VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

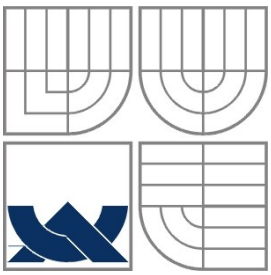
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

TVORBA MODELU (JESKYNÍ) POMOCÍ STEREOKAMERY

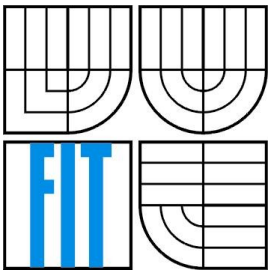
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Jiří Haša
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

TVORBA MODELU (JESKYNÍ) POMOCÍ STEREOKAMERY

(CAVE) MODELLING USING STEREOCAMERA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Jiří Haša

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Jaroslav Rozman

BRNO 2009

Abstrakt

Tato zpráva pojednává o problematice stereoskopického vidění, strojového vidění a rekonstrukce trojrozměrné scény ze stereo snímků. Text se nejprve zaobírá vytvořením matematického základu, ze kterého odvozuje postupy a řešení problémů vznikajících při tvorbě aplikací pro rekonstrukci scény.

Abstract

This document focuses on stereoscopic vision, machine vision and 3D scene reconstruction using stereo pair images. First we create basic mathematic framework which is later used to determine solutions for problems appearing during creation of scene reconstruciton application.

Klíčová slova

stereo, vidění, kamera, geometrie, triangulace, rekonstrukce, scéna

Keywords

stereo, vision, camera, geometry, triangulation, reconstruction, scene

Citace

Jiří Haša: Tvorba modelu (jeskyní) pomocí stereokamery, diplomová práce, Brno, FIT VUT v Brně, 2009

Tvorba modelu (jeskyní) pomocí stereokamery

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením
Ing. Jaroslava Rozmana
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Haša
26.5.2009

Poděkování

V této sekci bych rád poděkoval Ing. Jaroslavu Rozmanovi za cenné rady, připomínky a trpělivost při tvorbě této práce. Dále bych rád poděkoval Ing. Janu Hašovi za pomoc při odstraňování pravopisných chyb v tomto dokumentu

Obsah

Seznam ilustrací

1 Úvod

Účelem této práce je vysvětlit problematiku stereoskopického vidění a uvést postup, jakým se lze řídit při vytváření aplikace pro rekonstrukci scény z třírozměrného obrazu. Celý dokument je rozdělen do tří hlavních částí:

- Na úvod se uvedeme do problematiky stereoskopického vidění, tak, abychom věděli, že člověk není jediný, kdo řeší tuhle problematiku. Budeme hledat inspiraci v řešeních, která vynalezla příroda postupným vývojem a vysvětlíme si, na jakém principu fungují různé typy očí a jejich technický ekvivalent – kamera.
- Kapitola **Stereoskopické snímání** nás zasvětí do matematických principů, které lze využít pro výpočty týkající se rekonstrukce scény. Zavedeme si matematický model kamery, na kterém postavíme celou teorii týkající se rekonstrukce scény.
- V kapitole **Výpočet disparity** se budeme zabývat různými algoritmy, které dokáží nalézt odpovídající si body v levém a pravém obrazu a vytvořit mapu jejich disparity, která je nezbytná pro rekonstrukci scény. V této kapitole budeme využívat znalosti získaných v předchozí kapitole
- Kapitola Implementace se nám pokusí dát návod, jakým postupovat při vytváření aplikace na rekonstrukci scény ze stereo snímků. Zabývá se konkrétními problémy, na které může programátor při psaní podobné aplikace narazit a snaží se poskytnou snadná a elegantní řešení, pomáhající k vytvoření aplikace s možná největším využitím už vymyšlených a implementovaných funkcí tak, aby se programátor nemusel zabývat vymýšlením něčeho, co už bylo dříve vymyšleno a zdokumentováno.
- Na závěr zhodnotíme úspěšnost vytvořené aplikace a možný přínos této práce pro všechny, kdo se zabývají podobnou problematikou.

Vzhledem k tomu, že technika může neustále hledat inspiraci v technických řešeních, které vznikly v přírodě přirozeným vývojem druhů, podíváme se v této kapitole na to, jakým způsobem se u jednotlivých organizmů vyvinuly světlocitlivé orgány s funkcí oka. U jednotlivých typů oka se také zmíníme o možnosti vytvoření umělé varianty a jejího potenciálního využití.

Tato práce se mimo jiné zabývá snímáním skutečného světa kamerou, proto se podíváme zběžně na konstrukci kamery s důrazem na možné vady její optické soustavy (čočky), které budeme v pozdějších částech textu kompenzovat.

1.1 Inspirace přírodou

Během vývoje různých živočišných druhů příroda vyzkoušela různé možnosti vnímání světa . Nejen na blízku, ale i na dálku, což umožňovalo různým živočichům poznávat a reagovat na své okolí. Z počátku pouze na blízko, když reagovaly na dotek nějakého cizího předmětu, nebo na změnu prostředí, ve kterém se pohybovaly. Později se však rozvinuly orgány umožňující vnímání světa i prostřednictvím vlnění. Tyto změny umožnily takto vybaveným živočichům se mnohem lépe přizpůsobovat svému okolí a v důsledku toho měli i mnohem větší šanci na přežití.

V případě jednoho na světlo citlivého orgánu byli takové organizmy schopné zjišťovat přítomnost zdrojů světla, ale nebyly schopné zjistit z jednoho pohledu přesnou polohu takového zdroje. Proto se orientovaly pouze podle toho, jak intenzivně danou veličinu vnímaly. Často se pohybovaly různými směry a porovnávaly intenzitu vjemu.

K dalšímu významnému zlomu došlo, když se začaly objevovat druhy vybavené několika takto citlivými orgány, a nutno říci, že nesrovnatelně dokonalejšími orgány, než jejich předchůdci. V přírodě prosadily převážně druhy se dvěma složenýma očima a dvěma či osmi jednoduchýma.

1.1.1 Složené oči

Složenými na světlo citlivými orgány jsou vybaveny některé druhy hmyzu. Jde o párový orgán, kdy jednotlivé „oči“ jsou složené ze světlocitlivých šestiúhelníkových dílků, kdy každý dílek je schopen plného vnímání obrazu. Důsledkem takového uspořádání je mozaikovitý obraz a neschopnost zaostření. Tato nevýhoda se ukazuje hlavně v případě malých předmětů, které například vosy vůbec nevidí. Nespornou výhodou takového zraku je velké zorné pole umožňující takto vybavenému druhu i případné vidění směrem dozadu, aniž by musel pootočit hlavou. Vnímání pohybu i barev je značně odlišné od toho, jak vnímá svět člověk. Složené oko vnímá pohyb kvůli přesunu obrazu mezi jednotlivými buňkami, což umožňuje vnímat mnohem více impulzů a tedy reagovat na pohyb rychleji. Počet impulzů za sekundu u hmyzu se může blížit až k 200 (nejběžněji se 60-120).

Ilustrace 1: složené oko včely

Složené oko je vybaveno jinými světlocitlivými pigmenty, což má za následek schopnost vnímat světlo na jiných vlnových délkách, jako jsou například ultrafialové složky atp.

Využití tohoto typu oka pro robotiku nebo jiné průmyslové využití se nejeví jako snadný úkol, už jenom vzhledem k složitosti takového senzoru a také potřebnému výpočetnímu výkonu nutnému pro zpracování senzorem získaných informací.

1.1.2 Jednoduché oko

Přestože v názvu tohoto oka je jasně řečeno, že je jednoduché, je potřeba si uvědomit, že jednoduché zde znamená „nikoli složené“ a že i jednoduché oko má většinou velice složitou stavbu. Podle definice jde o oko s jedinou čočkou, která ale může být schopná měnit svoji geometrii a tedy zaostřovat oko na určitou vzdálenost. Protože pravděpodobně nejúspěšnějším živočišným druhem z hlediska přírodního výběru je člověk (a také proto, že tento pohled na věc je nám nejbližší), se budeme zabývat podrobněji právě lidským okem.

Pro člověka je oko orgánem, kterým získává až 80% informací o svém okolí. Světlocitlivost zajišťují tyčinky (vnímající intenzitu světla) a čípky (vnímající barvy), tyto specializované buňky umožňují člověku vnímat světelné vlny v rozsahu 400 – 760 nm. K dokonalosti zrakového vnímání jsou nezbytné části oka tvořící celý jeho optický systém (rohovka, komorová voda, čočka, sklivec), který soustřeďuje světlo do oblasti s foto receptory. Je třeba podotknout, že čípky jsou na světlo méně citlivé a tak za zhoršených světelných podmínek dochází k tomu, že se z vnímaného obrazu vytrácejí barvy.

Ilustrace 2: jednoduché oko

Z našeho pohledu činí jednoduché oko zajímavým především to, že se dá poměrně snadno vyrobit jeho senzorový ekvivalent (kamera), který může mít stejné, nebo dokonce ještě lepší vlastnosti. Dalším důvodem je, že takový senzor je snadný na pochopení a je tedy pro člověka mnohem přirozenější se orientovat v datech získaných takovým senzorem.

1.2 Technický ekvivalent vidění

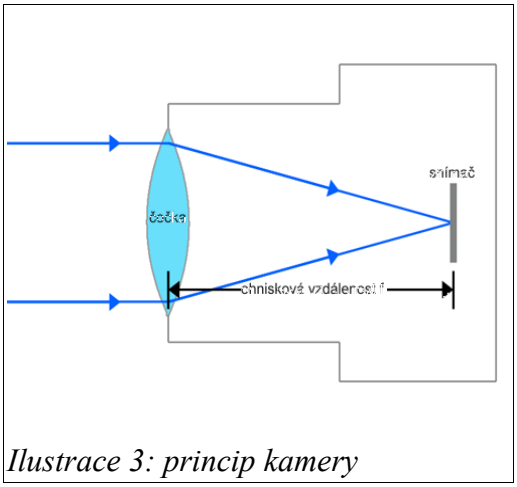
1.2.1 Kamera

Kamera je nástroj na snímání obrazových informací, ať už v podobě nehybných obrázků (fotografií) nebo v podobě filmu. Termín „kamera“ pochází z latinského camera obscura (temná komora). Kamera funguje na podobném principu jako jednoduché oko. Obsahuje světlocitlivý senzor, objektiv nahrazující čočku s měnitelnou ohniskovou vzdáleností a také clonu, která souží pro regulaci množství světla dopadajícího na senzor. Podívejme se teď na jednotlivé části kamery podrobněji:

1.2.1.1 Snímač obrazu

Snímač obrazu je matice polovodičových buněk citlivých na světlo, v nichž se po dobu expozice akumuluje elektrický náboj úměrný osvětlení buňky. Typ a vlastnosti snímače obrazu jsou dány jednak technologií jeho výroby, jednak způsobem sběru náboje z buněk.

Pro použití systémů strojového vidění v praxi většinou stačí znát základní vlastnosti dvou nejpoužívanějších snímačů obrazu, pro které se vžily zkratky CCD (Charged Coupled Device) a CMOS (Complementary Metal Oxide Semiconductor).



Ilustrace 3: princip kamery

Snímače obrazu CCD vysouvá náboj akumulovaný ve světlocitlivých buňkách pomocí soustavy analogových posuvných registrů. Výstupem je tedy analogový signál, který řídicí obvody kamery doplní potřebnou informací pro synchronizaci. Podle způsobu snímání řádků se kamery CCD dělí na kamery podle normy pro televizní vysílání s prokládaným řádkováním (interlaced) a kamery pracující metodou progressive scan. Kamera s prokládaným řádkováním poskytuje dva půlsnímky a ve strojovém vidění se téměř nepoužívá. Při snímání objektu v pohybu totiž mohou být svislé linie roztřepené, protože se stejný systém registrů používá pro sudé i liché řádky, a ty se tedy musí snímat v různých okamžicích. Kamera pracující metodou „progressive scan“ vysouvá náboj akumulovaný v buňkách všech obrazových řádků v jednom okamžiku.

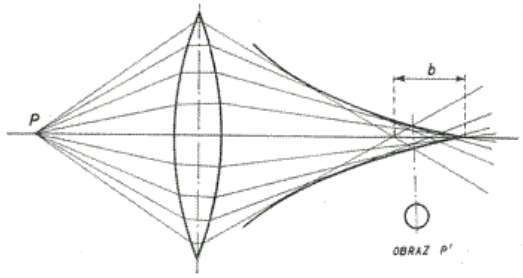
1.2.1.2 Optická soustava

Optická soustava, kterou ve většině případů představuje kamerový objektiv spolu s vhodným osvětlením, má za úkol vytvořit na snímači obrazu takový dvojrozměrný obraz předmětů rozmístěných v prostoru před kamerou. Jejím účelem je upravení svazku světelných paprsků tak, aby pokud možno s co nejmenším zkreslením dopadal na světlocitlivý senzor. Vzhledem k tomu, že celá optická soustava se dá teoreticky nahradit jednou čočkou, budeme dále v textu o optické soustavě hovořit jako o čočce.

Je potřeba si uvědomit, že optická soustava není nikdy zcela dokonalá a tak je obraz, který předává na senzor různým způsobem zkreslený. Nejčastějšími vadami čočky jsou sférická vada, astigmatizmus a zkreslení obrazu.

