ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA STROJNÍ

Studijní program: B2301 Strojní inženýrství

Studijní obor: Dopravní a manipulační technika

BAKALÁŘSKÁ PRÁCE

Řízení robotických manipulátorů na základě obrazové informace

Autor: **Václav MAŠEK**

Vedoucí práce: **Ing. Roman ČERMÁK, PhD.**

Akademický rok 2018/2019

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta strojní
Akademický rok: **2018/2019**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Václav MAŠEK**
Osobní číslo:	**S16B0090P**
Studijní program:	**B2301 Strojní inženýrství**
Studijní obor:	**Dopravní a manipulační technika**
Název tématu:	**Plánování trajektorie robotického ramene na základě obrazové informace.**
Zadávající katedra:	**Katedra konstruování strojů**

Zásady pro vypracování:

Základní požadavky:
Proveďte rešerši na trhu existujících řešení. Navrhněte a popište konfiguraci testovacího pracoviště, popište řešenou úlohu. Na základě literárních zdrojů vytvořte matematický model robota a popište teoretické základy pro řešení problému. Sestavte simulační model v SW MATLAB/Simulink a proveďte potřebné simulace. Navrhněte snímání obrazu s využitím low-cost komponent kompatibilních s MATLABem. Na základě získané vizuální informace naprogramujte pohyb robotického ramene pro danou úlohu.

Základní technické údaje:
Technické parametry jsou uvedeny v příloze zadání.

Osnova bakalářské práce:

1. Rešerše.
2. Teoretický popis problému.
3. Návrh a popis testovacího pracoviště, popis komponent.
4. Matematický a simulační model, ověření na testovací úloze.
5. Zhodnocení práce, závěr.

Rozsah grafických prací:	**dle potřeby**
Rozsah kvalifikační práce:	**30-40 stran A4**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam odborné literatury:

SICILIANO, B., KHATIB, O. *Springer Handbook of Robotics*. **Berlin: Springer, 2008**

CORKE, P. *Robotics, Vision and Control*. **Berlin: Springer, 2011**

Podkladový materiál, výkresy, katalogy, apod. poskytnuté zadavatelem úkolu.

Vedoucí bakalářské práce:	**Ing. Roman Čermák, Ph.D.** Katedra konstruování strojů
Konzultant bakalářské práce:	**Ing. Roman Čermák, Ph.D.** Katedra konstruování strojů

Datum zadání bakalářské práce:	**16. října 2018**
Termín odevzdání bakalářské práce:	**24. května 2019**

Západočeská univerzita v Plzni * Fakulta strojní *

L.S.

Doc. Ing. Milan Edl, Ph.D.
děkan

Doc. Ing. Václava Lašová, Ph.D.
vedoucí katedry

V Plzni dne 16. října 2018

Prohlášení o autorství

Předkládám tímto k posouzení a obhajobě bakalářskou práci, zpracovanou na závěr studia na Fakultě strojní Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů, uvedených v seznamu, který je součástí této bakalářské práce.

V Plzni dne: ……. . . . . . .

podpis autora

Poděkování:

Děkuji vedoucímu své bakalářské práce, Ing. Romanu Čermákovi, PhD. za cenné rady vedoucí k vypracování této bakalářské práce a za přínosné připomínky.

Dále bych rád poděkoval svému otci, Ing. Milanu Maškovi, za pečlivé pročtení textu a provedenou jazykovou korekturu.

ANOTAČNÍ LIST BAKALÁŘSKÉ PRÁCE

AUTOR	**Příjmení** Mašek	**Jméno** Václav	
STUDIJNÍ OBOR	B2301 „Dopravní a manipulační technika“		
VEDOUCÍ PRÁCE	**Příjmení (včetně titulů)** Ing. Čermák, Ph.D.	**Jméno** Roman	
PRACOVIŠTĚ	ZČU - FST - KKS		
DRUH PRÁCE	~~**DIPLOMOVÁ**~~	**BAKALÁŘSKÁ**	**Nehodící se škrtněte**
NÁZEV PRÁCE	Plánování trajektorie robotického ramene na základě obrazové informace.		

FAKULTA	strojní		**KATEDRA**	KKS		**ROK ODEVZD.**	2019

POČET STRAN (A4 a ekvivalentů A4)

CELKEM	82		**TEXTOVÁ ČÁST**	59		**GRAFICKÁ ČÁST**	23

STRUČNÝ POPIS (MAX 10 ŘÁDEK) **ZAMĚŘENÍ, TÉMA, CÍL POZNATKY A PŘÍNOSY**	Práce shrnuje historii a současná řešení průmyslových robotů a zpracování obrazu. Dále je představen vhodný SW a HW pro tuto práci. Součástí je taktéž teorie mechaniky průmyslových manipulátorů. Praktická část se zabývá návrhem a implementací algoritmu pro zpracování obrazu, modelování a řízení robotů. Součástí praktické části je též návrh konstrukce magnetických uchopovačů pro dva z těchto robotů.
KLÍČOVÁ SLOVA **ZPRAVIDLA JEDNOSLOVNÉ POJMY, KTERÉ VYSTIHUJÍ PODSTATU PRÁCE**	robot, zpracování obrazu, inverzní kinematika, průmyslové manipulátory, matlab, Denavit - Hartenbergovy parametry, koncové efektory

