VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

REALIZACE KAMEROVÉHO MODULU PRO MOBILNÍ ROBOT JAKO NEZÁVISLÉHO UZLU SYSTÉMU ROS - ROBOT OPERATING SYSTEM

REALIZATION OF CAMERA MODULE FOR MOBILE ROBOT AS INDEPENDENT ROS NODE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Ladislav Albrecht

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. Stanislav Věchet, Ph.D.

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Bc. Ladislav Albrecht**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**doc. Ing. Stanislav Věchet, Ph.D.**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Realizace kamerového modulu pro mobilní robot jako nezávislého uzlu systému ROS – Robot Operating System

Stručná charakteristika problematiky úkolu:

Hloubkové mapy vytvořené složením informací obsažených v obrazových datech z kamer patří k populárním doplňkovým senzorům pro mobilní roboty. Cílem předložené práce je realizovat kamerový modul jako hardwarově nezávislý senzorický vstup s možností doplnit mobilního robota více kamerami dle požadavků řešené úlohy. S využitím několika kamer jako nezávislých uzlů systému ROS (Robot Operating System) lze takto propojené kamery použít jako doplňkový detektor pro navigační úlohy robota, včetně tvorby hloubkové mapy. Každý uzel má být realizován pomocí samostatného hardwarového řešení umožňující jednoduché škálování navrženého řešení na dva a více kamer.

Cíle diplomové práce:

1. Seznamte se se systémem ROS.
2. Proveďte rešerši konektivity více kamer v rámci systému ROS.
3. Proveďte rešerši metod tvorby hloubkových map.
4. Připravte hardware systému tak aby bylo možno zpracovávat data minimálně ze dvou nezávislých kamer v rámci systému ROS.
5. Navrhněte metodiku tvorby hloubkové mapy ze získaných obrazů.
6. Navržené řešení prakticky otestujte.

Seznam doporučené literatury:

ROS.org | Powering the world's robots. [online]. 2.11.2016. Dostupné z: http://www.ros.org/

LIGHT, R. A., Mosquitto: server and client implementation of the MQTT protocol, The Journal of Open Source Software, vol. 2, no. 13, May 2017, DOI: 10.21105/joss.00265

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Stereo vize patří v dnešní době k jednomu z nejoblíbenějších prvků v oblasti mobilních robotů a výrazně přispívá k jejich autonomnímu chování. Cílem diplomové práce bylo navrhnout a realizovat kamerový modul jako hardwarově nezávislý senzorický vstup s možností doplnit soustavu dalšími kamerami, dále vytvořit hloubkovou mapu z páru kamer. Diplomová práce je rozdělena na teoretickou a praktickou část, včetně zhodnocení výsledků. Teoretická část přináší seznámení s frameworkem ROS, pojednává o metodách tvorby hloubkových map, a poskytne přehled nejpoužívanějších stereo kamer v robotice. Praktická část detailně popisuje přípravu experimentu a jeho vykonání. Jedná se zejména o přípravu hardwaru a softwaru. Dále popisuje kalibraci kamer a postup tvorby hloubkové mapy. Poslední kapitola obsahuje zhodnocení experimentu.

Klíčová slova

ROS, ROS uzel, stereo vize, hloubková mapa, kalibrace kamery, USB kamera, Raspberry Pi

Abstract

Stereo vision is one of the most popular elements in the field of mobile robots and significantly contributes to their autonomous behaviour. The aim of the diploma thesis was to design and implement a camera module as a hardware sensor input, which is independent, with the possibility of supplementing the system with other cameras, and to create a depth map from a pair of cameras. The diploma thesis consists of theoretical and practical part, including the conclusion of results. The theoretical part introduces the ROS framework, discusses methods of creating depth maps, and provides an overview of the most popular stereo cameras in robotics. The practical part describes in detail the preparation of the experiment and its implementation. It also describes the camera calibration and the depth map creating. The last chapter contains an evaluation of the experiment.

Keywords

ROS, ROS node, stereo vision, depth map, camera calibration, USB camera, Raspberry Pi

Bibliografická citace

ALBRECHT, Ladislav. *Realizace kamerového modulu pro mobilní robot jako nezávislého uzlu systému ROS - Robot Operating System.* Brno, 2020. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/124884. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Stanislav Věchet.

Čestné prohlášení

Prohlašuji, že jsem diplomovou práci na téma *Realizace kamerového modulu pro mobilní robot jako nezávislého uzlu systému ROS - Robot Operating System* vypracoval samostatně pod vedením vedoucího diplomové práce a s užitím uvedených podkladů a odborné literatury.

V Brně dne 26.6.2020 ……

Bc. Ladislav Albrecht

Poděkování

Tímto bych rád poděkoval doc. Ing. Stanislavu Věchetovi, Ph.D. za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování diplomové práce věnoval.

Dále bych chtěl poděkovat svojí rodině, která mi pomáhala při vykonávání experimentálních úloh.

Obsah

Úvod

Dnešní doba přináší mnoho zajímavých řešení v oblasti autonomního chování mobilních robotů. Příkladem jsou autonomní vozidla, různí pomocníci v domácnostech, nebo speciální aplikace v moderní armádě. Jedním z hlavních prvků u všech uvedených aplikací, ale i mnohých dalších, jsou hloubkové mapy. Hloubková mapa je snímek, který obsahuje informaci o vzdálenosti objektu od snímajících kamer. Kamery jsou v tomto případě dvě – systém strojového vidění funguje podobně jako lidské oči.

Hlavní cíl diplomové práce spočívá v realizaci kamerového modulu jako hardwarově nezávislého vstupu a následná tvorba hloubkové mapy. K provedení daných cílů bude využito možností frameworku ROS.

Teoretická část diplomové práce poskytuje v prvním řadě seznámení s frameworkem ROS. Distribuovaný design frameworku ROS je jedním z důvodů, proč byl zvolen pro realizaci cílů diplomové práce. Součástí seznámení s ROS jsou také statistiky uživatelů. Práce poskytne stručný přehled metrik z roku 2019, přehled nedávno vydaných verzí, komunikační systém, a v neposlední řadě vizualizační nástroje, kterými ROS jako framework disponuje. Dále jsou v diplomové práci popsány metody tvorby hloubkové mapy. Kapitola pojednávající o těchto metodách je rozdělena na metody aktivní a pasivní. Další kapitolu teoretické části tvoří rešerše populárních stereo kamer využívaných v robotice. Mluvíme o kamerách jako je Kinect, Bumblebee nebo ZED. Součástí kapitoly je také zajímavé řešení stereo vize pomocí Raspberry Pi Compute module.

Praktická část začíná specifikací hardwaru použitého při plnění cílů diplomové práce. Jedná se o tři jednodeskové počítače Raspberry Pi, každý se svojí připojenou USB kamerou, wifi router a laptop. Je zde také vyřešen zdroj pro napájení experimentální soustavy ze zásuvky pro zapalovač v autě (12 V). Součástí kapitoly o použitém hardwaru je také příprava experimentální soustavy a její implementace do osobního automobilu, a to včetně fotodokumentace. Dvě kamery byly přilepeny na střechu vozidla za čelním sklem, třetí kamera na čelní sklo zevnitř vozidla, zbylý hardware byl nainstalován na kompaktní dřevěnou desku nacházející se na zadním sedadle. Další kapitola praktické části se zabývá instalací ROSu. Poskytne také přehled nástrojů používaných při tvorbě této diplomové práce a vysvětlí tvorbu spouštěcího souboru (dále .launch soubor), který umožní spustit více uzlů najednou. Nezbytná při realizaci experimentu je časová synchronizace všech zařízení, nebo alespoň dvou Raspberry Pi, poskytujících levý a pravý obraz pro tvorbu hloubkové mapy, a master stanice, v tomto případě laptop. Důvodem je absence RTC (Real Time Clock) modulu v Raspberry Pi. Bez připojení k internetu, což je v běžném jedoucím automobilu běžné, si totiž Raspberry Pi nedokáže stanovit přesný čas. Důsledkem je jiný čas na každé stanici v systému. Časová synchronizace snímků (nezbytná pro tvorbu hloubkové mapy) z levé a pravé kamery není tím pádem možná. V posledním oddíle praktické části je detailně popsán postup kalibrace kamer a tvorba hloubkové mapy, a to včetně .launch souboru použitého při spouštění všech uzlů.

Poslední kapitola diplomové práce pojednává o vykonaném experimentu. Ten byl realizován jako série testů s testovacím vozidlem. Kapitola rozděluje experiment na tři části: vozidlo v klidu, při jednoduché manipulaci (například parkování) a za jízdy. Popisuje také podmínky při vykonávání experimentů, chování experimentální soustavy a kvalitu výsledné hloubkové mapy.

1. ROS

ROS je open-source meta-operační systém. Poskytuje služby na úrovni operačního systému, včetně hardwarové abstrakce, nízko-úrovňového ovládání zařízení, implementace běžně používaných funkcí, předávání zpráv mezi procesy a správu balíčků. Taktéž poskytuje nástroje a knihovny pro získávání, vytváření, psaní a spuštění kódu na více počítačích. ROS je v některých ohledech podobný robotickým frameworkům, jako jsou Player, YARP, Orocos, CARMEN, Orca, MOOS a Microsoft Robotics Studio [1].

Obr. 1-1 Loga uvedených robotických frameworků [2], [3], [4], [5], [6], [7], [8]

ROS běží v dnešní době hlavně na Unix platformách. Software pro ROS je primárně testován na Ubuntu a Mac OS X systémech, přestože ROS komunita přispívá podporou platforem, jako jsou Fedora, Gentoo, Arch Linux a jiné. Mluvíme zde o experimentálním provozu. Jsou to platformy, pro které je jen částečná podpora, nebo komunitou poskytnuty instalační příručky. Nutno ještě podotknout, že tyto příručky mohou být neúplné, nemusí být v souladu s nejnovějšími verzemi, nebo mohou instalovat pouze podmnožinu dostupných balíčků [9]. V tabulce (Tab. 1-1) je vyobrazeno několik platforem.

Plná podpora	Experimentální provoz		
Ubuntu	OS X (Homebrew)	OS X (MacPorts)	Android (NDK)
Ubuntu (armhf)	Debian	OpenEmbedded/Yocto	Arch Linux
	Windows	Ångström	UDOO
	Fedora	Gentoo	OpenSUSE
	Raspbian	QNX Realtime OS	Slackware
	FreeBSD		

Tab. 1-1 Plně a částečně podporované platformy [9]

Cílem tohoto meta-operačního systému je vytvořit jakýsi standard robotiky, takže při vytváření vlastní robotické aplikace již nemusíme znovu „vynalézt kolo“. Ten samý základní kód lze uplatnit u více druhů robotů, například robotické paže, dronu a podobně. Uzel lze naprogramovat v jazyku Python nebo C++. Komunikace mezi uzlem naprogramovaným v Pythonu a uzlem naprogramovaným v C++ je možná a snadná. Jakmile dojde k pochopení komunikace mezi jednotlivými uzly programu, může uživatel snadno rozšířit svůj projekt o další části. Výhodou je, že se může v budoucnu zaměřit na úplně jiný typ robotické aplikace a neztratí se. Široká škála dostupných ROS balíčků je jednou z hlavních výhod.

1.1. Distribuovaný systém

ROS byl navržen tak, aby byl co nejvíce distribuovaný a modulární. Díky modularitě se uživatel může rozhodnout, jak moc (nebo málo) ROS využije. Může si tedy vybrat, které balíčky mu budou užitečné, nebo které si naopak vytvoří sám. Distribuovaná povaha ROSu taktéž podporuje velkou komunitu uživatelsky přispívaných balíčků, které přidávají základnímu systému ROS na hodnotě [10].

Distribuovaný design umožňuje propojení více počítačů/zařízení, kde se každý počítač může chovat zcela nezávisle. Tyto počítače spolu vzájemně komunikují formou zpráv, které putují mezi zpracovávanými uzly jednotlivých zařízení. Master může být jeden, nebo jich může být i více [10].

V diplomové práci byl tento distribuovaný design ROSu testován. Jednalo se o jeden nadřazený master počítač a tři slave stanice. Na každé slave stanici běžel uzel pro USB kameru. Tyto slave stanice posílaly informace do master počítače, který data dále zpracovával. Jednou z výhod distribuovaného systému je, že jednotlivé počítače skutečně pracují samostatně. Není tedy ovlivněn celkový chod systému. Například co se dvou slave stanic poskytujících obraz USB kamer pro zpracování na hloubkovou mapu týče, není počet snímků za sekundu (FPS) nijak ovlivněn po připojení třetí slave stanice s kamerou. Více však v poslední kapitole o testování (Kap. 7).

1.2. ROS komunita

ROS komunita se v posledních několika letech rozrostla o velký počet uživatelů na celém světě. Komunita je velmi aktivní.

Obr. 1-2 Oficiální uživatelé ROSu ve světě [10]

Vývojáři ROSu poskytují každým rokem metriky, kterými měří aktivitu na svých stránkách a rovněž provádějí statistiky svého webu. Jsou publikovány ve formě reportů, které poskytují kvantitativní náhled na ROS komunitu. Následující odstavec bude pro zajímavost věnován nejnovější metrice, která byla vydána v červenci roku 2019 [11].

Metrika zaznamenala více než 1900 účastníků na e-mailovém seznamu *ros-users* (oficiální uživatelé), což je oproti roku 2014 (tedy během posledních 5 let) nárůst o 8 %. Stejná metrika udává více než 8400 lidí, spolupracujících na tvorbě online dokumentace (servery wiki.ros.org). Během posledních 5 let byl zaznamenán nárůst o 109 %. Počet osob, podílejících se při řešení různých problémů na stránkách www.answers.ros.org byl kolem 34000. Toto číslo každým rokem roste přibližně o 20%. Počet stránek webu www.wiki.ros.org činí 117000, z toho se 24 stránek upraví za den. Denně tyto wiki stránky shlédne 74000 lidí. Na webech www.answers.ros.org lze celkově nalézt 48000 otázek, z toho 32500 je zodpovězených. Za den přibude v průměru 21 otázek [12].

1.3. ROS verze

ROS verze neboli distribuce, je set ROS balíčků, podléhajících určité verzi. Účelem distribucí ROS je umožnit vývojářům pracovat s relativně stabilním kódovým základem. ROS distribuce jsou velmi podobné Linux distribucím. Všechny jsou plně kompatibilní s operačními systémy Ubuntu[13].

Verze	Datum vydání	Plakát	EOL datum
ROS Noetic Ninjemys	Květen 2020		Květen 2025
ROS Melodic Morenia	Květen 2018		Květen 2023
ROS Lunar Loggerhead	Květen 2017		Květen 2019
ROS Kinetic Kame	Květen 2016		Duben 2021
ROS Jade Turtle	Květen 2015		Květen 2017
ROS Indigo Igloo	Červenec 2014		Duben 2019

Tab. 1-2 Přehled posledních verzí ROS [13]

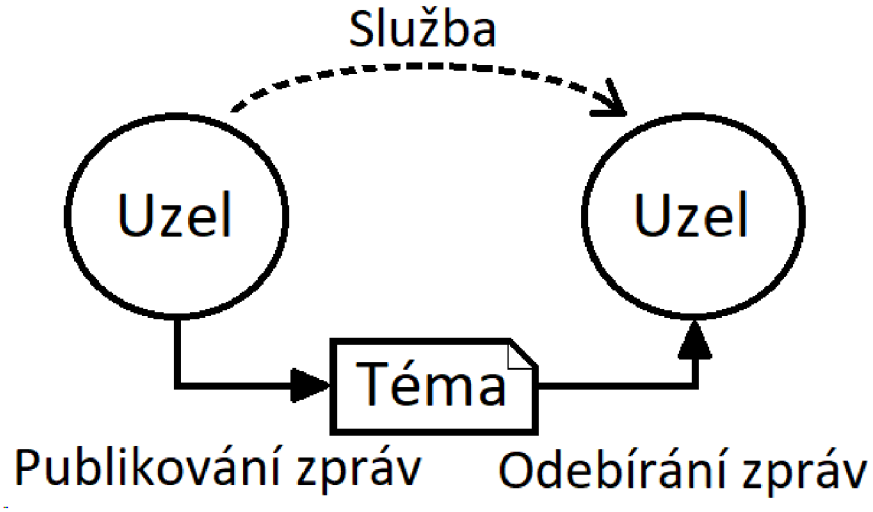
Nejnovější distribucí je Noetic Ninjemys (Tab. 1-2), která bude podporována až do května 2025. Toto datum se označuje jako EOL date (end of life – konec životnosti). V uvedené tabulce označuje zelená barva momentálně podporované verze a šedou barvou jsou zvýrazněny verze, kterým skončila životnost. Jedním z problémů nejnovější distribuce bývá, že některé balíčky nemusejí být pro danou verzi dostupné. Důvodem je to, že migrace balíčků z předchozích verzí zabere nějaký čas. Proto můžeme v tabulce vidět, že jsou podporovány tři verze. A to nejaktuálnější ROS Noetic Ninjemys, ROS Melodic Morenia, a ROS Kinetic Kame. Kinetic Kame začala v květnu 2016, tudíž od ní lze očekávat opravdu větší množství dostupných balíčků než u Noetic Ninjemys [13].

V této diplomové práci se pracovalo pouze s vydáním ROS Melodic Morenia.

1.4. Základní komponenty

Tato kapitola popíše základní kameny ekosystému ROSu, jelikož jde o velice rozsáhlé téma. ROS nabízí jisté rozhraní pro předávání zpráv, které poskytuje mezi–procesorovou komunikaci, díky čemuž je běžně označován jako middleware. Tato podkapitola popisuje základní služby, které tento middleware poskytuje [14, 15].

Komunikační systém je často jednou z nejdůležitějších věcí při vytváření robotické aplikace. Jak už bylo zmíněno, je ROS založen na komunikaci mezi jednotlivými uzly, procesory. Tato komunikace probíhá formou zpráv. Je založena na odesílání a přijímání (publishing/subscribing). Uzly si vyměňují zprávy prostřednictvím ROS tématu (ROS topic). Téma je anonymního charakteru, tím pádem uzly samy o sobě nevědí, s kým komunikují. Jednoduše řečeno, téma tzv. odděluje produkci informací od jejich spotřeby. Důsledek toho je prostý. Uzly, které data publikují, je publikují do příslušného tématu. Uzly, které data potřebují, se přihlásí k odběru na příslušné téma, které potřebnou informaci obsahuje. Na téma může být přihlášeno více uzlů. Výhoda je hlavně v přehlednosti systému. Témata jsou určena pouze pro jednosměrnou, sreamovanou komunikaci. Uzly, které potřebují provádět vzdálené volání procedur (přijímat a odesílat zároveň), využívají místo témat služby. ROS momentálně podporuje TCP/IP a UDP transport zpráv [14, 15].



Obr. 1-3 Komunikační systém ROSu [16]

Další služba se týká logování dat. Protože je systém publisher/subscriber anonymní a asynchronní, mohou být data jednoduše zachycena a později znovu přehrána bez jakékoliv změny v kódu. Jako příklad je uveden následující případ. Mějme úkol A, který čte data

z nějakého senzoru. Úkol B, který momentálně vyvíjíme, má zpracovávat data získaná díky úkolu A. Prostřednictvím logování můžeme jednoduše nahrát data poskytnutá úkolem A do nějakého souboru. Takto uložená data lze později několikrát opakovaně přehrávat (publikovat) z daného souboru, tím se nám ulehčí práce s úkolem B. Jedná se o silný návrhový vzor, který může výrazně snížit naše vývojové úsilí a zvýšit flexibilitu a modularitu v systému. Balíček, poskytující uvedené, se jmenuje *rosbag* [14].

Asynchronní chování odesílání/přijímání zpráv funguje v robotice dobře pro většinu komunikačních potřeb. Někdy je ale potřeba synchronní požadavek/odpověď mezi procesy. ROS middleware poskytuje řešení i pro tento problém, a to pomocí služeb (ROS services). Služba je definována párem zpráv (jedna pro požadavek, druhá pro odpověď). Poskytující ROS uzel nabídne službu pod jistým jménem, a klient volá službu, tím že pošle požadavek a čeká na odpověď. Knihovny klienta obvykle prezentují tuto iteraci uživateli, jakokdyby se jednalo o vzdálené volání procedury. Klient může navázat trvalé připojení ke službě, což umožňuje vyšší výkon za cenu menší odolnosti vůči změnám poskytovatele služeb. Služby jsou definovány .srv soubory. Nástroje v ROSu jsou *rossrv*, nebo *rosservice* [14, 17].

Dále mluvíme o službě *Parameter server*. Uzly využívají tuto službu k uložení a načtení parametrů i za chodu systému. Protože není navržen pro vysoce výkonné aplikace, je nejvhodnější pro statická data (typu integer, float, string, bool), jakými jsou třeba parametry konfigurace. V rámci diplomové práce to jsou například rozlišení, jas, gamma, ostrost apod.). V systému ROS je tato služba globálně dostupná pro všechna připojená zařízení [18].

1.4.1. Vizualizační nástroje

Jednou z nejsilnějších funkcí ROSu je sada nástrojů pro vývoj různých aplikací. Tyto nástroje podporují ladění, vykreslování, vizualizaci stavu vyvíjeného systému. Základní myšlenka publikování/odběru umožňuje spontánně prozkoumat data protékající systémem, což usnadňuje pochopení a ladění problémů v okamžiku jejich výskytu. Tyto nástroje využívají schopnost introspekce prostřednictvím rozsáhlé kolekce grafických a příkazových nástrojů, které zjednodušují vývoj a ladění. ROS lze také použít bez GUI (grafical user interface). Všechny základní funkce a nástroje pro introspekci jsou dostupné prostřednictvím jednoho z více než 45 nástrojů příkazového řádku. Existují příkazy pro spouštění skupin uzlů, představení témat a služeb, záznam a přehrávaní dat a řadu dalších situací. Pokud však dá uživatel přednost použití grafických nástrojů, poskytují *rviz* a *rqt* podobnou, i rozšířenou funkčnost. Grafické nástroje *rviz* a *rqt* budou dále krátce popsány [14].

- **rviz**

Je možná nejznámějším nástrojem ROS. Poskytuje general purpose, trojrozměrnou vizualizaci mnoha datových typů různých senzorů a jakéhokoliv robota popsaného URDF. Rviz umí vizualizovat mnoho běžných typů zpráv, jako jsou laserové skeny, množina bodů v trojrozměrném prostoru a obrazy z jedné nebo i více kamer. Také používá informace z *tf* knihovny, a to k zobrazení všech senzorových dat v běžném souřadnicovém systému dle uživatelského výběru, spolu s 3D vykreslením robota. Taková vizualizace všech získaných dat v jedné aplikaci jako je *rviz*, umožňuje rychle zjistit, co robot vidí, nebo identifikovat problémy, jako jsou nesprávné vyrovnání senzorů nebo nepřesnosti modelu robota. [14].

- **rqt**

Dalším populárním grafickým nástrojem je *rqt*. Jedná se o framework založený na *Qt* softwaru pro vývoj grafických rozhraní. Lze si vytvořit vlastní rozhraní, a to vytvořením a konfigurací rozsáhlé knihovny vestavěných modulů *rqt* do různých split-screen,

tabulkových nebo jiných rozložení. Napsáním vlastních *rqt* modulů lze také zavést nové komponenty rozhraní. Existuje několik *rqt* modulů. Zde budou některé stručně popsány [14].

