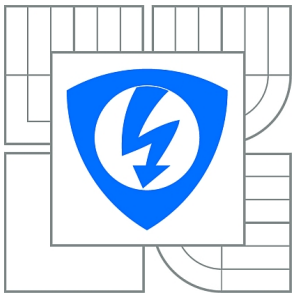
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SOUSTAVA KAMER JAKO STEREOSKOPICKÝ SENZOR PRO MĚŘENÍ VZDÁLENOSTI V REÁLNÉM ČASE

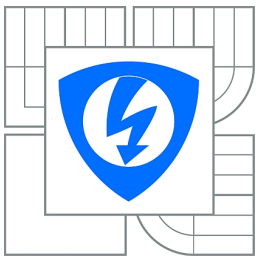
REAL-TIME DISTANCE MEASUREMENT WITH STEREOSCOPIC SENSOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MARTIN JANEČEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN HASMANDA
SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Martin Janeček	***ID:***	109664
Ročník:	2	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Soustava kamer jako stereoskopický senzor pro měření vzdálenosti v reálném čase

POKYNY PRO VYPRACOVÁNÍ:

Jako stereoskopický senzor bude v diplomové práci sloužit soustava dvou kalibrovaných kamer. Cílem studenta bude za pomocí takovéto soustavy zpracovat pořízené stereoskopické snímky do podoby hloubkové mapy. Výpočet hloubkové mapy bude z důvodu rychlosti výpočtu navržen pro GPU. Z takto vypočtené hloubkové mapy a známých kalibračních parametrů se dále určí vzdálenosti předmětů před senzorem. Pro vyšší přesnost bude uvažována hloubkové mapa se sub-pixelovou přesností.

DOPORUČENÁ LITERATURA:

[1] Andrew Harltey and Andrew Zisserman. Multiple view geometry in computer vision (2. ed.) . Cambridge University Press, 2006.
[2] BRADSKI, Gary; KAEHLER, Adrian. Learning OpenCV. [s.l.] : O'Reilly, 2009.
[3] CYGANEK, B.; SIEBERT, P., J.; An Introduction to 3D Computer Vision Techniques and Algorithms: Wiley 2010.

Termín zadání: ***Termín odevzdání:*** 29.8.2014

Vedoucí práce: Ing. Martin Hasmanda
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práce ukazuje kalibraci stereoskopického senzoru. Popisuje základní metody stereo korespondence za použití knihovny OpenCV. Obsahuje výpočet disparitních map pomocí procesoru nebo grafické karty (s použitím knihovny OpenCL).

KLÍČOVÁ SLOVA

Semi Global Block Matching, Block Matching, OpenCV, OpenCL, disparitní mapa, stereoskopie, kalibrace stereoskopického senzoru.

ABSTRAKT

Project shows calibration stereoscopic sensor. Also describes basic methods stereo-corespodation using library OpenCV. Project contains calculations of disparity maps on CPU or graphic card (using library OpenCL).

KEY WORDS

Semi Global Block Matching, Block Matching, OpenCV, OpenCL, disparity map, stereovision, calibration stereoskopic sensor.

BIBLIOGRAFICKÁ CITACE DÍLA

JANEČEK, M. *Soustava kamer jako stereoskopický senzor pro měření vzdálenosti v reálném čase.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 58 s. Vedoucí diplomové práce Ing. Martin Hasmanda.

PROHLÁŠENÍ O PŮVODNOSTI PRÁCE

Prohlašuji, že jsem svou diplomovou práci na téma „**Soustava kamer jako stereoskopický senzor pro měření vzdálenosti v reálném čase**“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonu (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne 28. srpna 2014 .....

podpis autora

PODĚKOVÁNÍ

Tímto děkuji vedoucímu Diplomové práce panu Ing. Martinu Hasmandovi za cenné připomínky, rady a odborné vedení, které mně po celou dobu zpracování poskytoval.

V Brně dne 28. srpna 2014 .....

podpis autora

FACULTY OF ELECTRICAL ENGINEERING
AND COMMUNICATION

BRNO UNIVERSITY OF TECHNOLOGY
TECHNICKA 12, CZ-61600 BRNO, CZECHIA

HTTP://WWW.SIX.FEEC.VUTBR.CZ

VÝZKUM POPSANÝ V TÉTO DIPLOMOVÉ PRÁCI BYL REALIZOVÁN V LABORATOŘÍCH PODPOŘENÝCH Z PROJEKTU SIX; REGISTRAČNÍ ČÍSLO CZ.1.05/2.1.00/03.0072, OPERAČNÍ PROGRAM VÝZKUM A VÝVOJ PRO INOVACE.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM GRAFŮ

ÚVOD

Tématem, které jsem si zvolil pro diplomovou práci, je „Soustava kamer jako stereoskopický senzor pro měření vzdálenosti v reálném čase“. Hlavním cílem diplomové práce je zpracovat stereoskopické snímky do podoby disparitní mapy. K dosažení tohoto cíle bude nezbytné nastudovat metody pro pořizování stereoskopických obrazů za pomocí dvojice kamer a nalézt tak vhodná řešení pro zpracování těchto výsledných snímků.

V úvodní části práce se věnuji způsobům pořizování stereoskopických obrazů tedy způsobům snímání scény. Jde o technologie, které jsou založeny na klasickém principu vnímání člověka světa kolem něj. Uvádím a definuji dvě možné metody snímání a to Toe-in a Off-axis. V rámci praxe pracuji s metodou Off-axis. Dále se věnuji problematice epipolární geometrie, která je základem pro následnou kalibraci kamer respektive pro mapování a popis stereoskopické soustavy a pro transformaci obrazu tedy rektifikaci. V rámci této problematiky se soustředím na význam fundamentální matice tedy mapování bodu v jednom obraze na korespondující epipolární přímku ve druhém obraze. Mapování vzájemně korespondujících bodů je důležité pro stanovení projektivity snímků. Na tuto problematiku navazuje zmíněný proces kalibrace kamer a jeho popis. Tento proces je základem pro získání vnitřních a vnějších parametrů kamer. Dále se v práci věnuji rektifikaci stereoskopického obrazu, díky které dochází ke zjednodušení úlohy při přiřazení bodu na pravém snímku bodu na levém snímku stereo páru. Zmíněný proces je nezbytný pro správné určení vnitřních a vnějších parametrů kamer v rámci kalibrace. Pořízené stereoskopické snímky spolu se zjištěnými parametry kamer následně pomocí stereo korespondenční metody využívám pro vytvoření hloubkové neboli disparitní mapy. V neposlední řadě je třeba vypořádat se se sub-pixelovou přesností vytvořené disparitní mapy. V závěru práce zmiňuji problematiku paralelního programování grafického procesoru GPU. Uvádím obecně funkce a využití GPU a také komparaci funkčních vlastností CPU a GPU. Ke zpracování shora uváděného využívám funkce knihovny OpenCV. Dále zmiňuji standard OpenCL, díky kterému je možné paralelní programování na CPU a GPU. Tyto poznatky jsou pak základem pro další zpracování již vytvořené aplikace, která umožňuje zpracování pořízených snímků do disparitní mapy. Po aplikování znalostí paralelního programování a funkčností OpenCL umožní aplikace vykreslit barevnou disparitní mapu scény v reálném čase.

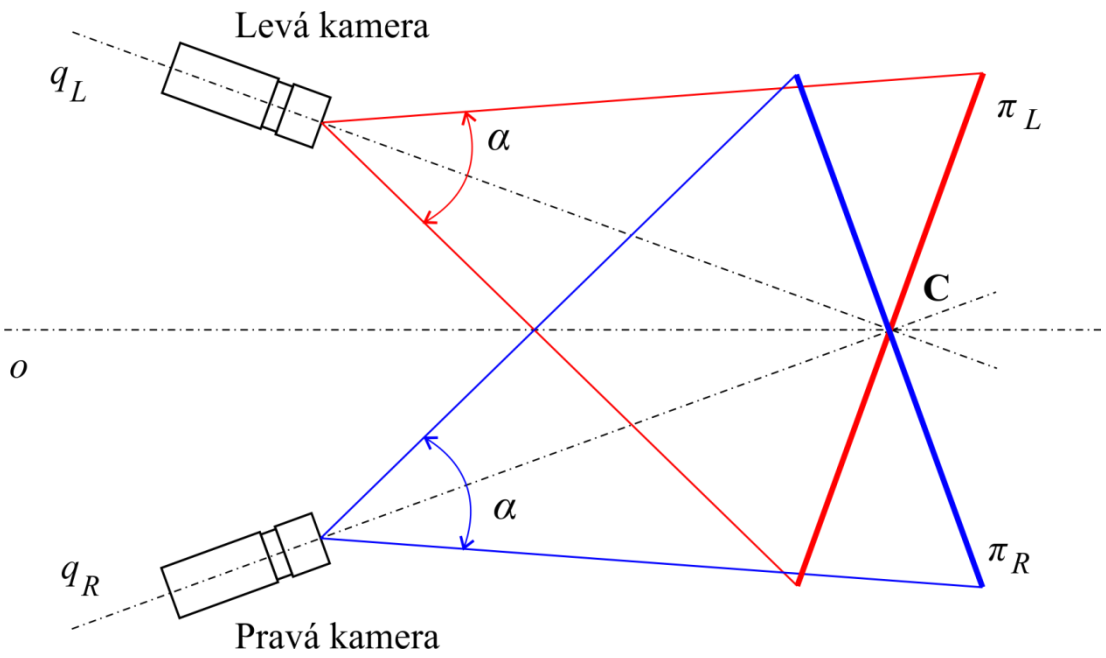
V samotném závěru práce uvádím grafy se statistickými údaji o chybách kalibrace kamerové soustavy, grafy s naměřenými chybami měření vzdálenosti, grafy rychlostí měření vzdálenosti, ukázky disparitních map a hodnotím dosažené cíle diplomové práce.

1 POŘÍZENÍ STEREOSKOPICKÝCH OBRAZŮ

Stereoskopická technologie nám umožňuje vytvořit určitý prostorový vjem. Celá technologie je založena na klasickém principu vnímání člověka světa kolem něj. Intenzita vjemu závisí na vzájemné vzdálenosti kamer a jejich vzdálenosti od průmětny. Existuje několik metod k pořízení těchto vjemů z prostoru. Každá z těchto metod vychází z různého rozestavění kamer. Ne všechny polohy kamer jsou ke snímání obrazu vhodné. V této práci uvedu dvě metody možné pro snímání prostoru dvěma kamerami. V praxi využívám metodu Off-axis.

1.1 Metoda Toe – in

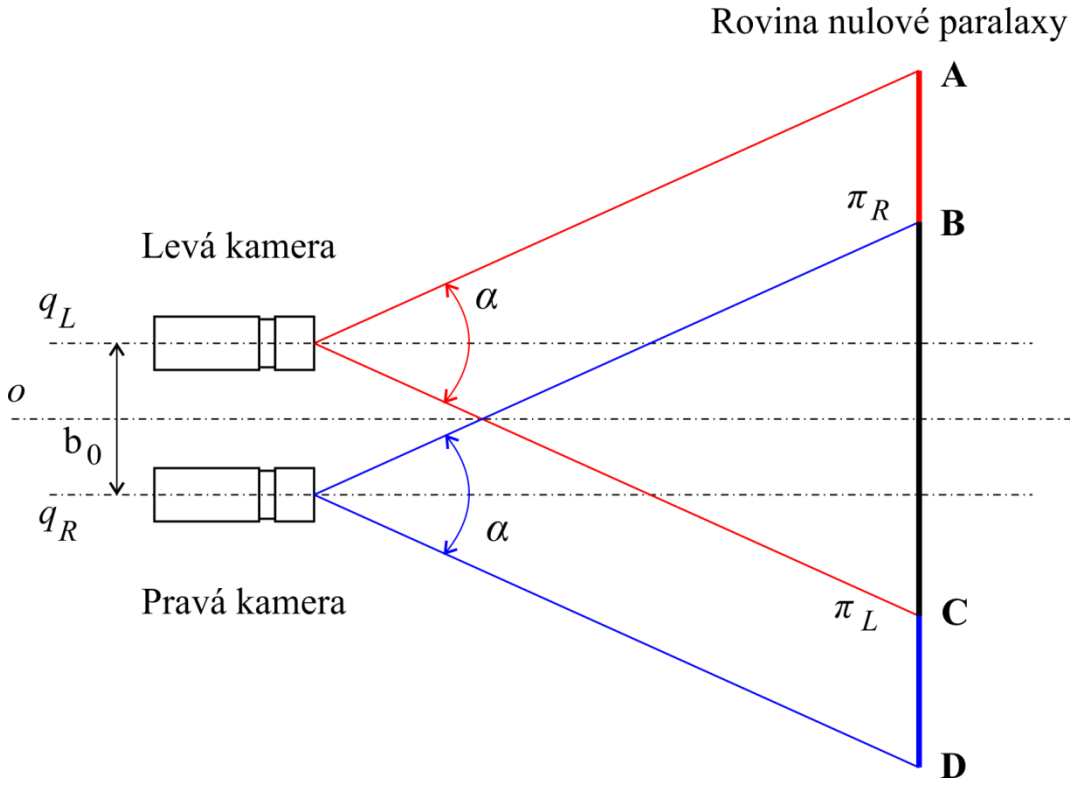
První z možných a častých metod je metoda Toe-in znázorněná na obrázku 1. Obě kamery jsou zaostřeny na totožný bod. Nevýhodou této metody je vznik vertikální paralaxy, která je následkem toho, že snímané body se nezobrazují v jedné rovině. Obraz tak může působit nepřirozeně [3].



Obr. 1: Metoda Toe-in

1.2 Metoda Off – axis

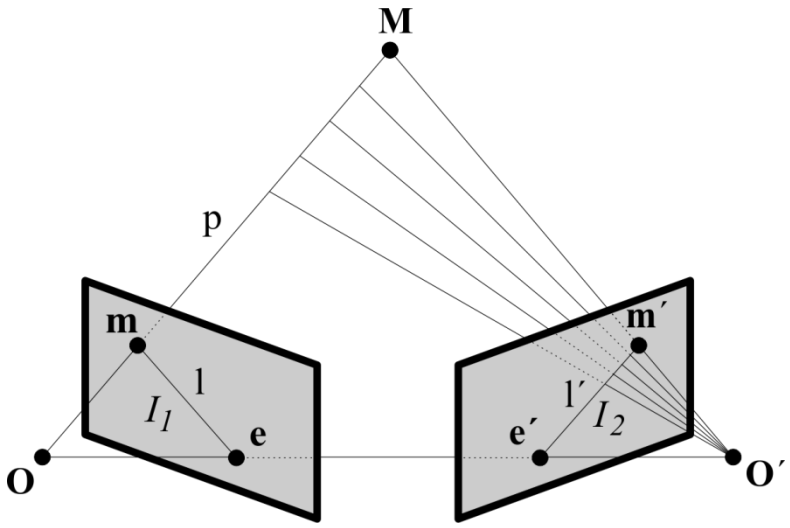
Vhodnější metodou pro snímání prostorového vjemu je metoda Off-axis ukázaná na obrázku 2. Objektivy kamer jsou nastaveny rovnoběžně a nehrozí zde proto vznik vertikální paralaxy. Obraz tedy působí přirozeně. Tato metoda je složitější v tom, že obrazy, které jsou kamerami snímány, se překrývají mezi body BC a zároveň každá z kamer snímá část obrazu, kterou druhá nesnímá [1].



