the building extraction and 2.32 of 3.04 for the road extraction. As LiDAR data acquisition technologies are on ongoing development, and the data volume is increasing consequently, it is in the opinion of the author that implementing computationally efficient methods will be critical in the years to come.

In conclusion, the proposed methods show results that are competitive with existing methods but exhibiting lower execution times. Therefore, it can be considered that the work presented in this thesis is a meaningful contribution to the state of the art.

Conclusions

The following paragraphs detail the conclusions that have been extracted.

1. Literature shows that an object-based classification tends to achieve higher accuracy than a pixel or point-based classification. Therefore, is it beneficial to group the LiDAR points into regions, or objects. For this, in Chapter 2, we proposed a region growing based segmentation method that modifies the growing criteria, namely the epicentre. Experiments were conducted to measure the improvement of the segmentation quality. An unsupervised evaluation was performed, and that the segmentation's global score was reduced a 3.23% (lesser is better). This is satisfactory, however, the research on unsupervised evaluation is still ongoing and it is difficult to assess the reliability of such evaluation. This is why we also analysed how this improved segmentation would improve the subsequent classification step. Tests showed that the overall classification accuracy was increased a 3.57%, highlighting a building classification improvement of nearly 10%. With this, we can state that **the use of the epicentre improves both the quality of the segmentation and the following classification**.

2. Most of the exiting classification methods use machine learning techniques so that they require training data. This need introduces important implications. First, the creation of labelled data is a difficult task with high economical and temporal costs, so that only a small amount of reference data are often available. And second, due to the heterogeneous characteristics of point cloud data, achieving high performance on datasets different from those used for training is not guaranteed. Thus, in Chapter 3, we explored what can be achieved without labelled data, by introducing a rule-based classifier. A set of heuristic rules are applied to the objects to categorise them into three classes, *ground*, *vegetation* and *buildings*. Tests showed the method can achieve an overall accuracy

above 80%, which makes it suitable as a general-purpose classifier and demonstrated that **satisfactory classification results can be achieved without learning strategies**. However, the comparison with exiting methods was difficult since accuracy measures were obtained in different datasets, which lead us to the next point.

3. There is a worrying lack of benchmarking standards in the classification field. We showed that authors use evaluation methodologies that differ in: the used dataset, the strategy of creation of the reference data (points per class, points inside delineated polygons, whole areas...) and in the percentages of reference data in respect to the total data. Fortunately, many of the recent studies use the Vaihingen dataset created by the International Society for Photogrammetry and Remote Sensing (ISPRS) 3D in 2010. However, the amount of labelled data is rather small. Other datasets have been made public recently but their adaption is yet to be seen. Another alternative being currently explored is crowdsourcing. Nevertheless, these options are still prone to human error and require a great effort. Because of that, the author sees the **automatic generation of synthetic data as the most promising alternative for creating reference data**. In this thesis, we show how a LiDAR simulator, namely the Heidelberg LiDAR Operations Simulator (HELIOS), can be used in the experiments to generate synthetic labelled data and to perform quantitative evaluation effectively. We believe that more research should be focused on this topic, and we are looking forward to new developments.

4. Many methods for road extraction have been proposed during the last years. However, existing methods lack automation and their applicability among datasets is rather unknown. To solve this issue, in Chapter 5, we have introduced a new method for the automatic extraction of road points, which estimates the optimal intensity threshold automatically. Experiments were carried out in ten study sites with different acquisition methods and degrees of urbanisation without changing a single parameter. We demonstrated that both road and non-road points follow a normal distribution and that skewness balancing can be used to distinguish between the two classes of points. We also showed that, as a rule of thumb, road points are expected to yield an intensity lower than half of the intensity range, after removing the intensity outliers. Road points were extracted even for sites with little presence of roads. Results showed completeness, correctness and quality of 93%, 83% and 78%, respectively. These values make our proposal to be of real practical importance, and it is only surpassed by Conditional

Random Fields (CRF) methods, which require training. Overall, **tests demonstrate the suitability of our approach as a general method for road point extraction**.

5. Road points are generally used as input for road delineation and characterisation. The topic of road modelling is still in its infancy. Tests for extracting the road centreline in the ISPRS Vaihingen dataset show that only one state of the art method achieved a quality value above 50%. Therefore is critical to perform more research in this topic. Because of that, in Chapter 6, we adapted two methods from image processing, namely Angular Texture Signature (ATS) and Active Contour Models or snakes, into LiDAR data, to extract the road centreline and to fill the possible gaps, respectively, and we presented a new method for the estimation of the road width. Experiments showed the potential of our approach. Future work must be conducted to improve the current results and to perform an exhaustive evaluation. The author believes that **there are a plethora of valuable image processing methods that are yet to be adapted into LiDAR data**.

6. Processing point cloud data is computationally demanding, mainly because of the irregular distribution of the point set. Therefore, how the algorithms are implemented plays a critical role in order to achieve reduced execution times. In this thesis, the implementations have being developed in a middle-to-low level language and use efficient spatial indexing and parallelisation. **Some conclusions can be taken from the performance study**. First, an efficient spatial indexing strategy is critical, as the methods associated with the query of neighbours spent a very significant portion of the overall execution time. Second, the speedup achievable by parallelisation scales well with the number of physical cores while Hyper-Threading provided modest improvements. Third, while the optimal parallelisation strategy (point-based or cluster-based) and the optimal loop scheduling (static, dynamic, or guided) are hardly the same for every case, we have found an hybrid approach (point-level guided for large clusters and cluster-level dynamic for the rest) to be optimal strategy for our experiments, and possibly it would also be optimal in many other cases.

7. With the proliferation of Unmanned Aerial Vehicle (UAV) systems, the demand for real-time LiDAR processing is increasing. In chapter 7, we addressed this issue for one of the most basic and demanded processing steps, namely ground-filtering. This was achieved by exploiting the scan line fashion acquisition pattern of the data with 1-D iterative spline interpolation. Experiments showed satisfactory results in both urban and

rural areas. To guarantee its accuracy is important to selected an appropriate scan-line length, formally known as swath width, when planning the survey. Guidelines for its selection are given and its influence was studied. We believe the swath width requirement is reasonable for most surveys. If this cannot be met, some non-ground objects might be wrongly classified as ground, especially if the biggest side of the object is aligned with the scan line direction. Furthermore, we have developed an embedded system implementation on a Zedboard with custom hardware acceleration on the FPGA. Extrapolating its performance if implemented in a high-end board, the method can process nearly 250,000 points per second, which is as many points as some current lightweight scanners can acquire, with low-energy consumption. Overall, we demonstrated that **near real-time processing, in particular ground filtering, is feasible**.

8. During the development of this thesis, we had trouble finding visualisation tool, simple, flexible and modifiable, that suited our needs. Many of the existing visualisation tools are not open source, so they are not modifiable, and those which are open source often require a noticeable learning curve giving its wide scope. Because of this, we opt for developing our solution, OLIVIA, introduced in Chapter 8. OLIVIA is an OpenGL based open-source tool written in Java that provides the backbone for any type of visualisation, that can be easily extended and tailored to allow the creation of custom visualisations, quickly and with minimum effort. Also, it supports 3D stereoscopic view, which is very useful for both visually inspecting the results and for performing engaging demonstrations. Although the Java implementation has advantages such as modularity, portability or developer reach, we also found disadvantages, particularly, a higher memory footprint. The Java Virtual Machine (JVM) tends to allocate an amount of memory far higher than the memory being used. Also, there is memory overhead associated with JVM tasks such as garbage collection or code optimisation during runtime. With all, **OLIVIA has been useful and it has been used actively during the development of this thesis**, both for evaluating the proposed methods and for performing public demonstrations at CiTIUS.

9. LiDAR data acquisition is ongoing development and future trends points towards more accurate and denser point clouds. Although increasing the pulse repetition rate is a top priority for LiDAR manufacturers, as this reduces acquisition time and costs, is in the opinion of the author that **more effort should be focused on generating more regularly distributed point clouds**. Achieving a regular point distribution not only

relies on the LiDAR manufacturers but also in the users that design the survey plan. It has been shown that an irregular point distribution can hinder the establishment of the neighbourhood model (see Chapter 4) and reduce computational performance (see Chapter 7). Furthermore, it is expected that the increase of density follows the law of diminishing returns so that after an optimal level of density is reached adding additional density will not give significant benefits.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Bibliography

[1] James B Abshire. Nasa's space lidar measurements of earth and planetary surfaces. In *Frontiers in Optics*, page SMB1. Optical Society of America, 2010.

[2] Advanced Micro Devices. Processor specifications. `https://www.amd.com/en/products/specifications/processors/`. Accessed: 2020-06-04.

[3] James Ahrens, Berk Geveci, Charles Law, C Hansen, and C Johnson. ParaView: An end-user tool for large-data visualization. *Visualization Handbook*, 717, 2005.

[4] Zeynep Akbulut, Samed Özdemir, Hayrettin Acar, and Fevzi Karsli. Automatic building extraction from image and lidar data with active contour segmentation. *Journal of the Indian Society of Remote Sensing*, 46(12):2057–2068, 2018.

[5] Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM*, 17(4):589–602, 1970.

[6] Cici Alexander, Kevin Tansey, Jörg Kaduk, David Holland, and Nicholas J. Tate. Backscatter coefficient as an attribute for the classification of full-waveform airborne laser scanning data in urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(5):423 – 432, 2010.

[7] Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485, 1967.

[8] Katharina Anders. Research using HELIOS. `https://github.com/GIScience/helios#research-using-helios`. Accessed: 2020-06-22.

[9] Andreas Ullrich, Martin Pfennigbauer, Peter Rieger. How to read your LIDAR spec – a comparison of single-laser-output and multi-laser-output LIDAR instruments. http://www.riegl.com/uploads/tx_pxpriegldownloads/RIEGL_Airborne_Laser_Scanning-what_to_expect_from_your_sensor_2013-03-08_03.pdf, 2013 (Accessed: 02/06/2020).

