

Univerzita Karlova

Přírodovědecká fakulta

Studijní program: Geoinformatika, kartografie a dálkový průzkum Země



Petr Havel, Jan Bartůšek

Prostorová indexace

1. Úvod

Tato práce se zabývá problematikou efektivního zpracování prostorových dat v geoinformatice, konkrétně analýzou 3D mračna bodů reprezentujícího vegetaci. Vstupní data jsou tvořena textovým souborem `tree_18.txt`, který obsahuje souřadnice jednotlivých bodů. Jednou z motivací této úlohy je řešení časové náročnosti, která vzniká při vyhledávání sousedních bodů v rozsáhlých datasetech. Bez použití optimalizačních algoritmů roste výpočetní čas kvadraticky, což činí analýzu velkých mračen neefektivní. Cílem práce je proto implementace a vzájemné porovnání několika metod prostorové indexace, které umožňují akcelerované vyhledávání, a jejich následné využití pro výpočet geometrických charakteristik mračna.

2. Zadání

Zadání úlohy požaduje zpracování bodového mračna definovaného body o souřadnicích x , y , z . Klíčovým úkolem je implementace šesti různých přístupů k prohledávání prostoru, přičemž je nutné postupovat bez využití externích knihoven pro prostorové indexování. Požadované metody zahrnují naivní hledání, akcelerované hledání s využitím voxelizace neboli gridu, a dále hledání s využitím hierarchických stromových struktur, konkrétně KD-tree, Octree a R-tree.

Na základě takto nalezených sousedů je nutné pro každý bod mračna vypočítat dvě základní charakteristiky. První z nich je prostorová hustota, která se určí z průměrné vzdálenosti d_{aver} k nejbližšímu bodu podle vztahu:

$$\rho = 1 / d_{\text{aver}}^3$$

Druhou charakteristikou je aproximovaná křivost, která se stanoví metodou analýzy hlavních komponent (PCA) na okolí třiceti nejbližších sousedů. Hodnota křivosti je definována jako podíl nejmenšího vlastního čísla ku součtu všech vlastních čísel kovarianční matice:

$$\kappa = \lambda_1 / (\lambda_1 + \lambda_2 + \lambda_3)$$

Součástí zadání je rovněž vizualizace vypočtené křivosti pomocí vhodné barevné škály a provedení benchmarku výpočetního času jednotlivých metod v závislosti na velikosti vstupní množiny dat.

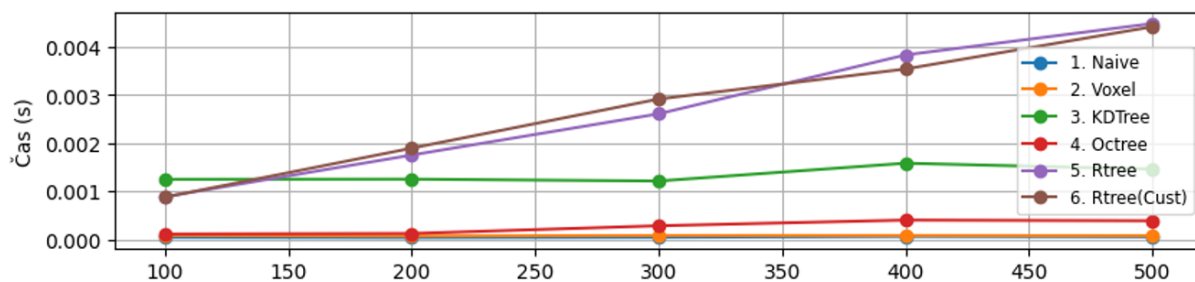
3. Využívané metody a implementace

Pro realizaci úlohy byl zvolen programovací jazyk Python, přičemž veškeré indexační struktury byly naprogramovány manuálně. Jako referenční metoda bylo implementováno naivní hledání, které pro každý dotazovaný bod iteruje přes všechny ostatní body v mračnu. Tento přístup s kvadratickou složitostí slouží především pro ověření správnosti výsledků optimalizovaných algoritmů.

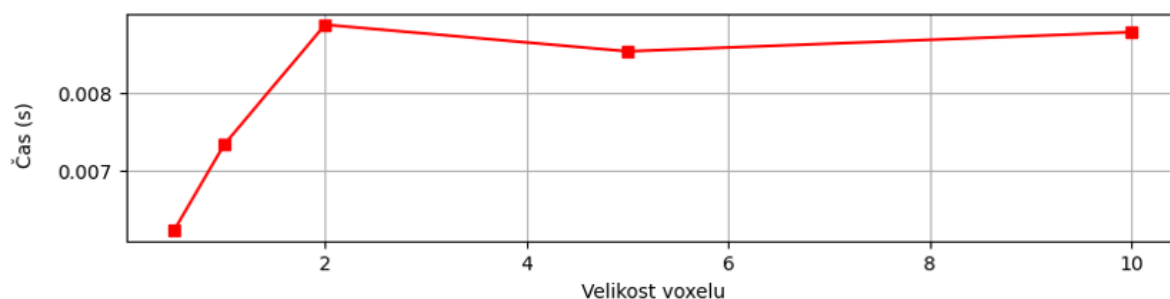
Druhou implementovanou metodou je Voxel Grid, který dělí prostor na pravidelnou mřížku buněk. Velikost buňky a počet buněk na hraně gridu jsou odvozeny z celkového počtu bodů tak, aby byla zajištěna efektivita přístupu. Pro rychlý převod 3D souřadnic na jednorozměrný index buňky byla použita hashovací funkce, která umožňuje ukládat body do hashovací tabulky a následně prohledávat pouze relevantní okolí bodu.

