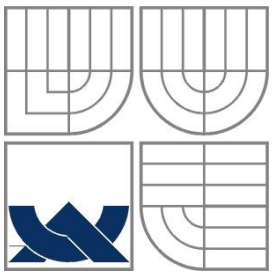
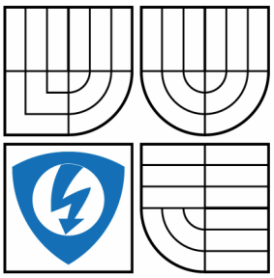


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ZPRACOVÁNÍ STEREO SNÍMKŮ NA GRAFICKÉ KARTĚ

GPU ACCELERATED ATEREO IMAGE PROCESSING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Jaromír Polák

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Miloslav Richter, Ph.D.

BRNO 2013

ORIGINÁLNÍ ZADÁNÍ DIPLOMOVÉ / BAKALÁŘSKÉ PRÁCE

Poznámka:

Červeným písmem je uvedeno, co má být napsáno resp. Aktualizováno!!

Abstrakt

Diplomová práce se zabývá 3D rekonstrukcí za použití stereo kamery. Tato práce má ukázat využitelnost GPU akcelerace při náročných aplikacích.

Klíčová slova

3D rekonstrukce, stereovidění, kalibrace, rektifikace, CUDA, GPU.

Abstract

This thesis deals with 3D reconstruction using stereo cameras. This Work is to show theusefulness of GPU acceleration for sophisticated algorithm.

Keywords

3D reconstructions, stereovision, CUDA, GPU.

Bibliografická citace:

POLÁK, J. *Zpracování stereo snímků na grafické kartě*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 55s. Vedoucí diplomové práce byl Ing. Miloslav Richter, Ph.D

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Zpracování stereo snímků na grafické kartě 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 jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **17. května 2013** ……
podpis autora

Děkuji vedoucímu diplomové práce Ing. Miloslavu Richterovi, Ph.D. a Ing. Tomáši Babincovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **17. května 2013** ……

podpis autora

Obsah

1 ÚVOD

Téma diplomové práce, které jsem si vybral, je z oboru počítačového vidění. Počítačové vidění se snaží o získání informací ze snímků či videa. Tyto informace dále zpracovává a vyhodnocuje. Jde například o zjištění barvy, tvaru, jasu, spektra, počtu, vzdálenosti a mnoho dalších. Počítačové vidění se dělí dále na mnoho kategorií, z nichž moje spadá do kategorie 3D vidění (stereovidění).

Stereo v překladu znamená „prostorový“. Stereo vidění tedy znamená prostorové vidění. Člověk je uzpůsoben prostorovému vidění pomocí očí. Tento lidský model používám v diplomové práci, místo očí však byly použity dvě kamery, které jsou pevně upevněny k sobě a jsou synchronizované. Jedná se o tzv. stereokameru, která získává dva snímky v jednom okamžiku z různých pohledů ze zkoumaného prostředí.

Cílem diplomové práce je vytvořit algoritmus, který bude schopný z nasnímaných snímků kalibrovat kameru, a prostřednictvím výše zmíněných algoritmů vytvořit 3D model nasnímané scény. Tento program poběží na CPU (Central Processing Unit). Za použití měření v programu Intel Parallel studio XE 2013 je možné získat přehled o náročnosti jednotlivých funkcí. Nejnáročnější funkce je třeba přeprogramovat na GPU (Graphic Processing Unit) pomocí architektury CUDA, a tím dosáhnout nebo se alespoň přiblížit real-time odezvě.

Problematika, kterou budu v diplomové práci řešit, je kalibrace kamery, rektifikace, detekce významných bodů, hledání korespondencí, výpočet disparity, implementace algoritmů pro paralelní zpracování na GPU a v neposlední řadě výpočetní náročnost.

Nyní stručně uvedu jednotlivé kroky řešení. Velmi důležitá je kalibrace stereokamery, která získá parametry pro další výpočty a měření. Následuje rektifikace a nalezení významných bodů v obraze. Dále identifikace korespondujících bodů, tedy bodů v jednom a druhém snímku, které si odpovídají, ze kterých vypočítám disparitní mapu a z ní a ze získaných parametrů z kalibrace jsem schopen vytvořit 3D model scény, který jsem nasnímal.

Diplomová práce je směřovaná na praktické využití paralelního výpočtu na grafické kartě. V práci bude použita grafická karta od firmy Nvidia s platformou CUDA. Tuto využitelnost je třeba prokázat měřením výpočetní náročnosti.

2 TEORETICKÁ ČÁST

V této kapitole provedu rozbor problematiky týkající se mé diplomové práce. Prvním odrazovým můstkem je kalibrace kamery, ta je velmi důležitá pro další výpočty a měření. Kalibrace určí vnitřní (např. typ zkreslení) a vnější (např. vzdálenost kamer) parametry kamery, které jsou důležité pro rektifikaci a výslednou 3D rekonstrukci.

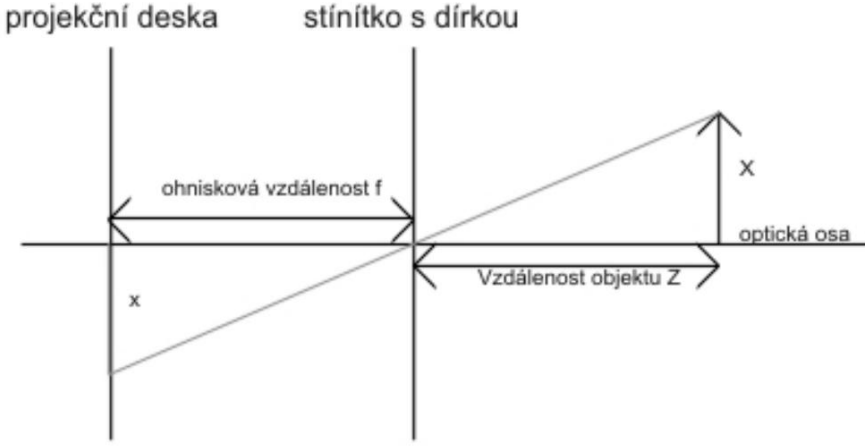
Po zjištění parametrů kamery bude provedena rektifikace. Jedná se o úpravu snímků, která zaručí, že sesouhlasí u obou snímků řádky, zajišťuje odstranění zkreslení a další. Předpokládám, že snímek z kamery bude zkreslený, a proto po rektifikaci vzniknou místa, kde snímek neponese žádnou informaci a bude muset být ořezán. Poté naleznu významné body v obraze, které mají ve snímku jasně definovanou pozici.

Dále musím najít korespondence, tedy významné body v jednom a druhém snímku, které si odpovídají. Z nalezených korespondencí je nutné vypočítat disparitní mapu, ze které mohu vyčíst vodorovný posun korespondujících pixelů z levého snímku vůči pravému snímku. Z disparitní mapy a ze získaných parametrů z kalibrace jsem schopen vytvořit 3D model scény, který jsem nasnímal.

Tento celý proces je velmi výpočetně náročný, proto se moje diplomová práce dále zabývá programováním na GPU, architekturou CUDA a paralelním výpočtem. Předpokládám, že tyto nástroje budou vhodné pro dosažení real-time odezvy výpočtů. Podobnou problematikou se zabývá i literatura [1],[3],[5],[20].

2.1 Matematický model kamery

Nejjednodušší model kamery se nazývá dírková komora.
Jak je vidět na obrázku č. 1, světlo dopadající na projekční desku prochází malým otvorem. Je zřejmé, že tedy na jeden bod projekční desky dopadá jenom jeden paprsek světla. Velikost získaného obrazu je závislá pouze na ohniskové vzdálenosti f. Zmiňovaná ohnisková vzdálenost však není stejná jako ohnisková vzdálenost v optice, avšak zde se používá pro zjednodušení.

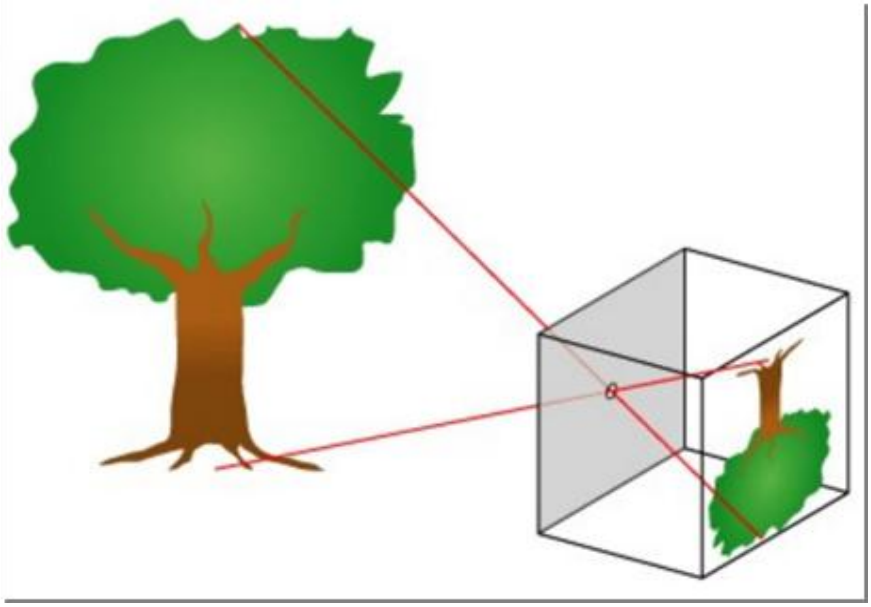


Obrázek 1: Matematický model kamery [1].

Na obrázku č. 1 je zobrazen ideální případ, kde ohnisková vzdálenost je stejná jako vzdálenost projekční desky. Dále můžeme vidět vzdálenost Z, která představuje vzdálenost objektu od kamery, a bod X, který znázorňuje velikost objektu. Za povšimnutí stojí podobnost trojúhelníků, které vznikly po stranách stínítka. Z této podobnosti získáme rovnici.

$$\frac{-x}{f} = \frac{X}{Z} \qquad [1]$$

Z této rovnice pak jednoduše získáme velikost obrazu. Záporné znaménko nám symbolizuje opačnou orientaci vzhledem k optické ose. To můžeme vidět na obrázku č. 2.



Obrázek 2: Praktická ukázka dírkové kamery [2].

Abychom se vyhnuli převrácenému obrazu, použijeme ekvivalentní model, jako je model dírkové kamery, který má jednodušší matematický model.
Bod průniku paprsku s projekční rovinou chápeme jako hlavní bod.

Obrázek 3: Ekvivalentní model dírkové kamery [1].

Podobně jako u prvního modelu platí podobnost trojúhelníků, avšak bez převráceného obrazu.

$$\frac{x}{f} = \frac{X}{Z} \qquad [2]$$

Jelikož žádný čip v kameře není přesně na optické ose, je nutné zavést další parametry cx a cy pro zobrazení posunutí senzoru. Tyto parametry vyjadřují rozdíl polohy středu souřadnicového systému optického senzoru a středu souřadnicového systému projekční roviny. Výsledkem je relativně jednoduchý model, kdy bod v trojrozměrném prostoru Q [X, Y, Z] je promítán na projekční rovinu. Tedy i na daný pixel o souřadnicích [x, y] na obrazovce dle rovnice:

$$x = f_x \frac{X}{Z} + c_x, \qquad y = f_y \frac{y}{Z} + c_y \qquad [3]$$

V následující rovnici jsou dvě ohniskové vzdálenosti *fx* a *fy*. Konstanta *fx* je konstantou kamery ve směru *x* a *fy* pro směr *y*. Tyto rovnice je vhodné přepsat do maticového zápisu:

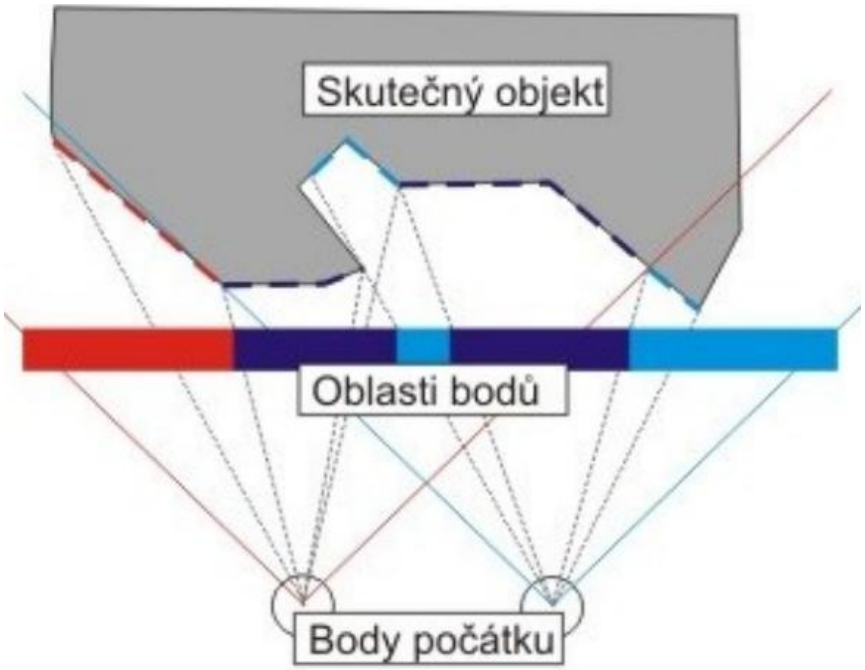
$$w\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \approx q = WQ \qquad [4]$$

Kde: q je projekce bodu Q v kameře,
Q je bod v obecném souřadnicovém systému,
W je projekční matice.

Uvedená transformace je počítána v homogenních souřadnicích. Tyto souřadnice nám umožní snadnější operace s maticemi. Rozšířením výsledného vektoru o jednu souřadnici získám váhový vektor. Po podělení jeho váhou získáme správný výpočet. Této operaci se říká normalizace vektoru v homogenních souřadnicích.

2.2 Model stereokamery

Jak již bylo řečeno, stereokamera nám zajišťuje synchronizované snímání scény pomocí dvou pevně uchycených kamer. Ze snímků pořízených ze stereokamery jsem schopen dopočítat prostorovou souřadnici pro všechny body, které jsou zobrazeny na obou snímcích.

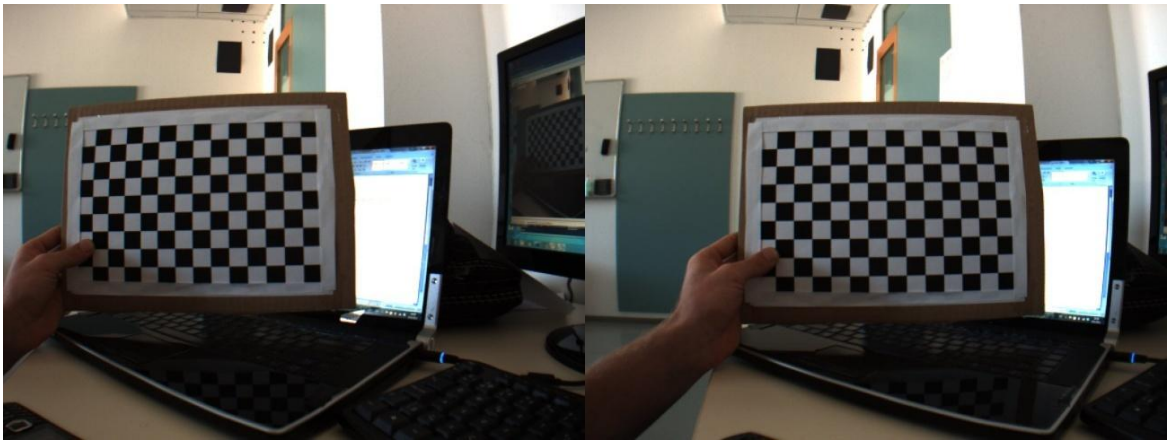


Obrázek 4: Znázornění snímání stereokamerou [1].

