TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Programování průmyslových robotů FANUC

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Autor práce: **Jaroslav Zima**
Vedoucí práce: Ing. Daniel Kajzr

Liberec 2020

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Programming of FANUC industrial robots

Bachelor thesis

Study programme: B2612 – Electrical engineering and informatics
Study branch: 2612R011 – Electronic Information and Control systems

Author: **Jaroslav Zima**
Supervisor: Ing. Daniel Kajzr

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Programování průmyslových robotů FANUC

Jméno a příjmení: **Jaroslav Zima**
Osobní číslo: M16000187
Studijní program: B2612 Elektrotechnika a informatika
Studijní obor: Elektronické informační a řídicí systémy
Zadávající katedra: Ústav mechatroniky a technické informatiky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Seznamte se s robotickým systémem FANUC instalovaným ve výukové buňce.
2. Navrhněte demo úlohu s podporou Roboguide offline programování.
3. Návrh validujte v reálné buňce.
4. Zpracujte stručný návod s postupem, jak analogické úlohy vytvořit.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30–40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] GREPL, Robert. Kinematika a dynamika mechatronických systémů. Brno: Akademické nakladatelství CERM, 2007. ISBN 978-80-214-3530-8.

[2] SICILIANO, Bruno a Oussama KHATIB. Springer handbook of robotics. Berlin: Springer, 2008. ISBN 978-3-540-23957-4.

[3] KARGER, Adolf a Marie KARGEROVÁ. Základy robotiky a prostorové kinematiky. Praha: ČVUT, 2000. ISBN 80-01-02183-1.

[4] FANUC Robotics – Accompanying Training Manual, Roboguide V6.40 Rev. B.

Vedoucí práce:	Ing. Daniel Kajzr Ústav mechatroniky a technické informatiky
Datum zadání práce:	10. října 2019
Předpokládaný termín odevzdání:	18. května 2020

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2019

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

1. června 2020 Jaroslav Zima

Abstrakt

Tato práce se zabývá vytvořením výukového materiálu, využitelného jak pro potřeby výuky, tak pro samostudium robotiky a programování s hlavním zaměřením na roboty značky FANUC. V práci je popsána problematika robotů a jejich programování. Praktická část je pak zaměřena na seznámení s možností offline programování za podpory programovacího softwaru Roboguide. Jsou připraveny dvě výukové demo úlohy. Jednotlivé programy a potřebné data jsou k dispozici v příloze.

Klíčová slova:

FANUC, průmyslové roboty, programování průmyslových robotů, Roboguide, iRVision, offline programování.

Abstract

This Bachelor's thesis is about creating teaching material reusable for both teaching and self-study robotics and programming, with a major focus on robots brand FANUC. Bachelor's thesis describes the issue of robots and their programming. The practical part is then aimed at familiarising yourself with the possibility of offline programming supported by the Roboguide programming software. Two teaching demo jobs are set up. Individual programmes and the necessary data are available in the annex.

Keywords:

FANUC, industrial robots, industrial robots programming, Roboguide, iRVision, offline programming.

Poděkování

Rád bych poděkoval všem lidem, kteří přímo či nepřímo pomohli zpracovat tuto práci. Jmenovitě vedoucímu mé Bakalářské práce Ing. Danielu Kajzrovi za odborné vedení, pomoc a rady při řešení problémů. Dále pak doc. Ing. Josefu Černohorskému, Ph.D. za věcné rady při práci na praktické části. Poslední, komu bych chtěl poděkovat je Ing. Jan Koprnický, Ph.D. za cenné rady ohledně problémů při práci se sázecím programem LaTeX.

Obsah

Seznam obrázků

Seznam zkratek

TUL	Technická univerzita v Liberci
FM	Fakulta mechatroniky
3D	Trojdimenzionální
PR	Průmyslový robot
GPM	Geometric Pattern Matching
TTT	Kinematická struktura translace, translace, tanslace
RTT	Kinematická struktura rotace, translace, translace
RRT	Kinematická struktura rotace, rotace, translace
RRR	Kinematická struktura rotace, rotace, rotace
SCARA	Selective compliance assembly robot arm
NC	Numerical control
CNC	Computer numerical kontrol
PC	Personal computer

1 Úvod

Má bakalářská práce se zabývá programováním průmyslových robotů FANUC. Cílem je vytvořit demo úlohu s podporou Roboguide offline programování. Výsledná úloha musí být schopna úspěšného provozu ve výukových buňkách v univerzitní laboratoři. Samotná práce pak může sloužit jako výuková pomůcka pro studenty, nebo pro samostudium. Práce je rozdělaná do dvou částí, teoretické a praktické.

V teoretické části se zabývám seznámením se světem robotů, jejich původem, popisem jejich konstrukce a vlastností. V závěru první části krátce shrnu pár informací o firmě FANUC a jejich produktech z odvětví robotiky. V druhé části, tedy té praktické je obsažen popis demonstračních úloh a příprav, které jim předcházely.

Návrhem a realizací první úlohy, která je zaměřena na offline programování s podporou softwaru Roboguide jsem obsáhl zadání. A dále jsem se rozhodl zvýšit svůj přínos bakalářskou prací a vytvořit druhou demo úlohu. Ta se zabývá návrhem a realizací programu s využitím nově nainstalované kamery na zápěstí robota v druhé výukové buňce.

Při tvorbě první úlohy jsem se na začátek seznámil s pracovištěm. Znalosti jsem následně využil při tvorbě virtuálního pracoviště. Vymodeloval jsem si vlastní 3D model buňky, který jsem později pro pracoviště využil. Navrhl a offline naprogramoval demo úlohu, kterou jsem na závěr otestoval ve výukové buňce. Výsledkem první úlohy je komplexní a stručný návod pro práci s robotem na pracovišti a práci se softwarem Roboguide.

Druhou úlohu jsem navrhl s využitím nově nainstalované kamery na zápěstí robota, provozovanou pomocí systému iRVision. iRVision je rozšířeni o takzvané strojní vidění, které umožňuje rozpoznávání objektu a určování jeho polohy, což ve své úloze také využívám. Pro tuto úlohu jsem výukovou buňku obohatil o poziční pole pro hrací kostky. Pole jsem navrhl dle potřebných rozměrů a sestrojil tak, aby vyhovovalo navržené úloze. V průběhu návrhu a testování možností kamery jsem zjistil, že hrací kostka, se kterou robot pracuje, nemá přijatelné optické vlastnosti. Na základě požadavků jsem navrhl novou kostku. V rámci projektu Bastlírna mají studenti přístup do univerzitní dílny, kde jsou k dispozici i 3D tiskárny. Využil jsem možnosti a kostku si zde dle návrhu nechal zhotovit. Úlohu jsem zrealizoval a otestoval ve výukové buňce. Výsledkem je opět popis návrhu, realizace a shrnutí výsledků. Buňky s roboty FANUC jsou na univerzitě instalovány relativně nedávno a tak není vytvořeno příliš úloh, na kterých by si studenti mohli vyzkoušet pracovat s robotem a tato práce by jim měla pomoci.

2 Průmyslové roboty

Nepostradatelní pomocníci v průmyslové výrobě a nejen zde. Jejich myšlenka sahá do dob Leonarda da Vinciho, první úspěšná realizace však přišla až v polovině dvacátého století.

Pod pojmem průmyslový robot si každý představí něco jiného. Většina si však představí žluté nebo oranžové robotické rameno, které se dnes běžně vidí v průmyslové výrobě. To je však jen jeden případ robota. V praxi lze průmyslovým robotem označit stroj ulehčující, či vykonávající práci za člověka.[4][5][11]

2.1 Původ slova robot a jeho význam v technice

Dnes je všeobecně známo, že slovo robot má původ zde u nás. Bylo použito v divadelní hře R.U.R Karla Čapka. Nicméně spousta lidí tento fakt nezná úplně přesně. Karel Čapek sice slovo využil ve své hře, autorem je však spisovatelův bratr Josef Čapek. Mylná informace je také ta, že Čapek ve své hře užívá životného tvaru vzoru páni, neboli ti roboti. V technické je pro robota využíván vzor hrad, tedy ty roboty.[4][5]

2.2 Asimovy zákony robotiky

Asimovy zákony, také označovány jako tři zákony robotiky, jsou pravidla, dle kterých by se měly roboty chovat. Byly sepsány Isaacem Asimovem, které uvedl v roce 1942 ve své povídce Hra na honěnou. V následujících letech se staly pilíři, kterými se řídili autoři nejrůznějších děl sci-fi žánrů zaměřených na roboty. Původní výklad zákonů je takovýto.

Robot nesmí zranit člověka ani ignorovat případné riziko jeho zranění. Podle druhého se robot musí řídit příkazy člověka. Podmínkou však je, že tím neporuší první zákon. V takovém případě je robot nucen příkaz neuposlechnout. Třetí zákon říká, že robot musí chránit sebe před poškození, pokud tím však neporuší první nebo druhý zákon.

V průběhu let pak došlo k nejrůznějším úpravám, obohacením jednotlivých zákonů i přidání nových. Nic ale nemění fakt, že jako ty původní stále platí tyto tři zákony.[4]

2.3 Robotizace

Průmyslová výroba od dob manufaktur prošla velkým vývojem. Asi nejzásadněji se v jejím vývoji zapsal Henry Ford se svým automobilem Ford Model T., díky němuž byly položeny základy pásové výroby a mechanizace, které vedly k dalšímu vývoji.

Následovala automatizace, která vyjadřuje samočinnost výrobního procesu a snaží se minimalizovat nutnost lidský zákroků. Vývoj nových technologií a nároky průmyslové výroby zapříčinily vzniku dalšího odvětví a tím byla robotizace.

Ta pomáhá odstraňovat nedostatky v činnostech, na které lidská síla nestačí. Samotným rozvojem robotizace se zabývá vědní disciplína označována jako robotika, zabývající se naukou o robotech, jejich koncepcí, návrhem a aplikacemi kde se využívají. Zahrnuje do sebe části z oblasti elektroniky, mechaniky a tvorby softwaru. Bez robotiky bychom nemohli provádět robotizace. Rozlišujeme tři úrovně robotiky.

Teoretická řeší roboty po stránkách teoretiky, jako je koncepce, umělé inteligence, senzorické systémy, simulace a podobně. Technická řeší výpočty spojené s roboty, jejich konstrukční řešení, provoz a údržbu. Aplikační řeší otázky aplikací a nasazení do provozu.[2][5]

2.4 Uplatnění robotů

V dnešní době je pro průmyslové roboty široké pole uplatnění. Místa, kde je robotizace více než nutná, jsou provozy s velkou mírou sériové výroby, dále výrobní procesy, pro které jsou již lidské zdroje vzhledem k nárokům na kvalitu a rychlost výroby nedokonalé.

Roboty jsou obzvlášť žádány tam, kde jsou podmínky pro lidi nepřípustné. Sem se řadí provozy s nadměrnými teplotami, znečištěným ovzduším a celkově prostory zvlášť nebezpečné. Nejčastějšími prostory pro aplikování průmyslových robotů jsou lakovny, svařovny, lisovny, kovárny, paletizační centra a montážní linky.[10][12]

2.5 Rozdělení z hlediska řízení

Jak už jsem uvedl, pod pojmem průmyslové roboty si většina lidí představí typicky oranžová nebo žlutá robotická ramena. Ty jsou však jen jeden z druhů průmyslových robotů.

Průmyslové roboty můžeme obecněji nazvat jako manipulátory. Ty však nemají ve světě přesně dané rozdělení. Nicméně je dnes hojně využíváno rozdělení do tří hlavních skupin, podle kterých je dělíme na ovládané člověkem, neměnným programem nebo proměnným programem.[2][10][12]

- Ruční manipulační zařízení uváděné do provozu za pomocí operátora, také označovaný jako teleoperátory, slouží k ulehčení opakovaných operací s těžkými břemeny. Jejich využití je často jednoúčelové, ale jsou i víceúčelové. Ty jsou však z podstaty věci často finančně nákladnější. [2][10][12]

- Robot s pevným programem je manipulátor, který vykonává činnost na základě pevně daného programu, bez přímého zásahu člověka. Samotný manipulátor lze jako mechanický celek v rámci jeho možností seřizovat. Nicméně, změna pracovního cyklu je velmi obtížná a často se neobejde bez zásahů v podobě výměn a úprav konstrukčních prvků.[2][10][12]

- Robot s proměnným programem a snadným přeučením pracovního cyklu. Je to volně programovatelný manipulátor, poskytující vysoký stupeň univerzálnosti. Tyto manipulátory jsou dnes na takové úrovni, že nemají problém vykonávat nejrůznější úlohy, učit se a na základě senzorických systémů upravovat, či předvídat svůj pracovní cyklus. Tyto roboty patří cenově k nejnákladnějším.[2][10][12]

2.6 Základní části průmyslových robotů

Základní části průmyslového robota jako celku jsou řídící jednotka, uživatelské rozhraní a sám průmyslový robot. Toto je základní sestava, v jaké může být průmyslový robot provozován. Dále lze provést rozšíření. Sem řadíme různé snímače, kamerové systémy a polohovací zařízení.[1][2][11]

- Řídící jednotka, vysílá příkazy pro ovládání pohonů a ostatních mechanizmů dle zadaného programu. Je schopna zpracovávat vstupní signály od senzorického systému, na základě kterých je schopna vykonávaný algoritmus měnit a upravovat. [1][2][11]

- Uživatelské rozhraní, označováno jako Teach-pendant, umožňuje přístup mezi člověkem a robotem. Lze skrze něj vizuálně kontrolovat parametry a ostatní důležité informace o zařízení, také dovoluje ručně řídit robota nebo programovat. Je pevnou součástí celku a nahrazuje tak jiné zařízení. Ve většině případů speciální počítač, často připojován skrze průmyslovou sběrnici, jakou je například ethernetový kabel. [1][2][11]

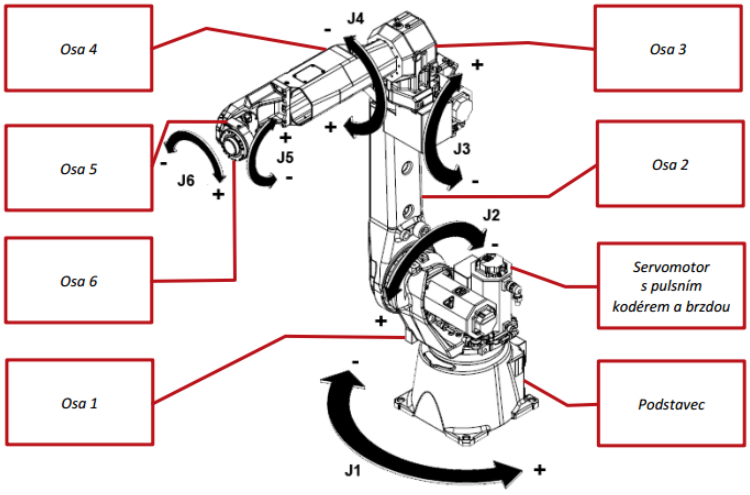
- Poslední součástí je robotické rameno neboli průmyslový robot, ten vykonává danou činnost, která je určena skrze řídící jednotku. Jednotka podává povely a robot je vykonává v definovaných osách nastavenou rychlostí. Pro zpětnou vazbu jsou pohony v každé ose spjaty se snímači polohy, díky nimž řídící jednotka dostává zpětně informaci o aktuální poloze. [1][2][11]

Obrázek 2.1: *Základní sestava průmyslového robota, upraveno z [1]*

3 Stavba PR

3.1 Konstrukční prvky robota

Robot, jakožto mechanický manipulátor, se skládá z několika součástí. Z podstavce označován také jako pata, za kterou je robot kotvený na své pracovní pozici, polohově řiditelných servomotorů s implementovanými převodovkami, absolutním enkodérem, brzdou, rameny a na konci zápěstím s montážní plochou, ke kterému se uchytí nástroj, obecně nazývaný jako efektor. [6]



Obrázek 3.1: *Konstrukční schéma šesti osého angulárního průmyslového robota, upraveno z [13]*

Schémata ostatních manipulátorů se mohou lišit, ale v základu se stále jedná o to, že robot je nějak ukotven. Má několik řiditelných os a aby mohl vykonávat svou činnost, musí být osazen koncovým nástrojem.

3.2 Kinematika robotů

Kinematika je odvětví mechaniky, zabývající se popisem pohybu. U robotů nám kinematika popisuje jeho pohyblivost a manipulační schopnosti. Zároveň charakterizuje jeho konstrukci. Pro vyjádření charakteru konstrukce se využívá termín kinematický řetězec, který nám reprezentuje množinu vzájemně spojených ramen skrze pohyblivé klouby neboli vazby.[2][3][11]

3.3 Kinematické řetězce

Rozdělujeme tři základní kinematické řetězce. Jedním z nich je otevřený kinematický řetězec, druhým je uzavřený řetězec a třetím speciální kombinace otevřeného s uzavřeným, který můžeme označit jako smíšený.[3][6][12]

- Otevřený kinematický řetězec, také označován jako sériová kinematika, má členy neboli ramena řazena v sérii za sebou v návaznosti na předchozí člen. Ramena jsou sériově spojeny skrze klouby. Tento řetězec začíná prvním kloubem u báze a posledním kloubem u zápěstí robota končí. Takovéto uspořádaní dnes využívá většina průmyslových robotů. Výhodou je flexibilnost robota a velké možnosti v rámci polohování koncového bodu. Nevýhodu je však nižší tuhost celé soustavy, odchylky v přesnosti polohování jsou řádově v desetinách milimetru. Výsledná odchylka na koncovém bodu robota je součtem jednotlivých odchylek předchozích členů.[3][6][12]

- Uzavřený kinematický řetězec, také označován jako paralelní kinematika, má své členy řazeny paralelně. Tedy tak, že koncový bod je veden několika paralelními rameny, která vychází z báze robota. Firma Fanuc, nabízí roboty se třemi, čtyřmi nebo šesti paralelními rameny. Výhodu je, že na rozdíl od otevřeného řetězce je zde vyšší tuhost a nepřesnosti jsou v jednotkách setin milimetrů. Nicméně, prostor poloh, kam je robot schopný se dostat, je dost omezený a proto jsou tyto roboty vhodné spíše na drobné operace. Jejich velikosti nedosahují takových rozměrů, jako roboty se sériovou kinematikou.[3][6][12]

- Smíšený kinematický řetězec je třetí zmiňovanou možností. Ve své podstatě se jedná o klasický otevřený řetězec, ale některá ramena jsou podpořena paralelními vzpěrami. Tato verze vyzvedává výhody sériové kinematiky a zároveň se snaží potlačovat její nedostatky. Jednoduše řečeno, snaží se co nejvíce zachovat jeho flexibilnost a volnost pohybu, ale zároveň se snaží dosáhnout větší tuhosti soustavy a snížení výsledné nepřesnosti. Tato koncepce se využívá při vysokých zatíženích, kdy dochází k velkým silovým působením. [3][6]

Obrázek 3.2: *Vlevo příklad sériové kinematiky, uprostřed paralelní a poslední smíšené, upraveno z [1]*

3.4 Kinematická dvojice

Označujeme tím vazbu v místě součásti robota, kde jsou navzájem pohyblivě spojeny dvě ramena robota. V praxi se nám nejčastěji vyskytují dvě takovéto vazby. Tou jednou je rotační označovaná jako velké R a translační, neboli posuvná označována jako velké T. Pro základní dosažení libovolné polohy koncového bodu, je nutné mít nejméně tři polohovatelné osy a další tři osy pro orientování. Pro základní polohování ve třech osách využíváme pět základních kinematických struktur. Jejich názvy nám zároveň charakterizují pracovní prostor. [3][6][8]

- TTT - kartézská kinematická struktura, je tvořena třemi translačními kinematickými dvojicemi neboli posunem ve všech osách.
- RTT - cylindrický pracovní prostor neboli část válcového prstence. Takovýto prostor nám vytvoří struktura o jedné rotaci a dvou posunech.
- RRT - sférický pracovní prostor jinak také část kulového vrchlíku. Takovýto prostor nám zajišťuje dvojí rotace a jeden posun.
- SCARA je speciální případ RRT struktury, její pracovní prostor však není, jako u předchozí část kulového vrchlíku, ale geometrický tvar na základě válce.
- RRR - angulární kloubový robot. Typická struktura, kterou si většina lidí představí pod pojmem průmyslový robot. Je tvořena čistě z rotačních vazeb

Obrázek 3.3: *Příklady strojů jednotlivých struktur, upraveno z [14]*

Na obrázku jsou jednak zobrazeny Kinematické struktury, ale také různé druhy manipulátorů, které bychom mohli nazvat průmyslovými roboty.