1.2.1.2.1. Sférická vada

Jde o typickou vlastnost čoček se sférickými nebo rovinnými povrchy. Projevuje se tím, že díky nesprávnému zakřivení čočky neprocházejí paprsky ohniskem. Konkrétně paprsky procházející blízko optické osy se protnou dále, než paprsky procházející čočkou dále od optické osy. Měřítkem velikosti této vady je maximální vzdálenost „ohnisek“ paprsků procházejících čočkou v různé vzdálenosti od optické osy. Velikost projevu této vady lze upravit zacloněním paprsků dále od optické osy, čímž se i zmenší vzdálenost ohnisek od teoretického ohniska čočky. Na obrazu je tato vada rozpoznatelná vzrůstajícím rozmazáním obrazu směrem k okraji (nebo ke středu, záleží na tom, kde přesně je umístěn snímač).



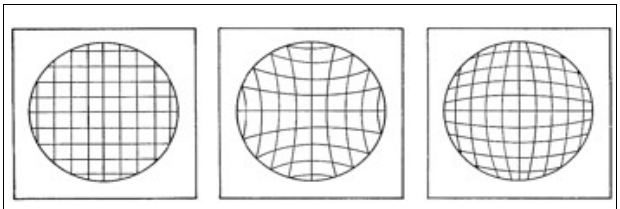
Ilustrace 4: sférická vada

1.2.1.2.2. Astigmatizmus

Projevuje se při zobrazování bodů ležících mimo optickou osu a to i při zobrazování velmi úzkým paprskem. Při zvětšování vzdálenosti zobrazovaného bodu od optické osy se posouvá i bod, do kterého se tento svazek paprsků sbíhá. Tento svazek paprsků má ještě navíc oválný průřez, takže zobrazovaný bod se nejen jeví rozmazaný, ale i zdeformovaný.

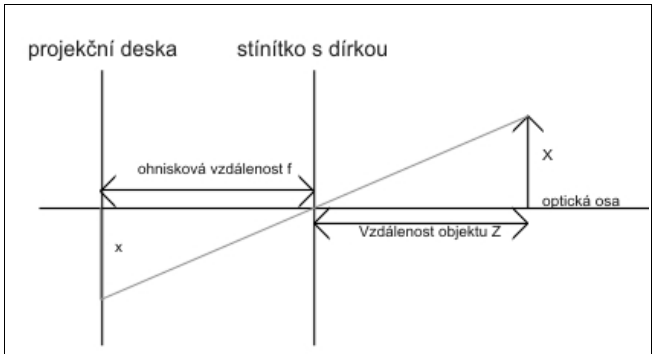
1.2.1.2.3. Zkreslení obrazu

Jedná se o vadu, která se může projevit i v případě, že jsou splněny podmínky bodového zobrazení. Ke zkreslení obrazu dochází kvůli rozdílnému příčnému zvětšení. Když si představíme, že kamera snímá pravoúhlou mřížku kolmou k optické ose, zobrazí se tato mřížka s poduškovitým (v případě vzrůstu příčného zvětšení směrem od optické osy) nebo soudkovitým zkreslením (pokud příčné zvětšení klesá).


Ilustrace 5: Správné, poduškovité a soudkové zobrazení

1.2.2 Matematický model kamery

Představme si nejjednodušší model kamery nazývaný dírková komora (pinhole camera model). V tomhle modelu světlo dopadající na projekční desku prochází malým otvorem. V takovém případě na jeden bod projekční desku dopadá pouze jeden paprsek. Taková kamera je vždy zaostřena a velikost získaného obrazu je závislá jenom na ohniskové vzdálenosti f. V ideálním případě je vzdálenost projekční desky rovna ohniskové vzdálenosti. Takový případ je ukázán na ilustraci 6, kde f je ohnisková vzdálenost, Z je vzdálenost objektu od kamery, X je velikost objektu a x je velikost obrazu objektu. Všimněme si podobnosti trojúhelníků vzniklých na obou stranách stínítka. Pokud vyjdeme z podobnosti trojúhelníků dostaneme se k rovnici



Ilustrace 6: dírková komora

$$\frac{-x}{f} = \frac{X}{Z} \tag{1.1}$$

,poté snadno vyjádříme vztah pro velikost obrazu, uvědomme si, že záporné znaménko zde neznamená zápornou vzdálenost, ale opačnou orientaci vzhledem k optické ose.

$$-x = f\frac{X}{Z} \tag{1.2}$$

Nyní můžeme přeskládat model dírkové kamery do formy, která je ekvivalentní, ale s jednodušším matematickým modelem. Bod, ve kterém se nacházela dírka se nyní bude nazývat střed projekce. V tomto modelu každý paprsek směřující do středu projekce prochází projekční deskou. Bod průniku paprsku s projekční deskou budeme nazývat hlavní bod. Tímto způsobem se vyhneme převrácenému obrazu při zachování jednoduchosti rovnice a z podobnosti trojúhelníků opět dostaneme vztah:

Ilustrace 7: model kamery s projekční rovinou

$$\boxed{\frac{x}{f}=\frac{X}{Z}} \tag{1.3}$$

Protože ale musíme počítat s tím, že žádný opticky čip v kameře není přesně na optické ose, je potřeba ještě zavést další parametry cx a cy pro vyjádření možného posunutí senzoru. Parametry vyjadřují rozdíl polohy středu souřadného systému optického senzoru a středu souřadného systému projekční roviny. Ve výsledku dostáváme relativně jednoduchý model, v němž je bod v trojrozměrném prostoru Q[X, Y, Z] promítán na projekční rovinu (a tedy i obrazovku) na pixel o souřadnicích [x,y] podle rovnic:

$$\boxed{x=f_x\left(\frac{X}{Z}\right)+c_x, y=f_y\left(\frac{Y}{Z}\right)+c_y} \tag{1.4}$$

To, že se zde uvádí dvě ohniskové vzdálenosti je důsledkem faktu, že jednotlivé pixely na levných snímacích čipech nemají čtvercový, ale obdélníkový tvar. Například ohnisková vzdálenost f_x je součinem fyzické ohniskové vzdálenosti čočky a rozměrem x obrazového snímače. Při rozměrové kontrole zjistíme, že fx se udává v pixelech:

$$\boxed{f_x[px]=s_x[\frac{px}{mm}]*F[mm]} \tag{1.5}$$

Uvědomme si, že s_x ani F nejsme schopni změřit bez toho, abychom fyzicky rozebrali kameru na součástky a tedy se musíme spolehnout na naměřené hodnoty f_x a f_y získané při kalibraci.

2 Stereoskopické snímání

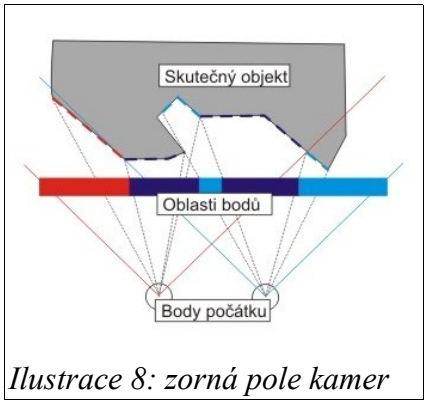
V této kapitole se zaměříme na snímání obrazu více kamerami. Podrobně se budeme věnovat problematice matematického modelu epipolární geometrie a zavedeme pojmy *esenciální matice* a *fundamentální matice*.

Jak jsme si vysvětlili v předchozích kapitolách, kamera je zařízení, které vytváří trojrozměrný obraz skutečného světa. Vezměme si hypotetický příklad, že se snažíme postavit robota, který by byl schopen se samostatně pohybovat v budově. Jistě bychom v takovém případě hledali nějaký vhodný senzor, který by robotovi, stejně tak jako příroda kdysi živočichům, umožnil vnímat prostředí, ve kterém se nachází a zároveň v tomto prostředí rozpoznávat jednotlivé objekty nebo překážky. Existují mnohé typy senzorů, které umožňují velice přesnou orientaci robota. Ty jsou ale buďto aktivní (to znamená, že robot vysílá nějaký signál a vnímá jeho odraz zpět), což může způsobovat to, že díky těmto senzorům by robot působil rušivě (ultrazvuk plaší psy, laser může poškodit zrak při náhodném odrazu...), nebo vyžadují úpravu prostředí, ve kterém by se měl robot pohybovat (majáky, speciální

podlaha...). Naskýtá se nám tedy otázka, zda by nešlo naučit robota vidět tak jako vidíme my? Kamery jsou v dnešní době dobře dostupné, a není tedy problém jich na robota několik namontovat. Umožňují vidění ve spektru, na které jsme zvyklí a o kterém víme, že poskytuje dostatek informací aktuálním prostředí a o objektech v něm. Narážíme zde ale na zásadní problém a to, jak bude robot vnímat prostor, když uvidí jenom dvojrozměrně?

2.1 Matematický základ stereoskopického snímání

Stereoskopické snímání je takové snímání, při kterém je trojrozměrná scéna snímaná současně dvěma synchronizovanými kamerami, které dohromady tvoří soustavu nazývanou dále **stereokamera.** Pokud nemáme k dispozici stereokameru, máme možnost nahradit ji jednou kamerou, kterou budeme postupně snímat scénu ze dvou bodů, tak jako bychom použili stereokameru. V takovém případě ale mějme na paměti, že jsme omezeni na snímání statických scén, protože mezi zachycením jednotlivých snímků se nesmí scéna změnit.

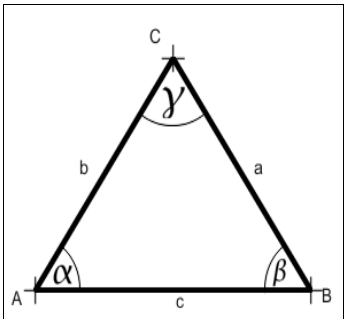


Ilustrace 8: zorná pole kamer

Snímky, které byly zaznamenány stereokamerou (nebo její jednokamerovou variantou) se nazývají stereosnímky a na rozdíl od standardních fotografií (nebo videa) jsme z nich schopni vypočítat třetí, prostorovou, souřadnici pro všechny body, které jsou vidět na obou snímcích. Mějme na paměti, že taková oblast je určena vzdáleností jedné kamery od druhé, rozdílem jejich orientace a také zorným polem. Na ilustraci je znázorněno, jakým způsobem se zorná pole kamer překrývají a zároveň je zde i ukázáno, u jakých částí objektu bude možné vypočítat vzdálenost bodu. Pro přehlednost je zde ukázáno i rozložení takových bodů v obrazové rovině. Červeně jsou znázorněny body snímané levou kamerou, zatímco modře jsou zobrazeny body snímané pravou kamerou. Fialově jsou zobrazeny body, u kterých bude možné měřit vzdálenost. Na první pohled je vidět, že měřitelná oblast je značně menší než celková snímaná oblast. Za zmínku stojí i skutečnost, že jakákoli překážka, která vystupuje z pozadí se sice projeví - ale stačí, aby některé objekty za ní byly zakryté a už nebude možné měřit jejich vzdálenost.

2.1.1 Triangulace

Víme, že pro zaměření bodu v prostoru lze využít techniku zvanou triangulace. Je to technika založená na geometrických vlastnostech trojúhelníku, umožňující spočítat polohu jednoho vrcholu, pokud známe úhly a polohy v ostatních dvou vrcholech nebo vzdálenosti všech bodů. Pro nás to znamená, že budeme potřebovat 2 senzory se známou polohou, kterými budeme zaměřovat třetí bod nacházející se v prostoru. Pro zaměření bodu musíme znát úhly, které naše senzory svírají s tímto bodem.

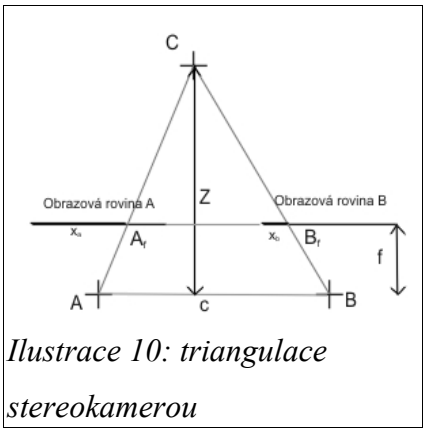


Ilustrace 9: geometrie trojúhelníku

Představme si, že naše senzory jsou umístěny v bodech $A[x_a,y_a]$ a $B[x_b,y_b]$ a pokoušíme se zjistit přesnou polohu bodu $C[x_c,y_c]$. Při znalosti velikosti úhlů α a β nalezneme rovnice polopřímek tvořených rameny úhlů α a β. Vyřešením soustavy těchto rovnic dostaneme souřadnice jejich průsečíku a tedy i bodu C.

2.1.2 Aplikace triangulace při stereoskopickém vidění

Nejvhodnější metodou pro zjišťování souřadnic skutečného bodu je bezesporu triangulace. Pokud jí chceme využít, musíme si ale nejprve zjistit všechna potřebná data pro výpočet, což znamená převést souřadné systémy kamer do jednoho společného souřadného systému, ve kterém bude prováděn výpočet. K tomu potřebujeme vědět ohniskové vzdálenosti obou kamer (v ideálním případě jsou stejné), vzdálenost kamer a souřadnice hledaného bodu na obou obrazech.