Na obrázku č. 4 je ukázka, jak se zorná pole kamer překrývají, a vidíme, u jakých bodů budeme moci počítat vzdálenost. Je to možné u fialových bodů, neboť tuto oblast snímají obě kamery. U modré a červené barvy snímá oblast pouze jedna z kamer, proto vzdálenost dopočítat nelze.
Stereo kamera, se kterou pracuji na diplomové práci, se nazývá Bumblebee 2 od firmy PTGREY. Kamera je připojena k pracovní stanici (PC) za pomocí sběrnice IEEE 1394a.

Obrázek 5: Použitá stereokamera Bumblebee 2 [17].

Po připojení k počítači a spuštění programu (*triclopsDemo.exe*) dodávaného k tomuto zařízení můžeme z kamery získat syrové (neupravené) snímky.
Neupravený snímek ze stereokamery můžeme vidět na obrázku č. 6. Tento snímek je potřeba rozdělit na dva - na snímek z levé kamery a snímek z pravé kamery.



Obrázek 6: Získaný snímek ze stereokamery (neupravené) vlevo z levého snímače, vpravo z pravého snímače.

2.3 Kalibrace kamery

Kalibrací kamery zjišťujeme vnitřní a vnější parametry kamery. Bylo by možné použít parametry poskytnuté výrobcem, avšak kalibrační parametry jsou zakódovány nepřístupně v hardware a použitý model kamery není zcela zřejmý. Parametry jsou zkompilované v DLL knihovnách a nespecifikovány v dokumentaci. Nechtěl jsem být omezen pouze na použití výpočetních nástrojů dodaných výrobcem. Proto jsem provedl vlastní kalibraci. Kalibrace je velice důležitý proces pro následující práci a měření.

2.3.1 Vnitřní parametry kamery

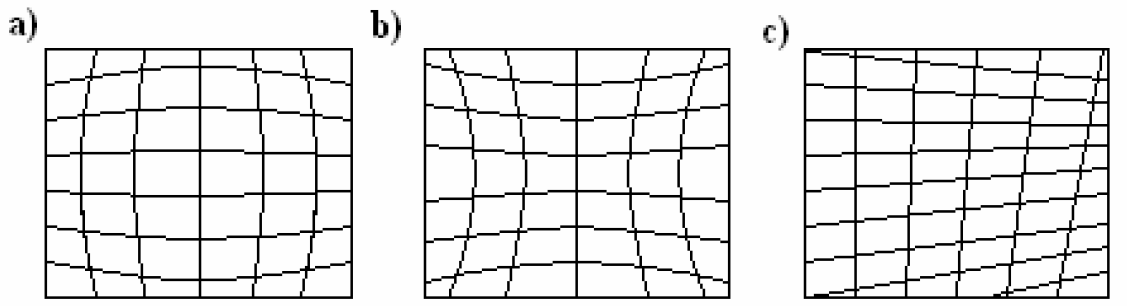
Abych mohl určit projekční matici W, z kapitoly 2.1 a koeficienty zkreslení, nasnímám vhodný vzor z několika různých pohledů a různých vzdáleností. Vhodný vzor je takový, ze kterého dokážeme určit co nejvíce parametrů, jako je například typ zkreslení (soudkovité, poduškovité, natočení, zkosení). Ve vzoru by mělo být hodně významných bodů (přechod černá-bílá). Dále popisuji šachovnicový vzor, se kterým v diplomové práci pracuji, a považuji jej za ideální. Na základě znalosti rozměrů šachovnice mohu určit neznámé parametry. Tento vzor je vhodný i proto, neboť v knihovně OpenCV je implementována funkce pro rozpoznání rohů nasnímané šachovnice. Tato funkce je podrobněji popsána ve zdroji [13].

2.3.2 Vnější parametry

Abych mohl provést rektifikaci obrazu, musím určit vzájemnou polohu kamer. Hledání polohy kamer provádíme sledováním kalibračního objektu danou kamerou. Hledáme takovou transformaci, která zobrazí body kalibračního objektu na objekt ve snímku. Jsou dva přístupy k určení této transformace: lineární a nelineární.
Lineární: pro každou dvojici bodu i jeho obrazu můžeme sestrojit rovnici, kde jsou zobrazeny parametry kalibrační matice.
Nelineární: hledá přímo parametry posunutí a natočení kamery v obecném souřadnicovém systému. Následně z těchto parametrů sestrojíme transformační matice.

2.3.3 Geometrické zkreslení

Geometrických zkreslení je celá řada, avšak během mé diplomové práce budu zohledňovat pouze poduškovité, soudkovité a natočení detektoru, neboť mají největší vliv. Toto zkreslení může mít za následky vzájemný pohyb snímače a předmětu, nevhodné zaostření, vadu optické soustavy, nelinearitu opticko-elektrického senzoru.

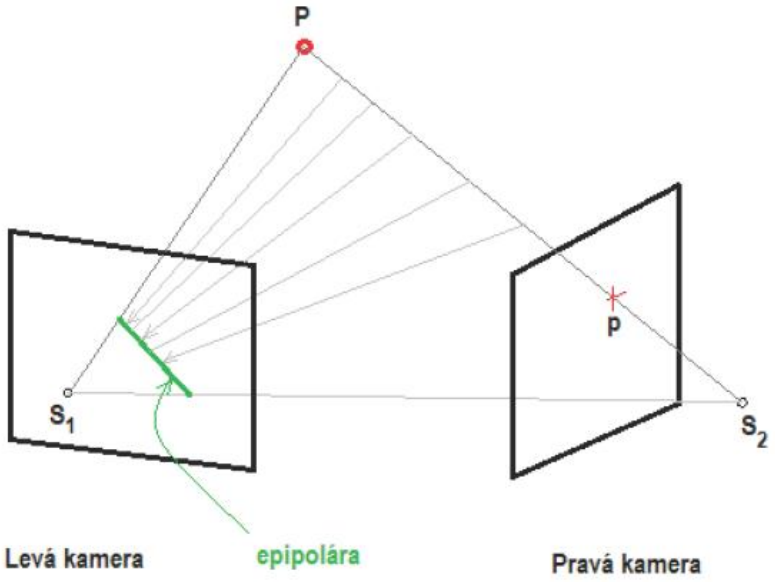


Obrázek 7: Geometrické zkreslení a) soudkovité, b) poduškovité, c) natočení detektoru [18].

Kalibrační objekt může mít dva tvary: rovinný a prostorový. U obou platí, že by měl být v obraze viditelný a co možná největší.

2.4 Epipolára

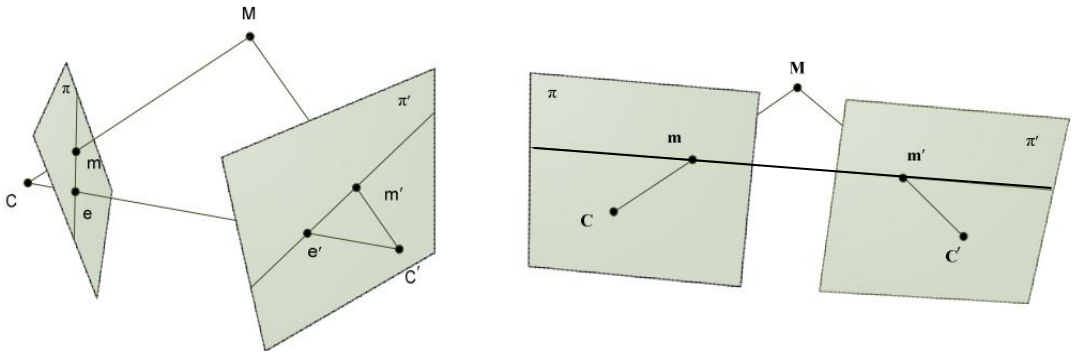
Epipolárou nazýváme přímku, na kterou se do prvního obrazu promítá bod z druhého obrazu. Stejný princip funguje i z druhého obrazu do prvního obrazu. Jde tedy o geometrii, která vznikla projekcí 3D scény. Využitím této přímky můžeme výrazně zjednodušit a zrychlit výpočet korespondujících bodů, neboť epipolára nám omezí oblast, kde tyto body budeme hledat.



Obrázek 8: Epipolární geometrie [3].

2.5 Rektifikace

Rektifikací se rozumí úprava snímku, která vede k přetransformování obrazů do stejné roviny (řádkové sesouhlasení) a vede k odstranění jeho vad. Rektifikace, která bude prováděna v diplomové práci, odstraní soudkovité zkreslení a srovná snímky z pravé a levé kamery tak, že korespondující body u obou snímků budou mít stejnou souřadnici Y. Tento fakt nám urychlí další výpočty.



Obrázek 9: Princip rektifikace [20].

2.6 Detekce významných bodů

Významný bod je takový, kde dochází k významným změnám jasu, podle kterých se snadno identifikuje. Významný bod je možné nalézt i při geometrickém zdeformování nebo při jasové změně. Existuje mnoho detektorů významných bodů, které se liší v parametrech: citlivost na šum, opakovatelnost, přesnost, citlivost na natočení apod.

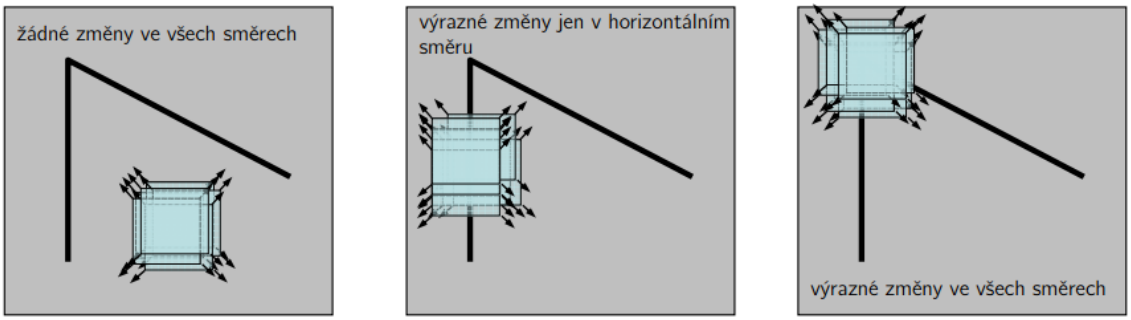
Moravcův detektor

Moravcův detektor je založen na hledání regionů, které jsou lokálním maximem ve vypočtených změnách intenzity daného obrázku. Jeho výhodami jsou jednoduchost, výpočetní nenáročnost a jeho nevýhodami je závislost na šumu a reakce na hrany. V dané rovnici je f (i, j) výstupní obraz a g (i, j) vstupní obraz.

$$f(i,j) = \frac{1}{8} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} \left(g(k,l) - g(i,j)\right) \qquad [5]$$

Harrisův detektor

Harrisův detektor je často používaný, neboť je nezávislý na rotaci, translaci a intenzitě jasu snímku. Dalším kladem je nízká výpočetní náročnost a odolnost proti šumu. Harrisův detektor je upravený Moravcův detektor. Změny se týkají „okénka“. Harrisův detektor používá okénko s Gaussovým vyvážením vnitřních hodnot. Gaussovo rozložení způsobí, že na okraji okénka mají body menší význam jako uprostřed. Dále používá autokorelaci, která zabraňuje anizotropní odezvě.



Obrázek 10: Princip Harrisova detektoru [21].

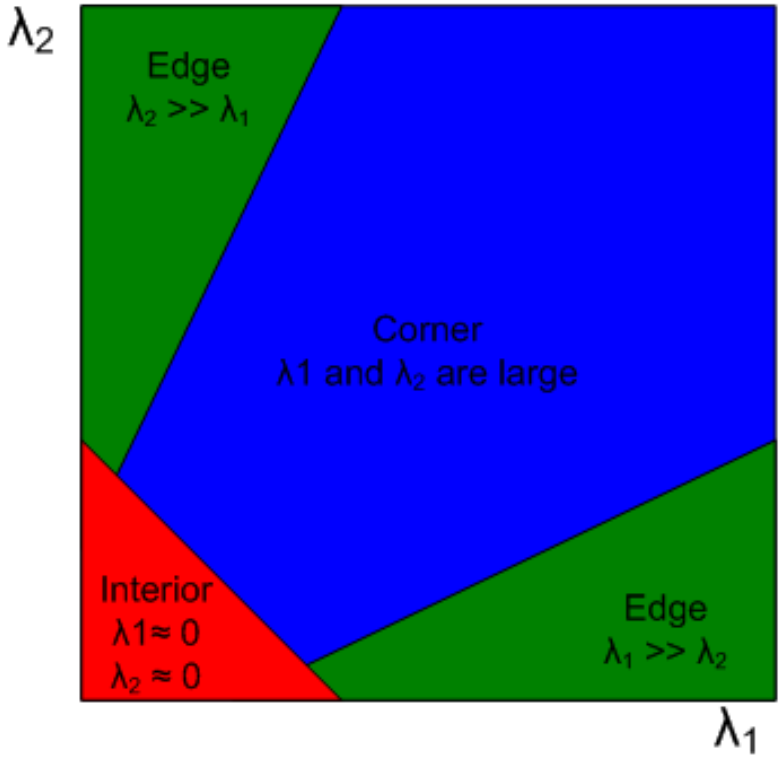
$$c(u,v) = \sum_{x,y} w(x,y) \times [f(x,y) - f(x+u, y+v)]^2 \qquad [6]$$

V rovnici jsou tyto proměnné: c(u,v) je autokorelační funkce,
w(x,y) je okénková gaussova funkce,
f(x,y) je obrazová funkce,
f(x+u,y+v) je posunutá obrazová funkce.

O tom, jestli daný bod je významný bod či nikoli, se rozhodne na základě hodnoticí funkce R(α ,β):

$$R(\alpha, \beta) = \alpha \cdot \beta - k \cdot (\alpha + \beta)^2 \qquad [7]$$

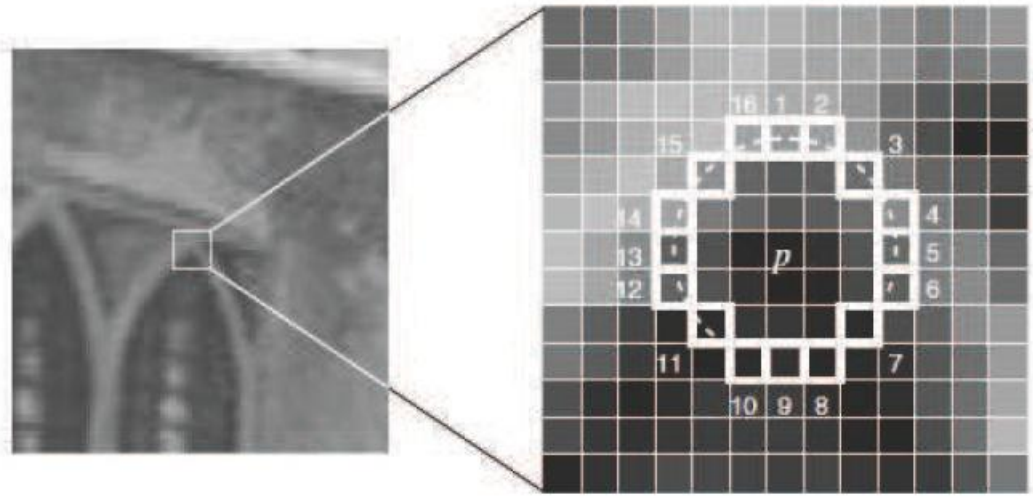
Kde α a β jsou vlastní čísla matice *C* a *k* je empiricky zjištěná konstanta.
Průběh hodnoticí funkce *R* je vidět na následujícím obrázku.



Obrázek 11: Výstup Harissova detektoru, určení rohu, přímky [19].

Fast Corner detektor

Vychází z analýzy bodů, které leží na kružnici kolem zkoumaného bodu. Detektor porovnává bod *p* s body na kružnici. Jestliže existuje *n* sousedících bodů na kružnici, které jsou tmavší nebo světlejší než bod *p*, tak je bod označený za významný.
Fast Corner detektor má dvě části: první část označí každý pixel na kružnici za světlejší, tmavší nebo stejný jako bod *p*. Druhá část vytvoří rozhodovací strom za pomocí algoritmu ID3. Ten rozhodne, který pixel nese nejvíce informací v rozhodování.



Obrázek 12: Fast Corner detektor [3].