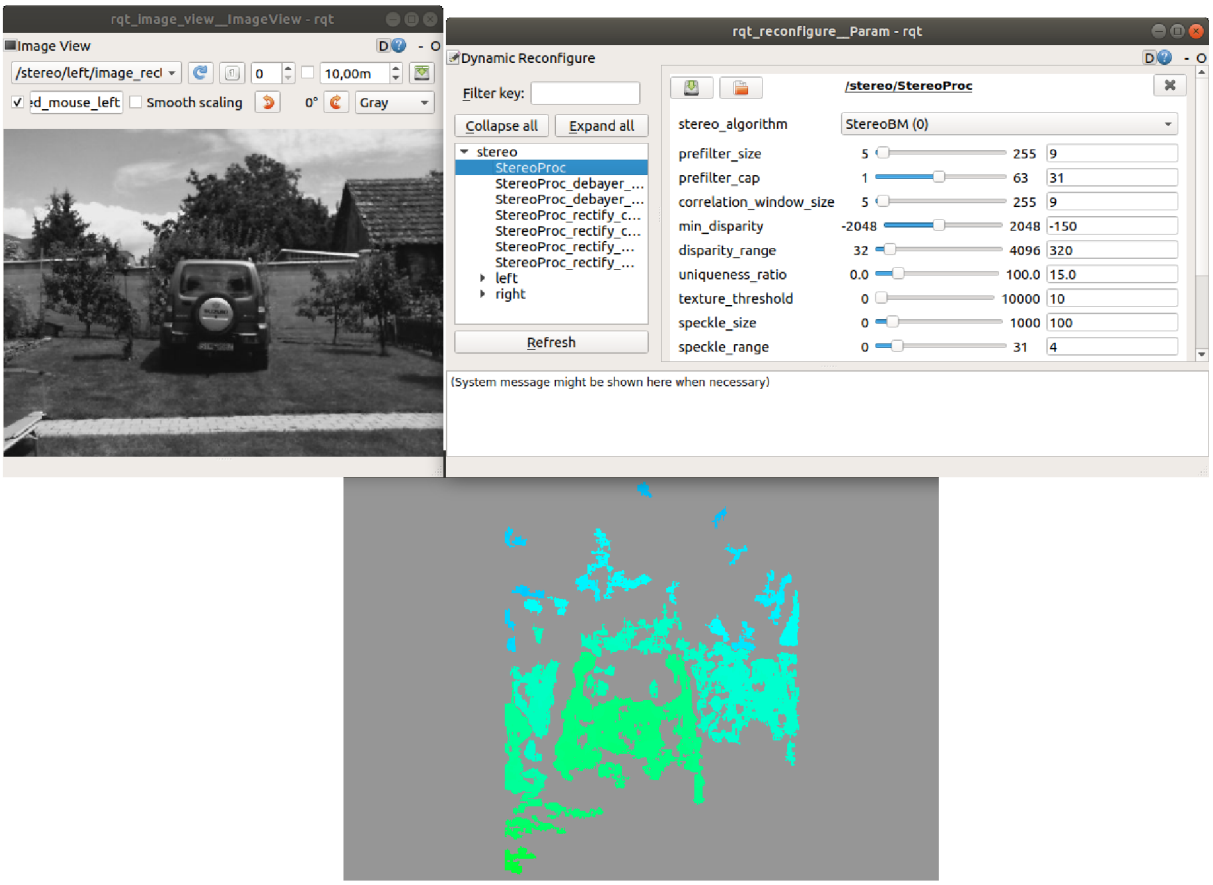
Modul *rqt_graph* poskytuje introspekci a vizualizaci živého systému ROS, zobrazuje uzly a propojení mezi nimi a umožňuje snadno ladit a porozumět běžícímu systému a jeho struktuře [14].

Modul *rqt_plot* dokáže sledovat například enkodéry, napětí, nebo cokoliv, co může být reprezentováno jako číslo měnící se v čase [14].

Pro monitorování a používání témat jsou k dispozici moduly *rqt_topic* a *rqt_publisher*. První z nich umožňuje sledovat a prozkoumat libovolný počet témat publikovaných v systému. Druhý umožňuje publikovat vlastní zprávy k libovolnému tématu, což usnadňuje experimentování s naším systémem, a to metodou „pokus-omyl“ [14].

Pro logování a přehrávání dat používá ROS formát .bag, jak už bylo zmíněno. Tento modul *rqt_bag* umožňuje zaznamenávat data do souborů typu .bag, následně je i přehrát, a dokonce i vizualizovat obsah tohoto souboru, včetně zobrazení obrazu a vykreslování numerické hodnoty v čase [14].

V této diplomové práci byl využíván modul *rqt_reconfigure*, kterým lze měnit například parametry hloubkové mapy za chodu systému. Další modul využíván v diplomové práci byl *rqt_image_view*. Pomocí tohoto modulu byly vyobrazeny témata ze všech kamer.



Obr. 1-4 Modul rqt_image_view (vlevo) a modul rqt_reconfigure (vpravo)

2. Metody tvorby hloubkových map

Hloubková mapa je obraz, který reprezentuje hloubkové uspořádání dané scény. Jedná se o relativní vyjádření hloubky daného obrazového bodu, korespondujícího s určitým bodem v prostoru. Zpravidla se jedná o obraz v šedých tónech, ale nemusí tomu tak být vždy. Zavedená konvence, která je často dodržována, udává světlejší barvou místo bližší k pozorovateli. Tmavší barvou se vyznačují objekty vzdálené. Důvodem je, že hloubková mapa je odvozena od horizontální paralaxy daného bodu. To platí pro pasivní metodu stereo snímání. Hloubková mapa však nemusí být vždy získána odvozením horizontální paralaxy. Mluvíme tak o aktivních metodách snímání [19].

Tato kapitola popisuje vybrané metody získávání hloubkových map. Pojednává o aktivních a pasivních metodách.

2.1. Aktivní metody

Aktivní metody snímání se vyznačují přidáním dodatečné informace do scény. Informace může být přidána prostřednictvím různých médií. Jedná se zejména o laser, infračervený zářič nebo projektor. O aktivních metodách snímání prováděných určitým světelným paprskem lze také mluvit jako o optických. Ty se pak dělí na koherentní a nekoherentní. Principem koherentních metod je interference, nekoherentních metod triangulace [19].

Jedná-li se o světelnou přidanou informaci, hodí se tyto metody především pro slabě osvětlené prostory. Nedostatečně silný zdroj přidávané světelné informace může mít za následek to, že se působením přímého denního světla může daná informace ztratit. Další nevýhodou jsou průhledné nebo lesklé povrchy [20].

Následující podkapitoly poskytnou přehled nejpoužívanějších aktivních metod snímání. Budou zde stručně popsány jednotlivé metody.

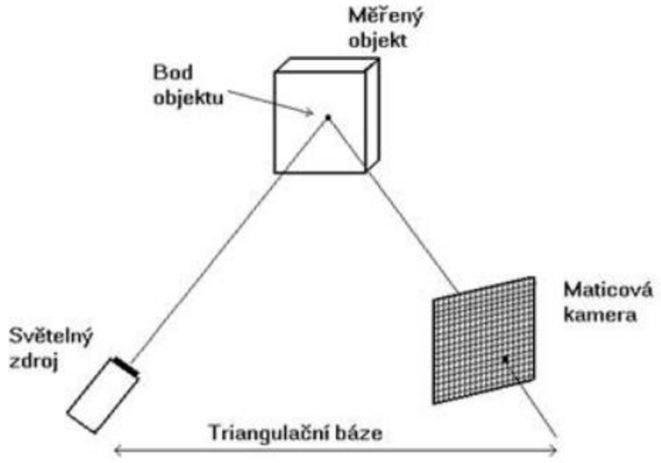
2.1.1. Aktivní triangulace

Technika aktivní triangulace spočívá ve fotogrammetrické rekonstrukci snímaného objektu nasvícením jeho povrchu světelným zdrojem a současným snímáním nějakou kamerou. Zdroj světla spolu s kamerou a osvětleným bodem tvoří tzv. triangulační trojúhelník (Obr. 2-1). Spojnici světelný zdroj – snímač nazýváme triangulační bází. Úhel zdroje světla svíraný s triangulační bází je neměnný, zatímco úhel kamery je určen proměnnou pozicí vysvíceného bodu na daném objektu. Velikost tohoto úhlu a znalost triangulační báze vedou ke snadnému určení Z-ové souřadnice objektu (hloubky) [21].

Vzhledem ke zdroji světla rozlišujeme tři metody triangulace [21]:

- 1D triangulace – světelný paprsek
- 2D triangulace – světelný pruh
- 3D triangulace – strukturovaný světelný svazek

Použitím strukturovaného světelného svazku označíme celý povrch najednou, není potřeba pracného skenování celé scény, jako u 1D nebo 2D varianty.



Obr. 2-1 Triangulační trojúhelník [21]

Nevýhodou aktivní triangulační metody a obecně triangulačních metod jsou konkavity v měřeném objektu, kde nemusí být promítaný bod, pruh, či vzor viditelný, a nelze tedy v tomto místě jednoznačně určit tvar povrchu objektu. Jsou to zkrátka místa, kam sice laser dosvítí, ale kamera nevidí, a obráceně [21].

2.1.2. Měření doby letu

Hloubku nějakého objektu lze také jednoduše stanovit z doby letu τ světelného paprsku. Touto dobou letu se rozumí čas od vyslání signálu senzorem, až po jeho návrat poté, co se odrazí od zaměřeného objektu. Světelný paprsek není podmínkou. Existují měření na základě doby letu infračerveného záření, ultrazvuku apod. Obecně platí základní vztah [21]:

$$z = c\frac{\tau}{2} \tag{2.1}$$

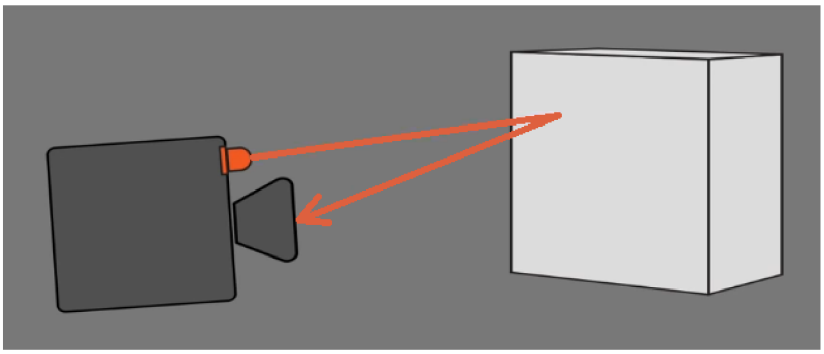
Vztah je platný jak pro techniky měření doby letu, tak pro interferometrické měření vzdálenosti. Nejdůležitějším faktorem je však měření doby letu τ. Jedná se vlastně o dobu letu modulovaného optického signálu, který představuje skupinovou rychlost. Rozlišujeme tři druhy modulace signálů [21]:

- pulsní
- pseudonáhodná
- spojitá

Ačkoliv je princip modulačních a interferometrických metod stejný, používají se modulační metody pouze u aplikací s menšími nároky na rozlišení a přesnost (řádově v cm). Na rozdíl od modulačních metod lze techniky založené na interferometrii aplikovat v širokém rozsahu i u měření s nano metrickou přesností. Hlavní důvod spočívá v tom, že měřicí přístroj založený na principu modulace je složen z optické části a z vysokofrekvenční elektronické části, která přináší do systému časová zpoždění ještě dříve, než je signál mixován a korelován. Je tedy třeba zařadit časově náročnou mechanickou kalibraci nebo kompenzovat chyby dalším referenčním kanálem. U metod interferometrických je toto mixování, korelace a srovnání s referenčním kanálem provedeno již přímo ve fotodetektoru. To má za důsledek téměř nulové zpoždění [21].

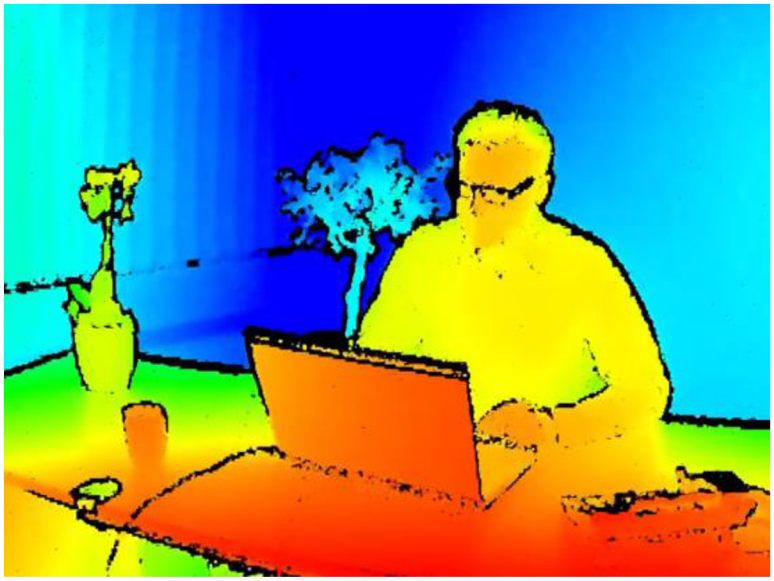
Nevýhoda těchto metod vyplývá z rychlosti světla, která je extrémně vysoká ($c = 300 \cdot 10^6$ m/s). Jsou tedy vysoké nároky na přesnost měřicího zařízení. Další nevýhodou, plynoucí z vlastností signálu, je rychlost zvuku, která může být značně ovlivněna různými podmínkami, jako je např. změna počasí. Za výhodu lze považovat existenci přístrojů, které mohou obě funkce (vysílaní a přijímání) plnit samy. Metoda je výpočetně nenáročná a je spolehlivější, mluvíme-li o větších vzdálenostech, a to na rozdíl od aktivní triangulace, kde se vzdáleností objektu od kamery přesnost klesala [20, 21].

Technologii měření doby letu využívají například TOF kamery (time of flight). Jednoduše řečeno, vysílá TOF kamera nejprve světelnou informaci (optický signál) do okolí, a následně odražený signál pozoruje. Vysílání světelného signálu je realizováno pulsy [22].



Obr. 2-2 Princip TOF kamery [22]

Výhoda TOF kamer spočívá v kompaktní konstrukci, jednoduchém používání, přesnosti cca 1 cm, a velké snímkovací frekvenci (FPS). Mezi nevýhody patří například rozptýlené světlo. Povrch nacházející se velmi blízko zachytávajícímu objektivu kamery dokáže odrazit světelnou informaci na vnitřní stěnu objektivu. Takto pohybující se informace dokáže způsobit odlesk v objektivu. Další problém může vzniknout při snímání vnitřních rohů. Zde se světelná informace z kamery může odrazit do objektivu vícekrát. Problém také vzniká při denním světle. Vysoká intenzita slunečního záření způsobuje rychlou saturaci pixelů senzoru. To způsobí, že kamera nevidí světelný signál odražený od objektu [22].



Obr. 2-3 Hloubková mapa při užití TOF kamery [22]

2.2. Pasivní metody

Principem pasivních metod je zisk dvou snímků ze dvou kamer s různou pozicí. Na rozdíl od aktivních metod zde není využito přídavné (světelné, zvukové) informace. Pasivní metody lze dělit podle vzájemné pozice kamer do dvou případů: obecný vztah kamer, normální postavení kamer [19].

V obecném vztahu kamer mohou být kamery vůči sobě posunuty ve všech třech směrech a mít různě natočenou osu snímání. V tomto případě není k tvorbě 3D snímku přímo získán levý a pravý obraz. Postup je vhodný zejména při vytváření modelu scény. Pro tvorbu 3D videa je vhodnější případ druhý – normální postavení kamer. V tomto případě se pozice objektivů liší pouze o stanovenou horizontální vzdálenost a jejich optické osy jsou rovnoběžné. Tato horizontální vzdálenost se nazývá stereobáze. Na rozdíl od předchozího případu, jsou zde oba stereo snímky použitelným výstupem pro zpracování 3D videa, a pozice všech stejných odpovídajících pixelů se liší pouze o horizontální paralaxu [19].

Pasivní metody jsou vhodné zejména pro kvalitně osvětlené prostory a scény s dostatečným kontrastem. Za dobrých podmínek vykazují menší výpočetní náklady oproti aktivním metodám. Při snímání (skenování) komplexnějšího modelu scény, většinou dvě kamery nestačí. Přidáváním dalších kamer se však zvýší výpočetní náročnost [20].

V této diplomové práci se k tvorbě hloubkové mapy používají kamery dvě. Jedná se tedy o tvorbu hloubkové mapy pasivní metodou.

2.2.1. Pasivní triangulace

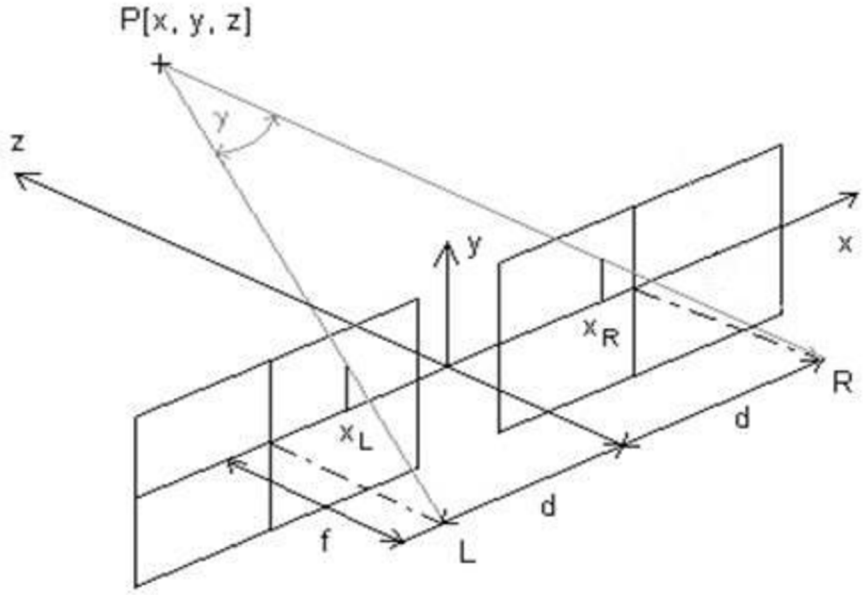
Techniky pasivní triangulace zahrnují různé formy digitální fotogrammetrie. Základní metody jsou [21]:

- více kamer se známou orientací
- více kamer se samokalibrací
- jedna kamera v různých polohách se samokalibrací

Více kamer je vhodné aplikovat u dynamických systémů, kde se využívá znalost relativních poloh nebo samokalibrujících se metod. Jedna pohyblivá kamera (schopná získávat snímky z více pohledů) se používá k zachycení statické scény. Za techniku se samokalibrací lze uvažovat i techniku použitou v diplomové práci, kde byly kamery před samotným snímáním zkalibrovány. Velmi důležité je však dávat pozor, aby se kamery nepohnuly, tedy dodržet jejich konstantní neměnnou vzájemnou polohu a natočení. U techniky se samokalibrací lze určit relativní umístění kamer vzhledem k měřené scéně. Nemusí být tedy dopředu známa poloha kamery. Pro kalibraci kamer se běžně používá kalibrační předmět. Nejčastěji se jedná o dokonale rovnou šachovnici s přesnými rozměry jednotlivých černo-bílých čtverců. Následující odstavec bude věnován nejčastěji používané technice, kterou je **stereovidění** [21].

Technika stereovidění se snaží napodobit smysl lidského zraku. Dvě kamery s rovnoběžnými optickými osami plní funkci lidských očí. Pro zajímavost lze uvést, že lidské oči jsou od sebe vzdáleny průměrně 63 mm [23]. Tyto dvě kamery pak získávají dva stereoskopické snímky, což jsou dva perspektivní obrazy. Úhel γ (Obr. 2-4) se nazývá úhlová paralaxa. Je to úhel, který svírají dva sdružené paprsky. Bod P představuje bod snímaného objektu (scény), body L a R ohniska kamer. Z obrázku je patrné, jak se velikost úhlové paralaxy mění se vzdáleností bodu od pozorovatele. Jestliže hodnota paralaxy klesne pod určitou mez, nemusí se prostorové vidění náležitě uplatnit. Obrázek (Obr. 2-4) znázorňuje

nejjednodušší případ, kdy jsou optické osy kamer rovnoběžné s osou *z* souřadnicového systému (hloubka), kamery mají stejné ohniskové vzdálenosti, a obě leží ve stejné rovině z = 0.



Obr. 2-4 Stereoskopické snímky [19]

Souřadnice bodu P lze určit pouze v případě, kdy se podaří najít bod P na jednom snímku a jemu odpovídající bod P na druhém snímku. Platí pak tyto vztahy [19]:

$$x = x_L \frac{2d}{x_L - x_P} \tag{2.2}$$

$$y = y_L \frac{2d}{x_L - x_P} \tag{2.3}$$

$$z = \frac{2df}{x_L - x_P} - f \tag{2.4}$$

Kde d je vzdálenost optické osy kamery od počátku souřadnicového systému, f je ohnisková vzdálenost kamery, souřadnice x_L a x_P definují snímaný bod v obrazové rovině z = 0. Rozdíl (x_L - x_P) představuje horizontální paralaxu.

Jestliže nejsou dodrženy dané požadavky, je nutno převést tyto případy na nejjednodušší. K tomu slouží různé korekční vztahy. Korespondenční problém automatického nalezení bodu v obrazech obou kamer je zjednodušen tím, že odpovídající body musí ležet na epipoláře (Obr. 2-6).

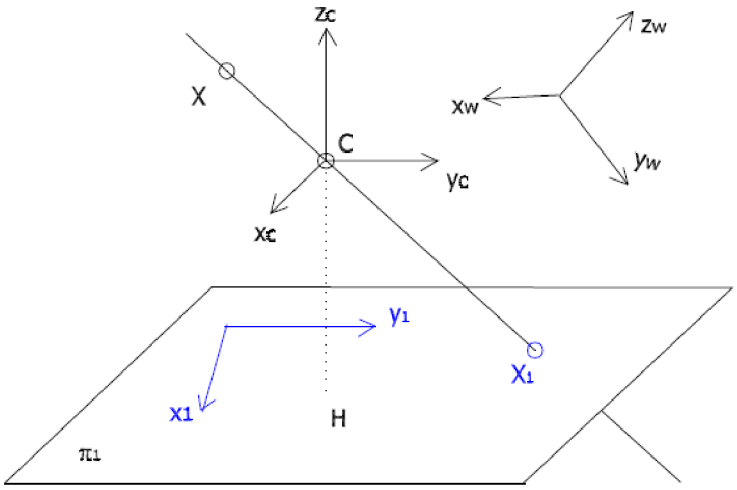
2.3. Prostorová rekonstrukce

Kapitola o epipolární geometrii, a to včetně jejích podkapitol, poskytuje teoretický přehled přístupu k rekonstrukci prostorové scény. Obecně existují dva přístupy k rekonstrukci prostorové scény. První možnost spočívá v určení projekčních matic kamer (získání vzájemných vztahů promítacích paprsků a pixelových souřadnic vzhledem k pevně danému repéru). Druhý přístup rekonstrukce prostorové scény je využití projektivity snímků, kterou určíme detekcí vzájemně si odpovídajících bodů. Výhodou je menší počet požadovaných parametrů k odhadnutí (netřeba znát vnitřní parametry kamery).

V této diplomové práci se omezíme pouze na případ první. Bude zde popsán vztah mezi souřadnicemi pozorovaného objektu a pixelovými souřadnicemi snímku, reprezentovaného projekční maticí kamery.

2.3.1. Projekční matice

Za předpokladu, že 3D scéna je ze středu C středově promítána na průmětnu π, počítáme geometrický model kamery. Jsou uvažovány tři souřadné systémy. Repér, pevně daný v obecné poloze vůči průmětně W < O_w, x_w, y_w, z_w > , druhý je repér objektivu kamery, jehož optická osa C < C, x_c, y_c, z_c > splývá s osou *z*, a třetí jsou souřadnice obrázku I < O, x_I, y_I >, které nemusí být vždy ortogonální [24].



Obr. 2-5 Projekční matice kamery [24]

Následující matice popisuje středové promítání X → X_I; $(w, x, y, z)^T \rightarrow (w_I, x_I, y_I)^T$, tedy bod X se zobrazuje do promítací roviny π_I:

$$P_{persp} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & -f & 0 & 0 \\ 0 & 0 & -f & 0 \end{pmatrix} = \begin{pmatrix} 0 & \tilde{P}_{persp} \end{pmatrix} \tag{2.5}$$

Parametr *f* je vzdálenost optického středu od průmětny, na obrázku (Obr. 2-5) je to vzdálenost |CH|. Tento parametr spolu se souřadnicí hlavního bodu patří k vnitřním parametrům kamery. Při ukládání obrazové informace do pixelových souřadnic je uvažováno celkem pět vnitřních parametrů kamery. Určují posunutí počátku, změnu měřítka ve směru os a afinní transformaci do neortogonální báze obrázku [24].