Ilustrace 10: triangulace stereokamerou

Vysvětleme si nyní postup výpočtu na příkladu uvedeném v ilustraci 9 Ilustrace . Body A,B jsou body projekce, obrazová rovina je ve vzdálenosti f. Souřadnice x_a a x_b jsou souřadnice projekcí bodu C na obrazových rovinách jednotlivých kamer. Body A_f a B_f jsou hlavními body paprsku, tedy průsečíky paprsků s obrazovými rovinami. Všimněme si, že místo toho, abychom hledali rovnice pro jednotlivé přímky nám opět stačí využít podobnosti trojúhelníků ABC a A_fB_fC. Délka úsečky AB (strana c trojúhelníka) je rovna vzdálenosti kamer. Délka úsečky A_fB_f se dá vyjádřit jako $c-(x_a-x_b)$. Když z těchto údajů sestavíme rovnici, vyjde nám:

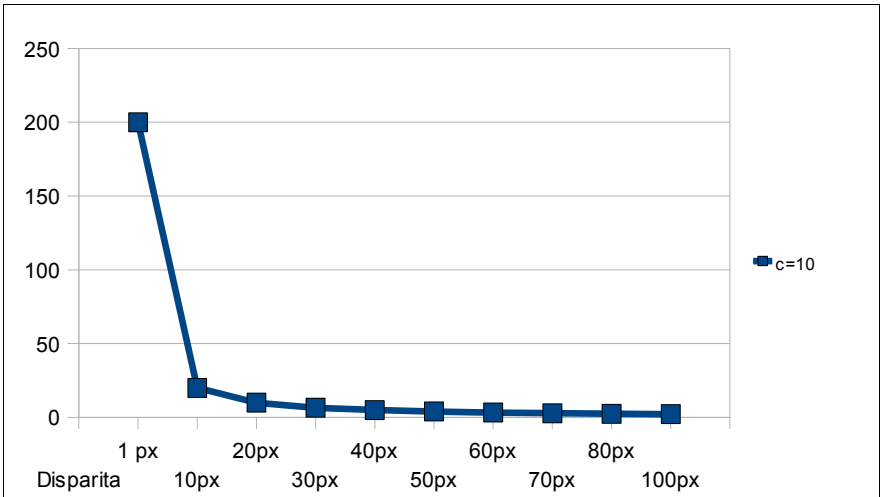
$$\frac{c-(x_a-x_b)}{Z-f}=\frac{c}{Z} \tag{2.1}$$

a tedy:

$$Z = \frac{cf}{x_a - x_b} \tag{2.2}$$

Tady zavedeme nový pojem **disparita** $d=x_a-x_b$, která je rozdílem mezi souřadnicemi x projekcí bodu v jednotlivých obrazech. Při pohledu na výše uvedenou rovnici je zřejmé, že vzdálenost Z bodu C je nepřímo úměrná disparitě, takže mezi těmito veličinami bude nelineární vztah. Vyplývá nám, že čím větší bude disparita, tím menší budou rozdíly ve vzdálenosti a naopak. Proto také disparita neposkytuje rovnoměrné měřítko a běžně tedy není použitelná jako hloubková mapa.

Za pomoci dosazování do tohoto vzorce jsme schopni vytvořit graf rozlišitelných vzdáleností



Ilustrace 11: graf závislosti vzdálenosti na disparitě

Je vidět, že přesnost měření vzdálenosti objektu stereokamerou s klesající disparitou klesá a pro každý krok je rozdíl přesnosti čím dál větší. Teoreticky jsme schopni tuto přesnost upravit nastavením změny vzdálenosti mezi kamerami, měli bychom ale brát v potaz to, že mezi dvojicí kamer vzniká slepá oblast, kam ani jedna kamera nevidí (platí pro rovnoběžné uspořádání kamer).

2.1.3 Projekce bodů na obrazovou rovinu

Pro každý snímek, ve kterém je šachovnice detekovaná můžeme popsat souřadný systém snímaného objektu v souřadném systému obrázku, na kterém se zobrazuje s pomocí translačního vektoru a rotační matice.

Natočení jakéhokoli objektu v prostoru lze popsat jeho rotací podle jednotlivých os **x**,**y**,**z** o odpovídající úhly $\boldsymbol{\psi},\boldsymbol{\varphi},\boldsymbol{\theta}$. tedy s pomocí tří rotací v dvourozměrném prostoru. Pokud budeme otáčet objekt postupně kolem jednotlivých os, bude jeho výsledné otočení dáno součinem tří matic $R_x(\boldsymbol{\psi}),R_y(\boldsymbol{\varphi}),R_z(\boldsymbol{\theta})$, kde:

$$R_x(\psi)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\psi & \sin\psi\\ 0 & -\sin\psi & \cos\psi\end{bmatrix} \quad (2.3)$$

$$R_y(\varphi)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\varphi & \sin\varphi\\ 0 & -\sin\varphi & \cos\varphi\end{bmatrix} \quad (2.4)$$

$$R_y(\theta)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta & \sin\theta\\ 0 & -\sin\theta & \cos\theta\end{bmatrix} \quad (2.5)$$

A tedy rotační matice R= $R_x(\psi)R_y(\varphi)R_z(\theta)$. Tato matice má tu vlastnost, že matice k ní inverzní je současně její transpozicí, takže $R^TR=RR^T=I$, kde I je jednotková matice.

Posunutí objektu v prostoru popisujeme translačním vektorem, který vznikne jako rozdíl souřadnic dvou bodů, tedy pro translační vektor z bodu A[A_x, A_y, A_z] do bodu B[B_x, B_y, B_z] platí, že T=A-B=[A_x-B_x, A_y-B_y, A_z-B_z].

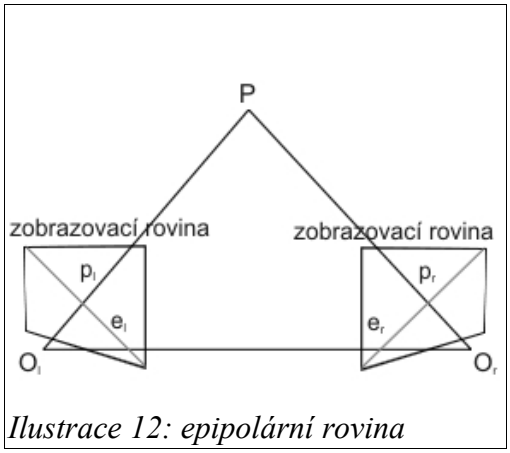
Pokud se nám podaří nalézt takové R a T, které vytvářejí relaci mezi každým zaměřeným bodem na šachovnici P_O a každým pozorovaným bodem v obraze P_C, můžeme zapsat tuto relaci jako

$$\boxed{P_C=R(P_O-T)} \quad (2.6)$$

Pokud zkombinujeme tuto rovnici s maticí pro vnitřní parametry matice, získáme základní model mapování bodů reálného objektu do souřadného systému obrázku na displeji. Matice vnitřních parametrů kamery má 4 hodnoty (f_x, f_y, c_x, c_y). Z počtu proměnných, které je nutné vypočítat vyplývá, že pro kalibraci jedné kamery nám budou stačit dva snímky (vzhledem k tomu, že R a T se mění s každým snímkem a matice pro vnitřní parametry kamery zůstává konstantní).

2.1.4 Epipolární geometrie

Epipolární geometrie je vnitřní projektivní geometrie mezi dvěma pohledy. Je nezávislá na struktuře scény, závisí pouze na vnitřních parametrech kamer a na jejich relativní pozici. V podstatě je to geometrie protínání obrazových rovin s rovinami, jejichž průsečíkem je báze (přímka spojující středy kamer)



Ilustrace 12: epipolární rovina

Každá kamera má svůj vlastní střed projekce O a vlastní zobrazovací rovinu. Skutečný bod P se promítá do projekčních rovin jednotlivých kamer jako bod p_l a p_r. Dalšími body ležícími na této rovině jsou epipóly e_l a e_r, které jsou na jednotlivých zobrazovacích

rovinách obrazy center projekce opačné kamery. Jak je vidět na ilustraci, e_l je projekcí bodu O_r a naopak.

Abychom porozuměli využití epipólů, měli bychom si nejprve uvědomit, že na levé nebo pravé obrazové rovině vidíme pouze projekci skutečného bodu P, a nevíme, jaká je jeho vzdálenost, tedy pokud se díváme na obrazovou rovinu pravé kamery, vidíme pouze bod p_r, což znamená, že skutečný bod P se může nacházet na libovolném místě polopřímky určené body p_r a P. Stejně tak, když se díváme na obrazovou rovinu levé kamery, vidíme pouze bod p_l, a opět skutečný bod P se může nacházet na na libovolném bodě polopřímky určené body p_l a P. Pokud se ale podíváme na projekcí oné polopřímky na druhé projekční rovině, vidíme, že její projekce je shodná s polopřímkou určené body e_r a p_r.

2.1.5 Matematický model epipolární geometrie

Epipolární geometrie je algebraicky reprezentovaná esenciální maticí E a z ní vycházející fundamentální maticí F. Obě matice v sobě zahrnují informace o vzájemné poloze a otočení kamer s tím, že Fundamentální matice v sobě zahrnuje i vnitřní vlastnosti obou kamer.

2.1.5.1 Esenciální matice

Pokud máme bod P, máme za úkol odvodit vztah, který je relací mezi jeho projekcemi p_l a p_r Vybereme si tedy souřadný systém jedné z kamer, ve kterém budeme výpočet provádět.. Vybereme si souřadnicový systém levé kamery se středem v O_l . V tomhle systému je lokace obrazu bodu P v zobrazovací rovině určena vektorem P_l a projekční bod druhé kamery je určen vektorem T. Bod P_r, jak jej vidí druhá kamera je určen vektorem P_r. Platí, že $P_r=R(P_l-T)$. V následujícím kroku zavedeme epipolární rovinu. Z definice roviny určené bodem a normálovým vektorem víme, že pro každý bod X ležící na této rovině platí vztah (X-A).n=0. Protože rozdíl souřadnic bodů ležících na dané rovině bude vždy vytvářet vektor kolmý na normálový vektor n a tedy jejich součin bude vždy roven nule. Za body X a A můžeme dosadit P_l a T. Pokud bychom měli vektor kolmý na P_l a T (jako třeba jejich vektorový součin), mohli bychom jej dosadit jako normálový vektor n epipolární roviny, čímž bychom dostali rovnici:

$$\boxed{(P_l - T)^T (T \times P_l) = 0} \tag{2.7}$$

Abychom vytvořili relaci mezi body q_l a q_l, což jsou body nacházející se na obrazech jednotlivých kamer, tím že nejdříve vytvoříme spojitost mezi obrazy Pl a Pr. Pr jsme zakreslili do roviny skrze rovnici $P_r=R(P_l-T)$, kterou můžeme upravit na tvar $P_l-T=P_rR^{-1}$. Poté zavedeme substituci $R^T=R^{-1}$ a dosadíme do rovnice epipolární roviny, čímž dáme do přímé souvislosti oba obrazy bodu P na jednotlivých zobrazovacích rovinách.

$$(P_r R^T)^T (T \times P_l) \tag{2.8}$$

Vektorový součin můžeme rozepsat jako násobení matic, nadefinujeme tedy matici S jako

$$T \times P_l = SP_l \Rightarrow SP_l \Rightarrow S = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{2.9}$$

Pokud substituujeme matici S do rovnice epipolární roviny, dostaneme $(P_r)^T RSP_l=0$. Esenciální matice E je definována jako součin RS, čímž vzniká rovnice $(P_r)^T EP_l=0$. Tato rovnice ale stále nevytváří relaci mezi body na obrazových rovinách jednotlivých kamer. Aby tohle platilo, zavedeme substituci $p_l=f_l P_l/Z_l$ a odpovídající substituci $p_r=f_r P_r/Z_r$, což nás vede k rovnici

$$\left(\frac{f_r P_r}{Z_r}\right)^T E \frac{f_l P_l}{Z_l} = 0 \tag{2.10}$$

,kterou můžeme vynásobit $Z_r Z_l / f_l f_l$, abychom se dostali ke konečnému vztahu definující vztah mezi jednotlivými obrazy bodu P:

$$p_r^T E p_l = 0 \tag{2.11}$$

2.1.5.2 Fundamentální matice

Matice E obsahuje veškeré informace o geometrii jedné kamery ve vztahu k druhé, ale neobsahuje žádné informace o kamerách samotných. Ve skutečnosti se ale zajímáme o výpočty v souřadném systému pixelů na obrázku, a odpovídající epipolární linii na druhém obrázku. Abychom mohli uvést tyto souřadné systémy do relace, potřebujeme znát vnitřní parametry obou kamer. Dosadíme tedy vztah q=Mp, kde q je bod na obrázku kamery, p je obraz bodu na zobrazovací rovině kamery a M je matice obsahující vnitřní parametry kamery (posunutí snímače...) do rovnice esenciální matice a dostaneme:

$$\left(\frac{q_r}{M_r}\right)^T E \left(\frac{q_l}{M_l}\right) = 0 \tag{2.12}$$

Rovnici uspořádáme zavedením fundamentální matice:

$$F = \frac{E}{M_r^T M_l} \tag{2.13}$$

Po substituci rovnice s esenciální maticí za rovnici s fundamentální maticí dostaneme výsledně:

$$q_r^T F q_l = 0 \tag{2.14}$$

V praxi to znamená, že fundamentální rovnice v sobě zapouzdřuje převod obrazového bodu z pixelového souřadného systému do souřadného systému skutečného bodu, následně aplikuje výpočet s pomocí esenciální matice, což je vlastně nalezení obrazu bodu na zobrazovací rovině druhé kamery a poté jej převede do souřadného systému obrazu vytvářeného druhou kamerou.

