UNIVERZITA KARLOVA
Přírodovědecká fakulta
Katedra aplikované geoinformatiky a kartografie

Studijní program:
Geoinformatika, kartografie a dálkový průzkum Země

Jakub Fuchsig, Šimon Kohout

Prostorová indexace

Úkol č. 2

Ústí nad Labem, Zákupy 2025

Úvod:

Práce s rozsáhlými prostorovými daty, jako jsou například mračna bodů získaná laserovým skenováním, přináší specifické výpočetní výzvy. Základní úlohou při analýze takových dat je často hledání nejbližších sousedů (Nearest Neighbor Search), které je nezbytné pro výpočet odvozených charakteristik, jako je prostorová hustota nebo křivost povrchu.

Pokud bychom k této úloze přistoupili naivním způsobem (metoda hrubé síly), museli bychom pro každý bod zkoumané množiny prohledat všechny ostatní body. To vede k velké časové složitosti. U malých datasetů je tento přístup akceptovatelný, avšak u reálných geodat, která často obsahují miliony bodů, se doba výpočtu stává enormní.

Z tohoto důvodu se v geoinformatice využívá prostorová indexace. Cílem indexace je organizovat data do pomocných datových struktur, které rozdělují prostor na menší, hierarchicky nebo geometricky uspořádané celky. To nám umožňuje při vyhledávání rychle eliminovat velké části prostoru, kde se hledaný bod s jistotou nenachází, a tím radikálně snížit počet nutných porovnání.

1. Zadání:

Geoinformatika, prostorová indexace
Pro zadané bodové mračno tvořené body $p\_i = [x_i, y_i, z_i]$ určete níže uvedené charakteristiky:

- Prostorovou hustotu,
- Křivost v každém bodě.

1. Výpočet charakteristik

Prostorová hustota

Pro výpočet prostorové hustoty mračna využijte průměrnou vzdálenost d_{aver} k nejbližšímu bodu:

$$\rho = \frac{1}{d_{aver}^3}$$

Aproximovaná křivost

Křivost určete metodou PCA (Principal Component Analysis) v každém bodě mračna ze vztahu:

$$\kappa = \frac{\lambda_1}{(\lambda_1 + \lambda_2 + \lambda_3)}$$

- Počet knn (k-nearest neighbors) volte 30.

2. Metody vyhledávání

Prohledávání bodového mračna bude využívat následující metody:

1. Naivní hledání
2. Akcelerované hledání s využitím voxelizace
3. Akcelerované hledání s využitím kd-tree

Poznámka: Veškeré výše uvedené vyhledávací struktury implementujte samostatně (tj. bez použití externích knihoven pro tyto struktury).

3. Zadání implementace a vizualizace

- Určení hustoty bodového mračna realizujte metodami 1–4*.
- Výpočet křivosti realizujte metodami 2–4*.
- Hodnoty křivosti v každém z bodů mračna vizualizujte s využitím vhodné barevné škály.
- Vizualizujte výpočetní čas jako funkci velikosti vstupní množiny.
- U metody 2 (voxelizace) vizualizujte čas také jako funkci velikosti voxelu

2. Řešení

Vstupní data pro testování algoritmů představují mračno bodů reprezentující objekt – konkrétně vzrostlý strom. Data byla poskytnuta v textovém souboru *tree_18.txt.*

- Formát**:** Textový soubor (ASCII), kde každý řádek reprezentuje jeden bod.
- Struktura**:** Tři sloupce oddělené tabulátorem nebo mezerou, odpovídající kartézským souřadnicím [x, y, z].
- Rozsah**:** Soubor obsahuje celkem 25 737 bodů.

Implementace byla provedena v programovacím jazyce Python. Jako výchozí bod byl použita kostra skriptu připravenou vyučujícím (**doc. Ing. Tomáš Bayer, Ph.D.**).

Tento poskytnutý základ zajišťoval:

- Načítání bodů ze souboru do paměti (funkce loadPoints).
- Základní vizualizaci mračna bodů a voxelů pomocí knihovny matplotlib.
- Definici funkce pro naivní vyhledávání nejbližšího souseda (getNN).

