VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KALIBRACE ROBOTICKÉHO PRACOVIŠTĚ

CALIBRATION OF ROBOTIC WORKSPACE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JAN UHLÍŘ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL KAPINUS**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Uhlíř Jan, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Kalibrace robotického pracoviště**
Calibration of Robotic Workspace
Kategorie: Uživatelská rozhraní

Zadání:

1. Prostudujte dostupné prostředky pro kalibraci 2D a 3D kamery a robotického manipulátoru v rámci scény.
2. Seznamte se s frameworkem ROS a jeho možnostmi v oblasti plánování pohybu robotického manipulátoru.
3. Vyberte vhodné metody a nástroje a navrhněte aplikaci umožňující kalibraci 2D a 3D kamery a robotického manipulátoru.
4. Navržené rozhraní implementujte a integrujte do robotického systému ARCOR.
5. Proveďte experimenty, demonstrujte a diskutujte vlastnosti vašeho řešení.
6. Vytvořte stručný plakát nebo video prezentující vaši diplomovou práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních tří bodů zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Kapinus Michal, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 16. ledna 2019

Abstrakt

Tato práce se zabývá problematikou kalibrace robotického pracoviště, zahrnující zaměření kalibračního objektu za účelem vzájemné kalibrace 2D nebo 3D kamery, robotického ramene a scény robotického pracoviště. Nejprve byla nastudována problematika týkající se kalibrací již zmíněných prvků. Dále byla provedena analýza vhodných metod realizujících tyto kalibrace. Výsledkem práce je aplikace robotického systému ROS poskytující metody pro tři různé typy kalibračních programů, jejichž funkcionalita je v závěru této práce experimentálně ověřena.

Abstract

This work is concerned by the issue of calibrating a robotic workplace, including the localization of a calibration object for the purpose of calibrating a 2D or 3D camera, a robotic arm and a scene of robotic workplace. At first, the problems related to the calibration of the aforementioned elements were studied. Further, an analysis of suitable methods for performing these calibrations was performed. The result of this work is application of ROS robotic system providing methods for three different types of calibration programs, whose functionality is experimentally verified at the end of this work.

Klíčová slova

kalibrace, mechanická kalibrace, kalibrace kamery, robotické pracoviště, robotický manipulátor, robotika, ROS, ROS-Indrustrial, MoveIt!, tf, OpenCV, ArUco, ARCOR, Template matching, ICP, PCL, point cloud

Keywords

calibration, mechanical calibration, camera calibration, robotic workspace, robotic manipulator, robotics, ROS, ROS-Indrustrial, MoveIt!, tf, OpenCV, ArUco, ARCOR, Template matching, ICP, PCL, point cloud

Citace

UHLÍŘ, Jan. *Kalibrace robotického pracoviště*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Kapinus

Kalibrace robotického pracoviště

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Kapinuse. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Uhlíř
21. května 2019

Poděkování

Touto cestou bych rád poděkoval vedoucímu mé diplomové práce Ing. Michalovi Kapinusovi, za jeho rady, ochotu, motivaci a čas věnovaný konzultacím.

Obsah

1 Úvod **3**

2 Kalibrace kamery a robotického ramene **4**
2.1 Kalibrace robotického ramene 4
2.2 Kalibrace vnitřních parametrů kamery 4
2.3 Kalibrace vnějších parametrů kamery 8
2.4 Vzájemná kalibrace kamery a robotického ramene 9

3 Robotický systém **11**
3.1 Obecný robotický systém 11
3.2 Průmyslový robot 12
3.3 Koncový efektor 12
3.4 Pracovní prostor 12
3.5 Kolizní prostor 13
3.6 Kinematika 13

4 Robotický software **14**
4.1 ROS 14
4.2 ROS-Industrial 16
4.3 MoveIt! 16
4.4 Podpůrné nástroje 18
4.5 ARCOR 20

5 Analýza existujících metod **22**
5.1 Automatický výpočet transformační matice 22
5.2 OpenCV Camera calibration 24
5.3 ArUco 25
5.4 Template matching 27
5.5 ICP 28

6 Návrh aplikace a metod pro kalibraci **30**
6.1 Definice problému 30
6.2 Návrh aplikace 30
6.3 Mechanická metoda 31
6.4 Metoda využívající 2D kameru 33
6.5 Metoda využívající 3D kameru 34

7 Implementace **36**

7.1 Struktura aplikace 36
7.2 Mechanická kalibrace 40
7.3 Kalibrace 2D kamery 44
7.4 Kalibrace 3D kamery 45
7.5 Integrace do systému ARCOR 46

8 Testování **48**
8.1 Scéna robotického pracoviště 48
8.2 Experimenty 52
8.3 Náměty k rozšíření 64

9 Závěr **66**

Literatura **67**

A Struktura zpráv **69**

Kapitola 1

Úvod

Hlavním cílem této diplomové práce je prostudovat dostupné prostředky pro vzájemnou kalibraci 2D nebo 3D kamery, robotického ramene a scény robotického pracoviště. Dále seznámit se s frameworkem `ROS` a jeho možnostmi v oblasti plánování pohybu robotického manipulátoru. Následně na základě získaných znalostí vybrat vhodné metody a nástroje a navrhnout aplikaci umožňující automatickou vzájemnou kalibraci kamer a robotického ramene. V dalším kroku byly zvolené metody implementovány do výsledné aplikace pro systém `ROS` sloužící ke kalibraci robotického pracoviště. Jedním z bodů implementace aplikace je také integrace do robotického systému `ARCOR`.

Návrh a tvorba této aplikace je motivována především snahou o zjednodušení celého procesu kalibrace robotického pracoviště a také snahou minimalizovat míru účasti člověka na tomto procesu. Jejím hlavním účelem je především zjistit vzájemné vztahy mezi souřadným systémem scény robotického pracoviště a vlastním souřadným systémem robotického manipulátoru, 2D nebo 3D kamery, skrze kalibrační objekt. To následně umožní jednoduché přiřazení takového libovolného prvku do již existující scény, přičemž v rámci ní nebude vázán na jedno konkrétní umístění.

Nejprve bude popsána problematika vzájemné kalibrace kamery a robotického ramene. Následovat bude popis robotického systému se všemi jeho součástmi. Dále bude popsána kapitola věnující se robotickému frameworku `ROS` a jeho možnostmi v oblasti plánování pohybu robotického ramene. Poté zde budou popsány jednotlivé vhodné metody realizující zmíněné kalibrace. Další kapitola se bude věnovat návrhu samotné aplikace a jednotlivých kalibračních metod navržených na základě analýzy v předchozí kapitole. Budou zde nastíněny modelové situace, pro které by navržené metody měly nalézt uplatnění, společně se způsobem jakým by toho měly dosáhnout. Následně bude popsáno, jak probíhala implementace výsledné aplikace. Zaměříme se podrobněji na její strukturu a jednotlivé kalibrační programy. Na závěr bude popsána sada experimentů, kde budou diskutovány vlastnosti výsledného řešení.

Kapitola 2

Kalibrace kamery a robotického ramene

Tato kapitola je věnována problematice vzájemné kalibrace kamery a robotického ramene. Postupně zde bude popsána kalibrace robotického ramene, společně s kalibrací vnitřních a vnějších parametrů kamery.

2.1 Kalibrace robotického ramene

Kalibrací robotického ramene získáme transformaci mezi jeho vnitřním souřadným systémem a souřadným systémem existující scény, uvnitř které se bude robot nacházet. Za účelem zjištění takové transformace je nutné zjistit vzájemnou rotaci a translaci mezi počátky těchto souřadných systémů. Získaná transformace nám následně dovoluje navádět robota pomocí souřadnic platných v rámci této scény. Tato transformace je dána následovně:

$$m' = [R|t]M' \tag{2.1}$$

$$\begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ 1 \end{bmatrix} \tag{2.2}$$

Kde:

- *m'* je vektor souřadnic bodu v souřadném systému robotického manipulátoru se souřadnicemi X_r, Y_r, Z_r
- $[R/t]$ jsou rotační matice a translační vektor popisující transformaci mezi souřadným systémem robotického manipulátoru a souřadným systémem scény
- *M'* je vektor souřadnic bodu v souřadném systému scény se souřadnicemi X_s, Y_s, Z_s

2.2 Kalibrace vnitřních parametrů kamery

Kalibrací kamery lze určit vztah mezi jednotkami typickými pro kameru (pixely) a jednotkami reálného světa (například milimetry). Vnitřní parametry kamery lze popsat jako sadu

parametrů určující její vnitřní vlastnosti a chování [11]. Popisují především její ohniskovou vzdálenost a polohu hlavního bodu kamery. Tyto parametry jsou dány ve formě matice a slouží pro transformaci souřadnic ze souřadného systému kamery do souřadného systému průmětny. Tato transformace je dána následovně:

$$m' = AM' \tag{2.3}$$

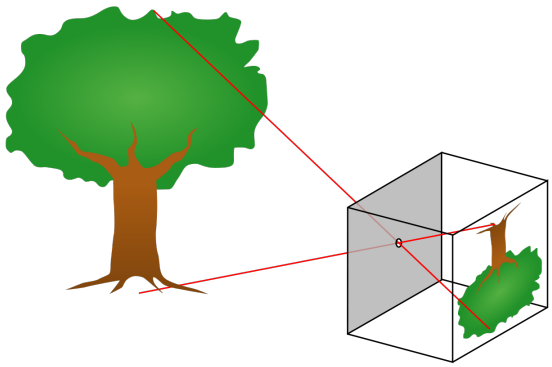
$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ z \end{bmatrix} \tag{2.4}$$

Kde:

- *m'* je vektor souřadnic sledovaného bodu v souřadném systému průmětny
- *A* je matice vnitřních parametrů kamery
  - f_x, f_y udávají ohniskovou vzdálenost v jednotkách kamery (pixely)
  - c_x, c_y určují hlavní bod
- *M'* je vektor souřadnic promítaného bodu v soustavě souřadnic kamery

Matematický model kamery

Používá se jednoduchý model kamery, označovaný jako *dírková kamera*, neboli *pinhole camera*. V tomto modelu dopadá paprsek světla na právě jeden bod projekční desky skrze úzký otvor kamery. Na projekční desce, která je u reálné kamery tvořena snímacím čipem, je zachycený obraz zobrazen středově. Velikost zobrazení zachycené scény je dána pouze ohniskovou vzdáleností *f*.

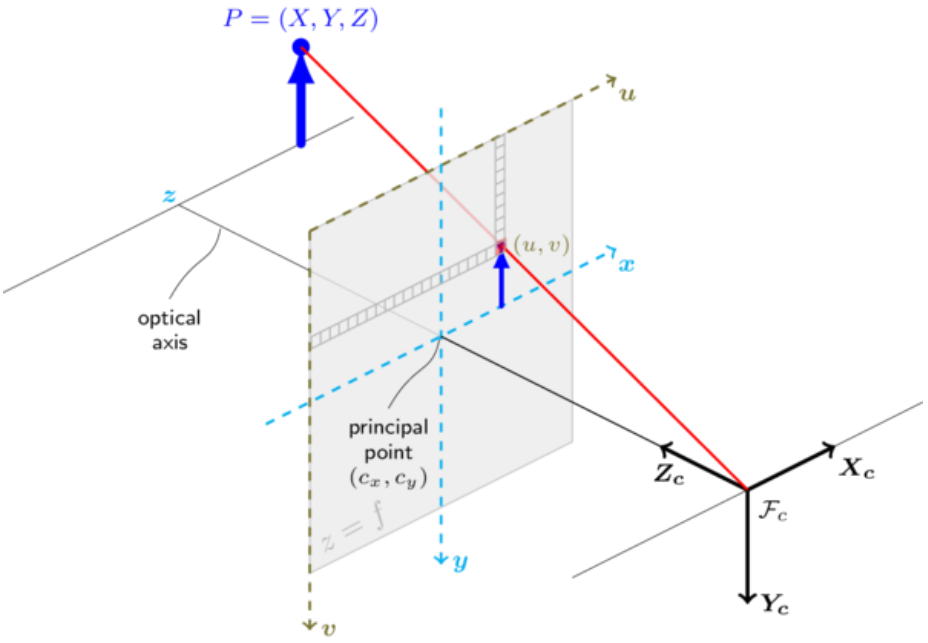


Obrázek 2.1: Princip dírkové kamery [1]

[1] Převzato z: https://cs.wikipedia.org/wiki/Soubor:Pinhole-camera.svg

Promítání souřadnic bodů na průmětnu

Vzdálenost průmětny od počátku souřadného systému kamery F_c je dána ohniskovou vzdáleností ve směru optické osy Z_c. Průnik průmětny s optickou osou se nazývá *hlavní bod*. Na obrázku 2.2 je zobrazen model promítání kartézských souřadnic bodů do obrazové roviny scény pomocí transformace perspektivy.



Obrázek 2.2: Model promítání souřadnic bodů na průmětnu [2]

Tato transformace je dána následovně:

$$sm' = A[R|t]M' \tag{2.5}$$

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ z \\ 1 \end{bmatrix} \tag{2.6}$$

Kde:

- s je faktor zvětšení
- m' je vektor souřadnic sledovaného bodu v zobrazovací rovině se souřadnicemi u,v v souřadném systému průmětny
- A je matice vnitřních parametrů kamery
  - f_x, f_y udávají ohniskovou vzdálenost v jednotkách kamery (pixely)

[2] Převzato z: https://docs.opencv.org/2.4/modules/calib3d/doc/

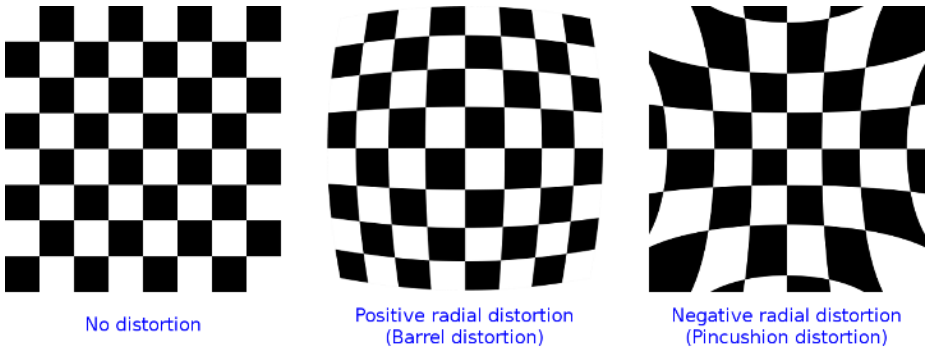
  - c_x, c_y určují hlavní bod
- $[R/t]$ jsou rotační matice a translační vektor popisující vnější parametry kamery
- M' je vektor souřadnic promítaného bodu v soustavě souřadnic reálného světa

Zkreslení obrazu

Narozdíl od jednoduchého modelu *dírkové kamery* dochází u skutečné kamery ke zkreslení výsledného obrazu vlivem fyzikálních a optických vlastností reálného světa. Nejčastěji uvažované je radiální a tangenciální zkreslení.

Radiální zkreslení

Radiální zkreslení je způsobeno nedokonalým tvarem čoček kamery. V tomto případě je úhel vystupujícího paprsku odlišný od úhlu vstupujícího paprsku a poloha zobrazeného bodu se ve výsledku mírně liší od skutečné polohy. Toto zkreslení je v hlavním bodě prakticky nulové a roste společně se vzdáleností od tohoto bodu, jak lze vidět na obrázku 2.3.



Obrázek 2.3: Ukázka radiálního zkreslení obrazu [3]

Pro eliminaci radiálního zkreslení se v rámci knihovny `OpenCV` používají následující vzorce:

$$x' = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.7}$$

$$y' = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.8}$$

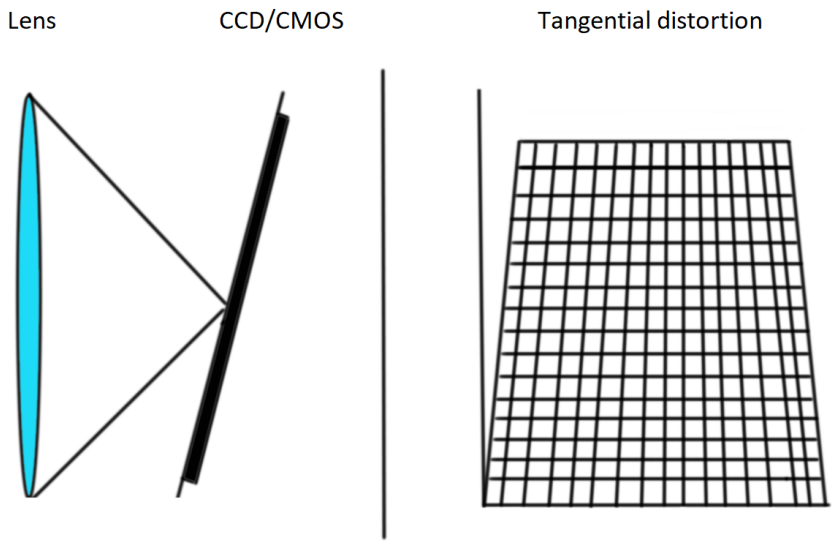
Kde:

- x, y jsou souřadnice v souřadném systému průmětny
- k_1, k_2, k_3 jsou koeficienty radiálního zkreslení
- x', y' jsou opravné souřadnice bez radiálního zkreslení

[3] Převzato z: https://docs.opencv.org/2.4/modules/calib3d/doc/

Tangenciální zkreslení

K tangenciálnímu zkreslení dochází z důvodu nedokonalosti konstrukce kamery. Nastává v případě, kdy obrazové objektivy nejsou dokonale paralelní se zobrazovací rovinou. Tento případ je znázorněn na obrázku 2.4. To způsobí odchylky v místech dopadu paprsku světla, což má za následek zkreslení ve formě odchýlení zobrazených poloh bodů od jejich skutečných poloh.



Obrázek 2.4: Princip tangenciálního zkreslení

V rámci knihovny `OpenCV` je tangenciální zkreslení eliminováno pomocí vzorce:

$$x' = x + [2p_1xy + p_2(r^2 + 2x^2)] \tag{2.9}$$

$$y' = y + [p_1(r^2 + 2y^2) + 2p_2xy] \tag{2.10}$$

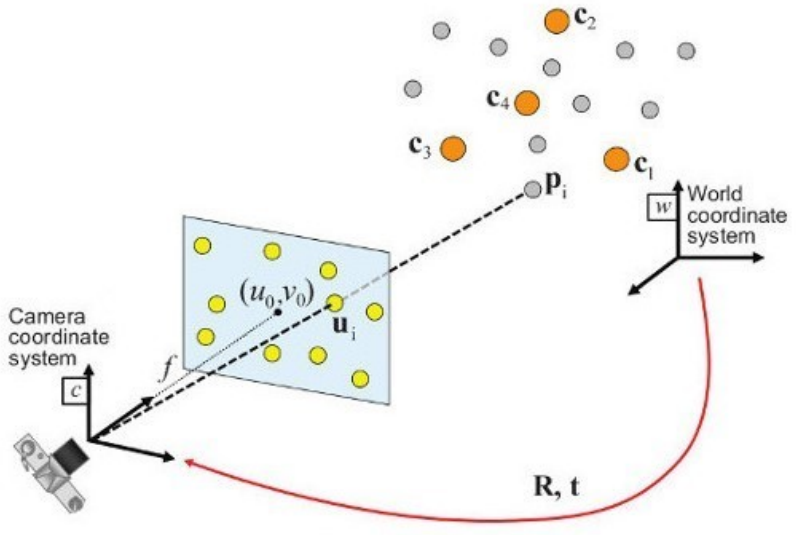
Kde:

- x, y jsou souřadnice v souřadném systému průmětny
- r je vzdálenost od počátku souřadného systému průmětny
- p_1, p_2 jsou koeficienty tangenciálního zkreslení
- x', y' jsou opravné souřadnice bez tangenciálního zkreslení

2.3 Kalibrace vnějších parametrů kamery

Narozdíl od kalibrace vnitřních parametrů kamery popsané v kapitole 2.2, u kalibrace vnějších parametrů kamery jde o definování polohy a orientace kamery v prostoru. Na základě

této kalibrace lze následně zjistit transformaci souřadnic bodů mezi souřadným systémem kamery a souřadným systémem reálného světa, jak je znázorněno na obrázku 2.5.



Obrázek 2.5: Transformace mezi souřadným systémem kamery a reálného světa [4]

Vztah pro tuto transformaci je daný následovně:

$$M' = [R|t]w \tag{2.11}$$

$$\begin{bmatrix} X \\ Y \\ z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.12}$$

Kde:

- M' je vektor souřadnic bodu v souřadném systému kamery
- $[R|t]$ jsou rotační matice a translační vektor popisující transformaci ze souřadného systému reálného světa do souřadného systému kamery
- w je vektor promítaného bodu v souřadném systému reálného světa

2.4 Vzájemná kalibrace kamery a robotického ramene

U vzájemné kalibrace kamery a robotického ramene jde o zjištění jejich vzájemných vztahů pro možnost převedení výsledků snímání kamery z jejího souřadného systému do souřadného

[4] Převzato z: https://docs.opencv.org/3.3.0/dc/d2c/tutorial_real_time_pose.html

systému robotického ramene. K tomu je zapotřebí nalezení transformační matice, která se skládá z matice popisující rotace a translačního vektoru. Tato transformační matice nám následně umožní navigovat robotické rameno na souřadnice získané z výsledků měření kamery. Tento typ transformace je často používán v počítačovém vidění a využívá upravené rovnice 2.11, která vypadá následovně:

$$w = R^{-1}[M' - t] \tag{2.13}$$

Kde:

- w je vektor promítaného bodu v souřadném systému reálného světa
- R^{-1} je inverzní rotační matice k matici R
- M' je vektor souřadnic bodu v souřadném systému kamery
- t je translační vektor popisující translaci mezi souřadným systémem reálného světa a souřadným systémem kamery

Kapitola 3

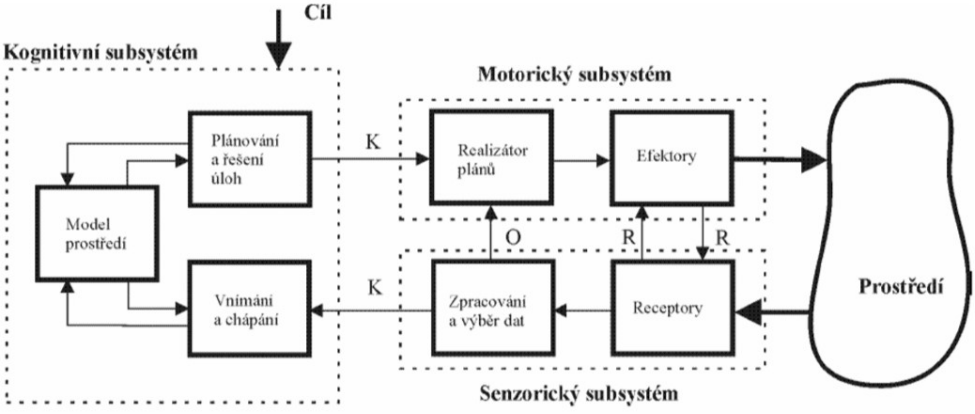
Robotický systém

V této kapitole se zaměříme na popis robotického systému a pracoviště, včetně všech jeho součástí a pojmů využívaných v rámci této diplomové práce. Zaměříme se také na definici a popis robotického ramene.

