



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

**ZOBRAZENÍ LIDAROVÝCH, KAMEROVÝCH
A VEKTOROVÝCH DAT Z ŽELEZNIČNÍHO
MOBILNÍHO MAPOVACÍHO SYSTÉMU**
THESIS TITLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ZUZANA MIŠKAŇOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ONDŘEJ KLÍMA, Ph.D.

BRNO 2025

Zadání bakalářské práce



Ústav: Ústav počítačové grafiky a multimédií (UPGM) 164356
Studentka: **Miškaňová Zuzana**
Program: Informační technologie
Název: **Zobrazení lidarových, kamerových a vektorových dat z železničního mobilního mapovacího systému**
Kategorie: Uživatelská rozhraní
Akademický rok: 2024/25

Zadání:

1. Seznamte se možnostmi vizualizace obrazových a geometrických dat z mobilních mapovacích systémů.
2. Navrhněte uživatelský systém pro vizualizaci agregovaných dat ze senzorů umístěných na čele vlaku včetně vektorových mapových dat.
3. Navržený systém implementujte v podobě uživatelské aplikace v prostředí Python s využitím existujících nástrojů a knihoven.
4. Proveďte experimenty s dodanými daty a vyhodnoťte vlastnosti a uživatelskou přívětivost aplikace.
5. Prezentujte dosažené výsledky.

Literatura:

Dle doporučení vedoucího.

Při obhajobě semestrální části projektu je požadováno:

Body 1, 2 a částečně 3.

Podrobné závazné pokyny pro vypracování práce viz <https://www.fit.vut.cz/study/theses/>

Vedoucí práce: **Klíma Ondřej, Ing., Ph.D.**

Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.

Datum zadání: 1.11.2024

Termín pro odevzdání: 14.5.2025

Datum schválení: 12.11.2024

Abstrakt

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém (slovenském) jazyce.

Abstract

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v anglickém jazyce.

Kľúčové slová

vizualizácia dát, dierkový model kamery, mračno bodov, Python, deck.gl, webové aplikácie, Dash

Keywords

data visualization, pinhole camera model, point cloud, Python, deck.gl, web applications, Dash

Citácia

MIŠKAŇOVÁ, Zuzana. *Zobrazení lidarových, kamerových a vektorových dat z železničního mobilního mapovacího systému*. Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ondřej Klíma, Ph.D.

Zobrazení lidarových, kamerových a vektorových dat z železničního mobilního mapovacího systému

Prehlásenie

Prehlasujem...

.....
Zuzana Miškaňová
11. apríla 2025

Poděkovanie

...

Obsah

1	Úvod	2
2	Vizualizácia dát z mobilných mapovacích systémov	3
2.1	Dierkový model kamery – teoretický základ perspektívneho zobrazenia	3
2.2	Vykreslovanie 3D dát v počítačovej grafike	6
2.3	Skreslenie kamery	6
2.4	Existujúce možnosti zobrazenia dát z mobilných mapovacích systémov v jazyku Python	6
2.5	Frameworky pre tvorbu interaktívnej webovej aplikácie v jazyku Python	8
3	Návrh aplikácie pre vizualizáciu dát z železničného mobilného mapovacieho systému	9
3.1	Návrh používateľského rozhrania	10
3.2	Problém nahrávania dát do aplikácie	10
4	Implementácia navrhnutej aplikácie	11
4.1	Optimalizácia vykreslovania mračna bodov priamym použitím jazyka JavaScript	12
4.2	Experimenty a vyhodnotenie výsledkov	13
5	Záver	14
	Literatúra	15

Kapitola 1

Úvod

Kapitola 2

Vizualizácia dát z mobilných mapovacích systémov

Výstupom mobilného mapovacieho systému je množstvo dát z rôznych senzorov. Dáta, ktoré boli poskytnuté k tvorbe tejto práce, obsahovali v prvom rade mračná bodov získané z lidaru. Aby bolo možné s týmito mračnami bodov pracovať a vhodne ich zobrazovať, je potrebné, aby mobilný mapovací systém zaznamenával údaje o svojej polohe. V tomto prípade boli k dispozícii údaje o transláciách a rotáciách kamery s časovými razítkami. Ďalej býva súčasťou mobilného mapovacieho systému klasická kamera. Pre zobrazenie mračna bodov takým spôsobom, aby sa zobrazenie čo najviac zhodovalo s kamerovým záznamom, je nutné poznáť parametre kamery – kalibračnú maticu a prípadne aj parametre skreslenia.

Pri vývoji aplikácie, ktorá má umožniť vizualizáciu týchto dát a navyše aj ďalších vektorových dát, je nutné poznáť základné princípy zobrazovania 3D dát v počítačovej grafike. Ďalej je potrebné vybrať si a naštudovať technológie, pomocou ktorých bude aplikácia vytvorená, a to konkrétnie niektorý z frameworkov pre tvorbu aplikácií v jazyku Python a vhodný framework pre zobrazenie grafických dát. Práve to je predmetom tejto kapitoly.