Vnější parametry kamery – posunutí $\vec{t}$ a ortogonální matice rotace R ∈ O(3,R), definují transformaci mezi pevně daným repérem W a repérem kamery C. Matice perspektivní projekce spolu s maticemi vnější a vnitřní kalibrace tvoří dohromady konečnou projekční matici P [24]:

$$P = \begin{pmatrix} 1 & 0 & 0 \\ x_0 & a & b \\ y_0 & 0 & c \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & -f & 0 & 0 \\ 0 & 0 & -f & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \vec{t} & R \end{pmatrix} = \tag{2.6}$$

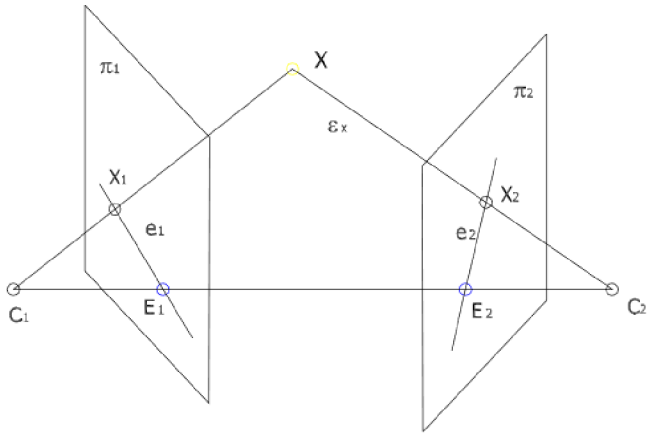
$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & -af & -bf & x_0 \\ 0 & 0 & -cf & y_0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \vec{t} & R \end{pmatrix} = (0\ K) \begin{pmatrix} 1 & 0 \\ \vec{t} & R \end{pmatrix} = K(\vec{t}\ R)$$

Matice K, typu 3 × 3, se nazývá matice kalibrace kamery. Vztah mezi homogenní souřadnicí bodu v prostoru a jeho obrazu v projektivní rovině lze vyjádřit jako [24]:

$$X_1 = PX\,; \quad P = K(\vec{t}\ R) \tag{2.7}$$

2.3.2. Epipolární geometrie

Ke stanovení epipolárních podmínek je využito vlastností dvoustředového promítání. Pro tento případ uvažujme dvě kamery, s různými maticemi kalibrace K_1, K_2, pozorující tutéž scénu (Obr. 2-6) [24].



Obr. 2-6 Epipolární geometrie [24]

Promítací (epipolární) rovina, tvořená promítacími paprsky daného bodu X, protíná průmětny ve sdružených přímkách (epipolárách). Vztah mezi průměty X_1 a X_2 bodu X odvodíme z rovnice (2.7) a za předpokladu, že pevný souřadný systém splývá se souřadným systémem první nebo druhé kamery. Vztah je popsán rovnicemi [24]:

$$X_1 = P_1X\,; \quad P = K(0\ I)X \tag{2.8}$$

$$X_1 = P_2X\,; \quad P = K(\vec{t}\ R)X \tag{2.9}$$

Vektor $\vec{t}$ a ortogonální matice rotace R určují transformaci mezi objektivy kamer. Známe-li matice kalibrace, lze normalizované souřadnice $\hat{X}_1$, $\hat{X}_2$ psát ve tvaru [24]:

$$X_1 = K_1 \hat{X}_1 \tag{2.10}$$

$$X_1 = K_2 \hat{X}_2 \tag{2.11}$$

Vztah mezi normalizovanými souřadnicemi $\hat{X}_1$, $\hat{X}_2$ je dán rovnicí [24]:

$$\hat{X}_2^T \cdot t_M R \cdot \hat{X}_1 = 0 \tag{2.12}$$

Jedná se o vektorový součin mezi normalizovanými obrazovými souřadnicemi a esenciální maticí E, kde t_M je asymetrická matice rozměru 3 × 3 vektoru posunutí [24].

$$E = t_M R \tag{2.13}$$

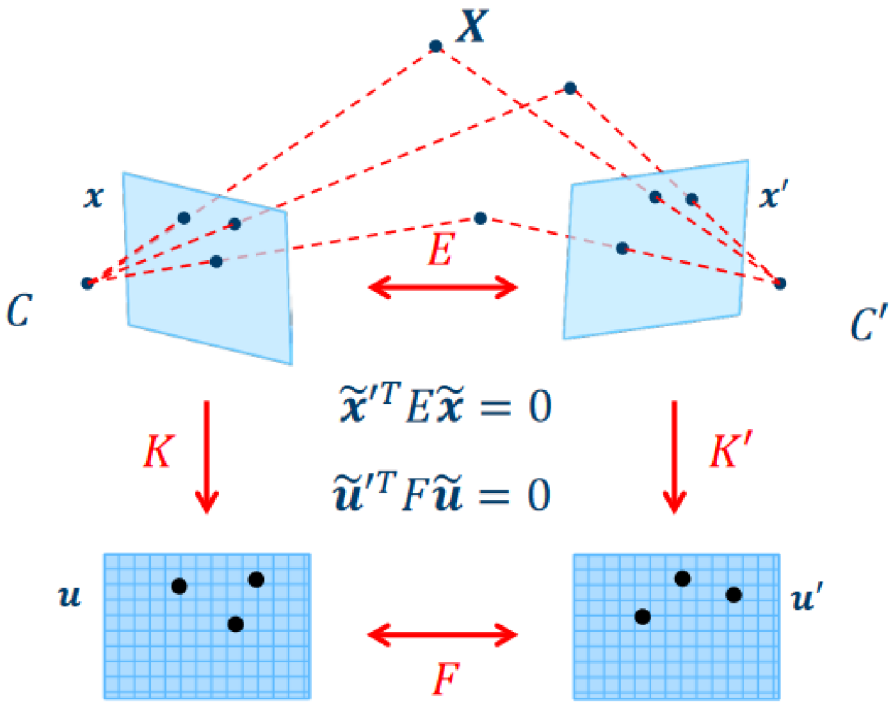
Uvažujme matice kalibrace K_1, K_2. Dosazením rovnic 2.10 a 2.11 do rovnice 2.12 získáme následující vztah [24]:

$$\hat{X}_2^T \left(K_1^{-T} E K_2^{-1}\right) X_1 = 0 \tag{2.14}$$

Prvek $(K_1^{-T} E K_2^{-1})$ v rovnici 2.14 nazýváme též fundamentální matice F [24]:

$$F = K_1^{-T} E K_2^{-1} \tag{2.15}$$

Esenciální matice E a fundamentální matice F jsou matice hodnosti 2, a rozměru 3 × 3. Fundamentální matice F je pro dvojici snímků stanovena jednoznačně. Dále určuje tato matice dvojici projekčních matic kamer P_1 a P_2. Přesný odhad fundamentální matice je pro stereo analýzu velmi důležitým úkolem. Pro její výpočet je potřeba celkem sedm parametrů [24].



Obr. 2-7 Esenciální matice E a fundamentální matice F [25]

3. Stereo kamery v robotice

Cílem této kapitoly bylo poskytnout přehled běžně používaných stereo kamer v robotice. Poslední část kapitoly je věnována řešení na Raspberry Pi.

Stereo vidění lze v dnešní době aplikovat v různých odvětvích. Nejvíce se tato technologie uplatňuje v automobilovém průmyslu, a to jak při samotné výrobě jednotlivých komponent, tak i ve vozidlech (autonomní vozy). Při výrobě mluvíme například o robotech, kteří jsou schopni určit polohu náhodně rozmístěných dílů na pásových nebo jiných dopravnících. Dalším příkladem může být určení polohy dílů při paletování nebo detekce poruch na povrchu materiálu (3D skenery) [26].

Stereovidění lze uplatnit také v armádě. Armáda USA začíná instalovat stereo kamery na některé bojová vozidla, ve spolupráci s Honeywell Aerospace. Jedná se o vozidla, u kterých bývá obvykle za jízdy zavřený poklop. Důvodem je, že řidič má běžně k dispozici pouze periskop a zrcátka, a tím pádem omezený výhled na okolní situaci. Uplatněním stereo vize, se zvýší bezpečnost členů posádky a efektivita vykonávání různých bojových nebo průzkumných misí. Systém pouze doplňuje již existující 360° senzory „prostorové představivosti“. Systém spočívá v umístění stereo kamer na předek vozidla, který 3D informaci zpracuje a promítne do levého a pravého oka pomocí dvojice holografických optických prvků, což umožňuje řidiči vnímat hloubku a vykazovat široké zorné pole, aniž by trpěl nevolností, nebo nepřiměřeným namáháním očí [27].

V obou zmíněných případech spočívá hlavní účel stereo vize v detekci objektů, zkoumání okolní scény, sledování nebo také v navigaci.

3.1. Kinect v1

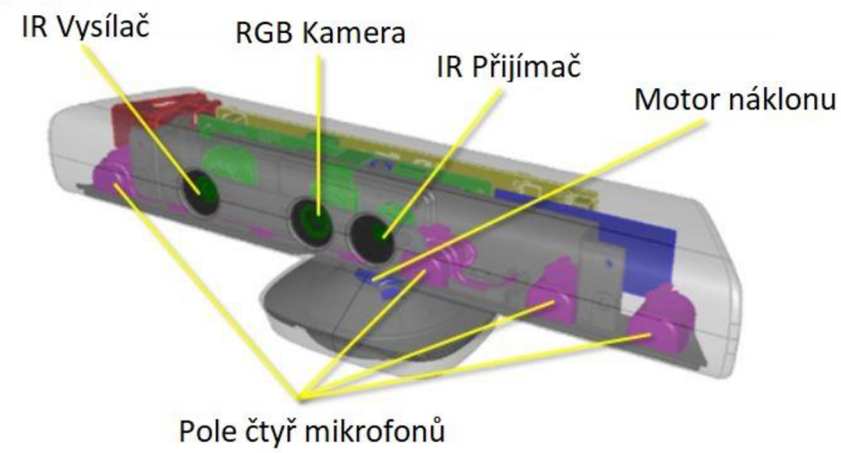
Toto zařízení našlo svoje uplatnění jak v herním průmyslu, tak v robotice. Mluvíme-li o prvním případě snímá tato stereo kamera hráčovy pohyby. Nejsou tím pádem potřeba vstupní zařízení jako myš, klávesnice, ovladač apod. Jedná se o hraní celým tělem. To znamená, že vstupy jsou realizovány pomocí různých gest. Zajímat nás bude především druhý případ, a to využití v robotice.

Obr. 3-1 Kinect v1 [28]

Společnost Microsoft začala vývoj zařízení Kinect jako tajný projekt v roku 2006, a to jako reakci na Nintendo Wii. O dva roky později zahájila projekt pod názvem „Project Natal“. Cílem projektu bylo vyvinout zařízení, které bude schopno rozlišit hloubku prostorové scény, sledování pohybu, rozpoznání tváří a rozpoznání řeči. V roce 2010 byl vydán Kinect pro herní konzoli Xbox. Jednalo se, a pořád tomu tak je, o průlomovou technologii v herním světě [29].

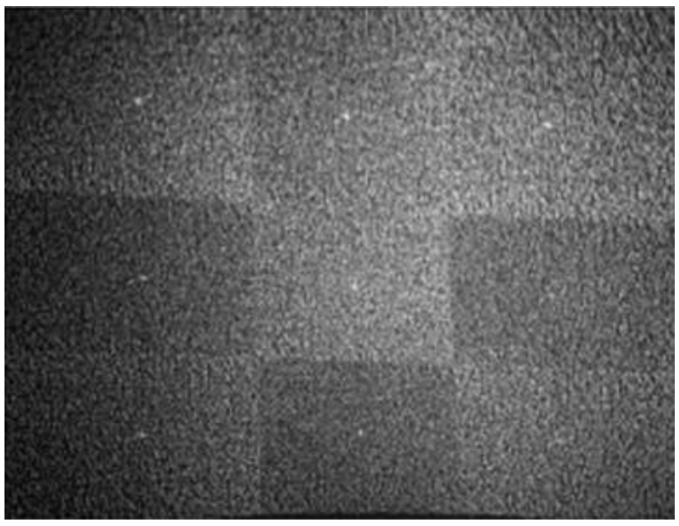
Pro zajímavost uvedeme, že se toto zařízení dostalo začátkem roku 2011 do Guinnessovy knihy rekordů, a to jako nejrychleji se prodávající herní periferie. V prvních 60 dnech od uvedení na trh (04.11.2010) se po celém světě v průměru prodávalo 133 tisíc kusů denně. Tento rekord drží dodnes [30].

Součástí Kinect jsou čtyři důležité prvky (Obr. 3-2), které společně detekují pohyb a zvuk. Infračervený přijímač a vysílač slouží především k tvorbě hloubkové mapy. Bližší informace o prostorové scéně může Kinect získat užitím RGB kamery, nebo mikrofonů. Součástí samostatného zařízení je také motor uložený v podstavě zařízení. Kinect je připojen k PC prostřednictvím USB rozhraní ve verzi 2.0. Potřebuje také externí zdroj napájení, protože USB porty neposkytují dostatečný výkon [29].



Obr. 3-2 Kinect v1, hardware [31]

Vysílač IR je zařízení, které Kinect využívá k projekci infračervených vln, které jsou použity při výpočtu hloubky prostorové scény. Vysílá uspořádaný strukturovaný vzor IR bodů (Obr. 3-3), a to při vlnové délce 830 nm. Vzor je uspořádán jako 9 sub-vzorů (3 × 3 rozložení), kde v každém z devíti sub-vzorů je uprostřed bod s největší intenzitou. Jestliže se tento bod potká s nějakým objektem, ztratí vzor svoje uspořádání (dopadnou na objekt) a na základě této deformace je IR přijímačem analyzovaná hloubka (vzdálenost mezi objektem a snímačem) [29].



Obr. 3-3 Infračervený vzor [29]

Jako přijímač IR záření slouží CMOS kamera, která je vybavena filtrem, propouštějícím pouze infračervené záření. Tato kamera je zařízením, které Kinect používá k zachytávání odraženého IR signálu, s jehož pomocí je pak rekonstruována prostorová scéna. Dokáže pracovat při rozlišení 1280×960 při rychlosti 12 snímků za sekundu. Pro nejběžnější aplikace se však doporučuje 640×480 při rychlosti 30 snímků za sekundu. Dokáže rozpoznat 2048 úrovní hloubky. Senzor má však omezenou funkčnost při odrazech slunečního světla od snímaných objektů [29].

RGB kamera funguje při stejných rozlišeních a za stejných rychlostí, jako CMOS kamera (IR přijímač). Uplatňuje se při získávání detailů snímané osoby, nebo objektů prostorové scény [29].

Součástí Kinect jsou také 4 mikrofony, umístěné lineárně vedle sebe podél spodní hrany zařízení. Jsou vybaveny akustickou lokací zdroje, dokážou tedy určit směr, odkud zvuk přichází. Kinect je dále vybaven trojosým akcelerometrem v rozsahu 2g. Kamery Kinectu poskytují úhel záběru 57,5 stupňů horizontálně a 43,5 stupňů vertikálně. Tento úhel může být přizpůsoben pomocí otočného motorku ve stojanu. Otáčí celým tělem Kinectu kolem horizontální osy v rozmezí -27 až 27 stupňů. Natočení motoru lze buď manuálně nastavit v PC, nebo automaticky podle výšky snímané osoby [29].

Kinect disponuje funkcí tzv. skeleton-tracking. Na zaznamenaném člověku dokáže najít 20 definovaných bodů (hlava, chodidla, ruce, klouby), které potřebuje k sestavení kompletního skeletu člověka. Kinect ve verzi 1 dokáže najednou zpracovávat 2 postavy.

Schopnost vidět okolí výrazně přispívá k autonomní povaze mobilních robotů. Jedním ze způsobů může být například současná lokalizace a mapování (SLAM). Senzory potřebné k uskutečnění takové úlohy jsou buď drahé a těžkopádné, nebo levné a nespolehlivé. Protože je Kinect vybaven IR vysílačem a přijímačem, je to skvělý zobrazovací nástroj, a představuje oproti složitému a drahému vybavení různých senzorů obrovskou úsporu. Jedním z praktických příkladů mobilního robota s autonomní povahou je TurtleBot (Obr. 3-4) od Willow Garage. Je to přizpůsobitelná mobilní robotická platforma, která jezdí na platformě iRobot Create a využívá otevřenou platformu ROS. TurtleBot osazen Kinectem, využívá Kinect k vidění okolí ve 3D k detekci a sledování lidí. Má také schopnost fotografovat okolí kolem sebe a vytvářet 360° panoráma [28].

Obr. 3-4 TurtleBot na podvozku iRobot, osazen zařízením Kinect [32]

3.2. Kinect v2

Jedná se o vylepšenou verzi Kinectu v1. Na obrázku níže (Obr. 3-5) je vyobrazen vedle původní v1 verze. Na první pohled je větší než jeho předchůdce, což lze považovat za menší nevýhodu.

Obr. 3-5 zleva: Kinect v2, Kinect v1 [34]

Oproti původní verzi se Kinect v2 chlubí vyšším rozlišením. Dosahuje až 1920×1080 (Full HD) při frekvenci 30 snímků za sekundu. Zorný úhel byl také vylepšen. Obě zařízení fungují spolehlivě i na Windows 10 OS a mohou pracovat společně na jednom PC. Kinect v2 přichází také s rozhraním USB ve verzi 3.0 [33, 34].

Jeden z hlavních rozdílů je způsob rekonstrukce prostorové scény. Zatím co Kinect v1 využívá infračervené vzory, Kinect v2 využívá metodu měření doby letu (time of flight, kapitola 2.1.2). Tato metoda je stabilnější, precizní a méně náchylná na interference (při použití více Kinectů) [33, 34].

Další novinkou je vylepšená funkce skeleton-tracking. Oproti původní verzi dokáže pojmout až 6 lidí v jednom záběru, každý člověk je reprezentován 26 body, včetně palců. Původní verze zvládla pouze 2 lidi, 20 bodů a žádné palce. Porovnání obou zařízení z hlediska funkce skeleton-tracking je na obrázku níže (Obr. 3-6) [33, 34].

Obr. 3-6 Porovnání funkce skeleton-tracking [34]

Z obrázku je na první pohled patrné, že Kinect v1 nedokáže zachytit třetí osobu. Dále jsou vidět u Kinectu v2 palce na rukou a ramena vypadají reálněji.

Následující tabulka (Tab. 3-1) přesně specifikuje parametry obou zařízení a provede vzájemné porovnání:

Parametr	**Kinect v1**	**Kinect v2**
Typ	Kinect for Windows 1	Kinect for Windows 2
RGB Kamera	640×480 @30fps	1920×1080 @30fps
IR kamera	320×240	512×424
Maximální hloubka	~4.5 m	~4.5 m
Minimální hloubka	40 cm (v módu „near“)	50 cm
Horizontální zorné pole	57 °	70 °
Vertikální zorné pole	43 °	60 °
Motor naklánění	ano	ne
Body skeletu člověka	20 bodů	26 bodů
Maximální počet skeletů	2	6
USB port	2.0	3.0
Podporovaný OS	Win 7, Win 8, Win 10	Win 7, Win 8, Win 10

Tab. 3-1 Porovnání Kinect v1 a Kinect v2 [33]

3.3. Bumblebee 2 a Bumblebee XB3

Stereo kamera Bumblebee 2 (Obr. 3-7) je vydána společností Point Grey Research. Jedná se o dvou senzorovou stereo kameru. Vzájemná vzdálenost obou objektivů je 12 cm. Poskytuje rovnováhu mezi kvalitou 3D dat, rychlostí zpracování obrazu, velikostí a cenou. Oproti původní Bumblebee kameře disponuje Bumblebee 2 téměř dvojnásobným počtem snímků za sekundu a GPIO konektorem pro externí ovládání a funkčnost blesku [35].

Obr. 3-7 Stereo kamera Bumblebee 2 [35]

Bumblebee XB3 (Obr. 3-8) je tří senzorová stereo kamera navržená za účelem větší flexibility a přesnosti. Disponuje 1,3 megapixelovými senzory (kamerami), a má dvě vzájemné vzdálenosti objektivů, jelikož jsou tři. Lze použít 12cm pro kratší vzdálenosti, a 24cm pro delší vzdálenosti. Z toho vyplývá i zvýšená přesnost zařízení a zmíněná flexibilita [35].

Obr. 3-8 Stereo kamera Bumblebee XB3 [35]

Kamery jsou předem kalibrovány pro zkreslení objektivu a nesprávné vyrovnání kamery. Zařízení komunikuje skrze IEEE 1394b/FireWire sběrnici. Každá Bumblebee kamera přichází s technologií FlyCapture v1.8 SDK, která umožní samotné získávání obrazu a ovládání kamery z externího zařízení (komunikaci) a se sadou knihoven Triclops SDK, které provádějí rektifikaci snímků a stereo zpracování [35].

Obr. 3-9 Stereo kamera Bumblebee 2 použitá s mobilním robotem [36]

Technologie Triclops Stereo SDK obsažená v obou modelech Bumblebee, poskytuje vysokorychlostní stereo jádro, které je schopno provádět SAD (Sum of Absolute Differences) stereo korelaci. Tato metoda je známa svojí rychlostí, robustností a jednoduchostí. Dokáže generovat velký počet úrovní hloubky [37].

FlyCapture SDK je sada pro vývoj softwaru umožňující získávání obrazu a ovládání samotného zařízení. Je obsažen ve všech modelech Bumblebee a je kompatibilní s Microsoft Windows OS. Obsahuje ovladače kamery, různé knihovny s API, demo programy a C/C++ vzorový zdrojový kód [37].

Následující tabulka (Tab. 3-2) zobrazuje základní specifikace všech modelů Bumblebee stereo kamer.

Parametr	Bumblebee 2		Bumblebee XB3
Kamera	Sony 1/3“ progressive scan CCD		
	2×		3×
Rozlišení	640×480 @48fps	1024×768 @20fps	1280×960 @16fps
Objektiv, ohnisková vzdálenost [mm]	2,5mm (100° HFOV) 3,8mm (65° HFOV) 6mm (43° HFOV)		3.8mm (70° HFOV) 6mm (50° HFOV)
Kalibrace	Předkalibrováno s chybou 0,1 RMS pixel error		
Vstupní napětí	8–30 V		

Tab. 3-2 Základní parametry stereo kamer Bumblebee [35, 37]

3.4. ZED stereo kamera

ZED je 3D kamera pro hloubkové snímání, sledování pohybu a 3D mapování v reálném čase. Je ideální pro použití v robotice, virtuální realitě (VR) a pro chytrou analýzu. Oproti většině stereo kamer vyniká ZED na delší vzdálenosti, a to až do 25 m. Navíc funguje spolehlivě i v exteriéru, na rozdíl od levnějších variant, které jsou spíše vhodné do vnitřních prostor. ZED stereo kamera zachytává dva dokonale synchronizované snímky scény a poskytuje je nadřazené stanici ve vysokém rozlišení, prostřednictvím USB rozhraní ve verzi 3.0. Použitím *Jetson Nano* počítače lze ZED přeměnit na Ethernet kameru. Obě kamery jsou vzájemně předkalibrovány. Video poskytnuté nadřazené stanici touto stereo kamerou je ihned použitelné pro zpracovávání hloubkové mapy [38].

Nyní přejdeme ke konkrétním parametrům kamery ZED, které jsou k nahlédnutí v tabulce (Tab. 3-3).

Parametr	Hodnota
Rozlišení	2208×1242 @15fps 1920×1080 @30fps 1280×720 @60fps 672×376 @100fps
RGB senzor	1/3“ 4MP CMOS
Ohnisková vzdálenost	2,8 mm
Vertikální zorné pole	max 60 °
Horizontální zorné pole	max 90 °
Diagonální zorné pole	max 100 °
Rozhraní	USB 3.0
Hloubková vzdálenost	0,5 – 25 m
Hloubková přesnost	< 4% do 15m vzdálenosti
Rozměry	175 × 30 × 33 mm
Napájení	USB, 5 V/380 mA

Tab. 3-3 Vybrané parametry stereo kamery ZED [38]

Kompatibilními operačními systémy jsou Windows 10, 8, 7, Ubuntu 18, 16, Debian, Jetson L4T. Podporované platformy: Unity, Unreal, OpenCV, ROS, TensorFlow, MATLAB, Python [38].

K práci se ZED kamerou v prostředí ROS, je zapotřebí Operační systém Ubuntu 16.04, nebo novější. Je třeba mít nainstalovanou aplikaci ZED SDK a k ní příslušnou sadu balíčků nazývaných CUDA. Doporučená distribuce ROSu je ROS Melodic. Balíčky pro framework ROS, k práci se ZED kamerou lze stáhnout z GitHub repositáře. Jedná se o tyto dva: *zed-ros-wrapper* a *zed-ros-examples* [39].

Po nainstalování příslušných balíčků, lze uzel ZED kamery spustit užitím nástroje roslaunch [39]:

```
~$ roslaunch zed_wrapper zed.launch
```

Data lze poté jednoduše zobrazit v grafickém prostředí *rviz*. Práce se ZED kamerou je podobná práci s jakoukoliv jinou kamerou (uzel publikuje stejná témata).