Další detektory

SIFT (Scale Invariant Feature Transforms) je invariantní vůči rotaci a změně měřítka.
SURF (Speed-Up Robust Features) detektor, který by měl dosahovat znatelně rychlejšího výpočtu než SIFT. Existuje implementace pro architekturu CUDA i v OpenCV.

Sobelova maska

Operátor počítá derivaci ze tří bodů (z centrálního bodu a z bodů okolo). Díky tomu je robustnější proti šumu. Má celkem 8 masek, které počítá pro každý bod, a vezme tu největší hodnotu z nich a dosadí jí do centrálního bodu.

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \quad \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix} \quad \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix}$$

Hranu detekuji vždy tou maskou, kde probíhá souběžně s nulami v masce.

2.7 Hledání korespondencí

Problematika hledání korespondencí je rozsáhlá. Záleží na mnoha faktorech. Například na použitém detektoru významných bodů, na zvoleném algoritmu hledání korespondencí, který závisí na typu scény, na výpočetní závislosti atd. Touto problematikou se zabývá i literatura [15],[22].

Metoda SAD je nejjednodušší algoritmus, který porovnává podobnost snímků. Je založena na rozdílu jasové úrovně v okolí hledaného bodu. Její nevýhodou je, že má podmínku stejného jasu a kontrastu ve snímcích a že je závislá na geometrickém zkreslení.

$$SAD(x, y, r, s) = \sum_{v=-n}^{n} \sum_{u=-n}^{n} \text{abs}(f_1(x+u, y+v) - f_2(r+u, s+v)) \quad [8]$$

[x, y] je souřadnice porovnávaného bodu v prvním obrázku,
[r, s] je souřadnice porovnávaného bodu v druhém obrázku,
f1 je obrazová funkce prvního obrázku,
f2 je obrazová funkce druhého obrázku.

Metoda SSD je podobná jako metoda SAD. Výsledkem je nezáporná hodnota rozdílu jasové úrovně v okolí daného bodu. Ideální korespondence nabývá hodnoty 0. Tato metoda je jednoduchá, má však podmínku stejného jasu a kontrastu, proto je závislá na geometrickém zkreslení**.**

Dle vztahu

$$SSD(x,y,r,s) = \sum_{v=-n}^{n} \sum_{u=-n}^{n} (f_1(x+u, y+v) - f_2(r+u, s+v))^2 \qquad [9]$$

[x, y] je souřadnice porovnávaného bodu v prvním obrázku,
[r, s] je souřadnice porovnávaného bodu v druhém obrázku,
f1 je obrazová funkce prvního obrázku,
f2 je obrazová funkce druhého obrázku.

Metoda NCC počítá korelaci okolí hledaného bodu. Je jednoduchá, nezávislá na změně jasu. Není invariantní vůči geometrickému zkreslení. Metoda nabývá hodnot <-1,1> a čím je okolí korespondujícího bodu podobnější, tím se hodnota blíží k 1.

$$NCC(x,y,r,s) = \frac{\sum_{v=-n}^{n} \sum_{u=-n}^{n} \left[\left(f_1(x+u, y+v) - \bar{f}_1(x,y) \right) * \left(f_2(r+u, s+v) - \bar{f}_2(r,s) \right) \right]}{\sqrt{\sum_{v=-n}^{n} \sum_{u=-n}^{n} \left(f_1(x+u, y+v) - \bar{f}_1(x,y) \right)^2 * \sum_{v=-n}^{n} \sum_{u=-n}^{n} \left(f_2(r+u, s+v) - \bar{f}_2(r,s) \right)^2}} \qquad [10]$$

[x, y] je souřadnice porovnávaného bodu v prvním obrázku,
[r, s] je souřadnice porovnávaného bodu v druhém obrázku,
f1 je obrazová funkce prvního obrázku,
f2 je obrazová funkce druhého obrázku.

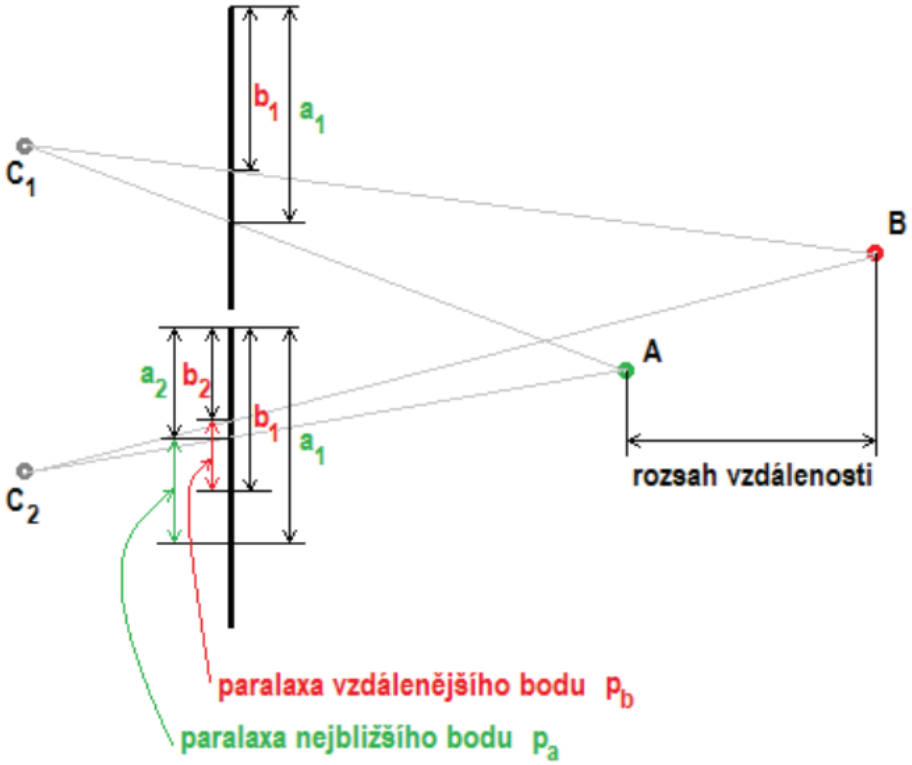
2.8 Disparitní mapa

Disparitní mapa nám určuje rozdíl polohy korespondujících pixelů z pravého a levého snímku. Tedy čím budou korespondující body od sebe vzdálenější (hodnota na ose x), tím bude bod ve výsledném obraze světlejší a předmět bude blíž kameře.

Určením maximální a minimální disparity omezíme okruh hledání korespondence. Tento fakt je znázorněn na obrázku č. 15 a je nedílnou součástí optimalizace této funkce. Na obrázku č. 13 můžeme vidět disparitní mapu, kde hodnota daného pixelu je rozdíl vodorovné polohy korespondujících pixelů. Tento disparitní snímek byl vytvořen z obrázku č. 14.

Obrázek 13:Disparitní mapa [9].

Obrázek 14: Vstupní obrázky pro disparitní mapu: teorie [převzato z example for CUDA 5.0].



Obrázek 15: Maximální a minimální paralaxa použita pro optimalizace hledání korespondujících bodů [1].

Výpočet 3D bodů

Při rekonstrukci bodů ve scéně vycházím z principu pasivní triangulace, viz kapitola níže. Za pomoci kalibrace kamery zjistím vnější a vnitřní parametry, a proto nebude těžké provést závěrečný výpočet.

$$x = x_l \frac{2d}{x_l - x_p} \qquad [11]$$

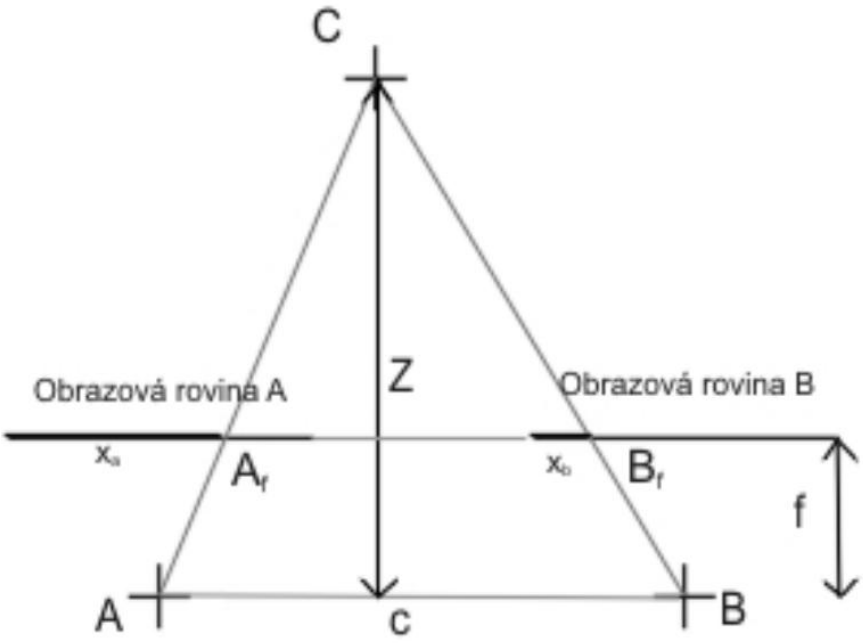
$$y = y_{lp} \frac{2d}{x_l - x_p} \qquad [12]$$

$$z = \frac{2df}{x_l - x_p} \qquad [13]$$

Kde : x_l je horizontální souřadnice levého snímku,
x_p je horizontální souřadnice pravého snímku,
y_{lp} je vertikální hodnota snímků, tato hodnota je stejná na obou snímcích,
2d je vzdálenost optických os,
f je ohnisková vzdálenost.

2.9 Triangulace

Jedna z metod pro zjišťování souřadnic snímaného bodu je triangulace. Před jejím použitím musíme znát vzdálenost kamer, ohniskovou vzdálenost obou kamer a souřadnice daného bodu na obou obrazech.



Obrázek 16: Triangulace [1].

Na obrázku č. 16 můžeme vidět body A, B - body projekce a dále obrazovou rovinu, která je ve vzdálenosti f. Souřadnice X_a X_b jsou souřadnice projekcí bodu C na obraze jednotlivých kamer. Body A_f a B_f jsou body paprsku, tedy průsečík paprsku s obrazovými rovinami. Vzdálenost kamer je strana c tohoto trojúhelníku. Délku úsečky A_f B_f můžeme vyjádřit pomocí c-(x_a-x_b), kde (x_a-x_b) se nazývá paralaxa.
Za použití těchto znalostí sestavíme rovnici:

$$\frac{c(x_a - x_b)}{Z - f} = \frac{c}{Z} \qquad [14]$$

Z této rovnice vyjádříme Z – což je vzdálenost snímaného objektu od kamery a dostáváme rovnici:

$$Z = \frac{cf}{x_a - x_b} \qquad [15]$$

2.10 Paralelní výpočet

V minulosti se výpočetní výkon zařízení zvyšoval taktovací frekvencí procesorů. To však s sebou nese i negativní stránky, a tou je například produkce ztrátového tepla v procesoru.

V dnešní době se pro zvýšení výkonu zařízení používá více exekučních jader, což vede k paralelnímu výpočtu. Paralelní výpočet však nemusí přinést očekávané zvýšení výpočetního výkonu. Ne na každý výpočet je vhodný. Předpokládám však, že při práci se snímky, kde potřebuji porovnávat velké množství pixelů, přinese značné zrychlení.

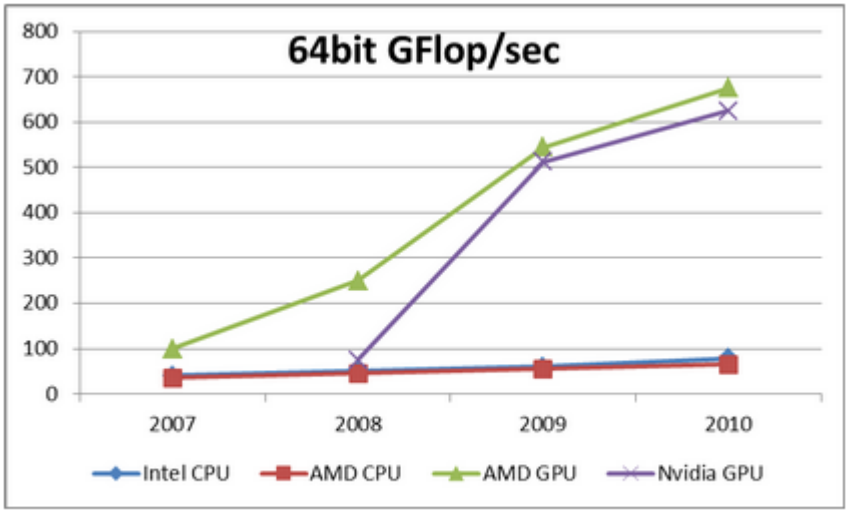
Je více standardů paralelního výpočtu. V diplomové práci se budu zabývat pouze architekturou CUDA od firmy NVIDIA - podle zadání.

2.11 Nvidia CUDA

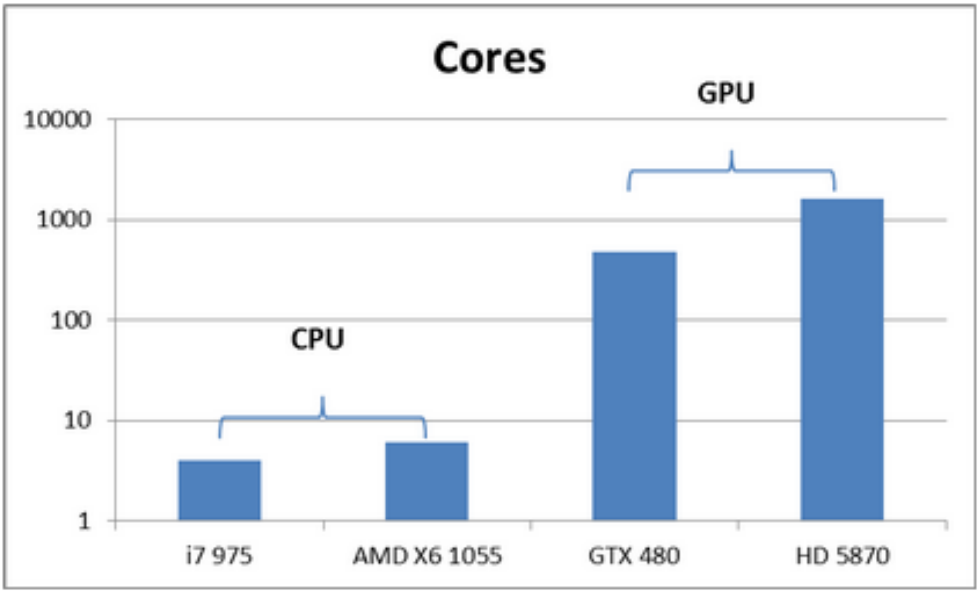
CUDA (Compute Unified Device Architecture) je softwarová a hardwarová architektura, která nám umožňuje na GPU spouštět programy napsané v jazycích CUDA C.

Architekturu CUDA balíků můžeme rozdělit na programovou a hardwarovou. Jádro grafického čipu je postaveno okolo pole stream multiprocessor. Tyto multiprocesory se skládají z několika jednodušších procesorů, CUDA jader. Počet těchto jader je dnes základním výkonnostním faktorem spolu s jejich frekvencí. Z programového hlediska je základní jednotkou vlákno. Na grafické kartě v současné době může běžet desetitisíce vláken. Je tedy zřejmé, že se jedná o strukturu, která je zaměřená na paralelní výpočet.