3.5 Souřadné systémy

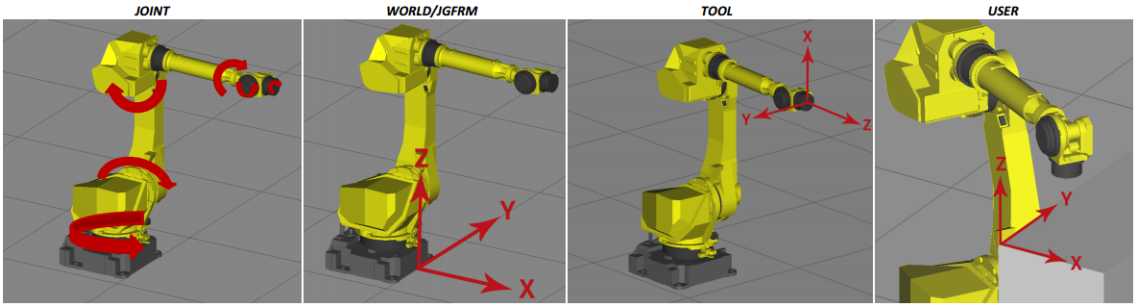
Robot může pracovat v několika souřadných systémech, díky kterým je schopný přesně určovat svou polohu. Informace o poloze je potřebné znát z důvodu jeho řízení. Bez souřadných systémů by nebyl schopný polohovat svůj koncový bod.

Systém WORLD/JGFRM, neboli světový. Je nezaměnitelně definovaný kartézský souřadný systém vetknut do paty robota.

Systém USER, vázaný na světový. Tento souřadný systém je definován uživatelem. Po nadefinování je robot a jeho polohy spjaty s počátkem tohoto systému. Počátek je také často označován jako origin.

Systém TOOL, je souřadný systém nástroje umístěný na středu zápěstí robota. Je stejně jako systém USER definován uživatelem. Definuje polohu středového bodu nástroje, zkráceně TCP z anglického Tool Center Point. Středový bod nástroje je nutný k dalšímu určování údajů o poloze. Pokud není systém TOOL definován, je jako platný systém koncového nástroje brán systém zápěstí robota.

Systém JOINT neboli osový. Stejně jako systém WORLD je i tento systém nezaměnitelně vetknut v každé ose robota. Poloha je pak vyjadřována pomocí úhlů natočení jednotlivých kloubů. [7][9]



Obrázek 3.4: *Souřadné systémy a jejich umístění, upraveno z[13]*

3.6 Pravidlo pravé ruky

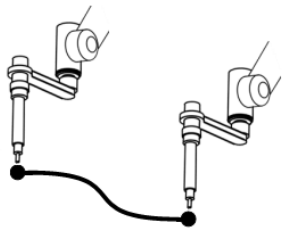
Pravidlo pravé ruky používáme, pokud chceme znát kladný směr os souřadného systému WORLD. Jeho použití je následovné. Z palce, ukazováčku a prostředníčku vytvoříme simulovaný pravoúhlý souřadný systém, ten umístíme tak, že palec míří nahoru od paty robota a ukazovaček směrem od přední strany robota. Kladný směr osy Z je pak reprezentován palcem, kladný směr osy X je reprezentován ukazováčkem a poslední kladný směr osy Y znázorňuje prostředníček.

Jak už jsem uvedl v kapitole 3.4, pro dosažení libovolné pozice je třeba šesti os. Osy pro napolohování označujeme XYZ, další tři osy jsou rotace kolem těchto os a jsou označovány jako WPR. Pro určení kladného směru rotace kolem jednotlivých os XYZ nám opět poslouží pravidlo pravé ruky. Palec necháme vztyčený a ostatní prsty nasměrujeme do dlaně, pokud pak palec srovnáme s kladným směrem dané osy, tak nám prsty reprezentují kladný směr rotace kolem dané osy. [7]

3.7 Pohyby

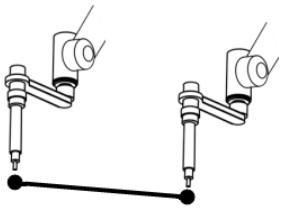
Dosažení jednotlivých poloh koncového bodu robot realizuje pomocí pohybů. Robot je schopen tří základních pohybů. [2][9][12]

Pohyb Joint, je zadávám pomocí dvou bodů, výchozího a cílového. Robot do definované cílové polohy dojede po libovolné trajektorii v nejkratším čase. Při tomto pohybu může být chování robota v jistých situacích nepředvídatelné.



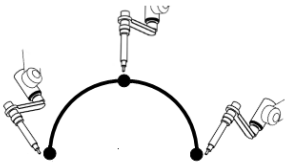
Obrázek 3.5: *Pohyb Joint.*

Pohyb Lineární, také je zadávám pomocí výchozího a cílového bodu. Robot do koncové polohy dojede po přímce definovanou rychlostí. Tento pohyb je vhodný, pokud potřebujme přesněji znát, jak se robot bude prostorem pohybovat. Bohužel se zde často můžeme setkat s problémem singularity. Jednou z možných příčin vzniku singularity je situace, kdy robot má několik možností jak se do cílového bodu dostat.



Obrázek 3.6: *Pohyb Lineární.*

Pohyb kruhový. Robot do definované koncové polohy dojede po kružnici. Na rozdíl od předchozích pohybů nelze pohyb po kružnici definovat jen výchozím a koncovým bodem, ale i třetím mezi-bodem, přes který TCP prochází. Na základě tohoto bodu je dopočítáván poloměr kruhového pohybu.



Obrázek 3.7: *Pohyb po kružnici.*

4 Programování průmyslových robotů

Programování průmyslových robotů je situace, kdy k robotu přicházíme s cílem vytvořit nějaký program či algoritmus, na jehož základě bude pak robot vykonávat svou činnost. Průmyslové roboty se napříč spektrem firem svou koncepcí od sebe mnohdy neliší. Jinak je tomu však z hlediska softwaru, uživatelských prostředí a programovacích jazyků. Nicméně co většina značek podporuje je možnost offline a online programování. [2][9][11]

4.1 Online programování

Online programování je druh programování, při kterém je využívána přímá interakce člověka a průmyslového robota. Programování spočívá v přímém navádění robota do potřebných poloh, které jsou ukládány do paměti řidicího systému robota. Když robot zná všechny potřebné body na požadované trajektorii, příkazy pro řízení nástroje na svém zápěstí, příkazy potřebné pro větvení programu, zkrátka jeli program připraven. Poté může řídící jednotka vysílat pokyny robotu. Při online programování jsou dvě možnosti, jak člověk může s robotem interagovat. [2][9][11]

- Jednou možností je, že člověk s robotem interaguje přes uživatelské rozhraní neboli Teach-pendant. Teach-pendant je specifické zařízení založené na bázi klasického počítače, kabelem připojený k řídící jednotce. Existuje mnoho druhů a možností, jak s Teach-pendanty pracovat. Programátor pak řídí veškeré úkony robota pomocí Teach-pendantu. [2][9][11]

- Druhou variantou, je místo použití Teach-pendatnu pro navádění robota do potřebných poloh, použít ruční navádění a robota do potřebné polohy dostat za pomocí vlastních rukou. Použití Teach-pendantu se programátor stejně nevyhne, polohy a ostatní příkazy je stejně třeba nahrát a to jen přes Teach-pendant. Nicméně toto nezlez aplikovat na všechny roboty. Sám robot k tomu musí být uzpůsoben. Speciálně pro tento druh online programování FANUC uvedl na trh svou řadu kolaborativních robotů, disponující touto funkcí učení. [2][9][11]

4.2 Offline programování

Jako druhá varianta se tu nabízí programovaní v režimu offline. Tato varianta využívá softwaru na bázi 3D simulačního prostředí, kde si uživatel vytvoří virtuální robotické pracoviště či robotickou buňku včetně všech jeho okolních příslušenství, jako jsou například bezpečností ohrazení, dopravníky, odkládací prostory a podobně.

Uživatel má možnost si své prostředí nakonfigurovat přesně dle reálného pracoviště, či jen vytvářet simulace možných nebo vyvíjených pracovišť. Pro realizaci simulovaného prostředí s reálným podkladem je tedy nezbytné, aby uživatel znal přesnou konfiguraci reálného pracoviště, včetně jeho přesných rozměrů a použitých zařízení.

Při simulaci je pak možné sledovat a ladit pracovní proces robota, včetně strojového času cyklu, testovat dosažitelnost jednotlivých poloh, možnosti drah pohybů, odhalovat kolize a podobně. Tvorba pracoviště je možná přímo v simulačním softwaru nebo je možné nahrát již vytvořený CAD model. [2][9]

4.2.1 Význam offline programování

Jaký význam má offline programování v nějakém softwaru, když můžeme využít robota. V dnešní době si velké podniky zabývající se průmyslovou výrobou nemohou dovolit na půl dne, ne-li na den odstavit výrobní linku, a zkoušet či ladit program a jeho proces. Často firmy vynakládají nemalé úsilí na to, aby dokázaly výrobní proces stabilizovat a vyhnuly se i těm sebemenším prostojům, které je stojí peníze. Investují velké úsilí do nekonečného zlepšování a studií ohledně úspor.

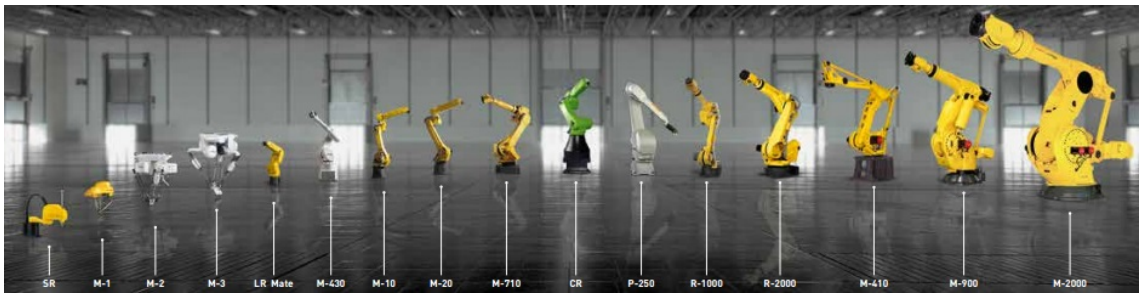
V těchto případech je takovýto simulační software ideálním řešením. Díky němuž nemusí přerušovat výrobu a zároveň mohou dále pracovat na zlepšeních, či s předstihem odsimulovat novou výrobní linku, která se bude teprve realizovat. Programátor si v simulačním softwaru vytvoří novu verzi programu a pak ji v době, kdy je linka v rámci údržby odstavena odzkouší. Když se odstávka linky v rámci údržby nebo jiné akce nekoná, tak je nutné výrobu pozastavit. Nicméně taková odstávka je v řádu minut. Programátor odzkouší nový program a zjistí případné nedostatky, nahraje zpět starý program a pokračuje se ve výrobě. Stále je to časově přijatelnější, nežli zastavení linky na půl dne. Efektivitu v rámci minimalizace korekcí na reálném modelu je možné docílit co nejpřesnějším simulačním modelem. Nicméně úplné zrušení korekcí se při testování na reálném modelu nevyhneme, vždy je třeba navržený program odladit a provést potřebné úpravy.

4.3 Historie firmy FANUC

FANUC je japonská firma datující svou produkci od roku 1956. V této době se zakladatel firmy FANUC Dr. Seiuemon Inaba podílel na vývoji technologie NC neboli Numeric Control, kterou využívaly obráběcí stroje před příchodem technologie CNC.

Výzkum v odvětví pohonů otevřel dveře vývoji robotů. Proto dnes po více než šedesáti letech od vniku firmy, je po celém světě na čtyři a půl miliony řídících jednotek CNC a více jak pul milionu robotů. Díky těmto úspěchu se také dnes značka Fanuc právem řadí mezi přední světové výrobce produktů pro oblast průmyslové automatizace.[1]

4.4 Přehled robotů firmy FANUC



Obrázek 4.1: *Srovnání velikostí PR Fanuc od nejmenších po největší, upraveno z[1]*

Roboty rozděluje firma do několika skupin odpovídajících požadavkům nejrůznějších aplikací. Samotné skupiny však nejsou přímým rozdělením. FANUC má na svém seznamu více než sto modelů, které spadají pod různé produktové řady. Na základě jejich technických parametru a specifikací se určuje, zda jsou pro danou aplikaci vhodné. Nemusí být tedy nutně jeden model zařazen do jediné skupiny, ale muže svými parametry vyhovovat více aplikacím.

Skupiny, do kterých Fanuc zařazuje své roboty jsou například kolaborativní, kloubové, delta, paletizační, lakovací, roboty pro svařování, roboty s vrchní montáží a nově i série robotů SCARA, která byla uvedla na trh v roce 2017.

FANUC také nabízí pro své roboty rozšíření o různé systémy, které dovolují přizpůsobit robota co nejpřesněji požadavkům aplikace. Je ho možné rozšířit o prvky zlepšující inteligenci a pohyblivost. Není tedy problém robota pomocí senzorických systému naučit vidět, odhalit překážky a učit se. Robotické pracovišťů je také možné rozšířit o polohovací zařízení, které ještě zvýší flexibilitu robotického centra. [1]

5 Úvod do praktické části

Nyní se budu věnovat popisu tvorby praktické části. Jelikož jsem vytvořil dvě úlohy, tak práci rozděluji do dvou částí a každá popisuje jednu výukovou úlohu.

První úloha slouží jako seznámení, na které ukážu práci s robotem v online režimu, jeho přípravu před offline programováním. Dále provedu seznámení a přípravu prostředí Roboguide pro offline programování. To zahrnuje vytvoření síťové komunikace mezi reálným robotem a našim PC, dále pak založení projektu, přípravu virtuálního pracoviště za použití vlastního 3D modelu rámové konstrukce buňky a realizace navržené manipulační úlohy. Nakonec je proveden export dat do robota a otestování úlohy.

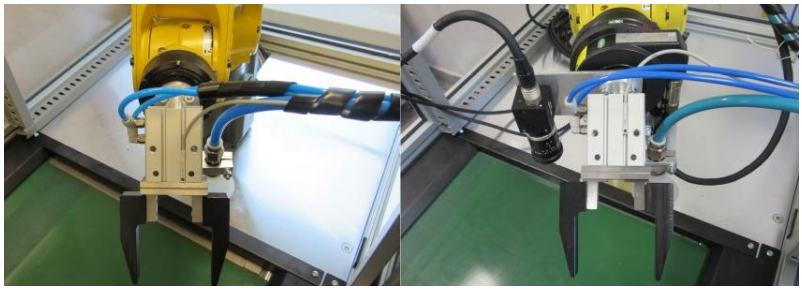
V druhé úloze se pak věnuji vytvoření úlohy s pomocí kamery, díky které budu vyhledávat objekty a rozeznávat tvary a počty teček. V jejím popisu je zahrnuta jak příprava samotné kamery a programu, tak i popis testů a úprav, které byly nutné pro zajištění stabilní funkce.

5.1 Příprava robota pro online programování

5.1.1 Seznámení s pracovištěm

Naším pracovištěm je výuková buňka osazená robotem Fanuc LR mate 200iD/4S. Tento robot je z řady nejmenších angulárních robotů se sériovou kinematikou, kterého FANUC nabízí. Možná zatížitelnost jsou čtyři kilogramy a dosah necelý metr. Zápěstí robota je osazeno efektorem s pneumatickými kleštěmi a pneumatickou přísavkou.

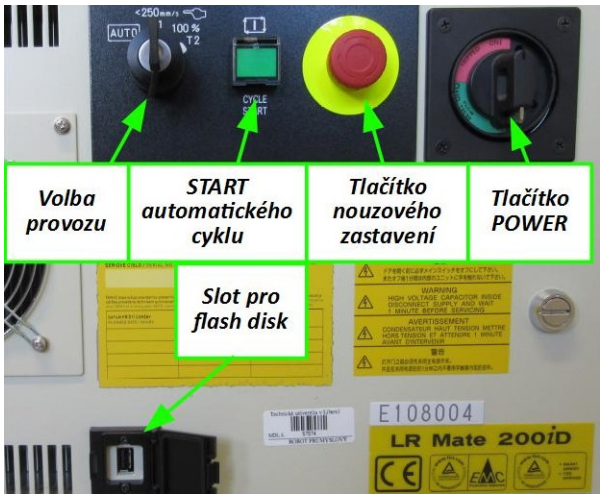
Učebna obsahuje tyto buňky tři, v základu jsou identické. Jejich konstrukce je z hliníkových montážních profilů, prostor buňky je zakryt ochranným plexisklem, přístup do buňky je skrze posuvnou bariéru. Ta je opatřena koncovými snímači. Při automatickém provozu se prostor robota uzavře, aby byl znemožněn přístup a byla splněna bezpečností podmínka. Pracovní plocha je opatřena dopravníkem. Ve vrchní části buňky je umístěna statická kamera značky COGNEX a osvětlení pracovního prostoru buňky. Jak už jsem uvedl, druhá buňka je nově osazena upraveným efektorem s novou kamerou a snímačem Force Sensor. Nová kamera narozdíl od té původní není staticky umístěna, ale je uchycena na zápěstí robota. Podrobné schéma výukové buňky vloženo v příloze. A.1



Obrázek 5.1: *Vlevo starý efektor, vpravo efektor rozšířen o senzor a kameru.*

5.1.2 Uvedení robota do provozu

Pro práci s robotem, je nejdříve nutného uvést ho do provozu. Provoz robota se ovládá skrze operátorský panel, který je součástí řídící jednotky. Na obrázku níže je vyobrazen spolu s popisem jednotlivých částí.



Obrázek 5.2: *Operátorský panel na řídící jednotce.*

Jako první na co bychom se měli zaměřit je přepínač *Volba provozu*, neboli režim, v jakém hodláme s robotem pracovat. Máme tři režimy AUTO, T1 a T2. Každý režim má svá specifika a podmínky pro svůj provoz. To se týká především omezení rychlosti a bezpečnostních podmínek.

- Režim AUTO složí jako automatický provoz. Robot vykonává spuštěný program, dokud neskončí, nebo ho sami nezastavíme. V tomto režimu robot nemá omezenou rychlost a může tedy přejíždět rychlostmi nastavenými v programu. Je tedy nutné, aby veškeré podmínky, které jsou pro tento režim potřeba, byly splněny. V našem případě je to bezpečnostní podmínka zavřených vrátek. Jsou hlídána snímačem, pokud bude robot spuštěn v režimu AUTO a dojde k otevření vrátek, systém okamžitě zareaguje a robota odstaví z provozu.
- Režimy T1 a T2 slouží jako seřizovací provozy při testování programů. T1 je s omezenou rychlostí na maximálně 250mm/s. T2 je provoz při kterém je

možné testovat program s nastavenou pracovní rychlostí. Pro uvedení robota do pohybu v těchto režimech je nutné stisknout takzvané tlačítko *Deadman*. Dále pro provoz nejsou nutná aktivní ochranná zařízení.

Při návrhu a testování je třeba mít režim zvolen na T1. Později, když jsou otestované pozice, lze provést test v režimu T2 plnou pracovní rychlostí a otestovat průjezdové rychlosti. Pokud je program otestován a veškeré nedostatky odstraněny, lze ho spustit v režimu AUTO.

5.1.3 Uvedení robota do pohybu

K samotnému ovládání a programování robota slouží takzvaný iPENDANT nebo také TEACHPENDANT. Dále ho budu označovat jen jako pendant. Části pendantu jsou červené tlačítko pro nouzové zastavení v případě neočekávané situace. Na opačné straně pendantu je otočný přepínač k aktivaci pendatnu při režimech T1 a T2, nebo deaktivaci pro režimu AUTO. Pendant disponuje dotykovým displejem a klávesnicí. Z druhé strany je slot pro flash disk a dvě tlačítka, neboli takzvané *Deadmany*. Tyto tlačítka jsou třípolohová a slouží jako bezpečnostní podmínka, pokud aspoň jedno nebude stisknuto, nelze s robotem v režimech T1 a T2 pohybovat ani spustit program. Zabraňují tedy nechtěnému pohybování robota.