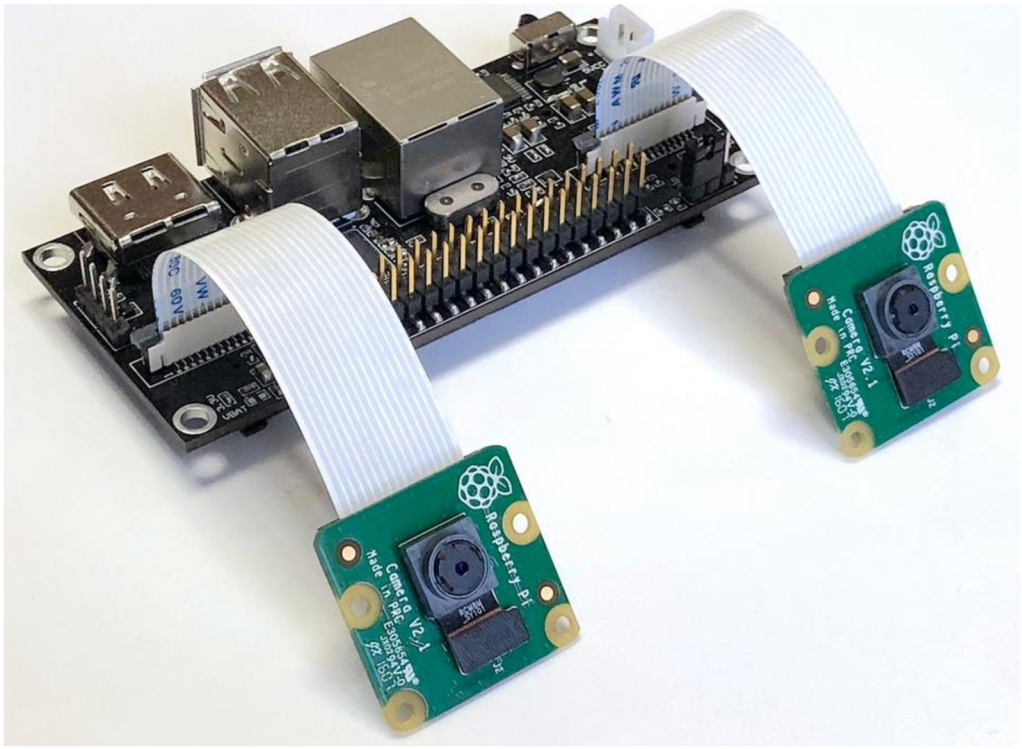
Obr. 3-10 Stereo kamera ZED [38]

3.5. Řešení stereo vize užitím Raspberry Pi

Tato podkapitola je věnována řešení Stereo Pi. Jedná se o spolehlivou a levnou technologii pro tvorbu hloubkové mapy.

Jedná se o nápad ruské firmy Virt2Real. Stereo Pi (Obr. 3-11) je open-source stereo kamera založená na Raspberry Pi. Dokáže zachytit, uložit, zpracovat a přehrát stereo videa a obrázky v reálném čase. Stereo Pi otevírá nespočet možností v robotice, virtuální realitě (VR), strojovém vidění, dronech, nebo třeba v tvorbě panoramatických videí [40].

Stereo Pi je nosič pro desku Raspberry Pi Compute Module (CM) a je kompatibilní se všemi dostupnými variantami tohoto modulu: od CM1 po CM3, včetně CM3+ modulu. Výpočetní modul je na desce Stereo Pi umístěn odspodu. S Raspberry Pi jako s jádrem běží Stereo Pi přirozeně na Raspbianu. Rozměry této desky jsou 90 mm na délku a 40 mm na šířku. Výška je u standardní edice 23 mm, u „slim“ edice 15 mm. Jako vstup videa jsou použity dva 15-pinové CSI-2 kamerové moduly, výstup je realizován klasickým HDMI rozhraním. Kompatibilními kamerami jsou Raspberry Pi kamery V1 (OV5647 senzor) a V2 (Sony IMX 216 senzor), a všechny kamery, disponující daným senzorem. Deska disponuje 40 GPIO piny, 2× USB typu A, a jeden USB pin header. Připojení k síti je zabezpečeno Ethernetem RJ45. Jako úložiště slouží slot pro SD kartu. Napájení je stejné jako u Raspberry Pi, pouze s rozdílem manuálního spínače. Stereo Pi je plně kompatibilní s Raspbian OS a má plnou podporu pro Python. Funguje dobře i s Ubuntu Mate [40].



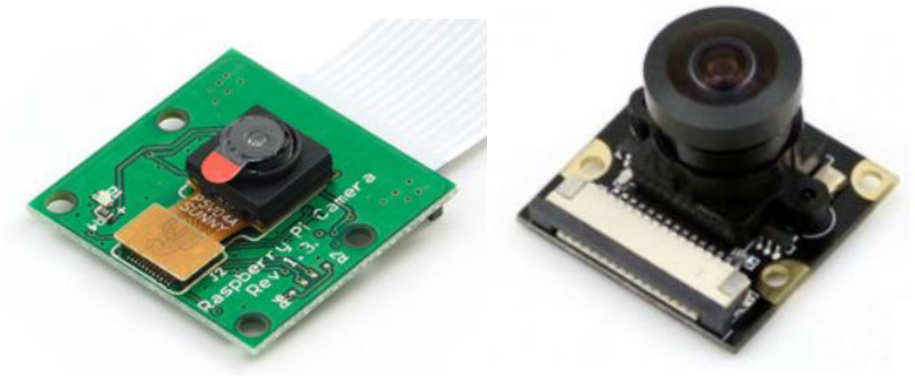
Obr. 3-11 Stereo Pi s připojenými Raspberry Pi kamerami V2 [41]

Obr. 3-12 Stereo Pi; zleva: slim edice, standardní edice [40]

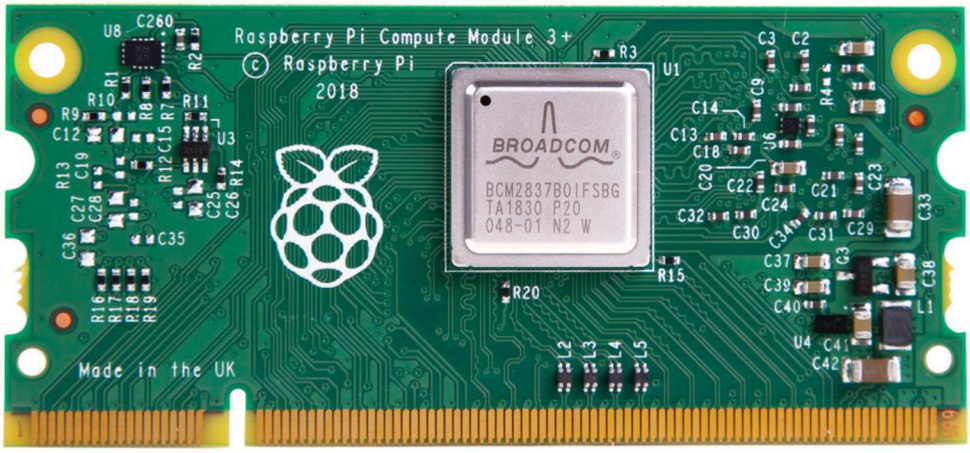
Nyní budou v tabulce (Tab. 3-4) porovnány dvě nejpoužívanější a zároveň doporučeny kamery pro Stereo Pi.

Parametr	**Raspberry Pi V1**	**Waveshare Raspberry Pi**
Typ senzoru	OmniVision OV5647 (5 MPx)	
Rozlišení	až do 1920 x 1080p @ 30 FPS	
Úhel pohledu	54 ° × 41 °	160 ° diagonálně
Rozměry nosiče	25 mm x 24 mm	

Tab. 3-4 Porovnání dvou kamer pro Raspberry Pi / Stereo Pi [40]



Obr. 3-13 kamera Raspberry Pi V1, kamera Waveshare model G [42, 43]



Obr. 3-14 Raspberry Pi Compute Module 3+ [44]

4. Hardware

Tato kapitola popisuje nejprve použitý hardware jednotlivě. Dále je zde objasněna architektura experimentální soustavy, a to včetně fotodokumentace. Experimentální soustava, o které kapitola pojednává, byla použita k tvorbě hloubkové mapy.

4.1. Použitý Hardware

4.1.1. Microsoft LifeCam HD-3000

Jedná se o běžně dostupnou webkameru firmy Microsoft vhodnou například k uskutečnění videohovoru. Komunikuje prostřednictvím sběrnice USB ve verzi 2.0.

Experimentální soustava obsahovala tyto kamery dvě. Obě byly stacionárně umístěny na čelním skle auta, aby se přibližně „dívaly“ stejným směrem. Kamery poskytly surový obraz, který byl dále zpracován až k výsledku. V tabulce (Tab. 4-1) lze nalézt parametry použité kamery.

Parametr	Hodnota
Max. podporované rozlišení	1280 × 720 @30fps
Typ senzoru	CMOS, 1,3 MPx
Automatické nastavení expozice	ano
Rozhraní	USB 2.0

Tab. 4-1 Parametry kamery Microsoft LifeCam HD-3000 [45]

Obr. 4-1 Microsoft LifeCam HD-3000 [45]

4.1.2. Sony Playstation 3 Eye Camera with EyeCreate (PS3)

Tato výkonná kamera byla použita jako třetí kamera v experimentální soustavě. Hlavním účelem bylo otestovat ROS jako distribuovaný systém. Kamera komunikuje prostřednictvím sběrnice USB 2.0.

Kamera byla umístěna ve vozidle, nezávisle od ostatních. Byla připojena na vlastní jednodeskový počítač Raspberry Pi. Poskytovala obraz, za účelem testování zbylých dvou kamer. Sledovala se snímkovací frekvence (FPS) ostatních dvou kamer, produkujících

hloubkovou mapu před připojením třetí kamery (včetně příslušné platformy) a po jejím připojení. V tabulce (Tab. 4-2) lze nalézt parametry použité kamery.

Parametr	Hodnota
Max. podporované rozlišení	640 × 480 @60fps
Fixed focus	ano
Dynamický rozsah	8–10bit
Rozhraní	USB 2.0

Tab. 4-2 Parametry kamery Sony PS3 Eye Camera [46]

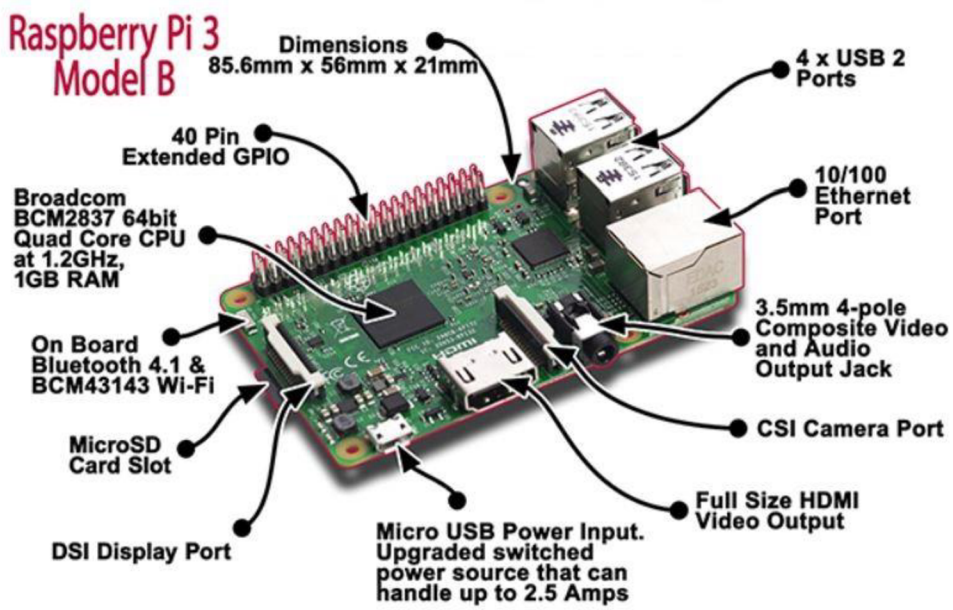
Obr. 4-2 kamera Sony PS3 Eye Camera [46]

4.1.3. Raspberry Pi 3 Model B

Jedná se o jednodeskový počítač vyvinutý britskou nadací Raspberry Pi Foundation. Na trh byl uveden v únoru roku 2016. Lze jej použít jako plnohodnotný počítač. Cena tohoto modelu se pohybuje kolem 1000 Kč. Standardním operačním systémem je Raspbian. Raspberry Pi 3 Model B využívá čtyřjádrového procesoru Broadcom BCM2837 64bit, o taktu 1.2GHz, a operační paměti 1GB. Dále disponuje micro SD portem, pro uložení pevného disku – SD paměťové karty o minimální velikosti 4GB. Raspberry Pi má širokou škálu možností. Vděčí za to zabudovanému Wi-Fi modulu a nízkoenergetickému Bluetooth modulu. Obsahuje i Fast Ethernet 100 Mbit/s pro připojení k počítačové síti. K nalezení jsou zde i 4 USB porty ve verzi 2.0, pro připojení různých vstupních a výstupních zařízení, jako jsou klávesnice, myš, nebo třeba USB kamera. Dále je zde přítomno 40 GPIO pinů, standardní HDMI rozhraní pro připojení k monitoru nebo TV, CSI port pro Raspberry Pi kameru. Raspberry Pi je napájeno ze zdroje poskytujícího maximálně 3,1 A stejnosměrného proudu. Napájení je realizováno micro-USB portem. Jednodeskový počítač Raspberry Pi 3 Model B je znázorněn na obrázku (Obr. 4-3) [47].

Při práci byly použity tři počítače Raspberry Pi, kde ke každému z těchto počítačů byla prostřednictvím USB rozhraní připojena kamera. Napájení Raspberry Pi bylo realizováno vytvořeným napájecím modulem, popsaným v podkapitole (Kap. 4.1.6), který poskytuje 3 USB porty pro napájení Raspberry Pi a jeden souosý konektor pro napájení Wifi routeru. Zdroj poskytuje pro stanovený účel dostatečný proud. K tvorbě hloubkové mapy ze zpracovaných obrazů sloužil počítač Raspberry Pi jako zprostředkovatel obrazu získaného

z kamery do nadřazeného počítače. Podobně tomu tak bylo i u třetí PS3 kamery, ale s tím rozdílem, že tato kamera sloužila pouze k poskytnutí jejího obrazu. Všechny Raspberry Pi sdílejí s nadřazeným počítačem společnou lokální off-line síť. Tři jednodeskové počítače Raspberry Pi společně s nadřazeným laptopem jsou pomocí Ethernetového kabelu připojeny k Wi-Fi routeru.



Obr. 4-3 Jednodeskový počítač Raspberry Pi 3 Model B [48]

4.1.4. TP-Link TL-WR840N

Jedná se o bezdrátový N router. Rychlost bezdrátového přenosu činí 300 Mb/s, což je ideální pro aplikace náročné na šířku pásma i pro běžné činnosti. Podporuje 4 režimy: router, extender dosahu, přístupový bod a WISP. V této diplomové práci pracoval router v režimu přístupového bodu. Obsahuje napájecí port (9 V; 0,6A DC), jeden WAN port a čtyři LAN porty [49].

Obr. 4-4 Wi-Fi router TP-Link TL-WR840N [49]

Jelikož je router v režimu přístupového bodu (access point), nevyužívá WAN. Router plnil důležitou úlohu. Připojení k internetu není v autě k dispozici. Byla proto vytvořena lokální off-line síť, a tím umožněna komunikace mezi jednotlivými platformami. IP adresy

připojených zařízení jsou přiděleny automaticky, DHCP protokol je tedy povolen. Pro jednotlivé MAC adresy všech zařízení jsou však vlastní IP adresy rezervovány. Jak laptop, tak i tři Raspberry Pi byly prostřednictvím ethernetu připojeny do LAN portů Wi-Fi routeru. Na obrázku (Obr. 4-5) lze vidět seznam připojených zařízení.

ID	Client Name	MAC Address	Assigned IP	Lease Time
1	Lada-Nitro-5	98:28:A6:10:11:9F	192.168.0.100	Permanent
2	ubuntu2	B8:27:EB:58:C3:94	192.168.0.102	Permanent
3	ubuntu1	B8:27:EB:CE:B1:09	192.168.0.101	Permanent
4	ubuntu3	B8:27:EB:E9:A1:AA	192.168.0.103	Permanent

Obr. 4-5 IP adresy připojených zařízení

4.1.5. Acer Nitro 5 AN515-51

Jako nadřazený počítač byl použit výkonný notebook, obsahující čtyřjádrový procesor Intel i5 sedmé generace o taktu 2,5 GHz, a 8 GB operační paměti (RAM). Notebook měl nainstalovány dva operační systémy, tedy spuštění notebooku probíhá v režimu dual-boot, kde si lze vybrat mezi původně nainstalovaným Windows 10 64bit, nebo pro účely této diplomové práce – Ubuntu 18.04.

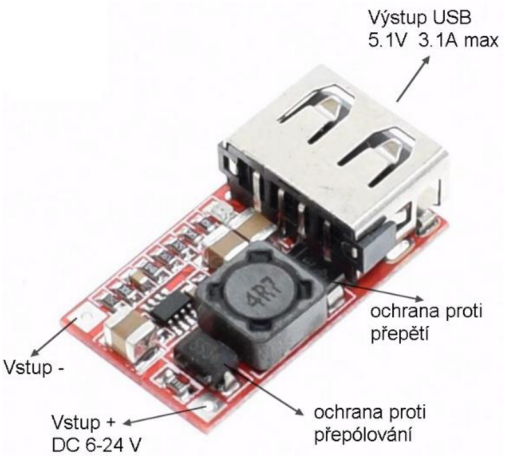
Úlohou nadřazeného počítače v experimentální soustavě byla příprava samotných Raspberry Pi, jako je instalace softwaru na tyto zařízení, provedení časové synchronizace všech zúčastněných zařízení. Přístup do terminálů Raspberry Pi byl tedy realizován skrze laptop. Dále zpracovával data ze tří jednodeskových počítačů Raspberry Pi. Nadřazený počítač byl připojen prostřednictvím ethernetu k routeru na lokální síť, odkud přijímal data k dalšímu zpracování. Data byla dále zpracovávána v ROSu.

Obr. 4-6 Acer Nitro 5 AN515-51 [50]

4.1.6. Napájení jednotlivých zařízení

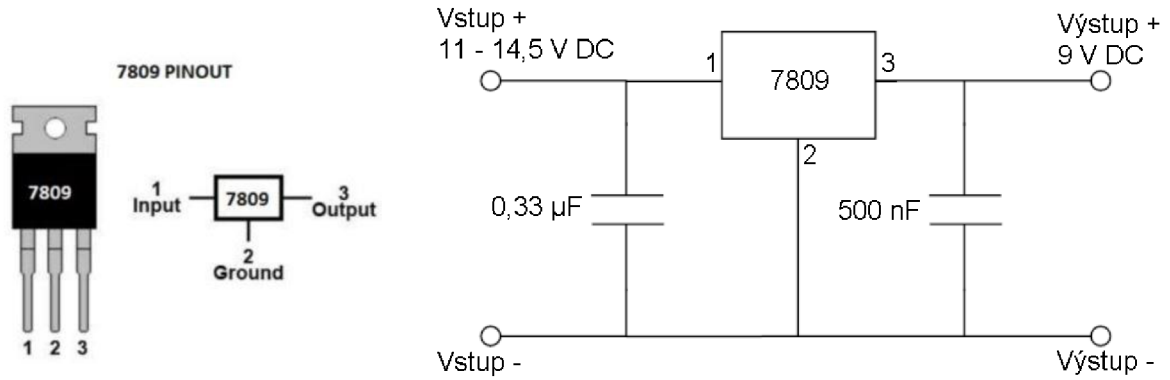
Experiment byl proveden v osobním automobilu. Bylo proto nutno vyřešit napájení jednotlivých zařízení. Laptop disponuje silnou baterií, zatímco Raspberry Pi a Wifi router potřebují specifické, pokud možno stabilní napájecí napětí. To bylo vyřešeno třemi USB moduly (Obr. 4-7) a jednoduchým obvodem složeným ze stabilizátoru napětí a dvou kondenzátorů (Obr. 4-8).

Pro napájení Raspberry Pi sloužil modul, který dokáže 6–24 V vstupního stejnosměrného napětí z autobaterie převést na stabilních 5 V stejnosměrných, a to při proudu až do 3,1 A. Vyznačuje se velmi vysokou účinností, nepřehřívá se. Obsahuje přepěťovou ochranu na výstupu, ochranu proti zkratu na výstupu a ochranu proti přepólování na vstupu.



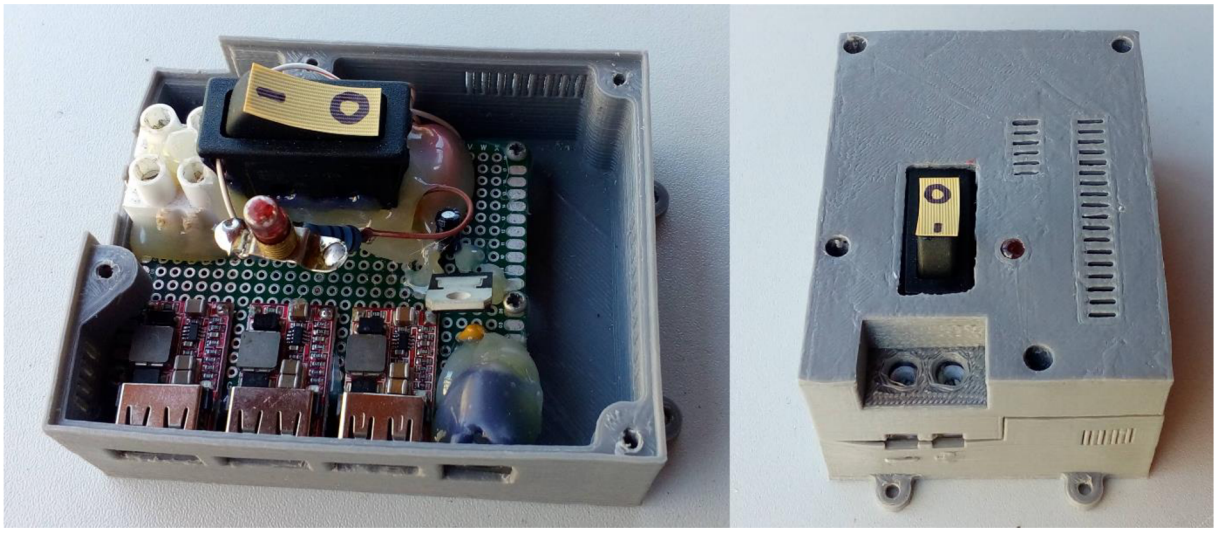
Obr. 4-7 Buck Step Down měnič DC 6-24V 12V/24V na 5V 3A USB [51]

Pro napájení Wifi routeru byl použit lineární stabilizátor napětí STMicroelectronics 7809-STM. Ten dokáže vstupní stejnosměrné napětí do 35 V převést na výstupní 9 V s výstupním proudem až do 1,5 A. Dále byl použit elektrolytický kondenzátor o kapacitě 0,33 µF a keramický kondenzátor o kapacitě 0,1 µF. Schéma obvodu podle obrázku (Obr. 4-8) bylo inspirováno příspěvkem na fóru [52].



Obr. 4-8 Lineární stabilizátor napětí použitý v obvodu s kondenzátory [52]

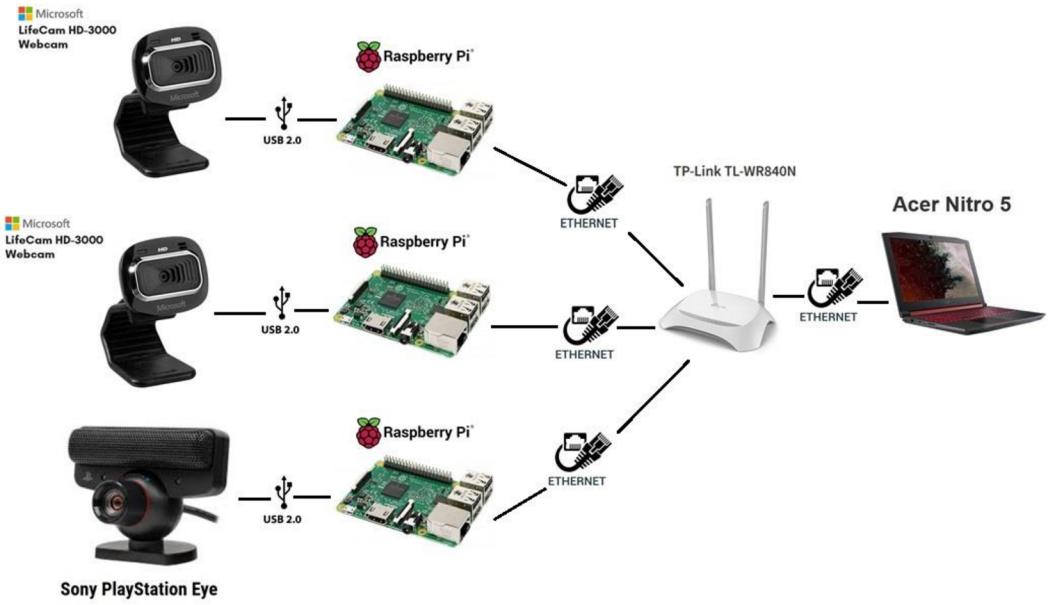
Všechny zmíněné komponenty byly nainstalovány na pájivé pole, a přivedeny ke společnému zdroji stejnosměrného napětí 12 V. Následně byly namontovány do ochranné krabičky, která byla vytištěna na 3D tiskárně. Více na obrázku (Obr. 4-9).