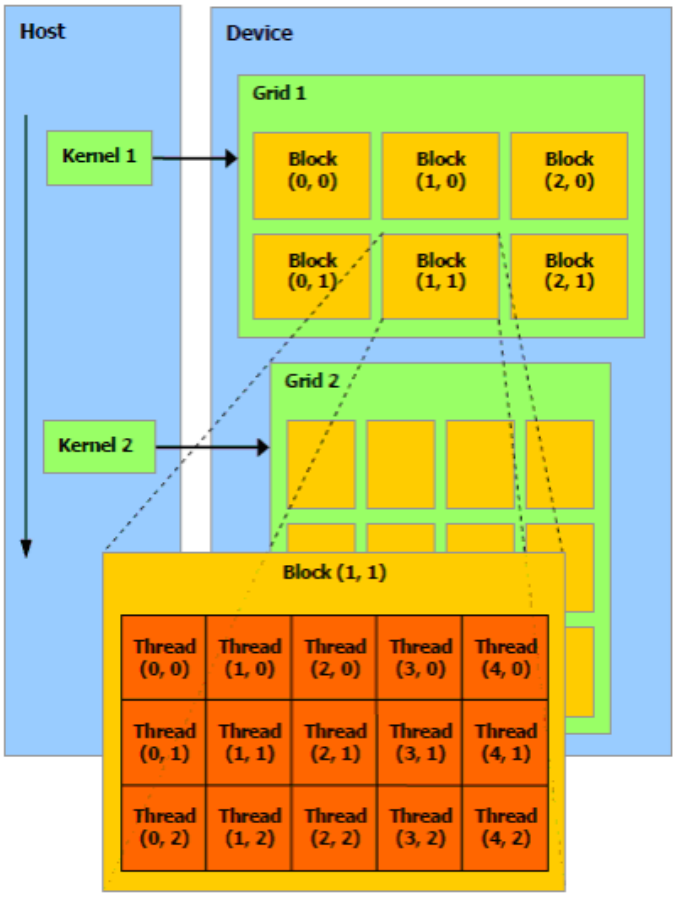
Podmínkou efektivního běhu na GPU, vzhledem k jeho struktuře, je maximální paralelní struktura navrhovaného algoritmu. Pro 3D rekonstrukci je to výborný předpoklad, neboť můžeme využít paralelní výpočet, a dosáhnout tak vyšší rychlosti oproti CPU.



Obrázek 17: Vývoj výpočetního výkonu u GPU a CPU [4].



Obrázek 18: Počet jader u CPU a GPU [4] , na ose y je logaritmické měřítko.

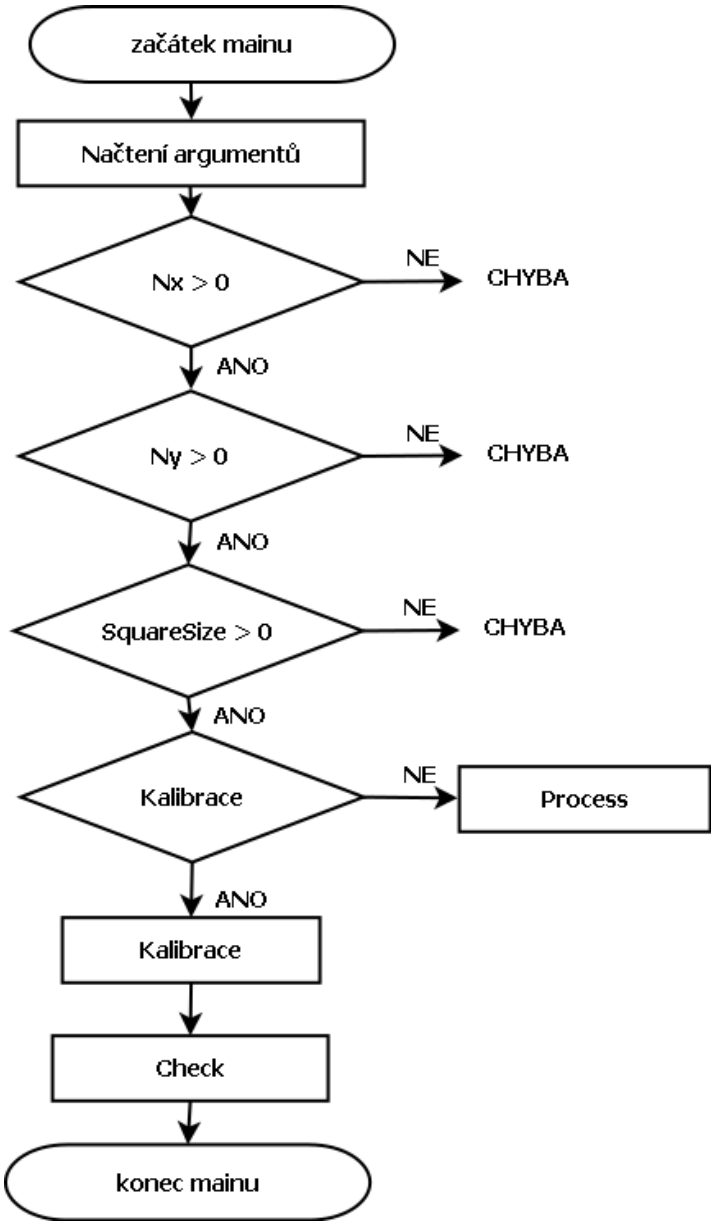


Obrázek 19: Model Standardu CUDA [14].

3 PRAKTICKÁ ČÁST

V této kapitole shrnu výsledky mé práce a postupy, které budou vysvětleny. Nejprve uvedu vývojové diagramy jednotlivých funkcí, pro lepší pochopení nastávající problematiky.

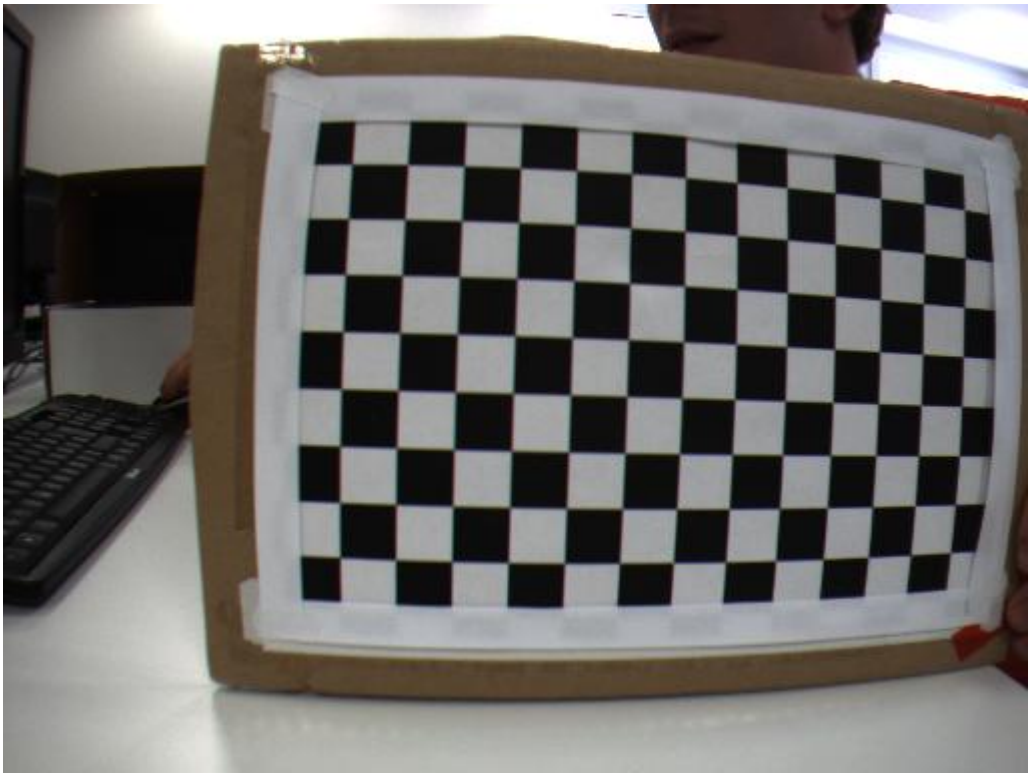
3.1 Funkce programu



Obrázek 20: Vývojový diagram funkce main.

Na předešlém obrázku je znázorněna funkce main pomocí vývojového diagramu. Ta nám slouží pro načtení parametrů, které se zadávají jako COMMAND ARGUMENTS. Prvním argumentem je seznam obrázků, v mé práci je to „list.txt“, které slouží ke kalibraci kamery.

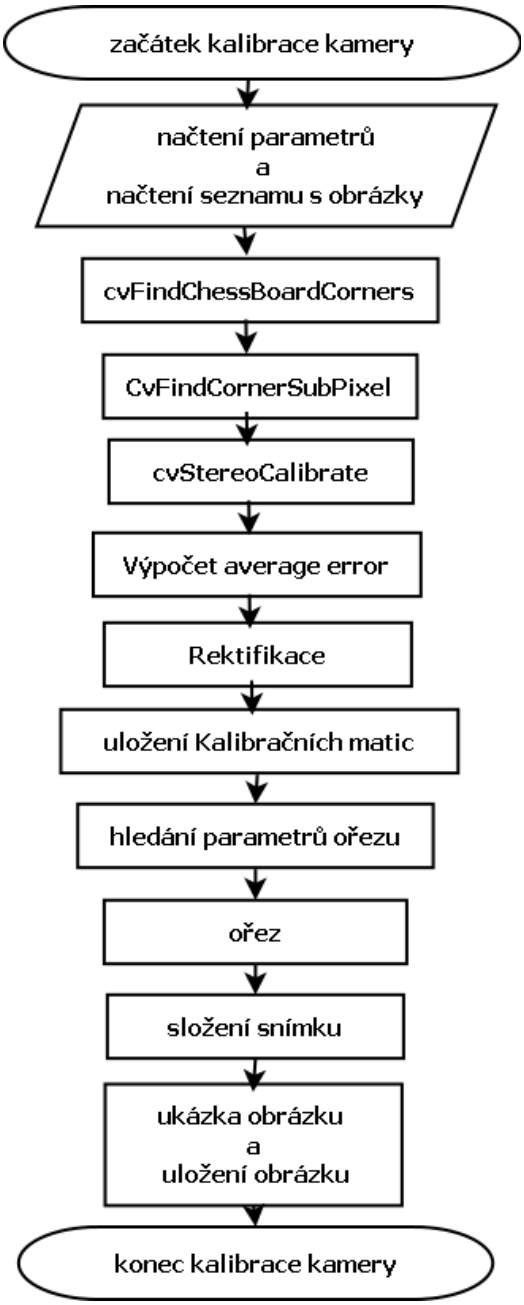
Druhý argument vyjadřuje počet přechodů v šachovnici z bílého čtverce na černý a obráceně. Ve svislém směru je to 8 přechodů, ve vodorovném směru 13 přechodů. Tyto parametry jsem odvodil z obrázku č. 21, kde je vidět kalibrační vzor, který jsem použil.



Obrázek 21: Ukázka kalibračního snímku z pravé kamery.

Předposledním parametrem je rozměr šachovnice, respektive rozměr jednoho čtverečku v cm. Do diplomové práce jsem použil šachovnici o rozměrech jednoho čtverce 2cm.

Poslední parametr nám udává, zdali budeme chtít prvotní kalibraci kamery nebo funkci process. Funkce process nám počítá 3D rekonstrukci, avšak ta potřebuje hodnoty uložené při kalibraci. Jestliže se poslední parametr rovná 0, je spuštěna kalibrace. Pokud je udaná jiná hodnota, spustí se funkce process. Ve vývojovém diagramu jsou vidět podmínky Nx, Ny, SquareSize > 0. Je to kontrola, zdali jsme tyto parametry, které jsou nutné pro kalibraci, zadali.



Obrázek 22: Vývojový diagram kalibrace kamery.

Princip kalibrace v OpenCV je podrobně popsán v [13] a je implementován do funkce cvCalibrateCamera() knihovny OpenCV, která je pro kalibraci použita. Pomocí této funkce zjistíme vnitřní i vnější parametry, fundamentální matici atd. Další mnou použité funkce v knihovně OpenCV jsou:

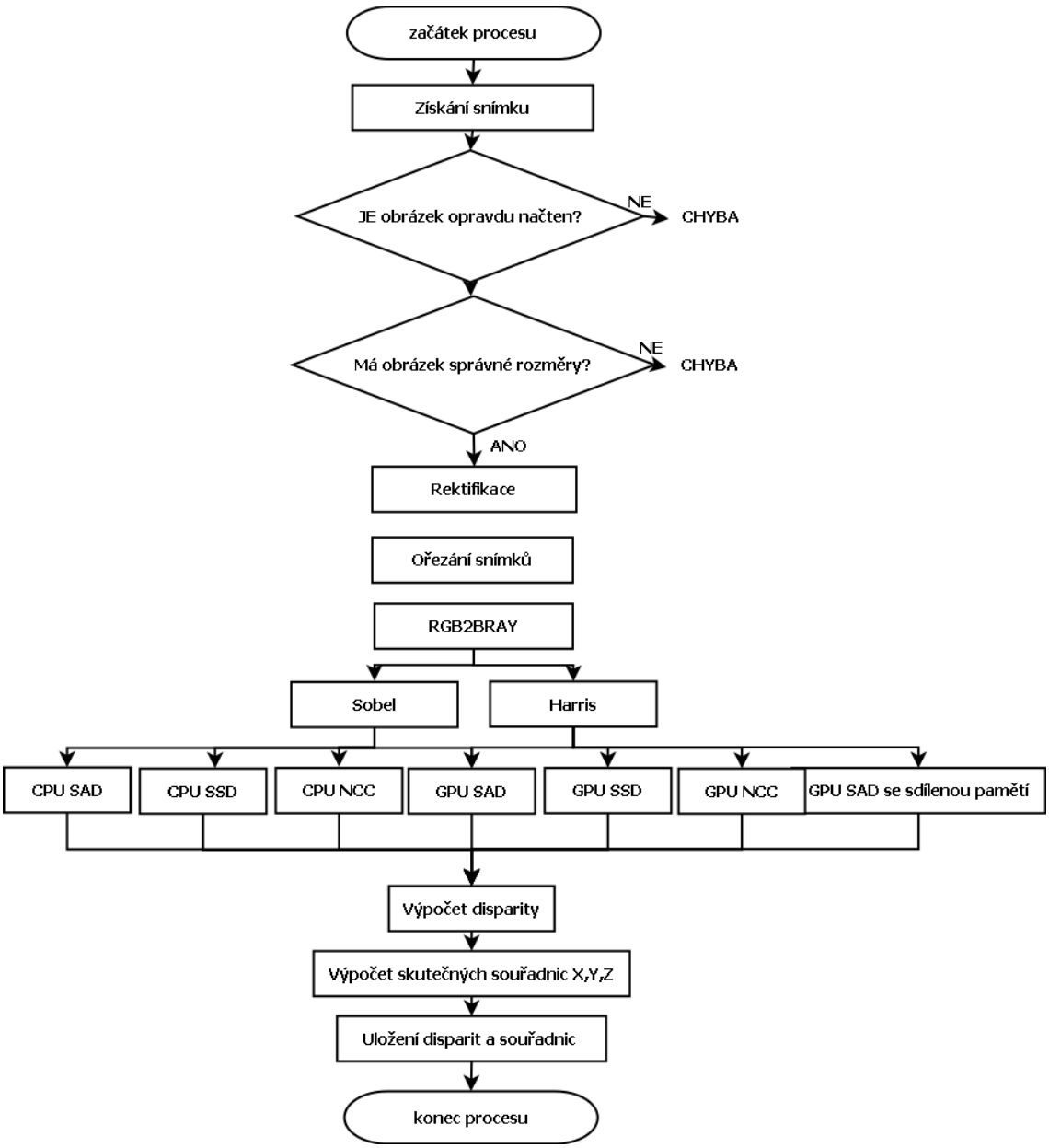
cvUndistortPoints() - ta transformuje souřadnice z originálních souřadnic do nezkreslených souřadnic,

cvComputeCorrespondEpilines() - počítá pro seznam bodů jednoho obrázku epipolární linii v druhém obrázku,

cvStereoRectify() - upraví snímky tak, že mají epipolární linie rovnoběžné,

cvInitUndistortRectifyMap() - počítá mapu, podle které lze přemapovat pixely pořízeného obrazu tak, že dojde jak k eliminaci zkreslení (distortion), tak i k rektifikaci obrazů z levé a pravé kamery (řádkové sesouhlasení),
cvRemap() - zmíněná funkce cvInitUndistortRectifyMap() nám vrací mapu, podle které je původní snímek přemapován, přetransformován tak, aby nebyl geometricky zdeformován.

Aby výsledný vývojový diagram nebyl příliš dlouhý a stal se přehlednějším, zařadil jsem pod rektifikaci hledání epipoláry, rektifikaci, transformaci souřadnic a přemapování. Jakmile proběhne kalibrace, uloží se kalibrační matice pro další práci.

