



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



164356

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
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. Presentujte 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.

Klíčové 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

24. januára 2025

Podakovanie

...

Obsah

Kapitola 1

Úvod

Kapitola 2

Vizualizácia dát z mobilných mapovacích systémov

[stručný popis, aké dáta je potrebné zobrazovať]

2.1 Dierkový model kamery – teoretický základ perspektívneho zobrazenia

Kamera je v počítačovej grafike pojem, 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 \mathbf{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 \mathbf{C} pretína obrazovú rovinu (obrázok ??). Z podobnosti trojuholníkov je možné odvodiť, že bod \mathbf{x} má súradnice $(fX/Z, fY/Z, f)^T$, postup je naznačený na obrázku ??.

Keďže 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 $(fX/Z, fY/Z)^T$.

Túto projekciu môžeme v homogénnych súradniciach zapísať pomocou násobenia matíc nasledovným spôsobom:

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

Maticu \mathbf{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 ??). Obrazom

text_prace/obrazky-figures/model_kamery1.jpg

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

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:

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

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


Maticu

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

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

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



text_prace/obrazky-figures/model_kamery2.jpg

Obr. 2.2: [TODO: prekresliť] Náčrt odvedenia súradníc bodu \mathbf{x} .

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 vedie k novému vyjadreniu projekčnej matice ako $\mathbf{P} = \mathbf{KR}[\mathbf{I} - \tilde{\mathbf{C}}]$, kde \mathbf{I} je jednotková matica 3×3 .

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

Zdrojom všetkých informácií v tejto sekcii je kniha *Multiple View Geometry in Computer Vision* od autorov R. Hartley a A. Zisserman [?].

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

Framework deck.gl a jeho nadstavba Pydeck

Javascriptový framework deck.gl je určený na zobrazovanie dát najmä na mapových podkladoch. Vyznačuje sa vysokou presnosťou a výkonnosťou. Pre akceleráciu využíva rozhrania WebGPU a WebGL2 [?].



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

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. Pre túto prácu je významná najmä vrstva **PointCloudLayer**, ktorá je určená na zobrazenie mračna bodov, a vrstva **LineLayer**, ktorá je určená na zobrazenie čiar.
- Pohľad (**View**). Definuje vlastnosti kamery, napríklad zorné pole a prednú a zadnú orezávaciu rovinu (*near plane* a *far plane*). Časť **ViewState** určuje polohu a orientáciu kamery. Typ pohľadu definuje spôsob interakcie vizualizácie s používateľom, napríklad pre zobrazenie trate z pohľadu strojvedúceho je ideálny typ **FirstPersonView**.

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

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

s veľkým množstvom dát, pretože sa aj po tej najmenšej zmene musia dáta a definícia vizualizácie nanovo prevádzať do formátu JSON a následne na javascriptový kód (obrázok ??), čo je veľmi časovo náročné.



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

2.3 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 žiaden vizuálny štýl a mnohé jeho komponenty sa priamo mapujú na HTML elementy, napríklad `dash.html.Div` a `dash.html.H1` [?][?].

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.

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 znázornené na obrázku ???. 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 je vytvoriť používateľskú aplikáciu. Výsledná aplikácia by mala byť webová, aby ju bolo možné spustiť jednoducho pomocou webového prehliadača. Mala by spĺňať nasledujúce body:

- Zobrazenie dát z mobilného mapovacieho systému. Tieto dáta sú tvorené mračnom bodov, kamerovým záznamom, vektorovými dátami a údajmi o pohybe vlaku a mali by byť zobrazené z pohľadu strojvedúceho vlaku.
- Umožniť používateľovi vybrať si konkrétnu pozíciu vlaku aj prehrať si animáciu pohybu vlaku.
- Umožniť používateľovi nahrať 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 a pole rotácií.
- Poskytnúť používateľovi možnosti prispôsobenia zobrazenia, ako napríklad zmeny viditeľnosti jednotlivých vrstiev (mračno bodov, vektorové dáta, kamerový záznam) a základné nastavenia zobrazenia mračna bodov (rôzne farebné škály podľa intenzity, veľkosť a priehľadnosť bodov, zobrazenie bodov len do určitej vzdialenosti) 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/dolava, nahor/nadol, ohnisková vzdialenosť), aby bolo možné toto zobrazenie prispôbiť 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 istej vzdialenosti pred vlakom a umožňovať prehrať animáciu jeho pohybu (podľa vektorových údajov o koľajniciach).
- Detekovať prekážky na trati.
- Umožňovať kolorizáciu mračna bodov podľa kamerového záznamu.

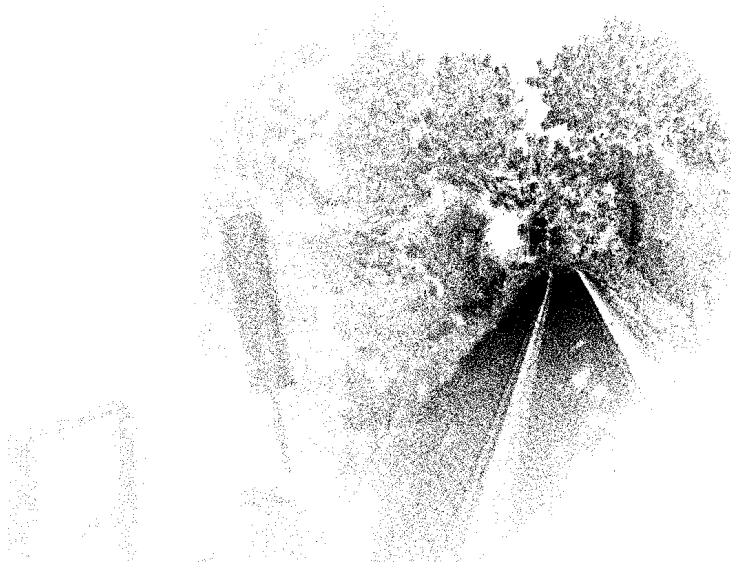
3.1 Návrh používateľského rozhrania

[návrh vrátane popisu zvolených farebných škál pre mračno bodov]

Kapitola 4

Implementácia navrhnutej aplikácie

Prvou fázou implementácie bolo napísanie skriptu v jazyku Python, ktorý vykresľoval mračno bodov bez použitia špeciálnych knižníc či frameworkov. K dispozícii pritom bola jedna sada ukázkových 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 vyzeráť výsledok zobrazenia (obrázok ??).



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 vykresľovania 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 os) a že je potrebné pri-

dať 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 vykresľovaní 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 dáta 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 vykresľovanie 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 zaberalo zhruba 3 sekundy, čo bolo spôsobené transformáciami medzi rôznymi formátmi, ako to bolo popísané v sekcii ??.

Tento čas sa podarilo významne zredukovať použitím knižnice Dash Deck bez knižnice Pydeck. Výsledok však stále nebol 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 zatažená aj zobrazovaním videa. Preto bolo rozhodnuté optimalizovať vykresľovanie dát vynechaním knižnice Dash Deck a použitím frameworku deck.gl priamo z jazyka Javascript.

4.1 Optimalizácia vykresľovania mračna bodov priamym použitím jazyka JavaScript

Pythonový framework Dash interne používa na vytváranie používateľského rozhrania JavaScript, konkrétne framework React [?]. Jeho tvorcovia ráтали 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 vykresľovania mračna bodov pred a po optimalizácii je popísané v tabuľkách ?? a ??.

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 vykresľovania 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 vykresľovania mračna bodov obsahujúceho 201 880 bodov **po optimalizácii**, teda s vykresľovaním 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 rýchlosti asi 6 snímok za sekundu.

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

4.2 Experimenty a vyhodnotenie výsledkov

Kapitola 5

Záver