Robot se může pohybovat v několika souřadných systémech. Volbu mezi systémy provádíme pomocí kláves SHIFT+COORD. Aktuálně zvolený souřadný systém se nachází vpravo nahoře na obrazovce. Na začátek doporučuji nastavit souřadný systém na systém JOINT. V tomto systému se robot pohybuje pouze v jednotlivých kloubech. Poté doporučuji vyzkoušet si i ostatní souřadné systémy.

Na začátek je dobré snížit rychlost a nenastavovat více jak dvacet procent. Nastavení rychlosti se provádí pomocí kláves ve spodní části klávesnice s označením +% a -%. Samotná nastavená rychlost je pak vidět na obrazovce nahoře vpravo a je vyjádřena procenty.

Předtím, než s robotem začneme pohybovat, je třeba zrušit chybový stav. To provedeme držením kláves Deadman+SHIFT a stisknutím klávesy RESET, poté lze s robotem pohybovat nebo spustit program. Pro ovládání pohybu slouží modrá tlačítka na pravé straně pendantu. Mají na sobě dvojí označení. Jako hlavní je znázorněna osa systému a její směr, v závorce je pak uvedená příslušná osa robota.

Před samotnou prací s robotem je třeba nejdříve nadefinovat náš nástroj pomocí TOOLFrame a také nadefinovat náš pracovní prostor pomocí USERFrame.

5.1.4 TOOLFrame

TOOLFrame je vytvoření souřadného systému nástroje, jinak také nadefinování polohy jeho koncového bodu a orientace vůči šesté ose. Můžeme se také setkat s pojmem TCP, neboli Tool Center Point, tento pojem jsem již zmiňoval výše v kapitole o souřadných systémech 3.5. Bez nadefinování nástroje je jako koncový bod brán souřadný systém šesté osy ramene robota.

Metod pro nadefinování nástroje je několik. Patří mezi ně tříbodová, šestibodová a pomocí přímého zadání hodnot například z datasheetu. V SETUP Frames

v adresáři OTHER si zvolíme TOOLFrame. Poté si nastavíme, že chceme souřadnice uložit na Frame na desáté pozici a klikneme na kolonku DETAIL. Tím se dostaneme do nastavení Framu. V kolonce METHOD zvolíme způsob, jakým nadefinujeme náš TOOLFrame. Já jsem zvolil šestibodovou (XZ). Konkrétní hodnoty Framu v příloze A.4b.

5.1.5 USERFrame

USERFrame je námi nadefinovaný kartézský souřadný systém, ve kterém pracujeme, a jsou k němu spřaženy naše polohy. Význam jeho vytvoření spočívá v tom, že pokud budeme hýbat nebo jiným způsobem měnit orientaci pracovní plochy robota, ale on sám zůstane na původním místě, je nutné přeučit jeho staré pozice na nové.

Pokud jsme si však pracoviště nadefinovali jako USERFrame, stačí ho pouze aktualizovat a není třeba složitě předělávat jednotlivé pozice v programu. Obdobným způsobem jako při vytváření TOOLFramu postupujeme i u USERFramu. Zvolíme si hodnotu Framu do kterého uložíme souřadnice a provedeme definování. Pro definování jsem využil tří bodovou metodu. Konkrétní hodnoty Framu v příloze A.4a.

5.2 Příprava prostředí v Roboguide

Před samotným programováním je nutné připravit si náš virtuální prostor tak, aby odpovídal výukové buňce. Je tedy zapotřebí naimportovat části našeho pracoviště. Je zde možnost využít knihovnu se základními konstrukčními částmi, jako jsou různé dopravníky, palety, vozíky, stojany, případné překážky a efektory.

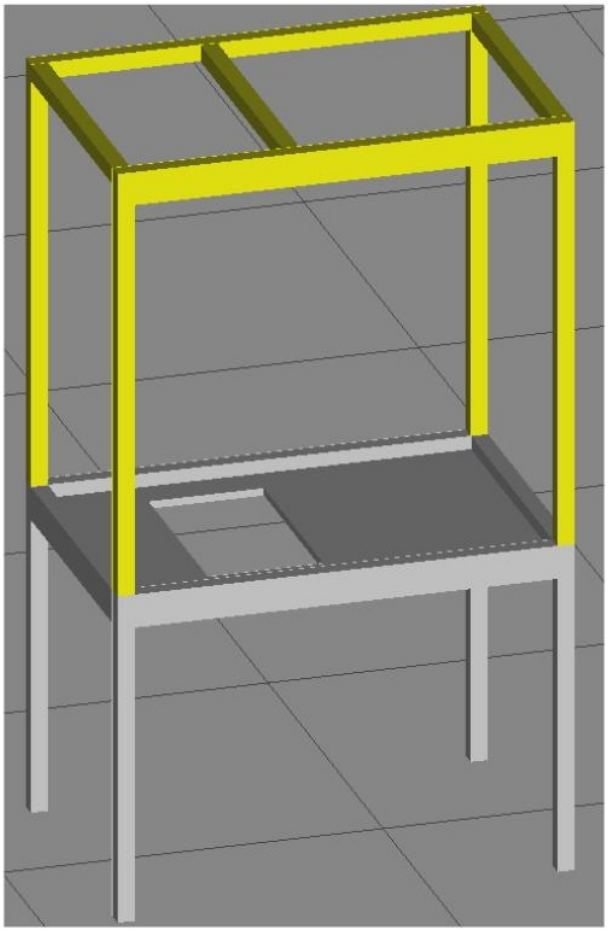
Součásti z knihovny jsou obecné a nevyhneme se úpravám velikostí a umístění. Je tedy výhodou pokud používáme vlastní modely, které jsou navrhnuty v reálných rozměrech, přesně dle potřeb a následné úpravy v Roboguidu jsou minimální a týkají se spíše úpravy pozice. V následujících odstavcích popíši kroky pro vytvoření našeho pracoviště. Tím se seznámíme s pracovním prostředí Roboguide a následná tvorba programu bude více méně práce čistě v pendantu, který již známe.

5.2.1 Modelace buňky

Ještě než začnu s popisem tvorby virtuálního pracoviště, rád bych uvedl, že od začátku jsem chtěl, aby virtuální pracoviště působilo realisticky. V průběhu tvorby pracoviště jsem zjistil, že knihovna s 3D modely neobsahuje takový rám, který by byl vhodný. 3D model spolu s výrobní dokumentací od výrobce nebyl k dispozici a tak jsem se rozhodl rám vymodelovat sám.

Rozměry a jednotlivé vzdálenosti buňky jsem si naměřil. Samotnou buňku jsem pak dle rozměrů vytvořil pomocí softwaru FreeCad. Buňku jsem vytvořil zjednodušenou a z praktických důvodů dvoudílnou. V Roboguidu je totiž možnost importovaná tělesa skrýt, aniž bychom je museli vymazat. Vrchní část je tedy možné v případě, že nám bude bránit při náhledu na robota skrýt. Postup jak nato je uveden v kapitole

Vložení objektů 5.2.6. Dále se samotné tvorbě buňky ve FreeCadu věnovat nebudu, zhotovený návrh bude k dispozici ve formátu STL jako příloha na CD.



Obrázek 5.3: *3D model rámu výukové buňky.*

5.2.2 Tvorba virtuálního prostředí

Tvorba virtuálního pracoviště začíná u založení projektu. Vytvoření nového projektu není nijak složité. Je to celkem 8 kroků, ve kterých získáme základní nastavení.

Pokud u založení nového projektu chceme, aby náš robot v Roboguidu měl stejné systémové nastavení jako reálný robot, tak je nejdříve nutné připojit náš počítač k reálnému robotu a vytvořit přes Robot Neighborhood připojení. Toto pak využijme při založení projektu a po založení náš virtuální robot převezme veškerá nastavení a informace uložené v paměti a všech registrech reálného robota.

Nicméně, pokud nehodláte pracovat reálně s robotem. Nebo toto využíváte jen jako studijní materiál pro domácí potřebu. Není podmínkou, že pro funkci výukové úlohy je nutné připojovat PC k robotu. Při zakládání projektu necháte v kroku dvě defaultní nastavení a projekt vytvoříte jen jako čistě virtuální prostředí. Potřebné hodnoty USERframe, TOOLframe a registrů je třeba nastavit ručně. Hodnoty pro nastavení framů naleznete v příloze A.4 a hodnoty a názvy registrů v příloze. A.3

5.2.3 Vytvoření komunikace mezi PC a robotem

Skrze síťový kabel se připojíme k našemu robotu. Otevřeme program Robot Neighborhood. V levém okně klikneme ve vlákně na adresář Robot Neighborhood, v pravé části se nám otevře okno s nadpisem Robot Group. Označíme kolonku Real Robot. Pokud nám rolovací okno nenabídne IP adresu 192.168.0.40 našeho robota, tak provedeme její zadání ručně. Nakonec napíšeme název například Bunka3 a potvrdíme pomocí tlačítka Add. Pokud vše proběhlo v pořádku dojde k vytvoření přístupového bodu, na který se budeme odkazovat. V okně vidíme název, cestu, status avaible a IP adresu našeho robota. Pokud je status unavaible nedošlo k automatickému nastavení adresy v síťovém adaptéru. Je tedy nutné tuto adresu zadat ručně.

Pokud je status avaible jsme připraveni při založení našeho projektu využít nastavení reálného robota. Vytvoření této komunikace je důležité i proto, abychom mohli importovat programy a nastavení z Roboguide do robota.

5.2.4 Založení projektu

V předchozím odstavci jsme si připojili robota k PC a nyní při vytvoření projektu můžeme využít data z backupu a přenést systémová nastavení do našeho virtuálního prostředí. Založení projektu se provádí v osmi krocích.

- Krok první, název projektu.
- Krok druhý, vytvoření konfigurace robota. V našem případě využijeme volbu „Create a robot from file backup“, do adresáře nastavíme přístupový bod, který jsme si vytvořili v RobotNeighborhood. Pokud nehodláme pracovat s reálným robotem, ponecháme kolonku s defaultním nastavením.
- Krok třetí, volba verze a zvolíme nejnovější.
- Krok čtvrtý, volba nástroje. Zaklikneme volbu nástroje později (Set Eoat later). Volit nástroj teď není třeba. Jeho volbu ukážu v kapitolách, kde se budu věnovat volbě a úpravě objektů.
- Krok pátý, volba našeho robota. Robota volíme podle toho, který se nachází v naší výukové buňce a tím je LR Mate 200iD/4s.
- Krok šestý lze jen přeskočit.
- Krok sedmý lze také jen přeskočit.
- Krok osmý, shrnutí. Dále pokračujme tlačítkem Finish

Tímto jsme úspěšně založili nový projekt. Nyní se před námi otevřela pracovní obrazovka s robotem. Prostředí je intuitivní a není složité na pochopení. Vlevo adresář s daty našeho projektu. V horní liště se nachází systémové záložky, spodní lišta obsahuje tlačítka s různými funkcemi, pro naši práci jsou stěžejní jen některá.

Mezi ně jednoznačně patří tlačítko *Show/Hide Teach Pendant* pomocí kterého zobrazujme a skrýváme virtuální Pendant, dále jsou tu tlačítka pro ovládání simulace, centrování pohledu, ikonka bílé myši nám ukáže nápovědu pro klávesové zkratky. Poslední využívanou ikonou je *Pravítko*. To využijeme především na začátku při úpravách pracoviště.

5.2.5 Vložení objektů

Objekty, které budeme v našem pracovišti potřebovat jsou: robot, efektor, dopravník, spodní a horní část buňky. Jediný objekt, který není nutné vkládat je sám robot. Ten se na pracoviště naimportoval po založení projektu. Ostatní objekty je třeba ručně vložit. Nejprve si všechny objekty naimportujme do prostředí.

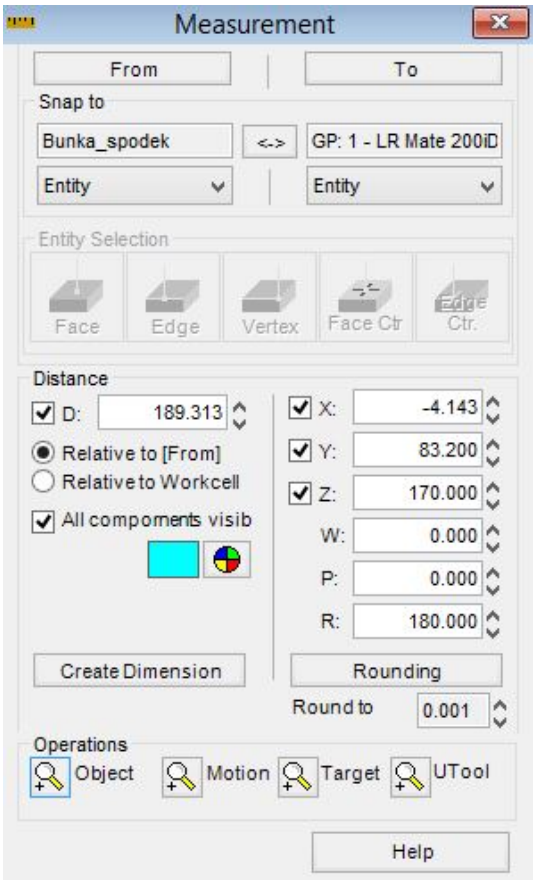
Je několik možností, jak provést import. Po levé straně je jak už jsem uvedl adresář obsahující náš projekt s programy, daty a ostatními komponenty týkající se našeho pracoviště. Nalezneme zde záložky *Machines, Fixtures, Parts, Obstacles* a *Workers*. Skrze tyto záložky lze vkládat různé objekty a v závislosti na tom, skrze co je importujme se dají nastavit jejich vlastnosti. Druhá možnost je, že v horní liště otevřeme záložku *Cell* a zobrazí se nám stejná nabídka jako je v adresáři na levé straně. Skrze tuto nabídku lze opět vkládat různé objekty. Vložení je obecně u všech stejné. Na dané záložce zvolíme, zda chceme využít knihovnu nebo budeme vkládat vlastní data. Pro využití knihovny zvolíme *CAD Library* a pro vlastní *Signle Cad files*.

Objekty vkládáme skrze záložku *Fixtures*. Jako první naimportujeme dopravník s označením *Conveyor09*, který nalezneme v *CAD Library*. Poté si pomocí *Single CAD File* naimportujeme obě části rámu buňky. Jelikož vkládáme části rámu pomocí *Single CAD File*, vkládáme vlastní 3D model. Vlastní modely je dobré uložit na známé místo, abychom věděli kde je najít.

Jako poslední si k robotu připojíme nástroj, to se provádí trochu odlišným způsobem. v adresáři se přes *Robot Controller1 -> GP: 1 – LR Mate 200iD/4s* dostaneme k záložce *Tooling*. V této záložce bude uložený náš nástroj. Dvojklikem otevřeme *UT: 10 (Eoat10)*. Nyní máme otevřené okno s parametry našeho budoucího efektoru, u kolonky *CAD File* pomocí ikony robota otevřeme okno s knihovnou základních konstrukčních částí. V záložce *EOATs* zvolíme *grippers* a vybereme nástroj s označením 36005f-200-2. Tímto máme všechny části naimportované. Samotné objekty se rozmístí náhodným způsobem soustředěně k počátku souřadného systému. Efektor je umístěn na poslední osu robota, jelikož je přidán jako nástroj nikoliv jako standardní objekt. Dále je nutné všechny objekty správně umístit, případně upravit jejich velikost.

5.2.6 Úprava velikostí a umístění

Objekty z knihovny, jsou často obecných rozměrů a nevyhneme se jejich úpravám a změně umístění. Tato úprava se provádí v okně vlastností objektů, které otevřeme buď kliknutím přímo na objektu, nebo skrze objekt v adresáři. Lze zde měnit barva, název, viditelnost a průhlednost. Ve spodní části jsou pak dvě pole s označením *Location* a *Scale*. Ty slouží pro změnu umístění či velikosti objektu. Umístění je zadáváno v milimetrech a velikost je nastavována poměrově. Pomocí *Lock All Location Values* lze uzamknout úpravy velikostí a umístění proti změnám.



Obrázek 5.4: *Nástroj pro odměřování vzdáleností.*

Pro správné umístění je třeba znát rozměry, ty lze zjistit pomocí funkce *Pravítko*, kterou nalezneme pod ikonou malého pravítka. V principu, volíme odkud a kam. Lze si také zvolit, jakým způsobem chceme, aby se nám bod kotvil na plochu, roh či hranu objektu. Můžeme si taky nastavit zda, chceme měřit z entity, neboli plochy, rohy a hrany. Nebo můžeme zvolit, aby byl bod kotven k originu. Jednotlivé vzdálenosti jsou následně vidět v tabulce měření. Zjištěné hodnoty využijeme pro umístění spodní části buňky, robota na pracovní plochu a úpravu dopravníku.

Nakonec zbývá upravit nástroj. Úpravu jsem vyřešil tak trochu nestandardně. Jak už jsem uvedl, výkresovou dokumentaci jsem bohužel nezískal a stejně to bylo s výkresem efektoru. Co se týče samotného rozměru, tak to nebyl zas takový problém. Stačilo posuvné měřítko a rozměry jsem zjistil. Nicméně nástroj jsem v Roboguidu stále neměl srovnaný a jediné co bylo k dispozici, byl souřadný systému nástroje, který jsem získal pomocí TOOLFramu. Souřadný systém levitoval před zápěstím a mě napadlo virtuální nástroj vycentrovat s osami TOOLFrame, tak aby odpovídal pozici v realitě. Robota v buňce jsem si srovnal tak, aby efektor byl těsně nad pracovní deskou. Tuto pozici jsem si nahrál do jednoduchého programu a ten přetáhl do Roboguidu. V Roboguide jsem si napolohoval robota na pozici uloženou v programu a provedl výškové srovnání nástroje tak, aby také byl těsně nad pracovní deskou stejně jako byl reálný robot. Tímto způsobem jsem si nastavil pozici nástroje. Program s polohou jsem poté smazal, již nebyl třeba. Nástroj z knihovny objektů není úplně vizuálně přesný, ale po úpravě jsou rozměrově stejné a pro náš účel je toto více než dostačující. Veškeré rozměry objektů jsou uvedeny v příloze A.2.

5.3 Tvorba programové části

Nyní se dostáváme k samotné programové části. Programování je o testování upravování, inovování prvotních představ. Často při tvorbě programu dochází k tomu, že tvůrce není spokojen nebo zjistí, že pokud chce svůj program ještě obohatit či vylepšit, musí nutně přetvořit část programu.

Podobným procesem jsem procházel při tvorbě úlohy i já. Začínal jsem od nejjednodušších příkazů. Ručně jsem učil robota jednotlivé pozice, a když program začal nabývat na počtu řádků a orientace v programu začínala být velmi nepříjemná, nahradil jsem opakující se pozice pozičním registrem. Cyklicky opakované části programu jsem vytvořil jako podprogramy, které jsem pak volal. Později jsem vytvořil algoritmus, který na základě různých podmínek automaticky vypočte ostatní polohy pomocí offsetu jedné výchozí pozice, a podobně jsem postupoval dále. Nebudu popisovat, jak jsem program tvořil krok po kroku, jak už jsem nadnesl, byl to vývoj a popis toho by byl nesmyslný. Zaměřím se raději na vysvětlení principu jednotlivých programů.

5.3.1 Návrh úlohy Pyramida

Úlohu Pyramida jsem navrhl tak, že robot si odebírá kostky z přesných poloh a následně je postaví do pyramidy a poté ji rozestaví zpět do původních poloh. Navíc pracuje ve smyčce a program je tak cyklicky vykonáván pořád dokola. Jak už jsem uvedl, rozhodl jsem se úlohu rozdělit na několik programů, každý program obstarává jistou část úlohy. Snažil jsem se program tvořit tak, abych na jednoduché úloze ukázal základní příkazy, jejich kombinace a možná řešení.