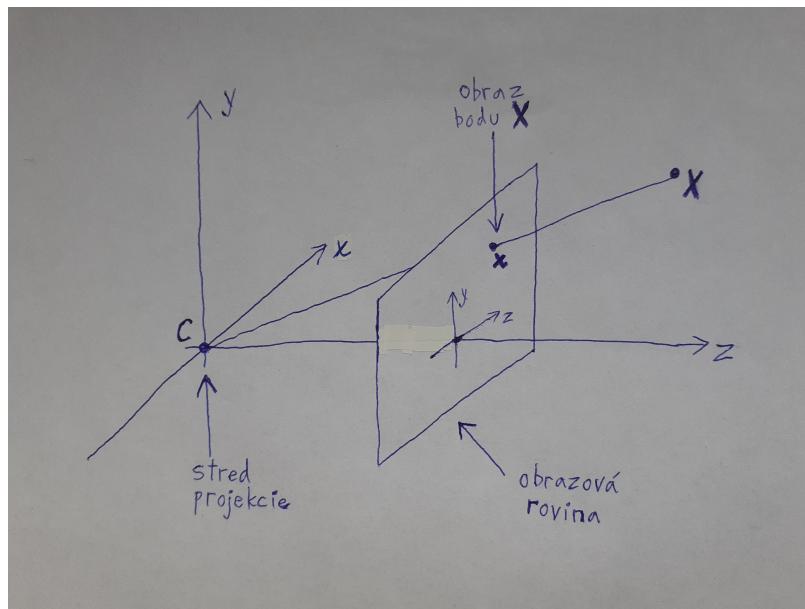
2.1 Dierkový model kamery – teoretický základ perspektívneho zobrazenia

Kamera je v počítačovej grafike pojmom, ktorý označuje projekciu bodov z trojrozmerného priestoru do roviny. Túto projekciu je možné vyjadriť pomocou matíc a je väčšinou bodová (*central projection*).

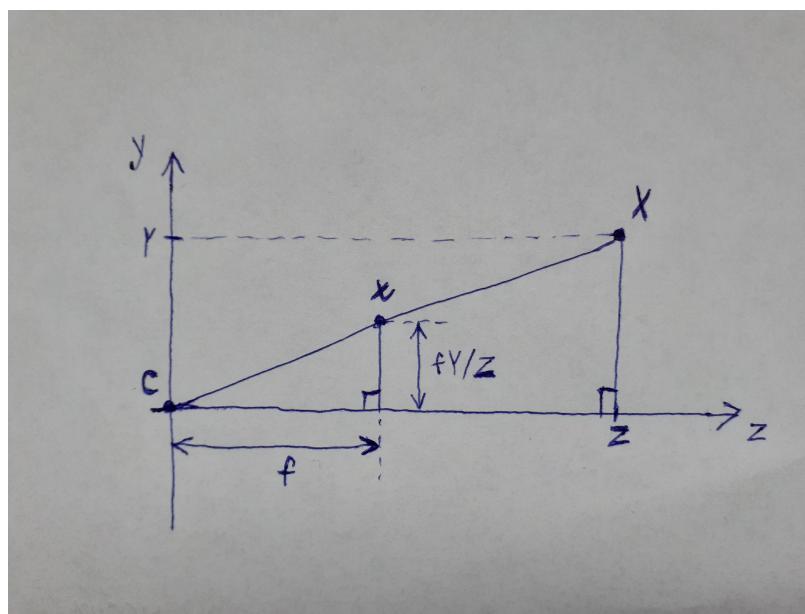
Existuje niekoľko rôznych modelov kamery, z ktorých najjednoduchší je dierkový model kamery (*pinhole camera model*). U tohto modelu je stred projekcie **C** (*camera centre*) v počiatku Euklidovského súradnicového systému a body sa premietajú do roviny $z = f$, ktorá sa označuje ako obrazová rovina (*image plane*).

Princíp zobrazenia je nasledovný: obrazom bodu $\mathbf{X} = (X, Y, Z)^T$ je bod \mathbf{x} , kde priamka vedúca bodom \mathbf{X} a stredom projekcie **C** pretína obrazovú rovinu (obrázok 2.1). Z podobnosti trojuholníkov je možné odvodiť, že bod \mathbf{x} má súradnice $(fX/Z, fY/Z, f)^T$, postup je naznačený na obrázku 2.2.

Kedže všetky obrazy bodov ležia v obrazovej rovine $z = f$, je možné poslednú súradnicu zanedbať a zapisovať súradnice bodu \mathbf{x} v súradnicovom systéme obrazovej roviny ako $(fX/Z, fY/Z)^T$.



Obr. 2.1: [TODO: prekresliť] Zakladný princíp dierkového modelu kamery.



Obr. 2.2: [TODO: prekresliť] Náčrt odvodenia súradníc bodu x.

