Zpracování obrazu

Měření vzdálenosti videokamerou

8. května 2016

Autoři: Roman Čižmarik, xcizma04@stud.fit.vutbr.cz
Tomáš Mlynarič, xmlyna06@stud.fit.vutbr.cz

Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

Obsah

Kapitola 1

Úvod

Cílem tohoto projektu je určování reálné vzdálenosti od objektů pomocí stereokamery. Stereokamera je speciální typ kamery, která se skládá z dvou objektivů. Popis použité stereokamery je v sekci 2.5. Z rozdílů mezi levým a pravým videokanálem kamery je možné sestrojit hloubkovou mapu a z ní následně určit reálnou vzdálenost objektů od kamery (viz kapitola 3). Před samotným měřením je však důležité kameru správně kalibrovat snímáním vzorů se známými rozměry, typicky šachovnice. Snímaná data budou zobrazovaná v interaktivním grafickém uživatelském rozhraní, kde si uživatel bude moct kliknutím na promítaný obraz zvolit bod, v kterém chce zjistit vzdálenost od kamery (viz 3.2).

Kapitola 2

Dostupné a použité technológie

V této kapitole jsou rozebrány dostupné technologie, kterými je možné problém řešit. Stručněji jsou popsány veřejně dostupné technologie a knihovny, nicméně způsob práce se stereokamerou je představen podrobněji.

2.1 Qt

Pro vytvoření uživatelského rozhraní a obecně celé aplikace byl zvolen framework Qt 5. Tento framework byl zvolen díky jednoduchosti použití s stereokamerou, jelikož kamera zpřístupňuje API v jazyce C, dostupnosti frameworku a zejména multiplatformnosti jelikož každý z členů týmu pracoval v jiném operačním systému. Vytvořená aplikace se tedy vždy přizpůsobí operačnímu systému, na kterém je spuštěna a působí nativním vzhledem.

2.2 OpenCV

Knihovna OpenCV byla zvolena díky jednoduché manipulaci s maticemi a tudíž možnosti zpracovávat data z kamery. Kamera umožňuje získávat data jako 2D matici, tzn. jako raw data. Ty jsou poté jednoduchými konverzemi přizpůsobeny oknu aplikace a frameworku Qt.

2.3 OpenGL

OpenGL je multiplatformní rozhraní komunikující s grafickou kartou. Využívá se na tvorbu aplikací, které potřebují překreslovat data v reálném čase. V tomto projektu byl tento framework použit z důvodu potřeby vykreslovat relativně velké množství dat v závislosti na snímkovací frekvenci (*FPS*). Kromě samotného vykreslování snímků z kamery je také potřeba provádět různé výpočty (hloubková mapa nebo vzdálenost) a proto je optimálním řešením vykreslovací a počítací úlohy od sebe oddělit.

2.4 ZED Stereo camera

Pro tento projekt bylo možné použít stereokameru společnosti Stereolabs[1]. Tato kamera disponuje možnostmi rozlišit hloubku až na vzdálenost 20m což by vzhledem k povaze aplikace bylo zřejmě nejvhodnější. Bohužel kvůli technickým nárokům nebylo možné tuto kameru použít, jelikož je potřeba k získávání dat mít NVIDIA grafickou kartu, která umožňuje výpočty CUDA a také USB 3.0. Ani jeden z řešitelů projektů nesplňoval oba tyto požadavky.

[1] Oficiální stránky – https://www.stereolabs.com/

2.5 DUO 3D

Projekt byl vypracován s použitím stereokamery společnosti *Code Laboratorie Inc.*, konkrétně kamerou DUO MLX. Tato kamera, velmi malých rozměrů ($52 \times 25 \times 13mm$), umožňuje komunikaci a přenos dat skrze USB rozhraní (umožňuje i starší verzi 2.0) [2]. Ovládání kamery je možné pouze programově, tzn. kamera neobsahuje žádná tlačítka. Kamera je k dispozici pro operační systémy *Windows*, *Linux* i *OS X* a nejsou na ni kladeny velké výpočetní nároky. Z tohoto důvodu byla zvolena pro vypracování projektu. Ke kameře existuje dokumentace na oficiálních stránkách. Nicméně ta se projevila jako silně nedostatečná a neúplná. Dokumentaci je možno najít na stránkách výrobce [1].