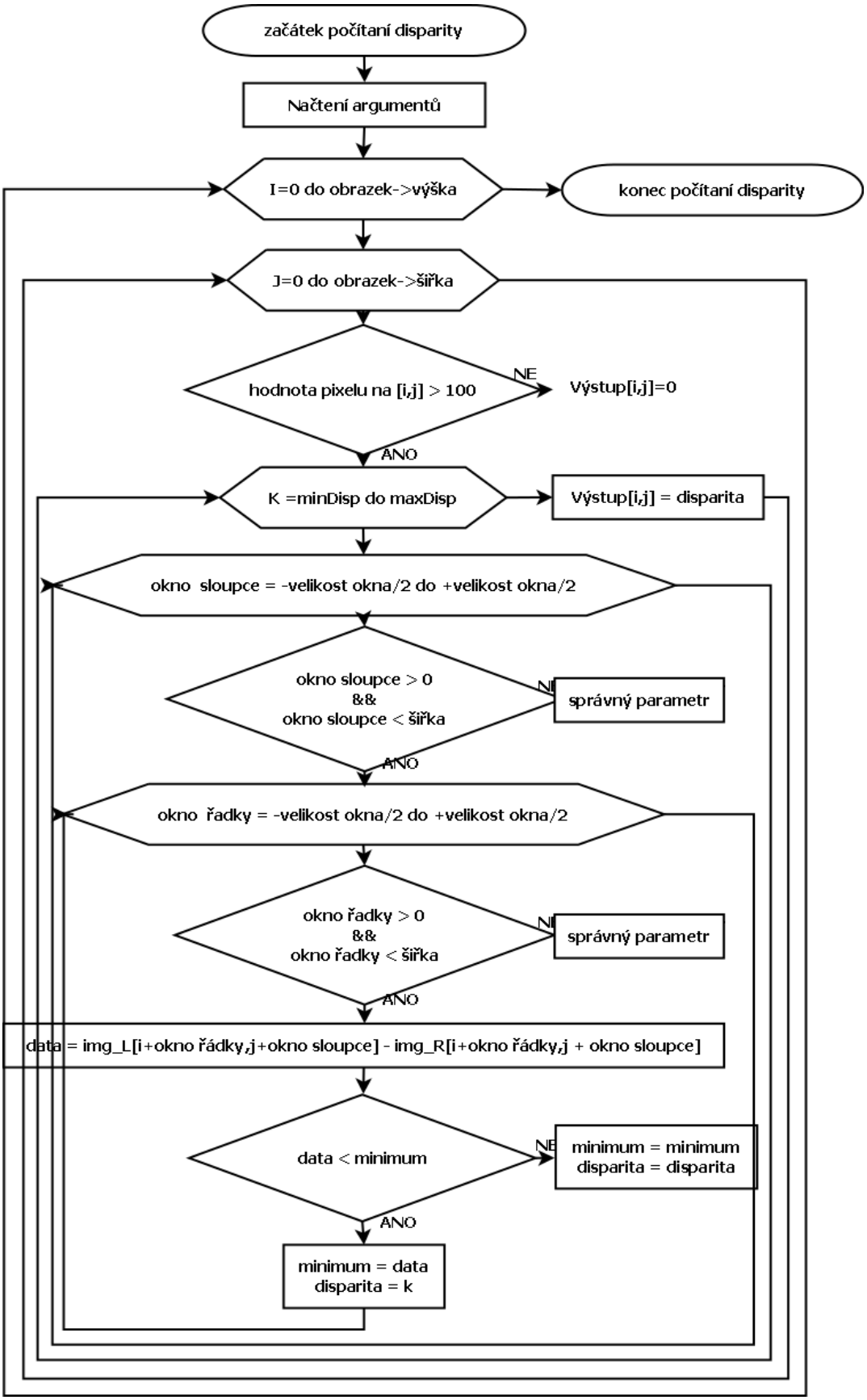


Obrázek 23: Vývojový diagram funkce process.

Ve funkci process načítám jednotlivé snímky (z pravé a levé kamery), kde chci počítat 3D rekonstrukci. Probíhá zde kontrola, jestli obrázek existuje a správně se načetl. Další podmínkou jsou správné rozměry, neboť při načítání kalibračních matic a parametrů jsou tyto parametry vázány na rozměr. Po správném načtení obrázků o správných rozměrech přejdeme k přemapování vstupních obrázků, z nichž jsou výsledkem rektifikované obrázky.

Zde ořežeme místa, která nenesou informaci. Vstupním parametrem Harrisova a Sobelova detektoru je šedotónový obrázek, proto je zde převod z RGB do GRAY.

Tyto detektory používám jako detektory významných bodů. Po zvolení jednoho z detektorů lze vybrat jednu z možných variant počítání korespondence nebo použít všechny a porovnat. Po zvolení metody se počítá disparita. Následně se ze známých kalibračních matic a vypočítané disparity spočítá souřadnicový systém objektu. Vše se uloží pro následné použití.



Obrázek 24: Vývojový diagram pro počítání disparity na CPU.

Na předešlém obrázku je vidět složitost výpočtu disparitní mapy. Algoritmus funguje na principu procházení pixelu po pixelu a jeho okolí v levém snímku se porovnává s daným pixelem a jeho okolím v pravém snímku. Pixely pro hledání korespondence z pravého snímku jsou omezeny od minimální disparity do maximální disparity. Porovnávané pixely však leží na jedné přímce díky předešlé rektifikaci.

V algoritmu je i zahrnut práh hodnoty vstupního obrázku, který nám urychluje algoritmus. Tento práh je nastaven dle použité metody vyhledávání významných bodů.

3.2 Kalibrace

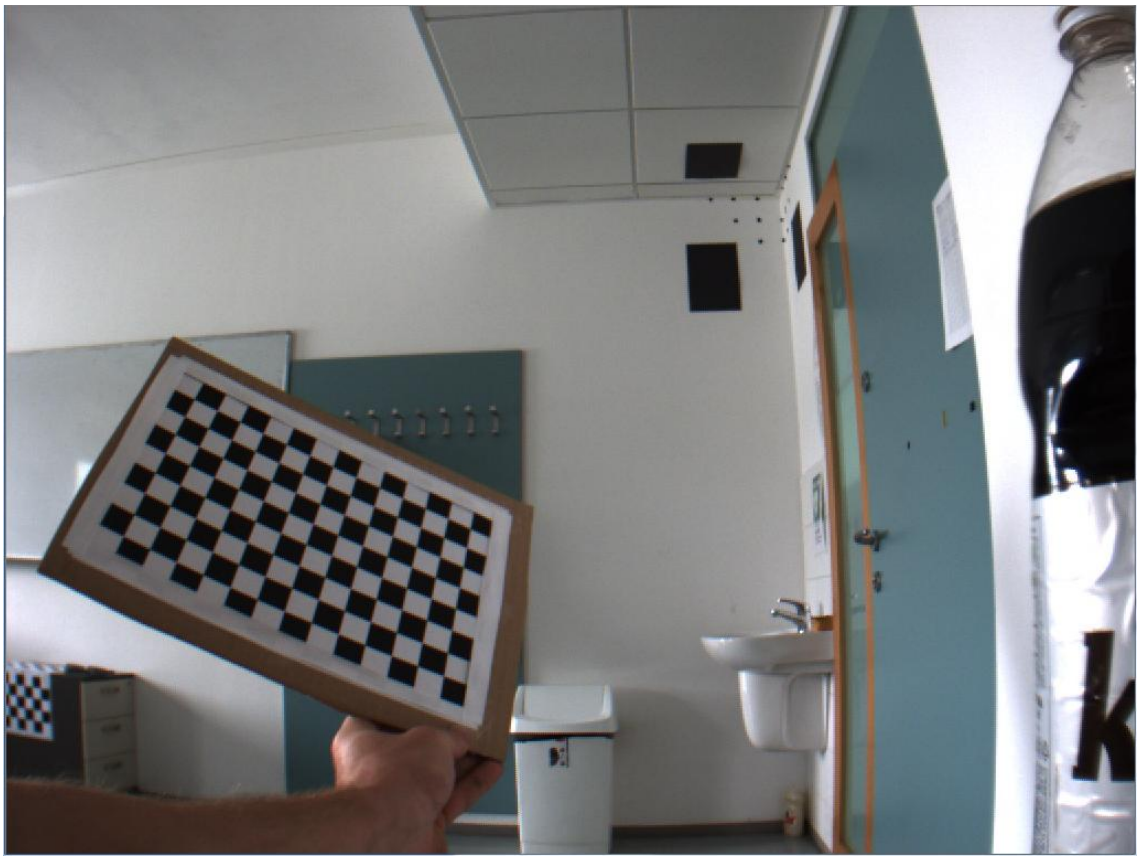
Kalibraci aparátu jsem prováděl tak, že jsem stereokamerou nasnímal šachovnicový vzor z různých úhlů a z různých vzdáleností. Je nutné, aby šachovnice byla vidět celá. Doporučuje se, aby šachovnice nebyla pouze uprostřed snímku, protože zkreslení je největší na jeho okraji. Kalibraci jsem musel provádět vícekrát, jelikož u první kalibrace jsem nedodržel doporučení, kterým je, aby snímaný šachovnicový vzor zabíral co možná největší plochu snímku.

Chyba kalibrace vycházela většinou přes 3pixely^2, což byla ve výsledku při rekonstrukci odchylka měřeného objektu (45x32cm) více než 15cm při vzdálenosti okolo 1 metru, což představuje více než 15% chybu. Po nasnímání nových vzorů jsem dosáhl chyby 0.3pixely^2, což ve výsledku nebyla chyba větší jak +-1cm při vzdálenosti okolo 1 metru. Při kalibraci jsem se inspiroval ze zdroje [13]. Počet snímků, ze kterých se počítaly kalibrační matice, bylo 14. Ukázka těchto snímků je v příloze, viz kapitola 8.3.

3.3 Rektifikace

Nejprve jsem použil funkci cvInitUndistortMap(), která počítá mapu deformace obrázku, a posléze pomocí této mapy a funkce cvRemap() jsem tuto deformaci odstranil. Vstupní obrázek můžeme vidět na obrázku č. 25. Pomocí funkce cvRemap získáme snímek, viz obrázek č. 26. Takto získaný obrázek je potřeba ořezat.

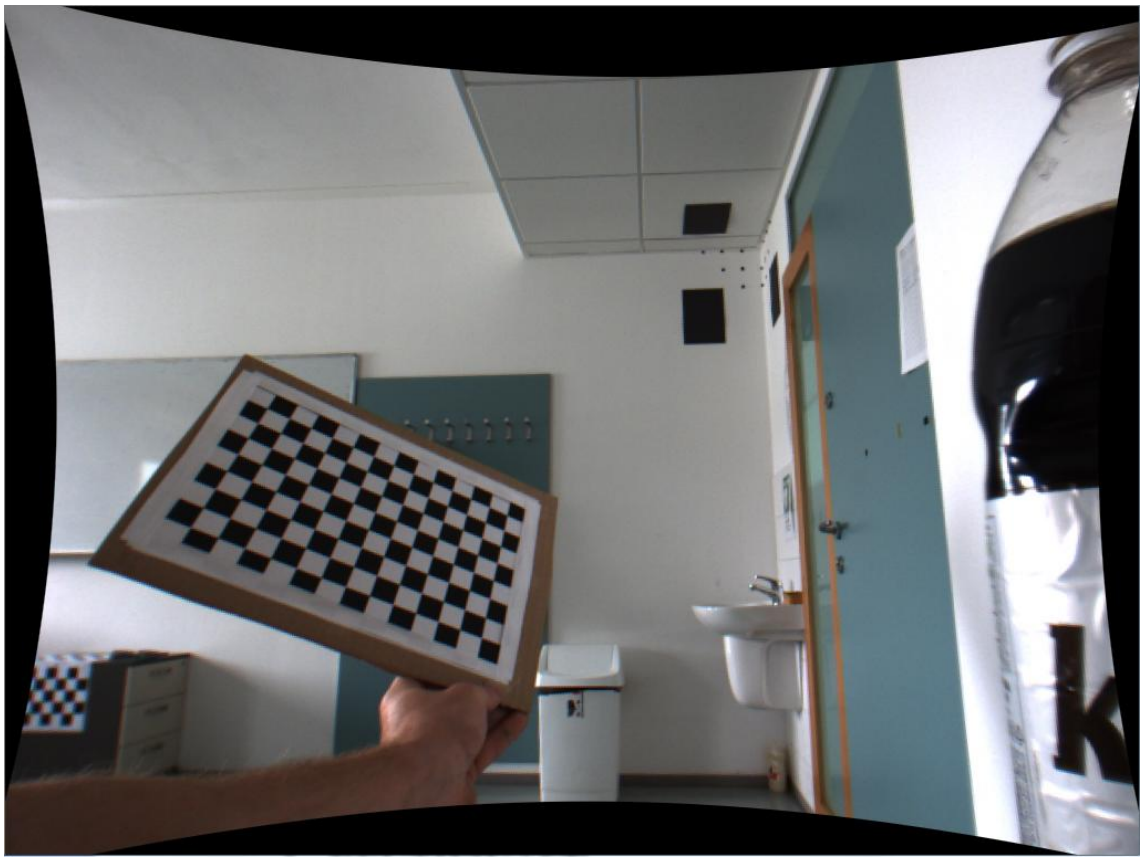
Zde musím zjistit parametry ořezu snímku z obou kamer. Abych zvolil správné parametry, použiji bílý obrázek, na který použiji funkci cvRemap(). Je nutné, aby měl tento obrázek stejné rozměry jako obrázek scény, který jsem získal z kamery. Touto operací jsem docílil toho, že pro každou kameru mám upravený bílý snímek podle její deformace. Poté jsem našel minima a maxima ořezu a originální snímky z levé a pravé kamery jsem ořezal se stejnými parametry ořezu. Ořezaný snímek je zobrazen na obrázku č. 28.



Obrázek 25: Vyjmutý obrázek z jedné kamery ze snímku ze stereokamery(zkreslený).

Na obrázku č. 25 je vyobrazen snímek, kde je zřetelně vidět geometrické zkreslení kamery. Snímek je pořízen z jedné kamery a to z levé, u pravé kamery může být toto zkreslení jiné.

Zkreslení je nejlépe pozorovatelné na dveřích v pravé části snímku.



Obrázek 26: Zrektifikovaný snímek.

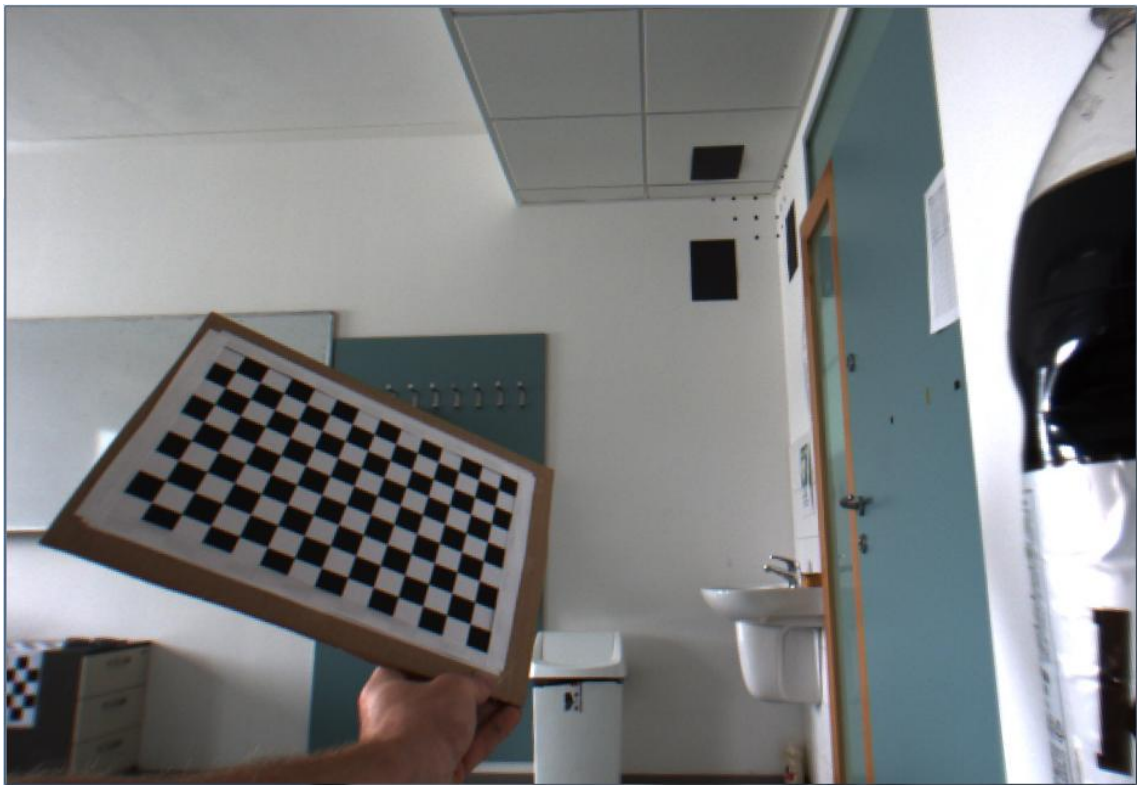
Obrázek č. 26 je zrektifikovaný snímek ze zkresleného předešlého snímku. Hlavní úlohu rektifikace však znázorňuje obrázek č. 29, kde vidíme jednak odstranění geometrického zkreslení, ale hlavně sesouhlasení řádků u obou snímků. Díky této skutečnosti je hledání korespondujících bodů snazší, neboť korespondující body musí ležet na stejném řádku.