Túto projekciu môžeme v homogénnych súradničach zapísat pomocou násobenia matíc nasledovným spôsobom:

$$\mathbf{x} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ f & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ f & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \mathbf{X} = \mathbf{P}\mathbf{X}$$

Maticu P nazývame projekčnou maticou kamery (*camera projection matrix*).

Rozšírenia základného dierkového modelu kamery

V praxi väčšinou chceme vyjadriť body na obrazovej rovine v súradnicovom systéme, ktorý nemá stred v bode $(0, 0, f)^T$, ale v ľubovoľnom bode $(-p_x, -p_y, f)^T$ (obrázok 2.3). Obrazom bodu $\mathbf{X} = (X, Y, Z)^T$ (v súradnicovom systéme kamery) potom bude bod $\mathbf{x} = (fX/Z + p_x, fY/Z + p_y, f)^T$ (v súradnicovom systéme obrazovej roviny). To je možné zahrnúť do projekčnej matice nasledovným spôsobom:

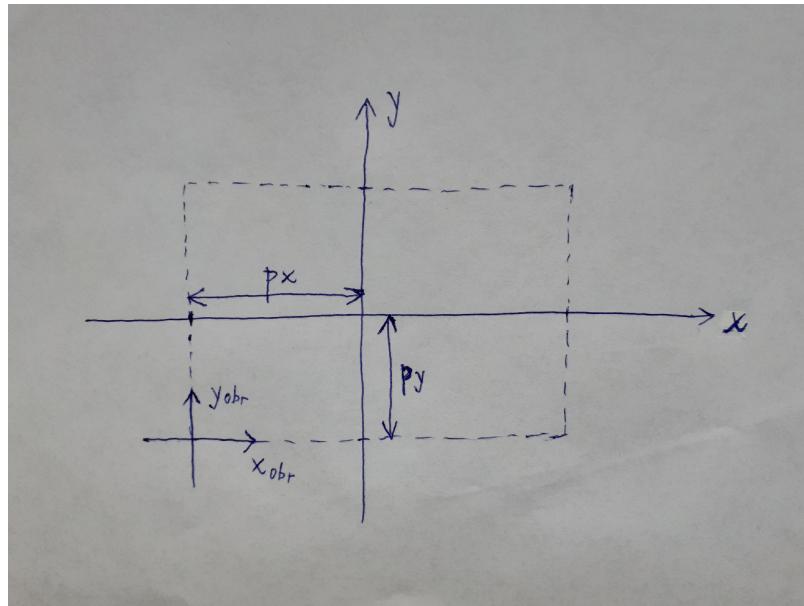
$$P = \begin{bmatrix} f & p_x & 0 \\ f & p_y & 0 \\ 1 & 0 & \end{bmatrix}$$

$$\mathbf{x} = \begin{pmatrix} fX/Z + p_x \\ fY/Z + p_y \\ 1 \end{pmatrix} = \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & p_x & 0 \\ f & p_y & 0 \\ 1 & 0 & \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Maticu

$$K = \begin{bmatrix} f & p_x \\ f & p_y \\ 1 & \end{bmatrix}$$

nazývame kalibračnou maticou kamery (*camera calibration matrix*) a medzi ňou a projekčnou maticou platí vzťah $P = [K|0]$, kde 0 predstavuje nulový stĺpcový vektor.



Obr. 2.3: [TODO: prekresliť] Súradnicový systém so stredom v bode $(-p_x, -p_y, f)^T$ obrazovej roviny.

Parametre f , p_x a p_y označujeme ako **vnútorné parametre kamery**.

Až doteraz sme predpokladali, že kamera má stred v počiatku súradnicovej sústavy a je „otočená“ v smere osi z , t. j. že obrazová rovina je rovnobežná s rovinou xy . V praxi to tak však nebýva a kamera má v súradnicovom systéme, v ktorom sú definované zobrazované

body, istú rotáciu a transláciu. V takom prípade rozlišujeme všeobecný súradnicový systém a súradnicový systém kamery (*world coordinate frame* a *camera coordinate frame*).

Ak $\tilde{\mathbf{X}}$ je vektor súradníc bodu \mathbf{X} vo všeobecnom súradnicovom systéme a vektor $\tilde{\mathbf{X}}_{\text{cam}}$ reprezentuje ten istý bod v súradnicovom systéme kamery, tak platí vzťah

$$\tilde{\mathbf{X}}_{\text{cam}} = \mathbf{R}(\tilde{\mathbf{X}} - \tilde{\mathbf{C}}),$$

kde $\tilde{\mathbf{C}}$ sú súradnice stredu kamery vo všeobecnom súradnicovom systéme a \mathbf{R} je rotačná matica 3×3 reprezentujúca orientáciu súradnicového systému kamery. To viedie k novému vyjadreniu projekčnej matice ako $\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I}] - \tilde{\mathbf{C}}$, kde \mathbf{I} je jednotková matica 3×3 .

