Vysoké učení technické v Brně
Fakulta strojního inženýrství
Ústav mechaniky těles, mechatroniky a biomechaniky

Bakalářská práce

Brno 2014

Bibliografická citace

KUMPÁN, P. Vytvoření 3D modelu prostředí pomocí senzoru Kinect. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2014. **??** s. Vedoucí bakalářské práce Ing. Michal Růžička.

Abstrakt

Je obecně známou věcí, že člověk bývá při zkoumání grafického návrhu rozptylován okolním textem, pokud mu dává nějaký smysl. Úkolem Lorem Ipsum je pak nahradit klasický smysluplný text vhodnou bezvýznamovou alternativou s relativně běžným rozložením slov. To jej dělá narozdíl od opakujícího se "Tady bude text. Tady bude text..."mnohem více čitelnějším. V dnešní době je Lorem Ipsum používáno spoustou DTP balíků a webových editorů coby výchozí model výplňového textu. Ostatně si zkuste zadat frázi "lorem ipsum"do vyhledavače a sami uvidíte. Během let se objevily různé varianty a odvozeniny od klasického Lorem Ipsum, někdy náhodou, někdy účelně (např. pro pobavení čtenáře).

Abstract

Navzdory všeobecnému přesvědčení Lorem Ipsum není náhodný text. Jeho původ má kořeny v klasické latinské literatuře z roku 45 před Kristem, což znamená, že je více jak 2000 let staré. Richard McClintock, profesor latiny na univerzitě Hampden-Sydney stát Virginia, který se zabýval téměř neznámými latinskými slovy, odhalil prapůvod slova consectetur z pasáže Lorem Ipsum. Nejstarším dílem, v němž se pasáže Lorem Ipsum používají, je Cicerova kniha z roku 45 před Kristem s názvem "De Finibus Bonurum et Malorum"(O koncích dobra a zla), konkrétně jde pak o kapitoly 1.10.32 a 1.10.33. Tato kniha byla nejvíce známá v době renesance, jelikož pojednávala o etice. Úvodní řádky Lorem Ipsum, "Lorem ipsum dolor sit amet...", pocházejí z kapitoly 1.10.32 z uvedené knihy.

Poděkování

Děkuji tomu že Lorem Ipsum má kořeny v klasické latinské literatuře z roku 45 před Kristem, což znamená, že je více jak 2000 let staré. Richard McClintock, profesor latiny na univerzitě Hampden-Sydney stát Virginia, který se zabýval téměř neznámými latinskými slovy, odhalil prapůvod slova consectetur z pasáže Lorem Ipsum. Nejstarším dílem, v němž se pasáže Lorem Ipsum používají, je

Čestné prohlášení

Prohlašuji na svou čest, že bakalářskou práci na téma *„Vytvoření 3D modelu prostředí pomocí senzoru Kinect“* jsem vypracoval samostatně, pod vedením vedoucího bakalářské práce pana Ing. Michala Růžičky a s použitím uvedených literárních a internetových zdrojů.

V Brně dne 5. 5. 2014 .....

Pavel Kumpán

Obsah

Úvod

Hlavním cílem práce bylo navrhnout, implementovat a otestovat metodu pro rekonstrukci prostředí za pomoci hloubkového senzoru Kinect. U částí systému kde to bylo vhodné a možné, byla metoda implementována jak ve verzi pro obvyklý procesor x86, tak i ve verzi s využitím možnosti výpočtu na grafické kartě.

V současné době se touto problematikou zajímají tři projekty:

- Kinect Fusion vyvinutý v Microsoft Research v roce 2011, použité algoritmy byly veřejně publikovány na konferencích o počítačovém vidění a zpracování obrazu. Kinect Fusion je společností Microsoft často používán pro efektní demonstraci možností Kinectu.
- KinFu je otevřený projekt založený na knihovně Point Cloud Library a algoritmech Kinect Fusion.
- ReconstructMe přestavuje komerční software vyvinutý rakouskou společností Profactor.