Naším úkolem bylo tento skript rozšířit o vlastní implementaci tříd a funkcí pro prostorovou indexaci a logiku pro výpočet hustoty, aniž bychom pro samotné vyhledávání využívali externí optimalizované knihovny (jako scipy nebo sklearn). Při řešení bylo využito i umělé inteligence (Google Gemini), jakožto pomocného nástroje. Veškeré nejistoty, opravy syntaxe či hrubé návrhy implementace, které se při tvorbě kódu objevovaly, byly konzultovány právě s AI. Tyto pasáže jsou označeny i ve vlastních kódech.

2.1 Naivní metoda

Prvním krokem implementace bylo vytvoření referenční metody pro vyhledávání nejbližšího souseda. Tento přístup, označovaný jako "naivní" nebo "hrubá síla", spočívá v iteraci přes všechny body v mračnu. Pro každý dotazovaný bod q je vypočtena Euklidovská vzdálenost ke všem ostatním bodům p_i ze sady P. Algoritmus si průběžně pamatuje bod s minimální nalezenou vzdáleností. I když je tato metoda implementačně triviální a zaručuje nalezení

přesného výsledku, její časová složitost $O(N^2)$ ji činí pro větší datové sady prakticky nepoužitelnou.

2.2 Voxelizace (Grid Index)

Jako první akcelerační strukturu jsme zvolili metodu voxelizace, která dělí prostor na pravidelnou mřížku (grid). Princip spočívá v přiřazení každého bodu do konkrétní buňky (voxelu) na základě jeho souřadnic. Pro efektivní správu těchto buněk jsme využili hashovací tabulku (v Pythonu reprezentovanou slovníkem), kde klíčem je unikátní jednodimenzionální index buňky a hodnotou seznam bodů v ní obsažených. Velikost buňky byla odvozena z celkového počtu bodů N a rozsahu souřadnic. Při samotném vyhledávání pak algoritmus neprochází celé mračno, ale pouze body uvnitř stejné buňky jako dotazovaný bod. Jedná se o metodu přibližnou (Approximate Nearest Neighbor), neboť neprohledáváme sousední buňky, což vede k výraznému zrychlení výpočtu $(O(N))$, avšak za cenu mírné ztráty přesnosti v případech, kdy leží nejbližší soused těsně za hranicí voxelu.

2.3 KD-Tree

Nejnáročnější částí úlohy byla implementace KD-stromu bez využití externích knihoven, což vyžadovalo vytvoření vlastní třídy pro uzel stromu (KDNode) a samotný strom (KDTree). Při návrhu konstrukce stromu jsme vycházeli z obecných algoritmických principů pro vkládání bodů do k-dimenzionálního prostoru (GeeksforGeeks 2025). Strom je konstruován rekurzivním dělením prostoru podle mediánu bodů, přičemž se v každé úrovni hloubky stromu cyklicky střídají dělící osy (x, y, z).

Samotné vyhledávání nejbližšího souseda (*get_nn*) pak probíhá průchodem stromem od kořene k listu. Klíčovým prvkem pro zajištění přesnosti bylo implementovat tzv. backtracking (návrat v rekurzi). Abychom zaručili nalezení skutečně nejbližšího bodu, implementovali jsme kontrolní podmínku popsanou u Rahmana (2024) a Tyagiho (2024). Ta ověřuje, zda "koule" o poloměru k aktuálně nejbližšímu sousedovi nezasahuje přes dělící rovinu do vedlejší větve.

Pokud ano, algoritmus prohledá i druhou stranu; pokud ne, větev se celá zahodí. Časová náročnost pro předpřipravení dat je $O(n \log n)$.

