Fakulta matematiky, fyziky a informatiky
Univerzita Komenského, Bratislava

Construction of 3D map

Ján Žižka

Bratislava 2007

Univerzita Komenského, Bratislava
Fakulta matematiky, fyziky a informatiky
Katedra aplikovanej informatiky

Diplomová práca

Construction of 3D map

Ján Žižka

Vedúci práce:
Doc. RNDr. Andrej Ferko, PhD.

Bratislava 2007

Prehlásenie

Diplomovú prácu som vytvoril samostatne s použitím uvedenej literatúry.

Bratislava, 5. máj 2007 .....

Ján Žižka

Poďakovanie

Na tomto mieste by som chcel poďakovať hlavne svojmu konzultantovi Doc. Andrejovi Ferkovi za trpezlivosť, prejavenú dôveru a cenné rady.

Rovnako chcem poďakovať mojej rodine za podporu a tolerantnosť pri písaní tejto práce a počas celého štúdia.

Abstrakt

Cieľom tejto práce bolo vyvinúť online rekonštrukčný systém, ktorý pomocou mobilného robota s jednou kamerou dokáže vytvoriť 3D model okolitého prostredia. Predpokladáme planárny pohyb kamery. Kalibráciu kamery vykonáme dopredu a v ďalšom procese je konštantná. Na výpočet pohybu používame vlastný algoritmus zohľadňujúci tieto predpoklady. Takto získame vyššiu presnosť aj rýchlosť výpočtu oproti klasickému všeobecnému prístupu. Navyše v porovnaní s podobnými prácami implementujeme algoritmus rekonštrukcie hustej štruktúry, ktorú robot používa na svoju ďalšiu navigáciu. Navrhli sme pôvodnú konštrukciu lacného robota s nosičom na notebook a kamerou a celý systém sme odskúšali v reálnych podmienkach. V dostupnej literatúre nie je známa rekonštrukcia hustej štruktúry z jednej kamery s dostačujúcou presnosťou v reálnom čase.

Kľúčové slová: Vizuálne modelovanie, Štruktúra a pohyb kamery, Rekonštrukcia scény, SLAM

Obsah

Zoznam obrázkov

Zoznam tabuliek

1 Úvod

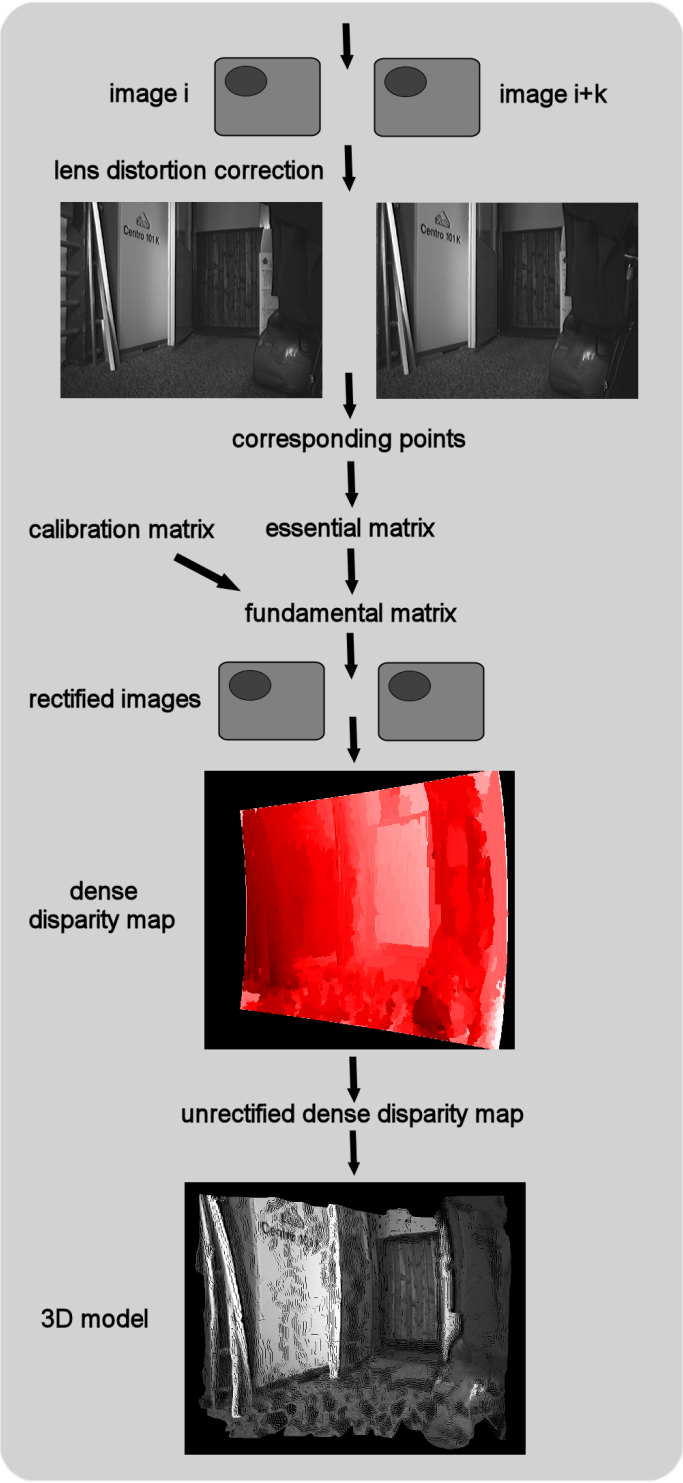
1.1 Motivácia a cieľ práce

Cieľom tejto práce bolo vyvinúť online rekonštrukčný systém, ktorý pomocou mobilného robota s jednou kamerou dokáže vytvoriť 3D model okolitého prostredia. Uplatnenia podobného systému možno nájsť v mnoho odvetviach automatizácie. Napríklad roboti priamo určení na prieskum či mapovanie komplexných priestorov alebo prostredia s nevhodnými, nebezpečnými podmienkami. Ďalej tiež autonómne mobilné stroje primárne plniace iný cieľ (strážne roboty, doručovatelia, sprievodcovia) môžu podobný systém využiť na navigáciu v okolitom prostredí. Nie je problém riešenie modifikovať a použiť na vizuálne modelovanie, čo v niektorých aplikáciách môže ušetriť mnoho času.

Na riešenie nášho problému sa zvyčajne používajú iné senzory ako kamera. Kvalitné výsledky sa dosahujú hlavne pomocou lasera. Náš systém sa snaží byť rádovo lacnejšou alternatívou ku tomuto riešeniu. Čiastočne odskúšaná je tiež metóda počítačového videnia - stereo. Avšak zatiaľ sme na plne automatický online systém počítajúci hustú rekonštrukciu nenarazili.

1.2 Problém rekonštrukcie a jeho riešenie

Štruktúra z pohybu (z anglického 'structure from motion') je definovaná ako rekonštrukcia pohybu kamery a aspoň niekoľkých bodov scény z nasnímaného videa (prípadne z niekoľko snímkov, dokonca možno neusporiadaných). Existuje niekoľko prístupov ako sa s týmto problémom vysporiadať, avšak všetky nám známe využívajú hľadanie korešpondencii istých častí obrazov na určenie vzťahov medzi týmito projekciami. Líšia sa následným výpočtom pohybu a štruktúry. Nami použitý prístup má najbližšie k [Pollefeys04], pretože autori ukázali reálnu funkčnosť tohto postupu. Zásadne však meníme charakter algoritmu, pôvodný, bežiaci čisto offline na online a v podstate všetky podúlohy riešime inak, hlavne rýchlejšie. Vybraný prichádzajúci snímok spracúvame ihneď, aby mal robot k dispozícii aktuálne informácie a dokázal sa vhodne navigovať okolitým priestorom. Celý proces spracovania ilustruje obrázok 1.



Obrázok 1: Postup spracovania

1.3 Uvažované ohraničenia a obmedzenia

Ak náš systém spĺňa napr. ohraničenia na pohyb kamery, je možné tieto zahrnúť do navrhovaného riešenia a tým dosiahnuť presnejšie výsledky, rýchlejšiu implementáciu alebo jednoduchší algoritmus. Ďalšími ohraničeniami sú špeciálne vlastnosti snímanej scény (štruktúra, textúry apod.), známe alebo nemenné vnútorné parametre kamery alebo vstup používateľa.

Nami uvažované ohraničenia sa vzťahujú na pohyb a orientáciu kamery. Kamera sa pohybuje iba v horizontálnom smere, teda jej pohyb je planárny, obrazová rovina je kolmá na rovinu pohybu, a tak rotuje iba okolo vertikálnej osi. Ďalej použité metódy predpokladajú, že kamera má vnútorné parametre počas procesu nemenné. Pre podobné aplikácie sú dostačujúce kamery s pevnou ohniskovou vzdialenosťou. Kameru teda kalibrujeme vopred, bližšie viď kapitola 2.1.

1.4 Prehľad dokumentu

Organizácia ďalšieho textu zodpovedá poradiu podúloh nášho algoritmu. Najprv naznačíme potrebné vedomosti pre prácu s kamerou a jej kalibráciu. V následných kapitolách 3-7 sa vždy na úvod snažíme čitateľa uviesť do témy, predstaviť existujúce riešenia, popísať ako ich používame či vylepšujeme, prípadne predvedieme svoje pôvodné riešenie. Kapitola 8 zhŕňa konečné výsledky práce.

