VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

VIZUÁLNÍ ODOMETRIE PRO ROBOTICKÉ VOZIDLO CAR4

VISUAL ODOMETRY FOR ROBOTIC VEHICLE CAR4

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Michal Szente

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Jan Najman

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Bc. Michal Szente**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Jan Najman**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Vizuální odometrie pro robotické vozidlo Car4

Stručná charakteristika problematiky úkolu:

Práce se bude zabývat aplikací algoritmů vizuální odometrie na experimentálním vozidle Car4. Předpokládá se využití jedné nebo dvou kamer (stereovize). Součástí práce je rešerše dostupných řešení v této oblasti a otestování vybraných algoritmů.
Pro zpracování dat je možné využít nástroje Computer Vision System Toolbox a Image Processing Toolbox v programu MATLAB.

Cíle diplomové práce:

1) Rešerše v oblasti vizuální odometrie (stereo i monokulární varianta). Výhody a nevýhody jednotlivých řešení.
2) Otestování algoritmů v praxi s využitím dostupných zařízení (kamery Basler, webkamera, Xtion, …). Srovnání zejména stereo a monokulární varianty, případně různých algoritmů:
– Rychlost výpočtu a přesnost odometrie
– Závislost na osvětlení prostředí
– Závislost na typu prostředí (indoor/outdoor, členitost)
– Robustnost (např. odolnost vůči pohybujícím se objektům ve statické scéně)
3) Aplikace vybraného řešení vizuální odometrie na vozidlo Car4, běh algoritmu v realtime

Seznam doporučené literatury:

BUBÁK, M.: Aplikace stereovize a počítačového vidění. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2014.

LAUKO, M.: Zhodnocení vlastností algoritmů stereovize v nástroji Computer Vision System pro MATLAB. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015

CORKE, P.: Robotics, Vision and Control: Fundamental Algorithms in MATLAB (Springer Tracts in Advanced Robotics), Springer, 2013

LIBVISO2: C++ Library for Visual Odometry 2. Andreas Geiger [online]. 2011 [cit. 2016-11-04]. Dostupné z: http://www.cvlibs.net/software/libviso/

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Táto práca sa zaoberá algoritmami vizuálnej odometrie a aplikáciou na experimentálne vozidlo Car4. Prvá časť obsahuje rešerše v tejto oblasti, na základe ktorej je zvolený postup riešenia. Ďalšie kapitoly uvádzajú teoretický návrh algoritmov monokulárnej a stereo vizuálnej odometrie.

Tretia časť sa venuje implementácii v programe MATLAB s využitím Image processing toolboxu. Po testoch na reálnych dátach je vybraný algoritmus aplikovaný na vozidlo Car4 v praktických podmienkach interiéru a exteriéru. Posledná časť zhŕňa výsledky práce a adresuje problémy, ktoré sú spojené s aplikáciou algoritmov vizuálnej odometrie.

Kľúčové slová

SFM , vizuálna odometria, , RANSAC, Car4, odometria, lokalizácia, esenciálna matica, stereovízia, triangulácia, SVD, počítačové videnie

Abstract

This thesis deals with algorithms of visual odometry and its application on the experimental vehicle Car4. The first part contains different researches in this area on which the solution process is based. Next chapters introduce theoretical design and ideas of monocular and stereo visual odometry algorithms.

The third part deals with the implementation in the software MATLAB with the use of Image processing toolbox. After tests done and based on real data, the chosen algorithm is applied to the vehicle Car4 used in practical conditions of interior and exterior. The last part summarizes the results of the work and address the problems which are asociated with the application of visual obmetry algorithms.

Key words

SFM, visual odometry, RANSAC, Car4, odometry, localization, essential matrix, stereo vision, triangulation, SVD, computer vision

Bibliografická citácia

SZENTE, M. *Vizuální odometrie pro robotické vozidlo Car4.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 71 s. Vedoucí diplomové práce Ing. Jan Najman.

Čestné prehlásenie

Prehlasujem, že som diplomovú prácu na tému „Vizuální odometrie pro robotické vozidlo Car4“ vypracoval samostatne s použitím odbornej literatúry a prameňov uvedených v zozname, ktorý tvorí prílohu tejto práce.

V Brne dňa 25.5.2017

.....

Michal Szente

Poďakovanie

Rád by som poďakoval svojmu vedúcemu Ing. Janu Najmanovi za jeho ochotu, cenné rady a vedenie počas práce. Ďalej by som rád poďakoval prof. RNDr. Miloslavu Druckmüllerovi CSc. za motiváciu a nadšenie pre oblasť spracovania obrazu a v neposlednom rade svojej rodine a blízkym za podporu počas tvorby tejto práce.

Obsah

1 Úvod

Estimácia polohy a pohybu robotickej sústavy patrí k tým najzákladnejším a najdôležitejším úkonom pre jej úspešné autonómne riadenie a navigáciu.

Vizuálna odometria (VO) sa radí do oblasti spracovania obrazu, ktorá si kladie za cieľ estimovať trajektóriu pohybujúcej sa sústavy (v našom prípade vozidla Car4). Ide o algoritmy, ktorých vstupom je tok snímkov z jednej alebo viacerých kamier. Kamera (kamery) je pevne umiestnená na pohybujúcej sa sústave. Výstupom algoritmu VO sú transformačné matice inkrementálne určujúce polohu kamery medzi jednotlivými po sebe nasledujúcimi snímkami. Tým je zároveň určená aj poloha našej sústavy.

Hlavná výhoda VO spočíva v jej odolnosti voči prešmyku kolies v piesočnom teréne, alebo pri strate adhézie. V prípade dodržania určitých podmienok a správneho nastavenia dokáže VO prevýšiť presnosť samostatných aplikácii GPS lokalizácie a low cost inerčných senzorov. Nepotrebuje žiadne pokrytie GPS alebo rádio signálom, ani žiadne navigačné bójky. Nevýhodou tejto aplikácie je závislosť na svetelných podmienkach, závislosť od štruktúry snímanej oblasti a neustále vnášanie chyby do estimácie (drift) najmä vplyvom diskrétneho charakteru čipu kamery.

Použitie VO limitovali v minulosti hlavne jej relatívne vysoké výpočtové nároky a s tým spojená cena a kompaktnosť. S vývojom výpočtovej techniky však vzrástol záujem o túto oblasť. NASA motivovaná výhodami VO vybavila týmto systémom Rovery v misiách na Mars [1]. Dnes tvorí VO atraktívnu voľbu v oblastiach autonómnych a poloautonómnych dronov, končatinových robotov, vozidiel a automobilov. Uplatnenie nachádza taktiež v augmentovanej realite alebo 3D skenovaní.

Mechatronické laboratórium na ÚMTMB FSI VUT v Brne disponuje experimentálnym vozidlom Car4, ktoré je predmetom záverečných prác a výskumu študentov. Jedným z dlhodobých cieľov je jeho (polo)autonómne riadenie, ktoré v súčasnosti prebieha ručne. Predkladaná diplomová práca má za cieľ prispieť do oblasti lokalizácie Car4 v priestore, kam spadajú aj algoritmy VO. Predpokladané je podľa možností využitie dostupných prostriedkov laboratória.

Prvá časť tejto práce prevedie čítajúceho rešeršou k danej téme, za ktorou nasleduje presnejšie definovanie ťažiska riešenia. Druhá časť práce poskytne podrobný rozbor algoritmov VO a ich štrukturovaný návrh. Ďalšia kapitola sa venuje praktickej aplikácii a konkrétnym riešeniam jednotlivých častí. Posledná časť vedie diskusiu a rozoberá výsledky. Záver zhŕňa dosiahnuté výsledky, ku ktorým sme v tejto diplomovej práci dospeli.

2 Rešeršné štúdie

Nasledujúca kapitola je venovaná čisto rešeršnej časti. Pretože sú prakticky všetky použité zdroje cudzojazyčné (angličtina), a pretože anglické pojmy nie vždy majú prirodzený alebo známy ekvivalent v jazyku slovenskom, budú niektoré termíny preložené, prípadne doplňené o ekvivalent v anglickom jazyku podľa najlepšieho svedomia. Po podrobnejšom predstavení a uvedení miesta vizuálnej odometrie v oblasti spracovania obrazu budú algoritmy VO rozdelené podľa niekoľkých kriterií. S uvedením výhod, nevýhod a problémov jednotlivých prístupov bude možné presnejšie definovať ťažisko riešenia.

2.1 Pojem vizuálna odometria

Odraz pre rýchle zorientovanie sa v oblasti VO poskytujú publikácie [2] [3] a [4] obsahujúce štrukturované rešerše a náčrty riešení. Hlavne z týchto zdrojov vychádzajú nasledujúce časti textu.

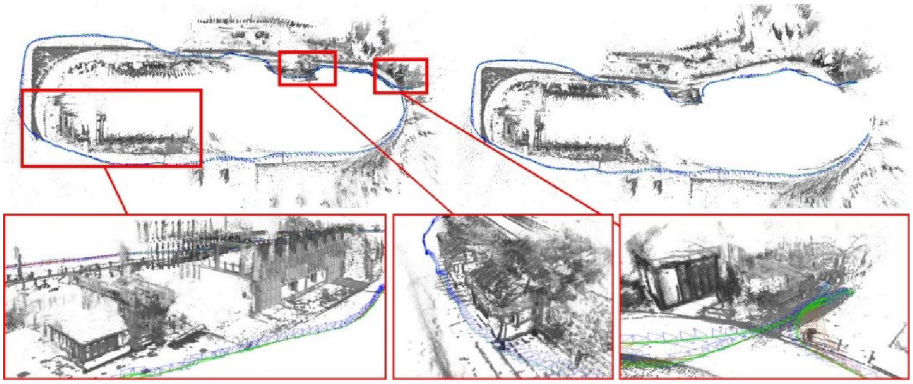
Algoritmy vizuálnej odometrie patria do skupiny všeobecnejších algoritmov, ktoré sú v literatúre spracovania obrazu známe pod pojmom SFM (Structure from motion). Medzi ne patria napríklad aplikácie ako 3D rekonštrukcia scény prípadne objektov. K tomuto sú využívané väčšinou snímky z kamier, ktoré sa pozerajú na scénu z rozdielnych miest, pričom medzi jednotlivými snímkami musí byť zaručený určitý podiel prekrytia (Obrázok 2.1). Z extrakcie významných bodov (zvaných tiež príznaky, objekty. en.: features) sú schopné rekonštruovať 3D model objektu alebo priestoru (podkapitola 2.6). Pre zlepšenie výsledku rekonštrukcie sa používa globálny optimalizačný proces zvaný bundle adjustment. Jeho aplikácia vyžaduje početne náročnú optimalizáciu mnohých parametrov a sledovanie tých istých príznakov počas viacerých snímkov.

Tieto procesy potrebujú poznať transformáciu kamery medzi snímkami. V prípade VO sa teda jedná o aplikáciu algoritmov SfM inkrementálne zisťujúcich polohu kamery pomocou extrakcie príznakov z toku snímkov. VO by sa preto dala charakterizovať aj ako určitá aplikácia SfM bez globálnej optimalizácie bundle adjustment, kvôli rýchlo meniacej sa scéne a výpočtovej náročnosti. Napriek tomu sa v literatúre zaoberajúcej sa VO pojem bundle adjustment často objavuje. Niektoré práce túto optimalizáciu úspešne využívajú napríklad v obmedzenom okne niekoľkých snímkov [5] [6].

Obrázok 2.1 SfM rekonštrukcia budovy [22]

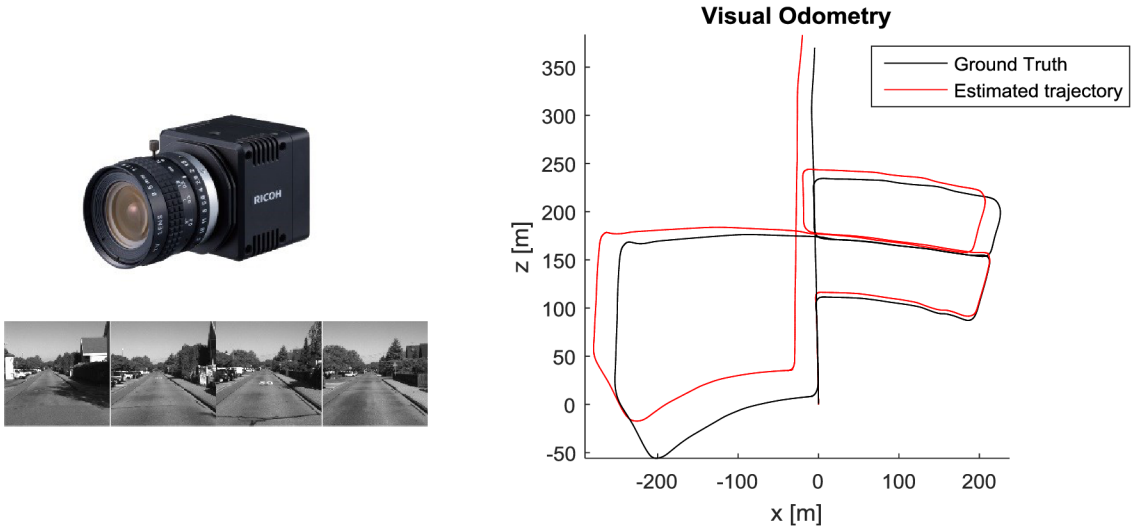
Počiatky algoritmov SfM sa datujú od 80. rokov 20. storočia, samotný pojem vizuálna odometria sa objavil až neskôr (2004). Nister ho použil vo svojej práci [7] pre svoju podobnosť s klasickou inkrementálnou odometriou. Toto meno rezonuje v mnohých prácach a štúdiách pre jeho prínos a výborné výsledky v tejto oblasti.

Ďalšou podobnou aplikáciou sú algoritmy SLAM (Simultaneous localization and mapping) známe pod pojmom Visual-SLAM (V-SLAM). Takáto aplikácia však vyžaduje pokročilejšie znalosti v oblasti spracovania obrazu a programovania, hlavne čo sa týka práce s globálnou 3D mapou. Tak isto si vyžaduje výhradne snímanie a pohyb v priestore pri ktorom dochádza k uzatváraniu trajektórie (loop closure) (Obrázok 2.2).



Obrázok 2.2 V-SLAM [23]

Samotná VO by sa preto dala charakterizovať aj ako V-SLAM bez cieľa vytvoriť globálnu mapu a bez nutnosti uzatvoriť smyčku trajektórie. Časom vznikali rôzne kombinácie alogritmov, využívajúce výhody jednotlivých riešení, avšak zvyšok tejto práce sa bude zaoberať čisto algoritmami VO ako takej (Obrázok 2.3)



Vstup
Sekvencia snímkov

Výstup
Trajektória kamery

Obrázok 2.3 Vizuálna odometria

2.2 Rozdelenie VO

Algoritmy VO možno rozdeliť podľa niekoľkých kritérií. Hlavným z nich je diferenciácia podľa typu a počtu kamier. Tie sa delia na:

- 1 monokulárna kamera
- 2 synchronizované kamery
- Kamera s 360° uhlom pozorovania
- Viac ako 2 synchronizované kamery

Nás pritom budú zaujímať algoritmy spojené s využitím jednej, alebo dvoch kamier a to z dôvodu dostupnosti hardware mechatronického laboratória a náročnosti na implementáciu. Spravidla sa algoritmus, ktorý využíva jednu kameru, nazýva *monokuárna VO* a algoritmus využívajúci dve kamery *stereo VO*.

Ďalej sa aplikácie rozlišujú podľa toho, či predpokladajú kamery kalibrované, alebo uvažujú všeobecnejší prípad bez dostupnej kalibrácie (takým by bol napríklad algoritmus na obrázku 2.1):

- Kalibrované (parametre kamery sú vopred známe)
- Všeobecné (parametre kamery nie sú známe)

Proces kalibrácie zahŕňa identifikáciu parametrov kamery (akých, to bude uvedené v neskoršej kapitole) a odstránenie zakrivenia obrazu vplyvom optiky. Takmer všetky algoritmy VO dnes počítajú s kalibrovaným prípadom. Nevýhodou tohto prístupu je potrebný čas na kalibráciu kamery. Výhodou na druhej strane je zjednodušenie algoritmu (menej neznámych parametrov) a zvýšená presnosť výsledkov. S dostupným software tretích strán sa tento proces výrazne zjednodušil.

Posledným spomenutým kritériom je princíp extrakcie informácií z obrazu:

- Sledovaním príznakov (feature-based)
- Sledovaním intenzity obrazu (appearance-based)
- Hybridné

Prvá metóda využíva opakovanú detekovateľnosť tých istých významných bodov v obraze a ich sledovanie v dvoch a viacerých snímkoch. Druhý typ využíva informácie o intenzite pixelov celého snímku. Hybridná metóda tieto prístupy kombinuje. Väčšina algoritmov v praxi využíva prvú metódu. Pre jej popularitu, dosahované výsledky, pokrytie literatúrou a dostupnosť hotových implementácií v SW sa bude práca zaoberať práve týmto prístupom.

2.3 Monokulárna VO

Všetky algoritmy počítajú rotáciu a transláciu kamery v rekonštruovanom 3D priestore. Algoritmy monovízie však túto scénu estimujú z dvojrozmernej projekcie medzi dvomi po sebe nasledujúcimi snímkami (preto sa niekedy označuje 2D-2D), medzi ktorými relatívnu polohu nepoznáme (naopak, snažíme sa ju vypočítať). To znamená, že 3D rekonštrukcia trpí stratou mierky (scale ambiguity). Podstata bude zrejmá v kapitole matematického modelu kamery. Táto nevýhoda je najzávažnejším problémom úspešnej implementácie a existuje niekoľko postupov, ako informáciu o mierke z obrazu získať:

- Detekcia objektov známej veľkosti
- Využitie algoritmov na princípe trifokálneho tenzora
- Znalosť výšky kamery nad rovinou

Prvý prístup je pre VO nepraktický, nakoľko sa musí mierka počítať v každej iterácii, je potrebné mať záber na známy objekt v každom snímku. Riešením by bolo rozmiestnenie rozlíšiteľných objektov známej veľkosti všade, kde by sa vozidlo pohybovalo.

Druhý prístup využíva samostatnú oblasť matematiky popisujúcu geometrické vzťahy medzi objektami (napr. priamky) v snímkoch. Tretí prístup využíva detekciu roviny v priestore. Estimáciou vzdialenosti kamery od roviny a jej porovnaním so skutočnou známou vzdialenosťou od roviny, je určená aj mierka scény, a tým pádom mierka pohybu.

Posledný prístup potrebuje však dva predpoklady. Musíme vopred poznať výšku kamery nad zemou a tá sa nesmie počas výpočtu meniť. Po druhé, musíme zaručiť jazdu nad relatívne plochým povrchom tak, aby mohla byť rovina z obrazu presne estimovaná.

Prvý predpoklad je v našom prípade jednoducho vyriešiteľný. Druhý predpoklad je rozumný, pokiaľ sa vozidlo bude pohybovať po lokálne rovnom teréne (cesta, chodba).