Parametre \mathbf{R} a $\tilde{\mathbf{C}}$ označujeme ako **vonkajšie parametre kamery**.

Zdrojom všetkých informácií, ktoré boli uvedené v tejto sekcií, je kniha *Multiple View Geometry in Computer Vision* od autorov R. Hartley a A. Zisserman [1].

2.2 Vykreslovanie 3D dát v počítačovej grafike

[world space, view space, clip space, screen space, view matrix, projection matrix, viewport transform, vzťah k dierkovému modelu kamery]

2.3 Skreslenie kamery

[parametre skreslenia, výpočet]

2.4 Existujúce možnosti zobrazenia dát z mobilných mapovacích systémov v jazyku Python

Framework deck.gl a jeho nadstavba Pydeck

Pri vývoji webovej aplikácie pre vizualizáciu väčšieho množstva dát, u ktorej má vykreslovanie prebiehať na strane klienta, teda vo webovom prehliadači, sa hodí priamo či nepriamo použiť niektorý z frameworkov pre zobrazovanie dát v jazyku JavaScript.

Takým vhodným frameworkom je napríklad deck.gl, ktorý je určený na zobrazovanie veľkých sád dát. Je zameraný najmä na zobrazovanie geografických dát mapových podkladov, ale hodí sa aj na iné typy dát. Vyznačuje sa vysokou presnosťou a výkonnosťou. Pre akceleráciu využíva rozhrania WebGPU a WebGL2 [2].

Vizualizácia dát v deck.gl sa skladá z dvoch základných častí:

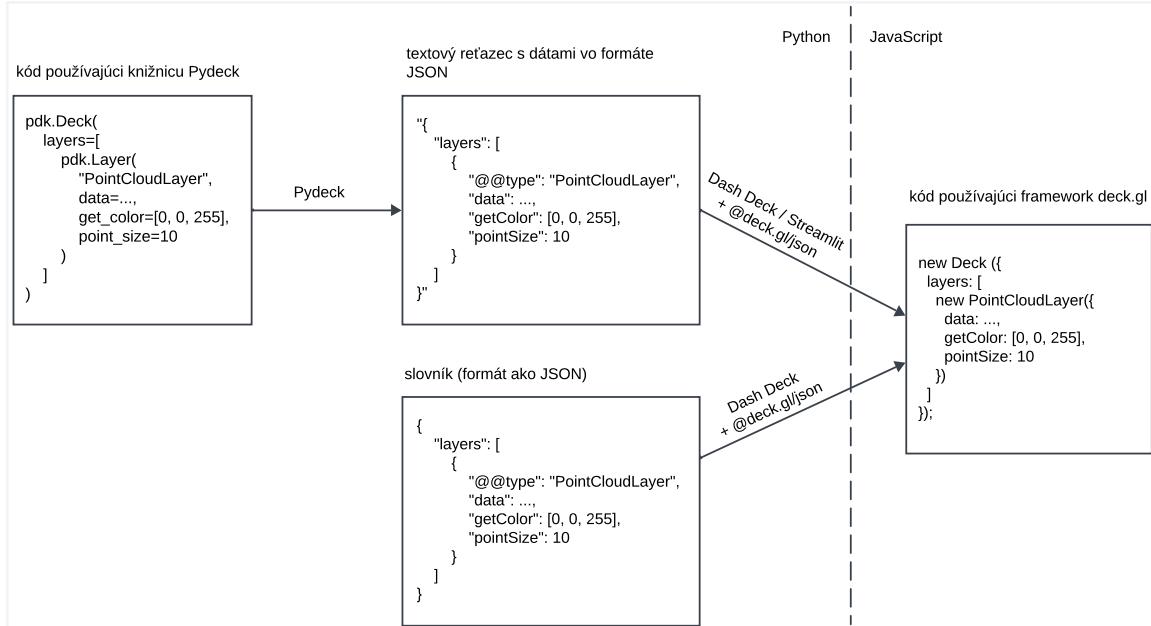
- **Vrstvy (Layers)**. Do vrstiev sa ukladajú zobrazované dáta. Framework deck.gl ponúka vyše 30 preddefinovaných typov vrstiev, ktoré zodpovedajú rôznym často sa vyskytujúcim typom dát. Pre túto prácu je významná najmä vrstva **PointCloudLayer**, ktorá je určená na zobrazenie mračna bodov, a vrstva **PathLayer**, ktorá je určená na zobrazenie trás.
- **Pohľad (View)**. Definuje vlastnosti kamery, napríklad zorné pole a prednú a zadnú orezávaciu rovinu (*near plane* a *far plane*). Je možné buď určiť všetky tieto vlastnosti osobitne, alebo rovno zadat vypočítanú projekčnú maticu. Časť **ViewState** určuje polohu a orientáciu kamery, pričom u orientácie je možné určiť iba dva uhly (*bearing* a *pitch*). Typ pohľadu definuje spôsob interakcie vizualizácie s používateľom, napríklad pre zobrazenie trate z pohľadu strojvedúceho je ideálny typ **FirstPersonView**.