2 Kamera

Na simuláciu funkcie kamery, to je mapovanie bodov 3D scény do 2D obrazovej roviny, používame model štrbinovej kamery. Tento však popisuje model reálnej kamery, ktorá mnoho krát vykazuje skreslenie šošovky (v našom prípade súdkovité skreslenie), nedostatočne, a preto túto deformáciu korigujeme. Fixná ohnisková vzdialenosť kamery poskytuje najostrejší obraz niekoľko metrov od obrazovej roviny, pretože tu predpokladáme výskyt snímanej scény.

Z praktického hľadiska je dôležité, aby kamera bola dostatočne citlivá, keďže nechceme používať žiadne prídavné svetlo. Aj napriek tomu môžeme správnu funkčnosť očakávať až niekde od 1 luxa. Ďalšou podmienkou pre náš výber kamery bola možnosť kontroly času expozície. V prípade absencie tejto funkcie by bolo potrebná softvérová korektúra kamerou automaticky nastavenej expozície a zisku,

pretože inak by rovnaký bod scény mal v rôznych obrazoch rôzne intenzity, čo výrazne zhorší funkčnosť ďalej použitých algoritmov.

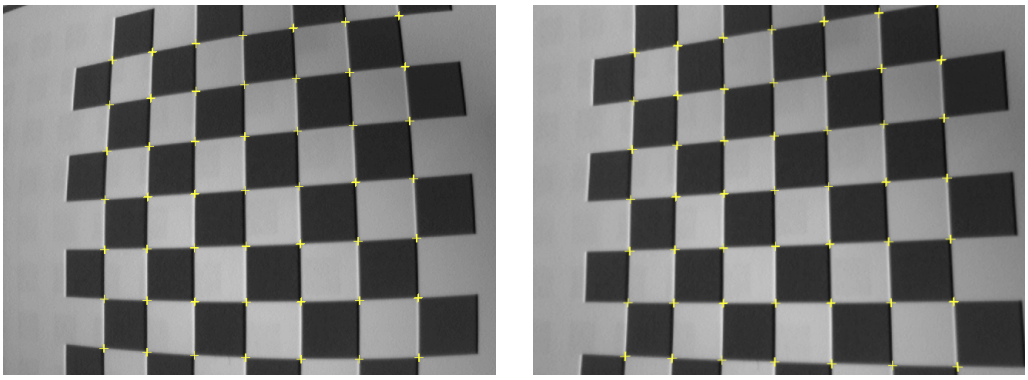
Iný nežiadúci efekt, pohybové rozostrenie, vzniká pri pohybe snímajúcej kamery. Najväčší problém nastáva pri rotácii kamery, kedy body vzdialené od kamery vytvárajú veľké rozostrenie pre ľubovoľne malý reálne použiteľný expozičný čas. Zatiaľ sa ako jediné riešenie ukazuje prednostne zvoliť menší expozičný čas a väčší polomer otáčania pri pohybe kamery.

2.1 Kalibrácia kamery

Pre použitú metódu [Zhang00] sme sa rozhodli kvôli jej nenáročnosti, nevyžaduje žiadne merania ani neobvyklé technické pomôcky. Postačuje presne poznať fyzické súradnice rohov navrhnutého rovinného kalibračného obrazu, tento zosnímať z aspoň dvoch rôznych polôh a algoritmu poskytnúť súradnice rohov kalibračného obrazu v týchto obrázkoch. Rovnako ako autor sme použili kalibračný obrazec pozostávajúci z 8x8 štvorcov usporiadaných v mriežke. Keďže autorom poskytovaný softvér "Microsoft Easy Camera Calibration Tool", riadený cez príkazový riadok, vyžaduje priamo súradnice, vytvorili sme jednoduchý program na extrakciu rohov kalibračnej mriežky z nasnímaných obrázkov a ich následné usporiadanie korešpondujúce s poradím fyzických súradníc týchto rohov. Takto je celá kalibrácia plne automatická a ukončená za pár sekúnd.

Momentálne používame druhú verziu, ktorá rovnako používa spomínanú metódu avšak zaobalenú v knižnici OpenCV, kvôli možnosti presnejšie identifikovať rohy kalibračného obrazca.

Výstupom algoritmu nie je len kalibračná matica kamery, ale tiež koeficienty popisujúce skreslenie šošovky, pomocou ktorých je možné toto odstrániť.



(a) **(b)**

Obrázok 2: Kalibračný obrazec s detekovanými rohmi pred a po korekcii skreslenia.

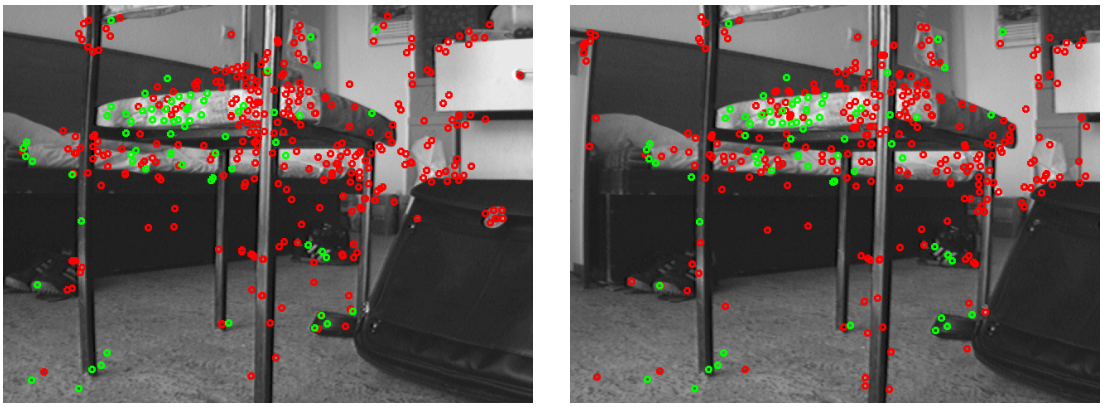
3 Hľadanie korešpondencie

Prvým krokom, ktorý nám umožní dať do súvisu jednotlivé zosnímané obrazy je nájdenie korešpondujúcich oblastí. Väčšinou sa jedná o snahu vyhľadať v dvoch obrazoch body, ktoré sú obrazom toho istého bodu v priestore. Máme dve možnosti líšiace sa hlavne predpokladanou vzdialenosťou obrazov, sledovanie význačných bodov, kde kladieme váhu na blízkosť skúmanej dvojice bodov, alebo ich hľadanie podľa lokálnych vlastností obrazu. Oba prístupy však používajú veľmi podobné metódy.

Osvedčený koncept rozdeľuje túto úlohu na dva kroky - detekcia význačných čŕt a ich následný popis, pomocou ktorého by sa mala dať ľahko vybrať najviac podobná spomedzi kandidátskych čŕt.

- Detektory. Od dobrého detektora význačných bodov sú očakávané dve základné vlastnosti - opakovateľnosť a hodnovernosť. Opakovateľnosťou je mienená schopnosť detekovať črtu v rôznych obrazoch a hodnovernosť značí, že vybraná význačná črta je dostatočne charakteristická na to, aby počet možných kandidátov na spárovanie bol malý.

- Deskriptory. Jeho úlohou je pre body označené detektorom vytvoriť istú popisnú informáciu použiteľnú na výber korešpondencie. Popis črty sa často reprezentuje ako vektor príznakov či vlastností. Je vhodné, aby bol deskriptor invariantný vzhľadom na rotáciu, škálovanie a afinné transformácie, pretože črty v rôznych obrazoch sú potom charakterizované takmer



(a) (b)

Obrázok 3: Význačné body vybrané 'Fast-Hessian' detektorom, zelené body sú hodnoverné korešpondencie párované pomocou euklidovskej vzdialenosti popisného vektora SURF.

rovnakým vektorom.

Medzi dnes bežne používané a dobre hodnotené detektory patria najmä 'Harris-Laplace' / 'Harris-Affine' detector a celkovo celá skupina založená na princípe Harrisovho detektora, 'Hessian-Laplace' / 'Hessian-Affine', 'maximally stable extremal region' detector (alebo MSER, pre krátkosť), 'an edge-based region' (EBR), 'an intensity extrema-based region' (IBR) a 'an entropy-based region' (salient regions). Bližšie informácie a tiež pomerne dôkladné hodnotenie je možné nájsť v [Mikolajczyk03]. Z ponúkaných možností sú pre nás hlavne z časového hľadiska akceptovateľné Kanade-Lucas-Tomasi(KLT) detektor [Birchfileld97] a MSER detektor.

Spomedzi deskriptorov si vedú najlepšie tie, založené na schéme SIFT ('Scale Invariant Feature Transform'). Dlho sme o týchto metódach neuvažovali, kvôli ich časovej náročnosti. Avšak posledný vývon ukázal ako pre nás ďalšiu zaujímavú alternatívu [BayTG06], kde autory predkladajú úplné riešenie, teda detektor - 'Fast-Hessian' a deskriptor 'SURF'. Približne trojnásobné urýchlenie (pre U-SURF, verzia neinvariantná vzhľadom na rotáciu, štvornásobné), predurčuje použitie týchto algoritmov aj v aplikáciách žiadajúcich niekoľko snímkov za sekundu. Toto všetko dokonca za cenu o málo kvalitnejšieho výsledku.