Základný algoritmus 2D-2D:

1. Získanie nového snímku I_k
2. Extrakcia a párovanie príznakov medzi snímkami I_k a I_{k-1}
3. Výpočet priestorovej rotácie R_ka translácie t_k
4. Získanie mierky s_k, aktualizácia t_k
5. Kontaktenácia transformácie
6. Opakovanie od bodu 1.

Hlavnou výhodou monokulárneho prístupu je cena techniky potrebnej k implementácii a tak isto jednoduchšie zavedenie na už existujúce systémy. Jednoduchší je taktiež kalibračný proces. Výhodu má monovízia oproti stereovízii rovnako aj v prípade, že vzdialenosť medzi kamerami stereo systému je výrazne menšia než vzdialenosť detekovaných príznakov. To vedie k degenerácii na monokulárny prípad.

2.4 Stereo VO

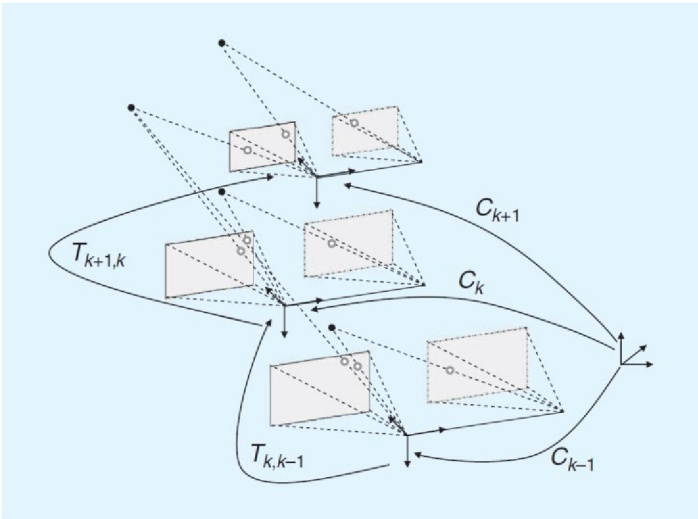
Stereovízia je v mnohom podobná zraku ľudí. Tento prístup, podobne ako ľudský mozog, dokáže estimovať aj hĺbku scény, resp. vzdialenosť objektov. Ak máme na scénu pohľad z dvoch miest a poznáme ich relatívnu vzájomnú polohu, je možné rekonštruovať 3D scénu absolútne, teda mierka scény je známa. Stereo VO sa označuje tiež ako 3D-3D algoritmus, pretože estimuje parametre pohybu z dvoch sád 3D bodov .

Počíta sa transformácia len jednej z dvoch kamier ako pohyb v rámci tohto 3D priestoru, a tým je určený aj pohyb nášho vozidla. Vyplýva z toho prvá nevýhoda, a to v prípade sledovania objektov, ktorých vzdialenosť je oveľa väčšia, než vzdialenosť medzi stereo kamerami. Tým pádom s narastajúcou vzdialenosťou bude narastať aj chyba estimovanej vzdialenosti. Druhou nevýhodou je náročnosť kalibrácie, tú však uľahčuje dostupný software (viac v kapitolách implementácie).

Základný algoritmus 3D-3D:

1. Získanie novej dvojice snímkov I_k^L, I_k^R
2. Extrakcia a párovanie príznakov medzi snímkami I_k^L, I_k^R
3. 3D triangulácia príznakov a ich spárovanie s 3D bodmi z predošlej dvojice $k-1$
4. Výpočet priestorovej rotácie R_k a translácie t_k medzi pármi 3D bodov
5. Konkatenácia transformácie
6. Opakovanie od bodu 1.

Ako vidieť hlavnou výhodou stereovízie je známa mierka a teda z princípu dokážeme estimovať pohyb kamier absolútne. Ďalšie algoritmy vznikali ako kombinácia oboch metód monokulárnej a stereo VO. Profitujú z presnosti 2D-2D metód pri estimovaní relatívneho pohybu a pre výpočet mierky využívajú informácie zo stereo kamier. To samozrejme prináša so sebou nevýhodu ceny a aplikácie stereo kamier.

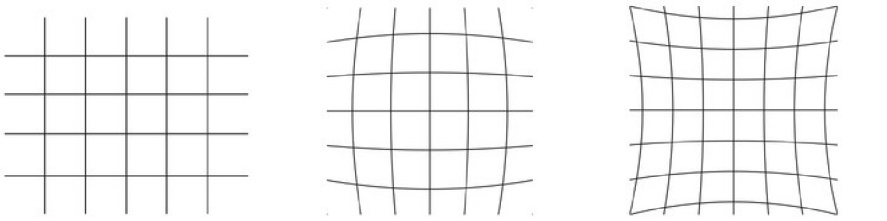


Obrázok 2.4 Ilustrácia problému stereovízie (C_k značí stred kamery, T_k tr. maticu) [2]

2.5 Kamera ako senzor

Cieľom tejto podkapitoly nie je detailné vysvetlenie princípu činnosti kamery ani rôzne konštrukčné rozdelenie. Uvedené budú problémy, s ktorými sa v spracovaní obrazu môžeme stretnúť. Z toho budú plynúť aj požiadavky pre úspešnú aplikáciu VO. Poznatky sú čerpané z prednášok prof. RNDr. Miloslava Druckmüllera, CSc., počítačových metód spracovania obrazu.

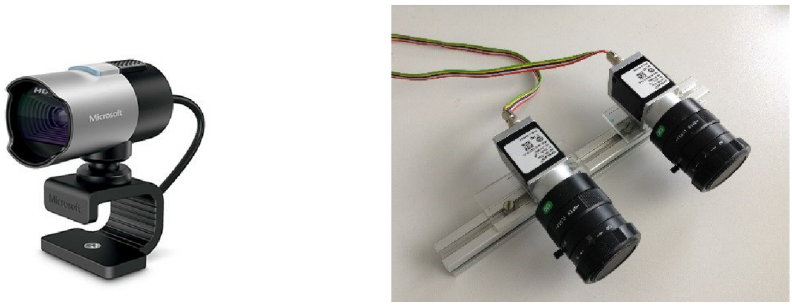
Aby mohli byť aplikované geometrické vzťahy spojené s vizuálnou odometriou, potrebujeme nahradiť kameru dierkovým modelom (en.: pinhole camera model). To v praxi znamená odstránenie zakrivenia obrazu (Obrázok 2.5), ktoré môže mať rôzny charakter a vzniká vplyvom optických členov. Informácie potrebné pre elimináciu tohto problému sú získané počas kalibrácie.



Bez zakrivenia | Súdkové zakrivenie (barrel distortion) | Vankúšové zakrivenie (pincushion dist.)

Obrázok 2.5 Zakrivenie obrazu vplyvom optiky

Ďalším problém môže predstavovať svetlosť scény, nakoľko čipy kamier využívajú svetlocitlivé súčiastky. V nedostatočne osvetlenej scéne elektronika kamery vyhodnotí prikrátk y expozičný čas a ten následne predĺži. Problém nastáva, pokiaľ sa sníma dynamická scéna a relatívne dlhým expozičným časom sa začne snímok vplyvom pohybu rozmazávať. Toto má, ako neskôr bude uvedené, negatívny dopad na detektory príznakov. V našom prípade máme k dispozícii Microsoft webcameru a Basler stereorig. Ďalšie informácie o hardware budú uvedené v časti praktickej implementácie.



Obrázok 2.6 Web kamera Microsoft a stereorig s kamerami Basler

2.6 Detekcia príznakov v obraze

Pretože potrebujeme sledovať zmenu obrazu, dôležitou časťou algoritmov VO je extrakcia informácií zo snímkov. Na to sa využívajú detektory významných príznakov. Pod pojmom príznak sa myslí identifikovateľná časť v obraze, ktorá je popísaná polohou v súradniciach obrazu a obsahuje informácie pre jej porovnanie a spárovanie s iným príznakom (deskriptor).

Tieto detektory potrebujú splniť niekoľko požiadaviek s ohľadom na stabilitu a opakovateľnosť. Potom, čo kamera podstúpi zmenu polohy v priestore, pozerá sa na scénu z iného miesta. Je preto dôležité, aby boli detekované v obraze fyzicky tie isté príznaky. Tak isto je žiadúce, pokiaľ opakovane detekujeme príznaky v nemeniacom sa obraze, aby boli extrahované vždy tie isté príznaky.

	Detektor rohov	Detektor objektov	Rotačne invariantný	Mierkovo inrvariatný	Afinne invariantný	Opakovateľnosť	Presnosť lokalizácie	Robustnosť	Efektívnosť
Harris	X		X			+++	+++	++	++
Shi-Tomasi	X		X			+++	+++	++	++
FAST	X		X	X		++	++	++	++++
SIFT		X	X	X	X	+++	++	+++	+
SURF		X	X	X	X	+++	++	++	++
CENSURE		X	X	X	X	+++	++	+++	+++

Tabuľka 2.1 Porovanie detektorov príznakov [3]

Časom vzniklo mnoho metód, stručné porovnanie je v tabuľke 2.1. Roh (en.: corner) je významná črta v obraze definovaná ako pretnutie dvoch a viacerých hrán. Objekt (en.: blob) je definovaný ako vzor v obraze, ktorý je jasne odlíšiteľný od susednej plochy, čo sa intenzity týka. V prípade, že sa pozeráme napríklad na šachovnicu, detektor by chápal body, v ktorých sa stretnú 4 štvorce, ako roh a samotné štvorce (čierne) ako objekty.

Jednotlivá invariantnosť znamená spomenutú opakovateľnosť detekcie, pokiaľ príznak v obraze podstúpi transformáciu (rotáciu, zmenu mierky, afinnú transformáciu). Dôležitá vlastnosť je tiež presnosť lokalizácie príznaku v súradniciach obrazu, ktorá je často vyžadovaná na subpixelovej úrovni. Robustnosť označuje odolnosť voči šumu, artefaktom po kompresii a rozmazaniu. Práve rozmazanie môže byť závažným problémom pre korektnú funkciu detektorov, pretože často využívajú detekciu hrán. Kedže je potrebné pre robustnosť a presnosť VO detekovať v jednom snímku niekoľko sto až tisíc príznakov, je vhodné mať detektor, ktorý je výpočtovo efektívny.

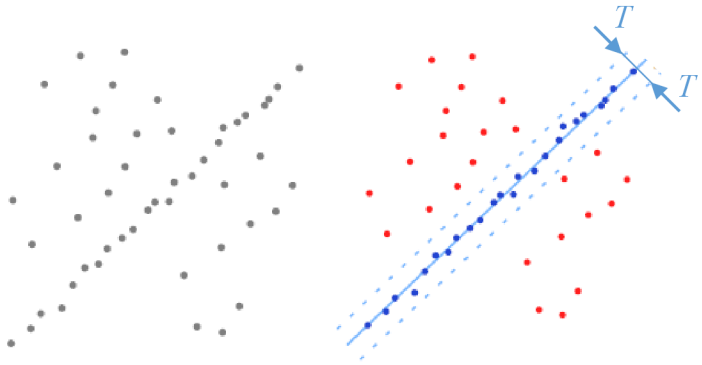
Výhodou rohov je ich výpočtová efektivita, na druhej strane sú však menej rozlíšiteľné medzi sebou v obraze s periodickým vzorom. Objekty sú výpočtovo náročnejšie na detekciu, avšak v obraze jednoznačnejšie rozlíšiteľné. Väčšina najpoužívanejších detektorov je

implementovaná v *MATLAB Image processing toolbox*. To značne urýchľuje prototyping a implementáciu (podkapitola 2.8).

Pre párovanie medzi snímkami sa používa porovnanie deskriptorov jednotlivých príznakov. Väčšinou sa jedná o určitý popis okolia príznaku formou okolitých pixelov alebo gradientov intenzity. Hľadanie prebieha lokálne alebo globálne. Lokálne hľadanie je, samozrejme, výpočtovo efektívnejšie, zlyháva však, pokiaľ došlo ku väčšiemu pohybu príznakov v obraze. Tak isto sú na deskriptory kladené požiadavky invariancie, jednoznačnosti a robustnosti. Podrobnejšie informácie sú dostupné v literatúre uvedenej na začiatku kapitoly a ďalšia diskusia bude prebiehať v kapitolách praktickej implementácie.

2.7 RANSAC

Random Sample Consensus (RANSAC) sa stal takmer nevyhnutnou súčasťou algoritmov VO. Jedná sa o metódu estimácie parametrov (v tomto prípade priamky), ktorá je dôležitá pre odstránenie nesprávne spárovaných príznakov (en.: outlier rejection) a vnáša do estimácie pohybu zásadný stupeň robustnosti..



Obrázok 2.7 Ilustrácia metódy RANSAC [24]

Je potrebné predstaviť jeho základný princíp, pretože sa s jeho implementáciou stretneme niekoľkokrát počas riešenia. Myšlienka tohto algoritmu je veľmi jednoduchá a pritom vysoko účinná. Uvedenie algoritmu v jeho forme pre VO nie je úplne jednoznačné na pochopenie. Nasledovný algoritmus bude preto vysvetlený na probléme prekladania bodmi priamkou.

Predstavme si namerané body na obrázku 2.7 vľavo, ktoré sa snažíme preložiť priamkou metódou najmenších štvorcov. Vidieť, že dáta obsahujú aj zjavne nesprávne body nepatriace do množiny bodov, ktoré nás zaujímajú. Pokiaľ by sme počítali parametre priamky zo všetkých bodov, výsledok by bol prinajlepšom nepresný, je však možné, že by trend bodov, ktoré nás zaujímajú, vôbec nesledoval. Preto použijeme nasledovný algoritmus, ktorý sa pokúsi body obsahujúce veľkú chybu odstrániť:

1. Náhodne vyberieme 2 body, z ktorých určíme parametre priamky
2. Spočítame vzdialenosti všetkých bodov od priamky
3. Zistíme počet bodov, ktorých vzdialenosť je menšia ako stanovená prahová hodnota T

4. Ak je počet bodov väčší než predošlý počet vyhovujúcich bodov, uložíme tieto body
5. Opakujeme od kroku 1 N krát

Parametrom algoritmu je počet iterácií N a to je aj jeho nevýhoda. Číslo N musí byť dostatočne veľké na to, aby sa v náhodnom procese výberu mohli objaviť 2 body, z ktorých priamka vyhovuje väčšine bodov. Tento algoritmus ďalej potrebuje poznať odhad hraničnej hodnoty T (akú vzdialenosť od správnej priamky sme ochotní tolerovať) a mať metriku na výpočet chybovosti bodu (vzdialenosť bodu od priamky). Nakoniec sa parametre vypočítajú pôvodnou metódou najmenších štvorcov. Vidieť, že takýto výsledok nebral do úvahy body zaťažené veľkou chybou, nevýhodou môže byť nedeterministický výsledok. Ďalším negatívom je zvýšenie časovej náročnosti na výpočet pri vysokom počte iterácií.

Tento algoritmus má však oproti tomu jednu zásadnú výhodu. Okrem odstraňovania párov príznakov, ktoré boli chybne spárované, odstraňuje príznakay resp. páry, ktoré nepodstúpili rovnakú transformáciu ako väčšina scény. To znamená, že estimácia pohybu pomocou algoritmu RANSAC je odolná voči pohybu neželaných predmetov, ktoré by inak viedli ku skresleným výsledkom. To smeruje ku značnému zvýšeniu robustnosti.

2.8 Software

Pre návrh a aplikáciu algoritmov VO bol zvolený software MATLAB. Jeho hlavnou výhodou je prostredie pre rýchle prototypovanie, debugging a odladenie. Nevýhodou oproti C/C++ môže byť pomalší výkon.

Image processing toolbox ponúka od verzie 2015b aj niektoré časti algoritmov VO. Trendom Mathworks je postupná implementácia algoritmov SFM a dopĺňanie tým palety použiteľných funkcií. Tak isto pribúdajú rôzne demá, ako je možné jednotlivé funkcie využiť. Dostupný je taktiež často používaný nástroj kalibrácie kamier. Táto práca uvažuje použitie verzie MATLAB 2015b nakoľko na palubnom počítači vozidla Car4 beží 32-bit operačný software a táto verzia MATLABu je posledná, ktorá takýto systém podporuje.

3 Analýza

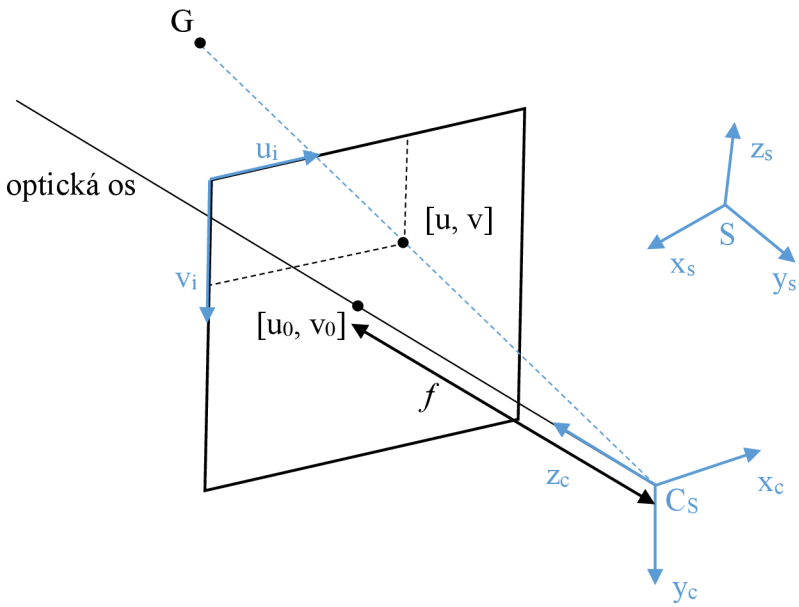
VO nie je triviálny algoritmus a obsahuje mnoho problémov, s ktorými treba počítať. Väčšinou je nutné zvoliť určitý kompromis medzi metódami. S ohľadom na pôvodné cieľe práce, dostupný hardware a software, v ktorom sa bude VO implementovať, bol zvolený nasledovný postup:

1. Teoretický rozbor a návrh algoritmov monokulárnej a stero VO založených na princípe sledovania príznakov
2. Implementácia algoritmov v programe MATLAB
3. Test algoritmov na reálnych dátach
4. Test algoritmov na dostupnom hardware pre potreby vozidla Car4

4 Teoretické riešenie

Následujúce kapitoly prevedú čítajúceho teoretickou podstatou algoritmov VO. Človek, ktorý nie je odborníkom v oblasti spracovania obrazu, sa môže v literatúre stretnúť s nejednoznačnými formuláciami, prípadne chýbajú dôležité poznámky potrebné pre praktickú implementáciu. Uvedené budú nevyhnutné matematické vzťahy doplnené o komentáre a poznatky nadobudnuté pri práci s nimi. Táto kapitola uvádza vždy odkazy na konkrétnu literatúru obsahujúcu podrobnejšie informácie.