2.2 Skutečná stereokamera

Když jsme si položili teoretický základ pro epipolární geometrii, máme přibližnou představu o tom, jakým způsobem bychom měli stereokameru připravit a jakým způsobem budeme převádět obrazy jednotlivých bodů, na souřadnice v třírozměrném souřadném systému. Předtím, než se pokusíme kameru zapojit, musíme se nejdříve podrobně podívat na to, s jakým zařízením budeme pracovat. Přestože lze jako stereokameru použít takřka libovolnou kombinaci i těch nejlevnějších USB kamerek, budeme se v této části zabývat stereokamerou, která je na úkoly z oblasti počítačového vidění a rekonstrukce obrazu přímo stavěná. Jedná se o kameru od společnosti VidereDesign STH-DCSG-VAR.

Ilustrace 13: stereokamera VidereDesign

2.2.1 Charakteristika stereokamery

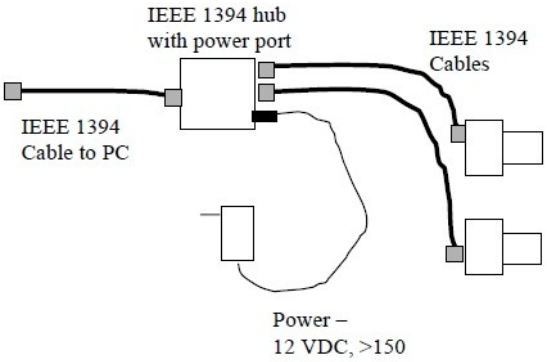
Stereo hlava VidereDesign je kompaktní nízkonapěťová stereokamera s rozhraním IEEE 1394 (Jinak také FireWire, rychlé komunikační rozhraní používané pro práci s videotechnikou a pevnými disky). Skládá se ze dvou samostatných kamer upevněných na společném hliníkovém rámu. O snímání obrazu se stará CMOS modul s fyzickým rozlišením bodů 640x480 umožňující expozici všech bodů snímače v přesně stejnou dobu. Tato vlastnost činí kameru vhodnou do prostředí s rychle se pohybujícími objekty.

CMOS snímače jsou MT9V022 senzory od firmy Micron Semiconductor, které se dodávají v barevné nebo monochromatické verzi. My máme k dispozici barevnou verzi snímače. Objektiv použitý na této kameře využívá standard C/CS a je tedy možné jej vyměnit. Kamera neposkytuje žádné jiné možnosti nastavení přímo na hardwaru.

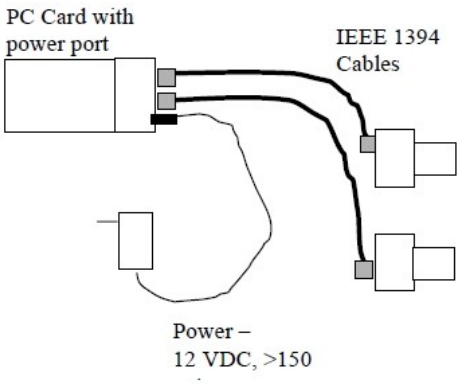
2.2.1.1 Zapojení kamery

Kamera vyžaduje, aby na hostitelském počítači byly 2 volné šestipinové konektory(F). Maximální délka kabelu je 4,5 metru. Pokud je nutné připojit kameru na delší vzdálenost, výrobce dodává i 10 metrový speciální kabel. Tato vzdálenost může být ještě navýšena přidáním 1394 opakovače. Při maximálním vytížení tato kamera zabere zhruba 60% šířky 400MBps pásma.

6-pinový kabel je nutný zejména proto, že kamera nemá žádný externí zdroj napájení, takže musí být napájena skrze 1394 rozhraní. Pokud je na počítači k dispozici pouze 4 pinový konektor, je nutné dokoupit 1394 hub, s externím napájením, aby stereokamera mohla fungovat. Tato situace se týká zejména notebooků.



Ilustrace 15: připojení kamery k notebooku



Ilustrace 14: připojení kamery PC 6 pinovými kabely

2.2.1.2 Optická soustava kamery

Ke kameře lze použít objektivy standardu C nebo CS. Objektivy standardu CS se připevňují 12,5mm od snímače, objektivy typu C se připevňují 17,5 mm od snímače, proto budou potřebovat prodlužovací prvek, který umožní je upevnit do této vzdálenosti.

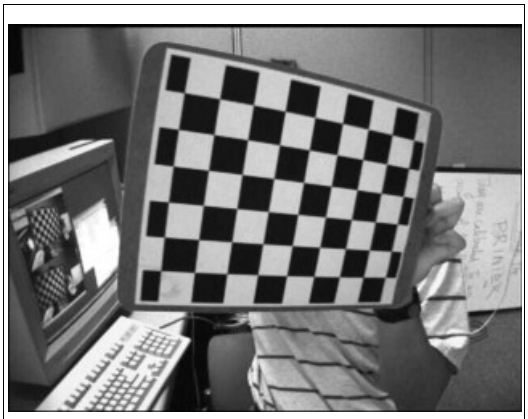
Optická propustnost F je měřítkem schopnosti objektivu propouštět světlo. Čím je F menší, tím více světla objektiv propouští a tím pádem je kamera schopna pracovat v tmavějším prostředí. Pro snímání v interiéru jsou vhodné hodnoty menší než 1,8. Ohnisková vzdálenost použité čočky přímo ovlivňuje všechny další charakteristiky optické soustavy kamery jako je zorný úhel a rozlišitelnost vzdálenosti objektů. Mějme na paměti, že čím větší je zorný úhel, tím horši je schopnost stereo systému rozlišovat vzdálenost snímaných objektů.

2.3 Snímání stereokamerou

Přestože čistě teoreticky bychom mohli začít snímat okamžitě po zapojení kamery, brzy bychom došli ke zjištění, že poskytované snímky nejsou přesně srovnané a že máme problémy s určováním disparity jednotlivých pixelů. Když jsme si uváděli matematický model a vlastnosti kamery, počítali jsme s ideálním uspořádáním, kterého se nám ale čistě hardwarovým nastavením podaří dosáhnout pouze velice zřídka. Někteří odborníci dokonce radí, že poté co si člověk kameru nastaví tak aby byly jejich optické osy naprosto rovnoběžné, měl by kameru nějakým způsobem zafixovat a dále s ní nemanipulovat. Opravdu přesné nastavení kamer může zabrat i více než hodinu čistého času. I tak se ale nikdo nevyhne poslední přípravě před snímáním složené z *kalibrace* a *rektifikace*.

2.3.1 Kalibrace kamery

V této kapitole se podrobně podíváme na postup, jakým lze kameru připravit pro snímání obrázků, které jsou co nejméně poznamenané vadami optické soustavy kamery a posunutím snímače. Poznamenejme, že kalibrace kamery už je funkce čistě softwarová a je pouze na tvůrci programu, jakým způsobem ji provede. Některé kamery mají kalibrační funkce uloženy přímo ve své paměti a jsou schopny ji provádět zcela automaticky. Přesto musíme počítat s tím, že ne všechny kamery jsou toho schopny a tedy musíme provést kalibraci sami.



Ilustrace 16: kalibrační obrazec

Vzhledem k tomu, že jsme využili knihovny OpenCV, budeme se zabývat tím, jakým způsobem se provádí kalibrace právě v rámci této knihovny. Metoda, kterou budeme používat spočívá ve snímání známého a dobře rozpoznatelného objektu kamerou. Na tomto místě nám dobře poslouží šachovnice o téměř libovolném počtu políček, kterou budeme s kamerou snímat. Co se týče velikosti šachovnice a počtu políček, bylo by dobré brát v potaz fakt, že pokud

Ilustrace 17: správně detekovaná šachovnice

chceme provést měření co možná nejpřesněji, měli bychom zajistit, aby šachovnice zabírala nezanedbatelnou část obrazu a počet detekovaných bodů (rohů mezi políčky) byl dostatečný počet pro provedení výpočtu. Pokud provedeme dostatečný počet snímání, jsme schopni vypočítat vnitřní parametry kamery.

Kalibrace stereokamery závisí na nalezení rotační matice R a translačního vektoru T mezi dvěma kamerami. Kalibrační funkce dokáže díky rozdílům v projekcích kalibračního obrazce (v našem případě šachovnice) matici R a vektor T nalézt. Platí, že pro každý bod v trojrozměrném prostoru můžeme odděleně použít kalibraci jedné kamery a bod P položit na souřadnice $P_l=R_lP+T_l$ a opačně $P_r=R_rP+T_r$. Už víme, že v epipolární geometrii platí, že $P_l=R^T(P_r-T)$. Z těchto rovnic můžeme vyjádřit rovnice $R=R_r(R_l)^T$ a $T= T_r-RT_l$.

Funkce, kterou budeme používat pro kalibraci stereokamery v OpenCV je *cvStereoCalibrate*, která nejprve provede kalibraci každé kamery zvlášť a poté vypočítané hodnoty dosadí do výše uvedených rovnic, čímž nalezne i rotační matici a translační vektor mezi obrazovými rovinami jednotlivých kamer. Vzpomeňme, že v teoretické části dokumentu jsme zjistili, že pro kalibraci jedné kamery jsou potřeba minimálně dva snímky, pro který bude tato funkce vracet přesně vypočítané hodnoty zatížené šumem v obrazu nebo zaokrouhlováním. Pokud bude vstupních obrazů více, funkce vrátí hodnoty, které přibližně a nejvhodněji odpovídají naměřeným skutečnostem.

Pro použité této funkce musíme znát několik faktů, které nám dají dohromady spojení mezi reálným souřadným systémem. Jsou jimi souřadnice jednotlivých bodů objektu v reálném souřadnicovém systému, souřadnice těchto bodů v nasnímaných obrazech a matice určující počet nalezených bodů v jednotlivých obrazech.

Jednotkou reálného souřadného systému může být jakákoli jednotka vzdálenosti, jako třeba centimetry nebo milimetry. Tím pádem budou všechny vypočítané hodnoty vztaženy k tomuto systému a veškerá projekce zpět do trojrozměrného souřadného systému provedená na základě těchto hodnot bude mapovat body z obrazu do systému s jednotkou použitou při kalibraci.

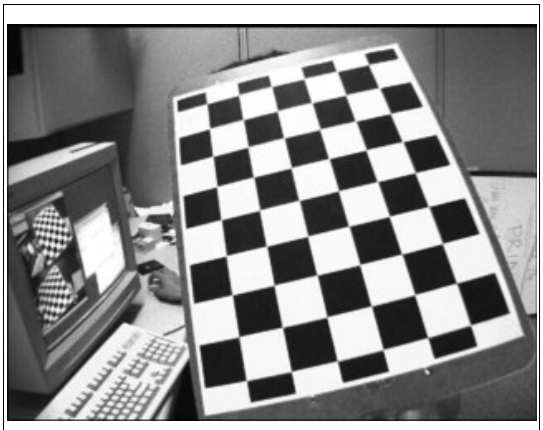
Z hodnot, které vrací kalibrační funkce můžeme zjistit

- matici s parametry obou kamery
- deformační parametry optické soustavy každé kamery
- rotační matici R
- translační vektor T
- esenciální matici E
- fundamentální matici F

všechny tyto hodnoty dohromady kompletně popisují optický systém naší stereokamery a jsou nutné pro další výpočty prováděné s těmito kamerami získanými stereo snímky.

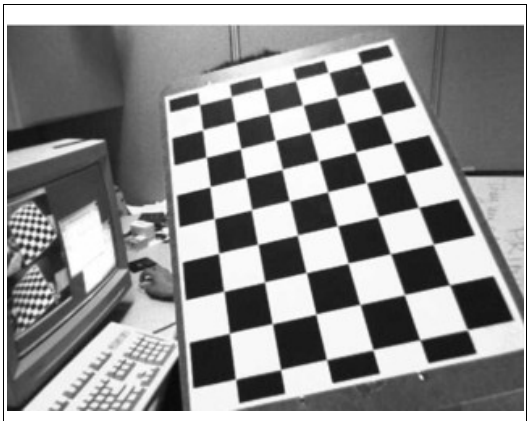
2.3.2 Odstranění deformace obrazu

Poté, co se nám podařilo získat parametry stereokamery, můžeme se pustit do odstranění deformace obrazu způsobené nepřesným uložením snímacího čipu a vadami čočky. K tomu nám dobře poslouží funkce *undistort*, které předáme vstupní obrázek, vnitřní parametry kamery a deformační parametry optické soustavy. Funkce spočítá takovou mapu deformačních koeficientů, že její účinek vyruší deformaci obrazu způsobenou chybami optické soustavy.



Ilustrace 18: deformovaný obrázek

Je potřeba si uvědomit, že kdybychom například při zpracovávání sekvence snímků, nebo videa volali stále dokola tuhle funkci, počítali bychom stále dokola jak výpočet koeficientů, tak jejich účinek na obrázek. Vzhledem k tomu, že koeficienty jsou vzhledem ke snímku neměnné (počítají se z parametrů kamery), je možné vypočítat si mapu deformačních koeficientů před zpracováváním obrazu a potom ji v každém snímku pouze aplikovat. K tomuto účelu slouží funkce *cvInitUndistortMap* a *cvRemap* .