Testovanie na nami zosnímaných dátach ukázalo podobné výsledky ako [Mikolajczyk05] a snáď ešte podobnejšie ako [MoreelsP05].

4 Geometria dvoch pohľadov

4.1 Epipolárna geometria a fundamentálna matica

Epipolárna geometria je navrhnutá ako formalizácia a preto aj zjednodušenie predstavy o geometrii dvoch pohľadov. Dáva nám odpoveď na otázku nasledujúcej úlohy: Majme daný obrazový bod x v prvom pohľade, ktorý je priemetom 3D bodu X. Ako je ohraničená pozícia korešpondujúceho bodu x' v pohľade druhom? Odpoveď znie: Bod jedného pohľadu definuje epipolárnu priamku v pohľade druhom, na ktorej leží korešpondujúci bod x'. Algebraickou reprezentáciou epipolárnej geometrie je fundamentálna matica. Na jej výpočet je navrhnutých mnoho algoritmov a problém je už dobre zvládnutý. Keďže naším cieľom nie je predkladať ďalší výklad, odkazujeme čitateľa na tému vyčerpávajúci zdroj [Hartley04]. To, čo je pre nás prínosné, sú rôzne metódy vyvinuté pri tomto výskume, ktoré môžeme použiť na odvodenie algoritmu pre náš špeciálny prípad. Ide o zabezpečenie numerickej stability, miery vhodné na odhad chyby výpočtu fundamentálnej matice, rôzne robustné algoritmy, nelineárne minimalizáčné metódy a mnoho ďalších poznatkov.

4.2 Súvisiace práce

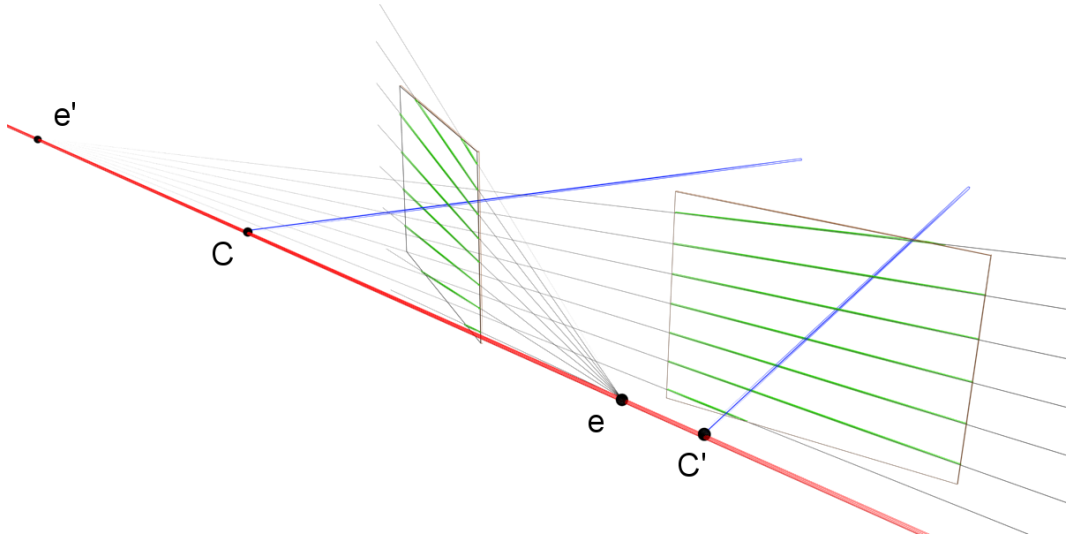
Autori [Montiel01] používajú podobné metódy ako my, avšak prvým krokom je detekcia úbežníkov postavená na nájdených hranách v obraze a predpoklade ich aspoň čiastočnej ortogonality v scéne. V reálnych scénach nie sú tieto predpoklady často splnené.

Výraznejšie iné postupy prezentujú [Li05] a [QuanWLS04]. Prvá práca je postavená na faktorizačnej metóde špeciálne pre planárny pohyb, čo ale vyžaduje výskyt každého bodu záujmu vo všetkých do výpočtu zahrnutých pohľadoch. Druhá predkladá veľmi zaujímavé riešenie, dekompozíciu 2D obrazu na jeden 1D projektívny a jeden 1D afinný obraz, z ktorých už pomerne jednoducho dokážeme rekonštruovať. Svoju teóriu autori overili experimentálne na reálnych dátach s pomerne peknými výsledkami.

Horšie sa nám odhadovala kvalita výstupu v [Oliensis96]. [VidalO02] je v podstate publikácia vylepšenej verzie algoritmu, ktorý tiež faktorizuje špeciálnu maticu. Algoritmus sme sa rozhodli neimplementovať aj kvôli jeho zložitosti.

4.3 Planárny algoritmus

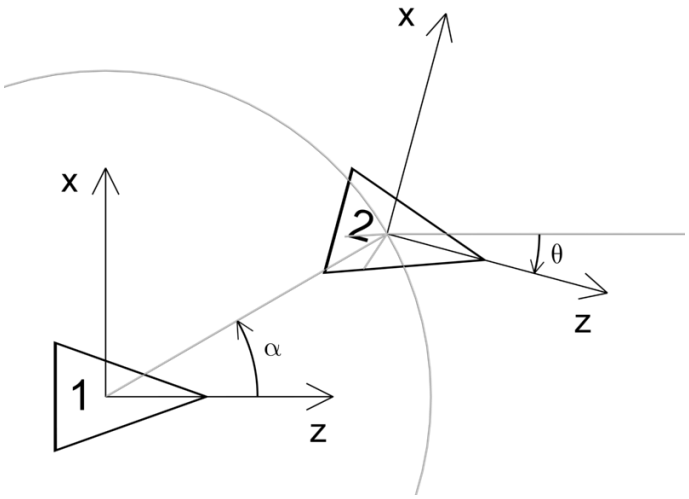
Ohraničenie pohybu kamery na planárny sa prejaví aj na epipolárnej geometrii takejto konfigurácie (viď Obrázok 4). Modré úsečky reprezentujú osi kamier, červená priamka prechádza stredmi premietania a zelenou sú vyznačené v obrazoch viditeľné časti niektorých epipolárnych priamok. Jednoduchou úvahou zistíme, že stredom oboch obrazových rovín prechádzajú horizontálne korešpondujúce epipolárne priamky a ostatné sú podľa nich súmerné.



Obrázok 4: Epipolárna geometria planárneho pohybu, C a C' sú stredy premietania, e, e' epilóly

Nech $x = PX$ je bod obrazu a $P = K[R|t]$ je matica kamery. Ak poznáme kalibračnú maticu K , potom bod $\hat{x} = K^{-1}x = [R|t]X$ nazývame obrazový bod vyjadrený v normalizovaných súradniciach. Uvažujme pár normalizovaných kamerových matíc $P = [I|0]$ and $P' = [R|t]$. Potom zodpovedajúca fundamentálna matica je maticou esenciálnou. Má tvar $E = [t]_x R$ a jej určujúca rovnica je $\hat{x}^T E\hat{x} = 0$ [Hartley04]. V prípade planárneho pohybu, predpokladajúc posun v $x - z$ rovine a rotáciu okolo osi y o uhol θ (viď Obrázok 5), esenciálna matica má nasledujúci tvar:

$$E = \begin{pmatrix} 0 & -t_z & 0 \\ t_z \cos\theta + t_x \sin\theta & 0 & t_z \sin\theta - t_x \cos\theta \\ 0 & t_x & 0 \end{pmatrix} \tag{1}$$

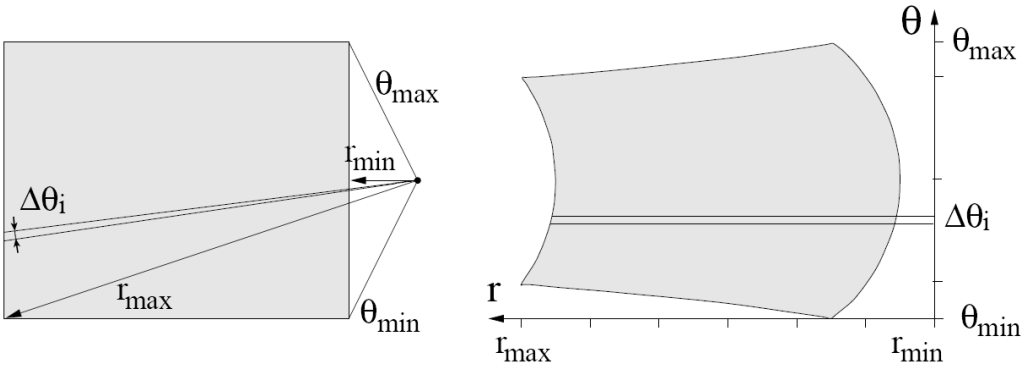


Obrázok 5: Parametre planárneho pohybu, θ je v tomto prípade záporná.