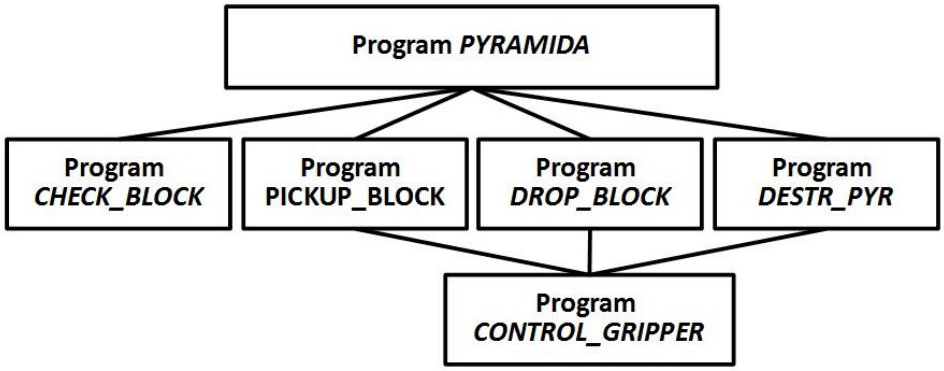
V úloze využívám volání programů, prvotní inicializaci použitých registrů a pozičních registrů. Registry využívám jako paměti čítačů, poziční registry využívám jako paměti pro pozice. Poziční registry využívám jednak jako statické proměnné pro stálé pozice, ale také jako proměnou pro ukládání aktuální polohy, která se v průběhu programu stále mění. Dále také využívám FOR cykly a větvení programu podmíněnými příkazy. Názvy a hodnoty registrů v příloze A.3.

5.3.2 Založení programu

Než začneme programovat, vytvoříme si programy, které budeme potřebovat. V Roboguidu je program možno vytvořit několika způsoby. První je, že si v horní liště otevřeme kolonku *TEACH* a zvolíme *Add TP program*. Program se nám objeví vlevo v adresáři *Programs*. Druhá možnost je pravým tlačítkem kliknout v adresáři na *Programs*, poté zvolit *Add TP program*. Dále stejné jako v předchozím případě. Poslední možnost je vytvořit program přímo v pendantu. Po vytvoření se nám program opět objeví v seznamu programu v adresáři *Programs*.

Jedním z výše uvedených způsobů je třeba vytvořit šest programů. Jednotlivé programy nesou názvy *PYRAMIDA, CHECK_BLOCK, PICKUP_BLOCK, DROP_BLOCK, DESTR_PYR, CONTROL_GRIPPER*.

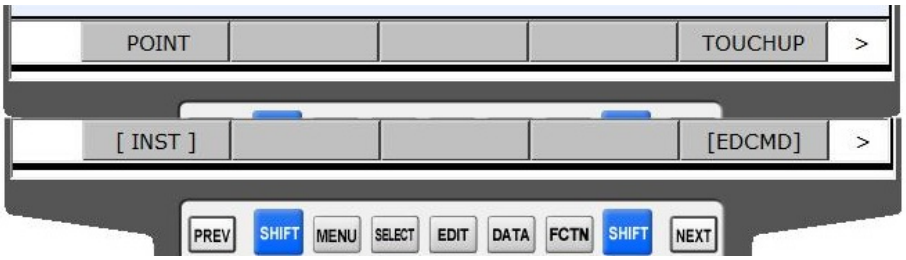
Následně popíšu principy jednotlivých programů. Hierarchie jednotlivých programů odpovídá schématu na obrázku 5.5.



Obrázek 5.5: *Hierarchie.*

5.3.3 Prostředky pro tvorbu programů

Program upravujeme pomocí virtuálního pendantu. Výhodou je, že virtuální pendant je přesnou softwarovou kopií toho reálného, s jehož obsluhou jsme se již seznámili. Pokud máme otevřený program, který hodláme upravovat, využijeme k tomu následující čtyři funkce ve spodní části dotykového displeje viz obrázek (5.6). Řadí s mezi ně POINT, TOUCHUP, EDCMD, INST. Každý pak obsahuje jinou sadu funkcí a příkazů.



Obrázek 5.6: *Funkce pro tvorbu programu.*

5.3.4 POINT

POINT slouží k uložení bodu s polohou, v jaké se nám aktuálně robot nachází. Po jeho otevření se nám zobrazí nabídka se čtyřmi možnostmi. Máme na výběr mezi lineárním a jointovým pohybem. Zároveň si můžeme zvolit, zda do bodu chceme najet úplně nebo kolem bodu jen projet. Po uložení bodu, můžeme měnit, zda má být lineární nebo jointový. Pokud označíme číselnou hodnotu bodu, tak ji můžeme změnit na jinou. Případně pomocí kolonky CHOICE , která se nám ukáže ve spodní části displeje vyměnit bod za libovolný poziční registr PR[...]. Dále lze upravovat rychlost, můžeme měnit jak hodnotu, tak jednotky ve kterých je uváděna. Standartě

je příkaz vkládán s rychlostí uváděnou v procentech. Poslední co lze změnit je přesnost při průjezdu bodem. Při použití možnosti FINE robot přesně najede do bodu a na krátkou dobu se zastaví. Pokud nepotřebujeme, aby robot do polohy najížděl přesně, a chceme jen bod minout, tak použijeme volbu CNT. Čím zadáme menší hodnotu CNT, tím se robot více přiblíží bodu.

5.3.5 TOUCHUP

Pokud stiskneme SHIFT a TOUCHUP, přepíšeme tím hodnoty pozice zvoleného příkazu, na hodnoty pozice ve kterých je robot právě najetý. Funguje to pouze na příkazy tykající se poloh, tedy na příkaz [P] neboli bod a na příkaz [PR] neboli poziční registr.

5.3.6 INST

Je to funkce, ve které se nachází široké škála instrukcí. Zde nalezne registry, vstupy a výstupy. Jsou zde příkazy na čekání, volání programů, skoky v programech, příkazy pro tvoření FOR cyklů a ostatní programové prostředky.

5.3.7 EDCMD

V této funkci nalezneme příkazy jako je *Insert* pro vložení prázdných řádků, *Delete* pro mazání a *Copy/Cut* pro kopírování nebo přesouvání částí programu. Dále tu nalezneme *Comment*pomocí něhož lze ukázat případně skrýt komentáře jednotlivých částí, jako jsou například registry. Pomocí *Remark* můžeme část programu zakomentovat a tím zabránit jeho vykonání. Toto jsou asi nejvyužívanější příkazy.

5.3.8 Program *PYRAMIDA*

Program *PYRAMIDA*, slouží jako hlavní tělo úlohy. Skrze tento program spouštíme robota a dochází k volání podprogramů. Hned na začátku zde dochází k volání podprogramu *CHECK_BLOCK*. Dále je zde prováděna inicializace použitých registrů a pozičního registru pro aktuální polohu. Následuje FOR cyklus, v němž je prováděn výpočet fáze, ve které se robot nachází. V tomto FOR cyklu je dle podmínky volán buď podprogram *PICKUP_BLOCK* nebo *DROP_BLOCK* v závislosti na stavu aktuální fáze. Zde v šesti krocích dochází k postavení pyramidy. Po jejím dokončení je vytvořen další FOR cyklus také se šesti cykly, ve kterém je volán program *DESTR_PYR* pomocí kterého dojde k rozestavění kostek do původních pozic. Proces stavby pyramidy končí příkazem JMP LBL[1], který nás přesune na příkaz LBL[1] za příkazem, který volá program *CHECK_BLOCK* a celý proces je opakován. K opakování bude docházet, dokud sami program nepřerušíme. Aby nedocházelo k nekonečné smyčce, lze povést jednoduchou úpravu programu. Úprava spočívá ve vytvoření dalšího FOR cyklu místo příkazu LBL. Tento FOR cyklus by měl za úkol například po třech opakováních program ukončit. Počet opakování lze nastavit dle libosti. Zdrojový kód v příloze A.5a.

5.3.9 Program *CHECK_BLOCK*

Program *CHECK_BLOCK* je úvodní program, který se provede jako první po spuštění hlavního programu PYRAMIDA. Program je napsán nejprimitivnějším způsobem a to tak, že si manuálně robota nasměruji do pozice a tu uložím. Tímto způsobem by se dala celá úloha programově napsat do jednoho programu. Program by však byl příliš dlouhý a nepřehledný. Záměrně jsem tedy program navrhl takto, abych ukázal, jak by se mohlo, ale nemělo programovat. Cíl tohoto programu je takový, že nemáme vizuálně určené pozice kam kostky postavit. Potřebujeme tedy, aby robot provedl nájezd do jednotlivých poloh kostek a my mu tam ty kostky ručně nastavili.

Program má dvě fáze. V té první provede nájezd do tří výchozích poloh, ze kterých si robot později kostky bere. V každé poloze se zastaví a vykoná příkaz *čekej pět vteřin*. V tomto čase my program přerušíme puštěním tlačítka SHIFT a *Deadman*. Kostku umístíme mezi klepeta a pokračujeme v programu. Robot po uplynutí pěti vteřin uchopí a pustí kostku, čímž si ji vycentruje do ideální pozice pro uchopení, tím při pozdějším nájezdu zamezíme nárazu. Takto to provedeme i u zbylých dvou poloh. Z poslední polohy se robot po kružnici vrátí na první pozici.

Nyní začíná druhá fáze programu. Robot opět postupně najede na jednotlivé pozice kostek, nově však se sníženou nájezdovou rychlostí. Naším úkolem je sledovat, zda nedojde ke kolizi klepeta a kostky. Teď už nedochází k uchopení a puštění kostky, pouze jen k nájezdu. Tato druhá fáze je jen čistě jako kontrola, zda kostky zůstaly ve správné pozici. Na konci se program vrátí zpět, odkud byl volán a pokračuje dále. Tento program je vykonán pouze jednou. V průběhu kdy bude docházet k opakované stavbě pyramidy, už tento program vlivem příkazu LBL bude vynechán. Slouží tak čistě jako inicializace poloh kostek. Zdrojový kód v příloze A.5b.

5.3.10 Program *PICKUP_BLOCK*

Program *PICKUP_BLOCK* je krátký podprogram volán z programu *PYRAMIDA*. Má za úkol provést výpočet polohy pro uchopení kostky. Program má určenou polohu první kostky v pozičním registru PR[24]. Na základě jednoduché podmínky závisející na fázi pak vypočte ostatní dvě polohy kostek. Tuto vypočtenou polohu pak dočasně uloží na poziční registr PR[29] pro aktuální polohu. Informace o poloze je pak předána podprogramu *CONTROL_GRIPPER*. Po vykonání podprogramu *CONTROL_GRIPPER*, je v těle *PICKUP_BLOCK* vynulován poziční registr PR[29], aby byl připraven pro další informace. Pokud bychom v pozičním registru nechali starou pozici, tak při dalším použití by docházelo ke špatně nastaveným novým pozicím. Program končí návratem na místo, odkud byl zavolán. Zdrojový kód v příloze A.6a.

5.3.11 Program *DROP_BLOCK*

Dalším programem je *DROP_BLOCK*, má obdobnou funkci jako *PICKUP_BLOCK*, dochází zde pomocí podmínky k výpočtu pozice, kam kostku položit. Jednotlivé pozice pro položení kostky jsou opět přepočítávány z jedné výchozí pozice uložené v PR[25] pro položení kostky. Po výpočtu pozice jsou souřadnice uloženy na pozičí registr PR[29], poté je volán podprogram *CONTROL_GRIPPER*, po vykonání podprogramu je v těle *PICKUP_BLOCK* vynulován poziční registr. Opět se vrací zpět do programu *PYRAMIDA*, kde pokračuje dále. Zdrojový kód v příloze A.6b.

5.3.12 Program *DESTR_PYR*

Předposledním programem je *DESTR_PYR*. Cílem tohoto programu je pyramidu rozestavit a kostky umístit do původních pozic. Při tvorbě jsem zvolil trochu odlišný postup jako u předchozích dvou. Zde nerozděluji výpočet výchozí a cílové pozice do dvou programů. Zde jsem vytvořil jeden program, ve kterém dochází k výpočtu polohy, následně je zavolán *CONTROL_GRIPPER*. Opět jsou zde polohy přepočítávány z výchozích pozic pomocí podmínek. Zdrojový kód v příloze A.6c.

5.3.13 Program *CONTROL_GRIPPER*

Poslední programem je *CONTROL_GRIPPER*. Tento program se nachází na nejnižší pozici v hierarchii. Jeho úkolem je převzít informaci o pozici z pozičního registru PR[29]. Robotu přikáže, najeď do pozice PR[29] tím se nasměruje nad pozici kostky. Poté provede offset v ose Z o -50mm a sníženou nájezdovou rychlostí se přesune do pozice, kdy kostku buď uvolní nebo uchopí. To zda chce kostku odebrat nebo položit obstarává podmínka, která rozhoduje v závislosti na aktuální fázi cyklu. *CONTROL_GRIPPER* je tedy univerzální program pro manipulaci s kostkou. Zdrojový kód v příloze A.6d.

5.4 Export dat z Roboguidu

Abychom mohli vyzkoušet naší úlohu ve výukové buňce, je třeba přenést naše programy a data z registru do reálného robota. Jsou dvě možnosti jak přesun provést. Tou první je soubory uložit na flash disk a ten připojit k robotu a data přetáhnout, toto řešení lze využít, pokud nejsme k robotu připojeni počítačem.

My jsme si na začátku vytvořili připojení a tak data exportujeme. Před samotným exportem je třeba ukončit běžící procesy jak v Roboguidu tak na reálném robotu, potřebné je také z programu vystoupit. Pokud bychom to neprovedli, může se stát, že export nebude proveditelný. Export se pak provádí tak, že pro export programu zvolíme pravým tlačítkem daný program, zvolíme *Export* a *To Robot*. Otevře se nám okno, kde bychom měli vidět cestu našeho připojovacího bodu, který jsme si vytvořili v Robot Neighborhood, poté stačí potvrdit pomocí *Export*. Obdobně provedeme export registrů. Ty nalezneme pod adresářem *Variables* a jsou označeny jako *Numeric Registers* a *Position Registers*.

5.5 Test programu ve výukové buňce

Jako první testujeme program v režimu T1 se sníženou rychlostí. Test můžeme provést tak, že se program po spuštění vykoná automaticky nebo můžeme zvolit krokování pomocí tlačítka STEP, při této volbě se pak mezi jednotlivými příkazy pohybujeme pomocí kláves FWD(krok vpřed) a BWD(krok vzad). Tlačítko FWD slouží i pro spuštění programu.

Pokud vše proběhlo v pořádku, provedeme test v režimu T2 s plnou rychlostí. Pokud máme kompletně otestován a odladěn program můžeme přikročit ke spuštění v automatickém cyklu.

Spuštěni provedeme takto, přepneme provoz na AUTO, vypneme pendant v levém horním rohu. Na rozvodové skříni nám svítí kontrolka FENCE, to značí, že vrátka jsou otevřená. Robotickou buňku uzavřeme a modrým tlačítkem RESET na rozvodové skříni resetujeme chybu a kontrolka FENCE zhasne. Na pendantu stiskneme RESET a na operátorském panelu stiskneme zelené tlačítko START cyklu. Tlačítko se rozsvítí a program bude vykonáván automaticky, bez naší obsluhy.

Pro zastavení programu stiskneme tlačítko HOLD, nikdy robota nezastavujeme tak, že otevřeme buňku. Tímto způsobem dochází k tvrdému zastavení robota a není to šetrné. To samé platí i pro zastavení robota v režimech T1 a T2, zde zastavujeme robota buď pomocí tlačítka HOLD nebo puštěním tlačítka SHIFT, nikdy však puštěním tlačítka *Deadman*, toto zastavení má stejné účinky jako otevření buňky v režimu AUTO. Zastavení robota tlačítkem *Deadman* se vykonává jen v krajních situacích.

6 Úloha s využitím kamery

Úloha s využitím kamery bude spíše než na obsažení nejrůznějších příkazu zaměřena na samotnou práci s kamerou. Při tvorbě programu pak využijeme příkazy, které jsme si procvičili v první úloze. Cílem úlohy bude manipulace s hrací kostkou.

Robot začne odebráním kostky z první pozice výchozího pole, následně provede hod kostkou. Po hodu provede vyfocení kostky, určí její souřadnice kostky, vyčte tečky na kostce a podle počtu určí číslo cílové pozice, na kterou kostku umístí.

Po umístění celý proces opakuje. Střídavě tedy umisťuje kostku mezi výchozí a cílovou pozicí. Dokud celkový součet čísel nebude větší jak deset. Tato podmínka se v programu dá lehce změnit na libovolné číslo.

6.1 Testování optických vlastností

Při seznamování se s kamerou a její obsluhou jsem před finálním vytvořením programu provedl několik testů a pokusů optických vlastností. Na začátku jsem měl k dispozici pozadí tvořené dopravníkem, hrací kostku a kalibrační mřížku. Povrch pásu dopravníku je tvořen lesklou zelenou gumou a hrací kostka je vytvořena z šedého plastu. Tečky na kostce jsou vytvořeny jako prohlubně a nejsou od zbytku kostky nijak jinak rozlišeny.

První problém vyvstal ze špatných světelných podmínek. Ty byly způsobeny umístěním osvětlení k jedné straně buňky a to způsobovalo nežádoucí odlesky a stíny robota. Při kalibraci pak jedna strana kalibrační mřížky byla více nasvícená než druhá a měla na sobě odlesk, což má dopad na přesnost kalibrace. Nicméně rozdíly v zaměření bodů kalibrační mřížky za použití rozumné doby expozice nebyly zas tak radikální, aby to mělo zásadní vliv na kalibraci. V příloze B.1 uvádím pár obrázku, jak kamera zobrazuje mřížku při různých dobách expozic a za různých osvětlení.

Cílem práce není testování vlivu optických vlastností na kvalitu kalibrace, ale uzpůsobení okolí a nastavení kamery tak, aby byla schopna vykonávat danou činnost. Proto jsem zvolil vhodný kompromis a pokračoval dále. V případě, že bych se v průběhu tvorby programu setkal s problémem funkce způsobeným špatnou kalibrací, byl bych nucen vyladit podmínky více a snažit se zpřesnit kalibraci. Naštěstí ten to případ nenastal.

Větší problém však byl, že pás dopravníku a kostka vytvářely špatný kontrast. To mělo za následek špatné rozpoznávání objektu. Jelikož barva kostky je šedivá a dopravník tmavě zelený, tak to způsobovalo, že se na pozadí kostka ztrácela. Ani experimentování s dobou expozice jsem nedošel k uspokojivému výsledku. Stále

jsem se setkával s problémem, že buď jsem kostku nenašel, nebo při snížení hodnot shody a kontrastu docházelo k tomu, že i jiné objekty jako například zmiňované odlesky to vyhodnotilo jako kostku, opět v příloze B.2 uvádím pár příkladů. Špatný kontrast mezi kostkou a jejími tečkami pak způsoboval problém při určování počtu teček. Vlivem těchto vlastností jsem navrhl úpravy, které měly za úkol odstranit tyto nedostatky ještě před samotnou tvorbou programu.

Jako první jsem se zaměřil na světlo. Odlesky jsem z části vyřešil odstíněním přímého dopadu světla, nicméně světlo jsem však nemohl zakrýt úplně a některé odlesky přetrvaly. Nasvícení pracovní plochy je nutné zachovat. Dále jsem se rozhodl upravit plochu dopravníku překrytím pomocí bílé čtvrtky. Bílá čtvrtka nevytvářela takové odlesky a vytvořila pěkné jednobarevné pozadí a po zvýšení doby expozice zcela zaniknou. Poté jsem testoval, jakým způsobem se toto projeví na kontrastu se šedivou kostkou. Dále jsem otestoval několik objektů různých barev a došel k závěru, že nevhodnější by bylo mít černou kostku, což je logické pokud se zamyslíme nad tím, že máme bílé pozadí. Nechal jsem si tedy vytisknout čistou černou kostku, kterou jsem poté opatřil tečkami. Kontrast mezi bílou a černou se mi osvědčil a tečky jsem volil také bílé. Po otestování šedivé a černé kostky na bílém pozadí jsem zjistil, že při lokaci samotné kostky dosahují obě podobných výsledků. Nicméně černá kostka je na tom s výsledky o trochu lépe. Jinak tomu však bylo při vyhledávání teček, tam černá kostka s bílými tečkami ve výsledcích výrazně převyšovala a pro další použití jsem zvolil černou kostku. V příloze B.3 jsou přiloženy fota černé a šedé kostky při testování vlivu doby expozice, jeden test jsem provedl s dobou sníženou a lze si povšimnou přetrvávajících náznaků odlesků a stínů, další dvě fota jsou při zvýšené době a zde je už krásně vidět jak krásně zafungovalo bíle pozadí a odlesky už nejsou znatelné. Poslední dvě fota jsou pořízeny v době, kdy už nebylo venku světlu a je krásně vidět, že absence denního světla ovlivní výsledný obraz.