To, že sa deck.gl na vizualizáciu dát z mobilného mapovacieho systému naozaj hodí, dokazujú ukážky na jeho webových stránkach. Použiteľnosť vrstvy PointCloudLayer demonštruje plynulá animácia mračna vyše 800 000 bodov.

Na webových stránkach deck.gl sa nachádza aj galéria projektov vytvorených pomocou tohto frameworku. Medzi nimi je Autonomous Visualization System, toolkit pre vývoj webových aplikácií pre vizualizáciu dát z autonómnych vozidiel [6]. Na webových stránkach tohto projektu je ukážka takej aplikácie, ktorá zobrazuje vozidlo, dáta z lidaru, vektorové dátá, ako napríklad trasu vozidla, a záznam z kamery. To je veľmi podobné aplikácii vyvíjanej v rámci tejto práce, ale je tu niekoľko rozdielov. Poprvé, mračno bodov nie je agregované, a teda sa v každom kroku animácie zobrazuje iba v tom momente nasnímaná časť, a podruhé, kamerový záznam sa zobrazuje oddelené, a nie na pozadí mračna bodov a vektorových dát.

Hoci je framework deck.gl primárne určený pre použitie v Javascripte, je možné ho použiť aj v jazyku Python, a to pomocou knižnice **Pydeck**. Tá je pomerne jednoduchá a podstatou jej činnosti je, že prevedie kód napísaný v jazyku Python do formátu JSON. Framework deck.gl má totiž modul @deck.gl/json, ktorý prijíma reprezentáciu vizualizácie vo formáte JSON a transformuje ju do javascriptového kódu (na definície funkcií a deck.gl objektov)¹.

Knižnica Pydeck je dobrým prostriedkom na vytvorenie jednoduchých vizualizácií, s ktorými môže používateľ interagovať pohybmi myši. Jej možnosti sú však oproti pôvodnému frameworku deck.gl veľmi obmedzené. Nie je vhodná na vytváranie zložitejších animácií s veľkým množstvom dát, pretože sa aj po tej najmenšej zmene musia dátá a definícia vizualizácie nanovo prevádztať do formátu JSON a následne na javascriptový kód (obrázok 2.4), čo je veľmi časovo náročné.



Obr. 2.4: Schéma vzťahov medzi technológiami Pydeck, Dash Deck a deck.gl a transformácií, ktorými prechádza definícia zobrazenia.

¹Ukážka rozhrania modulu @deck.gl/json je na <https://deck.gl/playground>.

2.5 Frameworky pre tvorbu interaktívnej webovej aplikácie v jazyku Python

Porovnanie frameworkov Streamlit a Dash

Streamlit a Dash sú frameworky, ktoré majú rovnaké zameranie: oba slúžia na tvorbu webových aplikácií pre prácu s dátami (*data apps*) v jazyku Python. Dash je oproti Streamlitu na nižšej úrovni abstrakcie, pretože sám o sebe nemá žiadnen vizuálny štýl a mnohé jeho komponenty sa priamo mapujú na HTML elementy, napríklad `dash.html.Div` a `dash.html.H1` [4][3].

Oba frameworky majú podporu pre Pydeck, u Streamlitu je priamo k dispozícii element `st.pydeck_chart` a Dash má na tento účel vytvorenú prídavnú knižnicu **Dash Deck**. Ukázalo sa však, že `st.pydeck_chart` podporuje iba pohľad `MapView`, ktorý je určený na zobrazenie dát na mape a nedá sa použiť na perspektívne zobrazenie bodov v trojrozmernom priestore. Preto je pre účely tejto práce element `st.pydeck_chart` prakticky nepoužiteľný.

Dash Deck má navyše tú výhodu, že umožňuje vynechať Pydeck a definovať zobrazenie iba pomocou slovníkov so štruktúrou zodpovedajúcou tej, ktorú vyžaduje modul `@deck.gl/json`, čo je tiež znázornnené na obrázku 2.4. To trochu zefektívni vykonávanie zmien vo vizualizácii, keďže taká reprezentácia umožní jednoduchšie vykonávanie úprav.

Popis frameworku Dash

[...]