Keď si uvedomíme, že maticu môžeme získať presne až na škálovanie, stačí nám namiesto o vektore (t_x, t_z) uvažovať iba o jeho smere α. Jednoduchou úpravou dostávame:

$$E = \begin{pmatrix} 0 & -\cos\alpha & 0 \\ \cos(\theta - \alpha) & 0 & \sin(\theta - \alpha) \\ 0 & \sin\alpha & 0 \end{pmatrix} \tag{2}$$

Elementy esenciálnej matice dostaneme ako riešenie systému homogénnych rovníc, pomocou singulárnej dekompozície ako nulový priestor matice sústavy. Je potrebné poznať najmenej tri korešpondencie naproti siedmym vo všeobecnom prípade. Súradnice vstupujúce do tohto výpočtu normalizujeme podobne ako pri všeobecnom osem bodovom algoritme. Ďalšou možnosťou je použitie nelineárnej metódy. V tomto prípade, keďže neznáme nie sú nezávislé, vystačíme iba s dvomi korešpondenciami. Uspokojíme sa aj s jednoduchým Newtonovým algoritmom, dokonca na veľkej časti oboru je funkcia monotónna a za štartovací bod môžeme voliť skoro ľubovoľné rozumné hodnoty. Naša množina korešpondencii môže (a väčšinou aj obsahuje) chybné dvojice bodov, je potrebné použitie robustnej metódy schopnej sa s týmto vysporiadať. Vybrali sme algoritmus RANSAC, pretože sa ukazuje ako dobré riešenie vo všeobecnom prípade výpočtu fundamentálnej matice.



Obrázok 6: Rektifikácia, prechod k polárnym súradniciam.

5 Rektifikácia

Po tom čo máme určenú epipolárnu geometriu je možné ohraničiť hľadanie korešpondencií pre bod jedného obrazu na priamku (epipolárnu priamku) v obraze druhom a naopak. Proces rektifikácie zo vstupných obrazov a epipolárnej geometrie vytvorí obrazy, v ktorých všetky epipolárne čiary sú rovnobežné s osou obrazu. Toto je typicky prevádzané aplikáciou jedinej lineárnej transformácie, avšak rektifikované obrazy sú pomerne značne skreslené. V prípade, že sa epipóly nachádzajú v obraze (čo je pri doprednom pohybe časté) nie je možné túto metódu použiť vôbec. Druhou možnosťou môže byť polárna rektifikácia [PollefeysKG99] [Oram01].

V oboch obrazoch nájdeme spoločné regióny pomocou zadanej fundamentálnej matice. Tá prichádza na vstup orientovaná alebo spolu s niekoľkými korešpondenciami z ktorých dokážeme epipolárne priamky zorientovať. Ďalej zistíme o aký uhol budeme epipolárnu priamku pri prenikaní s obrazom otáčať, volíme ho tak, aby sme v rektifikovanom obraze žiaden pixel nestratili. Prienik takto vybranej epipolárnej priamky s interpolovaným obrazom nám vytvára riadok rektifikovaného výstupu (Obrázok 6).

Napriek všetkým poskytovaným výhodám stále vyvstáva problém s možnou veľkosťou rektifikovaného obrazu. Ak počet pixelov pôvodného obrazu je n, potom výstupný obraz má medzi n a približne $3n$ pixelov, čo by nám v horších prípadoch spôsobilo neúnosné spomalenie ďalšieho kroku. Avšak tento stav môže byť vyšetrený ešte pred štartom procedúry a obrazy buď vylúčime z následného spracovania alebo ich zmenšíme na akceptovateľnú veľkosť.

6 Hustá rekonštrukcia scény

Tradičná definícia tejto úlohy pozostáva z výpočtu mapy disparít $d(x, y)$ z dvoch rektifikovaných obrazov vzhľadom na referenčný obraz. Existujú však aj ojedinelé prístupy, kde sa korešpondencie hľadajú priamo na epipolárnych priamkach v nerektifikovaných obrazoch. Úloha nie je triviálna ani z pohľadu kvality výsledku, ani z časového hľadiska. Metódy dostatočne kvalitné pre použitie na reálnych dátach sú väčšinou rádovo pomalšie od požiadaviek našej aplikácie. Obvykle je potrebné riešiť štyri podúlohy:

1. *Funkcia podobnosti.* Popisuje rozdiel či podobnosť pixelov alebo oblastí v rôznych obrazoch. Najbežnejšie používané zahŕňajú štvorce rozdielov intenzít, absolútne rozdiely intenzít a normalizovanú koreláciu, ktorá sa správa podobne ako suma štvorcov rozdielov. Vyvinuté sú tiež miery necitlivé na zisk kamery alebo iné vplyvy, ktorými môže byť obraz zaťažený. Najčastejšie sú to rôzne metódy vychádzajúce z gradientov. Toto je však riešiteľné aj prevedením predspracovania, ako sú rôzne korekcie alebo ekvalizácia histogramu. Ďalšia metóda [BirchfieldT98] ponúka mieru necitlivú na vzorkovanie obrazu. Pixel referenčného obrazu je porovnávaný s lineárne interpolovanou funkciu ostatného obrazu, pretože pri nedeliteľnom posune môžeme správny pixel minúť.

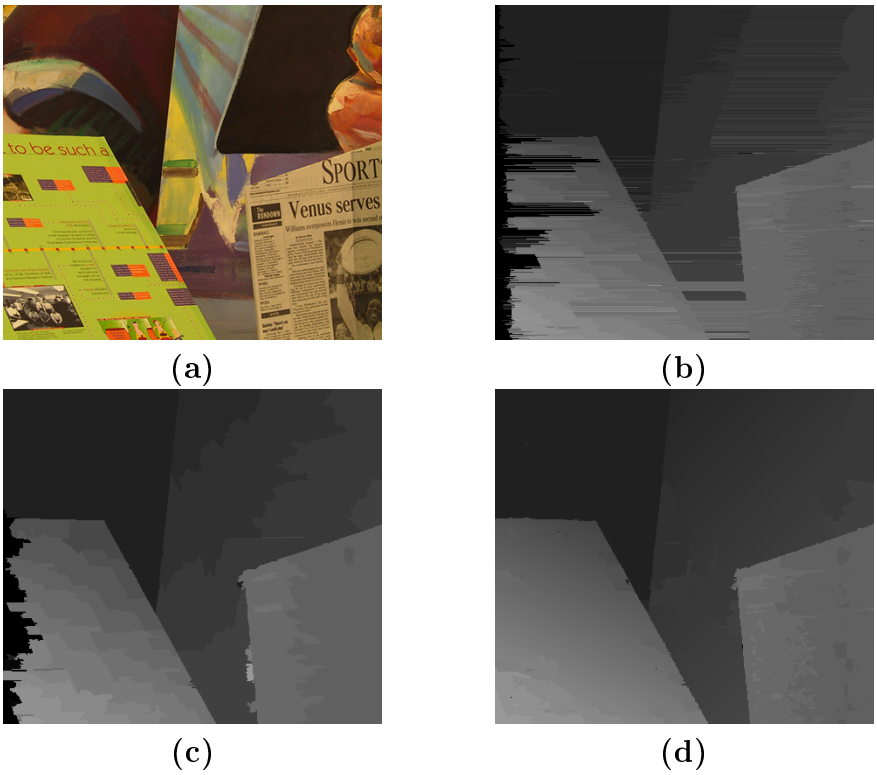
2. *Zhromažďovanie podpory podobnosti.* Suma alebo istý priemer podpory vybranej disparity v okolí daného bodu.

3. *Výpočet disparity.* Pre lokálne metódy tento krok obvykle znamená jednoduchý výber disparity s extremálnou cenou za túto voľbu spomedzi cien spočítaných v kroku 2. Naproti tomu pre globálne metódy je tento krok základom a zhromažďovanie podpory je ním často nahradené. Väčšinou sú postavené ako minimalizačné úlohy, ktorých cieľom je nájsť funkciu diparity d, ktorá minimalizuje globálnu energiu:

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d) \tag{3}$$

$$E_{data}(d) = \sum_{(x,y)} C(x, y, d(x, y)),$$

kde C je funkcia podobnosti a $E_{smooth}(d)$ zohľadňuje predpoklad hladkosti



Obrázok 7: Ľavý obrázok 'Venus' a hĺbkové mapy vyprodukované algoritmami DP na riadkoch, TreeDP a SegTreeDP

disparity. Existuje množstvo algoritmov, ktoré pre takto definovanú energiu nájdu (lokálne) minimum ('belief propagation', maximálny tok grafu, najlacnejší rez grafu).

- Globálna optimalizácia
- Dynamické programovanie
- Kooperatívne algoritmy

4. *Vylepšenie disparity.* Väčšinou sa jedná o odstránenie, vyčistenie zjavne chybných diparít. Tiež je možné dopočítať hĺbkovú mapu so 'sub-pixel' presnosťou.

Žiadna z metód globálnej optimalizácie nie je dostatočne rýchla na použitie v našej aplikácii. Jediným riešením je použitie GPU, čomu sme sa chceli vyhnúť. Z tohto hľadiska sa ukázali algoritmy využívajúce dynamické programovanie ako najlepšie riešenie. Prvým je pomerne klasický algoritmus - dynamické programovanie na riadkoch [BirchfieldT97], jeho výsledky nie sú uspokojujúce hlavne kvôli

častým chybným horizontálnym pasom. Ešte výraznejší nedostatok sa prejaví pri nedokonale rektifikovaných obrazoch, kedy sa výsledok stáva nestabilný. Preto sme sa rozhodli implementovať a prípadne vylepšiť ďalšie dva zložitejšie prístupy: dynamické programovanie na stromoch [Veksler05](TreeDP) a dynamické programovanie [1] na strome segmentov riadkov [DengL06](SegTreeDP). Hlavne posledne spomínaná metóda sa výsledkami blíži algoritmom globálnej optimalizácie, avšak beží v čase rádovo menšom.

