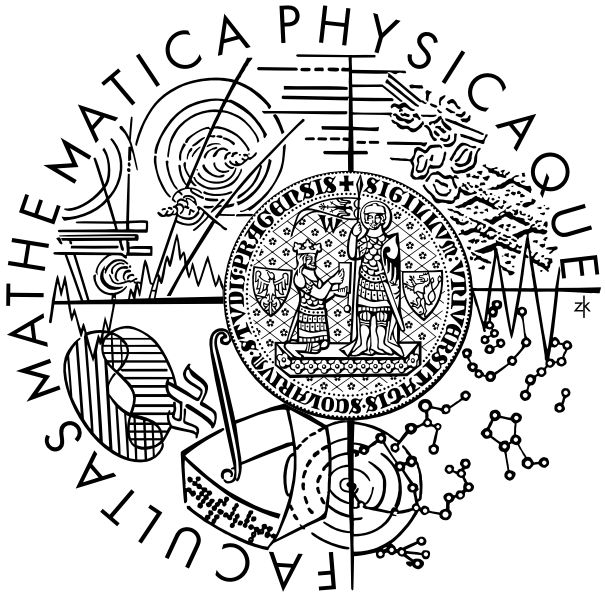
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Martin Zátopek

Optické snímání pohybu

Kabinet software a výuky informatiky

Vedoucí diplomové práce: RNDr. Josef Pelikán

Studijní program: Informatika

Děkuji RNDr. Josefu Pelikánovi za cenné připomínky a rady při tvorbě této práce.

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

V Praze dne 16. prosince 2005

Martin Zátopek

Obsah

Název práce: Optické snímání pohybu
Autor: Martin Zátopek
Katedra: Katedra software a výuky informatiky
Vedoucí diplomové práce: RNDr. Josef Pelikán
e-mail vedoucího: josef.pelikan@mff.cuni.cz

Abstrakt: Práce se zabývá optickým snímáním pohybu pomocí levných webkamer. Mapuje možnosti připojení více kamer k jednomu počítači a čtení obrazových dat pomocí rozhraní Video4Linux pro operační systém Linux. Dále se podrobně zabývá algoritmy kalibrace kamer, Direct linear transformation a Tsai kalibrací. Navrhuje a implementuje systém optického snímání pohybu, ke kalibraci využívá externí implementaci Tsai kalibrace Rega Willsona [5]. Součástí systému je popis kalibrační scény, postup poloautomatického získávání kalibračních dat, synchronizace a ukládání dat z více kamer připojených k počítačům umístěných v lokální síti. Následný postprocessing, rekonstruující 3D souřadnice značek, ukládá naměřená data do standardního translačního formátu trc.
Klíčová slova: optické snímání pohybu, USB web-kamera, Video4Linux, Direct linear transformation, Tsai kalibrace

Title: Optical motion capture
Author: Martin Zátopek
Department: Department of Software and Computer Science Education
Supervisor: RNDr. Josef Pelikán
Supervisor's e-mail address: josef.pelikan@mff.cuni.cz

Abstract: Main thema of the thesis lies in optical motion capture algorithms and their usage with cheap webcameras. Multiple cameras connected to a single computer are taken into account, while using Video4Linux on the Linux operating system for reading image data. Two camera calibration algorithms are inspected in detail: Direct Linear Transformation and Tsai calibration. Practical part of the work deals with design and implementation of an optical motion capture system, using Reg Willson's implementation [5] of Tsai calibration. Documentation for the system contains description of calibration scene, semiautomatical procedure of retrieving calibration data, synchronization and storing data from multiple devices available on a local network. Final reconstruction of marker's 3D coordinates uses standard translational format TRC.
Keywords: optical motion capture, USB web-camera, Video4Linux, Direct linear transformation, Tsai calibration

Kapitola 1

Úvod

Motorem lidského bádání vždy byla především snaha o usnadnění nebo zkvalitnění naší práce. Nejinak tomu bylo i u počátečních pokusů snímání pohybu. Fyziologové chtěli zkoumat odlišnosti v pohybových návycích lidí, animátoři by rádi vytvořili dokonalý pohyb svých postaviček, sportovci chtějí vědět, jak odhodit oštěp, aby letěl při dané síle co nejdále nebo jak pohybovat končetinami, aby rotace krasobruslaře byla co nejefektivnější. Všechny tyto problémy mají jednoho společného jmenovatele - pohyb. Pokud chceme cokoliv zkoumat, je výhodné mít přesný popis zkoumaného jevu; nejinak tomu je u pohybu.

Touto problematikou se zabývaly celé výzkumné týmy již od 70. let minulého století. V dnešní době se jedná v základním zadání snímání pohybu už o poměrně dobře zvládnutý problém, ovšem jen na úrovni profesionálního využití. Což je hlavní důvod nedostatku informací o dané problematice. Cílem této práce byla především snaha poodkrýt poměrně málo zdokumentovanou problematiku, vytvořit snad první ucelenější česky psaný dokument o snímání pohybu. A to vše podpořit implementací jednoduchého systému optického snímání pohybu provozuschopného v domácích podmínkách za využití běžně dostupného programového i hardwarového vybavení. Zadání práce dokonce omezovalo vstupní zařízení jen na využití levných webových kamer.

Mým cílem tedy nebylo objevit něco neznámého nebo vyvinout systém konkurující komerčním produktům, ale spíše proniknout do problematiky, jejíž know-how si firmy tak pečlivě střeží a ukázat, že se nejedná o nic složitého.

V úvodních kapitolách jsem se věnoval obecné definici snímání pohybu, jeho historii a základní typologii. Ve druhé části jsem popsal mé zkušenosti s připojováním USB kamer k počítači a způsobu získávání obrazových dat pomocí Video4Linux rozhraní. Ve třetí kapitole jsem se věnoval popisu obecně známých postupů kalibrace kamery, jeden pro jeho názornost (Direct linear transformation) a druhý pro jeho časté používání v praxi (Tsai kalibrace). Následuje popis mnou navrhované kalibrační scény a postupů při hledání významných kalibračních bodů. Ve čtvrté části je podrobný popis implementovaných algoritmů a návrh struktury vlastního systému optického snímání pohybu. V závěrečné části je popis dvou nejznámějších datových formátů optického snímání pohybu.

1.1 Co je to snímání pohybu - Motion Capture?

Motion capture (MC) je technologie pro digitální snímání pohybu objektů, obvykle lidí, případně zvířat. Původně bylo snímání pohybu vyvíjeno jako pomůcka ve výzkumu biomechaniky. Ale velmi brzy byl objeven jeho potenciál i odborníky v jiných oborech, především v počítačové animaci - našel tedy uplatnění v oblastech jako kinematografie a vývoj počítačových her.

Myšlenka samotného snímání pohybu je velmi jednoduchá. Obecný postup začíná v rozmístění značek na snímaný objekt, vytvoření prostředí, ve kterém se objekt může pohybovat, a zajištění snímání pozice případně orientace značek na pohybujícím se objektu. Takto vzniklá data se počítačově zpracovávají buď v reálném čase nebo jako postprocessing. Program má za úkol tato data správně interpretovat a uložit ve vhodném formátu, který obsahuje už jen pozice značek v prostoru.

Obrázek 1.1: Ukázka možného rozmístění značek na těle herce[1].

Nyní mohou animátoři aplikovat tato data na připravenou takzvanou kostru, kterou vybaví příslušným vzhledem. A my se pak jako konzumenti můžeme už jen obdivovat reálným a přirozeným pohybům robota v našem oblíbeném filmu nebo počítačové hře.

Pokud bych měl výše popsaný postup více zkonkretizovat, tak asi na příkladu optického snímání pohybu, kterým se podrobněji zabývám i v dalším textu. Značky mohou být v tomto případě aktivní (například emitující světlo) nebo pasivní (reflexní ping-pongové míčky). Tyto body se rozmístí tak, aby jejich zachycený pohyb v prostoru umožňoval rekonstrukci pohybu herce (loket, rameno, koleno, hlava atp., viz obrázek 1.1). Scéna s hercem se musí snímat nejméně dvěma kamerami, doporučuje se však alespoň osm kamer s vysokým framerate (právě tyto podmínky se budu snažit ve své práci eliminovat). Zpracování dat z optického MC probíhá jako postprocessing. V jednotlivých záběrech kamery se naleznou značky ze známé polohy kamer, identifikují se a rekonstruuje se jejich pozice v prostoru. Následně se tato data mapují na příslušnou kostru a dále vyžívají pro animátorské účely. Příkladem další technologie je magnetické snímání pohybu (přesněji elektromagnetický MC). Postup je obdobný, značky přímo vysílají svoji polohu a navíc i orientaci v prostoru do centrální řídící jednotky; s naměřeným daty se už pracuje velmi podobně. Magnetický MC umožňuje navíc realtime snímání pohybu.

V každém případě nejsou animátoři i při využití dat z MC bez práce, jak by se možná na první pohled mohlo zdát. Protože data jsou často zatížena chybami způsobené technikou snímání. Například u optického snímání pohybu se velmi často stává, že značka umístěná na hercově těle není zachycena na žádné nebo jen na jedné z kamer, může také dojít k chybné identifikaci markerů na jednotlivých záběrech. To všechno musí animátoři opravit a především vyřešit interakci scény s modelem. Ani samotné mapování markerů na kostru není bezproblémové. Model má pevné délky jednotlivých kostí, ale při snímání pohybu žádné takové předpoklady nejsou dodrženy. Dalším z problémů je nepřesnost v umístění jednotlivých značek. Nelze je umístit přesně do středu kolene nebo loktu, ale jen na povrch lidského těla. To jsou ale většinou nepřesnosti, které řeší sám animační software. Ani já se řešením těchto problémů postprocessingu v tomto textu nebudu podrobněji

zabývat.

Výhody MC jsou především v úspoře času animátorů. Animace postav, která dříve zabrala i několik měsíců, se dnes odehraje během jednoho dne práce s hercem a pár týdny čištění naměřených dat. A především výsledná animace dosahuje mnohem vyšších kvalit. Velmi snadno se takto zachycují pohyby, které se fyzikálně jen těžko popisují tak, aby mohly být nasimulovány. Například animace postavy, která skáče salto vzad a v rukách drží nunčaky, může být bez MC pro animátory nepřekonatelný problém.

Výhody MC se mohou snadno stát i jeho nevýhodami. Přirozenost pohybu může být v některých situacích i nežádoucí. Zároveň je problém ve snímání pouze reálného pohybu, herec nemůže předvádět létání v korunách stromů, ani nadlidskou sílu při boji se svým hereckým partnerem. Další neméně závažný problém je v manipulaci s již jednou naměřenými daty. Pokud animátor zjistí v období postprocessingu, že se mu daný záběr příliš nehodí, jen těžko ho může nějak výrazně upravit. Většinou se musí přistoupit k opakování celého procesu MC. Kromě animátora se na celkovém dojmu z výsledku MC podílí nemalou měrou i model se svými hereckými schopnostmi.

1.2 Historie [8]

Snímání pohybu je relativně mladá disciplína, její historie začíná na konci sedmdesátých let a bouřlivý rozvoj zaznamenala až v poslední dekádě minulého století. Ale myšlenka kopírovat pohyb člověka v animaci je mnohem starší. Už v roce 1915 Max Fleischer experimentuje s procesem, který se později začne označovat jako rotoscoping. Jedná se o jednoduché obkreslování políčka po políčku filmu se zaznamenaným pohybem člověka. V roce 1937 rotoscoping použili ve Walt Disney Studios ve filmu Snow White. Obdobné přístupy později převzala i ostatní studia, přestože to mnoho animátorů považovalo za podvod a ubíjení své tvůrčí práce.

Na začátku osmdesátých let laboratoře zkoumající biomechaniku začaly používat počítače k analýze lidského pohybu. Zde používaná zařízení a techniky se velmi brzy začaly využívat i v oboru počítačové grafiky. Například Tom Calvert, profesor kinesiologie a informatiky na Simon Fraser University, použil sadu potenciometrů připnutých na lidském těle ke studiu choreografie a abnormalit v jejím pohybu.

Velmi brzy se začínají používat komerční systémy optického snímání pohybu jako systémy Op-Eye a SelSpot. Tato technologie pracovala už na velmi podobném systému jako dnešní optické MC, omezení byla především v nízkém rozlišení a framerate tehdy dostupných kamer. Povětšinou bylo potřeba manuálně provádět postprocessing v případě, že se marker ztratil z dohledu kamer. Nízké rozlišení kamer se nahrazovalo zmenšováním zaznamenávané scény, kamery se jednoduše umístily blíže centru dění.

V roce 1988 deGraf/Wahrman vyvinuli "Mike the Talking Head" pro Sillicon Graphics, aby ukázali možnosti svého systému pro realtime animaci obličeje. Mike byl ovládán speciálním zařízením, které umožňovalo kontrolovat výraz úst, očí, pozice hlavy a celkového výrazu obličeje. Živým představením na SIGGRAPHu toho roku jednoznačně demonstroval připravenost systémů obdobného typu pro nasazení do produkční sféry.

Již v roce 1985 se pokoušeli v Jim Henson Productions sestrojit virtuální postavu pro svoji show, ale s minimálními úspěchy. Až v roce 1988 se jim podařilo využít systému Mike a zkonstruovat první v reálném čase ovládanou virtuální postavu Waldo.

O rok později Kleiser-Walczak vytvořili Dozo, počítačovou animaci tancující a zpívající ženy za mikrofonem. Pro realistický pohyb ženy se rozhodli použít optický motion capture systém založený na snímání reflexních značek umístěných na těle herce. Výše popsané problémy té doby zapříčinily dlouhou dobu nutnou pro postprocessing, ale výsledky byly na tu dobu ohromující.

Úspěch systému Waldo získal nové prostředky pro počítačovou animaci. Například společnosti Videosystem a French video v roce 1991 stály za vznikem dalšího realtime systému na animaci postav, na jehož základě vznikl pořad Mat the Ghost. V pořadu vystupovali jak živí herci, tak virtuální zelený duch, ovládaný datovou rukavicí, joystickem a sadou pedálů. Výroba jednoho sedmiminutového dílu trvala necelé dva dny.

Dalším významným krokem ve vývoji MC byl takzvaný "face waldo" (SimGraphics, 1992), systém realtime sledování výrazu obličeje snímaného pomocí mechanických senzorů připnutých přímo na obličeji herce. Velký ohlas vzbudila tato technologie především díky živému rozhovoru postavičky Maria ze světoznámé hry firmy Nintendo. Mario v reálném čase vtipkoval a reagoval na dotazy diváků.

Za posledních několik málo let udělal MC velký krok kupředu především díky technologickému pokroku ve vývoji hardwarových prostředků, kamery s vyšším rozlišením a framerate, počítače s rostoucí výkonností. Ale základní myšlenka se zatím příliš nemění. Probíhají výzkumy v optickém snímání pohybu bez využití pomocných značek rozmístěných na modelu. Další nemalou výzvou je recyklace již jednou naměřených dat vytvořením jakéhosi slovníku mnoha pohybů a z něho následně skládat nové, složitější pohyby, které by si zároveň zachovaly přirozenost své předlohy.

1.3 Typologie snímání pohybu

Z obecnosti problému snímání pohybu je zřejmé, že k němu lze přistupovat různými způsoby. Během let se objevila celá řada způsobů vhodných k zachycení pohybu. Každý z nich měl svá pro i proti, své silné a slabé stránky. V krátkosti se zde zmíním o těch nejvýznamnějších.

1.3.1 Elektromechanické snímání pohybu

Pohyb herce je snímán řadou mechanických zařízení, většinou potenciometrů měřících úhel v hlavních kloubních spojení člověka. Hlavní nevýhoda tohoto přístupu spočívá v nemožnosti měření globálních posunů. Pozice jednotlivých končetin je možné dopočítat z naměřených úhlů, ale pozici celého modelu jednoduše určit nelze. Nejčastějším řešením je doplnění snímacího modelu ještě o elektromagnetický senzor určující právě posun. S tím ovšem získává tato metoda další nevýhodu typickou pro elektromagnetický přístup, a to citlivost na přítomnost kovových předmětů. Další nezanedbatelná nevýhoda je ukryta v předpokladu, že vzájemnou polohu dvou kostí lze určit jedním úhlem, což například u ramenního kloubu může způsobit velké zkreslení. Pro herce není samozřejmě také nic příjemného být svázán v mechanickém korzetu nutným pro elektromechanické snímání pohybu. Zjevnou výhodou je možnost realtime použití naměřených dat, v podstatě neomezený prostor pro snímání pohybu a nehrozí zde ani nějaké překrývání značek, tolik typický problém optického MC.

1.3.2 Elektromagnetické (/magnetické) snímání pohybu

Pohyb je v tomto případě snímán pomocí vysílačů nízkofrekvenčního elektromagnetického pole vhodně rozmístěných na těle herce. Scéna je obklopena několika přijímači, které měří střed a orientaci jednotlivých vysílačů. Oproti většině jiných přístupů, kde lze polohu značek měřit pouze ve třech stupních volnosti, zde získáváme stupňů hned šest (pozice a orientace), proto se může snížit počet značek rozmístěných na těle herce. Zároveň žádná ze značek není při pohybu nikdy zakrytá, jako se to velmi často stává například u optického MC. Další výhodou je možnost realtime získávání dat. Na druhou stranu se tento přístup snímání pohybu potýká s problémy rušení elektromagnetického pole různými kovovými předměty poblíž scény, kolem nichž vzniká naindukované pole, jehož interference s měřenými póly vysílačů lze jen obtížně odstranit.

1.3.3 Optické snímání pohybu

Jedním z nejvýznamnějších způsobů snímání pohybu posledních let je bezesporu optický přístup. Herec má na svém těle rozmístěné buď pasivní reflexní značky, nebo aktivní diody emitující světlo. Scéna je snímána několika kamerami rozmístěnými kolem herce. Světelné prostředí musí být přizpůsobeno typu snímání pohybu, proto zde nesmí být žádné ostré světelné zdroje tak, aby se co nejsnadněji oddělovaly reflexní značky od pozadí nasnímaného obrazu. Nalezené pozice značek v kombinaci se znalostí o přesném umístění a parametrech kamer nám už stačí k určení přesné

polohy značek v prostoru. Tato metoda se považuje za nejpřesnější, a to díky kvalitě dnes dostupných kamer. Jde především o rostoucí rozlišení a framerate, tolik důležitý pro snímání rychlých pohybů. Optické snímání pohybu zatím nelze plně nasadit v realtime aplikacích kvůli dlouhé době postprocessingu, který se samozřejmě prodlužuje s rostoucím rozlišením a framerate. Další nepříjemností, se kterou se musí animátoři potýkat, jsou datové výpadky ve sledování jednotlivých značek zapříčiněné občasným zakrytím značky buď hercem nebo samotnou scénou. V takovém případě je nutný zásah animátora, případně softwaru, který odhadne pohyb značky v místech, kde o něm nemá žádné informace. Z popsaného je zřejmé, že optické snímání pohybu neměří rotace jednotlivých končetin, ale jen absolutní pozice, ze kterých lze zpětně rotace dopočítat. Výhody tohoto přístupu jsou především ve velké přesnosti naměřených dat, volnosti pohybu herce a v podstatě neomezeném počtu značek, není tedy problém mít na scéně i více herců.

Podrobnosti o tomto přístupu naleznete v dalším textu, kde se zabývám právě optickým snímáním pohybu za velmi špatných vstupních podmínek, nízkého rozlišení a framerate.