4.1 Projekčný model kamery



Obrázok 4.1 Súradnicové systémy

Na obrázku 4.1 je zobrazená 2D projekcia sledovaného bodu do súradníc snímku. Bod S značí stred pravotočivých svetových súradníc, v ktorých je vyjadrený pohyb kamery. Bod C_S značí stred súradníc kamery umiestnených v optickom strede kamery, jeho poloha je vyjadrená vo svetových súradniciach.

Osi sú orientované tak, že os z_c leží na priamke optickej osi a má kladný smer z pohľadu kamery sledujúcej scénu, os x_c smeruje vpravo a os y_c tak, aby tvorili pravotočivý súradnicový systém. Ďalej u_i a v_i sú osi súradnicového systému snímku, kde počiatok je orientovaný v ľavom hornom rohu snímku. Bod G, $G_C = [X_c, Y_c, Z_c]^T$ je 3D poloha sledovaného príznaku v súradniciach kamery a $G_S = [X_S, Y_S, Z_S]^T$ jeho poloha vo svetových súradniciach .

Označme $q = [u, v, 1]^T$ 2D projekciu príznaku v súradniciach obrazu, ktorú vypočítame z perspektívneho modelu kamery v tvare:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KG_C = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{4.1}$$

kde f značí ohniskovú vzdialenosť, λ je faktor hĺbky(Z) a u_0, v_0 sú súradnice projekčnej osi v súradniciach obrazu.

Matica K sa nazýva **kalibračná matica** a obsahuje vnútorné parametre kamery (en.: intrinsic parameters). Pokiaľ sú všetky členy matice K známe, hovoríme, že kamera je kalibrovaná . Práve na určenie týchto parametrov slúži kalibračný proces a drvivá väčšina algoritmov VO počíta s kalibrovaným prípadom. Polohu bodu G v súradniciach kamery získame z rovnice:

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = P_{ex} G_S = [R \mid t] \cdot \begin{bmatrix} X_S \\ Y_S \\ Z_S \end{bmatrix} \tag{4.2}$$

kde P_{ex} značí 3x4 maticu vonkajších parametrov kamery (en.: extrinsic parameters), ktoré obsahujú informácie o polohe kamery v priestore. R je 3x3 rotačná matica a t 3x1 vektor translácie $t = -RC_S$. Ak celý proces zlúčime, dostávame:

$$\lambda q = KP_{ex} G_S = PG_S \tag{4.3}$$

kde 3x4 maticu P nazývame **projekčná matica** kamery.

V tomto momente je jasnejší princíp VO. Ide o inverznú metódu, ktorá využíva 2D projekciu bodov (ich zmenu) na výpočet transformácie kamery v priestore. Je vhodné uviesť, že pokiaľ stred svetových súradníc S umiestnime do stredu kamery C_S, P_{ex} bude nulová matica s jednotkami na hlavnej diagonále. [8]

4.2 Triangulácia

Pre dve dané projekčné matice P_{k-1} a P_k , ktoré popisujú projekciu dvoch kamier alebo jednej kamery, ktorá sa pohla, chceme vypočítať 3D súradnice korešpondujúceho bodu G z 2D páru príznakov v snímku I_k a I_{k-1}. Projekčné matice majú tvar:

$$\begin{aligned} P_{k-1} &= K_1[I|0] \\ P_k &= K_2 P_{ex}^k \end{aligned} \tag{4.4}$$

Zostavíme maticu A:

$$A = \begin{bmatrix} u_{k-1}P_{k-1}^3 - P_{k-1}^1 \\ v_{k-1}P_{k-1}^3 - P_{k-1}^2 \\ u_k P_k^3 - P_k^1 \\ v_k P_k^3 - P_k^2 \end{bmatrix} \tag{4.5}$$

kde horný index značí riadok projekčnej matice. Súradnice bodu G v tvare $[a, b, c, d]^T$ sú prvky v poslednom stĺpci matice V z operácie $[U, D, V] = svd(A)$, kde *svd* (singular value decomposition) je operácia pri ktorej platí $A = UDV^T$. Posledným krokom je podelenie súradníc súradnicou *d* tak, aby bola poloha 3D bodu v tvare $X = [x, y, z, 1]^T$.

Triangulácia v prípade kalibrovanej stereokamery

Geometricky za kalibrovanú stereokameru považujeme pre naše potreby dve kamery paralelne umiestnené vedľa seba, poznáme ich kalibračné matice K_1, K_2 a vzdialenosť medzi nimi $B[m]$. Polohu 3D bodu $G = [x, y, z, 1]^T$ vypočítame ako [9]:

$$d\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = Q\begin{bmatrix} u_L \\ v_L \\ \delta \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -u_0 \\ 0 & 1 & 0 & -v_0 \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{1}{B} & 0 \end{bmatrix}\begin{bmatrix} u_L \\ v_L \\ \delta \\ 1 \end{bmatrix} \tag{4.6}$$

kde u_L, v_L sú súradnice príznaku v snímku a u_0, v_0, f sú vnútorné parametre ľavej kamery. Člen δ sa nazýva disparita a vypočítame ju ako $\delta = u_L - u_R$, kde u_R je u-súradnica toho istého príznaku v pravom snímku. Matica Q bola upravená pre náš súradnicový systém [10].

4.3 Esenciálna matica

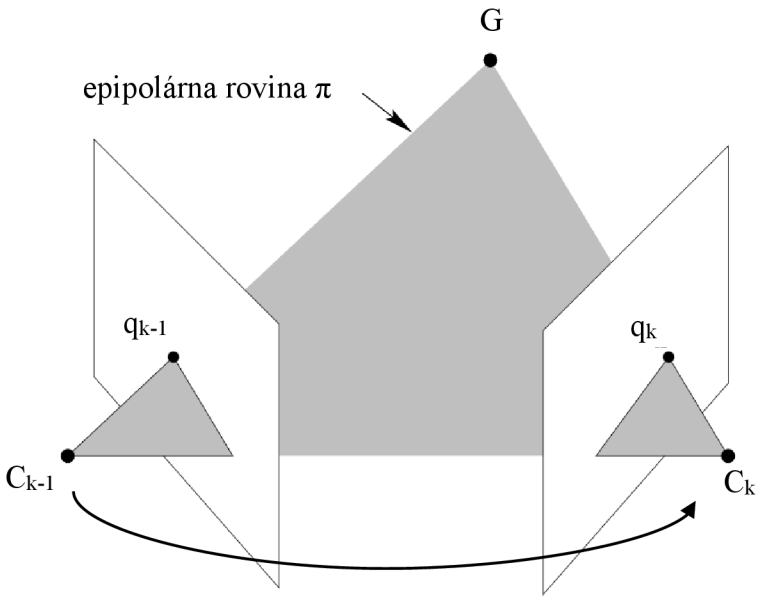
Pozorujme ten istý bod G v scéne, pomocou jednej kamery, ktorá sa pohla, alebo pomocou dvoch kamier. Majme pár príznakov, ktoré sú 2D projekciou tohto bodu so súradnicami $q_{k-1} = [u_{k-1}, u_{k-1}, 1]^T$ v snímku I_{k-1} a $q_k = [u_k, u_k, 1]^T$ v snímku I_k, kde snímok k bol vytvorený druhou kamerou, alebo tou istou kamerou po zmene polohy. Pre 2D projekcie bodu G musí platiť podmienka epipolarity (en.: epipolarity constraint). Pre nekalibrovaný prípad má tvar

$$q_k^T F q_{k-1} = 0 \tag{4.7}$$

kde F je 3x3 fundamentálna matica.Táto podmienka vyjadruje pozorovanie, že ten istý bod sa v druhom snímku zobrazí do bodu, ktorý leží v epipolárnej rovine π (Obrázok 4.2). Ak zavedieme $\tilde{q}_{k-1} = K_1^{-1} q_{k-1}$ a analogicky $\tilde{q}_k = K_2^{-1} q_k$, kde $K_1^{-1} K_2^{-1}$ sú inverzie známych kalibračných matíc kamery 1 a 2 ($K_1 = K_2$ pre jednu kameru, ktorá sa pohla), potom sa podmienka dá napísať v tvare pre kalibrovaný prípad:

$$\tilde{q}_k^T E \tilde{q}_{k-1} = 0 \tag{4.8}$$

kde E je 3x3 esenciálna matica a $\tilde{q}_{k,k-1}$ sú normalizované súradnice príznakov.



Obrázok 4.2 Ilustrácia podmienky epipolarity [8]

Esenciálna matica podobne ako fundamentálna obsahuje informácie o transformácii kamery z bodu C_{k-1}do bodu C_k. V prípade, že bod C_{k-1} umiestnime do počiatku súradnicového systému a súhlasne orientujeme osi, inak povedané, pokiaľ $P_{k-1}^{ex} = [I, 0]$ (kde I je 3x3 jednotková matica a 0 je 3x1 nulový vektor) potom matica E bude obsahovať informácie o vonkajších parametroch kameryP_k^{ex} v bode C_k.

$$E \cong R\hat{t} \tag{4.9}$$

kde znamienko ≅ značí rovnosť len do určitého násobku. Pri počítaní rotácie a translácie z esencálnej matice musíme zistiť tento násobok dodatočne. R značí 3x3 rotačnú maticu a $\hat{t}$ značí 3x3 antisymetrickú maticu translácie v tvare:

$$\hat{t} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{4.10}$$

Je užitočné uviesť z dôvodu kontroly výpočtu podmienky (4.11) a (4.12), ktoré musí každá korektná esenciálna matica spĺňať [11]:

$$2EE^TE - trace(EE^T)E = 0 \tag{4.11}$$

$$\det(E) = 0 \tag{4.12}$$

4.4 8-bodový algoritmus

Prvým krokom ku zisteniu transformácie kamery, je estimácia matice E z detekovaných párov príznakov v snímkoch. Podmienku epipolarity (4.8) je možné prepísať do tvaru:

$$\tilde{q}e = 0 \tag{4.13}$$

kde $\tilde{q} = \tilde{q}_k^T \tilde{q}_{k-1}$ značí 1x9 vektor v tvare po rozpísaní $\tilde{q} = [u_{k-1}u_k\,, v_{k-1}u_k\,, \dots\,, v_{k-1}.1\,, 1.1]$ a $e = [E_{11}, E_{12}, \dots, E_{33}]^T$ je 9x1 vektor s jednotlivými prvkami esenciálnej matice. Neznámy vektor e vypočítame z ôsmich (preto názov 8-bodový) párov $(a, b, \dots h)$ príznakov formovaním:

$$Ae = 0 \tag{4.14}$$

kde A je 8x9 matica v tvare:

$$A = \begin{bmatrix} \tilde{u}_{k-1}^a \tilde{u}_k^a & \tilde{v}_{k-1}^a \tilde{u}_k^a & \dots & 1.1 \\ \vdots & \ddots & \ddots & \vdots \\ \tilde{u}_{k-1}^h \tilde{u}_k^h & \tilde{v}_{k-1}^h \tilde{u}_k^h & \dots & 1.1 \end{bmatrix} \tag{4.15}$$

Pomocou operácie SVD vypočítame rozklad matice A na $[U_A D_A V_A] = svd(A)$. Estimovaný vektor prvkov e nájdeme ako posledný stĺpec matice V_S, tým pádom poznáme estimovanú maticu E_{est}. Aby však bola esenciálna matica určená korektne, potrebujeme výsledok upraviť nasledovne:

$$[U_E D_E V_E] = svd(E_{est}) \tag{4.16}$$

Vykonáme operáciu SVD matice E_{est} a vytvoríme novú esenciálnu maticu:

$$E = U_E diag(1,1,0) V_E^T \tag{4.17}$$

kde $diag(1,1,0)$ značí 3x3 nulovú maticu s prvkami v zátvorke na diagonále. V tomto kroku sa literatúra rozchádza a odporúča vytvoriť túto maticu aj ako $diag(s, s, 0)$ alebo ako $diag(s + r/2\,,\ s + r/2\,,\ 0\,)$ kde s a r sú prvé dva prvky na diagonále matice D_E. Podmienky (4.11) a (4.12) však spĺňal jedine postup (4.17).

Ďalej je potrebné vypočítať $[U, D, V] = svd(E)$. V prípade, že $det(UV^T) = -1$, je nutné prehodiť znamienka posledného stĺpca matice V a formovať novú esenciálnu maticu podľa vzoru (4.17). Týmto sme vypočítali korektnú esenciálnu maticu E.

Pokiaľ použijeme v predošlom postupe viac ako 8 párov príznakov, úloha je predeterminovaná a vedie k riešeniu najmenších štvorcov [8] [12]. V tomto bode je dôležité zdôrazniť potrebu normovania súradníc príznakov. Aj v prípade, že je použitý 8-bodový algoritmus pre všeobecný prípad (estimácia fundamentálnej matice), je vhodné použiť pred výpočtom normalizáciu, a to z dôvodov, ktoré sú jasné pri pohľade na tvorbu vektora $\tilde{q}$ (4.13).

V prípade niektorých členov môže dôjsť k násobeniu nepomerne veľkých čísel. Týmto dostaneme členy rádovo v jednotkách až po 10^9 v prípade riešenia najmenších štvorcov, čím sa zavádza do výpočtu numerický nepomer a nestabilita.

Zhrnutie 8-bodového algoritmu

1. Extrahujeme 8 a viac párov príznakov z dvoch snímkov
2. Sformujeme maticu A podľa (4.15)
3. Rozložíme maticu A pomocou SVD
4. Nájdeme esenciálnu maticu ako posledný stĺpec matice U z bodu 3.
5. Posilníme vnútornu podmienku podľa (4.17)
6. V prípade, že $det(UV^T) = -1$ prepočítame príslušne maticu E

5-bodový algoritmus

8-bodový algoritmus má negatívnu vlastnosť, a tou je degenerácia v prípade, že body, z ktorých počítame maticu E, sú koplanárne – teda ležia v jednej priestorovej rovine.

Pre výpočet esenciálnej matice sa dnes používa taktiež 5-bodový algoritmus [11]. Jeho výpočet nie je celkom priamočiary, má však výhodu oproti predošlému postupu, že nedegeneruje pri použití koplanárnych bodov a využíva početne efektívnejšiu metódu. To je dôležité najmä v algoritmoch RANSAC. Tento spolu s 8-bodovým algoritmom je implementovaný vo funkcii pre výpočet fundamentálnej matice v software MATLAB 2015b.

4.5 Výpočet esenciálnej matice pomocou RANSAC

V prípade, že by sme na výpočet esenciálnej matice použili všetky páry príznakov v snímkoch, výsledok by bol skreslený tak ako bolo naznačené v podkapitole 2.7. Metódou najmenších štvorcov (8-bodový algoritmus) by sme aproximovali 2 typy párov, ktoré sú nežiadúce. Prvým sú chybne spárované príznaky a druhým príznaky, ktoré patria telesám konajúcim neželaný pohyb v scéne (ľudia, zvieratá, autá atď..). Pre ich odstránenie je dnes využívaný algoritmus RANSAC. Pre VO má nasledovnú schému:

1. Náhodne vyberieme 5 párov príznakov
2. Vypočítame odhad esenciálnej matice pomocou 5-bodového algoritmu
3. Vypočítame Sampsonovu chybu všetkých párov podľa (4.18)
4. Ak je chyba menšia než prahová hodnota T, označíme pár za patriaci do množiny správnych párov a spočítame ich počet pre daný odhad esenciálnej matice
5. Body 1-4 opakujeme N krát a vyberieme množinu, ktorá obsahuje najviac párov
6. Vypočítame esenciálnu maticu z týchto párov pomocou 8-bodové algoritmu

Aby sme mohli odhadnúť chybu páru pre danú esenciálnu maticu, využijeme Samsonovu vzdialenosť, ktorá aproximuje chybu pri reprojekcii Taylorovym polynómom prvého stupňa a má tvar pre pár *i* [13]:

$$\varepsilon_i = \frac{\tilde{q}_k^T E \tilde{q}_{k-1}}{\sqrt{(\tilde{q}_k^T E)_1^2 + (\tilde{q}_k^T E)_2^2 + (E\tilde{q}_{k-1})_1^2 + (E\tilde{q}_{k-1})_2^2}} \tag{4.18}$$

kde $(\tilde{q}_k^T E)_1^2$ (a analogicky ďalšie členy v menovateli) je druhá mocnina prvého prvku vektora $\tilde{q}_k^T E$ a $\tilde{q}_{k,k-1}$ sú normalizované súradnice i-teho príznaku v snímku *k* a *k-1*. Prahová chyba *T* je volená a rozumná hodnota môže byť 0.002-0.001, ktorá odpovedá nenormovanej hodnote 1 pixel. Inak povedané, pokiaľ pár nespĺňa predpokladanú polohu v obraze o viac ako 1 pixel, je označený za nepatriaci do množiny správnych párov pre daný model matice *E*. Tak isto volíme parameter *N*, teda počet cyklov algoritmu tak, aby bolo s dostatočne vysokou pravdepodobnosťou zaručené, že najväčšia skupina párov (ich pohyb v rámci snímkov) reprezentuje reálny pohyb kamery.

4.6 Rozklad esenciálnej matice

Pokiaľ sme vypočítali korektnú esencialnu maticu z predošlých podkapitol, ďalším krokom bude rozložiť esenciálnu maticu na zložky rotácie a translácie.

Majme matice U, D, V z posledného kroku podkapitoly 4.4. Matematicky vyhovujú rovnici (4.9) 4 riešenia. Dve pre rotačnú maticu:

$$R_a = UWV^T \tag{4.19}$$
$$R_b = UW^TV^T \tag{4.20}$$

a dve možné znamienka pre vektor translácie

$$t = \pm [\hat{t}_{32}, \hat{t}_{13}, \hat{t}_{21}]^T \tag{4.21}$$

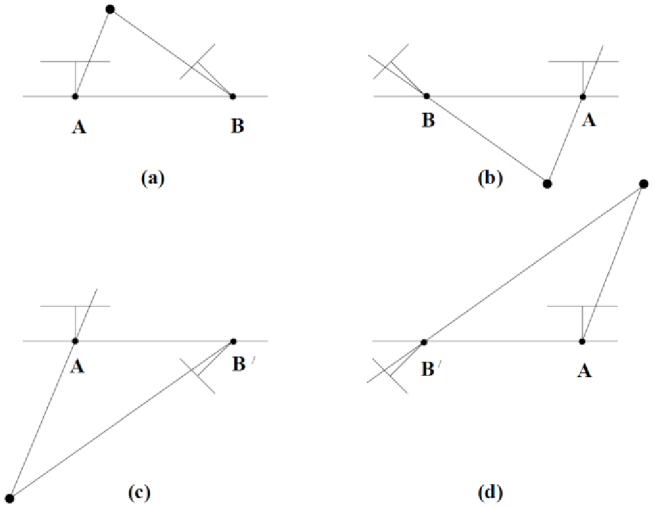
kde $\hat{t} = UZU^T$ a

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.22}$$

Z rovníc (4.19)-(4.21) tak dostávame 4 možnosti matice P_{ex} :

$$\begin{aligned} P_{ex}^1 &= [R_a| + t] \\ P_{ex}^2 &= [R_a| - t] \\ P_{ex}^3 &= [R_b| + t] \\ P_{ex}^4 &= [R_b| - t] \end{aligned} \tag{4.23}$$

Grafická interpretácia riešení (4.23) je na obrázku 4.3.