Obrázek 6.1: *Vlevo původní kostka, vpravo nová.*

6.2 Příprava před programovou částí

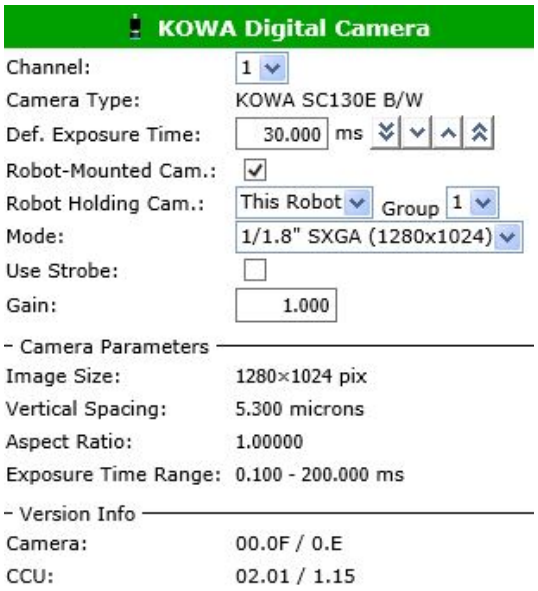
Samotná příprava před programovou tvorbou je prováděna v několika krocích. Je třeba nastavit kameru, provést kalibraci s nastavením framů a tvorba procesu, který bude kamera vykonávat. Toto bude obsaženo v následujících kapitolách.

6.2.1 Krok první připojit se ke kameře

Ke kameře se připojíme pomocí známé IP adresy. Tuto adresu zadáme do vyhledávače Internet Explorer. Tím se dostaneme na WEB SERVER našeho robota. Zde zvolíme kolonku *Vision Setup* a dostaneme se do prostředí. V dalších krocích provedeme nastavení naší kamery, její kalibraci a vytvoříme si proces, který bude kamera provádět.

6.2.2 Krok druhý nastavení kamery - Camera Setup Tools

V kolonce VTYPE zvolíme *Vision Setup Tools*. Zde si vytvoříme nový projekt. Projekt otevřeme a provedeme nastavení kamery. Zvolíme dobu expozice, nastavíme umístění kamery na robotu a zvolíme si rozlišení, v jakém bude naše kamera pracovat. Změny je nutné uložit pomocí SAVE. Konkrétní nastavení viz obrázek 6.2.



Obrázek 6.2: *Nastavení kamery použité pro mou úlohu.*

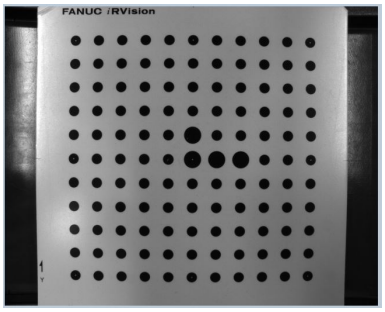
6.2.3 Krok třetí kalibrace kamery - Camera Calibration Tools

Dalším krokem je kalibrace kamery. Ve VTYPE zvolíme *Camera Calibration Tools*. Zde nastavíme, jakou kameru budeme používat, dobu expozice, *Grid Spacing* neboli rozteč bodů na kalibrační mřížce, hodnoty použitých Framů a jaký druh kalibrace budeme provádět. Konkrétní nastavení viz obrázek 6.3.

Obrázek 6.3: *Nastavení kalibrace použité pro mou úlohu.*

Kalibrace máme dvojího druhu. První je perspektivní kalibrace, využívá se, pokud bychom chtěli snímat tělesa s různou výškou. Já ve své úloze snímám tělesa v jedné výšce a tak jsem volil ortogonální kalibraci. Postup kalibrace je následovný.

Na pracovní plochu si do výšky plochy snímaných objektů vložíme kalibrační mřížku. Je třeba vyrovnat kameru, tak aby byla co nejvíce kolmo na mřížku. Orientaci mřížky je třeba nastavit tak, aby byla snímána jako je tomu na obrázku 6.4. Je také nutné, aby kamera viděla všechny kalibrační body mřížky.



Obrázek 6.4: *Správná orientace kalibrační mřížky.*

Poté si zvolíme vhodnou dobu expozice. Ta má za následek kvalitu obrazu, jeho nasvícení a tím i ostrost. Ideální doba nejde přesně určit. Výsledek obrazu dost ovlivňuji okolní světelné podmínky a optické vlastnosti těles. Je tedy třeba experimentovat a zhodnotit ideální dobu expozice.

Já jsem při volbě postupoval tak, že jsem zkoušel různé doby od nejmenší po největší. Při snížené době nebyl obraz tolik přesvícen a ostrost bodů kalibrační mřížky byla ideální. Nicméně, jak jsem psal výše, mřížka byla nasvícena nesouměrně a vytvářely se na ní odlesky. Toto se částečné vyřeší tím, že se zvýší doba expozice, tím se obraz více nasvítí a odlesky nebudou tak znatelné. Expozici ale není dobré mít nastavenou na příliš velkou hodnotu. Dochází pak k přesvícení a odlesky nejsou tolik vidět, ale sníží se ostrost kalibračních bodů a tím přesnost jejich zaměření. Je tedy nutné zvolit vhodný kompromis. Nakonec jsem zvolil nižší dobu, kdy jsou body lépe vidět.

Po nastavení expozice provedeme nastavení Framů. Na pendantu v MENU otevřeme adresář *iRVision*, zvolíme *Vision Utilities* a otevřeme *Automatic Grid frame Set*. Zde si zvolíme čísla USERFrame a TOOLframe do kterých se nám uloží hodnoty. Nastavíme expoziční dobu na hodnotu, kterou jsme si zvolili pro provedení kalibrace. Nastavíme *Grid Spacing* a nahrajeme naší polohu kamery na *Start Position*. Dále je třeba nastavit omezení úhlů a rozmezí prostoru v směru osy Z. Nastavení úhlů bude jistě v průběhu automatického vytvoření Framů snižovat. Pokud se s úhly dostanete pod hodnotu 10°, je třeba přenastavit fotící pozici tak, aby bylo umožněno nastavit úhly na vyšší hodnoty. Já jsem na všech osách nastavil omezení 30° a prostor v ose Z na 40mm.

Po nastavení hodnot stiskneme tlačítko FIND, čímž provedeme zaměření bodů kalibrační mřížky, poté pomocí tlačítka FCTN a zvolíme ABORT (ALL), tím zastavíme veškeré běžící procesy. Následně resetujeme chybový stav robota a pomocí EXECUTE dáme pokyn k vykonání automatického nastavení Framů. Rychlost je třeba snížit a nepřesahovat 15%. Vyšší rychlost má nežádoucí účinky na kvalitu nastavení Framů. V průběhu automatického nastavení může dojít k tomu, že se nám robot dostane na limit některé z os a zastaví se. Druhým případem je, že pozice, které se snaží dosáhnout, bude kolidovat s buňkou. Poté jak už jsem uvedl, je třeba snížit úhel limity a vrátit se zpět k provádění nastavení Framů. I já jsem musel v průběhu vytvoření Framů omezení úhlů přenastavovat. Samotný proces nastavení se provádí ve čtrnácti krocích. Po úspěšném nastavení Framů potvrdíme tlačítkem OK. Pokud máme na robotu nastavené Framy, přesuneme se zpět ke kalibraci. Nastavíme příslušná čísla Framů a provedeme nastavení pomocí SET a FIND. Každou změnu je třeba ukládat pomocí SAVE. Tímto máme nastavenou kalibraci. Pozor, pokud bychom nejdříve provedli nastavení kalibrace a až poté nastavení Framů, setkali bychom se později s chybovou hláškou „CVIS-172 Robot Pos (Z) is different Calib Pos“. Ta je způsobena tím, že kalibraci jsme provedli s jistými hodnotami Framů. Po automatickém nastavení Framů hodnoty přepíšeme, ale kalibrace byla provedena s hodnotami, které byly ve Framech obsaženy před přepsáním. Tyto rozdíly nám způsobí uvedenou chybu a nebudeme schopni využít kameru. Pro odstranění této chyby je nutné provést nové nastavení Framů a kalibrace. Uložení pozice po provedení nastavení Framů uložit do PR.

6.2.4 Krok čtvrtý vytvoření procesu kamery - Vision Proces Tools

Po nastavení Framů a kalibrace přichází na řadu *Vision Proces Tools*. Zde si vytvoříme funkce pro rozpoznávání kostky určení její pozice a funkci pro vyčtení naučených vzorů. K tomuto rozpoznávání využívám dva rozpoznávací nástroje. Tím prvním je *GPM Locator Tool* a druhým je *Blob Locator Tool*.

GPM Locator Tool(Geometric Pattern Matching)

GPM Locator Tool dále jen GPM je základní nástroj pro rozpoznávání a vyhledávání objektů. Princip funkce spočívá ve vyhledávání geometrické shody s naučeným objektem. Pro rozpoznání kostky je GPM z cela dostačují, dovoluje nastavení míry shody a práh kontrastu naučeného objektu s vyhledávaným objektem. Při nastavení hodnot, na základě kterých je vyhodnoceno nalezení tělesa postupujeme tak, že kostku umístíme přímo pod kameru. Pomocí *Teach* naučíme geometrii rozpoznávaného objektu a pomocí *Training Mask* zakážeme vnitřní plochu kostky, kde se nachází číslice. Tím jsem zajistil, že kamera bude hledat shodu pomocí geometrického tvaru obrysu kostky a nebude hledět na její obsah. Pokud při učení vzoru je kostka nasnímána špatně, například není zeleně ohraničen celý její obrys, je nutné otevřít v adresáři *2-D Single-View Vison Process* a správně nastavit dobu expozice. Opět správnou dobu expozice nezle přesně určit, chce to testovat a dle výsledků určit ideální dobu. Dále je třeba určit rozsah plochy, kde bude kostka hledána, to se provede skrze kolonku *Search Window*. Zde pomocí Set nastavíme plochu pro hledání. Po naučení vzoru a oblasti hledání je třeba nastavit origin kostky, ten je později využit pro nastavení referenční polohy, na základě které budou přepočítávány souřadnice nových poloh kostky. K nastavení originu slouží v nástroji GPM funkce *Cen Org*, automaticky se nastaví origin na ideální střed vzoru. Je zde také možnost pomocí *Set Org* nastavit umístění originu ručně, ale to není nutné. Kostka je pravidelných tvarů a automatické nastavení funguje v této situaci dobře. Nakonec je nutné provést nastavení referenční pozice kostky. To se provádí tak, že se v adresáři přesuneme do *2-D Single-View Vison Process*. Zde si kostku pomocí SNAP vyfotíme a u kolonky *Ref.Pos.Status* provedeme Set. Od této pozice budou prováděny výpočty nových pozic kostky. S kostkou nesmíme během procesu ani po jeho dokončení hýbat. Je třeba nejdříve efektor napolohovat do polohy kostky a tuto polohu si uložit. Později bude využívaná a offsetována na nové polohy kostky. V příloze B.4a a B.4b přikládám obrázky s nastavení *2-D Single-View Vison Process, GPM*.

Blob Locator Tool

Dalším korkem je zajistit vyčtení bílých bodů na kostce. Tyto body určují číslici, která na kostce padne. K tomu využijeme *Blob Locator Tool*, dále jen BLT vytvořeným pod funkcí GPM. Tím, že tuto funkci vytvoříme pod GPM zajistíme, že oblast, ve které budou body vyhledávány, se bude posouvat spolu s pozicí kostky.

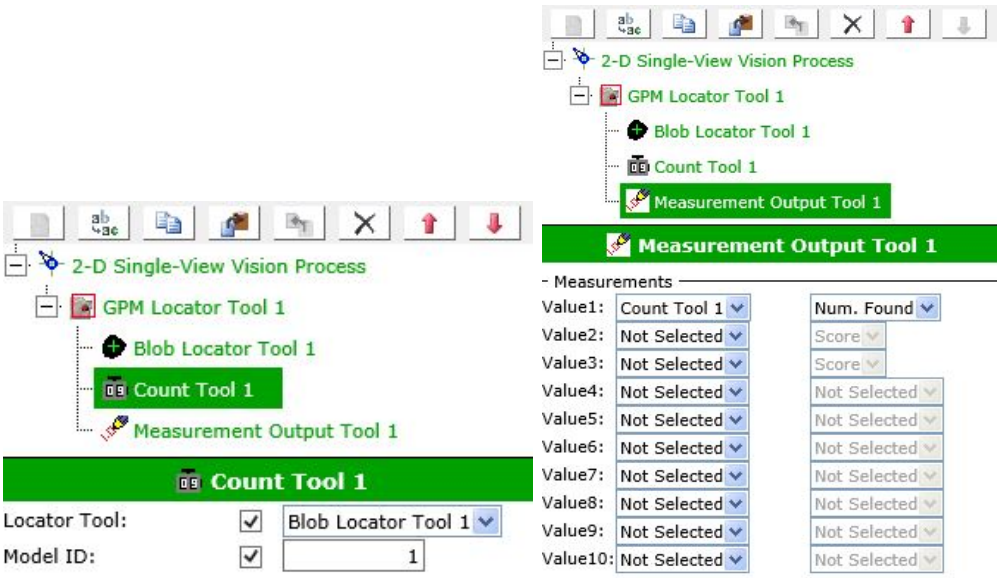
Oblast hledání nastavíme tak, že ji vložíme do plochy kostky, aniž by zahrnovala její hrany. Toto provedeme z toho důvodu, že pokud je kostka v krajní poloze a nenachází se přímo pod objektivem kamery, tak kamera kostku vidí z ostrého úhlu

a snímá i tečky na bocích kostky. Poté muže docházet ke špatnému vyhodnocení čísla. Tím, že oblast pro hledání omezil pouze na vrchní plochu, tak jsou tečky na bokách ignorovány. BLT totiž vyhledává objekty, na základě podobnosti s naučeným vzorem, je využíván pro nepravidelné tvary.

Dále pomocí *Teach* provedeme naučení vzoru, který bude hledat, což je naše tečka. Použité nastavení *Blob Locator Tool* je společně s *2-D Single-View Vison Process a GPM* v příloze B.4c.

6.2.5 Krok pátý vytvoření Count Tool a Measurement Output Tool

Abychom mohli v programu vyčítat počet nalezených teček, je třeba vytvořit *Count Tool*. Po jeho vytvoření si v okně *Locator Tool* zvolíme nástroj jehož nalezené objekty budeme počítat, nastavíme tedy náš *Blob Locator Tool* a zadáme hodnotu Model ID. Počet teček v programu vyčítáme z *Vision registru*, aby se nám nalezený počet uložil do tohoto registru, je třeba vytvořit *Measurement Output Tool*. V tomto nástroji je možno obsadit deset hodnot, zvolíme si Value1 do něj se nám budou nahrávat hodnoty z nástroje *Count Tool* a nastavíme hodnoty, které chceme z *Count tool* vyčítat. Jediné hodnoty, které nám to dovolí nastavit, jsou *Num. Found*, tedy náš počet teček. Tímto je celý proces nastavení našeho *Vision procesu* dokončen.



Obrázek 6.5: *Použité nastavení v Count Tool a Measurement Output Tool*

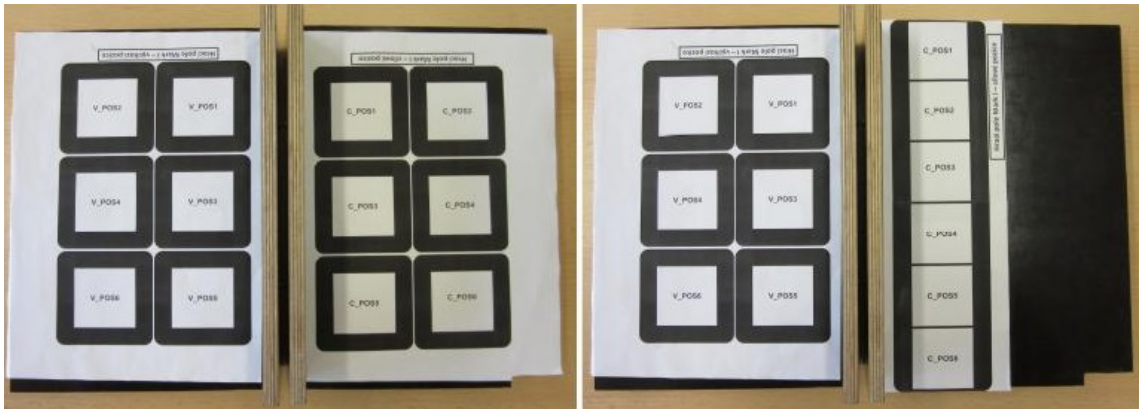
6.2.6 Krok šestý příprava před programování.

Nyní se dostáváme k poslednímu kroku před tvorbou programu. Jako první je nutné zkontrolovat, zda jsme nastavení na systém USER a máme nastaveny hodnoty Framů, které jsme si vytvořily.

Jako další krok je uložit si pozici pro focení, ve které jsme prováděli veškerá nastavení od kalibrace, až po tvorbu procesu. Polohu si dočasně uložíme do pozičního registru. Dále jak už jsem uvedl výše, je třeba vytvořit pozici pro uchopení kostky. Ručně nasměrujeme efektor robota do takové pozice, ze které budou později přepočítávány nové souřadnice, a efektor bude tímto způsoben polohován na nové polohy. Tuto pozici také uložíme například do pozičního registru. Nasměrování efektoru lze provést v systému JOINT nebo WORLD. Pozici však musíme nahrát v systému USER.

Nahrávání pozic do pozičních registrů však není příliš bezpečné, může se stát, že si hodnoty pozičního registru nechtěně přepíšeme jinou pozicí. To by pro nás znamenalo, že celý proces od kalibrace po nastavení Framů musíme provést znovu. Je tedy bezpečnější jednotlivé pozice nejprve uložit v nějakém pomocném programu, jako bod a až poté je příkazem přiřadit k pozičnímu registru. Takto máme ošetřeno, že pokud by se stalo, že informaci o poloze z PR ztratíme, není problém pozici si znovu naleznout pomocí naučeného bodu. Tyto body jsou pro správnou funkčnost vytvořeného programu důležité.

Pro úlohy jsem navrhl a sestavil dvě pole s vyznačenými výchozími a cílovými pozicemi kostek. Tyto pole jsou vloženy do dopravníku na opačné strany. Jsou opatřeny zarážkami, kterými jsem desky v dopravníku vycentroval tak, aby s krajem dopravníku svíraly pravý úhel a nedalo se s nimi jinak hýbat. Zároveň vymezují hrací prostor, kam robot hází kostku. Pole mají vyvýšené přepážky, které zabraňuji případnému přeskočení kostky. Před samotnou tvorbou jsem otestoval dosažitelnost jednotlivých pozic kostek na polích. U pole s výchozími pozicemi jsem kostky rozvrhl do dvou sloupců o třech řadách. U výchozích pozic s touto koncepcí není problém s dosažením krajních poloh, aniž by došlo k nějaké kolizi, nebo problému s dosažitelností. U pole s cílovými pozicemi jsem toto rozvržení realizovat nemohl. Zjistil jsem, že v krajní poloze by došlo k nárazu kamery do konstrukce buňky. Rozhodl jsem se, že jednotlivé kostky budou v jednom sloupci.

