Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE

Daniel Martinec

**Konzistentní rekonstrukce scény
z velkého množství snímků**

Katedra softwarového inženýrství
Vedoucí diplomové práce: Ing. Tomáš Pajdla
Studijní program: informatika

bla

Děkuji vedoucímu diplomové práce panu Ing. Tomáši Pajdlovi a doktorandům z Centra strojového vnímání FEL CVUT za vhodné připomínky k práci. Dále bych chtěl poděkovat svému bratrovi za sdílení počítače pro práci na tomto díle.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce

V Praze dne 15.12.2000 Daniel Martinec

bla

Abstrakt

This diploma thesis proposes a new method for recovery of projective shape and motion from multiple images of a scene by factorization of a matrix containing the images of all points in all views. This factorization is only possible when the image points are correctly scaled and when all image points in all images are known. This method uses epipolar geometry between images to recover these scalings.

An advantage of this method is that it works even if some images of points in views are unknown. The missing elements of the matrix are filled by operations of linear algebra only.

The method has an alternate in which no scalings of image points are recovered but only some of all scalings are used. These scalings are supposed to be computed by other methods.

The major theoretical contribution of this thesis is an analysis of recoverable configurations of missing points and scalings.

bla

Obsah

Kapitola 1

Úvod

Počítačové vidění prochází v posledních letech velkým rozvojem, jeho poznatky se stále více aplikují ve vědě i v průmyslu. Pomocí počítačového vidění se např. vyhodnocují snímky lidského těla v medicíně ke stanovení diagnózy, dále se využívá při řízení robotů nebo naváděných raket.

Informace o okolním světě, které se zpracovávají algoritmy počítačového vidění, se získávají pomocí kamer a zaznamenávají se buď do sekvencí nebo do jednotlivých obrázků. Aby bylo možné tyto obrázky využít, je potřeba porozumnět tomu, co zachycují. Často se proto vytváří jakási představa o prostorovém uspořádání zachycených objektů, tzv. třídimenzionální (dále jen 3D) model scény. Proces získání tohoto modelu se nazývá *rekonstrukce scény*.

Dnes může být počítačové vidění blízké prakticky komukoliv. Jediné, co člověk potřebuje k tomu, aby mohl vytvářet 3D-modely svého okolí, je fotoaparát nebo kamera a osobní počítač s programy pro rekonstrukci scény. Objekt(y), např. budovu nebo skupinu hraček, je potřeba nafotit nebo nafilmovat z několika stran a úhlů. Aby byly obrázky srozumitelné pro počítač, je nutné je převést do digitální formy, např. pomocí scaneru[1]. Nakonec se obrázky zpracují programy pro rekonstrukci scény, čímž se získá její 3D model. Od této chvíle má člověk něco více než jen nepohyblivé obrázky nebo film, který lze pouštět jen dopředu a zpět. Má 3D model scény, na který se může dívat z jakýchkoliv pozic a směrů, dokonce i zevnitř, a který může dále zpracovávat, např. vytvářet animace průchodu scénou.

Tato diplomová práce si neklade za cíl postihnout celou problematiku softwaru, který by potřeboval člověk z příkladu výše. Zabývá se pouze jedním z několika podproblémů rekonstrukce scény, viz 2.1.

[1] Dnes začínají být cenově dostupné i digitální fotoaparáty.

Kapitola 2

Definice problému a jeho rozbor

Rekonstrukce scény je často používaný proces v počítačovém vidění. Cílem tohoto procesu je z obrázků nějaké scény sestrojit její vnitřní 3D model. Obrázky se snímají kamerami různých typů.

Je několik důvodů, proč se problém rekonstrukce scény považuje za těžký, viz [3]. Obrázky scény jsou 2D, ale vnitřní model scény je 3D. Dále nemusejí být známé parametry kamer. Není jasné, co na různých snímcích patří k sobě, tj. co je obrazem toho samého bodu prostoru. Tento problém se nazývá problém *korespondence bodů*.

2.1 Proces rekonstrukce

Rekonstrukci scény lze rozdělit podle [3] na následující kroky:

1. Určení význačných bodů na snímcích.
2. Nalezení korespondencí význačných bodů mezi jednotlivými snímky. Tyto korespondence se zapisují do matice, tzv. *joint image matrix*, dále jen *JIM*.
3. Nalezení 3D modelu scény pomocí korespondencí význačných bodů.

Tato diplomová práce se zabývá pouze krokem 3 v procesu rekonstrukce, zejména se pak věnuje situaci, když některá políčka v JIM chybí.

Kapitola 3

Stav poznání

Algoritmy pro rekonstrukci scény lze rozdělit podle toho, kterých kroků rekonstrukce z 2.1 se týkají. Tyto kroky se zpravidla řeší odděleně, nicméně existuje algoritmus [6] (viz A.2), který řeší všechny z nich najednou. Kromě faktorizace (viz A.1), kterou používají všechny algoritmy, o kterých se zmíníme, je tento algoritmus postaven též na poznatcích o stereovidění, neboť lze použít pouze se specialním hardwarem.

V dalším přehledu se zaměříme na algoritmy, které řeší při rekonstrukci pouze krok 3. Tyto algoritmy tedy vycházejí z připravené JIM a hledají 3D model scény. Liší se podle typu kamery, kterou byly obrázky získány, a podle toho, zda fungují, i když chybí některá políčka v JIM.

Ortografická kamera

Problém rekonstrukce scény s ortografickou kamerou řeší např. algoritmus pánů Tomasi a Kanade [11] (viz 5.1) a Jacobsův algoritmus [7]. Oba algoritmy vykazují uspokojivé výsledky i při velkém šumu v datech a dokáží si dobře poradit i s chybějícími daty, viz 6.1, 6.2.

Perspektivní kamera

Problém rekonstrukce scény s perspektivní kamerou je mnohem těžší než problém rekonstrukce scény s ortografickou kamerou. Je to proto, že projekce bodu perspektivní kamerou platí až na nějaký skalární násobek. Tomuto násobku se říká *projektivní hloubka* nebo *měřítko*.

K odhadování měřítek se používají vztahy mezi snímky vyjádřené fundamentálními maticemi (algoritmus Triggse a Sturma [10], viz 7.1), trifokálními a kvadrifokálními tenzory (viz [4, 5]). Pomocí lineární algebry lze zachytit vztahy maximálně mezi čtyřmi obrázky (viz [4]). Za kvadrifokálním tenzorem už tedy žádný další nelze definovat.

Algoritmus pánů Mahamuda a Heberta [8] počítá měřítka iterativně bez použití tenzorů. Tento algoritmus ale funguje pouze pro slabě perspektivní kamery, tj. projektivní hloubky jsou velmi blízko jedné.

Pokud se podaří měřítka spočítat, je zbytek rekonstrukce snadný v případě, že je JIM kompletní. Nicméně existují algoritmy pro některé konfigurace korespondencí bodů (např. koláčová — cake [12]), avšak dobré obecné řešení tohoto problému nás teprve čeká. Proto je mu nejvíce soustředěna pozornost této diplomové práce.

Kapitola 4

Popis přínosu

Tato diplomová práce přináší nový algoritmus pro rekonstrukci scény s perspektivní kamerou. Jeho hlavním technickým přínosem je praktická metoda, která nevyžaduje, aby byly známé všechny obrazy bodů.

Algoritmus má dvě varianty. První varianta počítá projektivní hloubky neboli měřítka obrazů bodů pomocí fundamentálních matic a epipólů. Druhá varianta algoritmu žádná měřítka nepočítá, nýbrž využívá pouze znalosti těch měřítek, která jsou poskytnuta „zvenčí“.

Teoretickým přínosem práce je rozbor, pro jaké konfigurace chybějících bodů a měřítek je možné provést rekonstrukci. Vynikajícím výsledkem je např. to, že k rekonstrukci scény se dvěma obrázky stačí pouze pět společných bodů a tři společná měřítka.

Přínosem této práce je jistě naše „svazová“ formulace problému rekonstrukce z dat zatížených šumem, která přináší dodatečný (algebraický) vhled do metody publikované Jacobsem.

Byla vypracována analýza možných průběhů algoritmu a byl nalezen optimální postup doplňování chybějících dat.

Kapitola 5

Scéna s ortografickou kamerou

Předpokládejme, že máme n význačných bodů na m snímcích. Souřadnice těchto bodů $\{\mathbf{q}_j^i = (u_j^i, v_j^i)^T | i = 1, \ldots, m, j = 1, \ldots, n\}$ zapíšeme do matice J velikosti $2m \times n$:

$$J = \begin{pmatrix} \mathbf{q}_1^1 & \ldots & \mathbf{q}_n^1 \\ \vdots & & \vdots \\ \mathbf{q}_1^m & \ldots & \mathbf{q}_n^m \end{pmatrix} = \begin{pmatrix} u_1^1 & \ldots & u_n^1 \\ v_1^1 & \ldots & v_n^1 \\ \vdots & & \vdots \\ u_1^m & \ldots & u_n^m \\ v_1^m & \ldots & v_n^m \end{pmatrix}.$$

Chceme nalézt matici kamer $P \in \mathcal{R}^{2m \times 4}$ a matici souřadnic bodů v prostoru $X \in \mathcal{R}^{4 \times n}$ tak, že

$$J = PX.$$

Naši pozornost zaměříme na dva algoritmy pro rekonstrukci scény s ortografickou kamerou. Algoritmus Tomasi a Kanade [11] si zde ukážeme, Jacobsovu algoritmu [7] se budeme věnovat až v příští kapitole, protože je určen pro scény, v jejichž JIM chybí políčka.

5.1 Rekonstrukce podle Tomasi a Kanade

Tomasi a Kanade popsali algoritmus [11], který vychází z myšlenky, že JIM lze upravit do takového tvaru, o kterém dokázali, že má při absenci šumu hodnost maximálně tři. Takovou matici faktorizují pomocí svd a provedou zpětnou substituci. Algoritmus není potřeba upravovat pro případ, že je v datech šum. Následuje popis algoritmu.

Body v obrázcích upravíme odečtením jejich střední hodnoty:

$$\begin{aligned}\mathbf{t}^i &= \frac{1}{n}\sum_{j=1}^{n}\mathbf{q}_j^i \\ \tilde{\mathbf{q}}_j^i &= \mathbf{q}_j^i - \mathbf{t}^i,\end{aligned} \tag{5.1}$$

čímž získáme novou matici $\tilde{J} = [\tilde{\mathbf{q}}_j^i]$, kterou nazveme *centrovaná JIM*. Pomocí svd (viz větu A.1) rozložíme $\tilde{J}$ na

$$\tilde{J} = U\Lambda V^T,$$

kde U má velikost $2m \times n$, Λ je diagonální matice, jejíž položky jsou vlastní čísla $\lambda_1 \geq \ldots \geq \lambda_n$ setříděná v nerostoucím pořadí a V má velikost $n \times n$. O $\tilde{J}$ je v [11] dokázáno, že má při absenci šumu hodnost maximálně tři, proto jsou $\lambda_4 = \ldots = \lambda_n = 0$, takže lze výraz přepsat na

$$\tilde{J} = U'\Lambda' V'^T,$$

kde matice U' jsou první 3 sloupce U, diagonální matice Λ' má na diagonále čísla λ_1, λ_2 a λ_3 a V' jsou první 3 sloupce V.

Když je v datech šum, hodnost $\tilde{J}$ je vyšší než tři. Nejlepší možná aproximace ideální centrované JIM maticí hodnosti 3 ve smyslu nejmenšího součtu čtverců vzdálenosti (viz větu A.1, část 2) je taktéž součin

$$\hat{J} = U'\Lambda' V'^T.$$

Matici rotací kamer P a matici souřadnic bodů v prostoru (tj. model scény) X lze získat např. jako:

$$\begin{aligned}P &= U'\Lambda^{1/2} \\ X &= \Lambda^{1/2} V'^T.\end{aligned}$$

Matice P a X mají velikosti $2m \times 3$ a $3 \times n$ a platí

$$\tilde{J} = PX, \tag{5.2}$$

resp. $\hat{J} = PX$, když je v datech šum.[1]

[1] Ovšem rozklad (5.2) na matice kamer a bodů není jednoznačný, protože pro libovolnou invertibilní matici 3×3 Z jsou matice PZ a $Z^{-1}X$ také rozkladem $\tilde{J}$, neboť

$$(PZ)(Z^{-1}X) = P(ZZ^{-1})X = PX = \tilde{J}.$$

Je tedy vidět, že scénu lze rekonstruovat jednoznačně až na nějakou lineární transformaci Z. Tomasi a Kanade ve svém algoritmu hledají takovou transformaci Z, aby byly výsledné matice kamer $P' = PZ$ čisté rotace, tj. aby řádky P' byly jednotkové ortogonální vektory, a navíc aby referenční osy prvního snímku byly rovnoběžné se světovou soustavou souřadnic.

Podle 5.1 můžeme faktorizaci podle Tomasi a Kanade přepsat na:[2]

$$J = (P|\mathbf{t}) \begin{pmatrix} X \\ \mathbf{e}_n^T \end{pmatrix}, \tag{5.3}$$

kde

$$P = \begin{pmatrix} P^1 \\ \vdots \\ P^m \end{pmatrix},$$

přičemž P^i je matice 2×3 rotace i-té kamery, její řádky representují orientaci horizontální a vertikální referenční osy kamery,

$$\mathbf{t} = \begin{pmatrix} \mathbf{t}^1 \\ \vdots \\ \mathbf{t}^m \end{pmatrix},$$

kde $\mathbf{t}^i \in \mathcal{R}^{2\times 1}$ je translace i-té kamery podél roviny obrázku,

$$X = (\mathbf{x}_1 \dots \mathbf{x}_n)$$

je matice struktury, její sloupce jsou tří-dimenzionální souřadnice bodů scény vzhledem k jejich centroidu a $\mathbf{e}_n^T = (1, \dots, 1)$ je vektor n jedniček.

[2]když je přítomen šum, rovnost je pouze přibližná, což můžeme vyznačit použitím $\approx$ místo $=$

Kapitola 6

Doplňování chybějících políček

Při reálném snímání obrázků, ať už kamerou nebo fotoaparátem, je velmi pravděpodobné, že některé význačné body budou na některých obrázcích viditelné a na jiných nikoliv kvůli zákrytu scény sama sebou. Proto, aby byly algoritmy počítačového vidění použitelné v praxi, musejí se s problémem nevyplněných míst v JIM nějak vyrovnat.[1]

Obsah této kapitoly se týká pouze scény s ortografickou kamerou.

6.1 Doplňování JIM podle Tomasi a Kanade

Představme si scénu, jejíž některý bod není viditelný v některém snímku. Pokud je však dostatečně viditelný v ostatních snímcích, mělo by být možné získat jeho prostorové souřadnice. Pokud snímek s chybějícím obrazem bodu obsahuje dost obrazů jiných bodů, měli bychom umět spočítat pozici kamery, kterou byl snímek nasnímán. Potom ze souřadnic bodu a kamery bychom měli být schopni rekonstruovat chybějící obraz.

Tomasi a Kanade definují v [11] následující postačující podmínku pro to, aby mohl být rekostruován obraz bodu.

Podmínka pro rekonstrukci scény s ortografickou kamerou

Při absenci šumu lze rekonstruovat význačný bod $\mathbf{q}_j^i \in \mathcal{R}^{2\times1}$ (což je obraz bodu j scény v i-tém snímku), pokud je bod j viditelný alespoň v dalších

[1] Jednoduchým trikem, jak se zbavit prázdných míst v JIM, je vyškrtnout body, které chybí alespoň v jednom obrázku. Může se ale stát, že po aplikaci tohoto triku z JIM nic nezůstane. Toto nebezpečí klade tedy vysoké nároky na snímání scény. Dalším řešením je rozdělit JIM na několik kompletních podmatic, rekonstruovat je zvlášť a řešení spojit.

Tato řešení mají ale nevýhodu, že ignorují informace o scéně, které jsou v JIM obsaženy, což se projeví na kvalitě rekonstrukce, nemluvě o množství rekonstruovaných bodů.

třech snímcích i_1, i_2, i_3 a pokud existují nejméně tři další body j_1, j_2, j_3, které jsou viditelné ve všech čtyřech snímcích i_1, i_2, i_3, i.

Souřadnice zmíněných čtyř bodů ve čtyřech snímcích lze zapsat do matice velikosti 8×4, přičemž souřadnice neznámého bodu jsou např. na místě otazníku:

$$J = \begin{pmatrix} \mathbf{q}_1^1 & \mathbf{q}_2^1 & \mathbf{q}_3^1 & \mathbf{q}_4^1 \\ \mathbf{q}_1^2 & \mathbf{q}_2^2 & \mathbf{q}_3^2 & \mathbf{q}_4^2 \\ \mathbf{q}_1^3 & \mathbf{q}_2^3 & \mathbf{q}_3^3 & \mathbf{q}_4^3 \\ \mathbf{q}_1^4 & \mathbf{q}_2^4 & \mathbf{q}_3^4 & ? \end{pmatrix}, \mathbf{q}_j^i \in \mathcal{R}^{2\times 1}$$

Aplikací faktorizace (viz rovnice 5.3) na první tři řádky J, tj. na podmatici 6×4 dostaneme částečné podmatice rotací kamer a translací

$$P_{6\times 3} = \begin{pmatrix} P^1 \\ P^2 \\ P^3 \end{pmatrix}, \mathbf{t}_{6\times 1} = \begin{pmatrix} \mathbf{t}^1 \\ \mathbf{t}^2 \\ \mathbf{t}^3 \end{pmatrix}$$

a matici celé struktury $X = (\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4)$, takže

$$J_{6\times 4} = (P_{6\times 3}|\mathbf{t}_{6\times 1}) \begin{pmatrix} X \\ \mathbf{e}_4^T \end{pmatrix},$$

kde $\mathbf{e}_4^T = (1, 1, 1, 1)$.

Faktorizační metoda počítá prostorové souřadnice vzhledem k centroidu bodů, takže $\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 + \mathbf{x}_4 = 0$, ale souřadnice ve čtvrtém snímku jsou měřeny vzhledem k prvním třem bodům. Proto dříve než budeme moci spočítat rotaci čtvrté kamery, musíme ztotožnit oba počátky tak, že všechny souřadnice budeme brát vzhledem k centroidu

$$\mathbf{c} = \frac{1}{3}(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3)$$

tří bodů, které jsou viditelné ve všech čtyřech snímcích. Projekce $\mathbf{c}$ ve čtvtém snímku má souřadnice

$$\mathbf{p}'^4 = \frac{1}{3}(\mathbf{q}_1^4 + \mathbf{q}_2^4 + \mathbf{q}_3^4),$$

takže můžeme definovat nové souřadnice

$$\mathbf{x}_i' = \mathbf{x}_i - \mathbf{c}$$

v prostoru a

$$\mathbf{q}_i'^4 = \mathbf{q}_i^4 - \mathbf{p}'^4$$

ve čtvrtém snímku pro i=1,2,3. Potom rotaci čtvrté kamery P^4 nalezneme jako řešení 6-ti lineárních rovnic o 6-ti neznámých

$$(\mathbf{q}_1'^4 \mathbf{q}_2'^4 \mathbf{q}_3'^4) = P^4(\mathbf{x}'_1 \mathbf{x}'_2 \mathbf{x}'_3), \tag{6.1}$$

což je projekce prvních tří bodů čtvrtou kamerou, neboli propagace scény do čtvrté kamery (rovnici (6.1) jsme dostali redukcí vztahu (5.2)).

