Instrukcja użytkowania i sprawozdanie projektu Automatycznej Klasyfikacji Chmur Punktów dla CPK

HUTNICY
Mikołaj Klima, Michał Mróz, Miłosz Nowak i Tymon Szyler

Grudzień 2025

Spis treści

1 Instrukcja dla użytkownika końcowego (EXE)

1.1 Wymagania systemowe

- System operacyjny: Windows 10/11 (64-bit)
- RAM: minimum 8 GB (zalecane 16 GB)
- Miejsce na dysku: 500 MB wolnego miejsca
- **Nie wymaga instalacji Pythona ani żadnych bibliotek**

1.2 Przygotowanie i uruchomienie

1. Pobierz plik `ClassifyCloud.exe`
2. Umieść go w tym samym folderze co plik chmury punktów (.las lub .laz)
3. Kliknij dwukrotnie na plik `ClassifyCloud.exe`
4. **Pierwsze uruchomienie:** 3-5 minut (wczytanie i zapis cache)
5. **Kolejne uruchomienia:** 2-3 minuty (z cache)

1.3 Wyniki

Po zakończeniu w folderze pojawi sie `wyniki_YYYYMMDD_HHMMSS` zawierajacy:

- `klasyfikacja.png` - wizualizacje (3D, XY, XZ, histogram)
- `chmura_sklasyfikowana.las` - plik z przypisanymi klasami
- `statystyki.txt` - raport tekstowy

1.4 Klasy obiektów

ID	Nazwa	Kolor
0	Inne/Nieklasyfikowane	Jasnoszary
1	Ziemia/Trawa	Brazowy
2	Krawężnik	Złoty
3	Zieleń/Drzewa	Zielony
4	Słup	Czerwony
5	Budynek	Niebieski
6	Kable/Przewody	Czarny
7	Asfalt/Droga	Srebrny

Tabela 1: 8 klas + nieklasyfikowane

1.5 Przegladanie wyników

Otwórz plik .las w:

- CloudCompare - `https://www.cloudcompare.org/`
- QGIS z wtyczka Point Cloud - `https://qgis.org/`

1.6 Rozwiazywanie problemów

- **Nie znajduje pliku:** Sprawdź rozszerzenie .las/.laz i lokalizacje
- **Zamyka sie natychmiast:** Uruchom z PowerShell/CMD dla błedów
- **Brak pamieci:** Zamknij inne aplikacje (program próbkuje 500k punktów)

2 Instalacja ze źródeł

2.1 Wymagania

- Python 3.10+
- pip

2.2 Instalacja i uruchomienie

```
git clone https://github.com/[REPO]/cpk-classifier.git
cd cpk-classifier
pip install laspy numpy matplotlib scikit-learn
python classification_tool.py
```

2.3 Budowanie EXE

```
pip install pyinstaller
pyinstaller -onefile -clean -name=ClassifyCloud
    -exclude-module=tensorflow -exclude-module=torch
    classification_tool.py
```

Plik w `dist/ClassifyCloud.exe`

3 Opis rozwiazania technicznego

3.1 Architektura

Hierarchiczna klasyfikacja geometryczna bez uczenia maszynowego. Cechy lokalne + RGB zapewniaja szybkie i przewidywalne działanie.

3.2 Pipeline przetwarzania

1. Wczytanie i próbkowanie

- Wczytanie LAS/LAZ (laspy)
- Losowanie 500k punktów (seed=42)
- Ekstrakcja: XYZ, intensywność, RGB
- Zapis cache (pickle)

2. Cechy geometryczne (k=25 sasiadów, KD-Tree)

- Linearity - liniowość (kable, krawężniki)
- Planarity - płaskość (budynki, drogi)
- Sphericity - sferyczność (drzewa)
- Verticality - pionowość (budynki, słupy)
- Vertical axis - orientacja osi
- Cylinder radius - promień (słupy vs drzewa)
- Height continuity - ciagłość wysokości
- Local density - gestość punktów

Wzory:

$$Linearity = \frac{\lambda_1 - \lambda_2}{\lambda_1}, \quad Planarity = \frac{\lambda_2 - \lambda_3}{\lambda_1}, \quad Sphericity = \frac{\lambda_3}{\lambda_1}$$

gdzie $\lambda_1 \geq \lambda_2 \geq \lambda_3$ to wartości własne macierzy kowariancji.