3.1 Obecný robotický systém

Blokové schéma na obrázku 3.1 znázorňuje obecný robotický systém [12]. Takový systém se skládá ze 3 hlavních bloků:

1. Senzorický subsystém
2. Kognitivní subsystém
3. Motorický subsystém



Obrázek 3.1: Schéma obecného robotického systému [12]

Senzorický subsystém slouží pro snímání signálů z prostředí. Je tvořen receptory, které mění fyzikální signály z okolního prostředí na vhodné vnitřní signály. Součástí tohoto systému je také základní zpracování a výběr přijatých signálů.

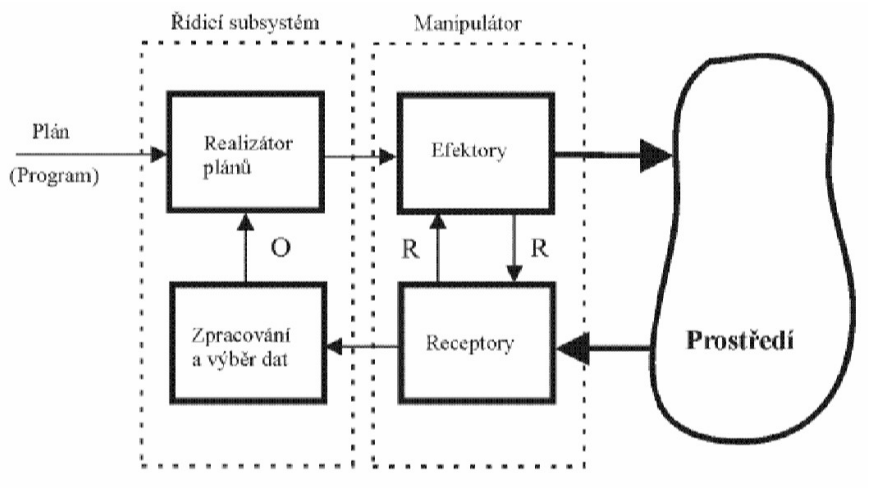
Kognitivní subsystém slouží jako vyšší nadřazené inteligentní řízení. Jeho cílem je především hlubší analýza informace získané ze senzorického subsystému. Součástí takové analýzy

je vnímání a chápání, což vyžaduje, aby měl robot k dispozici nějaký model prostředí. Na základě výsledku analýzy si stanoví své cíle práce, které jsou předány k plánování a později také k řešení.

Motorický systém slouží k interakci robota s okolním prostředím. Je tvořen efektory a realizátory plánů, které mají za úkol řízení těchto efektorů.

3.2 Průmyslový robot

Dle standartu `ISO 8373` [4] se jedná o automaticky řízený, reprogramovatelný, víceúčelový manipulátor pro použití v průmyslových automatizačních aplikacích, který je programovatelný ve třech nebo více pohybových osách. Může být stacionárně umístěný nebo mobilní. Na obrázku 3.2 je znázorněno obecné blokové schéma takového průmyslového robota. Jak můžeme vidět, toto schéma vychází ze schématu obecného robotického systému.



Obrázek 3.2: Schéma obecného průmyslového robotu [12]

3.3 Koncový efektor

Jde o zařízení nebo nástroj, který je připojen ke konci ramena robota a slouží k interakci s okolním prostředím. Jde o zařízení s nulovou úrovní inteligence, které má méně než tři pohyblivé osy, nebo více než tři osy, které ale nejsou programovatelné [4]. Typ koncového efektoru je určen typem činnosti robota, nejčastěji jde o určitý typ gripperu či přísavky pro manipulaci s objekty.

3.4 Pracovní prostor

Popisuje se jako prostor určený svým tvarem a objemem, který opíše referenční bod pracovní hlavice využitím všech možností svého pohybového systému [12]. Jedná se tedy o prostor, který je dán možnostmi pohybového rozsahu robotického manipulátoru ve všech směrech od své pevné základny.

3.5 Kolizní prostor

Je definován jako prostor určený svým tvarem a objemem, který vyplní konstrukce robotického manipulátoru v rámci své činnosti využitím všech možností svého pohybového systému [12]. Jedná se tedy o prostor, ve kterém se robotický manipulátor může pohybovat všemi svými částmi v rámci aktuálně prováděné činnosti.

3.6 Kinematika

Aktuální stav robota může být popsán pomocí kloubových souřadnic nebo pozicí koncového efektoru robota v prostoru. Pro robota se šesti klouby vypadá vektor q následovně:

$$q = [q_1, q_2, q_3, q_4, q_5, q_6]^T \tag{3.1}$$

Vektor P reprezentující pozici koncového efektoru je dán pomocí kartézských souřadnic a rotacemi kolem každé z os souřadného systému:

$$P = [x, y, z, \alpha, \beta, \gamma]^T \tag{3.2}$$

Vzájemnými vztahy mezi těmito dvěma vektory se zabývají následující dvě úlohy kinematiky.

Přímá úloha kinematiky

Přímá úloha kinematiky se zabývá jednoznačným zobrazením z prostoru kloubových souřadnic do prostoru kartézských souřadnic pro jednoznačné určení aktuální polohy koncového efektoru robota [12]. Toto zobrazení je dané vztahem:

$$P = f(g) \tag{3.3}$$

Toto zobrazení je pro každé uspořádání kloubů jednoznačné a existuje vždy právě jedno řešení této úlohy.

Inverzní úloha kinematiky

Inverzní úloha kinematiky řeší opačný problém, tedy nalezení aktuálních kloubových souřadnic na základě kartézských souřadnic polohy koncového efektoru [12]. Řešená úloha je tedy daná vztahem:

$$g = f(P) \tag{3.4}$$

Toto zobrazení už není zdaleka tak jednoznačné jako v předchozím případě. Pro některé případy polohy koncového efektoru může existovat potenciálně nekonečně mnoho řešení uspořádání kloubových souřadnic.

Kapitola 4

Robotický software

Tato kapitola se věnuje teoretickému přehledu o robotickém frameworku ROS a jeho možnostech v oblasti plánování pohybu robotického manipulátoru, společně s dalšími nástroji využitými v rámci této diplomové práce.

4.1 ROS

Robot Operating System (Robotový operační systém), neboli `ROS`, je abstraktní vrstva spojující operační systémem počítače s robotem [16]. Jedná se tedy o robotický middleware, označován také jako meta-operating system. Nejedná se tedy o plnohodnotný operační systém, v podstatě jde o open-source kolekci softwarových frameworků určenou pro vývoj software a správu balíků v oblasti robotiky. Poskytuje různé služby jako například abstrakci hardware a kontrolu zařízení na nejnižší úrovni [15]. Implementuje také komunikaci mezi jednotlivými procesy realizovanou pomocí zasílání zpráv.

`ROS` je oficiálně podporovaný pro unixové systémy, přičemž v období vzniku této práce je nejnovější stabilní distribucí `ROS Melodic Morenia` pro `Ubuntu 18.04`. Jeho verze jsou také testovány na systému `Mac OS X`. O podporu pro ostatní systémy, jako například `Fedora, Gentoo, Arch Linux` atd. se stará komunita. Microsoft aktuálně pracuje také na verzi pro operační systém `Windows`.

Pro tvorbu procesů jsou primárně podporovány programovací jazyky `C++` (`roscpp`), `Python` (`rospy`), a `Lisp` (`roslisp`). Ve vývoji jsou také stále ještě experimentální knihovny pro jazyky `C#, Java, Lua, Matlab`, nebo `JavaScript`.

ROS prozatím není real-time operační systém, i když je možné jej s takovým kódem integrovat. Na této podpoře se pracuje při vývoji verze ROS 2.0 vzhledem k čím dál vyšší nutnosti nízké latence a vysoké reaktivity v oblasti robotiky.

Procesy

Jednotlivé procesy v systému `ROS` jsou reprezentovány jako samostatné uzly (nodes), které spolu mohou navzájem komunikovat. Jedná se o *peer-to-peer* síť samostatných programů, které mohou běžet nezávisle jeden na druhém, a to také na více strojích [21]. Tato síť tvoří grafovou strukturu, tzv. *runtime graph*, mapující veškeré existující uzly v síti, včetně informací o jednotlivých sběrnicích (topics), ke kterým mohou být přihlášení jako *subscriber* (odběratel) nebo *publisher* (vydavatel). Veškerý přehled o těchto uzlech je uchováván ve speciálním uzlu *Master*. Tento uzel lze spustit příkazem `roscore`.

Zprávy

Jak již bylo zmíněno v předchozí kapitole, jednotlivé uzly spolu navzájem komunikují pomocí zasílání zpráv. Jedná se o jednoduché datové struktury s předem definovanými atributy [20]. Tyto atributy jsou tvořeny standartními datovými typy jako intiger, floating point, char, boolean atd. Podporována jsou také pole těchto primitivních typů. Je zde patrná podobnost se strukturami jazyka `C`.

Pro definici zprávy se používají soubory formátu `msg`. Jedná se o jednoduché textové soubory pro definici datové struktury tvořící zprávu. Programátorovi je tedy umožněno vytvořit si vlastní typy zpráv. Framework `ROS` obsahuje také předpřipravené nejčastěji používané typy zpráv. Každá zpráva může obsahovat záhlaví nesoucí dodatečná metada, například časové razítko a ID rámce.

Sběrnice

Aby mezi sebou mohly jednotlivé uzly komunikovat zasíláním zpráv, musejí se nejprve přihlásit pro publikování nebo odběr k určité sběrnici [23]. Pro uzly je odstíněno kdo je k dané sběrnici přihlášen, tyto informace vlastní jedině uzel `roscore`. Počet přihlášených uzlů k jednotlivým sběrnicím není nějak omezeno. Jeden proces může zároveň publikovat i odebírat ze stejné sběrnice.

Sběrnice jsou určeny především pro jednosměrnou a streamovanou komunikaci, neslouží pro vzdálená volání procedur jiných procesů. Pro každou sběrnici je předem definován typ zprávy, který bude využívat. Žádné jiné zprávy není možné přes takovou sběrnici přenést.

V současné době podporuje framework `ROS` přenos zpráv založený na protokolu `TCP/IP (TCPROS)` využívající trvalé připojení, a protokolu `UDP (UDPROS)` využívajícím nepotvrzované doručení a podporovaném pouze v roscpp. Spojení `TCPROS` je výchozí transportní mechanismus použitý v systému `ROS` a je nutné jej implementovat při vytváření vlastní klientské knihovny.

Služby

Jedná se o mechanismus eliminující nedostatky pouhého jednosměrného zasílání zpráv popsaného v předchozí kapitole. Používá se, pokud je nutná interakce na zaslaný požadavek [22]. Žádost a odpověď se provádí prostřednictvím služby, která je definována pomocí dvojice zpráv, jedna pro žádost (request) a druhá pro odpověď (response). Funguje tedy podobně jako vzdálené volání procedur. Služba je dostupná pomocí svého názvu tvořeného jedinečným řetězcem. Klient se přihlásí k dané službě, odešle na ni požadavek a čeká na odpověď. Služby jsou definovány pomocí souborů `srv`.

Ke službě je možné se přihlásit jednou v případě potřeby, nebo pomocí trvalého spojení. To má za následek vyšší výkon, ovšem v případě náhlé změny poskytovatele této služby dojde k chybě. Podobně jako u sběrnic měla každá sběrnice definovaný přesný typ zprávy se kterými může pracovat, má také služba přesně definovaný typ služby, kterou podporuje.

Akce

V předchozí kapitole jsme si popsali, co je to služba a kdy se používá. V některých případech, například pokud služba trvá delší dobu, může být vhodné mít nějakým způsobem možnost zrušit službu během provádění, nebo pravidelně získávat informace o jejím provádění. Za tímto účelem vznikly ve frameworku `ROS` akce (actions) [19], implementované

v balíku *actionlib*, který poskytuje nástroje pro vytváření serverů spravujících tyto dlouhodobé cíle, a který také poskytuje klientské rozhraní pro odesílání požadavků na tyto servery. Komunikace mezi klientem a serverem probíhá v tomto případě pomocí *ROS Action Protokol*. Klient a server disponují jednoduchým `API` pro odesílání požadavků klienta na server nebo pro realizaci těchto požadavků serverem zpětných volání.

Pro komunikaci klienta a serveru jsou používány tyto typy zpráv definující akci:

1. Cíl (goal) - popisuje představu o splnění cíle, například požadovaná cílová póza robota.
2. Zpětná vazba (feedback) - popisuje způsob, jak bude server informovat klienta o průběhu akce, například průběžné informování o aktuální pozici robota.
3. Výsledek (result) - narozdíl od zpětné vazby se klientovi odesílá pouze jednou, po ukončení akce.

URDF

Soubor `URDF`, neboli *Universal Robotic Description Format*, obsahuje kompletní specifikaci počítačového modelu robota, scény, nebo senzorů [18]. Jedná se o soubor ve formátu `XML`. Pro tento soubor je nutné mít validátor v některém z programovacích jazyků. Je tvořen množstvím různých balíčků a komponent. Hlavním omezením formátu `URDF` je fatk, že dokáže popsat pouze stromovou strukturu. Nelze tak například popisovat paralelní roboty. Zároveň se předpokládá, že robot sestává pouze z pevných propojených částí, což eliminuje veškeré flexibilní prvky. Specifikace zahrnuje:

- Kinematický a dynamický popis robota
- Vizuální popis robota
- Kolizní popis robota

4.2 ROS-Industrial

Jedná se o open-source projekt, který si klade za úkol rozšířit stávající funkcionalitu frameworku `ROS` pro použití v průmyslové robotice a automatizaci [17]. Poskytuje podporu a rozhraní pro většinu běžně používaných robotických manipulátorů, gripperů a senzorů. Mezi nejznámější podporované dodavatele patří například ABB, Universal Robots, Fanuc, Adept, nebo Motoman. Dále obsahuje softwarové knihovny pro automatickou kalibraci 2D a 3D senzorů, plánování pohybu, plánování cesty atd.

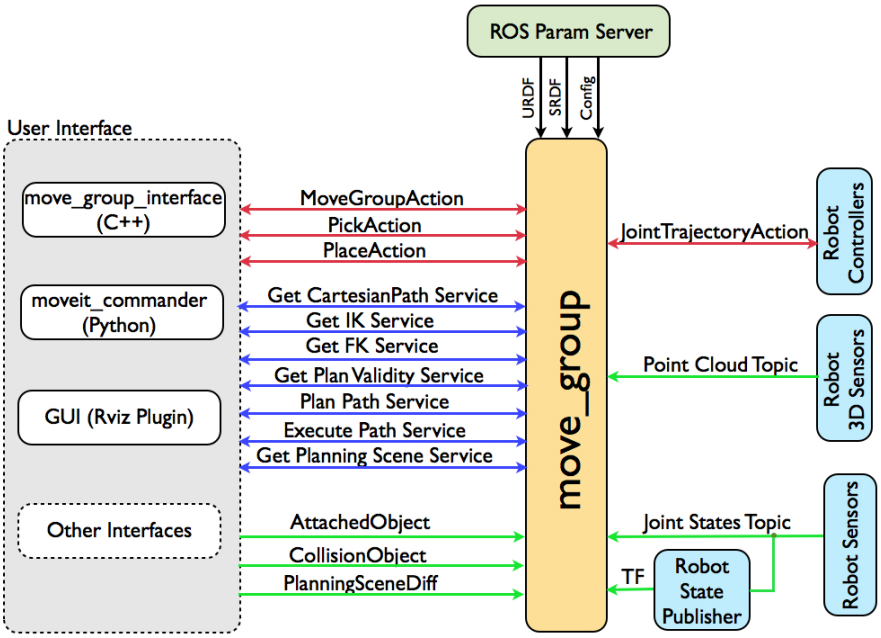
Obsahuje nástroje pro vývoj, simulaci a vizualizaci. Pro manipulátory používá vlastní řešení inversní kinematiky, a to také pro manipulátory s více než šesti stupni volnosti. Podporuje pokročilé zpracování 2D a 3D snímků. Při plánování pohybu poskytuje podporu pro eliminaci kolizí s kolizními objekty.

4.3 MoveIt!

Jde o open-source projekt prezentující se jako *state of the art* pro plánování pohybu a manipulace v oblasti robotiky [6]. Za úkol si klade poskytovat snadno použitelnou platformu pro vývoj robotických aplikací. Poskytuje `API` pro jazyky `C++` a `Python`. Dá se také ovládat skrze rozšíření GUI pro program Rviz.

Rozhraní robota

Obrázek 4.1 znázorňuje architekturu systému `MoveIt!` pro jeho primární uzel `\move_group`. Tento uzel slouží jako integrátor, který si načte jednotlivé komponenty robota a následně poskytuje příslušné akce skrze robotický systém [10]. K tomu využívá `API` `\move_group_interface` pro `rocpp`, nebo `moveit_commander` pro `rospy`. Veškerá komunikace mezi tímto uzlem a robotem je realizována pomocí sběrnic a akcí.



Obrázek 4.1: Architektura uzlu `\move_group` systému MoveIt! [1]

Jednou ze sběrnic které uzel `\move_group` naslouchá je sběrnice `/joint_states`, na kterou je v pravidelném intervalu publikován aktuální stav jednotlivých kloubů robota daný úhlem jejich natočení v radiánech.

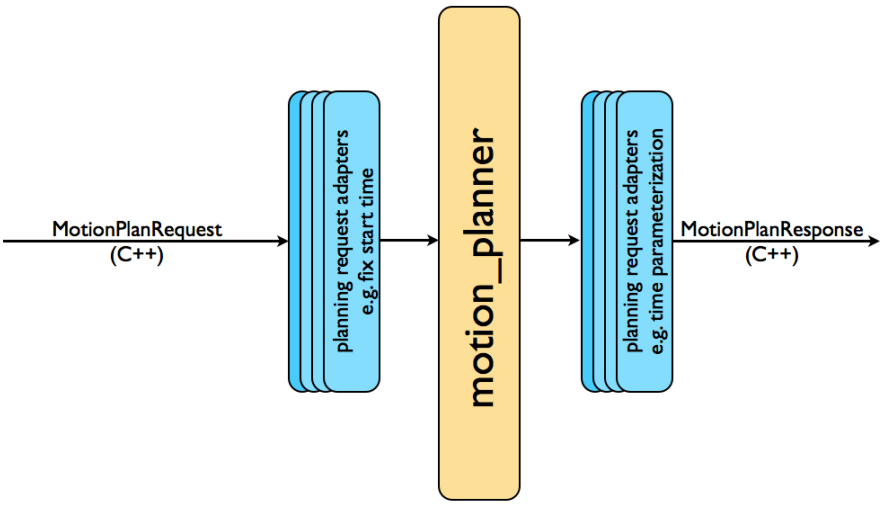
Dále také neustále monitoruje informace o transformacích, které získává z knihovny `TF` 4.4 systému `ROS`. Díky tomu je možné skrze `MoveIt!` vyčíst aktuální pózu robota. Dále může pomocí této knihovny interně řešit transformace mezi jednotlivými souřadnými systémy.

Tento uzel je navržen a implementován tak, aby byl nadále jednoduše rozšiřitelný. Ostatní věci jako řešení kinematických úloh, plánování *pick-and-place* instrukcí, nebo plánování pohybu jsou řešeny pomocí samostatných rozšíření. Tato skutečnost dovoluje větší flexibilitu pro programátora při jejich výběru, zároveň je každému umožněno vytvořit si vlastní rozšíření pro danou problematiku.

Plánování pohybu

Jak již bylo zmíněno v předchozí kapitole, MoveIt! pracuje s plánovači pohybu ve formě rozšíření [8]. To dovoluje použití různých typů plánovačů pohybu z více knihoven, se kterými se komunikuje pomocí akce nebo služby robotického systému skrz uzel `\move_group`. Jako výchozí jsou v `MoveIt!` použité plánovače z knihovny OMPL 4.4.

[1] Převzato z: https://moveit.ros.org/documentation/concepts/



Obrázek 4.2: Zpracování pohybu pomocí uzlu `motion_planner` [2]

Požadavek na plánování pohybu je dán požadovaným natočením jeho kloubů, případně jeho konečnou pózou, která je dána pozicí a rotací v prostoru jeho posledního článku, tzv. `end effector`. Zároveň je možné pro každý pohyb definovat různá omezení, jako rozsahy pohybu a rotace jednotlivých kloubů. Na základě tohoto požadavku je skrze uzel `\move_group` vrácena vygenerovaná trajektorie.

Detekce kolizí

Předcházení kolizí s kolizními objekty je uvnitř systému `MoveIt!` řešeno pomocí knihovny `FCL` (*Flexible Collision Library*). Tento proces je před uživatelem zcela skryt. [7]. Kontrola kolizí je podporována pro tyto objekty:

1. Meshes - polygonové sítě reprezentované pomocí vrcholů, hran a ploch.
2. Primitivní tvary - základní geometrické útvary jako krychle, válec, kužel a koule.
3. Octomap - objekt reprezentovaný 3D mřížkou oktantů.

4.4 Podpůrné nástroje

Rviz

Jedná se o 3D vizualizační nástroj, pomocí kterého lze prostorově zobrazovat data. Umožňuje zobrazovat například aktuální stav robota na jeho virtuálním modelu, ale také různá data ze sensorů nebo kamer, která lze odebírat z jednotlivých sběrnic systému `ROS`.

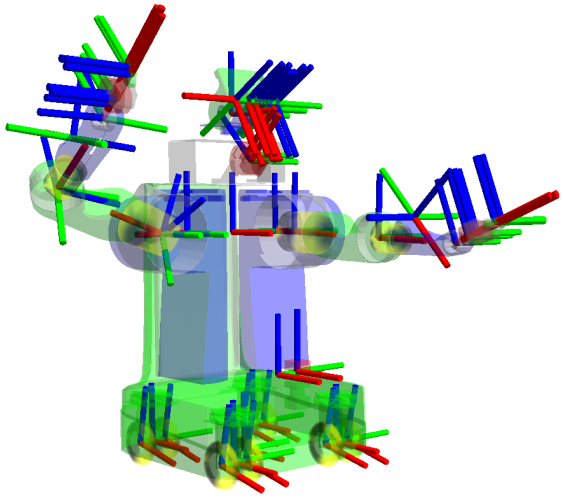
[2] Převzato z: https://moveit.ros.org/documentation/concepts/

OMPL

OMPL (*Open Motion Planning Library*) je open-source knihovna pro plánování pohybu. Tato knihovna implementuje řadu různých plánovačů pohybu, přičemž tyto plánovače realizují náhodné hledání cesty [25]. Tato skutečnost, spolu s tím že plánovače nemají žádné informace o reálném modelu robota, mohou zapříčinit zbytečně složité výsledné trajektorie i pro zdánlivě jednoduché pohyby. Jak již bylo zmíněno v předchozí kapitole, jedná se výchozí knihovnu pro plánovaní pohybu přímo integrovanou do systému `MoveIt!` [9]. Žádný z plánovačů neřeší problém týkající se prevence kolizí s kolizními objekty.

TF

Knihovna `TF` (*Transform Library*) slouží pro udržení přehledu o jednotlivých souřadných systémech v průběhu času a zároveň dovoluje provádět transformace mezi těmito systémy [26]. Tyto souřadné systémy se v robotickém systému označují jako rámce (`frames`). Na obrázku 4.3 je znázorněna vizualizace těchto souřadných systémů na modelu robota.