6.1 Dynamické programovanie na stromoch

Algoritmus vychádza z myšlienky aproximovať 4 súvislý systém susedností používaný v globálnych metódach pomocou stromu, na ktorom sa istý typ funkcií energie dá minimalizovať efektívne. Na rozdiel od DP na riadkoch sa teda jedná o globálnu optimalizáciu, čo prispieva ku kvalite výsledku. Bližšie popíšeme jednotlivé kroky algoritmu.

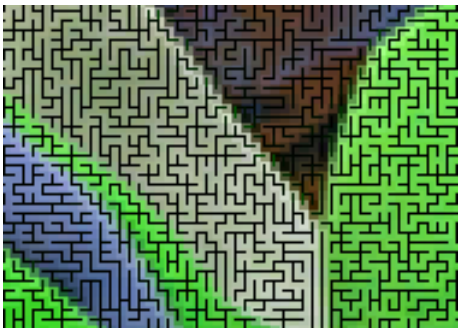
1. Obrazom preložíme 4 súvislý ováhovaný graf $G(W, F)$ tak, aby každý vrchol grafu zodpovedal jednému pixelu a hrany ich susednosti. Naším cieľom je odstrániť z grafu G hrany, ktoré najlepšie zodpovedajú nespojitostiam v hĺbkovej mape. Tu používame štandardný predpoklad - nespojitosti hĺbkovej mapy korešpondujú s nespojitosťami intenzít obrazu. Ak p a q sú susediace pixely obrazu, tak najjednoduchšie ich spoločnej hrane priradíme váhu $v_{pq} = |I(p) - I(q)|$. Testovali sme tiež váhovania odvodené od rôznych hranových detektorov, čo prinieslo jemne presnejšiu detekciu hraníc.

2. Spočítame minimálnu kostru grafu G, teda strom $T(V, E)$. Používame Kruskalov algoritmus.

3. Naším cieľom je minimalizovať na T funkciu energie všeobecne popísanú vzťahom 3. Nech $r \in V$ je koreň stromu T, hľadané minimum je nezávislé na jeho voľbe. Všetky vrcholy $v \in V$ okrem r majú rodiča $p(v)$. Ak $v \neq r$, potom minimum spomínanej funkcie energie pre podstrom zakorenený vo v môžeme písať ako funkciu od disparity vrcholu $p(v)$ - $d_{p(v)}$:

$$E_v(d_{p(v)}) = min_{d_v \in D} \left(Data(d_v) + Smooth(d_v, d_{p(v)}) + \sum_{w \in C_v} E_w(d_v) \right), \quad (4)$$

[1] skrátka DP



Obrázok 8: Strom pokrývajúci obraz

kde $C_v = \{w \mid (v,w) \in E\} - \{p(v)\}$ a D je množina uvažovaných disparít (väčšinou interval $\langle 0, dmax \rangle$). E_v budeme postupne vyhodnocovať od listov smerom ku koreňu a tým získavať minimálne energie a optimálne disparity (zatiaľ iba lokálne). Pre koreň sú však tieto hodnoty definitívne (všetky vrcholy sa podieľali na ich výpočte). Teraz spätne túto informáciu rozšírime od koreňa do celého stromu a tak zavŕšime globálnu distribúciu.

Keď označíme $|D| = k$ a dobre sa zamyslíme nad hľadaním minima pre konkrétne v, zistíme, že je potrebných k^2 testov. Z toho vyplýva celková zložitosť $O(k^2 n)$, kde n kladieme rovné počtu vrcholov stromu. Algoritmus s takouto časovou náročnosťou dokáže v praxi obstáť, avšak pre väčšie k sa stáva nevhodným pre úlohy nášho typu. Jednoduché riešenie ponúka správny (ale obmedzený) výber funkcie energie. Ak zvolíme

$$Smooth(d_p, d_q) = \begin{cases} 0 & \text{ak } d_p = d_q \\ w_{pq} & \text{inak} \end{cases}, \tag{5}$$

tak sme schopný pri pevne zvolenom $d_{p(v)}$ z k možností pre d_v vybrať správnu v konštantnom čase. Stačí rozhodnúť jednu nerovnosť medzi $E_v(d_{p(v)})$ a $E_v(d_v min) + w_{pq}$, kde pre $d_v min$ platí (vynechali sme $Smooth$ člen):

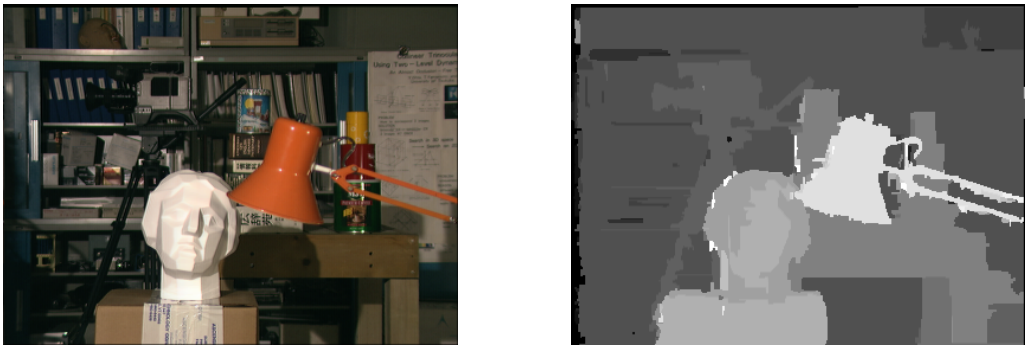
$$\forall d_v \; Data(d_v) + \sum_{w \in C_v} E_w(d_v) \geq Data(d_v min) + \sum_{w \in C_v} E_w(d_v min)$$

Avšak týchto k operácii, výber $d_v min$, prebehne iba raz na začiatku, pretože je nezávislý na $d_{p(v)}$, čo znamená, že zložitosť sme stlačili na $O(kn)$.

Testovali sme rôzne ďalšie ocenenia porušenia spojitosti disparity (w_{pq}) a ako jemné vylepšenie sa ukázalo $w_{pq}/(1.0 - abs(I(p) - I(q)))$. Teda zmena disparity

	Teddy			*Cones*		
	nonocc	*all*	*disc*	*nonocc*	*all*	*disc*
TreeDP	15.9	23.9	27.1	10.0	18.3	18.9
TreeDPMod	10.2	17.4	22.3	8.66	15.5	18.4

Tabuľka 1: Porovnanie výsledkov algoritmov v štandardnom teste. Naše úpravy zabezpečili lepšie skóre hlavne znížením odchýlky.



(a) **(b)**

Obrázok 9: Ľavý obrázok 'Tsukuba' a mapa disparít, modifikované dynamické programovanie na stromoch

bude viac akceptovaná na hranách spájajúcich pixely odlišnejšej intenzity.

Na kvalitu výsledku má tiež vplyv voľba funkcie podobnosti (kapitola 6). Pôvodne bol použitý veľmi jednoduchý, avšak dobre fungujúci výraz:

$$Data(d_p) = min\{|L(p) - R(p - d_p)|, \tau\}$$

Za cenu mierneho spomalenia sme ho nahradili spomínanou metódou, ktorá je necitlivá na vzorkovanie obrazu, čím sme dosiahli o niečo lepšiu stabilitu. Pôvodnú a našu implementáciu sme porovnali pomocou rozhrania [ScharsteinS02] na štandardnej sade obrázkov (Tabuľka 1).

Dynamické programovanie na strome segmentov riadkov obrazu

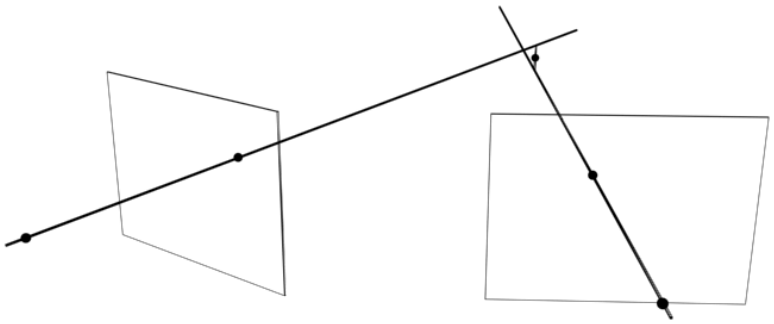
Algoritmus využíva rovnakú myšlienku ako horepopísaný TreeDP. Vylepšuje však počet zachovaných hrán grafu susedností z 50% na viac ako 80% v typickom obraze, čím dosahuje značne lepšie výsledky.

Prvým krokom je efektívna segmentácia riadkov obrazu. Tieto segmenty sa stávajú vrcholmi grafu, neskôr stromu v prístupe DP na stromoch. Namiesto

o disparite ako čísle uvažujeme o vektore priradenom každému segmentu, ktorý popisuje rovinu, do ktorej segment patrí.