3. Cechy RGB

- Brightness - jasność
- Greenness - zieleń (G - mean(R,B))
- RGB std - uniformność koloru

4. Analiza terenu

- Siatka 2D (2m)
- Minimum Z na komórke
- Wysokość nad gruntem (h)
- Grunt: h mniejsze 0.3m

5. Klasyfikacja hierarchiczna

Poziom 1 - Grunt (h mniejsze 0.3m):

- Intensity mniejsze mediana: Klasa 7 (Asfalt)
- Intensity wieksze-równe mediana: Klasa 1 (Ziemia)

Poziom 2 - Wysokie (h wieksze 2.0m):

- Budynki (5): verticality wieksze 0.85 AND planarity wieksze 0.6
- Kable (6): linearity wieksze 0.7 AND vertical_axis mniejsze 0.15 AND h wieksze 4.0
- Słupy vs Drzewa - scoring (próg 8 pkt):
  - Cylinder radius mniejsze 0.5m: +3
  - Linearity wieksze 0.75: +2
  - Neighbor sphericity mniejsze 0.08: +3
  - Brightness wieksze 0.35: +2
  - RGB std mniejsze 0.05: +2
  - Greenness mniejsze 0.0: +3
  - Local density mniejsze 80: +2
  - Height continuity wieksze 3.0: +2

  Jeśli wieksze-równe 8: Klasa 4 (Słup), inaczej: Klasa 3 (Drzewo)
- Zieleń: sphericity wieksze 0.05 OR greenness wieksze 0.03

Poziom 3 - Średnie (0.5m mniejsze h mniejsze-równe 2.0m):

- Zieleń: sphericity wieksze 0.04 OR greenness wieksze 0.02
- Budynki niskie: verticality wieksze 0.8 AND planarity wieksze 0.4

Poziom 4 - Niskie (0.05m mniejsze h mniejsze-równe 0.5m):

- Krawężniki (2) - scoring (próg 8 pkt):
  - Linearity wieksze 0.4: +3
  - Vertical axis mniejsze 0.4: +3

  - Wysokość 0.08-0.35m: +4
  - Intensity wieksze Q25: +2
  - Planarity wieksze 0.2: +2
  - Gradient wysokości wieksze P60: +2
  - Density 50-200: +1
  - Szary kolor: +1
  - Brak zieleni: +1
- Zieleń niska: greenness wieksze 0.01

6. Post-processing

- Usuwanie szumu: punkty z mniejsze 5 sasiadów tej samej klasy → Klasa 0

3.3 Wyniki

- Czas: 2-3 min (z cache), 3-5 min (bez)
- Próbka: 500k punktów
- Średnia pewność: około 0.75

3.4 Mocne strony

- Bez trenowania ML - gotowe od razu
- Deterministyczne
- Szybkie (2-3 min)
- Portable (jeden EXE)
- Standard LAS
- Scoring system (elastyczne progi)

3.5 Ograniczenia

- Próbkowanie 500k - dla wieksze 100M rozważ tiling
- Słupy vs drzewa w gestych lasach
- Brak klas specjalistycznych (znaki, bariery)

3.6 Rozszerzenia

1. Eksport IFC (w rozwoju)
2. Detekcja zmian (monitoring budowy)
3. Kontrola jakości BIM
4. Raportowanie PDF
5. Batch processing
6. GUI

3.7 Technologie

Python 3.12, laspy, NumPy, scikit-learn, Matplotlib, PyInstaller

4 Podsumowanie

Prototyp klasyfikacji 8 klas obiektów infrastrukturalnych bez ML. Cechy: łatwość (EXE), szybkość (2-3 min), zgodność (LAS), modularność. Gotowy do wdrożenia w pipeline przetwarzania 3D CPK.

Nota: IFC planowany jako dodatkowa funkcjonalność - ograniczenia ifcopenshell w PyInstaller. Możliwy w przyszłych wersjach.