[10] Zurui Ao, Yanjun Su, Wenkai Li, Qinghua Guo, and Jing Zhang. One-class classification of airborne lidar data in urban areas using a presence and background learning algorithm. *Remote Sensing*, 9(10), 2017.

[11] ArcGIS. Arcgis desktop - create and share maps, analytics, and data. https://desktop.arcgis.com/en/. Accessed: 2020-06-19.

[12] Samuel Audet. JavaCV: Java interface to OpenCV, FFmpeg, and more. https://github.com/bytedeco/javacv. Accessed: 2018-05-25.

[13] M. Awadallah, L. Abbott, and S. Ghannam. Segmentation of sparse noisy point clouds using active contour models. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 6061–6065, 2014.

[14] Mohammad Awrangjeb and Clive S Fraser. Rule-based segmentation of lidar point cloud for automatic extraction of building roof planes. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci*, 2(3/W1):1–6, 2013.

[15] Mohammad Awrangjeb and Clive S. Fraser. Automatic segmentation of raw lidar data for extraction of building roofs. *Remote Sensing*, 6(5):3716–3751, 2014.

[16] Mohammad Awrangjeb, Mehdi Ravanbakhsh, and Clive S. Fraser. Automatic detection of residential buildings using lidar data and multispectral imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(5):457 – 467, 2010.

[17] Peter Axelsson. Dem generation from laser scanner data using adaptive tin models. *International archives of photogrammetry and remote sensing*, 33(4):110–117, 2000.

[18] Zahra Azizi, Akbar Najafi, and Saeed Sadeghian. Forest road detection using lidar data. *Journal of Forestry Research*, 25(4):975–980, Dec 2014.

[19] Babcock International. Trusted to deliver. https://www.babcockinternational.com/. Accessed: 2019-07-30.

[20] E.P Baltsavias. Airborne laser scanning: basic relations and formulas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2):199 – 214, 1999.

[21] E.P. Baltsavias. Airborne laser scanning: existing systems and firms and other resources. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2):164 – 198, 1999.

[22] Marc Bartels and Hong Wei. Threshold-free object and ground point separation in LIDAR data. *Pattern Recognition Letters*, 31(10):1089 – 1099, 2010.

[23] S. Bechtold and B. Höfle. HELIOS: A multi-purpose lidar simulation framework for research, planning and training of laser scanning operations with airborne, ground-based mobile and stationary platforms. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-3:161–168, 2016.

[24] Fabrice Bellard, M Niedermayer, et al. FFmpeg. a complete, cross-platform solution to record, convert and stream audio and video. `https://www.ffmpeg.org/`, 2000. Accessed: 2018-05-25.

[25] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[26] P. J. Besl and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2):167–192, March 1988.

[27] Gary Bishop, Greg Welch, et al. An introduction to the kalman filter. *Proc of SIGGRAPH, Course*, 8(27599-23175):41, 2001.

[28] Blaise Barney and Lawrence Livermore. OpenMP tutorial. `https://computing.llnl.gov/tutorials/openMP/`, 2019 (Accessed: 04/06/2020).

[29] Andrew Blake and Michael Isard. *Active contours: the application of techniques from graphics, vision, control theory and statistics to visual tracking of shapes in motion*. Springer Science & Business Media, 2012.

[30] T. Blaschke. Object based image analysis for remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(1):2 – 16, 2010.

[31] Thomas Blaschke, Stefan Lang, Eric Lorup, Josef Strobl, and Peter Zeil. Object-oriented image processing in an integrated gis/remote sensing environment and perspectives for environmental applications. *Environmental information for planning, politics and the public*, 2:555–570, 2000.

[32] Bart Braden. The surveyor's area formula. *The College Mathematics Journal*, 17(4):326–337, 1986.

[33] G. Bradski. The Open Source Computer Vision Library (OpenCV). *Dr. Dobb's Journal of Software Tools*, 2000.

[34] Roberto Brunelli. *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

[35] Sandra Buján, Eduardo González-Ferreiro, Fabián Reyes-Bueno, Laura Barreiro-Fernández, Rafael Crecente, and David Miranda. Land use classification from lidar data and ortho-images in a rural area. *The Photogrammetric Record*, 27(140):401–422, 2012.

[36] James B Campbell and Randolph H Wynne. *Introduction to remote sensing*. Guilford Press, 2011.

[37] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *International journal of computer vision*, 22(1):61–79, 1997.

[38] A. P. Charaniya, R. Manduchi, and S. K. Lodha. Supervised parametric classification of aerial lidar data. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 30–30, June 2004.

[39] Nesrine Chehata, Li Guo, and Clément Mallet. Airborne lidar feature selection for urban classification using random forests. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(Part 3):W8, 2009.

[40] Chuanfa Chen, Yanyan Li, Wei Li, and Honglei Dai. A multiresolution hierarchical classification algorithm for filtering airborne LiDAR data. *ISPRS journal of photogrammetry and remote sensing*, 82:1–9, 2013.

[41] Chuanfa Chen, Yanyan Li, Na Zhao, Jinyun Guo, and Guolin Liu. A fast and robust interpolation filter for airborne lidar point clouds. *PLOS ONE*, 12(5):1–20, 05 2017.

[42] Dong Chen, Liqiang Zhang, Jonathan Li, and Rei Liu. Urban building roof segmentation from airborne lidar point clouds. *International journal of remote sensing*, 33(20):6497–6515, 2012.

[43] Yanming Chen, Liang Cheng, Manchun Li, Jiechen Wang, Lihua Tong, and Kang Yang. Multiscale grid method for detection and reconstruction of building roofs from airborne lidar data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(10):4081–4094, 2014.

[44] Yunhao Chen, Wei Su, Jing Li, and Zhongping Sun. Hierarchical object oriented classification using very high resolution imagery and lidar data over urban areas. *Advances in Space Research*, 43(7):1101 – 1110, 2009.

[45] Z. Chen and B. Gao. An object-based method for urban land cover classification using airborne lidar data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(10):4243–4254, Oct 2014.

[46] Simon Clode, Franz Rottensteiner, Peter Kootsookos, and Emanuel Zelniker. Detection and vectorization of roads from LiDAR data. *Photogrammetric Engineering & Remote Sensing*, 73(5):517–535, 2007.

[47] Simon P Clode, Emanuel E Zelniker, Peter J Kootsookos, and I Vaughan L Clarkson. A phase coded disk approach to thick curvilinear line detection. In *2004 12th European Signal Processing Conference*, pages 1147–1150. IEEE, 2004.

[48] Laurent D Cohen. On active contour models and balloons. *CVGIP: Image understanding*, 53(2):211–218, 1991.

[49] Russell G Congalton. A review of assessing the accuracy of classifications of remotely sensed data. *Remote sensing of environment*, 37(1):35–46, 1991.

[50] Michael Cramer. The DGPF-test on digital airborne camera evaluation–overview and test design. *Photogrammetrie-Fernerkundung-Geoinformation*, 2010(2):73–82, 2010.

[51] Louise H Crockett, Ross A Elliot, Martin A Enderwitz, and Robert W Stewart. *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc*. Strathclyde Academic Media, 2014.

[52] Jerome Demantke, Clément Mallet, Nicolas David, and Bruno Vallet. Dimensionality based scale selection in 3d LiDAR point clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(Part 5):W12, 2011.

[53] N. Demir and E. Baltsavias. Automated modeling of 3d building roofs using image and lidar data. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, I-4:35–40, 2012.

[54] Vivek Dey, Yonghui Zhang, and Maosheng Zhong. A review on image segmentation techniques with remote sensing perspective. In *ISPRS TC VII Symposium – 100 Years*, 2010.

[55] G Dial, L Gibson, and R Poulsen. Ikonos satellite imagery and its use in automated road extraction. *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, 3:357–367, 2001.

[56] Dirección General del Catastro Logotipo institucional del Ministerio de Hacienda. Resultados del procedimiento de regularización en municipios finalizados. `http://www.catastro.minhap.gob.es/esp/regu1.asp`. Accessed: 2020-05-19.

[57] Peter Doucette, Peggy Agouris, Anthony Stefanidis, and Mohamad Musavi. Self-organised clustering for road extraction in classified imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 55(5):347 – 358, 2001.

[58] Shouji Du, Yunsheng Zhang, Zhengrong Zou, Shenghua Xu, Xue He, and Siyang Chen. Automatic building extraction from lidar data fusion of point and grid-based features. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130:294–307, 2017.

[59] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[60] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[61] Richard O Duda, Peter E Hart, and David G Stork. Pattern classification, hoboken, 2001.

[62] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, July 1983.

[63] Ahmed F Elaksher, James S Bethel, et al. Reconstructing 3d buildings from lidar data. *International Archives Of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/A):102–107, 2002.

[64] G. M. Espindola, G. Camara, I. A. Reis, L. S. Bins, and A. M. Monteiro. Parameter selection for region-growing image segmentation algorithms using spatial autocorrelation. *International Journal of Remote Sensing*, 27(14):3035–3040, 2006.

[65] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, pages 226–231. AAAI Press, 1996.

[66] Kirill Grouchnikov et al. Substance Java look and feel. `http://insubstantial.github.io/insubstantial/substance/`. Accessed: 2018-05-25.

[67] Federal Emergency Management Agency (FEMA). Appendix 4B. Airborne Light Detection and Raning Systems . `https://www.fema.gov/media-library-data/20130726-1553-20490-5126/lidar_4b.txt`, 2009 (Accessed: 05/11/2019).

[68] António Ferraz, Clément Mallet, and Nesrine Chehata. Large-scale road detection in forested mountainous areas using airborne topographic lidar data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 112:23–36, 2016.

[69] S. Filin and G. Vosselman. Adjustment of airborne laser altimetry strips. In *ISPRS 2004 : proceedings of the XXth ISPRS congress : Geo-imagery bridging continents, 12-23 July 2004, Istanbul, Turkey. Comm. III WG III/3.*, pages 258–263. International Society for Photogrammetry and Remote Sensing (ISPRS), 2004.