2.5.1 Software Development Kit

Ke kameře je k dispozici sada nástrojů, *software development kit* (SDK), které obsahují knihovny pro překlad na danou platformu, které je poté možno přilinkovat k projektu, a také základní aplikace umožňující práci s kamerou:

- *DUODashboard* – Umožňuje rychlý přehled o kameře a nastavení, která umožňuje. Obsahuje několik sekcí pro zobrazení hloubkové mapy, klasického obrazu kamery a možnosti gyroskopu kamery.
- *DUOCalibration* – Aplikace umožňující nakalibrování kamery použitím známých vzorů pro kalibraci. Autoři tento vzor zpřístupňují na webových stránkách [2].
- *DUOFirmware* – Umožňuje jednoduché aktualizování firmware kamery.

SDK je možné stáhnout ze stránek výrobce po přihlášení k účtu[3], kterým byla kamera zakoupena. Po přihlášení je přehled o objednávkách a také několik licenčních kódů, které jsou potřebné pro spuštění kamery ve vlastní aplikaci (tj. kódy je možné zadat buď při kompilaci, nebo za běhu aplikace, nicméně před programovým spuštěním kamery).

2.5.2 API

Stereokamera obsahuje několik knihoven, které umožňují práci s kamerou prostřednictvím vlastních aplikací. Tyto knihovny jsou psané v jazyce *C*.

DUOLib(.h)

DUOLib Je knihovna pro práci s kamerou samotnou. Obsahuje funkce pro otevření/zavření spojení s kamerou, nastavení parametrů záznamu, jakými jsou šířka a výška snímků, FPS, podvzorkování (z angl. *binning*). Důležitou funkcí je statická callback funkce (řešena pomocí ukazatelé na funkci), která je kamerou volána při vytvoření nového snímku *DUOFrameCallback*. Během záznamu kamery je pak možné měnit některé parametry pomocí getter a setter funkcí (např. *GetDuoGain*, *SetDUOLedPWM* apod.).[4]

[2] Návod pro kalibraci kamery – https://duo3d.com/docs/articles/calibration
[3] Přihlášení a získání SDK – https://duo3d.com/account
[4] Celkový přehled všech funkcí – https://duo3d.com/docs/articles/api

Dense3D(.h)

Tato knihovna[5] umožňuje práci s hloubkovými daty, které jsou vypočítávány na hostitelském počítači. Podobně jako knihovna pro práci s kamerou je nejprve potřeba otevřít spojení (*Dense3DOpen*). Na rozdíl od práce s kamerou je potřeba zadat licenční kód kamery (viz 2.5.1). Kamera pak ve specifikované callback funkci umožňuje volání funkce *Dense3DGetDepth*, která vrací informace o hloubce z kamery.

Dense3DMT(.h)

Dle dokumentace[6] knihovna umožňuje podobnou práci s hloubkovými daty, nicméně pomocí více vláken. Jakým způsobem však knihovna nakládá s vlákny se nepodařilo z dokumentace zjistit. Získávání dat funguje podobným způsobem jako callback funkce knihovny *DUOLib*, tj. statická funkce *Dense3DFrameCallback*. Tato funkce pracuje v samostatném vláknu vytvořeném touto knihovnou a jako parametr je předán snímek obsahující jak klasické snímky z kamery (levé i pravé čočky), tak matici hloubkových dat. Více informací o tom, jak se data získávají nebo počítají se nepodařilo z dokumentace vyčíst.

[5] Dostupná specifikace na https://duo3d.com/docs/articles/dense3d-api

[6] Specifikace na https://duo3d.com/docs/articles/dense3dmt-api

Kapitola 3

Implementace

V této kapitole jsou nastíněny a popsány způsoby, jakými byla aplikace vypracována. Konkrétně se jedná o problematiku hloubkových map, dále je pak popsané ovládání a nakonec jakým způsobem byla práce rozdělena mezi členy týmu.

3.1 Hloubková mapa

V oboru zpracování obrazu se pod pojmem hloubková mapa rozumí obraz, který nese informaci o vzdálenosti plochy objektů ve scéně od pozorovatele. Na získání mapy je zapotřebí stereo obraz: dvojice snímků odpovídající vidění levého a pravého oka [7].