7 Konštrukcia 3D modelu

Jedná sa o vytvorenie 3D bodov z niekoľkých pohľadov a ich 2D súradníc v týchto obrazoch. Základný problém spočíva v nepresnosti nabalenej predchádzajúcimi algoritmami, ktorá spôsobí, že spätné projekcie 2D obrazov nevytvárajú pôvodné 3D vzory (Obrázok 10). Medzi štandardné riešenia patrí lineárna triangulácia alebo lepšie minimalizácia geometrickej chyby alebo jej odhadu (Sampsonova aproximácia). My momentálne používame jednoduchšiu metódu.



Obrázok 10: Rekonšktukcia trianguláciou, nepretínajúce sa spätné projekcie bodov.

8 Výsledky

Na otestovanie presnosti výpočtu lokalizácie sme implementovali simuláciu pohybu kamery pomocou virtuálnej OpenGL kamery. Obrazy náhodných (avšak známych) 3D bodov slúžia ako korešpondencie v 2D obrazoch. Zvolené polohy a orientácie kamery porovnávame s vypočítaným smerom pohybu a orientáciou. Algoritmus výpočtu esenciálnej matice vykazoval na takýchto syntetických scénach nasledujúce chyby:

- Chyba smeru posunu kamery: priemer 2.5 stupňa, medián 2.0 stupňa

Popis	Čas
Korekcia - výpočet neskreslených súradníc význačných bodov	1ms
Korekcia - celý obraz	26ms
Korešpondencia - spočítanie SURF na jednom obraze	125ms
Korešpondencia - výber podobných SURF vektorov	50-150ms
Korešpondencia - KLT, inicializácia	20ms
Korešpondencia - KLT, sledovanie medzi dvomi obrazmi	50ms
Lokalizácia - lineárne bez RANSAC	1ms
Lokalizácia - lineárne s RANSAC	150ms
Lokalizácia - nelineárne	60ms
Rektifikácia - lineárna	120-425ms
Rektifikácia - polárna	100-160ms
Disparita - DP na riadkoch, $D_{max} = 20 - 80$	50-110ms
Disparita - TreeDP, $D_{max} = 20 - 80$	200-500ms
Disparita - SegTreeDP, $D_{max} = 20 - 80$	160-500ms
3D - triangulácia	50ms

Tabuľka 2: Časy implementovaných algoritmov v typických prípadoch. Z tých, ktoré sú závislé na veľkosti obrazu, korešpondencie bežia na 640×480, ostatné na 320×240.

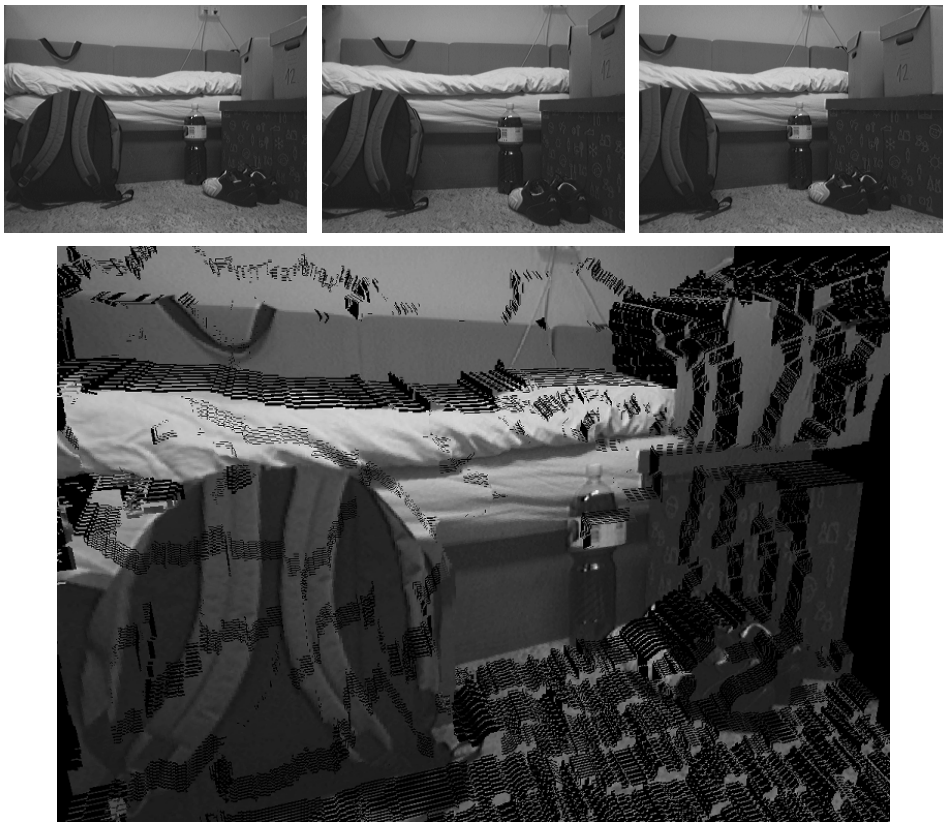
- Chyba orientácie kamery: priemer 1.1 stupňa, medián 0.9 stupňa

Je dôležité, že takáto chyba je dostatočne malá na to, aby ďalšie algoritmy pokračovali korektne. V reálnej scéne musíme očakávať horšie výsledky najmä kvôli šumu, nepresnostiam kalibrácie a korekcie a tiež nedokonalému planárnemu pohybu.

Kvôli rôznym ťažko odstrániteľným chybám (ako napríklad niektoré zlé korešpondencie) celý proces raz za čas zlyhá, čo je však do istej miery prirodzené už z podstaty problému (príkladom môže byť scéna pozostávajúca z bielej steny). Nepríjemnosti nastávajú pri nekonvexnostiach scény (z pohľadu robota), kedy sa aj miernym pootočením rázne mení pre robota viditeľná časť scény. Máme rozpracované ošetrenie tejto situácie sledovaním prudkých zmien v riedkej štruktúre význačných bodov.

Na druhej strane bezproblémový beh algoritmu zabezpečí pomerne kvalitnú rekonštrukciu (Obrázok 11 a 12).

Obrázok 11: Príklad reálnej zrekonštruovanej scény.



Obrázok 12: Vstupné obrazy, zrekonštruovaná scéna z vyhľadenej hĺbkovej mapy.

9 Implementácia

Keďže celý algoritmus je relatívne jednoducho štruktúrovaný, nebolo náročné rozdeliť jednotlivé úlohy do modulov, z ktorých každý možno v prípade nutnosti nahradiť, čo však aj tak časom spôsobovalo neporiadok v pomerne rozsiahlom kóde. Viac snahy bolo potrebné venovať efektívnosti implementácie. Momentálne je rozpracovaná aj verzia využívajúca dvojjadrové procesory.

9.1 Pracovné prostredie

Väčšina projektu bola vytvorená v prostredí 'Visual C++ 2005, Express Edition'. Kvôli efektívnosti a nepotrebe užívateľského vstupu sú vyprodukované aplikácie prevažne konzolové programy, čím sa tiež zvýši šanca skompilovania na iných platformách. To je podporené aj dobrou portabilitou knižníc.

9.2 Knižnice

9.2.1 OpenCV

Teda 'Open Source Computer Vision', využívame na komunikáciu s webkamerou, správu pamäte snímaných obrazov, základné obrazové transformácie, kalibráciu kamery a v prvých verziách aj na hustú rekonštrukciu pomocou implementovaného algoritmu dynamického programovania na riadkoch.

9.2.2 VXL

VXL (the Vision-something-Libraries) je menej známa knižnica vyvíjaná prevažne na popredných univerzitách, zameraná hlavne na počítačové videnie. Poskytuje však prostriedky na vývoj celých aplikácii, zahŕňa rôzne numerické algoritmy, dátové štruktúry, metódy spracovania obrazu, sledovanie význačných bodov, geometriu a algoritmy 3D rekonštrukcie, vlastný GUI dizajn a iné.

9.2.3 OpenGL

Našla uplatnenie pri rôznych testoch, simulácii a odhade chyby výpočtu pohybu kamery a zobrazení výslednej 3D scény.

10 Záver a budúca práca

Základné výsledky tejto práce sme prihlásili do súťaže o Cenu prof. Kuniiho, kde sa podarilo získať 3. miesto v novembri 2006. Poster [Zizka07] sme publikovali na medzinárodnej konferencii SCCG 2007. Poster uvádzame v prílohe. Vylepšená verzia práce získala ocenenie Laureát ŠVK FMFI 2007 a postúpila do medzinárodnej súťaže SVOČ v Olomouci.

Tak ako pri každom riešení problému počítačového videnia sme narazili na najzávažnejšie problémy pri experimentoch v reálnych podmienkach. Za jeden z prínosov pokladáme riešenia práve týchto problémov.

Vytvorili sme kompletný systém pre realtime rekonštrukciu prostredia pomocou mobilného autonómneho robota. Naša práca je unikátna vytváraním hustej štruktúry, ktorá sa používa na online navigáciu. V terajšej verzii však systém nie je schopný vysporiadať sa s naozaj všetkými podmienkami reálneho sveta.

Preto do budúcnosti plánujeme ďalej zlepšiť robustnosť lokalizácie a možno stabilitu algoritmu hustého sterea. Bližšie otestovať vzťah medzi 'šírkou sterea' a kvalitou rekonštrukcie. Je možnosť uvažovať nad integráciou predpokladu planarity ešte do skorších vrstiev spracovania, napríklad do sledovania význačných bodov. Alebo naopak, vyskúšať, či nie je zvládnuteľné všeobecné riešenie, hlavne kvôli širšej aplikovateľnosti.