Obrázok 4.3 Grafická ilustrácia 4 riešení dekompozície matice E [8]

Napriek tomu, že ide o riešenia matematicky korektné, zmysluplné je len jedno z nich a síce to, ktoré predstavuje obe kamery majúce sledovaný bod pred sebou (Obrázok 4.3 a)). To znamená, že zostavené matice (4.23) musíme testovať s triangulovaným bodom. Ak tento bod leží pred oboma kamerami, ide o hľadanú maticu P_{ex}. Postup je nasledovný:

1. Vyberieme 1 pár príznakov
2. Triangulujeme tento pár do 3D bodu pomocou metódy *svd* (podkapitola 4.2) s využitím všetkých matíc (4.23)
3. Vypočítame hĺbku 3D bodu v oboch kamerách podľa [8](s.162)
4. Správna matica P_{ex} je tá, pri ktorej obe hĺbky majú kladné znamienko

Je nutné ujasniť reprezentáciu výsledkov z predošlých odstavcov. Výsledkom je matica P_{ex}^{k} obsahujúca vonkajšie parametre kamery po pohybe, resp. kamery č. 2. Matica vonkajších parametrov kamery pred pohybom má tvar $P_{ex}^{k-1} = [I|0]$. Transformačnú maticu, ktorú hľadáme, vypočítame ako [14]:

$$T_{k-1,k} = \begin{bmatrix} (P_{ex}^{k})^{-1} \\ 0\ 0\ 0\ 1 \end{bmatrix} \tag{4.24}$$

4.7 Estimácia mierky

Transformačná matica (4.24), resp. translačná zložka tejto matice potrebuje pre absolútny popis transformácie $[m]$ známu mierku. Pre jej výpočet bude uvedený postup založený na estimácii výšky kamery nad zemou H_{est} [15]. Prvým krokom je detekcia roviny pod vozidlom. Predpokladáme, že poznáme n (kde $n \geq 3$) triangulovaných 3D bodov $X_n = [x_n, y_n, z_n]^T$, ktoré patria do roviny pod vozidlom, na ktorú má kamera dostatočný záber a poznáme skutočnú výšku kamery nad zemou H_{real}. Pre bod patriaci do roviny platí:

$$a(x - x_c) + b(y - y_c) + c(z - z_c) = 0 \tag{4.25}$$

kde (a, b, c) je normálový vektor roviny a $[x_c, y_c, z_c]^T$ je bod ležiaci v rovine. V tomto prípade sa jedná o ťažisko bodov X_n , ktoré sa snažíme preložiť rovinou. Súradnice ťažiska vypočítame ako aritmetický priemer jednotlivých súradníc bodov. Sformujeme maticu A 3xn nasledovne:

$$A = \begin{bmatrix} (x_1 - x_c) & (x_n - x_c) \\ (y_1 - y_c) & \dots (y_n - y_c) \\ (z_1 - z_c) & (z_n - z_c) \end{bmatrix} \tag{4.26}$$

Hľadané parametre (a, b, c) nájdeme ako posledný stĺpec matice U po operácii $[U, D, V] = svd(A)$. Rovnicu roviny prepíšeme do tvaru:

$$ax + by + cz + d = 0 \tag{4.27}$$

kde $d = -ax_c - by_c - cz_c$. [16]

Estimovaná výška kamery nad zemou H_{est}, je vzdialenosť bodu $[0,0,0]^T$ (teda poloha stredu kamery C_{k-1}) od roviny, ktorú sme práve vypočítali. Potrebnú mierku s vypočítame ako

$$s = \frac{H_{real}}{H_{est}} \tag{4.28}$$

4.8 Estimácia pohybu stereokamery

Predpokladajme dve množiny 3D bodov X_{k-1} a X_k tvoriacich páry, triangulovaných podľa (4.6) zo snímkov stereokamier I_{k-1}^L, I_{k-1}^R a I_k^L, I_k^R, ktoré sa pohli. Z pohľadu kamery body X_k voči X_{k-1} vykonali pohyb. Našim cieľom je zistiť transformáciu, ktorá popisuje zmenu polohy týchto párov a tým pádom aj relatívnu polohu kamier. Pohyb uvažujeme vždy relatívne ku ľavej kamere.

Snažíme sa nájsť rotáciu R a transláciu t pre všetky páry i=*1..n* takú, že

$$X_k^i = RX_{k-1}^i + t \tag{4.29}$$

Tieto páry však budú zaťažené šumom. Preto je cieľom nájsť parametre pohybu tak, že bude minimalizovaný MSE (mean square error) pre všetky body:

$$\epsilon_{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left\|X_k^i - (RX_{k-1}^i + t)\right\|^2 \tag{4.30}$$

Vytvoríme 3x3 kovariančnú maticu:

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(X_k^i - \mu_k)(X_{k-1}^i + \mu_{k-1})^T \tag{4.31}$$

kde n je počet párov, $\mu_k = [x_c, y_c, z_c]^T$ a analogicky μ_{k-1} sú súradnice centroidov pre body zo snímkov k a k-1. Člen $(X_k^i - \mu_k)$ má rozmer 3xn a $(X_{k-1}^i + \mu_{k-1})^T$ *nx3*. Keď vypočítame rozklad kovariančnej matice na matice $[U, D, V] = svd(\Sigma)$, rotačnú zložku vypočítame ako:

$$R = USV^T \tag{4.32}$$

kde $S = I$ je 3x3 jednotková matica a pokiaľ je $\det(U) \cdot \det(V) = -1$, člen v poslednom stĺpci a riadku matice je $S_{33} = -1$. Následne translačnú zložku vypočítame:

$$t = \mu_k - R^T\mu_{k-1} \tag{4.33}$$

Uvedený postup vychádza z [9][17]. Netreba zabúdať, že vypočítaná transformácia popisuje pohyb bodov a nie pohyb kamery, ktorý je inverzný k nemu (4.24).

Zhrnutie estimácie pohybu stereokamery

1. Extrahujeme 3D páry bodov zo stereokamier pomocou (4.6)
2. Vypočítame centroid pre každú sadu
3. Sformujeme kovariančnú maticu Σ podľa (4.31)
4. Rozložíme Σ pomocou operácie SVD
5. Vypočítame rotáciu R (4.32) a transláciu t (4.33)

Pohyb stereokamery a RANSAC

Pozorne čítajúci si všimne slabú stránku postupu z predošlých odstavcov. Predpokladom pre jeho správnu funkciu je bezchybné párovanie príznakov ako medzi ľavým a pravým snímkom (2 krát), tak medzi 3D pármi. Taktiež si tento postup neporadí s neželaným pohybom v scéne. Využitím metódy RANSAC, podobne ako bolo uvedené v podkapitole 4.5, je možné tieto problémy odstrániť v jednom kroku.

1. Náhodne vyberieme 3 páry 3D bodov
2. Vypočítame odhad pohybu (R_{est}, t_{est}) podľa podkapitoly 4.8
3. Spočítame chybu každého 3D bodu pre odhad z bodu 2 podľa (4.34)
4. Ak je chyba menšia než prahová hodnota *T,* označíme bod za patriaci do množiny správnych pre daný odhad
5. Body 1-4 opakujeme N-krát a vyberieme množinu s najväčším počtom bodov
6. Spočítame transformáciu množiny podľa podkapitoly 4.8

kde chybu bodu vypočítame ako vzdialenosť transformovaného bodu $X_k^{est} = R_{est}X_{k-1} + t_{est}$ voči skutočnej polohe spárovaného bodu X_k podľa (4.34) a prahovú hodnotu T volíme ako odhad prípustnej chyby v $[m]$.

$$r_i = \sqrt{(x_k^{est} - x_k)^2 + (y_k^{est} - y_k)^2 + (z_k^{est} - z_k)^2} \tag{4.34}$$

4.9 Konkatenácia transformačných matíc

Pri iteračnom výpočte polohy kamery C_k sa skladajú jednotlivé transformácie nasledovne:

$$\begin{aligned} R_k &= R_{k-1}R_{est} \\ t_k &= t_{k-1} + s_k R_k t_{est} \end{aligned} \tag{4.35}$$

$$C_k = T_k \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} R_k & t_k \\ 0\ 0\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.36}$$

kde s_k je mierka a $s_k \neq 1$ pokiaľ ide o monokulárnu VO.

4.10 Algoritmy VO

Táto podkapitola uvádza štrukturované osnovy jednotlivých algoritmov monokulárnej a stereo VO, ktoré sú založené na predošlých podkapitolách 4.1-4.9. V prípade hybridnej VO sa jedná o monokulárny algoritmus, ktorý rieši problém s mierkou (podkapitola 4.7) pomocou absolútnych informácií o hĺbke z 3D triangulovaných bodov zo sterevízie (značí skratka 3D-2D). Uvedené algoritmy budú v nasledujúcich častiach tejto práce implementované v programe MATLAB.

Algoritmus monokúlarnej VO 2D-2D

1. Detekujeme príznaky v snímkoch I_{k-1} , I_k a spárujeme ich
2. Vypočítame esenciálnu maticu E (podkapitola 4.5)
3. Rozložíme maticu E na rotáciu a transláciu (podkapitola 4.6)
4. Vypočítame mierku s_k (podkapitola 4.7)
5. Aktualizujeme polohu kamery (podkapitola 4.9)
6. Opakujeme od kroku 1

Algoritmus stereo VO 3D-3D

1. Detekujeme príznaky v stereo snímkoch I_{k-1}^L, I_{k-1}^R , spárujeme ich a triangulujeme do 3D bodov podľa (4.6)
2. Zopakujeme bod 1 pre stereo snímky I_k^L, I_k^R
3. Spárujeme 3D sady z krokov 1. a 2.
4. Vypočítame rotáciu a transláciu medzi 3D pármi (podkapitola 4.8)
5. Aktualizujeme polohu kamery (podkapitola 4.9)
6. Opakujeme od kroku 1

Algoritmus hybridný 3D-2D

1. Detekujeme príznaky v stereo snímkoch I_{k-1}^L , I_k^L a spárujeme ich
2. Vypočítame pohyb pomocou krokov 1-3 z 2D-2D algoritmu
3. Detekujeme príznaky v stereo snímkoch I_k^L, I_k^R a triangulujeme ich
4. Spárujeme 3D body triangulované v kroku 2 s 3D bodmi z 3. kroku
5. Vypočítame mierku s_k z pomeru 3D párov
6. Aktualizujeme polohu kamery (podkapitola 4.9)
7. Opakujeme od kroku 1

5 Praktická implementácia

Nasledujúca kapitola sa venuje praktickej implementácii v programe MATLAB. Vysvetlené budú základné funkcie, ktoré aplikujú teóriu z predošlej kapitoly. Pre názornosť a lepšie pochopenie budú doplňané obrazovým materiálom z jednotlivých krokov. Postup bude štrukturovaný podobne ako algoritmy uvedené v podkapitole 4.10. Tie kroky, ktoré sa niekoľkokrát opakujú budú vysvetlené len v provom prípade a ďalej bude na ne len odkázované.

Predtým je však nesmierne dôležité uviesť rozdiel medzi anotáciou v literatúre a MATLABe. Pri praktickej implementácii a používaní funkcií od Mathworks je potrebné na tento rozdiel myslieť, ušetrí tak riešiteľovi mnoho času.

$$\lambda\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}f & 0 & u_0\\0 & f & v_0\\0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}X_c\\Y_c\\Z_c\end{bmatrix}$$

Literatúra

$$\lambda[u \quad v \quad 1]=[X_C \quad Y_C \quad Z_C]\begin{bmatrix}f & 0 & 0\\0 & f & 0\\u_0 & v_0 & 1\end{bmatrix}$$

MATLAB

Pokiaľ bude uvedené meno funkcie v tvare `MS_nazovfunkcie` ide o funkciu knižnice alebo skript vytvorený počas riešenia diplomovej práce. Inak bude používaný tvar `nazovfunkcie` pokiaľ ide o funkciu od Mathworks.

5.1 Získavanie dát

Medzi prvé kroky patrí získanie snímkov z kamier (Obrázok 2.6). Obsluhovať tieto zariadenia výrazne zjednodušuje aplikácia *MATLAB Image Acquisition* . Popri práci v rozhraní GUI sa generuje kód s ekvivalentným výsledkom a tak je možné po čase obsluhovať kameru z prostredia editoru. Pre úspešnú komunikáciu webkamery Microsoft (LifeCam Studio) stačí nainštalovať *OS Generic Video Interface* podporu. Užitočné príkazy a funkcie je možné nájsť v zložke digitálnej prílohy (s. 67)

- `MS_getoneframe.m`
- `MS_getXframes.m`
- `MS_DAQ_webcam.m`

Získavanie snímkov zo stereorigu (BASLER acA1300-60gc) je pomerne zložitejšie. Základné informácie je možné nájsť v práci [18] , ktorá sa venuje aplikácii stereovidenia. Spojazdnenie stereorigu nie je priamočiare. Sú potrebné dva sieťové konektory a napájanie 12V UDC pre každú kameru. Týmito prostriedkami disponuje palubný počítač Car4. Návrh postupu je nasledovný:

1. Nainštalujeme *Pylon driver software suite* zo stránok Basler, uistíme sa, že nie je nainštalovaný *GigE Vision Hardware* balík v MATLABe
2. Manuálne nakonfigurujeme IP adresy portov a kamier pomocou *Pylon Camera Configuration Tool*, pričom je potrebné zachovať rovnakú masku podsiete

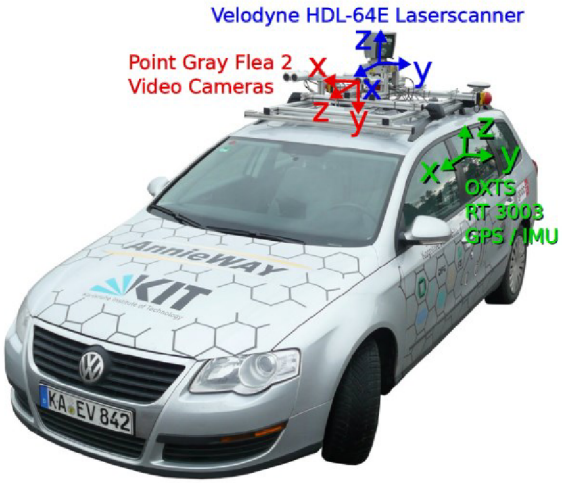
3. Odinštalujeme *Pylon driver* a nainštalujeme podporu kamier v MATLABe, konkrétne balík *GigE Vision Hardware*

Po tomto postupe by mali byť viditeľné obe kamery v GUI rozhraní *Image Acquisition*. Obsluha kamier je demonštrovaná v súbore `MS_stereorig.m`.

Zásadný problém nastáva pri získavaní synchrónnych snímkov, ktoré sú potrebné pre správnu funkciu stereo VO. Nakoľko sa jedná o bežné monokulárne kamery, ktoré nedisponujú hardware synchronizáciou, nie je možné takúto sadu snímkov získať so zaručeným načasovaním. Predošlé využitie tohto stereorigu predpokladalo statickú scénu. Jeho použitie pre algoritmy VO nie je vhodné. Pokúsiť sa obísť tento nedostatok by bolo možné pomocou externého hardware trigru.

KITTI databázy

Pre prvotné testy navrhnutých algoritmov boli využíté voľne dostupné databázy KITTI [19] [20]. Ponúkajú kalibrovaný záznam synchrónnych snímkov stereokamier umiestnených na automobile (Obrázok 5.1). Záznam obsahuje sekvencie snímkov z ulíc za bežnej premávky.



Obrázok 5.1 Automobil tvoriaci databázy KITTI [20]

Tieto databázy ponúkajú rýchlu dostupnosť dát pre test a ladenie algoritmov VO. Obsahujú taktiež presný záznam polohy vozidla, ktorý môže slúžiť pre veľmi podrobný kvalitatívny rozbor. Nevýhodou môže byť miera reprezentácie databáz pre danú aplikáciu car4. Pre správny výber databázy je potrebné stiahnuť z domovskej stránky v časti *odometry* súbory:

- *Odometry dataset grayscale* – sekvencie stereosnímkov
- *Odometry dataset calibration files* – kalibračné matice
- *Odometry ground truth poses* – záznam skutočnej polohy

5.2 Kalibrácia kamery

Procedúra kalibrácie by sa dala zhrnúť do nasledovného postupu:

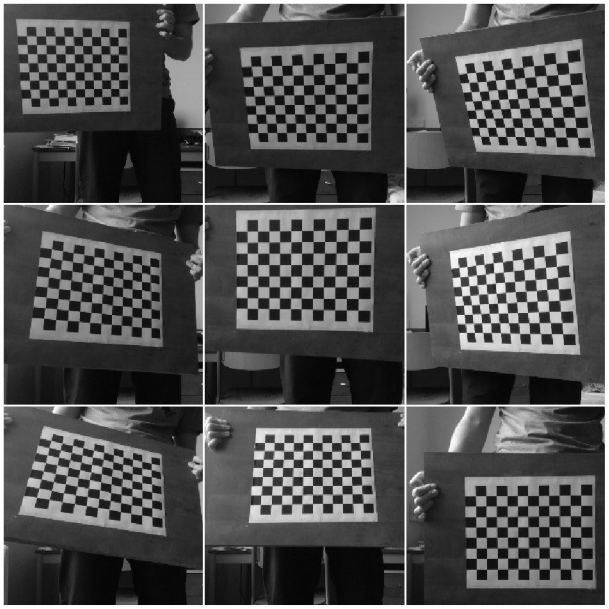
1. Vytvoríme kalibračný vzor, ktorý umiestnime na rovný povrch dosky
2. Pomocou statickej kamery, ktorú kalibrujeme, zachytíme 20-40 snímkov, pričom snímky musia obsahovať celý vzor s rôznym uhlom natočenia (Obrázok 5.2)
3. Snímky spolu s rozmerom jedného štvorca tvoria vstup pre kalibračný software
4. Výstupom sú interné parametre kamery spolu s informáciami o zakrivení obrazu

Pre aplikáciu boli použité dva spôsoby. Prvým bol voľne dostupný *Camera Calibration Toolbox* [21] často používaný nástroj implementovaný v MATLABe. Na webových stránkach nájdeme rozsiahly návod na používanie. Kalibrácia je vykonávaná z GUI a je časovo pomerne náročná. Veľa operácií je potrebných urobiť manuálne, napríklad detekciu rohov pre každý snímok. Na druhej stranu ponúka väčšiu kontrolu nad procedúrou a pokročilejšie nastavenia. Je potrebné sa však zorientovať v rozsiahlom návode a pre prípadné odstránenie zakrivenia obrazu, ktoré je dostupné len manuálne pomocou GUI, využiť iné spôsoby.