SUMMARY OF BACHELOR SHEET

AUTHOR	**Surname** Mašek	**Name** Václav	
FIELD OF STUDY	B2301 „Transport Vehicles and Handling Machinery“		
SUPERVISOR	**Surname (Inclusive of Degrees)** Ing. Čermák, Ph.D.	**Name** Roman	
INSTITUTION	ZČU - FST - KKS		
TYPE OF WORK	~~**DIPLOMA**~~	**BACHELOR**	**Delete when not applicable**
TITLE OF THE WORK	Trajectory planning of a robotic arm based on visual information.		

FACULTY	Mechanical Engineering		**DEPARTMENT**	Machine Design		**SUBMITTED IN**	2019

NUMBER OF PAGES (A4 and eq. A4)

TOTALLY	82		**TEXT PART**	59		**GRAPHICAL PART**	23

BRIEF DESCRIPTION **TOPIC, GOAL, RESULTS AND CONTRIBUTIONS**	Bachelor thesis summarizes history of robotic manipulators and image processing, including present industrial solutions for Vision - Based Control. Except that, thesis includes overview of suitable software and hardware. Theory of serial manipulators mechanics is also included. Practical part of thesis is based on design and implementation of image processing algorithm, creating mathematical models of three types of robots. In the last part were designed magnetic robot end effectors for two of used robots.
KEY WORDS	robot, image processing, inverse kinematics, industrial manipulators, matlab, Denavit - Hartenberg parameters, end effectors

Obsah

Seznam obrázků

Seznam Tabulek

Seznam příloh

Použitý software

MATLAB R2013a

Autodesk Inventor Professional 2019

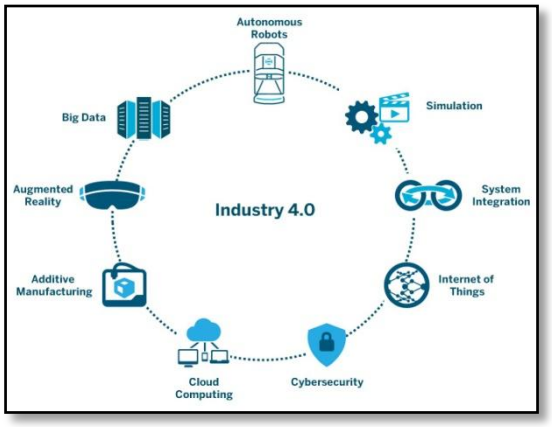
Použité zkratky a symboly

Zkratka	Význam	Český ekvivalent (je-li k dispozici)
2/2,5/3D	2/2,5/3 Dimensional	2/2,5/3 Rozměrný
CERN	Européen pour la recherche nucléaire	Evropská organizace pro jaderný výzkum
CMOS	Complementary Metal-Oxide Semiconductor	Doplňkový polovodič na bázi kovu a oxidu
CPU	Central Processing Unit	Processor
DoF	Degree of Freedom	Stupeň volnosti
GPIO	General - Purpose input/output	Univerzální vstup/výstup
GPU	Graphics Processing Unit	Grafický procesor
HDMI	High-Definition Multi-media Interface	Multimediální rozhraní s vysokým rozlišením
IoT	Internet of Things	Internet věcí
MSER	Maximally Stable Extremal Regions	Maximálně stabilní extrémní oblast
MRI	Magnetic Resonance Imaging	Magnetická rezonance
PC	Personal Computer	Osobní počítač
PLC	Programmable logic controller	Programovatelný logický automat
PWM	Pulse Width Modulation	Pulzně šířková modulace
SD	Secure Digital	Paměťová karta
USB	Universal Seriál Bus	Univerzální seriová sběrnice
VGA	Video Graphics Array	Grafické videopole

1. Úvod

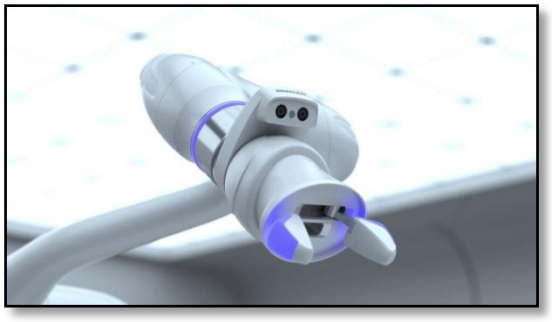
Robotické manipulátory hrají již několik desetiletí svoji nezastupitelnou roli ve výrobních závodech, kde postupně nahrazují lidské pracovníky v činnostech, které vyžadují vysokou přesnost, fyzickou zdatnost, případně jde o rutinní, neustále se opakující činnost nebo činnost v rizikovém či zdraví nebezpečném prostředí.