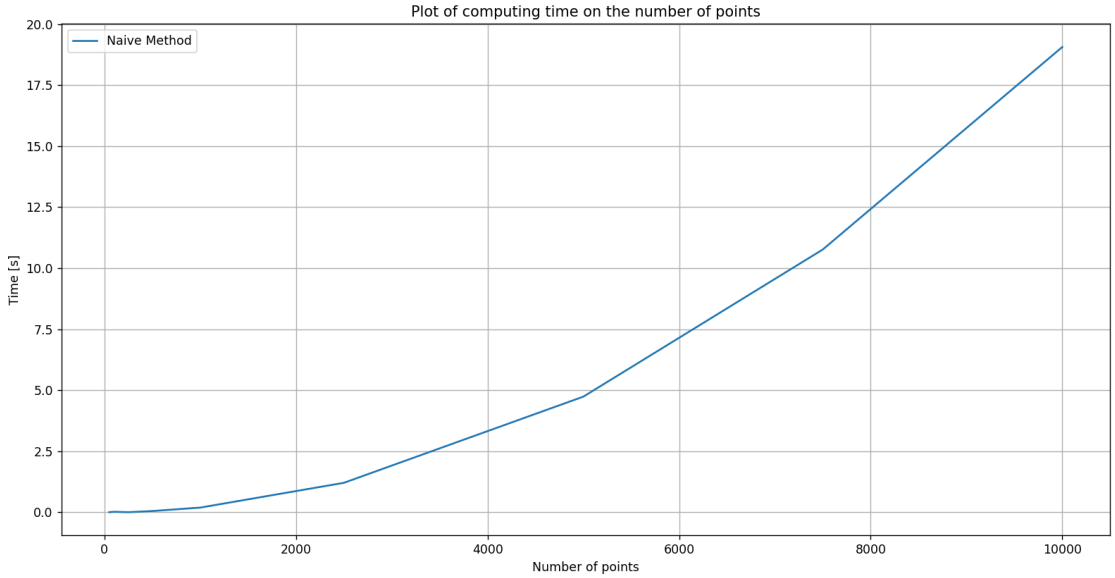
2.4 R-Tree (bonus)

Jedná se o jednu z nejpoužívanějších struktur pro indexaci prostorových dat. Na rozdíl od KD-Tree, který dělí prostor pomocí rovin, R-Tree je založen na struktuře vnořených min-max boxů. Princip spočívá v seskupování blízkých objektů do uzlů, díky čemuž při vyhledávání dokáže algoritmus efektivně ignorovat celé větve stromu, které se s dotazovanou oblastí neprotínají. Výhodou R-Tree je schopnost pracovat nejen s body, ale i s objekty, které mají rozlohu (linie či polygony). Nevýhodou naopak může být překryv obdélníků. V případě hledání určitého bodu, nemusí tento bod ležet v témže obdélníku. Pro zpracování úlohy bylo využito *rtree* knihovny společně s její dokumentací (RTree Documentantion 2019).

3. Výsledky

3.1 Naivní metoda

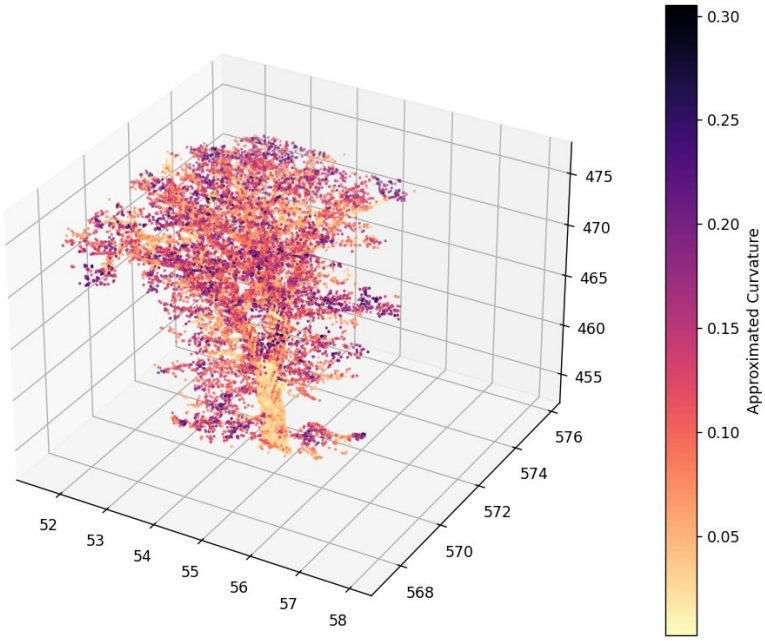
Při prohledávání bodového mračna touto metodou bylo hned na první pohled vidět, že se jedná o velice pomalou, avšak přesnou metodu. Této skutečnosti napovídá i výsledný graf, který zobrazuje závislost času na velikosti vstupních dat (Obrázek X). Je možné pozorovat exponenciální funkci, což dokazuje teoretickou časovou složitost $O(N^2)$. Při testování byl oproti ostatním rychlejším metodám snížený počet testovacích dat (od 50 do 10 000 bodů) a zároveň při měření samotného výpočetního času nebyl do měření zahrnut výpočet křivosti.