Verzia MATLAB 2015b (a novšia) ponúka svoje kalibračné funkcie menovite:

- `detectCheckerboardPoints`
- `generateCheckerboardPoints`
- `estimateCameraParameters`

Nevyžadujú takmer žiadny manuálny zásah a ich výstupom je štruktúra parametrov použiteľná priamo pre funkciu, ktorá odstraňuje zakrivenie obrazu `undistortImage` (Obrázok 5.3). Súbor `MS_calibrate` v zložke */Calib* demonštruje použitie týchto funkcií. V spomínanej zložke sa tiež nachádzajú snímky z kalibrácie jednotlivých kamier a niekoľko kalibračných vzorov rôznej veľkosti.



Obrázok 5.2 Ukážka kalibrácie

Na obrázku 5.3 je možné si všimnúť použitie čiernobielych snímkov. Metódy spomenuté v tejto práci nijako nevyužívajú informácie o farbe. Pre správnu funkciu preto postačujú čiernobiele snímky, čím sa znižuje aj dátový prenos pri DAQ. Vľavo snímok webkamery pred, vpravo po odstránení zakryvenia.

Obrázok 5.3 Odstránenie zakrivenia webkamery

5.3 Detekcia príznakov a párovanie

Pre tento krok ponúka MATLAB na výber implementácie hneď niekoľkých najznámejších operátorov. Postup by sa dal rozdeliť na tri časti:

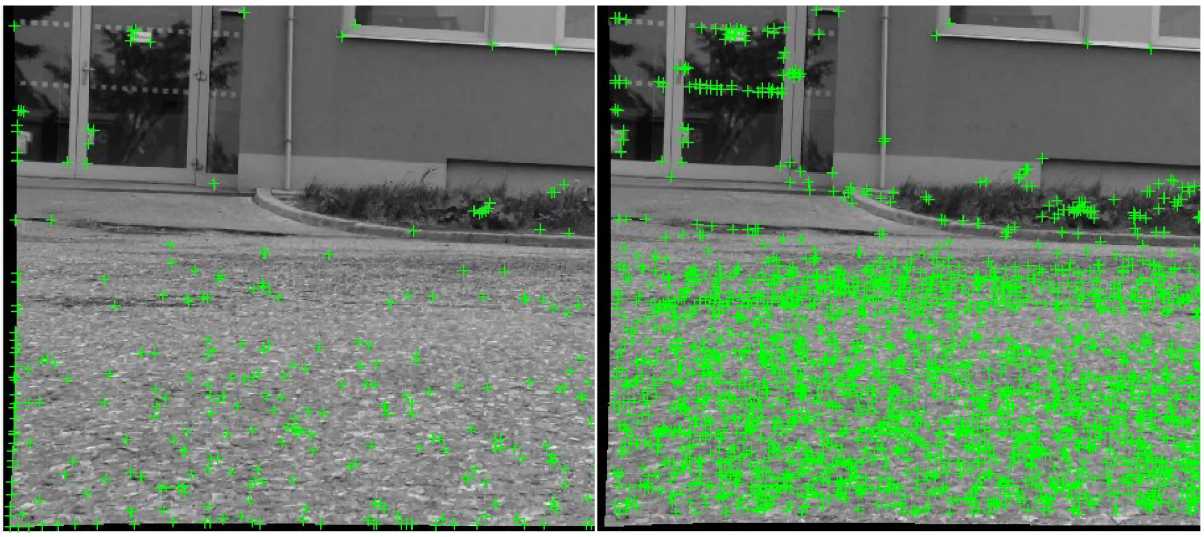
1. Detekcia príznakov
2. Extrakcia deskriptorov
3. Párovanie príznakov hľadaním podobného deskriptora

Detekcia príznakov

V tejto práci boli otestované tri v literatúre najpoužívanejšie detektory Harris, FAST a SURF s funkciami:

- `detectHarrisFeatures`
- `detectFASTFeatures`
- `detectSURFFeatures`

Vstupom týchto funkcií sú okrem samotného snímku dva typy parametrov. Parametre popisujúce kontrast/kvalitu detekovaných príznakov, prípadne iné atribúty, ktoré musí príznak spĺňať (napríklad v prípade SURF je to veľkosť objektov) a oblasť záujmu ROI (Region of Interest), v ktorom sa príznaky detekujú. Ten je dôležitý, pretože chceme vynechať čierny okraj snímku po odstránení zakrivenia. Vytvára totiž vysoký kontrast v snímku, čo vedie ku skresleniu detekcie. Vo všeobecnosti platí - čím nižšia požiadavka na kvalitu a kontrast, tým viac detekovaných príznakov.



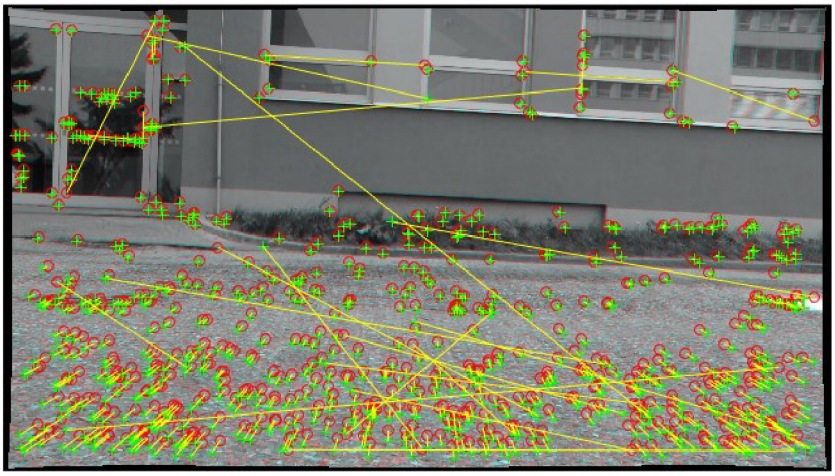
Obrázok 5.4 Detekcia príznakov (detail)

Výstupom týchto funkcií je poloha príznakov. Na obrázku 5.4 vidieť výstup detektora príznakov a rozdiel pri zahrnutí čierneho okraja v ROI pri rovnakom vstupnom parametri kontrastu. Vľavo definovaný vlastný ROI, vpravo detekcia v celom snímku. Tento krok je vykonaný pre oba snímky, medzi ktorými budeme príznaky párovať.

Extrakcia deskriptorov a párovanie

Pre extrakciu je použitá funkcia `extractFeatures`, ktorej vstupom je snímok a príznaky z predošlého kroku. Táto funkcia extrahuje deskriptor jednotlivých príznakov z oboch snímkov. Následne je použitá funkcia `matchFeatures`, ktorá na základe deskriptorov nájde páry príznakov medzi snímkami (Obrázok 5.5).

Každý príznak obsahuje index, ktorý je pre daný snímok jedinečný a pri opakovanej detekcii v snímku zachovaný (viď požiadavky na detektor podkapitola 2.6). Výstupom tohto kroku sú indexy pre obe sady deskriptorov tvoriacich páry. Tým môžeme extrahovať páry s ich polohou v snímku.



Obrázok 5.5 Prekryté snímky I_k a I_{k-1}, extrahované páry príznakov

Na obrázku 5.5 vidieť taktiež chybne spárované príznaky. Tento jav je možné do určitej miery potlačiť parametrami funkcie `matchFeatures`. Dosiahneme to zvýšením presnosti zhody deskriptorov. To vedie k poklesu spárovaných príznakov a tým aj tých, ktoré boli spárované správne. Dostávame sa tak ku dvom protichodným požiadavkam, kedy chceme čo najviac párov, ktoré budú mať najpresnejšiu zhodu.

5.4 Monokulárna VO 2D-2D

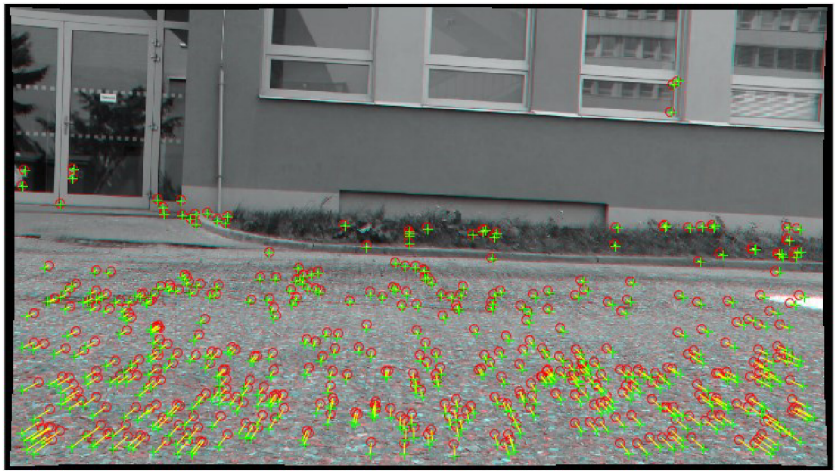
Detekcia príznakov v snímkoch I_{k-1} , I_k a spárovanie

Tento krok bol vysvetlený v predošlej podkapitole 5.3.

Výpočet esenciálnej matice E

MATLAB vo verzii 2015b ponúka výpočet fundamentálnej matice spolu s RANSAC algoritmom. Vzťah medzi fundamentálnou a esenciálnou maticou je jasný z rovnice (4.7) a (4.8). Z dostupných kalibračných matíc sa dá preto priamo vypočítať matica E. Funkcia má názov `estimateFundamentalMatrix` a jej princíp je podobný tomu, ktorý je popísaný v podkapitole 4.5.

Takto formovaná esencíalna matica nespĺňala podmienky (4.11) a (4.12) a z nej počítaná transformácia bola niekedy chybná. Obsahuje však jadro 5-bodového algoritmu a RANSAC, využitá bola preto len pre odstraňovanie chybných párov. Pre výpočet esenciálnej matice bola implementovaná funkcia `MS_norm8E` – 8-bodový algoritmus (podkapitola 4.4). Výstupom tohto kroku je správne určená matica E a odstránené chybné páry.



Obrázok 5.6 Odstránené chybné páry metódou RANSAC

Obrázok 5.7 Odstránenie chybných párov (KITTI)

Na obrázku 5.7 vidieť tiež odstránenie párov, ktoré konali iný pohyb než väčšina scény (biele auto v ľavom pruhu). Páry na dodávke pred kamerou boli ponechané, pretože vytvárali dojem nepohybujúcich sa bodov v diaľke.

Rozklad matice *E* na rotáciu a transláciu

Implementovaný bol postup z podkapitoly 4.6 pre rozklad a prvá časť podkapitoly 4.2 pre trianguláciu. Výber správnej matice vonkajších parametrov P_{ex} počíta funkcia `MS_chooseP`. Pre test, či bod leží pred kamerou, sa však nevyberá 1 ale viac bodov a správna matica sa zvolí podľa väčšinovej zhody, a to z dvôvodu, že v niektorých prípadoch neležal bod ani pred jednou kamerou.

Triangulácia je implementovaná vo funkcii od Mathworks `triangulate`, pre nadmerný prechod medzi anotáciu a štruktúrami premenných však bola implementovaná vlastná funkcia `MS_triangulate` s rovnakým výstupom. V tomto momente je možné veľmi jednoducho kontrolovať správnosť algoritmu. S danými 2D príznakmi, 3D triangulovanými bodmi a správnou projekčnou maticou sme schopní vykonať doprednú projekciu podľa rovnice (4.3).

Na obrázku 5.8 vidieť v jednom snímku 2D príznaky (žlté krúžky) a 3D body po doprednej projekcii (červené krížiky). Napriek tomu, že 3D rekonštruovaná scéna nemá korektnú mierku, teda neodpovedá fyzickej polohe bodov, musí vyhovovať rovnici (4.3) a vždy tieto 3D body zobraziť v snímku správne. Rovnice doprednej projekcie sú implementované vo funkcii `MS_Fwd_proj`.

Výsledná transformačná matica je vypočítaná podľa (4.24). Vyššie popísaný postup aplikuje funkcia `MS_mono`.

Obrázok 5.8 Kontrolná dopredná projekcia

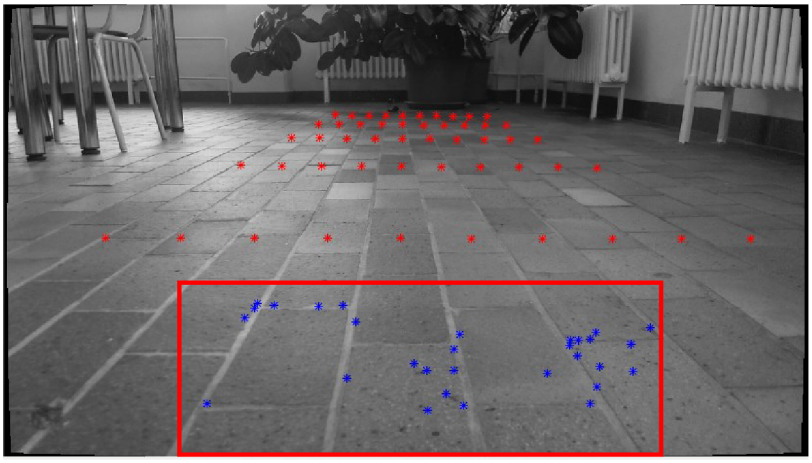
Výpočet mierky

Mierku podľa podkapitoly 4.7 počíta funkcia `MS_getscale`. Je, samozrejme, potrebné zaručiť, aby vstupom boli body detekované na rovine pod vozidlom. Správnosť funkcie je možné opäť sledovať pomocou doprednej funkcie a niekoľkými bodmi z estimovanej roviny.

Na obrázku 5.9 je vidieť 2 farby 3D bodov zobrazených na snímku doprednou projekciou. Modré body klasifikované do ROI oblasti (červený obdĺžnik) a červené body vypočítané z rovnice estimovanej roviny.

Aktualizácia polohy

Koniec cyklu algoritmu tvorí aktualizácia translácie mierkou a konkatenácia podľa podkapitoly 4.9.



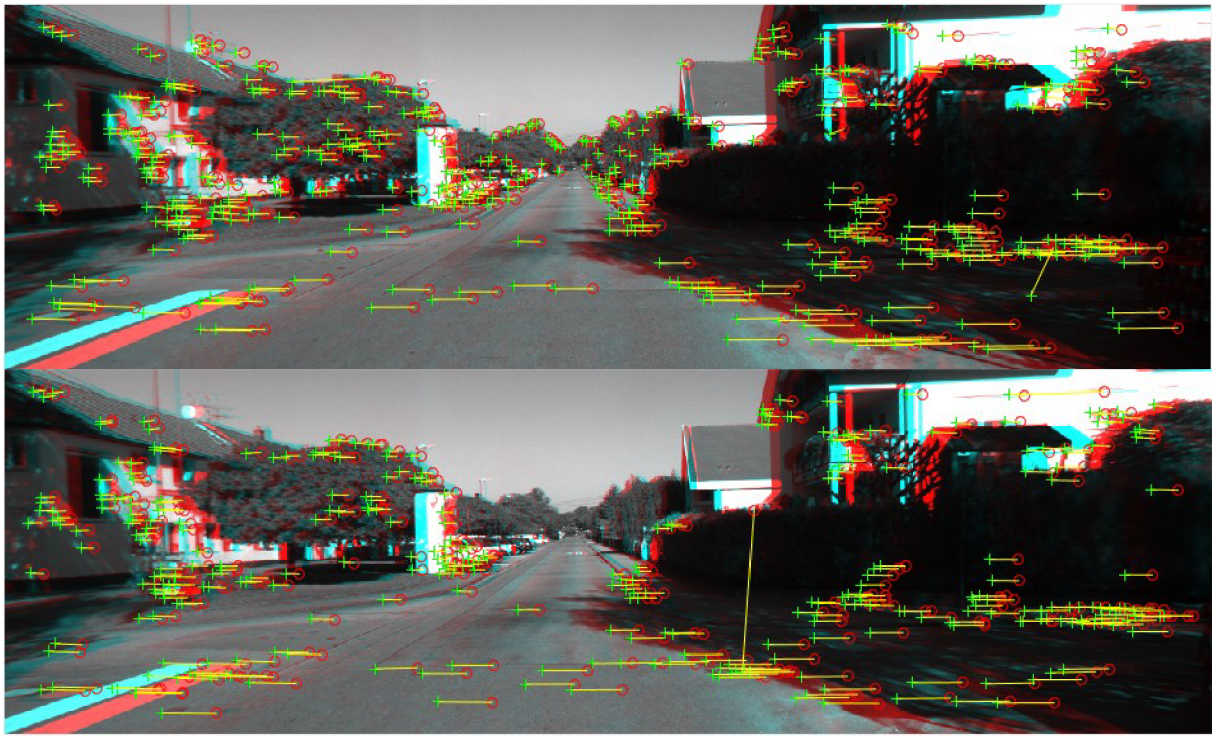
Obrázok 5.9 Estimácia mierky

5.5 Stereo VO 3D-3D

Detekcia príznakov v stereo snímkoch a triangulácia

Detekcia príznakov v jednotlivých snímkoch I_{k-1}^{L}, I_{k-1}^{R} a I_{k}^{L}, I_{k}^{R} vždy prebieha podľa podkapitoly 5.3 rovnako ako párovanie. Triangulovaním podľa rovnice (4.6) dostaneme 2 sady 3D bodov, ktoré relatívne voči sebe konali pohyb. Samozrejme, nie je zaručené, naopak, takmer isté, že tieto sady budú mať rozdielny počet bodov, niektorých aj nespoločných. Ďalej je potrebné uložiť pre každý 3D bod jeho 2D identifikátor z ľavého snímku. Po vykonaní tretieho párovania medzi ľavými snímkami I_{k-1}^{L}, I_{k}^{L} budeme vedieť spárovať aj 3D body. Párovanie teda nastáva 3 krát:

- Medzi I_{k-1}^{L} a I_{k-1}^{R} pre prvú sadu 3D bodov
- Medzi I_{k}^{L} a I_{k}^{R} pre druhú sadu 3D bodov
- Medzi I_{k-1}^{L} a I_{k}^{L} pre párovanie 3D bodov medzi sadami



Obrázok 5.10 Párovanie príznakov v stereosnímkoch

V hornej časti obrázku 5.10 vidieť páry detekované v snímkoch I_{k-1}^{L}, I_{k-1}^{R} a v dolnej I_{k}^{L}, I_{k}^{R}. Okrem toho vidieť v dolnej časti odstránenie bodov, ktoré sa nachádzali príliš ďaleko. Odstránenie je jednoduché, podľa prahovej hodnoty disparity v $[pix]$. Akonáhle má pár v obraze menší rozdiel súradníc u medzi ľavým a pravým snímkom, je odstránený.
Toto opatrenie má jednoduchý dôvod. Body ležiace oveľa ďalej než je vzdialenosť medzi kamerami, vnášajú do triangulácie veľkú chybu (snažíme sa popísať rozdielom pár pixelov informáciu o vzdialenosti niekoľko desiatok metrov).