Obrázek 4.3: Vizualizace jednotlivých souřadných systémů na robotovi PR2 [3]

Informace o vztazích mezi jednotlivými souřadnými systémy jsou v rámci knihovny `TF` uchovávány ve stromové struktuře. Pokud uživatel zažádá o transformaci z jednoho souřadného systému do druhého, `TF` tuto transformaci provede pouze tehdy, pokud v rámci této stromové struktury existuje cesta mezi těmito uzly, přičemž tyto uzly nemusí být v přímém vztahu otec a syn. Nejčastěji je knihovna `TF` využívána pro navigaci koncového efektoru robota pomocí jeho pózy, dané souřadnicemi a rotacemi v prostoru, z pohledu tzv. `word frame`, neboli ze souřadného systému světa. Jednotlivé rámce je také možné vytvářet pomocí zvolení jejich jména, počátku a rotace souřadného systému vzhledem ke světu (`word frame`).

PCL

Knihovna `PCL` (*Point Cloud Library*) slouží pro zpracování 2D/3D obrazu a práci s mračny bodů, neboli `point cloud` [13]. `PCL` nativně podporuje rozhraní OpenNI 3D a umožňuje tak

[3] Převzato z: http://wiki.ros.org/tf

získávat a zpracovávat data ze zařízení jako jsou například 3D kamery PrimeSensor, Microsoft Kinect nebo Asus XTionPRO. Obsahuje algoritmy pro porovnávání modelů, filtraci, segmentaci, rekonstrukci povrchů či extrakci příznaků.

Point cloud

Jedná se o datovou strukturu tvořenou množinou bodů v prostoru popsanými svými kartézskými souřadnicemi. Tento formát je často používaný pro výstup snímání stereo kamer a 3D skenerů. Může sloužit k vizualizaci, měření kontroly kvality, vytváření 3D CAD modelů atd. V případě, kdy je kamera schopna zaznamenávat také barvu, dostáváme data ve formátu 4D, kde čtvrtou složkou je hodnota RGB.

Obrázek 4.4: Ukázka dat reprezentovaných mračnem bodů

4.5 ARCOR

Jedná se o systém představující uživatelské rozhraní pro interakci mezi robotem a člověkem založený na rozšířené realitě [5]. Hlavními částmi systému `ARCOR` jsou projektor společně s dotykovou plochou, které společně tvoří prostor robotického pracoviště. Jeho hlavním cílem je poskytnout bezpečné, jednoduché a intuitivní ovládání robota i běžnému uživateli, který nemá s programováním robotů zkušenosti.

Architektura systému

Systém `ARCOR` je vytvořený pro robotický systém `ROS` a jeho architektura je tedy odvozena od hlavních myšlenek tohoto systému. Celý systém je tvořen z několika uzlů, které obstarávají různé funkce. Zde je ukázka několika vybraných uzlů systému:

- *art_brain* - hlavní uzel zajišťující komunikaci s robotem, řídí program, uchovává aktuální stav systému.
- *art_calibration* - uzel zajišťující kalibraci kamer pomocí AR markerů.

- *art_collision_env* - spravuje detekované a vložené kolizní objekty ve scéně.
- *art_db* - uzel sloužící jako permanentní úložiště objektů, programů atd.
- *art_table_pointing* - pomocí Kinectu vypočítá souřadnice, kde se uživatel dotkl stolu.
- *art_touch_driver* - čte data z dotykové podložky a publikuje je jako zprávy do systému ROS.

Součástí systému jsou také další dodatečné repozitáře:

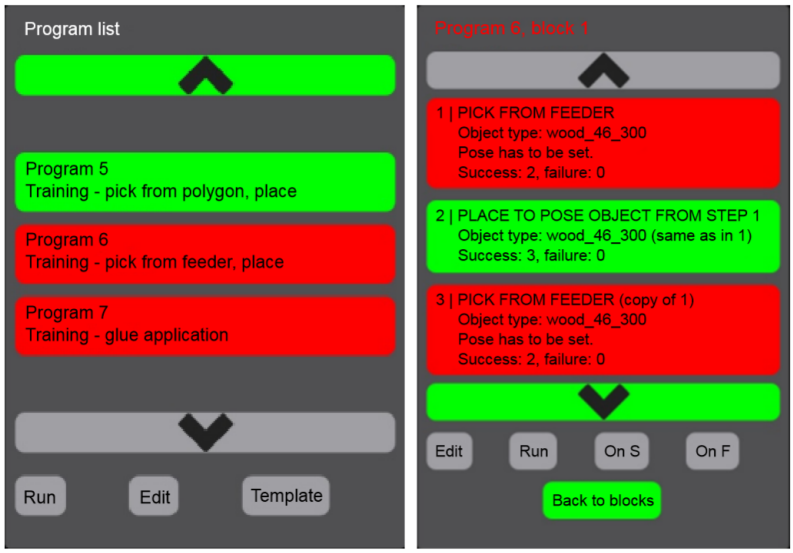
- *arcor-msgs* - obsahuje definice zpráv, akcí a služeb pro systém `ROS`.
- *arcor-utils* - obsahuje pomocné třídy pro programovací jazyk `Python`.
- *arcor-detectors* - obsahuje zdrojové soubory pro obsluhu detektorů.

Design systému

Systém `ARCOR` pracuje ve 2 hlavních módech:

1. Nastavení parametrů
2. Exekuce programu

Programování robota probíhá intuitivně skrze jednotlivé bloky instrukcí. Každý program je složen z 1..n bloků obsahujících 1..n instrukcí. Větvení a cyklení programu je založeno na vyhodnocování úspěchu či neúspěchu každého bloku programu. Ukázka této programové struktury je znázorněna na obrázku 4.5.



Obrázek 4.5: Programová struktura systému `ARCOR` [5]

Kapitola 5

Analýza existujících metod

V následující kapitole budou postupně popsána jednotlivá existující řešení věnující se vzájemné kalibraci robotického ramene a kamery spolu s existujícími metodami zabývajícími se problematikou lokalizace objektů ve scéně.

5.1 Automatický výpočet transformační matice

Jak je zřejmé z názvu, tato metoda, publikovaná v práci s názvem *Automatic Calculation of a Transformation Matrix Between Two Frames* [3], slouží pro výpočet transformace mezi dvěma souřadnými systémy. Tuto transformaci vypočítá ze dvou odpovídajících si množin souřadnic bodů v jednotlivých souřadných systémech. Výpočet probíhá pomocí lineární regrese a složitost této metody roste lineárně s množstvím zaznamenaných bodů. V rámci zmíněné publikované práci bylo zjištěno, že tato metoda produkuje deterministické výsledky. Zároveň bylo ověřeno, že při výpočtu transformační matice dochází k velice malé chybnosti.

V prvním kroku této metody zapíšeme matici transformace jako soustavu lineárních rovnic:

$$T_B^A = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{zy} & r_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

$$\begin{bmatrix} x_B \\ y_B \\ z_B \\ 1 \end{bmatrix} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{zy} & r_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ z_A \\ 1 \end{bmatrix} \tag{5.2}$$

$$\begin{aligned} x_B &= r_{xx}x_A + r_{xy}y_A + r_{xz}z_A + t_x \\ y_B &= r_{yx}x_A + r_{yy}y_A + r_{yz}z_A + t_y \\ z_B &= r_{zx}x_A + r_{zy}y_A + r_{zz}z_A + t_z \end{aligned} \tag{5.3}$$

Cílem lineární regrese je minimalizace čtverce reziduí. Pro lineární regresi je čtverec rezidua definován následující rovnicí:

$$R^2(m,b) = \sum_{i=1}^{n} [y_i - (mx_i + b)]^2 \tag{5.4}$$

Z definice 5.4 vyplývá následující soustava rovnic, kde n je počet párovaných vzorků souřadnic:

$$\begin{aligned} R^2_{xB} &= \sum_{i=1}^{n} [x_{Bi} - (r_{xx}x_{Ai} + r_{xy}y_A i + r_{xz}z_{Ai} + t_x)]^2 \\ R^2_{yB} &= \sum_{i=1}^{n} [y_{Bi} - (r_{yx}x_{Ai} + r_{yy}y_A i + r_{yz}z_{Ai} + t_y)]^2 \\ R^2_{zB} &= \sum_{i=1}^{n} [z_{Bi} - (r_{zx}x_{Ai} + r_{zy}y_A i + r_{zz}z_{Ai} + t_z)]^2 \end{aligned} \tag{5.5}$$

Minimální hodnotu získáme tak, že položíme deriváty čtverců rezidua z rovnic 5.5 rovny nule:

$$\begin{aligned} \frac{\partial R^2_{xB}}{\partial r_{xx}} &= 0 \\ \frac{\partial R^2_{xB}}{\partial r_{xx}} &= -2\sum_{i=1}^{n} [x_{Bi} - (r_{xx}x_{Ai} + r_{xy}y_{Ai} + r_{xz}z_{Ai} + t_x)]x_{Ai} \end{aligned} \tag{5.6}$$

$$\begin{aligned} \frac{\partial R^2_{xB}}{\partial r_{xy}} &= 0 \\ \frac{\partial R^2_{xB}}{\partial r_{xy}} &= -2\sum_{i=1}^{n} [x_{Bi} - (r_{xx}x_{Ai} + r_{xy}y_{Ai} + r_{xz}z_{Ai} + t_x)]y_{Ai} \end{aligned} \tag{5.7}$$

$$\begin{aligned} \frac{\partial R^2_{xB}}{\partial r_{xz}} &= 0 \\ \frac{\partial R^2_{xB}}{\partial r_{xz}} &= -2\sum_{i=1}^{n} [x_{Bi} - (r_{xx}x_{Ai} + r_{xy}y_{Ai} + r_{xz}z_{Ai} + t_x)]z_{Ai} \end{aligned} \tag{5.8}$$

$$\begin{aligned} \frac{\partial R^2_{xB}}{\partial t_x} &= 0 \\ \frac{\partial R^2_{xB}}{\partial t_x} &= -2\sum_{i=1}^{n} [x_{Bi} - (r_{xx}x_{Ai} + r_{xy}y_{Ai} + r_{xz}z_{Ai} + t_x)] \end{aligned} \tag{5.9}$$

Výsledky rovnic 5.6 - 5.9 zapíšeme do maticové podoby. Výsledné matice 5.11 - 5.13 následně vyjadřující jednotlivé řádky hledané transformační matice 5.1:

$$A = \begin{bmatrix} \sum_{i=1}^{n} x_{Ai}^2 & \sum_{i=1}^{n} x_{Ai}y_{Ai} & \sum_{i=1}^{n} x_{Ai}z_{Ai} & \sum_{i=1}^{n} x_{Ai} \\ \sum_{i=1}^{n} x_{Ai}y_{Ai} & \sum_{i=1}^{n} y_{Ai}^2 & \sum_{i=1}^{n} x_{Ai}z_{Ai} & \sum_{i=1}^{n} y_{Ai} \\ \sum_{i=1}^{n} x_{Ai}z_{Ai} & \sum_{i=1}^{n} y_{Ai}z_{Ai} & \sum_{i=1}^{n} z_{Ai}^2 & \sum_{i=1}^{n} z_{Ai} \\ \sum_{i=1}^{n} x_{Ai} & \sum_{i=1}^{n} y_{Ai} & \sum_{i=1}^{n} z_{Ai} & n \end{bmatrix} \tag{5.10}$$

$$\begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \\ t_x \end{bmatrix} = [A]^{-1} \begin{bmatrix} \sum_{i=1}^{n} x_{Bi}x_{Ai} \\ \sum_{i=1}^{n} x_{Bi}y_{Ai} \\ \sum_{i=1}^{n} x_{Bi}z_{Ai} \\ \sum_{i=1}^{n} x_{Bi} \end{bmatrix} \tag{5.11}$$

$$\begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \\ t_y \end{bmatrix} = [A]^{-1} \begin{bmatrix} \sum_{i=1}^{n} y_{Bi}x_{Ai} \\ \sum_{i=1}^{n} y_{Bi}y_{Ai} \\ \sum_{i=1}^{n} y_{Bi}z_{Ai} \\ \sum_{i=1}^{n} y_{Bi} \end{bmatrix} \tag{5.12}$$

$$\begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \\ t_z \end{bmatrix} = [A]^{-1} \begin{bmatrix} \sum_{i=1}^{n} z_{Bi}x_{Ai} \\ \sum_{i=1}^{n} z_{Bi}y_{Ai} \\ \sum_{i=1}^{n} z_{Bi}z_{Ai} \\ \sum_{i=1}^{n} z_{Bi} \end{bmatrix} \tag{5.13}$$

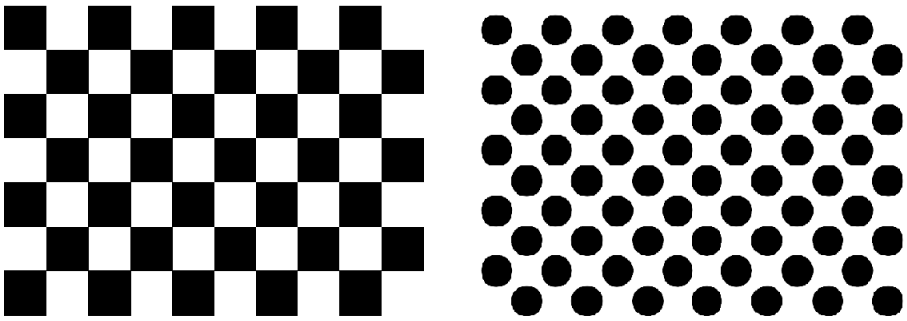
Jedním z omezení této metody je fakt, že pro výpočet transformace v trojrozměrném prostoru je nutné mít v obou množinách alespoň N bodů, přičemž platí:

$$N \geqslant 4 \tag{5.14}$$

Dalším omezením této metody je skutečnost, že matice `A`, popsána v rovnici 5.10, musí být invertibilní. To znamená, že zaznamenané body nesmějí v rámci obou systémů ležet ve stejné rovině. V tomto případě by byla porušena invertibilita matice `A` což by mělo za následek fakt, že matice 5.11 - 5.13 není možné vyřešit.

5.2 OpenCV Camera calibration

Pro kalibraci vnitřních parametrů kamery, popisujících její chování, je možné využít implementaci knihovny `OpenCV` [11]. Tato metoda používá pro kalibraci objekt zvaný kalibr. Jedná se o objekt s předem známými rozměry a strukturou. Nejčastěji má podobu černobílé šachovnice, jak je znázorněno na obrázku 5.1. Knihovna `OpenCV` nabízí několik předpřipravených snímků takových kalibrů.



Obrázek 5.1: Kalibr ve formě čtvercové a kruhové šachovnice [1]

[1] Převzato z: https://docs.opencv.org/3.1.0/d4/d94/tutorial_camera_calibration.html

Algoritmus v knihovně `OpenCV` postupně prochází jednotlivé snímky kalibru a hledá v něm významné body. V případě šachovnicového vzoru konkrétně rohy čtverců. Pokud použijeme vzor s černými kruhy, hledají se kruhy samotné. V obou případech je vstupním parametrem rozměr kalibru, výstupem jsou pozice jednotlivých vzorů. Algoritmus vyžaduje pořízení více snímků kalibru s různým natočením, protože podobné snímky vracejí podobné výsledky. U čtvercových obrazů jsou pozice rohů pouze přibližné, což ovšem lze vylepšit pomocí knihovních metod. Výsledkem kalibrace jsou 4 vnitřní parametry kamery, průměrná odchylka mezi skutečnými a vypočítanými polohami bodů a 5 koeficientů zkreslení, které byly popsány v kapitole 2.2.

Pro celkovou kalibraci musíme získat 4 vnitřní a 6 vnějších parametrů kamery [2]. Celkově tedy 10 parametrů. Pokud tedy máme K snímků šachovnice o N rozích. Pro každý snímek máme $6K$ vnějích parametrů a celkově poskytují $2NK$ omezení, musí platit následující rovnice:

$$2NK \geqslant 6K + 4 \tag{5.15}$$

$$(N - 3)K \geqslant 2 \tag{5.16}$$

Jelikož v případě čtvercové šachovnice máme pouze čtyři okolí, tedy N=4, musí platit následující:

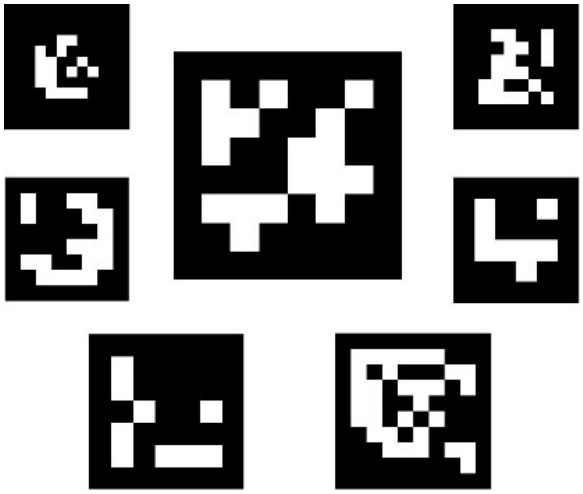
$$K \geqslant 2 \tag{5.17}$$

Potřebujeme tedy minimálně dva snímky pro celkovou kalibraci kamery. Ve skutečnosti pracujeme s více snímky pro větší přesnost a redukci šumu. Z tohoto důvodu je doporučeno pořídit alespoň deset snímků kalibračního obrazce v různých nakloněních. Z doporučení tedy vyplývá:

$$K \geqslant 10 \tag{5.18}$$

5.3 ArUco

Jedná se o jednu z nejpopulárnějších knihoven založenou na detekci čtvercových binárních markerů. Tyto markery je možné připevnit na hledaný objekt a ulehčit tak jeho detekci ve scéně pomocí kamery [1]. `ArUco` marker se skládá ze širokého černého okraje a vnitřní binární matice určující jeho identifikátor. Díky širokému černému orámování je snazší tento marker detekovat ve snímaném obraze a jeho binární struktura daná bílými a černými čtverci je odolná vůči chybám a nepřesnostem.



Obrázek 5.2: Ukázka ArUco markerů [2]

Na obrázku 5.2 jsou znázorněné ukázky `ArUco` markerů, které mohou být realizovány v různých velikostech. Tyto hodnoty udávají jak velká je matice kostek uvnitř markeru. Typicky se používají například rozměry 4x4, 5x5 nebo 6x6, přičemž každý jednotlivý čtverec reprezentuje jeden bit. Tento marker může být ve scéně libovolně natočený. Je tedy zapotřebí přesně identifikovat jednotlivé rohy markeru a zjistit jeho původní natočení, k čemuž se využívá přesného binárního popisu markeru.

Slovník

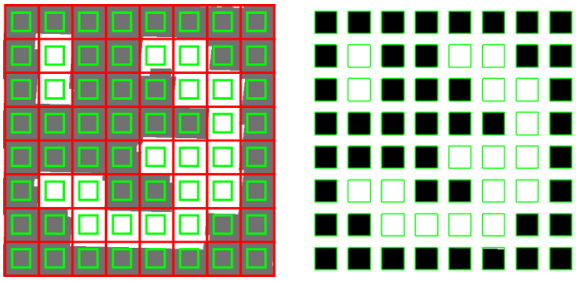
Aby bylo možné jednoznačně identifikovat takový marker ve snímané scéně, je zapotřebí udržovat slovník těchto značek. Tento slovník obsahuje binární reprezentaci všech markerů potenciálně využívaných v dané aplikaci. Jednoznačně určuje množinu markerů se kterou je aplikace schopná pracovat. `ArUco` modul obsahuje některé předdefinované slovníky obsahující základní markery různých velikostí. Identifikátor markeru zmíněný výše není tvořen konverzí binární posloupnosti na desetinné číslo, ale je určen pořadím tohoto markeru ve slovníku aplikace. Obecně je doporučené používat co možná nejmenší slovník, kvůli lepší rozlišovací schopnosti jednotlivých markerů a vyšší odolnosti vůči chybám.

Detekce markerů

Proces detekce `ArUco` markerů ve scéně navrací počet úspěšně detekovaných markerů daný umístěním každého z jeho čtyř rohů v původním pořadí společně s jednoznačným identifikátorem každého markeru. Tento proces se skládá ze dvou hlavních kroků:

1. Detekce kandidátů na markery
2. Analýza kandidátů na markery

[2] Převzato z: https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html



Obrázek 5.3: Extrakce bitové informace z `ArUco` markeru [3]

V prvním kroku je provedena analýza obrazu a jsou detekovány čtvercové útvary, jako potenciální kandidáti na markery. Tento proces je realizován pomocí adaptivního prahování pro segmentaci značek, jejichž obrysy jsou následně extrahovány. Pokud se tyto obrysy nepodobají čtvercovému útvaru, případně obsahují konvexní úhly, budou vyřazeny. Dále jsou filtrovány například příliš velké nebo příliš malé obrysy.

V druhém kroku dojde k analýze získaných kandidátů a k ověření, zda se skutečně jedná o `ArUco` markery. V rámci tohoto procesu jsou z každého kandidáta extrahovány bity dané vnitřní čtvercovou maticí. Nejprve je na marker uplatněna perspektivní transformace za účelem získání kanonické podoby markeru. Následně je tento kanonický obraz vymezen pomocí metody `Otsu`, kdy dojde k oddělení bílých a černých bitů. Obraz je následně rozdělen na buňky. Tento krok je znázorněný na obrázku 5.3. Množství černých nebo bílých pixelů v každé buňce slouží k určení, zda se jedná o bílý nebo černý bit. Nakonec jsou jednotlivé bity analyzovány, čímž se zjistí, zda značka patří do konkrétního slovníku použitého v aplikaci.

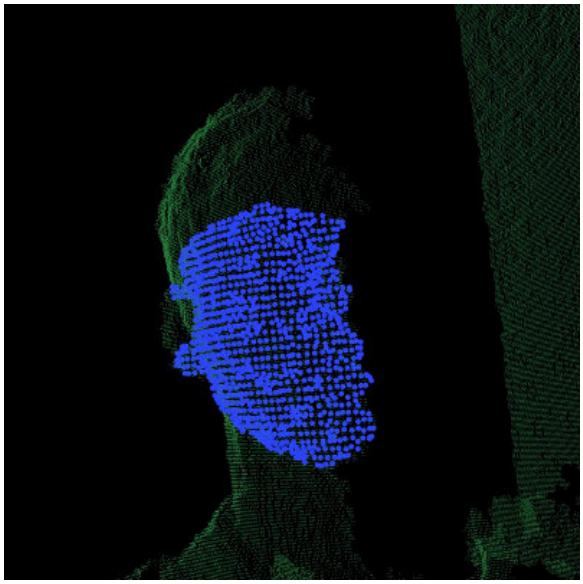
Odhad pózy markerů

Po úspěšné detekci jednotlivých markerů přichází na řadu získání jejich pózy v souřadném systému kamery. Pro tento krok je nutné znát vnitřní parametry kamery, dané maticí kamery a koeficienty zkreslení. Tyto vlastnosti lze zjistit kalibrací kamery. Modul `ArUco` poskytuje funkci pro odhad póz jednotlivých markerů. Vstupními parametry této funkce je vektor rohů získaný detekcí jednotlivých markantů, dále délka hrany markantu, matice kamery a vektor koeficientů zkreslení, získané její kalibrací.