Obrázek 1 – Měření časové náročnosti na počtu vstupních dat pro Naivní metodu

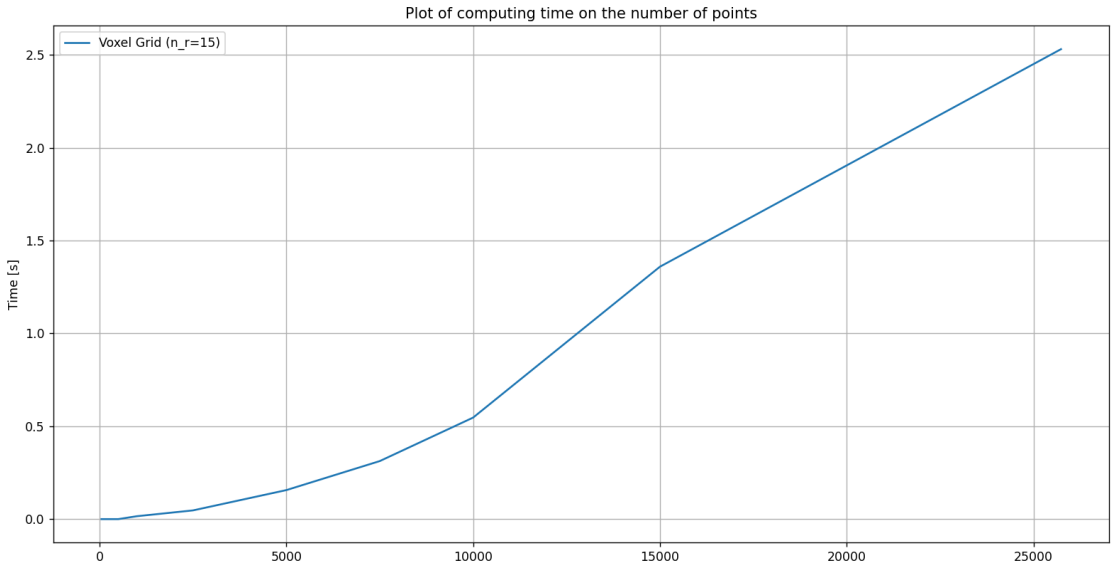
3.2 Voxelizace (Grid Index)

V teoretické části je zmíněné rozdělování prostoru do stejně velkých voxelů, do kterých jsou přiřazovány jednotlivé body podle jejich souřadnic. Ve skriptu bylo mimo výpočtu hustoty aproximována i křivost K metodou PCA v jednotlivých bodech pomocí 30 nejbližších bodů. V této části bylo nutné prohledávat i sousedící voxely aproximovaného bodu. Při vizualizaci obarveného mračna (Obrázek X) lze pozorovat, že body, které tvoří roviny (např. kmen) jsou obarveny světle, zatímco body, které jsou roztroušené v prostoru (např. koruna) jsou obarveny tmavěji.