Párovanie 3D bodov

Každý 3D bod oboch sád obsahuje identifikátor z pôvodného ľavého snímku. V tomto kroku sa jedná o hľadanie prieniku medzi troma množinami. Množinou identifikátorov prvej 3D sady, druhej 3D sady a párov medzi snímkami I_{k-1}^{L} a I_{k}^{L}. Výsledkom tohto kroku sú 3D páry, z ktorých sa môže počítať pohyb stereokamery.

Výpočet rotácie a translácie kamery

Tu je využitá schéma RANSAC podľa postupu, ktorý je popísaný v podkapitole 4.8. Estimáciu parametrov počíta funkcia `MS_Stereo_motion`. Poloha je aktualizovaná podľa podkapitoly 4.9.

5.6 Hybridná VO 3D-2D

Tento prístup kombinuje predošlé dva. V podstate ide o 2D-2D algoritmus, ktorý počíta ťažko odhadnuteľnú mierku z pomeru 3D triangulovaných bodov. Detekcia príznakov a párovanie prebieha podobne ako v algoritme 5.5, v tomto prípade nám však stačí triangulovať body pre jeden stereopár, kontkrétne I_{k-1}^{L}, I_{k-1}^{R}.

Estimácia pohybu prebieha podľa 2D-2D algoritmu medzi ľavými snímkami I_{k-1}^{L} a I_{k}^{L}. Z výsledkov estimácie je triangulovaná druhá sada 3D bodov funkciou `MS_triangulate` a spárovaná s prvou 3D sadou. Mierku z týchto 3D párov počíta funkcia `MS_Stereo_scale`, a to z pomeru najbližšie ležiacich 3D bodov pre minimalizovanie chyby.

Kedže zatiaľ nebolo zabránené chybe párovania medzi ľavým a pravým stereosnímkom, spočíta sa v prvej iterácii esenciálna matica medzi kamerami a chybe párovania je zabraňované počítaním Samsonovej chyby podobne ako v 2D-2D algoritme. Chybu počíta funkcia `MS_Sampson_E`.

5.7 Výber detektora

V grafe 5.1 vidieť výsledok monokulárnej VO pri použití rôznych detektorov. Pre test bol použitý KITTI dataset č. 3 s 800 snímkami (podkapitola 5.1). Z prvých testov plynie niekoľko dôležitých záverov.

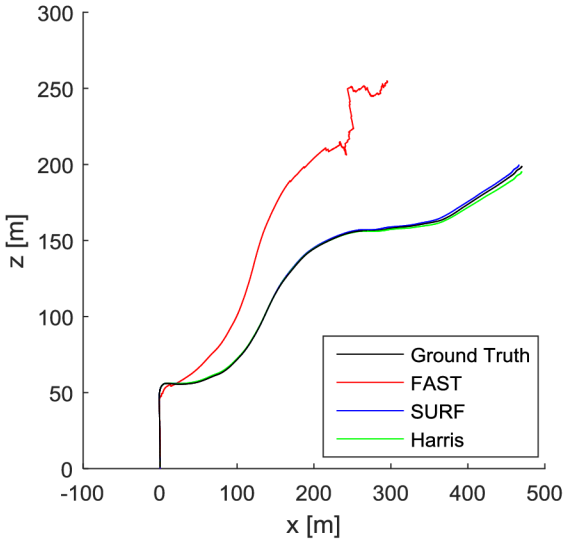
Takmer všetky algoritmy vo forme uvedenej v podkapitolách 5.4-5.6 zlyhávali na nedostatok alebo nerovnomerné rozšírenie príznakov v snímkoch. Príčinou je meniaca sa scéna a tým pádom nemeniace sa parametre, ktoré používajú funkcie na detekciu príznakov, dávajú rôzny výsledok. Naším cieľom by preto bolo nastaviť parametre pre daný snímok individuálne, pokiaľ dôjde ku zmene scény a poklesu detekovaných príznakov pod určité minimum.

Druhým problémom je nerovnomerné rozloženie príznakov. V obrázku 5.11 vidieť snímok z datasetu, ktorý obsahuje homogénne plochy (tieň, obloha). V týchto miestach dochádza ku absencii príznakov, pretože tieto miesta neobsahujú dostatočný kontrast alebo štruktúru. Kvôli tomu môže algoritmus VO degenerovať, prípadne nedetekovať príznaky v miestach, kde to potrebujeme (detekcia roviny pod vozidlom).

Pre výpočet bol použitý 2D-2D algoritmus bez počítania mierky. Mierka bola získavaná z údajov o skutočnej polohe. Tabuľka 5.1 zhŕňa výsledky použitia rôznych detektorov. Chyba v prvom stĺpci značí pomer estimovanej polohy koncového bodu ku skutočnej polohe. Chyba v druhom stĺpci predstavuje priemernú chybu smerových kosínusov jednotlivých ôs *x,y,z* a posledný stĺpec vyjadruje priemerný čas potrebný na detekciu jedného príznaku (priemerný počet príznakov na jeden snímok bol 1500).

Vidieť, že detektor FAST vykazoval najmenšiu presnosť (polohu príznakov vyjadruje celočíselne), zato bol najrýchlejší.. SURF a FAST mali podobný výkon, čo sa týka presnosti. SURF detekuje objekty, a tým vzniká vyššia časová náročnosť než Harris, ktorý detekuje rohy. Pre ďalšie prípady bol preto použitý detektor Harris.

Graf 5.1 Výkon detektorov



Tabuľka 5.1 Výkon detektorov

názov	chyba [%]	chyba [°]	čas/príznak [s]
FAST	33	[20, 7, 20]	1.10^{-6}
SURF	<1	[1, 1, 1]	$6{,}5.10^{-5}$
Harris	<1	[1, 1, 1]	1.10^{-5}

Obrázok 5.11 KITTI dataset 3

5.8 Adaptívna detekcia

Cieľom je zlepšiť rovnomernosť detekovaných príznakov a prispôsobovať detekciu meniacej sa scéne. Vytvorený bol preto algoritmus, ktorý rozdelí snímok na viacero ROI a detekuje príznaky pre každú časť zvlášť. Každá oblasť pritom obsahuje vlastný parameter, ktorý popisuje kvalitatívne požiadavky na detekciu. Tieto parametre sa pri poklese alebo prebytku príznakov v regióne menia. Počet týchto zmien je obmedzený, aby nedochádzalo ku zacykleniu. Samozrejme, tento úkon zvyšuje časovú náročnosť.

Algoritmus adaptívnej detekcie:

1. Detekujeme príznaky v danom ROI
2. Vykonáme maximálne N-krát:
   a) Ak je počet detekovaných príznakov menší než prahová hodnota, zmeníme parameter detektora
   b) Ak je počet detekovaných príznakov väčší než prahová hodnota, zmeníme parameter detektora
   c) Ak parameter saturuje na maximálnej alebo minimálnej hodnote, pokračuje sa ďalším krokom
3. Uložíme hodnotu parametra pre daný ROI
4. Uložíme detekované príznaky
5. 1.-4. opakujeme pre každý ROI

Obrázok 5.12 Adaptívna detekcia

Na obrázku 5.12 je zobrazený rozdiel v detekcii. Rozdelenie ROI je vidieť na obrázku 5.13. Dá sa nastavovať parametricky ako matica okien MxN s prípadným offsetom, pričom každé okno má svoj parameter, ktorý si predáva medzi iteráciami a aktualizuje. Údaje o polohách ROI počíta z týchto parametrov funkcia `MS_Image_init` . Detekcia podľa uvedeného algoritmu je implementovaná vo funkcii `MS_Harris_Segmented`. Časová náročnosť sa tým zvýšila približne na 1.10^{-4} s/príznak, odstraňuje však spomenuté nedostatky.

Obrázok 5.13 Rozdelenie ROI

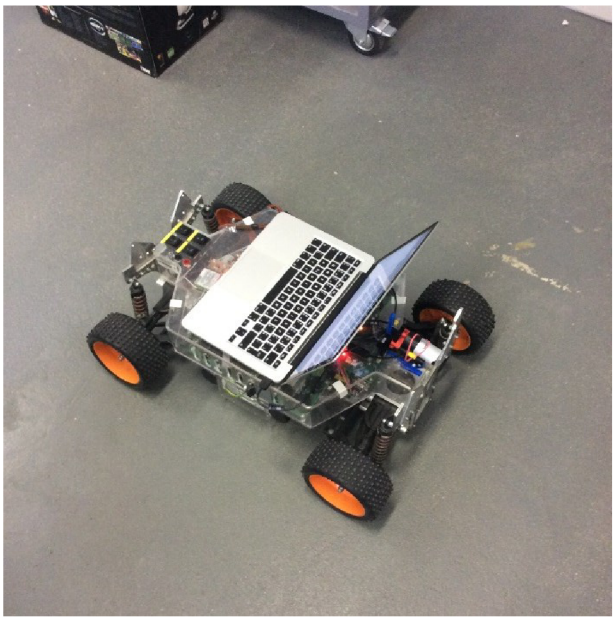
5.9 Výber algoritmu VO pre Car4

Zo zadania tejto práce a návrhu postupu v kapitole 3 vyplývajú nasledovné ciele, ktoré bolo potrebné splniť:

1. Návrh a implementácia alogritmov monokulárnej a stereo vizuálnej odometrie v programe MATLAB
2. Test algoritmov v praxi za rôznych podmienok a ich porovnanie
3. Aplikácia vybraných riešení na vozidlo Car4

Kvôli spomenutým nedostatkom hardware stereorigu (synchronizácia DAQ) bol vybraný pre praktickú aplikáciu na Car4 algoritmus monokulárnej VO s estimáciou mierky podľa podkapitoly 4.7 s použitím web kamery. Hĺbka ostrosti dostupnej web kamery Microsoft však nedovolila ostriť zároveň na vzdialené a bezprostredne blízke predmety. To je pre aplikáciu VO problém, pretože od ostrosti snímku závisí aj správanie sa detektorov príznakov.

Riešením bol DAQ pomocou kamery smartphone (Apple A1421) , ktorý bol vybavený aplikáciou (ProCam) pre fixné nastavenie zaostrenia, expozície a frekvencie snímkovania. Na základe rešerší a praktických testov, sa časť bodu 3 podarila naplniť do miery offline spracovania dát z kamery smartphone. Praktickú aplikáciu stereo VO nebolo možné uskutočniť pre nekvalitný hardware.



Obrázok 5.14 DAQ z jazdy Car4 pomocou web kamery

6 Výsledky

V prvej časti tejto kapitoly bude uvedený výkon troch algoritmov VO na dátach z KITTI databáz. Drúha časť popisuje výsledky aplikácie monokulárneho 2D-2D algoritmu na vozidlo car4 a rozoberá problémy a limity aplikácie v súčasnom stave práce. Posledná časť sumarizuje dosiahnuté výsledky, odpovedá na otázky, ktoré vyplývajú zo zadania, formuluje odporúčania pri práci s VO a navrhuje ďalšie kroky.

6.1 Testy na databázach KITTI

Algoritmy sledovali v každej iterácii približne $1 - 2{,}5.10^3$ príznakov. RANSAC využíval 3.10^3 iterácií s prípustnou chybou 1.10^{-3} $[-]$ v prípade mono algoritmu a 5.10^{-2} $[m]$ v prípade stereo VO. Rozdelenie okien ROI v snímku bolo nastavené na maticu $4x6$. Rozmer kalibrovaných snímkov v databázach KITTI je $370x1226$ $[pix]$.

V grafoch 6.1 a 6.2 vidieť trajektórie vypočítané jednotlivými metódami predstavenými v predošlých kapitolách. Použitý bol dataset č. 5 s 2761 snímkami a č. 9 s 1591 snímkami. Tabuľky 6.1 a 6.2 uvádzajú presnosť estimácie vzhľadom ku skutočnej trajektorii vozidla. Pre kvalitatívne ohodnotenie boli počítané nasledovné údaje:

$$e_{EP}[\%] = \frac{\sqrt{(C_{end} - T_{end})^2}}{T_{dist}} \cdot 100 \tag{6.1}$$

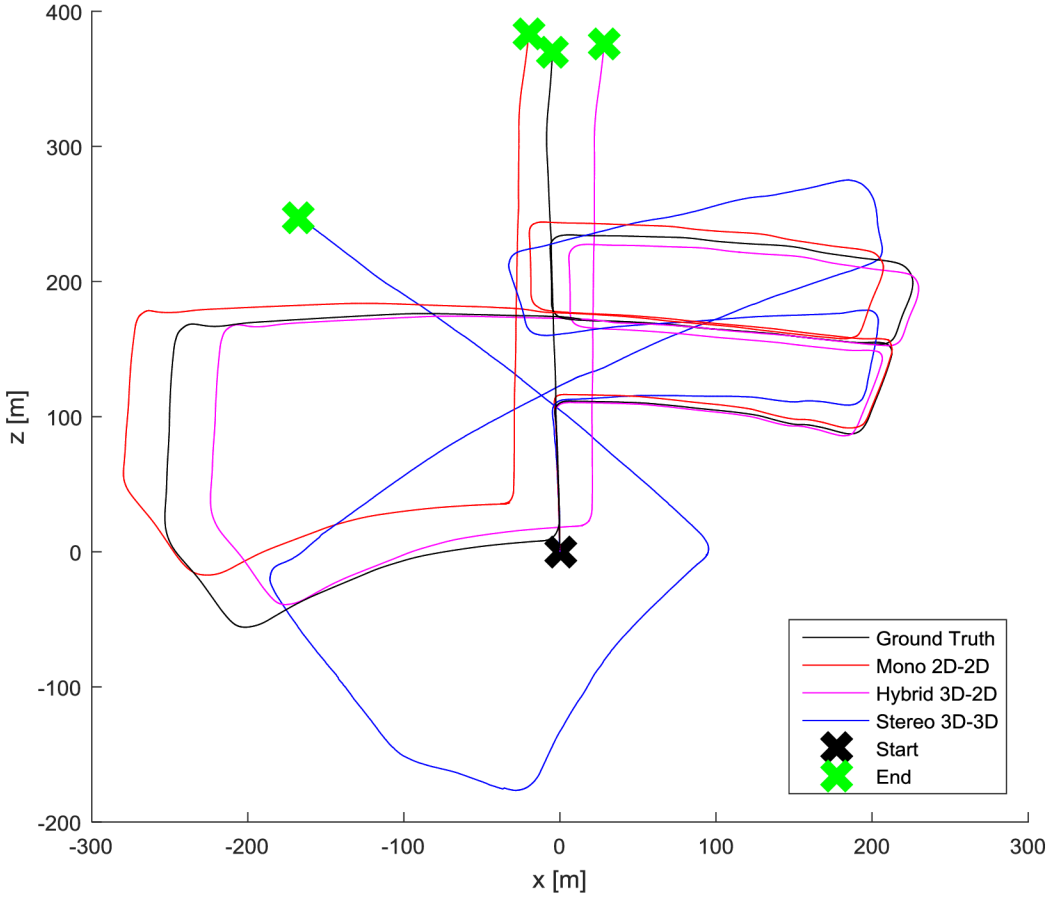
kde e_{EP} značí chybu koncového bodu vypočítanú ako pomer celkovej prejdenej dráhy vozidla T_{dist} a vzdialenosti medzi estimovanou polohou koncového bodu C_{end} ku skutočnej polohe koncového bodu T_{end}. Ďalej chyba prejdenej vzdialenosti e_T :

$$e_T[\%] = \frac{T_{est} - T_{dist}}{T_{dist}} \cdot 100 \tag{6.2}$$

kde T_{est} predstavuje estimovanú dĺžku trajektórie. Priemerná chyba natočenia vzhľadom k osám *x,y,z* bola počítaná z rozdielu smerových kosínusov dvoch vektorov ako:

$$e_{cos}[^\circ] = \frac{1}{n-1}\sum_{i=1}^{n-1} \cos^{-1}(v_{VO}^{i,i-1}) - \cos^{-1}(v_T^{i,i-1}) \tag{6.3}$$

kde $v_{VO}^{i,i-1}$ je smerový kosínus pre danú os estimovanej trajektórie vypočítaný pre vektor trajektórie medzi snímkom *i* a *i-1*. $v_T^{i,i-1}$ je smerový kosínus vektora vypočítaného zo skutočnej trajektórie medzi bodom *i* a *i-1*. e_{cos} je priemerný rozdiel uhlov v stupňoch a *n* počet snímkov.

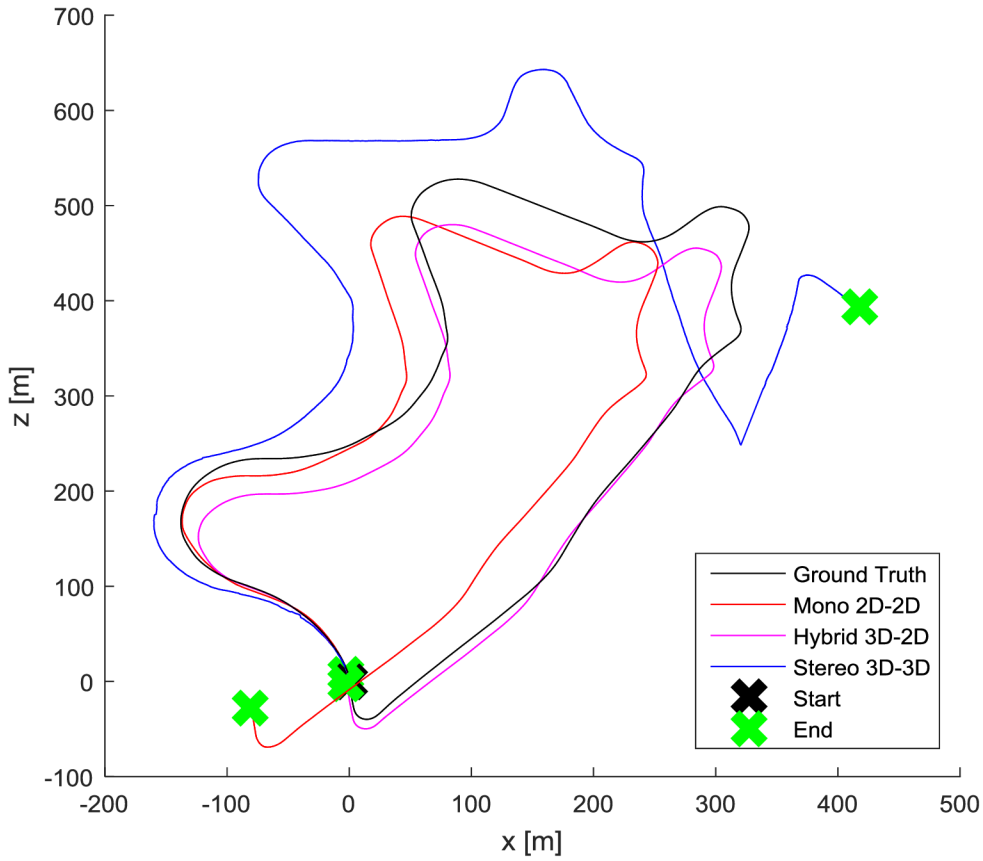


Graf 6.1 Porovnanie algoritmov VO (Dataset 5)

Algoritmus	e_{EP} [%]	e_T [%]	e_{cos} [°]		
			x	**y**	**z**
Mono VO 2D-2D	1,0	0,6	2,1	2,7	1,7
Hybridná VO 3D-2D	1,5	3,9	2,2	1,6	1,6
Stereo VO 3D-3D	9,3	1,3	14,5	2,0	17,8

Tabuľka 6.1 Porovnanie algoritmov VO (Dataset 5)

Obrázok 6.1 KITTI dataset 5 - ukážka



Graf 6.2 Porovnanie algoritmov VO (dataset 9)

Algoritmus	e_{EP} [%]	e_T [%]	e_{cos} [°]		
			x	**y**	**z**
Mono VO 2D-2D	5,1	6,2	0,9	1,0	0,7
Hybridná VO 3D-2D	0,8	7,1	0,5	0,9	0,4
Stereo VO 3D-3D	37,1	5,7	24,8	8,7	20,5

Tabuľka 6.2 Porovnanie algoritmov VO (dataset 9)

Obrázok 6.2 KITTI dataset 9 - ukážka

Z grafov 6.1 a 6.2 vidieť trajektórie estimované jednotlivými algoritmami VO spolu so skutočnou trajektóriou vozidla. Snímky zaznamenali občasnú premávku a chodcov, ktorí vytvárali neželaný pohyb v scéne. RANSAC estimácia parametrov si poradila s väčšinou týchto javov, napriek tomu algoritmus stereo 3D-3D zlyhal v úseku viditeľnom v grafe 6.2, čo bolo pravdepodobne spôsobené protiidúcim vozidlom (Obrázok 6.2).