5.4 Template matching

Nebo také *Template alignment*, je metoda, při které je snaha o zarovnání dřívějšího záznamu na aktuální snímek scény. Algoritmus začíná výpočtem lokálních deskriptorů a povrchových normál obou vzorků dat, na jejichž základě bude následně provedeno zarovnání [14]. Narozdíl od algoritmu `ICP` není tento algoritmus iterativní. Jeho výsledkem jsou rotační matice a translační vektor, které společně popisují vzájemný vztah mezi modelem a šablonou (template), spolu s hodnotou *fitness* značící úspěch zarovnání.

[3] Převzato z: https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html



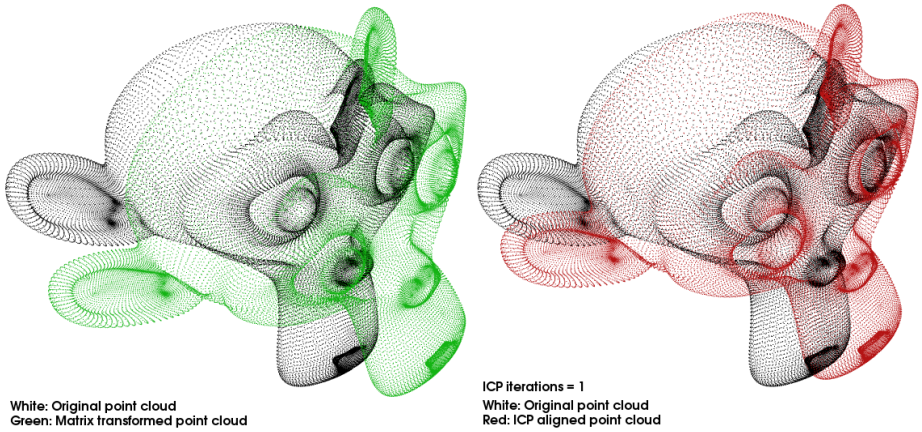
Obrázek 5.4: PCL template matching [4]

5.5 ICP

Algoritmus `ICP` (*Iterative Closest Point*) slouží pro zarovnání dvou mraků bodů [24]. Zarovnání probíhá iterativně v několika krocích, přičemž je usilováno o co nejmenší rozdíly mezi kartézskými souřadnicemi jednotlivých bodů v prostoru. Na obrázku 5.5 je ukázka provedení jedné iterace tohoto algoritmu. Jeden ze dvou vstupních objektů je zvolen jako fixní, zatímco na druhý jsou aplikovány různé transformace tak, aby se co nejvíce podobal fixnímu objektu. Jako fixní je nejčastěji zvolen snímek reálné scény pořízený 3D kamerou. Standartně se nejprve použije výše zmíněný `Template matching` jako odhad pro výchozí pozici objektu, na který se následně aplikuje algoritmus `ICP` pro co možná nejpřesnější zarovnání objektu. Jednotlivé kroky algoritmu jsou následující:

1. Pro každý bod z mraku bodů proběhne porovnání s jeho nejbližším referenčním bodem z fixního modelu.
2. Odhad kombinace rotace a posunutí pro jednotlivé body.
3. Proběhne transformace bodů porovnávaného modelu pomocí získané transformace.
4. Iterace algoritmu.

[4] Převzato z: http://pointclouds.org/documentation/tutorials/template_alignment.php



Obrázek 5.5: Iterative Closest Point [5]

Výstupem tohoto algoritmu je transformační matice popisující rotace a posunutí porovnávaného modelu vůči fixnímu. Existuje mnoho implementací tohoto algoritmu, přičemž nejpopulárnější jsou implementace *point-to-point* a *point-to-plane*. Důležitým faktorem rozhodujícím o úspěchu zarovnání pomocí algoritmu `ICP` je vhodně zvolený výchozí bod. z tohoto důvodu se často kombinuje s metodou `template matching`, popsané v kapitole 5.4, kdy se transformace získaná touto metodou použije právě jako zmíněný výchozí bod a algoritmus `ICP` se použije pro konečné zarovnání.

[5] Převzato z: http://pointclouds.org/documentation/tutorials/interactive_icp.php

Kapitola 6

Návrh aplikace a metod pro kalibraci

Cílem této kapitoly je popsat návrh jednotlivých metod pro vzájemnou kalibraci robotického ramene a 2D nebo 3D kamery pomocí detekce kalibru ve scéně robotického pracoviště. Nejdříve budou analyzovány požadavky kladené na výslednou aplikaci. Následně budou na základě těchto požadavků navrhnuté jednotlivé metody.

6.1 Definice problému

Hlavním úkolem aplikace je automatizace procesu kalibrace mezi robotickým ramenem, 2D nebo 3D kamerou, a existující scénou robotického pracoviště. Na základě této kalibrace dojde ke sjednocení souřadných systémů jednotlivých prvků scény a bude následně umožněna navigace robotického ramene uvnitř této scény. Aplikace se bude zabývat případem, kdy máme již existující robotickou scénu a v rámci ní známé souřadnice jednotlivých kalibračních objektů. Naším úkolem je zkalibrovat robota vůči této existující scéně. Robotický manipulátor má totiž typicky definovaný svůj vlastní souřadný systém a nelze jej tedy s jeho využitím bezpečně navigovat ve scéně, která je pro něj neznámá. Klíčovým prvkem zde bude právě onen objekt, tzv. *kalibr*, se známými souřadnicemi v souřadném systému scény pracoviště. Tento objekt tvoří prostředníka pro vzájemnou kalibraci jednotlivých komponent scény robotického pracoviště.

Výslednou aplikaci by mělo být možné použít nezávisle na zvoleném typu robotického ramene. Tento fakt je nutné zohlednit již při samotném návrhu jednotlivých metod aplikace a samotného kalibru. Metody nesmějí být závislé na konkrétních schopnostech určitého robota. Budeme tedy předpokládat pouze základní dostupné periferie jako 2D nebo 3D kameru a základní pohybové schopnosti robotického ramene s alespoň šesti stupni volnosti.

6.2 Návrh aplikace

Aplikace bude vytvořena pro robotický systém `ROS`. Bude se jednat o jednoduchou konzolovou aplikaci v rámci které bude možné zvolit z celkem tří různých metod využívajících odlišných principů pro vzájemnou kalibraci jednotlivých prvků scény robotického pracoviště. Kromě metod pracující s 2D nebo 3D kamerou vyžadující vzájemnou kalibraci kamery vůči scéně robotického pracoviště bude aplikace obsahovat také metodu, která pro kalibraci robota se scénou pomocí zaměření kalibračního objektu bude využívat pouze tělo samotného

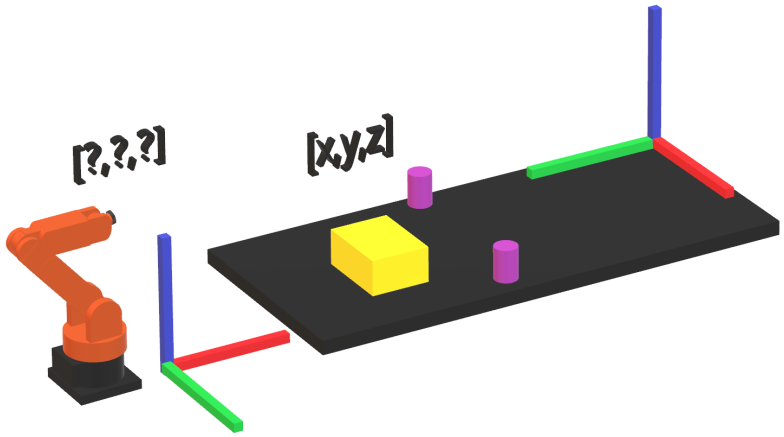
robota. Všechny tyto metody budou využívat některé z principů popsaných v kapitole 5. Výsledkem každé jednotlivé metody by měla být transformace mezi kalibrovaným objektem a existující scénou.

Aplikace by měla být plně funkční pro libovolné robotické rameno podporované robotickým systémem `ROS`. Z tohoto důvodu bude aplikace vystavena jako samostatný modul a nebude žádným způsobem využívat plánování pohybu konkrétního robota. Všechny metody popsané v následujících kapitolách budou tedy z pohledu pohybu robota navrženy jako semiautonomní. Bude se tedy využívat tzv. přímého programování, kdy je robotické rameno naváděné obsluhou. Plánování pohybu konkrétního robota bude využito v rámci testování jednotlivých metod.

Jelikož nelze zajistit jednotlivé ovládání pro různé typy 2D a 3D kamer, budou v rámci aplikace definované rozhraní pro tento typ periferií. Tato rozhraní budou obsahovat jednotlivé funkce, jejichž implementace bude nutná pro celkovou funkčnost aplikace.

6.3 Mechanická metoda

Na obrázku 6.1 je ilustrován modelový případ této kalibrační metody. Na levé straně je znázorněno rameno robotického manipulátoru s vlastním souřadným systémem. Naproti němu se nachází plocha představující scénu robotického pracoviště s vlastním souřadným systémem. Na této ploše se nachází jistý typ kalibračního objektu, pro něhož je známá transformace v souřadném systému scény. Našim cílem je skrze tento objekt nalézt transformaci mezi souřadným systémem robotického ramene a souřadným systémem scény a tím provést kalibraci robotického ramene. Protože robot v rámci této metody využívá pro vzájemnou kalibraci robota se scénou pomocí lokalizace známého objektu ve scéně pouze své vlastní tělo, není třeba provádět žádnou kalibraci vůči kameře. Princip této metody spočívá v tom, že si robot „osahá“ hledaný předmět pomocí svého koncového efektoru, čímž si jej zaměří vůči své poloze. Tato metoda je z principu použitelná na jakémkoliv typu robota a zároveň je také nejdostupnější, jelikož nepotřebuje žádné dodatečné periferie. Bude implementována jako semiautonomní metoda, což znamená, že je nutné počítat s navedením robota k hledanému předmětu obsluhou. Její hlavní limitací je především přesnost výpočtu úlohy přímé kinematiky pro získání polohy koncového efektoru, společně s faktem, že hledaný předmět ve scéně musí být vhodně umístěn v pracovním prostoru robota.



Obrázek 6.1: Ilustrace mechanické kalibrace robota

S využitím kalibračního objektu

V rámci této metody se předpokládá existence jistého typu kalibračního objektu uvnitř existující scény robotického pracoviště. Tento kalibrační objekt bude obsahovat sadu předem definovaných bodů, jejichž souřadnice jsou v této scéně známé. Pomocí koncového efektoru robotického ramene a uzlu `\move_group` systému `MoveIt!` následně zaměříme tyto body, čímž dostaneme jejich souřadnice v souřadném systému robotického ramene. K zaměření může dojít buď pomocí specializovaného gripperu, nebo pouze pomocí posledního článku řetězce robotického ramene. Nyní když známe jednotlivé souřadnice těchto bodů v obou souřadných systémech, můžeme vhodně zvolenou metodou vypočítat transformační matici mezi těmito dvěma systémy. Tato transformace bude sloužit ke kalibraci mezi robotickým ramenem a scénou robotického pracoviště.

S využitím 2D markerů

Narozdíl od předchozího případu budeme v této metodě uvažovat pouze dvojrozměrné markery umístěné na pracovní ploše ve scéně robotického pracoviště. Stejně jako v předchozí metodě budou souřadnice těchto dvojrozměrných markerů uvnitř této scény známé a postupně dojde k jejich zaměření robotickým ramenem. Tato metoda je narozdíl od metody využívající kalibrační objekt jednodušší na obsluhu. Další její výhodou je, že tyto dvojrozměrné značky mohou být součástí pracovní plochy po celou dobu obsluhy pracoviště robotickým ramenem, jelikož mu nebudou žádným způsobem omezovat pracovní a kolizní prostor.

Zaměření kolizních objektů

Typickou funkcí robotického manipulátoru nejčastěji bývá úloha tzv. *Pick and place.* v rámci níž robot z jednoho místa odebírá součástky a pokládá je na jiné místo, nejčastěji do nějakého boxu nebo krabice. V případě, že tento robotický systém není vybaven inteligentním kamerovým systémem, který by mu dovoloval reagovat na změny v okolním prostředí, bývají tato místa většinou přesně specifikovaná, stejně jako případné kolizní objekty. Pokud by došlo k situaci, že se toto místo pro odkládání změní, je nutné provést úpravu programu. Stejně tak pokud ve scéně vznikne překážka. V rámci této metody můžeme snadno zaměřit kolizní objekty ve scéně pomocí koncového efektoru robotického manipulátoru. Ze souřadnic v prostoru bude následně zjištěn pivot objektu, jeho rozměry a orientace. Na základě těchto údajů bude vytvořen kolizní objekt ve tvaru kvádru, který bude představovat kolizní obálku pro reálný předmět.

Zaměření kolizní plochy

Tato metoda, narozdíl od předchozí, slouží k zaměření typicky geometricky složitější plochy reprezentující například pracovní desku, na které se robotické rameno nachází. Tato metoda nám dovolí rychle a jednoduše zamezit kolizi robotického ramene s pracovní plochou, aniž bychom ji museli definovat jako model do systému `ROS`. Robot je postupně naveden na N bodů, přičemž musí platit:

$$N \geqslant 3 \tag{6.1}$$

Tyto body budou následně spojeny do plochy v 3D prostoru souřadného systému robotického manipulátoru, která bude definovat kolizní objekt pro pracovní plochu ve scéně robotického pracoviště.

Zaměření pracovního prostoru

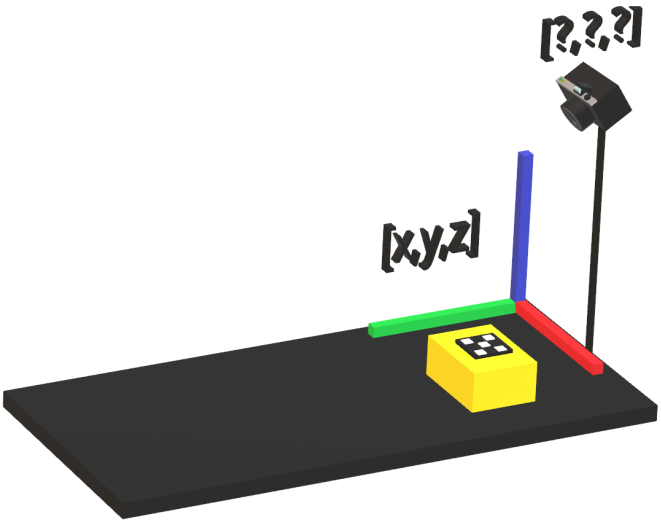
Jelikož vycházíme z předpokládané situace, kdy je robot jistým způsobem dodán do již existující scény pracoviště s vlastním souřadným systémem, bylo by vhodné mít metodu pro kalibraci pracovního prostoru v rámci této scény. To nám dovolí vymezit robotovi oblast, ve které se může bezpečně pohybovat. Protože je robot při pohybu ve scéně limitován především svojí geometrií, můžeme pro definici tohoto prostoru využít právě těla samotného robota. V rámci této metody je robot postupně naveden na N bodů, přičemž musí platit:

$$N \geqslant 2 \tag{6.2}$$

Ukládáním a následným spojením těchto bodů do plochy v 3D prostoru souřadného systému robotického manipulátoru. Následně dojde k definici pracovního prostoru robota v jeho souřadném systému.

6.4 Metoda využívající 2D kameru

V rámci této metody se zabýváme problematikou kalibrace 2D kamery. Nejdříve bude nutné provést kalibraci jejích vnitřních parametrů a následně provést vzájemnou kalibraci kamery a scény robotického pracoviště. Za tímto účelem bude provedeno sledování poloh značek `ArUco` pomocí 2D kamery. Tento případ je ilustrovaný na obrázku 6.2, kdy máme plochu představující scénu robotického pracoviště s vlastním souřadným systémem. Do této scény umístíme kameru a skrze `ArUco` marker, pro něhož je známá transformace, nalezneme transformaci mezi souřadným systémem kamery a scény. Tyto kalibrace budou provedeny pomocí funkcí implementovaných v knihovně `OpenCV`. Přesnost této metody závisí především na kvalitě použité kamery, algoritmů pro kalibraci a zvolené metodě detekce objektu. Oproti mechanické metodě je tato metoda nákladnější a výpočetně náročnější. Její použití by ovšem mělo být pro člověka pohodlnější.



Obrázek 6.2: Ilustrace kalibrace 2D kamery

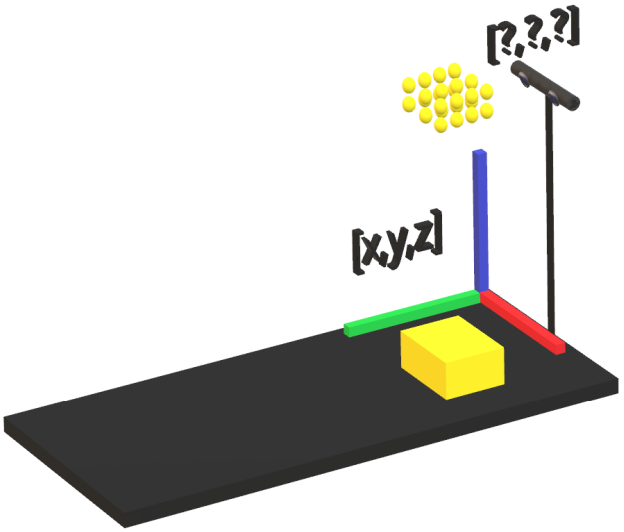
Stacionární kamera a 2D marker

Jako možné použití se jeví případ, kdy je 2D kamera stacionárně umístěna v robotickém pracovišti tak, aby měla dobrou viditelnost na hledaný objekt. Nejprve je nutné provést kalibraci vnitřních parametrů kamery `OpenCV`. Kalibrace vnitřních parametrů proběhne vyhodnocením několika snímků kalibrační matice. Výsledkem této kalibrace je matice vnitřních parametrů kamery společně s pěti koeficienty zkreslení kamery. Následně jako metodu pro detekci objektu zvolíme detekci 2D `ArUco` markeru pomocí funkcí z knihovny `OpenCV`, které nám vrátí pózu tohoto markeru v souřadném systému kamery. Pokud známe přesný model hledaného objektu, můžeme z předem dohodnuté polohy umístění `ArUco` markeru na objektu vypočítat souřadnice jeho středu. V dalším kroku se provede odhad pozice kamery pomocí funkce *solvePnP* za pomocí změřených souřadnic lokalizací `ArUco` markeru a reálných souřadnic známého objektu, na kterém je tento marker upevněn. Tím získáme vzájemnou kalibraci mezi souřadným systémem scény a souřadným systémem kamery.

Pokud doposud nemáme robotické rameno vzájemně zkalibrováno vůči scéně pracoviště, můžeme pro tuto kalibraci zvolit mechanickou metodu popsanou v předchozí kapitole. Následně můžeme zjistit vzájemnou transformaci mezi robotickým ramenem a kamerou.

6.5 Metoda využívající 3D kameru

Poslední uvažovanou metodou bude metoda využívající záznam z 3D kamery. V tomto případě budeme mít k dispozici data ve formátu mračna bodů. Konkrétně se bude jednat o data reprezentující samotný kalibrační objekt a o data reprezentující snímek tohoto objektu pomocí 3D kamery. Pro zpracování těchto dat budeme využívat funkce a algoritmy dostupné v knihovně `PCL` 4.4. Tento případ znázorňuje obrázek 6.3, kdy máme opět plochu s vlastním souřadným systémem představující scénu pracoviště. Do této scény umístíme 3D kameru a pořídíme snímek objektu, pro nějž je ve scéně známá transformace. Na základě zarovnání mračen bodů poté zjistíme transformaci mezi souřadným systémem kamery a scény. Tato metoda bude ze všech tří výpočetně nejnáročnější a nejnákladnější s ohledem na potřebné periferie.



Obrázek 6.3: Ilustrace kalibrace 3D kamery

Stacionární kamera a kalibrační objekt

Uvažujeme případ umístění kamery stacionárně do scény robotického pracoviště. Kameru musíme umístit do prostoru tak, aby mířila na místo, kde se nachází nám známý kalibrační objekt. Kamera tento objekt později nasnímá, čímž získáme 3D data ve formátu mračna bodů. Pomocí knihovny `PCL` a metodami popsanými v kapitole 5 následně zjistíme transformaci kalibračního z jeho nulové pozice do pozice zaznamenané 3D kamerou. Pomocí této transformace jsme schopni zjistit, kde se vůči tomuto objektu 3D kamera nachází. Jelikož zároveň známe také pozici tohoto objektu v rámci scény robotického pracoviště, dojde skrze tento objekt ke kalibraci 3D kamery a scény.

Stejně jako v případě použití 2D kamery, pokud nemáme doposud robotické rameno zkalibrováno vůči scéně pracoviště, lze pro tuto kalibraci využít mechanické metody popsané výše. Jakmile známe transformaci mezi souřadnými systémy kamery a scény pracoviště, a dále mezi souřadnými systémy kamery a scény pracoviště, můžeme zjistit také vzájemnou transformaci mezi souřadným systémem robotického ramene a 3D kamery, čímž dojde k jejich vzájemné kalibraci.

Kapitola 7

Implementace

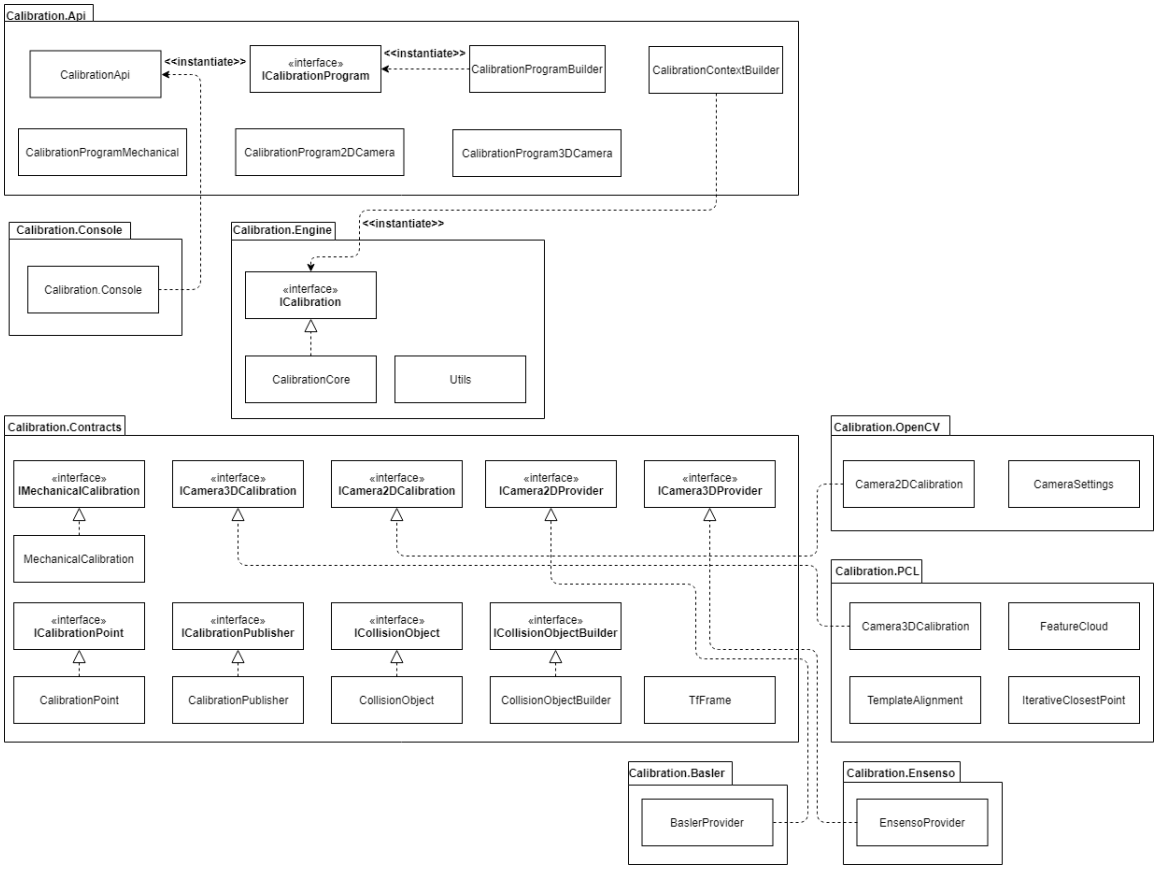
V této kapitole bude popsána implementace jednotlivých částí aplikace. Budou zde také popsány jednotlivé kalibrační objekty použité pro mechanickou kalibraci a kalibraci 3D kamery, společně s restrikcemi při jejich použití. Na závěr kapitoly bude popsána integrace výsledné aplikace do systému ARCOR.