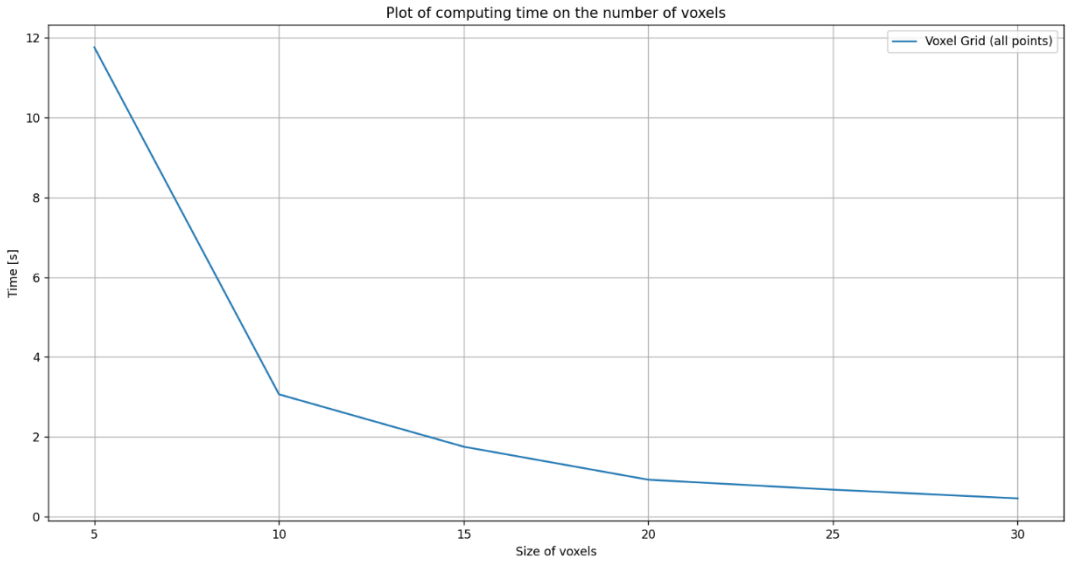
Obrázek 2 – Obarvení bodového mračna pomocí aproximované křivosti bodů metodou PCA pro metodu voxelizace

Pro tuto metodu byla testována i časová náročnost na výpočet hustoty a křivosti v bodech v závislosti na počtu bodů pro fixní počet voxelů 15 (Obrázek X). Výsledný ukazuje rostoucí trend blížící se k lineární závislosti. Na drobné výkyvy může mít vliv např. nerovnoměrné rozložení dat ve voxelech.



Obrázek 3 – Měření časové náročnosti na počtu vstupních dat pro metodu voxelizace

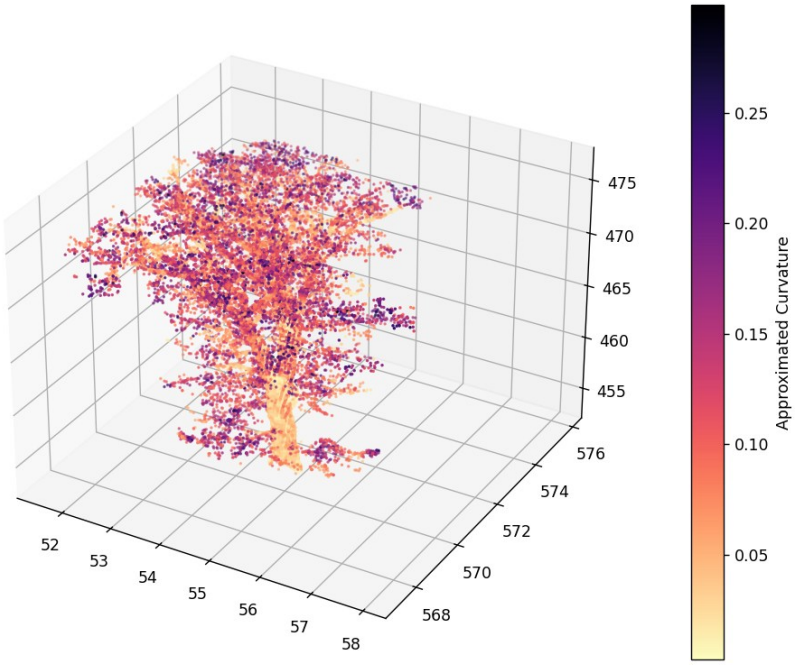
Pro tuto metodu byla měřena i časová závislost na velikosti, resp. počtu voxelů. Předpokládalo se, že při menším počtu voxelů bude prohledávání sousedních bodů připomínat výše zmíněnou Naivní metodu. S rostoucím počtem, a tudíž klesající velikostí voxelů by měla být využita rychlost prohledávání a výsledný čas by se měl zkracovat. Tomuto odpovídá i naměřen graf časové závislosti (Obrázek X).