6.2 Testy s vozidlom Car4

Vybraný algoritmus pre Car4 dosahoval pri testoch na dátach KITTI schopné výsledky. Pre prototypovanie, test a kvalitatívne posúdenie ponúkajú tieto datasety výbornú pomôcku, ktorá dokáže ušetriť mnoho času. Tieto záznamy však úplne neodrážajú podmienky a výzvy, ktorým využitie VO v praxi čelí a obsahujú záznamy výhradne z exteriéru.

Algoritmus pri aplikácii na snímkoch z jazdy s vozidlom Car4 značne zlyhával. V našom prípade bola frekvencia získavania dát 5Hz, vozidlo Car4 jazdilo rýchlosťou približne 0.5-1 m/s. Pri získavaní dát uzatváralo naše vozidlo trajektóriu, a to z dôvodu kontroly presnosti počítaním relatívnej chyby koncového bodu e_{EP}. Nasledujúce problémy museli byť odstránené:

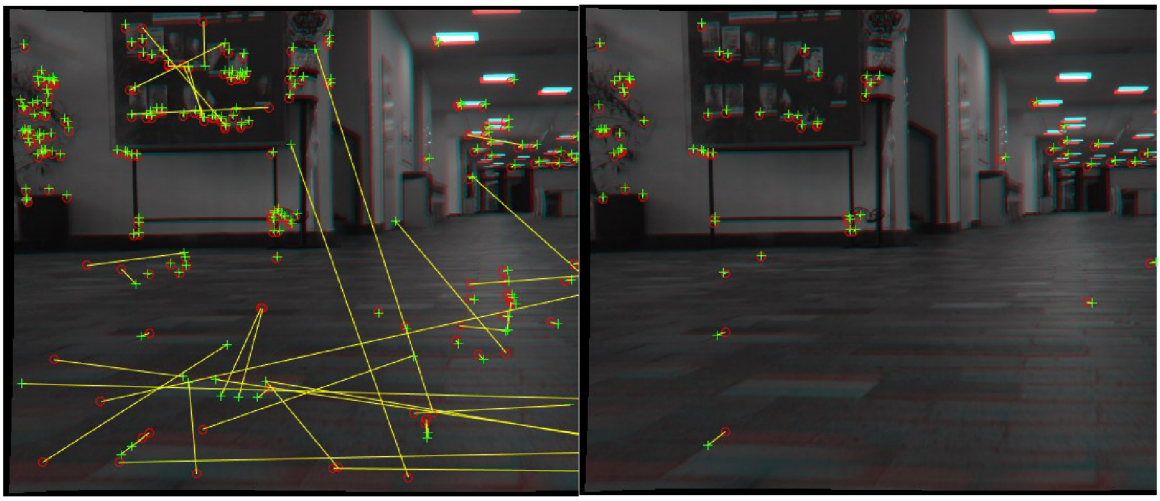
1. Bezhybný stav

Vozidlo môže kedykoľvek zastaviť, či už kvôli požiadavke, strate signálu, vplyvom chyby alebo náhlej zmene smeru. Pri triangulácii, kedy páry príznakov majú veľmi malý alebo nulový rozdiel v súradniciach dochádza k zlyhávaniu výpočtu. Aby sa scéna chovala nehybne, museli by triangulované body ležať v „nekonečnej“ vzdialenosti. Napriek tomu, že si v niektorých prípadoch algoritmus s týmto javom poradil, dochádzalo k častému nárastu chyby alebo pádu výpočtu.

Odstrániť tento problém sa podarilo sledovaním zmeny obrazu. Neúspešná implementácia využívala na tento krok rozdiel súradníc príznakov medzi snímkami I_k a I_{k-1}. Odhadnúť univerzálnu prahovú hodnotu zmeny bolo však náročné. Efektívne riešenie ponúklo až sledovanie zmeny pomocou korelácie časti snímku I_k a I_{k-1}. Kým korelácia týchto častí neklesla pod určitú hranicu (0,995), celý proces estimácie pohybu bol vynechaný. Výsledkom bola značne zvýšená robustnosť.

2. Pokles párov príznakov

Napriek relatívne dostatočnému počtu detekovaných príznakov dochádzalo v interiérových priestoroch k ich častému nespárovaniu a celkovému poklesu vplyvom neostrosti snímku (Obrázok 6.3). To viedlo k nedostatočnému počtu párov ROI roviny a zlyhávala estimácia mierky.

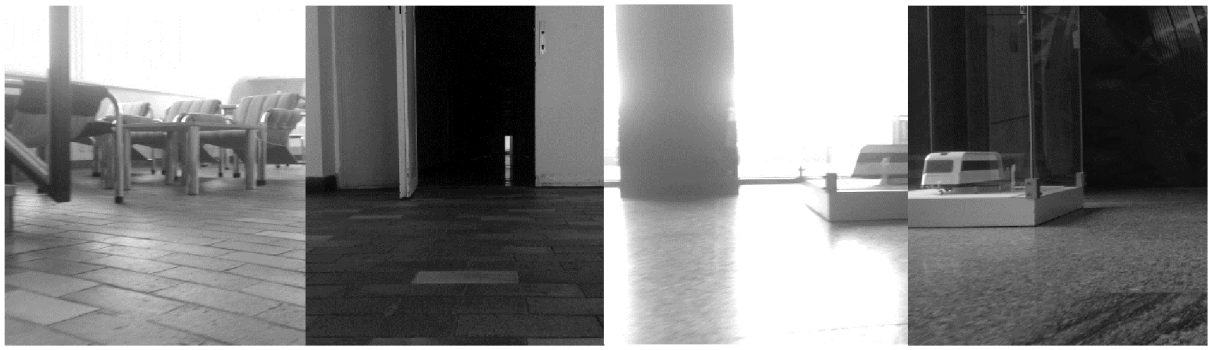


Obrázok 6.3 Nedostatok párov v rovine pod vozidlom (detail)

Neúspešným krokom pre odstránenie problému bolo využitie mierky z predošlej iterácie. Hodnota mierky sa výrazne mení v každom kroku cyklu. Rozumne eliminovať tento problém pomohlo podobné pravidlo. Pokiaľ je počet bodov v rovine menší než N (napr. 5), bude použitá translácia z predošlej iterácie. Inak povedané, pokiaľ algoritmus nevedel estimovať mierku, použil aktuálnu vypočítanú rotačnú maticu a transláciu z predošlého cyklu.

3. Meniaca sa svetlosť scény

Aby bol dosiahnutý ostrý snímok pri DAQ, je potrebné nastaviť relatívne krátky expozičný čas. Osvetlenie miestností veľakrát nestačilo na to, aby bol expozičný čas dostatočne krátky pre vyhotovenie ostrých snímkov. Automatická expozícia kamier nastavuje expozičný čas pre vyhotovenie svetlej snímky na úkor ostrosti. V prípade aplikácie VO to však nie je žiadúce a snímok by sme radšej podexponovali pre dosiahnutie najostrejšieho možného snímku pri miernom zhoršení štruktúry snímku a kontrastu.



Obrázok 6.4 Rozdiely vo svetlosti interiérov

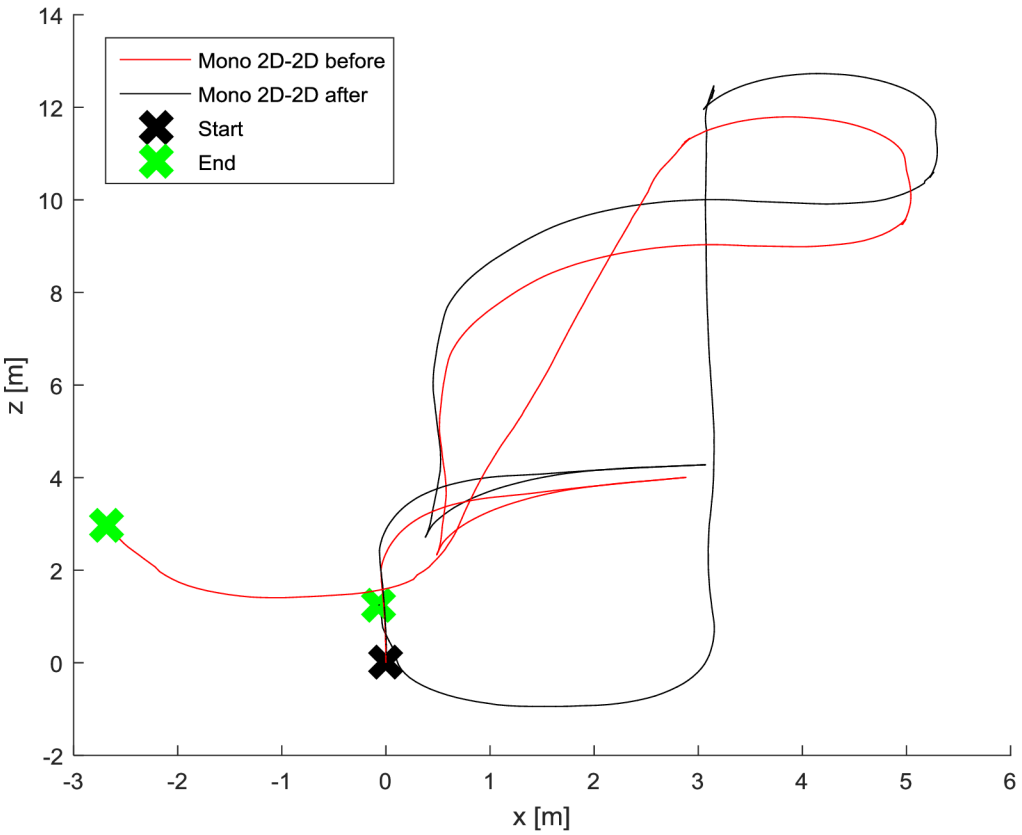
Ďalším problémom je aj veľká relatívna zmena vo svetelných podmienkach interiérov, a to najmä vplyvom okien (Obrázok 6.4). Tento rozdiel je pre ľudské oko ťažšie zaznamenateľný, pretože dokáže dobre vnímať oveľa dynamickejší rozsah. Konflikt je jasný - zníženie expozičného času povedie k ostrejšiemu snímku v dostatočne osvetlenom priestore, no aj k podexponovaniu v horšie osvetlených častiach. Zvýšenie expozičného času zlepší štruktúru snímku v tmavej časti miestnosti, no povedie k rozmazaniu pri pohybe

a preexponovaniu vo svetlých častiach miestnosti. Expozícia preto musela byť v interiéri nastavovaná manuálne na fixnú hodnotu, ktorá bola akýmsi kompromisom, väčšinou však podexponovaná v tmavších miestach miestnosti.

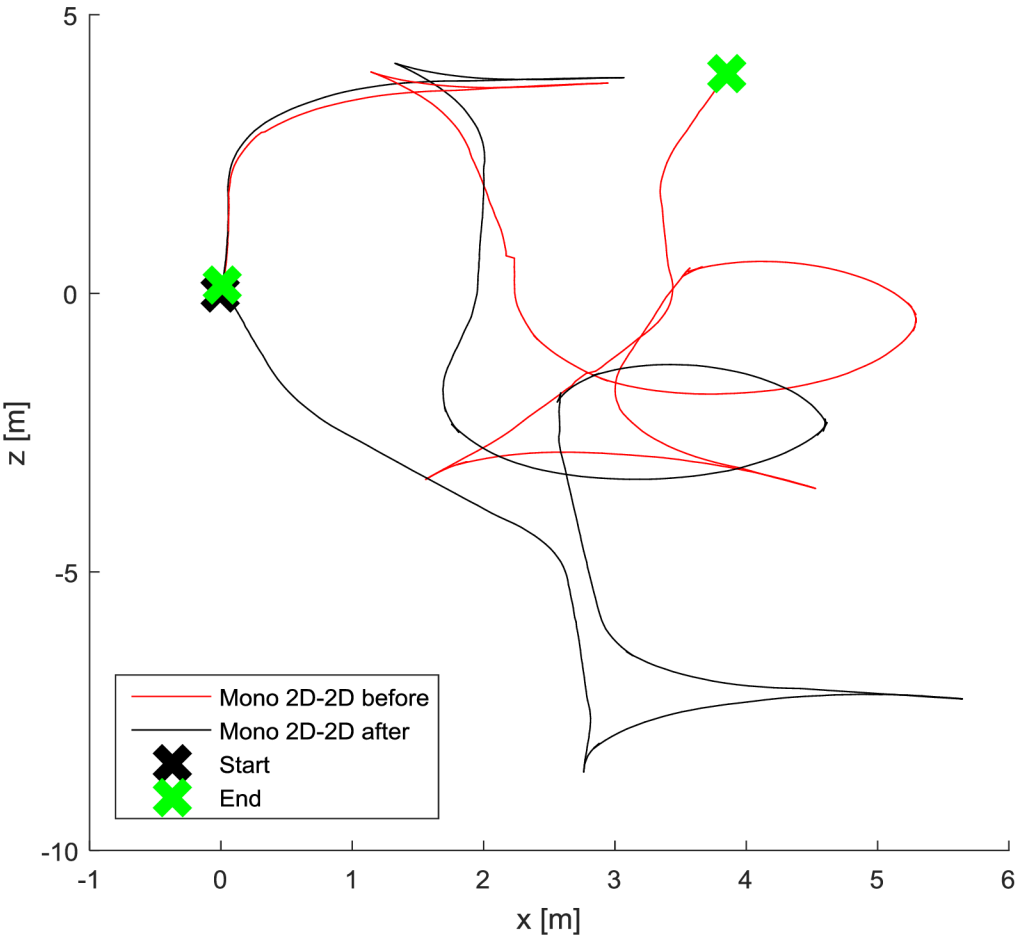
V exteriéri je tento problém o niečo jednoduchší, najmä vďaka zvýšenej a konzistentnej svetlosti od slnka. Napriek tomu dochádza k preexponovaniu pri pohľade na jasne osvietenú plochu. Pri použití automatickej expozície sú však časy uzávierky natoľko krátke, že nedochádza k rozmazaniu snímkov (Obrázok 6.5).

Obrázok 6.5 Automatická expozícia v exteriéri

Po zavedení týchto zmien došlo ku zvýšeniu presnosti VO z 14-20% relatívnej chyby koncového bodu na 3,4% v prípade (Graf 6.3) a na menej ako 1% v prípade (Graf 6.4). Ukážku zo záznamov vidieť na obrázku 6.5.



Graf 6.3 Mono VO s Car4 exteriér

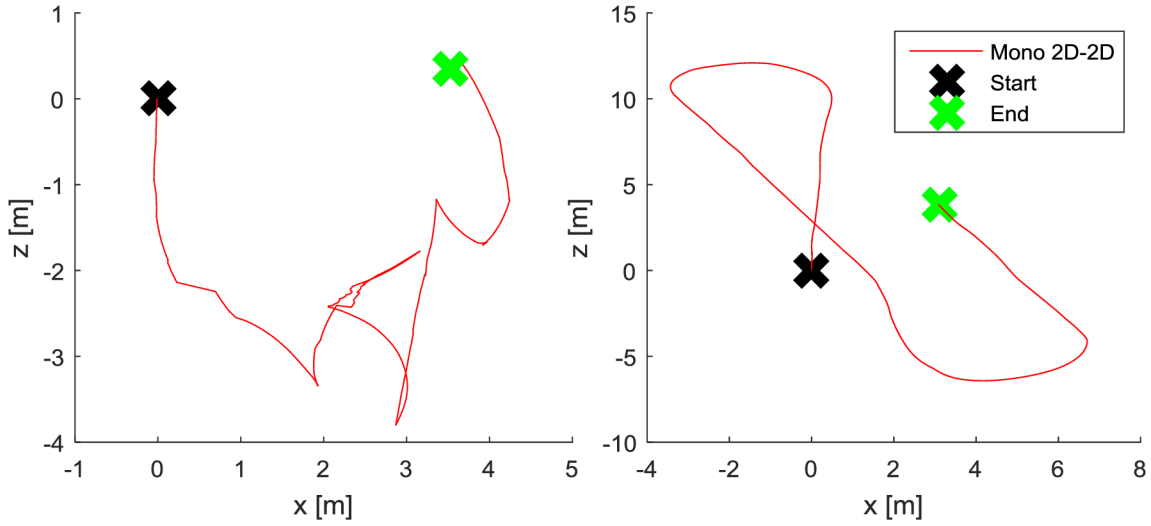


Graf 6.4 Mono VO s Car4 - exteriér

Aplikácia vybraného riešenia v interiéri je oveľa náročnejšia, a to z uvedených dôvodov na začiatku podkapitoly 6.2, ale aj kvôli samotnej štruktúre prostredia. Holé steny, podlaha a tmavé alebo príliš svetlé plochy značne redukujú detekciu príznakov, čo malo vplyv na výkon VO.

