Ablauf

Mittwoch, 24. April 2019 14:24

Allgemeines Ziel: Falsche Korrespondenzen zu identifizieren und entweder zu entfernen oder mit einem geringeren Gewicht zu versehen

1) Einladen der Punktwolke & Raussuchen passender Regionen
   - laspy → numpy → .csv
   - Cloud compare tools, Ausschneiden ggf. mit python

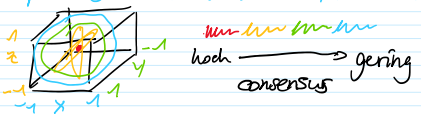
Beispiele:
XYZ gut, XZ gut, YZ gut, XY gut → gr. Gebäude?, Wald, Acker, Wasser, Fahrbahn, Gebäude abgerissen/neu gebaut, Bäume gefällt/neu gepflanzt

1.5) Erstellung einer manuellen Referenz-Transformation für alle Testgebiete
   - Cloud compare; mehrere Male für Sicherheit

2) Transformationsparameter mittels ICP $\Delta X, \Delta Y, \Delta Z$ ⇒ zuerst: entweder über Scanstreifen einer Punktwolke (easiest) oder ALS 2014/ALS 2016, später: ALS/DTM
   - mit python-pcl als Referenz zum eigenen
   - selbst programmiert: matching über kd-Baum mit ALS als Basis & Zuordnung von DTM, dann Ausgleichung andere ALS

a) fehlzuordnungen allg., z.B. Baumkronen zu zwei ALS (14/16)
b) systematische Fehler durch unterschiedliche Datenquellen, z.B. Lidar Punkte unter Baum vs DTM nur Baumkronen, DTM großes Rauschen in texturarmen Bereichen, DTM „rundet" alle Ecken aus

3) Robuste Ausgleichung mittels Maximum Consensus
   - Verschiebung in X,Y & Z in z.B. cm-Schritten um max. +-1m ⇒ 200^3 Berechnungen pro Testgebiet → Literaturrecherche!!!
   - Consensus Kriterium finden, z.B. | an Punktabständen unter x cm | Winkel zu. Normalenvektoren cos( ) < u_i, u_j >
     & in einer Matrix speichern
     - Distanz zu Punkt < ε
     später: Winkel der Normalenvektoren < ε
   - graphisch passende Darstellung für das Consensus Kriterium finden → Consensus Cube™
     - matplotlib → imshow für 2D, scatter für 3D?



& Gewichtung

4) Identifikation von Ausreißern/falschen Korrespondenzen ⇒ sollen falsche Zuordnungen entweder eliminieren/ausschließen oder zumindest für die Registrierung gewichten → Maximum Consens Kriterium / ICP Ausgleichung gewichten
   - „lokales Rauschen" mit PCA
     1) KD-Tree einer Punktwolke bauen & nächste Nachbarn zu Punkt i finden
     2) Koor auf Schwerpunkt reduzieren & Varianz-Kovarianz-Matrix aufstellen

$$x_i' = \bar{x} - x_i \quad y_i' = \bar{y} - y_i \quad z_i' = \bar{z} - z_i \rightarrow \begin{bmatrix} \Sigma x_i' x_i' & \Sigma x_i' y_i' & \Sigma x_i' z_i' \\ & \ddots & \\ & & \Sigma z_i' z_i' \end{bmatrix}$$

PCA-Varianz: „Eigenvektor zum kleinsten Eigenwert" ⇒ kleine = lokale Ebene

z.B. ICP: $\| \alpha (m - x_i) \|^2 \rightarrow \min$ mit α = Gewichtung

MC: Änderung/Anpassung des Kriteriums

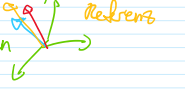
     3) PCA darauf berechnen
     4) 1-3 mehrmals & ggf. mit vers. Radien/kann wiederholen für robuste Ergebnisse
   - Normalenvektoren zu. korrespondierenden Punkten sollten ähnlich sein (durch PCA hat man diese eh schon fast) *
     Zuordnung verwerfen, wenn die Normalenvektoren sich zu stark unterscheiden
   - Klassen (sofern vorliegt oder berechnet werden kann)
     → Nutzen nur von best. Klassen (z.B. Dach, Fahrbahn) oder Zuordnungen je nach Klasse mit Unsicherheiten behaften (Dach = genau, Vegetation = ungenau)

* Normalenvektoren über PCA (s. oben) oder robust über RANSAC; Test mit vers. Radien

5) Vergleich der erreichten Transformationen visuell & quantitativ
   - ALS 2014 & ALS 2016 als „einfachen Fall"
   - ALS 2014 & ALS 2016 für Änderungen
   - ALS 2016 & DTM 2016 für allg. Beispiele
   - ICP, robuste Schätzung, Erweiterung vs. manuelle Referenzen checken

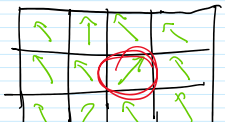
z.B. Region → ALS2016 farbcodiert mit Distanzen zum nächsten ALS 2014 Punkt

Profilansicht zu. beiden mit Residuen



Zeigen der vers. Transformationsvektoren im Vergleich

6) Test auf zusammenhängende Gebiete & Vergleich der Verschiebungen (optional)



Implementierung - grobe Struktur:

ALS: 20 pkte/m²
DM: 100 pkte/m²

def Matching:
- Sampling wegen Dichteunterschieden
- Sampling wegen Ausschlusskriterium, z.B. * Normalenvektor oder Klasse
- Zuordnung von Punkten (Distanz, Normalen)
- Gewichtung der Zuordnung!*

* lokales Rauschen, Normalenvektoren, Klassen, ...
* ggf. unabhängig zurückgeben

return Vektor mit ~~Zwischen~~ Übergabeparametern,
z.B. $\Delta x, \Delta y, \Delta z, \Delta n_x, \Delta n_y, \Delta n_z, \lambda$
oder gleich fertige Bestandteile für ICP/MC

def Optimierung:
- ICP main / Consensus main → call auf Matching + Subroutinen
- nimmt Vektor aus Matching & berechnet robuste Transformationsparameter
return $\Delta X, \Delta Y, \Delta Z$

def ICP:
- genaue Beschreibung von ICP

def CM:
- genaue Beschreibung von Maximum Consensus

Eigenwertzerlegung in python: sortiert nicht automatisch die Eigenwerte nach Größe!
Eigenvalue, eigenvector = np.linalg.eig(cov)
Idx = eigenvalue.argsort()[::-1]
Eigenvalue = eigenvalue[idx]
Sortiere Eigenvector ebenfalls nach idx