Ilustrace 19: obrázek po přemapování

Pokud chceme použít snímky, které získáme stereokamerou k rekonstrukci obrazu, pravděpodobně se budeme dříve nebo později zabývat zjišťováním disparity pro jednotlivé pixely obrazu. Všechny algoritmy pro počítání disparity poskytují různé výsledky v závislosti na tom, jakým způsobem jsou uspořádány snímky, které má funkce porovnat.

Ilustrace demonstrují, jak vypadá snímek, který nebyl ještě upraven a jak vypadá už upravený snímek po přemapování deformačními koeficienty. Všimněme si, že změna obrazu může být dost značná. Na ilustraci je vidět, že bylo zcela kompenzováno zakřivení šachovnice. Odstranění deformace nám poskytne výhodu při dalším zpracování těchto obrazů, zejména proto, že projekce snímaných objektů nebudou měnit svůj tvar v závislosti na tom, jakou částí čočky prochází jejich paprsky a podle toho, na kterou část optického senzoru tyto paprsky dopadají.

2.3.3 Rektifikace obrazu

Rektifikace obrazu je proces, při kterém se obrazy jednotlivých kamer srovnají tak, aby jednotlivé sobě odpovídající body měly stejnou y[px] souřadnici, což mimo jiné způsobí to, že se vykompenzuje rozdíl v Y souřadnicích projekčních rovin jednotlivých kamer. OpenV implementuje Hartleyho a Bougetovu metody rektifikace. Hartleyho metoda se používá v případě, že snímky byly získány nezkalibrovanými kamerami, což ale není náš případ, takže tuto metodu vynecháme a budeme se podrobněji věnovat Bougetově metodě.

2.3.3.1 Bougetova metoda

Bougetův algoritmus se snaží minimalizovat množství změn v jednotlivých obrazech (a tedy i i nepřesností pří zpětné projekci) při zachování co největší možné měřitelné oblasti. Aby dosáhl co nejmenšího počtu změn, nepracuje pouze s jedním obrazem, ale přizpůsobuje oba obrazy jejich teoretickému průměru. V praxi to vypadá tak, že algoritmus rozdělí rotační matici R na matice R_1 a R_2 .Každá kamera má tedy poloviční rotaci, takže jejich optické osy směřují rovnoběžně s vektorovým součtem jejich původních optických os. Abychom spočítali R_{rect} , takovou, aby přenesla epipól levé kamery do nekonečna a a srovnala horizontálně epipolární linie, vytvoříme rotační matici tím, že začneme se směrem epipólu e_1. Vezmeme souřadnice optické osy v obraze (c_x,c_y) jako počátek obrazu. Směr vektoru epipólu e1 je tedy rovnoběžný s translačním vektorem mezi středy projekce jednotlivých kamer. .

$$e_1 = \frac{T}{\|T\|} \tag{2.15}$$

Další vektor e_2 musí být svírat s e_1 pravý úhel. Je vhodné zvolit e_2 tak, aby svíral pravý úhel i s optickou osou. Toho dosáhneme vektorovým součinem e1 s vektorem optické osy a následnou normalizací výsledku.

$$e_2 = \frac{[-T_y T_x 0]^T}{\sqrt{T_x^2 + T_y^2}} \tag{2.16}$$

Třetí vektor e_3 je svírá pravý úhel s e_1 a e_2, takže jej snadno vypočítáme jejich vektorovým součinem:

$$e_3 = e_1 \times e_2 \tag{2.17}$$

Matice , která nasměruje epipóĺ do nekonečna je tedy

$$R_{Rect} = \begin{bmatrix} (e_1)^T \\ (e_2)^T \\ (e_3)^T \end{bmatrix} \tag{2.18}$$

Horizontálního zarovnání obrazů kamer je potom dosaženo vynásobením maticemi R_l a R_r, kde $R_l=R_{Rect}r_l$ a $Rr=R_{Rect}r_r$. Spočítáme také rektifikované matice M_{rect_l} a M_{rect_r} , ale vrátíme je vynásobené s projekčními maticemi P_l a P_r,

$$P_r = M_{Rectr} P_r' = \begin{bmatrix} f_{xr} & \alpha_r & c_{xr} \\ 0 & f_{yr} & c_{yr} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P_l = M_{Rectl} P_l' = \begin{bmatrix} f_{xl} & \alpha_l & c_{xl} \\ 0 & f_{yl} & c_{yl} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.19)$$

Projekční P převede 3D bod v homogenních souřadnicích na 2D bod v homogenních souřadnicích způsobem:

$$P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad (2.20)$$

Kde souřadnice na obrazu můžeme vypočítat jako x/w a y/w. Body v dvojrozměrném prostoru je možné zpět promítnout do 3D na základě jejich souřadnic v obrázku a vnitřních parametrů kamery. Tuto matici nazveme **reprojekční matice Q** a budeme ji využívat pro rekonstrukci třírozměrného obrazu.

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{-1}{-T_x} & \frac{c_x - c_x'}{T_x} \end{bmatrix} \quad (2.21)$$

Pokud máme k dispozici homogenní bod a jeho přiřazenou disparitu, jsme schopni promítnout bod zpět do trojrozměrného prostoru dosazením do rovnice:

$$Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \quad (2.22)$$

Souřadnice bodu ve 3D potom budou [X/W,Y/W,Z/W]

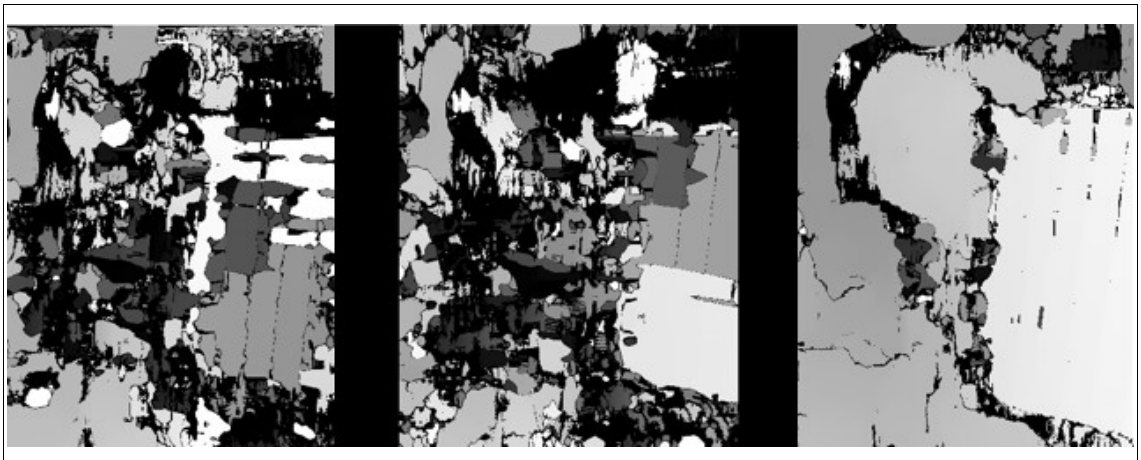
Tuto metodu popisuje funkce *cvStereoRectify,* která má jako vstupní parametry kamerové matice, deformační koeficienty, translační vektor T a rotační matici R. Funkce vraci

- projekční rovnice P_r a P_l
- rektifikační rotační matice R_r a R_l
- reprojekční matici Q

Po zavolání této funkce máme k dispozici všechna potřebná data k tomu, abychom přepočítali jednotlivé obrazy do jejich rektifikovaných podob. K tomu použijeme postupu velmi podobného jako u kalibrace, kdy jsme kompenzovali deformaci obrazu způsobenou optickou soustavou kamery. Tato zmínka o podobnosti postupu není náhodná, protože funkce *cvRemap* přepočítává obraz podle parametrů uložených v deformačním vektoru pro x a y souřadnice zvlášť. Budeme tedy po prostudování této kapitoly přepočítávat všechny deformace obrazu najednou, tak, abychom se nakonec při jednom zavolání funkce c*vRemap* získali rektifikovaný obraz bez artefaktů způsobených optickou soustavou kamery. Využijme tedy funkce *cvInitUndistortRectifyMap*, která při výpočtech zohledňuje jednak deformační parametry kamery a zároveň i úpravy nutné k tomu, aby obrazy byly rektifikované. Výsledné mapy stejně jako v kapitole o kalibraci předáme funkci *cvRemap*.

2.3.4 Shrnutí

Kalibrace a rektifikace jsou činnosti, které by se měly provést před každým snímáním, pokud je pravděpodobné, že se s kamerami mezi jednotlivými snímky pohnulo. Výstupní matice těchto funkcí v zásadní míře ovlivňují kvalitu výsledných snímků a tedy i vstupní data pro další kroky, jako je hledání disparity nebo nebo rekonstrukce třírozměrné scény. Vzhledem k tomu je vhodné při libovolných výpočtech pracovat s rektifikovanými a upravenými snímky Následující ilustrace ukazují, jaký vliv mají rektifikace a kalibrace vliv na výslednou podobu snímků a jako příklad jsou uvedeny i snímky ukazující mapu disparity pro různě ošetřené snímky.



Ilustrace 20: ukázka různě kvalitních map disparity

Na ilustraci 20 je vidět, že nijak neupravené snímky jsou naprosto nepoužitelné (vlevo), uprostřed je disparita vypočítaná ze snímků, u kterých byla odstraněna deformace čočkou. Takový snímek je naprosto nepoužitelný. Až rektifikované snímky zkalibrovaných kamer se dají použít (vpravo).

Ilustrace 21: neupravený stereo snímek

Ilustrace 22: stereo snímek po kalibraci a odstranění deformace

Ilustrace 23: rektifikovaný stereo snímek (horizontální linky přidány dodatečně)

3 Výpočet disparity

V této kapitole se podrobně podíváme na možné způsoby výpočtů disparity mezi jednotlivými body v obraze. Pro hledání disparity můžeme použít nejen algoritmy na to specializované, které vyžadují rektifikované a upravené snímky, ale také metody pro hledání a sledování významných bodů v obraze. Podrobně se podíváme na to, jak jednotlivé algoritmy fungují a na jakém principu rozhodují vzájemnou náležitost jednotlivých pixelů obrazu.

Než se podíváme na samotné algoritmy, připomeňme si nejdříve, co to vlastně disparita je a jaký význam má pro rekonstrukci trojrozměrné scény ze stereo snímku. Z kapitoly o triangulaci si pamatujeme, že obrazy bodu, které vidíme jsou pouze průnikem paprsků směřujícího z pozorovaného bodu do středu projekce jednotlivých kamer a obrazových rovin odpovídajících kamer. Trojúhelník tvořený středy projekcí a pozorovaným bodem je podobný trojúhelníku tvořeném průsečíky na obrazových rovinách. Základna tohoto trojúhelníku je **rovna rozdílu disparity** a vzdálenosti středů projekce kamer.

Disparita je zároveň elegantním způsobem, jak pro každý bod v obraze určit jeho odpovídající protějšek v obraze druhé kamery, zároveň mapa disparity vzhledem ke své dvourozměrnosti může být snadno normalizována a zobrazena pro kontrolu.

Jednotlivé algoritmy si rozdělíme do skupin podle jejich použitelnosti na jednotlivé body nebo celý obrázek. Do skupiny použitelné pro jednotlivé body patří jakékoli algoritmy schopné vyhledávat v obraze odpovídající si body. V tomhle textu se budeme zabývat algoritmem SURF. Do skupiny algoritmů použitelných na celé obrázky jsou Block matching (Kurt Konolige), Graph Cuts (Kolmogorov a Zabin) Pixel to pixel (Birchfield a Tomasi). Podívejme se nyní postupně na jednotlivé algoritmy.

3.1 Pixel to pixel Stereo

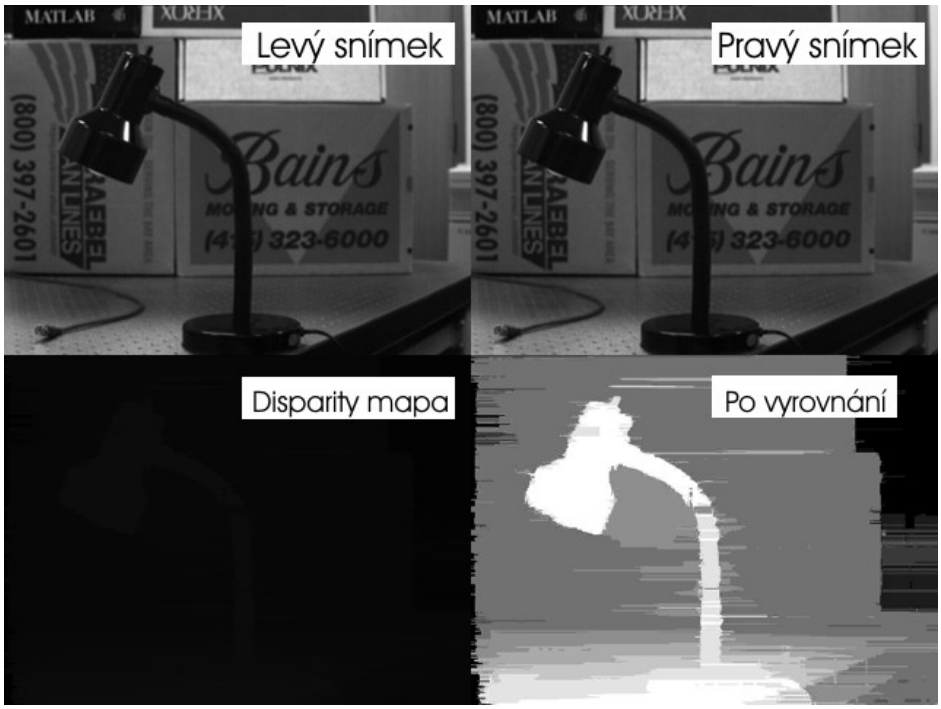
Původní algoritmus pixel to pixel stereo byl určen především pro použití v při detekci hran objektů. Proto pro tento algoritmus bylo důležité hlavně zachování detailů na obou obrázcích, aby mohly být hranice mezi objekty detekovány, musí být co nejvýraznější a tedy ostré. Algoritmus neaplikuje na vstupní obrázky žádnou filtraci. Pouze sekvenčně porovnává vždy dva pixely. Mapa disparity vytvořená tímto algoritmem tedy není příliš vhodný pro účely rekonstrukce trojrozměrné scény, ale rozhodně stojí za to jej vyzkoušet.