Kapitola 1

Zařízení Kinect

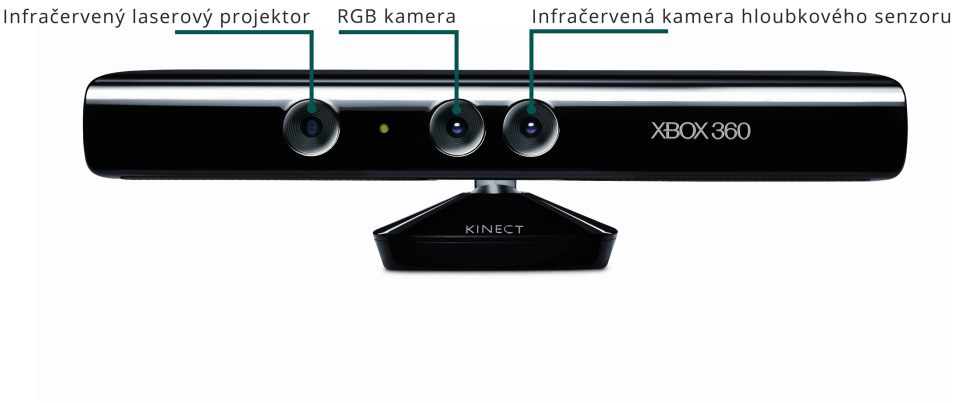
Kinect je hardwarový ovladač, který byl uveden jako doplněk ke hrací konzoli Xbox 360 společností Microsoft v listopadu roku 2010. SDK pro tvorbu aplikací třetích stran pro nekomerční užití Microsoft vydal v červnu 2011. V únoru 2012 byla uvolněna verze Kinectu přímo určená pro operační systém Windows a použití s PC. V roce 2014 se očekává uvedení nové verze Kinect 2.0 s vylepšenými parametry hardwaru.

1.1 Hardwarové parametry

Zařízení obsahuje RGB kameru o 8-bitovém maximálním rozlišení 1280×960 pixelů schopnou pracovat s frekvencí snímkování 12 Hz, případně lze použít rozlišení 640×480 a nižší s frekvencí 30 Hz. Dále se v zařízení nachází infračervený hloubkový senzor o maximálním rozlišení 640×480 pixelů a 11-bitové obrazové informaci s účinným snímáním objektů ve vzdálenosti 0,4 - 4 m. Zorný úhel je 43 ° vertikálně a 53 ° horizontálně. Maximální frekvence snímkování je 30 Hz.

Firmware Kinectu umožňuje detekci osob ve scéně i jednotlivých gest. Zařízení také obsahuje pole mikrofonů pro snímání zvuků okolí a je vybaveno servomotory pro autonomní změnu náklonu.

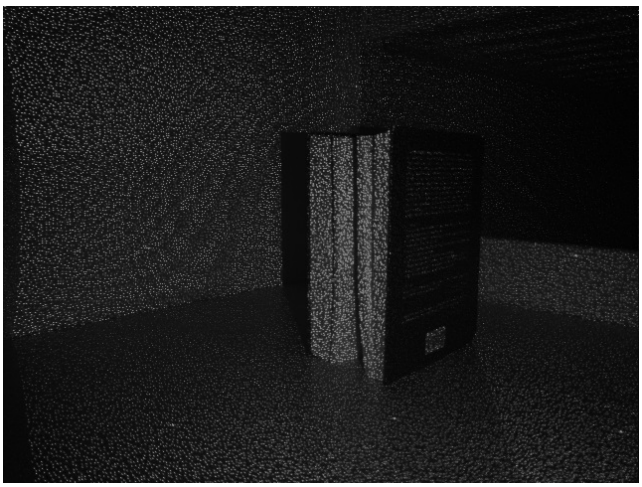
Pro tvorbu softwaru lze využít dvou nejrozšířenějších SDK. Oficiální Kinect SDK od společnosti Microsoft a komunitní otevřené SDK OpenKinect. Pro účely této práce bylo použito oficiální Kinect SDK.