1.4 Jak si udělat vlastní optické snímaní pohybu

Vlastní proces optického snímání pohybu lze rozdělit do několika fází, které zde v krátkosti popíši a podrobněji o nich budu mluvit v následujících kapitolách. Tam popíši nejen přesné algoritmy, ale především problémy, se kterými jsem se setkal, a návrhy jejich řešení.

Pokud vynechám spíše marketingovou fázi rozhodování, zda a jaký způsob snímání pohybu zvolit, je první přípravnou fází výběr snímaného modelu a především vhodné rozmístění značek na jeho povrchu tak, abychom byli z pozice značek v prostoru schopni co nejpřesněji rekonstruovat pohyb. Pro přesné určení rotace končetin je vhodné například značky na lokti zdvojit tak, abychom byli schopni při rekonstrukci určit přesnou polohu loketního kloubu jako střed úsečky určené právě těmito dvěma značkami.

Další stále ještě inicializační fází je rozmístění dostatečného množství kamer kolem scény. K rekonstrukci polohy značky v prostoru je nutné mít tuto značku zachycenou alespoň na dvou kamerách. Ale z důvodů častého zakrývání značek samotným modelem je nutné umístit do scény více kamer tak, abychom tuto podmínku zachovali po co nejdelší možnou dobu. Vhodný počet kamer se pohybuje od čtyř do dvanácti, podle druhu snímané scény. Pokud je značka identifikována v obraze více kamer, redundantnost informace se využívá ke zlepšení přesnosti. Přesnost určení polohy značky také závisí na vzájemné poloze kamer a snímané značky. Ideální případ nastává ve chvíli, kdy značka a dvě kamery svírají pravý úhel. Naopak pokud kamery svírají nulový úhel nebo je jejich umístění kolineární se značkou, pak nelze určit její umístění v prostoru. Proto je třeba dbát nejen na dostatek kamer snímajících scénu, ale i na jejich vhodné rozmístění.

Po rozmístění kamer je nutná jejich kalibrace - zjištění jejich přesné polohy a orientace v prostoru a zároveň je potřeba určit jejich optické vlastnosti, ohniskovou vzdálenost, zkreslení atp. To se nejčastěji děje poloautomatickým procesem, kdy se do scény umístí předmět známých rozměrů, jehož významné body se snadno identifikují v obrazu jednotlivých kamer. Ze znalosti polohy takto nalezených bodů a jejich obrazu v kameře lze dopočítat relativní pozici vůči sledovanému předmětu. Pokud ten prohlásíme za střed souřadného systému, můžeme dopočítat všechny potřebné parametry kamery. Jedná se o inverzní proces k určování polohy značek, který bude prováděn později v rámci postprocessingu.

Příprava samotné scény vyžaduje pouze nastavení vhodných světelných podmínek, vhodných pro fázi postprocessingu tak, aby šlo například pouhým prahováním oddělit pozadí scény od značek umístěných na těle modelu.

Po připojení kamer k jednomu centrálnímu záznamovému zařízení, například k počítači, je možné nasnímat pohyb modelu. Je třeba zajistit synchronizaci všech video vstupů pro pozdější zpracování. V této fázi se pouze nahrají videozáznamy pohybu ze všech kamer.

Následující část zpracování naměřených dat se označuje jako postprocessing. Hlavním úkolem je na základě nahraných videí identifikovat všechny značky a určit jejich polohu v čase. V první fázi je potřeba v jednotlivých políčkách záznamu oddělit značky od pozadí scény i samotného modelu. Protože se za značky často volí velmi reflexní body, stačí pouhým prahování odhalit "přepaly"

v záznamu. Jejich střed, definovaný například geometricky nebo nějakým korozním algoritmem, se považuje za 2D souřadnice značky v rámci jednoho filmového záznamu. Podobným způsobem se zpracují záznamy ze všech kamer z určitého časového okamžiku. Nyní stačí uvažovat dírkový model kamery, upravený o případné optické zkreslení a snažit se identifikovat jednotlivé značky v záběrech různých kamer. Pokud se podaří nalézt alespoň dvě kamery, které zachytily stejnou značku. Ze znalosti o umístění a orientace kamer lze už poměrně snadno určit polohu značky v prostoru. Tímto způsobem se určí poloha všech značek, u kterých to naměřená data umožní. Poslední problém nastává s identifikací jednotlivých značek v čase. Tam se předpokládá, že značky v časových intervalech zaznamenaných kamerami mění polohu jen v rámci malé translace. Za tohoto předpokladu lze programově odhadovat identifikaci jednotlivých značek v čase. Bohužel se ani při maximální snaze v reálném nasazení neubráníme zakrytí značek alespoň na krátkou dobu. V takovém případě je většinou nutný zásah operátora postprocessingu, který ručně přiřadí zaniklou značku k nově vzniklé, případně odhadne její průběh v čase, kdy byla zakryta.

Kapitola 2

Připojení levné USB kamery k počítači

V následujícím textu bude shrnutí mých zkušeností s instalací a používáním USB kamer. Konkrétně v mém případě jsem měl k dispozici webovou kameru Creative WebCam PRO[1]. Výběr kamery byl determinován především její dostupností na KSVI (Kabinet software a výuky informatiky), tímto děkuji za její zapůjčení panu RNDr. Josefu Pelikánovi. Jedná se o jeden z nejlevnějších modelů této značky, čímž byla splněna podmínka zadání diplomové práce na využití levných webových kamer.

2.1 Instalace

Protože jsem ve svém zadání nebyl omezen na konkrétní operační systém, první pokus o instalaci kamery proběhl na počítači s operačním systémem Windows XP. Jak už návod napovídá, ovladače pro jiné operační systémy se nedodávají, což jsem si ověřil i přímo na webových stránkách výrobce.

Samotná instalace probíhala velmi přímočaře přesně podle přiložené instalační příručky, na žádné problémy při instalaci jsem tedy nenarazil. Na instalačním disku je sada programů umožňující okamžité použití kamery, jako nahrávání videa, pořizování jednotlivých obrázků a program pro správu takto pořízených dokumentů. Kamera se k počítači připojuje dnes již standardním rozhraním USB verze 1.1 a je schopna snímat video v rozlišení až 640x480 při fps 13, případně 320x240 při fps 25.

Problémy nastaly až ve chvíli, kdy jsem se rozhodl hlouběji proniknout do programování a tedy využití dodávaných ovladačů pro potřeby mé diplomové práce. Při zjišťování možností snímání dat z videozařízení jsem narazil na zásadní problém spojený s implementací originálních ovladačů, které neumožňují současné připojení více kamer téhož typu k jednomu počítači. Nezbývalo než získat jiné ovladače, které umožní testování i v mých domácích podmínkách.

Uspěl jsem až u operačního systému Linux, pro který existují dokonce dva ovladače schopné obsluhovat moji kameru, jedná se o OV511 (`http://alpha.dyndns.org/ov511/`) a PWC (`http://www.smcc.demon.nl/webcam/`). Po prostudování vlastností a výhod jednotlivých ovladačů, jsem se rozhodl pro OV511, který byl ve svém vývoji o něco dále a díky tomu mnohem více využíván. Výběr distribuce byl ovlivněn především mými znalostmi operačních systémů typu UNIX, zvolil jsem distribuci SUSE Linux (`www.suse.com`), a to především pro jeho uživatelskou vstřícnost.

I v tomto případě byla instalace velmi snadná, protože SuSE samo detekovalo zařízení připojené na USB portu jako webkameru a zavedlo do jádra systému příslušné moduly plně automaticky, konkrétně se jedná právě o výše zmíněný ovladač OV511 a videodev modul. Pak již stačilo nainstalovat některý z řady programů dostupných často pod GNU licencí, které jsou většinou určeny

[1] Tato kamera není již normálně dostupná, proto již neexistují její oficiální stránky. Asi nejblíže z dnes ještě prodávaných modelů k ní má Creative WebCam PRO Ex `http://creative.com/products/product.asp?category=5&subcategory=32&product=243`

především pro televizní karty a tedy příjem několikakanálového obrazu. Asi nejznámější a nejrozšířenější je xawtv (`http://linux.bytesex.org/xawtv/`). Řadu dalších užitečných aplikací lze nalézt přímo na stránkách ovladače OV511 `http://alpha.dyndns.org/ov511/apps.html` nebo na `http://www.thedirks.org/v4l2/peopleprojects.htm`.

Pokud by distribuce sama nezavedla modul kamery do jádra, bylo by nutné ho zavést ručně pomocí příkazu *modprobe ov*511, který nahraje příslušný ovladač kamery a zároveň zkontroluje závislosti na dalších modulech. Úspěšné nahrání modulu lze ověřit příkazem *lsmod*, kterým se vypíší všechny dynamicky nahrané moduly jádra. Problém může nastat v případě, že jádro u sebe nemá tento modul zkompilovaný. V takovém případě je nutné překompilovat jádro i s ovladači kamery[2]. K většině těchto kroků budete potřebovat administrátorský přístup k operačnímu systému.

2.2 Použití

Po úspěšném zprovoznění kamery pod operačním systémem Linux pomocí ovladačů se známým API rozhraním jsem narazil na jiný, neméně závažný problém, který se mi ve svém důsledku nakonec ani vyřešit nepodařilo. Nedostatečný výkon kamery dosažený v prostředí Linux bohužel ani zdaleka neodpovídá výsledkům originálních ovladačů pod Windows (při rozlišení 320x240 pouhých 8 fps). Ale neměl jsem jinou možnost než využít prostředí, pod kterým můžu alespoň programovat, i když v neporovnatelně horších podmínkách. Na výkonu se pravděpodobně podepsala obecnost ovladačů, které dokáží sice obsloužit mnoho kamer, ale z důvodu nedostatečného nebo dokonce žádného kontaktu vývojářů OV511 s výrobcem zařízení, nejsou schopny z kamery dostat její maximum. Pro vyloučení negativního vlivu dalšího počítačového vybavení, které mám k dispozici, jsem provedl testy kamery i na jiným pracovních stanicích s v podstatě stejnými výsledky.

2.2.1 Video4Linux

Video4Linux, zkráceně V4L, je API (Application Program Interface) pro videozařízení pod Linuxem, tedy jednotný standard přístupu k ovladačům zařízení typu televizní karta, radiová karta a různé typy kamer. Pokud je na počítači nainstalován správný ovladač podporující V4L rozhraní a je připojena příslušná karta, pak není problém programovat aplikace nezávisle na konkrétním typu vstupního zařízení.

V dnešní době se mnohem častěji setkáváme s již novější verzí Video4Linux2 (V4L2), které je navrženo mnohem robustněji a přehledněji. Od verze jádra 2.6 se stala dokonce jeho nedílnou součástí. Navíc zpřístupňuje některé nové funkce videozařízení, jako např. časování jednotlivých snímků obrazu atp. Bohužel mnou používaný ovladač kamery OV511 v době psaní tohoto textu stále ještě nepodporoval nové V4L2. V mé práci mě to ale příliš neomezovalo. Proto následující text vychází pouze z V4L1, které jsem měl možnost používat.

2.2.2 Programování s V4L

V4L API se skládá ze systémových volání *open*, *read*, *ioctl* a *close*. Většina příkazů se provádí přes *ioctl* systémové volání. Je nutné mít nainstalován videodev modul, který je vlastní implementací V4L, nějaké vstupní videozařízení a k němu odpovídající ovladač, nejčastěji v podobě modulu jádra systému.

Jednoduchý postup použití V4L[3]:

1. Video zařízení je v systému reprezentováno nejčastěji souborem */dev/video*, což je většinou symlink na */dev/video*0.

[2] Podrobnější informace o tomto kroku v českém jazyce lze nalézt například na `http://www.abclinuxu.cz/clanky/navody/cesta-do-hlubin-kompilace-jadra-1`.

[3] http://linux.bytesex.org/v4l2/

```
fd = open("/dev/video", O_RDWR);
```

Je nutné zařízení otevřít nejen pro čtení, ale i pro zápis, protože budeme mapovat vnitřní paměť framebufferu zařízení, podrobněji později.

2. Přes jednotné přístupové rozhraní je nutné zjistit vlastnosti konkrétního připojeného videozařízení, tedy podporovanou část API.

```
struct video_capability cap;
ioctl(fd, VIDIOCGCAP, &cap);
```

Struktura video_capability obsahuje například jméno koncového zařízení (OV511+ USB Camera), jeho typ (VID_TYPE_CAPTURE, VID_TYPE_SUBCAPTURE - zařízení umí ukládat obraz případně jen jeho část do paměti), počet audio a video kanálu, minimální a maximální rozlišení.

3. Videozařízení, které podporuje zápis přímo do frame bufferu (cap.type == VID_TYPE_OVERLAY), můžeme předat informace o námi zpřístupněném frame bufferu - adresu, jeho velikost a organizaci.

```
struct video_buffer fbuf;
buf.base = ...;
fbuf.height = ...;
fbuf.width = ...;
fbuf.depth = ...;
fbuf.bytesperline = ...;
ioctl(fd, VIDIOCSFBUF, &fbuf);
```

Naopak přerušením VIDIOCGFBUF získáme aktuální nastavení frame bufferu.

4. Zařízení využívající V4L rozhraní může obecně přijímat signál z více kanálů (channels). Jejich počet jsme již získali v položce struktury cap.channels. Informace o jednotlivých kanálech lze získat voláním přerušení VIDIOCGCHAN s přednastaveným číslem požadovaného kanálu.

```
struct video_channel chan;
chan.channel = 0;
ioctl(fd, VIDIOCGCHAN, &chan);
```

Zbylé položky proměnné chan jsou vyplněny informacemi o nultém kanálu - jméno, počet tunnerů, typ TV/CAMERA, norma atp. Nastavení aktuálního vstupního kanálu je možné pomocí přerušení VIDIOCSCHAN, kde je postup obdobný.

5. Rozměry snímaného obrazu jsou popsány v struct video_window. Kromě velikosti obrazu jsou zde i informace o výřezu (clipping information), pokud jsou pro dané zařízení vůbec relevantní. Z mnoha parametrů struktury jsou nejvýznamější šířka a výška obrazu.

```
struct video_window win;
ioctl(fd, VIDIOCGWIN, &win);
win.width = 320;
win.height = 240;
ioctl(fd, VIDIOVSWIN, &win);
```

Tímto způsobem lze snadno nastavit konkrétní rozlišení. V případě nastavení nějakého nestandardního rozlišení může dojít k nedefinovanému chování. Možná rozlišení jsou závislá na konkrétním zařízení, což bohužel pomocí V4L nelze ověřit.

6. Vlastnosti snímaného obrazu lze získat ve struktuře struct video_picture voláním přerušení VIDIOCGPICT. Běžným způsobem lze jednotlivé vlastnosti i nastavit.

```
struct video_picture pic;
ioctl(fd, VIDIOCGPICT, &pic);
pic.brightness = pic.hue = pic.colour =
    pic.contrast = pic.whiteness = 32767;
pic.depth = 24;
pic.palette = VIDEO_PALETTE_RGB24;
ioctl(fd, VIDIOCSPICT, &pic);
```

Kromě palety a barevné hloubky jsou všechny hodnoty v rozsahu 0-65535.

7. V případě, že zvolený kanál má přístupný alespoň jeden tuner, je možné ho pomocí přerušení VIDIOCGTUNER a VIDIOCSTUNER a struktury struct video_tuner nastavovat. Podrobnější informace jsou v obsaženy v V4L API dokumentaci. Protože jsem měl k dispozici pouze jednoduchou USB kameru, podrobněji jsem se laděním nezabýval, podobně jako s audio výstupem.

8. Čtení obrazu ze zařízení může probíhat buď přímo pomocí operace read z file descriptoru fd, nebo rychlejším způsobem pomocí mapování paměti. Tento způsob zde popíši.

   Nejdříve je potřeba zjistit velikost vnitřní paměti zařízení a počet framů, do kterých je členěn.

   ```
   struct video_mbuf mbuf;
   ioctl(fd, VIDIOCGMBUF, &mbuf);
   ```

   Velikost je nyní uložena v mbuf.size a počet framů v mbuf.frames. Nyní již můžeme naalokovat sdílenou paměť

   ```
   void* map = mmap(0, mbuf.size, PROT_READ, MAP_SHARED, fd, 0)
   ```

   kde PROT_READ znamená, že paměť bude procesem jen čtena a parametr MAP_SHARED určuje sdílení paměti, tedy veškeré změny ve vnitřní paměti zařízení se projeví i v paměti procesu na adrese map. Podrobnější informace o použití příkazu mmap lze nalézt například na http://www.opengroup.org/onlinepubs/7908799/xsh/mmap.html.

   Nyní voláním přerušení VIDIOCMCAPTURE nastartujeme nahrávání aktuálního obrazu, a to podle mapování určeném ve struktuře struct video_mmap, ve které mimo jiné určíme číslo bufferu, který se má využít pro uložení obrazu. Toto volání můžeme provést vícekrát, a to podle počtu dostupných bufferů. Samotný návrat z VIDIOCMCAPTURE ještě neznamená dokončení vytváření obrazu, k tomu je určeno jiné přerušení VIDIOCSYNC. Jednoduchý model double-bufferingu by pak mohl vypadat třeba takto:

   ```
   int frame;
   struct video_mmap mmap;
   mmap.frame = 0;
   ioctl(fd, VIDIOCMCAPTURE, mmap);
   while(1) {
       mmap.frame = 1;
       ioctl(fd, VIDIOCMCAPTURE, mmap);
       frame = 0;
       ioctl(fd, VIDIOCSYNC, &frame);
       /* zpracování framu 0, během nahrávání framu 1 hardwarem */
       mmap.frame = 0;
       ioctl(fd, VIDIOCMCAPTURE, mmap);
       frame = 1;
       ioctl(fd, VIDIOCSYNC, &frame);
       /* zpracování framu 1, během nahrávání framu 0 hardwarem */
   ```

```
}
```

Samotná data v příslušném formátu lze pak číst přímo z adresy alokace sdílené paměti posunuté podle požadovaného framu.

```
buf = map + mbuf.offsets[frame];
```

2.2.3 Problémy s konkrétním zařízením

Při psaní jednoduché aplikace pro USB kameru Creative WebCam PRO jsem narazil na několik problémů. A to především na problém s chybným nastavením typu palety. Předpoklad, že formát VIDEO_PALETTE_RGB24 bude jako jeden ze standardů funkční, byl mylný. Jediné provozuschopné nastavení struct video_picture pro moji kameru se nakonec ukázalo pouze pic.palette = VIDEO_PALETTE_YUV420P.

Další neméně významný problém byl již zmiňovaný nedostatek v dosaženém výkonu - 320 x 240 x 8fps bohužel neposkytuje příliš velký komfort pro další práci.

Kapitola 3

Kalibrace kamery

Pro optické snímání pohybu potřebujeme co nejpřesnější informace o optických zařízeních, kterými pohyb sledujeme. Proto je nutné po vhodném rozmístění kamer snímajících pohyb určit jejich vlastnosti - kalibrace kamer. Obecně jí rozumíme algoritmické hledání vlastností kamery, které se dělí na vnitřní a vnější. Vnitřní parametry jsou nezávislé na umístění a pro danou kameru neměnné, jedná se například o ohniskovou vzdálenost, aspect ratio, různé deformace obrazu atp., ty stačí pro každou kameru určit pouze jednou. Vnější parametry jsou především pozice a orientace kamer v prostoru, kalibrace těchto parametrů je nutná po každé změně v umístění kamery. Algoritmy pro určení těchto parametrů byly publikovány už v 70. a 80. letech minulého století. Dále v této kapitole popíši asi dva nejznámější a nejpoužívanější algoritmy kalibrace kamer.