Obr. 2: Metoda Off-axis

2 EPIPOLÁRNÍ GEOMETRIE

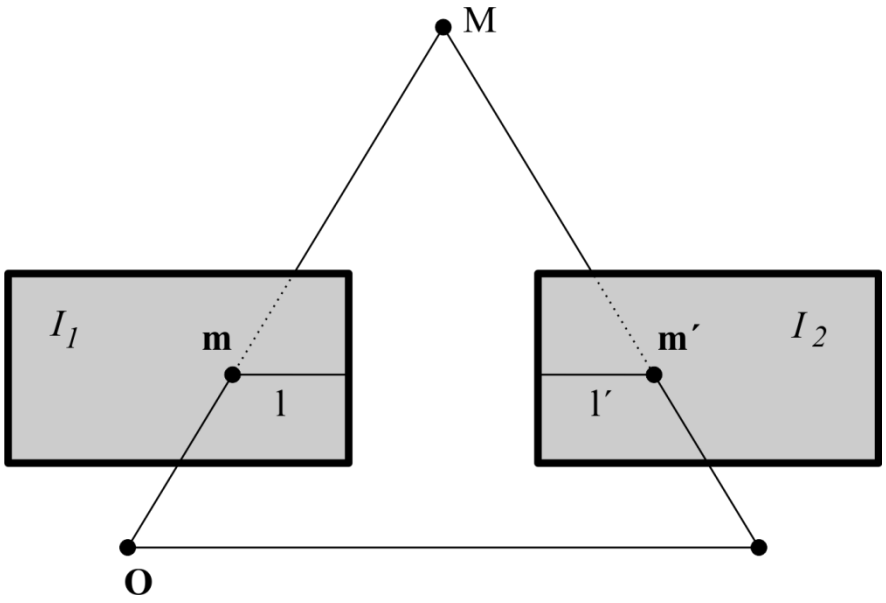
Epipolární geometrie mapuje a popisuje stereoskopickou soustavu. Tou je uspořádání protínajících se rovin, jejichž průsečíkem je báze. Bází je přímka, která spojuje středy kamer. Uspořádání rovin a nalezení báze je základem pro vyhledávání korespondujících bodů na snímcích stereopáru. Z důvodu odlišnosti scén si obrazy nemusí absolutně odpovídat. Cílem je tedy zmapovat dostatečné množství takovýchto bodů, které korespondují v obou snímcích a nalézt jejich odlišnosti. Získáme tak základ pro nalezení fundamentální matice. K dosažení cíle je rozhodující umístění obou kamer a jejich vnitřní parametry [4].



Obr. 3: Epipolární geometrie

Na uvedeném obrázku 3 je zobrazen princip epipolární geometrie. Středy kamer respektive body **O** a **O′** spojuje přímka shora nazvána jako báze. Středem kamery je bod středového promítání. Na obrázku je zobrazeno promítání bodu **M** do obrazové roviny $\mathbf{I_1}$ a $\mathbf{I_2}$. V rámci roviny $\mathbf{I_1}$ je bod **M** označen jako m a v rámci roviny $\mathbf{I_2}$ jako **m′**. Všechny uvedené body (**O**, **O′**, **M**, **m**, **m′**) leží v jedné epipolární rovině. Tato skutečnost je důležitá pro zpracování stereo páru na základě korespondencí. Takových rovin existuje nekonečně mnoho. Soubor těchto rovin je nazýván jako epipolární svazek. Průsečík epipolární roviny s rovinou obrazovou označujeme jako epipolární přímku. Body **e** a **e´** jsou označovány jako epipóly. Jde o body, které získáme protnutím báze s rovinou obrazu. Na níže uvedeném obrázku 4 je zřejmé, že parametry kamer mohou být nastaveny tak, že

epipolární přímky budou rovnoběžné s bází a epipóly budou ležet v nekonečnu. Na obrázku 3 je zřejmé, že všechny epipolární přímky budou procházet epipólem. Na uvedeném obrázku 4 tuto skutečnost tvrdit nemůžeme.



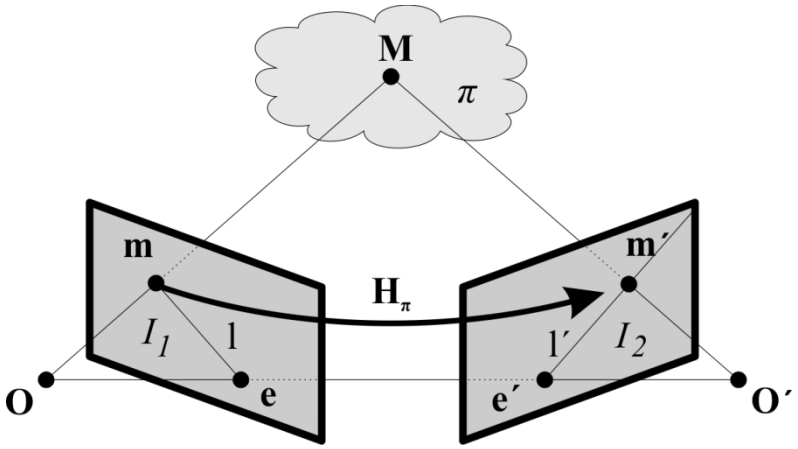
Obr. 4: Epipolární geometrie – epipóly v nekonečnu

2.1 Fundamentální matice

Fundamentální matice je mapování bodu v jednom obraze na korespondující epipolární přímku ve druhém obraze. Jde o vyjádření základních vlastností systému. Je základem pro matematický výpočet epipolární geometrie. Pro zjištění je třeba znát umístění kamer a jejich vnitřní parametry. Na základě fundamentální matice tedy můžeme dát do souvislosti dva obrazy jednoho bodu v jednom obraze na korespondující epipolární přímku v obraze druhém.

Existuje několik možností, jak fundamentální matici určit. Záleží na tom, zda jsou známy projekční matice či nejsou. Metody, které se využívají, jsou většinou nazývány podle počtu korespondujících bodů, které se používají k výpočtu.

Pro přesné určení fundamentální matice je důležité, aby korespondující páry byly stanoveny s velkou přesností, jinak by mohly být výpočty chybné.



Obr. 5: Epipolární geometrie – přenášení bodu

Na obrázku 5 je toto zřejmé. Paprsek respektive přímka vycházející ze středu kamery **O** a procházející bodem **m** se protíná s danou rovinou v bodě **M**. Bod **M** je pak v rámci přesunu v rovině π promítnut do bodu **m′**, který leží na epipolární přímce **l′**. Body **m** a **m´** jsou tedy obrazy 3D bodu **M**. Každý další bod $\mathbf{m_i}$ prvého obrazu a jemu korespondující bod $\mathbf{m_i'}$ budou ve vztahu vzájemné ekvivalence. Existuje tedy 2D homografie H_π mapující každý $\mathbf{m_i}$ na $\mathbf{m_i'}$, což znamená, že mapováním vzájemně si odpovídajících bodů stanovíme projektivitu snímků a to i kdybychom neznali vnitřní parametry kamery.

Máme-li dán bod **x′** a epipolární přímku **l'** procházející tímto bodem a epipólem **e'**, lze stanovit parametry epipolární přímky vztahem

$$l' = e' \times m' = [e']_x m'. \tag{1}$$

Protože bod **m′** může být stanoven ze vztahu

$$m' = H_\pi m, \tag{2}$$

lze předchozí vztah zapsat jako:

$$l' = [e']_x H_\pi m = Fm. \tag{3}$$

Z rovnice (3) je následně definována fundamentální matice, tato definice je vyjádřena vztahem:

$$F = [e']_x H_\pi \tag{4}$$

H_π je mapování jednoho bodu do druhého bodu v libovolné rovině.

Vztah (4) geometricky reprezentuje mapování z dvourozměrného prostoru (obrazový prostor $\mathbf{I_1}$) do jednorozměrného prostoru (epipolární přímka **l′**).

Ze vztahu vyjadřující fakt, že bod **m′** leží na přímce l′ $(l'^T m' = 0)$ a z rovnice 3 vyplývá vztah:

$$m'^T F m = 0 \tag{5}$$

Nalezením epipolárních přímek se velice usnadní hledání korespondujících si bodů v obrazech. Není totiž nutné hledat odpovídající bod v celém obraze, ale jen v blízkém okolí epipolární přímky. Po stanovení fundamentální matice a mapovacích matic je možné určit polohu bodu v prostoru [6].

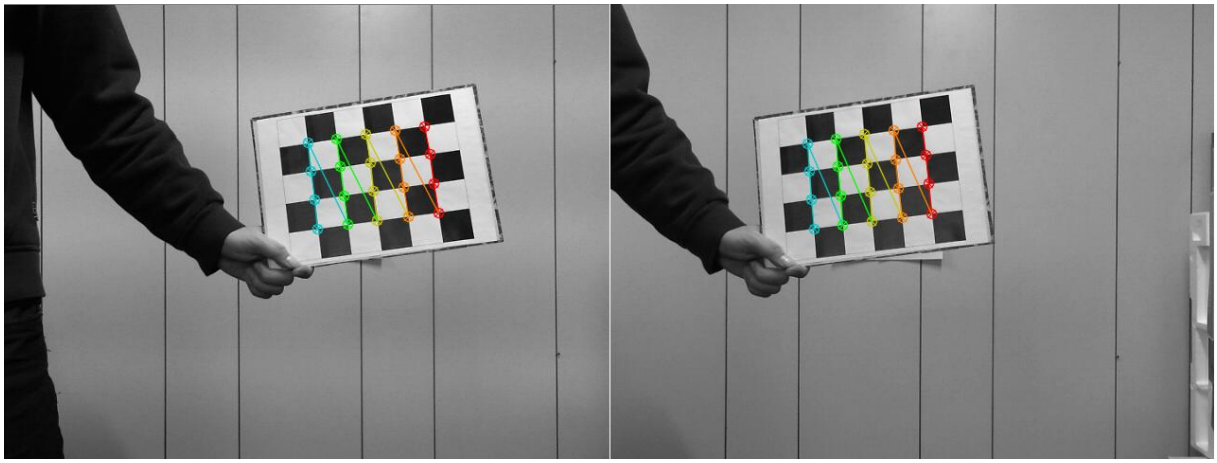
3 KALIBRACE KAMER

Kalibrace je proces, při kterém získáváme vnitřní a vnější parametry kamer potřebné pro další použití. V této práci používám pro snímání scény dvě kamery. Kalibrace se provádí tak, že aparátem nasnímáme scénu se známými obrazovými vzory, následně provedeme analýzu získaných stereografických obrazů.

3.1 Kalibrace pomocí obrazců

V rámci mé diplomové práce je používána při procesu kalibrace knihovna pro zpracování obrazu OpenCV. Jedná se o open source knihovnu, která operuje s množstvím funkcí pro zpracování obrazových dat. Knihovna je vydána pod licencí BSD. Ta umožňuje volné šíření a použití jednotlivých funkcí knihovny. Knihovna je jakožto multiplatformní software napsána v jazyce C a C++.

Základem kalibrace kamer je nasnímání určitého množství stereo párů, tedy dvojic fotek, zobrazujících totožnou scénu snímanou v totožný okamžik dvojicí kamer. Při využití výše zmiňované knihovny pro zpracování obrazu je doporučeno použít obrazce šachovnicového vzoru, které jsou snadno viditelné ve snímané scéně. Pro získání prostorových bodů potřebných ke kalibraci je třeba, aby tyto obrazce byly nastaveny pod různými úhly natočení a zároveň různě vzdáleny od snímajících kamer. Šachovnicový vzor je vhodný zejména proto, že program snadno zaznamená kontrastní body v místech křížení čtverců, viz obr. 6. Souřadnice takto nalezených bodů jsou pak základem pro další výpočty. Pokud je šachovnicový vzor na pevné a rovné ploše, lze předpokládat, že body jsou jednotlivě umístěny ve vzdálenosti Z=0 s osami X a Y spojenými mřížkou šachovnice. Na obrázku je ukázka snímků kalibračního vzoru použitých v této práci [2][5].

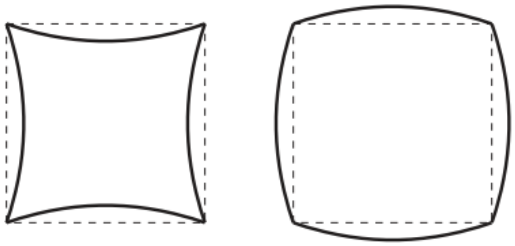


Obr. 6: Ukázka kalibračního vzoru

3.2 Výsledky kalibrace

Vnitřní parametry kamer

Vnitřní parametry kamer určují, jakým způsobem kamery deformují vstupní obraz. Jedním z těchto parametrů je radiální zkreslení. Je to odchylka, která nastává zkreslením čoček během počáteční projekce světla do obrazové roviny. Na obrázku 7 je znázorněno vlevo poduškovité a vpravo soudkovité radiální zkreslení.



Obr. 7: Ukázka radiálních zkreslení

Zkreslení je možné eliminovat pomocí kvalitnějšího objektivu nebo softwarově pomocí kalibrace. V modelové aplikaci v části programu pro kalibraci kamer je radiální zkreslení ve výstupu definováno a odstraňováno pomocí distorzního vektoru ($\boldsymbol{D_1}$, $\boldsymbol{D_2}$). Předpis tohoto zkreslení je $\sum_{i=1}^{6} k_i x^{2i}$.

$$\boldsymbol{D}_i = \left(k_1, k_2, p_1, p_2\left[, k_3[k_4, k_5, k_6]\right]\right), \tag{6}$$

k_1 - k_6 – definují radiální zkreslení,

p_1, p_2 – tangenciální zkreslení.

Dalším výstupem kalibrace je matice kamery ($\boldsymbol{M_1}$, $\boldsymbol{M_2}$), pro každou kameru.

$$\boldsymbol{M} = \begin{bmatrix} f_x^{(j)} & 0 & c_x^{(j)} \\ 0 & f_y^{(j)} & c_y^{(j)} \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

f_x, f_y – ohnisková vzdálenost v ose *x* a ose *y* (v pixelech),

c_x, c_y – pozice středu obrazu.

Vnější parametry (extrinsics)

Vnější charakteristiky kamerového systému určují relativní pozici a natočení jedné kamery vůči druhé. Natočení pravé kamery vůči levé určuje rotační matice (***R*).**

Dalším parametrem je translační vektor (***T***), který určuje posunutí pravé kamery vůči levé kameře (jestli jsou kamery vedle sebe nebo nad sebou).

$$\boldsymbol{T} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{8}$$

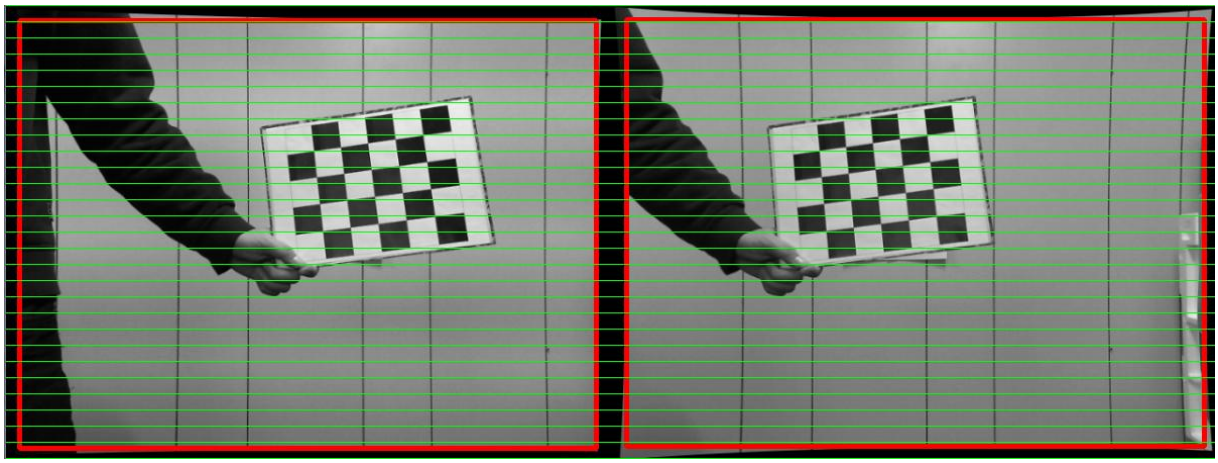
Další vnější parametry jsou výsledkem rektifikace a budou uvedeny níže.

4 REKTIFIKACE STEREOSKOPICKÉHO OBRAZU

Podstatou rektifikace je v širším slova smyslu jakákoliv přeměna polohy obrazových prvků. Pokud jde o stereosnímky, rektifikace umožňuje relativně snadné přiřazení bodu na pravém snímku korespondujícímu bodu na levém snímku stereo páru.