Na počátku 21. století došlo k nástupu takzvané 4. průmyslové revoluce. Ta se vyznačuje digitalizací, masivním využitím cloudového ukládání a zpracování dat včetně tzv. velkých dat (big data) v jednotkách petabytů a vyšších, aditivní výrobou, rozšířením umělé inteligence, prediktivní údržbou, aplikací virtuální a rozšířené reality, aplikacemi pokročilé senzoriky a vznikem tzv. kyberfyzikálních systémů, mezi něž patří i tzv. autonomní a kognitivní robotika. [28]



Obrázek 1 - Prvky 4. průmyslové revoluce [22]

Kognitivní roboti jsou osazeni senzory, pomocí kterých jsou schopni vnímat svoje okolí a reagovat na něj. Jde jak o senzory pro samotnou činnost robota (např. rozpoznávání výrobku a správnou koordinaci činnosti), tak i o senzory pro bezpečnost (t. j. pro kolaborativní robotiku) či o senzory pro údržbu (t. j. pro činnost jednotlivých subsystémů). Jedním z často využívaných senzorů bývá kamera, která snímá pracovní prostor robota, většinou ve viditelném nebo v tzv. blízkém infračerveném spektru.



Obrázek 2 - Aplikace robotického vidění [14]

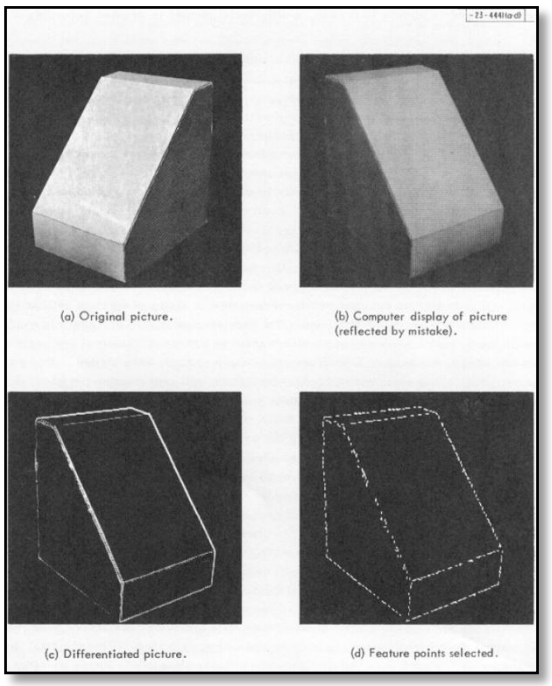
Kamerové systémy mohou být implementovány v robotickém pracovišti přímo z výroby, popřípadě mohou být nainstalovány dodatečně. Dodatečnou implementací kamer do systému lze rozšířit pole působnosti robotů starších generací o nové funkcionality a zvýšit tím i jejich flexibilitu. Robotické vidění navíc s výhodou funguje i jako další systém zpětné vazby.

Profesionální řešení systémů, které poskytují renomovaní výrobci, jsou často velice nákladná a výrazně svými parametry překračují požadavky na ně kladené jejich uživateli. Z těchto důvodů získává na atraktivitě aplikace levného a jednoduchého systému, fungujícímu

na principu levných kamer připojených přes USB rozhraní. Tato práce se zabývá návrhem systému využívajícího nízkonákladových (low-cost) komponent. Nutno zdůraznit, že periferie pracující přes rozhraní USB většinou nesplňují průmyslové nároky na rychlost zpracování dat, a proto je celá práce realizována především pro demonstrační a výukové účely. Přiměřeně tomu je přizpůsoben i ostatní použitý hardware i software.

1.1.Historie robotického vidění a zpracování obrazu

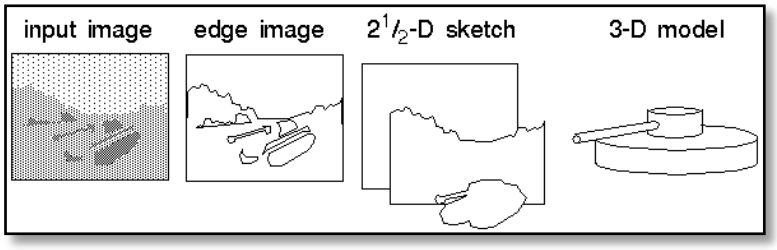
Za prvního autora softwaru na rozpoznávání obrazu je považován Larry Roberts, který se později angažovat například i jako jeden z vývojářů celosvětové internetové sítě. Ten se v rámci své dizertační práce na Massachusettském technologickém institutu (MIT) v roce 1960 věnoval výzkumu metod extrakce 3D dat z 2D obrazů různých mnohostěnů. [21]