7.1 Struktura aplikace

Aplikace je vytvořena pro robotický systém ROS, konkrétně verzi Kinetic, běžící pod operačním systémem Ubuntu 16.04 (Xenial), v programovacím jazyce C++. Jedná se pouze o konzolovou aplikaci, jelikož bude primárně sloužit pro integraci do větších robotických systémů (například do systému ARCOR) u kterých se předpokládá vlastní grafické uživatelské rozhraní. Program tvoří v systému ROS uzel \calibration_core, který pomocí uzlu \move_group systému MoveIt! komunikuje s robotickým ramenem a scénou robotického pracoviště. V případě mechanické kalibrace aplikace navíc komunikuje s dalšími uzly, které slouží pro přímou interakci s robotickým ramenem. V závislosti na zvoleném typu kalibračního programu vytvoří aplikace jednu z následujících sběrnic:

1. \calibration\mechanical
2. \calibration\camera2D
3. \calibration\camera3D

V rámci nich po ukončení kalibrace zveřejní výslednou transformaci mezi souřadným systémem scény a souřadným systémem kalibrovaného objektu. Tuto kalibraci posílá ve formátu typickém pro knihovnu TF, což dovoluje její snadné použití v systému. Výsledné kalibrace je možné také ukládat do souborů pro jejich zálohu a opětovné použití. Aplikace se skládá z více projektů, jak lze vidět na obrázku 7.1. Nyní budou popsány jednotlivé balíčky aplikace.



Obrázek 7.1: Diagram balíčků aplikace

Calibration.Api

Tento balíček v sobě obsahuje třídu `CalibrationProgramBuilder`, která je zodpovědná za instanciaci hlavního kalibračního programu. Chování tohoto kalibračního programu je definováno rozhraním `ICalibrationProgram`, které je implementované třemi třídami:

1. `CalibrationProgramMechanical` - implementuje program mechanické kalibrace.
2. `CalibrationProgram2DCamera` - implementuje program kalibrace 2D kamery.
3. `CalibrationProgram3DCamera` - implementuje program kalibrace 3D kamery.

kSoučástí tohoto balíčku je také třída `CalibrationContextBuilder`, která má na starosti instanciování třídy `CalibrationCore`, jejíž funkcionalita je definovaná rozhraním `ICalibration`. Tento objekt obsahuje veškeré jednotlivé kalibrační funkce, které výše uvedené třídy reprezentující jednotlivé kalibrační programy v závislosti na svém typu provolávají. Na závěr zde najdeme třídu `CalibrationApi`, která reprezentuje `API` (*Application Programming Interface*) výsledné aplikace. V rámci tohoto `API` existují tři metody, z nichž každá vyvolá jiný typ kalibračního programu. Konkrétně se jedná o metody:

1. `runMechanicalCalibration()` - spustí mechanickou kalibraci.
2. `runCamera2DCalibration()` - spustí kalibraci 2D kamery.
3. `runCamera3DCalibration()` - spustí kalibraci 3D kamery.

Calibration.Console

V rámci tohoto balíčku je implementován vstupní bod aplikace a její jednoduché konzolové uživatelské rozhraní ve formě spustitelného souboru `./Calibration.Console`. Tento soubor využívá implementované `API` popsané v předchozí kapitole. Aplikace uživatele provede pomocí jednoduchých voleb celým procesem kalibrace.

Calibration.Engine

Uvnitř následujícího balíčku je implementována stěžejní třída aplikace, konkrétně třída `CalibrationCore`. Tato třída je definována svým rozhraním `ICalibration`. Při instanciaci této třídy dojde k inicializaci aplikace v systému `ROS`. V rámci této inicializace vznikne v systému uzel `\calibration_core`. Tato třída ve svém konstruktoru přijímá objekty reprezentující funkcionalitu každého jednotlivého typu kalibračního programu. Tyto objekty budou popsány v rámci balíčku `Calibration.Contracts`. Dále dojde k načtení konfiguračního souboru `calibration_config.xml`, který v sobě obsahuje informace jako například:

- jméno pro inicializaci uzlu `\move_group` systému `MoveIt!`,
- jména rámců pro jednotlivé periferie,
- jména kalibračních objektů,
- názvy sběrnic pro publikování výsledných kalibrací,
- zvolenou metodu kalibrace

Na základě zvoleného typu kalibračního programu se vytvoří v systému `ROS` jednotlivé sběrnice, na které se později budou publikovat výsledné transformace. Dále zde dojde k inicializaci objektů rozhraní uzlu `\move_group` systému `MoveIt!` reprezentující robotický manipulátor a scénu pro plánování. Díky prvnímu jmenovanému jsme schopni komunikovat s robotickým ramenem. Druhý objekt nám dovoluje spravovat kolizní objekty ve scéně.

Dále je zde implementována třída `Utils` obsahující pomocné statické metody. Nejčastěji se jedná o metody pro konverzi datových typů vyskytujících se v aplikaci, nebo například zápis pořízených snímků kamerou do souboru.

Calibration.Contracts

Balíček obsahuje mimo jiné třídu `CalibrationPoint`, která reprezentuje datový typ pro souřadnice bodu v trojrozměrném souřadném systému robotického manipulátoru. Dále obsahuje třídu `TfFrame` pro definici názvů jednotlivých rámců využívaných v systému. Najdeme zde také třídy `CollisionObject` a `CollisionObjectBuilder`. První jmenovaná třída reprezentuje kolizní objekty ve scéně systému `ROS`. Druhá slouží pro jejich vytvoření a aplikování do systému. Dovoluje nám vytvořit kolizní objekt daný základními geometrickými útvary, jako jsou kvádr, koule, kužel, nebo válec. Tyto útvary jsou definovány pomocí zprávy systému `ROS`. Její struktura je znázorněna v tabulce A.3. Tato primitiva jsou využita především v mechanické kalibraci při zaměřování nových kolizních objektů ramenem robotického manipulátoru. Dále nám dovoluje vytvořit kolizní objekt definovaný trojúhelníkovou sítí, tzv. *mesh*, což slouží například pro definici kolizního objektu pracovní plochy, nebo pro vytvoření kolizní oblasti kolem robotického manipulátoru při definici jeho pracovního prostoru. Struktura zprávy definující takový kolizní objekt je znázorněna v tabulce A.4.

Dále zde najdeme tři rozhraní definující funkcionalitu tříd pro konkrétní typ kalibračního programu. Jedná se o rozhraní:

1. `IMechanicalCalibration` - rozhraní třídy mechanické kalibrace.
2. `ICamera2DCalibration` - rozhraní třídy kalibrace 2D kamery.
3. `ICamera3DCalibration` - rozhraní třídy kalibrace 3D kamery.

Rozhraní pro mechanickou kalibraci je implementováno třídou `MechanicalCalibration`, která je také součástí tohoto balíku. Obsahuje implementaci metody pro automatický výpočet transformační matice mezi dvěma souřadnými systémy, která byla popsána v kapitole 5.1. Dále obsahuje metody pro implementaci metod na zaměření kolizních objektů, plochy a pracovního prostoru, popsaných v kapitole 6.3. Rozhraní pro 2D kameru je poté implementováno třídou `Camera2DCalibration` umístěnou v balíku `Calibration.OpenCV`. Poslední rozhraní pro kalibraci 3D kamery je implementováno třídou `Camera3DCalibration` uvnitř balíku `Calibration.PCL`. Obě tyto třídy budou popsány v následujících kapitolách.

Dále se zde nachází třída `CalibrationPublisher`, která se stará o publikování výsledných kalibrací. Konkrétně lze tyto kalibrace publikovat na předem danou sběrnici systému `ROS`. Druhou možností je publikování kalibrace do souboru ve formátu `YAML`.

Nachází se zde také dvojice rozhraní `ICamera2DProvider` a `ICamera3DProvider`, definující funkcionalitu pro práci s 2D, respektive 3D kamerou. Jelikož není technicky možné implementovat třídu, která by byla schopná komunikovat s libovolnou 2D nebo 3D kamerou, je nutné při použití vlastní kamery respektovat toto rozhraní a implementovat jeho jednotlivé funkce.

Calibration.OpenCV

Tento balíček obsahuje třídy s metodami knihovny `OpenCV` sloužícími pro kalibraci 2D kamery. Je zde třída `Camera2DCalibration`, implementující rozhraní `ICamera2DCalibration` zmíněné v předchozí kapitole. V rámci této třídy jsou implementovány metody pro detekci `ArUco` markerů, jak je popsáno v kapitole 5.3. Dále jsou zde metody pro kalibraci vnitřních a vnějších parametrů kamery, které slouží pro korekci zkreslení obrazu kamery a odhadnutí její pozice v souřadném systému.

Calibration.PCL

V dalším balíčku jsou umístěny třídy obsahující metody knihovny `PCL` implementující algoritmus `ICP` popsaný v kapitole 5.5 a algoritmus pro `Template Alignment` popsaný v kapitole 5.4. Je zde přítomná třída `Camera3DCalibration` implementující rozhraní popsané v balíčku `Calibration.Contracts`.

Calibration.Basler

Jedná se o balíček obsahující třídu `BaslerProvider`, sloužící pro ovládání průmyslové 2D kamery Basler, která byla využívána v rámci aplikace. Pomocí této třídy lze inicializovat síťově připojenou kameru a pořizovat snímky z kamery, které následně slouží pro kalibraci vnitřních a vnějších parametrů kamery.

Calibration.Ensenso

Podobně jako předchozí balíček, také tento obsahuje třídu sloužící pro ovládání kamery, tentokrát průmyslové stereoskopické 3D kamery Ensenso. Jedná se o třídu `EnsensoProvider`, která provede inicializaci síťově připojené kamery a dovoloje pořizovat výškovou mapu snímané scény.

7.2 Mechanická kalibrace

Mechanická kalibrace byla implementována za účelem kalibrace ramene robotického manipulátoru vůči existující scéně robotického pracoviště skrze kalibrační objekt. V rámci aplikace je možné zvolit z několika typů kalibračního programu, potažmo z metod popsaných v kapitole 6.3, jejichž implementace bude popsána v následujících kapitolách. Je ovšem nutné zmínit, že kalibrací robotického ramene vůči scéně robotického pracoviště, ve smyslu nalezení transformace mezi jejich souřadnými systémy, se zabývá pouze jedna z nich. Ostatní metody slouží spíše k interakci s již zkalibrovanou scénou a k její jednoduché modifikaci. Nejprve bude popsán použitý kalibrační objekt, společně s kalibračním efektorem, které byly navrženy speciálně pro tento typ kalibrace. Následovat bude popis jednotlivých metod aplikace.

Kalibrační objekt

Za účelem mechanické kalibrace pomocí robotického ramene byl navržen a následně pomocí 3D tiskárny zhotoven kalibrační objekt ve tvaru krychle, jehož počítačový mode je znázorněný na obrázku 7.2. Tento objekt slouží pro zaměření čtyř přesně definovaných bodů, které leží v rozdílné výšce podél jeho hran. Díky tomu je lze využít v rámci metody automatického výpočtu transformační matice popsané v kapitole 5.1. Rozměry tohoto objektu jsou popsány v následující tabulce 7.1:

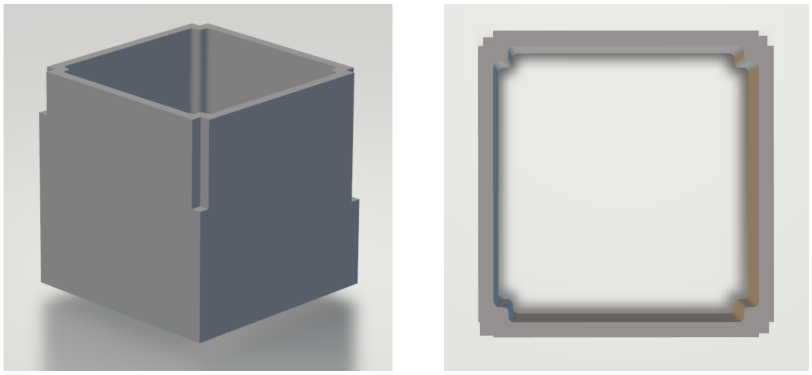
Rozměr	Délka [cm]
šířka	10.0
výška	10.0
hloubka	10.0

Tabulka 7.1: Rozměry kalibračního objektu

Jak již bylo zmíněno, podél hran kalibračního objektu jsou umístěny jednotlivé kalibrační body pro zaměření, každý v rozdílné výšce. V tabulce 7.2 je popsáno jejich rozložení:

Bod	Výška [cm]
p_1	2.0
p_2	4.0
p_3	6.0
p_4	8.0

Tabulka 7.2: Rozmístění kalibračních bodů



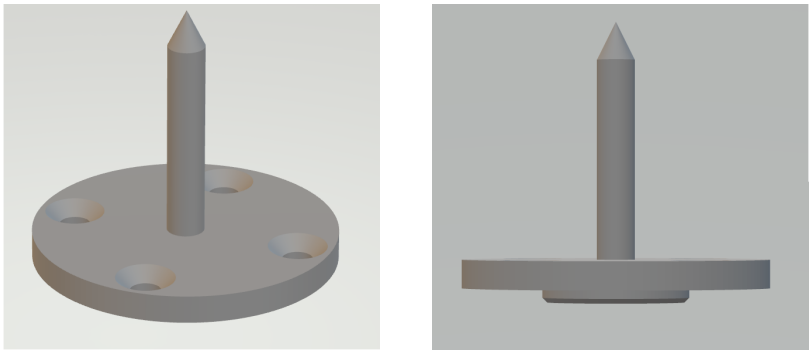
Obrázek 7.2: Kalibrační objekt pro mechanickou kalibraci

Kalibrační efektor

Pro co možná nejpřesnější zaměření kalibračních bodů na objektu z předchozí kapitoly byl navržen také speciální kalibrační efektor, který je znázorněný na obrázku 7.3. Jedná se o kruhovou násadu s trnem, jehož konec slouží k zaměření souřadnic pro kalibraci. Tento efektor byl také zhotoven na 3D tiskárně. Při použití se vloží jako součást počítačového modelu robotického ramene, což nám umožní získávat pomocí systému `MoveIt!` trojrozměrné souřadnice z konce jeho trnu.

Rozměr	Délka [cm]
poloměr	3.5
výška	0.5
délka trnu	5.0

Tabulka 7.3: Rozměry kalibračního efektoru



Obrázek 7.3: Kalibrační efektor

Kalibrace robotického ramene pomocí kalibračního objektu

Kalibrace ramene robotického manipulátoru využívá implementaci metody pro automatický výpočet transformační matice mezi dvěma souřadnými systémy, která je popsána v kapitole 5.1. Tato metoda očekává jako vstupní data dvě množiny odpovídajících souřadnic, každá v jiném trojrozměrném souřadném systému. Za účelem definice těchto souřadnic v souřadném systému scény robotického pracoviště byl navrhnut kalibrační objekt, který je umístěn

do existující scény. Tento kalibrační objekt obsahuje svůj vlastní rámec popisující jeho souřadný systém. Další rámce tvoří jeho jednotlivé kalibrační body. To nám dovoluje jednoduše přistupovat k tomuto objektu a příslušným kalibračním bodům pomocí knihovny `TF` a také jednoduše provádět transformace mezi kalibrem a výchozím souřadným systémem scény. Proces kalibračního programu probíhá následovně:

1. Uživatel v aplikaci zvolí kalibraci robotického ramene pomocí kalibračního objektu.
2. Aplikace uživatele vyzve k navedení ramene na kalibrační body.
3. Uživatel postupně navede robotické rameno na všechny čtyři kalibrační body.
4. Uživatel ukončí proces navádění ramene na kalibrační body.
5. Aplikace přistoupí pomocí knihovny `TF` ke kalibračním bodům v rámci scény.
6. Aplikace pomocí metody 5.1 vypočítá transformační matici.
7. Výsledná transformace je publikována mechanismem zpráv na sběrnici systému `ROS`.

Omezení této metody spočívá v nutnosti sesouhlasit obě množiny trojrozměrných souřadnic kalibračních bodů. Z toho důvodu je nutné zavést restrikci, která přesně definuje pořadí, v jakém se jednotlivé souřadnice kalibračních bodů budou ukládat.

Kalibrace robotického ramene pomocí 2D markerů

Stejně jako v předchozím případě za použití kalibračního objektu, i nyní je pro výpočet výsledné kalibrace využita implementace metody pro automatický výpočet transformační matice mezi dvěma souřadnými systémy z kapitoly 5.1. Rozdíl oproti předchozí metodě je v procesu zaměření kalibračních bodů. Tato metoda uvažuje dvourozměrné markery umístěné někde na pracovní ploše scény v dosahu robotického ramene. Stejně jako v případě kalibračních bodů umístěných na kalibračním objektu, i zde budou jednotlivé markery obsahovat svoje vlastní rámce definující jejich souřadný systém. Proces kalibračního programu má tedy tyto kroky:

1. Uživatel v aplikaci zvolí kalibraci robotického ramene pomocí 2D markerů.
2. Aplikace uživatele vyzve k navedení ramene na kalibrační markery.
3. Uživatel postupně navede robotické rameno na všechny tři kalibrační markery.
4. Uživatel ukončí proces navádění ramene na kalibrační markery.
5. Aplikace přistoupí pomocí knihovny `TF` ke kalibračním markerům v rámci scény.
6. Aplikace pomocí metody 5.1 vypočítá transformační matici.
7. Výsledná transformace je publikována mechanismem zpráv na sběrnici systému `ROS`.

Při použití této metody mechanické kalibrace je nutné vypořádat se s jejím hlavním omezením popsaném v kapitole 5.1. Toto omezení říká, že množiny porovnávaných bodů nesmějí v rámci jednoho souřadného systému ležet ve stejné rovině, což by platilo při jejich umístění na pracovní plochu. Stejně tak se zmiňuje o nutnosti zaměření alespoň čtyř bodů pro výpočet transformační matice. Obě tato omezení vyřešíme tím, že ke snímaným bodům umístíme jejich virtuální dvojice posunuté v ose z. Při navádění ramene na jednotlivé body poté budeme ukládat jak aktuální pozici koncového efektoru, tak jeho souřadnici posunutou v ose z o stejnou hodnotu jako virtuální dvojice kalibračního 2D markeru.

Zaměření kolizních objektů

Tato metoda slouží především pro snadné definování kolizních objektů uvnitř scény robotického pracoviště bez nutnosti zasahování do zdrojových kódů programu, nebo do počítačového modelu. Takto zaměřené kolizní objekty jsou definované zprávou popsanou v kapitole A.3. Je implementována jako součást programu pro mechanickou kalibraci robotického ramene. Proces zaměření kolizního objektu má následující kroky:

1. Uživatel v aplikaci zvolí zaměření nového kolizního objektu.
2. Aplikace uživatele vyzve, aby navedl robotické rameno na čtyři body:
   (a) Bod na konci horní hrany objektu v ose Y
   (b) Bod uprostřed na horní hraně objektu v ose Z
   (c) Bod na konci horní hrany objektu v ose X
   (d) Bod kdekoliv na povrchu, na kterém se objekt nachází.
3. Uživatel ukončí proces zaměřování kolizního objektu.
4. Aplikace vypočítá rozměry, polohu a orientaci objektu.
5. Aplikace vytvoří kolizní objekt a vloží jej do scény.

Aplikace ze zaměřených trojrozměrných souřadnic jednotlivých bodů vypočítá rozměry nového kolizního objektu, společně s jeho pivotem. Dále se vypočítají směrové vektory v jednotlivých osách. Pomocí těchto vektorů je poté vypočítán úhel mezi osami souřadného systému robotického manipulátoru a kolizního objektu. Pro výpočet úhlů mezi směrovými vektory je použitý následující vztah:

$$\cos\theta = \frac{\vec{u} \cdot \vec{u}}{||\vec{u}||||\vec{v}||} \tag{7.1}$$

Úhly vypočítané rovnicí 7.1 se použijí pro orientaci objektu v prostoru robotického manipulátoru. Jelikož jsou všechny nové kolizní objekty zaměřené robotickým ramenem, jsou pozicovány relativně od počátku jeho souřadného systému.

Zaměření kolizní plochy

Další z metod implementovaných v rámci mechanické kalibrace robotického ramene je metoda pro definici kolizní pracovní plochy ve scéně robotického pracoviště. Tato metoda nám dovoluje jednoduše definovat větší kolizní plochu různých tvarů, bez nutnosti zásahu do zdrojových kódů aplikace nebo do počítačového modelu. Pro definici kolizního objektu představující kolizní plochu je využita trojúhelníková síť. Tento objekt je tvořený zprávou popsanou v kapitole A.4. Nejsme tedy limitování pouze na geometrická primitiva. Metoda probíhá v následujících krocích:

1. Uživatel v aplikaci zvolí zaměření kolizní plochy.
2. Aplikace uživatele vyzve, aby navedl robotické rameno na body ohraničující zaměřovanou plochu.
3. Uživatel postupně navede rameno na tyto body a uloží jejich souřadnice.

