Segmentierung großer Punktwolken mittels Region Growing

MARKUS PÖCHTRAGER[1]

Zusammenfassung: Die stetig wachsenden Datensätze mit 3D-Punktwolken gewonnen durch terrestrisches Laserscanning und Airborne Laserscanning bieten einerseits vielfältige Möglichkeiten in der Datenanalyse und Datenprozessierung, erfordern andererseits jedoch eine hohe Effizienz der prozessierenden Algorithmen. Die Segmentierung von Punktwolken liefert eine Gruppierung von gleichartigen Punkten anhand eines Homogenitätskriteriums und bildet damit eine geeignete Grundlage für viele Prozessierungsketten. Dabei stehen alle Punkte über die Homogenität in Beziehung zueinander und müssen gemeinsam prozessiert werden. Um auch Datensätze behandeln zu können, die für eine Verarbeitung im Ganzen zu groß sind, wurde im Rahmen dieser Arbeit ein Konzept für eine Segmentierung von solchen großen Punktwolken vorgestellt, bei dem diese Kette zwischen den Punkten aufgebrochen wird. Für eine effiziente Prozessierung wird die Punktwolke zuerst in rechteckige Teilpunktwolken (Kacheln) aufgeteilt. Diese Kacheln werden unabhängig voneinander segmentiert und abschließend werden gleichartige Segmente aus benachbarten Kacheln zusammengeführt. Dabei wird auch die Punktwolke wieder vereint. Ziel dieses kachelbasierten Konzeptes ist auch, eine Möglichkeit der Parallelisierung zu schaffen. Wie diese Arbeit zeigt, bleiben die Ergebnisse von der Wahl der Kachelgröße unbeeinflusst und sind nur vom Homogenitätskriterium abhängig. Durch beliebige Wahl der Kachelgröße und parallele Prozessierung können Speicherbedarf und Laufzeit optimiert und die Anforderungen an den verarbeitenden Rechner reduziert werden.

1 Einleitung

Während Laserscanning zur Aufnahme von dreidimensionalen Punktwolken in verschiedensten Fachgebieten zunehmend Verbreitung findet, steigen die Anforderungen an Prozessierungs- und Auswertemethoden für die gewonnenen Punktwolken. Zum einen weil immer komplexere Prozessierungsketten entwickelt werden, zum anderen durch die immer größer werdenden Datenmengen. Für die automatisierte Prozessierung von großen Datenmengen werden Methoden, welche die Speicherung und Verwaltung von Punktwolken vereinfachen, benötigt. Dabei ist man unter anderem an Verfahren zur Datenkompression interessiert, um den Speicherbedarf zu reduzieren, aber auch an der Entwicklung von Speicherstrukturen, welche einen schnellen Zugriff auf die Punktinformationen ermöglichen. Es stellt sich hier die Frage, wie Strukturinformation in Punktwolken gefunden werden kann, um einen effizienteren Zugriff auf gleichartige Punkte zu ermöglichen. Wie VOSSELMAN et al. 2004 argumentiert, liefert die Segmentierung einen erheblichen Beitrag für diese Strukturfindung in Punktwolken. Bei der Segmentierung wird ein Datensatz auf Basis eines Homogenitätskriteriums in Segmente von Punkten mit gleicher Eigenschaft unterteilt. Das dieser Arbeit zugrunde liegende Segmentierungsverfahren *Seeded Region Growing* (ADAMS & BISCHOF 1994) ist Standard, aber durch den Arbeitsspeicherplatz beschränkt anwendbar. Für die Lösung dieses Problems wurde ein Konzept entwickelt, bei dem

[1] TU Wien, Department für Geodäsie und Geoinformation; Gußhausstraße 27-29, A-1040 Wien
E-Mail: markus.poechtrager@tuwien.ac.at

die Punktwolke in Teilpunktwolken (Kacheln) mit beliebiger Größe unterteilt werden kann. Die Segmentierung erfolgt dabei auf den Kacheln und es muss dadurch nie die gesamte Punktwolke im Arbeitsspeicher gehalten werden, sondern immer nur die gerade verarbeiteten Kacheln.