Obrázek 3 - Zpracování obrazu v práci Larryho Robertse [41]

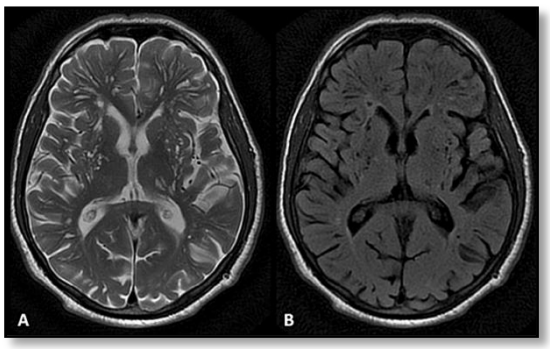
Dalším milníkem v oblasti počítačového vidění byla práce Davida Marra v roce 1978. David Marr byl neurovědec, který k problematice zpracování obrazu přistupoval na základě vizuálního vnímání světa lidmi a okolí podobně vnímajícími organismy. Spíše než o přesný algoritmus ve výše uvedené práci jde o vypracování modelu, popisujícího postup návrhu softwaru pro zpracování obrazu. [21]

V první fázi dle Marra měly být ve scéně detekovány hrany, přechody, rohy a další a měl být získán základní 2D nástin obrazu. V druhé, navazující fázi měl být pomocí stereoobrazu získán tzv. 2,5D obraz, ve kterém již měla být patrná hloubka obrazu, prostorové hrany apod. Posledním krokem mělo být kompletní rozpoznání objektů ve scéně a vytvoření jejich 3D modelů. [21]



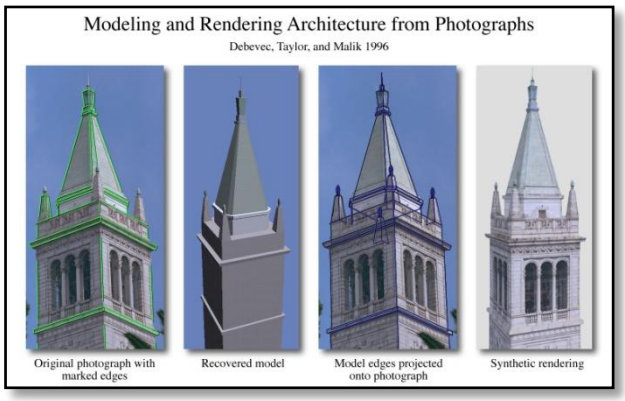
Obrázek 4 - Aplikace paradigmatu Davida Marra [34]

V roce 1979 získal Sir Godfrey N. Hounsfield společně s profesorem Allanem McLeodem Cormackem Nobelovu cenu za fyziologii a lékařství za vynález Výpočetní tomografie (Computed Axial Tomography). V 80. letech pak bylo dále pokračováno v rozvoji aplikací počítačového zpracování obrazu v lékařství. [32]



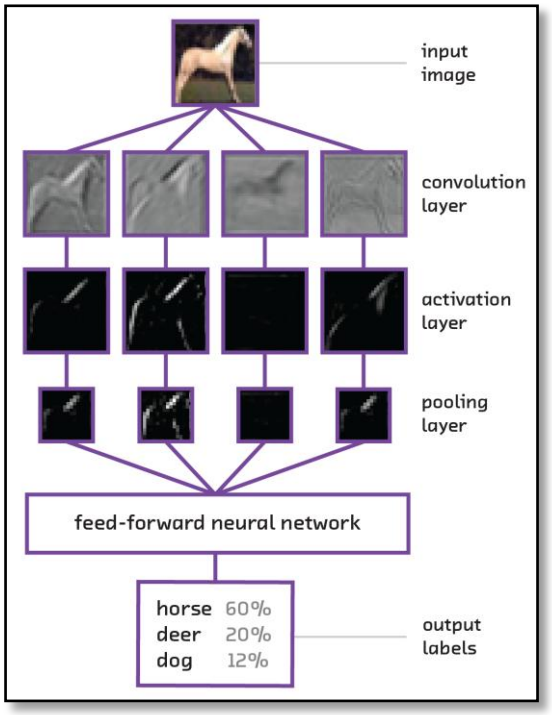
Obrázek 5 - Sken získaný pomocí výpočetní tomografie [27]

V 90. letech došlo k rozvoji technik počítačového zpracování obrazu Image - Based Modeling and Rendering, které umožňují vytváření 3D modelu ze souboru 2D obrazů a zároveň fotorealistické renderování scén. [49]



Obrázek 6 - Image Based Modeling and Rendering [10]