Klíčovým problémem je nalezení odpovídajících bodů v levém a pravém snímku, nazývaný „correspondence problem“[4][9]. Na jeho řešení se využívá tzv. „Block matching algorithm“ [4][8]. Hlavní myšlenkou tohoto algoritmu je porovnávání podobnosti mezi bloky stejných velikosti (matice $n \times m$) z levého a pravého snímku. K vyhodnocení podobnosti se využívá Střední kvadratická odchylka (MSE) Euklidovy vzdálenosti mezi hodnotami pixelů snímků. Hodnotou pixelu se rozumí informace o barvě. Přesnost algoritmu je vyšší o 20% — 25% pokud se při porovnávání využívá barevný obraz oproti šedotónovému obrazu. Hodnotu obrazu na pozici i, j můžeme zapsat jako koschan1996color:

$$F(i,j) = (R(i,j), G(i,j), B(i,j)) \tag{3.1}$$

Střední kvadratickou odchylku pak určíme následujícím vztahem:

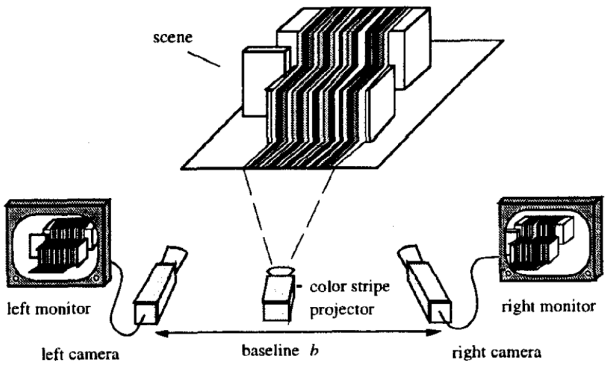
$$\begin{aligned} MSE_{color}(x,y,\Delta) = (&|R_r(x+i,y+j) - R_l(x+i+\Delta,y+j)|^2 + \\ &|G_r(x+i,y+j) - G_l(x+i+\Delta,y+j)|^2 + \\ &|B_r(x+i,y+j) - B_l(x+i+\Delta,y+j)|^2) \end{aligned} \tag{3.2}$$

kde *Delta* označuje rozdíl $(x_r - x_l)$ mezi posunem sloupců bloků v levém a pravém obrázku. Bloky se v prohledávané oblasti posouvají po pixelech.

Tento výpočet můžeme zjednodušit, pokud budeme předpokládat, že odpovídající pixelu se mohou vyskytovat pouze ve stejných sloupcích bloků. Pak je disparita D dvou bloků dána jako vodorovní vzdálenost s nejmenší MSE.

$$D = min\,\{MSE_{color}(x,y,\Delta)\} \tag{3.3}$$

Problematickou může byt ambiguita hodnot pixelů v daných blocích. Tento problém je možné eliminovat, pokud do scény přidáme další zdroj světla se známým spektrem, například duhovým, viz sekce 3.1.



Obrázek 3.1: Scéna s přidaným zdrojem světla (zdroj [4])

Po nalezení odpovídajících bodů můžeme vypočítat hloubkovou mapu triangulací [4][10]. Rozlišujeme dva základní typy hloubkových dat a to: řídké a husté hloubkové mapy. Řídké hloubkové mapy vznikají porovnáváním hran snímků, kdežto husté hloubkové mapy vznikají porovnáváním každého pixelu obou snímků. Husté hloubkové mapy jsou samozřejmě přesnější, ale také výpočetně náročnější [3][5].

3.2 Logika aplikace

Základní rozdělení uživatelského rozhraní je do dvou sekcí a to jako Vzdálenost (*Distance*) a také Hloubka (*Depth*).

Vzdálenost

V první sekci je dominantou interaktivní plocha zobrazující data z kamery v reálném čase. Kliknutím na tuto plochu se zobrazí bod (červený čtverec) s vypočtenou vzdáleností od kamery. Uživatel si může zvolit mezi měřením v jednom bodě nebo ve více bodech. V případě jednoho bodu se zobrazovací čtverec přesouvá podle posledního kliknutí. Při více bodech se při každém kliknutí přidá další bod měření. Dále je možné vybrat ze dvou módů výpočtu vzdálenosti: vestavěný (*built-in*) a vlastní (*computed*).

Vestavěná vzdálenost je získávána spolu se snímky kamery, které je možné číst v callback funkci definované před začátkem snímání (viz 2.5.2).

Vlastní vzdálenost je pak experimentálně vypočtena na základě rovnice:

$$distance = baseline * focal_lentgh/disparity.at(x, y)$$

kde hodnoty jsou $baseline = 30mm$, $focal_length = 2.05mm$ a *disparity* je získávána z dat kamery. Parametry kamery lze nalézt na [2]. Tímto způsobem je možné vypočítat hloubku v daném bodu (resp. bodech).