Pro pořízení scény je nutné správné nastavení kamer, podle metody Off-axis. Proto kamery by měly tvořit přesnou paralelní soustavu, tedy být vůči sobě posunuty pouze v horizontálním směru. To je však v reálném prostředí a díky připevnění kamer nedosažitelné. I když jsou kamery od jednoho výrobce a měly by mít stejné parametry, mohou se tyto parametry ve skutečnosti mírně lišit. Vzhledem k výše zmíněným okolnostem se přistupuje k stereoskopické rektifikaci obrazu.

Rektifikace má za cíl, aby korespondující body na snímcích stereo páru byly ve stejné výšce, viz obr. 8 (snímky budou koplanární). Pokud jde o soustavu kamer, které mají odchylky v rovnoběžnosti optických os nebo u kterých došlo k rotaci kamery, rektifikace srovná pořízené stereosnímky podle ypsilonových souřadnic korespondujících bodů. Stejný postup se uplatní i v případě, kdy kamery nejsou ve stejné vzdálenosti od snímaných bodů. Proces rektifikace má tedy za cíl sjednotit zobrazovací roviny kamer [4].



Obr. 8: Ukázka rektifikovaného snímku

Výsledkem rektifikace jsou další extristické parametry kamerové soustavy ($\boldsymbol{R_1}$, $\boldsymbol{R_2}$, $\boldsymbol{P_1}$, $\boldsymbol{P_2}$, $\boldsymbol{Q}$). Prvním výsledkem je rektifikační transformační matice ($\boldsymbol{R_1}$, $\boldsymbol{R_2}$) pro obě kamery. Určuje rotaci obrazu potřebnou pro rektifikaci (rotace okolo všech os) [2].

Další vypočítaný vnější parametr je nová projekční matice obou kamer (***P₁***, ***P₂***). Sjednocené (projekční) matice kamer tak, aby byly rektifikované.

$$\boldsymbol{P1} = \begin{bmatrix} f & 0 & cx_1 & 0 \\ 0 & f & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{9}$$

$$\boldsymbol{P2} = \begin{bmatrix} f & 0 & cx_2 & T_x * f \\ 0 & f & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{10}$$

f – ohnisková vzdálenost (v pixelech),

c_x, c_y – pozice středu obrazu.

$T_x * f$ – posun v ose x pro pravou kameru.

Dalším vnějším parametrem soustavy kamer je hloubková matice (***Q***), která přepočítává perspektivu (neshodnost dvou bodů v hloubce). Matice je potřebná při převádění snímků do prostorového vnímání.

$$\boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 & -c_{x1} \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{1}{|T|} & 0 \end{bmatrix} \tag{11}$$

f – ohnisková vzdálenost (v pixelech),

c_x, c_y – pozice středu obrazu,

$|T|$ – velikost translačního vektoru.

5 STEREO KORESPONDENČNÍ METODY

Stereovize je důležitou součástí pro monitorovací systémy, navigace a robotiku. Takové systémy mohou mít náročné požadavky na zpracování v reálném čase, a proto musí být funkce stereo korespondence navržené tak, aby běžely rychle. A právě proto nejsou vyrovnávací paměti součástí funkcí OpenCV, ale jsou drženy vedle, aby si je mohl tvůrce systému řídit podle potřeby.

5.1 Metoda Block-matching

Stereo korespondence respektive zobrazení odpovídající 3D bodu ve dvou různých pohledech kamer, lze vypočítat pouze ve vizuálních oblastech, ve kterých se pohledy obou kamer překrývají. To je jeden z důvodů, proč je nejvhodnější pro dosažení nejlepších výsledků situovat kamery blízko sebe a rovnoběžně tak, jak je to možné. Poté, co známe parametry soustavy kamer nebo velikosti objektů na scéně, můžeme odvodit hloubkovou (disparitní) mapu.

$$d = x^l - x^r \left(\text{nebo d} = \mathrm{x^l} - \mathrm{x^r} - \left(\mathrm{c_x^{left}} - \mathrm{c_x^{right}}\right)\right), \tag{12}$$

d – disparita,

$\mathrm{x^l}$ – pozice bodu z pohledu levé kamery,

$\mathrm{x^r}$ – pozice bodu z pohledu pravé kamery,

$\mathrm{c_x^{left}}$ – posunutí posunutí středu obrazu levé kamery,

$\mathrm{c_x^{right}}$ – posunutí posunutí středu obrazu pravé kamery.

Tento vzorec platí v případě, že se paprsky protínají v konečné vzdálenosti mezi korespondujícími body ve dvou různých pohledech kamer.

Stereokorespondenční algoritmus pracuje za pomocí malých oken (tzv. SAD – Sum of absolute differnces) respektive jejich součtů absolutních rozdílů. Najde odpovídající body mezi levými a pravými rektifikovanými obrazy. Tento algoritmus nachází pouze odpovídající si body mezi dvěma obrazy. Čím více je scéna strukturovaná, tím více bodů bude mít vypočítanou hloubku. Block-matching (BM) má tři kroky, které pracují s rektifikovanými obrazy stereo páru:

1. **Normalizace obrazového jasu a textury**

Během normalizace jsou ve vstupních obrazech snižovány rozdíly jasu a zvýrazňovány struktury. V této práci uvádím dva způsoby normalizace (Detekce hran, Prostá normalizace) a jsou podrobněji popsány v kapitole 5.3 Předzpracování obrazu.

2. **Hledání korespondencí podél vodorovných epipolárních přímek za použití SAD oken**

Korespondence se vypočítá za použití SAD oken. Parametrem `SADWindowSize` se nastaví délka hrany těchto oken, na které se zpracovávaný obraz rozdělí. Pro každý specifický tvar v levém obrazu hledáme odpovídající řádek v pravém obrazu. Po rektifikaci spolu řádky v obrázcích korespondují, takže odpovídající umístění v pravém obrazu musí být ve stejném řádku jako v levém obrazu. Takto odpovídající umístění lze nalézt v případě dostatečně výrazné struktury obrazu. Pokud má levý bod hledané struktury koordináty x0, y0, potom pro paralelně umístěnou kameru musí být objevena shoda (pokud bude existovat) na stejném řádku na pozici x0, nebo vlevo od ní.

První parametr, který řídí matching, je `minDisparity`. Určuje pozici, odkud se začnou hledat disparity. Výchozí hodnota je 0. Disparity jsou hledány přes tolik pixelů, kolik určuje `numberOfDisparities`. Disparity se nehledají po pixelech, ale na jejich částech (sub-pixelech), jejichž počet určí parametr `subPixelDisparities`.

Snížení hodnoty parametru `numberOfDisparities` může pomoci snížení času zpracování zkrácením vzdálenosti, na které se hledá disparita v rámci epipoláry. Velké disparity umožňují hledání blízkých předmětů.

Parametry `minDisparity` a `numberOfDisparities` určují `horopter`, což je hranice prostoru před kamerami, na které je možné pracovat. V této oblasti `horopteru` hledá stereo algoritmus shody. Všechno mimo tuto oblast je označeno jako „díra“, protože tam nelze spočítat vzdálenost od kamery. Oblast horopteru se zvětšuje snížením vzdálenosti mezi kamerami nebo zmenšením ohniskové vzdálenosti kamer nebo zvětšením `numberOfDisparities` nebo použitím kamer s větším rozlišením.

Korespondující body v rámci horopteru mají vnitřní omezení, nazývané omezení pořadí, které jednoduše říká, že pořadí charakteristických struktur se nemůže měnit z levého do pravého obrázku. Kvůli šumu může některá charakteristická struktura chybět na pravém obrazu, i když na levém je. Pořadí charakteristických struktur však zůstává

pořád stejné. Podobně mohou existovat charakteristické struktury na pravém obrazu, které nelze najít na levém. Jde o tzv. vsuvky. Avšak ani ty nemohly změnit pořadí charakteristických struktur, pouze je mohou rozšířit.

3. Koncová kontrola nalezených korespondencí

Koncová kontrola má za úkol odstranit špatně detekované shody a eliminaci špatných korespondencí. Většina shod při průřezu po epipoláře tvoří centrální vrchol obklopený na obou stranách strmě svažujícími se křivkami. K odstranění špatně detekované shody se použije funkce řízená parametrem `uniqueRatio`. Špatně detekované shody příliš vyčnívají nad ostatními shodami v dané oblasti. Tento parametr určí, které vyčnívají příliš, a proto se musí odstranit.

Parametr `textureThreshold` se používá jako mezní hodnota pro danou oblast, kterou musí disparita bodu dosáhnout, aby nebyl tento bod odstraněn a nezkresloval oblast, jako náhodný šum. Pro výpočet se používá SAD okno (nebo SSD – Sum of squared differences). Pokud je průřez po epipoláře v aktuálním okně pod touto hodnotou, nehledá se v daném okně žádná shoda.

Základní blokový matching má problém blízko hranic objektu, protože vyhledávací okno vidí na jedné straně popředí a na druhé straně pozadí objektu. Díky tomu jsou na hranicích objektu detekovány shody, tzv. `speckle` (zrnitá oblast). Těmto shodám se předchází nastavením parametru `speckleWindowsSize` a `speckleRange`. `SpeckleWindowsSize` určuje velikost okna, ve kterém jsou hledány okraje objektu. Například hodnota 9 znamená, že se jedná o čtverec 9x9 pixelů. Rozdíl minimální a maximální disparity uvnitř tohoto okna musí být menší nebo roven parametru speckleRange, jinak jsou všechny disparity z okna ignorovány.

Disparitní mapa se počítá z levého do pravého obrázku. Pokud je hodnota `disp12MaxDiff` nastavena na větší než -1, tak se provádí ještě výpočet z pravého do levého. Projdou se všechny spočítané disparity, a pokud jsou rozdíly mezi nimi větší než tento parametr, tak jsou ignorovány [2].

5.2 Semi Global Block-matching

SGBM (Semi Global Block-matching) algoritmus rozšiřuje obyčejný BM o více řádkové hledání disparity. Z toho vyplývají jeho hlavní negativa a pozitiva. Dosahuje sice lepších výsledků, ale je výrazně pomalejší a výrazně náročnější na paměť. Je velmi dobře paralelizovatelný.

1. Výpočet pixelové korespondence

Pro všechny body na zkoumané epipoláře se spočítají korespondence pro všechny potencionální nezáporné disparity. Spočítá se váha pro pixelovou korespondenci pro každý bod a pro všechny nezáporné disparity z definovaného rozsahu.

$$C[x, y, d] = |L(x) - R(x - d)|, \tag{13}$$

C – korespondence,

L – hodnota bodu pro levý obrázek,

R – hodnota bodu pro pravý obrázek,

d – disparita.

2. Hledání cest

SGBM spočítá váhu cesty, což je minimum korespondencí bodů, které jsou od počítaného bodu daným směrem a váhy dané cesty před tím než dospěla k danému bodu podle vzorečku níže. Většinou se počítá pro 8 směrů:

$$L_r(p, d) = C(p, d) + \min \begin{cases} L_r(p - r, d), \\ L_r(p - r, d + 1) + P_1 \\ L_r(p - r, d - 1) + P_1 \\ min_i L_r(p - r, i) + P_2 \end{cases}, \tag{14}$$

P_1 , P_2 – konstanty pomáhající odstranění nespojitosti,

L_r – váha cesty (nejmenší rozdíl),

C – korespondence.

3. Disparita

Samotná disparita se pak určí jako cesta s minimální váhou. OpenCV umožňuje spustit SGBM pouze na CPU. Parametr `FullDP` vynucuje výpočet výdajů ve všech osmi cestách, jinak je vypočítáno pouze 5 cest. Stejně jako je tomu u BM, OpenCV umožňuje použít filtr horizontálního Sobelova operátoru [10].

5.3 Předzpracování obrazu

První fází algoritmů BM nebo SGBM je předzpracování obrazu. Během ní jsou ve vstupních obrazech snižovány rozdíly jasu a zvýrazňovány struktury.

5.3.1 Detekce hran

Jednou z možností předzpracování obrazu je detekce hran, a pro ni lze využít Sobelův operátor. Tento operátor je aproximace druhé derivace. Výsledkem je vektor gradiendu nebo norma tohoto vektoru. Aproximace se provádí násobením bodu a jeho okolí maticí pro horizontální nebo vertikální směr. Matice $\boldsymbol{G_x}$ pro horizontální a matice $\boldsymbol{G_y}$ pro vertikální aproximaci:

$$\boldsymbol{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \tag{15}$$

$$\boldsymbol{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \tag{16}$$

Tyto matice jsou základní.

Celková aproximace **G** se spočítá:

$$\boldsymbol{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \begin{bmatrix} A_{i-1,i-1} & A_{i,i-1} & A_{i+1,i-1} \\ A_{i-1,i} & A_{i,i} & A_{i+1,1} \\ A_{i-1,i+1} & A_{i,i+1} & A_{i+1,i+1} \end{bmatrix}, \tag{17}$$

$$\boldsymbol{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \begin{bmatrix} A_{i-1,i-1} & A_{i,i-1} & A_{i+1,i-1} \\ A_{i-1,i} & A_{i,i} & A_{i+1,1} \\ A_{i-1,i+1} & A_{i,i+1} & A_{i+1,i+1} \end{bmatrix}, \tag{18}$$

$$\boldsymbol{G} = \sqrt{G_x^2 + G_y^2}. \tag{19}$$

Sobelovy operátory mohou být větší než 3x3, např.:

$$\begin{bmatrix} +1 & +4 & +7 & +4 & +1 \\ +2 & +10 & +17 & +10 & +2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -10 & -17 & -10 & -2 \\ -1 & -4 & -7 & -4 & -1 \end{bmatrix}, \tag{20}$$

$$\begin{bmatrix} +1 & +4 & +9 & +13 & +9 & +4 & +1 \\ +3 & +11 & +26 & +34 & +26 & +11 & +3 \\ +3 & +13 & +30 & +40 & +30 & +13 & +3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & -13 & -30 & -40 & -30 & -13 & -3 \\ -3 & -11 & -26 & -34 & -26 & -11 & -3 \\ -1 & -4 & -9 & -13 & -9 & -4 & -1 \end{bmatrix}, \tag{21}$$

Ve stereokorespondenčních metodách se používá jen horizontální matice, protože se pohybujeme pouze na ose *x* [13].

Základní vzorec pro předzpracování obrazu:

$$min\left[max\left(X - I_{cap}\right), I_{cap}\right], \tag{22}$$

I_{cap} – parametr `preFitlerCap`.

Základní vzorec (22) je upraven tak, že X je nahrazeno G_x:

$$min\left[max\left(G_x - I_{cap}\right), I_{cap}\right]. \tag{23}$$

5.3.2 Prostá normalizace

Prostá normalizace upravuje hodnotu bodu na základě průměru hodnot jeho okolí. Základní vzorec (22) je upraven tak, že X je nahrazeno $I_c - \bar{I}$:

$$min\left[max\left(I_c - \bar{I} - I_{cap}\right), I_{cap}\right], \tag{24}$$

I_{cap} – parametr `preFitlerCap`,

$\bar{I}$ – průměr spočítaný z bodů v rámci této oblasti,

I_c – hodnota, na kterou se přepočítá hodnota aktuálního bodu [2].

5.4 Odhad disparity pomocí sub-pixelové přesnosti

Disparity jsou spočítány jako rozdíly korespondencí obrázků a jsou zaokrouhleny na celá čísla. U vzdálenějších objektů nebo kamer s menším rozlišením dochází k tomu, že disparita daného bodu nemůže být vždy celočíselná.

A naopak u objektů, které jsou skoro rovnoběžné s kolmicí na osu kamerového soustavy, dochází k viditelným skokům, kde je nespojitost disparit dobře vidět.