Obrázek 1.1: Zařízení Kinect

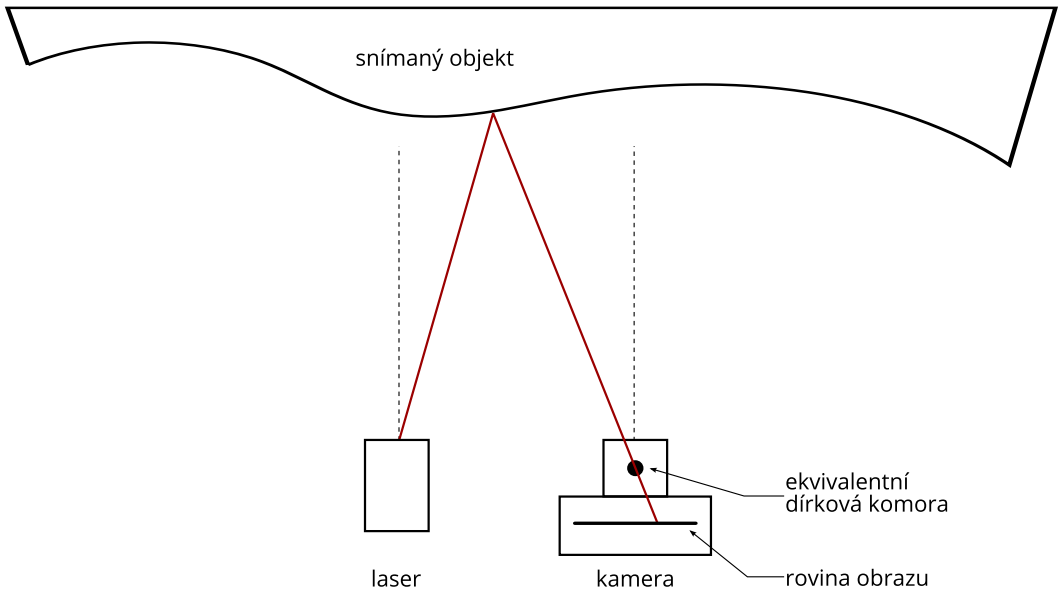
1.2 Hloubkový senzor

V této práci byl primárně využíván hloubkový senzor zařízení. Skládá se z infračerveného laserového emitoru a monochromatického CMOS senzoru. Díky použití infračerveného rozsahu spektra jej lze využívat relativně nezávisle na světelných podmínkách. Emitor na snímanou scénu stále promítá vzor bodů.

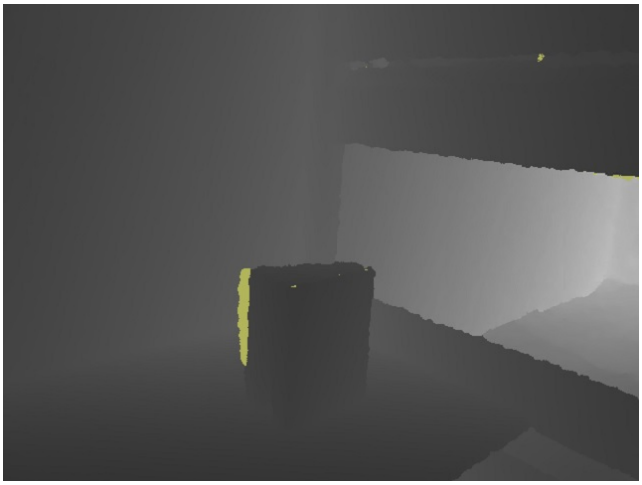


Obrázek 1.2: Snímek promítaného vzoru v infračerveném spektru

Body je následně snímán CMOS senzorem a z porovnání rozdílu mezi vzorem snímaným a promítaným jsou pomocí stereo triangulace vypočteny vzdálenosti pro jednotlivé pixely snímaného obrazu. Ty jsou poté do déle předány jako pole 16-bitových hodnot které tvoří takzvanou hloubkovou mapu. První tři bity obsahují informaci o hráči, nachází-li se bod na nějakém. Horních třináct bitů pak obsahuje vzdálenost bodu scény od senzoru v milimetrech.