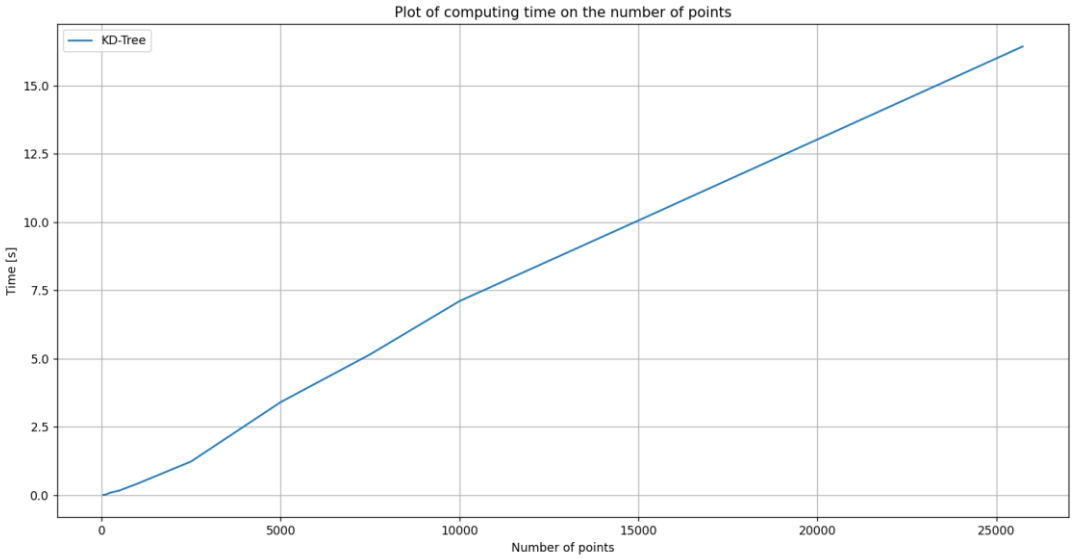
Obrázek 4 – Měření časové náročnosti na počtu, resp. velikosti voxelů pro metodu Voxelizace

3.3 KD-Tree

O něco pomalejší metoda se jevila právě KD-Tree. I pro tuto metodu byly počítány veškeré charakteristiky. Stejně jako u předešlé metody bylo vizualizováno obarvené bodové mračno v závislosti na aproximovanou křivost jednotlivých bodů (Obrázek X). Jelikož teoretické zpracování všech dat vychází na $O(n \log n)$, byla i tato část testována. Do testovací sady byly vloženy počty bodů od 50 do celého bodového mračna. Výsledným grafem (Obrázek X) byla křivka připomínající lineární funkci s rychlejších růstem v nižších hodnotách, tedy graf odpovídá teorii.



Obrázek 5 – Obarvení bodového mračna pomocí aproximované křivosti bodů metodou PCA pro metodu KD-Tree



Obrázek 6 – Měření časové náročnosti na počtu vstupních dat pro metodu KD-Tree

3.4 R-Tree (bonus)

Výrazně nejrychlejší byla metoda R-Tree, která dosahovala výpočtu hustoty kolem dvou sekund. Jelikož se jednalo o bonusový úkol nebyla tedy aproximována šikmost v jednotlivých bodech. Toto rozhodnutí mělo za následek mírné snížení výpočetního času oproti ostatním metodám.