Algoritmus vnímá problém stereo korespondence bodů jako problém odpovídajících si pixelů v horizontální linii. Proto je nezbytně nutné, aby byly obrazy rektifkované a snímané kalibrovanými

kamerami. Vzhledem k účelu algoritmu je pixelové rozlišení dostatečné pro výpočet mapy disparity. Korespondence je zakódovaná jako sekvence uspořádaných dvojic (x_l, x_r) značící, že intenzity pixelů na x_l a x_r jsou obrazy jednoho bodu. Nespárované pixely jsou označeny jako překryté. Disparita je pak vypočítána jako x_l-x_r . Každé sekvenci je potom přiřazena hodnota γ(M), která ukazuje, jak je nepravděpodobné, že sekvence ukazuje obsahuje správně spárované pixely. Pro γ(M) vyhodnocení se používá jednoduchá hodnotící funkce přiřazující každému překrytí postih a odměnu za každý správně přiřazený pixel:

$$\gamma(M) = N_{occ}\,\kappa_{occ} - N_m\,\kappa_r + \sum_{i=1}^{N_m} d\,(x_{li}, x_{ri}) \tag{3.1}$$

κ_{occ} je velikost postihu za překryv, N_{occ} je počet překryvů, N_m je počet odpovídajících si pixelů v sekvenci a κ_r . Tato funkce způsobuje, že se může stát, že celému objektu je přiřazena jedna disparita, je to proto, že každá změna disparity znamená mimo jiné i překryv, takže pokud se hodnoty pixelů začínají lišit, je pravděpodobné, že bude započata nová sekvence.



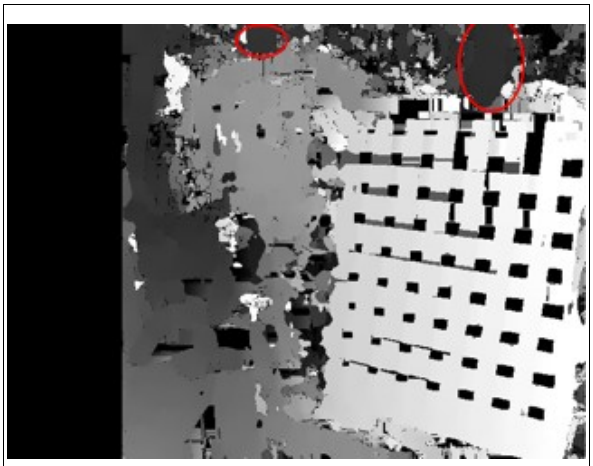
Ilustrace 24: pixel to pixel stereo

Tato funkce je v opencv implementována jako *cvFindStereoCorrespondence* s třetím parametrem CV_DISPARITY_BIRCHFIELD. Ostatní jsou 2 zdrojové obrázky, jeden výstupní . Číselné parametry jsou podle pořadí: postih za překryv, odměna za souhlas, definice sady s velkou spolehlivostí, definice sady se střední spolehlivostí, definice sady s malou spolehlivostí.

3.2 Block matching

V OpenCV je implementována metoda porovnávání bloků od Kurta Konoligeho. Funguje na principu oken součtu absolutního rozdílu. Tento algoritmus vykazuje rozdílné vysledky pro různě texturované scény. Čím je scéna složitější, tím je algoritmus lepší a tím více bodů na scéně bude mít správně vypočítanou disparitu. Příkladem takové scény může být třeba objekt položený před knihovnout, kde jsou jednotlivé knížky rozpoznatelné, nebo scéna obsahující přírodní objekdy, které mají obvykle bohatě texturovaný povrch. Míně vhodný je tento algoritmus do umělých prostředí, kde se vyskytují velké plochy bez výrazných textur, klasický jde o hladké zdi v pokoji a tak podobně.
Funkce algoritmu se skládá ze tří fází:

1. filtrování vstupních snímků, vyvážení jasu a zlepšení ostrosti textur
2. hledání odpovídajícího okna na horizontální epipolární linii
3. filtrování pro odstranění chybných výsledků



Ilustrace 25: disparita metodou block matching (zvýrazněná přidáno dodatečně)

V první fázi jsou obrazy normalizovány, aby se redukovaly rozdíly v osvětlení. V druhé fázi probíhá samotná detekce s pomocí posouvajícího se okna. Podotkněme, že tento algoritmus vyžaduje, aby vstupní obrázky byly rektifikované, jinak není možné se pohybem po x souřadnici obrazu pohybovat zároveň i po epipolární linii vztažené k aktuálnímu bodu na levém snímku. Pro rovnoběžně umístěné kamery odpovídá nulová disparita nekonečnu, protože procházející paprsky jsou rovnoběžné a tedy se protnou v nekonečnu. Nastavením minimální disparity a počtu disparit se vytvoří oblast, která má být prohledávána . Tímto je možné algoritmus nastavit pro snímání objektů pouze v určité vzdálenosti. Pokud nebude disparita daného bodu intervalu <minimální disparita; minimální disparita + počet disparit>, bude bod označen jako ležící v nekonečnu. Podle vzorce, který jsme si uvedli už dříve

$$Z = \frac{cf}{d} \tag{3.2}$$

si snadno můžeme vypočítat vzdálenosti, pro jaké bude algoritmus vytvářet mapu. Kromě toho algoritmus sleduje, v jakém pořadí se shody nacházejí. Překrytí může sice způsobit, že některé pixely

nebudou mít žádný odpovídající protějšek, ale nemůže způsobit změnu pořadí shod. To znamená, že algoritmus si ověřováním pořadí shod hlídá, jestli někde nedošlo k chybě.

Na ilustraci je mapa disparit vypočítaná metodou block matching. Všimněme si prosím červeně zvýrazněných oblastí, které ukazují místa, kde se na povrchu prakticky bez textury nachází vertikálně umístěné lišty, které algoritmus správně detekuje. Černé čtvercové oblasti na šachovnici jsou způsobeny tím, že dané pixely mají příliš velkou disparitu a jsou tedy vynechány.

Algoritmus Block matching implementovaný v openCV je dobře nastavitelný rychlý algoritmus, který vypočítá disparitu pro obrázky v rozlišení 640x480 v řádu desetin, maximálně jednotek vteřin. Je proto vhodný pro tvorbu map disparity v aplikacích, kde je vyžadována především rychlost, a dobře se může uplatnit i jako algoritmus pro zjišťování překážek u mobilních robotů.. Dokladem budiž, že firma VidereDesign implementuje tento algoritmus do některých stereo hlav. Mezi nevýhody tohoto algoritmu patří bezesporu nízká přesnost a i to, že je obtížné algoritmus vhodně nastavit. Při nevhodném nastavení rychle klesá kvalita výsledné mapy.

3.3 Graph Cuts

Algoritmus staví problém hledání disparity jako minimailizaci energií použitím MRF (Markov Random Fields). Využívá toho, že disparita se ve stereo snímcích mění pouze v daném rozsahu, a sousední pixely mají rozdíl svých disparit malý vyjma pixelů na hranách objektů. K výpočtům používá Pottův energetický model

Tento algoritmus vytváři velmi přesné a přehledné mapy disparity pro rektifikované snímky. Bohužel výpočet trvá velmi dlouho a pohybuje se v řádu desítek vteřin pro obrazy v rozlišení 640x480.

Ilustrace 26: disparita metodou graph cuts

3.4 Využití SURF pro zjišťování disparity

Všechny metody, které jsme si doposud popsali vyžadují rektifikované snímky a pracují na celých obrázcích. Podívejme se teď na trochu odlišný postup, který nevyžaduje ani rektifikované snímky ani žádné filtrování. Zkusme využít pro výpočet disparity algoritmus, který se používá pro detekci objeku v obraze. Vzhledem ke svým charakteristikám se hodí algoritmus SURF.

Ilustrace 27: využití SURF při sledování pohybu kamery

SURF znamená Speeded Up Robust Features a jak název napovídá jde o algoritmus na detekci významných bodů v obraze. To, že jsou tyto body robustní znamená, že je velká pravděpodobnost jejich opětovné detekce v jiném obraze.

Vzhledem k tomu, že algoritmus zjišťování a porovnávání bodů je velmi rychlý, je tato metoda vhodná i pro porovnávání pozice kamery snímající to samou scénu. Ilutrace 27 ukazuje, jaké body jsou algoritmem detekovány. Zeleně jsou zvýrazněny body, které nemají v předchozím snímku žádný odpovídající bod, červené linky spojují spárované body.

Toho lze využít při zjišťování disparity daných bodů. Pokud máme k dispozici 2 snímky, můžeme body v nich detekovaná vzájemně porovnat. Jejich disparita pak bude tvořena velikostí vektoru rozdílu souřadnic těchto bodů. Jediným problémem, který by při této metodě mohl nastat je špatně detekovaný pár. Proto by bylo potřeba zjistit, jestli vektor vycházející z daného bodu odpovídá směrově vektorům, vycházejícím z bodů v určitém okolí.

4 Popis implementace programu

4.1 Požadavky na program a diskuze nad problémy

Cílem této práce je vytvořit program, který bude schopen s pomocí stereokamery vytvářet 3D model prostředí. Přitom máme vycházet z význačných bodů detekovaných v obraze a reprezentovat nepřesnosti v učení polohy bodu.

Celý problém se dá rozdělit na 3 podproblémy a ty dále na několik dílčích problémů:

- Získání stereo snímku
  - Zpřístupnění kamery
  - Získání snímku z video streamu
  - Kalibrace kamery
  - Odstranění deformace a rektifikace snímků
  - Sledování pohybu kamery
- Zpracování stereo snímku
  - Detekce významných bodů v obou snímcích
  - Nalezení párů významných bodů
  - Filtrování chybných párů
  - Nalezení mapy disparity
  - Korekce významných bodů
  - Přepočet souřadnic 2D bodů na 3D souřadnice modelu
- Vytvoření a zobrazení modelu.
  - Vytvoření polygonové sítě modelu
  - Potažení modelu texturou
  - Zobrazení modelu podle nastavené polohy kamery

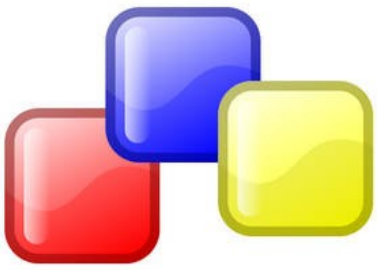
Z tohoto rozdělení se naskýtá rozdělení programu na tři vzájemně propojené celky, které by mohly být reprezentovány třídami. Každá ze tříd bude zapouzdřovat funkce nutné k řešení jednotlivých podproblémů. Aplikace by měla běžet na OS windows, a měla by být uživatelsky přívětivá, takže aplikace by měla být formulářového typu s možnostmi zobrazování aktuálních snímků na kamerách a renderování třírozměrné scény. Vzhledem k tomu, že všechny tyto operace jsou výkonově náročné a je pravděpodobné, že aplikace bude zpracovávat větší množství dat, bude lepší ji implementovat v kompilovaném jazyce, jako je C nebo C++... Bylo by vhodné využít už existujících knihoven, protože jinak bychom zabředli do psaní přístupu k webkamerám, directX apod.

Vzhledem k tomu, že existuje knihovna přímo určená k psaní takových aplikací týkajících se zpracovaní videa a počítačového vidění, bude při tvorbě tohoto programu využita. Její název je OpenCV. Vzhledem k tomu, že po mých špatných zkušenostech jsem nechtěl kombinovat Visual studio a OpenCV, bylo potřeba najít jiné prostředí, ve kterém by mohla být aplikace vytvořena. Volba padla na wxDev C++, které je vývojovým nástupcem už osvědčeného Dev C++. Zobrazování bude vzhledem bezproblémovosti tvorby aplikací pro OpenGL prováděno právě touto knihovnou. Přibližme si nyní zmiňované knihovny wxWidgets, OpenGL a OpenCV .

4.2 Využívané prostředky a knihovny

4.2.1 Wxwidgets

Knihovna wxWidgets je multiplatformní knihovna pro tvorbu formulářových aplikací. Je přenositelná, ale přesto aplikace v ní vytvořené vypadají nativně. Je použitelná pro C++, Python, Perl, Ruby a Javu. Tuto knihovnu používá spousta multiplatformních aplikací, z nihž můžeme uvést jako příklad VLC media player, BitTorrent a podobně. Multiplatformní možnosti této knihovny sice nebudou v našem případě pravděpodobně využity, přesto ale je knihovna vhodná zejména pro jednoduchost práce s ní. Mezi další výhody této knihovny patří to, že je volně šiřitelná a dobře dostupná ke stažení.