Obrázek 1.3: Princip hloubkového senzoru

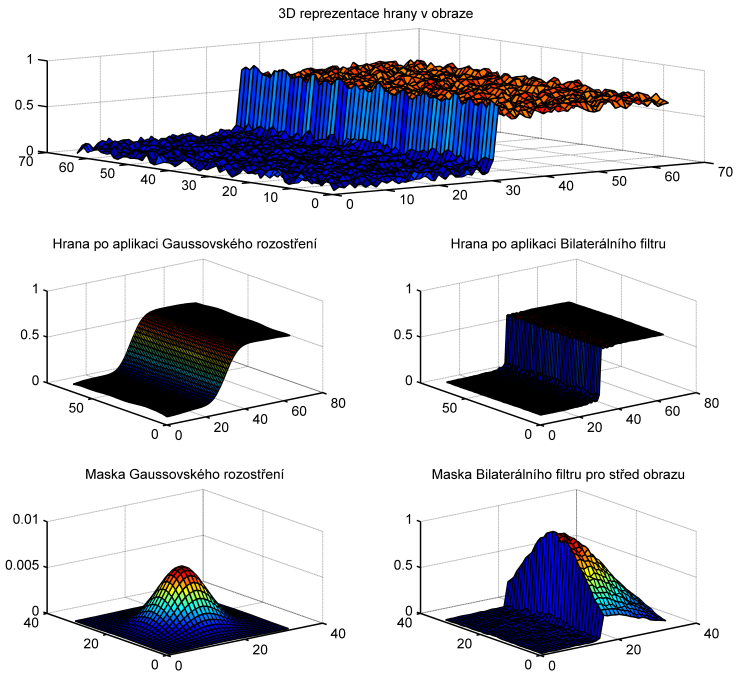


Obrázek 1.4: Hloubkový snímek s hloubkou vyjádřenou pomocí odstínů šedi

Kapitola 2

Popis metody

V této kapitole bude zevrubně popsána implementovaná metoda.



Obrázek 2.1:

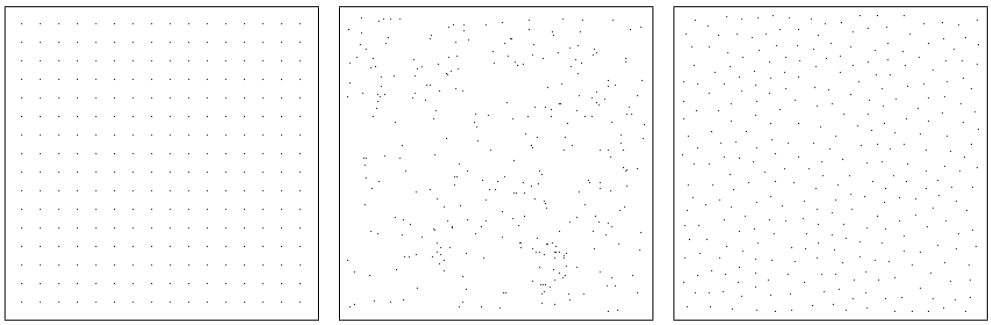
2.1 Stereo triangulace

2.2 Filtrování obrazových dat

Nasnímaná data nejprve zpracujeme pomocí filtru, ukázalo se že tento krok výrazně zlepšuje rychlost konvergence a kvalitu výstupů dalších kroků algoritmu. Na hloubkový sken v pravidelném rastru se můžeme dívat jako na obrázek o jednom barevném kanálu, což nám umožňuje použít grafické filtry běžně používané pro zpracování obrazu. Naším hlavním cílem je redukce šumu. K tomu se často používá Gaussovo rozostření. Nedostatkem tohoto filtru je však to, že dochází ke ztrátě informace o hranách v obraze. Tento nedostatek řeší bilaterální filtr, který zachovává ostré hrany, ale odstraňuje šum.

(a) Původní obrázek (b) Gaussovo rozostření (c) Bilaterální filtr

Obrázek 2.2: Porovnání Gaussova rozostření a bilaterálního filtru



(a) Mřížka (b) Náhodný výběr (c) Poisson Disk

Obrázek 2.3: Porovnání vzorku vytvořeného algoritmem Poisson Disc se vzorkem pravidelné mřížky a náhodného výběru

Bilaterální filtr má jakožto nelineární filtr v základní implementaci vysokou časovou náročnost $O(m * N^k)$ (m je počet pixelů v obraze, N je poloměr jádra filtru, k je počet rozměrů obrazu), což je pro účely filtrace v reálném čase zcela nedostatečné. Proto jsme použili přístup, který využívá vzorkování filtrované oblasti a časovou náročnost zlepšuje na $O(m * N * k)$. Pro vzorkování je použita metoda Poisson Disc. Vzorky získané touto metodou mají rovnoměrnější rozdělení než vzorky vygenerované pomocí pseudonáhodných čísel a oproti vzorkům s pravidelným mřížkovým rozložením nevytvářejí v obrazu fragmenty.