[70] Sagi Filin. Surface clustering from airborne laser scanning data. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/A):119–124, 2002.

[71] Sagi Filin. Recovery of systematic biases in laser altimetry data using natural surfaces. *Photogrammetric Engineering & Remote Sensing*, 69(11):1235–1242, 2003.

[72] Raphael A. Finkel and Jon Louis Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974.

[73] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[74] The American Society for Photogrammetry & Remote Sensing (ASPRS). LAS Specification Version 1.4. In *ASPRS Board Meeting; ASPRS: Bethesda, MD, USA*, Nov 2011.

[75] Free Software Foundation, Inc. Gsl - gnu scientific library. `https://www.gnu.org/software/gsl/`. Accessed: 2019-07-30.

[76] Peter Friess. Toward a rigorous methodology for airborne laser mapping. *Proceedings EuroCOW*, pages 25–27, 2006.

[77] Fugro. Fugroviewer. `https://www.fugro.com/about-fugro/our-expertise/technology/fugroviewer`. Accessed: 2019-07-30.

[78] E. Funk. and A. Börner. Infinite 3d modelling volumes. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: VISAPP, (VISIGRAPP 2016)*, pages 246–253. INSTICC, SciTePress, 2016.

[79] D Gatziolis and HE Andersen. A guide to lidar data acquisition and processing for the forests of the pacific northwest.(no. pnw-gtr-768). *US Department of Agriculture, Forest Service, Pacific Northwest Research Station, Portland, OR. https://doi.org/10.2737/PNW-GTR-768*, 2008.

[80] JD Gautier and BW Parkinson. Using the gps/ins generalized evaluation tool (giget) for the comparison of loosely coupled, tightly coupled and ultra-tightly coupled integrated navigation systems. In *Proceedings of the 59th annual meeting of the institute of navigation and CIGTF 22nd guidance test symposium*, pages 65–76, 2001.

[81] Laurie Gibson. Automation technology with ikonos satellite imagery. In *Sensors, Systems, and Next-Generation Satellites V*, volume 4540, pages 650–660. International Society for Optics and Photonics, 2001.

[82] Alejandro Lorenzo Gil, Laia Núñez-Casillas, Martin Isenburg, Alfonso Alonso Benito, José Julio Rodrigo Bello, and Manuel Arbelo. A comparison between lidar and photogrammetry digital terrain models in a forest area on tenerife island. *Canadian Journal of Remote Sensing*, 39(5):396–409, 2013.

[83] Daniel Girardeau-Montaut. CloudCompare: 3D point cloud and mesh processing software. `https://www.danielgm.net/cc/`. Accessed: 2018-05-25.

[84] C. Glennie. Rigorous 3d error analysis of kinematic scanning lidar systems. *Journal of Applied Geodesy*, 1(3):147 – 157, 2008.

[85] GNU Project. Options that control optimization. `https://gcc.gnu.org/onlinedocs/gcc-7.5.0/gcc/Optimize-Options.html`, 2020 (Accessed: 04/06/2020).

[86] Gobierno de La Rioja. IDE Rioja. `https://iderioja.github.io/clasificacion_lidar/`. Accessed: 2019-04-25.

[87] S Gothel, R Santina, and P Koutsolampros. JOGL: Java binding for the OpenGL API. `http://jogamp.org/jogl/www/`. Accessed: 2018-05-25.

[88] Susan L Graham, Peter B Kessler, and Marshall K Mckusick. Gprof: A call graph execution profiler. *ACM Sigplan Notices*, 17(6):120–126, 1982.

[89] E Grilli, F Menna, and F Remondino. A review of point clouds segmentation and classification algorithms. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:339, 2017.

[90] Bo Guo, Xianfeng Huang, Fan Zhang, and Gunho Sohn. Classification of airborne laser scanning data using jointboost. *ISPRS Journal of Photogrammetry and Remote Sensing*, 100:71 – 83, 2015. High-Resolution Earth Imaging for Geospatial Information.

[91] Li Guo, Nesrine Chehata, Clément Mallet, and Samia Boukir. Relevance of airborne lidar and multispectral image data for urban scene classification using random forests. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(1):56 – 66, 2011.

[92] Qinghua Guo. Luquillo CZO Rio Blanco and Rio Mameyes airborne lidar. National Center for Airborne Laser Mapping (NCALM), distributed by OpenTopography. https://doi.org/10.5069/G9BZ63ZR, 2010.

[93] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, 1984.

[94] Ayman F Habib, Ana P Kersting, Ahmed Shaker, and Wai-Yeung Yan. Geometric calibration and radiometric correction of lidar data and their impact on the quality of derived products. *Sensors*, 11(9):9069–9097, 2011.

[95] Robert M. Haralick and Linda G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100 – 132, 1985.

[96] GJ Hay and G Castilla. Object-based image analysis: strengths, weaknesses, opportunities and threats (swot). In *Proc. 1st Int. Conf. OBIA*, pages 4–5, 2006.

[97] Y. He, C. Zhang, and C. S. Fraser. An energy minimization approach to automated extraction of regular building footprints from airborne lidar data. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3:65–72, 2014.

[98] Marcus Hebel and Uwe Stilla. Pre-classification of points and segmentation of urban objects by scan line analysis of airborne lidar data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. 38 (3/W4), pp. 187-192.*, 01 2008.

[99] Jeff Hecht. Beam: The race to make the laser. *Opt. Photon. News*, 16(7):24–29, Jul 2005.

[100] C. Heipke, H. Mayer, C. Wiedemann, and O. Jamet. Evaluation of automatic road extraction. In *International Archives of Photogrammetry and Remote Sensing*, pages 47–56, 1997.

[101] Bernhard Höfle, Thomas Geist, Martin Rutzinger, and Norbert Pfeifer. Glacier surface segmentation using airborne laser scanning point cloud and intensity data. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(Part 3):W52, 2007.

[102] Bernhard Höfle, Werner Mücke, Marieke Dutter, Martin Rutzinger, and Peter Dorninger. Detection of building regions using airborne lidar–a new combination of raster and point cloud based gis methods. In *GI-Forum 2009-International Conference on Applied Geoinformatics, Salzburg. accepted*, 2009.

[103] Bernhard Höfle and Norbert Pfeifer. Correction of laser scanning intensity data: Data and model-driven approaches. *ISPRS journal of photogrammetry and remote sensing*, 62(6):415–433, 2007.

[104] Bernhard Höfle and Martin Rutzinger. Topographic airborne lidar in geomorphology: A technological perspective. *Zeitschrift für Geomorphologie, Supplementary Issues*, 55(2):1–29, 04 2011.

[105] Gerard J. Holzmann. The power of ten – rules for developing safety critical code. *NASA/JPL Laboratory for Reliable Software*, 2006.

[106] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon, and R. B. Fisher. An experimental comparison of range image segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):673–689, July 1996.

[107] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4507–4515, 2017.

[108] Han Hu, Yulin Ding, Qing Zhu, Bo Wu, Hui Lin, Zhiqiang Du, Yeting Zhang, and Yunsheng Zhang. An adaptive surface filter for airborne laser scanning point clouds by means of regularization and bending energy. *ISPRS Journal of Photogrammetry and Remote Sensing*, 92:98 – 111, 2014.

[109] X. Hu, X. Li, and Y. Zhang. Fast filtering of lidar point cloud in urban areas based on scan line segmentation and gpu acceleration. *IEEE Geoscience and Remote Sensing Letters*, 10(2):308–312, March 2013.

[110] X Hu and CV Tao. A reliable and fast ribbon road detector using profile analysis and model-based verification. *International Journal of Remote Sensing*, 26(5):887–902, 2005.

[111] Xiangyun Hu, Yijing Li, Jie Shan, Jianqing Zhang, and Yongjun Zhang. Road centerline extraction in complex urban scenes from LiDAR data based on multiple features. *IEEE Transactions on Geoscience and Remote Sensing*, 52(11):7448–7456, 2014.

[112] Xiangyun Hu, C Vincent Tao, and Yong Hu. Automatic road extraction from dense urban area by integrated processing of high resolution imagery and lidar data. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35(B3):288–292, 2004.

[113] Xiangyun Hu and Lizhi Ye. A fast and simple method of building detection from lidar data based on scan line analysis. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:W1, 2013.

[114] Xiangyun Hu, Lizhi Ye, Shiyan Pang, and Jie Shan. Semi-global filtering of airborne lidar data for fast extraction of digital terrain models. *Remote Sensing*, 7(8):10996–11015, 2015.

[115] Zhenyang Hui, Youjian Hu, Shuanggen Jin, and Yao Ziggah Yevenyo. Road centerline extraction from airborne LiDAR point cloud based on hierarchical fusion and optimization. *ISPRS Journal of Photogrammetry and Remote Sensing*, 118:22–36, 2016.

[116] Zhenyang Hui, Youjian Hu, Yao Ziggah Yevenyo, and Xianyu Yu. An improved morphological algorithm for filtering airborne lidar point cloud based on multi-level kriging interpolation. *Remote Sensing*, 8(1), 2016.

[117] Ibeo Automative Systems GmbH. Ibeo lux hd - technical facts. `http://www.abott-mf.com/images/pdf/IbeoLUXHD.pdf`. Accessed: 2019-06-03.

[118] Intel Corporation. Products formerly comet lake. `https://ark.intel.com/content/www/us/en/ark/products/codename/90354/comet-lake.html#@Desktop`. Accessed: 2020-06-04.

[119] Intel Corporation. Getting started with OpenMP. `https://software.intel.com/content/www/us/en/develop/articles/getting-started-with-openmp.html`, 2012 (Accessed: 04/06/2020).

[120] ISPRS Working Group III/3. Filtertest - test sites. `https://www.itc.nl/isprs/wgIII-3/filtertest/downloadsites/`. Accessed: 2019-01-24.

[121] Albert V Jelalian. *Laser radar systems*. Artech House, 1992.