Ilustrace 28: logo wxWidgets

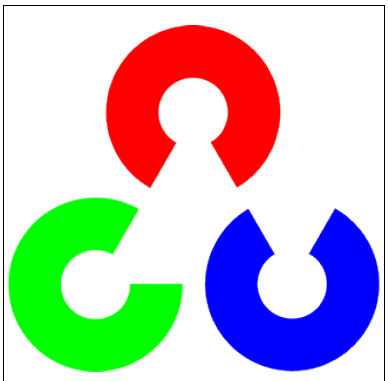
Vývoj wxWidgets začal v roce 1992 na University of Edinburgh. Původně začala jako projekt pro vytváření přenosttelných aplikací mezi *nixovymi systémy a Windows.

4.2.2 OpenCV

OpenCV je zkratka pro Open Computer Vision library. Jde o knihovnu specializovanou na počítačové vidění a zpracování obrazu. Knihovna je multiplatformní a dá se stáhnout ve verízh pro Linux, Windows, OS X, přičemž se vyvíjí verze pro Python, Ruby, Matlab a jiné.

Vývoj OpenCV začal původně jako výzkumný záměr firmy Intel pokročit ve vývoji takových výpočetně náročných operací, jako jsou ray-tracing v reálném čase a 3D zobrazovací panely. Mezi hlavní přispěvatele projektu patřil mimo jiné tým vyvíjející Intel Performance Library a veliký počet ruských optimalizačních expertů. V počátečních fázích byly cíle projektu definovány jako:

- Pokrok ve výzkumu počítačového vidění poskytnutím nejen otevřeného, ale také optimalizovaného kódu, pro základní úkony počítačového vidění tak, aby už nebylo nutné dokola vyvíjet to, co už bylo vymyšleno.
- Rozšíření znalostí o počítačovém vidění poskytnutím základní infrastruktury, na které by mohli vývojáři dále vyvíjet své aplikace
- Umožnit pokrok komerčních aplikací týkajících se počítačového vidění



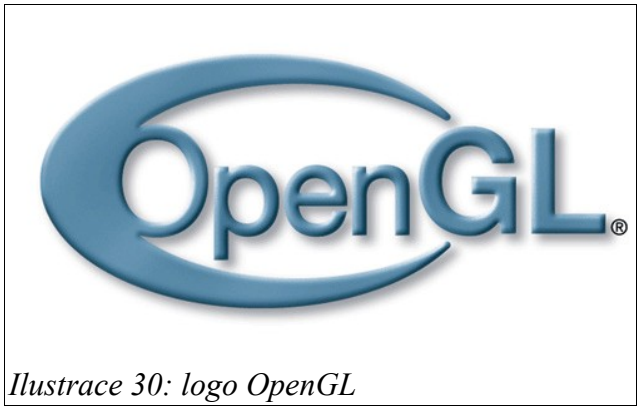
Ilustrace 29: logo OpenCV

První alfa verze OpenCV byla vydána v roce 2000 na IEEE Conference on Computer Vision and Pattern recognition. Dále následovalo několik beta verzí v letech 2001 – 2005. Verze 1.0 byla vydána v roce 2006 a její vývoj byl oficiálně ukončen.

V roce 2008 se dostalo OpenCV podpory od Willow Garage, která od té doby pokračuje ve vývoji. Pre – release verze byla vydána říjnu 2008.

4.2.3 OpenGL

OpenGL je zkratkou pro Open Graphics Library, byla navržena firmou Silicon Graphics inc. Jako rozhraní k akcelerovaným grafickým kartám, nebo grafickým subsystémům. Byla navržena s důrazem na to, aby byla použitelná na různých typech grafických akcelerátorů a aby bylo možné ji použít i v případě, že grafický akcelerátor není dostupný. Knihovna OpenGL je použitelná na Linuxu, OS/2 a Windows. Z programátorského hlediska je knihovna navržena tak, aby byla použitelná takřka v libovolném programovacím jazyce, primárně jsou však k dispozici hlavičkový soubor pro C/C++.



Ilustrace 30: logo OpenGL

OpenGL se chová jako stavový automat. To znamená, že knihovna si uchovává vnitřní nastavení, dokud není změněno. To umožňuje změnu nastavení vykreslování během vykreslování. Další výhodou je, že funkce OpenGL mají menší počet parametrů. Vykreslování samotné se provádí procedurálně. Postupným voláním funkcí se vykreslí výsledný rastr, který je uložen ve framebufferu.

Přes svoji přenositelnost ovšem OpenGL nezaručuje, že při použití identického programu na různých platformách dosáhneme stejného výsledku, protože na různých platformách jsou čísla reprezentována odlišnými způsoby.

Vykreslování v OpenGL se provádí s pomocí primitiv, což jsou základní tělesa, ze kterých se skládají složitější objekty a tedy i scény. Na výsledné objekty (ale i primitiva) lze aplikovat transformace, jako je rotace, translace či změna měřítka.

Z hlediska datové reprezentace poskytuje OpenGL samotné pouze základní rozhraní. Existují však různé rozšiřující knihovny. Mezi nejznámější rozšiřující knihovnu patří GLU, kterou budeme v tomto při tvorbě programu využívat.

4.3 Implementace

V této části textu se budeme zabývat samotnou implementací programu. Bude rozdělena na tři hlavní části týkající se implementace jednotlivých tříd. Na úvod zopakujme, že aplikace bude vytvořena v jazyce C++ s využitím knihoven wxWidgets, OpenCV a OpenGL. Půjde o aplikaci formulářového typu.

4.3.1 Získání stereo snímku

Získaní stereo snímku je základním problémem, který je nutné vyřešit. Jako zdroj dat můžeme považovat buďto živý vstup z kamery nebo nebo soubory z disku. Co se vstupu z kamery týče, musíme počítat jak s kamerami připojenými přes USB, tak s kamerami připojenými přes FireWire (v našem případě kamera Videre Design). Třída, kterou se budeme zabývat musí být schopna získávat data ze všech vyjmenovaných vstupů a navíc tato data upravovat a zobrazovat. Zkrátka musí zapouzdřovat všechna data a operace týkající se práce s kamerou. Protože stereokamera, přestože umístěna na jeden rám, je vlastně dvojice samostatných kamer, budeme ke každé kameře přistupovat zvlášť a tedy bude tato třída napsána pouze pro práci s jednou kamerou. Proto bude pro každou kameru zvlášť vytvořena nová instance této třídy.

4.3.1.1 Získání snímku

Pro získání snímku vytvoříme funkci, která bude zapouzdřovat práci se soubory, USB a FireWire. Funkce bude při svém spuštění pouze kontrolovat, v jakém režimu se nachází a podle toho bude odebírat data a ukládat je do vnitřního bufferu objektu. Pro čtení z USB kamery bude použita funkce openCV *cvQueryFrame*. OpenCV obsahuje i funkci pro práci se soubory, takže pro načtení obrázku ze souboru bude použita funkce *cvLoadImage* . Problém nastává při čtení dat z FireWire, z kterého OpenCV nedokáže nativně číst. Pro vyřešení tohoto problému využijeme volně dostupného

ovladače pro FireWire kamery **CMU 1394**, s pomocí kterého dokážeme přistupovat ke kameře. Naštěstí formát dat získaný tímto ovladačem je kompatibilní s formátem dat používaným OpenCV, takže pro získaní snímku nám stačí pouze inicializovat hlavičku IplImage ukazatelem na získaná data a dále jsme schopni s těmito daty zacházet jako s obyčejným obrázkem. Mějme však na paměti, že ukazatel ukazuje na buffer snímku v paměti, to znamená, že data v něm se mění. Musíme tedy stejně jako při práci s USB kamerou data zkopírovat do jiné oblasti paměti a tam je poté zpracovávat. Krom toho, že tato funkce obstarává získání obrázku, musí být také schopna tento obrázek upravovat, hlavně upravovat podle map pro rektifikaci a odstranění deformace obrazu. Zaveďme tedy další kontrolní proměnnou, která bude sloužit jako přepínač postprocessingu. Další věc, na kterou si musíme dát pozor jsou rozměry načítaných dat. Uživatel aplikace má možnost si zvolit, v jakém rozlišení chce data snímat a dále zpracovávat. Proto budou všechny nasnímané obrázky smrštěny (nebo roztaženy) tak, aby bylo splněno nastavení od uživatel. Samozřejmě pokud si uživatel zvolí nějaké rozlišení snímání dat, pokusí se aplikace změnit nastavení kamery na zvolené rozlišení.

4.3.1.2 Vykreslování videa

Poté co jsme schopni načíst data do paměti, bylo by dobré, kdybychom byli schopni je periodicky vykreslovat na obrazovku. Vzhledem k tomu, že periodické vykreslování znamená fungování v uzavřeném cyklu, musíme umožnit běh této třídy jako samostatného vlákna. Knihovna OpenCV umožňuje vykreslování obrázku svými funkcemi, ovšem pouze do okna vytvořeného OpenCV, což se nám nehodí. Potřebujeme, aby se video vykreslovalo v určité části aplikace a to v okně, kde se budou nacházet i jiné prvky než samotný obrázek. Proto musíme najít způsob, jakým provést vykreslovaný obrázek do podoby, které rozumí wxWidgets. Tímto formátem dat není nic jiného, než pole hodnot RGB. Proto nejprve z obrázku extrahujeme RGB data a nad těmito daty inicializujeme strukturu obrázku wxWidgets. Poté co jsou data převedena už není problém použít je jako pozadí pro blokový prvek uživatelského prostředí a tak vytvořit okénko se zobrazovaným videem.

Další věc, kterou by funkce měla umět je přidání informací do obrázku, čehož snadno dosáhneme tím, že do obrázku přidáme text, opět díky funkci OpenCV. Musíme si přitom ale dávat pozor, abychom tímto přidáním informací neznehodnotili obrázek, se kterým se možná ještě bude pracovat, proto vnitřní obrázek při zavolání této funkce nejprve zkopírujeme a potom teprve budeme provádět jednotlivé úpravy.

4.3.1.3 Sledování pohybu kamery

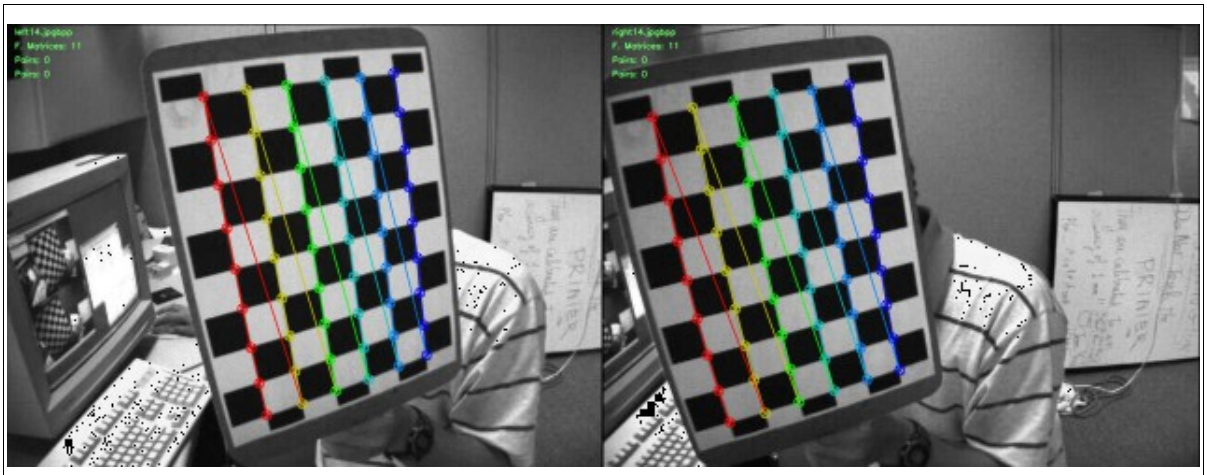
Kamera by měla být schopna sledovat vlastní pohyb. Proto bude mít implementovánu funkčnost možnost využít detektoru SURF pro zjištění a sledování jednotlivých bodů v obraze. Kamera bude mít 2 buffery pro uložení deskriptorů a souřadnic detekovaných bodů. Jeden pro

aktuální snímek a jeden pro předchozí. Vzhledem k tomu, že by bylo neefektivní přesouvat data mezi těmito buffery při každém snímku, bude si kamera držet záznam o tom, jestli se provádí sudé nebo liché snímání a podle toho bude přistupovat k bufferu 0 nebo 1. Přehození bufferu se provádět jednoduchým odečtením aktuálního stavu od jedničky, takže bude zajištěno střídání po snímku. Jako malý dodatek bude kamera schopna vykreslovat sledované body a jejich spojení s odpovídajícími body v předchozím snímku. Tato funkce je časově a výpočetně náročná, proto bude implicitně vypnuta a uživateli bude ponechána možnost ji zapnout.

4.3.1.4 Kalibrace

Vzhledem k tomu, že třída zapouzdřující práci s kamerou dokáže pracovat pouze s jednou kamerou, je nutné pro kalibraci vytvořit jinou třídu, která bude schopna ovládat obě kamery a pracovat s jejich daty. Tato třída bude provádět jenom jednu jedinou věc a tou je kalibrace stereokamery. Tudíž musí nejprve zastavit zobrazování videa, přepnout režim získávání dat a poté provést samotnou kalibraci. Ke kalibraci je použita funkce *cvStereoCalibrate*. Poté co proběhne kalibrace ještě funkce inicializuje mapy pro korekci snímků jednotlivých kamer a společně s maticemi výstupními maticemi je předá zpět do tříd zodpovědných za získávání videa a nastaví, že mají získané snímky automaticky upravovat.