3.1 Direct linear transformation (DLT) [2]

První z nich je algoritmus Direct Linear Transformation publikovaný v [1]. Pro kalibraci je nutné znát dostatečné množství kalibračních bodů se známou polohou v 3D souřadnicích scény a zároveň jejich 2D souřadnice v referenčním obrazu. Algoritmus pracuje s ideálním modelem dírkové kamery. Tedy bod **O** o souřadnicích $[x, y, z]$ je promítán do zobrazovací roviny na souřadnice $[u, v]$ přes střed projekce (viz obrázek 3.1).

Dále rozlišujeme dvě souřadné soustavy. Jednu pro scénu a druhou pro kameru, jedním z výsledků kalibrace je právě převod z jedné na druhou.

Předpokládejme, že střed projekce **N** leží ve scénických souřadnicích na $[x_0, y_0, z_0]$. Vektor **A** nabývá tedy souřadnice $[x - x_0, y - y_0, z - z_0]$ (viz obrázek 3.2).

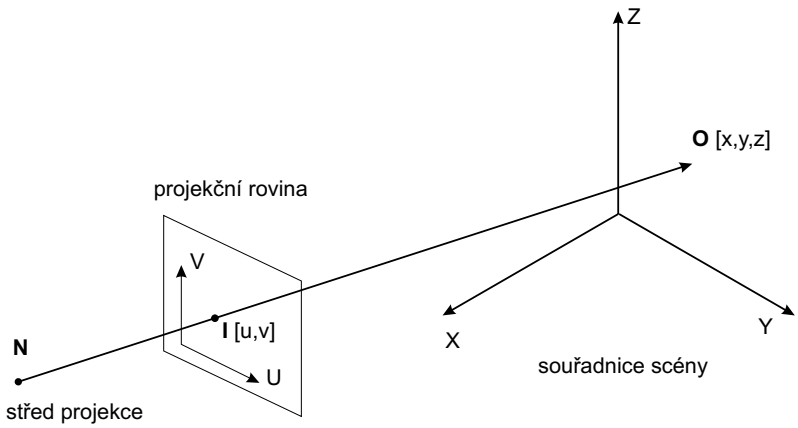
Na obrázku 3.3 vidíme souřadnice významných bodů v souřadném systému kamery. **N** je již popsaný střed promítání, **P** je střed roviny promítání - průsečík roviny promítání s její kolmicí procházející bodem **N** (hlavní osa). Ohnisková vzdálenost d je vzdálenost mezi body **N** a **P**. Vektor **B** je tedy definován jako $[u - u_0, v - v_0, -d]$.

Protože body **O**, **I** a **N** jsou kolineární, lze jejich vztah vyjádřit jako

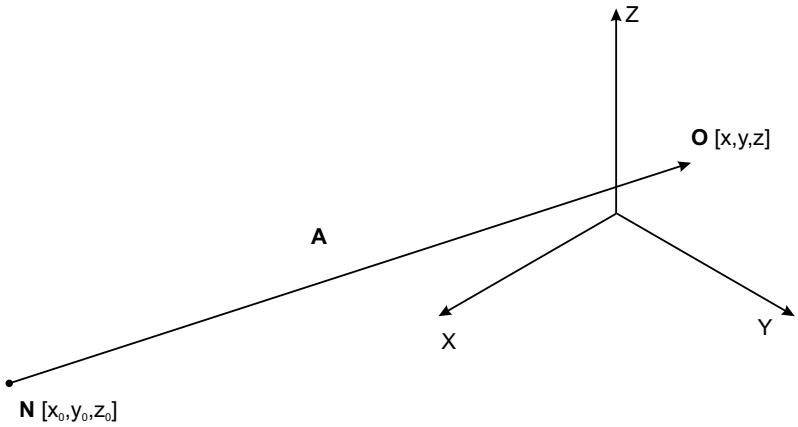
$$\mathbf{B} = c\mathbf{A}, \tag{3.1}$$

kde c je škálovací konstanta. Dosud jsme ale vektory **A** a **B** popisovali v různých souřadných soustavách, proto je nutné převést vektor **A** do soustavy kamery. K tomu nám poslouží matice rotace

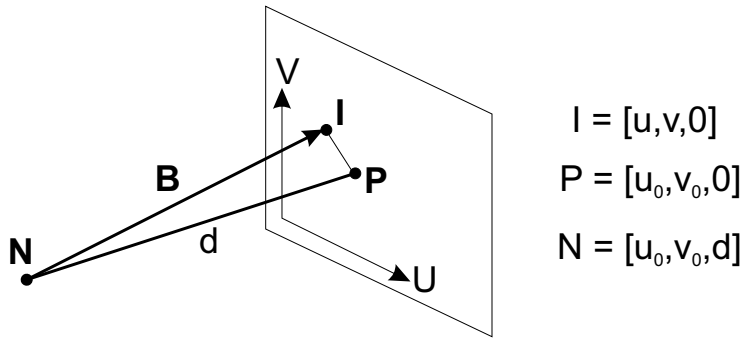
$$\begin{aligned} \mathbf{T}_{K/S} &= \begin{bmatrix} r & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \\ \mathbf{A}^{(K)} &= \mathbf{T}_{K/S}\mathbf{A}^{(S)} = \begin{bmatrix} r & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \mathbf{A}^{(S)}, \end{aligned} \tag{3.2}$$



Obrázek 3.1: Model dírkové kamery.



Obrázek 3.2: Vektor **A** ze středu projekce do prostoru scény.



Obrázek 3.3: Popis souřadné soustavy kamery.

kde $\mathbf{A}^{(K)}$je vektor $\mathbf{A}$ popsaný v souřadnicích kamery a $\mathbf{A}^{(S)}$je vektor $\mathbf{A}$ popsaný v souřadnicích scény.

Dosazením (3.2) do (3.1) dostáváme

$$\begin{bmatrix} u-u_0 \\ v-v_0 \\ -d \end{bmatrix} = c \begin{bmatrix} r & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x-x_0 \\ y-y_0 \\ z-z_0 \end{bmatrix} \tag{3.3}$$

nebo jinak

$$\begin{aligned} u-u_0 &= c\left[r_{11}(x-x_0)+r_{12}(y-y_0)+r_{13}(z-z_0)\right] \\ v-v_0 &= c\left[r_{21}(x-x_0)+r_{22}(y-y_0)+r_{23}(z-z_0)\right] \\ -d &= c\left[r_{31}(x-x_0)+r_{32}(y-y_0)+r_{33}(z-z_0)\right]. \end{aligned} \tag{3.4}$$

Vyjádřením konstanty c ze třetí rovnice a dosazením do dvou předchozích získáme

$$\begin{aligned} u-u_0 &= -d\,\frac{r_{11}(x-x_0)+r_{12}(y-y_0)+r_{13}(z-z_0)}{r_{31}(x-x_0)+r_{32}(y-y_0)+r_{33}(z-z_0)} \\ v-v_0 &= -d\,\frac{r_{21}(x-x_0)+r_{22}(y-y_0)+r_{23}(z-z_0)}{r_{31}(x-x_0)+r_{32}(y-y_0)+r_{33}(z-z_0)}. \end{aligned} \tag{3.5}$$

V tuto chvíli je nutné si uvědomit, že proměnné u, v, u_0 a v_0 z (3.5) jsou ve stejných jednotkách jako v scénických souřadnicích (např. v mm). Ale kamera nám dává souřadnice v pixelech, proto je nutné zavést převod

$$\begin{aligned} u-u_0 &\Rightarrow \lambda_u(u-u_0) \\ v-v_0 &\Rightarrow \lambda_v(v-v_0) \\ u-u_0 &= -\frac{d}{\lambda_u}\,\frac{r_{11}(x-x_0)+r_{12}(y-y_0)+r_{13}(z-z_0)}{r_{31}(x-x_0)+r_{32}(y-y_0)+r_{33}(z-z_0)} \\ v-v_0 &= -\frac{d}{\lambda_v}\,\frac{r_{11}(x-x_0)+r_{12}(y-y_0)+r_{13}(z-z_0)}{r_{31}(x-x_0)+r_{32}(y-y_0)+r_{33}(z-z_0)}, \end{aligned} \tag{3.6}$$

kde λ_ua λ_v jsou konverzní konstanty pro osy U a V. Předpokládá se, že mohou byt pro každou z os různé. Přidáním dvou proměnných jsme dosáhli toho, že proměnné u, v, u_0 a v_0 v (3.6) mohou být v libovolných jednotkách.

Poměrně snadnými úpravami lze z (3.6) odvodit

$$\begin{aligned} u &= \frac{L_1x+L_2y+L_3z+L_4}{L_9x+L_{10}y+L_{11}z+1} \\ v &= \frac{L_1x+L_2y+L_3z+L_4}{L_9x+L_{10}y+L_{11}z+1}, \end{aligned} \tag{3.7}$$

kde

$$\begin{aligned} L_1 &= \frac{u_0r_{31}-d_ur_{11}}{D} \\ L_2 &= \frac{u_0r_{32}-d_ur_{12}}{D} \\ L_3 &= \frac{u_0r_{33}-d_ur_{13}}{D} \\ L_4 &= \frac{(d_ur_{11}-u_0r_{31})x_0+(d_ur_{12}-u_0r_{32})y_0+(d_ur_{13}-u_0r_{33})z_0}{D} \\ L_5 &= \frac{v_0r_{31}-d_vr_{21}}{D} \\ L_6 &= \frac{v_0r_{32}-d_vr_{22}}{D} \end{aligned}$$

$$\begin{aligned}
L_7 &= \frac{v_0 r_{33} - d_v r_{23}}{D} \\
L_8 &= \frac{(d_v r_{21} - v_0 r_{31})\, x_0 + (d_v r_{22} - v_0 r_{32})\, y_0 + (d_v r_{23} - v_0 r_{33})\, z_0}{D} \\
L_9 &= \frac{r_{31}}{D} \\
L_{10} &= \frac{r_{32}}{D} \\
L_{11} &= \frac{r_{33}}{D} \\
[d_u, d_v] &\equiv \left[\frac{d}{\lambda_u}, \frac{d}{\lambda_v}\right] \\
D &= -\left(x_0 r_{31} + y_0 r_{32} + z_0 r_{33}\right). \qquad (3.8)
\end{aligned}$$

Koeficienty L_1 až L_{11} v (3.7) se nazývají DLT parametry a vyjadřují vztah mezi scénickou soustavou a soustavou kamery.

Tento model je vhodné ještě rozšířit o deformace obrazu vznikající především u levnějších optických zařízení. Úpravou (3.7) získáme

$$\begin{aligned}
u - \Delta u &= \frac{L_1 x + L_2 y + L_3 z + L_4}{L_9 x + L_{10} y + L_{11} z + 1} \\
v - \Delta v &= \frac{L_1 x + L_2 y + L_3 z + L_4}{L_9 x + L_{10} y + L_{11} z + 1}, \qquad (3.9)
\end{aligned}$$

kde Δu a Δv jsou optické chyby. Ty můžeme vyjádřit například takto

$$\begin{aligned}
\Delta u &= \xi\left(L_{12} r^2 + L_{13} r^4 + L_{14} r^6\right) + L_{15}\left(r^2 + 2\xi^2\right) + L_{16}\xi\eta \\
\Delta v &= \eta\left(L_{12} r^2 + L_{13} r^4 + L_{14} r^6\right) + L_{15}\eta\xi + L_{16}\left(r^2 + 2\eta^2\right),
\end{aligned}$$

kde

$$\begin{aligned}
[\xi, \eta] &= [u - u_0, v - v_0] \\
r^2 &= \xi^2 + \eta^2.
\end{aligned}$$

Přidali jsme tedy pět nových proměnných L_{12} až L_{16}, z nichž první tři se vztahují k optickému zkreslení (optical distortion) a dvě k decentralizaci (de-centering distortion).

Dosažené výsledky především v (3.8) lze využít nejen k samotné kalibraci kamery, ale také k zpětnému určení 3D souřadnic bodu ve scéně (bude naznačeno později).

3.1.1 Kalibrace

Vhodným upravením (3.9) získáme

$$\begin{aligned}
\frac{1}{R}u &= \frac{1}{R}\left(L_1 x + L_2 y + L_3 z + L_4 - L_9 u x - L_{10} u y - L_{11} u z\right) + \Delta u \\
\frac{1}{R}v &= \frac{1}{R}\left(L_5 x + L_6 y + L_7 z + L_8 - L_9 v x - L_{10} v y - L_{11} v z\right) + \Delta v,
\end{aligned}$$

kde

$$R = L_9 x + L_{10} y + L_{11} z + 1,$$

což lze ekvivalentně zapsat pomocí matic

$$\frac{1}{R}\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{R}\begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & -ux & -uy & -uz \\ 0 & 0 & 0 & 0 & x & y & z & 1 & -vx & -vy & -vz \end{bmatrix}$$

$$\begin{bmatrix} \xi r^2 R & \xi r^4 R & \xi r^6 R & \left(r^2+2\xi^2\right) R & \xi\eta R \\ \eta r^2 R & \eta r^4 R & \eta r^6 R & \eta\xi R & \left(r^2+2\eta^2\right) R \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_{15} \\ L_{16} \end{bmatrix}. \tag{3.10}$$

Za předpokladu, že máme n kontrolních bodů můžeme (3.10) rozepsat

$$\begin{bmatrix} \frac{x_1}{R_1} & \frac{y_1}{R_1} & \frac{z_1}{R_1} & \frac{1}{R_1} & 0 & 0 & 0 & 0 & \frac{-u_1 x_1}{R_1} & \frac{-u_1 y_1}{R_1} & \frac{-u_1 z_1}{R_1} \\ 0 & 0 & 0 & 0 & \frac{x_1}{R_1} & \frac{y_1}{R_1} & \frac{z_1}{R_1} & \frac{1}{R_1} & \frac{-v_1 x_1}{R_1} & \frac{-v_1 y_1}{R_1} & \frac{-v_1 z_1}{R_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{x_n}{R_n} & \frac{y_n}{R_n} & \frac{z_n}{R_n} & \frac{1}{R_n} & 0 & 0 & 0 & 0 & \frac{-u_n x_n}{R_n} & \frac{-u_n y_n}{R_n} & \frac{-u_n z_n}{R_n} \\ 0 & 0 & 0 & 0 & \frac{x_n}{R_n} & \frac{y_n}{R_n} & \frac{z_n}{R_n} & \frac{1}{R_n} & \frac{-v_n x_n}{R_n} & \frac{-v_n y_n}{R_n} & \frac{-v_n z_n}{R_n} \end{bmatrix}$$

$$\begin{bmatrix} \xi_1 r_1^2 & \xi_1 r_1^4 & \xi_1 r_1^6 & r_1^2+2\xi_1^2 & \xi_1\eta_1 \\ \eta_1 r_1^2 & \eta_1 r_1^4 & \eta_1 r_1^6 & \eta_1\xi_1 & r_1^2+2\eta_1^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \xi_n r_n^2 & \xi_n r_n^4 & \xi_n r_n^6 & r_n^2+2\xi_n^2 & \xi_n\eta_n \\ \eta_n r_n^2 & \eta_n r_n^4 & \eta_n r_n^6 & \eta_n\xi_n & r_n^2+2\eta_n^2 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_{15} \\ L_{16} \end{bmatrix} = \begin{bmatrix} \frac{u_1}{R_1} \\ \frac{v_1}{R_1} \\ \vdots \\ \frac{u_n}{R_n} \\ \frac{v_n}{R_n} \end{bmatrix}. \tag{3.11}$$

K řešení této soustavy rovnic je tedy nutné mít n kontrolních bodů se známými scénickými souřadnicemi $[x_i, y_i, z_i]$. Dále je nutné zajistit rozmístění kontrolních bodů v prostoru, nelze použít koplanární sadu bodů. Soustavu lze zjednodušit vypuštěním některých DLT parametrů, do úvahy přicházejí především parametry L_{12} až L_{16}. Dále je potřeba si uvědomit, že R_i jsou funkcí proměnných L_9, L a L_{11} a vyskytují se v matici koeficientů. Proto nelze tento systém rovnic řešit přímo, je nutné použít iterativní přístup.

Způsobů jak řešit úlohu (3.11) je několik. Například metodou nejmenších čtverců[1]. Zjednodušme zápis (3.11)

$$\mathbf{XL} = \mathbf{Y},$$

potom lze $\mathbf{L}$ vyjádřit takto

$$\begin{aligned} \mathbf{XL} &= \mathbf{Y} \\ \left(\mathbf{X}^t\mathbf{X}\right)\mathbf{L} &= \mathbf{X}^t\mathbf{Y} \\ \left(\mathbf{X}^t\mathbf{X}\right)^{-1}\left(\mathbf{X}^t\mathbf{X}\right)\mathbf{L} &= \left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t\mathbf{Y} \\ \mathbf{L} &= \left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t\mathbf{Y}. \end{aligned}$$

3.2 Tsai kalibrace [3]

Další velmi často používaná kalibrace kamery byla představena Tsaiem v [4]. Algoritmus z vnitřních parametrů detekuje ohniskovou vzdálenost a souřadnice hlavního bodu (principal point), u levných kamer velmi významné koeficienty optického zkreslení a navíc i scale faktory v obou osách (chyby vzniklé při vzorkování z optického zařízení do obrazu). Orientace a pozice kamery jsou samozřejmostí. Algoritmus určí nejlepší možné řešení výše uvedených parametrů z množiny testovacích bodů se známým umístěním v kalibrační scéně. Pracuje ve dvou hlavních fázích, část parametrů odhadne metodou minimálních čtverců a teprve ve druhé fázi iterativní nelineární optimalizací zpřesňuje odhad všech parametrů, jako startovací hodnota první iterace se využije výsledek první fáze algoritmu. Postup výpočtů se liší pro koplanární množinu testovacích bodů a pro body rozmístěné v prostoru. Koplanární body skýtají jistá omezení, podrobněji dále v textu.

Velmi často používanou implementaci toho algoritmu napsal Reg Willson [5].

[1] http://mathworld.wolfram.com/LeastSquaresFitting.html

3.2.1 Vztah souřadnic kamery a obrazu

Souřadný systém kamery má svůj střed ve středu promítání, z-tovou osu podél optické osy, x-ovou a y-ovou rovnoběžně s příslušnými osami obrazu. Vztah mezi soustavami je dán jednoduchou přímou úměrou

$$\begin{aligned} \frac{x_I - x_0}{f} &= \frac{x_C}{z_C} \\ \frac{y_I - y_0}{f} &= \frac{y_C}{z_C}, \end{aligned} \tag{3.12}$$

kde f je ohnisková vzdálenost, $[x_0, y_0]$ hlavní bod (principal point - průsečík kolmice s obrazovou rovinou procházející středem projekce), $[x_I, y_I]$ obrazové souřadnice, $[x_C, y_C, z_C]$ souřadnice kamery. Neznámými proměnnými jsou zde ohnisková vzdálenost f a hlavní bod $[x_0, y_0]$.

3.2.2 Vztah souřadnic scény a kamery

Jedná se o převod mezi soustavou centrovanou v kameře a libovolnou další soustavou vhodnou pro scénu. Tento převod je určen rotací a posunutím, transformace má tedy 6 stupňů volnosti, tři pro otočení a tři pro posun.