Dále byly implementovány stromové struktury, které dělí prostor hierarchicky. KD-tree dělí prostor rekurzivně pomocí nadrovin kolmých k osám souřadného systému, což je velmi efektivní pro hledání k-nejbližších sousedů. Metoda Octree dělí každý uzel prostoru na osm oktantů a je vhodná i pro nerovnoměrně rozložená data. Poslední strukturou je R-tree, respektive jeho zjednodušená varianta, která organizuje body do hierarchie ohraničujících boxů a umožňuje rychle vyloučit celé oblasti prostoru, které se nepřekrývají s oblastí dotazu. Výpočet křivosti byl následně realizován s využitím knihovny NumPy pro operace lineární algebry, konkrétně pro výpočet vlastních čísel kovarianční matice sousedních bodů.

4. Interpretace výsledků a grafů



Graf 1: Benchmark – Čas hledání vs. počet bodů

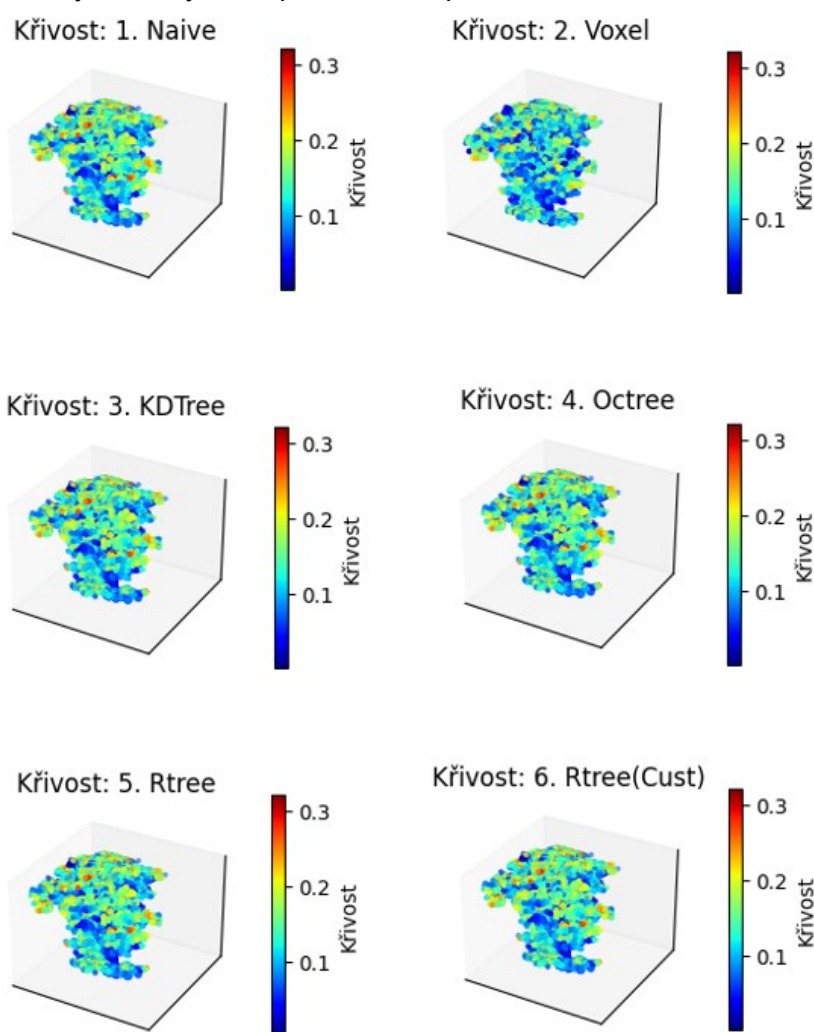


Graf 2: Benchmark – Čas vs. velikost voxelu

Analýza výsledků začíná posouzením výpočetní náročnosti jednotlivých metod (Graf 1: Benchmark – Čas hledání vs. počet bodů), která je znázorněna na grafu závislosti času hledání na počtu bodů. Při testování na menších vzorcích dat o velikosti sto až pět set bodů je patrný zdánlivě paradoxní jev, kdy naivní metoda hrubé síly dosahuje srovnatelných, či dokonce lepších časů než komplexní stromové struktury. Tento výsledek je způsoben tím, že