Obě tyto metody nastavují vzdálenost objektům třídy *DistancePoint*, které jsou alokovány v seznamu, který je sdílený mezi zobrazovacím widgetem a callbackem nového snímku. Samotné vykreslení dat poté probíhá tím způsobem, že se do upravené *thread-safe* fronty ukládají snímky získané z kamery a pak je vykreslovací widget informován o novém snímku. Tento widget poté vybírá snímky z bufferu a vykresluje je kombinovaně pomocí *OpenGL* a *QTPainteru*. Snímek je v *OpenGL* jako textura, která je aplikovaná na jednoduchý čtverec.

Hloubka

V sekci Hloubka se zobrazují hloubková data z kamery. Tato plocha není interaktivní a slouží spíše jako zajímavost. Hloubková mapa je vypočítána z matice disparit, kterou umožňuje získat kamera. Data jsou pak převedena pomocí nastaveného počtu disparit na hodnoty šedotónového obrázku. V další fázi je pak mapa obarvena pomocí vyhledávací tabulky (eng. *lookup table*), která je předpočítána při startu aplikace.

3.3 Měření a výsledky

Na základě několika pokusů byly zhotoveny výsledky měření kamerou. Měření probíhala uvnitř místnosti za dne (sluneční osvětlení). Experiment byl také vyzkoušen i večer, tzn. s umělým osvětlením, nicméně toto měření vykazovalo naprosto nepoužitelné výsledky (nahodilé), pravděpodobně z důvodu nedostatku světla.

Následující tabulky a grafy představují naměřené hodnoty pro několik předmětů: kniha, obal vinylu (čtvercový tvar), ruka. Tabulka 3.1 a graf 3.2 představují hodnoty naměřené zabudovaným algoritmem kamery, dále pak tabulka 3.2 a graf 3.3 představují algoritmus vlastní.

Z grafů je pak možné zjistit, že kamera měří celkem přesně do vzdálenosti 100cm, při vzdálenosti větší než 150cm kamera začíná špatně měřit vzdálenost menších předmětů, a při vzdálenosti 200cm pak menší předměty přestávají být rozpoznatelné.

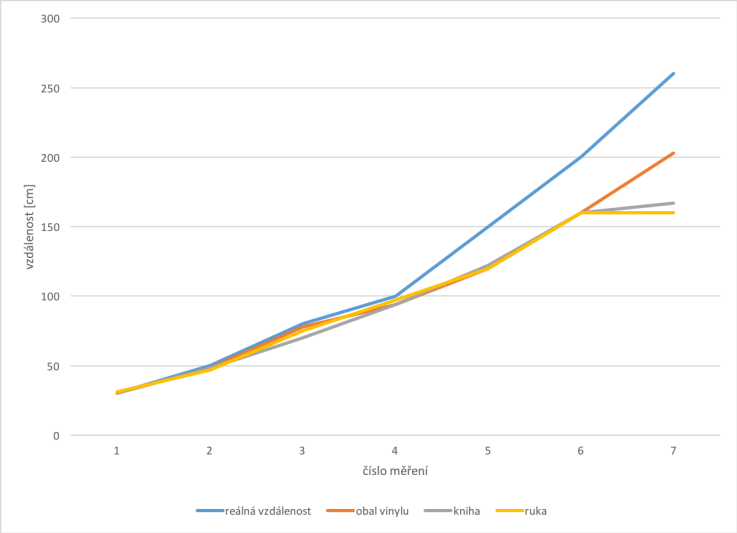
Při vlastním algoritmu je situace odlišná, jelikož jsou naměřené výsledky vždy trochu nepřesné, nicméně nenastává situace, že by se výsledky razantně měnily při větších vzdálenostech.

reálná vzdálenost	30	50	80	100	150	200	260
obal vinylu	30,15	48	77,9	94	120	160	203
kniha	30,02	48,00	70	94,09	122,00	160,00	167,00
ruka	31,00	47,00	75,00	97,00	120,00	160,00	160,00