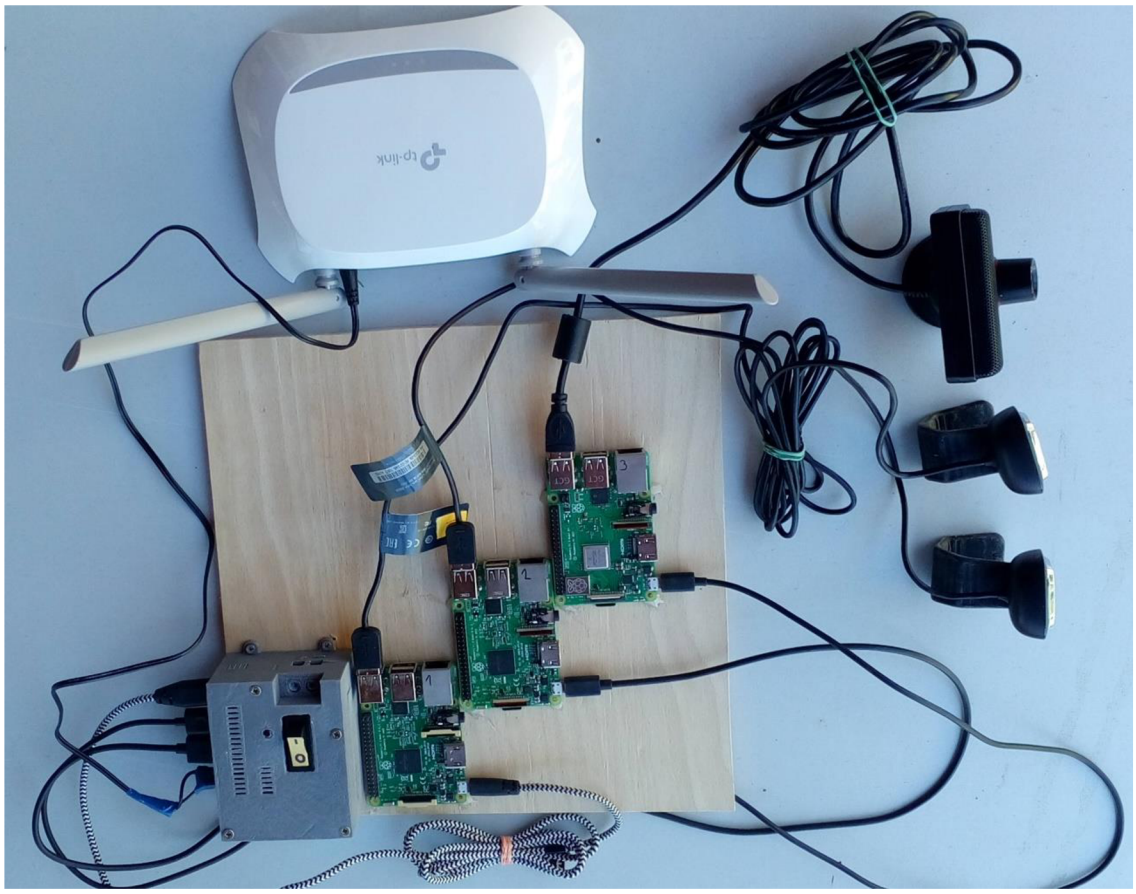
Obr. 4-9 Napájecí obvod pro experimentální soustavu

4.2. Experimentální soustava

Jak již bylo zmíněno, bylo při zpracovávání této diplomové práce využito tří USB kamer. Každá kamera byla prostřednictvím USB rozhraní připojena k jednodeskovému počítači Raspberry Pi (Slave). Tyto jednodeskové počítače byly pak prostřednictvím ethernet portů připojeny přímo k Wi-Fi routeru k lokální síti, kde byl ethernetem připojen i nadřazený počítač (Master). Schéma propojení jednotlivých komponentů je znázorněno na obrázku (Obr. 4-10). Tento obrázek tedy představuje architekturu experimentální soustavy. Na dalším obrázku (Obr. 4-11) je zachycena reálná experimentální soustava. Pro větší přehled zde nejsou zapojeny ethernety ani nadřazený počítač.



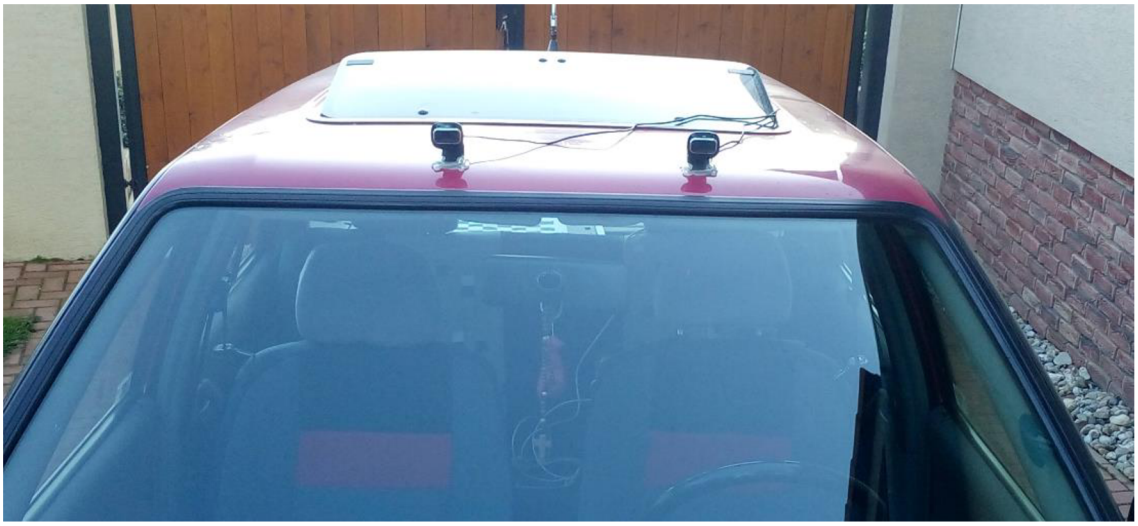
Obr. 4-10 Schéma experimentální soustavy



Obr. 4-11 Experimentální soustava, bez zapojených ethernet kabelů

4.3. Provedení ve vozidle

Dvě kamery Microsoft LifeCam byly umístěny pevně na střeše vozidla (Obr. 4-12), nad čelním sklem. USB kabely z obou kamer byly vedeny přes střešní okno přímo do Raspberry Pi. Třetí kamera byla přilepena k čelnímu sklu zevnitř (Obr. 4-14).

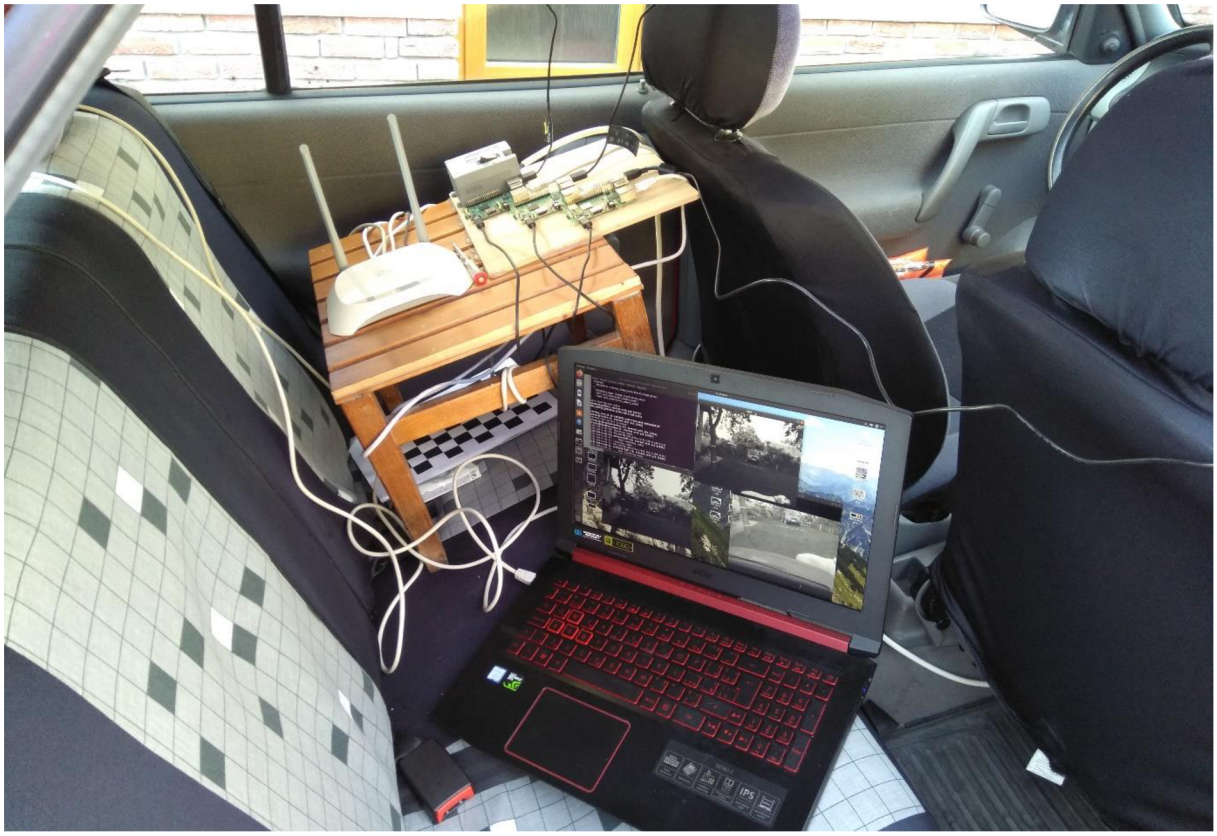


Obr. 4-12 Kamery na střeše vozidla

Obr. 4-13 Kamery na střeše vozidla + třetí kamera na čelním skle zevnitř

Obr. 4-14 Třetí kamera na čelním skle zevnitř vozidla

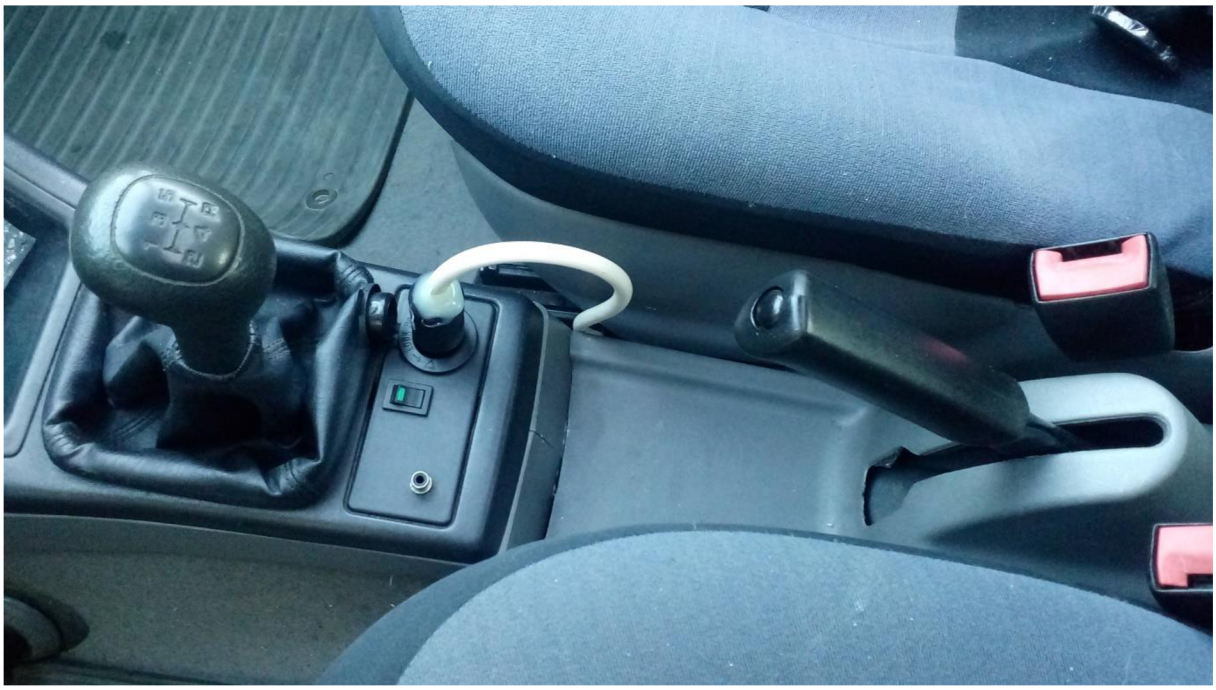
Experimentální soustava (Obr. 4-11) byla připevněna na malou dřevěnou stoličku. Jako celek byla soustava položena na levé zadní sedadlo (Obr. 4-15). Vyvedený ethernetový kabel pro připojení k laptopu byl dostatečně dlouhý na to, aby umožnil volný pohyb notebooku po vozidle (nebo na vozidle, v případě kalibrace kamer, kdy byl laptop po celou dobu na střeše).



Obr. 4-15 Experimentální soustava na zadním sedadle

Obr. 4-16 Experimentální soustava na zadním sedadle

Zdroj pro napájení soustavy byl vyveden kabelem pod sedadlem spolujezdce a zapojen do zásuvky pro zapalovač (Obr. 4-17), která poskytuje 12 V napětí z autobaterie. V zapalovači je rovněž pojistka na 3,5 A.



Obr. 4-17 Zásuvka zapalovače pro napájení experimentální soustavy

Součástí elektronické přílohy k diplomové práci je taky internetový odkaz na video, které názorně ukazuje spuštění experimentální soustavy.

5. Software

Obsah této kapitoly je zaměřen na software použitý k tvorbě hloubkové mapy v rámci této diplomové práce. Kapitola nastíní přípravu softwaru. Základem celé práce bylo použití operačního systému Ubuntu ve verzi 18.04, nainstalovaného na všech zařízeních. Stejně tomu bylo u ROSu. ROS byl všeobecně, včetně jeho výhod, popsán v první kapitole. Kapitola o použitém softwaru se zaměří na ROS z praktičtějšího hlediska, co se týče například samotné instalace, používání a orientace, tvorby pracovního prostředí, komunikace mezi uzly.

5.1. ROS. Instalace ROSu

V rámci diplomové práce proběhly veškeré instalace na operačním systému Ubuntu 18.04. Instalace byla provedena z příkazového řádku operačního systému, v našem případě mluvíme o terminálu. Verze ROSu použitá v této práci je ROS Melodic Morenia. Dále budou podrobně popsány všechny kroky instalace programu. Při instalaci bylo postupováno podle instrukcí na ROS portálu [53].

Prvním krokem nastavíme náš počítač, aby přijal software z *packages.ros.org*. Příkaz vyžaduje sudo oprávnění:

```
~$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
$(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-
latest.list'
```

Jako druhý krok nastavíme klíč repositáře, pomocí kterého systém ověřuje, zdali skutečně stahuje data z OSRF (*Open Source Robotics Foundation):*

```
~$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80'
--recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

Konečně se dostáváme k samotné instalaci. ROS obsahuje mnoho různých knihoven a nástrojů. Už při instalaci si lze zvolit, co všechno bude zahrnovat. V diplomové práci zahrnuje naše verze ROS všechny základní balíčky, plus: *rqt*, *rviz*, robot- generic knihovny, 2D/3D simulátory and 2D/3D vnímání. Doporučuje se i začátečníkům. Nainstalujeme příkazem:

```
~$ sudo apt install ros-melodic-desktop-full
```

Kromě balíčků nainstalovaných při běžné instalaci, existuje možnost jak doinstalování jiného požadovaného balíčku, tak i seznam všech dostupných balíčků. Učiníme tak příkazy:

```
~$ sudo apt install ros-melodic-PACKAGE
~$ apt search ros-melodic
```

Balíčky použité v této diplomové práci byly nainstalovány příkazy:

```
~$ sudo apt install ros-melodic-uvc-camera
~$ sudo apt install ros-melodic-camera-calibration
~$ sudo apt install ros-melodic-stereo-image-proc
```

ROS je nainstalován, zbývá ještě poslední krok. Abychom propojili operační systém s ROS systémem, byl upraven .bashrc script, který se vykonává při každém spuštění terminálu. Úpravu *.bashrc* souboru lze zapsat jako samostatný příkaz:

```
~$ echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
~$ source ~/.bashrc
```

Pro nahlédnutí do *.bashrc* souboru za účelem editace nebo kontroly lze použít linuxový příkaz *gedit* (otevření v textovém editoru) nebo *nano* (přímo v terminálu). Příkazy vypadají následovně:

```
~$ sudo gedit ~/.bashrc
~$ sudo nano ~/.bashrc
```

5.2. Spuštění a práce v ROS

Práce v ROSu spočívá především v používání konzolových nástrojů. Jak vyplývá z názvu, jsou nástroje zadávány jako příkazy do příkazového řádku operačního systému. To umožňuje například spouštět uzly, provádět diagnostiku uzlů apod. Typický příklad konzolového nástroje nese na prvním místě název nástroje (Tab. 5-1), zbylá místa patří argumentům. Při složitějších příkazech, jako je například volání kalibrátoru (obsahují mnoho parametrů), není důležité pořadí argumentů. Následující tabulka stručně popíše jednotlivé nejpoužívanější nástroje ROS.

Nástroj	Funkce
roscore	spustí jádro ROS, používán samostatně bez argumentů
rosrun	slouží k spuštění uzlů jednotlivě.
roslaunch	slouží k spuštění .launch souborů (více uzlů najednou), také spustí jádro ROS
rostopic	zobrazí informace o tématech
rosnode	zobrazí informace o uzlech
rosparam	dokáže ladit hodnoty parameter serveru
rosservice	informace o službách
rosmsg	informace o zprávách
rosbag	nástroj k logování dat
roscd	umožní navigaci mezi balíčky ROS, není nutná znalost cílové adresy v PC. funguje obdobně jako linuxový příkaz „cd“.
rosed	umožní editovat zadaný soubor v balíčku ROS, není nutná znalost cílové adresy v PC.

Tab. 5-1 přehled nejpoužívanějších nástrojů ROS [54]

5.2.1. Nástroj roslaunch

Roslaunch je nástroj pro snadné spouštění více uzlů ROS lokálně i vzdáleně (přes SSH protokol) umožňující také nastavit parametry *parameter serveru*. Spouští konfigurační soubory v značkovacím jazyce XML (přípona .launch, dále .launch soubor), kde jsou specifikované uzly, které se mají spustit, parametry, které se mají nastavit, a také stroje, na kterých by měly být příslušné uzly spuštěny. Mnoho ROS balíčků přichází již s vypracovanými .launch soubory, které lze spustit [55]:

```
~$ roslaunch název_balíčku spouštěcí_soubor.launch
```

Tento spouštěcí soubor je realizován použitím značek (tags), kde každá vlastní příslušné atributy. Použitím atributů jsou specifikovány vlastnosti značek. Atributy dělíme na povinné a volitelné. Nástroj roslaunch čte .launch soubor v jednom průchodu. Značky jsou vyhodnoceny sériově. Pokud existuje více nastavení pro tentýž parametr, použije se hodnota, která byla zadána jako poslední [55].

Značka	**Povinné atributy**	**Volitelné atributy**	**Funkce**
<launch>	–	–	Kořenový prvek. Slouží k vymezení oblasti pro ostatní značky.
<node>	pkg, type, name	args, machine, respawn, respawn_delay, required, ns, clear_params, output, cwd, launch-prefix	Definuje uzel, který chceme, aby se spustil.
<machine>	name, address, env-loader	default, user, password, timeout	Definice zařízení, na kterém chceme spustit uzel.
<remap>	from, to	–	Přejmenování témat publikovaných uzlem.
<param>	name	value, type, textfile, binfile, command	Nastavení parametrů parameter serveru.
<group>	–	ns, clear_params	Uspořádá více uzlů do jedné skupiny, pomáhá mezi nimi hromadně nastavovat parametry.

Tab. 5-2 Značky .launch souboru + příslušné atributy [55]

Praktické použití .launch souboru je k nahlédnutí v kapitole pojednávající o přípravě experimentu (Kap. 6.6.1).

5.3. Tvorba balíčku

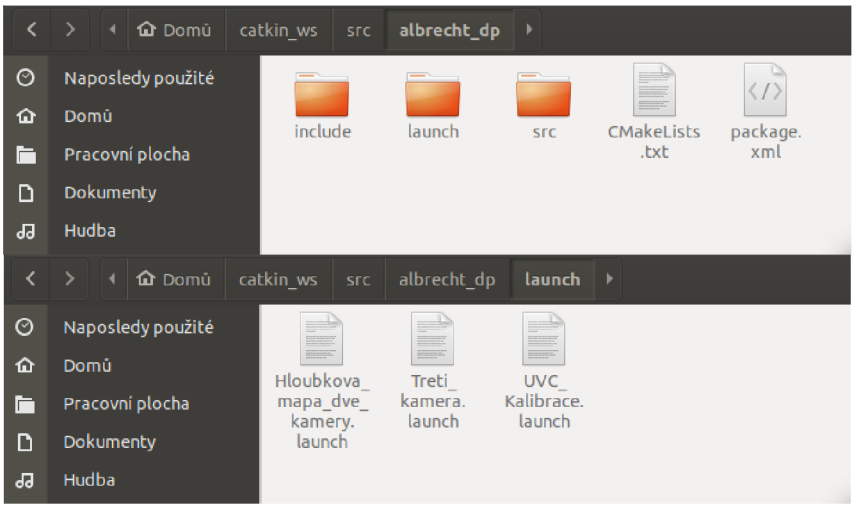
Jako první krok bylo vytvořeno pracovní prostředí:

```
~$ cd
~$ mkdir -p ~/catkin_ws/src
~$ cd ~/catkin_ws/
~$ catkin_make
~$ source devel/setup.bash
```

Složka *catkin_ws/src* je místo pro umístění uživatelem vytvořených balíčků. Zde byl vytvořen balíček s názvem *albrecht_dp*. Název se doporučuje psát pouze malými písmeny.

```
~$ cd ~/catkin_ws/src
~$ catkin_create_pkg albrecht_dp std_msgs rospy roscpp
~$ cd ~/catkin_ws/
~$ catkin_make
~$ source ~/catkin_ws/devel/setup.bash
```

Členy v příkazech výše, *std_msgs, rospy, roscpp,* se nazývají závislosti. Poslední řádek zde není nutný, při instalaci ROSu byla tato úprava již provedena v .bashrc souboru (Kap. 5.1). Složka pracovního prostředí, včetně všech podadresářů, jsou součástí elektronické přílohy k diplomové práci.



Obr. 5-1 Struktura adresářů pracovního prostředí, umístění .launch souborů v adresáři

Spuštění projektu vypadá následovně. První .launch soubor na obrázku (Obr. 5-1) spustí uzly dvou kamer, včetně hloubkové mapy. Druhý .launch soubor přidá do systému uzel třetí kamery. Třetí .launch soubor spustí pouze uzly dvou kamer, před samotným zadáním příkazu kalibrátoru (Kap. 6.5).