11 Prílohy

11.1 Hardvér

Náš autonómny mobilný robot (Kyklop) je navrhnutý hlavne na testovanie rôznych metód počítačového videnia. Najmä kvôli výpočtovej náročnosti sa predpokladá použitie notebooku ako riadiacej jednotky. Implementovaný algoritmus má možnosť nezávisle riadiť dve hnané kolesá a snímať prostredie pomocou webkamery. Riadenie je elegantne zabezpečené cez USB rozhranie.

Z ponúkajúcich sa alternatív návrhu konštrukcie(viď Obrázok 13) sme sa rozhodli pre umiestnenie notebooku priamo na robota. Ostatné možnosti(hlavne vzdialená kontrola cez bezdrôtovú sieť) by síce istú časť práce zjednodušili, ale na druhej strane ani problémy s komunikáciou nie sú zanedbateľné. Ako vážnejšia prekážka by sa ukázala rýchlosť prenosu. Samotná kamera pri tridsiatich snímkoch za sekundu(nekomprimované a farebné) vyžaduje skoro 30MB/s. Z tohto dôvodu sú rozmery robota v pôdoryse o niečo väčšie ako rozmery bežného notebooku a výška približne 30cm. Kvôli nízkej hmotnosti a relatívnej pevnosti sme na stavbu použili prevažne hliníkové profily.

Na pohon slúžia najväčšie dostupné modelárske servá. Pôvodné ovládanie (šírka impulzu zodpovedá natočeniu výstupného rotora v rozmedzí $0 - 180$ stupňov) sme odstránením fyzického dorazu a elektronického obmedzenia zmenili na riadenie rýchlosti šírkou vysielaného impulzu. Maximálna rýchlosť pohybu je okolo 25cm za sekundu.

Elektronika zabezpečujúca komunikáciu medzi hardvérom robota a notebookom je pripravená na pripojenie ďalších vstupov(napríklad senzory) a výstupov. Je napájaná priamo z USB rozhrania a od napájania výkonnej časti galvanicky oddelená, čím sme sa vyhli komplikáciám so stabilitou napätia. Ostatné napájanie poskytujú štyri nabíjateľné NiCd monočlánky(teda 4×1.2V) s kapacitou 3000mAh.

11.2 Construction of 3D map, poster publikovaný na Spring Conference on Computer Graphics

Poster nájdete na nasledujúcej strane.

Construction of 3D Map

Ján Žižka*
Comenius University, Mlynska dolina, SK-842 48 Bratislava, Slovakia
Faculty of Mathematics, Physics and Informatics

Abstract

The aim of this work is to develop the online structure from motion system, which can create a 3D model of surrounding environment using a mobile robot with a single camera. We assume that camera motion is constrained to plane. Camera calibration is completed in advance and it remains constant. For camera movement computation we have introduced our original algorithm based on these assumptions. This method offers better speed and a higher precision. Additionally, unlike other known solutions, we create dense structure reconstructed.

Keywords: Visual modeling, Structure-from-Motion, Dense reconstruction, SLAM

1 Introduction

One can find similar systems applied in many branches of automation. For instance, the robots dedicated to search or mapping of complex, dangerous, or human-unsuitable environments. Another applications include mobile machines primarily intended for another task (security guard, delivery, tourguide robot). They can utilize the mobile system for the navigation in the surroundings. Typical solutions of our problem use sensors different from camera. High quality results are obtained with lasers. Our system intends to offer much cheaper alternative. In the literature, there is partially tested shape-from-stereo approach. To our best knowledge, we have not found any fully automatic solution with dense reconstruction.

2 Calibration and lens distortion correction

We have chosen the calibration method [Zhang 2000]. It is precise enough and simple in using resources. However, it was necessary to implement procedures for extraction and correct classification of corners of calibration object. The lens has not a wide field of view, but it dislayed the barrel distortion. This was fixed using the OpenCV library.

3 Feature tracking

Thanks to having control of robot motion, we can secure by selection of smoothness and speed very good tracking conditions. This minimizes the number of outliers in correspondences and loss of features being tracked. For our work we have selected free KTL feature tracking toolkit [Birchfileld] and modified Lucas-Kanade optical flow algorithm implemented in OpenCV.

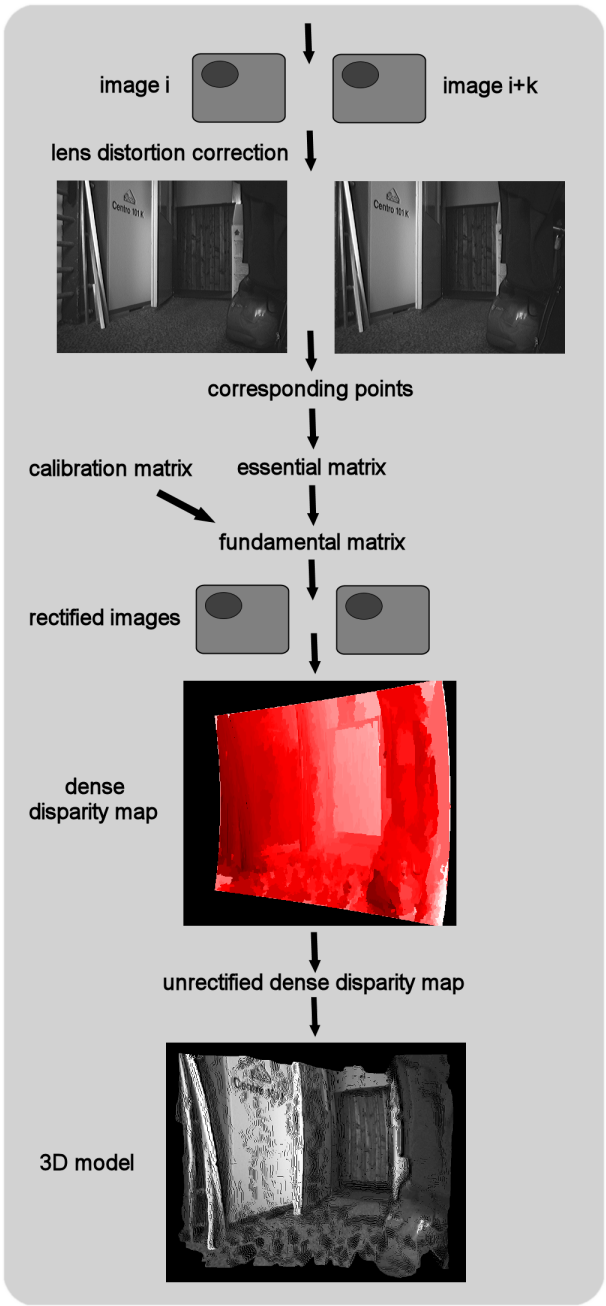


Figure 1: Processing pipeline

*e-mail: jzizka@gmail.com

4 Pose estimation

Let $x = PX$ be a point in the image and $P = K[R|t]$ be decomposed camera matrix. If the calibration matrix K is known, then we can obtain the point $\hat{x} = K^{-1}x = [R|t]X$ - image point expressed in normalized coordinates. Consider a pair of normalized camera matrices $P = [I|0]$ and $P' = [R|t]$. The corresponding fundamental matrix is essential matrix. It has the form $E = [t]_x R$ and its defining equation is $\hat{x}'^T E\hat{x} = 0$ [Hartley and Zisserman 2004]. In the case of planar motion, assuming translation in $x-z$ plane and rotation around y-axis by an angle θ, the essential matrix has the following form:

$$E = \begin{pmatrix} 0 & -t_z & 0 \\ t_z\cos\theta + t_x\sin\theta & 0 & t_z\sin\theta - t_x\cos\theta \\ 0 & -t_x & 0 \end{pmatrix} \quad (1)$$

The elements of the essential matrix can be obtained as a least squares solution of a system of homogeneous equations. At least three point correspondences are required. We have used RANSAC algorithm for the robust estimation of the essential matrix.

5 Rectification

Once the epipolar geometry has been determined it is possible to constrain the match for a point in one image to lie on a line (the epipolar line) in the other image and vice versa. The process of rectification makes all matching epipolar lines coincident and parallel with an image axis. This is typically performed by applying a single linear transformation, but rectified images are heavily distorted. The second option can be polar rectification [Oram 2001]. In spite of all the benefits, there are still problems with size of rectified image. If the number of pixels in the original image is n then the rectified image has between n and approximately $3n$ pixels. This can be determined before the start of procedure and the images can be either dropped or resized.

6 Dense matching

This is the most time consuming step. The majority of developed methods compute disparity in several tens seconds. Dynamic programming on a scanline is a sort of computational efficient approach, but its performance is far from the state of the art. Small difference in alignment of images induces unstable result and therefore this method is not suitable for our purpose. More stable and a bit higher quality results provides dynamic programming on a tree [Veksler 2005]. The processing time of our implementation for each frame is only 500ms. Another sufficiently fast method [Deng and Lin 2006] can be compared with the quality of results of the best algorithms. It uses a 3-parameter linear transform label space which can well model slanted planes in the scene and give a sub-pixel disparity map as the results.

7 Conclusion

We have created a complete system for real-time reconstruction of environment with the mobile autonomous robot. Our work is unique in creating dense reconstruction for online navigation. However, in current version the system is not ready to handle really all the real-world conditions. For future work we plan to improve the robustness of the pose estimation and eventually also the dense matching.