K odstranění těchto neduhů se používá právě sub-pixelová disparita. Je to přepočet aktuálních hodnot disparit na základě už spočítaných disparit a jejich proložení polynomem obvykle druhého stupně. Pro dané okno (velikost 5 až 11 bodů) se po jeho proložení najde minimum, viz obr. 9. To určuje novou hodnotu disparity ve středovém bodě okna:

$$ad_x^2 + bd_x + c = v_x. \tag{25}$$

Derivace (hledáme minimum):

$$2ad_x + b = 0, \tag{26}$$

$$d_x = \frac{-b}{2a}. \tag{27}$$

Vytvoření soustavy rovnic pro tři okolní body:

$$\begin{cases} ad_{i-1}^2 + bd_{i-1} + c = m_{i-1} \\ ad_i^2 + bd_i + c = m_i \\ ad_{i+1}^2 + bd_{i+1} + c = m_{i+1} \end{cases}. \tag{28}$$

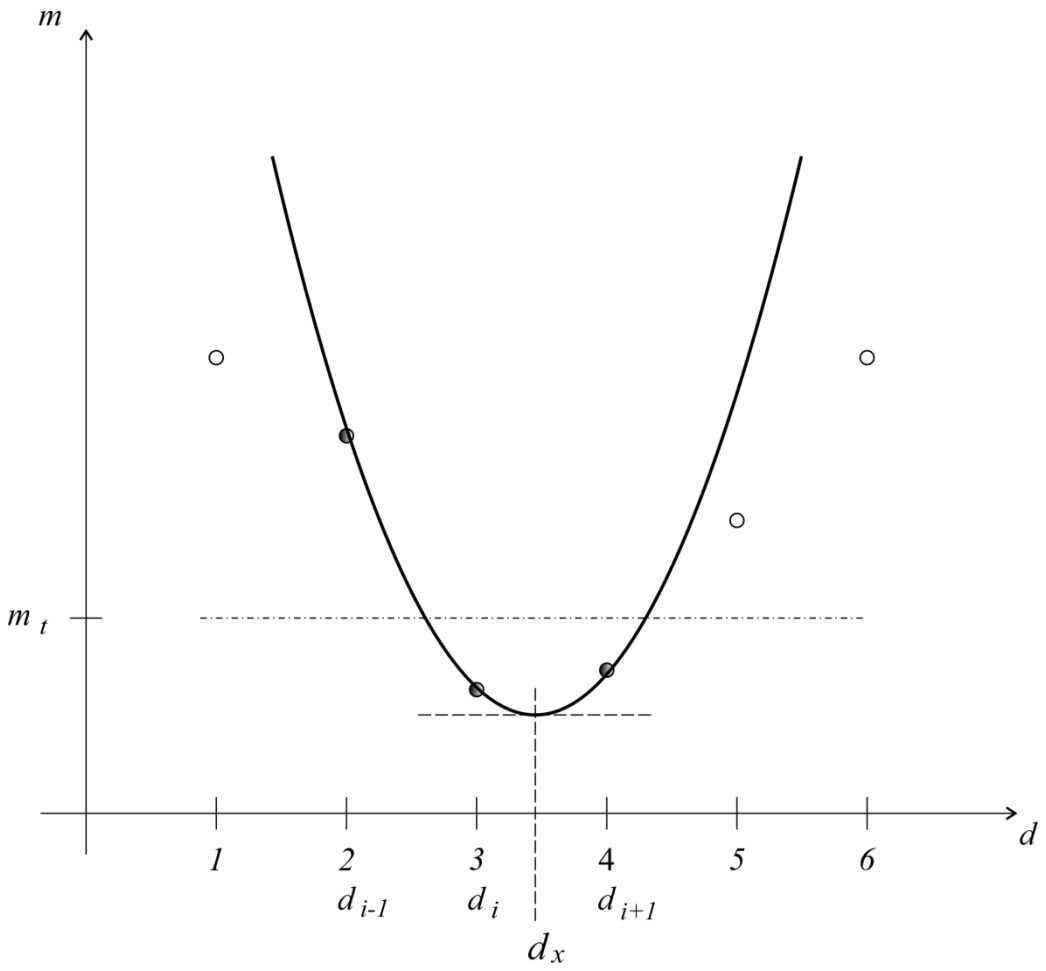
Aplikujeme Kramerovo pravidlo a zároveň posuneme střed systému koordinátů do bodu $[d_i, 0]$, z toho vyplývá:

$$\frac{b'}{a'} = -\frac{\begin{vmatrix} 1 & m_{i-1} & 1 \\ 0 & m_i & 1 \\ 1 & m_{i+1} & 1 \end{vmatrix}}{\begin{vmatrix} -1 & m_{i-1} & 1 \\ 0 & m_i & 1 \\ 1 & m_{i+1} & 1 \end{vmatrix}} = -\frac{m_{i-1} - m_{i+1}}{m_{i-1} - 2m_i + m_{i+1}} \tag{29}$$

Dosadíme zpět do rovnice výpočtu d_x a přičteme posunutí z minulého kroku:

$$d_x = d_i + \frac{m_i - m_{i+1}}{2(m_{i-1} - 2m_i + m_{i+1})} \tag{30}$$

Tím získáme nové minimum, viz obr. 9 [11].



Obr. 9: Odhad disparitního minima (polynomiální funkcí)

6 ZPRACOVÁNÍ DAT NA GPU

Tato část práce je věnována přiblížení problematiky paralelního programování na GPU a obecně jeho funkcím a využití.

GPU respektive grafický procesor (graphic processing unit) je procesor, který zajišťuje promítání dat uložených v operační paměti na zobrazovacím zařízení. Výkonné grafické procesory se v současné době využívají i k jiným výpočtům, než jsou výpočty nutné pro zobrazování dat. Grafický procesor je používán v mnoha zařízeních. Jeho konstrukce je závislá na konkrétních potřebách a požadovaném výkonu.

V osobním počítači může být GPU jako rozšiřující grafická karta nebo jako grafická karta integrována na základní desce počítače v podobě integrovaného grafického čipu.

Použitím paralelního programování a následujícím zpracováním těchto programů na GPU lze několikanásobně zkrátit dobu náročných a dlouhotrvajících výpočtů [8].

6.1 CPU vs. GPU

CPU a GPU byly postupem času vyvíjeny každý pro jiný účel. CPU je využíváno jako výpočetní jednotka a pomocí přerušení se stará o jiná zařízení. Má za cíl zpracovat pouze jednu operaci a to co nejrychleji.

GPU jakožto specializovaný procesor je přizpůsoben ke zrychlování vykreslování grafiky, takže musí být schopen zpracovat operaci s velkým množstvím informací za co nejkratší dobu. Je optimalizován pro výpočty čísel s plovoucí desetinnou čárkou.

GPU poskytuje velké zvýšení výkonu v určitých operacích, obsahuje velké množství SIMD procesorů (pro jednu instrukci je několik datových toků) a podpora pro paralelismus je zabudována v hardwaru. CPU sice poskytuje uspokojivý výkon ve všech operacích, ale nemá zase takovou podporu pro paralelismus.

Grafické karty se dlouhou dobu používaly pouze pro zobrazování grafického obsahu. Postupným vývojem se dopracovaly k tomu, že je dnes možné je používat i k jiným operacím. Tato nová schopnost grafických karet se označuje jako GPGPU neboli general-purpuse computing on graphics processing unit (obecné výpočty na grafických kartách). Snahou je zpravidla nahradit CPU u výpočtů, kde je GPU lepší volbou [7].

Grafická karta z hlediska výpočetního stroje se skládá ze streaming multiprocesorů (compute unit) a streaming procesorů (scalar procesor) [15].

6.2 Paralelismus

Výpočetní operace lze řešit sériově nebo paralelně. Sériové řešení výpočtu je složeno ze samostatných kroků. Každý z nich se provádí po dokončení předcházejícího. Většina programů používaných na osobních počítačích funguje na principu sériového zpracování. Programy se skládají ze sady instrukcí, kdy je jedna po druhé zpracovávána na CPU. Druhým způsobem zpracování programu je paralelní řešení výpočtu. V tomto případě je řešený program rozdělen na části a některé instrukce jsou prováděny ve stejnou dobu tedy paralelně. Jinak řečeno, pro paralelní výpočet je ve stejnou dobu použito vícero výpočetních zdrojů.

Aplikováním paralelních výpočtů je možné zlepšit výkon mnoha programů. Například při násobení matic se může urychlit doba výpočtu. Jednotlivé operace násobení matic jsou totiž na sebe nezávislé, a proto mohou být provedeny současně. Avšak výpočty, ve kterých jsou jednotlivé kroky na sebe vázány, nebudou pro paralelní výpočet vhodné.

Je více možností k vytvoření paralelního výpočetního prostředí. Je možné využít více jádrový procesor, výpočetní jednotky v rámci jednoho počítače nebo sadu počítačů propojenou počítačovou sítí. Různé techniky mohou být kombinovány. V této práci operuji s paralelizmem v rámci jednoho počítače, s využitím CPU, GPU a standardu OpenCL.

6.3 OpenCL

OpenCL (Open Computing Language) je otevřený standard, díky kterému je možné paralelní programování různorodých počítačových systémů. Podporuje velkou škálu aplikací. Hraje velkou roli v grafických aplikacích, které kombinují obecné paralelní výpočty s vykreslováním grafiky. Součástí OpenCL je API (Application Programming Interface) pro lepší fungování paralelních výpočtů v různorodých procesech.

Původní návrh standardu OpenCL pochází od společnosti Apple. Od roku 2008 se o vývoj tohoto standardu stará organizace Khronos s aktuální verzí OpenCL 2.0.

Tento standart umožňuje paralelní programování na CPU, GPU a jiných výpočetních jednotkách, které jsou uspořádány do jediné platformy. OpenCL je plnohodnotný framework, který slučuje API, programovací jazyk, knihovny a runtime systém pro podporu vytváření programů.

Architektura OpenCL

1. Platformní model

Jedním z hlavních modelů Standard OpenCL je platformní model. Základem tohoto modelu je hostitelský systém (host). Ten tvoří jedno nebo více OpenCL zařízení (compute device). OpenCL zařízení dále obsahují jednu nebo více výpočetních jednotek (compute unit). Jednotky jsou rozděleny do dílčích prvků zpracování dat (processing elements). Těmito prvky mohou být SSE jednotky procesoru nebo jádra multiprocesorů grafické karty. Proces OpenCL probíhá na hostitelském systému. Poté jsou hostitelem odeslány příkazy k provádění výpočtů na prvcích výpočetní jednotky.

OpenCL zahrnuje podporu pro smíšené verze platforem. V rámci jedné platformy mohou být zařízení s různými verzemi specifikace OpenCL. Tato zařízení jsou specifikována dílčími identifikátory. Těmito identifikátory jsou verze platformy, verze přístroje a verze jazyka OpenCL C, kterou zařízení podporuje. Verze platformy prezentuje dostupné verze runtime OpenCL C. Verze zařízení představuje údaje o schopnostech zařízení. Tyto údaje poskytuje OpenCL zařízení po inicializaci formou informací `clGetDeviceInfo`. Verze jazyka OpenCL C určuje, jakou verzi jazyka může developer využívat [14].

2. Vykonávací model

Dalším z hlavních modelů Standard OpenCL je vykonávací model. Základními komponenty tohoto modelu jsou hostitelský program a kernely (jádra). Hostitelský program pracuje na hostitelském systému a koordinuje kernely. Součástí kernelů jsou dílčí instance, což jsou běžící části programu na výpočetním prvku. Tyto instance jsou nazývány jako pracovní položky. Každá z nich má přiděleno jedno konkrétní globální ID. Pracovní položky jsou uspořádány do jednotlivých pracovních skupin s cílem vytvořit lepší přehled o využívání celého indexovaného prostoru (NDRange). Stejně jako jednotlivé instance i pracovní skupiny mají přiděleno unikátní ID. Každou pracovní položku je tedy možné určit na základě ID pracovní skupiny a přiděleného ID položky v rámci pracovní skupiny nebo pomocí globálního ID.

Pomocí funkce `clCreateContext` hostitelský program vytvoří tzv. kontext zařízení. Kontext je prostředí umožňující přístup k zařízení OpenCL. Na základě dostupných informací kontext určuje, jak budou zpracovány fronty příkazů, respektive jak a na kterých zařízeních dojde ke spuštění kernelů. Kromě jiného má kontext též monitorovat paměť a zajišťovat správu paměti.

3. Paměťový model

V rámci OpenCL mohou kernely využívat 4 druhy paměti. Mezi tyto patří:

- Globální paměť (Global Memory) – tato paměť je nejhůře dostupnou co do rychlosti přístupu. Všem instancím ve všech pracovním skupinám umožňuje jak čtení, tak i zápis.
- Konstantní paměť (Constant Memory) – jde o neměnnou, tedy konstantní část globální paměti.
- Lokální paměť (Local Memory) – tato oblast paměti je přístupná pouze pracovním skupinám a je sdílená jejich pracovním položkám. Mohou zde být alokovány proměnné. Přístup do lokální paměť je rychlejší než u konstantní paměti.
- Privátní paměť (Private Memory) – tato paměť je nejlépe dostupnou co do rychlosti přístupu. Je dostupná pro každou instanci.

4. Programovací model

OpenCL operuje primárně s datově paralelním programovacím modelem nebo s úlohově paralelním programovacím modelem. Zásadním rozdílem je, že datově paralelní model umožňuje spouštět a zpracovávat souběh různých instancí téhož kernelu, zatímco úlohově paralelní model umožňuje sice také spouštět vícero instancí, avšak nemůže jít o instance téhož kernelu.

Programování OpenCL je složeno ze dvou částí. V první části host soubor zjišťuje zařízení a jejich nastavení. Dále určí, na kterém zařízení by se měl program spustit. Ve druhé části se programuje kernel. Jde o samostatnou funkci programu, která se spustí na zařízení. Programování má jistá omezení z hlediska programovacího jazyka. Probíhá v C99 dialektu programovacího jazyka C. Možná je spolupráce s OpenGL a dalšími API. Host soubor nemusí být naprogramován v C, ale je možné použít jazyky Java, Python, C++ a další.

Výhodou OpenCL je díky abstrakci nezávislost na zařízení, kdy je možné využívat OpenCL na grafických kartách AMD a Nvidia, se správnou implementací i na různých procesorech [8].

6.4 Výpočet sub-pixelové přesnosti pomocí CPU a GPU

Použitá funkce knihovny OpenCV vypočítá disparitní mapu se sub-pixelovou přesností pouze pomocí CPU. Pro GPU jsem si musel tento výpočet doplnit.

Funkce knihovny používají pro výpočet sub-pixelové přesnosti SAD [11]:

$$D_{SAD} = \sum_{(i,j)\in U} \left| I_1(x+i, y+j) - I_2\left(x + d_x + i, y + d_y + j\right) \right|. \quad (31)$$

Pro výpočet na GPU jsem zvolil SSD [11]:

$$D_{SSD} = \sum_{(i,j)\in U} \left(I_1(x+i, y+j) - I_2\left(x + d_x + i, y + d_y + j\right) \right)^2. \quad (32)$$

Nevýhodou tohoto postupu je nutnost ručního volání knihovny OpenCL, které zbytečně prodlužuje čas výpočtu, místo toho aby si tuto funkci volala sama knihovna OpenCV. Návrh algoritmu v podobě zdrojového kódu kernelu pro GPU je obsažen jako příloha 3.

7 PRAKTICKÁ ČÁST

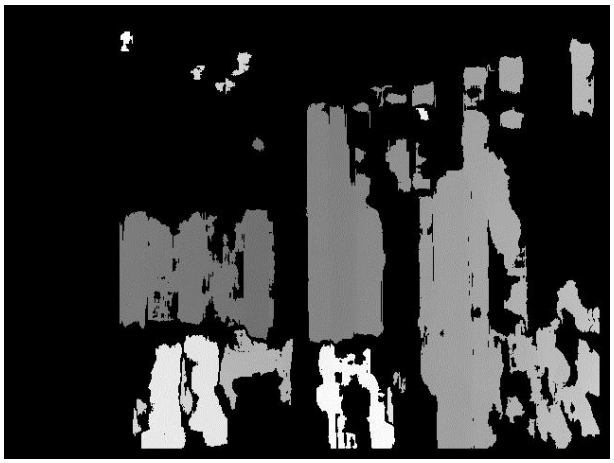
Cílem práce je navrhnout aplikaci, která na základě výpočtů zobrazí disparitní (hloubkovou) mapu snímané scény. Jakožto stereoskopický senzor scénu snímají dvě webové kamery Microsoft LifeCam Studio, model 1425. Aplikace je navržena tak, že v prvním kroku se kalibruje kamerová soustava a v druhém kroku je možné pořizovat disparitní mapu jako jednotlivé snímky nebo video.