```
~$ roslaunch albrecht_dp Hloubkova_mapa_dve_kamery.launch
~$ roslaunch albrecht_dp Treti_kamera.launch
```

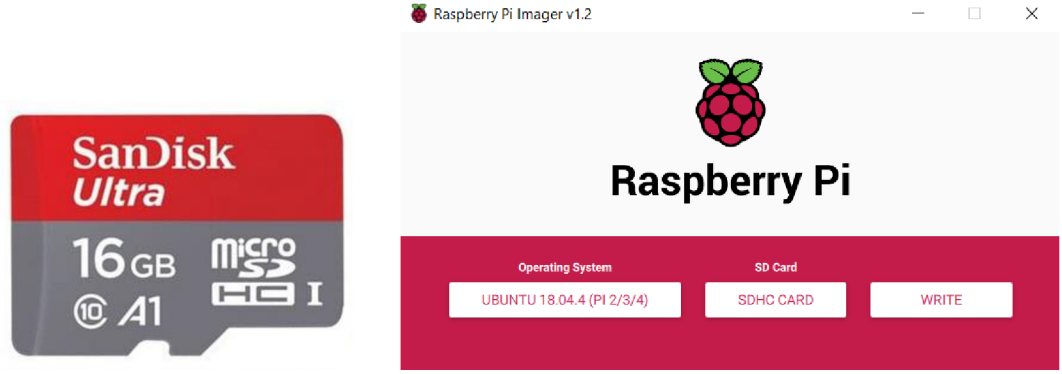
Při tvorbě balíčku bylo postupováno podle instrukcí na ROS portálu [56]

6. Příprava experimentu

Jak již bylo v předchozích kapitolách zmíněno, bylo nutno nejdříve sestavit experimentální soustavu. Tři kamery byly stacionárně umístěny na vozidle (Obr. 4-13). Dvě kamery na čelním skle, třetí kamera vzadu. Mluvíme-li o dvou kamerách na čelním skle, důležité bylo jak vzájemné vertikální natočení, tak i horizontální natočení. Dva identické pixely od dvou obrazů by měly ležet nejlépe ve stejné horizontální úrovni. Natočení okolo vertikální osy určuje, do jaké vzdálenosti budeme schopni „prostorově vidět".

6.1. Instalace Ubuntu na Raspberry Pi

Nejprve proběhla instalace operačního systému na SD kartu k jednodeskovému počítači Raspberry Pi. SD karta SanDisk Ultra A1 o velikosti 16 GB byla naformátována. Dále byl z oficiálních stránek Ubuntu stažen operační systém. Jedná se o Ubuntu Server 18.04.4 v 32bit verzi pro Raspberry Pi, v podobě souboru s příponou .img. Ten bylo nutno vypálit na již zmíněnou SD kartu. K tomu posloužil nástroj Raspberry Pi imager. Nyní je SD karta připravena pro zasunutí do Raspberry Pi, po kterém následuje první spuštění zařízení, včetně konfigurace prvního uživatelského nastavení (jméno uživatele, heslo a root přístup).



Obr. 6-1 SD karta SanDisk Ultra a Raspberry Pi imager GUI [57, 58]

K připravenému Raspberry Pi byly připojeny USB, monitor (HDMI), Wi-Fi router (ethernet) a zdroj (5 V, 2,2 A). Na monitoru již bylo vidět první spuštění operačního systému Ubuntu a jeho první konfigurace. Po uplynutí zhruba 5 minut vyzval systém k zadání uživatelského jména a následně i hesla. Nejdříve bylo nutno zadat „ubuntu" jak pro jméno, tak pro heslo, až poté systém vyzval k novému, uživatelem zvolenému heslu. Stejný postup byl dále aplikován pro druhý a třetí jednodeskový počítač Raspberry Pi, tudíž i zbylé SD karty. Klíčové údaje ke všem počítačům Raspberry Pi a k nadřazenému počítači (tak, jak jsou reprezentovány v .launch souboru) jsou zaznamenány v tabulce (Tab. 6-1).

Zařízení	pozice	uživatelské jméno	heslo	IP adresa ve společné síti
Laptop	master	lada	(–)	192.168.0.100
rpi1	slave	ubuntu1	machine1	192.168.0.101
rpi2	slave	ubuntu2	machine2	192.168.0.102
rpi3	slave	ubuntu3	machine3	192.168.0.103

Tab. 6-1 Přístupové údaje a IP adresy všech zařízení

6.2. Komunikace s Raspberry Pi

Jako další krok, a to za předpokladu správného připojení jednotlivých komponent, bylo zapotřebí připojit se z nadřazeného počítače k podřazeným jednodeskovým počítačům. Spojení uvedených platforem bylo realizováno zabezpečeným komunikačním protokolem SSH (Secure Shell). Jedná se o protokol, pomocí kterého je zprostředkován přístup k terminálu (příkazovému řádku) Raspberry Pi z nadřazeného počítače. Pro běžné aplikace, v této diplomové práci, se podřazené jednodeskové počítače ovládaly pouze pomocí terminálu. Příkaz, který poskytuje přístup do Raspberry Pi terminálu a využívá SSH protokol, vypadá následovně:

```
~$ ssh ubuntu1@192.168.0.101
~$ ssh ubuntu2@192.168.0.102
~$ ssh ubuntu3@192.168.0.103
```

Při prvním připojení k jednodeskovému počítači, je automaticky zapsán SSH klíč do souboru ~/ssh/*known_hosts*, a to podle jednoho ze tří algoritmů. V současné době existují tři algoritmy: ssh-rsa, ssh-dss, ssh-ecdsa. Zadáním výše uvedeného příkazu do terminálu užije systém výchozí algoritmus ssh-ecdsa k zaznamenání klíče.

Při spouštění projektu užitím spouštěcího souboru (.launch soubor) příkazem *roslaunch* může nastat problém, kdy se nelze připojit k jednomu nebo k oběma jednodeskovým počítačům. Systém nemůže najít zařízení o dané IP adrese v souboru hostitelských klíčů (*known_hosts*). Hlavním důvodem tohoto problému je příkaz roslaunch, který využívá paramiko. Paramiko je implementace SSH protokolu v Pythonu (2.7, 3.4+). Nepodporuje totiž hostitelské klíče v ecdsa formátu. Proto bylo nutno klíče zapsat užitím jiného algoritmu. Před provedením následujícího příkazu je vhodné stávající soubor s hostitelskými klíči smazat. Příkazy je nutno nejprve samostatně poslat v samostatných terminálech [59].

```
~$ ssh -oHostKeyAlgorithms='ssh-rsa' host
```

Konkrétní příkazy tedy zní:

```
~$ ssh -oHostKeyAlgorithms='ssh-rsa' ubuntu1@192.168.0.101
~$ ssh -oHostKeyAlgorithms='ssh-rsa' ubuntu2@192.168.0.102
~$ ssh -oHostKeyAlgorithms='ssh-rsa' ubuntu3@192.168.0.103
```

6.2.1. Konfigurace .bashrc souboru

Poté co jsme se úspěšně vzdáleně připojili do terminálů jednotlivých Raspberry Pi, bylo nutno upravit .bashrc soubory u všech počítačů. Tato konfigurace zabezpečila komunikaci ROS uzlů mezi master a slave počítačem. Dále určila, který z počítačů je nadřazený. Poslední řádky .bashrc souboru, a tím pádem finální úprava pro nadřazený počítač, vypadají potom takto:

```
source /opt/ros/melodic/setup.bash
source ~/catkin_ws/devel/setup.bash
export ROS_HOSTNAME="192.168.0.100"
export ROS_IP="192.168.0.100"
export ROS_MASTER_URI="http://192.168.0.100:11311"
```

Pro Raspberry Pi s IP adresou 192.168.0.101 (rpi1) platí analogicky:

```
source /opt/ros/melodic/setup.bash
source ~/catkin_ws/devel/setup.bash
export ROS_HOSTNAME="192.168.0.101"
export ROS_IP="192.168.0.101"
export ROS_MASTER_URI="http://192.168.0.100:11311"
```

Síťová konfigurace byla nastavena dle instrukcí na ROS Wiki portálu [60].

6.2.2. Remote env-loader

V případě realizace úlohy užitím .launch souboru (nástroj roslaunch) stačí pouhá editace .bashrc souboru. Avšak v případě vytvoření určitého scriptu, který načte pracovní prostředí dané platformy, není nutné .bashrc soubor editovat.

V rámci této práce byly provedeny obě konfigurace (úprava .bashrc souboru a vytvoření env-loader.sh scriptu) Nemají na sebe vzájemný vliv, spolehlivost a celkové chování při práci se soustavou vzroste. Také to lze provést lidově řečeno „pro jistotu“.

Již zmiňovaný soubor s názvem remote-env-loader.sh byl vytvořen na všech třech Raspberry Pi počítačích, a to pod adresou */home/ubuntuX/catkin_ws/devel/*, kde X značí číslo zařízení (1,2,3). Příklad scriptu, který se v této diplomové práci používá, je pro rpi1 [61]:

```
#!/bin/bash

export ROS_IP=192.168.0.101
export ROS_MASTER_URI=http://192.168.0.100:11311

source /home/ubuntu1/catkin_ws/devel/setup.bash

exec "$@"
```

Ještě je třeba učinit soubor spustitelným, příkaz k tomu zní:

```
~$ chmod +x remote-env-loader.sh
```

6.3. Reprezentace Raspberry Pi v .launch souboru

Po shrnutí všech poznatků z předchozích kapitol lze vytvořit .launch soubor. Jedná se o script v .XML jazyce, který je vykonán nástrojem roslaunch (více v kapitole 5.2.1). Nachází-li se více počítačů v experimentální soustavě (případ této diplomové práce), nutno je nejdřív definovat.

Následující úryvek .launch souboru definuje nejdříve zařízení včetně přístupových údajů (Tab. 6-1), které je v další části scriptu použito jako argument pro spuštění uzlu kamery. Jedná se tu o *rpi1*, na kterém je spuštěn uzel levé kamery s názvem *left*, nainstalovaného balíčku *uvc_camera*, typu *uvc_camera_node*.

```
<launch>

<machine
   name = "rpi1"
   address = "192.168.0.101"
   password = "machine1"
   user = "ubuntu1"
   env-loader = "/home/ubuntu1/catkin_ws/devel/remote-env-loader.sh">
</machine>

<node name="left" pkg="uvc_camera" type="uvc_camera_node" machine = "rpi1">
   < ... />
</node>

</launch>
```

6.4. Časová synchronizace

Samotné Raspberry Pi nemá RTC modul (real time clock) a nemá ani baterii. Když se všechna Raspberry Pi zapnou, mají podobný interní čas, ale ne stejný. Raspberry Pi využívá modul, kterým si dokáže přes internet synchronizovat čas. Problémem je, že připojení k internetu není v testovaném prostředí k dispozici.

Bylo nutno synchronizovat čas jiným způsobem. K tomu posloužil nástroj chrony, kterému je tato podkapitola věnována. Tento software je kompatibilní s operačními systémy, jakými jsou Linux, FreeBSD, NetBSD, macOS nebo Solaris [62].

6.4.1. Chrony

Chrony je flexibilní implementace NTP (Network time protocol). Dokáže synchronizovat systémový čas s NTP servery, referenčními hodinami, nebo s manuálním vstupem (čas nastaven pevně, např z klávesnice). Zvládá dobře širokou škálu podmínek, například: přerušovaná síťová připojení, silně přetížené sítě, měnící se teploty (běžné počítačové hodiny jsou citlivé na teplotu) a systémy, které neběží nepřetržitě nebo na virtuálním počítači. Mezi nevýhody patří například nemožnost synchronizovat čas broadcastem nebo multicastem a podpora relativně malého počtu platforem [62].

Typická přesnost mezi dvěma zařízeními synchronizovanými přes internet je v řádu jednotek milisekund. Přes LAN spojení, mluvíme o desítkách mikrosekund. Jestliže nastavíme za referenční hodiny hardwarové, můžeme se pohybovat i v menších číslech (až nanosekundách) [62].

Celá aplikace se skládá ze dvou částí [62]:

- **chronyd**

Jedná se o daemon běžící na pozadí, který zabezpečuje síťovou výměnu dat s NTP servery a klienty, nastavování systémových i hardwarových hodin, generuje statistiky a informace o zdrojích.

- **chronyc**

Klient sloužící na komunikaci s daemonem chronyd. Představuje jednoduchý příkazový nástroj, pomocí kterého je možné za běhu nastavovat různé parametry, dočasně měnit konfiguraci daemona, ručně nastavovat aktuální čas systémových i hardwarových hodin, zobrazovat generované statistiky atd.

6.4.2. Provedení synchronizace

Prvním krokem byla samotná instalace chrony. Dále bylo třeba upravit soubor s konfigurací, vytvořený při instalaci. Při instalaci a samotné konfiguraci byl velmi užitečný oficiální manuál k softwaru, podle kterého bylo postupováno. Je dostupný zde: [63].

```
~$ sudo apt install chrony
```

Při instalaci se vytvořil soubor s konfigurací, který bylo potřeba upravit:

```
~$ sudo nano /etc/chrony/chrony.conf
```

Dále bylo nutno nakonfigurovat, aby stanice běžela v lokálním módu, tj. není synchronizována k žádnému vnějšímu zdroji. Využívá systémový čas. Na master stanici vypadala konfigurace následovně:

```
local stratum 8
commandkey 1
manual
allow 192.168.0
smoothtime 0.01 0.01
keyfile /etc/chrony/chrony.keys
driftfile /var/lib/chrony/chrony.drift
logdir /var/log/chrony
maxupdateskew 1
rtcsync
makestep 0.01 1
```

Slave stanice byla nastavena tak, aby se časově synchronizovala s master stanicí. Byl přidán nový serverový vstup s IP adresou master stanice. Konfigurace pro jednu ze slave stanic vypadala následovně:

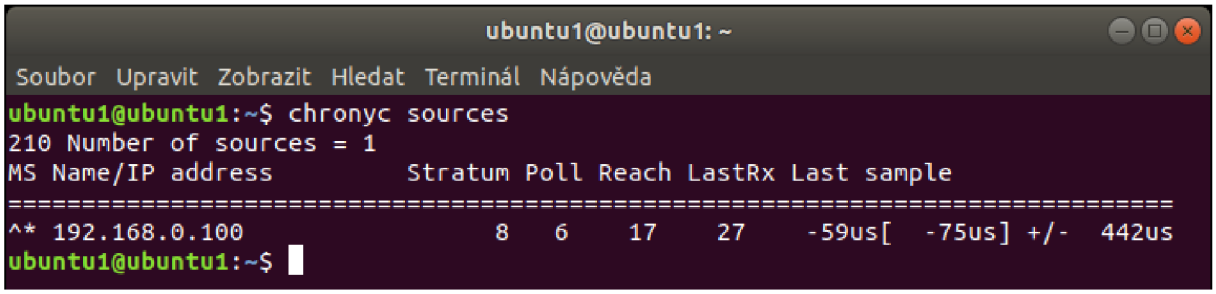
```
server 192.168.0.100 iburst
keyfile /etc/chrony/chrony.keys
driftfile /var/lib/chrony/chrony.drift
logdir /var/log/chrony
maxupdateskew 1
commandkey 24
rtcsync
makestep 0.01 1
```

Po uplatnění změn v konfiguraci u všech zařízení bylo potřeba službu chrony restartovat. To bylo provedeno z root uživatele.

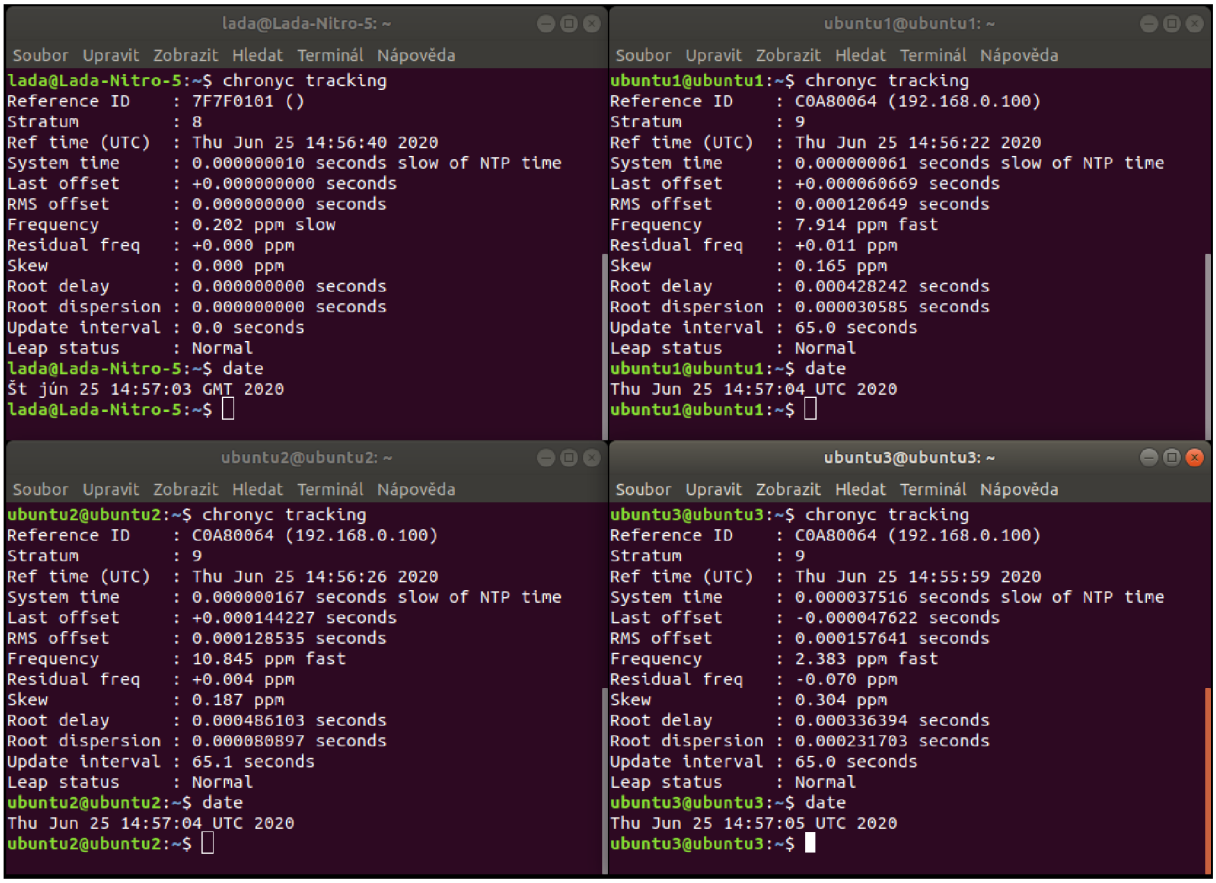
```
~# systemctl restart chrony
~# systemctl enable chrony
```

Tím byly všechny slave stanice časově synchronizovány s master stanicí. Pomocí následujících dvou příkazů, lze zjistit zdrojové NTP servery, nebo sledovat aktuální stav synchronizace. Výstupy obou příkazů jsou znázorněny na obrázcích níže (Obr. 6-2, Obr. 6-3).

```
~$ chronyc sources
~$ chronyc tracking
```



Obr. 6-2 Výstup příkazu "chronyc sources" na rpi1



Obr. 6-3 Výstup příkazu "chronyc tracking"

Výstupy v terminálu značí, že slave stanice mají k dispozici skutečně jediný zdroj času, a tím je master stanice s IP adresou 192.168.0.100. Rovněž vystupuje tato IP adresa vystupuje jako Referenční ID na výstupu všech slave terminálů (ubuntu1,2,3). Referenční čas

značí, kdy proběhlo poslední měření z referenčního zdroje, resp. kdy byl uložen záznam. Toto měření probíhá v intervalu „update interval“.

6.5. Stereo kalibrace kamer

Pro nejlepší výsledek v podobě kvalitní hloubkové mapy bylo nutno obě kamery zkalibrovat. Samotná stereo kalibrace se skládá ze dvou částí – kalibrace každé kamery samostatně a stereo rektifikace obrazů. Princip stereo rektifikace spočívá v získání vektoru vzájemného posunutí obou kamer a matice vzájemného natočení kamer, pomocí kterých lze obrazy jaksi „vyrovnat“. Jde o to, aby jednotlivé stejné pixely z levého a pravého obrazu byly ve stejné horizontální poloze, nebo tomu alespoň co nejblíže. Při samotné tvorbě hloubkové mapy lze nastavit parametry, které berou do úvahy tuto nepřesnost.

Součástí ROSu je také knihovna *image_pipeline*, která byla použita pro kalibraci kamer a pro tvorbu hloubkové mapy.

Na obrázku (Obr. 4-12) jsou obě kamery umístěny stacionárně a vhodně vzájemně vertikálně a horizontálně natočeny. Pro provedení stereo kalibrace bylo nutno spustit na každém Raspberry Pi uzly příslušných USB kamer. V této diplomové práci produkují uzly USB kamer témata (topics), jako jsou například *left/image_raw*, *left/camera_info* a *right/image_raw*, *right/camera_info*. Pro korektní zadání nástroje pro vyvolání kalibrace byla tato témata přejmenována užitím příkazu *remap from*. To lze zapsat do .launch souboru následujícím způsobem:

```
<launch>

<machine
   name = "rpi1"
   ... >
</machine>

<node name="left" pkg="uvc_camera" type="uvc_camera_node" machine = "rpi1">
   <param name=" ... "/>
   <remap from="left/image_raw" to="stereo/left/image_raw" />
   <remap from="left/camera_info" to="stereo/left/camera_info" />
</node>