Jako reprezentaci rotace lze použít ortonormální matici a pro posun jednoduchý vektor, potom převod můžeme zapsat třeba takto

$$\begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_S \\ y_S \\ z_S \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \tag{3.13}$$

kde $[x_C, y_C, z_C]^T$jsou souřadnice v soustavě kamery, $[x_S, y_S, z_S]^T$ jsou souřadnice v soustavě scény, matice $R = [r_{ij}]$ je orthonormální matice rotace a $[t_x, t, t_z]^T$ je vektor posunu. Zde jsou neznámými právě matice rotace a vektor posunu.

3.2.3 Horizontální scale faktor

U většiny moderních elektronických kamer se setkáváme s problémem nepřesného přenosu navzorkovaných dat.

Ve většině typických CCD a CMOS kamerách se původně diskrétní signál v analogové formě prahuje nízkofrekvenčním filtrem (low pass filtred), aby se dosáhlo rozmáznutí video výstupu a tedy zakrytí přechodu mezi jednotlivými buňkami senzoru. Tento analogový signál je pak digitalizován v frame grabberu. Vzorkování v horizontálním směru ve frame grabberu není bohužel většinou stejné jako rozmístění prvků v optickém senzoru a tento rozdíl není většinou určen ani výrobcem. Horizontální vzdálenost mezi pixely je obecně jiná než vzdálenost mezi buňkami senzoru.

Na druhou stranu ve vertikálním směru je vzorkování správné. Proto poměr velikosti buněk obrazu v horizontálním směru a ve vertikálním směru není přímo určen poměrem velikosti buněk senzoru. Tento poměr je třeba také určit v rámci kalibrace kamery. Modifikovaná rovnice (3.12)

$$\frac{x_I - x_0}{f} = s\frac{x_C}{z_C}, \tag{3.14}$$

kde s je právě neznámý poměr velikosti pixelu v x-ovém a y-ovém směru.

Tento poměr není možné určit v případě koplanárních vstupních dat kalibrace kamery, v takovém případě se musí řešit zvlášť.

3.2.4 Vztah souřadnic scény a obrazu

Aplikací vzorce pro vztah souřadnic kamery a obrazu (3.12) a vzorce pro vztah souřadnic scény a kamery (3.13) získáme

$$\frac{x_I - x_0}{f} = s\frac{r_{11}x_S + r_{12}y_S + r_{13}z_S + t_x}{r_{31}x_S + r_{32}y_S + r_{33}z_S + t_z}$$

$$\frac{y_I - y_0}{f} = \frac{r_{21}x_S + r_{22}y_S + r_{23}z_S + t_y}{r_{31}x_S + r_{32}y_S + r_{33}z_S + t_z}. \tag{3.15}$$

3.2.5 Optické zkreslení

Doteď jsme při všech úvahách předpokládali ideální model dírkové kamery, ovšem většina kamer vykazuje v menší či větší míře optické zkreslení. U optických systémů se sférickou čočkou se vyskytuje geometrické zkreslení v paprskovitém směru od středu obrazu (hlavního bodu). Zobrazovaný bod má buď větší (pin-cushion distortion), nebo menší (barrel distortion) vzdálenost od hlavního bodu než odhadovaný výsledek výše odvozených vzorců, chyba roste se vzdáleností a je menší ve směrech více rovnoběžných s hlavními osami obrazu.

Chybu způsobenou radiálním zkreslením (radial lens distortion) nejčastěji odhadujeme takto

$$\begin{aligned} \delta x &= x\left(\kappa_1 r^2 + \kappa_2 r^4 + \ldots\right) \\ \delta y &= y\left(\kappa_1 r^2 + \kappa_2 r^4 + \ldots\right), \end{aligned}$$

kde x a y jsou měřeny od středu zkreslení a r je vzdálenost bodu od středu zkreslení, které je většinou na stejném místě jako hlavní bod. Ve vzorci se vyskytují pouze sudé mocniny r, a to většinou jen první nebo první a druhý člen.

Elektro-optické systémy často trpí i tangentním zkreslením (tangential distortion). Podobně jako radiální zkreslení roste i toto se vzdáleností od středu zkreslení

$$\begin{aligned} \delta x &= -y\left(\epsilon_1 r^2 + \epsilon_2 r^4 + \ldots\right) \\ \delta y &= x\left(\epsilon_1 r^2 + \epsilon_2 r^4 + \ldots\right). \end{aligned}$$

Odhady parametrů κ_i, ϵ_j těchto mocninných řad jsou součástí výpočtu kalibrace. I když existují i samostatná řešení.

3.2.6 Postup algoritmu

Použitím známé kalibrační scény s množinou významných testovacích bodů získáme dvojice bodů ve scéně a jejich obrazy. Toto jsou hlavní vstupní data kalibrace. V první fázi se pokusíme odhadnout některé parametry metodou minimální čtverců, což lze řešit velmi rychle nalezením pseudoinverzní matice. V tomto inicializačním kroku se například nebudeme vůbec zabývat ortonormalitou rotační matice, nebudeme se dokonce ani snažit minimalizovat chybu výpočtu, jde především o zjednodušení následující fáze pomocí jednoduché soustavy lineárních rovnic. Výsledek této fáze bude použit jen jako startovní vstup iteračního procesu ve druhé optimalizační části algoritmu.

Zbytek parametru bude řešen tedy až ve druhé části algoritmu již pomocí nelineární optimalizace. Algoritmus se mírně liší pro koplanární vstup. Pokud nebude řečeno jinak, v následujícím textu předpokládáme množinu testovacích bodů rozmístěnou v prostoru, nikoli v rovině.

3.2.7 Hledání rotace a části posunu

Pro tuto fázi potřebujeme odhad pozice hlavního bodu $[x_0, y_0]$. Pro většinu optických prvků je velmi dobrým odhadem střed obrazu. Použijme značení

$$\begin{aligned} x_I' &= x_I - x_0 \\ y_I' &= y_I - y_0, \end{aligned} \tag{3.16}$$

potom z (3.12) a z (3.14) dostaneme

$$\begin{aligned} \frac{x_I'}{f} &= s\frac{x_C}{z_C} \\ \frac{y_I'}{f} &= \frac{y_C}{z_C}. \end{aligned}$$

Dále uvažujme jen nad směrem bodu od hlavního bodu, to vede k nezávislosti na ohniskové vzdálenosti f a zároveň na radiálním zkreslení

$$\frac{x_I^{'}}{y_I^{'}} = s\frac{x_C}{y_C}.$$

Dále z (3.15) a předchozího máme

$$\frac{x_I^{'}}{y_I^{'}} = s\frac{r_{11}x_S + r_{12}y_S + r_{13}z_S + t_x}{r_{21}x_S + r_{22}y_S + r_{23}z_S + t_y}, \tag{3.17}$$

což lze dále upravit na

$$\begin{aligned}(x_S y_I^{'})sr_{11} + (y_S y_I^{'})sr_{12} + (z_S y_I^{'})sr_{13} + y_I^{'}st_x \\ -(x_s x_I^{'})r_{21} - (y_S x_I^{'})r_{22} - (z_S x_I^{'})r_{23} - x_I^{'}t_y = 0.\end{aligned} \tag{3.18}$$

Na toto se můžeme dívat jako na homogenní rovnici o osmi neznámých sr_{11}, sr_{12}, sr_{13}, r_{21}, r_{22}, r_{23}, st_x a t_y. Pro každý testovací bod v kalibrační scéně takto získáme jednu rovnici.

Protože máme homogenní soustavu rovnic, řešením je vždy podprostor, násobením se nedostaneme z prostoru řešení. Proto můžeme jednu proměnnou zafixovat (např. $t_y = 1$), převést homogenní soustavu na nehomogenní o sedmi neznámých, tím snížíme dimenzi prostoru řešení a získáme právě jeden výsledek.

Když získáme nějaké řešení, označme ho $sr_{11}^{'}$, $sr_{12}^{'}$, $sr_{13}^{'}$, $r_{21}^{'}$, $r_{22}^{'}$, $r_{23}^{'}$, $st_x^{'}$ a $t_y^{'} = 1$, samozřejmě jsme se nezbavili problému se škálovatelností výsledku, jakýkoli násobek tohoto řešení je stále řešením. Scale faktor s můžeme odhadnout za předpokladu normality řádků rotační matice

$$\begin{aligned} r_{11}^2 + r_{12}^2 + r_{13}^2 &= 1 \\ r_{21}^2 + r_{22}^2 + r_{23}^2 &= 1. \end{aligned}$$

Potřebujeme nalézt normalizační faktor c, který nám zajistí platnost těchto rovnic pro námi nalezené řešení soustavy (3.18). Snadno lze nahlédnout, že

$$\begin{aligned} c &= 1/\sqrt{r_{21}^{'2} + r_{22}^{'2} + r_{23}^{'2}} \\ c/s &= 1/\sqrt{(sr_{11}^{'})^2 + (sr_{12}^{'})^2 + (sr_{13}^{'})^2}. \end{aligned} \tag{3.19}$$

Nyní už by neměl být problém dopočítat odhad faktoru s.

3.2.8 Ortonormalita rotační matice

Mějme dány dva vektory $\mathbf{a}$ a $\mathbf{b}$, chceme najít jim příslušné ortonormální vektory $\mathbf{a}^{'}$ a $\mathbf{b}^{'}$, které se co nejméně liší od jejich vzorů $\mathbf{a}$ a $\mathbf{b}$

$$\begin{aligned} \mathbf{a}^{'} &= \mathbf{a} + k\mathbf{b} \\ \mathbf{b}^{'} &= \mathbf{b} + k\mathbf{a} \\ \mathbf{a}^{'} \cdot \mathbf{b}^{'} &= \mathbf{a}\cdot\mathbf{b} + k(\mathbf{a}\cdot\mathbf{a} + \mathbf{b}\cdot\mathbf{b}) + k^2\mathbf{a}\cdot\mathbf{b} = 0. \end{aligned}$$

Bohužel řešení této kvadratické rovnice je numericky nestabilní pro "téměř" ortonormální vektory, $\mathbf{a}\cdot\mathbf{b}$ je blízké nule. V takovém případě můžeme použít následující aproximaci výsledku

$$k \approx -\frac{1}{2}\mathbf{a}\cdot\mathbf{b}.$$

Poté, co provedeme ortogonalizaci prvních dvou řádků rotační matice, můžeme ještě provést jejich normalizaci (3.19). Poslední řádek rotační matice získáme jednoduše vektorovým součinem prvních dvou již kolmých a znormovaných vektorů.

3.2.9 Koplanární testovací body

Výše naznačený algoritmus nelze přímo použít pro koplanární množinu testovacích bodů. Znemožňuje to odhadnout scale faktor s, proto budeme dále předpokládat, že souřadnice testovacích bodů jsou již upraveny podle tohoto faktoru.

U koplanární množiny testovacích bodů můžeme bez újmy na obecnosti předpokládat $z_S = 0$ pro každý bod. Po tomto zjednodušení získáme z (3.17)

$$\frac{x_I^{'}}{y_I^{'}} = \frac{r_{11}x_S + r_{12}y_S + t_x}{r_{21}x_S + r_{22}y_S + t_y},$$

což opět upravíme na

$$(x_S y_I^{'})r_{11} + (y_S y_I^{'})r_{12} + y_I^{'}t_x - (x_s x_I^{'})r_{21} - (y_S x_I^{'})r_{22} - x_I^{'}t_y = 0.$$

Opět máme lineární homogenní soustavu rovnic, tentokrát o šesti neznámých r_{11}, r_{12}, r_{21}, r_{22}, t_x a t_y. Podobně jako u nekoplanárního vstupu si jednu z proměnných zafixujeme (např. $t_y = 1$) a řešíme nehomogenní soustavu lineárních rovnic o pěti proměnných. Stačí nám tedy jen pět testovacích bodů, při větším počtu lze opět využít předeterminování úlohy a metodou minimálních čtverců minimalizovat chybu.

3.2.10 Rekonstrukce rotační matice pro koplanární vstup

Nyní potřebujeme dopočítat rotační matici 3 x 3 z levé horní podmatice 2 x 2. Z předpokladu ortonormality rotační matice víme

$$\begin{aligned} r_{11}^2 + r_{12}^2 + r_{13}^2 &= 1 \\ r_{21}^2 + r_{22}^2 + r_{23}^2 &= 1 \\ r_{11}r_{21} + r_{12}r_{22} + r_{13}r_{23} &= 0, \end{aligned}$$

tedy

$$\begin{aligned} r_{11}^{'2} + r_{12}^{'2} + r_{13}^{'2} &= k^2 \\ r_{21}^{'2} + r_{22}^{'2} + r_{23}^{'2} &= k^2 \\ r_{11}^{'}r_{21}^{'} + r_{12}^{'}r_{22}^{'} + r_{13}^{'}r_{23}^{'} &= 0. \end{aligned} \tag{3.20}$$

Tato soustava tří rovnic o třech neznámých vede na kvadratickou rovnici pro k^2

$$k^4 - k^2(r_{11}^{'2} + r_{12}^{'2} + r_{21}^{'2} + r_{22}^{'2}) + (r_{11}^{'}r_{22}^{'} + r_{12}^{'}r_{21}^{'})^2 = 0. \tag{3.21}$$

Zpětným dosazením do (3.20) získáme

$$\begin{aligned} r_{13}^{'2} &= k^2 - (r_{11}^{'2} + r_{12}^{'2}) \\ r_{23}^{'2} &= k^2 - (r_{21}^{'2} + r_{22}^{'2}). \end{aligned}$$

Potřebujeme, aby pravá strana těchto rovnic byla kladná, proto správné řešení naší úlohy je větší z kořenů kvadratické rovnice (3.21)

$$k^2 = \frac{1}{2}\Big((r_{11}^{'2} + r_{12}^{'2} + r_{21}^{'2} + r_{22}^{'2}) + \sqrt{\left((r_{11}^{'} - r_{22}^{'})^2 + (r_{12}^{'} + r_{21}^{'})^2\right)\left((r_{11}^{'} + r_{22}^{'})^2 + (r_{12}^{'} - r_{21}^{'})^2\right)}\Big).$$

Dále už jen stačí normalizovat první dva řádky matice pomocí k a dopočítat jejich vektorovým součinem třetí řádek rotační matice.

Ještě je zde problém s nejednoznačností znamének r'_{13} a r'_{23}. Můžeme získat jen znaménko jejich násobku použitím třetí rovnice z (3.20)

$$r'_{13}r'_{23} = -(r'_{11}r'_{21} + r'_{12}r'_{22}),$$

čímž jsme nejednoznačnost omezili na dvojznačnost. Ale my potřebujeme přesnou podobu rotační matice.

Jedno z možných řešení je promítnutí známých testovacích bodů zpět do obrazu pomocí námi spočítané rotační matice. Pokud jsme si znaménka určili správně, výsledná poloha bude velmi blízko dříve naměřené. V opačném případě jsme špatně spočítali i první dva prvky třetího řádků rotační matice a mnoho testovacích bodů bude vycházet v jiném kvadrantu obrazu. Jedna z možných metrik této chyby, zkusíme jiná znaménka pokud

$$\sum_{i=0}^{N} (x_{I_i} x_{P_i} + y_{I_i} y_{P_i}) < 0.$$

3.2.11 Odhad ohniskové vzdálenosti a vzdálenosti scény

Doposud se nám podařilo odhadnout celou matici rotace a první dva prvky vektoru posunutí. Nyní odhadneme třetí z-tovou souřadnici posunu a ohniskovou vzdálenost z (3.15) a (3.16)

$$\begin{aligned} \frac{x'_I}{f} &= s\frac{r_{11}x_S + r_{12}y_S + r_{13}z_S + t_x}{r_{31}x_S + r_{32}y_S + r_{33}z_S + t_z} \\ \frac{y'_I}{f} &= \frac{r_{21}x_S + r_{22}y_S + r_{23}z_S + t_y}{r_{31}x_S + r_{32}y_S + r_{33}z_S + t_z}, \end{aligned}$$

roznásobením získáme

$$\begin{aligned} s(r_{11}x_S + r_{12}y_S + r_{13}z_S + t_x)f - x'_I t_z &= (r_{31}x_S + r_{32}y_S + r_{33}z_S)x'_I \\ (r_{21}x_S + r_{22}y_S + r_{23}z_S + t_y)f - y'_I t_z &= (r_{31}x_S + r_{32}y_S + r_{33}z_S)y'_I. \end{aligned}$$

Na toto už můžeme nahlížet jako na soustavu rovnic o dvou neznámých, k řešení nám teoreticky stačí jediný testovací bod. Pokud využijeme více testovacích bodů, můžeme takto předeterminovanou úlohu řešit opět metodou minimálních čtverců.

Pokud nemáme odhad scale faktoru s (např. v případě koplanárního vstupu), můžeme použít jen rovnice pro y'_I.

Nyní nám už zbývá nalézt jen hlavní bod $[x_0, y_0]$, koeficienty mocninných řad optických zkreslení a především iterativní minimalizace chyb v odhadech.

Nutnou podmínkou správného odhadu f a t_z je nenulový rozsah hloubky testovacích bodů ve scéně (z_C). Kdyby byly všechny body v jedné hloubce, nebylo by možné rozpoznat vliv ohniskové vzdálenosti a posunu, měli bychom jen jejich poměr f/t_z. Přesnost výpočtu roste právě s rozptylem hloubky bodů ve scéně. Pro koplanární množinu testovacích bodů to znamená, že rovina nesmí být kolmá na optickou osu.

3.2.12 Nelineární optimalizace

Nyní bychom měli minimalizovat chybu, rozdíl mezi naměřenou pozicí testovacího bodu v obrazu $[x_I, y_I]^T$a pozicí $[x_P, y_P]^T$predikovanou naším odhadem správného řešení. Všechny vnitřní i vnější parametry kamery budeme upravovat tak, abychom minimalizovali

$$\sum_{i=1}^{N}(x_{I_i} - x_{P_i})^2 + \sum_{i=1}^{N}(y_{I_i} - y_{P_i})^2.$$

Toto lze řešit iterativní numerickou optimalizací, jako například modifikovaný Levenberg-Marquardt[2].

[2] http://mathworld.wolfram.com/Levenberg-MarquardtMethod.html

Obrázek 3.4: První neúspěšný pokus o detekci významných bodů v kalibrační scéně.