Kapitola 3

Návrh aplikácie pre vizualizáciu dát z železničného mobilného mapovacieho systému

Cieľom tejto práce bolo vytvoriť používateľskú aplikáciu. Výsledná aplikácia mala byť webová, aby ju bolo možné spustiť jednoducho pomocou webového prehliadača. Mala by splňať nasledujúce body:

- Zobrazenie dát z mobilného mapovacieho systému. Tieto dátá sú tvorené mračnom bodov z lidaru, kamerovým záznamom, údajmi o pohybe vlaku a ďalšími vektorovými dátami a mali by byť zobrazené z pohľadu strojvedúceho vlaku.
- Umožniť používateľovi vybrať si konkrétnu pozíciu vlaku aj prehrať si animáciu pohybu vlaku.
- Umožniť používateľovi nahráť súbory s dátami na zobrazenie: súbor s mračnom bodov, video z kamery na čele vlaku, súbor s vektorovými dátami, textové súbory s údajmi o pohybe vlaku – pole translácií, pole rotácií a pole zodpovedajúcich časových razítok.
- Zobrazovať dva typy dát z lidaru – buď malé, postupne nasnímané mračná bodov s časovými razítkami, alebo iba jedno celistvé mračno bodov, ktoré vzniklo ich spojením (u oboch typov ale ide o agregované dátá – všetky body sú v jednej spoločnej súradnicovej sústave). Umožniť používateľovi prepínať medzi týmito dvoma typmi.
- Poskytnúť používateľovi možnosti prispôsobenia zobrazenia, ako napríklad zmeny viditeľnosti jednotlivých vrstiev (mračno bodov, vektorové dátá, kamerový záznam) a základné nastavenia zobrazenia mračna bodov (rôzne farebné škály podla intenzity, veľkosť a priehľadnosť bodov, zobrazenie bodov len do určitej vzdialenosťi) aj vektorových dát (farba a hrúbka čiar). Užitočná by bola aj možnosť doladiť parametre zobrazovania mračna bodov a vektorových dát (posun aj otočenie kamery doprava/doľava, nahor/nadol, posun dopredu/dozadu), aby bolo možné toto zobrazenie prispôsobiť záznamu z kamery. Kamera totiž môže byť na vlaku rôzne umiestnená a údaje z nej môžu byť oproti údajom z mračna bodov posunuté.
- Zobrazovať prejazdný profil vlaku v predikovanej polohe v rôznych vzdialostiach pred vlakom - 25, 50, 75 a 100m. Zobrazovať aj čiaru spájajúcu predikované polohy. Umožniť používateľovi nahráť súbory s týmito predikovanými polohami.

- Intuitívnosť a používateľská prívetivosť.

3.1 Návrh používateľského rozhrania

[návrh vrátane popisu zvolených farebných škál pre mračno bodov]

3.2 Problém nahrávania dát do aplikácie

Dáta, ktoré má vyvíjaná webová aplikácia zobrazovať, sú rozdelené do väčšieho množstva súborov rôznych typov. Nahrávať ich do aplikácie klasickým spôsobom po jednom či po nejakých skupinách by bolo sice možné, ale pre používateľov časovo náročné a veľmi nepraktické.

Kedže je aplikácia určená úzkej cieľovej skupine používateľov, môžeme si dovoliť predpokladať, že používateľ bude bud' spúšťať server webovej aplikácie lokálne, alebo bude mať k serveru aspoň prístup. Za tohto predpokladu má problém s nahrávaním dát veľmi elegantné riešenie, a tým je **projektový súbor**, teda textový súbor, v ktorom sú špecifikované cesty k dátovým súborom uloženým na serveri. Používateľovi stačí nahrať tento súbor a aplikácia podľa neho automaticky nahráť všetky potrebné dátové súbory.

Presný formát projektového súboru bolo potrebné navrhnúť. Kompletný návrh je uvedený v prílohe [DOPLNIT]. Bol zvolený formát TOML, ktorý je jednoduchý, praktický a dobre čitateľný [5]. U dátových súborov, ktorých môže byť rôzny počet, napríklad súbory s dátami z lidaru, sa do projektového súboru uvedie, v akom priečinku sa nachádzajú, ako sú pomenované a koľko ich je. Predpokladá sa, že sú tieto súbory jednotne pomenované a očíslované (napríklad `pcd_0.pcd`, `pcd_1.pcd`, ...).

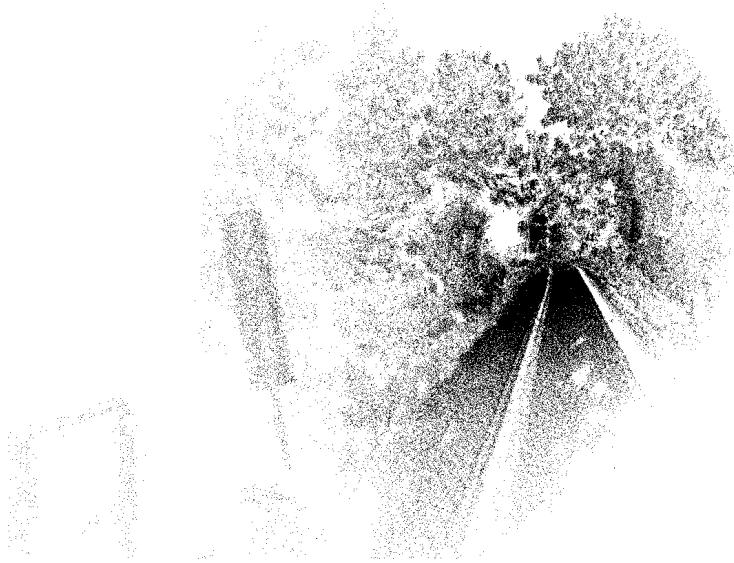
Keby však bol projektový súbor jediným možným spôsobom nahrávania dát do aplikácie, malo by to isté nevýhody – napríklad pre vyskúšanie zmeny v jednom dátovom súbore by bolo potrebné meniť projektový súbor a znova ho nahrávať.