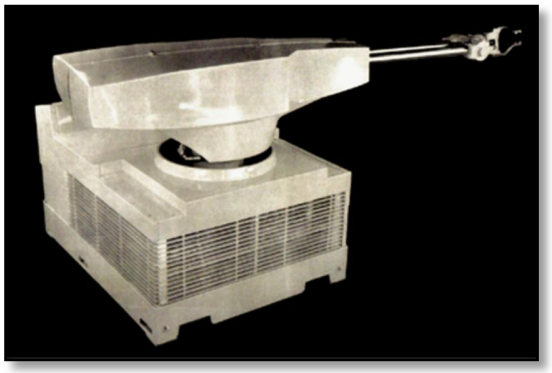
Ve 21. století začaly klasickým metodám zpracování obrazu, založených na geometrických vlastnostech obrazu, konkurovat metody strojového učení a tzv. neuronové sítě. K nárůstu jejich užívání došlo především z důvodu vyššího výpočetního výkonu dnešní počítačové techniky umožňujícího zpracování většího množství tzv. učebních dat. [49]



Obrázek 7 - Zpracování obrazu neuronovou sítí [20]

1.2. Průmyslové roboty a manipulátory

Počátky činnosti, kterou by bylo možno nazvat robotikou, lze datovat dlouho do historie. Za skutečně prvního průmyslového robota je považován robot Unimate, kterého si v roce 1954 nechal patentovat George Devol, spoluzakladatel společnosti Unimation. První robot této společnosti byl instalován v roce 1961 v automobilce General Motors, kde byl používán k vyjímání výrobků z licí formy. [47]

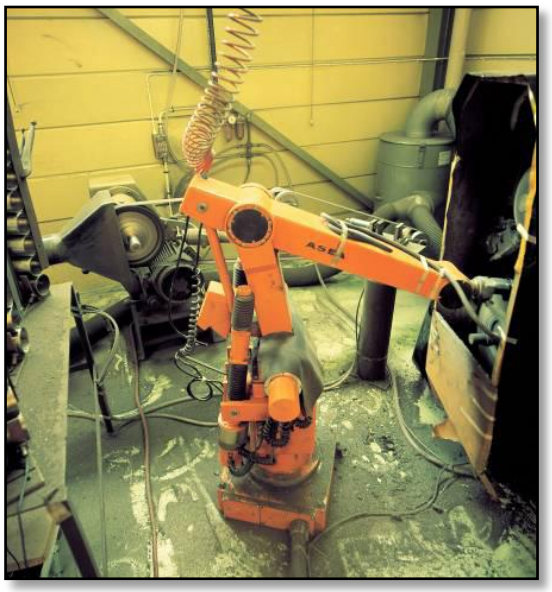


Obrázek 8 – Robot Unimate [50]

Dalším významným počinem v konstrukci robotů byla tzv. Stanfordská paže (Stanford Arm), sestrojená v roce 1969 Victorem Scheinmanem, studentem strojního inženýrství. Tento robot měl šest stupňů volnosti, ale na rozdíl od dnes obvyklé konstrukce šestiosých robotů neměl šest rotačních vazeb, ale pět rotačních a jednu posuvnou. [47]

Obrázek 9 - Stanfordská paže [48]

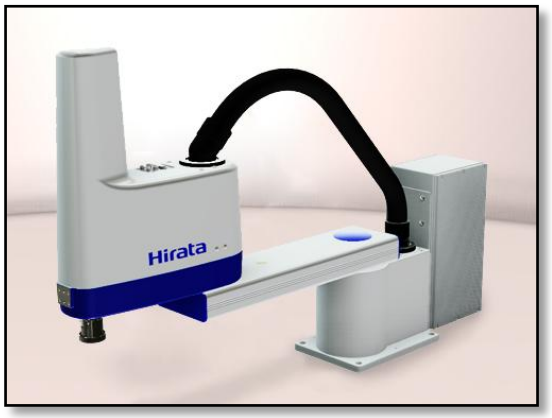
V roce 1973 společnost ASEA (dnešní ABB Group) představila robota IRB-6. Jednalo se o prvního robota řízeného mikropočítačem. Tento robot se již dokázal po zadané trajektorii pohybovat s dostatečnou přesností, aby mohl vykonávat činnosti typu obráběcí či svařovací operace. [47]



Obrázek 10 - ASEA IRB – 6 [3]

V roce 1978 sestrojil univerzitní profesor Hiroshi Makino robotický manipulátor SCARA (Selective Compliance Assembly Robot Arm). Toto zařízení má čtyři stupně volnosti (tři rotační a jednu posuvnou vazbu) a je dodnes široce využívané při paletování či montáži elektronických a mechanických zařízení.