Translaci čtvrté kamery získáme vynásobením rovnice 5.3 zprava vektorem $\eta_4 = (1,1,1,0)^T$:

$$J\eta_4 = PX\eta_4 + \mathbf{t}\mathbf{e}_4^T\eta_4.$$

Protože $\mathbf{e}_4^T\eta_4 = 3$, dostáváme

$$\mathbf{t} = \frac{1}{3}(J - PX)\eta_4.$$

Translaci čtvrté kamery získáme z rádků 7 a 8 této rovnice, kde nula ve čtvrté souřadnici η_4 způsobila, že nepotřebujeme znát chybějící místo v J. Nyní známe tvar scény i pohyb kamer, chybějící bod v JIM získáme projekcí:

$$\mathbf{q}_4^4 = P^4\mathbf{x}_4 + \mathbf{t}^4.$$

Postup, který jsme provedli, nyní shrneme. Faktorizací prvních třech řádků J jsme získali úplný tvar scény a částečný pohyb, který jsme rozšířili (propagovali) do posledního snímku. Pak jsme doplnili chybějící místo v J.

Alternativní postup je aplikovat faktorizaci na první tři sloupce J, spočítat všechny translace a rotace a z nich získat tvar scény a nakonec chybějící políčko v J.

Řešení za přítomnosti šumu

Metodu propagace řešení představenou v předchozí části je možné použít i na $2m \times n$ JIM s $m \geq 4$ a $n \geq 4$. Rovnice (6.1) však bude přeurčená. Pokud bude v JIM šum, tato nadbytečnost bude přínosem, protože rovnici lze řešit ve smyslu nejmenší chyby čtverců a tak redukovat vliv šumu.

Metodu propagace obecné $2m \times n$ JIM je možné shrnout následovně. Na začátku se vybere co největší podblok JIM bez chybějících dat, který se rozloží faktorizací. Nalezené řešení se poté rozšiřuje na chybějící místa v JIM. Nicméně kvůli šumu bude výsledný tvar scény i pohyb ovlivněn pořadím doplňování. Proto může být nutné výsledné řešení vylepšit minimalizací nejpříkrejšího sestupu rezidua

$$\|J - (P|\mathbf{t})\begin{pmatrix} X \\ \mathbf{e}_n^T \end{pmatrix}\|,$$

viz rovnici (5.3).

Zůstávají však dva problémy. Prvním z nich je výběr počátečního plného podbloku, na kterém se provede faktorizace. Lze ukázat, že problém nalezení největšího úplného podbloku matice s chybějícími políčky je NP-těžký (lze na něj snadno redukovat problém maximální kliky, viz [7]). Druhým problémem je postup propagace řešení. Nicméně díky vylepšení v posledním kroku algoritmu to není až tak podstatné v případě, že se na začátku vybral dostatečně velký podblok pro dobré počáteční řešení.

6.2 Doplňování JIM podle Jacobse

Jacobsův algoritmus rekonstuuje scénu s šumem a s velkým množstvím chybějících políček. Funguje pro ortografickou kameru.

Zatímco předchozí algoritmus nalezl nejdříve řešení pro část JIM a dále postupně doplňoval chybějící políčka, v tomto algoritmu se doplnění políček a rekonstrukce celé scény nechá až na konec a nejdříve se hledá báze dimenze 3, kterou by měla mít rekonstruovaná JIM. Pomocí této báze je už snadné doplnit JIM a faktorizací rekonstruovat scénu.

Jacobsův algoritmus má dva význačné rysy:

1. Jacobs ve svém algoritmu využívá všechny informace o scéně najednou. Nejedná se o iterativní ani filtrovací algoritmus.[2]

2. Používají se pouze operace lineární algebry (Jacobs mluví o „linear fitting“). Proto také může být výsledný algoritmus lineární.

Na jednoduchém příkladu ukážeme, jak je možné doplnit prázdná místa v matici nízké hodnosti. Nechť má naše matice hodnost 1 a políčka v ní chybí na místech s otazníky:

$$\begin{pmatrix} 2 & 6 \\ ? & 3 \\ ? & 3 \end{pmatrix}$$

Protože má matice hodnost 1, sloupcová báze obsahuje jediný vektor, za který můžeme položit druhý sloupec. Aby byla hodnost matice stejná jako velikost její báze, musí být všechny sloupce nějakou lineární kombinací vektorů sloupcové báze. V našem příkladě tedy musí být první sloupec nějakým

[2] V *iterativním algoritmu* se nalezne řešení celého problému nějakým počátečním výpočtem, dále se toto řešení opakovaně vylepšuje a konverguje se k řešení, které lokálně minimalizuje součet čtverců vzdálenosti mezi známými daty a řešením.

Ve *filtrovacím algoritmu* se provede počáteční výpočet pro prvních několik snímků, v následujících krocích se řešení vylepšuje přidáváním dalších snímků.

násobkem druhého. Z prvního řádku vypočteme tento násobek:

$$\begin{aligned} 2 &= 6q \\ q &= \frac{1}{3} \end{aligned}$$

Jediným správným doplněním naší matice je tedy

$$\begin{pmatrix} 2 & 6 \\ 1 & 3 \\ 1 & 3 \end{pmatrix}$$

V dalším textu budeme předpokládat, že JIM, se kterou budeme pracovat, bude centrovaná JIM, o které již víme, že má za nepřítomnosti šumu hodnost tři.

V JIM bez šumu chybí políčka

Nyní si probereme jednodušší situaci, když v JIM sice není šum, ale některé její části chybí. Protože není přítomen žádný šum, můžeme předpokládat, že JIM je skutečně obrazem nějaké reálné scény a kdybychom znali celou matici, byla by její hodnost rovna třem. Naším cílem je tedy doplnit chybějící místa JIM tak, že zachováme vlastost býti centrovanou JIM, což znamená zachovat hodnost tři.

Mějme $J \in \mathcal{R}^{m \times n}$, jejímž doplněním chceme získat novou matici Q stejných rozměrů hodnosti tři[3] . Podíváme se, co geometricky znamenají sloupce JIM a jak se projeví, když nějaká souřadnice chybí:

- Kompletnímu sloupci J odpovídá vektor m-dimenzionálního prostoru (dále jen m-D prostor).

- Pokud nemáme nějakou souřadnici, sloupci odpovídá afinní podprostor m-D prostoru. Tento podprostor je rovnoběžný s osou m-D prostoru, ve které chybí souřadnice, viz obrázek 6.1(a) a 6.1(b)[4].

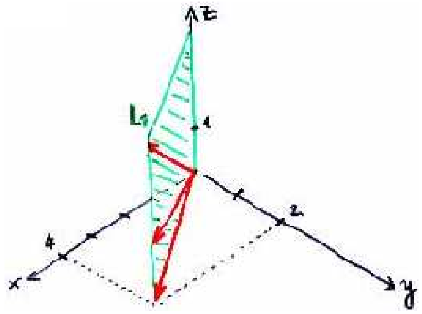
Hledáme bázi J, pomocí které bychom podobným způsobem jako výše spočítali prázdná místa J. Tato báze musí mít velikost tři, takže jejím vektorům odpovídá 3-D lineární prostor, označme ho L. Je to prostor, který je

[3] Jacobsův algoritmus ale funguje na doplňování libovolné matice na libovolnou hodnost. (Je samozřejmě vhodné požadovat od algoritmu doplnění na nějakou rozumnou hodnost.)

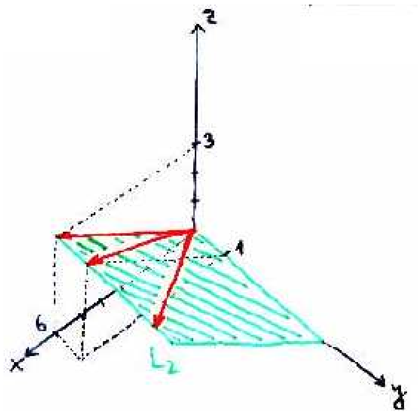
[4] Naším cílem je pracovat s hodností 3, ale protože na 2-D obrázcích se těžko ukazuje, co chybějící políčka geometricky znamenají, příklady budou pouze pro hodnosti 1 a 2. LO v příkladech označuje lineární obal.

$$J = \begin{pmatrix} 4 & 6 \\ 2 & ? \\ ? & 3 \end{pmatrix}, h(J) = 1$$

$$\mathbf{L_1} = LO\begin{pmatrix} 4 & 4 \\ 2 & 2 \\ \mathit{0} & \mathit{1} \end{pmatrix} \qquad \mathbf{L_2} = LO\begin{pmatrix} 6 & 6 \\ \mathit{0} & \mathit{1} \\ 3 & 3 \end{pmatrix}$$

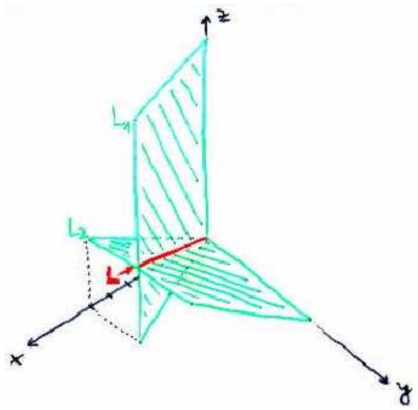


(a) Prostor sloupce 1

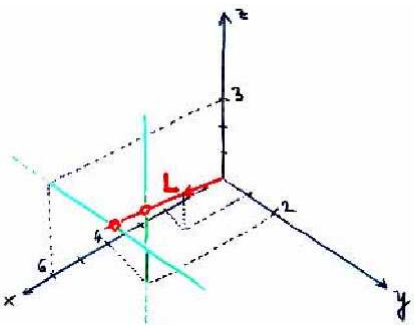


(b) Prostor sloupce 2

$$\mathbf{L} \subseteq \mathbf{L_1}, \mathbf{L_2} \Longrightarrow \mathbf{L} \subseteq \mathbf{L_1} \cap \mathbf{L_2} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \Longrightarrow Q = \begin{pmatrix} 4 & 6 \\ 2 & \mathit{3} \\ \mathit{2} & 3 \end{pmatrix}$$



(c) Prostor L



(d) Doplnění JIM

Obrázek 6.1: Prostory sloupců JIM, prostor L a doplnění JIM nejblíže k afinním podprostorům sloupců J v tom smyslu, že součet čtverců vzdálenosti k nim je minimalní. L je tedy v průniku afinních podprostorů sloupců, viz obr. 6.1(c).

Sloupec Q je, stejně jako sloupec J, nějakou lineární kombinací vektorů báze J, což je také báze L. Proto je sloupec Q vektor v L, který je nejblíže k afinnímu podprostoru odpovídajícího sloupce J. Jeho nalezením získáme chybějící políčka sloupce Q, viz obr. 6.1(d).

6.2.1 Algoritmus pro nalezení prostoru L

Není snadné nalézt prostor L neboli bázi J, protože J není kompletní. Kdybychom měli tři kompletní lineárně nezávislé (dále jen LN) sloupce, měli bychom bázi J. Nicméně, i když je sloupec J nekompletní, nějakou užitečnou informaci obsahuje. Viděli jsme, že nekompletnímu sloupci odpovídá podprostor. Pokud vezmeme tři sloupce jako generátory, bude jim odpovídat podprostor dimenze minimálně tři.

Budeme hledat trojice sloupců, které kladou na L omezující podmínku „L leží v lineárním obalu (dále jen LO) těchto sloupců“. L bude obsažen v LO každé takové trojice a proto i v jejich průniku. Když se průnik zmenší na dimenzi 3, budeme mít L přesně.

Zbývá upřesnit, jak poznáme správnou trojici sloupců. Prostor L je obsažen v LO sloupců, když každý sloupec trojice přidá k LO (jeden) vektor společný s L, který je různý od počátku. Pokud souřadnice v nějakém sloupci trojice chybí, můžeme do ní dosadit cokoliv, takže se tak generuje celá dimenze odpovídající chybějící souřadnici. Podmínka na obsah L v LO v této dimenzi je tedy vždy splněna a tuto dimenzi nemá smysl uvažovat. Zavedeme si přesný pojem pro trojici, která nás bude zajímat.

Definice 1 (Generující trojice) *Trojici sloupců nazveme generující trojicí, pokud po odstranění řádků s chybějícími políčky tvoří zbývající části sloupců bázi prostoru dimenze 3.*

Poznámky (Generující trojice):

1. Generující trojice má tu vlastnost, že prostor L jistě leží v jejím LO.

2. Podobným způsobem můžeme definovat generující k-tice sloupců s tím rozdílem, že zbývající části sloupců budou tvořit bázi prostoru dimenze k.

3. Tato definice má samozřejmě smysl jen pro sloupce k-tic velikosti alespoň $k + 1$.

Pro srozumitelnost ukážeme význam definice na příkladu. Zobrazit prostory slopců JIM hodnosti 3 by bylo obtížné, omezíme se tedy na JIM hodnosti 2. Matice J na obr. 6.2 by měla mít hodnost 2 a hovořili bychom tedy o generujících dvojicích a o dimenzi prostoru 2. Nemohli bychom použít za generující dvojici sloupce 1 a 3, protože po odstranění řádků s chybějícími políčky zbyde

$$\begin{pmatrix} 3 & 3 \\ 2 & 2 \end{pmatrix},$$

což je báze prostoru dimenze pouze 1, ale použili jsme sloupce 1 a 2, které tvoří bázi prostoru dimenze 2:

$$\begin{pmatrix} 3 & 1 \\ 2 & 2 \\ 1 & -1 \end{pmatrix}.$$

Síla podmínky závisí na počtu chybějících políček. Pokud žádné nechybí, známe L přesně. Platí, že $L \subseteq \bigcap_{i \in T} S_i$, kde T je množina trojic indexů sloupců J, S_i je lin. prostor tvořený i-tou trojicí sloupců.

Kvůli šumu se však průnik $\bigcap_{i \in T} S_i$ brzy vyprázdní. Tuto situaci vyřešíme přechodem ke komplementu: průnik lineárních podprostorů můžeme nalézt jako komplement ke sjednocení jejich komplemetů. V následující části ukážeme, že takový způsob nalezení průniku je korektní. Tento způsob má tu výhodu, že nám umožní za přítomnosti šumu nalézt takovou aproximaci $\bigcap_{i \in T} S_i$, u které si budeme moci ohlídat dimenzi (budeme chtít dimenzi tři), i když samotný průnik je prázdný.

6.2.2 Výlet do teorie svazů

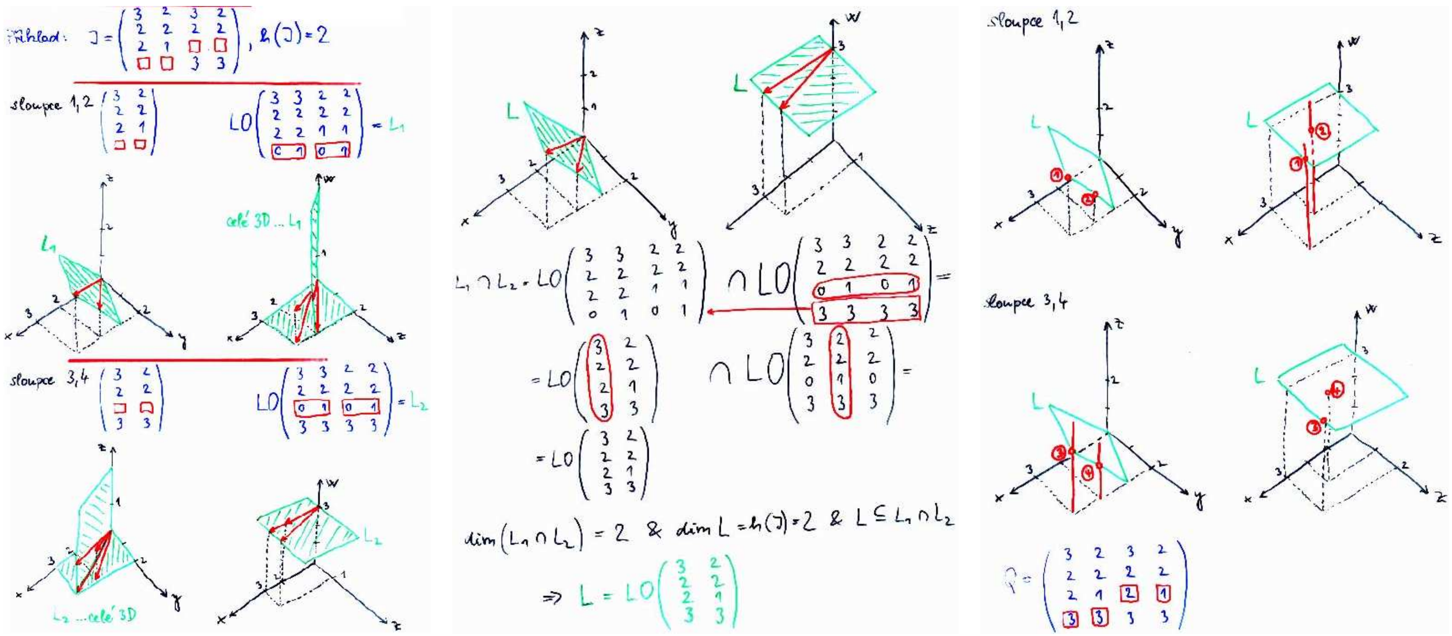
Na systém podprostorů lineárního prostoru se můžeme dívat jako na modulární komplementární svaz. V [9] lze nalézt následující definici.

Definice 2 (Modulární orthokomplementární svaz) *Nechť (M, $\wedge$, $\vee$, $\leq$, $'$, 0, 1) je trojice taková, že částečně uspořádaná množina (M, $\wedge$, $\vee$, $\leq$) je svaz (tj. každá dvojice prvků $a, b \in M$ má v M infimum a supremum[5]) s 0 a 1 a navíc $' : M \to M$ je unární operace s vlastnostmi:*

$$\begin{aligned} a \leq b \quad &\Rightarrow \quad b' \leq a', \\ a \vee a' \quad &= \quad 1, \\ a \wedge a' \quad &= \quad 0. \end{aligned} \tag{6.2}$$

[5]**Definice (infimum):** $c = a \wedge b$ je *infimum* $\overset{\text{def}}{\equiv}$ $c \leq a, c \leq b$ a navíc platí: Pokud $d \leq a, d \leq b$, pak $d \leq c$.