Preto by mala výsledná aplikácia kombinovať oba prístupy, projektový súbor aj klasické nahrávanie dát. Pre jednoduchosť a prístupnosť by mala umožniť nahranie základných súborov klasickým spôsobom. Naopak pre pokročilejšie použitie by mala mať možnosť nahrania projektového súboru, podľa ktorého by mala nahrať všetky špecifikované dátové súbory. Potom by mal používateľ prehľadne vidieť všetky nahrané súbory a mať možnosť tieto súbory jednotlivou ľubovoľne nahradzovať inými súbormi.

Kapitola 4

Implementácia navrhnutej aplikácie

Prvou fázou implementácie bolo napísanie skriptu v jazyku Python, ktorý vykresloval mračno bodov bez použitia špeciálnych knižníc či frameworkov. K dispozícii pritom bola jedna sada ukážkových dát, ktorá sa skladala z mračna bodov, kalibračnej matice a postupnosti translácií a rotácií, ktoré predstavovali pohyb vlaku po trati. Mračno bodov obsahovalo 201 880 bodov s intenzitou od 0 do 42 a bolo definovaných 500 polôh vlaku. K tomu boli dodané obrázky, ktoré ukazovali, ako by mal vyzerať výsledok zobrazenia (obrázok 4.1).



Obr. 4.1: Referenčný obrázok, ktorý bol dodaný na začiatku práce spolu s mračnom bodov a údajmi o pohybe kamery v ňom. Obrázok je tu pre lepšiu viditeľnosť s invertovanými farbami a zväčšeným kontrastom.

Cieľom tejto prvotnej fázy bolo najmä oboznámenie sa s dátami a základnými princípmi vykreslovania bodov. Výsledky, ktoré sa nakoniec podarilo dosiahnuť, boli podobné referenčným obrázkom, aj keď nie úplne identické. Ukázalo sa, že údaje o pohybe kamery majú iný súradnicový systém ako mračno bodov (líšilo sa poradie ôs) a že je potrebné pri-

dat isté posunutie a otočenie, aby sa výsledky podobali referenčným obrázkom (posunutie kamery do stredu vlaku, podľa dodaných dát bola naľavo od stredu).

Ďalej už nasledovala práca s existujúcimi knižnicami a frameworkami v jazyku Python. Prvotným plánom bolo použiť knižnicu Pydeck (pre vizualizáciu dát) s frameworkom Streamlit (pre tvorbu GUI). Pydeck bol zvolený preto, že je nadstavbou nad javascriptovým frameworkom deck.gl, ktorý pri vykreslovaní dát využíva hardwarovú akceleráciu (GPU) a má dobrú výkonnosť. Ukázalo sa však, že Streamlit nepodporuje Pydeck v plnej miere, a teda sa nedá využiť. Preto bol namiesto Streamlitu použitý framework Dash, ktorý už mal podporu pre Pydeck dostatočnú.

Pomocou tohto frameworku bola vytvorená jednoduchá webová aplikácia zobrazujúca ukážkové dátá a umožňujúca základné nastavenia vzhľadu. Do tejto aplikácie bolo zahrnuté aj zobrazovanie kamerového záznamu, pre tento účel bolo provizórne použité video z prejazdu tej istej trate natočené v roku 2012. Pri tom sa zistilo, že vykreslovanie dát pomocou kombinácie knižníc Pydeck a Dash Deck tak, ako je to vo vzorových príkladoch v dokumentácii knižnice Dash Deck, je značne neefektívne. Jedno vykreslenie mračna bodov zaberala zhruba 3 sekundy, čo bolo spôsobené transformáciami medzi rôznymi formátmi, ako to bolo popísané v sekcii [2.4](#).

Tento čas sa podarilo významne zredukovať použitím knižnice Dash Deck bez knižnice Pydeck. Výsledok však stále neboli dosť dobrý na to, aby mohla hladko bežať animácia pohybu vlaku, a ani napriek rôznym pokusom sa ho už nepodarilo v rámci tejto kombinácie technológií vo významnejšej miere vylepšiť. Čas vykreslenia mračna bodov bol pritom úmerný počtu bodov a príčina neefektivity bola v spôsobe implementácie knižnice Dash Deck. Bolo navyše potrebné rátať s tým, že aplikácia bude zaťažená aj zobrazovaním videa. Preto bolo rozhodnuté optimalizovať vykreslovanie dát vynechaním knižnice Dash Deck a použitím frameworku deck.gl priamo z jazyka Javascript.