Po implementaci těchto dvou tříd máme k dispozici dostačující zdroj dat umožňující kdykoli dodat potřebné snímky bez toho, abychom se museli starat o to, jakým způsobem jsou snímky obstarány. V případě, že aplikace zrovna neprovádí žádné výpočty, je zobrazováno aktuální video.



Ilustrace 31: probíhající kalibrace s pomocí obrázků ze souboru

4.3.2 Zpracování stereo snímku

V této kapitole se budeme věnovat různým způsobům zpracování stereo snímků. Půjde především o zjišťování map disparit a hledání odpovídajících si bodů. Funkce této třídy má smysl až v případě, že máme k dispozici rektifikované snímky. Podotkněme, že tento objekt bude schopen převzít kontrolu nad oběma kamerami, zastavit jejich automatické překreslování, provést výpočty a poté je zase spustit.

4.3.2.1 Detekce významných bodů

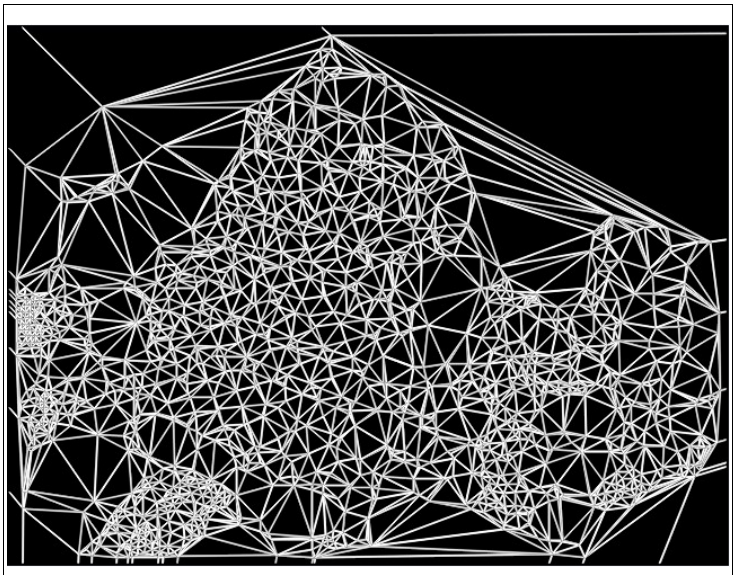
Pro detekci významných bodů v obou obrazech je použit, stejně jako u detekce pohybu kamer algoritmus SURF. Tato třída je schopna použít už vypočítané hodnoty v bufferech jednotlivých objektů kamery nebo vypočítat nové.

4.3.2.2 Sestavení mapy disparity

Pokud bude uživatel chtít, může pro zpřesnění vykreslované mapy využít vytváření podle mapy disparity. Vzhledem k tomu, že jednotlivé algoritmy poskytují značně se lišící výsledky, musí být implementovány všechny. Nastavení těchto algoritmů je je opět na uživateli.

4.3.2.3 Projekce do 3D

Poté, co jsou k dispozici vybrané významné body a mapa disparit, provedeme zavoláním funkce *cvReprojectImageTo3D* projekci mapy disparity do třírozměrných souřadnic. Tím dostaneme k dispozici mapu bodů nacházejících se v třírozměrném souřadném systému. Shrňme si nyní situaci. Máme k dispozici vybrané body a mapu disparit. Pro naše účely tohle ale nestačí, potřebujeme jednotlivé body propojit do polygonů a tak ještě provedeme Delaunayovu triangulaci. OpenCV dokáže provádět Delaunayovu triangulaci pouze v dvourozměrném systému, proto bude triangulace provedena nad množinou detekovaných

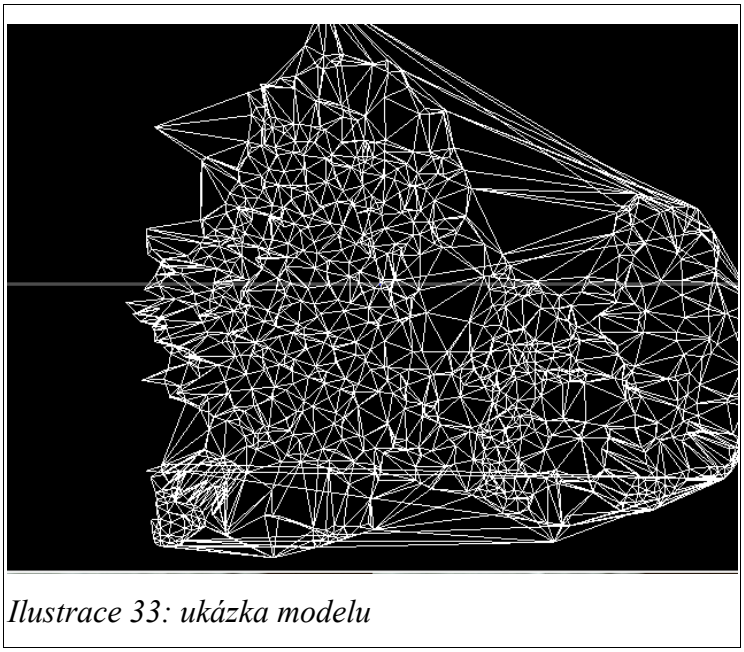


Ilustrace 32: 2D trojúhelníková síť

významných bodů, čímž nám vznikne množina hran mezi těmito body. Abychom zjistili, jaké body odpovídají jaké hraně, musíme body dohledávat tím způsobem, že pro každou jednotlivou hranu se podíváme, jaké jsou souřadnice bodů, které spojuje ve 2D a souřadnice 3D bodů, které odpovídají těmto bodům , můžeme zjistit tak, že se podáváme na hodnoty 3D souřadnic v mapě na souřadnicích 2D bodu hrany. Postupným procházením jednotlivých trojúhelníků jsme pak schopni vytvořit matici trojúhelníků, kterými bude tvořen povrch modelu. Kromě toho musí být v matici těchto bodů ještě souřadnice pro texturování. Jakmile máme sestavenou i tuto strukturu, můžeme ji spolu s nastavením kamery a aktuálním snímkem z levé kamery předat objektu zajišťujícím vykreslení modelu. Na přiložené ilustraci je dobře vidět, jakým způsobem bude polygonální síť rozdělena.

4.3.2.4 Vykreslování modelu

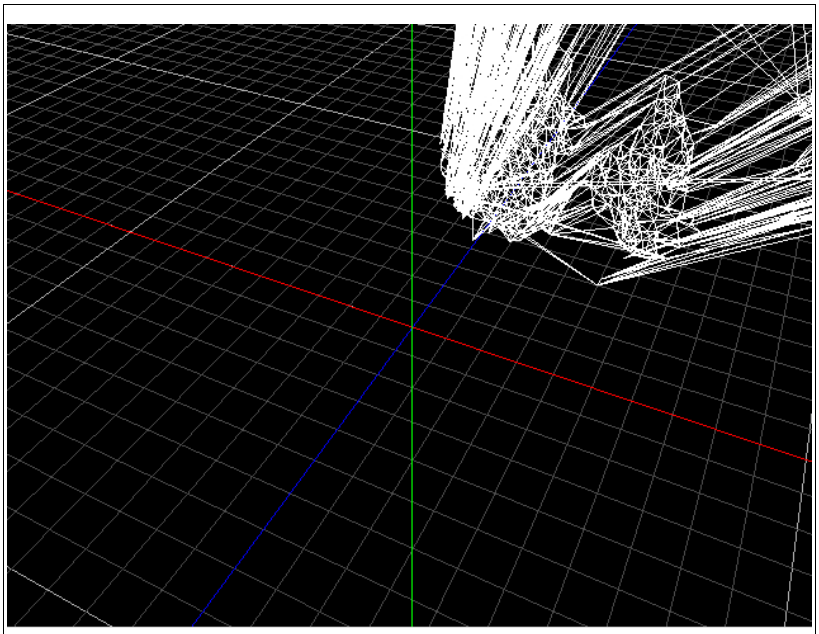
Vykreslování modelu je prováděno s pomocí OpenGL. K dispozici dostaneme matici polygonů objektu, takže jediné, co zbývá je zapouzdření těchto dat do jednoho objektu. Tyto objekty, můžeme je nazývat fragmenty modelu, budou poté sekvenčně vykreslovány tím způsobem, že se vždy nejdříve nastaví vykreslování podle nastavení kamery, obsahujícího translační vektor a rotační matici. Poté je provedeno vykreslení fragmentu modelu v cyklu po jednotlivých polygonech. Na ilustraci je vidět už hotový model vzniklý ze stereo snímku s šachovnicí. Z tohoto úhlu je zřetelně vidět vyčnívající obdélníkový objekt v popředí. Pokud bychom změnili úhel pohledu, naskytl by se nám pohled, jako je na ilustraci 34. Tam je zcela zřetelně vidět, že objekt před kamerou je obdélníkový objekt a že blízko za ním se nachází další objekt. Z tohoto pohledu dokonce můžeme říci, že přibližně 60 cm před kamerou je šachovnice a další objekt se nachází 30cm za ní. Kamera se nachází ve středu souřadného systému.



Ilustrace 33: ukázka modelu

5 Závěr a shrnutí výsledků

Závěrem lze říci, že s pomocí stereokamery lze dobře vytvářet modely prostředí, kde jednotlivé objekty vystupují z pozadí, a zároveň lze i sledovat povrchové vlastnosti těchto objektů, což činí ze stereokamery nástroj vhodný pro navigaci autonomních robotů. Pokud jsme ochotni věnovat výpočtu více času (nebo máme k dispozici rychlejší procesor), můžeme vytvořit i poměrně přesné modely okolního prostředí.



Ilustrace 34: ukázka modelu, pohled s odstupem

Z hlediska přesnosti je snímání stereokamerou nejvíce zatíženo odchylkami při výpočtu map disparity a kvalitou kalibrace. Bohužel ale i při sebelepším nastaveném aplikace i kamery vznikne hodně šumových dat, které sice částečně odfiltrovat (například body se zápornou vzdáleností), ale přesto obsahují výsledná hodně šumu.

Další věc, se kterou se musíme umět vyrovnat je klesající přesnost stereoskopického systému s klesající disparitou. V našem programu provádíme znázornění této nepřesnosti nejjednodušším způsobem a to tak, že pro každý bod se vypočítají souřadnice bodů, které mají vždy o 1 menší a větší disparitu. Protože možná odchylka se udává jako hodnota nejmenšího dílku, je nalezen střed vzdálenosti těchto bodů a toho je poté vykreslena odchylka.

Knihovna OpenCV se ukázala jako nenahraditelný a neocenitelný nástroj při práci se stereo kamerou a zpracování stereo snímků jednak pro rychlost s jakou je schopna zpracovávat získaná data a také proto, že je volně dostupná, což z ní při její kvalitě činí perfektní nástroj nejen pro výzkumné týmy, ale i domácí nadšence a kutily zabývající se robotikou a počítačovým viděním.

Zjistili jsme také, že přesnost výsledného modelu je značně ovlivněna každou činností, která je s modelem prováděna. Pokud ale dokážeme stereo snímek správně zpracovat, jsme schopni dosáhnout přesnosti v řádu centimetrů pro blízké objekty.

Tato práce může vzhledem k její ojedinělosti dobře sloužit všem, kteří se někdy budou pokoušet o rekonstrukci obrazu s pomocí stereokamery, protože ukazuje shrnutí řešení problémů, se kterými se lidé zabývající se rekonstrukcí scény můžou setkat. Úspěšně se nám podařilo nejen položit matematický základ tomu, jednotlivým výpočtům, ale i uvést pořadí funkcí a shrnutí jejich významu ať už při získávání stereo snímků, nebo při samotné rekonstrukci.

Literatura

[1] Internetová encyklopedie *Wikipedia*: [online]
URL: <www.wikipedia.org> [cit. 2009-20-5]

[2] Internetové noviny *Digimanie* [online]
URL: <www.digimanie.cz> [cit. 2009-22-5]

[3] Internetové noviny *Elektrorevue* [online]
URL: <www.elektrorevue.cz> [cit. 2009-24-5]

[3] Internetové noviny *Root* [online]
URL: <www.root.cz> [cit. 2009-24-5]

[4] Stan Birchfield, Carlo Tomasi: *Depth Discontinuities by Pixel-to-Pixel Stereo* [online]
URL: <http://ai.stanford.edu/~birch/publications/p2p_iccv1998.pdf>

[5] Sudipta N. Sinha: *Graph Cut algorithms in Vision, Graphics and machine learning* [online]
URL: <http://cs.unc.edu/~ssinha/pubs/SinhaGraphCutsIP2004.pdf > [cit. 2009-22-5]

[6] Gary Rost Bradski, Adrian Kaehler: *Learning OpenCV, Computer Vision with OpenCV Library*. O'Reilly, 2008, ISBN 0-596-51613-4

[7] OpenCV, *online dokumentace* [online]
URL: <http://opencv.willowgarage.com/wiki/> [cit. 2009-24-5]

Seznam příloh

Příloha 1. CD obsahující:

- zdrojové texty
- manuál k programu
- programovou dokumentaci