Infimum je největší dolní závora. Supremum je definováno duálně.



(a) Prostory sloupců

(b) Prostor L

(c) Doplnění JIM

Obrázek 6.2: Prostory dvojic sloupců JIM, prostor L a doplnění JIM

Pak M se nazývá modulární orthokomplementární svaz.

Ukážeme, že přechod ke komplementu je v tomto svazu definován korektně.

Věta 1 (Přechod ke komplementu) *Pro modulární orthokomplementární svaz platí*

$$(a \wedge b)' = a' \vee b'. \tag{6.3}$$

Důkaz: Zřejmě $a \wedge b \leq a$, použitím (6.2) dostaneme $a' \leq (a \wedge b)'$, podobně $b' \leq (a \wedge b)'$, takže dohromady máme důkaz první nerovnosti ve větě:

$$a' \vee b' \quad \leq \quad (a \wedge b)'.$$

Zřejmě $a' \leq a' \vee b'$, použitím (6.2) máme $(a' \vee b')' \leq a$, podobně $(a' \vee b')' \leq b$, dohromady dostáváme

$$\begin{aligned} (a' \vee b')' &\leq a \wedge b \\ (a \wedge b)' &\leq ((a' \vee b')')' \\ (a \wedge b)' &\leq a' \vee b' \end{aligned}$$

jako druhou nerovnost ve větě. □

Příkladem modulárního orthokomplementárního svazu je $(M(\mathcal{R}^n), \cap, LO, \subseteq, \perp, \emptyset, 1)$ – svaz lineárních podprostorů v $\mathcal{R}^n$, který je uspořádán inkluzí, jako infimum je použit průnik, jako supremum lineární obal a komplementem je orthogonální doplněk.

6.2.3 Algoritmus pro nalezení prostoru L — pokračování

Tímto skončil náš výlet do teorie svazů. Větu o komplementu, kterou jsme dokázali, nyní použijeme při hledání $L \subseteq \bigcap_{i \in T} S_i$. Podle (6.2):

$$L \subseteq \bigcap_{i \in T} S_i \Longrightarrow (\bigcap_{i \in T} S_i)^{\perp} \subseteq L^{\perp}$$

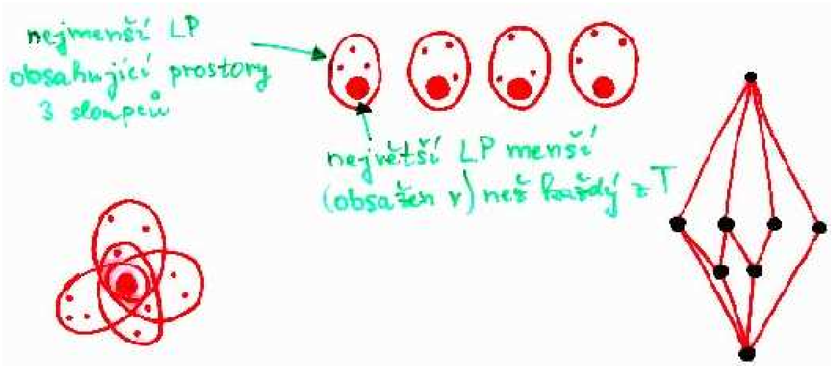
Podle(6.3):

$$(\bigcap_{i \in T} S_i)^{\perp} = LO_{i \in T}(S_i^{\perp})$$

Opět použitím(6.2) dostáváme:

$$L \subseteq (LO_{i \in T}(S_i^{\perp}))^{\perp} = \bigcap_{i \in T} S_i \tag{6.4}$$

Jak mohou vypadat průniky lineárních prostorů a jejich svaz je ukázáno na obr. 6.3.



Obrázek 6.3: Průniky lineárních prostorů a jejich svaz

Počítání L

Lineární podprostory budeme reprezentovat maticemi, jejichž sloupce odpovídají vektorům, které tyto prostory generují. Tyto generátory tedy musí obsahovat nějakou bázi prostoru, jejich velikost však může být větší než je velikost báze. $\bigcap_{i\in T} S_i$ budeme počítat podle (6.4), přičemž budeme chtít, aby měl výsledný průnik dimenzi tři. Nalezneme nejdříve reprezentaci průniku dvou prostorů, kterou zobecníme.

Nechť je prostor S_i dimenze d reprezentován maticí M_i. Reprezentaci jeho doplňku, prostoru S_i' $(= S_i^{\perp})$, nalezneme jako $N_i = u(:, d+1 : end)$, kde $[u, d, v] = svd(M_i)$. Sjednocení dvou prostorů $S_i \vee S_j$ $(= LO(S_i, S_j))$ je reprezentováno jako $[M_i M_j]$. Průnik dimenze tři dvou prostorů $S_i \wedge S_j$ $(= S_i \cap S_j)$ nalezneme jako $u(:, 4 : end)$, kde $[u, d, v] = svd([N_i N_j])$[6]. Reprezenaci N 3D-průniku z prostorů $\bigcap_{i\in T} S_i$, kde T je množina indexů, nalezneme jako $u(:, 4 : end)$, kde $[u, d, v] = svd([N_1 N_2 \ldots N_z])$.

$\bigcap_{i\in T} S_i$ má dimenzi 3, proto je $L = \bigcap_{i\in T} S_i$, neboť je v něm obsaženo.

Konstrukce Q

Označíme-li maticovou reprezentaci prostoru L jako l, J_i i-tý sloupec J, p indexy nechybějících políček v J_i, J_i^p pouze nechybějící políčka v J_i, l^p podmatici l z řádků s indexy p, potom můžeme odhadnout i-tý sloupec Q jako

$$Q_i \approx l((l^p)^+ J_i^p),$$

kde $(l^p)^+$ je pseudoinverze k l^p.

[6]Použití věty (6.4) je zřejmější po rozepsání provedeného postupu: $S_i \vee S_j = (S_i \wedge S_j)'' = (S_i' \vee S_j')'$

6.2.4 Přínos algoritmu

1. Algoritmus řeší některé konfigurace chybějících bodů v JIM, které jsou pro filtrační nebo iterativní metody neřešitelné.

   V příkladu označuje 1 známou souřadnici a 0 souřadnici neznámou:

Snímek	**Souřadnice**	**Bod 1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	x	1	1	1	1	1	1	0	0	0
1	y	1	1	1	1	1	1	0	0	0
2	x	1	1	1	0	0	0	1	1	1
2	y	1	1	1	0	0	0	1	1	1
3	x	0	0	0	1	1	1	1	1	1
3	y	0	0	0	1	1	1	1	1	1

   Důvod neřešitelnosti pro uvedené metody je následující. Z 6-ti společných bodů ve dvou snímcích sice můžeme odhadnout afinní strukturu těchto bodů, ale jenom 3 z těchto 6-ti bodů se objeví v dalším snímku, takže projekce této afinní struktury do něj není možná.

2. Druhým přínosem algoritmu je to, že využije více informace, díky čemuž dává přesnější výsledky (to lze samozřejmě tvrdit jen tehdy, když uvedené metody naleznou řešení).

3. Dalším důležitým přínosem algoritmu je fakt, že lze použít na doplňování libovolné matice na určitou hodnost, viz 6.2, poznámka 3.

4. Algoritmus může pracovat v lineárním čase.

   Algoritmus probíhá v lineárních krocích, což jsou operace s trojicemi sloupců velikosti $3m$, každý krok tedy probíhá v čase $O(m)$. Maximální počet kroků je stejně jako trojic, $\binom{n}{3}$, takže maximální čas algoritmu je $O(m\binom{n}{3})$. To je ale výpočetně velmi náročné pro JIM s velkým n. Tento problém se řeší tak, že se vybírá pouze kn náhodných trojic, kde k je nějaká konstanta. Tento přístup dává dostatečné přiblížení variantě se všemi trojicemi, viz 9.3.1, experiment 1. Čas algoritmu je potom lineární s velikostí vstupních dat: $O(mn)$.

6.2.5 Srovnání s algoritmem Tomasi a Kanade

Algoritmy si jsou podobné v tom, že oba hledají bázi JIM. Různí se však ve způsobu hledání.

V algoritmu Tomasi a Kanade se po nalezení částečného řešení doplňují chybějící políčka v dalším sloupci (nebo řádku) tak, že se nalezne lineární kombinace báze pro sloupce (nebo řádky) podmatice, která nejlépe odpovídá nechybějícím políčkům nového sloupce (řádky). To, jak se dobře se podaří nalézt bázi JIM, velmi závisí na výběru počátečního podbloku JIM, který se faktorizuje. Pokud byly v tomto bloku velké chyby, metodou propagace řešení se rozšíří i do ostatních částí JIM a není síly, která by řešení této chyby zbavila, i když se v posledním kroku algoritmu chyba minimalizuje příkrým sestupem[7].

Oproti tomu Jacobsův algoritmus hledá bázi rovnoměrně ve všech datech, takže je chyba rovnoměrně rozptýlena. Proto dává Jacobsův algoritmus lepší výsledky, viz [7].

6.2.6 Varianty algoritmu

Jacobsův algoritmus je možné modifikovat podle toho, jestli se doplňuje JIM, její transpozice, nebo zda se berou páry obrázků.

Varianta s páry obrázků spočívá v tom, že JIM transponujeme, čímž dostaneme matici $n \times 2m$. i-tý obrázek přispívá dvěma sloupci $2i-1$ a $2i$. Zkouší se pouze trojice sloupců mezi párů obrázků, tj. pro obrázky i a j máme 4 trojice sloupců $(2i-1, 2i, 2j-1)$, $(2i-1, 2i, 2j)$, $(2i-1, 2j-1, 2j)$ a $(2i, 2j-1, 2j)$.

Tento přístup má tu výhodu, že párů obrázků je pouze $\binom{m}{2}$, takže počet trojic ve variantě s páry je $4\binom{m}{2}$, což je málo v porovnání s počtem všech trojic v transpozici JIM: $\binom{2m}{3}$. Vyžívá se přitom skutečnosti, že pokud chybí x-ová souřadnice význačného bodu, chybí i jeho y-ová souřadnice. Je proto vhodné brát sloupce ze společného obrázku.

6.2.7 Urychlení Jacobsova algoritmu

Volání $svd([N_1 N_2 \dots N_z])$ při hledání prostoru L v 6.2.3 je pomalé a počítají se kromě báze i její koeficienty, které však při hledání báze nulového prostoru nepotřebujeme.

Matice $N = [N_1 N_2 \dots N_z]$ má rozměry $2m \times K$, a typicky je $2m \ll K$, tedy N je daleko širší než vyšší. Místo svd lze použít řádově rychlejší (pro velké K) výpočet pomocí vlastních vektorů.

[7] Metoda příkrého sestupu najde pouze lokálně, nikoliv globálně optimální řešení.

Hledáme U, $[U, S, V] = svd(N)$ a tedy

$$N = USV^T.$$

Vynásobme to zprava transpozicí sebe sama, V se zruší:

$$NN^T = USS^TU^T$$

Uvažujme teď vlastní čísla matice NN^T, tedy

$$[W, D] = eig(NN^T).$$

Je

$$NN^TW = WD$$

a tedy také

$$NN^T = WDW^T.$$

Protože NN^T je symetrická, vlastní čísla budou reálná a W bude orthonormální. Tedy

$$U = W.$$

Tedy $[U, S, V] = svd(N)$ je nahrazeno $[U, SS^T] = eig(NN^T)$.

Vtip je v tom, že NN^T má rozměry jen $2m \times 2m$, ten nepohodlný velký rozměr matice K tedy vůbec do eig nevstupuje. Svd trvá tak dlouho proto, že počítá velkou matici V a S, kterou vůbec nepotřebujeme. K (počet trojic sloupců vybraných z JIM) může být desítky tisíc a rozklad N bude stále stejně rychlý.

Svd se obyčejně upřednostňuje před eig proto, že je numericky stabilnější. Ale stabilita je zde více než vyvážená malostí matice NN^T oproti N. (Rozdíl ve stabilitě se v testech neprojevil.)

Kapitola 7

Scéna s perspektivní kamerou

Předpokládejme, stejně jako v situaci s ortografickou kamerou, že máme n význačných bodů na m snímcích. Souřadnice těchto bodů $\{\mathbf{q}_j^i = (u_j^i, v_j^i)^T | i = 1, \ldots, m, j = 1, \ldots, n\}$ zapíšeme do matice J velikosti $2m \times n$:

$$J = \begin{pmatrix} \mathbf{q}_1^1 & \ldots & \mathbf{q}_n^1 \\ \vdots & & \vdots \\ \mathbf{q}_1^m & \ldots & \mathbf{q}_n^m \end{pmatrix}.$$

Chceme nalézt matici kamer $P \in \mathcal{R}^{3m\times 4}$ a matici souřadnic bodů v prostoru $X \in \mathcal{R}^{4\times n}$. Promítání perspektivní kamerou je složitější než v případě ortografické kamery, protože projekce bodu $\mathbf{x}_j$ i-tou kamerou platí až na nějaký skalární násobek: $\lambda_j^i \mathbf{q}^i = P^i \mathbf{x}_j$. Skalár λ_j^i se nazývá *projektivní hloubka* nebo *měřítko* bodu X_j v i-tém obrázku. Promítnutím všech bodů a všech kamer dostaneme podmínku pro P a X:

$$\begin{pmatrix} \lambda_1^1 \mathbf{q}_1^1 & \ldots & \lambda_n^1 \mathbf{q}_n^1 \\ \vdots & & \vdots \\ \lambda_1^m \mathbf{q}_1^m & \ldots & \lambda_n^m \mathbf{q}_n^m \end{pmatrix} = \begin{pmatrix} P^1 \\ \vdots \\ P^m \end{pmatrix} (\mathbf{x}_1 \ldots \mathbf{x}_n), \tag{7.1}$$

zkráceně $\lambda J = PX$. Výraz λJ neoznačuje maticové násobení, ale matici na levé straně rovnice (7.1), které budeme říkat *JIM s měřítky*.

Rekonstrukce scény s perpsektivní kamerou se provádí tak, že se nejdříve spočítají projektivní hloubky pomocí bifokálních (viz [10, 4]), trifokálních nebo kvadrifokálních tenzorů (viz [4]). Tím získáme matici λJ hodnosti 4 (v přítomnosti šumu je tato hodnost vyšší), kterou lze faktorizovat (viz A.1).

Odhad měřítek lze obecně provést až na nějaké násobky sloupců a trojic řádků odpovídajícím jednotlivým obrázkům. Vynásobení j-tého sloupce JIM s měřítky nenulovým skalárem totiž odpovídá vynásobení $\mathbf{x}_j$ stejným číslem, viz rovnice (7.1). Podobně vynásobení i-té trojice řádků JIM s měřítky nenulovým skalárem odpovídá vynásobení $\mathbf{P}^i$ stejným číslem.

7.1 Odhad měřítek podle Triggse

Triggs ve svém algoritmu [10] odhaduje projektivní hloubky z fundamentalních matic a epipólů, které se souhrně označují jako bifokální tenzory. Dosazením rovnice $\lambda^i \mathbf{q}^i = P^i Q$, což je projekce bodu Q i-tou kamerou, do matice 6×5

$$\left(\begin{array}{c|c} P^i & \lambda^i \mathbf{q}^i \\ P^j & \lambda^j \mathbf{q}^j \end{array} \right) = \left(\begin{array}{c|c} P^i & P^i Q \\ P^j & P^j Q \end{array} \right) = \left(\begin{array}{c} P^i \\ P^j \end{array} \right) (I_{4\times 4}|Q) \tag{7.2}$$

dostáváme, že tato matice má hodnost maximálně 4. Proto mají všechny její podmatice 5×5 nulovou hodnost a jejich determinanty jsou rovny nule. Označme

$$\mathbf{t}^k = \lambda^k \mathbf{q}^k, \tag{7.3}$$

dále nechť P^k_a, t^k_a jsou a-té řádky matice P^k a vektoru t^k, kde $k = i, j$. Vezměme tyto tři (ze šesti) determinantů podmatic matice (7.2):

$$\begin{vmatrix} P^i_{23} & \mathbf{t}^i_{23} \\ P^j & \mathbf{t}^j \end{vmatrix} = \begin{vmatrix} P^i_{13} & \mathbf{t}^i_{13} \\ P^j & \mathbf{t}^j \end{vmatrix} = \begin{vmatrix} P^i_{12} & \mathbf{t}^i_{12} \\ P^j & \mathbf{t}^j \end{vmatrix} = 0. \tag{7.4}$$

Rozvojem prvního determinantu podle posledního sloupce dostáváme:

$$\begin{vmatrix} P^i_3 \\ P^j \end{vmatrix} \mathbf{t}^i_2 - \begin{vmatrix} P^i_2 \\ P^j \end{vmatrix} \mathbf{t}^i_3 + \begin{vmatrix} P^i_{23} \\ P^j_{23} \end{vmatrix} \mathbf{t}^j_1 - \begin{vmatrix} P^i_{23} \\ P^j_{13} \end{vmatrix} \mathbf{t}^j_2 + \begin{vmatrix} P^i_{23} \\ P^j_{12} \end{vmatrix} \mathbf{t}^j_3 = 0. \tag{7.5}$$

Označíme-li

$$[F]_{aa'} = \begin{vmatrix} P^i_{bc} \\ P^j_{b'c'} \end{vmatrix}, \qquad [\mathbf{e}]^a = \begin{vmatrix} P^i_a \\ P^j \end{vmatrix},$$

kde čísla abc a $a'b'c'$ jsou sudé permutace 123, můžeme (7.5) přepsat a upravit na

$$\begin{aligned} [\mathbf{e}]^3 \mathbf{t}^i_2 - [\mathbf{e}]^2 \mathbf{t}^i_3 + [F]_{11} \mathbf{t}^j_1 + [F]_{12} \mathbf{t}^j_2 + [F]_{13} \mathbf{t}^j_3 &= 0 \\ ([F]_{11} [F]_{12} [F]_{13}) \mathbf{t}^j &= -([\mathbf{e}]^3 - [\mathbf{e}]^2) \begin{pmatrix} \mathbf{t}^i_2 \\ \mathbf{t}^i_3 \end{pmatrix}. \end{aligned}$$

Po rozvoji a stejné úpravě dalších dvou determinantů (7.4) dostaneme

$$\begin{aligned} ([F]_{21} [F]_{22} [F]_{23}) \mathbf{t}^j &= ([\mathbf{e}]^3 - [\mathbf{e}]^1) \begin{pmatrix} \mathbf{t}^i_1 \\ \mathbf{t}^i_3 \end{pmatrix} \\ ([F]_{31} [F]_{32} [F]_{33}) \mathbf{t}^j &= -([\mathbf{e}]^2 - [\mathbf{e}]^1) \begin{pmatrix} \mathbf{t}^i_1 \\ \mathbf{t}^i_2 \end{pmatrix}. \end{aligned}$$

Poslední tři rovnice zapíšeme do jedné a použitím (7.3) dostáváme:

$$\begin{aligned}
\begin{pmatrix} [F]_{11} & [F]_{12} & [F]_{13} \\ [F]_{21} & [F]_{22} & [F]_{23} \\ [F]_{31} & [F]_{32} & [F]_{33} \end{pmatrix} \mathbf{t}^j &= \begin{pmatrix} 0 & -[\mathbf{e}]^3 & [\mathbf{e}]^2 \\ [\mathbf{e}]^3 & 0 & -[\mathbf{e}]^1 \\ -[\mathbf{e}]^2 & [\mathbf{e}]^1 & 0 \end{pmatrix} \mathbf{t}^i \\
(F\mathbf{t}^j) &= [\mathbf{e}]_\times \mathbf{t}^i \\
(F\mathbf{q}^j)\lambda^j &= (\mathbf{e} \wedge \mathbf{q}^i)\lambda^i, \qquad (7.6)
\end{aligned}$$

což je základní vztah mezi epipolárami, kde F je fundamentální matice a $\mathbf{e}$ je epipól pro i-tý a j-tý snímek.