7.1 Obarvení disparitní mapy

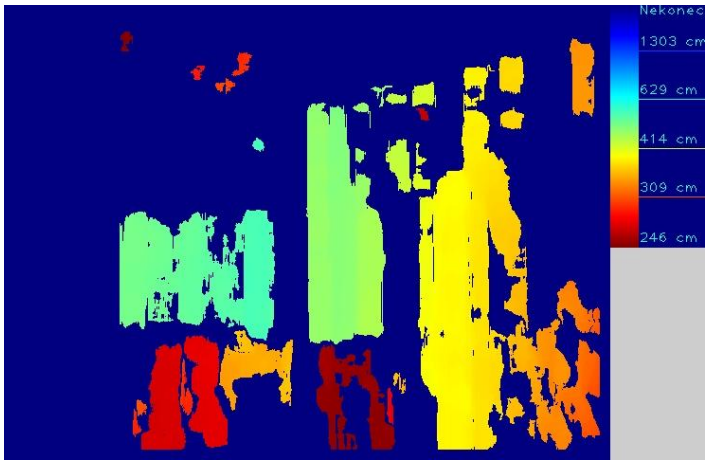
Výstupem BM jsou jednobarevné disparitní mapy, ve kterých je znázorněna disparita různě šedými tóny (obr. 10). Pro lidské oko je lepší použít barevné spektrum, protože více vyniknou jednotlivé hloubky obrazu dané barevným spektrem.

Při zpracování zabírají disparity jen malou část spektra obrazu. Je třeba toto spektrum rozšířit natolik, aby byl obraz pro člověka přehlednější. První fází je převedení na jednobarevnou disparitní mapu, která má barevnou hloubku 256 odstínů (obr. 11). Nízké hodnoty disparit bodů jsou obarveny tmavšími tóny. Naopak vysoké hodnoty disparit bodů jsou obarveny světlými tóny.

Obr. 10: Disparitní mapa – neupravená



Obr. 11: Disparitní mapa – upravená, 256 tónů šedi v celém spektru



Obr. 12: Disparitní mapa – upravená, 256 tónu barevné mapy Jet

Druhou fází je převedení šedotónové disparitní mapy na barevnou (obr. 12), pomocí barevné mapy Jet (obr. 13). Po této úpravě jsou nízké hodnoty disparit bodů obarveny modře. Naopak vysoké hodnoty disparit bodů jsou obarveny červeně.

Obr. 13: Barevná mapa Jet

Tato barevná mapa obarví disparitní mapu tak, že lidské oko je schopné snáze pozorovat menší rozdíly v disparitách, než tomu bylo u předchozích způsobů zobrazení [12].

7.2 Určení vzdálenosti předmětu

Výstupem BM algoritmů je disparitní mapa, která obsahuje vzdálenosti bodu od kamerové soustavy. Vzdálenosti bodu od kamer jsou závislé na rozmístění kamer a na jejich natočení. Určení vzdálenosti předmětu před kamerovou soustavou je znázorněno na obrázku 14.

Výpočet vzdálenosti od kamerové soustavy:

$$z = f\frac{t}{d}, \tag{33}$$

f – ohnisková vzdálenost,

t – vzdálenost mezi kamerami,

d – disparita,

z – výsledná vzdálenost.

Parametry f a t určuje kalibrace kamer. Čím přesněji se soustava kamer zkalibruje, tím přesněji se určí tyto dva parametry a tím přesněji se vypočítají vzdálenosti z disparitní mapy. Tyto parametry jsou součástí hloubkové matice proto, aby tato matice dokázala převádět vzdálenosti z R^2 prostoru kamer do R^3 prostoru, jehož střed je v levé kameře. Natočení levé kamery určuje osu z. Do rovnice níže se dosadí matice $\boldsymbol{Q}$ a vektor $\boldsymbol{V}$.

$$\boldsymbol{Q} * \boldsymbol{V} = [x \quad y \quad z \quad w], \tag{35}$$

$$\boldsymbol{V} = \begin{bmatrix} M_x \\ M_y \\ d \\ 1 \end{bmatrix}, \tag{34}$$

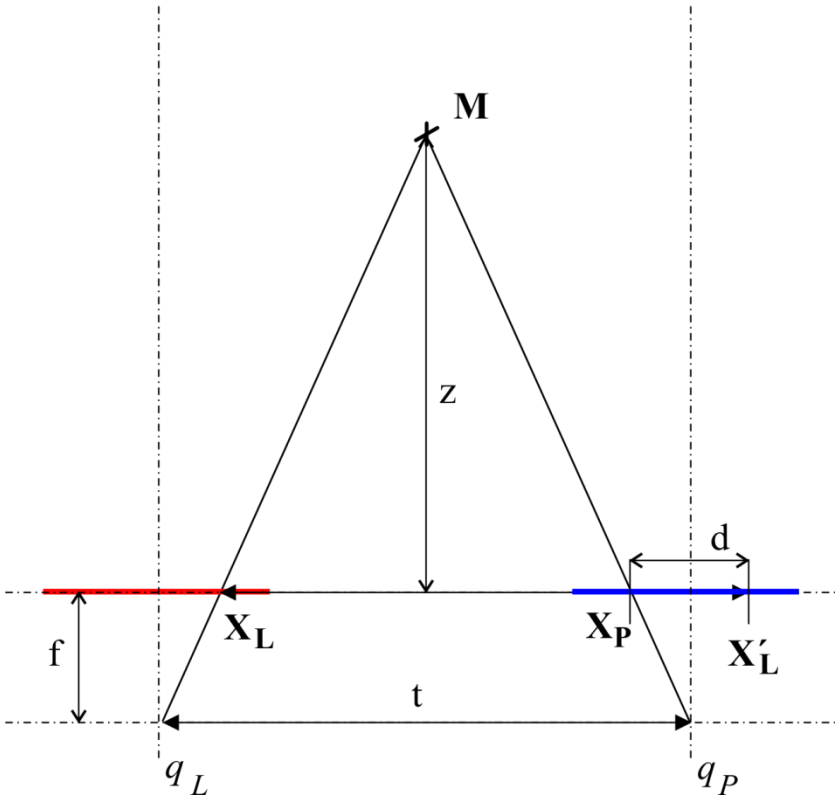
$\boldsymbol{Q}$ – hloubková matice,

$\boldsymbol{M_x}$, $\boldsymbol{M_y}$ – pozice bodu v obrázku.

Výsledkem je vektor $\boldsymbol{p}$.

$$\boldsymbol{p} = \left[\frac{x}{w} \quad \frac{y}{w} \quad \frac{z}{w}\right], \tag{36}$$

$\boldsymbol{p}$ – fyzická pozice, vektor vůči levé kameře [11].



Obr. 14: Určení vzdálenosti bodu

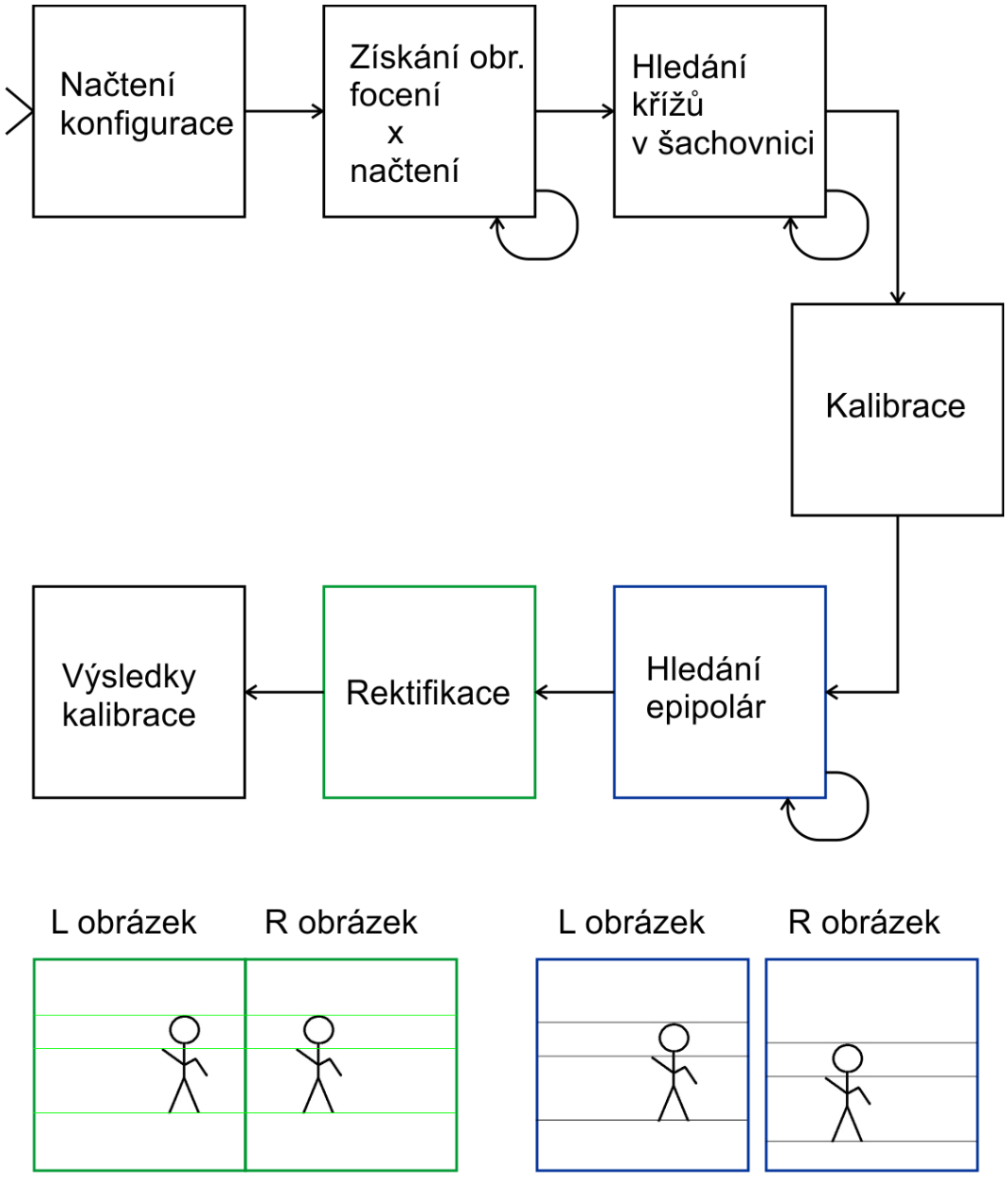
7.3 Popis programu

Program je rozdělen na dvě části. V první části se provádí snímání scény a následná kalibrace kamerové soustavy z nasnímaných fotek. V druhé části pak následuje výpočet disparitní mapy z použitých stereo snímků. Ovládání programu je popsáno v příloze 2.

Kalibrace kamerového systému

Níže uvedený diagram kalibrace (obr. 15) názorně zobrazuje postup programu při kalibraci kamerové soustavy. V programu kalibrace je možné nastavit, jestli dá program kamerám pokyn pouze snímat a následně kalibrovat nebo jenom snímat nebo pouze spustí kalibraci pro pořízené snímky jiným steroskopickým senzorem. Dále je možné nastavit počet pořizovaných dvojic snímků, z nichž má být následně provedena kalibrace soustavy. Pro kalibraci používá program automatizovanou metodu, kdy ve scéně program hledá šachovnici. Proto je dalším parametrem k nastavení počet křížů šachovnice na ose *x* a na ose *y*. Pokud nastavíme možnost snímat a kalibrovat, program po spuštění aktivuje kamery a zobrazí v samostatných oknech (z levé a pravé kamery) snímanou scénu. V dalších dvou oknech jsou totožné obrazy scény, ale zvětšené dle nastavení proměnné zoom. Po zmáčknutí klávesy „u“ na klávesnici, program začne ukládat požadovaný počet dvojic snímků v nastaveném intervalu a klávesou „x“ je možné předčasně ukončit snímání.

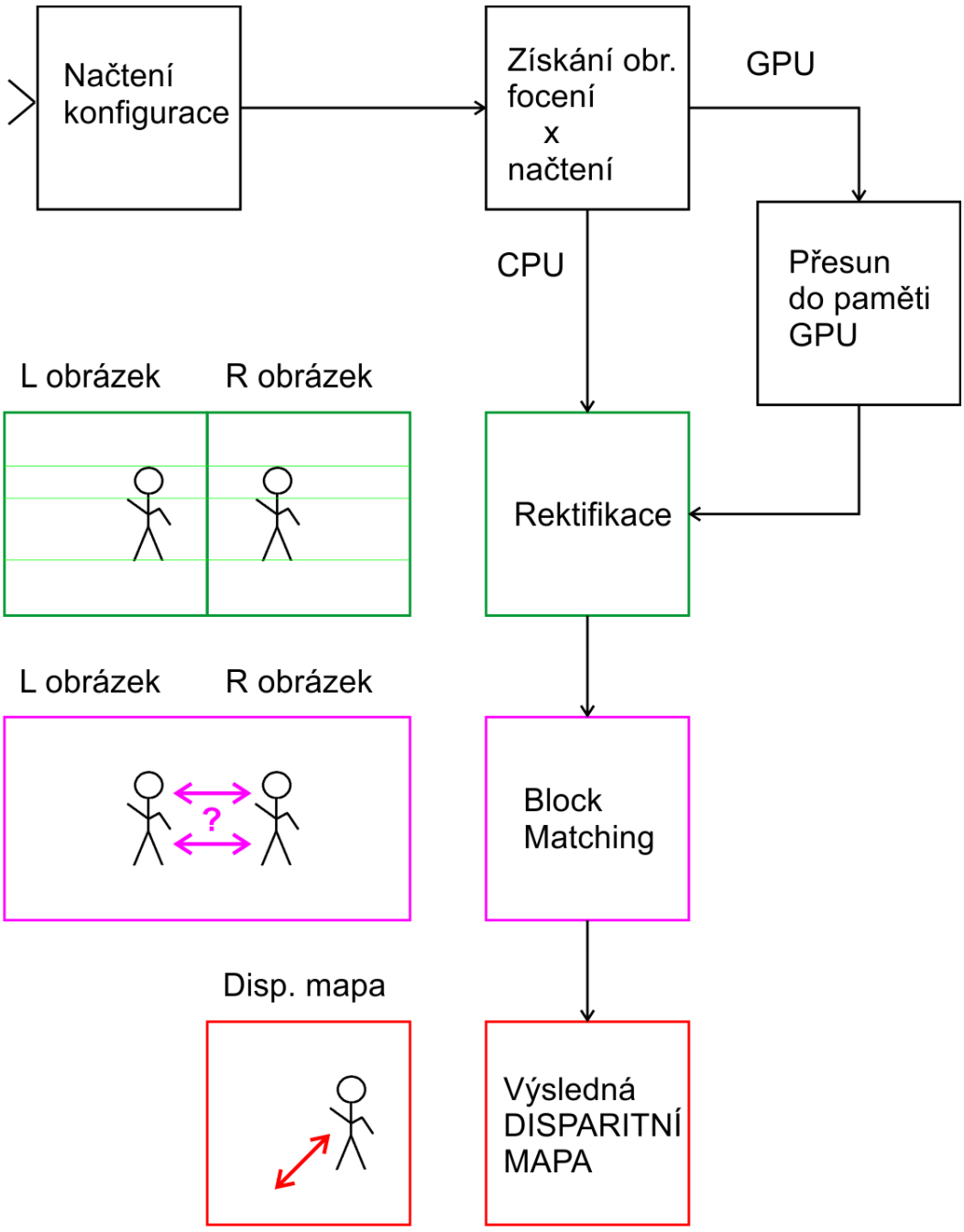
V dalším kroku programu se automaticky spustí kalibrace (pokud je nastaveno). Zobrazí se okno, ve kterém se zobrazuje zpracovaná fotografie s nalezenými body v místech křížení na šachovnici (pokud ji program na fotografii nalezne). Až se projdou všechny fotografie, zobrazí se další okno, kde se bude zobrazovat výstup kalibrace. Ten se skládá z dvojice snímků a to snímku s radiálním zkreslením a snímku s odstraněným radiálním zkreslením. Výsledkem kalibrace jsou soubory extrinsics, intrinsics s vnějšími a vnitřními parametry kamer a parametry radiálního zkreslení. Navíc se vytvoří soubor info, kde jsou uložené informace o průběhu a chybovosti zpracování.