[122] P. Jiangui and G. Guang. A method for main road extraction from airborne lidar data in urban area. In *2011 International Conference on Electronics, Communications and Control (ICECC)*, pages 2425–2428, Sep. 2011.

[123] Andreas Jochem, Bernhard Höfle, Martin Rutzinger, and Norbert Pfeifer. Automatic roof plane detection and analysis in airborne LiDAR point clouds for solar potential assessment. *Sensors*, 9(7):5241–5262, 2009.

[124] Brian Johnson and Zhixiao Xie. Unsupervised image segmentation evaluation and refinement using a multi-scale approach. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(4):473 – 483, 2011.

[125] J. Jung, E. Pasolli, S. Prasad, J. C. Tilton, and M. M. Crawford. A framework for land cover classification using discrete return lidar data: Adopting pseudo-waveform and hierarchical segmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(2):491–502, Feb 2014.

[126] Mostafa Kabolizade, Hamid Ebadi, and Salman Ahmadi. An improved snake model for automatic extraction of buildings from urban aerial images and lidar data. *Computers, Environment and Urban Systems*, 34(5):435–441, 2010.

[127] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Fluids Engineering*, 82(1):35–45, 03 1960.

[128] Kirsi Karila, Leena Matikainen, Eetu Puttonen, and Juha Hyyppä. Feasibility of multispectral airborne laser scanning data for road mapping. *IEEE Geoscience and Remote Sensing Letters*, 14(3):294–298, 2017.

[129] Alireza G. Kashani, Michael J. Olsen, Christopher E. Parrish, and Nicholas Wilson. A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors*, 15(11):28099–28128, 2015.

[130] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[131] Keqi Zhang, Shu-Ching Chen, D. Whitman, Mei-Ling Shyu, Jianhua Yan, and Chengcui Zhang. A progressive morphological filter for removing nonground measurements from airborne lidar data. *IEEE Transactions on Geoscience and Remote Sensing*, 41(4):872–882, April 2003.

[132] M. Kim, M. Madden, and T. Warner. Estimation of optimal image object size for the segmentation of forest stands with multispectral ikonos imagery. In Thomas Blaschke, Stefan Lang, and Geoffrey J. Hay, editors, *Object-Based Image Analysis: Spatial Concepts for Knowledge-Driven Remote Sensing Applications*, pages 291–307. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[133] Seongjoon Kim, Seonghong Min, Geunhan Kim, Impyeong Lee, and Chulmin Jun. Data simulation of an airborne lidar system. In *Laser Radar Technology and Applications XIV*, volume 7323, page 73230C. International Society for Optics and Photonics, 2009.

[134] Donald T Knight. Rapid development of tightly-coupled gps/ins systems. In *Proceedings of Position, Location and Navigation Symposium-PLANS'96*, pages 300–305. IEEE, 1996.

[135] K. Koster and M. Spann. Mir: an approach to robust clustering-application to range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):430–444, May 2000.

[136] K. Kraus and N. Pfeifer. Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53(4):193 – 203, 1998.

[137] Pankaj Kumar, Conor P McElhinney, Paul Lewis, and Timothy McCarthy. An automated algorithm for extracting road edges from terrestrial mobile lidar data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 85:44–55, 2013.

[138] LaboraTe. Laboratorio do Territorio (LaboraTe). `http://laborate.usc.es/`. Accessed: 2018-07-30.

[139] Jet Propulsion Laboratory. Jpl institutional coding standard for the c programming language. `http://lars-lab.jpl.nasa.gov/JPL_Coding_Standard_ext.pdf`. Accessed: 2019-10-29.

[140] Andrew R. G. Large and George L. Heritage. *Laser Scanning – Evolution of the Discipline*, chapter 1, pages 1–20. John Wiley & Sons, Ltd, 2009.

[141] I Lee and T Schenk. Autonomous extraction of planar surfaces from airborne laser scanning data. In *Proc. ASPRS annual conference, St. Louis, MO, USA*, 2001.

[142] Lay Khoon Lee, Siau Chuin Liew, and Weng Jie Thong. A review of image segmentation methodologies in medical image. In Hamzah Asyrani Sulaiman, Mohd Azlishah Othman, Mohd Fairuz Iskandar Othman, Yahaya Abd Rahim, and Naim Che Pee, editors, *Advanced Computer and Communication Engineering Technology*, pages 1069–1080, Cham, 2015. Springer International Publishing.

[143] Nan Li, Norbert Pfeifer, and Chun Liu. Tensor-based sparse representation classification for urban airborne lidar points. *Remote Sensing*, 9(12), 2017.

[144] Y Li, X Hu, H Guan, and P Liu. An efficient method for automatic road extraction based on multiple feature from LiDAR data. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 41, 2016.

[145] Yong Li, Bin Yong, Huayi Wu, Ru An, and Hanwei Xu. Road detection from airborne lidar point clouds adaptive for variability of intensity data. *Optik*, 126(23):4292 – 4298, 2015.

[146] XG Lin, JX Zhang, ZJ Liu, and J Shen. Semi-automatic extraction of ribbon roads form high resolution remotely sensed imagery by cooperation between angular texture signature and template matching. *Int. Arch. Photogramm. Remote Sens*, 34:309–312, 2008.

[147] Xiangguo Lin, Jing Shen, and Yong Liang. Semi-automatic road tracking using parallel angular texture signature. *Intelligent Automation & Soft Computing*, 18(8):1009–1021, 2012.

[148] J. Lintz and D.S. Simonett. *Remote sensing of environment*. Addison-Wesley Pub. Co., Advanced Book Program, 1976.

[149] S. K. Lodha, D. M. Fitzpatrick, and D. P. Helmbold. Aerial lidar data classification using adaboost. In *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007)*, pages 435–442, Aug 2007.

[150] Suresh K. Lodha, Darren M. Fitzpatrick, David P. Helmbold, David P. Helmbold, and David P. Helmbold. Aerial lidar data classification using adaboost. In *Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling*, 3DIM '07, pages 435–442, Washington, DC, USA, 2007. IEEE Computer Society.

[151] Bharat Lohani and RK Mishra. Generating lidar data in laboratory: Lidar simulator. *Int. Arch. Photogramm. Remote Sens*, 52:12–14, 2007.

[152] Oscar G. Lorenzo, Jorge Martínez, David L. Vilariño, Tomás F. Pena, José C. Cabaleiro, and Francisco F. Rivera. Landing sites detection using LiDAR data on many-core systems. *The Journal of Supercomputing*, 73(1):557–575, Jan 2017.

[153] JL Lovell, DLB Jupp, GJ Newnham, NC Coops, and DS Culvenor. Simulation study for finding optimal lidar acquisition parameters for forest height retrieval. *Forest Ecology and Management*, 214(1-3):398–412, 2005.

[154] Ruijin Ma. Dem generation and building detection from lidar data. *Photogrammetric Engineering & Remote Sensing*, 71(7):847–854, 2005.

[155] Wei Ma and Qingquan Li. An improved ball pivot algorithm-based ground filtering mechanism for lidar data. *Remote Sensing*, 11(10), 2019.

[156] Hans-Gerd Maas. Methods for measuring height and planimetry discrepancies in airborne laserscanner data. *Photogrammetric Engineering and Remote Sensing*, 68(9):933–940, 2002.

[157] T. H. Maiman. Stimulated Optical Radiation in Ruby. *Nature*, 187:493–494, August 1960.

[158] Clément Mallet, Frederic Bretar, and Uwe Soergel. Analysis of full-waveform lidar data for classification of urban areas. *Photogrammetrie Fernerkundung Geoinformation*, 5:337–349, 2008.

[159] Clément Mallet and Frédéric Bretar. Full-waveform topographic lidar: State-of-the-art. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(1):1 – 16, 2009.

[160] E. Maltezos, A. Doulamis, N. Doulamis, and C. Ioannidis. Building extraction from lidar data applying deep convolutional neural networks. *IEEE Geoscience and Remote Sensing Letters*, 16(1):155–159, 2019.

[161] G. Mandlburger, H. Lehner, and N. Pfeifer. A comparison of single photon and full waveform lidar. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5:397–404, 2019.

[162] Jorge Martínez, Francisco F Rivera, José C Cabaleiro, David L Vilariño, Tomás F Pena, and David Miranda. A rule-based classification from a region-growing segmentation of airborne LiDAR. In *Image and Signal Processing for Remote Sensing XXII*, volume 10004, page 100040F. International Society for Optics and Photonics, 2016.

[163] J. Martínez, O. G. Lorenzo, D. L. Vilariño, T. F. Pena, J. C. Cabaleiro, and F. F. Rivera. A developer-friendly "open lidar visualiser and analyser" for point clouds with 3d stereoscopic view. *IEEE Access*, 6:63813–63822, 2018.

[164] Ali Akbar Matkan, Mohammad Hajeb, and Saeed Sadeghian. Road extraction from lidar data using support vector machine classification. *Photogrammetric Engineering & Remote Sensing*, 80(5):409–422, 2014.

[165] Helmut Mayer, Stefan Hinz, Uwe Bacher, and Emmanuel Baltsavias. A test of automatic road extraction approaches. *International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences*, 36(3):209–214, 2006.

[166] Andreas Mayr, Martin Rutzinger, Magnus Bremer, Sander Oude Elberink, Felix Stumpf, and Clemens Geitner. Object-based classification of terrestrial laser scanning point clouds for landslide monitoring. *The Photogrammetric Record*, 32(160):377–397, 2017.

[167] Donald Meagher. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129–147, 1982.

[168] Xuelian Meng, Nate Currit, and Kaiguang Zhao. Ground filtering algorithms for airborne LiDAR data: A review of critical issues. *Remote Sensing*, 2(3):833–860, 2010.

[169] Xuelian Meng, Le Wang, José Luis Silván-Cárdenas, and Nate Currit. A multi-directional ground filtering algorithm for airborne lidar. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(1):117 – 124, 2009.

[170] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Apache Maven*. Alpha Press, 2010.