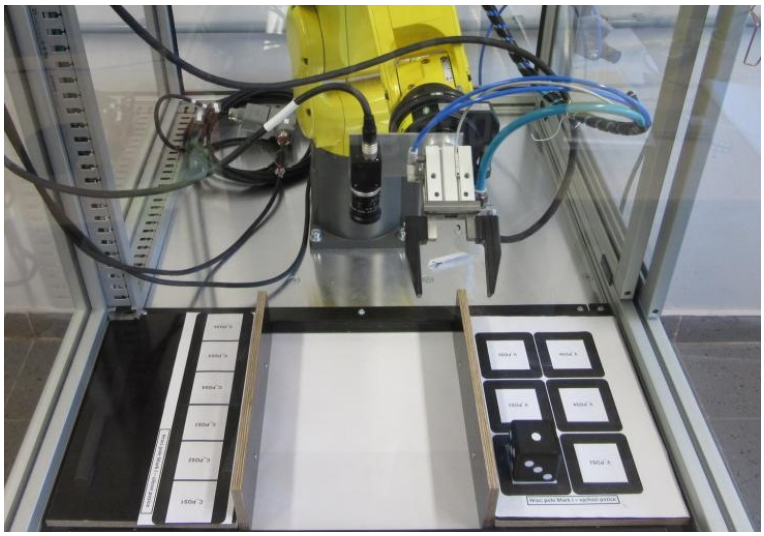


Obrázek 6.6: *Vlevo hrací pole s původním návrhem pozic, vpravo s upravenými cílovými pozicemi.*

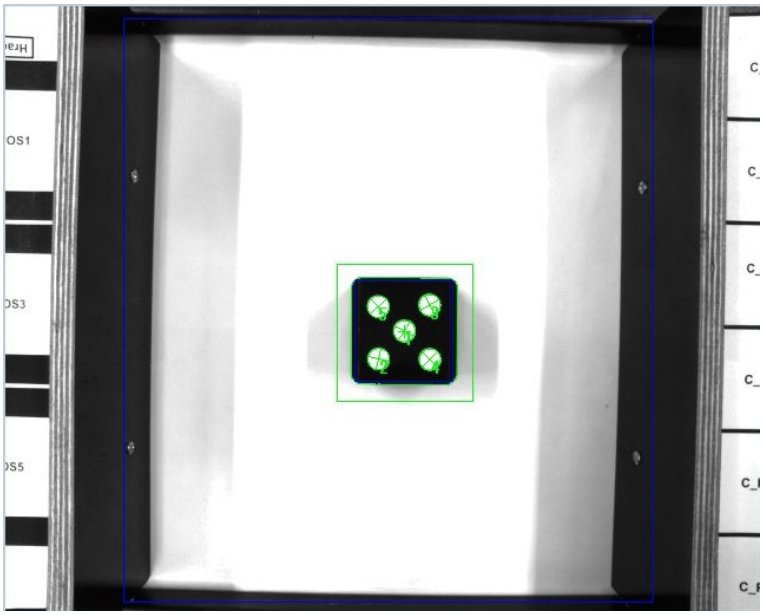
Tím jsem získal dostatečný prostor pro nasměrování efektoru na pozice kostek. Po otestování jsem se nesetkal s problémem dosažitelnosti nebo kolize.

6.3 Tvorba programu Kamera

Samotná úloha je rozvržena do dvou programových částí. První částí je hlavní tělo programu s názvem *CAM_ZIMA*, z něho je pak volán program obsluhující kameru s názvem *CAM_ZIMA_OBSLUHA*. Tyto programy si vytvoříme a následně do nich budeme psát náš kód. Stejně jako u úlohy pyramida vysvětlím principy programů, veškeré zdrojové kódy a obsahy registrů budou obsaženy v příloze B.5.



Obrázek 6.7: *Pracoviště připraveno pro spuštění programu.*



Obrázek 6.8: *Pohled kamery na pracovní plochu z fotící pozice.*

6.3.1 Program *CAM_ZIMA*

Tento program slouží jako hlavní tělo a využívá se pro spuštění úlohy. Program začíná nastavením robota do polohy, která se příkazem uloží do pozičního registru. Následuje inicializace registrů, které jsou později využity pro napolohování robota do první pozice na výchozím poli. Dalším krokem je skok do části programu, kde robot provede uchopení kostky na první pozici výchozího pole. Po uchopení se provede zavolání podprogramu *CAM_ZIMA_OBSLUHA* na obsluhu kamery. Po provedení tohoto podprogramu se pomocí skoku přemístí program do části, kde se provede zvýšení hodnoty registru určující cyklus. V závislosti na cyklu jsou nastaveny proměnné buď na pozice výchozího pole, nebo cílového. Tyto proměnné jsou pak přiřazeny v pozičních příkazech obslužné části programu, která má na starosti napolohování a manipulaci s kostkou. Tím jsem zajistil, že manipulaci s kostkou na výchozím i cílovém poli obstarává jedna část programu a není tam obsažena dvakrát, jednou s pozicemi výchozího pole a podruhé s pozicemi cílového pole. Takto program opakovaně vykonává manipulaci s kostkou, dokud nezaznamená, že součet hozených číslic překonal nastavenou hodnotu, aktuálně je to číslo 25. Poté kostku umístí na první pozici výchozího pole a program ukončí. Zdrojový kód v příloze B.6a.

6.3.2 Program *CAM_ZIMA_OBSLUHA*

Tento program má na starosti obsluhu kamery a je volán z programu *CAM_ZIMA*. Jako první se po jeho zavolání provede nasměrování do pozice *SAVE_POS*, dále pokračuje přemístěním do pozice, ve které kostku upustí, následně se přesune do pozice pro focení. Zde provede snímek pomocí příkazu VISION RUN_FIND. Dále příkazem *VISION GET_OFFSET*, vypočítá novou polohu kostky a provede aktualizaci vision registru, ze kterého získáme pomocí příkazu *R[21]=VR[1].MES[1]* počet nalezených teček a ten se nám uloží do zvoleného registru. Pokud kamera nevyhodnotí přítomnost objektu, program skočí na definovaný LBL. Po výpočtu nových souřadnic se robot nasměruje na pozici kostky a provede její uchopení, následně se vrací na pozici *SAFE_POS* a vrací se zpět do hlavního programu. A takto stále dokola dokud je program volán. Zdrojový kód v příloze B.6b.

7 Nezrealizované návrhy na úpravy

Možností pro rozvoj a návrh úloh je tu mnoho, rád bych uvedl pár příkladů, které mě později napadly a mohli by posloužit jako inspirace pro ostatní. První co mi přišlo na mysl, je spojení první a druhé úlohy do sebe, zachovala by se stavba pyramidy, která by nově mohla být tvořena ze šesti kostek. Ty by se umístily nahodile do definovaného prostoru, jediné co by bylo zapotřebí, aby každá kostka měla jiné číslo a čísla se neopakovala. Robot by si je potom v definovaném prostoru vyfotil, automaticky by určil polohy jednotlivých kostek a podle čísel například od nejmenší po největší umístil do pyramidy.

Jako druhá se nabízí úprava mé úlohy Kamera. Úprava by spočívala ve vytvoření perspektivní kalibrace s tím, že bychom na začátku neměli určenou pozici kostky pro start, ale bylo by ji třeba lokalizovat pomocí kamery. Dále by se úloha dala obohatit o Force Sensor, aby hlídal kolizi. Senzor by hlídal nevyžádané silové působení, na něhož by robot zareagoval a zastavil svůj pohyb.

Další úpravy by byly spíše mechanické a týkali by se mnou navržených hracích polí. Při jejich návrhu a konstrukci jsem hledět spíše na funkčnost než estetický dojem a tak by se dal vytvořit nový návrh a nechat například vytisknout na 3D tiskárně.

V průběhu tvorby několikrát padl pojem špatné světelné podmínky, v závislosti na nich mě napadlo vyzkoušet osvětlení přímo od objektivu, dalo by se tedy otestovat, jaký vliv by takovéto osvětlení mělo na výsledek.

8 Závěr

Vytvořil jsem dvě demo úlohy, které jsou provozu schopné ve výukových buňkách univerzitní laboratoře robotiky.

V první se zabývám offline programováním s podporou softwaru Roboguide, při tvorbě této úlohy jsem se seznámil s pracovištěm, které jsem si nadefinováním USERFramu a TOOLFramu připravil pro offline programování, dále jsem pak provedl připojení PC k reálnému robotu a pokračoval návrhem virtuálního pracoviště.

Pracoviště jsem navrhl tak, že je shodné s výukovou buňkou, k tomu jsem využil 3D model rámu buňky, který jsem sám navrhl v softwaru FreeCAD přesně dle jejich rozměrů. Následně jsem navrhl a offline naprogramoval jednoduchou manipulační úlohu, ve které jsem se naučil používat základní programové operace a příkazy.

Výslednou úlohu jsem exportoval do reálného robota a provedl test v online režimu. Při testu jsem postupoval obezřetně, nejdříve jsem v režimech T1 a T2 vyzkoušel dosažení pozic a nájezdové rychlosti. Nakonec jsem provedl test v automatickém cyklu. Úlohu jsem analogicky popsal v textové zprávě. Při návrhu a testování jsem se nesetkal s žádným větším problémem.

Dále jsem pokračoval druhou úlohou, ve které se zaměřuji na využití nově nainstalované kamery. Samotné úloze předcházely testy a experimenty s nastavením kamery, kdy jsem zkoumal, jak se různá nastavení projeví na výsledku. Dobral jsem se několika nedostatků, které byly potřeba před finální tvorbou úlohy odstranit.

Úpravy se týkaly zlepšení optických vlastností okolí a kostky, se kterou robot pracuje. Když byly tyto nedostatky odstraněny, zaměřil jsem se na tvorbu vision procesu. Vyzkoušel jsem si několik nástrojů a různá nastavení parametrů, abych mohl určit přijatelné nastavení. Samotný proces mi pak lokalizuje kostku a vyčítá počet teček na kostce, které reprezentují číslo.

Když jsem měl vision proces navrhnutý přišla na řadu programová tvorba, zde jsem využil znalosti získané v první úloze, které jsem aplikoval na úlohu s kamerou. Výslednou úlohu jsem pak otestoval v jednotlivých režimech, abych odzkoušel správnou funkčnost.

V průběhu realizace jsem se setkal hned s několika překážkami, které mě potkaly jak při tvorbě vision procesu, tak při tvorbě programové části. Velmi jsem se potýkal s nedostatkem prostoru v robotické buňce. Jednak mě limitoval při manipulaci stísněný prostor a další co mě omezovalo při manipulaci s robotem, jsou kabely vedoucí ke kameře a Force Sensoru. Návrh jsem musel uzpůsobit těmto omezením.

Dále jsem se potýkal s častým problémem singularity a dosažitelností poloh, to jsem nakonec vyřešil přepracováním pozice robota, kdy provádí focení, pozici jsem upravil tak, že jsem provedl několik testů a zhodnotil, která bude nejvíce vyhovovat.

Dále jsem byl nucen přepracovat rozvržení pozic na pozičních polích z důvodu kolize kamery a ochranného plexiskla. Nakonec jsem se se všemi omezeními a problémy vypořádal a navrhl jsem funkční úlohu s využitím kamery, jejíž návrh jsem opět analogicky zpracoval do textové zprávy.

Výsledkem jsou dvě provozuschopné demo úlohy s popsaným postupem tvorby, na kterých si studenti mohou vyzkoušet manipulaci s robotem a obsluhu kamery. Zároveň je možné úlohy využít pro prezentaci robotické laboratoře při dnech otevřených dveří.

Literatura

[1] FANUC | The Factory Automation Company - Fanuc. Object moved [online]. Dostupné z: https://www.fanuc.eu/cz/cs

[2] DVOŘÁK, Radim. „*Programování průmyslových robotů KUKA*.“ Brno, 2015. Bakalářská práce (Bp.). Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav výrobních strojů, systémů a robotiky. [online] [cit. 30.05.2020]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=102446

[3] CVEJN, Jiří. „*Paralelní kinematické struktury výrobních strojů*.“ Brno, 2011. Bakalářská práce (Bp.). Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav výrobních strojů, systémů a robotiky. [online] [cit. 30.05.2020]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=41313

[4] RUBIŠAR, Václav. „*Obrábění kompozičních materiálů pomocí robotů*.“ Brno, 2014. Diplomová práce (Dp.). Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav strojírenské technologie. [online] [cit. 30.05.2020]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=84063

[5] VANĚK, Josef. „*Aplikace robotů v automatizovaném testování řídicích jednotek automobilů*.“ Brno, 2011. Bakalářská práce (Bp.). Západočeská v Plzni, Fakulta elektrotechnická, Katedra aplikované elektrotechniky a telekomunikací. [online] [cit. 30.05.2020]. Dostupné z: https://otik.zcu.cz/bitstream/11025/18843/1/BP_Josef_Vanek_finalni%20verze.pdf

[6] SMUTNÝ, Vladimír. „*Robotika - Úvod do kinematiky*.“ Praha. Výukový text. Centrum strojového vnímání, Český institut informatiky, robotiky a kybernetiky (CIIRC), České vysoké učení technické v Praze, Fakulta elektrotechnická. [online] [cit. 30.05.2020]. Dostupné z: http://cmp.felk.cvut.cz/cmp/courses/ROB/roblec/kinematika-notecz.pdf

[7] FORMÁNEK, Josef. „*Podklady a grafická vizualizace k určení souřadnicových systémů výrobních strojů* .“ In: http://www.home.zcu.cz/ formanek [online].©ZČU v Plzni, poslední revize 12. 06. 2014 Praha.[cit. 16.4.2018]. Dostupné z: http://home.zcu.cz/~formanek/mmvyuka-arvt/Data/ivk-arvt-soubory/17-F.pdf

[8] HELLEBRAND, Petr. „*Výukový text - Kinematika robotů. In: http://skola.hellebrand.cz/* .“ Praha, Střední průmyslová škola na Proseku. [online] [cit. 30.05.2020]. Dostupné z: http://skola.hellebrand.cz/text1415/rt/RT-4-10-02_Robotika-2-Kinematika.pdf

[9] HELLEBRAND, Petr. „*Výukový text - Řízení robotů. In: http://skola.hellebrand.cz/* .“ Praha, Střední průmyslová škola na Proseku. [online] [cit. 30.05.2020]. Dostupné z: http://skola.hellebrand.cz/text1415/rt/RT-4-10-03_Robotika-3-Rizeni.pdf

[10] HELLEBRAND, Petr. „*Výukový text - Roboty a manipulátory. In: http://skola.hellebrand.cz/* .“ Praha, Střední průmyslová škola na Proseku. [online] [cit. 30.05.2020]. Dostupné z: http://skola.hellebrand.cz/text1415/rt/RT-4-10-01_Robotika-1-Uvod_a_cleneni.pdf

[11] Střední průmyslová škola strojírenská Kolín, Výukový text 3. ročníku oboru 23-41-M/001 Strojírenství„*AUTOMATIZACE A ROBOTIZACE I.* “[online]. [cit. 30.05.2020]. Dostupné z: http://www.sps-ko.cz/!podklady/ARO_prorok/Pr%C5%AFmyslov%C3%A9%20roboty.pdf

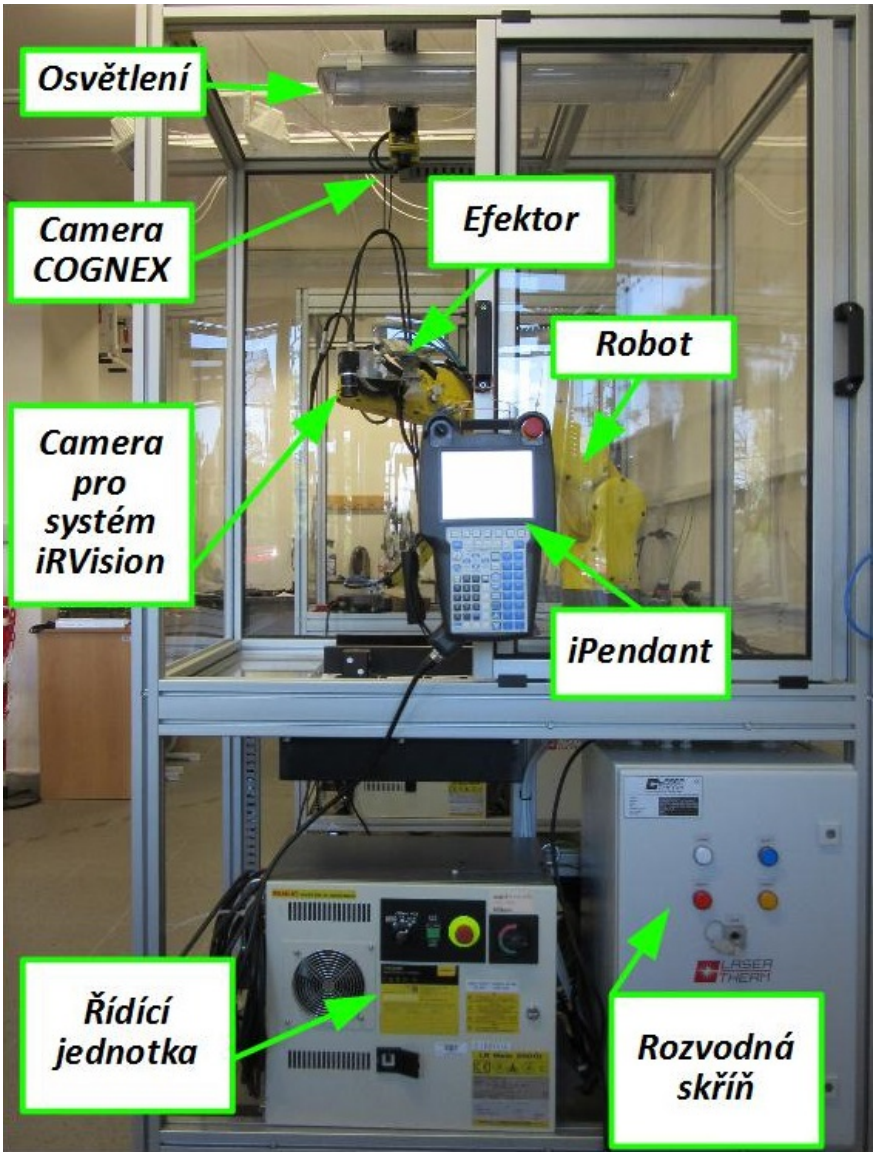
[12] KOHOUT, Luděk. „*Učební text VOŠ a SPŠ Kutná Hora - Roboty a manipulátory. In: http://www.edumat.cz/texty.php* .“ VOŠ, SPŠ a JŠ Kutná Hora. [online] [cit. 30.05.2020]. Dostupné z: http://www.edumat.cz/texty/Roboty_manipulatory.pdf

[13] Lase Therm„*Obsluha a programování průmyslových robotů Fanuc, příručka pro základní školení.*“ | lasertherm.cz

[14] ČERNOHORSKÝ, Josef „*Základy robotiky, Kinematika a topologie robotů.*“|Výukový text Technická Univerzita v Liberci