Obr. 15: Diagram pro získání kalibračních dat

Matching

Další částí programu je matching, viz obr. 16. Vytváří disparitní mapy. Celý program je konfigurovatelný přes jediný soubor. Při spuštění programu z příkazové řádky, lze uvést jméno konfiguračního souboru, jako parametr. Pokud nebude jeho jméno uvedeno, je použito jako výchozí jméno "config.txt". Pokud nebude konfigurační soubor nalezen, použijí se výchozí hodnoty parametrů a ty se uloží do konfiguračního souboru.

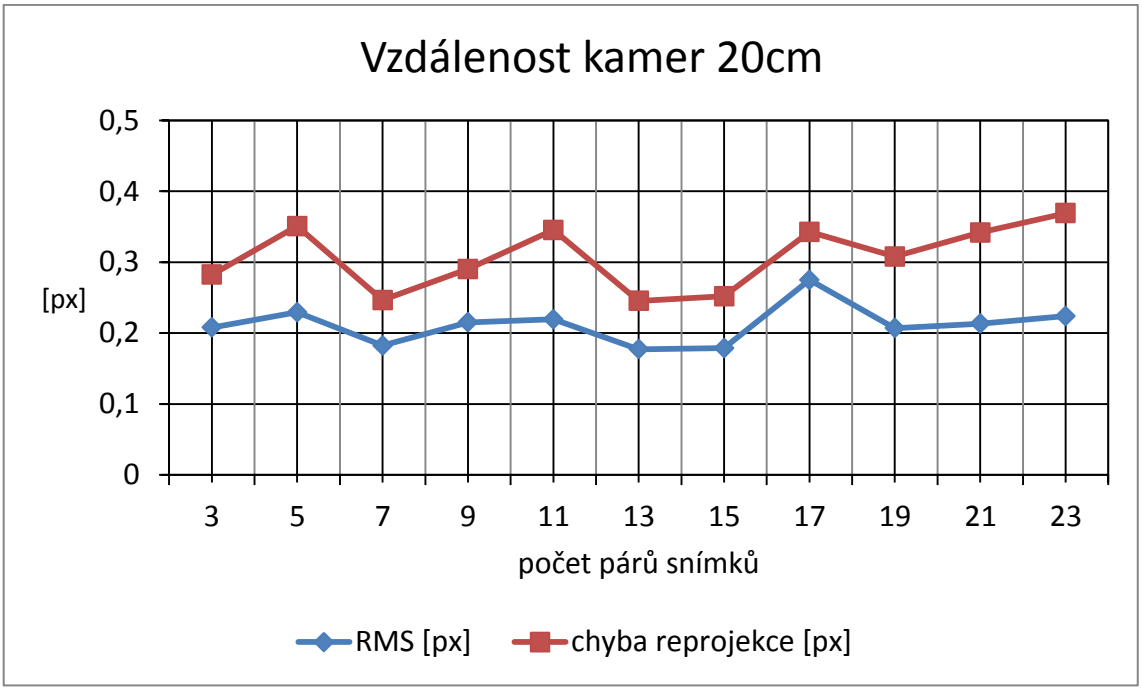


Obr. 16: Diagram hledání stereokorespondencí

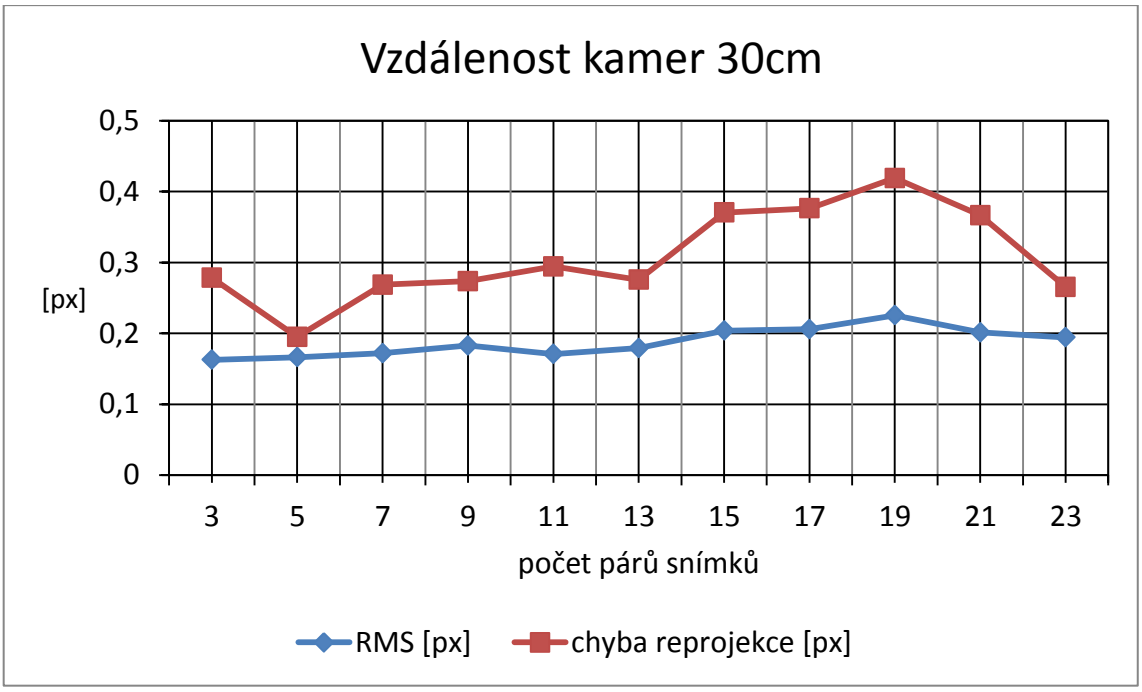
7.4 Statistické hodnoty z kalibrací

Jedním z cílů této práce je změřit hodnoty chyb kalibrace kamerové soustavy. Dále z těchto dat odvodit vhodný počet stereo snímků, kterými se bude stereoskopický senzor kalibrovat s nejmenší chybou reprojekce.

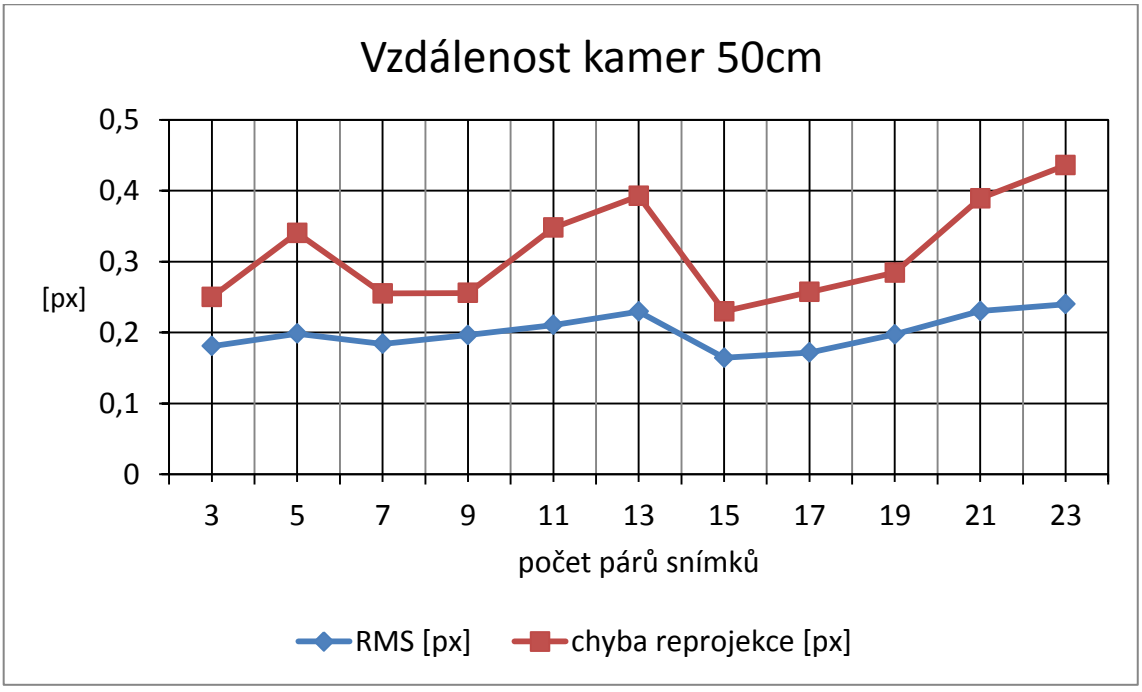
Názorný postup, jakým způsobem program pracuje při kalibraci a následné zpracování a uložení kalibračních dat, je znázorněn na obrázku 15.



Graf 1: Chyby kalibrace (20cm)



Graf 2: Chyby kalibrace (30cm)



Graf 3: Chyby kalibrace (50cm)

Chyba reprojekce odchylka ve výpočtu epipolárních přímek. RMS je kvadratický průměr chyby reprojekce.

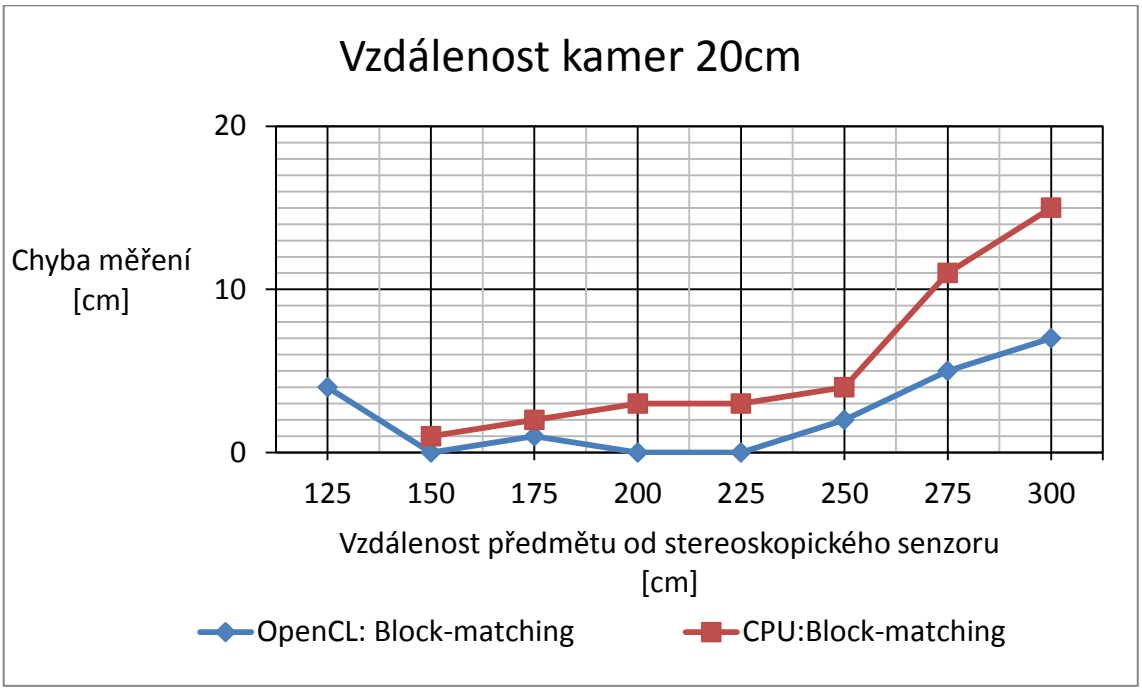
Vhodným počtem snímků pro kalibraci se ukázalo být 15 až 17 párů snímků a to jak z hlediska časové náročnosti kalibrace, tak i z hlediska chyby reprojekce.

Při měření jsem zjistil, že kalibraci může ovlivnit případná prodleva mezi snímáním obou kamer. Tuto chybu jsem maximálně zredukoval tím, že nejprve obě kamery postupně snímají snímky a teprve poté se snímky obou kamer uloží do paměti počítače. Tím se zmenší prodleva mezi jednotlivým pořízením snímků a dojde ke zvýšení přesnosti při kalibraci a hlavně při snímání dymamické scény.

7.5 Hodnocení chyb měření vzdálenosti

Kamery byly upevněny na stojánek do pevné stabilní polohy a zkalibrovány pomocí kalibrační aplikace. Vzdálenost mezi kamerami, ve které proběhla kalibrace a následné měření, je uvedena vždy v hlavičce grafu.

Měření probíhalo tak, že předmět byl umístěn před kamery do dané vzdálenosti a tato vzdálenost předmětu byla změřena aplikací pro měření vzdálenosti v reálném čase. Měřily se vždy středy větších plochých objektů, protože měření vzdálenosti okrajů objektů je nepřesné. Následně byly výsledky měření porovnány se skutečnými vzdálenostmi a rozdíl měření, tedy chyba měření, byla zanesena do grafu.

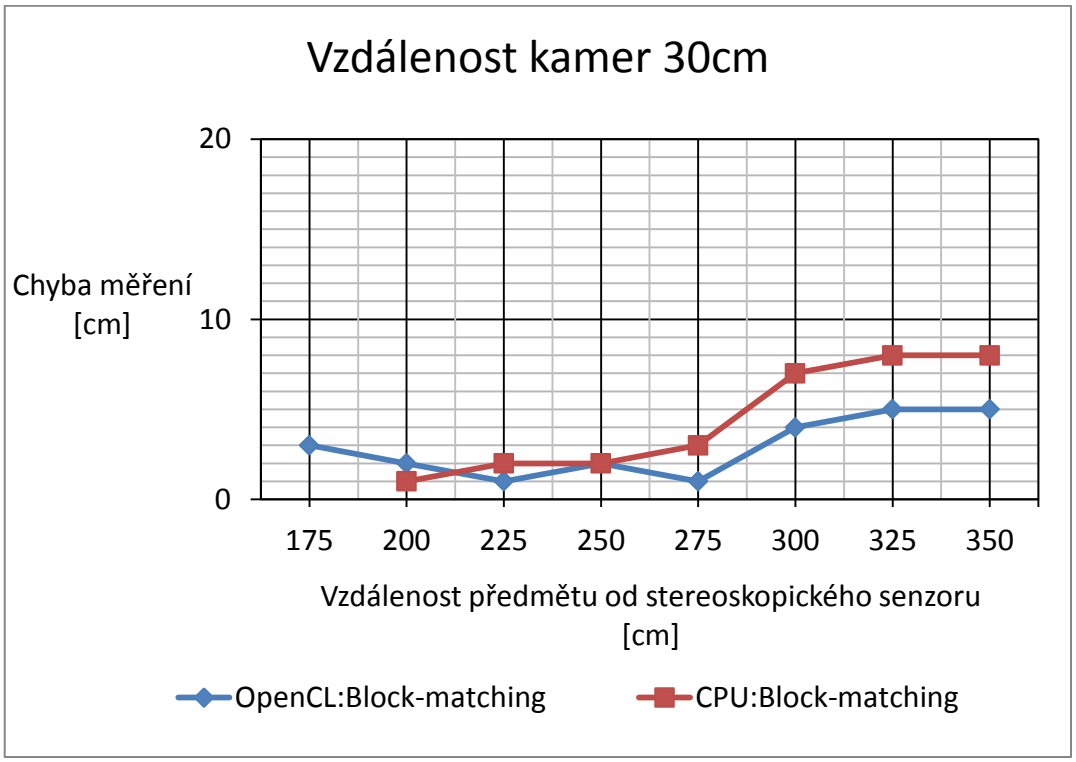


Graf 4: Závislosti chyby měření na vzdálenosti předmětu (20 cm)

Pro toto rozložení kamerové soustavy je možné pozorovat předmět na disparitní mapě v případě zpracování algoritmem OpenCL:Block-matching (výpočet na grafické kartě) na vzdálenosti 125 cm. Na grafu 4 můžeme sledovat průběh chyby měření, která se při vzdálenosti předmětu od stereoskopického senzoru v rozmezí 150 - 225 cm pohybuje od 0 do 1 cm. Lze tedy konstatovat, že měření je relativně přesné. Při měření větší vzdálenosti se chyba lineárně zvyšuje.

V případě algoritmu CPU:Block-matching (výpočet na procesoru) mohou být předměty sledovány od vzdálenosti 150 cm od stereoskopického senzoru. Chyba měření je zde větší než u předchozí metody, jak je patrno z grafu.