[171] S Moafipoor, DA Grejner-Brzezinska, and CK Toth. Tightly coupled gps/ins/ccd integration based on gps carrier phase velocity update. In *Proceedings of the 2004 National Technical Meeting of The Institute of Navigation (ION), San Diego, CA, January*, pages 1094–1102, 2004.

[172] Domen Mongus, Niko Lukač, and Borut Žalik. Ground and building extraction from lidar data based on differential morphological profiles and locally fitted surfaces. *ISPRS Journal of Photogrammetry and Remote Sensing*, 93:145–156, 2014.

[173] Domen Mongus and Borut Žalik. Parameter-free ground filtering of lidar data for automatic dtm generation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67:1 – 12, 2012.

[174] Michel Morgan and Klaus Tempfli. Automatic building extraction from airborne laser scanning data. *International Archives of Photogrammetry and Remote Sensing*, 33(B3/2; PART 3):616–623, 2000.

[175] K. W. Morin. Calibration of airborne laser scanners. *Masters Thesis, The University of Calgary, UCGE Report #20179*, 2002.

[176] Abu Saleh Mohammad Mosa, Bianca Schön, Michela Bertolotto, and Debra F Laefer. Evaluating the benefits of octree-based indexing for LiDAR data. *Photogrammetric Engineering & Remote Sensing*, 78(9):927–934, 2012.

[177] Marius Muja and David G Lowe. FLANN, fast library for approximate nearest neighbors. In *International Conference on Computer Vision Theory and Applications (VISAPP 09)*, volume 3. INSTICC Press, 2009.

[178] National Oceanic & Atmospheric Administration (NOAA). Attachment Y. Light Detection and Ranging (LIDAR) Requirements. `https://www.ngs.noaa.gov/RSD/LIDAR_SOW_NGSDec2009.pdf`, 2000 (Accessed: 05/11/2019).

[179] Todd Neff. *The Laser That's Changing the World: The Amazing Stories Behind Lidar, from 3D Mapping to Self-Driving Cars*. Prometheus Books, 2018.

[180] Ross Nelson. How did we get here? an early history of forestry lidar1. *Canadian Journal of Remote Sensing*, 39(sup1):S6–S17, 2013.

[181] NetBeans. NetBeans Integrated Development Environment. https://netbeans.org/. Accessed: 2018-05-25.

[182] Gurdeep Singh Nick Diakopoulous, Mythili Bagavandas. Interactive: The top programming languages 2019. https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2019. Accessed: 2019-11-08.

[183] J. Niemeyer, F. Rottensteiner, U. Soergel, and C. Heipke. Contextual classification of point clouds using a two-stage crf. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-3/W2:141–148, 2015.

[184] Joachim Niemeyer, Franz Rottensteiner, and Uwe Soergel. Contextual classification of lidar data and building object detection in urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 87:152 – 165, 2014.

[185] Joachim Niemeyer, Franz Rottensteiner, Uwe Sörgel, and Christian Heipke. Hierarchical higher order crf for the classification of airborne lidar point clouds in urban areas. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives 41 (2016)*, 41:655–662, 2016.

[186] Joachim Niemeyer, Jan Dirk Wegner, Clément Mallet, Franz Rottensteiner, and Uwe Soergel. Conditional random fields for urban scene classification with full waveform lidar data. In Uwe Stilla, Franz Rottensteiner, Helmut Mayer, Boris Jutzi, and Matthias Butenuth, editors, *Photogrammetric Image Analysis*, pages 233–244, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[187] Stephen O'Grady. The RedMonk programming language rankings: January 2018. http://redmonk.com/sogrady/2018/03/07/language-rankings-1-18/. Accessed: 2018-05-25.

[188] OpenMP Architecture Review Board. The OpenMP API specification for parallel programming. http://openmp.org, 2019 (Accessed: 10/06/2019).

[189] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017 (Accessed: 03/05/2019).

[190] OpenTopography. Opentopography high-resolution topography data and tools. https://opentopography.org/. Accessed: 2019-03-19.

[191] Jens Overby, Lars Bodum, Erik Kjems, and PM Iisoe. Automatic 3d building reconstruction from airborne laser scanning and cadastral data using hough transform. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34(Part B3):296–301, 2004.

[192] Nikhil R Pal and Sankar K Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277 – 1294, 1993.

[193] Jihye Park, Impyeong Lee, Yunsoo Choi, et al. Automatic extraction of large complex buildings using lidar data and digital maps. In *Remote Sensing and Spatial Information Sciences, ISPRS*. Citeseer, 2006.

[194] M. Pauly, M. Gross, and L. P. Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization, 2002. VIS 2002.*, pages 163–170, Oct 2002.

[195] Thomas J. Pingel, Keith C. Clarke, and William A. McBride. An improved simple morphological filter for the terrain classification of airborne lidar data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 77:21 – 30, 2013.

[196] Shi Pu, George Vosselman, et al. Automatic extraction of building features from terrestrial laser scanning. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5):25–27, 2006.

[197] Lindi J Quackenbush, Iungho Im, and Yue Zuo. Road extraction: A review of LiDAR-focused studies. *Remote Sensing of Natural Resources*, pages 155–169, 2013.

[198] T. Rabbani, F.A. van den Heuvel, and G. Vosselman. Segmentation of point clouds using smoothness constraints. In H.G. Maas and D. Schneider, editors, *ISPRS 2006 : Proceedings of the ISPRS commission V symposium Vol. 35, part 6 : image engineering and vision metrology, Dresden, Germany 25-27 September 2006*, volume 35, pages 248–253. International Society for Photogrammetry and Remote Sensing (ISPRS), 2006.

[199] Matthias Rentsch and Peter Krzystek. Precise quality control of lidar strips. In *Proceedings ASPRS 2009 Annual Conference*, pages 9–13, 2009.

[200] Stephen E Reutebuch, Hans-Erik Andersen, and Robert J McGaughey. Light detection and ranging (lidar): an emerging tool for multiple resource inventory. *Journal of Forestry*, 103(6):286–292, 2005.

[201] RIEGL Laser Measurement Systems GmbH. Riegl minivux-1uav infosheet. `http://www.riegl.com/uploads/tx_pxpriegldownloads/RIEGL_miniVUX-1UAV_Infosheet_2018-10-01.pdf`. Accessed: 2019-06-03.

[202] James Ring. The laser in astronomy. *New Scientist*, 18(344):672–673, 1963.

[203] J. Rogers. Truro and Provincetown, MA: LiDAR in Salt March Environments airborne lidar. National Center for Airborne Laser Mapping (NCALM), distributed by OpenTopography. `https://doi.org/10.5069/G9B56GNB`, 2010.

[204] Marco Roggero. Object segmentation with region growing and principal component analysis. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/A):289–294, 2002.

[205] Franz Rottensteiner and Christian Briese. A new method for building extraction in urban areas from high-resolution lidar data. In *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, volume 34, pages 295–301. Natural Resources Canada, 2002.

[206] Esteban Rozas, Francisco F. Rivera, José C. Cabaleiro, Tomás F. Pena, and David L. Vilariño. Comparative study of building footprint estimation methods from LiDAR point clouds. In *Image and Signal Processing for Remote Sensing XXIII*, 2017.

[207] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[208] M. Rutzinger, B. Höfle, and N. Pfeifer. Object detection in airborne laser scanning data: an integrative approach on object - based image and point cloud analysis. In T. Blaschke, S. Lang, and G.J. Hay, editors, *Object based image analysis : spatial concepts for knowledge driven remote sensing applications (Lecture Notes in Geoinformation and Cartography)*, pages 645–662. Springer, 2008.

[209] Martin Rutzinger, Bernhard Höfle, and Norbert Pfeifer. Detection of high urban vegetation with airborne laser scanning data. *ForestSAT07*, 2007.

[210] Martin Rutzinger, Markus Maukisch, Frederic Petrini-Monteferri, and Johann Stötter. Development of algorithms for the extraction of linear patterns from airborne laser scanning data. In *In: Kellerer-Pirklbauer, A., Keiler, M., Embleton-Hamann, C. and Stötter, J. (eds), Geomorphology for the Future - Conference Proceedings. Innsbruck University Press, Innsbruck*, pages 161–168, 2007.

[211] F. Samadzadegan, M. Hahn, and B. Bigdeli. Automatic road extraction from lidar data based on classifier fusion. In *2009 Joint Urban Remote Sensing Event*, pages 1–6, May 2009.

[212] Farhad Samadzadegan, Behnaz Bigdeli, and Pouria Ramzi. A multiple classifier system for classification of lidar remote sensing data using multi-class svm. In *Proceedings of the 9th International Conference on Multiple Classifier Systems*, MCS'10, pages 254–263, Berlin, Heidelberg, 2010. Springer-Verlag.

[213] Aparajithan Sampath and Jie Shan. Segmentation and reconstruction of polyhedral building roofs from aerial lidar point clouds. *IEEE Transactions on geoscience and remote sensing*, 48(3):1554–1567, 2010.

[214] Philipp Schaer, Jan Skaloud, S. Landtwing, and Klaus Legat. Accuracy estimation for laser point cloud including scanning geometry. *Mobile Mapping Symposium 2007, Padova*, 2007.

[215] Jie Shan and Sampath Aparajithan. Urban dem generation from raw lidar data. *Photogrammetric Engineering & Remote Sensing*, 71(2):217–226, 2005.

[216] Jie Shan and Charles K Toth. *Topographic laser ranging and scanning: principles and processing*. CRC press, 2018.

[217] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524, 1968.

[218] George Sithole and George Vosselman. Automatic structure detection in a point-cloud of an urban landscape. In *Remote Sensing and Data Fusion over Urban Areas, 2003. 2nd GRSS/ISPRS Joint Workshop on*, pages 67–71. IEEE, 2003.

[219] George Sithole and George Vosselman. Experimental comparison of filter algorithms for bare-earth extraction from airborne laser scanning point clouds. *ISPRS journal of photogrammetry and remote sensing*, 59(1-2):85–101, 2004.