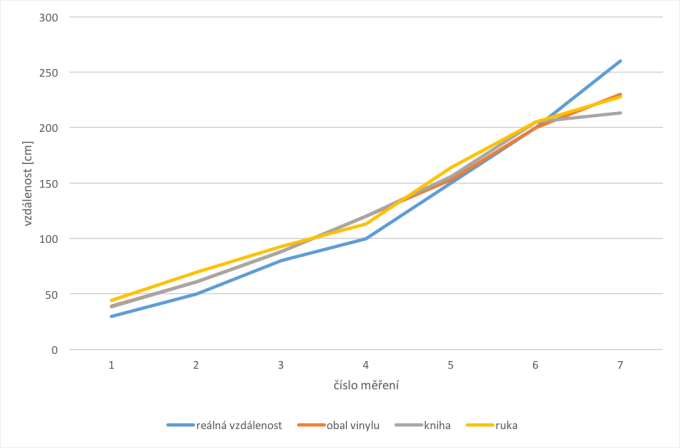
Tabulka 3.1: Výsledky měření předmětů vestavěným algoritmem [cm]

reálná vzdálenost	30	50	80	100	150	200	260
obal vinylu	39	61,00	88	120,00	153	200	230
kniha	38,4	61,00	88	120,00	156	205	213
ruka	44	69,7	92,8	113	164	205	228

Tabulka 3.2: Výsledky měření předmětů vypočítaným algoritmem [cm]



Obrázek 3.2: Porovnání výsledků vestavěného měření



Obrázek 3.3: Porovnání výsledků vypočítaného algoritmu

3.4 Rozdělení práce v týmu

K řešení projektu sice byly k dispozici 2 stereokamery, nicméně pouze jednu se podařilo zprovoznit a pracovat s ní. Z tohoto důvodu bylo rozdělení práce komplikované a práce probíhala víceméně současně.

Roman Čižmarik

Nastudoval problematiku získávání vzdálenosti ze stereo obrázků, připravil zrychlené vykreslování snímků pomocí OpenGL.

Tomaš Mlynarič

Studoval funkcionalitu stereo kamery DUO3D, připravil rozhraní pro práci s kamerou skrze Qt aplikaci, zdokonalil vykreslování pomocí OpenGL, konkrétně pak kombinaci vykreslování pomocí Qt Painteru a OpenGL.

3.5 Závěr

Cílem tohoto projektu bylo navrhnout a implementovat aplikaci, která měří vzdálenost pomocí stereokamery. Tento cíl se podařilo splnit. Výsledná aplikace demonstruje dva rozdílné přístupy k získávání reálné vzdálenosti od kamery. Experimentálně byla dokázána přesnost jednotlivých algoritmů (viz. 3.3). Přesnost měření různě ovlivňují následující faktory: osvětlení, vzdálenost od kamery, velikost měřeného objektu a nastavení kamery.

Možným a užitečným rozšířením by mohla být například funkce měření vzdálenosti mezi zadanými body nebo získávání informací o celém objektu na základě hloubky. V aktuální verzi je možné zadat více bodů měření, avšak vzdálenost je vždy vypočítaná vzhledem ke kameře.

Literatura

[1] Code Laboratories Inc. Duo api docs [online]. https://duo3d.com/docs/articles/api, [cit. 2016-05-02].

[2] Code Laboratories Inc. Duo mlx [online]. https://duo3d.com/product/duo-minilx-lv1, [cit. 2016-05-08].

[3] John R Jordan and Alan C Bovik. Using chromatic information in dense stereo correspondence. *Pattern Recognition*, 25(4):367–383, 1992.

[4] Andreas Koschan, Volker Rodehorst, and Kathrin Spiller. Color stereo vision using hierarchical block matching and active color illumination. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 1, pages 835–839. IEEE, 1996.

[5] Masatoshi Okutomi, Osamu Yoshizaki, and Goji Tomita. Color stereo matching and its application to 3-d measurement of optic nerve head. In *Pattern Recognition, 1992. Vol. I. Conference A: Computer Vision and Applications, Proceedings., 11th IAPR International Conference on*, pages 509–513. IEEE, 1992.

[6] Jay Rambhia. Stereo ranging [online]. http://www.jayrambhia.com/blog/stereoranging/, [cit. 2016-05-08].

[7] Dimitrios Tzovaras, Nikos Grammalidis, and Michael G Strintzis. Disparity field and depth map coding for multiview 3d image generation. *Signal Processing: Image Communication*, 11(3):205–230, 1998.

[8] Wikipedia. Block-matching algorithm — Wikipedia, the free encyclopedia [online]. https://en.wikipedia.org/wiki/Block-matching_algorithm, [cit. 2016-05-02].

[9] Wikipedia. Correspondence problem — Wikipedia, the free encyclopedia [online]. https://en.wikipedia.org/wiki/Correspondence_problem, [cit. 2016-05-02].

[10] Wikipedia. Triangulace — Wikipedia, the free encyclopedia [online]. https://cs.wikipedia.org/wiki/Triangulace, [cit. 2016-05-02].