Robot vyžaduje malé zástavbové rozměry a vyznačuje se vysokou dosažitelnou rychlostí pohybů bez újmy na opakovatelnosti, t. j. bez nežádoucí odchylky koncového efektoru během pracovního cyklu. [47]



Obrázek 11 - Robot typu SCARA [19]

Dnes jsou nejčastěji používanými průmyslovými roboty sériové manipulátory se šesti stupni volnosti, jako je například robot na obrázku č. 12. Jejich výhodou je relativně velký pracovní prostor a fakt, že jejich stupně volnosti přibližně odpovídají rozložení stupňů volnosti lidské paže. Proto mohou snadno nahrazovat činnosti vykonávané lidmi a konat pohyb i po velmi složitých trajektoriích.

Obrázek 12 - Moderní robot s 6 stupni volnosti [37]

Dalšími často využívanými roboty jsou paralelní manipulátory typu robota na obrázku č. 13. Vyznačují se vysokou pracovní rychlostí a nízkou setrvačností pohyblivých částí manipulátoru. Za nevýhodu lze považovat jejich velice omezený pracovní prostor.

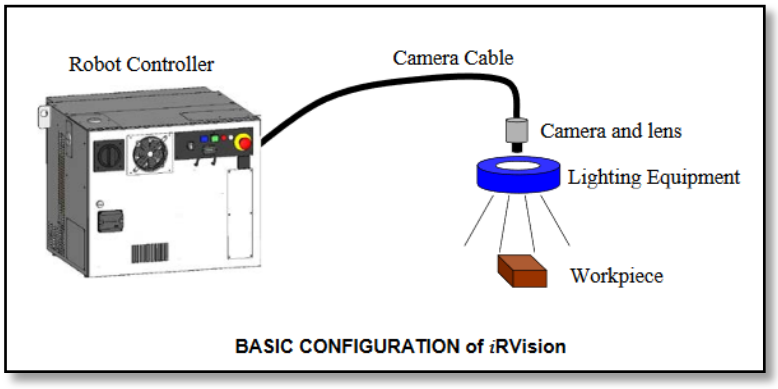
Obrázek 13 - Paralelní manipulátor [25]

2. Současná řešení robotického vidění

V dnešní době je na trhu k dispozici velké množství specializovaných průmyslových mechatronických systémů pro strojové vidění, defektoskopii, třídění a manipulaci. Níže jsou uvedeny tři příklady komerčních průmyslových systémů pro robotické vidění.

2.1.Fanuc

Firma Fanuc nabízí ke svým robotům kamerový systém iRVision (integral Robot Vision). Nabízeno je několik verzí systému, je možný výběr mezi monochromatickou či barevnou kamerou, kterou je možné umístit jak fixně, tak i na paži robota. Mezi volitelné vybavení patří například osvětlovač. Firma Fanuc garantuje spolehlivost svého kamerového systému i pro špatně identifikovatelné předměty, například předměty s vysokou světelnou odrazivostí či předměty znečištěné.



Obrázek 14 - Systém Fanuc iRVision v základní konfiguraci [17]

Systém byl navržen pro komunikaci s řídící jednotkou robota R - 30 iB Plus / R – 30 iB Mate Plus. Systém umí v základu číst čárové kódy, vyhledávat předměty v prostoru zabíraném kamerou a třídit je podle jejich geometrie. V případě využití barevné kamery umí předměty třídit i podle barvy. Umožňuje též vizuální kontrolu předmětů (t. j. např. provádění defektoskopie či vyhledávání tzv. zmetků). Pro nastavení systému je potřeba naučit software specifikace hledaných předmětů, ať už jde o geometrické tvary či barvu předmětů.

Další funkcí, kterou tento systém obsahuje, je možnost vybrat si mezi funkcemi Fixed Frame Offset, případně Tool Offset. V prvním případě je pomocí softwaru vypočítána poloha předmětu vůči základně robotu pro úlohu inverzní kinematiky. Funkce Tool Offset se využívá v případě složitějších koncových efektorů, jejichž souřadný systém se může v čase měnit a úloha vizuálního navádění je proto formulována vůči souřadnému systému efektoru. [17]

2.2.Keyence

Firma Keyence produkuje široký sortiment měřících přístrojů, senzorických systémů a systémů pro robotické vidění a dalších výrobků v oblasti měřící techniky. Vyrábí několik modelových řad systémů robotického vidění, a sice řady CV – X, XG – X, XG – 8000, XG – 7000 a CV – 5000. Tyto systémy poskytují v nejjednodušší variantě pouze funkci rozpoznání obrazu pro účely manipulace s objekty, ve vyšších verzích jsou implementovány například i nástroje pro obrazovou defektoskopii apod.

Dodávaný software pro strojové vidění nabízí možnost multispektrálního skenování, 2D a 3D vizuální kontrolu defektů, vzhledu a měření, vizuální řízení (Vision - Based Control), čtení čárových a QR kódů a možnost rozpoznávání znaků (Optical Character Recognition). [26]

Obrázek 15 - Systém Kayence CV - X s příslušenstvím [26]

2.3.ABB

Firma ABB Group nabízí systém robotického vidění ABB Integrated Vision, který je založen na kamerách Cognex In-Sight. Celý systém umožňuje současně využívat až tří kamer a je zejména vhodný k třídění, k vizuální inspekci součástek, jejich přesnému zarovnávání a dalším obdobným úkonům. V závislosti na komplexnosti úlohy proběhne celý proces od pořízení snímku k vyhodnocení obrazu v čase od 50ms do 2s. V závislosti na zamýšleném určení systému výrobce dodává kamery s rozlišením 800x600 Px (typ DSQC1020) nebo 1280x1024 Px (typ DSQC1021).