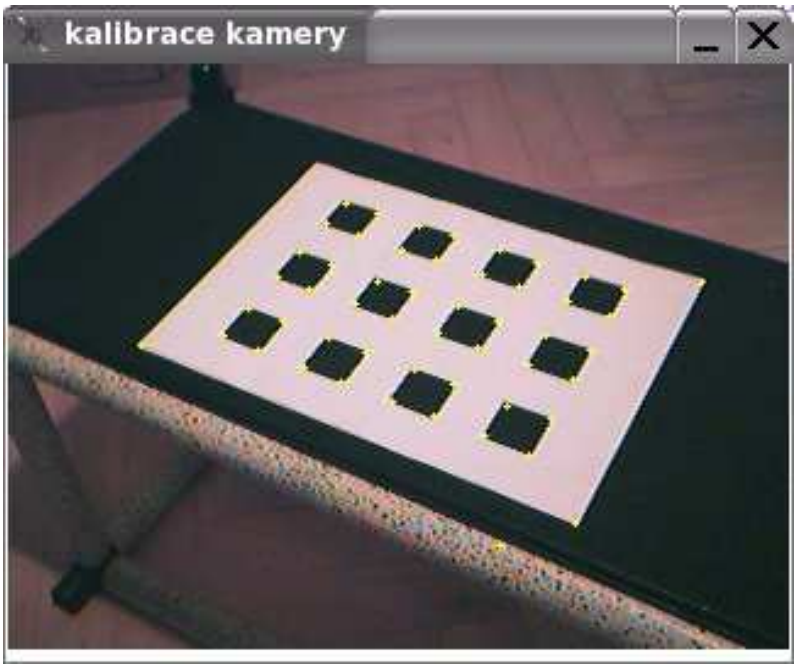
3.3 Hledání kalibračních bodů

Pro všechny výše zmíněné algoritmy kalibrace kamery je nutné zajistit dostatečně velkou množinu kalibračních bodů. Bodů se známou polohou v souřadnicích scény a jejich polohou v nasnímaném obrazu. Rozhodl jsem se využít často citovanou, používanou a tudíž i dobře otestovanou implementaci Tsai algoritmu Rega Willsona [5], jejímž vstupem jsou právě pětice $(x_i, y_i, z_i, u_i, v_i)$.

3.3.1 Volba kalibrační scény

První pokusy jsem prováděl na klasické šachovnici vytištěné na papír formátu A4 velikosti 4x5 (viz obrázek v dodatku A). Předpokladem pro moji volbu bylo snadné rozpoznání rohů jednotlivých polí šachovnice. Na takto získaný obraz jsem aplikoval Cannyho hranový detektor pro nalezení významných hran. Dále jsem použil Gandalf čárový detektor, který aplikovaný na výstup hranového detektoru vrací seznam úseček v obrazu. Bohužel přes velkou snahu o využití všech parametrů jednotlivých výše zmíněných algoritmů se mi nedařilo odstranit množství nalezených hran. Částečný úspěch jsem slavil až s binárním prahováním původního obrazu testovací scény. Využil jsem toho, že černobílá šachovnice má velký barevný odstup jednotlivých polí a správně nastaveným prahem se mi podařilo odstranit většinu málo významných hran v obrazu. Samozřejmě se nepodařilo odstranit všechny nepotřebné hrany, ale redukce byla významná. Postup na takto upravený binární obraz byl stejný, Canny hranový detektor, Gandalf čárový detektor. Při významném snížení počtu nalezených hran zůstala nezměněna kvalita nalezených hran v šachovnici. Jediná, nikoliv nevýznamná, nevýhoda tohoto vylepšení je v nestabilitě v osvětlení scény. Při změně světelných podmínek se pro kvalitní výstup detektorů musí upravit i meze binárního prahování. Což trochu znepříjemňuje použitelnost mé utility na kalibraci kamery.

Ještě než jsem se pokusil o automatické hledání vhodného prahu v závislosti na světelných podmínkách, vytvořil jsem poslední část hledání rohů šachovnice. Ta spočívá v hledání průsečíků přímek určených úsečkami nalezenými Gandalf čárovým detektorem. Samozřejmě jsem aplikoval omezení tak, aby průsečík nebyl "příliš daleko" od konců úseček. Bohužel se mi nepodařilo dosáhnout dostatečně kvalitních výsledků (viz obrázek 3.4). Úspěšnost nalezení rohu šachovnice se pohybovala okolo 80%, přestože jsem se snažil ručně nastavit detektory tak, aby dosahovaly co nejlepších výsledků.



Obrázek 3.5: Výsledek hledání významných bodů pomocí Harris rohového detektoru.

Úspěch jsem začal slavit až při použití Harris rohového detektoru, který je konstruován přímo na hledání rohů v obrazu. Při vhodném nastavení prahu tohoto detektoru jsem dosáhl velmi snadno výborných výsledků. I ve snímku s velmi vysokým úhlem pohledu na kalibrační scénu Harris našel v podstatě všechny rohy kalibrační scény. Jediný drobný problém nastával v občasném zdvojení nalezeného rohu v případě mé šachovnice (viz obrázek 3.5). Tuto malou chybu jsem odstranil nahrazením šachovnice, kalibrační scénou obsahující pouze izolované černé čtverce (viz obrázek v dodatku A).

Přestože Tsai algoritmus umožňuje použít i koplanární množinu bodů, pro lepší výsledky a přesné určení všech požadovaných parametrů kamer jsem vytvořil scénu s testovacími body rozmístěnými v prostoru, složenou ze dvou výše navržených vzorů s izolovanými černými čtverci (viz obrázek 3.7).

3.3.2 Harris rohový detektor [7]

Tento algoritmus na hledání rohů v obrazu odvodil Chris Harris a Mike Stephens v roce 1988 ze staršího Moravcova detektoru rohů. Ten spočíval v jednoduché myšlence měření průměrné změny intenzity při posunu lokálního okna nad obrazem. Mohly nastat tři případy:

1. Jestliže je obraz v lokálním okně a jeho blízkém okolí jednobarevný (konstantní intenzita), potom všechny posuny budou vykazovat jen velmi malé změny intenzity.

2. Jestliže je okno nad hranou, potom změny intenzity při posunu podél takovéto hrany budou malé, ale při posunu kolmo na hranu získáme velké změny.

3. Jestliže je okno nad rohem, případně nad izolovaným bodem, potom všechny posuny budou vykazovat výrazné změny v intenzitě.

Toto lze snadno matematicky vyjádřit. Označíme-li intenzitu obrazu I a její změnu $E_{x,y}$ při posunu ve směru (x, y), snadno odvodíme, že

$$E_{x,y} = \sum_{u,v} w_{u,v}(I_{u+x,v+y} - I_{u,v})^2,$$

kde w určuje okno, nad kterým se změna intenzity počítá (jednička uvnitř okna a nula mimo). Posuny se nejčastěji uvažují ve všech osmi směrech o jeden pixel $[(1,0),(0,1),(1,1),...]$. Samotný Moravcův detektor už jen hledá lokální maxima v $\min\{E\}$, která přesáhnou jistý práh. Takovéto body jsou pak označeny za rohy.

Harris hranový detektor vylepšuje tento přístup hned v několika bodech:

1. Nevýhodu pouze diskrétních posunů jen v osmi směrech se podařilo odstranit následujícím způsobem.

$$\begin{aligned} E_{x,y} &= \sum_{u,v} w_{u,v}(I_{u+x,v+y} - I_{u,v})^2 \\ &= \sum_{u,v} w_{u,v}[xX + yY + O(x^2, y^2)]^2, \end{aligned}$$

   kde první gradienty odhadneme

$$\begin{aligned} X &= I \otimes (-1, 0, 1) = \frac{\delta I}{\delta x} \\ Y &= I \otimes (-1, 0, 1)^T = \frac{\delta I}{\delta y}. \end{aligned}$$

   Odtud lze E pro malé posuny zapsat takto

$$E_{x,y} = Ax^2 + 2Cxy + By^2,$$

   kde

$$\begin{aligned} A &= X^2 \otimes w \\ B &= Y^2 \otimes w \\ C &= (XY) \otimes w. \end{aligned}$$

2. Šum zapříčiněný čtvercovým binárním oknem lze potlačit kruhovým vyhlazovacím Gaussovským oknem

$$w_{u,v} = e^{-\frac{u^2+v^2}{2\sigma}}.$$

3. Problém s příliš snadnou záměnou hrany s rohem se podařilo také odstranit, u změny intenzity se neuvažuje prosté minimum, ale rozptyl.

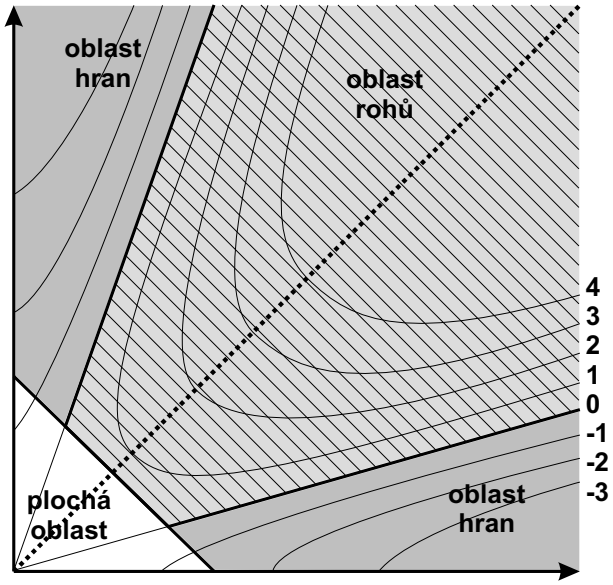
   Změna intenzity E lze pro malé posuny také zapsat jako

$$E_{x,y} = (x, y)M(x, y)^T,$$

   kde M je symetrická matice 2x2

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}.$$

   Takto lze na E nahlížet jako na lokální autokorelační funkci, která je určena maticí M. Nechť α, β jsou vlastní čísla matice M. Uvažujme prostor (α, β). Ideální hrana by zde měla vysoké α a nulové β (případně naopak). V reálném případě z důvodů šumu, vzorkování, kvantizace intenzity dosáhneme jen vysoké hodnoty v jednom z vlastních čísel a nízké ve



Obrázek 3.6: Reprezentace závislosti vlastních hodnot matice M na typu zkoumané oblasti a vrstevnice ohodnocovací funkce R.

druhém. Na druhou stranu v případě rohu v obrazu zaznamenáme vysoké hodnoty v obou vlastních číslech (viz obrázek 3.6).

Součástí Harris rohového detektoru je i funkce, která udává míru rohu/hrany v daném bodě, označme ji R. Pokud chceme zachovat rotační invariantnost, musí být funkcí vlastních čísel. Abychom se vyhnuli explicitnímu počítání vlastních čísel matice M, lze využít vztahů pro determinant a trace

$$\begin{aligned} det(M) &= \alpha\beta = AB - C^2 \\ tr(M) &= \alpha + \beta = A + B. \end{aligned}$$

Navržená ohodnocovací funkce pak vypadá takto

$$R = det(M) - \kappa\, tr(M)^2,$$

kde κ je nějaká vhodná konstanta (Harris uvádí 0.04). Na obrázku 3.6 jsou vidět jednotlivé vrstevnice funkce R. V oblasti rohů nabývá funkce kladných hodnot, v oblasti hran záporných a v místech bez větších změn intenzity hodnot blízkých nule. Za rohy pak označíme pixely, kde R dosahuje lokálního maxima ve všech osmi směrech. Na takto nalezené kandidáty lze samozřejmě uplatnit nějaké rozumné prahování, případně vybrat n nejvýraznějších rohů.

3.4 Implementace Tsai algoritmu

Existuje více implementací Tsai algoritmu pro kalibraci kamer, řada z nich se pokouší o optimalizace výpočtů, případně o zařazení některých dalších fází kalibrace, jako například získávání kalibračních dat. Asi nejznámější, velmi často používaná a proto i dobře otestovaná implementace vznikla na fakultě informatiky Carnegie Mellon University. Jejím autorem je Reg Willson [5].

Tento balík obsahuje rutiny nejen pro kalibraci kamer, ale zároveň je schopný vypočítaný model kamery využít pro další výpočty, například převod souřadnic v modelu kamery do souřadnic scény

nebo určení 3D souřadnice objektu v obraze kamery. Tyto a další výpočty jsem dále využil i při vlastním systému optického snímání pohybu.

Pro správnou funkci tohoto balíku je nutné nejprve zjistit parametry kalibrované kamery:

- Ncx počet elementů senzoru kamery v x-ovém směru
- Nfx počet pixelů ve frame bufferu v x-ovém směru
- dx velikost senzorového elementu kamery v x-ovém směru (v mm/sel)
- dy velikost senzorového elementu kamery v y-ovém směru (v mm/sel)
- dpx efektivní velikost pixelu v x-ovém směru (v mm/pixel)
- dpy efektivní velikost pixelu v y-ovém směru (v mm/pixel)
- Cx, Cy souřadnice středu paprskovitého optického zkreslení; průsečík promítací roviny kamery a z-tové osy souřadného systému kamery

Přesnost těchto proměnných nelze podcenit, jak jsem se sám mohl přesvědčit. Bez znalostí těchto nastavení bohužel nelze provést Tsai kalibraci s dostatečnou přesností tak, abychom dosáhli v následujících fázích snímání pohybu alespoň uspokojivých výsledků.

U kvalitních kamer jsou proměnné Ncx, dx a dy často uvedeny přímo v uživatelské dokumentaci. U low-end kamer, které se snažím využívat já při testování, nejsou tato data bohužel přímo dostupná. I přes několik pokusů kontaktovat přímo výrobce se mi je nepodařilo zjistit. Reg Wikson v dokumentaci ke své implementaci Tsai algoritmu uvádí, že přesné nastavení konstant není pro jednoduché převody mezi 2D pozicí v obraze a 3D světem potřeba. Ve většině případů tedy stačí v nastavení konstant respektovat pouze aspect ratio obrazu, tedy poměr mezi dpx a dpy. Dobrým odhadem je poměr skutečně nasnímané plochy ve scéně kamery k rozlišení obrazu:

$$\frac{s_x/w_x}{s_y/w_y},$$

kde s_x je rozlišení obrazu v horizontálním směru, s_y ve vertikálním směru, podobně pro w_x a w_y, kde se jedná o velikost nasnímaného prostoru například v milimetrech.

Takto naměřená hodnota by měla být poměrně dobrým odhadem pro dpx/dpy. Nyní můžeme za dpx dosadit nějakou pravděpodobnou hodnotu například $10\mu m$ a zpětně snadno dopočítat příslušnou dpy. Konstanty dx a dy určíme podobným způsobem. V podstatě určujeme jen jejich poměr. Dále položíme $Ncx = Nfx$, $sx = 1$ a Cx, Cy nastavíme na střed obrazu. Protože jsem používal kalibraci pro nekoplanární množinu kalibračních bodů, dojde při kalibraci i k zpřesňování proměnných sx, Cx a Cy. S takto odhadnutými vnitřními parametry kamery jsem byl při kalibraci schopný dosahovat rozumných výsledků.

Při vytváření množiny kalibračních bodů je nutné určit počátek souřadnic scény a orientaci os. Pro použití této implementace je nutné zvolit pravotočivý souřadný systém.

Jak už jsem zmiňoval, tato knihovna umožňuje kalibraci kamer na základě koplanární i nekoplanární množiny kalibračních bodů. Dále obsahuje prostředky pro urychlení kalibrace pomocí přednastavených vnitřních parametrů z předchozí kalibrace. V každé následující kalibraci téže kamery se již dopočítávají pouze externí parametry (pozice a rotace). Ve své práci jsem tuto funkčnost nevyužil, a proto ani netestoval, jak koplanární kalibrační body ani externí kalibraci, protože ze zadání mého problému jsem nebyl při kalibraci nijak časově omezen.

Pro základní kalibraci je potřeba buď koplanární množina bodů mohutnosti alespoň pět nebo nekoplanární množina o sedmi prvcích. Pro "plně optimalizovaný výpočet" je potřeba v obou případech nejméně jedenácti bodů.

Pro zpřesnění odhadu paprskovitého zkreslení a středu obrazu je potřeba volit kalibrační data tak, aby byla rovnoměrně distribuovaná přes celý obraz. Dále je potřeba neumisťovat střed souřadného systému scény do blízkosti středu obrazu kamery. Ovšem toto omezení lze snadno odstranit

pomocí jednoduchého numerického posunu souřadnic všech kalibračních bodů a tím vychýlit střed souřadného systému.

Další podmínky na kalibrační scénu jsou kladeny z důvodu odlišení vlivu ohniskové vzdálenosti čočky modelu dírkové kamery f od vlivu vzdálenosti kamery od scény Tz. Proto je potřeba mít kalibrační data rozmístěna v prostoru tak, aby se při jejich snímání dostatečně projevilo perspektivní zkreslení. Za vhodný rozdíl vzdáleností nejbližšího a nejvzdálenějšího kalibračního bodu od kamery považuji zhruba vzdálenost kamery od kalibrační scény. Pro lepší představu budiž příkladem koplanární kalibrační scéna, jejíž rovina je rovnoběžná s rovinou obrazu kamery, kolmá na hlavní osu promítání. V takovém případě je zřejmé, že nelze určit f ani Tz, pouze jejich poměr. V takovém případě se doporučuje úhel snímané roviny alespoň 30 stupňů.

3.5 Popis aplikace pro kalibraci kamer

Výsledkem mé snahy o kalibraci kamery je jednoduchá aplikace *cameraCalibration*, která slouží v podstatě jen jako generátor vstupních dat (uspořádaných pětic $(x_i, y_i, z_i, u_i, v_i)$) pro Tsai knihovnu.

Programu je nutné na vstupu předat soubor obsahující výše popsané informace o kalibrované kameře. Formát tohoto souboru jsem navrhl tak, aby byl snadno čitelný a editovatelný uživatelem. Příklad nastavení mojí kamery, které jsem používal při testování:

```
type=WebCam Pro
Ncx=320
Nfx=320
dx=0.013
dy=0.0133883
dpx=
dpy=
Cx=160
Cy=120
sx=1.0
```

Všechny parametry jsem popsal v předchozím textu. Jediné nové pole *type* označuje typ kamery, nemá žádný funkční význam, používá se pouze v Tsai knihovně pro chybové výpisy. Pokud nejsou, podobně jako v mém případě, parametry *dpx* a *dpy* určeny, jejich hodnoty se dopočítají podle vzorců:

$$\begin{aligned} dpx &= dx * Ncx/Nfx \\ dpy &= dy. \end{aligned}$$

Druhým povinným parametrem kalibrace kamery je cesta k souboru obsahujícího souřadnice kalibračních bodů v prostoru. V následující kapitole je krátký popis nástroje generujícího tato data pro mou kalibrační scénu.

Po spuštění kalibračního procesu se na obrazovce objeví záběr z kamery, ve kterém je bílými body vyznačen výsledek Harris rohového detektoru. Nyní je potřeba takto nalezené rohy identifikovat se zadanými kalibračními značkami. Tento proces jsem se nepokoušel nijak automatizovat. Uživateli je umožněno měnit za běhu nastavení hranového detektoru. Dva vstupní parametry lze ovládat pomocí kláves plus, minus a děleno, krát na numerické klávesnici. Změny o nastavení jsou vypisovány na standardní výstup. Uživatel je vyzván k označení postupně všech značek kliknutím myši na příslušný nalezený roh (viz obrázek 3.7).

Tímto postupem se získají dvojice 3D souřadnic značek v prostoru a jejich 2D souřadnic v obraze jako vstup pro Tsai knihovnu. Výsledek je zapsán do výstupního souboru, který lze poté využít jako vstup klientské části systému snímání pohybu popsaném v kapitole 4.