4. Uživatel ukončí proces zaměřování kolizní plochy.

5. Aplikace z uložených bodů vytvoří trojúhelníkovou síť pro kolizní objekt a vloží jej do scény.

Zaměření pracovního prostoru

Stejně jako v předchozí metodě, i zde využíváme pro definici kolizního objektu trojúhelníkovou síť. Samotný proces definice pracovního prostoru probíhá analogicky s definicí kolizní plochy. Rozdíl je pouze v logice vytvoření kolizního objektu. Zatím co v případě definice kolizní plochy vytváříme trojúhelníkovou síť spojující body do roviny ve stejné výšce tak, aby nám vznikla plocha, v tomto případě vytváříme ke každému zaměřenému bodu jeho obraz posunutý v ose z. Výsledným spojením těchto bodů do trojúhelníkové sítě dostaneme plochu tvořící ohraničení zaměřeného prostoru. Toto ohraničení nám zajistí, že robotické rameno nebude schopné naplánovat pohyb za jeho plochu, čímž dojde k vymezení pracovního prostoru ramene bez nutnosti zásahu do počítačového modelu. Proces definice pracovního prostoru probíhá následovně:

1. Uživatel v aplikaci zvolí zaměření pracovního prostoru.

2. Aplikace uživatele vyzve, aby navedl robotické rameno na body ohraničující pracovní prostor.

3. Uživatel postupně navede rameno na tyto body a uloží jejich souřadnice.

4. Aplikace ke každému zaměřenému bodu vytvoří jeho dvojici posunutou v ose z.

5. Uživatel ukončí proces zaměřování pracovního prostoru.

6. Aplikace z uložených bodů vytvoří trojúhelníkovou síť pro kolizní objekt a vloží jej do scény.

7.3 Kalibrace 2D kamery

Kalibraci 2D kamery v rámci této aplikace lze využít jak pro kalibraci jejích vnitřních parametrů, tak pro kalibraci jejích vnějších parametrů. První zmíněná kalibrace se zabývá korekcí obrazu snímaného kamerou, tedy problémem popsaném v kapitole 2.2. K jeho řešení používá metodu z kapitoly 5.2. Druhá kalibrace se věnuje problému odhadnutí pozice kamery v globálním trojrozměrném souřadném systému, což je problematika popsaná v kapitole 2.3. Zde je k řešení použita metoda popsaná v kapitole 5.3. Pro úspěšné spuštění kalibračního programu pro 2D kameru je nutné poskytnou implementaci rozhraní `ICamera2DProvider`, pomocí něhož dochází k pořizování snímků skrze kameru.

Kalibrace vnitřních parametrů

Tato metoda slouží ke korekci výsledného obrazu snímaného kamerou. Jejím výstupem je matice vnitřních parametrů kamery, společně s pěti koeficienty zkreslení. Obojí je popsáno v kapitole 2.2. Ke kalibraci se využívá některého z typů kalibračních obrazců, které jsou znázorněné na obrázku. 5.1. Tento kalibrační obrazec je několikrát nasnímán podle pravidel z rovnice 5.15. Proces kalibrace vnitřních parametrů kamery je následující:

1. Uživatel v aplikaci zvolí kalibraci vnitřních parametrů 2D kamery.
2. Aplikace načte konfiguraci kalibrace ze souboru `default.xml`.
3. Aplikace uživatele vyzve, aby nasnímal kalibrační vzor v různých úhlech.
4. Uživatel postupně pořizuje a ukládá snímky kalibračního vzoru.
5. Aplikace z uložených snímků kalibračního vzoru vypočítá kalibraci vnitřních parametrů kamery a pět koeficientů zkreslení.
6. Výsledné hodnoty jsou zapsány do souboru `out_camera_data.xml`.

Výsledná matice vnitřních parametrů kamery společně s pěti koeficienty zkreslení jsou následně použity při kalibraci vnějších parametrů kamery. Pokud tedy v aplikaci neprovedeme nejdříve kalibraci vnitřních parametrů kamery, musíme soubor `out_camera_data.xml` s těmito hodnotami programu poskytnout zvlášť.

Kalibrace vnějších parametrů

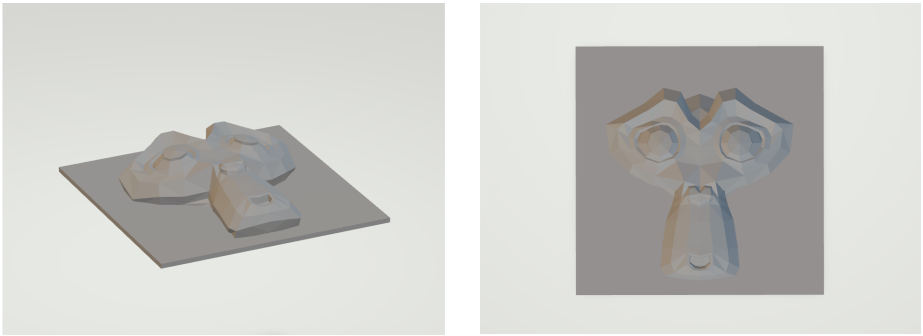
7.4 Kalibrace 3D kamery

Kalibrace 3D kamery je v aplikaci implementována za účelem kalibrace jejích vnějších parametrů, neboli zjištění pozice a orientace kamery v globálním trojrozměrném souřadném systému skrze kalibrační objekt.

Kalibrační objekt

Tento kalibrační objekt, znázorněný na obrázku 7.4, byl navržený a následně zhotovený na 3D tiskárně za účelem kalibrace 3D kamery. Hlavní motivací pro použití tohoto motivu obličeje, připomínající tvar písmene `Y`, byl fakt, že se jedná o asymetrický, výškově proměnlivý a zároveň dosti vypovídající motiv. Lze tedy předpokládat, že bude při pořizování mračna bodů ze stacionárně umístěné kamery nad kalibrem poskytovat potenciálně dobré výsledky snímaní.

Rozměr	Délka [cm]
šířka	10.0
výška	2.0
hloubka	10.0



Obrázek 7.4: Kalibrační objekt pro kalibraci 3D kamery

Kalibrace vnějších parametrů

Pro kalibraci vnějších parametrů 3D kamery jsou implementovány metody popsané v kapitolách 5.4 a 5.5. První z těchto metod, tedy `Template Alignment`, slouží pro výchozí odhad zarovnání dat ve formátu mračna bodů počítačového modelu kalibračního objektu z předchozí kapitoly a výsledného snímku tohoto modelu pořízeném 3D kamerou. Výsledkem této metody je transformační matice, která se použije pro transformaci dat počítačového modelu kalibračního objektu. V druhé fázi se použije metoda `ICP` na transformovaná data počítačového modelu a dříve pořízený snímek kalibračního objektu 3D kamerou. Metoda pracuje v několika iteracích, ve kterých se snaží o co nejpřesnější zarovnání mezi jednotlivými body předložených dat. Kalibrační proces má následující kroky:

1. Uživatel v aplikaci zvolí kalibraci 3D kamery.
2. Aplikace pořídí pomocí 3D kamery snímek pracovní plochy s kalibračním objektem.
3. Aplikace provede odfiltrování pracovní plochy z výsledného snímku.
4. Aplikace aplikuje metodu `Template Alignment`.
5. Aplikace aplikuje metodu `ICP`.
6. Aplikace provede násobení výsledných transformací obou metod, čímž získá finální transformaci.
7. Výsledná transformace je publikována mechanismem zpráv na sběrnici systému ROS.

7.5 Integrace do systému ARCOR

Jedním z bodů této diplomové práce byla také integrace výsledné aplikace a jejich jednotlivých metod do robotického systému `ARCOR`. Za tímto účelem bylo využito `API` z balíčku `Calibration.Api` popsané v kapitole 7.1. Jak se zmiňuje tato kapitola, v rámci `API` jsou implementovány tři metody, které spouští jednotlivé typy kalibračního programu. Stěžejním úkolem v rámci této integrace bylo pochopit architekturu a chování robotického systému `ARCOR`.

Architektura systému

Jak je zmíněno v kapitole 4.5, tento systém se skládá z několika samostatných uzlů systému `ROS` s různou funkcionalitou a zodpovědností. Konkrétně nás zajímají jednotlivé uzly z balíčku `art_calibration` věnující se kalibraci systému, především pak:

- `calibration.py`
- `cell_calibration.py`

První z výše zmíněných uzlů má na starosti autonomní kalibraci celého systému. Tento uzel si z parametrického serveru `rosparam` systému `ROS` zjistí jednotlivé definované periferie systému, interně označované jako tzv. *cells*, neboli buňky. Může se jednat o periferie jako je například kinect, robotický manipulátor, nebo kamery. Prvním úkolem tedy bylo zadefinovat do systému nové označení buněk pro kalibraci ramene robotického manipulátoru,

společně s 2D a 3D kamerou. Toho bylo docíleno upravením inicializačního konfiguračního souboru, tzv. *setup* souboru.

Druhý zmíněný soubor představuje uzel pro kalibraci těchto periferií systému. Uzel reprezentující tento objekt je v rámci skriptu `calibration.py` vytvořen pro každou jednotlivou buňku získanou z parametrického serveru. V rámci jeho instanciace je vytvořen její kalibrační uzel. Tento uzel se pokusí získat z parametrického serveru transformaci mezi jeho a globálním souřadným systémem. V případě že se zde tato kalibrace nenachází, pokusí se uzel provést autonomní kalibraci. Jejím výstupem je transformace ve formátu knihovny `TF`, kterou následně umístí na parametrický server. Ukončení kalibračního procesu je indikováno také nastavením vnitřního příznaku `calibrated` na hodnotu `True`, což značí že je daný uzel úspěšně zkalibrován. Následně je tato hodnota publikována do systému.

Řešení

Na základě těchto poznatků byl vytvořen skript `external_calibration.py`, který slouží pro kalibraci nově definovaných periferií. V rámci jednotlivých instancí tohoto uzlu pro kalibraci ramene robotického manipulátoru společně s 2D a 3D kamerou je zkontrolováno, zda jsou v souborovém systému aplikace přítomny soubory ve formátu `YAML` obsahující jednotlivé transformace. Pokud ano, tyto transformace jsou z těchto souborů načteny a dále publikovány do systému. V opačném případě je vytvořena instance `API` kalibrační aplikace. Na základě toho, o jakou buňku, respektive o jakou periferii se aktuálně jedná, je následně tato aplikace spuštěna s odpovídajícím kalibračním programem. Běh aplikace a jednotlivých metod je stejný, jako je popsán v kapitolách 7.2 - 7.4. Výsledkem běhu aplikace je publikování výsledné transformace do souboru a mechanismem zasílání zpráv na odpovídající sběrnici systému.

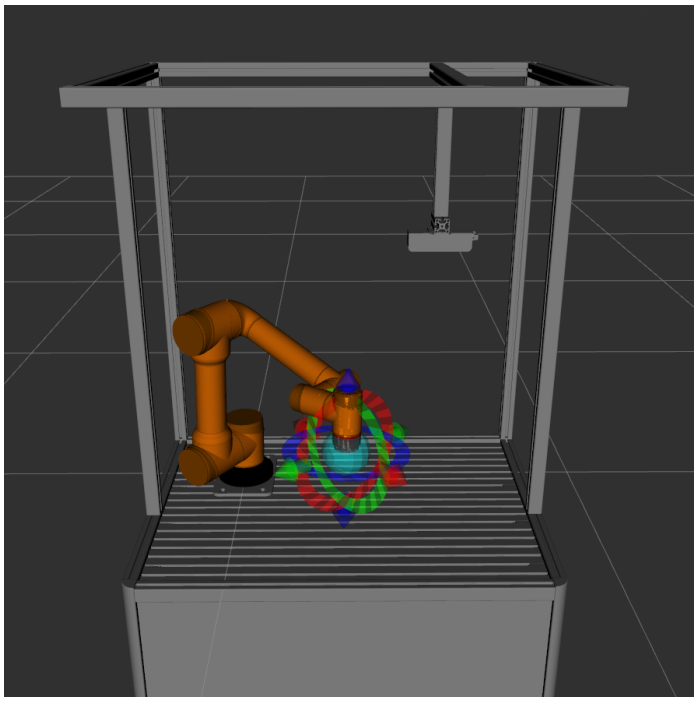
Jelikož je celý systém `ARCOR` implementovaný v jazyce `Python`, zatímco tato kalibrační aplikace je implementována v jazyku `C++`, bylo nutné najít a vhodně zvolit způsob vzájemného propojení obou systémů. Za tímto účelem byl v systému `ARCOR` pomocí knihovny `ctypes` implementován tzv. *wrap* kódu jazyka `C++`, který slouží pro vytvoření instance `API` a vyvolání některé z kalibračních metod. Tento *wrap* je implementován třídou `CalibrationApi`. Uvnitř konstruktoru této třídy definujeme datové typy jazyka `C` pro návratové hodnoty a argumenty jednotlivých metod, aby je bylo možné odchytit v programu. V tomto případě program po úspěšném procesu kalibrace vrací hodnotu `True`, která se využije pro nastavení vnitřního příznaku `calibrated`.

Kapitola 8

Testování

8.1 Scéna robotického pracoviště

Testování probíhalo v rámci scény robotického pracoviště zobrazeného na obrázku 8.1, které poskytla zadavatelská firma Kinalisoft s.r.o.. Toto pracoviště obsahovalo rameno kolaborativního robotického manipulátoru umístěné na pracovní ploše. Scéna byla ohraničená konstrukcí, na níž byla stacionárně umístěna stereoskopická 3D kamera. K tomuto pracovišti byl poskytnutý konfigurační soubor `URDF`, jehož úpravou lze do počítačového modelu přidávat nové rámce a kolizní objekty. Jednotlivé periferie tohoto robotického pracoviště, na nichž kromě testování probíhal i samotný vývoj aplikace, budou podrobněji popsány v následujících kapitolách.



Obrázek 8.1: Model robotického pracoviště pro testování

Veškeré prvky byly v této scéně umístěny relativně vzhledem k základně robotického ramene. Jelikož tato aplikace počítá se scénářem, kdy robotické rameno nemá žádnou před-

stavu o tom, jak vypadá reálná scéna kolem něj, byl za účelem simulování této situace vytvořen nový rámec `table`, který představoval nový počátek souřadného systému scény robotického pracoviště. V tabulce 8.1 je znázorněna transformace mezi ním a původním počátkem souřadného systému scény, který představoval rámec `world`. Z této tabulky lze vyčíst, že rámec `table` byl záporně posunutý v osách x a y vůči rámci `world`, přičemž si zachoval jeho výšku a orientaci. I když v rámci systému `ROS` jsou rotace v transformacích vyjádřeny pomocí kvaternionu, pro jednoduchost bude ve všech tabulkách s transformacemi znázorněna rotace v Eulerových úhlech, které určují rotace kolem os souřadného systému. Tato transformace bude následně sloužit jako referenční hodnota pro ověření přesnosti vypočítaných transformací v rámci experimentů popsaných v kapitole 8.2.

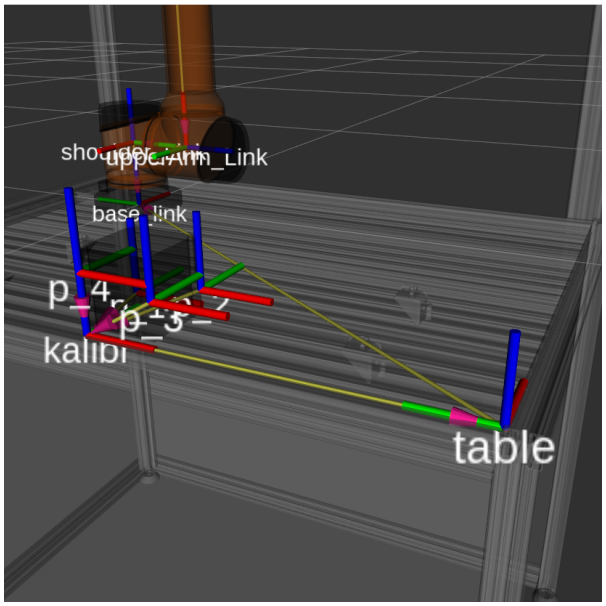
Translace	Hodnota [cm]
x	-53.5
y	-83.0
z	0.0
Rotace	Hodnota [◦]
x	0.0
y	0.0
z	0.0

Tabulka 8.1: Transformace počátku souřadného systému rámce `table`

Na obrázku 8.2 dále vidíme umístění kalibračního objektu do scény. Protože chceme jeho zaměřením získat transformaci mezi robotickým ramenem a scénou, umístíme jej do prostoru pracoviště relativně vzhledem k vytvořenému rámci `table`. Lze si také všimnou, že pro kalibrační objekt byl také vytvořený nový rámec `kalibr`, společně s rámci pro jednotlivé kalibrační body. V tabulce 8.2 je znázorněna transformace mezi rámcem kalibračního objektu `kalibr` a rámcem scény `table`. Lze z ní vyčíst že kalibrační objekt byl od počátku souřadného systému scény posunut v ose y a zároveň byl orotován.

Translace	Hodnota [cm]
x	0.0
y	50.0
z	0.0
Rotace	Hodnota [◦]
x	0.0
y	0.0
z	90.0

Tabulka 8.2: Transformace počátku souřadného systému rámce `kalibr`



Obrázek 8.2: Rámec scény pracoviště, kalibračního objektu a kalibračních bodů

Robotické rameno

K vývoji a testování aplikace bylo zadavatelskou firmou poskytnuto kolaborativní robotické rameno AUBO-i5, zobrazené na obrázku 8.3. Jedná se robotický manipulátor se šesti stupni volnosti s možností ručního navádění, ovládáním přes přiložený pendant, nebo pomocí robotického systému `ROS`. V tabulce 8.3 jsou popsány jeho technické parametry.

Vlastnost	Hodnota
Dosah	924 [cm]
Nosnost	5 [kg]
Opakovatelnost	+/- 0,05 [mm]
Lineární rychlost	2,8 [m/s]

Tabulka 8.3: Technické parametry kolaborativního robotického ramene AUBO-i5

Obrázek 8.3: Kolaborativní robot AUBO-i5 [1]

Použitá 2D kamera

Za účelem testování kalibrace 2D kamery byla zadavatelskou firmou poskytnuta průmyslová RGB 2D kamera Basler acA1920-40gc zobrazená na obrázku 8.4. Tuto kameru lze ovládat skrze ethernetového připojení. Její technické parametry jsou znázorněny v tabulce 8.4.

Vlastnost	Hodnota
Rozlišení	1920 x 1200 [px]
Snímková frekvence	42 [fps]
Typ snímače	CMOS

Tabulka 8.4: Technické parametry 2D kamery Basler acA1920-40gc

Obrázek 8.4: Průmyslová 2D kamera Basler acA1920-40gc [2]

Použitá 3D kamera

V rámci testování aplikace pro kalibraci 3D kamery byla použitá průmyslová stereoskopická kamera Ensenso N35. Stejně jako v případě 2D kamery Basler, také tuto kameru lze ovládat skrze ethernetové připojení. Kamera dovoluje snímat scénu ve formě souřadnic tvořící výškovou mapu. Taková data lze poté jednoduše převést do formátu mračna bodů a využívat je v aplikaci. Tabulka 8.5 znázorňuje její technické parametry.

Vlastnost	Hodnota
Rozlišení	1280 x 1024 [px]
Snímková frekvence	30 [fps]
Typ snímače	CMOS

Tabulka 8.5: Technické parametry 2D kamery Ensenso N35

Obrázek 8.5: Průmyslová stereoskopická 3D kamera Ensenso N35 [3]

[1]Převzato z: https://www.aubo.cz/produkty/aubo-i5/
[2]Převzato z: https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca1920-40gc/

8.2 Experimenty

Následující kapitola se věnuje experimentálnímu ověření funkcionality jednotlivých metod implementovaných v této kalibrační aplikaci. Experimenty jsou prováděné za účelem ověření zda byly splněny všechny požadavky kladené na výslednou aplikaci. Postupně budou otestovány všechny metody kalibračních programů. Výsledky těchto experimentů budou vizualizovány a diskutovány.

Cílem této diplomové práce bylo nastudovat vhodné metody pro vzájemnou kalibraci ramene robotického manipulátoru, scény robotického pracoviště, 2D a 3D kamery, a ty následně implementovat do aplikace pro robotický systém `ROS`. Aplikace kromě kalibrací jednotlivých prvků také umožňuje jednoduchou interakci se scénou robotického pracoviště za použití robotického ramene.

Experimenty jsou navržené především pro ověření funkčnosti, použitelnosti a efektivity implementovaných metod. Výsledné transformace jednotlivých kalibrací budou porovnány s referenčními hodnotami, čímž bude ověřena jejich přesnost.

Kalibrace robotického ramene pomocí kalibračního objektu

Tento experiment modeluje situaci, kdy máme k dispozici již existující scénu robotického pracoviště a její počítačový model systému `ROS`. Nyní chceme do toho systému vložit libovolný robotický manipulátor. Za tímto účelem musíme navzájem zkalibrovat jejich souřadné systémy. K tomu slouží kalibrační objekt popsaný v kapitole 7.2. Umístění tohoto objektu do scény je popsáno v kapitole 8.1. Pomocí kalibračního koncového efektoru umístěném na robotickém rameni postupně zaměříme jednotlivé body, jak je znázorněno na obrázku 8.6. Takto zaměřené body jsou společně s jejich odpovídajícími dvojicemi ze souřadného systému rámce `table` předány algoritmu z kapitoly 5.1. Výsledky jednotlivých měření jsou zaznamenány v tabulkách 8.6.

Obrázek 8.6: Zaměření kalibračního objektu

[3]Převzato z: https://en.ids-imaging.com/ensenso-n35.html

Měření č.1	
Translace	Hodnota [cm]
x	-53,6571845391
y	-81,4054031416
z	0,0876445679924
Rotace	Hodnota [○]
x	-1,5927081
y	0,4950265
z	0,3910686

Měření č.2	
Translace	Hodnota [cm]
x	-53,4056460418
y	-81,4782808614
z	0,216282726542
Rotace	Hodnota [○]
x	0,3219217
y	0,1784386
z	-0,0005013

Měření č.3	
Translace	Hodnota [cm]
x	-53,9824429591
y	-81,4397913251
z	0,6602461719880
Rotace	Hodnota [○]
x	-2,0096776
y	1,3170185
z	0,1233366

Měření č.4	
Translace	Hodnota [cm]
x	-53,8424429591
y	-81,4438926779
z	0,356318877453
Rotace	Hodnota [○]
x	-1,3112701
y	0,5361885
z	0,0012346

Tabulka 8.6: Výsledné transformace kalibrace pomocí kalibračního objektu

Z jednotlivých naměřených hodnot byla vypočítána průměrná hodnota znázorněná v tabulce 8.7. Tato hodnota byla následně odečtena od referenční hodnoty transformace z tabulky 8.1. Tento výsledek je znázorněný v tabulce 8.8. Ze získaných hodnot je zřejmé, že kalibrace dosahuje poněkud malé chyby činící 2 milimetry v případě translace na ose x. Daleko větší chyby bylo dosaženo při výpočtu translace na ose y, kde už chyba dosahovala průměrně 1,5 centimetru. Tato chyba mohla být způsobena nepřesným zaměřením nového počátku souřadného systému rámce `table`, nebo nepřesným umístěním samotného kalibračního objektu v rámci reálné scény. V případě vypočítaných průměrných hodnot rotací dosahuje tato metoda oproti referenční transformaci největší chyby u rotace kolem osy x, která činí zhruba -1 stupeň.