Při vzdálenosti kamer 20 cm končí rozlišovací schopnost soustavy kamer obou algoritmů pro vzdálenosti předmětu větší než 300 cm.



Graf 5: Závislosti chyby měření na vzdálenosti předmětu (30 cm)

Pro nastavení kamerové soustavy, uvedené na grafu 5, je možné pozorovat předmět na disparitní mapě v případě zpracování algoritmem OpenCL:Block-matching (výpočet na grafické kartě) od vzdálenosti 175 cm. Z grafu je patrné, že nejnižší chyba měření je naměřena ve vzdálenosti předmětu od stereoskopického senzoru v rozmezí od 200 cm do 275 cm. Chyba měření se při této vzdálenosti pohybuje od 1 do 2 cm. S větší vzdáleností se chyba zvětšuje.

V případě algoritmu CPU:Block-matching mohou být předměty sledovány od vzdálenosti 200 cm od stereoskopického senzoru. Chyba měření je v tomto případě srovnatelná s předchozí metodou, avšak při měření větších vzdáleností od 275 cm se chyba zvětšuje více než u předešlé metody.

Při vzdálenosti kamer 30 cm končí rozlišovací schopnost soustavy kamer obou algoritmů pro vzdálenosti předmětu větší než 500 cm. Při měření na vzdálenosti 500 cm se však projevuje chyba měření asi 20 cm u metody OCL:Block-matching a asi 36 cm u

metody CPU:Block-matching. Tyto hodnoty v grafu uvedeny nejsou, neboť pro reálné měření jsou irelevantní a mají pouze doplňující charakter.

Z měření vyplývá, že pro vzdálenosti předmětů od stereoskopického senzoru do 2,5m je vhodnější menší vzdálenost mezi kamerami.

7.6 Měření doby trvání výpočtu disparitní mapy

Výpočet disparitní mapy provádím na třech počítačových sestavách. Jedná se o notebook a dvě počítačové sestavy.

Testovací sestava č. 1: PC1

- Procesor: AMD Athlon X2 250 (3GHz)
- Grafická karta: AMD Radeon HD 7770
- Operační systém: MS Windows 7 64-bit

Testovací sestava č. 2: Notebook

- Procesor: Intel Core i5 2410M (2,3GHz)
- Grafická karta: NVIDIA GeForce 540M
- Operační systém: MS Windows 7 64-bit

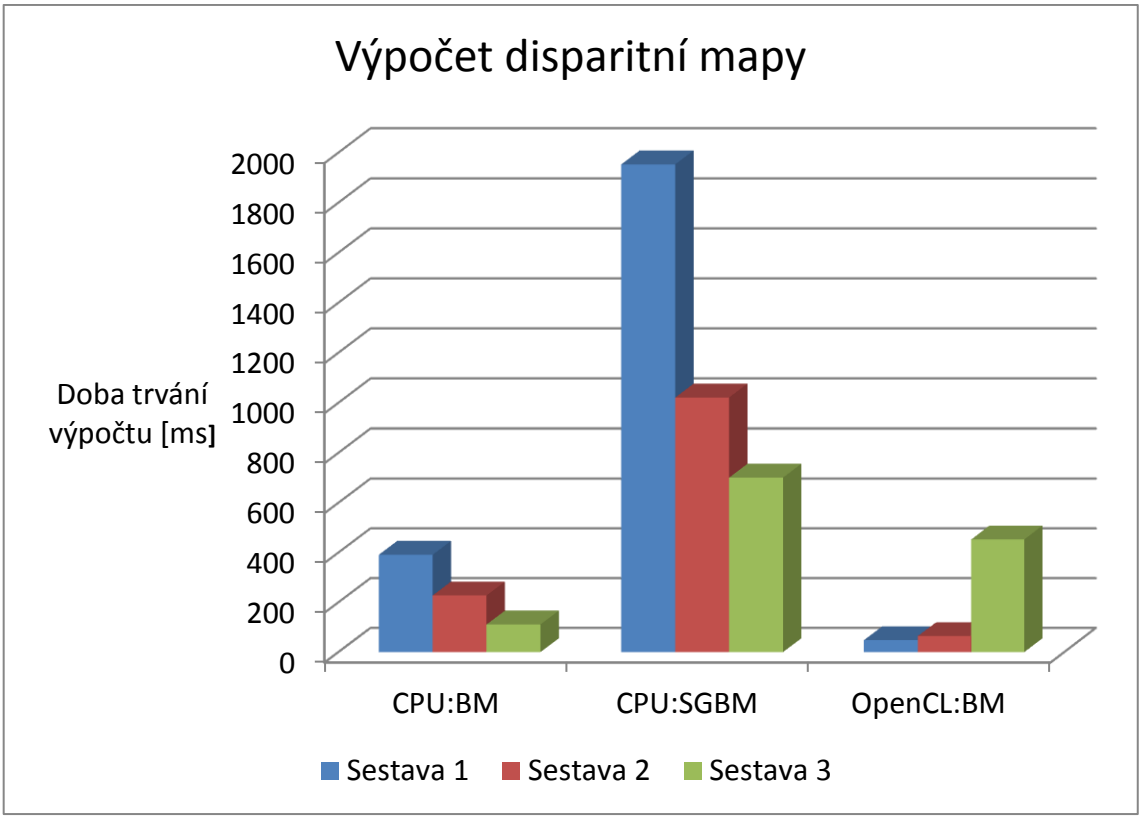
Testovací sestava č. 3: PC2

- Procesor: Intel Core i5 3570 (3,4GHz)
- Grafická karta: Intel HD Graphics 2500
- Operační systém: MS Windows 7 64-bit

V rámci měření jsou použity videosoubory pořízené pro levou a pravou kamerou o 347 snímcích. Každá sestava je zpracovává s výchozími parametry. Z výsledného času se vypočítá průměrná doba zpracování jedné disparitní mapy. V grafu je zakreslen čas, za jaký byla vykreslena disparitní mapa.

Tab. 1: Doba trvání výpočtu disparitní mapy na různých sestavách

Operace	Výpočet disparitní mapy		
Výpočet pomocí	CPU:BM	CPU:SGBM	OpenCL:BM
	Doba trvání výpočtu [ms]		
Sestava 1	391,11	1953,06	47,83
Sestava 2	227,64	1021,01	64,22
Sestava 3	111,13	700,95	452,67



Graf 6: Doba trvání výpočtu disparitní mapy

Na grafu 6 jsou vidět průměrné časy výpočtu disparitních map. Na prvních dvou částech grafu je vidět vliv zvyšujícího se výkonu procesorů. Procesor v sestavě 1 je nejstarší a následující dva jsou novější, ale procesor ze sestavy 2 je o jednu generaci Intel i5 starší a je mobilní a proto nemá tak velký výkon. Výpočet BM na procesoru je

algoritmicky méně náročný než SGBM, proto jsou doby výpočtu výrazně menší a nárůst výkonu lineárnější.

Na třetí části grafu je vidět pokles výkonu na třetí sestavě, protože nemá samostatnou grafickou kartu, ale OpenCL výpočty provádí procesor.

Nejrychleji proběhl výpočet na samostatných grafických kartách při použití OpenCL. Může za to vysoký stupeň paralelismu, který nám tyto karty umožňují využít. Zároveň je vidět rozdíl 34% výpočetního výkonu mezi desktopovou (sestava 1) a notebookovou grafikou (sestava 2). Notebooková grafika je navržena tak, aby měla malo spotřebu energie na úkor výkonu.

7.7 Výstupy programu

V této kapitole jsou zařazeny ukázky výsledných disparitních map pořízené stereoskopickou soustavou, tedy párem kalibrovaných kamer. Všechny obrázky v této kapitole jsou vytvořeny pomocí modelové aplikace.

Na obrázku 17 je zachycena modelová scéna s různě vzdálenými předměty od kamer. Kamery jsou upevněny na stojánku a jsou od sebe vzdáleny 30 cm. Na obrázcích níže je ukázána funkce modelové aplikace pro měření vzdálenosti v reálném čase a tvoření disparitní mapy.

Obr. 17: Fotografie z levé a pravé kamery

Soustava kamer byla softwarově nakalibrována a kalibrační data jsou přístupná modelové aplikaci pro výpočet disparitní mapy. Aplikace při každém výpočtu disparitní mapy rektifikuje stereosnímky. Rektifikovaný stereosnímek je zobrazen na obrázku 18.

Obr. 18: Rektifikovaný stereopár

V programu lze nastavit anaglyfické zobrazení aktuálně snímané scény tak, jak je vidět na obrázku 19. Anaglyfické zobrazení je vlastně snímek, na kterém je červené spektrum barev z levé kamery, zelené a modré spektrum barev z pravé kamery.

Obr. 19: Anaglyfické zobrazení

V příloze 1 jsou představeny disparitní mapy vytvořené modelovou aplikací. Měření vzdáleností v reálném čase je předvedeno v ukázkovém videu, jako příloha této práce na přiloženém DVD.

8 ZÁVĚR

Dle postupů uvedených v úvodu práce, jsem soustavu kamer zkalibroval. Pořízené stereoskopické snímky respektive snímky obecné scény spolu se zjištěnými parametry kamer byly základem pro dosažení cíle práce a to vytvoření hloubkové neboli disparitní mapy.

Grafy chybovosti kalibrace shora uvedené poukazují na chyby reprojekce, které nastaly při kalibrování kamer v závislosti na počtu snímků použitých při kalibraci. Bylo zjištěno, že při menším než optimálním počtu snímků chybovost roste. Stejný výsledek byl zjištěn při větším počtu snímků, než je počet optimální.

Grafy chybovosti měření vzdálenosti ukazují, že přesněji měří Block-matching algoritmus vypočítávaný pomocí OpenCL. Stereoskopická soustava je schopná měřit vzdálenost předmětu před ní s nízkou chybou měření ve vzdálenosti, která je asi 7 větší než je vzdálenost mezi kamerami.

Pro dosažení lepších výsledků můžeme použít náročnější algoritmy pro výpočet disparitní mapy, ale obávám se, že budou velmi výrazně prodlužovat čas výpočtu. Dále by je možné použít kamery určené pro stereo snímání, některé dnes prodávané se dají synchronizovat, aby snímaly zároveň. Přístup k fotografiím přes rozhraní USB není úplně nejvhodnější, protože má velké prodlevy a neumožňuje přímý přístup do paměti kamer.

Dokumentace nasnímané scény a z ní pocházejících disparitních map je uvedena shora.

Závěrem lze říci, že uvedeného cíle práce bylo dosaženo.

ZDROJE

[1] BOURKE, Paul. *Calculating Stereo Pairs* [online]. 1999 [cit. 2013-12-08]. Dostupný z WWW: <http://local.wasp.uwa.edu.au>.

[2] BRADSKI, Gary; KAEHLER, Adrian. *Learning OpenCV*. [s.l.] : O'Reilly, 2009.

[3] ČÍŽEK, Petr. *Prostorové zobrazování.* [s.l.], 2005. 59 s. Diplomová práce. Dostupný z WWW: <http://herakles.zcu.cz/~skala/VID/Data/PetrCizek_DP2005.pdf>.

[4] HARTLEY, Richard; ZISSERMAN, Andrew. *Multiple View Geometry in Computer Vision : Second Edition* . [s.l.] : [s.n.], 2003. 672 s.

[5] LANGANIÉRE, Robert.*OpenCV 2 Computer Vision Application Programing Cookbook* Pack Publishing, 2011. ISBN 978-184-9513-241.

[6] ING. ŘÍHA, Kamil; ING. HUJKA, Petr. *Elektrorevue* [online]. 2005 [cit. 2014-01-22]. *Epipolární geometrie* Dostupné z WWW:<http://www.elektrorevue.cz>.

[7] TRIOLET D., *"Nvidia CUDA: preview," BeHardware*, [online] 2007 [cit. 2014-04-26]. Dostupné z WWW: <http://www.behardware.com>.

[8] WOOLLEY, Cliff. *Introduction to OpenCL*, [online] 2011 [cit. 2013-12-22]. Dostupné z WWW: <http://www.cc.gatech.edu>.

[9] OpenCL Specification 2.0[online]. 2011 [cit. 2014-01-28] Dostupné z WWW: <http://www.khronos.org>.

[10] LUNOKHOD, Semi-Global Matching, [cit. 2014-05-26] Dostupné z WWW: <http://lunokhod.org>.

[11] CYGANEK, B.; SIEBERT, P., J.; *An Introduction to 3D Computer Vision Techniques and Algorithms*: John Wiley&Sons 2009, ISBN: 978-0-470-01704-3

[12] OpenCV 2.4.9.0 documentation [online] [cit. 2014-05-20] Dostupné z WWW: <http://docs.opencv.org>.

[13] Configuring Image Processing Properties [online] [cit. 2014-05-21] Dostupné z WWW: <http://www1.adept.com >.

[14] Mac OS X Developer Library [online]. 2013 [cit. 2014-05-15]. OpenCL Programming Guide for Mac OS X. Dostupné z WWW: <http://developer.apple.com/library/mac/#documentation/Performance/Conceptual/OpenCL_MacProgGuide/Introduction/Introduction.html>.

[15] MANYTHREADS. *Part 2: OpenCL™ – Memory Spaces* [online]. 27. 9. 2010 [cit. 2014-04-22]. Dostupné z WWW: <http://www.codeproject.com/Articles/122405/Part-2-OpenCL-Memory-Spaces>

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

API	Application Programming Interface
BM	Block-mathing
CPU	Central Processing Unit
GPU	Graphic Processing Unit
OpenCL	Open Computing Language
OpenGL	Open Graphics Library
OpenCV	Open Source Computer Vision
SAD	Sum of absolute differences
SGBM	Semi Global Block-mathing
SIMD	Single Intruction, Multiple Data
SSD	Sum of squared differences
SSE	Streaming SIMD Extensions

SEZNAM PŘÍLOH

Příloha 1 – Doplňující obrázky pro kapitolu 7.7 – ukázky disparitních map.

Příloha 2 – Ovládání programu, parametry nastavení, tabulka možností změn parametrů nastavení výpočtu disparitní mapy.

Příloha 3 – Zdrojový kód kernelu pro GPU.

Příloha 4 – Zdrojové kódy vytvořených funkcí vytvořených v prostředí MS Visual Studio 2010. Příloha je umístěna na přiloženém DVD.

Příloha 5 – Zkompilované programy včetně ukázkové konfigurace a vstupních dat. Příloha je umístěna na přiloženém DVD.

Příloha 6 – Ukázková videa. Příloha je umístěna na přiloženém DVD.

Příloha 7 – Ukázkové fotografie, kalibrační fotografie. Příloha je umístěna na přiloženém DVD.

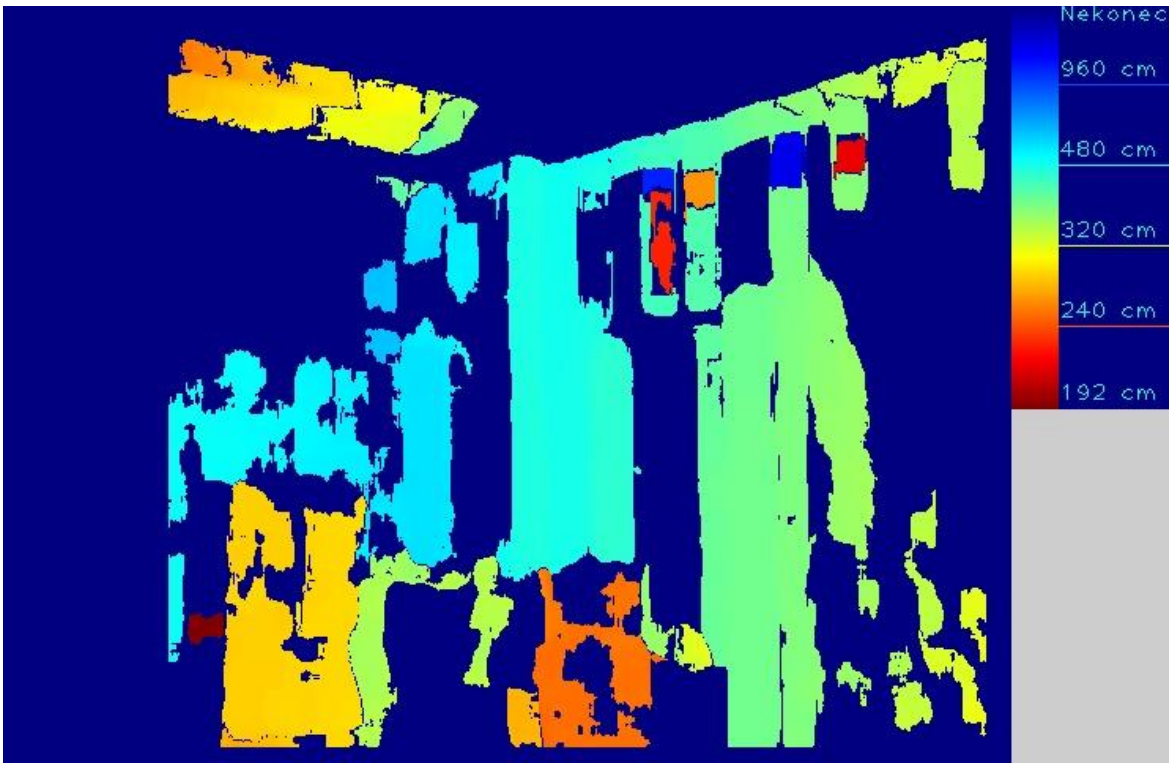
Příloha 8 – Naměřené grafy. Příloha je umístěna na přiloženém DVD.