Tato rovnice říká dvě věci:

1. **Rovnost až na násobek:** Epipolára bodu $\mathbf{q}_j$ ve snímku i je přímka protínající bod $\mathbf{q}_i$ a epipól $\mathbf{e}_{ij}$.

2. **Rovnost měřítek:** Pokud do (7.6) dosadíme správná měřítka, obě strany rovnice se budou rovnat přesně — ne jen až na násobek. Díky tomu můžeme spočítat měřítka z fundamentálních matic a epipólů.

Potřebujeme nejméně m fundamentálních matic a epipólů, které sjednotí násobky měřítek m snímků v jediný celkový násobek. Mezi m snímky bychom sice mohli nalézt až $m(m-1)/2$ fundamentálních matic a epipólů, to by bylo však příliš výpočetně náročné, neboť návíc bychom měli přeurčenou soustavu. Nejsnazším řešením je vzít páry po sobě jdoucích snímků a nebo vybrat jeden snímek, říkejme mu *centrální*, a brát fundamentální matice a epipóly tohoto snímku a ostatních.

V obou případech nalezneme sice ne nejrobustnější řešení, ale máme tu výhodu, že je odhad měřítek triviální. Řešením vektorové rovnice (7.6) metodou nejmenších čtverců pro λ^i vzhledem k λ^j dostáváme:

$$\lambda^i = \frac{(\mathbf{e} \wedge \mathbf{q}^i) \cdot (F\mathbf{q}^j)}{\|\mathbf{e} \wedge \mathbf{q}^i\|^2}. \qquad (7.7)$$

Následuje celý algoritmus podle Triggse:

1. V prvním kroku algoritmu se určí páry obrázků. Do páru se vezmou vždy po sobě jdoucí obrázky v sekvenci[1] (nebo se určí centrální obrázek c a do párů s ním se vezmou všechny zbývající).

[1] U (video-)sekvencí si jsou po sobě jdoucí obrázky velmi podobné, lze tedy očekávat, že nejvíce společných význačných bodů budou mít právě po sobě jdoucí obrázky.

2. Měřítka bodů obrázku na začátku řetězu (nebo bodů v cetrálním obrázku) se nastaví na nějakou počáteční hodnotu, např. 1: $\lambda_p^1 = 1$ (nebo $\lambda_p^c = 1$) pro $p = 1, \ldots, n$.

   Spočítájí se fundamentální matice F^{ij} a epipóly $\mathbf{e}^{ij}$ pro každý pár obrázků pomocí např. osmibodového [2] nebo sedmibodového [3] algoritmu.

3. Postupně se spočítají měřítka bodů v druhém až posledním obrázku (nebo bodů v druhém obrázku každého páru s cetrálním obrázkem) podle rovnice (7.7):

$$\lambda_p^i = \frac{(\mathbf{e}^{ij} \wedge \mathbf{q}_p^i) \cdot (F^{ij} \mathbf{q}_p^j)}{\|\mathbf{e}^{ij} \wedge \mathbf{q}_p^i\|^2}, p = 1, \ldots, n.$$

7.2 Numerická stabilita faktorizace

Volnost při výpočtu měřítek až na nějaké násobky sloupců a trojic řádků odpovídajícím obrázkům neovlivní rekonstruovaný pohyb a tvar, viz úvod kapitoly 7. To ale platí pouze při absenci šumu. V přítomnosti šumu bude mít JIM s měřítky hodnost přibližně čtyři a takové skalární přenásobení matice ovlivní výsledek faktorizace. Vyplatí se proto aplikovat vhodné přenásobení.

Cílem je zajistit numerickou stabilitu takovým přenásobením, aby všechny řádky a sloupce JIM s měřítky měly v průměru stejnou důležitost. To lze docílit následujícím iterativním postupem, $(w_{ij})_{i=1,\ldots,3m,j=1,\ldots,n} = \lambda J$:

1. Přenásob každý sloupec l tak, aby $\sum_{i=1}^{3m} w_{il}^2 = 1$.

2. Přenásob každou trojici řádků $(3k-2, 3k-1, 3k)$ tak, aby $\sum_{l=1}^{n} \sum_{i=3k-2}^{3k} w_{il}^2 = 1$.

3. Pokud se políčka JIM s měřítky změnily významně, opakuj body 1 a 2.

7.3 Algoritmus pro chybějící políčka

Pokud v JIM chybějí nějaká políčka, nelze provést faktorizaci, i kdyby bylo možné spočítat měřítka u nechybějících políček. Nabízí se tedy myšlenka provést faktorizaci pouze na řádky a sloupce JIM, u kterých se podařilo spočítat měřítka. Tím však dostaneme rekonstrukci pouze části scény a navíc jsme nevyužili některé informace o scéně.

Jacobsův algoritmus na doplňování chybějící políček JIM ortografické kamery funguje pro doplňování obecné matice dané hodnosti, viz 6.2, poznámka 3. Proto pomocí Jacobsova algoritmu můžeme doplnit chybějící políčka včetně měřítek ještě před faktorizací.

Algoritmus rekonstrukce sestává z následujících dvou kroků:

1. Spočítají se měřítka podle Triggse[2].

2. Doplní se chybějící políčka Jacobsovým algoritmem.

Tento proces lze opakovat, protože při další iteraci je možné, že se díky doplněným políčkům z předchozí iterace podaří spočítat nějaká další měřítka a díky nim doplnit další chybějící políčka. Opakování se provádí dokud algoritmus nezkonverguje, tj. dokud nastává nějaká změna, totiž že se od minulého kroku doplnil nějaký bod.

Pokud je scéna, kterou rekonstrujeme, sekvencí, bude jistě nejlepší, aby byl Triggsův algoritmus aplikován na řetěz po sobě jdoucích snímků (viz 7.1, pozn. 1). Pokud tomu tak není, bývá lepší určit v prvním kroku Triggsova algoritmu centrální obrázek. Může se na první pohled zdát, že je velká volnost při tomto výběru. Ovšem ne všechny možnosti jsou nejšťastnější, např. pokud by měl být centrální obrázek ten, který nemá žádné společné význačné body s jakýmkoliv jiným obrázkem, potom bychom totiž rekonstrukcí scény nic nezískali.

Zbývá tedy najít dobrý způsob výběru centrálního obrázku, tedy otázka, co je vhodný centrální obrázek. Pro daný obrázek lze snadno zjistit, kolik měřítek by šlo spočítat, kdyby byl tento obrázek centrální, a dále, kolik políček by šlo doplnit (popř. jak velkou kompletní podmatici bychom získali). Druhý údaj označme jako *sílu obrázku.*

Dobrý algoritmus rekonstrukce scény v počítačovém vidění využívá co nejvíce informace o scéně, pokud možno spravedlivě pro všechna data (viz [7]). Odtud je zřejmé, že je nejvýhodnější provést co nejméně iterací tohoto algoritmu, protože při opakování se zvýhodňují již dříve použitá data.[3] Proto je optimální vybrat za centrální ten obrázek, který má největší sílu, neboť potom se při každé iteraci algoritmu doplní maximální počet chybějících políček a počet iterací bude minimální.

[2] Měřítka je možné počítat i pomocí trifokálních nebo kvadrifokálních tenzorů.

[3] To je dáno charakterem algoritmu. V první iteraci se rekonstruje nějaká podmatice JIM, která se dalšími iteracemi rozrůstá. Ty části JIM, které se zrekonstruují nejdříve, přispívají svým šumem k rekonstukci později rekonstruovaných částí.

7.3.1 Síla obrázku

V matici $I \in \mathcal{R}^{m \times n}$ mějmě informaci, která políčka v JIM chybí: hodnota 0 nechť značí, že příslušný význačný bod v JIM chybí, v opačném případě nechť je hodnota 1. Např.

$$I = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

zkoumejme sílu druhého obrázku.

Vychází se z toho, že pro výpočet fundamentální matice mezi dvěma obrázky potřebuje sedmibodový algoritmus sedm (osmibodový osm) společných význačných bodů, takže přesně víme, které fundamentální matice nelze spočítat, aniž bychom je počítali[4]. Proto nastavme do příslušných řádků matice I nuly:

$$I = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

Dále víme, že měřítka lze spočítat pro všechny význačné body centrálního obrázku (krok 2 Triggsova algoritmu) a jen pro ty význačné body druhého obrázku v každém páru, které mají společníka v centrálním obrázku (krok 3). Do políček příslušejících všem ostatním význačným bodům opět nastavme nuly:

$$I = \begin{pmatrix} \mathbf{0} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

O Jacobsově algoritmu víme, že nutnou podmínkou k tomu, aby se mohl určitý sloupec matice účastnit nějaké čtveřice, která klade podmínku na prostor L (viz 6.2.1), musí být ve sloupci alespoň 4 nenulová políčka, aby mohla tato políčka tvořit spolu s nenulovými políčky ostatních sloupců čtveřice matici dimenze alespoň 4. Protože políčku I odpovídá v JIM políčko 3×1, víme, že sloupec JIM odpovídající sloupci I, ve kterém je pouze jedna jednička, se nebude takové čtveřice účastnit. Proto nastavme takové sloupce na nuly:

$$I = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \mathbf{0} \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

[4] až na případ degenerované konfigurace význačných bodů

Po odstranění všech nulových sloupců a řádků, neboť ty se jistě nezrekonstují, dostaneme maximální matici, která by měla šanci být celá zrekonstruovaná Jacobsovým algoritmem[5]. Počet nul v této matici vyjadřuje sílu obrázku. V našem příkladě je síla druhého obrázku rovna jedné (popř. 24, což je $3 * 8$):

$$I = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Ovšem může se stát, že Jacobsův algoritmus neuspěje, takže nedoplní jediné políčko JIM[6] . Pokud jsou ve scéně ještě nějaké obrázky s nenulovou sílou, můžeme zkusit vybrat za centrální ten s největší sílou a pokus o doplnění zopakovat atd. Pokud už žádný další obrázek s nenulovou sílou není, algoritmus zkonvergoval.

[5]kdyby nedošlo k degeneraci význačných bodů při výpočtu fundamentálních matic

[6]V některých případech stojí za to spustit Jacobsův algoritmus ještě na transpozici JIM, popř. ho spustit vícekrát.

Kapitola 8

Když jsou známa některá měřítka

Představme si, že máme rekonstruovat scénu s perspektivní kamerou, ale někdo nám navíc předloží měřítka některých význačných bodů.

Otázka zní, jestli je možné při rekonstrukci nějakým způsobem využít dodaná měřítka, resp. zda by znalost těchto měřítek umožnila celou rekonstrukci, aniž bychom potřebovali počítat další měřítka (např. Triggsovým algoritmem).

Pokud to je možné, naskýtá se další otázka, a to otázka nutných a postačujících podmínek pro rekonstrukci, tedy z kolika a kterých měřítek jde scéna zrekonstruovat.

8.1 Využití známých měřítek

Jacobsův algoritmus doplňování neznámých míst JIM je založen na myšlence hledání generujících trojic, tj. takových trojic sloupců, v jejichž LO leží prostor L (viz definici 1 s poznámkou 1). Budeme chtít použít podobnou myšlenku na hledání neznámých měřítek v JIM s měřítky, tentokrát ale se čtveřicemi, protože JIM s měřítky má hodnost čtyři. Ze stejného důvodu bude mít také prostor L dimenzi čtyři.

8.1.1 Situace se dvěma obrázky

Pro začátek předpokládejme JIM se dvěma obrázky, která je kompletní. Vyberme čtveřici sloupců A, ve které známe všechna měřítka až na jedno. Tučné písmo značí místo neznámého měřítka $\boldsymbol{\lambda_4^2}$, bez jehož znalosti se pokusíme vytvořit omezující podmínku na L:

$$A = \begin{pmatrix} \lambda_1^1 u_1^1 & \lambda_2^1 u_2^1 & \lambda_3^1 u_3^1 & \lambda_4^1 u_4^1 \\ \lambda_1^1 v_1^1 & \lambda_2^1 v_2^1 & \lambda_3^1 v_3^1 & \lambda_4^1 v_4^1 \\ \lambda_1^1 w_1^1 & \lambda_2^1 w_2^1 & \lambda_3^1 w_3^1 & \lambda_4^1 w_4^1 \\ \lambda_1^2 u_1^2 & \lambda_2^2 u_2^2 & \lambda_3^2 u_3^2 & \lambda_{\mathbf{4}}^{\mathbf{2}} u_4^2 \\ \lambda_1^2 v_1^2 & \lambda_2^2 v_2^2 & \lambda_3^2 v_3^2 & \lambda_{\mathbf{4}}^{\mathbf{2}} v_4^2 \\ \lambda_1^2 w_1^2 & \lambda_2^2 w_2^2 & \lambda_3^2 w_3^2 & \lambda_{\mathbf{4}}^{\mathbf{2}} w_4^2 \end{pmatrix}.$$

Pokud nahradíme neznámé měřítko nulou a přidáme ještě jeden sloupec s nulami v místě prvního obrázku a s hodnotami JIM na místě druhého obrázku:

$$B = \begin{pmatrix} \lambda_1^1 u_1^1 & \lambda_2^1 u_2^1 & \lambda_3^1 u_3^1 & \lambda_4^1 u_4^1 & 0 \\ \lambda_1^1 v_1^1 & \lambda_2^1 v_2^1 & \lambda_3^1 v_3^1 & \lambda_4^1 v_4^1 & 0 \\ \lambda_1^1 w_1^1 & \lambda_2^1 w_2^1 & \lambda_3^1 w_3^1 & \lambda_4^1 w_4^1 & 0 \\ \lambda_1^2 u_1^2 & \lambda_2^2 u_2^2 & \lambda_3^2 u_3^2 & 0 & u_4^2 \\ \lambda_1^2 v_1^2 & \lambda_2^2 v_2^2 & \lambda_3^2 v_3^2 & 0 & v_4^2 \\ \lambda_1^2 w_1^2 & \lambda_2^2 w_2^2 & \lambda_3^2 w_3^2 & 0 & w_4^2 \end{pmatrix}, \tag{8.1}$$

dostaneme pětici sloupců, v jejichž LO má ležet prostor L. Následující věta říká, kdy to nastane.

Lemma 1 *Nechť existuje takové $\lambda_4^2 \in \mathcal{R}$, že sloupce A jsou generující čtveřicí. Potom prostor L leží v LO sloupců matice B.*

Důkaz: Pokud je předpoklad splněn, A je generující čtveřicí a podle poznámky 1 k definici 1 prostor L leží v LO sloupců matice A. Sloupce matice A jsou lineárními kombinacemi sloupců B, proto L leží i v LO sloupců B. □

Poznámka k lemmatu 1: $\lambda_4^2 = 0$ neuvažujeme, protože nulová měřítka nemají pro rekonstrukci smysl. Stejně tak v následujících větách.

Nevíme ale, kdy existuje vhodné λ_4^2 pro splnění předpokladu lammatu. Odpověď dává následující charakterizační lemma.

Lemma 2 *Neznámé měřítko $\lambda_4^2 \in \mathcal{R}$ takové, že sloupce A jsou generující čtveřicí, existuje $\Longleftrightarrow$ matice B má hodnost 5.*

Důkaz:

1. ($\Longrightarrow$) Podle poznámky 1 k definici 1 prostor L leží v LO sloupců matice A s doplněným správným $\lambda_4^2 \neq 0$. Protože v A nechybí žádné položky, sloupce A jsou (jednou z mnoha) bází prostoru L. Podle lemmatu 1 leží

prostor L v LO sloupců B, takže sloupce B jsou (také) bází prostoru L. Pro spor nechť je hodnost B menší než 5.

Pokud je hodnost B menší než 4, bude dimenze prostoru L menší než 4, protože B je bází L. To je ale spor s tím, že L má dimenzi 4, protože JIM s měřítky má hodnost 4.

Nechť tedy má B hodnost 4. První tři sloupce B jsou jistě linárně nezávislé (dále jen LN), protože se shodují s prními třemi LN sloupci A. Bez újmy na obecnosti předpokládejme, že čtvrtým LN sloupcem je čtvrtý sloupec B. Máme dvě báze L: sloupce A a první čtyři sloupce B. Protože se tyto báze shodují v prvních třech vektorech, měl by se jejich čtvtý vektor lišit maximálně o násobek, aby báze generovaly stejný prostor. Označme $\mathbf{p}$ čtvrtý sloupec A a $\mathbf{q}$ čtvrtý sloupec B. $\mathbf{p}$ ale nemůže být násobkem $\mathbf{q}$, protože $\mathbf{p}$ obsahuje vždy nenulové číslo na místě homogenní souřadnice v 6. políčku, kdežto $\mathbf{q}$ má v tomto políčku vždy nulu. To je spor s tím, že naše dvě báze generují stejný prostor L.

2. ($\Longleftarrow$) Pokud B má hodnost 5, jistě lze nalézt takové λ_4^2, že A bude mít hodnost 4 a bude tudíž generující čtveřicí, protože poslední sloupec A je lineární kombinací dvou posledních sloupců B. □

Obě lemmata dohromady říkají:

Důsledek 1 *Pokud je hodnost B rovna 5, jistě L leží v LO sloupců matice B.*

8.1.2 Obecné řešení

Budeme ještě předpokládat, že v JIM s měřítky nechybí žádné význačné body. Tento požadavek odstraníme až na konci sekce.

Umíme tedy klást podmínku na L, když ve čtveřici sloupců schází jedno měřítko. Pokud jich schází více, je postup konstrukce matice B obdobný: za každé chybějící měřítko se na místo jeho výskytu v A dají nuly a přidá se sloupec se samými nulami a hodnotami JIM na místě obrázku, v kterém chybělo dané měřítko. S posledním měřítkem ve sloupci uděláme to, že v něm ponecháme pouze hodnoty JIM bez měřítek. Příklad B pro 4 neznámá měřítka viz 8.3.

Lemmata z 8.1.1 můžeme zobecnit. Důkazy neuvádíme, ale byly by vedeny indukcí.