4.1 Optimalizácia vykreslovania mračna bodov priamym použitím jazyka JavaScript

Pythonový framework Dash interne používa na vytváranie používateľského rozhrania JavaScript, konkrétnie framework React [\[3\]](#). Jeho tvorcovia rátali s tým, že vývojári, ktorí ho budú používať, budú niekedy chcieť priamo použiť javascriptový kód. Preto to tento framework umožňuje viacerými spôsobmi:

- Klientske callbacky. Podobajú sa tým štandardným, ale na rozdiel od nich bežia iba na klientovi a píšu sa v jazyku JavaScript. Štandardné callbacky bežia na serveri a píšu sa v Pythone.
- Skripty v jazyku JavaScript, ktoré je možné pripojiť k generovanej HTML stránke. Spustia sa automaticky pri otvorení stránky v prehliadači.
- Vytváranie vlastných komponentov používateľského rozhrania pomocou frameworku React.

V tejto práci bola pre jednoduchosť využitá prvá a druhá možnosť. Bol napísaný skript `visualisation.js`, v ktorom sú definované funkcie pre zobrazenie a zmeny zobrazenia dát pomocou deck.gl. Odkazy na tieto funkcie sú potom priradené do objektu `window`, aby bolo možné volať ich z klientskych callbackov. Tento objekt teda tvorí rozhranie skriptu `visualisation.js`, ktoré je využívané zvyškom kódu aplikácie.

Pre správne pripojenie zdrojových kódov frameworku deck.gl k javascriptovému kódu tak, aby mohol fungovať v rámci aplikácie napísanej vo frameworku Dash, je nutné použiť bundler. V rámci tejto práce bol použitý bundler Webpack. Výstupnému skriptu je potrebné nastaviť príponu `.mjs`, aby ho Dash spustil ako modul.

Meranie výkonnosti vykreslovania mračna bodov pred a po optimalizácii je popísané v tabuľkách 4.1 a 4.2.

Nastavenie parametra <code>interval</code> [ms]	Čas, za ktorý prebehla celá animácia (100 snímok) [s]	Priemerný čas vykreslenia jedného snímku [ms]
1000	101	1010
800	80	800
750	76	760
700	72	720
650	71	710
600	70	700

Tabuľka 4.1: Výkonnosť vykreslovania mračna bodov obsahujúceho 201 880 bodov **pred optimalizáciou**, teda iba pomocou kódu v jazyku Python. Použité technológie: Python, Dash, Dash Deck. Z nameraných údajov vyplýva, že minimálny čas potrebný vykreslenie jednej snímky je asi 750 ms, čo znamená, že rýchlosť nedosahuje ani 2 snímky za sekundu.

Nastavenie parametra <code>interval</code> [ms]	Čas, za ktorý prebehla celá animácia (500 snímok) [s]	Priemerný čas vykreslenia jedného snímku [ms]
250	126	252
200	101	202
150	76	152
125	68	136
100	63	126
75	62	124

Tabuľka 4.2: Výkonnosť vykreslovania mračna bodov obsahujúceho 201 880 bodov **po optimalizácii**, teda s vykreslovaním dát v jazyku Javascript. Použité technológie: Python, Dash, Javascript, deck.gl. Z nameraných údajov vyplýva, že minimálny čas potrebný vykreslenie jednej snímky je asi 150 ms, čo zodpovedá rýchlosťi asi 6 snímok za sekundu.

Riadenie animácie pomocou komponentu `Interval` frameworku Dash je pomerne fažkopádne, pretože funguje na základe komunikácie medzi klientom a serverom a tým animáciu spomaľuje. Preto by bolo vhodné ho niečím nahradieť, napríklad riadením animácie v Javascripte iba na klientskej strane.

4.2 Experimenty a vyhodnotenie výsledkov

Kapitola 5

Záver

Literatúra

- [1] HARTLEY, R. a ZISSELMAN, A. *Multiple View Geometry in Computer Vision*. 2. vyd. Cambridge: Cambridge University Press, 2003. 152-156 s. ISBN 978-0-521-54051-3.
- [2] OPENJS FOUNDATION. *DECK.GL* online. 2024. Dostupné z: <https://deck.gl/>. [cit. 2025-01-14].
- [3] PLOTLY. *Dash Python User Guide* online. 2025. Dostupné z: <https://dash.plotly.com/>. [cit. 2025-01-17].
- [4] SNOWFLAKE INC.. *Streamlit* online. 2024. Dostupné z: <https://streamlit.io/>. [cit. 2025-01-17].
- [5] PRESTON-WERNER, T.; GEDAM, P. et al. *TOML v1.0.0* online. 1. novembra 2021. Dostupné z: <https://toml.io/en/v1.0.0>. [cit. 2025-04-11].
- [6] UBER ATG. *Autonomous Visualization System* online. Dostupné z: <https://avs.auto/>. [cit. 2025-04-11].