2.3 Převod hloubkové mapy na body v prostoru

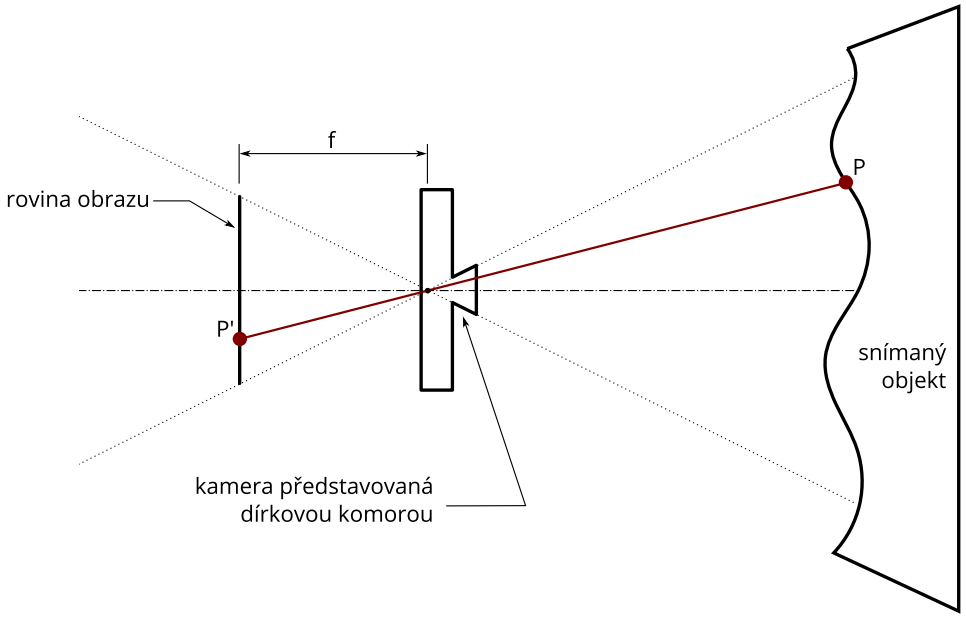
Pro převod dat z hloubkové mapy je hloubkový senzor Kinectu nahrazen zjednodušeným modelem dírkové komory - viz obrázek 2.4. Předpokládá se, že každý reálný bod P o souřadnicích (x, y, z) je promítán na obrazovou rovinu ležící ve vzdálenosti f od dírky jako bod P' o souřadnicích (u, v, d). Mezi souřadnicemi reálného bodu P a jeho obrazu na obrazové rovině P platí následující vztahy

$$P(x) = \frac{P'(d) \cdot (P'(u) - c_x)}{f} \tag{2.1}$$

$$P(y) = -\frac{P'(d) \cdot (P'(v) - c_y)}{f} \tag{2.2}$$

$$P(z) = P'(d) \tag{2.3}$$

kde (c_x, c_y) jsou souřadnice osy ohniska infračervené kamery Kinectu na snímku - tedy střed snímku. Souřadnice bodu P' u a v jsou souřadnicemi na hloubkové mapě v pixelech.



Obrázek 2.4: Geometrický model Kinectu

Souřadnice d je hodnota pole hloubkové mapy která odpovídá reálné vzdálenosti v milimetrech. Souřadnice bodu P jsou v milimetrech a vztahují se k lokálnímu souřadnému systému jehož počátek představuje právě dírka modelové dírkové komory.

2.4 Spojování skenů

Pro sestavení scény byl použit algoritmus ICP (iterative closest point - iterace přes nejbližší bod). Jedná se o iterační algoritmus pro spojování bodových mračen.

Vstupní data představují dvě množiny souřadnic bodů p_i a q_i, $i \in 1, ..., n$ ve trojrozměrném prostoru. Množina bodů **P** představuje základní snímek, množina **Q** nový snímek, který chceme zarovnat ke snímku **P**.

Problém představuje zarovnání nového snímku Q k předchozímu snímku P. P a Q obsahují souřadnice bodů vzhledem k aktuální poloze senzoru. Snímky však byly pořízeny v různých polohách senzoru. Chceme tedy body nového snímku posunout lokálního souřadného systému senzoru do globálního souřadného systému modelované scény, který pro nás představuje souřadný systém snímku P.