2 Konzept

Das Konzept dieser Arbeit basiert auf einem *Split-and-Merge* Ansatz. Dafür wird die Punktwolke im ersten Schritt in Kacheln unterteilt (*Split*). Alle Kacheln zusammen bilden eine lückenlose und überlappungsfreie Abdeckung der gesamten Punktwolke. Im nächsten Schritt wird auf jeder dieser Kacheln eine von anderen Kacheln unabhängige Segmentierung durchgeführt. Abschließend werden die Punkte aus den segmentierten Kacheln wieder zu einer Punktwolke vereint. Gleichartige Segmente – also Segmente, bei denen zumindest zwischen zwei Punkten das Homogenitätskriterium erfüllt ist – aus benachbarten Kacheln werden dabei zusammengeführt (*Merge*). Ein Ziel dieses Konzeptes muss sein, dass die Ergebnisse dieser *Split-and-Merge* Segmentierung den Ergebnissen einer Segmentierung auf der gesamten Punktwolke entsprechen (CHEN & PAVLIDIS 1990). Das Segmentierungsergebnis soll also von der gewählten Kachelgröße unabhängig sein.

2.1 Datenorganisation

Das Konzept wurde auf Basis der Software OPALS (PFEIFER et al. 2014) entwickelt, kann aber grundsätzlich auch auf anderen Datenstrukturen etabliert werden. Der verwendete *OPALS Data Manager* übernimmt die Aufteilung und persistente Speicherung der Punktwolke in Kacheln und verwaltet außerdem für jeden Punkt eine Liste mit Attributinformationen. Ist eine Kachel der Punktwolke im Arbeitsspeicher geladen, können diese Punktattribute – wie z.B. Intensität, Farb-Kanäle oder die Segment-ID – gelesen oder geschrieben werden. Beim Laden einer Kachel wird mit den darin liegenden Punkten *on-the-fly* ein *kd*-Baum im Arbeitsspeicher aufgebaut. Die Datenstruktur des *kd*-Baumes liefert einen schnellen Index für Punkte im *k*-dimensionalen Raum und ermöglicht eine effiziente Nachbarschaftssuche in der Punktwolke (BENTLEY 1975). Gerade bei der Segmentierung mit *Seeded Region Growing* spielt die Suche von benachbarten Punkten für einen Ausgangspunkt (Saatpunkt) eine entscheidende Rolle.

2.2 Segmentierung mit Seeded Region Growing

Die Segmentierung mit *Seeded Region Growing* teilt die Punktwolke von einem Saatpunkt ausgehend in disjunkte Segmente ($S_i \cap S_j = \emptyset$, *für* $i \neq j$). Die Vereinigung aller Segmente S_i ist dabei wieder die ursprüngliche Punktmenge. Für die Segmentierung auf den Kacheln werden zunächst alle Punkte einer Kachel geladen und mit ihnen der *kd*-Baum aufgebaut. Anschließend werden alle Punkte p_i mit $i = 1,\ldots,m$ in der Baumstruktur durchlaufen. Wurde ein Punkt p_i noch keinem Segment zugewiesen, wird ein neues Segment S_i mit p_i als Saatpunkt angelegt. Für jeden Saatpunkt p_i eines Segmentes wird eine Nachbarschaftssuche ausgeführt. Die Nachbarschaftssuche liefert eine Liste von Punkten p_n mit $n = 1,\ldots,m$ die ein symmetrisches Nachbarschaftskriterium *NH* erfüllen (z.B.: Punkte innerhalb einer Kugel mit Suchradius r). Für die Nachbarschaft *NH* zweier Punkte muss folgende Bedingung erfüllt sein:

$$NH(p_i, p_n) == NH(p_n, p_i)$$

Ist ein Punkt p_n ein Nachbarpunkt von p_i, so muss auch p_i ein Nachbarpunkt von p_n sein.
Erfüllt ein benachbarter Punkt p_n zusätzlich auch ein gewähltes Homogenitätskriterium P, wird dieser Punkt zum Segment S_i hinzugefügt und reiht sich außerdem in eine Liste zukünftiger Saatpunkte des Segmentes SQ_i ein. Das Homogenitätskriterium muss ebenfalls symmetrisch sein, damit dieses *Split-and-Merge* Konzept unabhängig von der Wahl der Kacheln immer die gleichen Ergebnisse liefert. Auch hier gilt also folgendes:

$$P(p_i, p_n) == P(p_n, p_i)$$

Erfüllt ein Punkt p_n das Homogenitätskriterium zu p_i, so muss auch p_i das gleiche Kriterium zu p_n erfüllen.
Nachdem alle Nachbarpunkte von p_i auf Homogenität überprüft und gegebenenfalls zum Segment hinzugefügt wurden, wird aus der Liste SQ_i der nächste Saatpunkt des Segmentes S_i gewählt. Das Segment wächst dadurch in alle Richtungen, bis keine neuen Punkte das Homogenitätskriterium erfüllen. Danach wird das Segment abgeschlossen und mit dem nächsten Punkt p_i aus dem *kd*-Baum, welcher noch nicht prozessiert wurde, ein neues Segment initialisiert. Die Segmentierung erfolgt auf jeder Kachel also sequentiell. Ein neues Segment wird erst dann angelegt, wenn das vorherige abgeschlossen ist.
Während bei *Seeded Region Growing* eine Parallelisierung auch über die Verwendung von *n* Saatpunkten, von denen ausgehend die Segmente parallel wachsen, denkbar wäre, wird sie in diesem Konzept über die unabhängige Segmentierung auf den *n* Kacheln erreicht.

2.3 Zusammenführung der Kacheln

Um das erwartete Segmentierungsergebnis auf der gesamten Punktwolke zu bekommen, müssen abschließend die gleichartigen Segmente aus den Kacheln vereint werden. Für eine effiziente Lösung dieser Aufgabenstellung wird eine Nachbarschaftsbeziehung zwischen den Kacheln aufgebaut. Dafür wird jede Kachel in neun Teilbereiche unterteilt (**Fehler! Verweisquelle konnte nicht gefunden werden.**). Neben dem inneren Bereich (0), in der jene Punkte liegen, die für die Zusammenführung keine Relevanz haben, gibt es acht Randbereiche (1)-(8). Die Größe der Randregionen ergibt sich aus dem Suchradius r der Nachbarschaftssuche in der Segmentierung. Alle Punkte, die bei der Segmentierung in zumindest einer Nachbarkachel eine Rolle spielen können, liegen dadurch in Randregionen.
Für jede Kachel (K0) werden außerdem die angrenzenden Kacheln ermittelt (K1-K8) und mit benachbarten Kachelregionen sogenannte *Merge*-Nachbarschaften gebildet. Im Wesentlichen werden 4er- (**Fehler! Verweisquelle konnte nicht gefunden werden.** – *rot*) und 6er-Nachbarschaften (**Fehler! Verweisquelle konnte nicht gefunden werden.** – *grün* und *blau*) unterschieden. Die beteiligten Randbereiche in einer *Merge*-Nachbarschaft können über die geschaffene Index-Struktur, bestehend aus Kachel-ID und Randbereichs-ID, schnell ermittelt werden:

z.B.: [K0, Region 1] → [[K8, Region 3], [K1, Region 5], [K2, Region 7]])

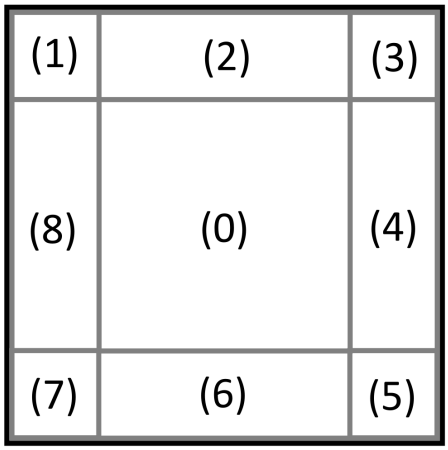

Abb. 1: Kachel-Regionen

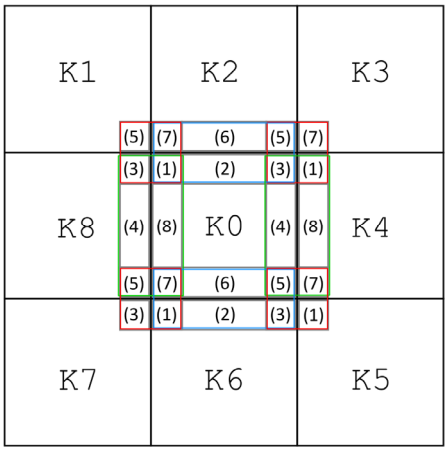

Abb. 2: Kachel-Nachbarschaft

Im tatsächlichen Zusammenführungsschritt werden dann alle Kacheln der Punktwolke durchlaufen. In jeder Kachel werden die Randbereiche (Kachel-Region 1-8) unabhängig voneinander - parallel oder sequentiell - geladen. Für jede Kachel-Region werden außerdem die relevanten Nachbarregionen geladen und mit allen Punkten dieser *Merge*-Nachbarschaft ein lokaler *kd*-Baum aufgebaut. Mit den Punkten im *kd*-Baum wird eine zweite, lokale Segmentierung durchgeführt. Dabei werden zwei Segmente zusammengeführt wenn ein Punkt p_i und sein benachbarter Punkt p_n innerhalb des Suchradius folgende Bedingungen erfüllen:

i. Der Punkt p_n ist noch nicht prozessiert
ii. Die Punkte p_i und p_n liegen nicht im gleichen Segment
iii. Die Segmente der Punkte p_i und p_n besitzen noch keine Zuordnung (Mapping) zueinander
iv. Das Homogenitätskriterium $P(p_i, p_n)$ wird erfüllt

Alle Segmente, die durch die Zusammenführung zu einem Segment vereint wurden, werden in einem Mapping-Eintrag zusammengefasst. Jeder Eintrag in der Mapping-Liste beinhaltet also zwei oder mehr Segment-IDs. Über diese Mapping-Liste können die endgültigen Segment-IDs für die Punkte bestimmt werden.

3 Ergebnisse

Die Tauglichkeit des Konzeptes wird im Rahmen der Arbeit mit unterschiedlichen Testdatensätzen geprüft. Wir erwarten dabei, dass die Ergebnisse von der Wahl der Kachelgröße unbeeinflusst bleiben. Zusätzlich wird der Einfluss der Kachelgröße auf die Laufzeit analysiert.

3.1 Segmentierungsergebnis

Der für den anschaulichen Vergleich der Segmentierungsergebnisse bei unterschiedlichen Kachelgrößen verwendete, synthetische Testdatensatz *Ring (100x100 m)* besteht im Wesentlichen aus drei Teilbereichen. Einem inneren kreisförmigen Bereich, einem in z-Achse um 1 m angehobenen Ring, sowie dem äußeren Randbereich. Eine Segmentierung mit dem einfachen Homogenitätskriterium

$$P(p_i, p_n) = |p_i.z - p_n.z| < 0.01$$

wird für verschiedene Kachelgrößen (10 m, 50 m, 100 m) durchgeführt und das Ergebnis zahlenmäßig (Tab. 1) und visuell (**Fehler! Verweisquelle konnte nicht gefunden werden.** und **Fehler! Verweisquelle konnte nicht gefunden werden.**) analysiert.
Die unterschiedlichen Farbwerte in der Punktwolke repräsentieren die den Punkten zugeordneten Segment-IDs. Alle drei Segmentierungsergebnisse zeigen die erwarteten, gleichen Segmente. Zur Veranschaulichung der unterschiedlichen Berechnungsvorgänge sind in **Fehler! Verweisquelle konnte nicht gefunden werden.** die Segmente vor der Zusammenführung zu sehen.

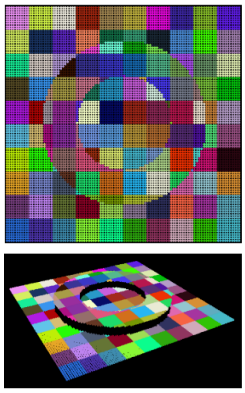

Abb. 3: Segmentierungsergebnisse vor der Zusammenführung; Kachelgrößen 10m, 50m; Grundriss und 3D

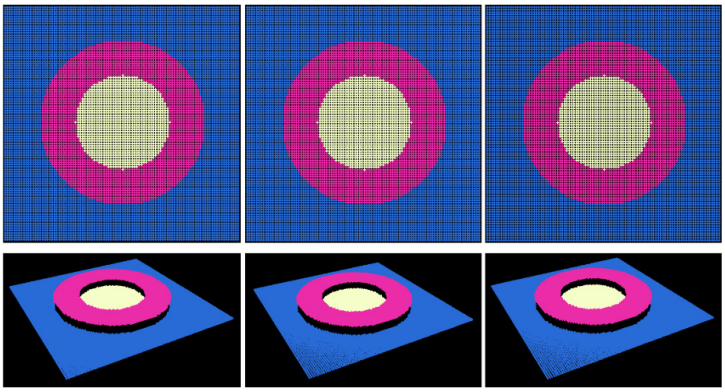

Abb. 4: Segmentierungsergebnisse nach der Zusammenführung; Kachelgrößen 10m, 50m und Segmentierung ohne Kachelung; Grundriss und 3D