Figure 2: Tsukuba disparity map, modified dynamic programming on a tree

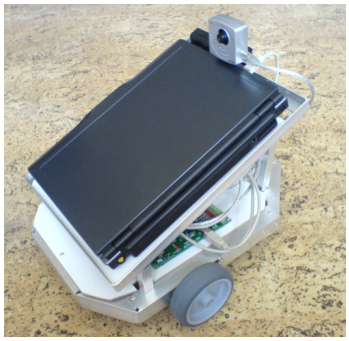


Figure 3: Constructed robot

Acknowledgment

This research was partially supported by APVT grant Rozpoznávanie a sledovanie tváre ako súčasť univerzálneho systému na spracovanie videosekvencií No. APVT-20-031304.

References

BIRCHFILELD, S. *KLT: An Implementation of Kanade-Lucas-Tomasi Feature Tracker*.

DENG, Y., AND LIN, X. 2006. A fast line segment based dense stereo algorithm using tree dynamic programming. In *ECCV (3)*, 201–212.

HARTLEY, R. I., AND ZISSERMAN, A. 2004. *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press, ISBN: 0521540518.

ORAM, D. 2001. Rectification for any epipolar geometry. In *BMVC*.

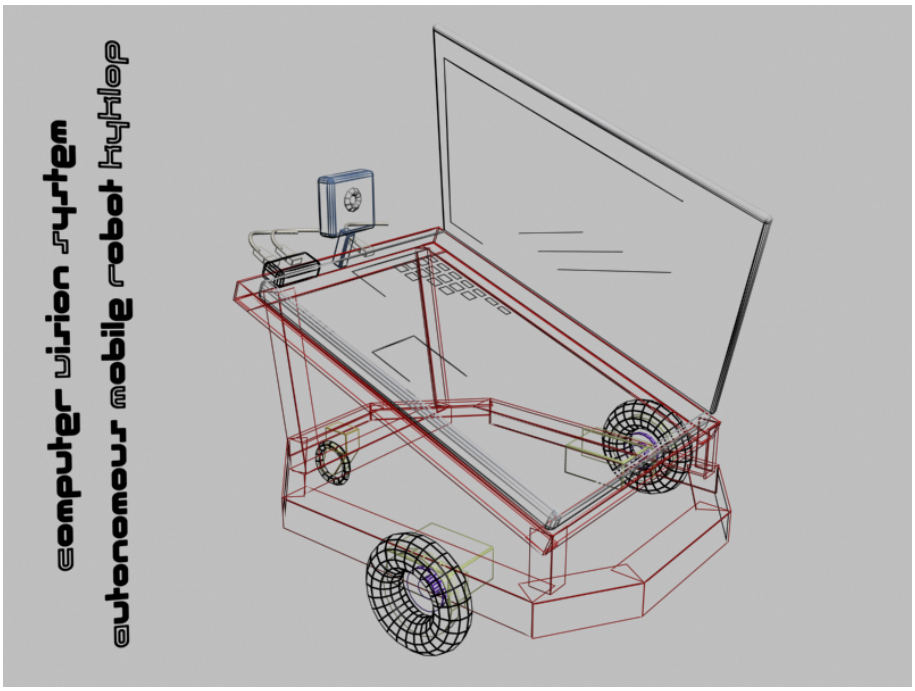
VEKSLER, O. 2005. Stereo correspondence by dynamic programming on a tree. In *CVPR (2)*, 384–390.

ZHANG, Z. 2000. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell. 22*, 11, 1330–1334.

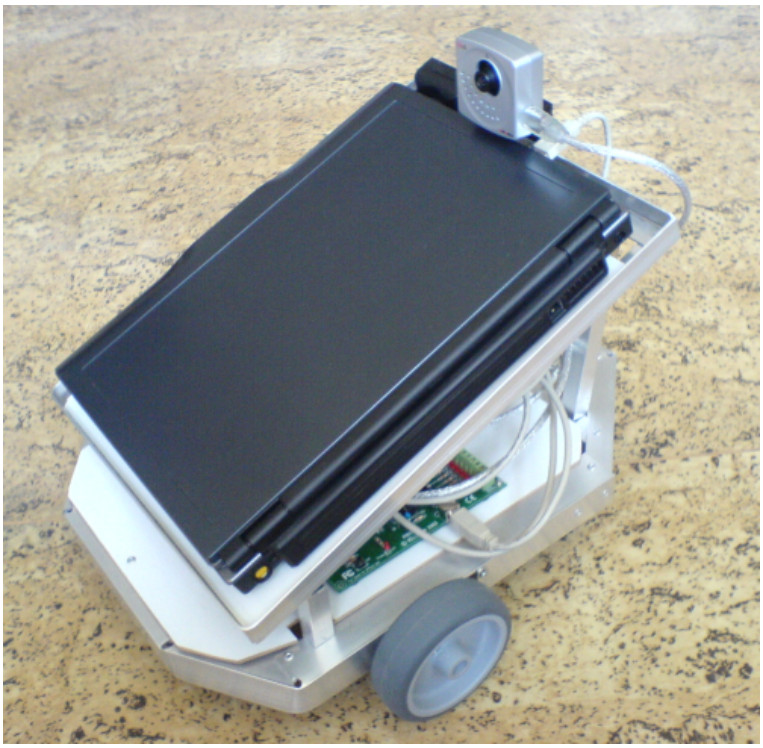
11.3 Zdroje

Online zdroje: `jzizka.googlepages.com`

- Text tejto práce vo formáte pdf.
- Poster [Zizka07] publikovaný na konferencii SCCG 2007.
- Práca postupujúca na SVOČ 2007.
- Citované a iné súvisiace články.
- Vyvinuté aplikácie užitočné pre rekonštrukciu.



Obrázok 13: Návrh konštrukcie robota, drôtený model.



Obrázok 14: Skonštruovaný robot.

Literatúra

[BayTG06] Herbert Bay, Tinne Tuytelaars, and Luc J. Van Gool. Surf: Speeded up robust features. In *ECCV (1)*, pages 404–417, 2006.

[BiberFD05] Peter Biber, Sven Fleck, and Tom Duckett. 3d modeling of indoor environments for a robotic security guard. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, page 124, Washington, DC, USA, 2005. IEEE Computer Society.

[BirchfieldT97] Stan Birchfield and Carlo Tomasi. Depth discontinuities by pixel-to-pixel stereo. In *ICCV*, pages 1073–1080, 1997.

[BirchfieldT98] Stan Birchfield and Carlo Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(4):401–406, 1998.

[Birchfileld97] Stan Birchfield. *KLT: An Implementation of Kanade-Lucas-Tomasi Feature Tracker.*

[DavisonRMS07] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[DengL06] Yi Deng and Xueyin Lin. A fast line segment based dense stereo algorithm using tree dynamic programming. In *ECCV (3)*, pages 201–212, 2006.

[Hartley04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[Li05] Jian Li and Rama Chellappa. A factorization method for structure from planar motion. In *WACV-MOTION '05: Proceedings of the IEEE Workshop on Motion and Video Computing (WACV/MOTION'05) - Volume 2*, pages 154–159, Washington, DC, USA, 2005. IEEE Computer Society.

[Mikolajczyk03] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors, 2003.

[Mikolajczyk05] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1/2):43–72, 2005.

[Montiel01] J. M. M. Montiel and A. Zisserman. Automated architectural acquisition from a camera undergoing planar motion. In *Int. Symp. on Virtual and Augmented Architecture (VAA01)*, pages 207–218, Dublin, Ireland, June 2001. http://www.robots.ox.ac.uk/∼vgg.

[MoreelsP05] Pierre Moreels and Pietro Perona. Evaluation of features detectors and descriptors based on 3d objects. In *ICCV*, pages 800–807, 2005.

[NisterNB04] David Nister, Oleg Naroditsky, and James Bergen. Visual odometry. *CVPR*, 01:652–659, 2004.

[Oliensis96] J. Oliensis. Structure from linear or planar motions, 1996.

[Oram01] Daniel Oram. Rectification for any epipolar geometry. In *BMVC*, 2001.

[Pollefeys04] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *Int. J. Comput. Vision*, 59(3):207–232, 2004.

[PollefeysKG99] Marc Pollefeys, Reinhard Koch, and Luc J. Van Gool. A simple and efficient rectification method for general motion. In *ICCV*, pages 496–501, 1999.

[QuanWLS04] Long Quan, Yichen Wei, Le Lu, and Heung-Yeung Shum. Constrained planar motion analysis by decomposition. *Image Vision Comput.*, 22(5):379–389, 2004.

[ScharsteinS02] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[Thrun03] Sebastian Thrun. Robotic mapping: a survey. pages 1–35, 2003.

[Veksler05] Olga Veksler. Stereo correspondence by dynamic programming on a tree. In *CVPR (2)*, pages 384–390, 2005.

[VidalO02] Rene Vidal and John Oliensis. Structure from planar motions with small baselines. In *ECCV (2)*, pages 383–398, 2002.

[Zhang00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, 2000.

[ZhichaoB06] *Qualitative vision-based mobile robot navigation*, 2006.

[Zizka07] Ján Žižka. Construction of 3d map, poster. In Martin Samuelčík, editor, *Spring Conference on Computer Graphics*, pages 33–34. Comenius University, Bratislava, 2007.