Následující obrázek č. 27 ukazuje hodnoty potřebného ořezu u pravého a levého snímku. Z těchto hodnot jsem vybral ty, které splní podmínku ořezů u obou snímků.

Příklad uvedu u parametru ořezu UP. U levého snímku je hodnota rovna 32 a u pravého 24. Tedy ořez musím provést 32 pixelů zeshora, neboť tuto hodnotu zahrnuje 24 pixelů (z pravého snímku). Poté jsem si nechal vykreslit levý a pravý snímek, abych ověřil funkčnost a správnost úvahy, viz obrázek č. 29.

```
up1: 32   down1: 361  left1: 21   right1: 490
up2: 24   down2: 353  left2: 22   right2: 488
```

Obrázek 27: Hodnoty minima a maxima ořezu z obou snímků.



Obrázek 28: Zrektifikovaný a ořezaný obrázek, detail odstranění geometrických vad.

Obrázek 29: Ukázka zrektifikovaného obrázku, kde došlo k řádkovému sesouhlasení.

Výsledná rektifikace pravého a levého snímku je zobrazena, viz obrázek č. 29. Zde je odstraněno geometrické zkreslení a došlo k sesouhlasení řádků u pravého a levého snímku.

3.4 Korespondence

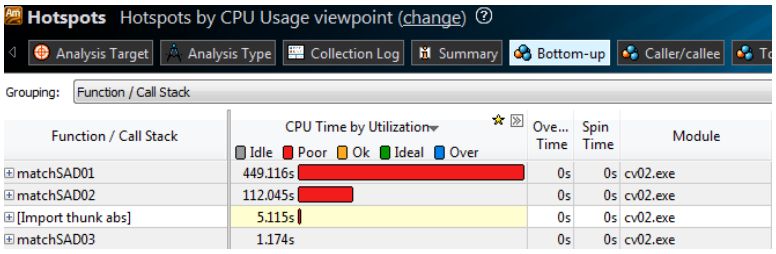
Hledání korespondence je jeden z větších problémů pro 3D rekonstrukci. Jednak se jedná o výpočetně náročnou metodu a dále pak o problematiku spojenou s prohlášením korespondence za pravdivou.

Během mé diplomové práce bylo vyzkoušeno více metod. Jedná se o metody zmíněné v teoretické části mé diplomové práce: metoda SAD, SSD, NCC. Nejprve jsem si implementoval svoji funkci SAD, která počítala podle vzorce [8]. Tato metoda bez jakékoliv optimalizace, tedy bez zohlednění rektifikace (hledání +/- 2 řádky od aktuálního pixelu v levém snímku) a bez omezení maximální a minimální disparity a prahu trvala na zařízení s komponenty, které je v příloze, viz hardware zařízení č. 1, skoro 7.5 minuty pro jediný obrázek.

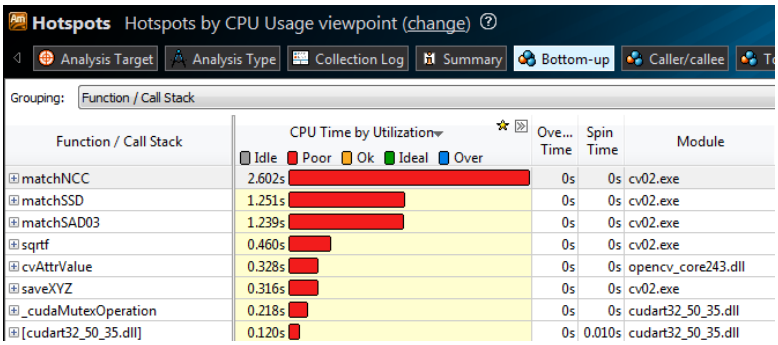
Poté jsem přešel k optimalizaci pomocí zohlednění výsledku rektifikace. Na stejném zařízení se blížil čas výpočtu ke 2 minutám. Poslední optimalizací bylo omezení hledání disparity minimem a maximem a nastavení prahu vstupního obrázku, kdy nad určitý práh se dále nepočítá a výsledný práh na daných souřadnicích je roven 0. Po těchto optimalizacích trval výpočet: 1,2s.

Výsledky jsou vidět na obrázku č. 30, kde funkce matchSAD01 byla bez optimalizací a metoda SAD03 s optimalizací. Na obrázku č. 31 je vidět výpočetní náročnost metod SAD, SSD. Výsledkem tohoto měření je fakt, že právě tato část je pro dosažení real-time odezvy nutná pro GPU-akceleraci. Ta by měla zajistit rychlejší výpočet.

Právě tento druh algoritmu podle literatury vyhovuje použití paralelního výpočtu. Je to zřejmé už z vývojového diagramu, viz obrázek č. 24. Zde je vidět, kolik cyklů je nutné projít pro prohledání okolí daného pixelu a jeho rozdílu s pixelem z druhého snímku. Proto se nadále moje práce zabývá akcelerací výpočtu hledání korespondencí. Opět postupuji od nejlehčí metody, a tou je metoda SAD a postupně přidám i metody SSD a NCC.



Obrázek 30: Výpočetní náročnost operací SAD bez optimalizací a po optimalizaci.



Obrázek 31: Výpočetní náročnost jednotlivých metod hledání korespondence.

3.5 GPU akcelerace

Architektura CUDY je popsána v teoretické části. Zde popíši, jak se s ní pracuje. Základem každé CUDA aplikace je tzv. kernel. Jde o funkci, kterou definujeme pomocí klíčového slova __global__. Toto slovo nám říká, že tato funkce bude spouštěna na GPU-device a ne na CPU-host. Kernel (funkce) je spuštěna tolikrát, kolik je jednotlivých vláken (threads). Každé toto vlákno zpracovává instrukce v kernelu. Každé vlákno má své ID. ID vlákna se dá zjistit pomocí threadIdx.x, tím zjistíme pouze jednu složku. ThreadIdx je typu dim3, což říká, že je 3-dimenzionální (x,y,z).

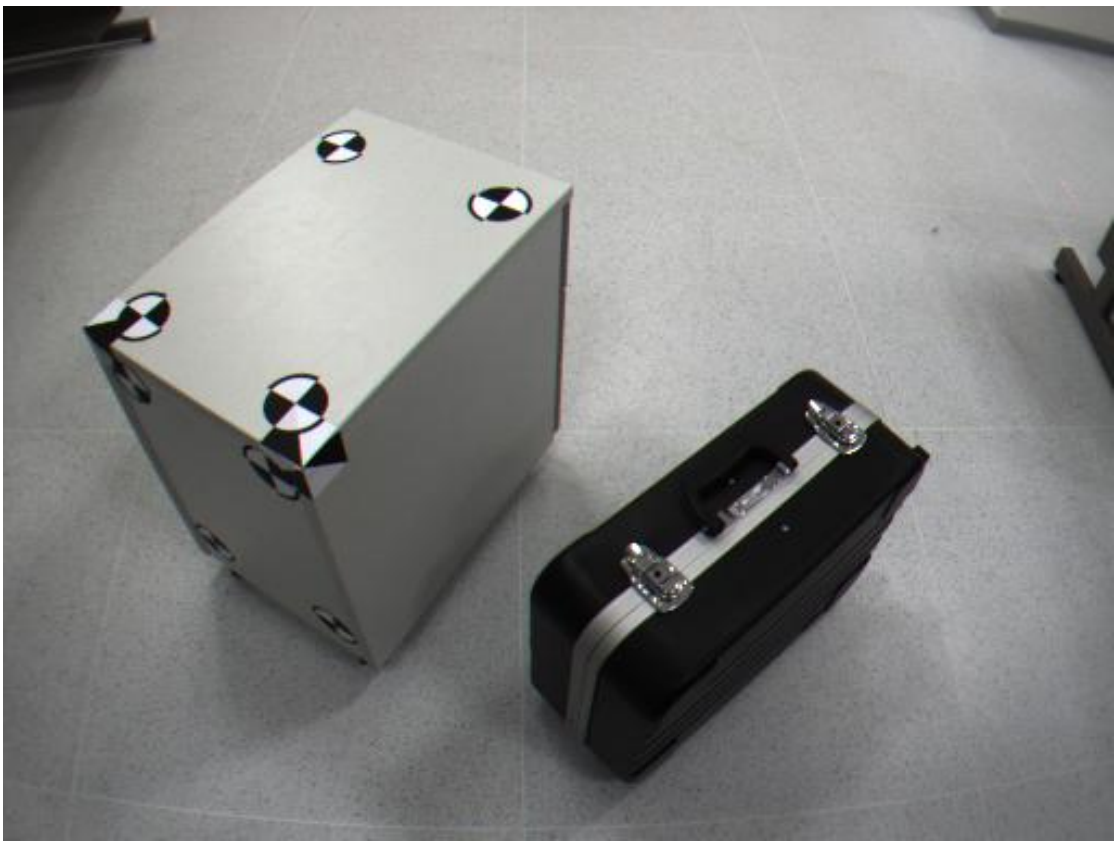
V mé práci používám pouze jednu dimenzi. Kernel se volá pomocí ostrých závorek <<<počet bloků,počet vláken>>>, je to speciální syntaxe CUDA programů. K lepšímu pochopní je vhodný obrázek č. 19.

Sám o sobě paralelní program přináší obrovské zrychlení, avšak umožňuje i další urychlení. Jelikož data, se kterými pracuje, se nacházejí na globální paměti. Globální paměť je výrazně pomalejší jako sdílená paměť neboli shared memory. Sdílená paměť je přístupná pouze jednomu bloku, avšak všem vláknům toho bloku.

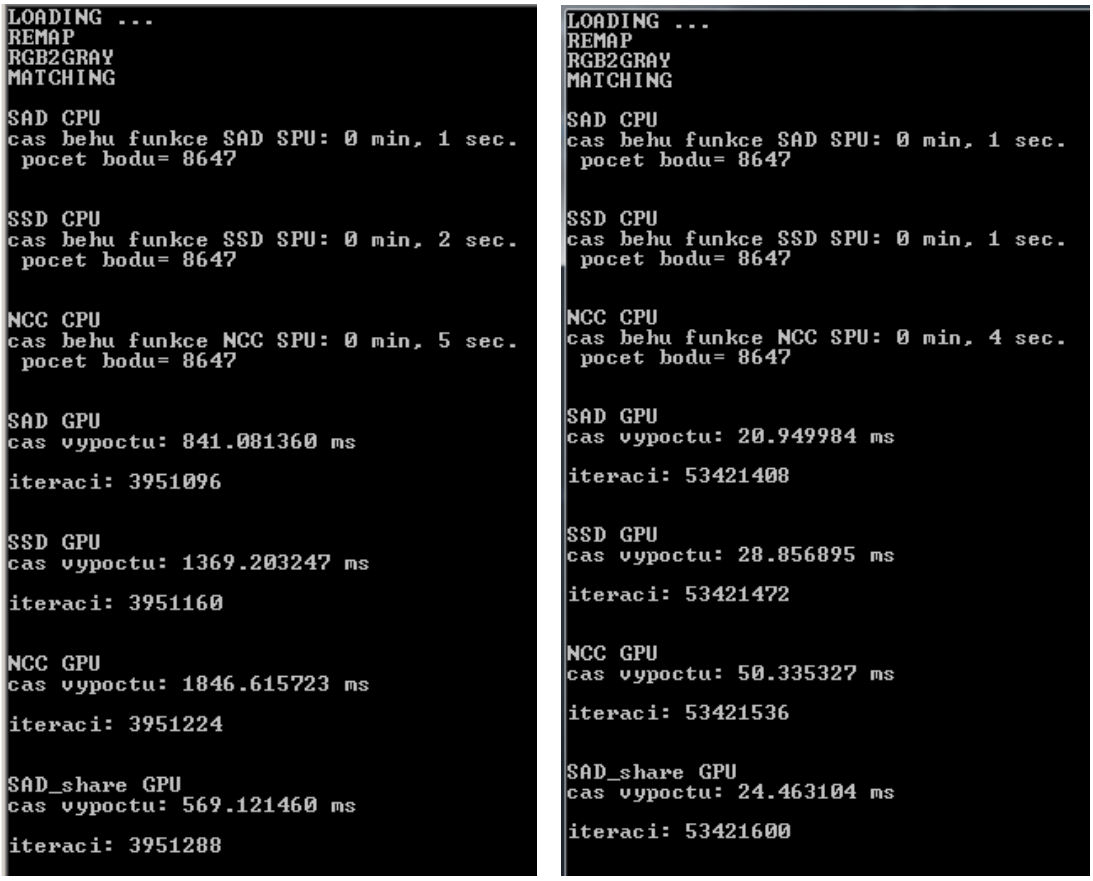
Hledání korespondencí jsem implementoval tak, že každé vlákno počítá rozdíl daného pixelu (je zde brán ohled na rektifikaci a minimální a maximální disparitu) a jeho okolí vůči všem možným pixelům a jejich okolí v pravém snímku. Po úspěšné implementaci jsem dosáhl času kolem 20ms u metody SAD. Metoda SSD je pomalejší, blíží se k 30ms a metoda NCC je kolem 50ms.

Při implementaci metody SAD mohu konstatovat, že tato rychlost je dostatečná pro dosažení real-time odezvy na hardware zařízení č. 1. Avšak u implementace není využita sdílená paměť, která by měla tento výpočet ještě urychlit. Jak již bylo zmíněno, tak sdílená paměť je společná pro jeden blok i pro všechna vlákna toho bloku, tedy jsem pro lepší práci využil těchto poznatků. Do jednoho bloku jsem přiřadil počet vláken stejný jako je šířka obrázku, avšak musí být menší než povolený počet vláken na blok. Následně jsem zkopíroval data z globální paměti do sdílené paměti, pak už jsem jen pracoval s daty ve sdílené paměti. Každé vlákno kopíruje svoji část dat. Je velmi důležité použít po zkopírování dat funkci __syncthreads(), ta nám zaručuje, že počká na vlákna v bloku, která ještě běží, aby všechna vlákna měla svoji instrukci hotovou.

Zde se však moje diplomová práce rozchází s teorií, neboť práce se sdílenou pamětí by měla být výrazně rychlejší než práce s globální pamětí, avšak na Hardware zařízení č. 1 se výpočetní výkon použitím sdílené paměti nezrychlil. Tento fakt je možné vidět na obrázku č. 33, kdy se jedná o rozdíl 5ms, kde sdílená paměť je pomalejší. Proto jsem algoritmus vyzkoušel i na jiném zařízení, viz hardware zařízení č. 2. Zde je vidět, že práce se sdílenou pamětí urychlila algoritmus a to výrazně, viz obrázek č. 33, jedná se o urychlení o více jak 30%.



Obrázek 32: Vstupní obrázek do kernelu.