Lemma 3 *Nechť A je nějaká čtveřice sloupců a nechť lze doplnit neznámá měřítka A tak, že sloupce A jsou generující čtveřicí. Potom prostor L leží v LO sloupců příslušné matice B.*

Lemma 4 *Nechť A je nějaká čtveřice sloupců. Neznámá měřítka A lze doplnit tak, aby byly sloupce A generující čtveřicí $\Longleftrightarrow$ příslušná matice B o rozměrech $3m \times l$, kde $l < 3m$, má hodnost l.*

Důsledek 2 *Pokud je hodnost matice B o rozměrech $3m \times l$, kde $l < 3m$, rovna l, jistě L leží v LO sloupců matice B.*

Pojem generující čtveřice můžeme používat jen na matice hodnosti 4, ale JIM s měřítky, v které jsou neznámá měřítka nahrazena nějakým nenulovým číslem, má vyšší hodnost. Pojem generující čtveřice je tedy nedostatečný, proto definujeme pojem perspektivní čtveřice.

Definice 3 (Perspektivní čtveřice) *Uvažujme scénu s m obrázky bez chybějících bodů. Nechť ve čtveřici sloupců A neznáme několik měřítek. Pokud má příslušná matice B o rozměrech $3m \times l$, kde $l < 3m$, hodnost l, nazveme matici A perspektivní čtveřicí.*

Následuje charakterizační věta perspektivní čtveřice, která je završením dosavadních poznatků.

Věta 2 (Podmínka na L) *Nechť A je perspektivní čtveřicí.*

1. *Prostor L jistě leží v LO příslušné matice B.*

2. *Doplnění neznámých měřítek tak, aby A byla též generující čtveřicí, vždy existuje.*

Důkaz:

1. Viz důsledek 2.

2. Viz lemma 4.

□

Dále budeme pracovat jen s pespektivními čtveřicemi.

Algoritmus doplnění JIM s některými měřítky bude v dalších krocích stejný jako Jacobsův algoritmus: budou se stejným způsobem počítat komplementy a jejich sjednocení.

Pokud bude chybět nějaký význačný bod (ne jen měřítko) ve čtveřici, příslušné tři řádky, do kterých souřadnice význačného bodu zasahují, se nebudou brát do podmínky na L, tj. ani do sloupců matice A a následně ani B.

8.2 Minimální sada měřítek

Zabývejme se nyní otázkou, kolik měřítek a kterých je nutné znát k úspěšné rekonstrukci.

V sekci 6.2.3 jsme lineární prostory reprezentovali maticemi, jejichž sloupce je generují. *Komplementem matice M* budeme rozumět matici, která reprezentuje komplement prostoru, který generují sloupce matice M.

Komplement matice B (8.1), která přísluší perspektivní čtveřici A, má velikost 6×1, protože B velikosti 6×5 má hodnost 5. Kdybychom znali chybějící měřítko, byla by velikost komplementu A větší: 6×2. Je tedy vidět, že došlo ke ztrátě informace. Kdybychom neznali dvě měřítka v matici A, B by mělo velikost 6×6 a hodnost 6, takže by byl jeho komplement prázdný a tudíž nepoužitelný.

Potřebujeme, aby sjednocení komplementů N ze sekce 6.2.3 mělo dimenzi 2, protože L je 4D-podprostor prostoru dimenze 6. K tomu postačí dvě různé perspektivní čtveřice. Různé v tom smyslu, že generují prostory, které nejsou totožné. Ty můžeme vybrat z pěti nezávislých (upřesnění tohoto pojmu přijde za chvíli) sloupců, jak lze vidět na následujícím příkladě. V matici $I \in \mathcal{R}^{m \times n}$ mějme tentokrát informaci o známých (1) a neznámých (0) měřítkách.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Pokud v nějakém sloupci čtveřice známe pouze jediné měřítko, vyzískáme stejně informace, jako bychom neznali žádné, viz konstrukce matice B v Proto takové měřítko nebudeme započítávat, ani vyznačovat jedničkou.

Nyní uvažujme JIM se třemi obrázky. L je 4D-podprostor prostoru dimenze 9, N musí mít dimenzi 5. Pokud neznáme jedno měřítko ve čtveřici, B má rozměry 9×5, takže jeho komplement má dimenzi 4. Opět stačí dvě různé perspektivní čtveřice, tedy 5 sloupců. Stejně bychom mohli ve zvyšování počtu obrázků pokračovat, takže dostáváme první omezení ze spoda pro rekonstrukci tímto algoritmem:

Věta 3 *K rekonstrukci scény z alespoň dvou obrázků algoritmem 8.1 je třeba znát alespoň 5 nedegenerovaných význačných bodů společných všem obrázkům a z nich alespoň 3 vhodná měřítka společná všem.*

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Poznámky k větě 3:

1. Nedegenerovanými význačnými body s vhodnou podmnožinou společných měřítek se myslí takové body, pro něž lze vybrat dostatek perspektivních čtveřic pro úplnou rekonstrukci. To je řečeno poněkud „kruhem“, ale zatím nebylo blíže zjištěno, co to znamená. Tato otázka je předmětem k dalšímu bádání.

2. Toto se umí i nelineárním postupem s využitím trifokálního tenzoru. Naše metoda je lineární, viz 6.2.4, bod 4. To je náš přínos, vlastně využití těch známých měřítek.

3. Dalším přínosem je to, že potřebujeme pouze 5 bodů a 3 společná měřítka. Oproti tomu nejlepší existující lineární algoritmus potřebuje 7 bodů, tzv. sedmibodový algoritmus.

V úvahách můžeme pokračovat snížením počtu společných měřítek na dvě. Ukáže se, že je třeba zvýšit počet obrázků i bodů.

Věta 4 *K rekonstrukci scény z alespoň tří obrázků algoritmem 8.1 je třeba znát alespoň 6 nedegenerovaných význačných bodů společných všem obrázkům a z nich alespoň 2 měřítka společná všem.*

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Důkaz: Čtveřici, která obsahuje jen jeden sloupec se známými měřítky, odpovídá B o rozměrech 9×10, takže nemůže mít hodnost 10, což je podmínka pro perspektivní čtveřici. Stejně není perspektivní čtveřice, která neobsahuje žádný sloupec se známými měřítky. Perspektivní jsou pouze čtveřice, které obsahují dva sloupce se známými měřítky, protože jim odpovídá B o hodnosti 8 a doplněk B hodnosti 1.

N má hodnost 5, takže nám „různých“ perspektivních čtveřic stačí 5. Ty lze vybrat ze šesti sloupců: dva sloupce jsou obsaženy vždy, ze zbývajících čtyř vybíráme dva, což dává $\begin{pmatrix} 4 \\ 2 \end{pmatrix} = 6$ možností. Pět sloupců nestačí, protože poskytují pouze $\begin{pmatrix} 3 \\ 2 \end{pmatrix} = 3$ možnosti. □

Poznámka k větě 4: To je také úspěch, protože k výpočtu trifokálního tenzoru lineární metodou je potřeba 7 bodů, kdežto zde jen 6 a 2 společná měřítka.

8.3 Bez měřítek to nepůjde

Ptejme se dále. Je možná taková konfigurace známých měřítek, v které není známé žádné měřítko v nějakém obrázku?

Odpověď bohužel zní ne. Bez újmy na obecnosti to ukážeme na konfiguraci, v které známe všechna měřítka ve všech obrázcích kromě posledního. Důkaz, že pro tuto konfiguraci nelze rekonstruovat poslední obrázek, bude stačit, protože ostatní konfigurace jsou pro rekonstrukci jenom těžší.

Vezměme nějakou čtveřici sloupců JIM s měřítky pro m obrázků. Tučné písmo bude opět značit neznámá měřítka $\lambda^{\mathbf{m}}_{\mathbf{1}}, \ldots, \lambda^{\mathbf{m}}_{\mathbf{4}}$ v posledním obrázku:

$$A = \begin{pmatrix} \lambda^1_1 u^1_1 & \lambda^1_2 u^1_2 & \lambda^1_3 u^1_3 & \lambda^1_4 u^1_4 \\ \lambda^1_1 v^1_1 & \lambda^1_2 v^1_2 & \lambda^1_3 v^1_3 & \lambda^1_4 v^1_4 \\ \lambda^1_1 w^1_1 & \lambda^1_2 w^1_2 & \lambda^1_3 w^1_3 & \lambda^1_4 w^1_4 \\ \vdots & & & \vdots \\ \lambda^{\mathbf{m}}_{\mathbf{1}} u^m_1 & \lambda^{\mathbf{m}}_{\mathbf{2}} u^m_2 & \lambda^{\mathbf{m}}_{\mathbf{3}} u^m_3 & \lambda^{\mathbf{m}}_{\mathbf{4}} u^m_4 \\ \lambda^{\mathbf{m}}_{\mathbf{1}} v^m_1 & \lambda^{\mathbf{m}}_{\mathbf{2}} v^m_2 & \lambda^{\mathbf{m}}_{\mathbf{3}} v^m_3 & \lambda^{\mathbf{m}}_{\mathbf{4}} v^m_4 \\ \lambda^{\mathbf{m}}_{\mathbf{1}} w^m_1 & \lambda^{\mathbf{m}}_{\mathbf{2}} w^m_2 & \lambda^{\mathbf{m}}_{\mathbf{3}} w^m_3 & \lambda^{\mathbf{m}}_{\mathbf{4}} w^m_4 \end{pmatrix}.$$

Z matice A vytvoříme stejným způsobem jako v 8.1 matici B pro podmínku pro prostor L^1:

$$B = \begin{pmatrix} \lambda^1_1 u^1_1 & \lambda^1_2 u^1_2 & 0 & \lambda^1_3 u^1_3 & 0 & \lambda^1_4 u^1_4 & 0 \\ \lambda^1_1 v^1_1 & \lambda^1_2 v^1_2 & 0 & \lambda^1_3 v^1_3 & 0 & \lambda^1_4 v^1_4 & 0 \\ \lambda^1_1 w^1_1 & \lambda^1_2 w^1_2 & 0 & \lambda^1_3 w^1_3 & 0 & \lambda^1_4 w^1_4 & 0 \\ \vdots & & & & & & \vdots \\ \lambda^m_1 u^m_1 & 0 & u^m_2 & 0 & u^m_3 & 0 & u^m_4 \\ \lambda^m_1 v^m_1 & 0 & v^m_2 & 0 & v^m_3 & 0 & v^m_4 \\ \lambda^m_1 w^m_1 & 0 & w^m_2 & 0 & w^m_3 & 0 & w^m_4 \end{pmatrix}.$$

Pro každý sloupec komplementu B musí platit, že jeho skalární součin s každým sloupcem B je roven nule. Označme

$$\mathbf{b} = (a^1 b^1 c^1 \ldots a^m b^m c^m)^T$$

nějaký sloupec komplementu B. Skalární součiny $\mathbf{b}$ se třetím, s pátým a se sedmým sloupcem B dávají následující soustavu rovnic:

$$\begin{aligned} u^m_2 a^m + v^m_2 b^m + w^m_2 c^m &= 0 \\ u^m_3 a^m + v^m_3 b^m + w^m_3 c^m &= 0 \\ u^m_4 a^m + v^m_4 b^m + w^m_4 c^m &= 0, \end{aligned}$$

[1]V následující matici je použito měřítko λ^m_1 jako bychom ho znali, neboť jedno měřítko v obrázku můžeme nastavit, např. na 1, viz 7. Ale i kdybychom neměli použít toto měřítko, B by byla o jeden sloupec větší a na zbytek důkazu by to nemělo vliv.

krátce

$$S\mathbf{x} = \mathbf{0},\ \text{kde } S = \begin{pmatrix} u_2^m & v_2^m & w_2^m \\ u_3^m & v_3^m & w_3^m \\ u_4^m & v_4^m & w_4^m \end{pmatrix} \text{ a } \mathbf{x} = \begin{pmatrix} a^m \\ b^m \\ c^m \end{pmatrix}.$$

Pokud je matice S regulární, musí se $\mathbf{x}$ rovnat nule. Poslední tři políčka $\mathbf{b}$ jsou tedy nulová a nulové jsou i poslední tři řádky komplementu B. To znamená, že jsme nezískali žádné omezení na prostor L co se týče posledního obrázku, takže ho ani nemůžeme rekonstruovat.

Zbývá už jen dokázat, že S je regulární. Pokud S není regulární, B nemá hodnost 7, protože S^T je obsažena v B jako podmatice na takových místech, kde zbytky sloupců jsou nuly. To je spor s tím, že A je perspektivní čtveřicí, takže by B mělo mít podle definice 3 hodnost 7.

Kapitola 9

Experimenty a demonstrace

Algoritmy se testovaly na reálných a umělých scénách. Do snímků scén byl přidáván náhodný a normální šum a byly odebírány význačné body, čímž se testovala robsustnost algoritmů.[1]

9.1 Jaké chyby sledovat

Označme J správnou JIM (v literatuře „ground truth"), $\tilde{J}$ zašuměnou JIM, $\tilde{J} = J + E$, kde E je nějaký šum, J' rekonstruovanou JIM. Chceme, aby $J' = J$ bez ohledu na to, jak vypadá $\tilde{J}$.

Matici JIM můžeme považovat za bod ve vekt. prostoru $\mathcal{V}$, když její sloupce sloučíme v jediný vektor. Metrika $d : \mathcal{V} \times \mathcal{V} \longrightarrow \mathcal{R}$ definovaná předpisem

$$d(I, J) = \sum_i \sum_j (I_{i,j} - J_{i,j})^2$$

je známá Frobeniova norma. (Jedná se vlastně jen o vzdálenost dvou bodů ve vekt. prostoru.) Máme dva procesy,

- zašumění: $J \Longrightarrow \tilde{J}$
- a rekonstrukci: $J' \Longleftarrow \tilde{J}$,

které vedou ke sledování vzdáleností $d(J, J')$, $d(J, \tilde{J})$ a $d(\tilde{J}, J')$. Pro sledování kvality algoritmů rekonstrukce je nejdůležitější vzdálenost $d(J, J')$, chyba reprojekce. Ovšem ne vždy máme k dispozici správnou JIM J, zvláště při testech na reálných scénách. Tehdy budeme považovat za chybu reprojekce pouze její aproximaci $d(\tilde{J}, J')$.

[1] Odolnost proti záměně korespondencí testována nebyla.

9.2 Počítání chyb

Do JIM se přidává šum E s normálním $N(0, \sigma^2)$ a s náhodným $R(0, \sigma^2)$ rozdělením. Šum má nulovou střední hodnotu, rozptyl chceme regulovat vzhledem k velikosti obrázků, protože je přirozené chtít, aby byl šum v obrázku tím větší, čím větší je obrázek. To, proč je toto opatření důležité, se vysvětlí vzápětí.

U reálných scén lze předpokládat, že souřadnice význačných bodů budou mít na všech snímcích zhruba stejný rozsah, který je zpravidla určen velikostí obrázků v pixelech. Pokud je tato podmínka splněna, je spravedlivé rozmístění šumu ve všech obrázcích zaručeno bez jakýchkoliv normalizací.

U umělých scén, získaných pouhým promítnutím scény volně (nebo náhodně) umístěnou kamerou, může ale dojít k situaci, kdy se rozsah souřadnic bodů na některých snímcích velmi liší od rozsahu souřadnic na jiných snímcích. Rovnoměrné přidání šumu do takové scény proto způsobí, že je do některých snímků zavlečena malá a do některých snímků velká chybu vzhledem k rozsahu jeho souřadnic. Některé algoritmy jsou citlivé na takové nerovnoměrnoměrnosti a proto při testech vykazují horší výsledky, čímž ztěžují vyvozování správných závěrů o vlastnostech algoritmu.

Chceme tedy mít pro všechny scény stejný metr, tak aby bylo počítání chyb invariantní vzhledem k velikosti obrázků. Řešením je přidávat do každého obrázku scény jiný šum, a to s rozptylem vynásobeným velikostí obrázku.

9.2.1 Co považovat za velikost obrázku

Je třeba stanovit, co považovat za velikost obrázku a jaké vlastnosti by měla tato veličina splňovat. Je přirozené od velikosti očekávat, že bude invariantní vzhledem k posunutí popř. otočení, tj. že velikost obrázku se nezmění, když ho posuneme nebo otočíme. Invarianci vzhledem ke zkosení a natažení nebudeme považovat pro naše účely počítání chyb za relevantní. Dalším požadavkem je robustnost vzhledem k nepřítomnosti nějakého bodu, protože v JIM body často chybí a je tedy rozumné chtít, aby se velikost obrázku příliš nezměnila, pokud z něj odebereme nějaký význačný bod.

Asi nejsnazší způsob je vzít rozdíl minimální a maximální souřadnice bodů v různých dimenzích. Tento přístup je invariantní vzhledem k posunutí, nedává však uspokojivé výsledky pro scény, kde je většina bodů nahuštěna blízko sebe a jen několik z nich je značně vzdálených od tohoto shluku. Posunutí nebo absence jednoho ze vzdálených bodů by velmi ovlivnilo velikost takového obrázku a tudíž i množství přidávaného šumu.

Nebudeme tedy započítávat jenom extrémní body na okrajích, ale všechny body. Následující dva přístupy splňují výše zvýšené požadavky. Velikost obrázku můžeme chápat buď jako průměrnou Eukleidovskou vzdálenost bodů od jejich těžiště nebo jako průměr (nebo součet druhých mocnin apod.) standardních odchylek v různých dimenzích. První přístup má všechny požadované vlastnosti (invariance vzhledem posunutí a otočení, robustnost vůči chybějícím význačným bodům), druhý také všechny kromě invariance vzhledem k otočení. Obě možnosti nyní rozebereme.

1. Velikost obrázku jako průměrná Eukleidovská vzdálenost bodů od jejich těžiště.

   Je potřeba uvažovat vzdálenost od těžiště obrázku a ne od počátku soustavy souřadné kvůli invarianci vzhledem k posunutí. Velikost obrázku počítaná vzhledem k těžišti bodů je totiž menší než velikost stejného obrázku vzhledem k bodu daleko od těžiště.

2. Velikost obrázku jako průměr (nebo součet druhých mocnin apod.) standardních odchylek v různých dimenzích.

   Proč je zapotřebí počítat std. odchylky zvlášť v různých dimenzích? Jinými slovy, nestačilo by vzít std. odchylku z normalizované JIM[2]?

   Důvod tkví opět v invarianci vzhledem k posunutí. Při posunutí obrázku v jedné dimenzi by se hodnota velikosti nežádoucím způsobem změnila, jak to ukazuje následující tabulka, kde $\vec{x}$ a $\vec{y}$ označují vektory souřadnic obrázků ve dvou dimenzích.

