Praca domowa 4 – Ocena dokładności odwzorowania dachów

Opis ogólny

Ćwiczenie polega na stworzeniu algorytmu do automatycznej oceny jakości zamodelowania połaci dachowych w modelach LOD 2.

Dane wejściowe

Jako dane wejściowe do zadania posłużą dwa zbiory danych – modele budynków wykonane w standardzie CityGML LOD 2 oraz chmura punktów LIDAR z lotniczego skanowania laserowego. Proszę o samodzielne wybranie danych dla dowolnego obszaru – jeden kafelek danych LIDAR oraz jedna sekcja danych CityGML LOD2.

Elementy zadania

W ramach realizacji zadania należy przygotować skrypt który:

1. Odczyta dane z chmury punktów w formacie LAS/LAZ. Do wykonania obliczeń będą nam potrzebne tylko współrzędne punktów oraz informacje o tym do jakiej klasy pokrycia terenu należą.
    - Przy odczycie należy pozostawić tylko punkty należące do klasy *budynek* (kod klasy – 6)
    - Po filtracji na podstawie klasyfikacji potrzebne będą już tylko współrzędne punktów
2. Odczyta i wyekstrahuje z danych CityGML informacje o geometrii dachów (bldg:RoofSurface)
    - Niestety geopandas nie będzie w stanie odczytać danych LOD2 – trzeba postawić na bardziej prymitywne podejście
    - Pozostawiam dowolność w kwestii wyboru rozwiązania do wczytywania danych XML – jeżeli ktoś chce robić to samemu przez open(...) to nie ma problemu. Pod kątem wczytywania danych XML mogę polecić dwa rozwiązania – ElementTree z pakietu xml w Python, oraz BeautifulSoup z pakietu beautifulsoup4. Można też zobaczyć jak wczytywanie odbywa się w CityGML2OBJ i skorzystać z wybranych fragmentów kodu.
    - W teorii dachy w CityGML mogą mieć bardzo złożoną geometrię z tytułu obsługiwanych przez standard typów geometrii. Nie chcemy skupiać się na tworzeniu najlepszego na świecie parsera do CityGML, więc możemy przyjąć parę założeń/uproszczeń:
        - Budynek zawiera w sobie elementy typu bldg:RoofSurface – dach płaski ma jeden taki obiekt, dach wielopołaciowy - wiele
        - Każda połać dachu powinna składać się z jednego bldg:lod2MultiSurface
        - W lod2MultiSurface powinien znajdować się zawsze jeden obiekt gml:MultiSurface zawierający jeden gml:SurfaceMember opisany przez gml:Polygon (sprawdziłem parę arkuszy dla Krakowa i nie trafiłem innych przypadków)
        - W razie wystąpienia jakichś odstępstw, można je pominąć lub obsłużyć – bez różnicy
3. Wyznaczy błąd zamodelowania połaci względem chmury punktów
    - Na podstawie geometrii połaci wycinamy tylko punkty znajdujące się w jej obrysie (zawierają się w rzucie geometrii połaci na płaszczyznę XY
    - W punkty połaci dachu wpasowujemy płaszczyznę – można wpasować ją metodą najmniejszych kwadratów we wszystkie punkty, albo wybrać losowe 3 i na ich podstawie wyznaczyć jej parametry matematycznie (i tak wszystkie punkty powinny leżeć na jednej płaszczyźnie)
    - Dla każdego wybranego punktu z chmury wyznaczamy jego odległość od wpasowanej płaszczyzny. Proszę zwrócić uwagę na to żeby wyznaczać tzw. *Signed distance*.
    - Odrzucamy błędy grube – punkty na ścianach, punkty daleko odstające od płaszczyzny dachu. Można założyć, że jeżeli odległość punktu od płaszczyzny jest większa niż 1m to taki punkt uznajemy za błąd gruby i nie uwzględniamy go w dalszych badaniach

    - Wyznaczamy wartość średnią odległości dla pozostałych po odrzuceniu błędów grubych punktów.
4. Przedstawi wartości błędów w formie wizualizacji kartograficznej
    - Nie narzucam docelowej formy – liczę na Państwa inwencję twórczą 😊
    - Proszę tylko, żeby wykaz nie był w formie wypisania id – wartość błędu. Musi być chociaż kontekst przestrzenny
    - Zachęcam do skorzystania z Pythona do wizualizacji – prosty kartogram można nawet stworzyć w geopandas, ale znają Państwo też Open3D, więc może jakieś kolorowanie lub słupki 😉
    -

Inne informacje

- Za zadanie można zdobyć 3 punkty
- W razie wystąpienia jakichś problemów w kwestii założeń co do zadania, merytoryki itd. proszę pisać na Teams
- W razie czego proszę się kierować zasadą, że jeżeli skrypt zadziała dla większości budynków to jest OK.
- Zanim zaczną Państwo implementację polecam najpierw zobaczyć sobie jak zbudowany jest budynek w pliku LOD 2 – czym różni się budynek o dachu płaskim od wielospadowego.
- W przypadku korzystania z BeautifulSoup4, zalecam doinstalowanie parsera *lxml* przez pip install
- Można korzystać z innych zewnętrznych bibliotek – jeżeli nie instalowaliśmy ich w naszym środowisku na zajęcia to proszę tylko obok pliku Python z kodem przesłać jeszcze requirements.txt

Przydatne linki

- https://docs.python.org/3/library/xml.etree.elementtree.html
- https://beautiful-soup-4.readthedocs.io/en/latest/
- https://www.geeksforgeeks.org/parsing-tables-and-xml-with-beautifulsoup/
- https://www.cuemath.com/geometry/distance-between-point-and-plane/ - signed distance otrzymamy jeżeli nie będziemy liczyć wartości bezwzględnej z licznika we wzorze
- https://medium.com/@michaelscheinfeild/fit-plane-3d-05fcd0304ae0
- https://numpy.org/doc/stable/reference/generated/numpy.matmul.html - proszę zwrócić uwagę na różnicę w działaniu * i @ dla tablic numpy
- https://laspy.readthedocs.io/en/latest/complete_tutorial.html
- https://laspy.readthedocs.io/en/latest/examples.html#filtering

Przekazanie zadania

Jako wynik przekazujemy paczkę ZIP zawierającą:

1. Skrypt Python realizujące pracę domową
2. Zrzut ekranu/filmik stworzonej wizualizacji z ewentualnym opisem jeżeli do jej stworzenia było wykorzystane jakieś dodatkowe narzędzie/program
3. Dane lidar i GML na których ćwiczenie było realizowane