Příloha 9 – Elektronická verze této práce. Příloha je umístěna na přiloženém DVD.

PŘÍLOHA 1

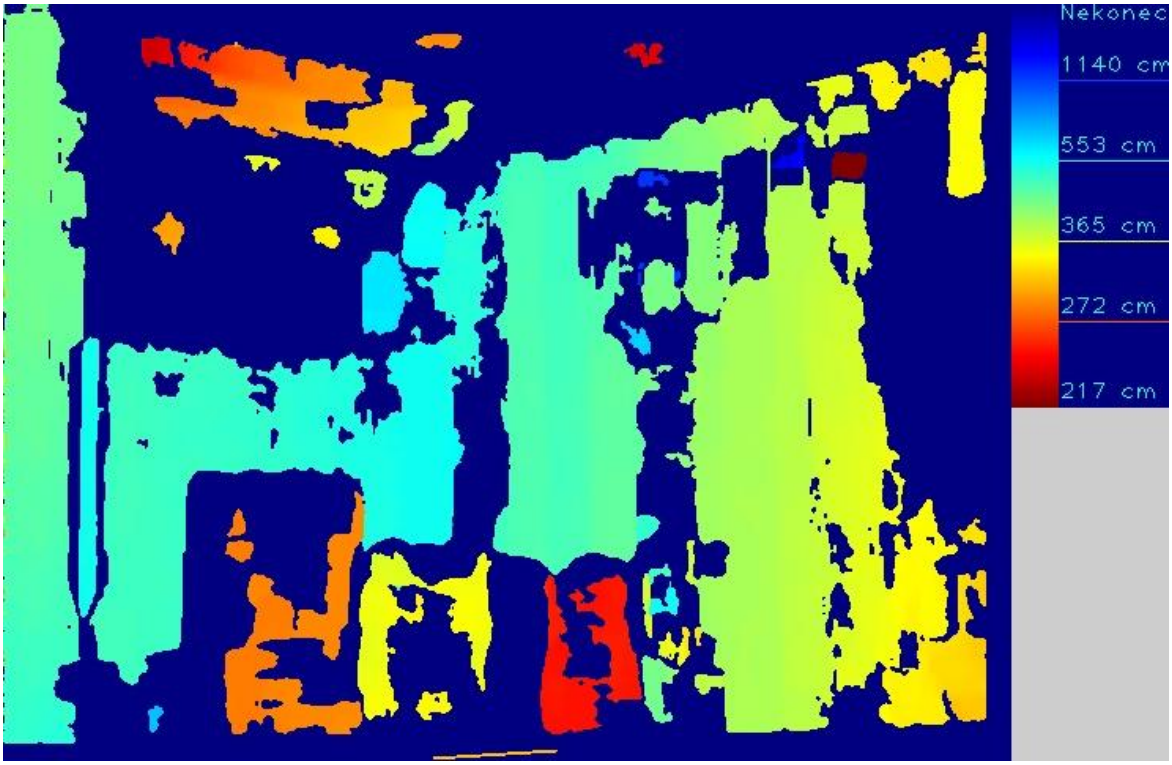
Disparitní mapa (CPU:BM)

Disparitní mapa obarvená (CPU:BM)

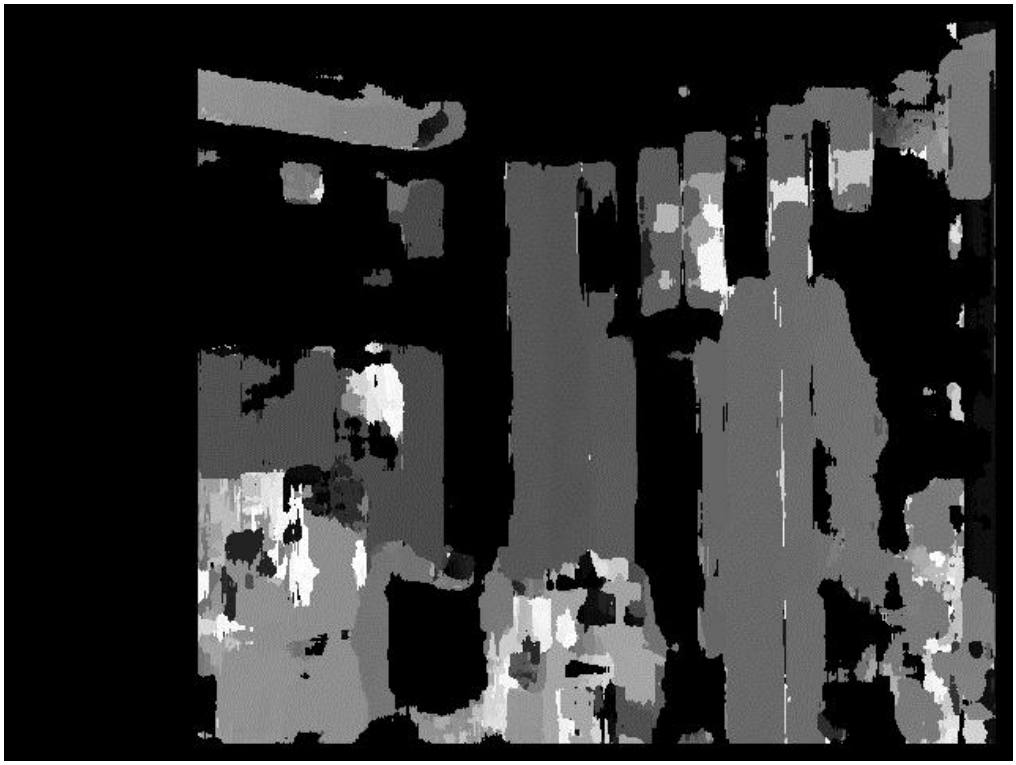


Disparitní mapa (CPU:SGBM)

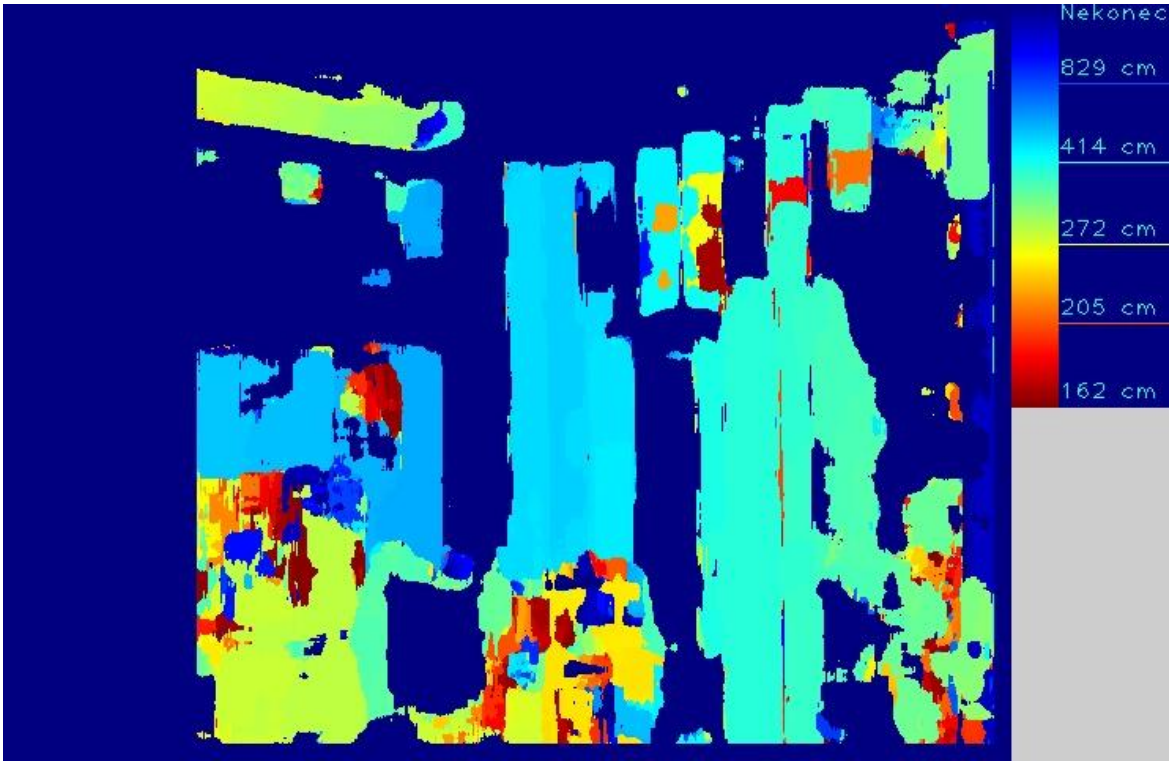
Disparitní mapa obarvená (CPU:SGBM)



Disparitní mapa (OpenCL:BM)



Disparitní mapa obarvená (OpenCL:BM)



PŘÍLOHA 2

Ovládání programu:

Program opakovaně snímá stereosnímky z nastaveného vstupu, vypočítá z nich disparitní mapu a zobrazí ji. Pokud je vypnutý videomód, čeká po zobrazení disparitní mapy na vstup uživatele. Po stisknutí klávesy „mezerník“ se aktuálně zpracovaný snímek uloží. Po stisknutí klávesy „enter“ se aktuálně zpracovaný snímek neuloží a načte se další stereosnímek. Na ostatní klávesy reaguje program nezávisle na nastavení videomódu, viz tabulka 1. Klávesou „x“ je možné program kdykoliv ukončit.

Parametry nastavení:

1. **Systémové parametry:**

- calibrationDirectory – určuje jméno adresáře, ve kterém jsou uložena data z kalibrace
- photoDirectory – určuje adresář, kam se mají ukládat nasnímané fotografie
- leftCameraIndex – index levé kamery (pokud bude použit vstup z kamer)
- rightCameraIndex – index pravé kamery (pokud bude použit vstup z kamer)
- leftCameraInput – cesta ke zdroji dat pro levou kameru, buď cesta k videosouboru nebo formát cesty k fotografiím, kde se %d nahradí číslem snímku
- rightCameraInput – cesta ke zdroji dat pro pravou kameru, buď cesta k videosouboru nebo formát cesty k fotografiím, kde se %d nahradí číslem snímku
- openCLDeviceIndex – index zařízení použitého pro OpenCL, pokud je zadaná „1“, je použita autodetekce (najde se zařízení s maximálním počtem výpočetních jednotek), výpis všech zařízení pro OpenCL uživatel nalezne v souboru opencl_devices.txt po prvním spuštění programu
- maxFPS – určuje počet snímků za sekundu, které budou použity, pokud se bude zaznamenávat barevná disparitní mapa nebo vstup z kamer do videosouboru
- videoMode – zapíná video mód, viz. dále input - určuje druh vstupu, „0“ pro vstup z kamer, „1“ pro načítání fotek ze souboru, „2“ pro načítání obrázků z videosouboru
- storeVideo – nastavením hodnoty „1“ program ukládá barevnou disparitní mapu do videosouboru (pokud jsou vstupní obrazová data pořizována z kamer, ukládají se i tyto data do samostaných videosouborů)

2. **Zobrazovací parametry:**

- Anaglyf – pro každý snímek se zobrazí okno s Anaglyfem
- BWDisparityMap – pro každý snímek se zobrazí okno s šedotónovou disparitní mapou
- RectifiedInput – zobrazí okno s rektifikovanou vstupní dvojicí snímků
- Legend – nastavením hodnoty „1“ zapíná/vypíná zobrazení legendy, která ukazuje vzdálenosti pro minimální a maximální nalezenou disparitu a pro 4 hodnoty mezi nimi
- MouseDistance – 0/1 vypíná/zapíná zobrazování vzdálenosti objektu od kamer při pohybu myši v okně barevné disparitní mapy

3. **Startovní parametry matchingu:**

- OpenCL – 0/1 vypíná/zapíná použití OpenCL
- BM – 0/1 určuje metodu block matchingu SGBM/BM,
- preFilterCap, preFilterSize, SADWindowSize, minDisparity, numberOfDisparities, textureThreshold, averageTexThreshold, uniquenessRatio, speckleWindowSize, speckleRange, disp12MaxDiff, fullDP, XSobelFilter jsou vstupní parametry pro block matchingové metody, viz tabulka 1.

4. **Parametry koncové filtrace:**

- UseRangePostFiltering – 0/1 vypíná/zapíná ořezání disparitní mapy v daných mezích před převedovem na 256 stupňovou šedotónovou disparitní mapu
- MinDistance – určuje minimální vzdálenost, která bude viditelná
- MaxDistance – určuje maximální vzdálenost, která bude viditelná
- RealUnits – 0/1 určuje typ jednotek vzdálenosti (disparity/ centimetry)

Tabulka možností změn parametrů nastavení výpočtu disparitní mapy

Parametr	Klávesa pro zvýšení hodnoty	Klávesa pro snížení hodnoty	Hodnota změny
preFilterCap	Q	W	1
preFilterSize	B	N	2
SADWindowSize	E	R	2
minDisparity	U	I	16
numberOfDisparities	O	P	16
textureThreshold (CPU)	A	S	1
uniquenessRatio	D	F	1
speckleWindowSize	G	H	10
speckleRange	J	K	5
disp12MaxDiff	C	V	3
Přepínání BM/SGBM	M		true/false
Přepínání OCL/CPU	T		true/false
FullDP (pro SGBM)	L		true/false
XSobel filtr	1		true/false
texture Threshold (OCL)	2	3	*2 / *(1/2)
UseRange	5		true/false

PŘÍLOHA 3

```
__kernel void subPixelKernel(
      __global unsigned int* cols,
      __global unsigned int* rows,
      __global unsigned char* disp,
      __global unsigned char* dest,
      __global unsigned char* left,
      __global unsigned char* right)
{
      int gid = get_global_id(0); // ziskani globalniho indexu
celeho pole

      int area_length = 1;
      int area_size = area_length * 3;

      int y = gid / cols[0];
      int x = gid - (cols[0] * y);

      if (x < area_size / 2 || x > cols - area_size / 2 || y <
area_size / 2 || y > rows - area_size / 2) {
            dest[gid] = 0;
            return;
      }

      float sumA = 0;
      float sumB = 0;
      float sumC = 0;
      for(int i=0; i<=area_length; i++) {
         for(int j=-area_length; j<=area_length; j++) {
                  int indexA = (y + j) * cols[0] + (x + i);
                  int indexB = indexA + area_length;
                  int indexC = indexB + area_length;
                  sumA += sqrt((float)(left[indexA] - right[indexA -
disp[indexA]]));
                  sumB += sqrt((float)(left[indexB] - right[indexB -
disp[indexB]]));
                  sumC += sqrt((float)(left[indexC] - right[indexC -
disp[indexC]]));
         }
     }

      dest[gid] = disp[gid] + (unsigned char)((sumB - sumC) / 2 *
(sumA - 2 * sumB + sumC));
}
```