[220] Jeong-Heon Song, Soo-Hee Han, KY Yu, and Yong-Il Kim. Assessing the possibility of land-cover classification using LiDAR intensity data. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/B):259–262, 2002.

[221] Spectral Sciences Inc. MODTRAN: MODerate resolution atmospheric TRANsmission. http://modtran.spectral.com/, 2019 (Accessed: 04/11/2019).

[222] Jason M Stoker, Qassim A Abdullah, Amar Nayegandhi, and Jayna Winehouse. Evaluation of single photon and geiger mode lidar for the 3d elevation program. *Remote Sensing*, 8(9):767, 2016.

[223] Supervisely. Releasing first online 3D Point Cloud labeling tool in Supervisely. https://medium.com/deep-systems/releasing-first-online-3d-point-cloud-labeling-tool-in-supervisely 2019 (Accessed: 19/06/2020).

[224] Tasman District Council. Golden Bay, Tasman, New Zealand airborne lidar. AAM New Zealand Limited, distributed by OpenTopography. https://doi.org/10.5069/G9N877WS, 2017.

[225] Alexandru Telea, Cristian Sminchisescu, and Sven Dickinson. Optimal inference for hierarchical skeleton abstraction. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 4, pages 19–22. IEEE, 2004.

[226] TerraScan. Terrascan – software for lidar data processing and 3d vector data creation. http://www.terrasolid.com/products/terrascanpage.php. Accessed: 2020-06-19.

[227] The American Society for Photogrammetry & Remote Sensing (ASPRS). LAS Specification 1.4 - R15. http://www.asprs.org/wp-content/uploads/2019/07/LAS_1_4_r15.pdf, 2019 (Accessed: 29/10/2019).

[228] TIOBE. TIOBE index for may 2018. https://www.tiobe.com/tiobe-index/. Accessed: 2018-05-25.

[229] Tobias Knerr. OSM2World. http://osm2world.org/, 2011 (Accessed: 04/05/2019).

[230] Ivan Tomljenovic, Bernhard Höfle, Dirk Tiede, and Thomas Blaschke. Building extraction from airborne laser scanning data: An analysis of the state of the art. *Remote Sensing*, 7(4):3826–3862, 2015.

[231] D. Tóvári and N. Pfeifer. Segmentation based robust interpolation - a new approach to laser data filering. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Proceedings of ISPRS Workhop "Laser Scanning 2005", Enschede, The Netherlands 2005. Vol. 36-3/W19*, pages –, 2005.

[232] UDEM. Victor Harbour South Australia airborne lidar. The Coastal Urban Digital Elevation Modelling in High Priority Regions (UDEM). Cooperative Research Centre for Spatial Information (CRCSI), distributed by OpenTopography. https://doi.org/10.5069/G9251G47, 2011.

[233] A Ullrich and M Pfennigbauer. Linear lidar versus geiger-mode lidar: impact on data properties and data quality. In *Laser Radar Technology and Applications XXI*, volume 9832, page 983204. International Society for Optics and Photonics, 2016.

[234] S. Upadhayay, M. Yadav, and D. P. Singh. Road network mapping using airborne LiDAR data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-5:707–711, 2018.

[235] Nina Varney, Vijayan K Asari, and Quinn Graehling. Dales: A large-scale aerial lidar data set for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 186–187, 2020.

[236] Velodyne Lidar, Inc. Velodyne puck (vlp-16). https://www.goetting-agv.com/dateien/downloads/63-9229_Rev-H_Puck%20_Datasheet_Web.pdf. Accessed: 2019-06-03.

[237] G. Vosselman, B.G.H. Gorte, G. Sithole, and T. Rabbani. Recognising structure in laser scanning point clouds. In M. Thies, B. Koch, H. Spiecker, and H. Weinacher, editors,

ISPRS 2004 : proceedings of the ISPRS working group VIII/2 : laser scanning for forest and landscape assessment, Freiburg, October 3-6, 2004, pages 33–38. University of Freiburg, 2004.

[238] George Vosselman. Building reconstruction using planar faces in very high density height data. *International Archives of Photogrammetry and Remote Sensing*, 32(3; SECT 2W5):87–94, 1999.

[239] George Vosselman, Maximilian Coenen, and Franz Rottensteiner. Contextual segment-based classification of airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 128:354 – 371, 2017.

[240] George Vosselman and Hans-Gerd Maas. *Airborne and terrestrial laser scanning*. Whittles Publishing: Dunbeath, Scotland, 2010.

[241] Vozpópuli. Golpe al fraude en el ibi: Hacienda destapa más de tres millones de obras y piscinas no declaradas. `https://www.vozpopuli.com/economia-y-finanzas/operativo-Catastro-Golpe-fraude-IBI-Hacienda-millones-obras-piscin` `0_1207979276.html`. Accessed: 2020-05-19.

[242] W. Wagner, A. Ulrich, Thomas Melzer, Christian Briese, and K. Kraus. From single-pulse to full-waveform airborne laser scanners: Potential and practical challenges. *Int. Archives Photogrammetry, Remote Sensing, Geoinformation Sci.*, 35:414–419, 01 2004.

[243] Chisheng Wang, Qiqi Shu, Xinyu Wang, Bo Guo, Peng Liu, and Qingquan Li. A random forest classifier based on pixel comparison features for urban lidar data. *ISPRS journal of photogrammetry and remote sensing*, 148:75–86, 2019.

[244] G. Wang, Y. Zhang, J. Li, and P. Song. 3d road information extraction from lidar data fused with aerial-images. In *Proceedings 2011 IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services*, pages 362–366, June 2011.

[245] Ruisheng Wang. 3d building modeling using images and lidar: a review. *International Journal of Image and Data Fusion*, 4(4):273–292, 2013.

[246] Weixing Wang, Nan Yang, Yi Zhang, Fengping Wang, Ting Cao, and Patrik Eklund. A review of road extraction from remote sensing images. *Journal of Traffic and Transportation Engineering (English Edition)*, 3(3):271 – 282, 2016.

[247] Yan Wang, Donghui Xie, Guangjian Yan, Wuming Zhang, and Xihan Mu. Analysis on the inversion accuracy of lai based on simulated point clouds of terrestrial lidar of tree by ray tracing algorithm. In *2013 IEEE International Geoscience and Remote Sensing Symposium-IGARSS*, pages 532–535. IEEE, 2013.

[248] Yuan Wang, Siying Chen, Yinchao Zhang, He Chen, Pan Guo, and Jian Yang. Automatic road extraction for airborne LiDAR data. In *International Symposium on Photoelectronic Detection and Imaging 2013: Laser Sensing and Imaging and Applications*, volume 8905, page 890528. International Society for Optics and Photonics, 2013.

[249] Shen Wei. Building boundary extraction based on LiDAR point clouds data. *Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37:157–161, 2008.

[250] Martin Weinmann, Boris Jutzi, and Clément Mallet. Feature relevance assessment for the semantic interpretation of 3D point cloud data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 5:W2, 2013.

[251] Congcong Wen, Lina Yang, Xiang Li, Ling Peng, and Tianhe Chi. Directionally constrained fully convolutional neural network for airborne lidar point cloud classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 162:50–62, 2020.

[252] Timothy G. Whiteside, Guy S. Boggs, and Stefan W. Maier. Comparing object-based and pixel-based classifications for mapping savannas. *International Journal of Applied Earth Observation and Geoinformation*, 13(6):884 – 893, 2011.

[253] George M Williams Jr. Limitations of geiger-mode arrays for flash ladar applications. In *Laser Radar Technology and Applications XV*, volume 7684, page 768414. International Society for Optics and Photonics, 2010.

[254] John O Woods and John A Christian. Glidar: an opengl-based, real-time, and open source 3d sensor simulator for testing computer vision algorithms. *Journal of Imaging*, 2(1):5, 2016.

[255] C. Wu, Y. Wu, C. Wu, and C. Lin. A gradient vector flow snake based multi-level morphological active contour algorithm. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 5806–5809, 2017.

[256] Teng Wu, Xiangyun Hu, and Lizhi Ye. Fast and accurate plane segmentation of airborne lidar point cloud using cross-line elements. *Remote Sensing*, 8(5):383, 2016.

[257] B. Xiang, J. Yao, X. Lu, L. Li, and R. Xie. Segmentation-based classification for 3d urban point clouds. In *2016 IEEE International Conference on Information and Automation (ICIA)*, pages 172–177, Aug 2016.

[258] Xilinx. Xilinx Zynq-7000 SoC Technical Reference Manual. UG585 (v1.12.2) July 1, 2018. `https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf`, 2018 (Accessed: 10/06/2019).

[259] Xilinx. UG1267 - ZCU104 Board User Guide (v1.1) . `https://www.xilinx.com/support/documentation/boards_and_kits/zcu104/ug1267-zcu104-eval-bd.pdf`, 2018 (Accessed: 24/06/2019).

[260] Chenyang Xu and Jerry L Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on image processing*, 7(3):359–369, 1998.

[261] Jianhua Yan, Keqi Zhang, Chengcui Zhang, Shu-Ching Chen, and Giri Narasimhan. Automatic construction of 3-d building model from airborne lidar data through 2-d snake algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, 53(1):3–14, 2014.

[262] W. Y. Yan and A. Shaker. Radiometric correction and normalization of airborne lidar intensity data for improving land-cover classification. *IEEE Transactions on Geoscience and Remote Sensing*, 52(12):7658–7673, Dec 2014.

[263] Wai Yeung Yan and Ahmed Shaker. Radiometric normalization of overlapping lidar intensity data for reduction of striping noise. *International Journal of Digital Earth*, 9(7):649–661, 2016.

[264] Wai Yeung Yan, Ahmed Shaker, and Nagwa El-Ashmawy. Urban land cover classification using airborne LiDAR data: A review. *Remote Sensing of Environment*, 158:295–310, 2015.

[265] Yetao Yang, Ke Wu, Yi Wang, Tao Chen, and Xiang Wang. Two-layered graph-cuts-based classification of lidar data in urban areas. *Sensors*, 19(21):4685, 2019.