Translace	Hodnota [cm]
x	-53,721929124775
y	-81,4418420015
z	0,33012308599385
Rotace	Hodnota [○]
x	-1,147933525
y	0,631668025
z	0,128784625

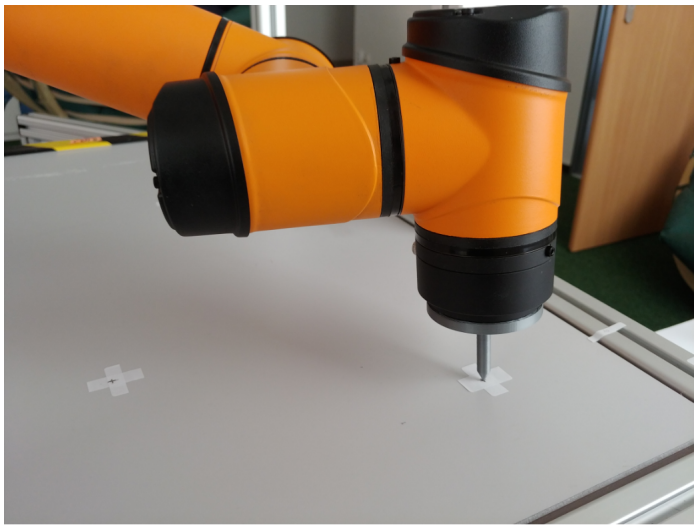
Tabulka 8.7: Průměrná hodnota výsledné transformace

Translace	Hodnota [cm]
x	-0,221929124775
y	1,5581579985
z	-0,33012308599385
Rotace	Hodnota [◦]
x	1,147933525
y	-0,631668025
z	-0,128784625

Tabulka 8.8: Rozdíl mezi referenční transformací a průměrnou hodnotou měření

Kalibrace robotického ramene pomocí 2D markerů

V rámci tohoto experimentu máme stejnou modelovou situaci jako v předchozím případě. Narozdíl od experimentu zmíněném výše, nyní používáme pro mechanickou kalibraci robotického ramene 2D markery rozmístěné na pracovní ploše. Každý tento marker je zanesen do scény robotického pracoviště relativně vzhledem k souřadnému systému rámce `table`. Opět postupně zaměříme tyto markery pomocí kalibračního efektoru, jak je znázorněno na obrázku 8.7. Jak bylo popsáno v kapitole 7.2, k poslednímu zaměřenému bodu uložíme jeho virtuální obraz posunutý v ose z, čímž eliminujeme omezení metody pro automatický výpočet transformační matice z kapitoly 5.1.



Obrázek 8.7: Zaměření kalibračních 2D markerů

Měření č.1	
Translace	Hodnota [cm]
x	-53,5365004107
y	-83,1903694383
z	0,373486809505
Rotace	Hodnota [◦]
x	-0,2589639
y	-0,0186199
z	-0,082166

Měření č.2	
Translace	Hodnota [cm]
x	-53,5242382155
y	-83,3227976696
z	0,366651221384
Rotace	Hodnota [◦]
x	-0,3246734
y	-0,0145326
z	-0,0080626

Měření č.3	
Translace	Hodnota [cm]
x	-53,5123573086
y	-83,2497244718
z	0,234532333121
Rotace	Hodnota [◦]
x	-0,1527584
y	-0,0163694
z	-0,0117742

Měření č.4	
Translace	Hodnota [cm]
x	-53,5483813176
y	-83,2634426361
z	0,305605697768
Rotace	Hodnota [◦]
x	-0,2184679
y	0,0177831
z	0,0584544

Tabulka 8.9: Výsledné transformace kalibrace pomocí 2D markerů

Stejně jako v případě kalibrace pomocí kalibračního objektu, také zde byla v rámci vyhodnocení metody provedena celkově čtyři měření, ze kterých byla následně vypočítána průměrná hodnota, která je znázorněna v tabulce 8.10. Ta byla následně odečtena od referenční transformace z tabulky 8.1 a tento rozdíl byl zaznamenán do tabulky 8.11. Jak je z těchto hodnot zřejmé, metoda mechanické kalibrace robotického ramen pomocí 2D markerů dosáhla daleko lepších výsledků, jako v případě použití kalibračního objektu. Oproti původní chybě translace na ose x, která činila 2 milimetry, v případě této metody činí chyba v průměru pouze 3 desetiny milimetru. Zmenšila se také chyba v případě translace na ose y, konkrétně z původního 1,5 centimetru v průměru na 2 milimetry. Poněkud razantně klesla také nepřesnost v rotacích, jejichž hodnoty jsou již zanedbatelně malé.

Translace	Hodnota [cm]
x	-53,5303693131
y	-83,25658355395
z	0,3200690154445
Rotace	Hodnota [◦]
x	-0,2387159
y	-0,0079347
z	-0,0108871

Tabulka 8.10: Průměrná hodnota výsledné transformace

Translace	Hodnota [cm]
x	-0,0303693131
y	-0,25658355395
z	-0,3200690154445
Rotace	Hodnota [◦]
x	0,2387159
y	0,0079347
z	0,0108871

Tabulka 8.11: Rozdíl mezi referenční transformací a průměrnou hodnotou měření

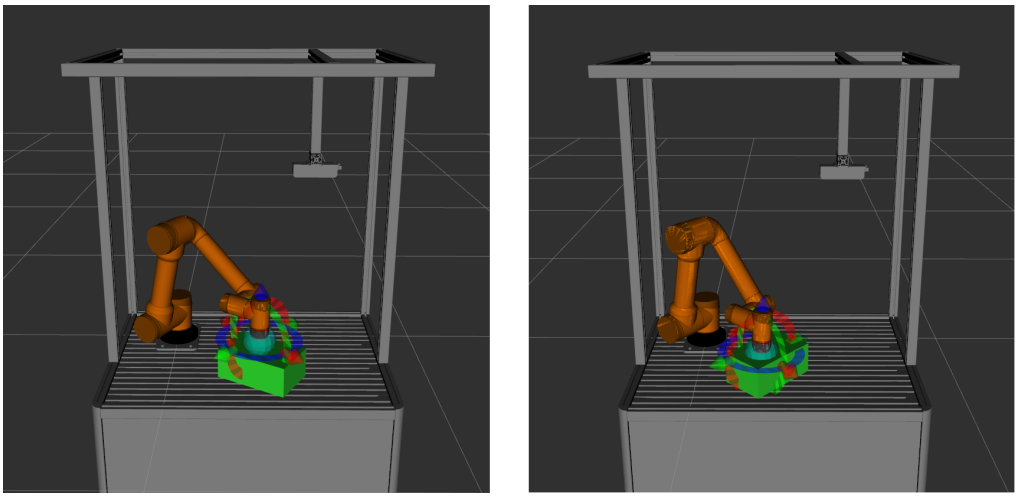
Zaměření kolizního objektu

Cílem tohoto experimentu bylo pomocí robotického ramene zaměřit libovolný objekt v reálné scéně, vytvořit pro něj odpovídající kolizní model a následně jej vložit do počítačové

scény pracoviště. Při provádění experimentu jsem postupoval podle kroků popsaných v kapitole 7.2. Na fotografii znázorněné na obrázku 8.8 můžeme vidět reálnou scénu robotického pracoviště, společně se zaměřovaným kolizním objektem.

Obrázek 8.8: Zaměření kolizního objektu - reálná scéna

Výsledky experimentu jsou zobrazené na obrázku 8.9, přičemž první obrázek zleva odpovídá výše zmíněné fotografii. Lze provést snadnou vizuální kontrolu polohy robotického ramene a zaměřované krabice, abychom mohli prohlásit, že experiment proběhl úspěšně. Druhý obrázek potom demonstruje zaměření této krabice s rozdílným natočením ve scéně, pro ověření, zda funguje výpočet rotace kolizního objektu. Tato metoda byla úspěšně vyzkoušena ve všech čtyřech kvadrantech souřadného systému robotického manipulátoru.

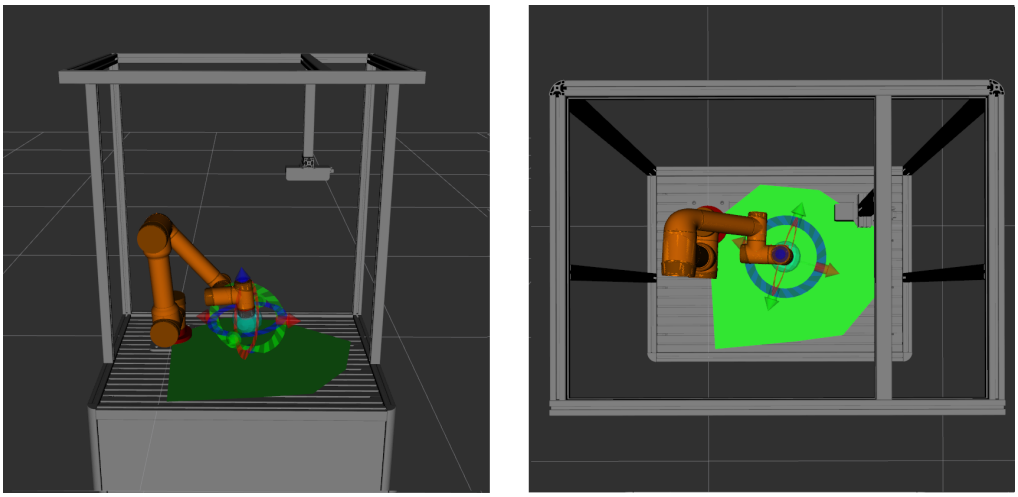


Obrázek 8.9: Zaměření kolizního objektu v různých rotacích

Zaměření kolizní plochy

Následující experiment ověřuje funkcionalitu metody z kapitoly 7.2 pro zaměření kolizní plochy. Při provádění tohoto experimentu bylo robotické rameno navedeno postupně na několik souřadnic vymezujících kolizní plochu. Tyto souřadnice byly postupně ukládány a

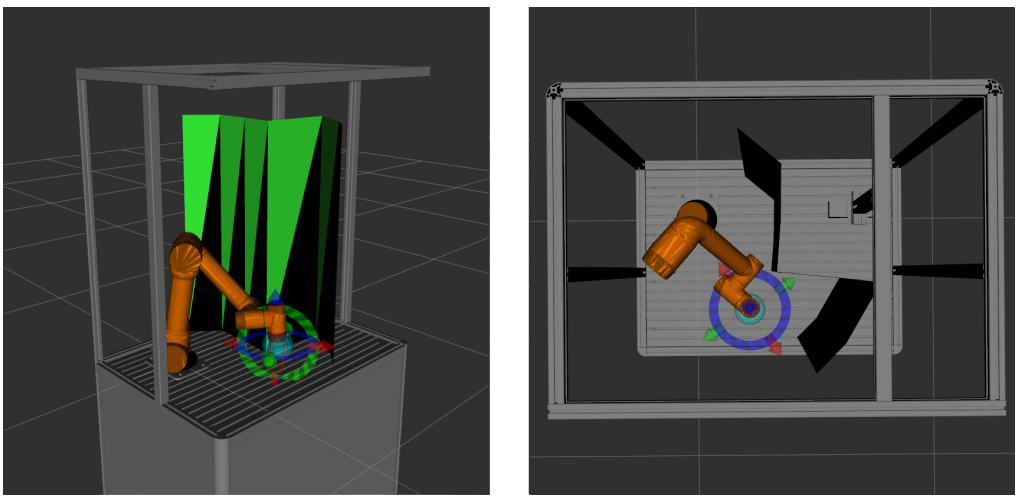
následně z nich byla vytvořena trojúhelníková síť představující kolizní objekt této plochy. Jelikož při ručním navádění robotického ramene nezaměříme všechny souřadnice ve stejné výšce, jsou hodnoty těchto souřadnic v ose z zprůměrovány, aby výsledný objekt skutečně tvořil rovnou plochu. Výsledky tohoto experimentu jsou znázorněny na obrázku 8.10, kdy došlo k úspěšnému zaměření kolizní plochy v rámci robotického pracoviště.



Obrázek 8.10: Zaměření pracovního plochy

Zaměření pracovní plochy

Dalším experimentem bylo ověření metody pro zaměření oblasti vymezující pracovní plochu pro robotické rameno. Tento experiment probíhal velice podobně jako předchozí, s tím rozdílem že nyní je pro každou uloženou souřadnici uložen její obraz posunutý v ose z o hodnotu jednoho metru. Tato hodnota byla zvolena na základě dosahu použitého robotického manipulátoru. Výsledky experimentu jsou znázorněny na obrázku 8.11. Lze na nich spatřit nově vytvořený kolizní objekt z trojúhelníkové sítě, který tvoří virtuální hranici, která nedovoluje robotickému ramenu plánovat pohyb skrze ni.



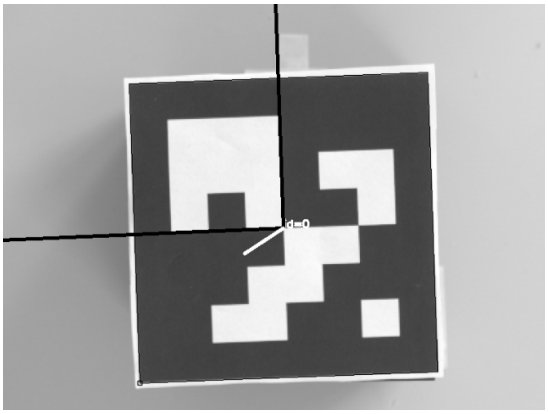
Obrázek 8.11: Zaměření pracovního prostoru

Kalibrace 2D kamery

V případě experimentu zaměřeného na kalibraci 2D kamery pomocí `ArUco` markeru byl tento marker umístěn na kalibrační objekt sloužící pro mechanickou kalibraci. Společně s ním byl umístěn pod konstrukci s připevněnou 3D kamerou, na kterou byla umístěná také již zmíněná 2D kamera, jak je znázorněno na obrázku 8.12. Ta nebyla v rámci poskytnutého robotického pracoviště aktivně využívána, a tudíž nebyla zanesena ani v počítačovém modelu. Nebyla také známá žádná referenční transformace mezi ní a rámcem `word` představujícím souřadný systém scény robotického pracoviště. Z tohoto důvodu byla umístěna právě na konstrukci s 3D kamerou, od které jsem měl poskytnutou referenční transformaci mezi ní a scénou. Pomocí těchto informací byla následně ověřena přesnost výsledné kalibrace.

Obrázek 8.12: Umístění 2D kamery ve scéně pracoviště

Výsledkem procesu detekce `ArUco` markeru, který je znázorněn na obrázku 8.13, je seznam jeho čtyř rohů, které jsou vráceny po směru hodinových ručiček od levého horního rohu zvýrazněného čtverečkem. Společně s rohy je odhadnuta také translace a rotace markeru. Tyto údaje následně slouží pro výpočet polohy kamery v prostoru. Za tímto účelem je nutné dodat k detekovaným rohům jejich reálné souřadnice. Tyto souřadnice se opět vezmou z kalibračního objektu, podobně jako u mechanické kalibrace. Tentokrát se ovšem všechny kalibrační body posunou v ose z na hodnotu deseti centimetrů, aby odpovídaly souřadnicím rohů markeru.



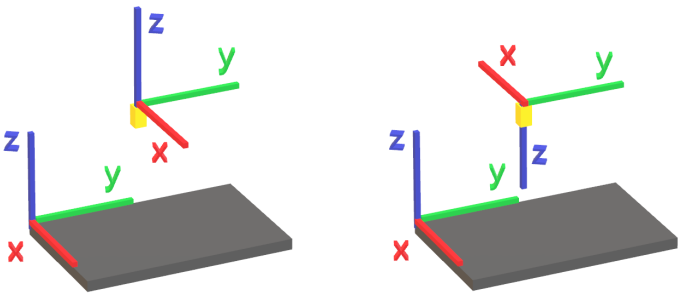
Obrázek 8.13: Detekce `ArUco` markeru

Výsledkem kalibrace 2D kamery je transformace znázorněná v tabulce 8.12. Jedná se o transformaci mezi rámcem kamery a rámcem `table` představujícím nový počátek souřadného systému scény. Jak již bylo zmíněno, k této 2D kameře nebyla poskytnuta žádná referenční transformace pro ověření přesnosti výpočtu.

Translace	Hodnota [cm]
x	44,6509731633
y	-18,3603031271
z	77,091945519
Rotace	Hodnota [○]
x	-178,6059689
y	-0,1540016
z	-178,9261405

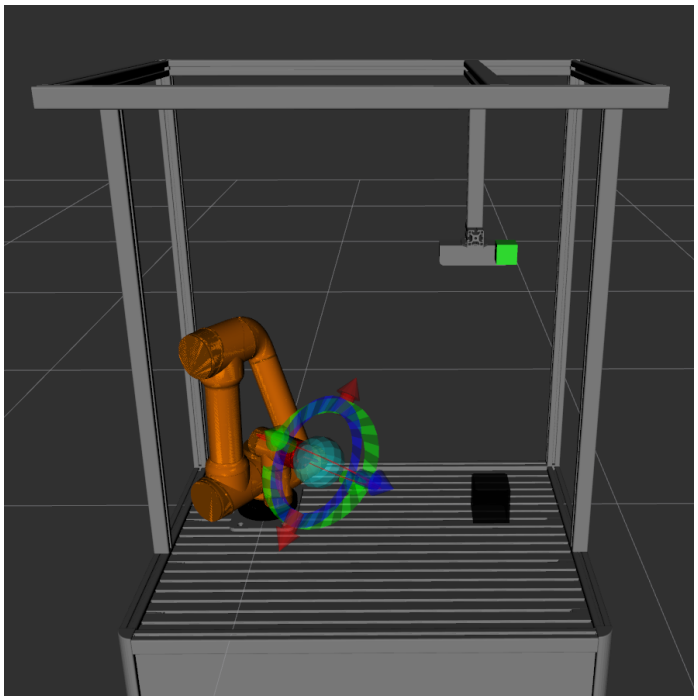
Tabulka 8.12: Výsledná transformace kalibrace 2D kamery

Z této transformace je patrné, že abychom se dostali do počátku souřadného systému rámce `table`, musíme provést translaci v kladném směru na osách x a z, v záporném pak na ose y. Dále je zřejmé, že je nutné provést rotaci o téměř 180 stupňů kolem os x a y. Pro zjednodušení si tuto rotaci vizualizujeme na obrázku 8.14, který znázorňuje situaci před a po aplikování rotace. Z tohoto obrázku je nyní patrné, že předem zmíněné kladné a záporné translace v jednotlivých osách vizuálně odpovídají novému uspořádání os souřadného systému kamery. Translace byla ověřena také pomocí měření v reálné scéně robotického pracoviště.



Obrázek 8.14: Vizualizace rotací z výsledné transformace 2D kamery

Za účelem ověření vypočítané transformace byl vytvořen kolizní objekt reprezentující kameru, který byl následně umístěn do počátku souřadného systému kamery. Tyto souřadnice byly získány právě pomocí vypočítané transformace z tabulky 8.12, kdy byl počátek souřadného systému kamery převeden na souřadnice rámce `table`. Z obrázku 8.15 lze ověřit, že se kamera skutečně nachází na předpokládaném místě, které je zachyceno na fotografii z obrázku 8.12.



Obrázek 8.15: Ověření výsledné kalibrace 2D kamery

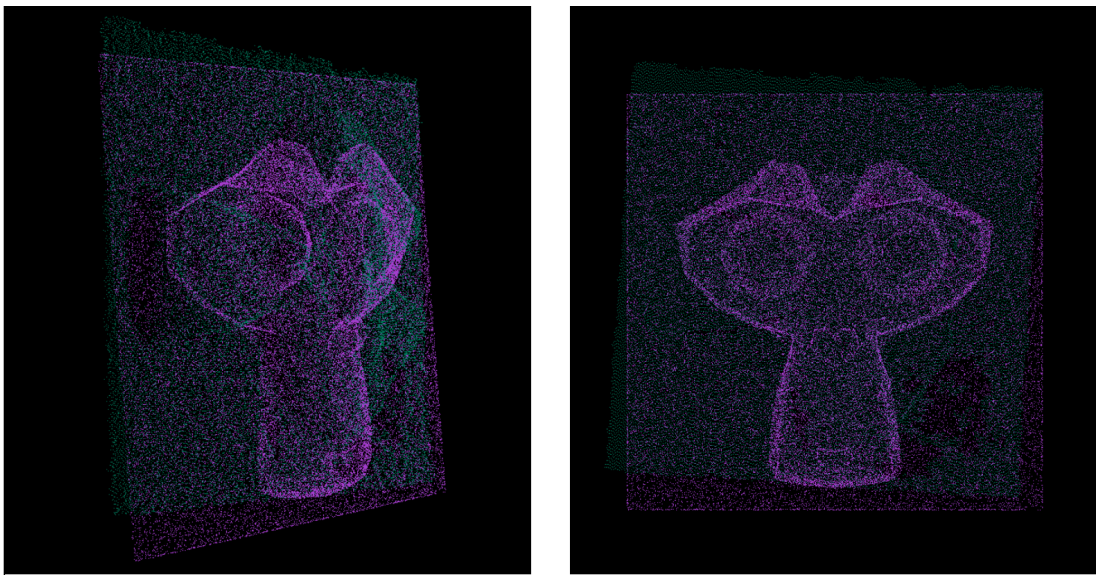
Kalibrace 3D kamery

Poslední experiment této kapitoly se týká ověření funkčnosti kalibrace 3D kamery. V tomto případě nahradíme `ArUco` marker použitý v předchozím experimentu za kalibrační objekt z kapitoly 7.4. Tento objekt opět umístíme na kalibrační krychli pro mechanickou kalibraci. Zároveň tento kalibrační objekt vložíme do počítačového `URDF` modelu, kde mu nastavíme vlastní rámec reprezentující jeho souřadný systém. Pro kontrolu přesnosti této kalibrace použiji referenční transformaci mezi souřadným systémem 3D kamery a robotického manipulátoru, kterou mi poskytla zadavatelská firma. Tato transformace je znázorněna v tabulce 8.13.

Translace	Hodnota [cm]
x	-2,61717
y	-61,9076
z	73,8186
Rotace	Hodnota [○]
x	-3,1415909
y	0,0000103
z	-3,1375951

Tabulka 8.13: Transformace počátku souřadného systému 3D kamery

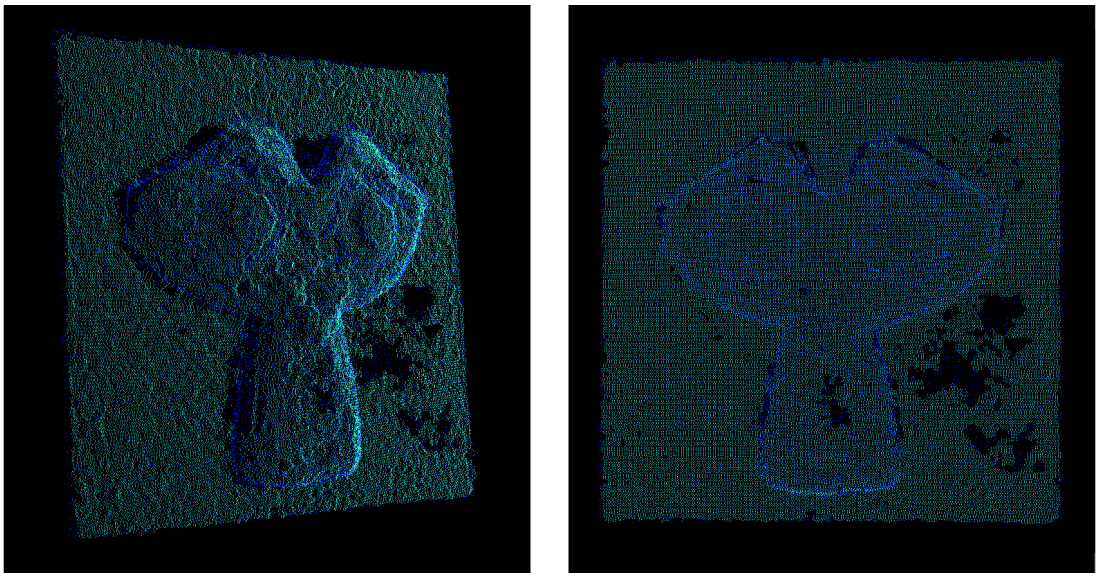
Během experimentů několikrát došlo k neúspěšnému zarovnání mračna bodů tak, jak je znázorněno na obrázku 8.16. K těmto případům docházelo náhodně, aniž by se mezi jednotlivým měřením jakkoliv manipulovalo s kalibračním objektem. Podrobným pozorováním jsem zjistil, že k těmto případům dochází na základě špatného výsledku algoritmu `Template Alignment`, který vracel neplatné transformace pro zarovnávaný objekt. S těmito výsledky si již algoritmus `ICP` nedokázal poradit. Tento nežádoucí jev byl částečně eliminován úpravou parametrů pro oba algoritmy. Jednalo se především o změnu velikosti rádiusu pro výpočet normál a rysů (tzv. *feature*), vzhledem k rozlišení a hustotě jednotlivých mračen bodů. Na základě výsledků měření byla upravena také minimální vzorkovací vzdálenost pro algoritmus `Template Alignment`. Čím vyšší hodnota, tím rychleji algoritmus funguje, ovšem ztrácí se tím přesnost zarovnání. Nastavil jsem tedy nižší hodnotu ve snaze dostat přesnější zarovnání mračen bodů na úkor výpočetního času. Dále jsem zjistil poměrně vysokou citlivost kamery na okolní osvětlení, kdy přímé světlo dokázalo způsobit ve výsledném snímku prázdné oblasti bez bodů, nebo výrazně zašumělo výsledné mračno bodů.