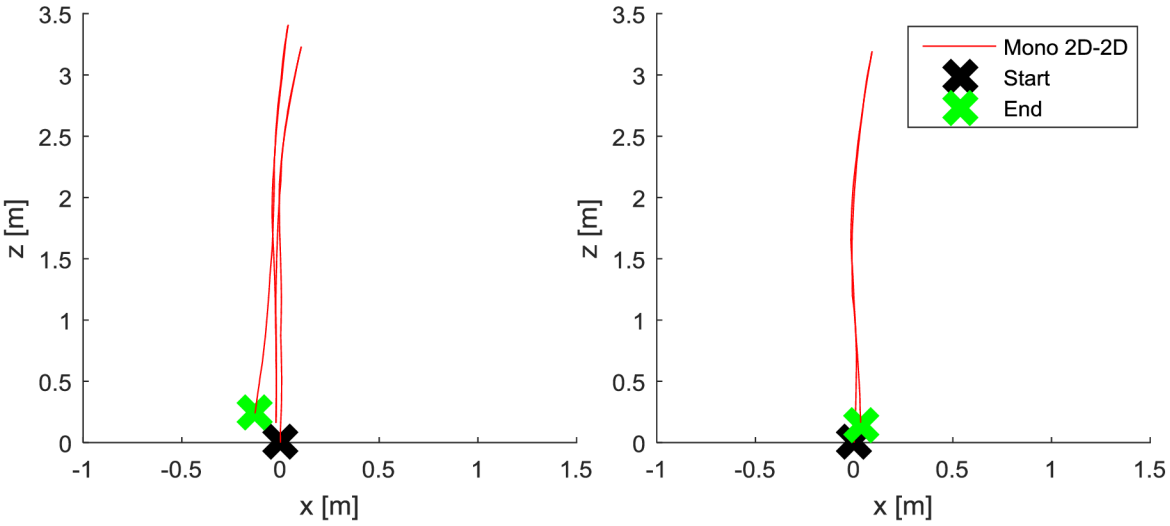
Hlavný problém však tvorí neželané rozmazanie snímku, napriek zníženému času uzávierky podexponovaním. Chyba polohy koncového bodu sa pohybovala v rozmedzí 10-40%, riešenia často zlyhávali napriek opatreniam, ktoré boli predstavené počas riešenia a v predošlých kapitolách. Názorný výsledok spolu s ukážkou snímkov je vidieť v grafe 6.5 a obrázku 6.6. Lokálne tieto algoritmy fungovali s relatívne malou chybou približne 2,5-4% na jednoduchej kalibrovanej trojmetrovej vzdialenosti (Graf 6.6) v prostredí na obrázku 6.6 vpravo.

Limit vybraného riešenia pre Car4 predstavuje aj výpočet mierky. Pre správne fungovanie je potrebný predpoklad pohybu nad rovinou.



Graf 6.5 Mono VO s car4 interiér

Obrázok 6.6 Ukážka snímkov interiéru



Graf 6.6 Kalibrovaný úsek (interiér)

6.3 Realtime aplikácia

Tabuľka 6.3 uvádza porovnanie časovo najnáročnejších krokov podľa MATLAB profiler.

	jednotky	**Mono 2D-2D**	**Hybrid 3D-2D**	**Stereo 3D-3D**	**Mono (car4)**
snímok	[pix]	370x1226			1080x1920
čas/iterácia	[s]	1,854	2,720	3,502	4,84
RANSAC	[s (%)]	1,08 (58,9)	1,17 (43,3)	0,91 (26,1)	2,92 (60,4)
RANSAC N	[-]	3000			
detekcia príznakov	[s (%)]	0,60 (32,3)	1.10 (40,5)	1,53 (43,6)	1,39 (28,8)
párovanie	[s (%)]	0,08 (4,3)	0,16 (5,8)	0,47 (13,4)	0,33 (6,8)

Tabuľka 6.3 Časová náročnosť algoritmov

Je vidieť postupný nárast časovej náročnosti u 3D-2D a 3D-3D metódy, ktorý je spôsobený častejšou detekciou príznakov a ich párovaním. U stereovízie sa jedná o zdvojnásobenie vstupných snímkov a ztrojnásobenie párovania. Nemalý vplyv má na to aj samotný rozmer snímku. Pri použití menších snímkov (480x640) pre vybrané riešenie monokulárnej VO 2D-2D však tento čas neklesol pod 0,3s.

Druhým, časovo náročným krokom, je estimácia parametrov metódou RANSAC, ktorá je, samozrejme, priamo úmerná počtu dovolených iterácií. Minimalizovať počet iterácií nie je jednoznačný úkon. Pri počte 500 iterácií došlo ku zhoršeniu chyby koncového bodu na 14% a chyby trajektórie na 12% v prípade mono algoritmu 2D-2D. Pri tomto počte vyžadoval uvedený krok vo vybranom algoritme približne 0,2 s.

Požadovaná frekvencia algoritmov VO je 5 (naše testovanie) a 10Hz (KITTI). Závisí to, samozrejme, od požiadavky na rýchlosť pohybu vozidla. Implementácia v tejto práci momentálne nedokáže dosahovať realtime výkon a dáta boli spracované offline. Do uvedených časov je potrebné pripočítať ešte čas potrebný pre DAQ. V prípade webkamery a stereorigu sa časy pre získanie jedného snímku pohybovali v rozmedzí 0,07-0,15 (webkamera) a 0,5-1,3s (stereorig). Pre získanie bola použitá funkcia `getsnapshot`, ukážky získavania dát obsahujú skripty v zložke digitálnej prílohy */Others*. V prípade, že bol použitý korektný postup, DAQ pomocou príkazov *MATLAB Image acqusition toolbox* nemusí byť správna cesta pre samotné získavanie snímkov.

Pre zrýchlenie by bolo najrozumnejšie zameranie sa na nasledovné body:

- Detekcia príznakov a ich párovanie implementáciou v iných jazykoch. Efektívne prototypovanie v MATLABe nemusí poskytovať rýchlosť kompilovaného C/C++ kódu, ktorý poskytuje väčšiu kontrolu nad alokáciou alebo rýchlejší prístup ku knižniciam.

- Zníženie počtu iterácií, prípadne využitie adaptívneho RANSAC algoritmu [2], ktorý upravuje počet ďalších iterácií na základe počtu správnych párov v jednotlivých iteráciach
- Implementácia 5-bodového algoritmu (volaný v každej RANSAC iterácii) v inom jazyku (C/C++)
- Využitie iného software pre DAQ

6.4 Návrh pokračovania

Oblasť vizuálnej odometrie ponúka zaujímavú alternatívu pri riešení navigácie a určovaní polohy vozidla pre potreby (polo)autonómneho riadenia. V prípade pokračovania výskumu v tejto oblasti a ďalšej implementácie, sú navrhované nasledovné kroky:

- Vyriešenie rozmazávania snímkov pri DAQ v interiéroch použitím vhodného hardware s vyššou svetelnosťou optiky a čipu
- Software adaptívna zmena expozície v interiéroch s ohľadom na potreby VO (scénu je možné mierne podexponovať, neprekročiť pritom prahovú hodnotu expozície)
- Preskúmanie možnosti využiť stereorig ako monokulárnu variantu
- Zvýšenie presnosti stereo algoritmu, prípadne jeho aplikácia s vhodným hardware
- Implementácia inej estimácie mierky v prípade monokulárnej VO (hĺbkový senzor kinect, trifokálny tenzor)
- Kombinácia VO s iným senzorom polohy
- Implementácia častí algoritmov v inom jazyku (C/C++)
- Implementácia lokálnej optimalizácie parametrov bundle adjustment

7 Záver

Cieľom tejto práce bolo navrhnúť algoritmy vizuálnej odometrie a implementovať ich v programe MATLAB, ďalej aplikovať ich s použitím dostupného hardware na experimentálne vozidlo Car4 a prispieť tým do sféry lokalizácie vozidla v priestore pre potreby ďalšieho výskumu (polo)autonómneho riadenia.

Rešeršná časť vysvetľuje samotný pojem VO a uvádza čítajúceho do širšej oblasti spracovania obrazu, kam tieto algoritmy patria. Následne sú algoritmy rozdelené podľa niekoľkých kritérií ako počet kamier a princíp funkčnosti. Ďalej uvádza základnú štruktúru algoritmov VO so zameraním na monokulárnu a stereo variantu. Porovnáva taktiež výhody a nevýhody jednotlivých riešení. Táto časť súčasne rozoberá aj problematiku spojenú s kamerovým systémom a vysvetľuje princíp detekcie príznakov a RANSAC algoritmu. Z výstupov rešeršnej stati bolo možné následne vhodnejšie definovať postup riešenia.

Druhá časť sa zaoberá teoretickým pozadím jednotlivých častí algoritmov. Návrh je postavený na osnovách z rešeršnej časti. Obsah tvoria matematické vzťahy, na ktorých je VO založená, spolu s literatúrou, z ktorej vychádzajú. Doplnené sú o komentáre a praktické rady, ktoré boli nadobudnuté počas práce. Literatúra často predpokladala pokročilejšie znalosti v tejto oblasti a tak neuvádzala niektoré podstatné detaily. Orientáciu zhoršovala rôzna anotácia a nekoherentnosť jednotlivých riešení. Výber vhodného riešenia a úpravu anotácie pomohla vyriešiť často až praktická implementácia alebo využitie literatúry, ktorá na chyby priamo poukazovala.

Výstupom tejto časti sú zjednotené teoretické postupy a matematické vzťahy využiteľné pre praktickú aplikáciu v programe MATLAB. Na konci tejto časti je definovaná konkrétna podoba troch algoritmov – monokulárnej, stereo a hybridnej VO.

Praktická implementácia obsahuje konkrétne riešenia jednotlivých krokov algoritmov, ktoré boli uvedené na konci kapitoly teoretického návrhu. Vysvetľuje rozdiel medzi anotáciou literatúry a programu MATLAB, uvedené sú tiež postupy získavania dát, využitie záznamov z jazdy osobného vozidla (databáz KITTI) a problémy spojené s dostupným hardware. Pre správne fungovanie stereo algoritmov bolo potrebné zaručiť synchrónne získavanie snímkov, ktorým však stereorig nedisponoval.

Táto kapitola uvádza tiež funkcie a vizualizáciu krokov, ktoré využívajú tri algoritmy uvedené na konci kapitoly teoretického návrhu. Počas riešenia práce museli byť implementované vlastné funkcie kvôli nedostatočnej kontrole alebo nevhodnosti dostupných funkcií. Adresované sú tiež početné problémy spojené s praktickou aplikáciou a spracovaním obrazu. Problém nedostatočnej detekcie príznakov a meniacej sa scény odstraňuje predstavenie adaptívnej detekcie. Na základe praktických skúseností, dostupnosti hardware a prvotných testov algoritmov na databázach KITTI bolo zhodnotené zadanie diplomovej práce a vybraný algoritmu monokulárnej VO s predpokladom lokálnej roviny pod vozidlom pre aplikáciu na vozidlo Car4.

Kapitola výsledkov zhŕňa výkon implementovaných algoritmov, ktoré boli testované na databázach KITTI, porovnáva ich presnosť a robustnosť. Ďalej sú v tejto časti uvedené výsledky z aplikácie vybraného riešenia na jazdu s vozidlom Car4 v interiéri a exteriéri. Ostrenie použitej web kamery nevyhovovalo požiadavkám VO, pre získavanie dát preto bola použitá kamera zo smartphone. Porovnaný je výkon vybraného mono algoritmu v závislosťi na svetlosti a type scény. Adresovaná je kľúčová potreba získavania ostrých snímkov, ktorá v opačnom prípade znižuje robustnosť. Ďalej je uvedená časová náročnosť algoritmov, ktoré nedosahovali realtime frekvenciu a preto museli byť dáta spracované offline. Posledné časti sú venované návrhom pokračovania výskumu so zameraním na zrýchlenie algoritmov a využitie vhodnejšieho hardware.

Napriek uvedeným problémom s hardware sa podarilo pôvodné ciele naplniť do vysokej miery a odpovedať na všetky body zadania. Výstupom práce je podrobný a štrukturovaný popis častí algoritmov VO, ktorý zjednocuje a dopĺňa riešenia z literárnych prameňov. Poskytuje tiež prehľad a rozbor problémov tejto oblasti, ktorý môže slúžiť pre rýchlejšie a kvalitnejšie prototypovanie, návrh a aplikáciu VO ale aj SFM a V-SLAM algoritmov všeobecne. Práca tiež ponúka konkrétne riešenia pre lokalizáciu vozidla Car4 s bohatou knižnicou funkcií v programe MATLAB.

Zoznam použitej literatúry

[1] CHENG, Yang, Mark MAIMONE a Larry MATTHIES. Visual odometry on the Mars Exploration Rovers. *2005 IEEE International Conference on Systems, Man and Cybernetics*. 2005, (1), 903-910. DOI: 10.1109/ICSMC.2005.1571261.

[2] SCARAMUZZA, Davide a Friedrich FRAUNDORFER. Visual Odometry: Part I: The First 30 Years and Fundamentals. *IEEE ROBOTICS & AUTOMATION MAGAZINE*. 2011, , 80-92. DOI: 10.1109/MRA.2011.943233.

[3] FRAUNDORFER, Friedrich a Davide SCARAMUZZA. Visual Odometry: Part II: Matching, Robustness, Optimization, and Applications. *IEEE ROBOTICS & AUTOMATION MAGAZINE*. 2012, , 78-90. DOI: 10.1109/MRA.2012.2182810.

[4] O. A. AQEL, Mohammad, M. SARIPAN a Napsiah BT. ISMAIL. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*. 2016, (5), 1-26. DOI: 10.1186/s40064-016-3573-7.

[5] TRIGGS, Bill, Philip F. MCLAUCHLAN, Richard I. HARTLEY a Andrew W. FITZGIBBON. Bundle Adjustment — A Modern Synthesis. *Vision Algorithms: Theory and Practice (LNCS)*. 1999, (1883), 298-372. DOI: 10.1007/3-540-44480-7_21.

[6] PERSSON, Mikael, Tommaso PICCINI, Michael FELSBERG a Rudolf MESTER. Robust Stereo Visual Odometry from Monocular Techniques. *IEEE Intelligent Vehicles Symposium (IV)*. COEX, Seoul, Korea, 2015, (, 6.

[7] NISTER, D., O. NARODITSKY a J. BERGEN. Visual Odometry. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2004, , 1-8. DOI: 10.1109/CVPR.2004.1315094.

[8] HARTLEY, Richard a Andrew ZISSERMAN. *Multiple View Geometry in Computer Vision*. Second Edition. Cambridge (New York): Cambridge University Press, 2004. ISBN 978-0-511-18618-9.

[9] SÜNDERHAUF, Niko a Peter PROTZEL. *Stereo Odometry – A Review of Approaches*. Technical Report 3/07. Chemnitz: Chemnitz University of Technology, 2007.

[10] IOCCHI, Luca. Stereo Vision: Triangulation. In: *Dipartimento di Ingegneria informatica, automatica e gestionale - Università di Roma* [online]. Rím, 1998 [cit. 2017-05-04]. Dostupné z: http://www.dis.uniroma1.it/~iocchi/stereo/triang.html

[11] NISTÉR, David. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2004, **26**(6), 756-770. DOI: 10.1109/TPAMI.2004.17.

[12] LONGUET-HIGGINS, H. A computer algorithm for reconstructing a scene from two projections. *Nature*. 1981, (293), 133-135. DOI: 10.1038/293133a0.

[13] BOTTERILL, Tom, Steven MILLS a Richard GREEN. Refining essential matrix estimates from ransac. *Image and Vision Computing New Zealand*. 2011, , 500-505.

[14] SIMEK, Kyle. Dissecting the Camera Matrix, Part 2: The Extrinsic Matrix. In: *Http://ksimek.github.io/* [online]. b.r. [cit. 2017-04-30]. Dostupné z: http://ksimek.github.io/2012/08/22/extrinsic/

[15] KITT, Bernd. Monocular Visual Odometry using a Planar Road Model to Solve Scale Ambiguity. *Proc. of the European Conference on Mobile Robots*. 2011.

[16] SÖDERKVIST, Inge. Using SVD for some fitting problems. In: *Luleå University of Technology*. [online]. b.r. [cit. 2017-05-01]. Dostupné z: https://www.ltu.se/cms_fs/1.51590!/svd-fitting.pdf

[17] SORKINE-HORNUNG, Olga a Michael RABINOVICH. Least-Squares Rigid Motion Using SVD. In: *Interactive Geometry Lab* [online]. ETH Zurich, 2017 [cit. 2017-05-01]. Dostupné z: https://igl.ethz.ch/projects/ARAP/svd_rot.pdf

[18] BUBÁK, Martin. *Aplikace stereovize a počítačového vidění*. Brno, 2014. Vysoké učení technické v Brně. Vedoucí práce Ing. Josef Vejlupek.

[19] GEIGER, Andreas, Philip LENZ a Raquel URTASUN. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, , 8.

[20] GEIGER, Andreas, Philip LENZ, Christoph STILLER a Raquel URTASUN. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*. 2013, , 6.

[21] BOUGUET, Jean-Yves. Camera Calibration Toolbox for Matlab. In: *Jean-Yves Bouguet's WWW Homepage* [online]. b.r. [cit. 2017-05-02]. Dostupné z: http://www.vision.caltech.edu/bouguetj/calib_doc/

[22] SNAVELY, Noah, Steven M. SEITZ a Richard SZELISKI. Modeling the World from Internet Photo Collec. *International Journal of Computer Vision*. 2007, , 1-22. DOI: 10.1007/s11263-007-0107-3.

[23] ENGEL, Jakob, Thomas SCHÖPS a Daniel CREMERS. LSD-SLAM: Large-Scale Direct Monocular SLAM. *European Conference on Computer Vision (ECCV)*. 2014, , 1-16.

[24] Robust estimation: Looking for a line in corrupted data. In: *Robust estimation — openMVG library* [online]. b.r. [cit. 2017-05-15]. Dostupné z: http://openmvg.readthedocs.io/en/latest/_images/robustEstimation.png

Obsah digitálnej prílohy

Digitálna príloha obsahuje MATLAB skripty a funkcie, ktoré boli vytvorené počas riešenia tejto práce. Ďalej obsahuje funkčné ukážky riešení uvedených v kapitolách 4-6 a vzorky dát z KITTI databázy a z jazdy s vozidlom Car4. Doplnené sú aj užitočné skripty a funkcie vytvorené pre uľahčenie DAQ a spracovanie záznamov.

názov zložky	**obsah**
Calib	skripty implementujúce kalibráciu kamier kalibračné vzory
Data	vzorky dát z KITTI databázy a jazdy s Car4
Demos	skripty algoritmov VO
Others	užitočné skripty a funkcie spojené s DAQ a úpravou obrazu
VO_lib	knižnica funkcií VO, ktorá bola implementovaná podľa kapitoly 4

Tabuľka P.1 Digitálna príloha

Zoznam obrázkov

Zoznam použitých skratiek

RANSAC	Random Sample Consensus	konsenzus náhodnej vzorky
ROI	Region of Interest	oblasť záujmu
SFM	Structure from Motion	štruktúra na základe pohybu
SLAM	Simultaneous Localization and Mapping	simultánna lokalizácia a mapovanie
SVD	Singular Value Decomposition	rozklad na singulárne hodnoty
VO	Visual Odometry	vizuálna odometria