Tab. 1: Einfluss der Kachelgröße auf Segmente – Datensatz Ring

Kachel-größe [m]	Anz. d. Kacheln	Segmente vor *Merge*	Segmente Final
10	100	140	3
50	4	12	3
100	1	-	3

3.2 Laufzeitanalyse

Unter der Annahme, dass keine signifikanten Laufzeitunterschiede zwischen der Segmentierung des gesamten Datensatzes und der verteilten Segmentierung auftreten, wurden auf einem herkömmlichen Business-Notebook (Intel® Core™ i5 M560 @ 2.67 GHz, 4 GB RAM) Laufzeittests durchgeführt. Obwohl der Prozessor eine Parallelisierung auf 4 Threads ermöglicht,

werden dabei die Berechnungen in einem einzelnen Thread ausgeführt. Eine Parallelisierung würde sowohl die Segmentierung, als auch die Zusammenführung erheblich beschleunigen. An dieser Stelle sind wir jedoch am direkten Laufzeitvergleich interessiert. Für die Laufzeitanalyse wird eine ALS Punktwolke mit etwa 2.5 Millionen Punkten, aufgenommen bei einer Befliegung im Naturschutzgebiet *Zonser Grind* am Niederrhein, verwendet.

Tab. 2: Einfluss der Kachelgröße auf die Laufzeit – Datensatz *Niederrhein*

Kachel-größe [m]	Anz. d. Kacheln	Segmente vor *Merge*	Segmente Final	Laufzeit Segmentierung auf Kacheln [s]	Laufzeit Zusammen-führung [s]	Laufzeit Gesamt [s]
75	90	26409	1019	1309	1075	2384
100	56	21012	1019	1234	775	2009
125	36	19256	1019	1310	506	1816
150	30	15456	1019	1317	559	1876
200	20	12960	1019	1259	402	1661
250	12	9575	1019	1316	212	1528

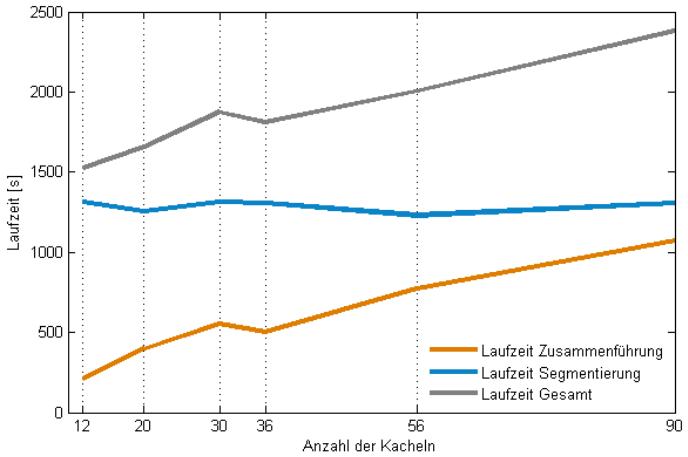

Abb. 5: Laufzeitvergleich, Segmentierung bei unterschiedlichen Kachelgrößen - Datensatz *Niederrhein*

Die Ergebnisse der Laufzeitanalyse zeigen, dass die Kachelgröße – wie erwartet – keinen Einfluss auf die Rechenzeit der Segmentierung hat. Die Laufzeit für die Zusammenführung nimmt hingegen mit steigender Kachelanzahl linear zu. Der Grund dafür liegt in der für die Zusammenführung relevanten Randbereichsfläche, welche mit zunehmender Kachelanzahl ebenfalls linear wächst (**Fehler! Verweisquelle konnte nicht gefunden werden.**). Mit den wachsenden Randbereichen steigt auch die Anzahl der Punkte, die in der Zusammenführung erneut auf Homogenität überprüft werden.