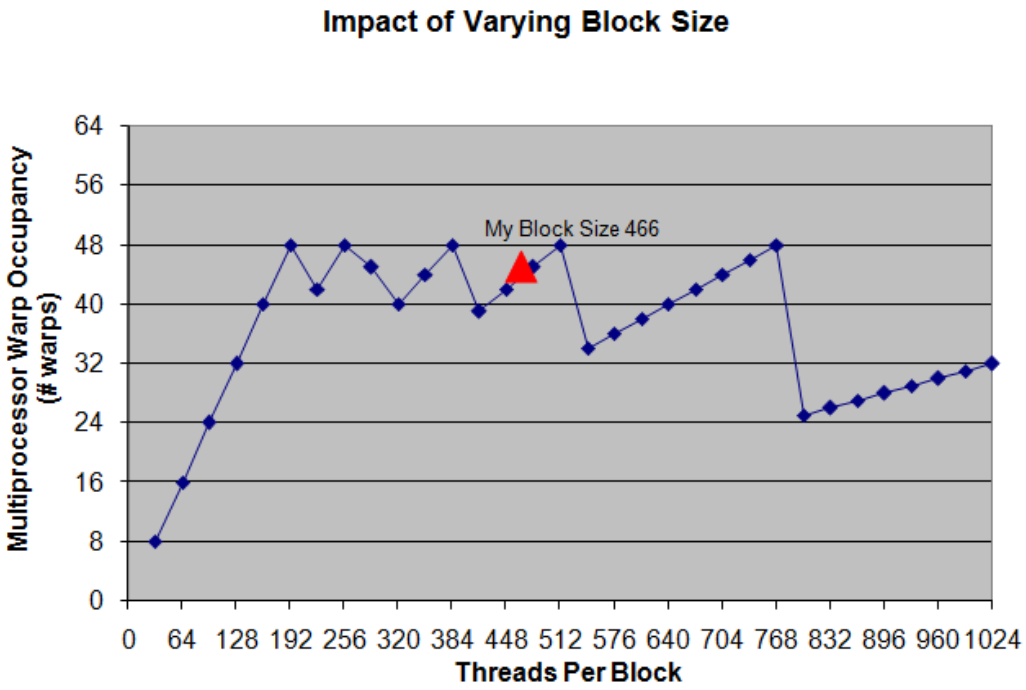
Obrázek 33: Výpočetní náročnost funkcí CPU a GPU na hardwarové zařízení č. 2 vlevo a hardwarové zařízení č. 1 vpravo.

Na obrázku č. 33 je vidět výpočetní náročnost jednotlivých funkcí pro hledání korespondencí. Měření funkcí na CPU není tak přesné jako na GPU proto bylo nutné použít program Intel Parallel Studio XE 2013, který funkce běžící na CPU přesně změřil.

Tabulka 1:Vlastnosti GPU zařízení č.1.

Physical Limits for GPU Compute Capability:	**2,1**
Threads per Warp	32
Warps per Multiprocessor	48
Threads per Multiprocessor	1536
Thread Blocks per Multiprocessor	8
Total # of 32-bit registers per Multiprocessor	32768
Register allocation unit size	128
Register allocation granularity	warp
Registers per Thread	63
Shared Memory per Multiprocessor (bytes)	49152
Shared Memory Allocation unit size	128
Warp allocation granularity	2
Maximum Thread Block Size	1024

Z uvedené tabulky č. 1 mohu vyčíst, že na první zařízení (viz kapitola 8.1) je možné využít na blok celých 1024 vláken. Podle obrázku č. 34 je však tento počet nevyhovující, neboť lepší využitelnost je při počtu 512 či 768.



Obrázek 34: Využitelnost GPU při změně velikosti bloku na zařízení č. 1.

V mé práci je využito 512 vláken, neboť tento počet je možné využít i na zařízení č. 2, viz tabulka č. 2. Tím je podle tabulek 3 a 4 využitelnost 100%.

Tabulka 2: Vlastnosti GPU zařízení č. 2.

Physical Limits for GPU Compute Capability:	**1,2**
Threads per Warp	32
Warps per Multiprocessor	32
Threads per Multiprocessor	1024
Thread Blocks per Multiprocessor	8
Total # of 32-bit registers per Multiprocessor	16384
Register allocation unit size	512
Register allocation granularity	block
Registers per Thread	124
Shared Memory per Multiprocessor (bytes)	16384
Shared Memory Allocation unit size	512
Warp allocation granularity	2
Maximum Thread Block Size	512

Díky CUDA_Occupancy_calculator.xls, který přikládám v příloze, je patrné, že při málo využité sdílené paměti se využitelnost bloků nemění, a tím pádem předpokládám, že ani výkon nenarůstá. Faktem zůstává, že sdílená paměť je výrazně rychlejší než globální paměť. Je však nutné říci, že při práci se sdílenou pamětí se v kernelu objevují další podmínky, for cykly a další prostředky, díky nimž výpočetní náročnost roste.

Během testování práce se sdílenou pamětí se prokázalo, že při opakované práci s hodnotou ze sdílené paměti je rychlejší oproti globální. Tím se potvrzuje, že při malém využití sdílené paměti nepřináší výpočetní přínos.

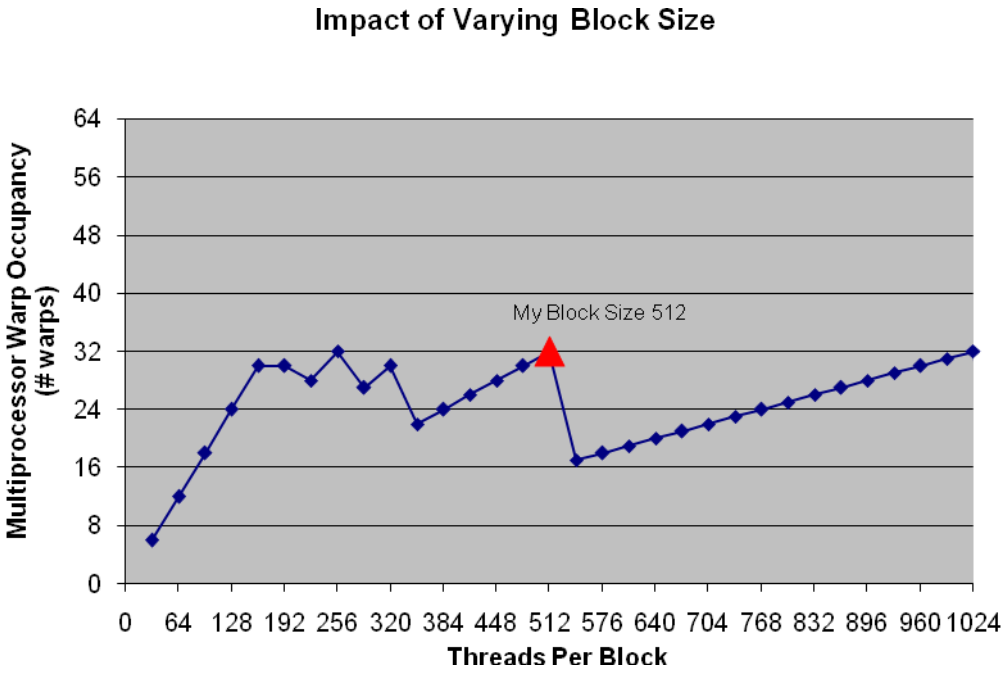
V mé práci se mi nepodařilo tuto paměť využít pro konečné zrychlení algoritmu. Proto je nejlepší dosažená odezva téměř rovna 20ms.

Tabulka 3: Využitelnost jednotlivých bloků na zařízení č. 1.

3.) GPU Occupancy Data is displayed here and in the graphs:	
Active Threads per Multiprocessor	1536
Active Warps per Multiprocessor	48
Active Thread Blocks per Multiprocessor	3
Occupancy of each Multiprocessor	100%

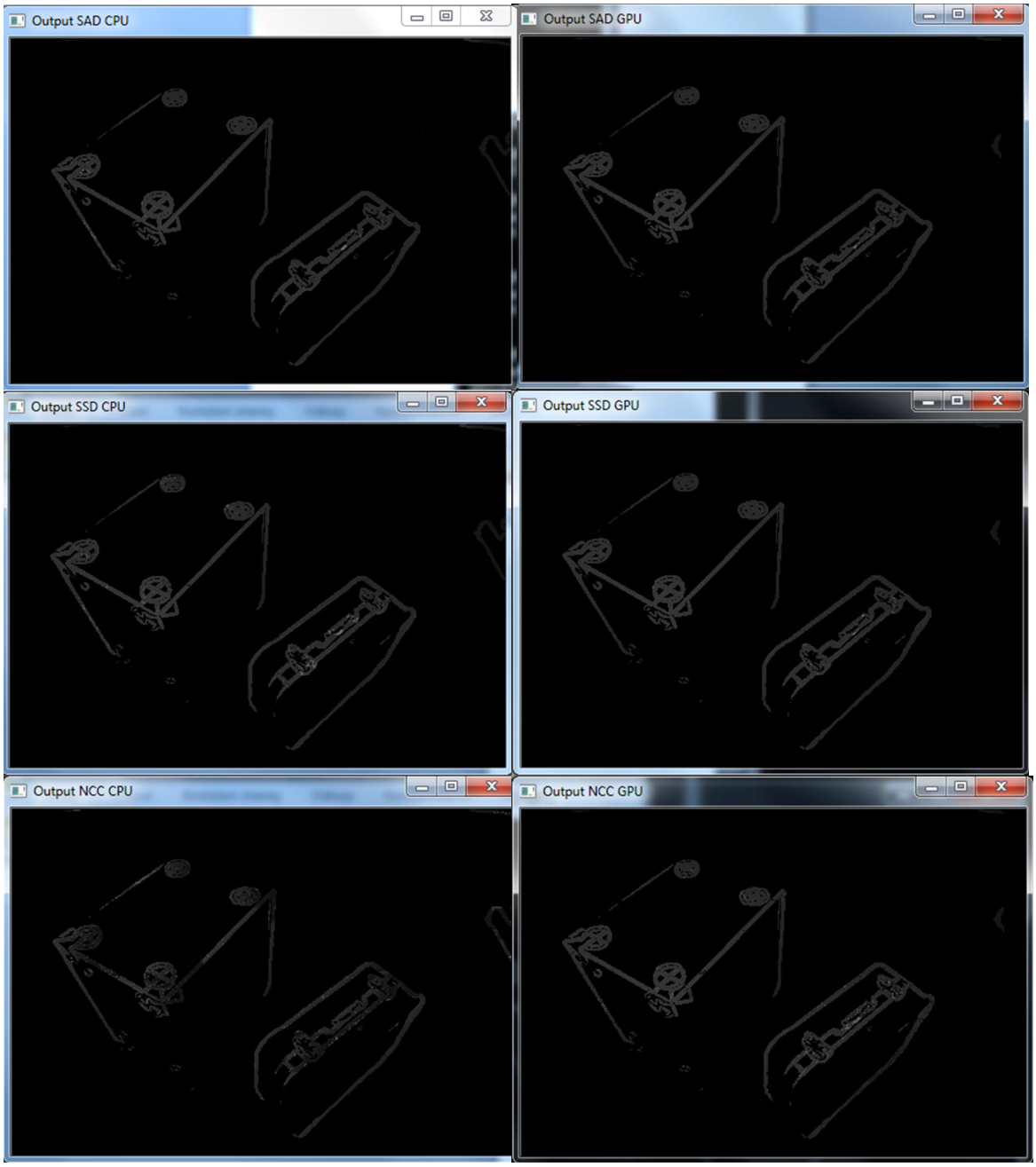
Tabulka 4: Využitelnost jednotlivých bloků na zařízení č. 2.

3.) GPU Occupancy Data is displayed here and in the graphs:	
Active Threads per Multiprocessor	1024
Active Warps per Multiprocessor	32
Active Thread Blocks per Multiprocessor	2
Occupancy of each Multiprocessor	100%



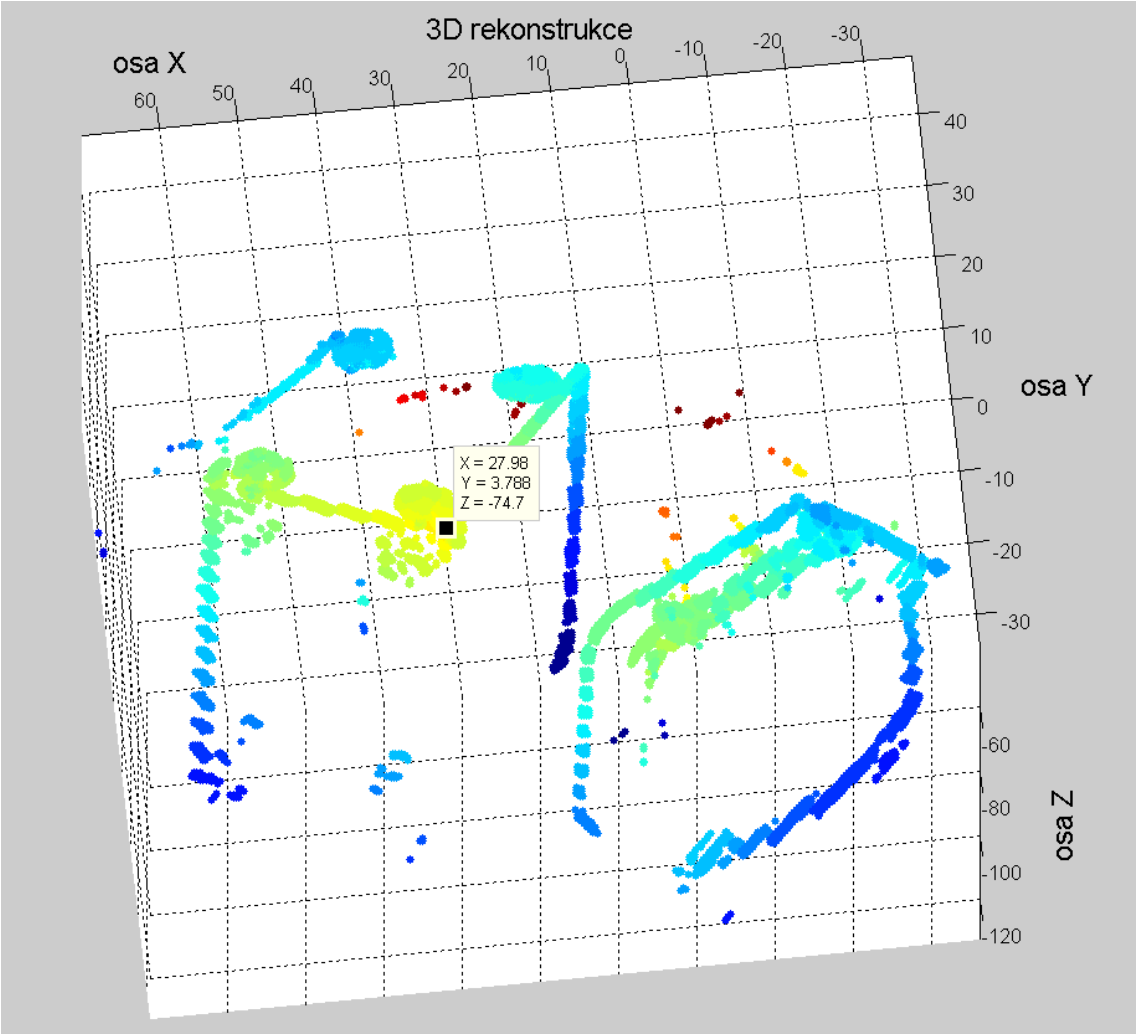
Obrázek 35: Využitelnost GPU při změně velikosti bloku na zařízení č. 2.

U zařízení č. 2 mi sdílená paměť přinesla výpočetní výkon, avšak v porovnání s výpočetní času na zařízení č. 1 je tento výpočetní přínos zanedbatelný.



Obrázek 36: Porovnání metod hledání korespondencí, výsledkem je disparitní mapa.

Na předešlé šestici obrázků jsem vykreslil jednotlivé metody hledání korespondence. Až na malé výjimky se obrázky příliš neliší. Jednotlivé metody se liší ve výpočetní náročnosti.