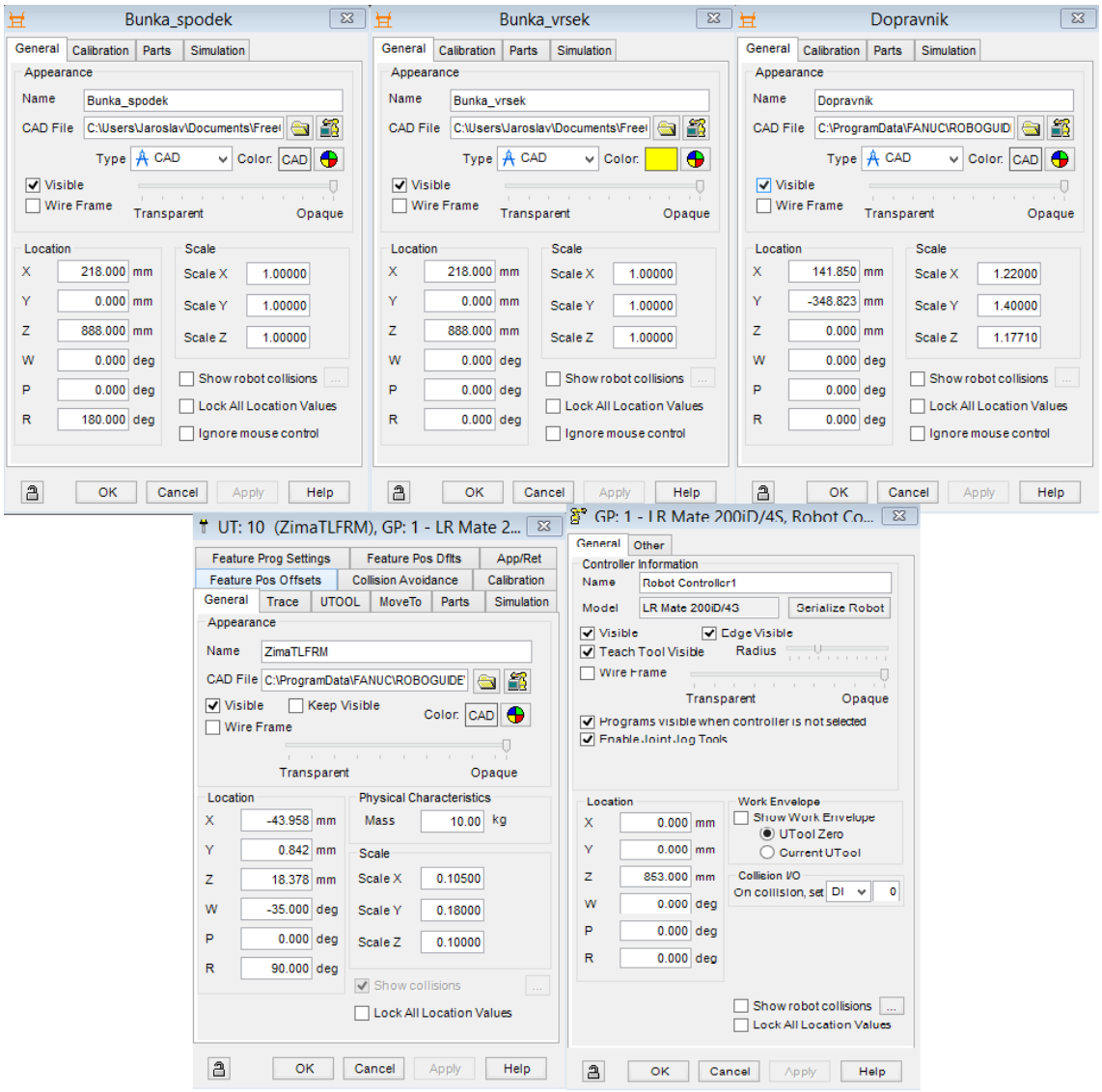
A Přílohy k úloze Pyramida

A.1 Popis součástí druhé výukové buňky



Obrázek A.1: *Obsah druhé výukové buňky.*

A.2 Hodnoty umístění a rozměrů těles v Roboguide



Obrázek A.2: *Rozměry a umístění jednotlivých objektů v Roboguide.*

A.3 Seznam a hodnoty využitých registrů

PR[22]	UF:F	UT:F		CONF:FUT	00-1
X	-60.000	mm	W	-.000	deg
Y	80.000	mm	P	.001	deg
Z	150.000	mm	R	-.000	deg

PR[23]	UF:F	UT:F		CONF:FUT	00-1
X	129.992	mm	W	-.000	deg
Y	79.994	mm	P	.001	deg
Z	89.994	mm	R	-.000	deg

PR[24]	UF:F	UT:F		CONF:FUT	00-1
X	49.990	mm	W	.025	deg
Y	-70.000	mm	P	.001	deg
Z	10.000	mm	R	.000	deg

PR[25]	UF:F	UT:F		CONF:FUT	00-1
X	220.000	mm	W	-.000	deg
Y	45.000	mm	P	.000	deg
Z	10.000	mm	R	.000	deg

PR[26]	UF:F	UT:F		CONF:FUT	000
X	0.000	mm	W	.000	deg
Y	150.000	mm	P	0.000	deg
Z	0.000	mm	R	0.000	deg

PR[27]	UF:F	UT:F		CONF:NDB	000
X	0.000	mm	W	0.000	deg
Y	70.000	mm	P	0.000	deg
Z	0.000	mm	R	0.000	deg

PR[28]	UF:F	UT:F		CONF:FUT	000
X	0.000	mm	W	0.000	deg
Y	0.000	mm	P	0.000	deg
Z	-50.000	mm	R	0.000	deg

PR[29]	UF:F	UT:F		CONF:FUT	00-1
X	0.000	mm	W	0.000	deg
Y	0.000	mm	P	0.000	deg
Z	0.000	mm	R	0.000	deg

(a) *Hodnoty pozičních registrů*

```
PR[ 22:Home_pos        ]=R
PR[ 23:Safe_pos        ]=R
PR[ 24:Pos1            ]=R
PR[ 25:Pos2            ]=R
PR[ 26:150Y            ]=R
PR[ 27:70Z             ]=R
PR[ 28:-50Z            ]=R
PR[ 29:Actual_pos      ]=R
```

(b) *Názvy pozičních registrů*

```
R[ 20:Citac_cyklu      ]=0
R[ 21:Citac_PICKUP     ]=0
R[ 22:FOR_PICKUP       ]=0
R[ 23:Citac_DROP       ]=0
R[ 24:Citac_DESTR      ]=0
R[ 25:FOR_DESTR        ]=0
```

(c) *Použité číselné registry*

Obrázek A.3: *Hodnoty a názvy registrů*

A.4 Hodnoty USERframe a TOOLframe

```
SETUP Frames
    User Frame          Direct Entry    1/7
    Frame Number: 9
 1  Comment:                 ZimaUSFRM
 2  X:                         272.258
 3  Y:                         -82.045
 4  Z:                        -299.184
 5  W:                          -0.000
 6  P:                           0.000
 7  R:                          -0.000
    Configuration:           F U T, 0, 0, 1
```

(a) *Hodnoty USERframe*

```
SETUP Frames
    Tool Frame          Direct Entry    1/7
    Frame Number:10
 1  Comment:                 ZimaTLFRM
 2  X:                         111.394
 3  Y:                           0.947
 4  Z:                         126.048
 5  W:                         179.760
 6  P:                         -55.796
 7  R:                        -179.646
    Configuration:           N D B, 0, 0, 0
```

(b) *Hodnoty TOOLframe*

Obrázek A.4: *Hodnoty USERframe a TOOLframe*

A.5 Programy úlohy Pyramida

```
                                                    1/27
   1:  CALL CHECK_BLOCK
   2:L  PR[22:Home_pos] 300mm/sec FINE
    :
   3:  LBL[1]
   4:  R[20:Citac_cyklu ]=0
   5:  R[21:Citac_PICKUP]=0
   6:  R[22:FOR_PICKUP]=0
   7:  R[23:Citac_DROP]=0
   8:  R[24:Citac_DESTR]=0
   9:  R[25:FOR_DESTR]=0
  10:  PR[35]=PR[29:Actual_pos]-
    :  PR[29:Actual_pos]
  11:  FOR R[20:Citac_cyklu ]=1 TO 6
  12:  IF (R[20:Citac_cyklu ]=1 OR
    :  R[20:Citac_cyklu ]=3 OR
    :  R[20:Citac_cyklu ]=5) THEN
  13:  R[70]=R[70]+1
  14:L  PR[23:Safe_pos] 300mm/sec
    :  CNT100
  15:  CALL PICKUP_BLOCK
  16:  ELSE
  17:  R[23:Citac_DROP]=
    :  R[23:Citac_DROP]+1
  18:L  PR[23:Safe_pos] 300mm/sec
    :  CNT100
  19:  CALL DROP_BLOCK
  20:  ENDIF
  21:  ENDFOR
  22:L  PR[23:Safe_pos] 100mm/sec
    :  CNT100
  23:  FOR R[20:Citac_cyklu ]=7 TO 12
  24:  CALL DESTR_PYR
  25:  ENDFOR
  26:  JMP LBL[1]
[End]
```

(a) *Program PYRAMIDA*

```
                                                    1/37
   1:L  PR[22:Home_pos] 100mm/sec FINE
    :
   2:  RO[1:chapadla open]=ON
   3:L  P[1] 300mm/sec FINE
   4:L  P[2] 150mm/sec FINE
   5:  WAIT    5.00(sec)
   6:  RO[2:chapadla_close]=ON
   7:  WAIT     .50(sec)
   8:  RO[1:chapadla open]=ON
   9:L  P[3] 300mm/sec FINE
  10:L  P[4] 300mm/sec FINE
  11:L  P[5] 150mm/sec FINE
  12:  WAIT    5.00(sec)
  13:  RO[2:chapadla_close]=ON
  14:  WAIT     .50(sec)
  15:  RO[1:chapadla open]=ON
  16:L  P[6] 300mm/sec FINE
  17:L  P[7] 300mm/sec FINE
  18:L  P[8] 150mm/sec FINE
  19:  WAIT    5.00(sec)
  20:  RO[2:chapadla_close]=ON
  21:  WAIT     .50(sec)
  22:  RO[1:chapadla open]=ON
  23:L  P[9] 300mm/sec FINE
  24:C  P[10]
    :   P[11] 300mm/sec FINE
  25:L  P[12] 300mm/sec FINE
  26:L  P[13] 30mm/sec FINE
  27:  WAIT     .50(sec)
  28:L  P[14] 300mm/sec FINE
  29:L  P[15] 300mm/sec FINE
  30:L  P[16] 30mm/sec FINE
  31:  WAIT     .50(sec)
  32:L  P[17] 300mm/sec FINE
  33:L  P[18] 300mm/sec FINE
  34:L  P[19] 30mm/sec FINE
  35:  WAIT     .50(sec)
  36:L  P[20] 300mm/sec FINE
[End]
```

(b) *Porgram CHECK_BLOCK*

Obrázek A.5: *Zdrojové kódy programu PYRAMIDA a CHECK_BLOCK*

```
                                1/14
  1:  IF (R[23:Citac_DROP]=1) THEN
  2:  PR[29:Actual_pos]=PR[25:Pos2]
  3:  ELSE
  4:  IF (R[22:FOR_PICKUP]=2) THEN
  5:  PR[29:Actual_pos]=PR[25:Pos2]+
   :  PR[27:70Z]
  6:  ELSE
  7:  PR[29:Actual_pos]=PR[25:Pos2]
  8:  PR[29,2:Actual_pos]=
   :  PR[29,2:Actual_pos]+35
  9:  PR[29:Actual_pos]=
   :  PR[29:Actual_pos]-PR[28:-50Z]
 10:  ENDIF
 11:  ENDIF
 12:  CALL CONTROL_GRIPPER
 13:  PR[29:Actual_pos]=
   :  PR[29:Actual_pos]-
   :  PR[29:Actual_pos]
[End]
```

(a) *Program PICKUP_BLOCK*

```
                                1/14
  1:  IF (R[23:Citac_DROP]=1) THEN
  2:  PR[29:Actual_pos]=PR[25:Pos2]
  3:  ELSE
  4:  IF (R[22:FOR_PICKUP]=2) THEN
  5:  PR[29:Actual_pos]=PR[25:Pos2]+
   :  PR[27:70Z]
  6:  ELSE
  7:  PR[29:Actual_pos]=PR[25:Pos2]
  8:  PR[29,2:Actual_pos]=
   :  PR[29,2:Actual_pos]+35
  9:  PR[29:Actual_pos]=
   :  PR[29:Actual_pos]-PR[28:-50Z]
 10:  ENDIF
 11:  ENDIF
 12:  CALL CONTROL_GRIPPER
 13:  PR[29:Actual_pos]=
   :  PR[29:Actual_pos]-
   :  PR[29:Actual_pos]
[End]
```

(b) *Program DROP_BLOCK*

```
                                1/26
  1:  IF (R[20:Citac_cyklu ] MOD 2=1)
   :  THEN
  2:  IF (R[20:Citac_cyklu ]=7) THEN
  3:  PR[29:Actual_pos]=PR[25:Pos2]
  4:  PR[29,2:Actual_pos]=
   :  PR[29,2:Actual_pos]+35
  5:  PR[29:Actual_pos]=
   :  PR[29:Actual_pos]-PR[28:-50Z]
  6:  ELSE
  7:  IF (R[20:Citac_cyklu ]=9) THEN
  8:  PR[29:Actual_pos]=PR[25:Pos2]+
   :  PR[27:70Z]
  9:  ELSE
 10:  PR[29:Actual_pos]=PR[25:Pos2]
 11:  ENDIF
 12:  ENDIF
 13:  ELSE
 14:  IF (R[20:Citac_cyklu ]=8 OR
   :  R[20:Citac_cyklu ]=10) THEN
 15:  PR[29:Actual_pos]=PR[24:Pos1]
 16:  R[24:Citac DESTR]=
   :  R[24:Citac_DESTR]+1
 17:  FOR R[25:FOR_DESTR]=2 DOWNTO
   :  R[24:Citac_DESTR]
 18:  PR[29:Actual_pos]=
   :  PR[29:Actual_pos]+PR[26:150Y]
 19:  ENDFOR
 20:  ELSE
 21:  PR[29:Actual_pos]=PR[24:Pos1]
 22:  ENDIF
 23:  ENDIF
 24:  CALL CONTROL_GRIPPER
 25:  PR[29:Actual_pos]=
   :  PR[29:Actual_pos]-
   :  PR[29:Actual_pos]
[End]
```

(c) *Program DESTR_PYR*

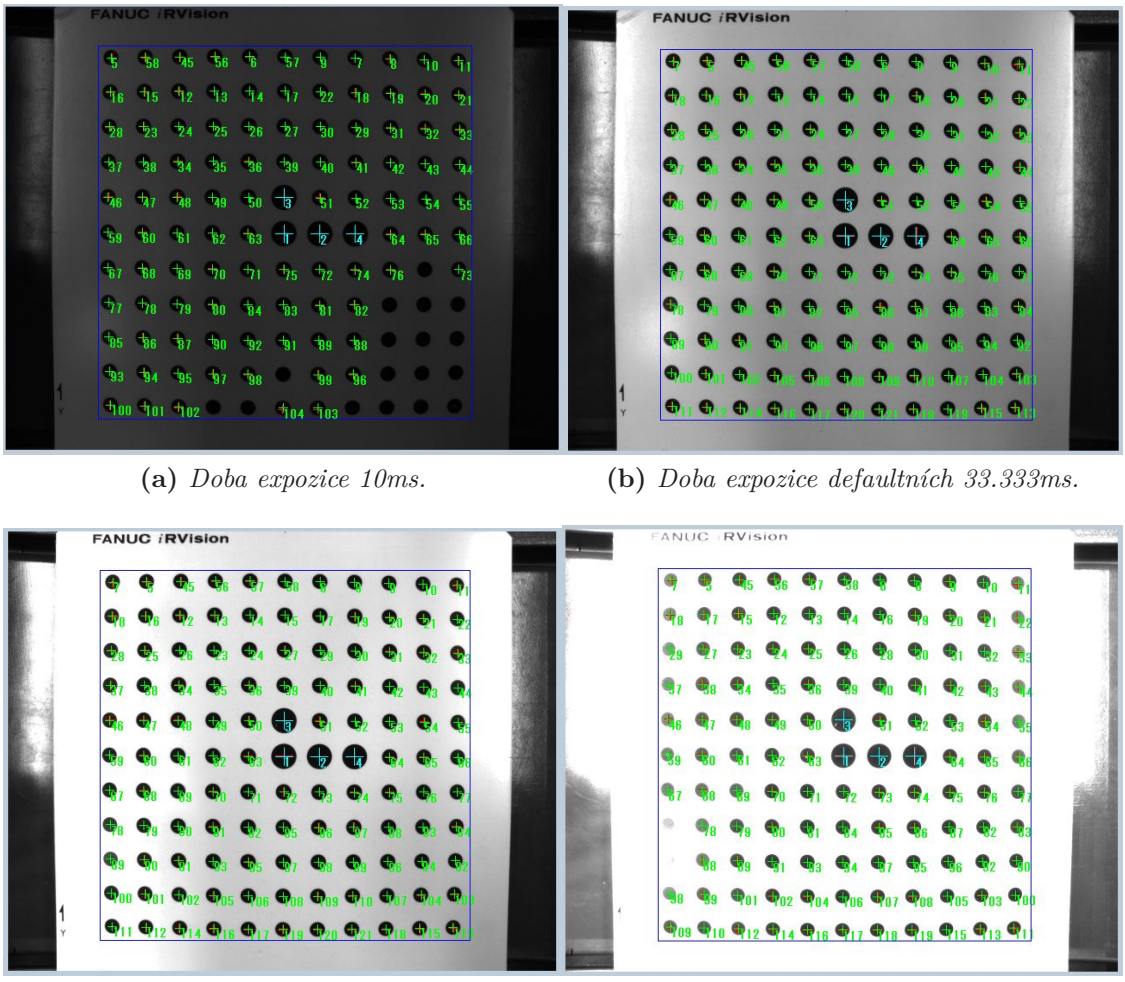
```
                                1/16
  1:  IF (R[20:Citac_cyklu ]=1 OR
   :  R[20:Citac_cyklu ]=3 OR
   :  R[20:Citac_cyklu ]=5 OR
   :  R[20:Citac_cyklu ]=7 OR
   :  R[20:Citac_cyklu ]=9 OR
   :  R[20:Citac_cyklu ]=11) THEN
 2:L  PR[29:Actual_pos] 300mm/sec
   :  FINE
 3:L  PR[29:Actual_pos] 30mm/sec FINE
   :  Offset,PR[28:-50Z]
 4:  WAIT    .80(sec)
 5:  RO[2:chapadla_close]=ON
 6:  WAIT    .50(sec)
 7:L  PR[29:Actual_pos] 300mm/sec
   :  FINE
 8:  ELSE
 9:L  PR[29:Actual_pos] 300mm/sec
   :  FINE
10:L  PR[29:Actual_pos] 30mm/sec FINE
   :  Offset,PR[28:-50Z]
11:  WAIT    .80(sec)
12:  RO[1:chapadla open]=ON
13:  WAIT    .50(sec)
14:L  PR[29:Actual_pos] 300mm/sec
   :  FINE
15:  ENDIF
[End]
```

(d) *Program CONTROL_GRIPPER*

Obrázek A.6: *Zdrojové kódy zbylých programu*

B Přílohy k úloze Kamera

B.1 Příklady nasnímaných obrazů pomocí kamery

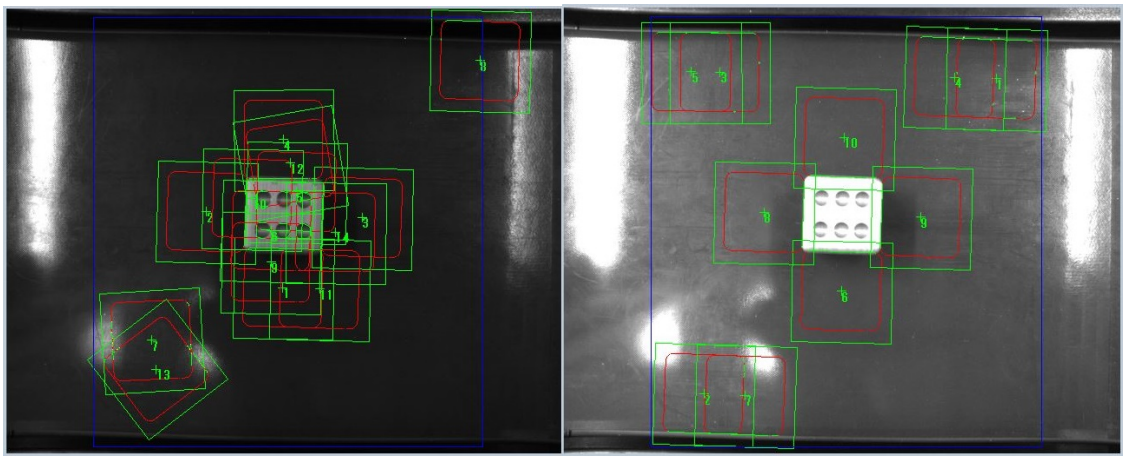


(a) *Doba expozice 10ms.* **(b)** *Doba expozice defaultních 33.333ms.*

(c) *Doba expozice 50ms.* **(d)** *Doba expozice 200ms.*

Obrázek B.1: *Vliv doby expozice na zobrazení mřížky a zaměření jejích bodů.*

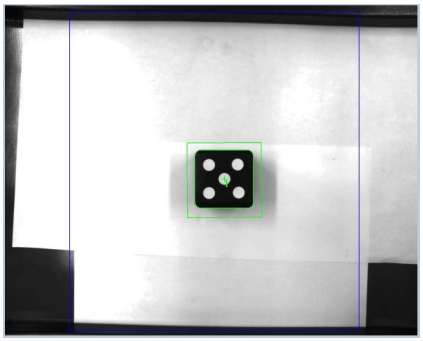
B.2 Špatné vyhodnocení kostky



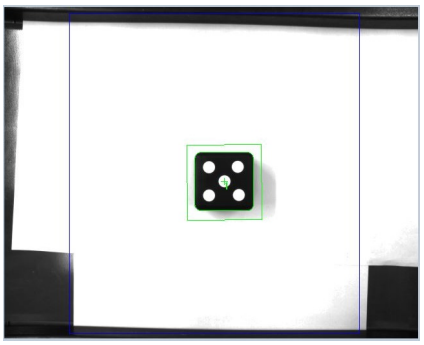
(a) *Expozice 50ms, socre 50% a contrast 10.* **(b)** *Expozice 150ms, socre 20% a contrast 35.*