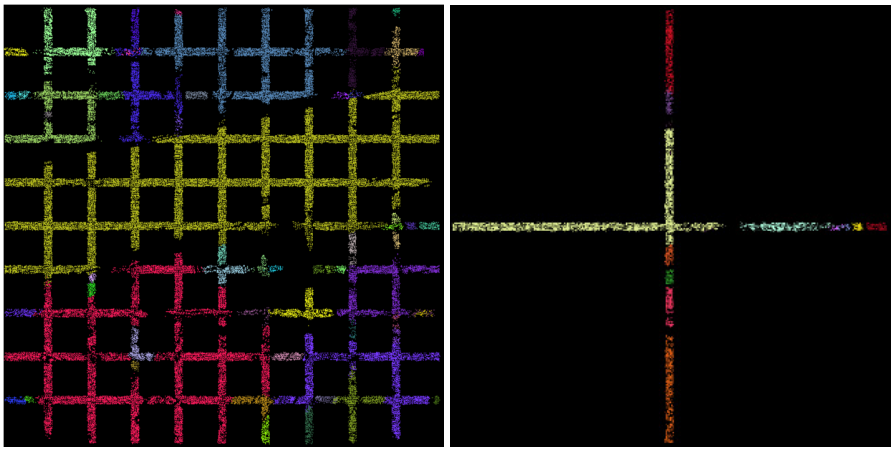

Abb. 6: Vergleich der Randbereiche aller Kacheln; Farbwerte ≙ Segment-IDs; 1 m Suchradius; 10 m Kacheln (li) und 50 m Kacheln (re)

4 Fazit & Ausblick

Wie die Ergebnisse bestätigen, liefert das entwickelte Konzept nach Segmentierung und Zusammenführung unabhängig von der gewählten Kachelgröße die gleichen Segmente. Die Ergebnisse hängen somit nur vom gewählten Homogenitätskriterium und dem gewählten Suchradius ab. Durch eine geeignete Wahl der Kachelgröße kann die Laufzeit für die Berechnung optimiert werden. Zusätzlich bietet das vorgestellte Konzept großes Potenzial für Parallelisierung. Sowohl die Segmentierung als auch die Zusammenführung der Kacheln ermöglichen eine verteilte Berechnung auf mehreren Threads. Bei guter Skalierung kann dabei, bei einer Verteilung auf *n* Threads, die Laufzeit auf ein *n*-tel gesenkt werden.

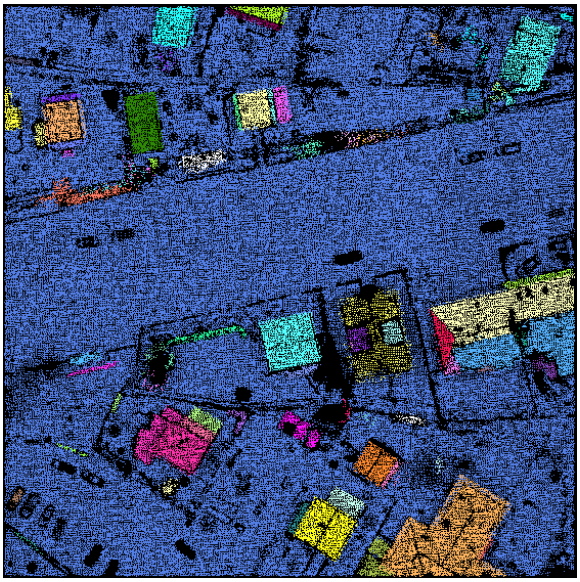

Abb. 7: Segmentierungsergebnis Testdatensatz Wien; Homogenitätskriterium $|p_i.\mathrm{NormalZ} - p_j.\mathrm{NormalZ}| < 0.01$; Suchradius r = 1 m; Unterschiedliche Farbwerte repräsentieren Segment-IDs

5 Literaturverzeichnis

ADAMS, R. & BISCHOF, L., 1994: Seeded region growing. IEEE Transactions on Pattern Analysis and Machine Intelligence **16**(6), 641-647.

BENTLEY, J. L., 1975: Multidimensional Binary Search Trees Used for Associative Searching. Communications of the ACM **18**(9), 509-517.

CHEN, M.-H. & PAVLIDIS T., 1990: Image seaming for segmentation on parallel architecture. IEEE Transactions on Pattern Analysis and Machine Intelligence **12**(6), 588-594.

PFEIFER, N., MANDLBURGER, G., OTEPKA, J. & KAREL, W., 2014: OPALS – A framework for Airborne Laser Scanning data analysis. Computers, Environment and Urban Systems **45**, 125-136

VOSSELMAN, G., GORTE, B.G., SITHOLE, G. & RABBANI, T., 2004: Recognising structure in laser scanner point clouds. International archives of photogrammetry, remote sensing and spatial information sciences **46**(8), 33-38