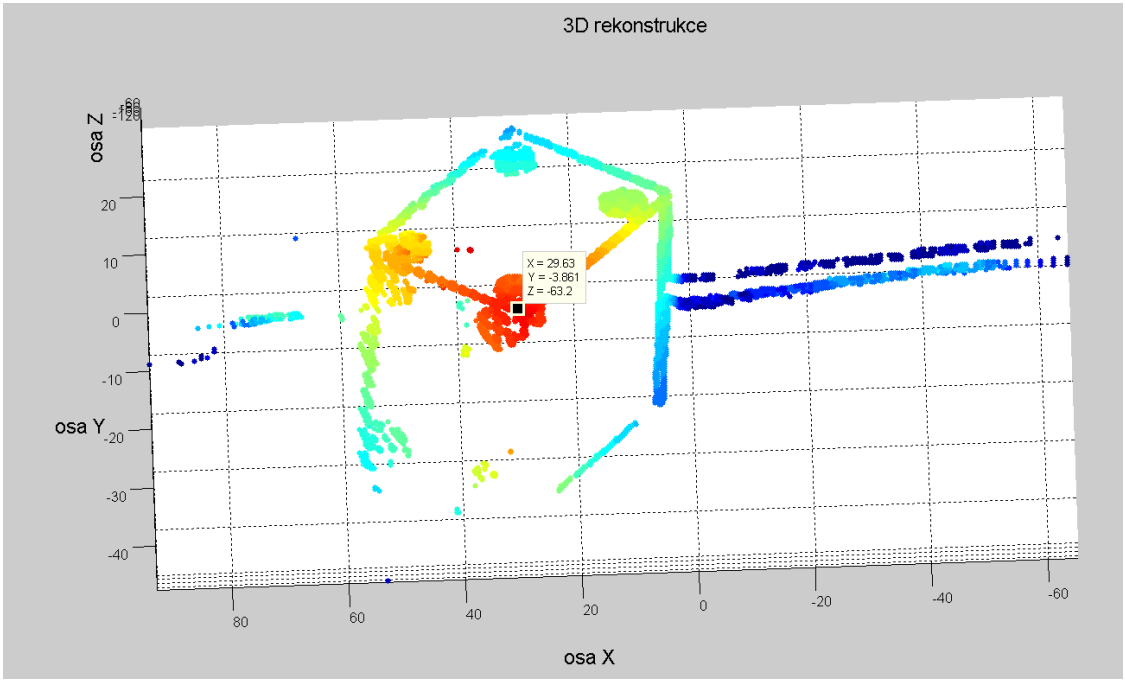


Obrázek 37: Výstupní 3d model scény.

Na obrázku č. 37 je vidět výsledná 3D rekonstrukce. 3D rekonstrukce je prováděna za použití programu MATLAB. Nejprve jsem však musel uložit z Visual Studia data pro zobrazení.

Pro ukládání do textového souboru s příponou txt, slouží funkce saveXYZ. Ta projede celý snímek a ze znalosti vzorců [11], [12], [13] a znalosti kalibrační matice Q, kterou zjistíme při kalibraci, není již problém s vytvořením 3D rekonstrukce. Do Matlabu poté importujeme data, která rozdělíme na tři souřadnice X, Y, Z pomocí příkazu *X=O(:,1); Y=O(:,2); Z=O(:,3),* kde proměnná O znázorňuje importovaná data.

Dalším příkazem je *scatter3(X, Y, Z, 10, Z),* který zobrazí importovaná data. Tato data pak můžeme libovolně otáčet a zjišťovat souřadnice předmětu, jeho vzdálenost atd. Po kontrole pomocí Euklidovské vzdálenosti jsem si vypočetl rozměry „bedny“, kterou jsem měřil. Rozdíl se při delší straně pohyboval okolo 0.5cm, při kratší straně byla odchylka kolem 1cm. Nevýhodou použití Matlabu při 3D rekonstrukci je obtížné otáčení zrekonstruovanou scénou, nastavení pohledu atd.



Obrázek 38: Výstupní 3D model scény č. 2.

4 ZÁVĚR

V mé diplomové práci se zabývám zpracováním stereo snímků. Stereo snímky jsou pořízeny z dvojice kamer, které jsou k sobě pevně uchyceny a jsou synchronizované. Za použití přídavného softwaru *Intel ® Parallel Studio XE 2013* jsem měřil výpočetní náročnost jednotlivých funkcí.

Během diplomové práce jsem se seznámil s problematikou kalibrace. Následně jsem v práci provedl rektifikace. Rektifikace byla velmi důležitá, neboť díky ní se výrazně snížila výpočetní náročnost na CPU, tedy i následně na GPU. Tento rozdíl je možné vidět na obrázku č. 33, výpočet na CPU trval 1,2s a na GPU 0.02s. Bez zohlednění rektifikace se při hledání korespondence (pouze +/- 2řádky) blížil čas výpočtu k 8 minutám. Po rektifikaci se čas snížil na 2 minuty.

Další optimalizace, uskutečněná omezením hledáním disparity a prahem vstupního obrazu, posunula výpočetní čas ještě níž¨, a to cca k 1,2s. Vstupní snímek pro detekci významných bodů podléhá Harrisově detektoru nebo Sobelově masce. I přes veškeré optimalizace se výpočetní čas neblíží real-time odezvě.

Vyhledávání korespondujících bodů (metoda SAD) bylo nejvíce výpočetně náročné, proto jsem se dále zaměřil na přeprogramování na paralelní výpočet za použití platformy CUDA. Po vytvoření funkčního paralelního běhu dané funkce se mi podařilo dosáhnout rychlosti výpočtu 20ms. Při dosažení této rychlosti a funkčnosti jsem ještě vyzkoušel další metody hledání korespondence, a to metody SSD, NCC. Výsledky těchto metod i jejich výpočetní náročnost je zobrazena na obrázku č. 33 a obrázku č. 36.

Pro dosažení rychlejší odezvy byla vyzkoušena práce se sdílenou pamětí u platformy CUDA. Výsledek nepřinesl urychlení, protože velikost potřebné sdílené paměti byla zanedbatelná a nárůst složitosti programu při použití sdílené paměti byl převyšující než předpokládaný efekt. Je zde velký prostor pro vhodné nastavení běhu kernelu.

Vizualizace 3D rekonstrukce byla vytvořena v programu Matlab viz obrázek č. 37. Při přeměření výsledné zrekonstruované scény se ve vzdálenosti cca 1metr od kamery lišil rozměr měřeného objektu (45x32cm) +/- 1cm.

Navazující práce by se dále mohla zaměřit na paralelní programování, práci se sdílenou pamětí, neboť je to zajímavé odvětví. Dále by bylo vhodné v programu pomocí knihovny OpenGL vytvořit 3D rekonstrukci, abychom se obešli bez Matlabu. Poté již jen celý program upravit pro práci s videem.

Cílem diplomové práce bylo ověření na reálném příkladě využitelnosti paralelního programování na GPU a jeho výpočetního přínosu.

V praxi je využití 3D rekonstrukce velké, ať už jde o měření vzdálenosti objektů, jeho rozměrů či tvarů apod.

5 LITERATURA

[1] HAŠA, Jiří. *Tvorba modelu Jeskyní pomocí stereokamery*. Brno: Vysoké učení technické v Brně, Fakulta Informačních technologií, 2009. 42s. Vedoucí diplomové práce byl Ing. Jaroslav Rozman.

[2] Geometry of image formation [online]. 3.11.2008 [cit. 2012-12-15]. Dostupné z: http://cmp.felk.cvut.cz/cmp/courses/XE33PVR/WS20072008/Lectures/Geometry/pinhole.pdf

[3] BASTL, J. *Online 3D rekonstrukce*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 67s. Vedoucí diplomové práce byl Ing. Petr Petyovský

[4] Graphics Processing Unit [online].2008 [cit. 2012-12-15]. Dostupné z: http://www.conficore.com/index.php?option=com_content&view=article&id=48&Itemid=61)

[5] AMBROŽ, Ondřej. *Rekonstrukce 3D objektu z obrazových dat*. Brno: Vysoké učení technické v Brně, Fakulta Informačních technologií, 2010. 59s. Vedoucí bakalářské práce byl Ing. Michal Španěl

[6] NVIDIA CUDA™ : Programming Guide [online]. Version 2.1. [s.l.] : [s.n.], 12/8/2008 [cit. 2011-04-20]. Dostupné z WWW: <http://developer.download.nvidia.com/compute/cuda/2_1/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.1.pdf>.

[7] BRADSKI, G. - KAEHLER, A. Learning OpenCV. Cambridge: O´Reilly, 2008. 577 s. ISBN 978-0-596-51613-0.

[8] OWENS. Computer Vision IT412. [online]. 29.10.1997 [cit. 2012-05-03]. Dostupné z: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT9/

[9] OpenCV Wiki [online]. c2010, WWW stránky pro podporu OpenCV společenství [cit. 2012-05-03]. Dostupné z:

[10] CUDA BY EXAMPLE : An Introducton to General/Purpose GPU Programming [online]. 28/07/010 [cit. 2012-05-3]. Dostupné z WWW: http://developer.download.nvidia.com/books/cuda-by-example/cuda-by-example-sample.pdf

[11] Hlaváč V., Šonka M.: Počítačové vidění,Grada, 1992, ISBN 80-85424-67-3

[12] ŽÁRA, J., BENEŠ, B., SOCHOR, J., FELKEL, P. Moderní počítačová grafika. Computer press, 2004. 628 s. ISBN 80-251-0454-0.

[13] BRADSKI, Gary a Adrian KAEHLER. O´REILLY. *Learning OpenCV*. Sebastopol: O´Reilly Media, 2008. ISBN 978-0-596-51613-0.

[14] NVIDIA: NVIDIA CUDA Compute Unified Device Architecture Programming Guide [online]. Verision 1.1. NVIDIA, 29.11.2007 [cit. 2012-04-28]. Dostupné z: http://www.mpe.mpg.de/~umaio/manual_NVIDIA_CUDA_Programming_Guide_1.1.pdf

[15] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense twoframe stereo correspondence algorithms. IJCV, 47(1/3):7–42, April–June 2002.

[16] Úvod do technologie CUDA [online]. 2009 [cit. 2012-04-28]. Dostupné z: http://www.root.cz/serialy/uvod-do-technologie-cuda/

[17] Bumblebee datasheet [online]. 2012 [cit. 2012-12-15]. Dostupné z: http://www.ptgrey.com/products/bumblebee2/bumblebee2_xb3_datasheet.pdf.

[18] *Počítačové vidění* [online]. Brno 16.4.2008. [cit. 2012-12-15]. Dostupné z: http://www.uamt.feec.vutbr.cz/vision/TEACHING/MPOV/Pocitacove_videni_S.pdf

[19] Corner detection [online]. 2012 [cit. 2012-12-15]. Dostupné z: http://kiwi.cs.dal.ca/~dparks/cornerdetection/harris.htm

[20] HASMANDA, Martin. *Zpracování stereoskopické videosekvence*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 66s. Vedoucí diplomové práce byl Ing. Kamil Říha, Ph.D.

[21]http://midas.uamt.feec.vutbr.cz/POV/lectures-pdf/10_Lokalni_priznaky_a_korespondence.pdf

[22] K. Takita, M. A. Muquit, T. Aoki, and T. Higuchi, "A subpixel correspondence search technique for computer vision applications," IEICE Trans. Fundamentals, vol. E87-A, no. 8, pp.1913–1923, Aug. 2004.

6 SEZNAM PŘÍLOH

Příloha 1. CD/DVD, které obsahuje: zdrojový kód,
Cuda_Occupancy_calculator.xls
dokumentaci

7 SEZNAM OBRÁZKŮ

8 PŘÍLOHY

8.1 Hardware zařízení č.1

System Information

Time of this report: 5/7/2013, 13:10:19
Machine name: PC-E617-09
Operating System: Windows 7 Professional 64-bit (6.1, Build 7601) Service Pack 1 (7601.win7sp1_gdr.130318-1533)
Language: Czech (Regional Setting: Czech)
System Manufacturer: ATComputers
System Model: H55M-S2V
BIOS: Award Modular BIOS v6.00PG
Processor: Intel(R) Core(TM) i5 CPU 650 @ 3.20GHz (4 CPUs), ~3.2GHz
Memory: 4096MB RAM
Available OS Memory: 3832MB RAM
Page File: 3460MB used, 4465MB available
Windows Dir: C:\Windows
DirectX Version: DirectX 11
DX Setup Parameters: Not found
User DPI Setting: Using System DPI
System DPI Setting: 96 DPI (100 percent)
DWM DPI Scaling: Disabled
DxDiag Version: 6.01.7601.17514 32bit Unicode

Display Devices

Card name: NVIDIA GeForce GTX 560
Manufacturer: NVIDIA
Chip type: GeForce GTX 560
DAC type: Integrated RAMDAC
Device
Key: Enum\PCI\VEN_10DE&DEV_1201&SUBSYS_83B51043&REV_A1
Display Memory: 2638 MB
Dedicated Memory: 978 MB
Shared Memory: 1659 MB
Current Mode: 1920 x 1080 (32 bit) (60Hz)
Monitor Name: Obecný monitor PnP
Monitor Model: HP S2231

Monitor Id: HWP2905
Native Mode: 1920 x 1080(p) (60.000Hz)
Output Type: DVI
Driver Name: nvd3dumx.dll,nvwgf2umx.dll,nvwgf2umx.dll,nvd3dum,nvwgf2um,nvwgf2um
Driver File Version: 9.18.0013.1106 (English)
Driver Version: 9.18.13.1106
DDI Version: 11
Driver Model: WDDM 1.1
Driver Attributes: Final Retail
Driver Date/Size: 2/26/2013 00:32:38, 18055184 bytes

Drive: C:
Free Space: 165.2 GB
Total Space: 276.0 GB
File System: NTFS
Model: WDC WD5000AAKX-001CA0 ATA Device

Drive: D:
Free Space: 200.7 GB
Total Space: 200.8 GB
File System: NTFS
Model: WDC WD5000AAKX-001CA0 ATA Device

8.2 Hardware zařízení č.2

System Information

Time of this report: 5/8/2013, 16:52:20
Machine name: MACA-LENOVOPC
Operating System: Windows 7 Home Premium 32-bit (6.1, Build 7601) Service Pack 1 (7601.win7sp1_gdr.130318-1533)
Language: Czech (Regional Setting: Czech)
System Manufacturer: LENOVO
System Model: 20023
BIOS: Ver 1.00PARTTBLI
Processor: Intel(R) Core(TM)2 Duo CPU T6600 @ 2.20GHz (2 CPUs), ~2.2GHz
Memory: 4096MB RAM

Available OS Memory: 3036MB RAM
Page File: 4694MB used, 1440MB available
Windows Dir: C:\windows
DirectX Version: DirectX 11
DX Setup Parameters: Not found
User DPI Setting: Using System DPI
System DPI Setting: 96 DPI (100 percent)
DWM DPI Scaling: Disabled
DxDiag Version: 6.01.7601.17514 32bit Unicode

Display Devices

Card name: NVIDIA GeForce G210M
Manufacturer: NVIDIA
Chip type: GeForce G210M
DAC type: Integrated RAMDAC
Device
Key: Enum\PCI\VEN_10DE&DEV_0A74&SUBSYS_389F17AA&REV_A2
Display Memory: 1728 MB
Dedicated Memory: 466 MB
Shared Memory: 1262 MB
Current Mode: 1366 x 768 (32 bit) (60Hz)
Monitor Name: Obecný monitor PnP
Monitor Model: unknown
Monitor Id: AUO20EC
Native Mode: 1366 x 768(p) (60.006Hz)
Output Type: Internal
Driver Name: nvd3dum.dll,nvwgf2um.dll,nvwgf2um.dll
Driver File Version: 9.18.0013.0694 (English)
Driver Version: 9.18.13.694
DDI Version: 10.1
Driver Model: WDDM 1.1
Driver Attributes: Final Retail
Driver Date/Size: 12/14/2012 10:55:16, 15316840 bytes

Drive: C:
Free Space: 3.9 GB
Total Space: 106.4 GB
File System: NTFS
Model: Hitachi HTS543232L9A300

Drive: D:

Free Space: 28.3 GB
Total Space: 183.5 GB
File System: NTFS
Model: Hitachi HTS543232L9A300

8.3 Ukázka kalibračních snímků