[266] Zhishuang Yang, Wanshou Jiang, Bo Xu, Quansheng Zhu, San Jiang, and Wei Huang. A convolutional neural network-based 3d semantic labeling method for als point clouds. *Remote Sensing*, 9(9):936, 2017.

[267] Bailang Yu, Hongxing Liu, Jianping Wu, Yingjie Hu, and Li Zhang. Automated derivation of urban building density information using airborne lidar data and object-based method. *Landscape and Urban Planning*, 98(3-4):210–219, 2010.

[268] Chunsun Zhang, Yuxiang He, and Clive S Fraser. Spectral clustering of straight-line segments for roof plane extraction from airborne LiDAR point clouds. *IEEE Geoscience and Remote Sensing Letters*, 15(2):267–271, 2018.

[269] Hui Zhang, Jason E Fritts, and Sally A Goldman. Image segmentation evaluation: A survey of unsupervised methods. *computer vision and image understanding*, 110(2):260–280, 2008.

[270] Jixian Zhang and Xiangguo Lin. Advances in fusion of optical imagery and lidar point cloud applied to photogrammetry and remote sensing. *International Journal of Image and Data Fusion*, 8(1):1–31, 2017.

[271] Jixian Zhang, Xiangguo Lin, and Xiaogang Ning. SVM-based classification of segmented airborne LiDAR point clouds in urban areas. *Remote Sensing*, 5(8):3749–3775, 2013.

[272] Keqi Zhang, Jianhua Yan, and Shu-Ching Chen. Automatic construction of building footprints from airborne LiDAR data. *Geoscience and Remote Sensing, IEEE Transactions on*, 44(9):2523–2533, 2006.

[273] Qiaoping Zhang and Isabelle Couloigner. Benefit of the angular texture signature for the separation of parking lots and roads on high resolution multi-spectral imagery. *Pattern recognition letters*, 27(9):937–946, 2006.

[274] Wuming Zhang, Jianbo Qi, Peng Wan, Hongtao Wang, Donghui Xie, Xiaoyan Wang, and Guangjian Yan. An easy-to-use airborne lidar data filtering method based on cloth simulation. *Remote Sensing*, 8(6), 2016.

[275] Y. J. Zhang, X. D. Xiong, and X. Y. Hu. Rigorous lidar strip adjustment with triangulated aerial imagery. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-5/W2:361–366, 2013.

[276] Zhiqiang Zhang, Xinchang Zhang, Ying Sun, and Pengcheng Zhang. Road centerline extraction from very-high-resolution aerial image and lidar data based on road connectivity. *Remote Sensing*, 10(8):1284, 2018.

[277] et al Zhang Hong. GIS algorithm elements, 2006.

[278] Jiaping Zhao and Suya You. Road network extraction from airborne lidar data using scene context. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–16. IEEE, 2012.

[279] Jiaping Zhao, Suya You, and Jing Huang. Rapid extraction and updating of road network from airborne LiDAR data. In *2011 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–7, Oct 2011.

[280] Ruibin Zhao, Mingyong Pang, and Jidong Wang. Classifying airborne lidar point clouds via deep features learned by a multi-scale convolutional neural network. *International Journal of Geographical Information Science*, 32(5):960–979, 2018.

[281] Zhen Zhen, Lindi J. Quackenbush, and Lianjun Zhang. Trends in automatic individual tree crown detection and delineation—evolution of lidar data. *Remote Sensing*, 8(4), 2016.

[282] Qian-Yi Zhou and Ulrich Neumann. Fast and extensible building modeling from airborne lidar data. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, pages 1–8, 2008.

[283] P. Zingaretti, E. Frontoni, G. Forlani, and C. Nardinocchi. Automatic extraction of lidar data classification rules. In *14th International Conference on Image Analysis and Processing (ICIAP 2007)*, pages 273–278, Sep. 2007.

[284] Iman Zolanvari, Susana Ruano, Aakanksha Rana, Alan Cummins, Aljosa Smolic, Rogerio Da Silva, and Morteza Rahbar. Dublincity: Annotated lidar point cloud and its applications. In *30th British Machine Vision Conference (BMVC)*, 09 2019.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

List of Figures

Fig. 1.1 Basic operation of the laser rangefinder using the TOF method [216]. 22
Fig. 1.2 Comparison between the two signals using the Phase-based method [216]. . 23
Fig. 1.3 Main components of an ALS system [104]. 24
Fig. 1.4 Deflection devices and resulting scanning patterns [240]. 25
Fig. 1.5 Workflow of data integration. 26
Fig. 1.6 Reference frames of the ALS system [214]. 28
Fig. 1.7 ALS Flight Planning. 33
Fig. 1.8 Profiles along the X direction over a building covered by strips 1 (in blue), 2 (in red), 3 (in green), and 4 (in magenta) (a) Before and (b) after the geometric adjustment [94]. 34
Fig. 1.9 Averaged intensity rasters (gray-scaled), empty cells filled with a median filter; intensity a) before and b) after a data-driven radiometric normalisation [103]. 35
Fig. 1.10 Illustration of the laser pulse return signal and the recorded signal strength [264]. 36
Fig. 1.11 Recursive partition into octants. Left: 3D scheme. Right: 2D scheme [78]. . 38
Fig. 1.12 Example of multi-threading: Fork join model. Image from [28]. 40

Fig. 2.1 Neighbourhood models. From left to right: k-nearest, sphere, and cylinder. . 46
Fig. 2.2 Segmentation overview. 48
Fig. 2.3 Normal vectors: (a) before and (b) after orientation correction. The wrongly oriented normal vectors (in red) are corrected using an artificial viewpoint (in blue). 49
Fig. 2.4 Results of (a) the first region-growing procedure, (b) the second region-growing procedure, and (c) their combination. Regions are drawn in different colours. 50
Fig. 2.5 Visualisation of the concept of epicentre: region's points (in red), region's centroid (in big red), epicentre (in blue), last added point (in purple), candidate point (in black). 51
Fig. 2.6 ISPRS dataset coloured by (a) height, (b) intensity, and (c) classification labels. 52
Fig. 2.7 Label adaptation: Original ground-truth labels (left) and adapted labels (right). 56
Fig. 2.8 Influence of the epicentre on quality. 58

Fig. 3.1 Overview of the classification strategy. 62
Fig. 3.2 Alpha Shapes algorithm visualisation [249]. 64
Fig. 3.3 Alpha shapes algorithm judgement [249]. 65
Fig. 3.4 Extracted boundaries for (a) single plane building, (b) a multi-plane building, and (c) trees. Points coloured by segment. Boundary points are highlighted with a bigger size. 65
Fig. 3.5 Height Jump Detection (HJD) for a roof segment: (a) visualisation of the neighbour query for one boundary point and (b) resultant jump labels for every boundary point. Points coloured by region. Boundary points are highlighted in a bigger size. The limits of the vertical cylinder are shown only for visualisation purposes as it has infinite height. 66
Fig. 3.6 Rule-based classification flow. 67
Fig. 3.7 Alcoy: Classification results (left) and buildings ground truth (right.) 69
Fig. 3.8 Vaihingen: Classification results (left) and buildings ground truth (right.) . . 70
Fig. 3.9 Histogram of cluster sizes. For visualisation purposes, only clusters with size in the [0,1000] range are shown. Clusters sizes in this example range from 1 to nearly 33,000. 72

Fig. 4.1 Overview of the building extraction. 81

Fig. 4.2 Visualisation of opening. The opening is used to remove all of the vegetation while preserving the structure of the included building as well as the sloping ground surface on the right. A ten meter linear structuring element was used. Image from [195]. 82

Fig. 4.3 Echo ratio visualisation: (a) Basic and (b) Slope adapted versions [102]. . . 85

Fig. 4.4 Erosion for an area of the Alcoy site: Candidate road points (a) before and after the (b) first, (c) second, and (d) third iteration of the erosion. Point coloured by intensity. 86

Fig. 4.5 Clustering results for an area of the Alcoy site (see Section 4.2.2 for details about the data). Clusters are shown in different colours. 88

Fig. 4.6 Vaihingen ground truth [251] 89

Fig. 4.7 Vaihingen results: T_p in green, F_p in blue, and F_n in red. 91

Fig. 4.8 92

Fig. 4.9 93

Fig. 4.10 Dimensionality of the points for an area of the Vaihingen site: (a) Top view and (b) profile view. Points are coloured by dimensionality: 0 in white (isolated points), 1 in blue, 2 in red, and 3 in green. 96

Fig. 5.1 Flowchart of the pipeline filtering. 101

Fig. 5.2 Flowchart of the bidirectional skewness balancing. 103

Fig. 5.3 Normalised histogram (HST) of Truro data (see Section 5.2.1) (a) before and (b) after removing the outliers. 104

Fig. 5.3 Effects for the Carola site. The histogram is shown (a) after removing outliers, (b) after removing values lower than the threshold obtained by the forward skewness balancing without tail removal, (c) after removing the right tail, and (d) after removing values lower than the threshold obtained by the forward skewness balancing with tail removal. 106

Fig. 5.4 Results of the intensity filtering for Carola data (a) before and (b) after the removal of the right tail. Points are coloured by a cold-to-hot gradient of their intensity values. Some road segments enclosed in the highlighted white areas are recovered. 106