</launch>
```

Nástroj v ROSu pro stereo kalibraci lze zapsat příkazem níže. Včetně parametrů [64]:

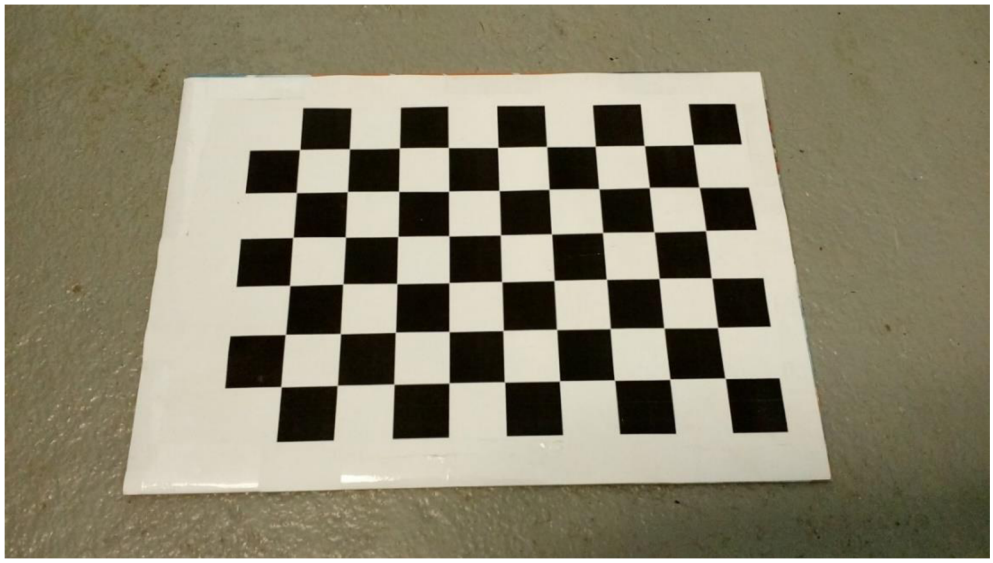
```
~$ rosrun camera_calibration cameracalibrator.py -approximate
0.1 -size 9x6 -square 0.024 right:=/stereo/right/image_raw
left:=/stereo/left/image_raw right_camera:=/stereo/right
left_camera:=/stereo/left
```

Parametr **approximate** umožní kalibrátoru pracovat s obrazy, které nejsou dokonale synchronizovány v čase. Nyní je nastaven na hodnotu 0,3 sekund. V tomto případě, vezme kalibrátor do úvahy i obrazy o takových časových známkách, které se liší do 0,1 sekund.

Parametr **size** udává rozměry šachovnice použité při kalibraci. Šachovnice použitá v této diplomové práci (Obr. 6-4) měla rozměry 9×6 čtverců.

Parametr **square** udává délku hrany jednoho čtverce na šachovnici v metrech. Šachovnice použitá v této diplomové práci byla ze čtverců o hraně 0,024 m.

Nakonec se definují pravá, levá kamera, a konkrétní publikovaná témata (topics).



Obr. 6-4 Šachovnice přilepená na pevném podkladu pro kalibraci kamer

6.5.1. Provedení stereo kalibrace dvou kamer

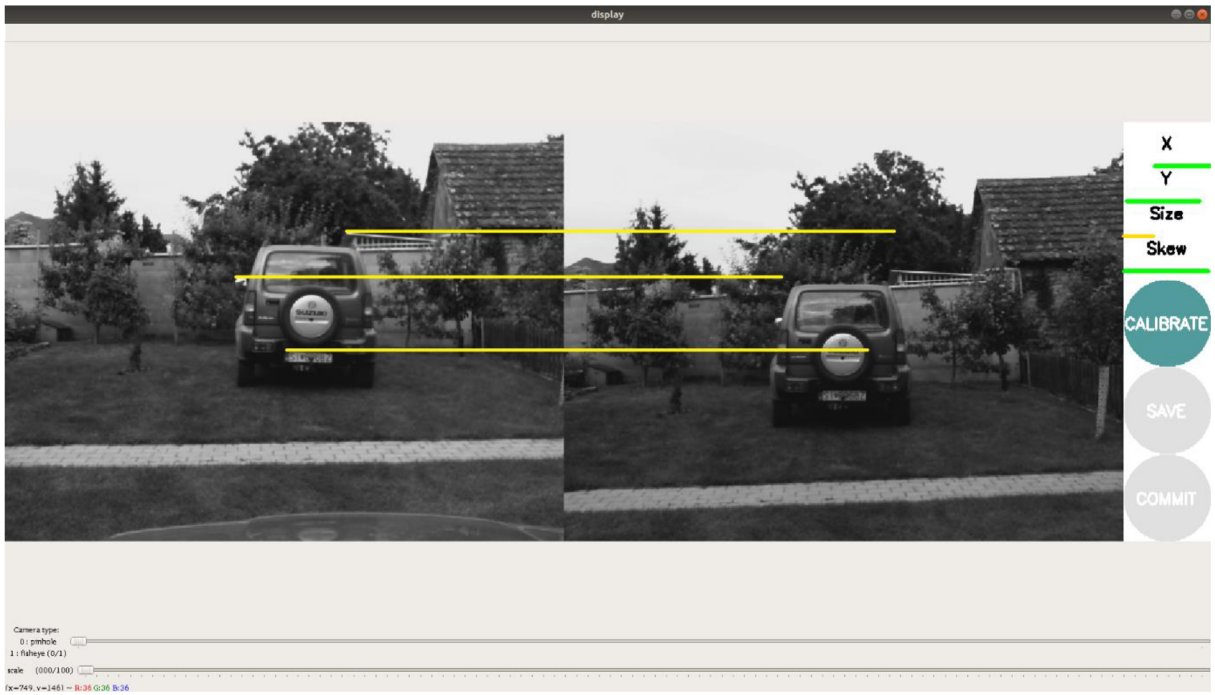
Po vyvolání již zmíněného příkazu se spustil nástroj na kalibraci kamer. Otevřelo se grafické uživatelské rozhraní (Obr. 6-5). Okno rozhraní je rozděleno na obrazy z levé a pravé kamery. V pravém dolním rohu jsou tři tlačítka/funkce. Funkce *CALIBRATE* se zpřístupní poté, co systém nashromáždí dostatečné množství snímků se šachovnicí. Tato funkce nejdřív spočítá, a následně uplatní výsledné parametry kalibrace na aktuální obraz v okně kalibrátoru. Funkce *SAVE* uloží výsledek kalibrace do složky (Kapitola 6.5.2). Poslední funkce *COMMIT*, nahraje výsledky do připojených kamer. Vytvoří tím soubor ve složce /.ros/camera_info/<camera_name>.yaml každého ze zúčastněných zařízení.

Jestliže je na záběru přítomna šachovnice, nástroj ji zachytí a oba snímky – z pravé i levé kamery, uloží. Snímků je několik. Záleží, jak dlouho kalibraci provádíme. Snímky se ukládají pod názvy left00X a right00X. Jsou k nalezení v souboru calibrationdata.tar.gz, který se vytvoří po rozkliknutí tlačítka *SAVE* v okně kalibrátoru. Rovněž zde najdeme i výsledky kalibrace pro pravou a levou kameru.

Pro poskytnutí dostatku kvalitních snímků pro potřebnou kalibraci bylo třeba šachovnici zobrazovat pod různými úhly, v různých vertikálních a horizontálních polohách. Pro kvalitnější výsledek byla šachovnice rovnoměrně přilepena na rovnou, pevnou desku (Obr. 6-5). Množství snímků potřebných pro výpočet je znázorněno i v samotném okně kalibrátoru, a to vpravo nahoře. Jedná se o ukazatele: *X* (pohyb šachovnice v obou horizontálních směrech), *Y* (pohyb šachovnice v obou vertikálních směrech), *Size* (přiblížení a oddálení šachovnice), *Skew* (natočení šachovnice). Jakmile se tyto ukazatele „naplní“, systém oznámí zpřístupněním tlačítka *CALIBRATE* dostatek snímků.

Obr. 6-5 Grafické uživatelské rozhraní ROS kalibrátoru

Obrázek 6-5 značí dostatečný počet snímků pro kalibraci. Podle ukazatelů (vpravo nahoře GUI), lze určit kolik snímků šachovnice v různých polohách je zapotřebí udělat. Není zapotřebí naplnit všechny ukazatele. Na uvedeném příkladu je ukazatel *Size* slabý. Důvodem je malý počet snímků šachovnice zblízka. To nám ale nebrání v úspěšné kalibraci. Další obrázek (Obr. 6-6) znázorňuje tutéž scénu bez šachovnice před kalibrací.



Obr. 6-6 Levá a pravá scéna před kalibrací

Na obrázku (Obr. 6-6) jsou pomocí žlutých čar vyneseny pozice výrazných objektů levé scény. Díky nim je patrné, že je pravá kamera znatelně natočena směrem nahoru. Další

obrázek (Obr. 6-7) obsahuje tutéž scénu po kalibraci (po rozkliknutí funkce *CALIBRATE*). Žluté čáry nyní spojily identické objekty obou scén. Mluvíme tedy o rektifikaci obrazů obou kamer. Takto rektifikované kamery jsou dokonale zarovnány, mají stejnou ohniskovou vzdálenost a jejichž ohniskové body odpovídají původním.

Obr. 6-7 Levá a pravá scéna po kalibraci

6.5.2. Výsledky stereo kalibrace

Úspěšná stereo kalibrace dvou kamer se projevuje na výsledném obrazu tím, že jednotlivé shodné pixely z obou obrazů jsou ve stejné horizontální výšce. Jako příklad lze vzít rohy, nebo hrany objektů zachycených oběma kamerami (Obr. 6-7). Obrazy jsou rektifikovány. Neúspěšná kalibrace může mít za následek prázdný, nebo nerozpoznatelný obraz výsledné hloubkové mapy.

V souboru s uloženými výsledky pak můžeme najít matici kamery, matici rektifikace, matici projekce, vektor koeficientů zkreslení, vektor translace a rotace vzájemné polohy obou kamer. Tento soubor je součástí elektronické přílohy k diplomové práci. V elektronické příloze se dále nachází internetový odkaz na video – názornou ukázku kalibrace kamer.

6.6. Tvorba hloubkové mapy

Již bylo řečeno, že ROS využívá k tvorbě hloubkové mapy knihovnu *image_pipeline*. Jedním z nástrojů uvedené knihovny je také balíček *stereo_image_proc*. Tento balíček obsahuje také stejnojmenný uzel *stereo_image_proc*, který byl v této diplomové práci použitý. Uzel leží mezi ovladači kamer a uzly pro zpracování obrazu.

Uzel *stereo_image_proc* odstraňuje zkreslení (užitím parametrů kamer získaných při kalibraci) a zbarví nezpracované (surové) obrazy u obou kamer jednotlivě. U dobře kalibrovaných stereo kamer je zkreslení kombinováno s rektifikací. Obrazy jsou poté transformovány tak, aby byly řádky jejich skenování uspořádány pro rychlé stereo zpracování. Uzel *stereo_image_proc* také počítá hloubkovou mapu z příchozích stereo dvojic obrazů pomocí algoritmu *OpenCV block matching* (BM). Zpracované stereo dvojice lze nejlépe zobrazit užitím *stereo_view* nástroje [65].

Uzel také vytvoří bodové mraky, které lze zobrazit v prostředí *rviz* [65].

Předpokládáme správnou síťovou konfiguraci experimentální soustavy (úprava .bashrc souborů). Před samotným voláním nástroje pro tvorbu hloubkové mapy bylo zapotřebí spustit jádro ROSu *roscore* na master stanici. Poté byly spuštěny uzly USB kamer na každém Raspberry Pi samostatně.

Následující příkazy poskytují pouze stručný přehled (stručnou posloupnost příkazů), jak se dopracovat k výsledku. Nejsou zde zahrnuty parametry pro správný chod systému, jako jsou přejmenování témat (topics), parametry uzlů USB kamer, parametry hloubkové mapy apod. Tyto příkazy mají pouze informativní charakter o tom, jak balíček *image_proc* použít (jak byl použit v rámci této diplomové práce). Kompletní výsledná konfigurace, včetně všech důležitých parametrů, je zapsána jako .launch soubor v XML jazyce. Ten je potom vyvolán příkazem *roslaunch*. Znění .launch souboru lze nalézt v následující podkapitole (6.6.1), kde jsou zmiňované parametry i vysvětleny. Kompletní .launch soubory jsou k nalezení v elektronické příloze k diplomové práci.

```
~$ roscore
```

Spuštění uzlu USB kamery na každé slave stanici jednotlivě (Raspberry Pi):

```
~$ rosrun uvc_camera uvc_camera_node
```

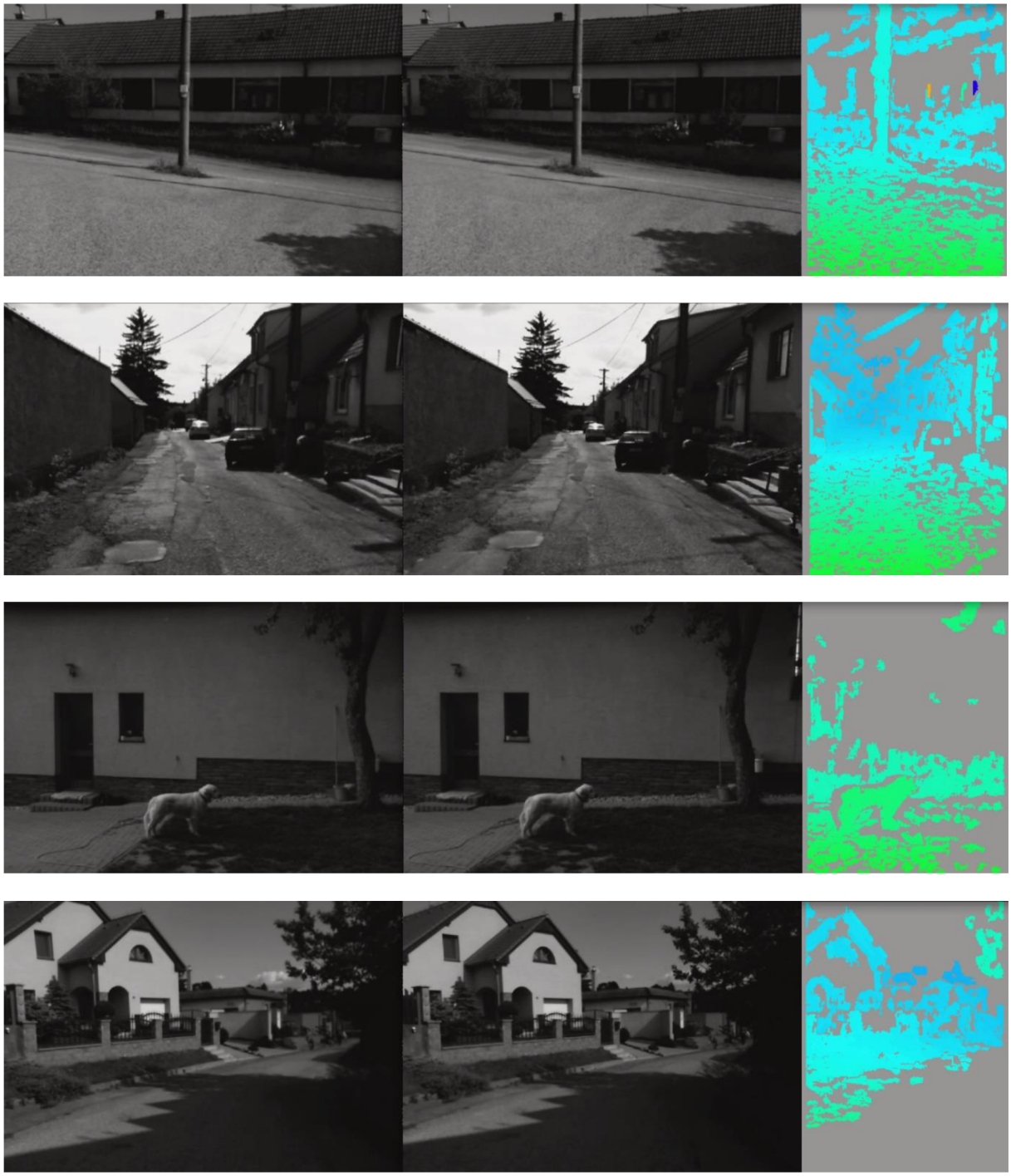
Nyní předpokládáme, že jsou obrazy obou kamer správně zkalibrovány. Potom lze přistoupit k uzlu *stereo_image_proc*. Příkaz lze zadat na master stanici v novém okně terminálu:

```
~$ ROS_NAMESPACE=stereo rosrun stereo_image_proc
stereo_image_proc stereo:=stereo image:=image_rect
_approximate_sync:=true
```

Uzel *stereo_image_proc* byl spuštěn a běží. Užitím nástroje *stereo_view* v novém okně terminálu zobrazíme výsledek:

```
~$ rosrun image_view stereo_view stereo:=stereo
image:=image_rect _approximate_sync:=True
```

Výsledek je v podobě tří oken. Zobrazí se témata *stereo/left/image_rect*, *stereo/right/image_rect,* a nakonec samotná hloubková mapa. Následuje několik ukázek výsledků na fotografiích. Na každé fotografii jsou obrázky v pořadí zleva: levá kamera, pravá kamera, hloubková mapa.



Obr. 6-8 Výsledky experimentu hloubkové mapy

6.6.1. Spuštění experimentu užitím .launch souboru

Tato podkapitola je věnována celému spouštěcímu souboru potřebnému pouze k tvorbě hloubkové mapy. Některé části tohoto .launch souboru byly v diplomové práci už připomenuty, a to za účelem lepší představy konkrétní problematiky. Jednalo se například o příkazy *remap*, nebo reprezentace (definice) externí stanice. Tato podkapitola slouží k popsání .launch souboru jako souvislého celku.

V prvním kroku byly definovány externí stanice. Zadaly se údaje jako: název stanice, příslušná IP adresa v lokální síti, přihlašovací heslo, přihlašovací jméno a nakonec adresa k env-loaderu. První část .launch souboru je vyobrazena zde:

```
<launch>

<machine
   name = "rpi1"
   address = "192.168.0.101"
   password = "machine1"
   user = "ubuntu1"
   env-loader = "/home/ubuntu1/catkin_ws/devel/remote_env_loader.sh">
</machine>

<machine
   name = "rpi2"
   address = "192.168.0.102"
   password = "machine2"
   user = "ubuntu2"
   env-loader = "/home/ubuntu2/catkin_ws/devel/remote_env_loader.sh">
</machine>
```

Druhým krokem bylo spuštění uzlů obou USB kamer na příslušných externích zařízeních. Toto proběhlo pro každou kameru zvlášť – každá kamera měla svoji příslušnou externí stanici. Kamery byly spuštěny při určitých parametrech. Pro ukázku zde byly použity všechny nastavitelné parametry uzlu *uvc_camera_node*, balíčku *uvc_cam*. Za zmínku však stojí především název kamery, který byl důležitý hlavně při kalibraci, kde se právě podle tohoto názvu pojmenoval soubor s kalibrační konfigurací dané kamery. Velkou výhodou je, že v .launch souboru není třeba specifikovat adresu konfigurace kalibrace příslušné kamery. Systém si ji defaultně najde v adresáři */.ros/camera_info/<camera_name>.yaml*. Dále jsou tu rozlišení kamery, autofocus, jas, gamma a saturace. Posledním velmi důležitým parametrem, bylo ID USB zařízení (kamery) připojeného k Raspberry Pi. Jestliže je připojena jenom jedna kamera, hodnota se defaultně nastaví na */dev/video0*. Nakonec je zapotřebí přejmenovat témata na společnou skupinu *stereo*, kvůli správnému fungování kalibrátoru a uzlu *stereo_image_proc*.

Druhá část .launch souboru je kvůli stručnosti a přehlednosti znázorněna pouze pro levou kameru. Pro pravou kameru bude konfigurace analogická. Změní se pouze: název uzlu (name), zařízení (machine), jméno kamery (camera_name) a nakonec v *remap* sekci všechny *stereo/left* na *stereo/right*. Druhá část je vyobrazena zde:

```
<node pkg="uvc_camera" type="uvc_camera_node" name="left" machine = "rpi1">
   <param name="camera_name" type="string" value="left" />
   <param name="frame" type="string" value="left" />
   <param name="fps" type="int" value="30" />
   <param name="skip_frames" type="int" value="0" />

   <param name="width" type="int" value="640" />
   <param name="height" type="int" value="480" />

   <param name="power_line_frequency" value="50"/>
   <param name="auto_exposure" value="0" />
   <param name="auto_gain" value="10" />

   <param name="auto_white_balance" value="0" />
   <param name="brightness" value="80" />
   <param name="backlight_compensation" value="0" />
   <param name="contrast" value="32" />
   <param name="saturation" value="10" />
   <param name="gamma" value="50" />
   <param name="sharpness" value="1" />

   <param name="device" type="string" value="/dev/video0" />
   <remap from="image_raw" to="stereo/left/image_raw" />
   <remap from="camera_info" to="stereo/left/camera_info" />
   <remap from="set_camera_info" to="stereo/left/set_camera_info" /></node>
```

Třetí a poslední část .launch souboru obsahovala spuštění uzlu *stereo_image_proc* a uzlu *image_view*. Uvedené parametry byly získány užitím již zmíněného modulu *rqt_reconfigure*. Tento nástroj umožňuje měnit tyto parametry v reálném čase (Obr. 1-4). Parametry *approximate_sync* a *queue_size* souvisí se synchronizací snímků. Parametr *approximate_sync* je vhodné nastavit na hodnotu „True“, pokud levá a pravá kamera neprodukují snímky přesně synchronizované v čase. Hodnotu *queue_size* se doporučuje zvyšovat, pokud informace o snímku putuje sítí podstatně déle než informace o kameře.

Třetí část .launch souboru, tak jak byla použita ke tvorbě hloubkové mapy:

```
<node name="StereoProc" pkg="stereo_image_proc" type="stereo_image_proc" ns="stereo" >
   <param name="approximate_sync" value="True"/>
   <param name="queue_size" value="5000"/>

   <param name="disparity_range" value="320"/>
   <param name="min_disparity" value="-150"/>
   <param name="correlation_window_size" value="9"/>
   <param name="speckle_range" value="4" />
   <param name="speckle_size" value="100" />
   <param name="texture_treshold" value="10" />
   <param name="uniqueness_ratio" value="15.0" />
</node>
```

```
<node name="hloubkova_mapa" pkg="image_view" type="stereo_view" output="screen">
   <param name="approximate_sync" value="True"/>
   <param name="queue_size" value="5000"/>

   <remap from="stereo" to="stereo"/>
   <remap from="image" to="image_rect"/>
 </node>