Při svých pokusech o kalibraci kamery jsem narazil na několik problémů. Výsledky mých prvních pokusů nebyly ani zdaleka uspokojivé. Přesnost kalibrace jsem odhadoval nejčastěji pomocí

Obrázek 3.7: Kalibrační scéna s nalezenými kalibračními body.

výpočtu umístění kamery v souřadnicích scény, tedy převod bodu $[0, 0, 0]$ v souřadném systému kamery do scénického a následném fyzickém přeměření skutečnosti. Ukolébán představou, že stačí 11 značek pro nekoplanární kalibraci, jsem zvolil pouze osm bodů z každé roviny. Dosahované výsledky byly velmi tristní. Trochu lepších výsledků jsem dosáhl, když jsem zvýšil počet kalibračních bodů. Ale opravdu uspokojivých výsledků jsem dosáhl, až když jsem začal uplatňovat všechna pravidla popsaná v odstavci 3.4. Především se jedná o dodržení správného hloubkového rozmístění značek v prostoru. Vzdálenost nejbližšího a nejvzdálenějšího bodu musí být obdobná jako vzdálenost kamery od nejbližší značky. Kameru jsem tedy musel více přiblížit ke kalibrační scéně. Problém pak byl zabrat všechny kalibrační body, proto jsem omezil množství bodů ve vertikální rovině jen na první řadu čtverců. Tímto způsobem jsem dosahoval odchylek v určení pozice kamery v rozmezí asi pěti procent.

Popis vstupních přepínačů *cameraCalibration*:

- x - horizontální rozlišení kamery (standardně *320*)
- y - vertikální rozlišení kamery (standardně *240*)
- d - název vstupního zařízení, název souboru reprezentujícího kameru (standardně */dev/video*)
- i - soubor s vnitřními parametry kamery
- c - soubor s kalibračními daty (povinný parametr)
- o - název výstupního souboru
- h - stručná nápověda

3.5.1 Pomocná rutina na generování vstupních dat kalibrace

Protože jsem testoval kalibraci kamer na řadě různých kalibračních scén s různými rozměry, neustálé a pracné přepočítávání souřadnic významných bodů v prostoru mě přinutilo vytvořit jednoduchý nástroj na automatické generování vstupních dat kalibrace. Program je určen pro kalibrační scénu, se kterou jsem dosahoval nejlepších výsledků, disjunktní černé čtverce v rovině (viz obrázek v dodatku A). Vstupními parametry jsou rozměry čtverců, jejichž rozestupy, odsazení od počátku souřadného systému a jejich počet. Protože jsem se setkal s drobnou nepřesností při rozpoznávání rohů čtverců, jejíž umístění bylo Harrisovým detektorem nalezeno mírně posunuté ke středu tmavého čtverce, přidal jsem ještě možnost významné body kalibrační scény posunout obdobným způsobem. Posunutí je určeno v procentech rozměru čtverců.

Kapitola 4

Snímání pohybu

Po úspěšné kalibraci alespoň dvou kamer můžeme přistoupit k samotnému snímání pohybu.

4.1 Hledání značek ve 2D obrazu

Ve většině systémů optického snímání pohybu se pro detekci významných bodů-značek používá prahování obrazu. Tento způsob jsem zvolil i já jako nejvhodnější a nejstabilnější metodu odlišení značek od pozadí. Výhodou je snadná implementace a nenáročnost na strojový čas, díky čemuž jsem neměl problémy upočítat tuto fázi snímaní pohybu v reálném čase. Nevýhodou jsou nároky kladené na osvětlení a barevné ladění celé scény. Je potřeba zvolit takové pozadí, aby významné body byly vždy výrazně světlejší než jejich okolí. V ideálním případě by bylo vhodné použít značky emitující světlo, případně silně reflexní plochy s dobrým osvětlením scény. Já jsem pro testování ve svých skromných domácích podmínkách používal bílé ping-pongové míčky, případně reflexní samolepky (běžná součást cyklistického vybavení). V každém případě bylo nutné mít pozadí snímaného objektu dostatečně tmavé.

Prvním krokem ve zpracování obrazu je tedy prosté prahování. Obraz se převede dle klasického vzorce z RGB modelu na černobílý:

$$0.299 * R + 0.587 * G + 0.114 * B.$$

A nyní stačí vhodně nastaveným prahem vytvořit binární obraz, jehož bílé části v ideálním případě reprezentují značky a černé pozadí.

4.1.1 Binární morfologie

Binární obraz vzniklý prostým prahováním velmi často vykazuje různé nežádoucí artefakty, způsobené například šumem nebo obecně nekvalitním snímacím prvkem v kameře. Pro jejich odstranění jsem s výhodou využil binárních morfologických algoritmů. Implementoval jsem dvě nejznámější operace matematické morfologie - dilatace a eroze. Obě operace jsou určeny strukturním elementem B (viz obrázek 4.1) a binárním obrazem X.

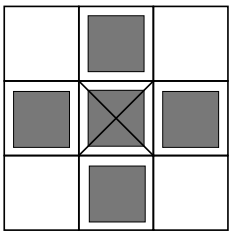
Dilatace $X \oplus B$ je bodovou množinou všech možných vektorových součtů pro dvojice pixelů, vždy pro jeden z množiny X a jeden z množiny B

$$X \oplus B = \{p \in \varepsilon^2 | p = x + b, x \in X, b \in B\}.$$

Dilatace se používá samostatně k zaplnění malých děr, úzkých zálivů a jako stavební kámen složitějších operací.

Eroze $X \ominus B$ je duální operací k dilataci, skládá dvě množiny podle předpisu

$$X \ominus B = \{p \in \varepsilon^2 | \forall b \in B : p + b \in X\}.$$



Obrázek 4.1: Strukturní element binární morfologie využívaný systémem.

Eroze se používá pro zjednodušení struktury objektů - objekty tloušťky 1 se ztrácejí, a tak se složitější objekty rozdělí na několik jednodušších.

Dilatace a eroze nejsou navzájem inverzní zobrazení. Jejich kombinací získáme další významné morfologické transformace - otevření a uzavření. Výsledkem obou je zjednodušený obraz obsahující méně detailů. Erozi následovanou dilatací označujeme jako *otevření* množiny X strukturním elementem B

$$X \circ B = (X \ominus B) \oplus B.$$

Otevření oddělí objekty spojené úzkou šíjí a tak zjednoduší strukturu objektů.

Dilatace následovaná erozí je uzavřením množiny X strukturním elementem B a je definována jako

$$X \bullet B = (X \oplus B) \ominus B.$$

Uzavření spojí objekty, které jsou blízko u sebe, zaplní malé díry a vyhladí obraz tím, že zaplní úzké zálivy.

Obě operace tedy slouží k odstranění detailů v obraze, čehož jsem využil při zpracování binárního obrazu vzniklého prahováním. Nejlepších výsledků při pokusech o různé kombinace výše popsaných operací jsem dosahoval při použití otevření i uzavření současně.

4.1.2 Řádkové semínkové vyplňování

Nyní, když máme oddělené značky od pozadí scény v binárním obrazu bez zbytečného šumu, potřebujeme k dokončení této fáze spočítat přesné souřadnice značky. Předpokládá se, že značky tvoří disjunktní oblasti bílé barvy v binárním obrazu a skutečná poloha je v těžišti této oblasti. Diskrétní výpočet těžiště tělesa lze vyjádřit například takto:

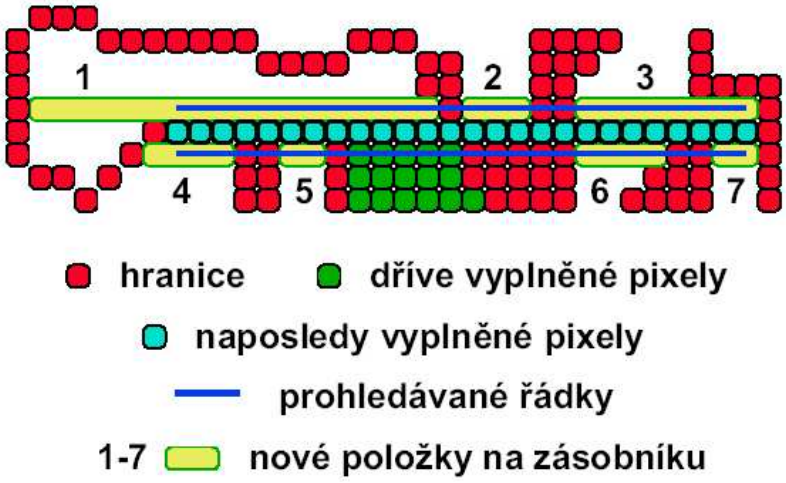
$$(x_t, y_t) = \frac{\sum_{i=1}^{n}(x_i, y_i)m_i}{\sum_{i=1}^{n} m_i},$$

kde (x_t, y_t) jsou souřadnice těžiště tělesa, skládajícího se z n částí po řadě s těžišti v souřadnicích (x_i, y_i) a hmotností m_i. Pro náš konkrétní případ budou jednotlivé elementy výpočtu všechny pixely příslušné značky o konstantní hmotnosti $m_i = 1$. Výsledný vzorec

$$(x_t, y_t) = \frac{\sum_{i=1}^{n}(x_i, y_i)}{n}$$

vyjadřuje počítaný průměr všech bodů reprezentující těžiště. Pro jeho výpočet je nutné nalézt všechny pixely značky. K tomu jsem využil algoritmus řádkového semínkového vyplňování [9, 10].

Rekurzivní naivní algoritmus vyplňování plochy v rastrovém obrazu jednoduše vybarví semínko a rekurzivně se zavolá na všechny své sousedy. Řádkový algoritmus nevyužívá přímo rekurzi, zásobník si staví sám, ale jeho velikost je mnohem nižší než velikost zásobníku při rekurzivním volání naivního algoritmu. Na zásobník se dávají celé řádkové úseky vyplňované plochy, tedy trojice $(minX, maxX, Y)$. V každém kroku se vyplní jeden úsek a na zásobník se přidají nové úseky, které se naleznou v bezprostředně přilehlých řádcích (viz obrázek 4.2). Vstup algoritmu semínko, se využije jako inicializační prvek zásobníku $(seminkoX, seminkoX, seminkoZ)$.



Obrázek 4.2: Schéma řádkového vyplňování (zdroj [10]).

Výhody tohoto algoritmu jsou především v jeho rychlosti a v nižších nárocích na velikost využité paměti.

4.2 Navržená architektura klient-server

Proces popsaný v předchozí kapitole se provádí pro každou kameru zvlášť a jeho výsledky je nutné synchronizovat a jednotně zpracovávat. Přestože nic na první pohled nebraní tomu, abychom vše prováděli v rámci jednoho počítače, rozhodl jsem se rozdělit celý proces snímání pohybu do jednotlivých aplikací.

Obsluha každé kamery je svěřena vždy jedné běžící aplikaci - klientovi. Sdružujícím a synchronizačním prvkem je server, na který se klienti připojují (viz obrázek 4.3). Komunikace mezi jednotlivými prvky je zajištěna pomocí TCP/IP spojení, což umožňuje rozmístění běžících komponent na více počítačů a zároveň je neomezuje v jejich integraci jen do jednoho centrálního počítače (viz obrázek 4.4).

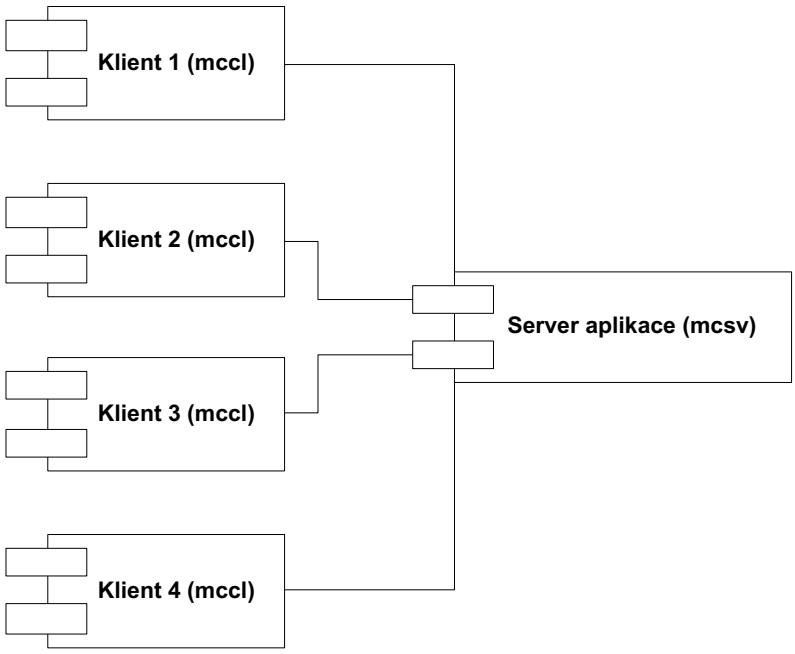
Všechny vyvinuté aplikace jsou konzolového typu. Tedy většina vstupních parametrů se předává přímo z příkazové řádky, případně pomocí konfiguračních souborů.

4.2.1 Klient

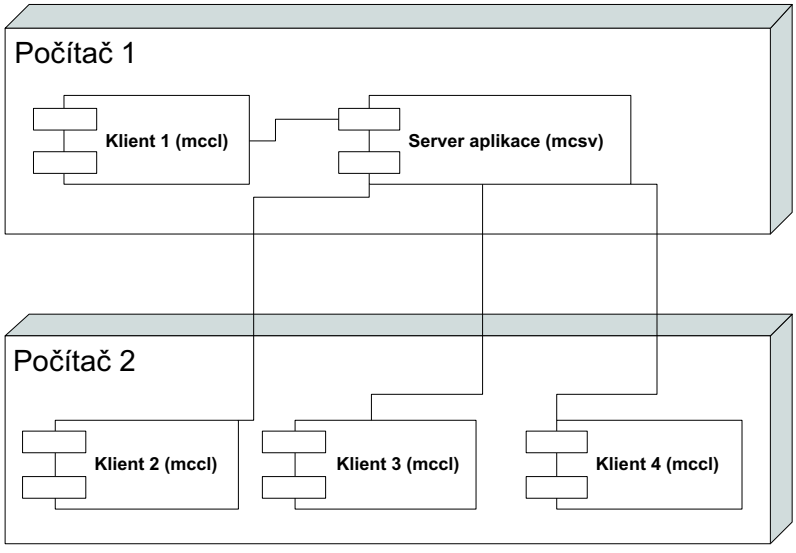
Klientská aplikace obsluhuje právě jednu kameru. Pro její spuštění je nutný již běžící server, jehož adresa je předána buď při spuštění klienta z příkazové řádky, nebo pokud adresu explicitně neuvedeme, klient se ho pokusí pomocí broadcastu najít v lokální síti automaticky. Po spojení klienta a serveru dojde k inicializační komunikaci, jejíž hlavním úkolem je oznámit serveru kalibrační data obsluhované kamery. Tato data jsou předána klientovi pomocí výstupního souboru aplikace pro kalibraci kamery.

Pro zobrazení výstupu z kamery využívám knihovnu Simple DirectMedia Layer (SDL - `http://www.libsdl.org/`), jejíž balíky je nutné mít pro bezproblémový běh systému nainstalované. Jedná se o nízkoúrovňovou multiplatformní multimediální knihovnu zpřístupňující kromě jiného 2D video framebuffer. Okno příslušné velikosti se objeví na obrazovce hned po spuštění klienta.

Jak už bylo naznačeno v předchozí kapitole, hlavním úkolem klienta je posílat data z nasnímaného obrazu serveru. Snahou bylo minimalizovat množství přenášených dat. V ideálním případě by šlo o přenos pouze 2D souřadnic značek. Tento cíl se nakonec podařilo splnit, a to především díky velmi nízkému rozlišení a obnovovací frekvenci dostupných kamer. Mezi jednotlivými obrazy



Obrázek 4.3: Klient-server architektura.



Obrázek 4.4: Možné rozmístění aplikací na počítače v lokální síti.

není problém upočítat algoritmus podrobně popsaný v kapitole 4.1 i na velmi průměrném hardwaru.

Okno zobrazující aktuální snímky z kamery už nemůže být samozřejmě využíváno na směrování kamery na pozorovaný cíl, protože v této fázi je kamera již zkalibrováná a data o její poloze jsou už zaslána i serveru pro další zpracování. Jediný úkol tohoto okna je v možnosti správně nastavit algoritmus na hledání značek, jehož jediným vstupním parametrem je hranice binárního prahování. Proto se do okna posílá obraz až po prahování a středy značek jsou zvýrazněny černým pixelem. Klávesy plus a minus jsou odchytávány pomocí SDL přímo v okně a informace o změně a aktuálním stavu prahu je vypisována do konzole. Cílem tohoto nastavení by mělo být správně odfiltrované pozadí od výrazných značek ve scéně. Nic nebrání úpravám prahu i v průběhu snímání a zasílání dat serveru.

Klient běží ve třech hlavních vláknech (plus vlákna obsluhovaná knihovnou SDL) - vlákno pro síťovou komunikaci, pro zpracování dat z kamery a jedno synchronizační vlákno. Síťové vlákno slouží především k příjímání synchronizačních zpráv od serveru. Vlákno na obsluhu kamery jednoduše snímá obraz a na každý snímek aplikuje algoritmus hledání značek v obrazu, a to bez ohledu na to, zda si server tato data vyžádal. Obraz upravený prahováním rozšířený o informace spočtených souřadnic posílá SDL knihovně. Synchronizační vlákno má jediný úkol - v okamžiku, kdy klient přijme žádost serveru o snímek z kamery, vezme informace nejbližšího spočítaného snímku a zašle je serveru. Podrobnosti synchronizace zasílání dat jsou uvedeny v následující kapitole.

Popis vstupních přepínačů klienta *mccl*:

- x - horizontální rozlišení kamery (standardně *320*)
- y - vertikální rozlišení kamery (standardně *240*)
- d - název vstupního zařízení, název souboru reprezentujícího kameru (standardně */dev/video*)
- i - soubor s kalibračními daty (povinný parametr)
- s - adresa serveru
- p - číslo portu serveru
- h - stručná nápověda

Příklad spuštění klienta *mccl -d /dev/video1 -i calibdata.txt -s localhost -p 12345.*

4.2.2 Server

Server je centrálním prvkem celého procesu snímání obrazu. Jeho běžící instance je nutná pro úspěšné spouštění klientů. Server si zaznamenává informace o všech připojených klientech, jedná se především o kalibrační informace příslušných kamer nutné pro pozdější zpracování naměřených dat.

Server od klienta může získat seznam 2D souřadnic značek v jednom obrazu. Hlavním úkolem serveru je tyto informace synchronizovat a zapisovat do výstupního souboru, který může být později použit pro rekonstrukci 3D souřadnic značek (viz následující kapitola).

Synchronizace spočívá v uspořádání příchozích dat do jednotlivých obrazů tak, aby si co nejpřesněji časově odpovídala. Toto je zajištěno pomocí výzev. Server na začátku vytváření obrazu nejprve zprávou vyzve všechny připojené klienty k zaslání nejaktuálnějších dat z jejich kamer. Klient po obdržení této výzvy v co nejkratším čase zašle aktuální naměřený stav 2D souřadnic značek. Server čeká na odpověď v podobě souřadnic od všech kamer, tato data zapíše do výstupního souboru a opakuje celý proces vytváření obrazu. Může se tedy stát, že vlivem pomalého síťového spojení nebo nižšího framerate jedné z kamer se nevyužijí data ze všech obrazů všech kamer. Navržený postup ale zajišťuje nutnou podmínku úspěšné detekce značek, kdy časové rozdíly mezi jednotlivými daty nejsou vyšší než $1/f$ sekund, kde f je snímková frekvence nejpomalejší kamery systému.