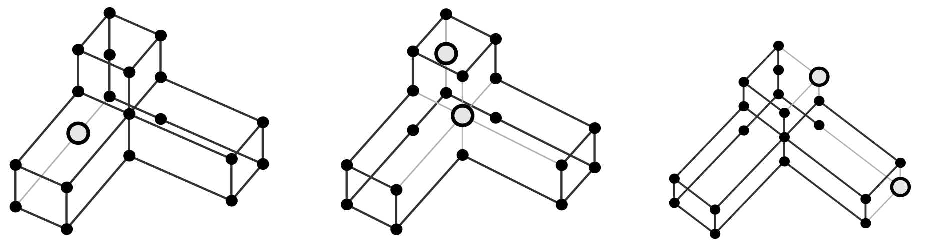
Způsob výpočtu	**Velikost původního obrázku**	**Velikost obrázku posunutého v dimenzi x o t**
zvlášť po dimezních	$mean(std(\vec{x}), std(\vec{y}))$	$mean(std(\vec{x}), std(\vec{y}))$
dohromady	$std(union(\vec{x}, \vec{y}))$	$std(union(\vec{x}+t, \vec{y}))$

Poznámka: K tomu, aby se zde definavaný pojem velikosti blížil vizuální velikosti objektu na obrázku, je vhodné brát čtyřnásobky uvedených hodnot.

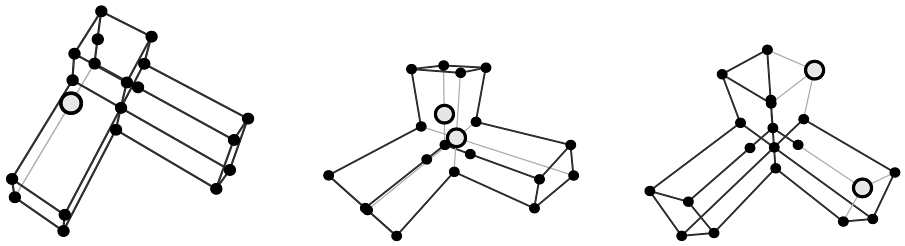
9.3 Experimenty s umělými scénami

K experimentům byla použita jednoduchá scéna s kostkami o 20-ti bodech, která byla snímána ortografickou a perspektivní kamerou. Příklady obrázků scény jsou na obr. 9.1(a) a 9.1(b). Na těchto obrázcích snímaly kamery scénu

[2] *Normalizovaná JIM* se definuje pro JIM scén s perspektivní kamerou. Obsahuje pouze x- a y-souřadnice význačných bodů, které jsou vyděleny homogenní souřadnicí.



(a) Ortografická kamera



(b) Perspektivní kamera

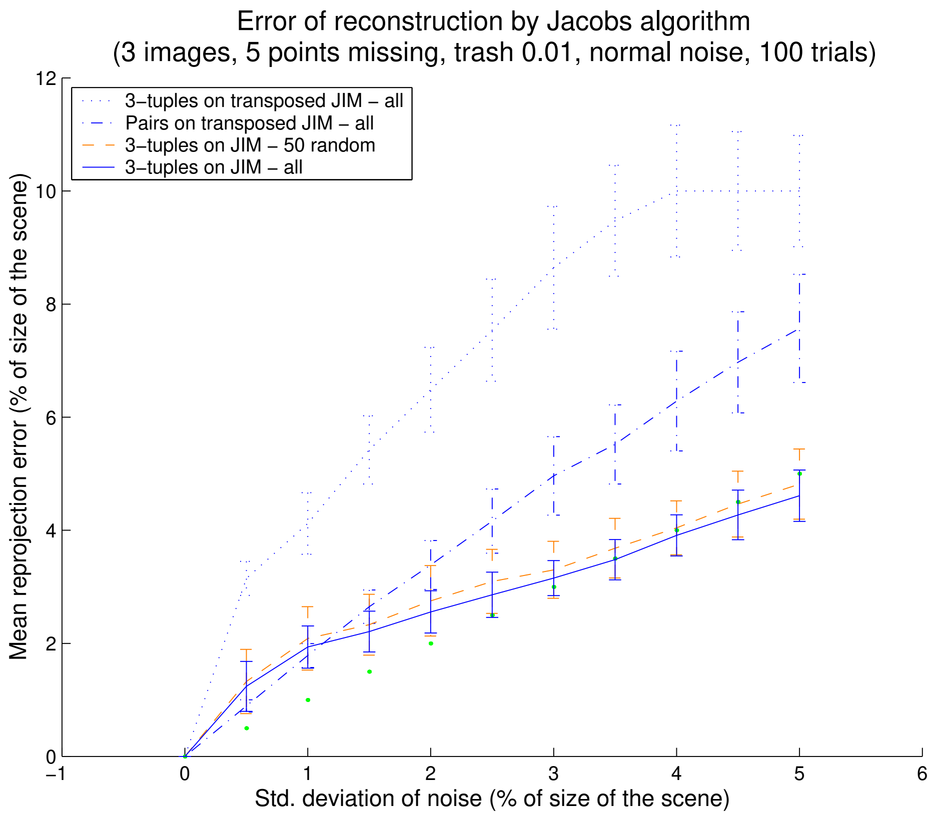
Obrázek 9.1: Scéna s kostkami

ze stejných pozic a úhlů, obrázky se liší pouze druhem kamery. Větší body se světlým vnitřkem označují neznámé body scény, které odpovídají chybějícím políčkům v JIM.

Grafy byly vypracovány pro JIM se třemi a s pěti obrázky.[3] Na každém grafu je pod názvem algoritmu uveden počet obrázků scény, počet chybějících políček v JIM, druh přidávaného šumu a počet opakování spuštění algoritmu pro stejné množství šumu. Díky tomu, že se algoritmus spouští vícekrát pro stejný šum, je možné spočítat i směrodatnou odchylku chyby algoritmu. Ta je vyznačena svislým intervalem u každé hodnoty grafu funkce. Na grafech, které se nějak týkají Jacobsova algoritmu, je uveden jestě práh pro svd, více viz 9.3.1.

Na vodorovné ose je std. odchylka přidávaného šumu v procentech velikosti scény. Myslí se tím, že se přidává do každého obrázku takové množství

[3]Výsledky pro více obrázků nejsou zahrnuty především z důvodu nedostatku místa, ale také proto, že se už křivky grafů dalším zvyšováním počtu obrázků výrazně nemění. To, že algoritmy fungují i pro vyšší počet obrázků, je ukázáno experimenty s reálnými scénami.



Obrázek 9.2: Chyba reprojekce Jacobsova algoritmu pro 3 obrázky

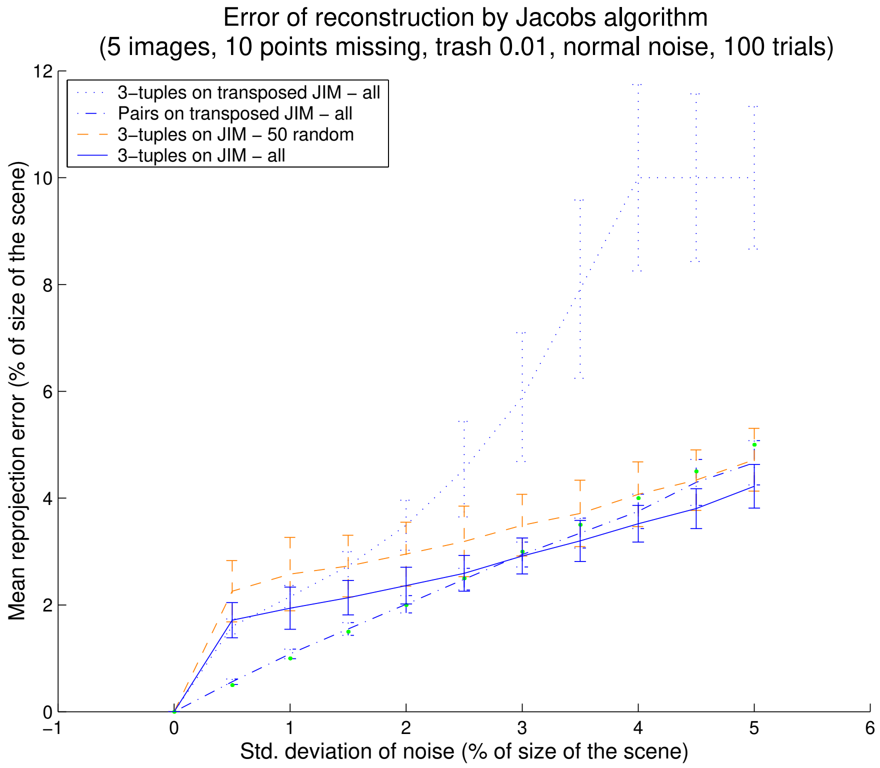
šumu, které je přímoúměrné jeho velikosti. Na svislé ose je chyba algoritmu, opět v procentech velikosti scény. Chyba zašuměné scény je také zobrazena drobnými zelenými body, aby bylo patrné, jestli se grafy funkcí dostanou pod původní šum.

V grafech je vynesena vždy jen chyba reprojekce neboli vzdálenost $d(J, J')$, protože ostatní dvě vzdálenosti si jsou velmi podobné.

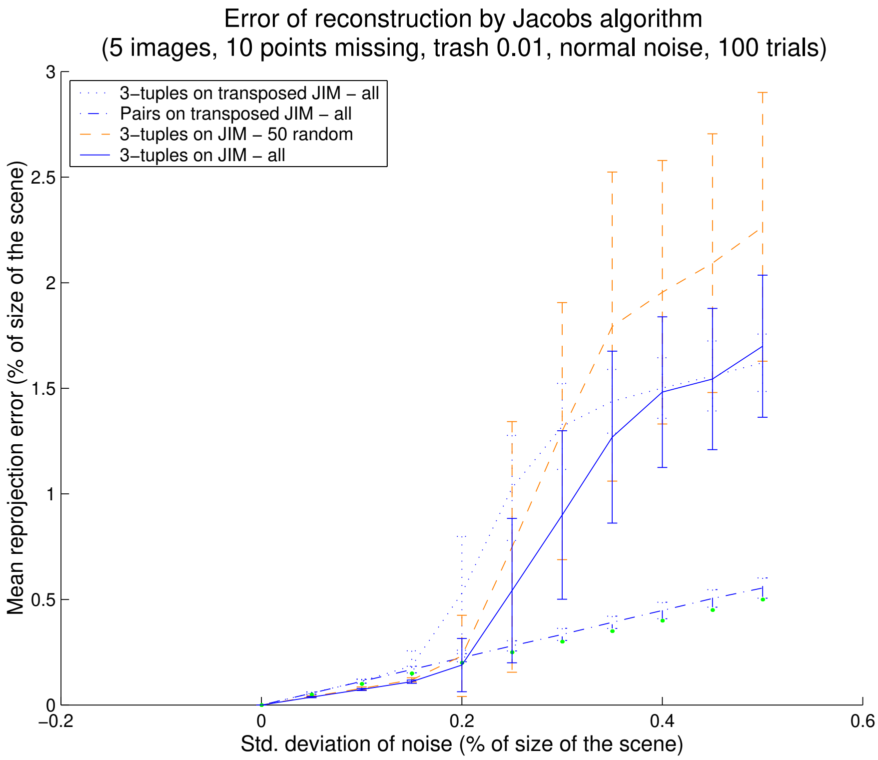
9.3.1 Jacobsův algoritmus

V Jacobsově algoritmu se vybírají trojice sloupců, ale u velkých JIM se v rozumném čase nemusí podařit vybrat všechny, což je ideální případ, a proto se jich vybere jen několik náhodných. Tento počet se omezuje jednak velikostí hledaného nulového prostoru N, jednak počtem pokusů pro výběr trojic[4] nebo párů. Znamená to, že pokud se podaří nagenerovat dostatečně velký nulový prostor, algoritmus končí, stejně jako v případě, že se zkusilo dost

[4]Navíc ne všechny trojice jsou generující.



(a) Šum do pěti procent



(b) Šum do půl procenta

Obrázek 9.3: Chyba reprojekce Jacobsova algoritmu pro 5 obrázků trojic.

Experiment 1 *V tomto experimentu se testuje vliv šumu a počtu obrázků na chybu reprojekce Jacobsova algoritmu. Porovnávají se všechny tři varianty Jacobsova algoritmu z 6.2.6: varianta s JIM, s transpozicí JIM a s páry obrázků. Navíc se srovnávají výsledky při použití všech nebo několika náhodných trojic sloupců.*

Algoritmus, jehož chyba reprojekce je zobrazena na obrázcích 9.2 a 9.3, je omezen tím, že nulový prostor může mít maximálně 50-tinásobek počtu sloupců JIM nebo její transpozice. Je testováno maximálně 1000 náhodných výběrů. Na obrázcích není uvedena chyba pro 50 náhodných trojic v transpozici a 50 náhodných párů obrázků. Je to proto, že počet trojic v transpozici nebo párů obrázků, které je možno vytvořit s pěti obrázky o dvaceti bodech je o mnoho menší než 1000, takže se zkoušejí všechny.

Z těchto grafů je patrné, která varianta algoritmu je nejlepší. Pokud je šum vyšší než zhruba jedno nebo tři procenta podle počtu obrázků, je nejlepší varianta s JIM. V oblasti velmi malých šumů je jasně nejlepší varianta s páry. Je velmi důležité, že tu chyba algoritmu roste lineárně s šumem, bez skoků jako u jiných variant, jejichž průběh je patrný na detailním obr. 9.3(b). Z obrázků 9.2 a 9.3(a) lze usuzovat, že s dalším zvyšováním počtu obrázků se bude varianta s páry stále zlepšovat oproti ostatním i pro vysoký šum.

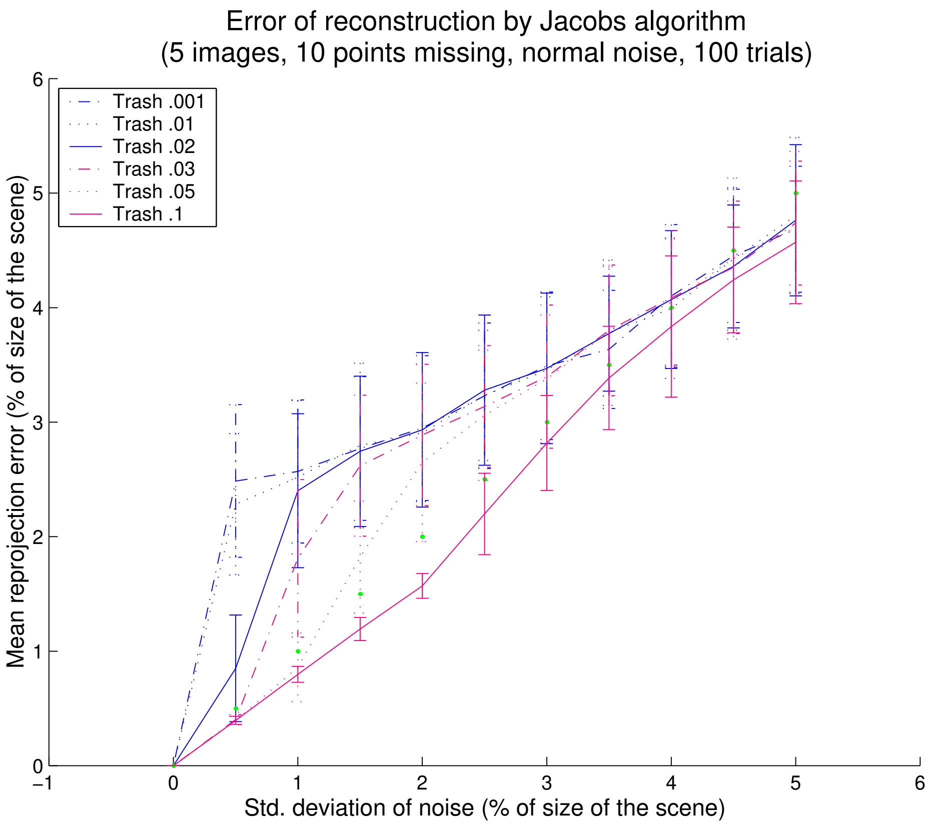
Je překvapivé, jak dobré výsledky dává Jacobsův algoritmus, když nejsou použity všechny trojice. Je to vidět na variantě s JIM, kde si jsou křivky pro několik náhodných a pro všechny trojice velmi blízko. V této variantě se vyzkoušelo vždy něco mezi deseti a dvaceti procenty všech trojic.

Práh pro svd

Jacobsův algoritmus je závislý na jedné konstantě, která se musí v programu určit explicitně. Jedná se o rozpoznání generující trojice ze sekce 6.2.1, kde je potřeba rozhodnout o dimenzi prostoru tj. hodnosti matice. Za přítomnosti šumu je hodnost vždy plná, proto se použije svd. Nalezená vlastní čísla je potřeba rozdělit na ta, která ještě přísluší vlastním vektorům uvnitř prostoru, a ta, která přísluší již vektorům vně. Tento práh se určuje explicitně.

Jacobs ve svém kódu doporučuje maximálně hodnotu prahu 0.1. Optimální nastavení prahu ale závisí na konkrétní scéně, množství šumu a na souřadnicovém systému, který ovlivňuje velikost vlastních čísel matic trojic sloupců.

Experimentálně lze ale vypozorovat vliv prahu na efektivitu algoritmu pro konkrétní scénu. Z výsledků lze potom stanovit optimální nastavení prahu.



Obrázek 9.4: Jacobsův algoritmus s různými prahy pro svd

Experiment 2 *Tento experiment demonstruje vliv nastavení prahu Jacobsova algoritmu na chybu reprojekce při různém šumu.*

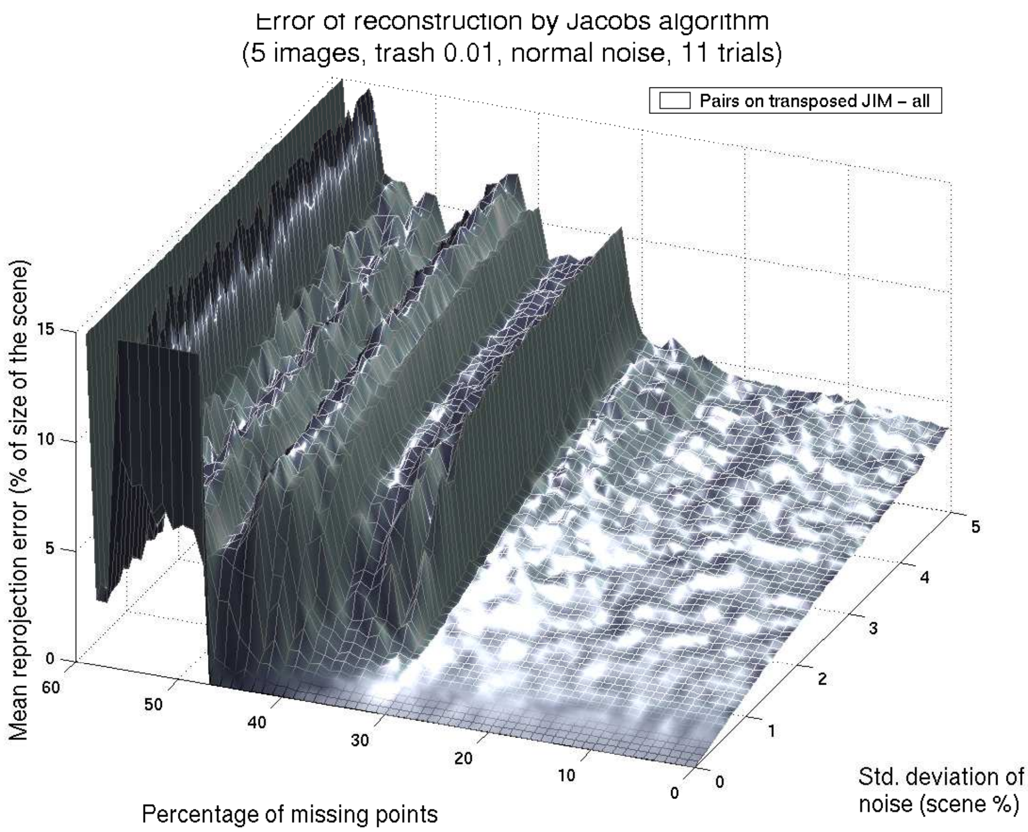
Na obrázku 9.4 je průběh varianty s JIM.