2.4.1 Výběr bodů pro zarovnání

Pro výběr bodů se kterými ICP pracuje lze zvolit několik strategií

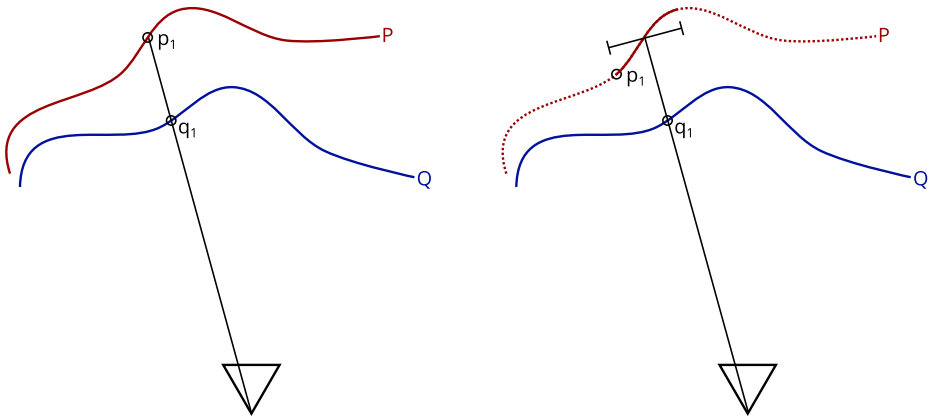
- Použití všech bodů je časově nejnáročnější, ale dává velmi dobré výsledky a je odolné proti šumu.
- Použití podmnožiny bodů s pravidelným nebo náhodným vzorkováním.
- Výběr zajímavých bodů na základě obrazové informace. Může se jednat o body s vysokou intenzitou gradientu hloubky nebo jasové složky barevné informace (je-li dostupná).
-

V systému bylo použito náhodné vzorkování metodou Poisson Disk (viz 2.3)

2.4.2 Získání párů bodů

Algoritmus ICP, jak už název napovídá, pracuje nad páry nejbližších bodů v jednotlivých snímcích. Tedy ke každému bodu q_i z množiny **Q** nového snímku musíme získat nejprve nejbližší bod p_i z množiny základního snímku **P**. V ideálním případě se jedná v reálném prostředí o totožné body.

- Hledání v celém množině je nejjednodušším přístupem. K nalezení bodu p_i musíme prohledat celou množinu P. Rychlejší variantou je nejprve body rozdělit do stromové struktury, například K-d stromu nebo oktalového stromu a problém se redukuje na prohledávání stromu s výrazně nižší časovou náročností. Výhodou toho přístupu jsou velmi kvalitní výsledky, možnost zarovnávat i snímky mezi nimiž je velká transformace. Nicméně pro real-time použití je tento přístup zcela nepoužitelný a své místo nachází především v aplikacích kde je výpočetní čas méně důležitý v porovnání s přesností.
- Metody založené na významných bodech jako Lucas-Kanade pracují s
- Projektivní asociace vychází z předpokladu že mezi snímky je velmi malá transformace a bod se na jednom snímku nachází na velmi podobných souřadnicích jako na druhém. Prohledávání okolí projekce je rozšířením projektivní asociace. Místo toho abychom za nejbližší bod prohlásili přímo bod na souřadnicích $[u, v]$, prohledáme oblast v jeho okolí a nalezneme nejbližší bod.



(a) Projektivní asociace (b) Prohledávání okolí projekce

Obrázek 2.5: Princip projektivní asociace a prohledávání okolí projekce

2.4.3 Popis algoritmu ICP

Relaci množin **P** a **Q** můžeme vyjádřit jako

$$p_i = \mathbf{R}q_i + \mathbf{t} + \mathbf{V}_i \tag{2.4}$$

kde **R** je čtvercová matice rotace, **T** je vektor translace a **V** představuje vektor šumu. Naším cílem je nalézt takovou transformaci vyjádřenou pomocí $[\hat{\mathbf{R}}, \hat{\mathbf{t}}]$, která minimalizuje kriteriální funkci

$$e = \sqrt{\sum_{i=1}^{n} ||p_i - \hat{\mathbf{R}}q_i - \hat{\mathbf{t}}||^2} \tag{2.5}$$

Nejprve určíme těžiště $\bar{p}$, $\bar{q}$ množin **P** a **Q**.

$$\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p_i \bar{q} = \frac{1}{n}\sum_{i=1}^{n} q_i \tag{2.6}$$

Určíme vektory mezi jednotlivými body množin a jejich těžišti.