Činnost serveru zajišťuje opět několik vláken. Jedno vlákno odpovídá na broadcast dotazů klientů. Další vlákno se stará o vstup z klávesnice. Server je ovládán ze standardního vstupu pomocí několika příkazů - *start* spouští proces zaznamenávání dat, server začne rozesílat výzvy a příchozí data ukládat do výstupního souboru, *stop* tento proces snímání ukončuje a poslední rozpracovaný obraz zapíše do souboru, příkaz *exit* zastavuje běh serveru. Jedno z dalších vláken obsluhuje nově připojované klienty, vytvoří pro ně příslušné datové struktury a hlavně nastartuje nové vlákno, které obsluhuje především síťovou komunikaci s konkrétním klientem. Poslední, nikoliv však významem, je vlákno zajisťující součinnost všech klientských vláken a případné ukládání dat do souboru.

Příklad výstupního souboru je uveden v přílohách. Formát jsem navrhl tak, aby byl snadno čitelný nejen pro strojové zpracování, ale i pro člověka. Na prvním řádku souboru je vždy uveden počet kamer, které se účastnily snímání pohybu.

```
CameraCount: 2
```

Následuje blok kalibračních informací všech kamer, jejich formát jsem ponechal stejný, jako je používán v Tsai knihovně. Jedná se o 19 čísel, každé na jednom řádku. Jednotlivé kamery jsou odděleny jedním prázdným řádkem.

```
320.000000    //Ncx
320.000000    //Nfx
0.013000      //dx
0.013388      //dy
0.013000      //dpx
0.013388      //dpy
160.000000    //Cx
120.000000    //Cy
0.913305      //sx
5.556490      //f
-0.004061     //kappa1
70.568381     //Tx
-33.763495    //Ty
481.087563    //Tz
-2.503010     //Rx
-0.416255     //Ry
-0.340617     //Rz
0.000000
0.000000[1]
```

Dále jsou už vlastní data zaslaná od jednotlivých kamer v průběhu snímání pohybu. Každý obraz začíná řádkem udávajícím jeho pořadí. Následují řádky s 2D souřadnicemi značek, jeden řádek odpovídá informacím získaných z jedné kamery. Je zde uveden počet nalezených značek a jejich souřadnice.

```
Frame: 1
marksCount: 1 x = 182.787804 y = 147.869018
marksCount: 1 x = 131.484481 y = 173.380941
```

Popis vstupních přepínačů serveru *mcsv*:

- p - číslo portu, na kterém očekává klienty (standardně *12345*)
- o - výstupní soubor (standardně *out.mcd*)
- h - stručná nápověda

Příklad spuštění serveru *mcsv -p 12345 -o output.mcd.*

[1] Význam jednotlivých polí je popsán v kapitolách 3.4 a 4.3.1

4.3 Postprocessing naměřených dat

Postprocessing je poslední fáze optického snímání pohybu. Protože se jedná o poměrně výpočetně náročnou operaci, u které je navíc velmi často nutný zásah operátora, provádí se až po získání kompletních dat. Obecně je do postprocessingu zahrnován i proces identifikace jednotlivých značek s kostrou předlohy a případné nutné výpočty spojené s deterministickou nepřesností naměřených dat způsobenou umístěním značek mimo středy kloubních spojení atp. Já jsem se ve své práci omezil pouze na rekonstrukci 3D souřadnic značek.

Vstupní data této fáze zpracování kromě samotných 2D souřadnic nalezených značek obsahují samozřejmě i informace získané při kalibraci kamer. Z nich lze snadno spočítat umístění kamery, přesněji středu promítání dírkového modelu, v souřadnicích scény. Postup tohoto výpočtu je naznačen v kapitole 4.3.1 v bodě 6 aplikovaný na vstup $[0, 0, 0]$. Pro 2D souřadnice značky v obraze lze, tentokrát podle bodu 8, vypočítat jeho reprezentaci v prostoru scény. Je zřejmé, že značka reprezentovaná těmito 2D souřadnicemi leží někde na polopřímce určené souřadnicemi kamery a její reprezentací v prostoru scény. Pokud chceme určit přesné souřadnice značky v prostoru, je nutné mít danou značku zachycenou v záznamu alespoň dvou kamer. Z každého takového záznamu získáme polopřímku možného umístění značky a pokud nejsou kamery umístěny nevhodným způsobem, například kolineární umístění kamer a značky, pak průnik polopřímek jednoznačně určuje umístění významného bodu ve scéně. Toto by platilo v ideálním případě, ale protože v průběhu celého procesu snímání pohybu pracujeme s nepřesnými daty, v obecném případě k průniku těchto polopřímek dojde jen velmi zřídka. Proto je nutné i zde tolerovat jistou chybu a za průnik považovat i paprsky míjející se "dostatečně" těsně. Pokud je značka zachycena na záznamech více kamer, lze toho s výhodou využít pro zvýšení přesnosti výpočtu.

Aplikace následného zpracování funguje jako filtr. Na vstupu očekává data generovaná serverovou částí systému, popsanou v předchozí kapitole, a na výstup zapisuje data v trc datovém formátu popsaném v kapitole 5.2. V hlavičce těchto dat jsou informace o počtu kamer účastnících se optického snímání pohybu a především jejich kalibrační informace. Pro každou kameru vzniká datová struktura obsahující tyto hodnoty. Protože Tsai kalibrační knihovna nepředpokládá výpočty nad více než jednou kamerou současně, je nutné její globální proměnné definující vlastnosti kamery předpočítat do vlastních datových struktur a při volání vnitřních metod knihovny tyto proměnné nastavovat podle kamery, nad kterou chceme výpočet provádět. Kromě těchto proměnných si do vlastních vnitřních struktur v inicializační části běhu předpočítám i souřadnice kamery v souřadném systému scény, z důvodů naznačených v předchozím odstavci. Druhou fází běhu algoritmu je už vlastní zpracování vstupních 2D souřadnic značek. Každý obraz se zpracovává individuálně. Pro každou z kamer se spočítají všechny paprsky odpovídající značkám nalezených v obraze. Pro každou dvojici paprsků z dvou různých kamer se spočítá jejich průnik. Průnikem se zde nemyslí přímo jejich geometrický průnik, protože ten nastává jen velmi zřídka i v případě, že paprsky označují stejnou značku ve scéně, ale bod v prostoru ležící nejblíže oběma polopřímkám. Při jeho výpočtu je nutné najít dva body, každý na jedné z polopřímek, které mají minimální vzdálenost od druhého paprsku (viz obrázek 4.5) . Označme je P_1 a P_2. Polopřímky nechť jsou definovány bodem A resp. B a vektorem V resp. W. Není obtížné nahlédnout, že úsečka určená hledanými body musí být kolmá na obě polopřímky a splňovat tedy

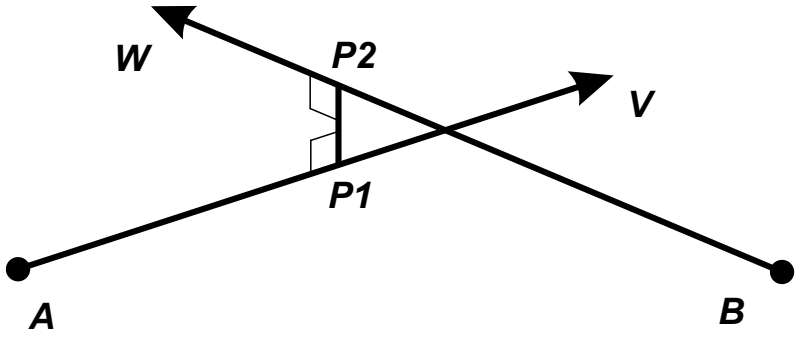
$$\begin{aligned}(P_2 - P_1) \bullet V &= 0 \\ (P_2 - P_1) \bullet W &= 0.\end{aligned}$$

Pro parametrické vyjádření bodů P_1 a P_2 máme soustavu dvou rovnic pro dvě neznámé

$$\begin{aligned}(B + t \cdot W - (A + s \cdot V)) \bullet V &= 0 \\ (B + t \cdot W - (A + s \cdot V)) \bullet W &= 0.\end{aligned}$$

Z nichž už snadno odvodíme vztahy pro hledané body

$$s = \frac{(A \bullet W - B \bullet W) \cdot (W \bullet V) + (B \bullet V - A \bullet V) \cdot (W \bullet W)}{(V \bullet V) \cdot (W \bullet W) - (V \bullet W)^2}$$



Obrázek 4.5: Vzdálenost dvou mimoběžek.

$$t = \frac{(B \bullet V - A \bullet V) \cdot (W \bullet V) + (A \bullet W - B \bullet W) \cdot (V \bullet V)}{(V \bullet V) \cdot (W \bullet W) - (V \bullet W)^2}.$$

Kromě bodů P_1 a P_2 se pro další výpočty předpočítává i střed úsečky jimi určené. Dále se paprsky všech kamer shlukují do skupin tak, aby ve skupině byl nejvýše jeden paprsek z každé kamery a zároveň vzdálenost bodů P_1, P_2, příslušející každé dvojici z množiny, byla menší než předem určený práh. Pro každou takto nalezenou skupinu paprsků se spočítá průměr množiny středů všech dvojic. Ten označíme za odhad pozice hledané značky ve scéně.

Poslední fází zpracování každého obrazu je zápis nalezených souřadnic do výstupního souboru. Data jsou zapisována ve formátu trc podrobněji popsaném v kapitole 5.2.

Popis vstupních přepínačů postprocessingu *mc*:

- i - vstupní soubor s 2D souřadnicemi značek, výstup serverové části systému
- o - výstupní soubor trc formátu snímaného pohybu
- t - práh pro shlukování paprsků do množin určujících značku (standardně *30* (mm))
- h - stručná nápověda

Příklad spuštění postprocessingu *mc -i input.mcd -o output.trc.*

4.3.1 Transformační funkce dostupné v Tsai kalibrační knihovně

Tsai kalibrační knihovna obsahuje kromě samotných kalibračních rutin i řadu transformačních funkcí, které jsem s výhodou využil v postprocessingu nasnímaných dat. Všechny tyto funkce využívají strukturu konstant zadaných uživatelem a především pak výsledek samotné kalibrace.

- f ohnisková vzdálenost dírkového modelu kamery
- $kappa1$ první koeficient paprskovitého optické zkreslení
- Cx, Cy souřadnice středu paprskovitého optického zkreslení; průsečík promítací roviny kamery a z-tové osy souřadného systému kamery
- sx konstanta vyjadřující chybu při vzorkování v horizontálním směru vůči směru vertikálnímu
- R matice rotace mezi souřadným systémem scény a kamery
- T vektor vzájemného posunu středu souřadného systému scény a kamery
- dpx, dpy velikost pixelu ve frame bufferu (v mm/pixel)

Nyní bude následovat seznam několika transformačních funkcí, u kterých bude většinou i matematické vyjádření výpočtu (následující text vychází ze zdrojového kódu knihovny):

1. *Převod ze zkreslených souřadnic na nezkreslené souřadnice senzoru kamery.* Vzorec pro tuto operaci je zmíněn již v teoretických kapitolách o kalibraci kamery

$$\begin{aligned} X_u &= X_d(1 + kappa1\left(\sqrt{X_d} + \sqrt{Y_d}\right) \\ Y_u &= Y_d(1 + kappa1\left(\sqrt{X_d} + \sqrt{Y_d}\right), \end{aligned}$$

   kde souřadnice označené indexem d jsou zkreslené a souřadnice s indexem u jsou nezkreslené.

2. *Převod z nezkreslených na zkreslené souřadnice senzoru kamery.* Jedná se o inverzní operaci předchozí transformace. Její řešení vede k problému nalezení kořenů polynomiální rovnice třetího řádu. K odvození autor použil Cardanovu metodu [12].

3. *Převod z nezkreslených na zkreslené souřadnice v obrazu.* Řešení této podúlohy spočívá v převedení na předchozí problém. Nejdříve je potřeba převést souřadnice obrazu na souřadnice senzoru

$$\begin{aligned} X_u &= dpx\left(X_{fu} - C_x\right)/sx \\ Y_u &= dpy\left(Y_{fu} - C_y\right), \end{aligned} \tag{4.1}$$

   kde souřadnice označené indexem fu jsou nezkreslené souřadnice v obrazu. Na takto získané souřadnice stačí už jen zavolat transformaci 2 a převést zpět do obrazových souřadnic

$$\begin{aligned} X_{fd} &= X_d sx/dpx + C_x \\ Y_{fd} &= Y_d/dpy + C_y. \end{aligned} \tag{4.2}$$

   Tyto vzorce přímo vycházejí z definice proměnných dpx, dpy, C_x, C_y a sx.

4. *Převod ze zkreslených na nezkreslené souřadnice v obrazu.* Jedná se opět o pouhou inverzní operaci předchozí transformace. Nejdříve se převedou zkreslené souřadnice obrazu na zkreslené souřadnice senzoru obdobně jako v 4.1. Dále se využije transformace popsaná v 1 k převodu na nezkreslené souřadnice senzoru a nakonec zpět do obrazových souřadnic pomocí 4.2.

5. *Převod ze souřadnic scény do souřadnic kamery.* Tento převod je jednoduše implementován řešením rovnice

$$(X, Y, Z)_C = R(X, Y, Z)_W + T,$$

   kde indexem C jsou označeny souřadnice kamery a indexem W jsou označeny souřadnice ve scéně.

6. *Převod ze souřadnic kamery do souřadnic scény.* Algebraicky je to opět pouze inverzní operace k předchozí transformaci. Dle komentářů ve zdrojovém kódu byl inverz získán pomocí programu Macsyma[2].

7. *Převod ze souřadnic scény na obrazové souřadnice.* Pomocí 5 převedeme souřadnice scény do souřadného systému kamery $(X, Y, Z)_W \rightarrow (X, Y, Z)_C$. Převod do senzorových souřadnic lze přímo ze shodnosti trojúhelníků zapsat takto

$$\begin{aligned} X_u &= f\frac{X_C}{Z_C} \\ Y_u &= f\frac{Y_C}{Z_C}. \end{aligned}$$

   Dále pomocí transformace popsané v 2 odstraníme zkreslení $(X_u, Y_u) \rightarrow (X_d, Y_d)$ a převedeme do obrazových souřadnic (viz 4.2).

[2] http://www.scientek.com/macsyma/mxmain.htm

8. *Převod z obrazových souřadnic na souřadnice scény.* Postup je, jak už název napovídá, inverzní obdobou předchozího problému. Zároveň je zřejmé, že takto zadaná úloha nemá jednoznačné řešení. Řešením je celá množina bodů ležících na přímce procházející středem promítání. Proto je nutné vstup rozšířit například o z-tovou souřadnici hledaného bodu. Prvním krokem je přechod od obrazových do senzorových souřadnic (viz 4.1). Výpočet inverzní operace zbývajících dvou kroků z předchozí transformace byl opět získán pomocí Macsyma.

Kapitola 5

Motion capture formáty dat [6]

V praxi je možné se setkat s celou řadou výstupních datových formátů jako výsledkem procesu sledování pohybu. Řada z nich je standardizována a jejich podpora je implementována v 3D programech typu Softimage, Maya, 3D Studio Max atp. Zároveň tyto programy umožňují snadný převod z jednoho formátu na druhý.

Data z optického snímání pohybu se v průběhu zpracování mění. Z původního nasnímaného obrazu se extrahují 2D souřadnice značek. Ty se pak převádějí na 3D souřadnice za pomocí výstupu z více kamer. V tuto chvíli máme pro každou značku 3D souřadnice a jejich změny v čase, změny jsou pouze translace ve společné kartézské soustavě, proto se tato data někdy také označují jako globální translační data. Pro tento typ dat zde uvedu jeden příklad formátu jejich uložení. Ale pro praktické využití nasnímaného pohybu je většinou potřeba globální translační data dále upravit tak, aby odpovídala požadované kostře. Pro takto upravená data existuje již celá řada více či méně standardizovaných formátů, z nichž jeden v krátkosti také popíši. Přístup k ukládání se i zde dále dělí na hierarchický a globální. Hierarchický vychází z kořenového bodu, nejčastěji bodu ve středu trupu kostry, který je určen translací a všechny zbylé segmenty kostry jsou určeny už jen svou lokální rotací a délkou vzhledem ke svému předku v rámci hierarchie kostry. Globální přístup rezignuje na přirozenou hierarchii kostry, rotace a posun všech segmentů jsou určeny nezávisle.

Nejčastěji používaný formát optického snímání pohybu dneška je asi Acclaim, kombinace .amc a .asf souborů. Další formáty, na které můžete narazit, jsou Biovision (.bva a .bvh) a Motion Analysis (.trc a .htr). Mnoho společností si vytváří další proprietární datové typy, které co nejlépe vyhovují jejich konkrétním potřebám.

5.1 Acclaim datový formát

Tento datový formát, jak už název napovídá, vznikl v Acclaim a Biomechanics pro jejich vlastní optický systém snímání pohybu. V dnešní době je to asi nejrozšířenější formát podporovaný snad všemi animačními aplikacemi, jako jsou Maya, Alias, Softimage, 3D Studio Max, Nichimen, Prisms a Houdini. V mnoha optických MC systémech se stal primárním formátem.

Skládá se ze dvou souborů, Acclaim skeleton format (.asf) a Acclaim motion capture (.amc). První z nich obsahuje definici kostry, druhý pak samotná data jejího pohybu. Proto k jednomu .asf souboru může příslušet více .amc datových souborů.

.asf soubor se dělí na osm sekcí: *version*, *name*, *units*, *documentation*, *root*, *bonedata*, *hierachy* a *skin*. Podrobněji v následujícím příkladu se stručným komentářem.

```
# AST/ASF file generated using VICON BodyLanguage
//komentář je uveden znakem # a může být kdekoliv souboru
:version 1.10 //verze datového formátu
:name VICON //jméno kostry
:units //sekce s jednotkami
```

```
 mass 1.0 //multiplikátory pro hmotnost a délku
 length 0.45
 angel deg //úhly budou ve stupních (rad = radiány)
:documentation //dokumentace ke kostře
 .ast/.asf automatically generated from VICON data using
 VICON BodyBuilder and BodyLanguage model BRILLIANT.MOD
:root //informace o kořenovém uzlu
 order TX TY TZ RX RY RZ //pořadí transformací kořene
 axis XYZ //orientace rotace
 position 0 0 0 //pozice v globálních souřadnicích
 orientation 0 0 0 //natočení
:bonedata //definice všech segmentů vůči kořenovému uzlu
 begin //začátek definice prvního segmentu (kosti)
  id 1 //identifikátor segmentu (volitelný)
  name lowerback //jméno segmentu
  direction 0 1 0 //směrový vektor segmentu
  length 2.07313 //délka segmentu
  axis 0 0 -5.3486 e-035 XYZ
                //globální orientace os a jejich pořadí
  dof rx ry rz //povolené stupně volnosti a jejich pořadí
  limits (-inf inf) //meze jednotlivých stupňů volnosti
         (-inf inf)
         (-inf inf)
  bodymass 3.0 //hmotnost segmentu
  cofmass 1.0 //těžiště segmentu
 end //konec definice prvního segmentu
 begin
  id 2
  name lfemur
  ...
 end
 ...
:hierarchy //definice samotné hierarchie segmentů
 begin //první je vždy předek následovaný jeho potomky
  root lhipjoint rhipjoint lowerback
  lhipjoint lfemur
  lfemur ltibia
  ltibia lfoot
  lfoot ltoes
  ...
 end
:skin <filename> //seznam 3D modelů příslušejících této kostře (volitelný)
      <filename>
      ...
```

Řada položek bývá různými programy také různě interpretována, lze se setkat i s řadou rozšíření podle typu užití datového formátu. Podrobnější informace lze nalézt například v [6].