Obrázek 16 – Systém robotického vidění ABB Integrated Vision [40]

Systém lze v zásadě nastavit dvěma způsoby. Prvním způsobem je použití tzv. „pendantu," což je vstupně – výstupní periferie řídicího systému robota, přes kterou může operátor robotického pracoviště robota přímo ovládat či programovat. Druhou cestou je metoda offline programování přes vývojové prostředí RobotStudio prostřednictvím programovacího jazyka RAPID. [40]

Obrázek 17 - Pendant ABB IRC5 [2]

3. Řídící desky

Řídící systém většiny průmyslových robotů a manipulátorů obvykle bývá realizován formou profesionálního řešení pomocí PLC, jako například Fanuc 90-30 a za účelem programování často výrobce vyvíjí vlastní proprietární jazyk.

Pro potřebu této bakalářské práce bude řízení manipulátoru realizováno pomocí jednoduchého open source mikrokontroleru. Toto řešení je cenově nenáročné, pro zadanou testovací úlohu není potřeba vysoký výkon profesionálních PLC a implementace řešení je vzhledem k absenci nutnosti externích programovacích toolboxů pro MATLAB/Simulink jednoduchá.

Níže je předložena rešerše několika variant mikrokontrolerů, které je možné použít k realizaci úlohy. Vzhledem k povaze problému, k potřebné výpočetní rychlosti a k potřebnému množství vstupně – výstupních pinů (GPIO) byl pro praktickou realizaci experimentu vybrán mikrokontroler Arduino Uno.

Je nutné zmínit, že Arduino Uno je použito pouze k ovládání výstupních periferií robota. Veškeré výpočty související s rozpoznáváním obrazu i s výpočtem trajektorií jsou externě zpracovány v prostředí MATLAB/Simulink, které je spuštěno na stolním počítači, k němuž je Arduino Uno připojeno.

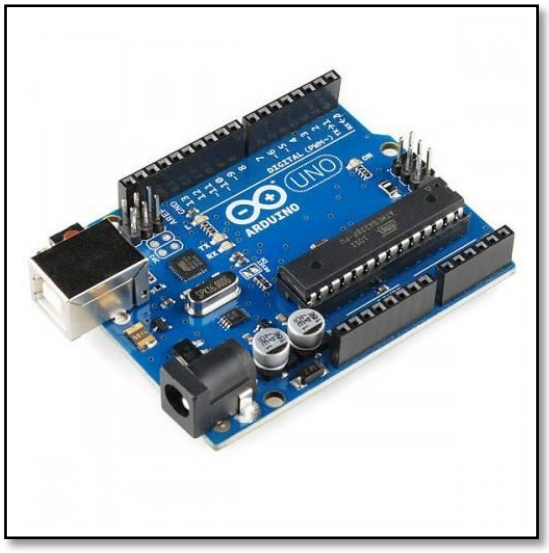
3.1. Arduino

Arduino je Open Source projekt jednoduché prototypovací desky. Nadace projektu Arduino byla založena v roce 2005 v Itálii. Tento systém ve vekém nahradil předtím používané prototypovací desky. Je tomu tak především kvůli nižší ceně a programováním ve snadno osvojitelném jazyce Wiring, který syntaxí vychází z jazyka C++. Lze ho však programovat i pomocí jiných programovacích jazyků. Níže jsou popsány 2 desky, které jsou svými parametry vhodné pro řešenou úlohu. [51]

3.1.1. Arduino Uno

Arduino Uno je vývojová deska založená na mikročipu ATmega328P o taktu 16 MHz. Obsahuje 14 vstupně/výstupních pinů (6 pro PWM, 6 analogových). Logická jednička je nastavena na +5 V, ale lze využít i napájení 3,3 V pro periferie, které nelze provozovat při napětí 5V. Maximální dosažitelný výstupní proud jednoho pinu je 20 mA při napětí 5 V a 50 mA při napětí 3,3 V. V případě potřeby vyššího výstupního výkonu lze desku osadit tzv. „shieldy," což jsou hardwarové součástky rozšiřující možnosti základní desky. V této byl použit tzv. PWM servo shield, což je modul, ke kterému lze připojit externí napájení a pomocí něhož lze řídit servomotory s větším proudovým zátížením. Programová paměť flash Arduina Uno má kapacitu 32 kB. [5]

Arduino Uno nemá dostatečný výpočetní výkon ke zpracování obrazu. Proto se používá jen jako převodníková karta, zpracovávající signály z externího PC, které vykonává výpočty nutné ke zpracování obrazu a k výpočtu trajektorií robota.