$$p_c i = p_i - \bar{b} \tag{2.7}$$

$$q_c i = q_i - \bar{t} \tag{2.8}$$

Následně vypočteme korelační matici **H** z těchto vektorů.

$$\mathbf{H} = \sum_{i=1}^{n} q_{ci} p_{ci}^T \tag{2.9}$$

Pomocí singulárního rozkladu můžeme matici **H** rozložit na vektory **U** a **V**.

$$\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{2.10}$$

Ze kterých pak vypočteme matici rotace.

$$\hat{\mathbf{R}} = \mathbf{V}\mathbf{U}^T \tag{2.11}$$

Pokud je $\det(\hat{\mathbf{R}}) = -1$ pak došlo k selhání výpočtu. To může nastat, je-li v obraze příliš mnoho šumu, nebo jsou-li množin bodů rovinné. V tomto případě lze rotaci určit

$$\hat{\mathbf{R}} = \mathbf{U}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\mathbf{U}\mathbf{V}^T) \end{pmatrix}\mathbf{V}^T \tag{2.12}$$

Nejvhodnější translace je pak taková, která těžiště množiny **Q** posune do těžiště množiny **P**.

$$\hat{\mathbf{t}} = \bar{p} - \hat{\mathbf{R}}\bar{q} \tag{2.13}$$

Získanou transformaci můžeme aplikovat na jednotlivé prvky množiny **Q**

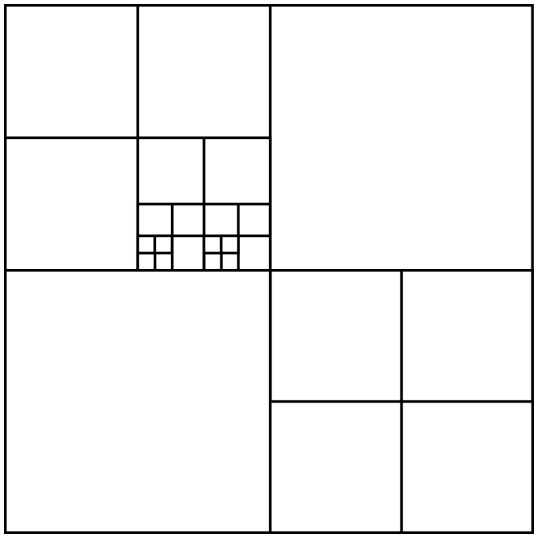
$$q_i = \mathbf{R}p_i + \mathbf{t} \tag{2.14}$$

Vypočteme hodnotu kriteriální funkce a je-li dostatečně malá algoritmus ukončíme. Není-li tomu tak, opakujeme algoritmus do té body než jsou snímky dostatečně zarovnané nebo nedosáhneme maximálního počtu iterací.

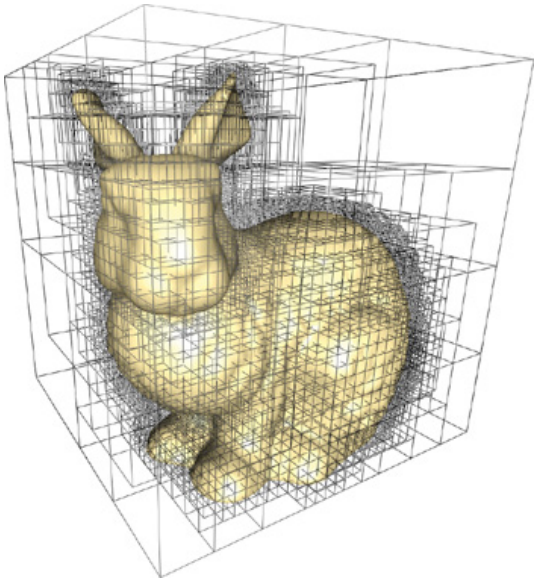
2.5 Reprezentace bodového mračna v programové paměti

Pro uchování zarovnaných snímků byla použita datová struktura *oktalový strom* (též *oktantový strom*, *octree*). Jedná se o stromovou strukturu používanou v počítačové grafice pro ukládání a reprezentaci trojrozměrných scén. Oktalový strom je o analogickým rozšířením 1-rozměrných binárních stromů.