Veškeré charakteristiky jednotlivých metod jsou shrnuty přehledně do Tabulky 1 níže.

Tabulka 1 – Naměřené charakteristiky jednotlivých metod prohledávání

	Method			
	Naive	**Voxelization**	**KDTree**	**Rtree**
Density	5881,44372	4226,82384	5881,44372	5757,56359
Average distance	0,05540	0,06185	0,05540	0,05579
Time of Computing (s)	150*	30	30	2
Time of Computing samples (s)	120**	60	90	x

* U naivní metody nebyla aproximována křivost v jednotlivých bodech pomocí PCA, tento čas je tedy jen pro výpočet hustoty

** U naivní metody byl snížený testovací vzorek počtu bodů v grafu od 50 do 10 000

4. Závěr

V tomto úkolu bylo prozkoušeno několik metod prohledávání bodového mračna. Jednalo se o Naivní metodu a 3 akcelerované metody (Voxelizace, KD-Tree a R-tree). Pro tři zmíněné metody byly dopočítány a různé charakteristiky souhrnně zobrazeny v tabulce. Byly vytvořeny i testy, díky kterým bylo možné pozorovat výpočetní časovou závislost jednotlivých metod na počtu vstupních dat, resp. velikosti voxelů. Z našich výsledků vychází, že metoda voxelizace byla nejrychlejší, avšak za cenu nižší přesnosti při výpočtu hustoty (viz Tabulka 1). Zatímco metoda KD-Tree byla o něco pomalejší, výsledek výpočtu hustoty bodového mračna ukazuje, že je velice přesná. Vypracován byl i bonusový úkol použití metody R-Tree z knihovny. Tato metoda byla výrazně rychlejší než ostatní metody a její přesnost výpočtu hustoty se od Naivní metody či metody KD-Tree lišila jen velice málo.

Zdroje a použitá literatura:

1. Odborná literatura a podklady

- BAYER, Tomáš. *Prostorová indexace s využitím gridu* [online]. Praha: Katedra aplikované geoinformatiky a kartografie, PřF UK, [cit. 2025-12-03]. Dostupné ze studijních materiálů předmětu.

2. Internetové zdroje

- APXML. *Practice: Implementing a KD-Tree*. APXML, [cit. 2025-12-03]. Dostupné z: https://apxml.com/courses/advanced-python-programming-ml/chapter-4-advanced-data-structures-algorithms-ml/practice-implementing-kd-tree
- COMPUTERPHILE. *k-d Trees* . YouTube, 20. 1. 2014 [cit. 2025-12-03]. Dostupné z: https://www.youtube.com/watch?v=TLxWtXEbtFE
- GEEKSFORGEEKS. *Search and Insertion in K Dimensional tree*. GeeksforGeeks, 2024 [cit. 2025-12-04]. Dostupné z: https://www.geeksforgeeks.org/search-and-insertion-in-k-dimensional-tree/
- RAHMAN, Muneeb ur. *k-d Trees for nearest neighbor search*. Medium, 24. 12. 2024 [cit. 2025-12-03]. Dostupné z: https://medium.com/@notesbymuneeb/k-d-trees-for-nearest-neighbor-search-df4fc459da51
- RTREE. Spatial indexing for Python. Read the Docs, 2025 [cit. 2025-12-05]. Dostupné z: https://rtree.readthedocs.io/
- TYAGI, Pradyumna. *Approximate Nearest Neighbors (ANN) Algorithm using KD-Trees from Scratch in Python*. Medium, 9. 8. 2024 [cit. 2025-12-04]. Dostupné z: https://medium.com/@prxdyu/approximate-nearest-neighbors-ann-algorithm-using-kd-trees-from-scratch-in-python-41d3f0a34214