pro takto malý počet bodů je kvadratická složitost naivního algoritmu zanedbatelná, neboť moderní procesory zvládnou statisíce porovnání v řádu milisekund. Naproti tomu sofistikovanější metody, zejména R-Tree implementovaný v čistém Pythonu, trpí vysokou režijní zátěží spojenou s rekursí a vytvářením velkého množství objektů. Lze však předpokládat, že s dalším nárůstem počtu bodů by se křivka naivního hledání strmě zvedala, zatímco stromy jako KD-Tree nebo Octree by si zachovaly lineární či logaritmický růst, což z nich činí jedinou použitelnou volbu pro velká data.

Druhý graf, zaměřený specificky na metodu Voxel Grid, ilustruje vliv velikosti buňky na celkový čas výpočtu (Graf 2). Z průběhu křivky je zřejmé, že s rostoucí velikostí voxelu dochází ke zpomalování algoritmu. Tento trend má logické opodstatnění v principu fungování mřížky. Při nastavení malé velikosti voxelu obsahuje každá buňka jen minimální počet bodů, což umožňuje velmi rychlé prohledání relevantního okolí. Naopak, pokud je voxel příliš velký, spadne do jedné buňky velké množství bodů a algoritmus je nucen uvnitř této buňky provádět neefektivní sekvenční porovnávání všech kandidátů. Tím se vytrácí výhoda prostorové indexace a metoda degraduje na úroveň naivního přístupu. Pro optimální výkon je tedy klíčové nastavit velikost buňky tak, aby korespondovala s průměrnou hustotou mračka.



Obrázek 1: Grafy vizualizace křivosti pomocí zadáných metod

Další část analýzy se věnuje vizuální kontrole správnosti a interpretaci vypočtených hodnot křivosti. Série šesti vizualizací, reprezentující každou z implementovaných metod, vykazuje naprostou shodu v barevném rozložení, což potvrzuje, že všechny algoritmy – od naivního po

R-Tree – identifikovaly pro každý bod totožné sousedy a fungují korektně. Z hlediska geometrické interpretace odpovídají modré oblasti nízkým hodnotám křivosti, což indikuje hladké povrchy, jako je kmen stromu nebo silné větve, které lokálně připomínají válec či rovinu. Oblasti zbarvené žlutě až červeně naopak značí vysokou křivost. Tyto hodnoty se vyskytují v místech s velkým rozptylem normál, typicky v koruně stromu, kde se nachází shluky listů, tenké větvičky a místa větvení, která netvoří spojitou plochu.

Zajímavý vhled do přesnosti jednotlivých algoritmů přineslo srovnání numerických výsledků výpočtu prostorové hustoty. Zatímco metody založené na exaktním vyhledávání – tedy naivní přístup, KD-Tree, Octree i obě varianty R-Tree – dospěly ke zcela shodné průměrné hodnotě hustoty 6060.374341, metoda využívající voxelizaci vykázala odlišný výsledek 5507.542382. Tento rozdíl je způsoben principem fungování voxel gridu, který byl v rámci optimalizace rychlosti implementován s omezeným poloměrem prohledávání sousedních buněk. Pokud se nejbližší bod nachází až za hranicí prohledávaných voxelů (což se může stát v řidších oblastech mračna), algoritmus nenalezne globálně nejbližšího souseda, ale pouze lokální optimum. Výsledek tak potvrzuje teoretický předpoklad, že voxelizace v této konfiguraci funguje jako metoda přibližného vyhledávání (Approximate Nearest Neighbor), která vyměňuje absolutní matematickou přesnost za nižší výpočetní náročnost, zatímco stromové struktury zajistily nalezení skutečného nejbližšího souseda ve všech případech.

5. Závěr

V rámci řešení úlohy byly úspěšně implementovány všechny požadované vyhledávací struktury a aplikovány na zadané mračno bodů. Z provedených benchmarků vyplynulo, že naivní metoda je pro větší objemy dat časově neúnosná, zatímco indexační stromy jako KD-tree a Octree redukovat čas hledání řádově na zlomek původní hodnoty. U metody voxelizace se potvrdilo, že její efektivita silně závisí na správně zvolené velikosti buňky, kdy příliš malé voxely zvyšují režii procházení prázdného prostoru a příliš velké voxely degradují výkon na úroveň naivního hledání. Vizualizace křivosti prokázala schopnost algoritmů správně identifikovat geometrické rysy stromu, kdy byly zřetelně odlišeny ploché a válcové části kmene od členitějších oblastí větví a listů. Výsledné řešení tedy splňuje všechny body zadání a poskytuje robustní nástroj pro analýzu prostorových dat.