Struktura se skládá z kořene, který si můžeme představit jako krychli zahrnující určitý prostor. Kořen má 8 potomků - buněk (*oktantů*, *octants*), které kořenovou krychli dělí na 8 menších krychlí. Každý z oktantů může mít svých 8 potomků, nebo může být koncovým uzlem, neboli listem (*leaf*). V listu jsou pak uloženy souřadnice všech vertexů, které se v dané oblasti prostoru vymezené krychlí uzlu nacházejí.



Obrázek 2.6: Struktura oktalového stromu ve dvourozměrné podobě



Obrázek 2.7: Model králíka zachycený ve struktuře oktalového stromu

Kapitola 3

Implementace

Navržená metoda byla implementována pod platformou .NET s použitím jazyka C#. Pro práci se zařízením Kinect bylo využito *Kinect for Windows SDK* , k maticovým výpočtům posloužila knihovna math.net. Kód pro paralelní zpracování jsem napsal pomocí OpenCL, propojení s .NET aplikací realizuje knihovna Cloo.

Podrobný popis implementace výše představené metody je mimo rozsah této práce. Zaujímá-li se čtenář o tuto oblast blíže, doporučuji nahlédnout přímo do zdrojových kódů, které jsou k práci přiloženy případně lze hlavní vývojovou vět nalézt také na GitHubu: `https://github.com/PavelKumpan/KinectDepth`. V této kapitole jsou alespoň zevrubně přiblíženy zajímavé nebo méně obvyklé části projektu.

3.1 Paralelizace

3.1.1 Krátký úvod do výpočtů na grafických kartách

I přes rapidní nárůst výkonu běžně dostupných procesorů v posledních letech jsou zde stále oblasti výpočtů u kterých je výpočetní náročnost úlohy na běžném procesoru mimo únosnou mez. Společným rysem velké části těchto úloh je to, že se jedná o úlohy paralelní, tedy takové, které je možné řešit na více výpočetních jednotkách zároveň a do jisté míry nezávisle. Přestože jsou vícevláknové procesory dnes již standardem, nejsou pro takovéto výpočty zcela ideálně použitelné. Počet vláken je zde stále relativně malý (v jednotkách až desítkách), kvůli čemu nemůžou konkurovat z principu paralelně navrženým grafickým kartám se stovkami až tisíci výpočetních jader.

Grafické karty nebyly dříve navrhovány tak, aby umožnily přímé programování.

Společnost nVidia uvedla své řešení pro programování nad grafickou kartou v roce 2007 pod označením CUDA. Konkurenční technologie Stream od AMD (ATI) spatřila světlo světa roce 2008. Nutnost standardizace vedla k vytvoření jednotného rozhraní a jazykové specifikace OpenCL spravovaného konsorciem Khronos. To je nyní podporováno grafickými kartami nVidia, AMD a Intel.

3.1.2 Struktura programu na GPU

Program který využívá grafickou kartu lze v zásadě rozdělit na tři části:

- *kernel*,
- *data*,
- *hostitelská aplikace*.

Kernel je podprogram v jazyce OpenCL C který běží na grafické kartě a realizuje potřebné výpočty. Kernely se zapisují do souborů s příponou **.cl*.

```
__kernel void vector_add_gpu (__global const float* src_a,
                              __global const float* src_b,
                              __global float* res,
                             const int num)
{
   const int idx = get_global_id(0);
   if (idx < num)
      res[idx] = src_a[idx] + src_b[idx];
}
```

Kód 3.1: Jednoduchý kernel pro součet dvou vektorů

Data se kterými je výpočet prováděn je třeba nejprve zkopírovat z operační paměti RAM do paměti grafické karty. Výsledky výpočtů jsou poté opět překopírovány z paměti GPU do RAM. *Hostitelská aplikace* se stará o agendu. Jedná se o běžnou aplikaci běžící na CPU, která pomocí rozhraní OpenCL načítá, kompiluje a spouští kernely, kopíruje data a poskytuje uživatelské rozhraní.

Kapitola 4

Výsledky měření

	Typ	**Procesor**	**Kapacita operační paměti**	**Dedikovaná grafická karta**
počítač 1	notebook	Intel core i7 3632QM	8 GB	nVidia GeForce GT740M
počítač II	notebook	Intel core i3 370M	4 GB	nemá

Tabulka 4.1: Parametry počítačů použitých k testům

Kapitola 5

Závěr

Seznam obrázků

Seznam zdrojových kódů

Seznam tabulek