Z tohoto obrázku se vyšší prahy jeví pro tuto scénu s pěti obrázky jako lepší. Např. zvyšováním prahu z 0.01 by se graf varianty s JIM na obr. 9.3(b) stával lineárním i v oblasti do jednoho procenta šumu, neboť v tomto rozsahu vykazuje tato varianta velmi dobré výsledku na obr. 9.4.

Závislost na chybějících datech

Čím více bodů chybí, tím je rekonstrukce těžší. Počet bodů, které lze rekonstruovat, klesá s rostoucím počtem chybějících políček v JIM. Od určité hranice nelze rekonstuovat žádné body scény. Čím je více obrázků, tím více bodů může chybět.

Experiment 3 *Experiment ukazuje závislost kvality rekonstrukce Jacobsovým algoritmem na šumu a na množství chybějících dat.*



Obrázek 9.5: Závislost Jacobsova algoritmu na chybějících datech

Body byly odebírány rovnoměrně z obrázků a dbalo se také na to, aby byly jednotlivé body rovnoměrně zastoupeny. Počet chybějících význačných bodů se ve všech obrázcích lišil maximálně o 1, počet obrazů každého bodu se také lišil maximálně o 1. Čím více obrázků máme, tím více bodů může chybět. 60 % chybějících bodů v pěti obrázcích je možné maximum pro rekonstrukci všech bodů, protože každý bod musí být alespoň ve dvou obrázcích, aby byl Jacobsovým algoritmem doplněn[5].

Na obr. 9.5 je vidět, že rekonstrukce v případě scény s pěti obázky na 60-ti procentech vždy selhala, což je vyznačeno hodnotou grafu 15.

Na hranici 30-ti % chybějících dat se zvedá nápadné pohoří s několika lokálními maximy okolo 10-ti %. Je způsobeno tím, že některé body sice byly rekonstruovány, bohužel ale tak špatně, že přispěly velkou chybou. Díky dalšímu přibývání chybějících bodů se tyto body nerekonstruovaly vůbec, do chyby reprojekce se nezapočítaly, a to způsobilo opět pokles. Významné lo-

[5]odpovídá mu sloupec se čtyřmi neprázdnými políčky, což je více než potřebné tři pro rekonstrukci sloupce

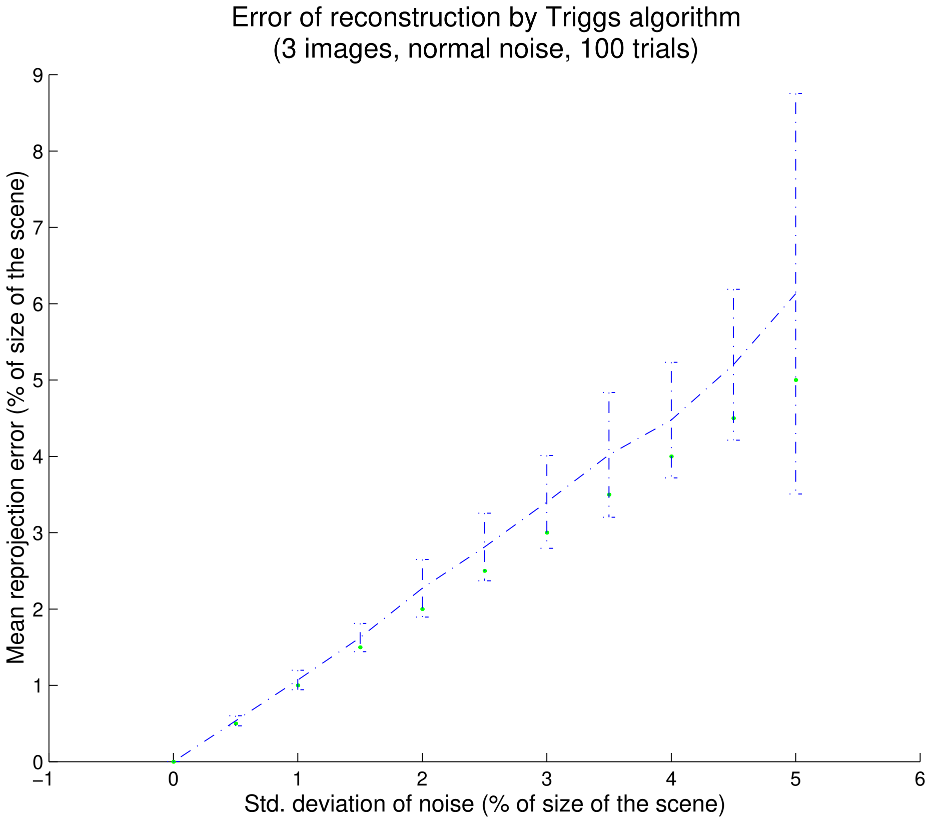
kální minimum je mezi 57 a 58 procenty chybějících dat s chybou pouze 3 %. Zajímavé je, že algoritmus nefungoval pro nulový šum v oblasti 48 – 56 % chybějících dat.

9.3.2 Triggsův algoritmus

Triggsův algoritmus nezávisí na žádných vnitřních nastavitelných parametrech jako je práh pro svd u Jacobsova algoritmu. Fundamentální matice a epipóly se počítají jednoduchým osmibodovým algoritmem, výpočet měřítek je také jednoduchá lineární algebra, Proto bychom mohli očekávat, že průběh grafu funkce závislosti na šumu neměl být tak divoký jako u Jacobsova algoritmu.

Experiment 4 *V tomto experimentu se testuje vliv šumu na kvalitu Jacobsova algoritmu.*

Výsledky testů jsou na obr. 9.6.



Obrázek 9.6: Chyba reprojekce Triggsova algoritmu

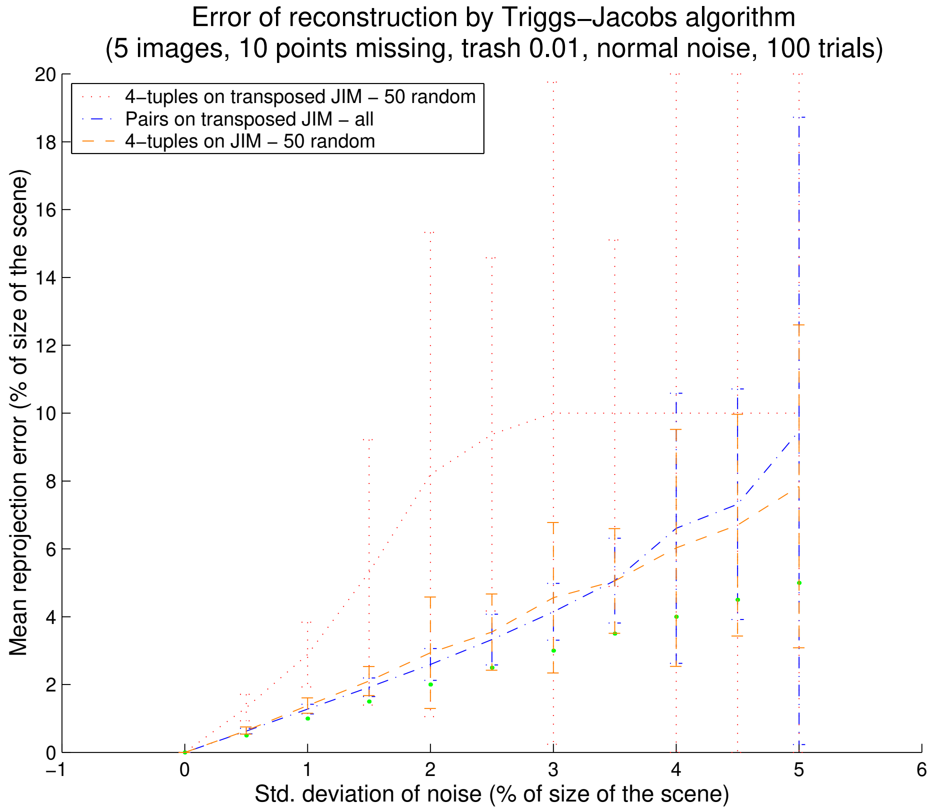
Z grafu je vidět, že chyba reprojekce Triggsova algoritmu roste lineárně se šumem ve scéně. Tento graf je pro JIM se třemi obrázky, přidáváním dalších obrázků se chyba zmenšuje.

9.3.3 Algoritmus pro chybějící políčka

Vzhledem k tomu, že jsme viděli, že Triggsův algoritmus má lineární chybu s rostoucím šumem, bychom mohli očekávat, že se algoritmus 7.3 pro chybějící políčka, který vychází z Triggsův a Jacobsova algoritmu, bude chovat podobně jako Jacobsův algoritmus.

Kvůli opakování volání algoritmu na podmatice JIM lze očekávat spíše změnu k horšímu, protože se chyba z minulých volání takto spíše kumuluje než že by se kompletně rozptýlila.

Experiment 5 *Experiment má demonstrovat vliv šumu na kvalitu rekonstrukce algoritmem 7.3 pro chybějící políčka. Srovnávají se různé varianty Jacobsova algoritmu doplňování políček.*



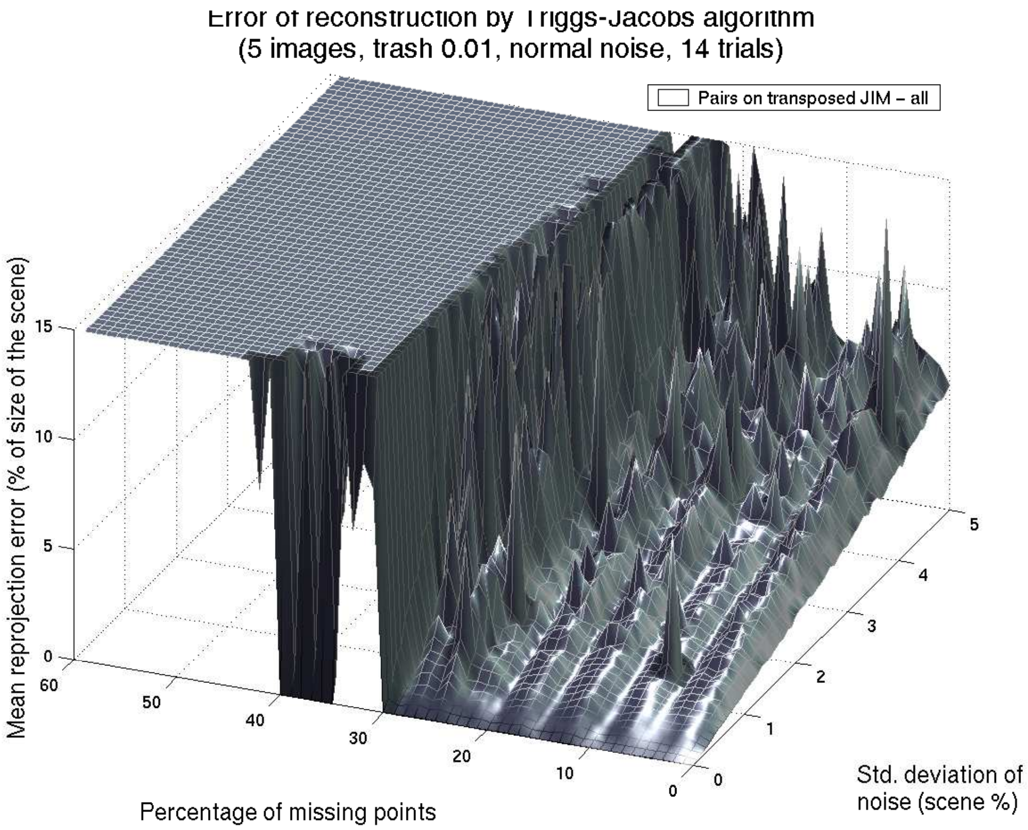
Obrázek 9.7: Chyba reprojekce algoritmu s chybějícími políčky

Test byl proveden pro pět obrázků tak, aby se algoritmus 7.3 pro chybějící políčka zavolal při každé rekonstrukci vícenásobně, aby byla otestována robustnost vůči počtu iterací. To se dosáhlo v případě testu na pevném množství chybějících dat tím, že v každém obrázku chyběl alespoň jeden bod, u testu závislosti na chybějících datech nebylo potřeba žádné opatření.

Test nebyl proveden pro všechny čtveřice v JIM a v její transpozici kvůli výpočetní náročnosti při 100 opakování testu. Navíc aproximace náhodnými trojicemi se v experimentu 1 Jacobsova algoritmu ukázala jako dostatečná.

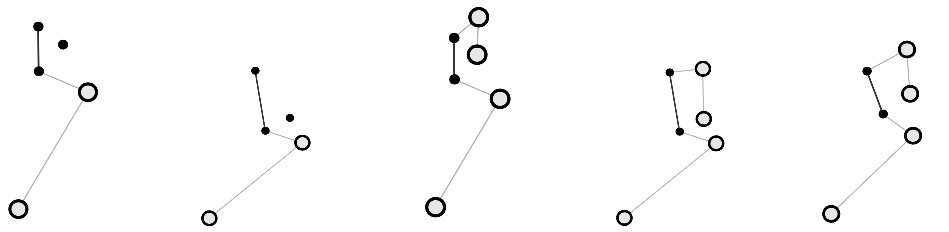
Výsledek testu je na obr. 9.7. Potvrzuje očekávání, že bude průběh chyby jednotlivých variant podobný jako u Jacobsova algoritmu, viz experiment 1.

Závislost na chybějících datech



Obrázek 9.8: Závislost algoritmu s chybějícími políčky na chybějících datech

Vše, co platí pro závislost Jacobsova algoritmu na chybějících datech v 9.3.1, platí i pro algoritmus s chybějícími políčky. Navíc ale, kvůli opakovanému volání, se bude, jak už bylo řečeno výše, chyba z minulého volání



(a) Pro dva obrázky (b) Pro tři obrázky

Obrázek 9.9: Minimální konfigurace měřítek

kumulovat, což se u pohoří v grafu experimentu 3 projeví jeho velkým zvýšením, protože už tam je vysoké.

Experiment 6 *Experiment ukazuje závislost kvality rekonstrukce algoritmem 7.3 pro chybějící políčka na šumu a na množství chybějících dat.*

Experiment byl proveden na stejných datech jako experiment 3, proto, aby bylo možné srovnání, zda se pohoří z experimentu 3 objeví na stejných místech.

Body byly odebírány rovnoměrně z obrázků stejným způsobem jako v experimentu 3. Čím více obrázků máme, tím více bodů může chybět. Opět 60 % chybějících bodů v pěti obrázcích je možné maximum pro rekonstrukci všech bodů. Graf chyby reprojekce byl omezen maximální hodnotou 15, aby byl dobře viditelný průběh chyby v oblasti s menším množstvím chybějících dat.

Výsledek experimetnu je na obr. 9.8. Pro množství chybějících dat nad 30 procent začíná ono pohoří z experimentu 3.

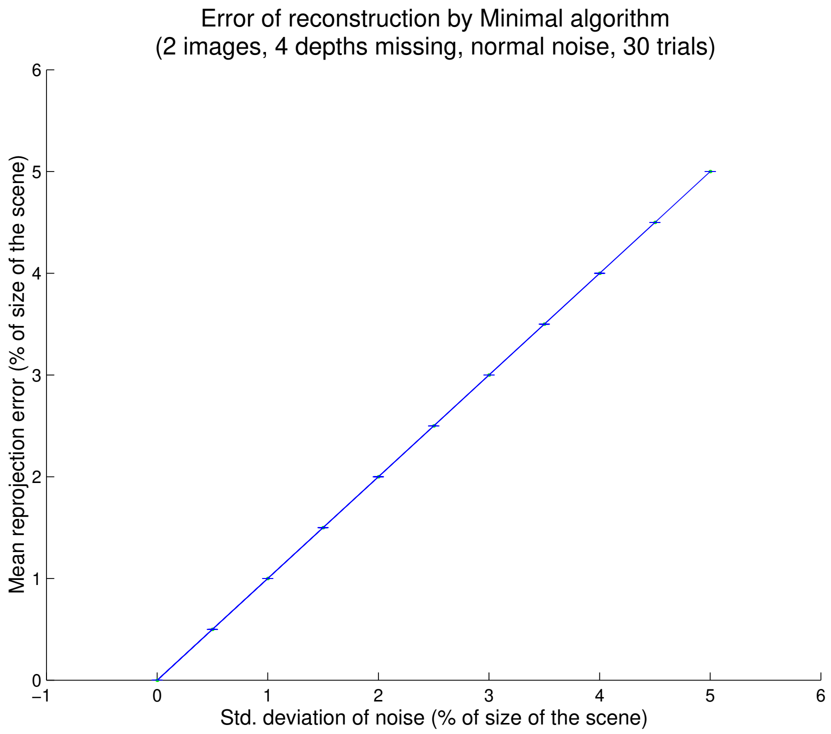
Opět je zajímavé, že se objevila oblast, tentokrát mezi 30 – 35 % chybějících dat, kde algoritmus nefungoval pro nulový šum a pak se opět na malém úseku chytil.

Špičaté výčnělky lokálních maxim jsou způsobeny především malým počtem opakování algoritmu.