</launch>
```

Jedny z nejdůležitějších parametrů při tvorbě hloubkové mapy [65]:

Parametr *disparity_range* představuje rozlišení 3D scény. Čím větší je, tím víc hloubkových úrovní lze rozlišit. Pohybuje se v násobcích 16.

Parametr *min_disparity* lze nastavit kladně i záporně. Kladný je vhodný pro objekty blíž ke kamerám, záporný pro objekty, které jsou od kamer dál.

Parametr *correlation_window_size* určuje velikost korelačního okna odpovídající si dvojici pixelů obou snímků. Hodnoty jsou liché, od 5 do 255. Čím větší hodnota, tím hladší výsledek. Jestliže je příliš velká, mohou se některé objekty z výsledné scény úplně ztratit. Moc malá však může mít za následek jakýsi šum na výsledném obrazu.

Parametry *speckle_range*, *speckle_size*, *texture_treshold*, *uniqueness_ratio* souvisí s konečnou filtrací.

Celý spouštěcí soubor byl tímto způsobem popsán. K nahlédnutí v úplném znění je v příloze k diplomové práci. Při experimentech s třetí kamerou byl vytvořen .launch soubor pro třetí kameru, vytvořen analogicky podle souboru zmíněného výše. Všechny spouštěcí soubory jsou součástí elektronické přílohy k diplomové práci. Mimo spouštěcích souborů je v příloze také internetový odkaz na YouTube video, s názornou ukázkou hloubkové mapy. Jedná se o sestřih videí (přibližně 5 minut).

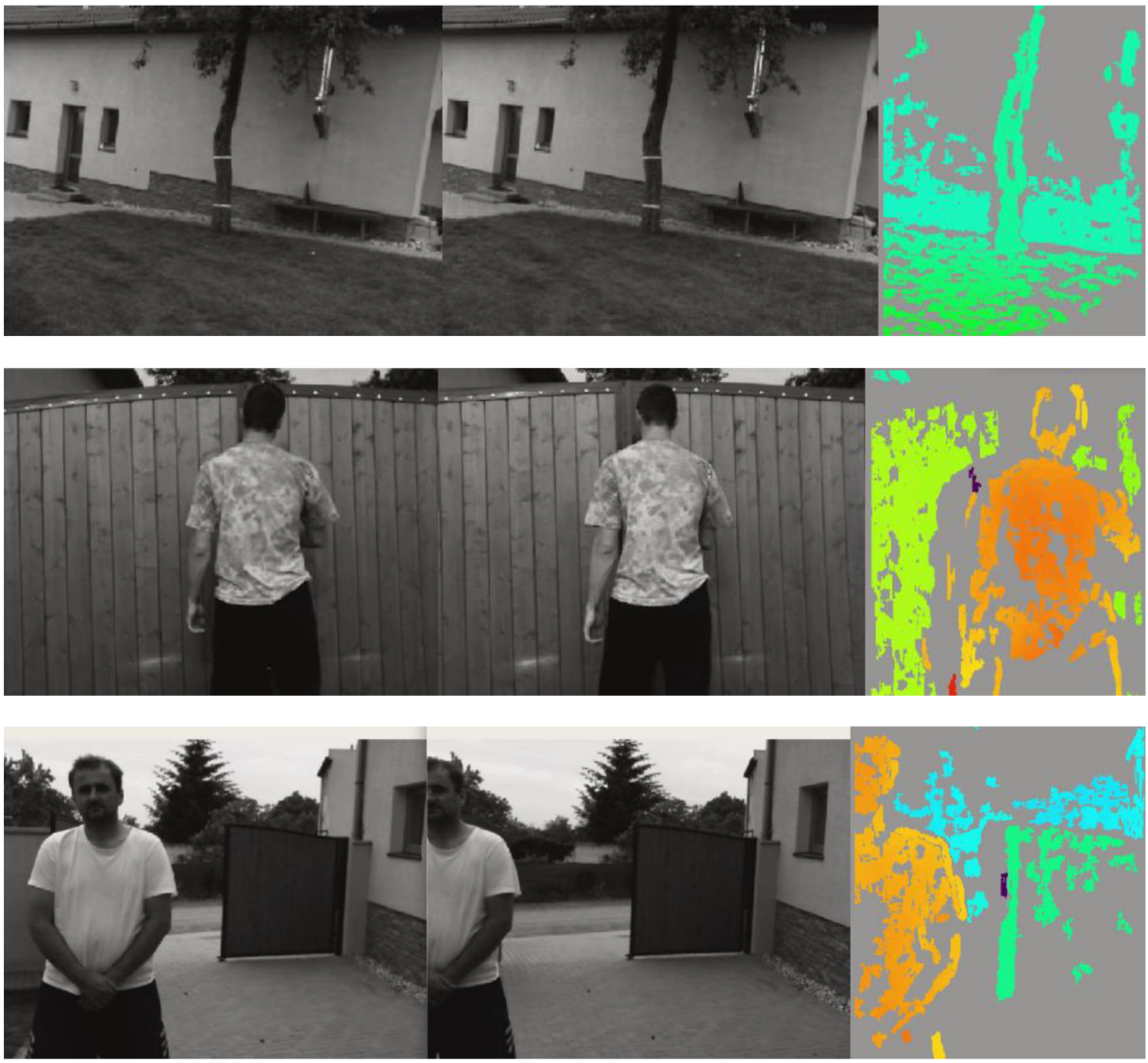
7. Zhodnocení experimentu

Experiment byl proveden pouze na osobním automobilu značky Škoda Felicia Kombi. Řada experimentů byla provedena na stojícím vozidle. Důležité zde bylo zkontrolovat experimentální soustavu včetně všech kamer, provést kalibraci, naladit správné parametry. Kontrolovalo se především elektrické zapojení celé soustavy, umístění kamer, stav baterie laptopu a počasí. Další experimenty byly provedeny při jednoduché manipulaci s vozidlem (například parkování), při objížďce obce (do 50 km/h) nebo na silnici mimo obec (do 90 km/h). Při zkouškách byl model experimentální soustavy občas doplněn o třetí kameru, která snímala pouze směr jízdy. Účelem připojení dodatečné kamery bylo otestování systému (sledování chování) z hlediska výkonu. Sledovala se především snímkovací frekvence (FPS) výsledných obrazů. Doplnění o třetí kameru spočívalo ve spuštění ROS uzlu na třetím Raspberry Pi. K tomu byl připraven .launch soubor, který spustí uzel USB kamery a uzel rotace obrazu (protože kamera byla nalepena „vzhůru nohama“, viz Obr. 4-14). Publikovaný obraz z kamery byl poté zobrazen užitím grafického nástroje *rqt_image_view*. Uzel *image_view*, jak je například použit v .launch souboru v poslední kapitole, způsoboval celkové zpomalení celého systému. S každým spuštěným uzlem *image_view*, snímkovací frekvence všech streamů (včetně hloubkové mapy) citelně klesla. Mluvíme zde o nezanedbatelných cca 15 % původní snímkovací frekvence, zatímco při použití grafického nástroje *rqt_image_view* není poznat žádná změna ani při obrazech ze všech tří kamer. Nutno však podotknout, že oba tyto nástroje slouží hlavně pro vizualizaci při vývoji. U hotové soustavy, kdy se autonomní vozidlo podle získaných dat chová, nemusí být spuštěny žádné vizualizační prostředky. Závěrem pro případ s třetí kamerou tedy je, že se snímkovací frekvence soustavy nezmění ani po připojení třetího počítače Raspberry Pi, který do systému přispěje svým uzlem kamery.

Snímání ze stojícího vozidla spočívalo v provádění série pokusných pohybů před stojícím vozidlem. Lze zde mluvit o simulaci chodce pohybujícího se před vozidlem nebo o „cizím autě“ pohybujícím se před testovacím vozidlem. Realizace probíhala za pomoci figuranta, pohybujícího se před vozidlem. Test se skládal z pohybů do stran, přiblížení nebo vzdálení se od kamery. Kamery zaznamenaly nebezpečnou vzdálenost (silueta člověka na hloubkové mapě nabude červené barvy) v poloze člověka přibližně 30 cm od „nejpřednějšího“ bodu vozidla (přední RZ). K sérii experimentů lze zařadit i snímání vozidla pohybujícího se před naším testovacím vozidlem. Při snímání ze stojícího vozidla se hloubková mapa jevila jako nejkvalitnější, hlavním důvodem je klidový stav vozidla – vzájemná poloha dvou kamer na střeše totiž nebyla ovlivněna otřesy způsobenými nerovností vozovky, protivětrem, brzděním, zrychlením. Případ stojícího vozidla je vhodný pro kalibraci kamer, nebo nastavení konfigurace systému.

Snímání při jednoduché manipulaci s vozidlem se liší pouze minimálně od předchozího příkladu. Pohyby vozidla byly tak malé a rychlost vozidla tak pomalá, že vzájemná poloha dvou kamer nebyla nijak ovlivněna. Kamery spolehlivě detekovaly veškeré překážky před vozidlem. Při experimentech se jednalo zejména o kmen stromu, zeď, přibližování se k jinému vozidlu/člověku nebo také zavřenou bránu. Tento případ může simulovat například parkování. Kromě předku vozidla lze použít další páry kamer, například na zadek auta. Otazníkem u tohoto případu je, zda není k parkování lepší užít jiných senzorů (ultrazvuk apod.). Z osobní zkušenosti mohu říct, že se s touto konfigurací parkuje pohodlně. Jak bylo řečeno v předchozím odstavci, lze relativně přesně zastavit 30 centimetrů před objektem.

Snímání za jízdy bylo zajímavější. Testovací jízdy probíhaly při objížďkách obce a na silnicích. V rámci chodu systému nebyly zjištěny žádné znatelné komplikace. Při celkové snímkovací frekvenci uzlu zobrazujícího hloubkovou mapu, která kolísala kolem 10 FPS, se rychlost mírně projevovala na kvalitě výsledku. Druhým problémem, kromě nízkého FPS, byly nerovnosti vozovky. Otřes jedoucího vozidla, způsobený nerovností vozovky, se přenese na dvě snímající kamery umístěné na střeše. Pozorovatelné chyby jsou také při silnějším zabouchnutí dveří. Tento problém je způsoben tím, že se jedná o dvě samostatné kamery. Jejich vzájemná poloha, napříč všem snahám, nemusí být po celou dobu fixní. Kamera Microsoft LifeCam HD-3000 je umístěna na pohyblivém stojanu, kterým lze kameru natáčet kolem horizontální osy. Objektiv kamery lze také otáčet, a to pouze podle vertikální osy. Tuhost otáčení je však dostatečně velká na to, aby se dalo experimentovat s hloubkovou mapou. Kamery, o kterých pojednává kapitola o stereo kamerách (Kap. 3), zejména: Kinect, Bumblebee nebo ZED, jsou k tomuto účelu vhodnější, a to zejména díky statické vzájemné poloze obou senzorů, zabudovaných v jednom společném krytu. Tyto kamery často přicházejí už předkalibrovány. Kamera ZED je navíc vhodná do vnějších prostor. Protivítr zde překvapivě nehrál žádnou roli, vzhledem k tomu, jak jsou kamery citlivé na sebemenší dotyk/otřes.



Obr. 7-1 Příklady detekce: kmen stromu, člověk stojící blízko před vozidlem

Závěr

Cílem této diplomové práce bylo realizovat kamerový modul jako hardwarově nezávislý senzorický vstup do systému. To všechno za možnosti doplnění systému o další vstupy. Při práci byly využity tři kamery jako nezávislé uzly systému ROS. V rámci sekundárního cíle bylo vytvořit hloubkovou mapu ze dvou obrazů nezávislých kamer s využitím možností frameworku ROS.

Vraťme se k prvnímu bodu teoretické analýzy. ROS poskytuje služby na úrovni operačního systému. Cílem tohoto meta-operačního systému bylo vytvořit určitý standard robotiky, který znatelně ulehčí práci při vytváření různých robotických aplikací. Princip funkce spočívá v komunikaci jednotlivých uzlů mezi sebou. Uzel lze chápat jako script v jazyce C++ nebo Python. Velkou výhodou a zároveň hlavní předností tohoto systému je jeho distribuovaný design a modularita. Uživatel může svůj projekt snadno rozšířit o další části. To bylo také otestováno v rámci této diplomové práce, když byl systém doplněn o třetí nezávislou kameru s příslušným počítačem Raspberry Pi. Připojení třetí platformy do již běžícího systému nijak neovlivnilo chod systému. Jedním z důležitých aspektů frameworku ROS je také široká komunita uživatelů z celého světa. Tito uživatelé se podílejí na tvorbě nových balíčků, rozšiřují webové stránky wiki.ros.org a přispívají svým řešením různých problémů na server answers.ros.org. Za jednu z mála nevýhod frameworku ROS lze považovat plnou podporu pouze pro jeden operační systém, tím je Ubuntu. Druhý bod analýzy rozdělil metody tvorby hloubkových map na aktivní a pasivní. Z druhého bodu vyplývá, že aktivní jsou ty metody, kde je do scény přidána dodatečná informace. Touto přídavnou informací se rozumí například laser, infračervený zářič nebo projektor. Vzhledem k tomu, že se jedná o světelnou informaci, jsou tyto metody vhodné spíše pro slabě osvětlené prostory. Jako příklady aktivních metod jsou uvedeny metoda aktivní triangulace a měření doby letu. Pasivní metody se liší od aktivních tím, že místo dodatečné světelné informace mají druhou kameru. Jedná se tedy o systém dvou kamer. Hloubku prostorové scény vyhodnocují podobně jako lidské oči. Jako příklad je uvedena metoda pasivní triangulace. Třetí bod analýzy se zabývá stereo kamerami, používanými v robotice. Cílem kapitoly bylo poskytnout přehled stereo kamer populárních v robotice. Kapitolu lze zároveň chápat jako porovnání různých variant, včetně posledního bodu kapitoly, který je věnován řešení stereo vize pomocí Raspberry Pi. Tento jednoduchý nápad ruských vývojářů dokáže plně nahradit drahé varianty, jako je Kinect, nebo ZED kamera. Funguje spolehlivě při vysokém rozlišení a za vysoké snímkovací frekvence. Lze ho uplatnit s kamerami běžně dostupnými pro Raspberry Pi. Doporučuje se však užití Raspberry Pi V1 kamery a Waveshare model G kamery.

Z hlediska praktické části proběhl experiment bez jakýchkoli komplikací. Mezi použitý hardware patří tři jednodeskové počítače Raspberry Pi 3 Model B (slave stanice), tři USB kamery (2× Microsoft LifeCam, 1× Sony PlayStation3 Eye Camera), Wi-Fi router a laptop (master stanice). Při instalaci softwaru na jednotlivé počítače Raspberry Pi se jako nejrychlejší jevila konfigurace 32bit Ubuntu server 18.04, nainstalován na zmíněné SD kartě SanDisk Ultra A1, 16 GB. Po upravení .bashrc souborů na každém počítači v soustavě a po vytvoření *remote_env_loader.sh* scriptů bylo možné realizovat propojení všech platforem v rámci systému ROS. Nezávislosti kamer bylo dosaženo připojením každé kamery na samostatný jednodeskový počítač Raspberry Pi, kde každý z těchto počítačů byl ethernetem připojen na Wi-Fi router (Obr. 4-10). Komunikace mezi uzly jednotlivých zařízení byla bezproblémová. Kvůli absenci RTC modulu u Raspberry Pi bylo nutno tyto počítače časově synchronizovat s master stanicí. Software k tomu určený se jmenuje chrony. Pomocí chrony

bylo možno tyto počítače synchronizovat s přesností mikro až nanosekund. Následně byla provedena kalibrace kamer a poté hloubková mapa.

Celkově se testovaly tři pohybové stavy vozidla: stojící vozidlo, jednoduchá manipulace, jízda. Při testování poskytovaly dvě kamery na střeše vozidla snímky pro zpracování hloubkové mapy na master stanici. Systém běžel bez komplikací. V průběhu testování bylo připojeno k experimentální soustavě třetí Raspberry Pi i s kamerou (třetí kamera). Následně byl spuštěn uzel USB kamery. Třetí kamera přilepená na čelním skle vozidla snímala okolí ve směru jízdy. Po připojení třetího kamerového modulu a následném spuštění uzlu na příslušné platformě se průběh testování hloubkové mapy nijak znatelně nezměnil. Uzel USB kamery totiž běžel na externí stanici (třetí Raspberry Pi). Master stanici stačí tedy pouze informace o běžícím uzlu, pokud nemluvíme o vizualizaci na ploše master stanice. Vizualizace výsledků probíhala poněkud jinak. Vizualizace poskytovaných obrazů každé kamery byla provedena užitím nástroje *rqt_image_view*. Původní varianta v podobě uzlu *image_view* nedosahovala tak dobrých výsledků. S každým spuštěním uzlu *image_view* bylo dosaženo 15% zpomalení snímkovací frekvence zobrazované hloubkové mapy.

Celková kvalita hloubkové mapy závisela na vnějších podmínkách (počasí, slunce, nerovnosti vozovky apod.). Hloubková mapa spolehlivě detekovala blízké, ale i vzdálenější objekty. Na základě série pokusů znamenala červenající se silueta člověka vzdálenost snímaného člověka 30 cm od přední RZ vozidla (bod, který je na vozidle nejvíce vepředu).

Seznam použitých zdrojů

[1] Introduction. *ROS Wiki: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/ROS/Introduction

[2] Player logo. In: *The Player Project* [online]. The Player Project [cit. 2020-06-15]. Dostupné z: http://playerstage.sourceforge.net/

[3] YARP logo. In: *YARP* [online]. [cit. 2020-06-15]. Dostupné z: https://www.yarp.it/

[4] Orocos Project logo. In: *The Orocos Project: Smarter control in robotics & automation!* [online]. [cit. 2020-06-15]. Dostupné z: https://www.orocos.org/

[5] CARMEN logo. In: *CARMEN: Robot Navigation Toolkit* [online]. [cit. 2020-06-15]. Dostupné z: http://carmen.sourceforge.net/

[6] Orca logo. In: *Orca: Components for Robotics* [online]. [cit. 2020-06-15]. Dostupné z: http://orca-robotics.sourceforge.net/

[7] The MOOS logo. In: *The MOOS: Cross Platform Software for Robotics Research* [online]. [cit. 2020-06-15]. Dostupné z: http://www.robots.ox.ac.uk/~mobile/MOOS/wiki/pmwiki.php/Main/HomePage

[8] Microsoft Robotics Developer Studio logo. In: *Wikipedia: the free encyclopedia* [online]. [cit. 2020-06-15]. Dostupné z: https://en.wikipedia.org/wiki/Microsoft_Robotics_Developer_Studio

[9] Installation. *ROS Wiki: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/Installation

[10] Is ROS For Me? *ROS: Powering the world's robots* [online]. [cit. 2020-06-15]. Dostupné z: https://www.ros.org/is-ros-for-me/

[11] Metrics. *ROS Wiki: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/Metrics

[12] FOOTE, Tully. Community Metrics Report. In: *ROS* [online]. Červenec 2019 [cit. 2020-06-15]. Dostupné z: http://download.ros.org/downloads/metrics/metrics-report- 2019-07.pdf

[13] Distributions. *ROS Wiki: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/Distributions

[14] Core Components. *ROS: Powering the world's robots* [online]. [cit. 2020-06-15]. Dostupné z: https://www.ros.org/core-components/

[15] Topics. *ROS Wiki: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/Topics

[16] Concepts. *ROS: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/ROS/Concepts

[17] Services. *ROS: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/Services

[18] Parameter Server. *ROS: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/Parameter%20Server

[19] *Tvorba prostorového obrazu a zisk 3D informace* [online]. Brno [cit. 2020-06-15]. Dostupné z: http://www.urel.feec.vutbr.cz/web_documents/studium/predmety/MVDK/FRVSg/kompletni1.pdf. Elektronická podpora. Fakulta elektrotechniky a komunikačních technologií VUT v Brně.

[20] NĚMEC, Pavel. *Hloubková mapa s využitím zařízení Kinect* [online]. Brno, 2017 [cit. 2020-06-15]. Dostupné z: https://dspace.vutbr.cz/bitstream/handle/11012/67644/final-thesis.pdf?sequence=8&isAllowed=y. Bakalářská práce. Fakulta elektrotechniky a komunikačních technologií VUT v Brně. Vedoucí práce Libor Boleček.

[21] KALOVÁ, Ilona a Karel HORÁK. *Optické metody měření 3D objektů* [online]. 12.04.2005 [cit. 2020-06-15]. Dostupné z: http://www.elektrorevue.cz/clanky/05023/index.html#kap2.1.1

[22] NGUYEN, Hubert. What Is a ToF Camera? In: *Übergizmo* [online]. 04.04.2019 [cit. 2020-06-15]. Dostupné z: https://www.ubergizmo.com/articles/what-is-tof-camera-sensor/

[23] EVANS, Laura. What is pupillary distance and how to measure it. In: *All About Vision* [online]. Srpen 2019 [cit. 2020-06-15]. Dostupné z: https://www.allaboutvision.com/eye-care/measure-pupillary-distance/

[24] VORÁČKOVÁ, Šárka. *Aplikace epipolární geometrie* [online]. [cit. 2020-06-15]. Dostupné z: https://adoc.tips/2-rekonstrukce-ze-dvou-kalibrovanych-pohled.html

[25] PSAHL, Thomas. Lecture 7.2 Triangulation.UiO Universitetet i Oslo[online].2016-3-7 [cit. 2020-06-15]. Dostupné z:https://www.uio.no/studier/emner/matnat/its/UNIK4690/v16/forelesninger/lecture_7_2-triangulation.pdf.

[26] BRUCE, David. *3D strojové vidění pro průmyslové roboty* [online]. 05.06.2019 [cit. 2020-06-15]. Dostupné z: https://www.mmspektrum.com/clanek/3d-strojove-videni-pro-prumyslove-roboty.html

[27] Bradley Driving in Stereo. *Army Guide* [online]. 06.07.2018 [cit. 2020-06-15]. Dostupné z: http://www.army-guide.com/eng/article/article_3231.html

[28] CONG, Robert a Ryan WINTERS. *How Does The Xbox Kinect Work* [online]. [cit. 2020-06-15]. Dostupné z: https://www.jameco.com/Jameco/workshop/howitworks/xboxkinect.html

[29] GIORI, Clemente. *Kinect in Motion – Audio and Visual Tracking by Example* [online]. Packt Publishing, 25. 4. 2013n. l. [cit. 2020-06-15]. Dostupné z: https://books.google.sk/books?id=zLRgqqYB5pcC&printsec=frontcover&hl=cs#v=onepage&q=Natal&f=false

[30] Fastest-selling gaming peripheral. *Guinness World Records* [online]. Leden 2011 [cit. 2020-06-15]. Dostupné z: https://www.guinnessworldrecords.com/world-records/fastest-selling-gaming-peripheral

[31] HAN, Trann. Microsoft Kinect. In: *Nhan Tran* [online]. [cit. 2020-06-15]. Dostupné z: https://www.trannhan.com/project/simon-says-interactive-kinect-ros-turtlebot/

[32] TurtleBot. In: *ROBOTS: Your Guide to the World of Robotics* [online]. [cit. 2020-06-15]. Dostupné z: https://robots.ieee.org/robots/turtlebot/

[33] SZYMCZYK, Matthew. How Does The Kinect 2 Compare To The Kinect 1? *Zugara* [online]. 09.12.2014 [cit. 2020-06-15]. Dostupné z: http://zugara.com/how-does-the-kinect-2-compare-to-the-kinect-1

[34] The difference between Kinect v2 and v1. *The Ghost Howls* [online]. 02.12.2016 [cit. 2020-06-15]. Dostupné z: https://skarredghost.com/2016/12/02/the-difference-between-kinect-v2-and-v1/

[35] Stereo Products. *Voltrium Systems: Machine Vision Components Provider* [online]. [cit. 2020-06-15]. Dostupné z: https://voltrium.wordpress.com/machine-vision/home/stereo-products/

[36] WINKLEHNER, Thomas. Mobile robotic platform with Bumblebee 2 stereo camera. In: *ResearchGate* [online]. Listopad 2009 [cit. 2020-06-15]. Dostupné z: https://www.researchgate.net/figure/a-Mobile-robotic-platform-with-Bumblebee-2-stereo-camera-b-Pre-processing-pipeline_fig1_220844639

[37] Bumblebee: STEREO VISION CAMERA SYSTEMS. In: *Universitat Politècnica de Catalunya* [online]. [cit. 2020-06-15]. Dostupné z: https://www.upc.edu/sct/ca/documents_equipament/d_186_id-488.pdf

[38] Meet ZED. *Stereo Labs* [online]. [cit. 2020-06-15]. Dostupné z: https://www.stereolabs.com/zed/

[39] Getting Started with ROS and ZED. *Stereo Labs* [online]. [cit. 2020-06-15]. Dostupné z: https://www.stereolabs.com/docs/ros/

[40] SEROV, Sergey a Kirill SHIRYAEV. StereoPi: An open source stereoscopic camera based on Raspberry Pi. *Crowd Supply* [online]. [cit. 2020-06-15]. Dostupné z: https://www.crowdsupply.com/virt2real/stereopi

[41] StereoPi. In: *Raspberrypi.org* [online]. 28.06.2018 [cit. 2020-06-15]. Dostupné z: https://www.raspberrypi.org/forums/viewtopic.php?t=216940

[42] Camera Module v1.3 5Mpix for Raspberry Pi. In: *Nettigo* [online]. [cit. 2020-06-15]. Dostupné z: https://nettigo.eu/products/camera-module-v1-3-5mpix-for-raspberry-pi

[43] Waveshare Raspberry Pi Camera. In: *RLX* [online]. [cit. 2020-06-15]. Dostupné z: https://rlx.sk/sk/raspberry-pi/4873-rpi-camera-g-fisheye-lens-waveshareraspberry-pi-camera-module-fisheye-lens-wider-field-of-view.html

[44] Raspberry Pi Compute Module 3+. In: *Farnell* [online]. [cit. 2020-06-15]. Dostupné z: https://sk.farnell.com/productimages/large/en_GB/2989461-40.jpg

[45] Microsoft LifeCam HD-3000. In: *TSBohemia* [online]. [cit. 2020-06-15]. Dostupné z: https://www.tsbohemia.cz/microsoft-lifecam-hd-3000_d161915.html?fulltextword=microsoft%20lifecam

[46] Sony PlayStation 3 Eye Camera with EyeCreate. *Amazon* [online]. [cit. 2020-06-15]. Dostupné z: https://www.amazon.co.uk/Sony-PlayStation-Eye-Camera-EyeCreate/dp/B000W3YQ1Y

[47] Raspberry Pi 3 Model B. *Raspberrypi.org* [online]. [cit. 2020-06-15]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[48] Raspberry Pi 3 Model B. In: *Robodyne* [online]. [cit. 2020-06-15]. Dostupné z: http://www.robo-dyne.com/en/shop/raspberry-pi-3/

[49] Bezdrátový N router 300Mbit/s: TL-WR840N. *TP-link* [online]. [cit. 2020-06-15]. Dostupné z: https://www.tp-link.com/cz/home-networking/wifi-router/tl-wr840n/

[50] Acer Nitro 5. In: *Benchmark* [online]. [cit. 2020-06-15]. Dostupné z: https://www.benchmark.pl/produkt/acer-nitro-5-nh-q3lep-001-240gb-m-2-1tb-

[51] Buck Step Down měnič DC 6-24V 12V/24V na 5V 3A USB. *Arduino-Shop* [online]. [cit. 2020-06-15]. Dostupné z: https://arduino-shop.cz/arduino/2036-buck-step-down-menic-dc-6-24v-12v-24v-na-5v-3a-usb.html?mena=2

[52] How can I power my 9 volts DC internet router via a 12v battery? *Quora* [online]. [cit. 2020-06-15]. Dostupné z: https://www.quora.com/How-can-I-power-my-9-volts-DC-internet-router-via-a-12v-battery

[53] Ubuntu install of ROS Melodic. *ROS: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/melodic/Installation/Ubuntu

[54] ROS Command-line tools. *ROS: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/ROS/CommandLineTools

[55] Roslaunch. *ROS: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/roslaunch

[56] Creating a ROS Package. *ROS: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/ROS/Tutorials/CreatingPackage

[57] SanDisk Ultra A1 16GB. In: *Mall.cz* [online]. [cit. 2020-06-15]. Dostupné z: https://www.mall.cz/i/39856953/1000/1000

[58] Raspberry Pi imager. *Raspberrypi.org* [online]. Raspberry Pi Foundation [cit. 2020-06-15]. Dostupné z: https://www.raspberrypi.org/downloads/

[59] A is not in your SSH known_hosts file. In: *ROS Answers* [online]. 30.08.2012 [cit. 2020-06-15]. Dostupné z: https://answers.ros.org/question/41446/a-is-not-in-your-ssh-known_hosts-file/

[60] NetworkSetup. *ROS Wiki: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/ROS/NetworkSetup

[61] Remote Machines Running ROS nodes. *GitHub: Wiki* [online]. [cit. 2020-06-15]. Dostupné z: https://github.com/pandora-auth-ros-pkg/pandora_docs/wiki/Remote-Machines-Running-ROS-nodes

[62] Introduction. *Chrony* [online]. [cit. 2020-06-15]. Dostupné z: https://chrony.tuxfamily.org/

[63] Manual for version 2.3: Isolated networks. *Chrony* [online]. [cit. 2020-06-15]. Dostupné z: https://chrony.tuxfamily.org/manual.html#Isolated-networks

[64] How to Calibrate a Stereo Camera. *ROS: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/camera_calibration/Tutorials/StereoCalibration

[65] MIHELICH, Patrick, Kurt KONOLIGE a Jeremy LEIBS. Stereo_image_proc. *ROS: Documentation* [online]. [cit. 2020-06-15]. Dostupné z: http://wiki.ros.org/stereo_image_proc

[66] Kalibrace kamer. In: *Youtube* [online]. 24.06.2020 [cit.2020-06-24]. Dostupné z: https://www.youtube.com/watch?v=h0_jUSO4u4o. Kanál uživatele Ladislav Albrecht.

[67] Spuštění systému. In: *Youtube* [online]. 24.06.2020 [cit.2020-06-24]. Dostupné z: https://www.youtube.com/watch?v=p3CH0UUhbj0. Kanál uživatele Ladislav Albrecht.

[68] Hloubková mapa. In: *Youtube* [online]. 24.06.2020 [cit.2020-06-24]. Dostupné z: https://www.youtube.com/watch?v=5HHxFmdHRZw. Kanál uživatele Ladislav Albrecht.

Seznam obrázků

Seznam tabulek

Elektronické přílohy

Součástí elektronické přílohy k diplomové práci v souborovém formátu .zip je:

- Složka pracovního prostředí *catkin_ws*, včetně vytvořeného balíčku s názvem *albrecht_dp* a všech spouštěcích souborů (.launch) potřebných k vykonání experimentu.

- Složka obsahující konfigurační soubory potřebné pro časovou synchronizaci všech zařízení na lokální síti, která nemá přístup k internetu. Bližší instrukce jsou k nalezení v textovém dokumentu *readme.txt*, který je rovněž součástí složky.

- Složka obsahující výsledky stereo kalibrace dvou kamer. Přítomny jsou také fotografie, které si ROS kalibrátor ukládal pro následný výpočet.

- Textový dokument *internetové_odkazy_na_videa.txt*, který obsahuje odkazy na příslušná videa. Jedná se o:
    - video kalibrace kamer [66]
    - video spuštění systému [67]
    - sestřih videí hloubkové mapy [68]