Fig. 5.5 Comparison of the intensity values distribution between (a) the urban site of Vaihingen and (b) the rural site of Alcoy. The dominant point class, road or non-road, determines if the distribution is left-concentrated or right-concentrated, respectively. 108
Fig. 5.6 Flowchart of the forward skewness balancing. 109
Fig. 5.7 Visualisation of a circular neighbourhood with radius r for different points in the road: (a) a point in the centerline, (b) a point in the road edge, and (c) a point in a sharp bend of 90 degrees. 111
Fig. 5.8 Level of urbanisation vs. skewness. The red line is the 1D polynomial fitted line. Note that the adjustment is far from perfect but it is just shown to help to visualise the trend. 113
Fig. 5.9 TIN computed with FugroViewer [77] of (a) Vaihingen and (b) Carola sites. Point of view over a road. Carola data show unusual high curvature. 115
Fig. 5.10 Aerial images of the tested data (left) and extracted road points (right). . . . 116
Fig. 5.10 Aerial images of the tested data (left) and extracted road points (right). . . . 117
Fig. 5.10 Aerial images of the tested data (left) and extracted road points (right). . . . 118
Fig. 5.10 Aerial images of the tested data (left) and extracted road points (right). . . . 119
Fig. 5.11 Intensity correction issues in Arzúa. The change in intensity is clearly visible at the overlap of the flight strips (highlighted with red arrows). The road segment inside this high-intensity strip cannot be identified by our method. . 120
Fig. 5.12 Planar (black) and non-planar (red) points for different curvature thresholds. The scene corresponds to an area of east Trabada with tilled areas with nearby road segments. 121
Fig. 5.13 Influence of MRW parameter in the quality of the results for Vaihingen and Truro study sites. 125

Fig. 6.1 Overview of the road modelling. 130
Fig. 6.2 ATS: (a) Set of rectangular regions centred for a given pixel and (b) the graph of its ATS value. The size of the rectangular regions is 5 by 20 pixels. Image from [273]. 131
Fig. 6.3 Adapted ATS: (a) Set of rectangles and (b) rectangle features. 132
Fig. 6.4 ATS road tracking: ATS rectangle set and extracted centreline points (in pink). 135
Fig. 6.5 Extracted centreline for a road segment. 136
Fig. 6.6 Extracted centreline for a road segment. 137

Fig. 6.7 Extracted centreline for a road segment. 138

Fig. 6.8 Visualisation of the finite central difference calculation. Note that for visualisation purposes, it has been assumed that the points inside the neighbourhood are regularly spaced. 142

Fig. 6.9 Application of the partial derivatives: (left) LiDAR data coloured by intensity and (right) magnitude of the intensity gradient coloured in a cold-to-heat heat map. It can be seen how the areas with the largest intensity gradient correspond to the surroundings of the road markings, where materials with very high intensity (painted asphalt) and very low intensity (raw asphalt) coincide. 143

Fig. 6.10 External forces: (a) synthetic data and (b) resulting external forces (white arrows). 144

Fig. 6.10 Snakes results for some road gaps in the Alcoy site. The LiDAR points classified as ground and road, coloured by intensity, are shown. The initial contours are shown in the left side (in red) and the final contours are shown on the right side (in green). 146

Fig. 6.11 Geometric derivation of the road width. Centreline points (in violet), road points (in black), and boundary points (in red). 147

Fig. 6.12 Width estimation for a road segment. 148

Fig. 6.13 Snake results varying the number of contours to: (a) 5 (b) 10 (c) 20 (d) 50 (e) 100 (f) 150. 150

Fig. 6.14 Snake results varying the maximum number of iterations to: (a) 1 (b) 2 (c) 3 (d) 5 (e) 10 (f) 50. 151

Fig. 7.1 Scanning mechanisms and their resulting scan-line ground patterns [9]: Scan lines are captured perpendicularly to the flight course. The (a) oscillating mirror and (b) rotating polygon mechanisms are commonly used in airborne scenarios. 155

Fig. 7.2 Space extend of the LAS data: A flight line (single-digit numbered) is formed by a set of consecutive scan lines. Odd- and even-numbered stripes are flown in opposite directions. The darker shading shows stripe overlap. Generally, LAS data is organised in tiles (two-digit-numbered rectangles) [79]. 156

Fig. 7.3 Flowchart of the iterative spline ground filtering. 157

Fig. 7.4 Visualisation of the right neighbour calculation for the point p_{63}. Green dots mean that the distance is lower than the previous one while red dots mean that this distance is larger. 158
Fig. 7.5 Comparison between Akima and cubic interpolations. 160
Fig. 7.6 Spline (orange) of a scan line after (a) the initial interpolation, (b) the push-down stage, and (c) the push-up stage. The knots of each stage are also shown: seed knots (green), pushdown added knots (yellow), and push-up added knots (blue, red and purple). The colours of the push-up added knots match the different knot addition processes shown in Figure 7.7: p_i (blue), p_{i-1} (red) and p_j (purple). 162
Fig. 7.7 Flowchart of the push-up stage. 163
Fig. 7.8 Error distribution in the (a) urban and (b) rural synthetic data and (c–e) sources of error. Legend: Type I error (red), Type II error (blue), ground points (light grey), and non-ground points (dark grey). 165
Fig. 7.9 Alcoy results. 166
Fig. 7.10 Trabada results. 167
Fig. 7.11 Vaihingen results. 167
Fig. 7.12 Visible errors highlighted in yellow. 169
Fig. 7.13 Sources of point accumulation: (**a**) oscillating mirror deceleration and (**b**) building facades. 171
Fig. 7.14 Hardware implementation in the Zynq-7000 platform. 172
Fig. 7.15 System architecture. 173
Fig. 7.16 Neighborhood processing unit (Neighbourhood Search Core (NCORE)) architecture. 174
Fig. 7.17 Stream Processing Unit (SPU) structure. 175
Fig. 7.18 Error distribution in the synthetic Heidelberg's urban site for different scan-line lengths. Legend: Type II error (blue), ground points (light grey), and non-ground points (dark grey). 178
Fig. 7.19 Kappa and total error in the Heidelberg's site for different scan-line lengths. 179
Fig. 7.20 Segments distribution (yellow) of a scan line for different scan-line lengths. Scan-line segments fully covered by the building are highlighted in purple. . 179

Fig. 8.1 Package structure of Olivia. 187

Fig. 8.2 Screen capture of a segmentation visualisation showing the panel composition: control panel (1), render panel (2), and visualisation panel (3). 189
Fig. 8.3 Demonstrating classification results using the 3D stereoscopic mode in the Airborne LiDAR Solutions stand. II Edition of Inside the Lab, a Technological Demonstration Day at CiTIUS. 191
Fig. 8.4 Octree nodes showed in green: All (left), only leafs (right). 192
Fig. 8.5 Roof point (in blue) and its neighbours (in green) for different methods: Triangular Irregular Network (left), sphere (center), and orientated cylinder (right). 193
Fig. 8.6 Normal vectors (white lines). Points coloured by cluster. 194
Fig. 8.7 Usage of both selection and camera mirroring. Comparing the same cluster changing the intensity threshold in the segmentation process. 195
Fig. 8.8 Computed hulls and areas using different algorithms; from left-to-right to top-to-bottom: convex, concave, exterior, voronoi. 196
Fig. 8.9 Result of the LSDA, the suitability for landing is represented in different colours. The approximation cone is shown for the selected point. 197
Fig. 8.10 A detailed view of the approximation cone: landing position (blue bars), valid approximation angles (in green) and invalid approximation angles (in red). 198
Fig. 8.11 Rendering speed of Olivia for different graphic devices. 199
Fig. 8.12 Memory usage of Olivia for host and device memory. 199

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

List of Tables

Tab. 1.1 LAS 1.4 Format definition. 29
Tab. 1.2 PDRF 6. The fields that are bit fields are encoded into two bytes. 30
Tab. 1.3 Classification Flags Bit Field Encoding (PDRF 6-10) 30
Tab. 1.4 ASPRS Standard Point Classes (PDRF 6-10) 31

Tab. 2.1 Segmentation parameters and their default values. 53
Tab. 2.2 Results of the unsupervised metrics. 55
Tab. 2.3 Label adaptation. 56
Tab. 2.4 Results of the classification-derived metrics. 57
Tab. 2.5 Normalised evaluation metrics and Segmentation Goodness. 58

Tab. 3.1 Classification methods 63
Tab. 3.2 Classification parameters and their default values. 68
Tab. 3.3 Computational stages and their execution times. 71
Tab. 3.4 Execution times and speedups of the paralellised stages. 73
Tab. 3.5 Segmentation GProf output. Routines for neighbour queries are highlighted. 74
Tab. 3.6 Classification GProf output. Routines for neighbour queries are highlighted. 75
Tab. 3.7 Performance comparison with other existing methods. 75

Tab. 3.8 Different methodologies to evaluate the classification accuracy. 77

Tab. 4.1 Parameters and their default values. 90
Tab. 4.2 Quality metrics of the quantitative evaluation. 90
Tab. 4.3 Computational stages and their execution times. 94
Tab. 4.4 Execution times and speedups of the paralellised stages. 95

Tab. 5.1 Point clouds included the dataset. 112
Tab. 5.2 Results of the quantitative evaluation. 114
Tab. 5.3 Execution times of the bidirectional skewness balancing with different step sizes. 121
Tab. 5.4 Computational stages and their execution times for each site. 122
Tab. 5.5 Computational stages and their overall execution times. 122
Tab. 5.6 Execution times and speedups of the paralellised stages. 123
Tab. 5.7 Skewness balancing insights and results of the road extraction. 124
Tab. 5.8 Quality metrics of road extraction compared to other methods. 126

Tab. 6.1 ATS tracking parameters. 135
Tab. 6.2 Snakes parameters. 144

Tab. 7.1 Errors and kappa coefficients for the synthetic samples. 165
Tab. 7.2 Point clouds for the qualitative evaluation. 166
Tab. 7.3 Execution times and throughput in the workstation with and without considering input and output (IO). 168
Tab. 7.4 Execution times and throughput in the Zedboard for the Alcoy, Trabada, and Vaihingen data with and without considering input and output (IO). 170
Tab. 7.5 List of field-programmable gate array (FPGA) resources used. 174
Tab. 7.6 Hardware acceleration for the Vaihingen data. 176
Tab. 7.7 Power dissipation of the Zedboard for the Vaihingen data. 176
Tab. 7.8 Mean total error and kappa coefficients compared to other methods. 181

Tab. 8.1 Popular open source visualisation tools and Olivia. 184