Druhý .amc soubor již obsahuje konkrétní data pohybu příslušející kostře definované v .asf souboru. Způsob ukládání těchto dat bude nejlépe vidět z následujícího příkladu.

```
:FULLY-SPECIFIED //typ formátu dat
:DEGREES //jednotky rotace
1 //číslo framu
root 29.6858 15.6774 16.4701 -174.691 -28.7904 -179.576
```

```
lowerback -0.545052 -0.0306724 0.526889
upperback 0.478258 -0.0474607 030806175
...
2
root 29.6836 15.6803 16.4578 -174.717.717 -29.1093 -179.366
lowerback -0.582749 -0.0504433 0.0860232
upperback 0.438491 -0.076839 0.06367
...
```

Soubor je rozdělen k jednotlivým framům příslušejících částí uvozených jejich číslem. V každé sekci jsou pak vyjmenovány všech segmenty kostry se svojí aktuální polohou. Pro kořenový uzel jsou to data v pořadí určeném polem *:root order TX TY TZ RX RY RZ* v .asf souboru. Pro ostatní uzly v hierarchii kostry se udávají data v pořadí jejich stupňů volnosti (*:bonedata dof rx ry rz*).

5.2 .trc datový formát

Tento formát vznikl pro Motion Analysis optical MC systém. Je typickým příkladem skladu jen translačních dat. První část souboru je hlavička s obecnými informacemi, jako například frekvence nasnímaných dat v souboru, případně frekvence v originálním snímání, počet framů a značek ve scéně, jednotky, ve kterých se udávají souřadnice, atp. Jak už jsem naznačil, hlavička neobsahuje jen informace o datech v aktuálním souboru, ale i datech, ze kterých byl tento soubor vygenerován.

```
PathFileType 4 (X/Y/Z) /usr/data/trc/sample.trc
DataRate CameraRate NumFrames NumMarkers Units
30.0     60.0       55        24         mm
OrigDataRate OrigDataStartFrame OrigNumFrames
60.0         0.0                600
```

Z předchozího příkladu hlavičky lze například vyčíst, že data mají aktuálně 30 obrázků za sekundu, soubor obsahuje 55 obrázků a jednotkou jsou milimetry. Originální data byla ovšem nasnímána v 60 fps a obsahovala 600 framů.

Bezprostředně za hlavičkou následuje sekce se samotnými daty, organizovanými do sloupců. První sloupec je číslo obrázku, druhý čas vzniku obrázku, dále jsou již jen trojice souřadnic jednotlivých značek v globálních souřadnicích.

```
Frame# Time  HeadTop                    HeadLeft
             X1      Y1       Z1       X2      Y2       Z2
1      0.817 230.937 1208.980 -574.766 334.829 1166.965 -594.169 ...
2      0.85  240.007 1210.762 -569.765 340.597 1167.553 -589.941 ...
3      0.883 247.311 1213.39  -561.436 350.318 1165.927 -577.866 ...
...
```

Tento příklad má tedy celkem 74 sloupců, souřadnice pro 24 značek plus čas a číslo framu. To, že časy obrázků začínají až od 0,817, má zpětný vztah k originálním datům. Znamená to, že tento převzorkovaný výřez dat začíná v originálním zdroji framem s číslem 49. Existuje zde tedy možnost zpětného dohledání původních naměřených dat.

Kapitola 6

Závěr

Cíle vytyčené v zadání diplomové práce, tedy prostudovat problematiku optického snímání pohybu a vytvořit funkční systém provozuschopný i za velmi špatných podmínek určených především levnými webovými kamerami, které jsem měl k dispozici pro testování, se až na drobné nedostatky popsané níže, podařilo splnit.

Prvním úkolem, který bylo nutné vyřešit na cestě zpracování dané problematiky, bylo zprovoznění zapůjčených testovacích webových kamer. První pokusy pod operačním systémem Windows XP byly úspěšné pouze z uživatelského hlediska, nikoli však z programátorského. Při vyhledávání informací o možnostech práce s videozařízením pod Windows jsem narazil na problémy s připojením více kamer stejného typu k jednomu počítači. Originální ovladače nejsou schopny spravovat současně více než jednu kameru. Po prostudování možností, které mi nabízejí jiné OS, jsem se rozhodl pro Linux, konkrétně pro distribuci SUSE, kde jsem na rozdíl od jiných distribucí zaznamenal jen minimální nedostatky na uživatelské úrovni. A především mi nic nebránilo připojit v podstatě neomezené množství kamer. Dále jsem měl k dispozici volně dostupné rozhraní pro práci s videozařízeními Video4Linux. Toto rozhraní mi umožňovalo bezproblémové čtení dat přímo z kamery. Přestože linuxový ovladač nedosahoval výsledků svého originálního protějšku z Windows, fakt, že jsem schopný ovládat více kamer najednou, byl pro mojí další práci zásadní.

První pokusy o získání jakýchkoli materiálů na téma optické snímání pohybu povětšinou vedly na dokumenty spíše marketingového charakteru, případně na velmi povrchní popis myšlenky snímání pohybu. První úspěchy jsem zaznamenal až při vyhledávání informací o problému, který je skrytý zdánlivě na pozadí samotného optického snímání pohybu. Jedná se o kalibraci kamery. V osmdesátých letech minulého století vznikl asi nejznámější algoritmus kalibrace kamery prezentovaný R. Y. Tsaiem v roce 1986 [4], se kterým jsem se podrobně seznámil a popsal jeho hlavní myšlenku v kapitole3.2. Ve své navazující práci jsem využíval volně dostupnou implementaci Tsai algoritmu od Rega Willsona [5].

Po úspěšné kalibraci již víme, kde jsou kamery rozmístěny, známe jejich orientaci a vnitřní optické vlastnosti. Řešení snímání pohybu je už relativně přímočará myšlenka, kterou jsem se pokusil realizovat v systému popsaném v kapitole 4.

Nejdříve bylo ovšem nutné alespoň částečně zautomatizovat přípravu kalibračních dat nutných pro vstup kalibračního algoritmu. Touto fází příprav jsem se zabýval poměrně dlouho. Bylo nutné navrhnout kalibrační scénu s vhodně rozmístěnými významnými body, které je možné snadno automaticky rozpoznávat. V konečné fázi jsem pro identifikaci využil Harris rohový detektor (viz kapitola 3.3.2) a scénu o dvou kolmých rovinách s černými disjunktními čtverci (viz obrázek 3.7) s důrazem na větší počet rozpoznávaných bodů v horizontální rovině scény, a to z důvodu zvětšení hloubkového rozptylu bodů nutného pro naměření dostatečného perspektivního zkreslení. Nevýhodou mé kalibrační aplikace zůstává nutná asistence uživatele z důvodu identifikace významných bodů se značkami nalezenými rohovým detektorem. Pro přípravu množiny souřadnic kalibračních bodů v mnou navrhované scéně jsem vytvořil jednoduchý nástroj generující 3D souřadnice bodů ve scéně. Přesto plná automatizace celého procesu není bohužel zajištěna. Dosahované výsledky kalibrace byly přes počáteční velké nedostatky, s ohledem na kvalitu snímacích

prvků a k nim dostupných informací, uspokojivé. Při správném umístění mé kalibrační scény a tedy správném rozmístění kalibračních bodů v prostoru jsem dosahoval odchylek v odhadnutých souřadnicích kamer 5% vzdálenosti kamery od počátku souřadného systému.

Teprve po úspěšné kalibraci alespoň dvou kamer jsem mohl přistoupit k pokusům o vlastní optické snímání pohybu. Nejprve bylo nutné vyřešit synchronizaci naměřených dat z obecně různých kamer a zároveň dostát podmínkám zadání diplomové práce a umožnit využití více sítově propojených počítačů. Rozhodl jsem se proto pro klasickou klient-server aplikaci, kde každá kamera je reprezentována právě jedním klientem komunikujícím s centrální synchronizační serverovou částí systému. Abych minimalizoval velikost po síti přenášených dat, umístil jsem logiku rozpoznávání 2D souřadnic značek v obrazu kamery do klientské části systému a směrem k serveru už nepřenášel surová obrazová data. Oddělení značek od pozadí scény jsem dosáhl pomocí tří výpočetně jednoduchých kroků - prahováním, vyhlazení binárního obrazu algoritmy známe z matematické morfologie a spočítáním těžiště nalezených značek. Již s takto jednoduchým návrhem jsem dosahoval dostatečně kvalitních výsledků a výpočetní nenáročnost zajišťuje dostatečnou časovou rezervu v real-time zapracovaní obrazu klientem. Synchronizaci naměřených dat zajišťuje server pomocí výzev, na které klienti reagují zasíláním množiny nalezených značek v aktuálním snímku. Tento způsob komunikace umožňuje mít v systému kamery s různou obnovovací frekvencí. Server nad přijatými daty už žádné výpočty nedělá, pouze je uloží do souboru, který slouží jako podklad pro následný postprocessing.

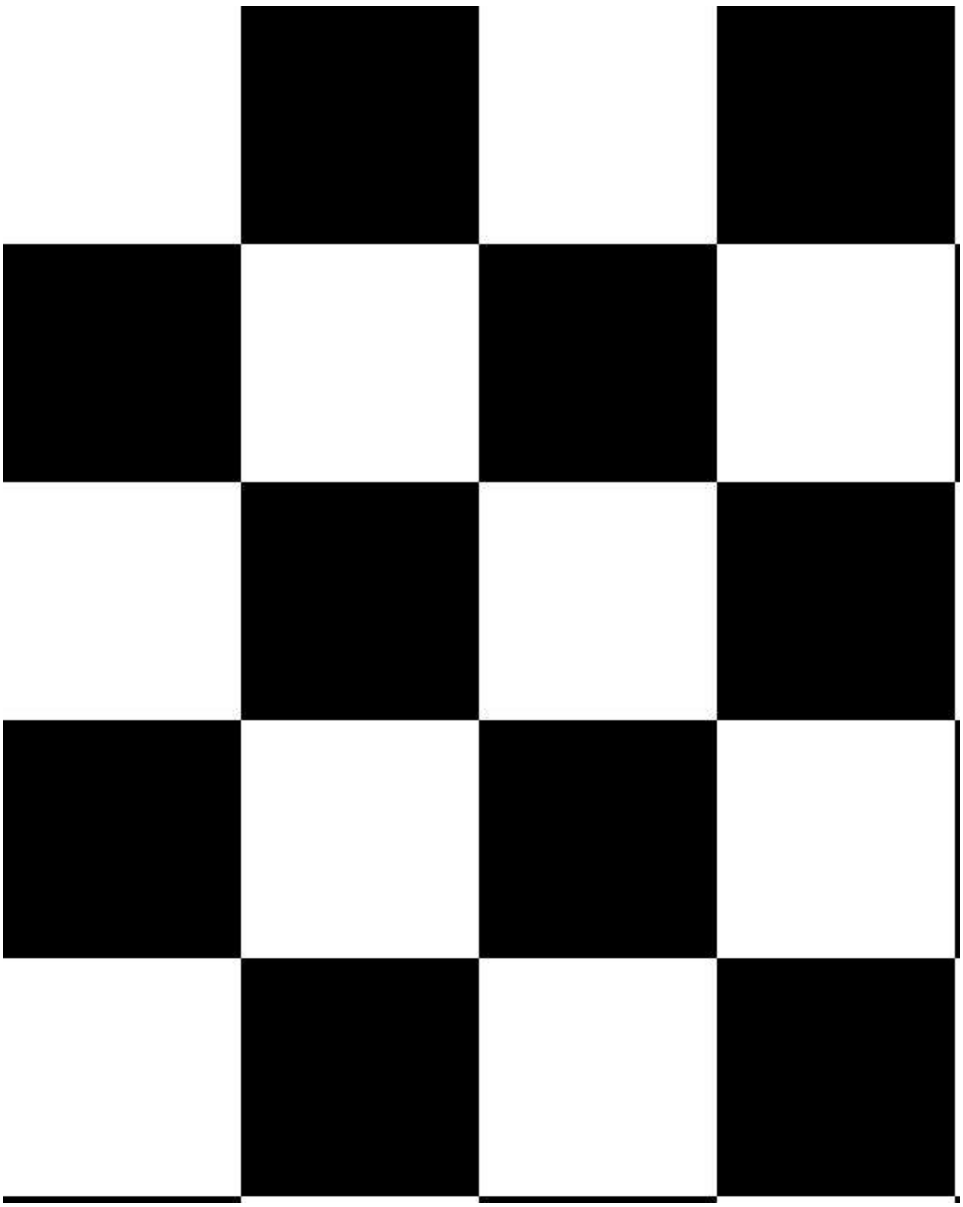
Protože poslední fáze zpracování již synchronizovaných dat je obecně velmi výpočetně náročný úkol a velmi často je nutný uživatelský zásah do procesu, neprovádí se v reálném čase s měřením dat, ale až po skončení snímání celé scény, tzv. postprocessing. Já jsem se ve své práci omezil jen na vyřešení základního úkolu této fáze, a to na rekonstrukci 3D souřadnic nalezených značek. V kapitole 4.3.1 jsem popsal odvození několika geometrických výpočtů, které jsem využil při rekonstrukci pozice značek. Myšlenka spočívá v předpočítání všech paprsků reprezentujících nalezené souřadnice značek a hledání shluků paprsků z různých kamer míjejících se v dostatečně malé vzdálenosti (podrobněji v kapitole 4.3). Výsledná data se zapisují ve formátu trc (kapitola 5.2) do výstupního souboru.

V průběhu psaní celého systému optického snímání pohybu jsem narážel na celou řadu problémů, z nichž se mi ne všechny podařilo zcela vyřešit. Z těch významných nedostatků bych jmenoval nízké rozlišení a především nízkou obnovovací frekvenci dosaženou u testovacích kamer, zapříčiněnou využitím neoficiálních ovladačů dostupných pod OS Linux. Dále bych zmínil nutnost asistence uživatele při kalibraci kamer, jejíž řešení by znamenalo implementovat další netriviální algoritmy nesouvisející přímo s danou problematikou. Za nedostatek by asi šla považovat i přímočarost výpočtů v postprocessingu, kde jsem si ale již od počátku nekladl velké ambice pro přímé nasazení do praktického využití. Zde bych viděl asi největší prostor pro navazující práci, jejímž cílem by mohla být především identifikace značek v čase a jejich mapování na předem popsanou kostru snímané předlohy, což by, s případným využitím kvalitnějších kamer, byl poslední krok k aplikaci navržených postupů v praxi.

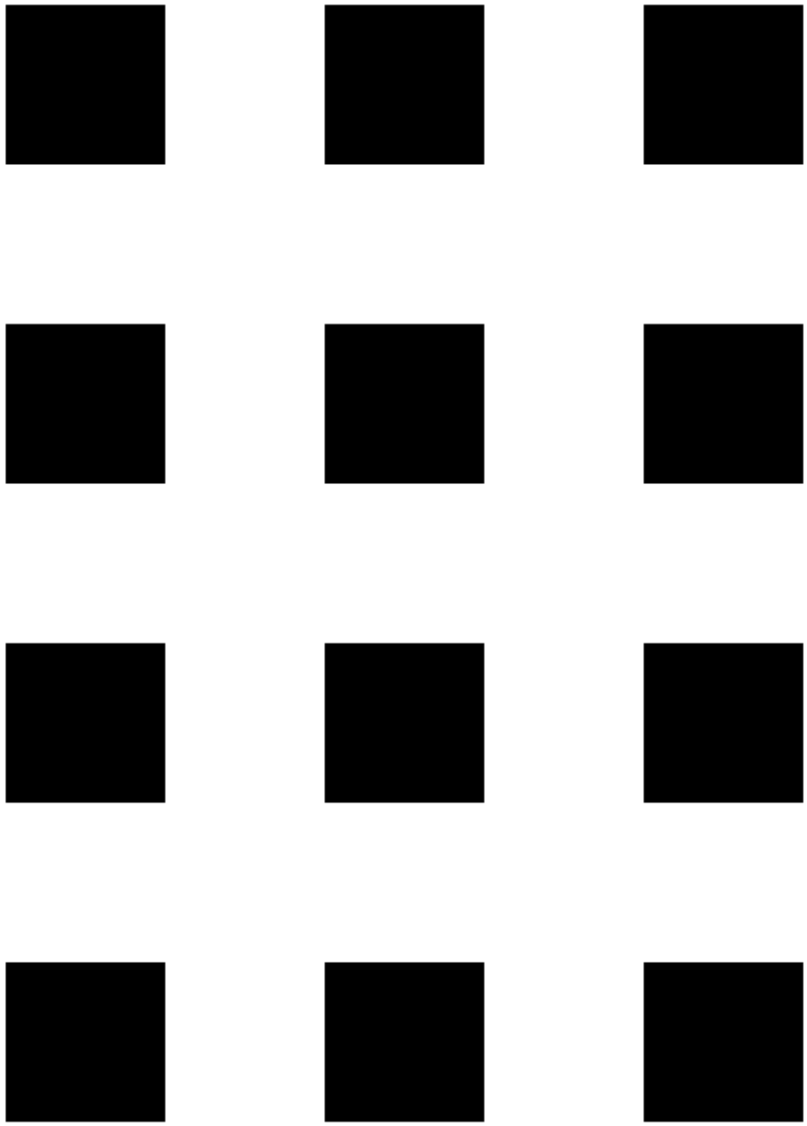
Dodatek A

Typy kalibrační scény

- jeden z prvních pokusů o sestavení kalibrační scény



- výsledný návrh kalibrační scény, s jejímž rozpoznáváním jsem dosahoval nejlepších výsledků



Seznam obrázků

Literatura

[1] Abdel-Aziz, Y.I., & Karara, H.M. (1971): *Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry*, Proceedings of the Symposium on Close-Range Photogrammetry (pp. 1-18). Falls Church, VA: American Society of Photogrammetry.

[2] Young-Hoo Kwon (1998): *Direct Linear Transformation (DLT)*, `http://kwon3d.com/theory/dlt/dlt.html`

[3] Berthold K. P. Horn (2000): *Tsai's camera calibration method revisited*, `http://www-mtl.mit.edu/researchgroups/itrc/ITRC_publication/horn_publications.html`

[4] Roger Y. Tsai (1986): *An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision*, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL, 1986, pages 364-374

[5] Reg Willson (1995): *Tsai Camera Calibration Software*, `http://www-2.cs.cmu.edu/~rgw/TsaiCode.html`

[6] Alberto Menache (1999): *Understanding motion capture for computer Animation and video games*, Morgan Kaufmann

[7] Chris Harris, Mike Stephens (1988): *A combined corner and edge detector*, Plessey research roke manor, United Kingdom, The Plessey company, 1988

[8] David J. Sturman (2003): *A Brief History of Motion Capture for Computer Character Animation*, MEDIALAB

[9] Jiří Zára, Bedřich Beneš, Petr Felkel (1998): *Moderní počítačová grafika*, Computer Press, 1998

[10] Josef Pelikán (1995): *Vyplňování souvislé oblasti,* prezentace k přednášce Počítačová grafika I, KSVI MFF UK Praha

[11] Václav Hlaváč, Miloš Sedláček (2002): *Zpracování signálů a obrazů*, Vydavatelství ČVUT, 2002

[12] Heinrich Tietze (1965): *Famous Problems of Mathematics*, Graylock Press, New York 1965. (kapitola 10, strana 211)