Obrázek B.2: *Rozpoznání mnoho objektů.*

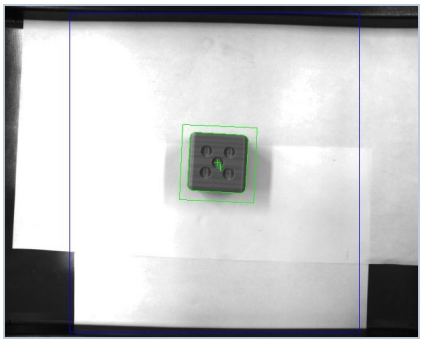
B.3 Vlivy doby expozice a absence denního světla na výsledný obraz



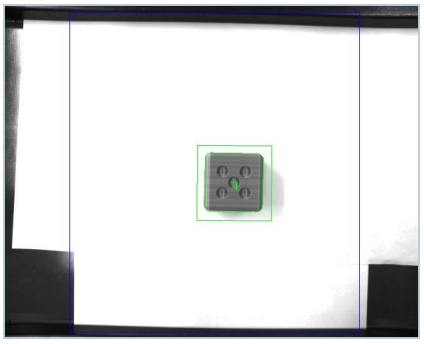
(a) *Doba expozice 45ms.*



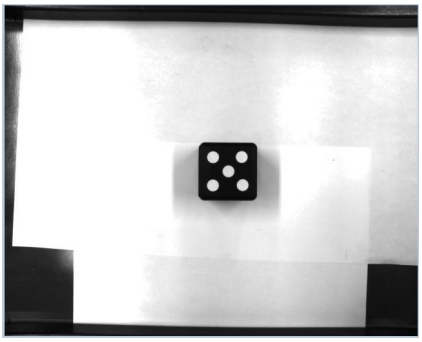
(b) *Doba expozice 60ms.*



(c) *Doba expozice 45ms.*



(d) *Doba expozice 60ms.*

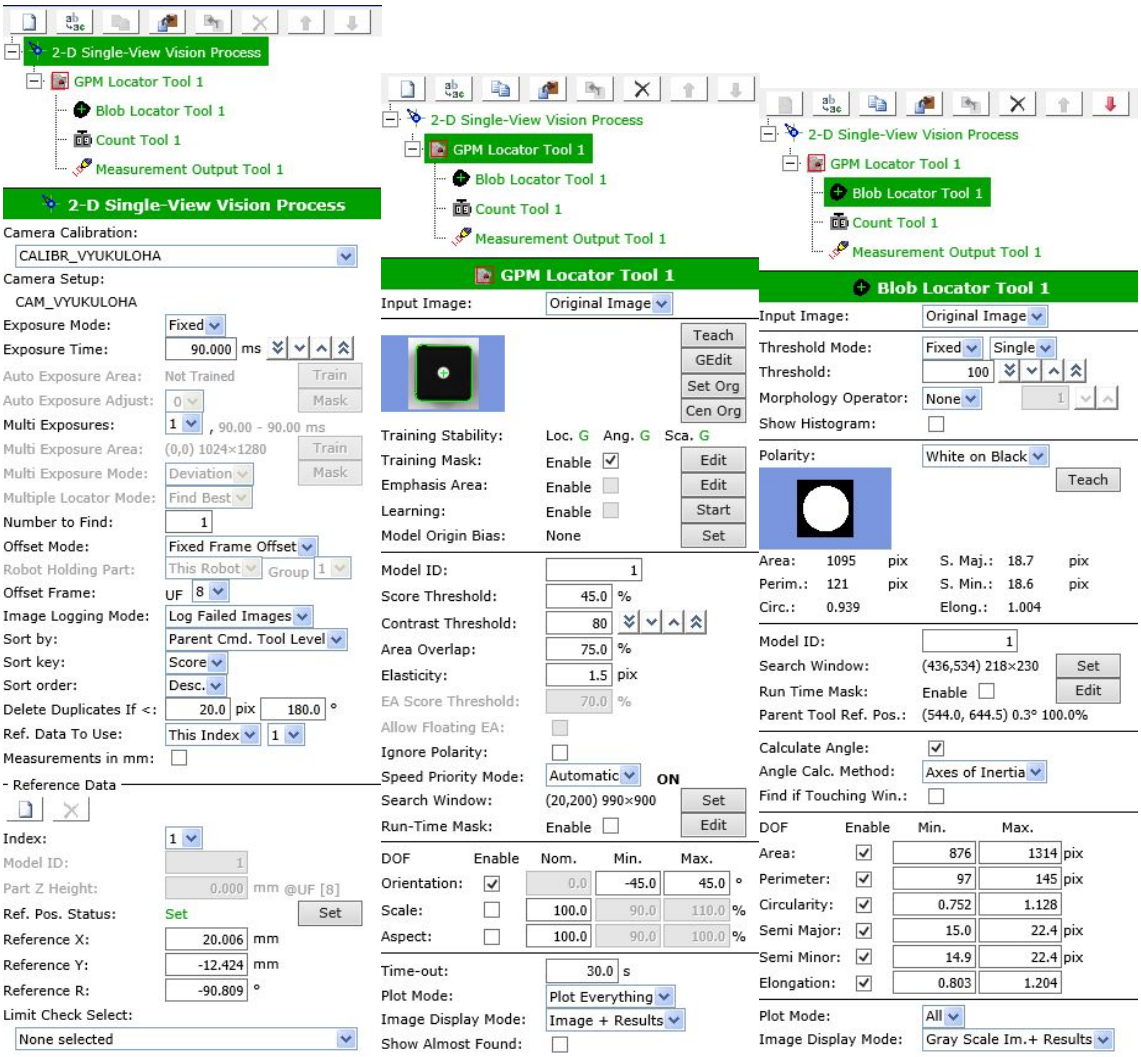


(e) *Doba expozice 60ms bez denního světla.*

(f) *Doba expozice 90ms bez denního světla.*

Obrázek B.3: *Vliv doby expozice při testování kostek.*

B.4 Konkrétní nastavení nástrojů pro lokaci kostky a teček



(a) *2-D Single-View Vison Process .* **(b)** *GPM Locator Tool.* **(c)** *Blob Locator Tool.*

Obrázek B.4: *Použitá nastavení 2-D Single-View Vison Process, GPM Locator Tool a Blob Locator Tool.*

B.5 Použité číselné a poziční registry a jejich hodnoty

PR[22:OFFSET_KOSTKA]=R		PR[32:V_POS3]=R	
PR[23:V/S_POS]=R		PR[33:V_POS4]=R	
PR[24:HOD]=R		PR[34:V_POS5]=R	
PR[25:SAFE_POS]=R		PR[35:V_POS6]=R	
PR[26:SNAP_POS]=R		PR[36:C_POS1]=R	
PR[27:REF_POS_KOSTKA]=R		PR[37:C_POS2]=R	
PR[28:V_POS]=R		PR[38:C_POS3]=R	
PR[29:C_POS]=R		PR[39:C_POS4]=R	
PR[30:V_POS1]=R		PR[40:C_POS5]=R	
PR[31:V_POS2]=R		PR[41:C_POS6]=R	
R[20:CYKL]=0			
R[21:NUM_FOUND]=0			
R[22:SET_PR]=0			
R[23:SOUCET]=0			
R[24:REF_SOUCET]=0			

(a) *Použité poziční a číselné registry s jejich názvy.*

PR[21] UF:F UT:F			CONF:FUT 000	
X	0.000	mm W	0.000	deg
Y	0.000	mm P	0.000	deg
Z	-30.000	mm R	0.000	deg

PR[22] UF:F UT:F			CONF:FUT 000	
X	79.839	mm W	-56.080	deg
Y	-340.348	mm P	.725	deg
Z	-57.202	mm R	-2.493	deg

PR[23] UF:F UT:F			CONF:FUT 000	
X	258.467	mm W	-56.081	deg
Y	-330.530	mm P	.733	deg
Z	93.743	mm R	-1.295	deg

PR[24] UF:F UT:F			CONF:FUT 000	
X	12.250	mm W	-53.942	deg
Y	-467.573	mm P	3.539	deg
Z	-40.894	mm R	-10.247	deg

PR[25] UF:F UT:F			CONF:FUT 001	
X	5.654	mm W	-.113	deg
Y	-3.929	mm P	-.033	deg
Z	-150.794	mm R	-.000	deg

PR[26] UF:F UT:F			CONF:FUT 001	
X	-2.497	mm W	-.115	deg
Y	3.260	mm P	-.033	deg
Z	-.085	mm R	-.001	deg

PR[27] UF:F UT:F			CONF:FUT 000	
X	79.839	mm W	-56.080	deg
Y	-340.348	mm P	.725	deg
Z	-57.202	mm R	-2.493	deg

PR[28] UF:F UT:F			CONF:FUT 001	
X	-120.001	mm W	-55.466	deg
Y	-321.765	mm P	-.169	deg
Z	90.880	mm R	-.724	deg

PR[29] UF:F UT:F			CONF:FUT 000	
X	258.467	mm W	-56.081	deg
Y	-330.530	mm P	.733	deg
Z	93.743	mm R	-1.295	deg

PR[30] UF:F UT:F			CONF:FUT 001	
X	-120.000	mm W	-55.466	deg
Y	-242.000	mm P	-.169	deg
Z	-22.000	mm R	-.724	deg

PR[31] UF:F UT:F			CONF:FUT 001	
X	-200.000	mm W	-55.466	deg
Y	-242.000	mm P	-.169	deg
Z	-22.000	mm R	-.724	deg

PR[32] UF:F UT:F			CONF:FUT 001	
X	-120.000	mm W	-55.466	deg
Y	-322.000	mm P	-.169	deg
Z	-22.000	mm R	-.724	deg

PR[33] UF:F UT:F			CONF:FUT 001	
X	-200.000	mm W	-55.466	deg
Y	-322.000	mm P	-.169	deg
Z	-22.000	mm R	-.724	deg

PR[34] UF:F UT:F			CONF:FUT 001	
X	-119.990	mm W	-55.468	deg
Y	-402.017	mm P	-.170	deg
Z	-22.000	mm R	-.724	deg

PR[35] UF:F UT:F			CONF:FUT 001	
X	-200.000	mm W	-55.466	deg
Y	-402.000	mm P	-.169	deg
Z	-22.000	mm R	-.724	deg

PR[36] UF:F UT:F			CONF:FUT 000	
X	258.000	mm W	-56.081	deg
Y	-200.000	mm P	.733	deg
Z	-20.000	mm R	-1.296	deg

PR[37] UF:F UT:F			CONF:FUT 000	
X	258.000	mm W	-56.081	deg
Y	-252.000	mm P	.733	deg
Z	-20.000	mm R	-1.296	deg

PR[38] UF:F UT:F			CONF:FUT 000	
X	258.000	mm W	-56.081	deg
Y	-304.000	mm P	.733	deg
Z	-20.000	mm R	-1.296	deg

PR[39] UF:F UT:F			CONF:FUT 000	
X	258.000	mm W	-56.081	deg
Y	-356.000	mm P	.733	deg
Z	-20.000	mm R	-1.296	deg

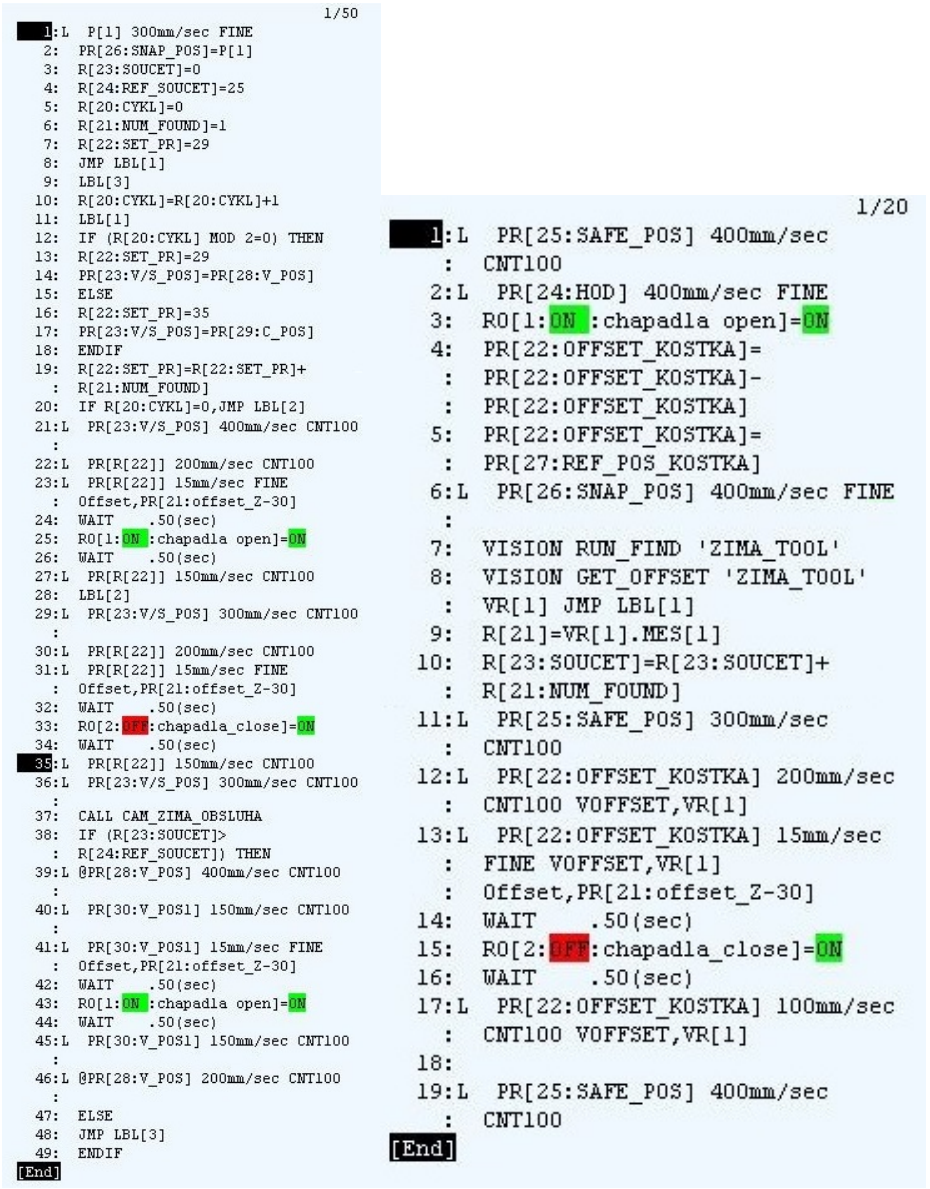
PR[40] UF:F UT:F			CONF:FUT 000	
X	258.000	mm W	-56.081	deg
Y	-408.000	mm P	.733	deg
Z	-20.000	mm R	-1.296	deg

PR[41] UF:F UT:F			CONF:FUT 000	
X	258.000	mm W	-56.081	deg
Y	-460.000	mm P	.733	deg
Z	-20.000	mm R	-1.296	deg

(b) *Obsah použitých pozičních registrů.*

Obrázek B.5: *Seznam registrů a jejich hodnot pro úloh Kameru.*

B.6 Programy úlohy Kamera



```
                                                                1/50
   1:L  P[1] 300mm/sec FINE
   2:   PR[26:SNAP_POS]=P[1]
   3:   R[23:SOUCET]=0
   4:   R[24:REF_SOUCET]=25
   5:   R[20:CYKL]=0
   6:   R[21:NUM_FOUND]=1
   7:   R[22:SET_PR]=29
   8:   JMP LBL[1]
   9:   LBL[3]
  10:   R[20:CYKL]=R[20:CYKL]+1
  11:   LBL[1]
  12:   IF (R[20:CYKL] MOD 2=0) THEN
  13:   R[22:SET_PR]=29
  14:   PR[23:V/S_POS]=PR[28:V_POS]
  15:   ELSE
  16:   R[22:SET_PR]=35
  17:   PR[23:V/S_POS]=PR[29:C_POS]
  18:   ENDIF
  19:   R[22:SET_PR]=R[22:SET_PR]+
    :   R[21:NUM_FOUND]
  20:   IF R[20:CYKL]=0,JMP LBL[2]
  21:L  PR[23:V/S_POS] 400mm/sec CNT100
    :
  22:L  PR[R[22]] 200mm/sec CNT100
  23:L  PR[R[22]] 15mm/sec FINE
    :   Offset,PR[21:offset_Z-30]
  24:   WAIT    .50(sec)
  25:   RO[1:ON :chapadla open]=ON
  26:   WAIT    .50(sec)
  27:L  PR[R[22]] 150mm/sec CNT100
  28:   LBL[2]
  29:L  PR[23:V/S_POS] 300mm/sec CNT100
    :
  30:L  PR[R[22]] 200mm/sec CNT100
  31:L  PR[R[22]] 15mm/sec FINE
    :   Offset,PR[21:offset_Z-30]
  32:   WAIT    .50(sec)
  33:   RO[2:OFF:chapadla_close]=ON
  34:   WAIT    .50(sec)
  35:L  PR[R[22]] 150mm/sec CNT100
  36:L  PR[23:V/S_POS] 300mm/sec CNT100
    :
  37:   CALL CAM_ZIMA_OBSLUHA
  38:   IF (R[23:SOUCET]>
    :   R[24:REF_SOUCET]) THEN
  39:L @PR[28:V_POS] 400mm/sec CNT100
    :
  40:L  PR[30:V_POS1] 150mm/sec CNT100
    :
  41:L  PR[30:V_POS1] 15mm/sec FINE
    :   Offset,PR[21:offset_Z-30]
  42:   WAIT    .50(sec)
  43:   RO[1:ON :chapadla open]=ON
  44:   WAIT    .50(sec)
  45:L  PR[30:V_POS1] 150mm/sec CNT100
    :
  46:L @PR[28:V_POS] 200mm/sec CNT100
    :
  47:   ELSE
  48:   JMP LBL[3]
  49:   ENDIF
[End]
```

(a) *Program CAM_ZIMA*

```
                                                                1/20
   1:L  PR[25:SAFE_POS] 400mm/sec
    :   CNT100
   2:L  PR[24:HOD] 400mm/sec FINE
   3:   RO[1:ON :chapadla open]=ON
   4:   PR[22:OFFSET_KOSTKA]=
    :   PR[22:OFFSET_KOSTKA]-
    :   PR[22:OFFSET_KOSTKA]
   5:   PR[22:OFFSET_KOSTKA]=
    :   PR[27:REF_POS_KOSTKA]
   6:L  PR[26:SNAP_POS] 400mm/sec FINE
    :
   7:   VISION RUN_FIND 'ZIMA_TOOL'
   8:   VISION GET_OFFSET 'ZIMA_TOOL'
    :   VR[1] JMP LBL[1]
   9:   R[21]=VR[1].MES[1]
  10:   R[23:SOUCET]=R[23:SOUCET]+
    :   R[21:NUM_FOUND]
  11:L  PR[25:SAFE_POS] 300mm/sec
    :   CNT100
  12:L  PR[22:OFFSET_KOSTKA] 200mm/sec
    :   CNT100 VOFFSET,VR[1]
  13:L  PR[22:OFFSET_KOSTKA] 15mm/sec
    :   FINE VOFFSET,VR[1]
    :   Offset,PR[21:offset_Z-30]
  14:   WAIT    .50(sec)
  15:   RO[2:OFF:chapadla_close]=ON
  16:   WAIT    .50(sec)
  17:L  PR[22:OFFSET_KOSTKA] 100mm/sec
    :   CNT100 VOFFSET,VR[1]
  18:
  19:L  PR[25:SAFE_POS] 400mm/sec
    :   CNT100
[End]
```

(b) *Program CAM_ZIMA_OBSLUHA*

Obrázek B.6: *Zdrojový kód programu CAM_ZIMA a CAM_ZIMA_OBSLUHA*

C Obsah přílohy

- Spodní i vrchní část buňky ve formátu STL
- Kompletní projekt s úlohou Pyramida
- Program CAM_ZIMA a CAM_ZIMA_OBSLUHA ve formátu TP