Obrázek 8.16: Výsledek neúspěšného zarovnání mračna bodů

Jednotlivými úpravami popsanými v předchozím odstavci jsem docílil vizuálně stabilních výsledků zarovnání, jak je znázorněno na obrázku 8.17. Metodu v tomto stavu jsem již použil pro experimentální kalibraci 3D kamery. Výsledky jednotlivých měření jsou znázorněné

v tabulce 8.14. Jelikož referenční transformace se vztahuje k rámci souřadného systému robotického ramene, vypočítal jsem pomocí získané transformace nulové souřadnice počátku souřadného systému kamery relativně vůči rámci `world`. Tyto hodnoty jsou znázorněny v tabulce 8.16 a slouží pro porovnání přesnosti kalibrace.



Obrázek 8.17: Výsledek úspěšného zarovnání mračna bodů

Měření č.1	
Translace	Hodnota [cm]
x	-1,21078722978
y	3,34396882678
z	63,4821892512
Rotace	Hodnota [◦]
x	3,1167278
y	0,006136
z	-2,6833274

Měření č.2	
Translace	Hodnota [cm]
x	-3,40358765331
y	4,26393824772
z	63,5847760415
Rotace	Hodnota [◦]
x	3,1198376
y	0,0120088
z	3,0518079

Měření č.3	
Translace	Hodnota [cm]
x	-2,23183547762
y	2,56981485737
z	63,364889374
Rotace	Hodnota [◦]
x	-3,1106934
y	0,0157247
z	-2,8599648

Měření č.4	
Translace	Hodnota [cm]
x	-4,7654333048
y	4,69315712053
z	63,4659829673
Rotace	Hodnota [◦]
x	3,1283154
y	0,0104145
z	-3,1031236

Tabulka 8.14: Výsledné transformace kalibrace 3D kamery vzhledem ke kalibru

Translace	Hodnota [cm]
x	-2,902910916
y	3,7177197631
z	63,474459408
Rotace	Hodnota [∘]
x	1,56354685
y	0,011071
z	-1,398651975

Tabulka 8.15: Průměrná hodnota výsledné transformace

Z výsledků jednotlivých kalibrací je patrný poměrně vysoký rozptyl hodnot translace na osách x a y, zatím co translace na ose z zůstává poměrně stabilní. K tomuto rozptylu docházelo přesto, že s kalibračním objektem po dobu experimentu nebylo žádným způsobem manipulováno. Vizualizace výsledků jednotlivých zarovnání mračen bodů vypadaly velice stabilně, podobně jako je znázorněno na obrázku 8.17. Přesto i z pohledu vypočítaných rotací se tato metoda jeví jako poněkud nestabilní. Z tohoto důvodu je vhodné pro lepší výsledek kalibrace provést více jednotlivých měření a jejich výsledek následně zprůměrovat.

Měření č.1	
Translace	Hodnota [cm]
x	-3,21053
y	-60,6588
z	73,4822
Rotace	Hodnota [∘]
x	-3,1276182
y	-0,0087044
z	0,4566579

Měření č.2	
Translace	Hodnota [cm]
x	-1,40162
y	-61,73592
z	73,5848
Rotace	Hodnota [∘]
x	-3,1310706
y	-0,0115311
z	-0,0914474

Měření č.3	
Translace	Hodnota [cm]
x	-2,23354
y	-59,4314
z	73,474475
Rotace	Hodnota [∘]
x	-3,12997635
y	-0,0072873
z	0,1705861

Měření č.4	
Translace	Hodnota [cm]
x	-0,76415
y	-61,3057
z	73,466
Rotace	Hodnota [∘]
x	-3,1307202
y	-0,0104789
z	0,0368642

Tabulka 8.16: Výsledné transformace kalibrace 3D kamery vzhledem k rámci `world`

Jak můžeme vidět v tabulce 8.18, transformace 3D kamery pomocí metody využívající zarovnání mračen bodů je oproti referenční transformaci nepřesná o něco málo přes půl centimetru v případě translace na ose x. Na ose y tato nepřesnost narostla na asi 1 centimetr. Nejlepší odhad má tato metoda pro výškové umístění objektu, což lze vidět z průměrné hodnoty nepřesnosti translace na ose z, která činí přibližně 3 milimetry. Jak již bylo zmíněno, tato metoda je poněkud náchylná na vlivy okolního prostředí a na zašumění výsledných dat. Určitá nepřesnost mohla být do měření zanesena také ručním zaměřením

počátku souřadného systému rámce `table` a relativně vůči němu zaměřením a umístěním kalibračního objektu pod kameru.

Translace	Hodnota [cm]
x	-1,90246
y	-60,782955
z	73,50186875
Rotace	Hodnota [◦]
x	1,56354685
y	0,011071
z	-1,398651975

Tabulka 8.17: Průměrná hodnota výsledné transformace vzhledem k rámci `world`

Translace	Hodnota [cm]
x	-0,71471
y	-1,124645
z	0,31673125
Rotace	Hodnota [◦]
x	-4,70513775
y	-0,0110607
z	-1,738943125

Tabulka 8.18: Rozdíl mezi referenční transformací a průměrnou hodnotou měření

8.3 Náměty k rozšíření

Tato kapitola se věnuje možnému budoucímu vývoji a vylepšení této aplikace. Jelikož se aplikace zaměřuje celkově na tři různé typy kalibrací a s nimi spojenou problematiku, je možné se do budoucna soustředit pouze na jednu z oblastí a navrhnout pro ni nové efektivnější metody. Další možnou oblastí budoucího vývoje je vylepšení integrace aplikace do systému `ARCOR`. Jednotlivé návrhy a jejich potenciální přínosy budou postupně popsány v následujících kapitolách.

Použití kalibračního objektu s protikusem

Jedním z možných vylepšení mechanické kalibrace ramene robotického manipulátoru by mohl být návrh kalibračního objektu společně s protikusem umístěným na koncovém efektoru robotického ramene. Tento protikus by přesně zapadal do otvorů v kalibračním objektu, čímž by se minimalizovala nepřesnost měření. Tyto protikusy by se daly použít také v případě kalibrace pomocí 2D markerů. V takovém případě by například mohly přesně odpovídat tvarem značce určující marker na pracovní ploše. V úvahu by připadala i varianta, kdy by se z 2D markerů staly jakési kalibrační otvory, případně zarážky, do kterých by se přiložil protikus upevněný na rameni. V takovém případě by se mohlo jednat o rychlou a pohodlnou metodu pro zkalibrování robotického manipulátoru, za předpokladu že máme pevně definovanou scénu robotického pracoviště, v rámci které se robot často mění nebo přesouvá.

Vizualizace a zaměření kalibračních bodů systémem ARCOR

V případě prohloubení integrace této aplikace do systému `ARCOR` lze využít především poskytovaného grafického uživatelského rozhraní. Obě zmíněné metody pro mechanickou kalibraci ramene robotického manipulátoru jsou omezeny skutečností, že jednotlivé kalibrační body je nutné zaměřovat v přesně daném pořadí, v jakém jsou definovány v rámci aplikace. Tato informace je ovšem běžnému uživateli skryta a musí na ni být nějakým způsobem upozorněn. Jedním možným způsobem je tuto informaci zanést přímo na kalibrační objekt. Daleko lepší a efektivnější způsob by ale mohlo být právě využití grafického uživatelského rozhraní a skutečnosti, že systé `ARCOR` umí spolupracovat s projektorem. V takovém případě by bylo možné jistým způsobem zvýraznit bod, který se aktuálně zaměřuje. Ještě efektivnější by tento postup byl v případě použití 2D markerů.

Dále by bylo vhodné umožnit uživateli potvrdit zaměření kalibračního bodu přímo skrze grafické uživatelské rozhraní. Aktuálně je nutné toto zaměření potvrdit na klávesnici. V rámci systému `ARCOR` by k tomuto účelu mohla sloužit buď dotyková plocha stolu, nebo případně virtuální realita.

Zaměření kolizního objektu pomocí 3D kamery

Podobně jako je u metody mechanické kalibrace pro zaměření kolizního objektu využíváno robotické rameno, mohla by pro zaměření dalších objektů ve scéně sloužit také 3D kamera, případně zařízení typu LIDAR. Tato kamera by mohla být upevněna na rameno robotického manipulátoru, čímž by byla známa rotace mezi zaměřeným objektem a roboitckým ramenem, která by se následně využila pro vložení tohoto objektu do virtuální scény. Tato metoda by vyžadovala nastudování a implementaci vhodných metod pro detekci objektů v prostoru z výsledných mračen bodů získaných snímáním scény kamerou. Touto metodou by bylo možné poměrně jednoduše a přesně zaměřovat nové objekty a tím následně vytvářet virtuální obraz této reálné scény pro robotické rameno.

Výpočet transformace kalibračního objektu pomocí neuronové sítě

Neuronové sítě jsou poslední dobou stále více aktuálním tématem a nachází uplatnění v mnoha odvětvích. V rámci této aplikace by bylo možné použít neuronové sítě pro odhad pozice kalibračního objektu v prostoru. Neuronová síť by se naučila z jednotlivých vzorků veškeré možné kombinace translací a rotací daného objektu. Na základě transformace zjištěné neuronovou sítí a známé transformace mezi souřadným systémem scény a kalibračním objektem lze následně zjistit pozici kamery. Tento přístup by bylo možné aplikovat jak na fotografie pořízené 2D kamerou, tak na mračna bodů získané snímáním 3D kamerou. Takto naučená neuronová síť by poté mohla nahradit jak metodu pro kalibraci 2D kamery skrze `ArUco` marker, tak metodu pro kalibraci 3D kamery využívající algoritmy `Template Alignment` a `ICP`.

Kapitola 9

Závěr

Cílem této práce bylo navrhnout aplikaci určenou pro robotický systém `ROS` sloužící pro vzájemnou kalibraci robotického manipulátoru, 2D nebo 3D kamery a scény robotického pracoviště. K tomuto účelu mi bylo od zadavatelské firmy poskytnuto vybavení ve formě kolaborativního robotického ramene, průmyslové 2D a 3D kamery, společně s konstrukcí představující reálnou scénu robotického pracoviště, včetně jejího počítačového modelu.

Ve své práci jsem začal studiem problematiky výpočtu transformací mezi jednotlivými souřadnými systémy. K tomuto účelu mi posloužily různé internetové zdroje a odborné publikované články, ze kterých jsem získal základní přehled o problematice. Na základě těchto poznatků jsem napsal první kapitolu této práce, která se zmíněné problematice věnuje.

V dalším kroku jsem se zaměřil více na robotický systém a dostupný software, jelikož mým úkolem bylo také nastudovat problematiku týkající se plánování pohybu robotického manipulátoru. Především jsem se ale věnoval základům práce se systémem `ROS` a s ním souvisejícími věcmi, jako jsou framework `MoveIt!` a různé dostupné knihovny pro práci s transformacemi a daty z kamer. Jelikož práce na aplikaci zahrnovala také integraci do systému `ARCOR`, bylo nutné nastudovat také funkcionalitu a architekturu tohoto systému.

Dále jsem analyzoval jednotlivé vhodné metody pro realizaci výše zmíněných kalibrací. Na základě této analýzy byla navrhnuta aplikace a její jednotlivé kalibrační metody. V případě mechanické kalibrace robotického ramene byly navrženy také metody pro modifikaci virtuální scény tohoto pracoviště. Kromě samotných metod jsem také navrhnul a následně vyrobil dva kalibrační objekty, společně s kalibračním koncovým efektorem, určené pro mechanickou kalibraci a kalibraci 3D kamery.

Následovala samotná implementace této aplikace. V práci je popsána její celková struktura, včetně jednotlivých balíčků, ze kterých se skládá, společně s obsaženými kalibračními metodami. Jsou zde také popsané průběhy procesů každé kalibrační metody, jejich účel, funkcionalita, případně jednotlivé restrikce. Během vývoje jsem tyto metody neustále testoval na poskytnutém robotickém pracovišti. Výsledná aplikace byla následně integrována do systému `ARCOR` skrze její aplikační rozhraní.

Na závěr práce jsem za účelem ověření funkcionality jednotlivých metod navrhnul a popsal sadu experimentů. U každého experimentu jsou diskutovány jeho výsledky. Většina je doprovázena obrázky a fotografiemi pro lepší představu o podobě a výsledku experimentu. Aplikaci je možné do budoucna rozšířit o další metody, případně lze vylepšit integraci do systému `ARCOR`. Na toto téma jsem v práci sepsal samostatnou podkapitolu, kde jsou popsány mé konkrétní návrhy.

Literatura

[1] OpenCV: Detection of ArUco Markers. [Online; navštíveno 2.12.2018].
URL https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html

[2] Bradski, G.: ICRA 2010 OpenCV Tutorial. 2010, [Online; navštíveno 30.10.2018].
URL http://opensource-robotics.tokyo.jp/ros.org/wiki.ros.org/attachments/Events(2f)ICRA2010Tutorial/ICRA_2010_OpenCV_Tutorial.pdf

[3] Cashbaugh, J.; Kitts, C.: Automatic Calculation of a Transformation Matrix Between Two Frames. *IEEE Access*, ročník 6, 2018: s. 9614–9622.

[4] International Organization for Standardization. Technical Committee Automation systems and integration. Subcommittee Robots and robotic devices: *ISO 8373: Robots and Robotic Devices - Vocabulary*. ISO, 2012.

[5] Materna, Z.; Kapinus, M.; Beran, V.; aj.: Interactive Spatial Augmented Reality in Collaborative Robot Programming: User Experience Evaluation. In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, IEEE, 2018, s. 80–87.

[6] About MoveIt! [Online; navštíveno 10.11.2018].
URL https://moveit.ros.org/about/

[7] MoveIt! Concepts - Collision checking. [Online; navštíveno 13.11.2018].
URL https://moveit.ros.org/documentation/concepts/#collision-checking

[8] MoveIt! Concepts - Motion planning. [Online; navštíveno 13.11.2018].
URL https://moveit.ros.org/documentation/concepts/#motion-planning

[9] MoveIt! Concepts - OMPL. [Online; navštíveno 16.11.2018].
URL https://moveit.ros.org/documentation/concepts/#ompl

[10] MoveIt! Concepts - Robot interface. [Online; navštíveno 10.11.2018].
URL https://moveit.ros.org/documentation/concepts/#robot-interface

[11] Camera calibration With OpenCV. [Online; navštíveno 26.10.2018].
URL https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

[12] Orság, F.: *Robotika - studijní opora*. FIT VUT v Brně, 2006.

[13] Point Cloud Library (PCL). [Online; navštíveno 26.11.2018].
URL http://pointclouds.org/about/

[14] Aligning object templates to a point cloud. [Online; navštíveno 15.12.2018].
URL
http://pointclouds.org/documentation/tutorials/template_alignment.php

[15] Quigley, M.; Conley, K.; Gerkey, B.; aj.: ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, ročník 3, Kobe, Japan, 2009, str. 5.

[16] About ROS. [Online; navštíveno 5.11.2018].
URL http://www.ros.org/about-ros/

[17] ROS-Industrial. [Online; navštíveno 7.11.2018].
URL https://rosindustrial.org/

[18] URDF. [Online; navštíveno 5.11.2018].
URL http://wiki.ros.org/urdf

[19] ROS Actions. [Online; navštíveno 6.11.2018].
URL http://wiki.ros.org/actionlib

[20] ROS Messages. [Online; navštíveno 5.11.2018].
URL http://wiki.ros.org/Messages

[21] ROS Nodes. [Online; navštíveno 10.11.2018].
URL http://wiki.ros.org/Nodes

[22] ROS Services. [Online; navštíveno 6.11.2018].
URL http://wiki.ros.org/Services

[23] ROS Topics. [Online; navštíveno 6.11.2018].
URL http://wiki.ros.org/Topics

[24] Rusinkiewicz, S.; Levoy, M.: Efficient variants of the ICP algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, IEEE, 2001, s. 145–152.

[25] Sucan, I. A.; Moll, M.; Kavraki, L. E.: The open motion planning library. *IEEE Robotics & Automation Magazine*, ročník 19, č. 4, 2012: s. 72–82.

[26] Transform Library. [Online; navštíveno 21.11.2018].
URL http://wiki.ros.org/tf

Příloha A

Struktura zpráv

Pro jednotlivé úkony aplikace, jako například publikace výsledné transformace mezi souřadnými systémy, nebo definice nového kolizního objektu a jeho umístění na konkrétní pozici do scény robotického pracoviště, je využíván mechanismus zpráv systému `ROS`. V následujících kapitolách je popsána struktura zpráv, které jsou v aplikaci využívány.

geometry_msgs/TransformStamped

Výsledná transformace je publikována pomocí mechanismu zpráv systému `ROS`. V tabulce A.1 je vidět struktura zasílané zprávy ve formátu `tf::TransformStamped` knihovny `TF`. Jedná se o zprávu obsahující v hlavičce (`header`) časové razítko, společně se jménem rámce souřadného systému. Kolonka pro rámec potomka (`child_frame_id`) obsahuje název cílového rámce souřadného systému, pro který je transformace platná.

Typ	Název
std_msgs/Header	header
string	child_frame_id
geometry_msgs/Transform	transform

Tabulka A.1: Struktura zprávy `geometry_msgs/TransformStamped`

geometry_msgs/Transform

Samotná transformace je obsažena v kolonce `transform`, která je součástí zprávy z předchozí kapitoly. Uvnitř ní se nachází složky `translation` a `rotation`, které popisují translaci, respektive rotaci pro transformaci mezi souřadnými systémy.

Typ	Název
geometry_msgs/Vector3	translation
geometry_msgs/Quaternion	rotation

Tabulka A.2: Struktura zprávy `geometry_msgs/Transform`

moveit_msgs/CollisionObject

Kolizní objekt je v robotické scéně definován zprávou `moveit_msgs/CollisionObjec`. Jak již bylo zmíněno v předchozí kapitole, kolizní objekt lze definovat pomocí geometrických primitiv, nebo pomocí sítě trojúhelníků mesh. Strukturu prvního jmenovaného znázorňuje tabulka A.3. Zpráva obsahuje hlavičku nesoucí časové razítko a jméno rámce pro který je daný kolizní objekt platný. Dále obsahuje textové `id` kolizního objektu a typ geometrického primitiva, kterým je tvořen. Popis struktury zprávy znázorňující toto primitivum je obsažen v tabulce A.6. Zpráva obsahuje také pozici, na kterou je kolizní objekt ve scéně umístěn. Struktura zprávy pro tuto pozici je popsána v tabulce A.8.

Typ	Název
std_msgs/Header	header
string	id
object_recognition_msgs/ObjectType	type
shape_msgs/SolidPrimitive[ ]	primitives
geometry_msgs/Pose[ ]	primitive_poses
byte	operation

Tabulka A.3: Struktura zprávy `moveit_msgs/CollisionObject` pro primitiva

Tabulka A.4 znázorňuje strukturu zprávy pro kolizní objekt tvořený sítí trojúhelníků. Od předchozí zmíněné zprávy se liší pouze v definici geometrického objektu použitého pro kolizní objekt. Struktura zprávy pro síť trojúhelníků je popsána v tabulce A.7.

Typ	Název
std_msgs/Header	header
string	id
object_recognition_msgs/ObjectType	type
shape_msgs/Mesh[ ]	meshes
geometry_msgs/Pose[ ]	mesh_poses
byte	operation

Tabulka A.4: Struktura zprávy `moveit_msgs/CollisionObject` pro mesh

shape_msgs/SolidPrimitive

Tato zpráva, jejíž struktura je popsána v tabulce A.6, popisuje geometrické primitivum, které je použito v rámci zprávy definující kolizní objekt z tabulky A.3. Její políčko `type` definuje právě onen typ použitého primitiva. Hodnoty kterých může nabývat jsou znázorněny v tabulce A.5:

uint8	Typ
1	kvádr
2	koule
3	válec
4	kužel

Tabulka A.5: Typy geometrických primitiv

Políčko `dimensions` poté obsahuje rozměry tělesa v jeho jednotlivých osách. V případě koule nebo kužele obsahuje také hodnotu poloměru.

Typ	Název
uint8	type
float64[ ]	dimensions

Tabulka A.6: Struktura zprávy `shape_msgs/SolidPrimitive`

shape_msgs/Mesh

Narozdíl od předchozí zprávy, která definovala geometrické primitivum použité pro definici kolizního objektu, následující zpráva za tímto účelem definuje síť trojúhelníků. Strukturu této zprávy můžeme vidět v tabulce A.7. Ta obsahuje políčko `triangles`, ve kterém jsou uloženy pole obsahující tři indexy, jejichž hodnoty definují body tvořící trojúhelník. Tyto body se nacházejí na odpovídajících indexech v políčku `vertices`, kde jsou uložené jejich trojrozměrné souřadnice.

Typ	Název
MeshTriangle[ ]	triangles
geometry_msgs/Point[ ]	vertices

Tabulka A.7: Struktura zprávy `shape_msgs/Mesh`

geometry_msgs/Pose

Poslední zprávou využívanou v rámci této aplikace je zpráva `geometry_msgs/Pose`, která definuje pozici a orientaci v souřadném systému některého z rámců. Její struktura je znázorněna v tabulce A.8. Tato zpráva může sloužit jednak pro navádění ramene robotického manipulátoru, ale také například pro umísťování kolizních objektů do scény systému `ROS`. Obsahuje políčko `position`, v rámci kterého je uložená trojrozměrná souřadnice. V druhém políčku `orientation` je naopak uložena orientace v prostoru.

Typ	Název
geometry_msgs/Point	position
geometry_msgs/Quaternion	orientation

Tabulka A.8: Struktura zprávy `geometry_msgs/Pose`