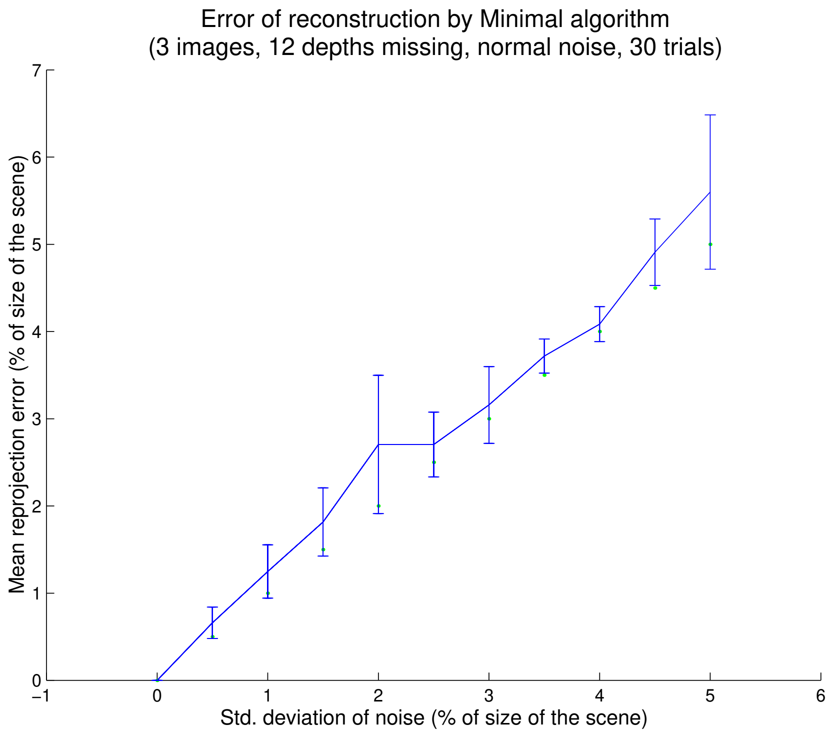
9.3.4 Když jsou známá některá měřítka

Věty 3 a 4 ze sekce 8.2 demonstrujeme na scénách s minimálním počtem bodů a měřítek, které tyto věty vyžadují.

Na obr. 9.9(a) a 9.9(a) jsou výřezy několika bodů z obrázků perspektivní scény s kostkami 9.1.



(a) Pro dva obrázky



(b) Pro tři obrázky

Obrázek 9.10: Chyba reprojekce algoritmu s měřítky

Obrázek 9.11: Příklad obrázku scény „domek“

Experiment 7 *Tento experiment chce ukázat závislost kvality rekonstrukce algoritmem s měřítky 8.1.2 na šumu pro minimální nutné konfigurace bodů a měřítek podle vět 3 a 4.*

Experiment je prováděn na scénách s minimálními předpoklady jednotlivých vět, které jsou na obr. 9.9(a) a 9.9(a). Pro větší scény, které tuto scénu obsahují, se chyba reprojekce zmenšuje.

Na obrázcích 9.10(a) a 9.10(b) jsou výsledky experimentu. To, že chyba pro nulový šum je v obou případech nulová, svědčí opravdu o tom, že použité konfigurace měřítek byly nedegenerované.

9.4 Experimenty s reálnými scénami

Experimenty s reálnými scénami byly provedeny na scéně „domek“.

Tato scéna měla 15 obrázků a 168 bodů. V JIM chybělo 82.26 % význačných bodů. Na obrázcích 9.11 a 9.12 jsou příklady dvou obrázků scény s rozlišením 2003 × 2952 pixelů. Menší plná kolečka značí známé korespondence, větší kolečka se světlým vnitřkem značí rekonstrukce neznámých bodů. Ty samozřejmě před rekonstrukcí neznáme, ale na obrázcích jsou ukázány proto, aby bylo zjevné jejich množství.

Obrázek 9.12: Příklad obrázku scény „domek“

9.4.1 Algoritmus pro chybějící políčka

Algoritmus 7.3 pro chybějící políčka je jediným algoritmem popsaným v této práci, který je schopen kompletně rekonstruovat perspektivní scénu s chybějícími políčky.

Experiment 8 *Tento experiment ukazuje robustnost algoritmu pro chybějící políčka na reálné scéně „domek“. Porovnávaly se všechny tři varianty Jacobsova algoritmu, který algoritmus pro chybějící políčka volá.*

Pro variantu s páry bylo výpočně únosné testovat všechny čtveřice, protože scéna měla jen 15 obrázků, proto se rekonstrukce opakovala pouze jednou. Ostatní varianty algoritmu se prováděly tak dlouho, dokud matice N z 6.2.3 neměla velikost alespoň 250-tinásobek počtu sloupců JIM nebo její transpozice. Kvůli zjištění standardní odchylky se tyto metody pouštěly stokrát.

Chyby reprojekce jednotlivých variant v % velikosti scény ukazuje tabulka 8. Je vidět, že nejlépe si vedla varianta s páry. Varianta s JIM na tom není o moc hůře, nemůžeme dokonce ani vyloučit, že by vyšla lépe, kdybychom použili všechny čtveřice. Varianta s transpozicí JIM má sice dobrý průměr, ale relativně velkou std. odchylku. Pokud by ji někdo měl používat, bylo by asi dobré nechat provést rekonstrukci touto metodou několikrát.

Varianta algoritmu	**Průměrná chyba v % vel. scény**	**Standardní odchylka v % vel. scény**
Čtveřice sloupců JIM – náh.	0.1452	0.0187
Páry obrázků – všechny	0.1380	0.0000
Čtv. sl. transp. JIM – náh.	1.8991	6.0030

Tabulka 9.1: Algoritmus pro chybějící políčka — různé varianty

Chyby v jednotlivých obrázcích v pixelech:

Obrázek	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Součet	230	321	292	274	225	174	231	278	202	264
Maximální	11.9	10.8	15.3	14.3	9.8	14.6	15.3	15.2	8.5	15.6
Medián	1.13	0.93	0.76	1.05	0.00	0.00	1.00	1.28	0.83	0.95
Průmer	1.37	1.91	1.74	1.63	1.34	1.04	1.38	1.66	1.21	1.58

Celkový součet chyby reprojekce: 2495 pixelů
Průměrná chyba reprojekce: 2.52 pixelů na bod

Tabulka 9.2: Algoritmus pro chybějící políčka — varianta s páry

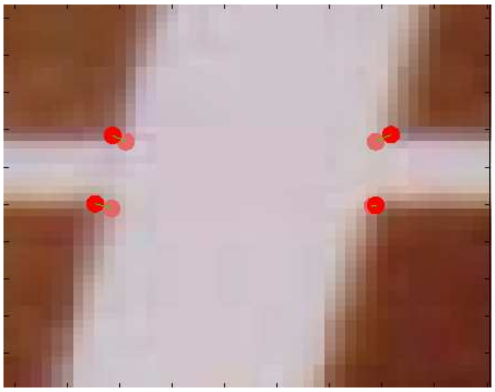
Chyby varianty s páry v jednotlivých obrázcích a celkové chyby této varianty jsou v tabulce 9.2. Detaily ukázek rekonstrukce nechybějících a chybějících bodů jsou na obrázcích 9.13(a) a 9.13(b).

Na obrázku 9.13(a) je vidět, jak blízko jsou restaurované body (nakreslené tmavě) původním (nakreslené světleji). Pravý dolní bod ze čtveřice má chybu reprojekce menší než 1 pixel.

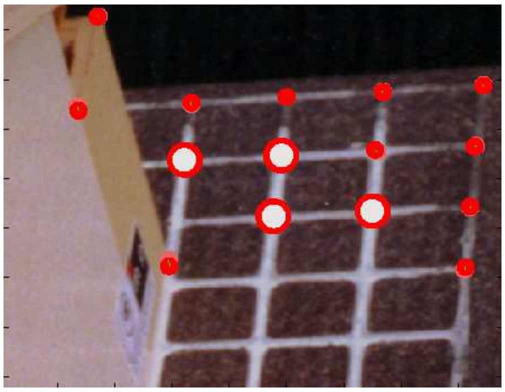
Obrázek 9.13(b) demonstruje, že se neznámé body (velké se světlým vnitřkem) rekonstrují smysluplně: na jiných obrázcích byly v tomto obrázku neznámé body umístěny v rozích dlažby chodníku, kde je také vidíme rekonstruované.

Můžeme říct, že průměrná chyba reprojekce 2.52 pixelů na bod je velmi dobrým výsledkem. Na druhou stranu je potřeba podotknout, že data scény „domek“ byla pořízena velmi pečlivě a v praxi se člověk setká spíše s horšími daty. Nicméně tento experiment otestoval algoritmus pro chybějící políčka dobře minimálně ve smyslu množství chybějících dat, neboť 82 % chybějících dat není málo.

Algoritmus bez použití síly obrázku z 7.3.1 měl v tomto experimentu zhruba desetinásobnou chybu.



(a) Nechybějící body: okno



(b) Chybějící body: chodník

Obrázek 9.13: Rekonstrukce scény „domek“ algoritmem pro chybějící políčka

Kapitola 10

Implementace

Popsané algoritmy byly implementovány v jazyku Matlab.

Programovací jazyk Matlab je určen zejména pro vytváření algoritmů náročných na výpočty. Obsahuje velkou podporu pro práci s maticemi a vektory a umožňuje snadnou vizualizaci výpočtů. Je též přenositelný mezi různými platformami.

Implementované algoritmy se ukázaly jako vhodné pro širší používání, proto se plánuje jejich integrace do systému RecX [13], který byl vytvořený v Centru strojového vnímání FEL ČVUT. Tento software je určen pro správu korespondencí virtuální kamery s možností následné projektivní rekonstrukce. Obsahuje podporu pro získávání a editaci význačných bodů v obrázcích, určování jejich korespondencí a kromě toho také sadu algoritmů pro rekonstrukci.

Předpokládá se, že rozhraní funkcí a používání globálních proměnných bude muset být ještě změněno, aby odpovídalo standardům systému RecX, jehož budou součástí. Přesné rozhraní funkcí zde proto nebude uvedeno.

10.1 Implementované nástroje

Kromě implementací samotných algoritmů byla vyvinuta podpora pro jejich spouštění a testování.

Všechny popsané algoritmy používají informace o scéně obsažené v několika maticích. Tyto matice se liší především podle typu kamery, kterou byla scéna snímána, ale také např. tím, zda je potřeba udržovat informace o chybějících políčkách matic. Navzdory různorodosti dat o scénách se s těmito maticemi zachází jednotným způsobem.

Nástroje pro generování scén, pro jejich vizualizaci s podporou doplňujících a ladících informací a nástroje pro počítání chyb mají také jednotné používání. Nechybí ani nástroje k různorodému a mnohonásobnému testování algoritmů s ukládáním mezivýsledků na disk.

Experimenty s reálnými scénami byly prováděny s podporou systému RecX, neboť v něm byly vytvořeny korespondence bodů pro scénu „domek“.

Kapitola 11

Závěr

V této práci byly zkoumány souvislosti mezi k-ticemi neúplných sloupců matice scény a hledanou kompletní maticí scény. Ukázalo se, že Jacobsovu myšlenku přechodu ke komplementu lze použít i pro scénu s perspektivní kamerou a že se lze s její pomocí obejít bez znalosti některých měřítek.

Pomocí nových poznatků je možné rekonstruovat v lineárním čase scény s perspektivní kamerou a s velmi malým počtem bodů a známých měřítek.

11.1 Otevřené problémy

Studovaná látka ukazuje i směr dalšího bádání.

Není ujasněn pojem „nedegenerovaná konfigurace význačných bodů“ ze sekce 8.2, poznámka k větě 3. Byly sice ukázány příklady takových konfigurací (viz experiment 7), ale pro úplné pochopení problému to je málo.

S tím souvisí i otázka, zda je možné vypočítat velmi malý počet (2 nebo 3) společných měřítek v různých obrázcích. Pokud to lze, bude se jednat o metodu dosud neznámou, protože klasické lineární algorimy potřebují minimálně 6 společných bodů (viz [4]).

Otázka praktického použití také není zcela zodpovězena. Pokud neexistuje metoda pro výpočet měřítek z klasicé JIM, je možné se ptát, jestli v praxi existují problémy, v nichž lze předpokládat nějaká další omezení, která by bylo možné k rekonstrukci využít.

Ztraceným se zatím nejeví ani nápad navrhnout metodu pro měření některých měřítek přímo ve scéně, opět pro úzkou třídu praktických úloh.

11.2 Algoritmus pro chybějící políčka a měřítka

Nicméně alespoň nějaké praktické využití lze vidět už teď. Jedná se o spojení algoritmu pro chybějící body s algoritmem s chybějícími měřítky. Toto spojení bylo sice už naznačeno na konci 8.1.2, ale doposud nebyl popsán jeho přínos pro rekonstrukci, ani nebylo implementováno. Zde je ukázka přínosu tohoto řešení.

V následující JIM s měřítky nechť místa s λ odpovídají políčkům JIM se známými měřítky, místa s 1 políčkům se známými body, ale neznámými měřítky, a místa s 0 políčkům, kde neznáme vůbec nic.

$$\begin{pmatrix} \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & 0 & 0 & 0 & 0 & 0 & 1 & 1 & \dots & 1 \\ \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & 1 & 1 & \dots & 1 \end{pmatrix}$$

Měřítka v místech s λ dokáže vypočítat algoritmus pro chybějící políčka, když za centrální obrázek zvolíme druhý obrázek, protože osmibodový algoritmus potřebuje 8 společných bodů mezi obrázky. Algoritmus doplní políčka v pouze v nejlevější podmatici 9×13.

Ovšem místa s 1 jsou obnovitelná pomocí algoritmu s chybějícími měřítky. Vyberme sloupce 6 až 8, dva sloupce s jedničkami a řádky prního a třetího obrázku. Dostaneme pětici

$$\begin{pmatrix} \lambda & \lambda & \lambda & 1 & 1 \\ \lambda & \lambda & \lambda & 1 & 1 \end{pmatrix},$$

která je celá rekonstruovatelná podle věty 3 ze sekce 8.2. Následně jsou obnovitelné i zbývající místa s 0. Popsané kroky však bude zobecněný algoritmus provádět najednou.

Poznamenejme, že lineární algoritmus na kompletní rekonstrukci výše popsané scény dosud nebyl publikován.

Literatura

[1] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, 1989.

[2] R. Hartley. In defence of the 8-point algorithm. In *5th ICCV*, pages 1064–1070, Cambridge, Massachusetts, June 1995.

[3] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2000.

[4] A. Heyden. A common framework for multiple view tensors. In *ECCV*, 1998.

[5] A. Heyden. Tutorial: Multiple view geometry. In *ICCV*, September 1999.

[6] Pui-Kuen Ho and Ronald Chung. Stereo-motion with stereo and motion in complement. In *IEEE TPAMI*, volume 22, pages 215–220, February 2000.

[7] D. Jacobs. Linear fitting with missing data: Applications to structure from motion and to characterizing intensity images. In *CVPR*, pages 206–212, 1997.

[8] S. Mahamud and M. Hebert. Iterative projective reconstruction from multiple views. In *CVPR*, 2000.

[9] Pavel Pták. Některé matematické struktury používané v kybernetice, duben 1999.

[10] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *ECCV96(II)*, pages 709–720, 1996.

[11] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. In *IJCV(9), No. 2*, pages 137–154, November 1992.

[12] M. Urban, T. Pajdla, and V. Hlaváč. Projective reconstruction from multiple views. In *Technical report CTU-CMP-1999-5*. CMP, FEL ČVUT, Karlovo náměstí 13, Praha, Czech Republic, December 1999.

[13] T. Werner, T. Pajdla, and M. Urban. Rec3d: Toolbox for 3d reconstruction from uncalibrated 2d views. In *Technical report CTU-CMP-1999-4*. CMP, FEL ČVUT, Karlovo náměstí 13, Praha, Czech Republic, December 1999.

Příloha A

Dodatky

A.1 Faktorizace

Věta 5 (Faktorizace) *Mějme matici J velikosti $k \times n$.*

1. *Za předpokladu $k \geq n$ může být J rozložena pomocí svd*[1] *na matici U velikosti $k \times n$, diagonální matici velikosti $n \times n$ a matici V velikosti $n \times n$:*

$$J = U \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & \\ 0 & \dots & & \lambda_n \end{pmatrix} V^T,$$

   kde $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k \geq 0$ jsou vlastní čísla matice J a $U^T U = V^T V = V V^T = I$, kde I je jednotková matice $n \times n$.

   Pokud předpoklad $k \geq n$ není splněn, rozklad lze provést na transpozici J.

2. *Nejlepší aproximace J maticí Q hodnosti r $(n > r \geq 1)$ ve smyslu nejmenšího součtu čtverců vzdálenosti*[2] *je*

$$Q = U \begin{pmatrix} \lambda_1 & 0 & \dots & & & 0 \\ 0 & \ddots & & & & \\ \vdots & & \lambda_r & & & \vdots \\ & & & 0 & & \\ & & & & \ddots & \\ 0 & & \dots & & & 0 \end{pmatrix} V^T,$$

[1] singular value decomposition, rozklad matice pomocí vlastních čísel a vektorů

[2] tj. Frobeniova norma $||Q - J||$ je minimální

upraveno

$$Q = U'\Lambda'V'^T,$$

kde U' je prních r sloupců matice U, V' je prních r sloupců matice V a

$$\Lambda' = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & \\ 0 & \dots & & \lambda_r \end{pmatrix}$$

Poznámka: *Tato aproximace poskytuje dostatečnou informaci jen tehdy, pokud je poměr mezi r-tým a $(r+1)$-ím největším vlastním číslem J dostatečně velký.*

Důkaz: Viz [1]. □

A.2 Algoritmus Ho

Tento algoritmus [6] řeší všechny kroky rekonstrukce scény z 2.1. Hledá tedy význačné body, přiřazuje je k sobě a odhaduje 3D model scény. Algoritmus využívá také speciální hardware. Jsou to dvě kamery připevněné vedle sebe a nasměrované stejným směrem, který je kolmý na spojnici kamer. Scéna se snímá z relativně velké vzdálenosti, takže lze předpokládat afinní model (ortografická kamera).

Postup algoritmu je následující:

1. Spojenými kamerami se nasnímá pár sekvencí n snímků.

2. Naleznou se význačné body pro každou sekvenci zvlášť.

   K tomu je potřeba, aby byly sekvence dostatečně husté, aby rozdíly mezi jednotlivými snímky byly relativně malé.

   Vzniknou tak dvě JIM J a J', které se kromě malého posunu v souřadnicích kvůli různému umístění kamer liší v tom, že mají nejspíše nestejný počet sloupců (nalezených význačných bodů) a sloupce jsou v jiném pořadí.

   Nyní by bylo možné faktorizovat J a J' zvlášť, ovšem pouze pokud by byly správně určeny význačné body. Proto se použijí stereo-korespondence v následujícím kroku.

3. Určení stereo korespondencí je ekvivalentní správnému přiřazení sloupců J se sloupci J'. A to není těžký problém.

   Autor algoritmu použil epipolární geometrii páru kamer, kterou zjistil po jeho konstrukci. Pomocí ní a párů nasnímaných obrázků nalezl jednoznačné stereo korespondence, jejichž faktorizací sestavil báze pro J a J'. Ostatní sloupce JIM dopočítal z bází.

Algoritmus pracuje v lineárním čase vzhledem k velikosti nasnímaných dat.