Obrázek 18 - Arduino Uno [6]

3.1.2. Arduino Yún

Arduino Yún je vývojovou deskou konstrukčně vycházející z jednoduššího Arduina Uno. Stejně jako Uno je vybaveno procesorem ATmega32u4 o frekvenci 16MHz. Mimo něj je však navíc vybaveno čipem Atheros AR9331, na němž je možně spustit unixový operační systém Linino. Dále je deska vybavena WiFi modulem, ethernet modulem a 20 digitálními input/output piny (7 z nich lze použít pro PWM modulaci, 12 je analogových). [7]

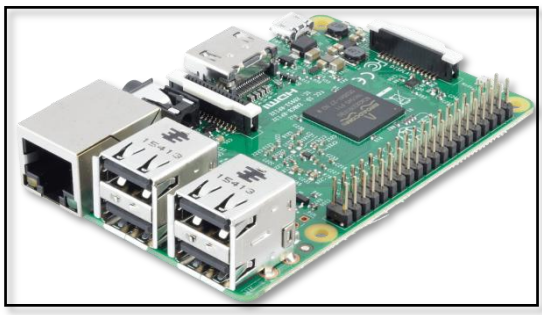
Ačkoliv by čip Atheros při mezním zatížení mohl zvládnout jednoduchou úlohu zpracování obrazu, v praxi toto řešení není vhodné z důvodu jeho nízkého výkonu, omezené paměti RAM a absenci datového úložiště (pouze SD karta) a z nich se odvíjející nevyhovění nárokům na zvládnutí úlohy v akceptovatelném čase. Nezanedbatelnou položkou je taktéž vysoká cena desky. Využití Arduina Yún tak spočívá především v oblasti IoT měření, kdy lze například posílat naměřená data přímo na vzdálený server. [7]

Obrázek 19 - Arduino Yún [8]

3.2. Raspberry Pi

Raspberry Pi je miniaturní počítač, který se od všech ostatních v tomto výběru liší tím, že konstrukčně obsahuje veškerý hardware, který obsahují klasické počítače. Je na něm možné spustit a provozovat operační systém Raspbian (různé verze), který je odvozen od operačního systému Debian. Kromě operačních systémů na bázi linuxu na něj lze nainstalovat i jiné systémy, např. odvozené od OS Windows (např. Windows 10 IoT Core).

V nejnovější verzi 3 má Raspberry Pi 1,2 GHz 64 bit CPU, 1 GB RAM a 400MHz GPU. Díky tomu je schopno realizovat i náročné výpočetní operace nutné pro strojové vidění i bez nutnosti spolupráce s externím PC. Operační systém je stejně jako u Arduina Yún nahrán na SD kartě. Konektivita je u Raspberry zajištěna čtyřmi USB 2.0 porty, HDMI výstupem, síťovým adaptérem, Wi-Fi a 17 GPIO vstupy/výstupy. Nevýhodou je, že pracuje pouze s logikou 3,3V, zatímco většina dnešních senzorů a periferií je konstruována pro použití s logikou 5V. [42]

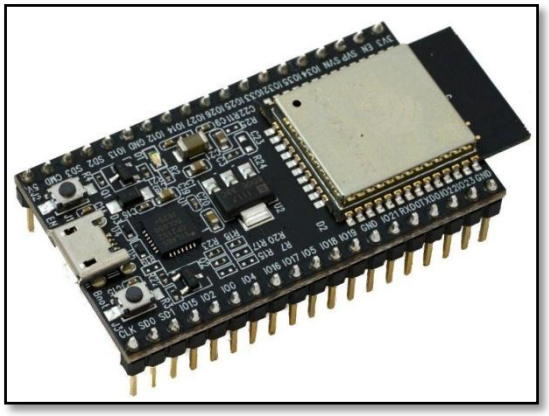


Obrázek 20 - Raspberry Pi model 3 B+ [43]

3.3. ESP 32

ESP 32 je vývojová deska vyvinutá společností Espressif. Obsahuje Wi-Fi a Bluetooth moduly. Jelikož tato deska byla primárně navrhnuta pro energeticky úsporné IoT aplikace, má nižší odběr energie než ostatní výše zmíněné desky.

Deska je osazena 32 bit dvoujádrovým procesorem Xtensa o výkonu 600 MIPS. Jako výstup slouží 34 GPIO pinů, které stejně jako v případě desky Raspberry Pi pracují pouze s logikou 3,3 voltu. I když nejnovější verze ESP 32 umožňuje připojit VGA kameru OV7670, z hlediska možnosti zpracování obrazu je tato deska pro Vision - Based Control obrazu prakticky nepoužitelná. [16]



Obrázek 21 - ESP 32 [15]

4. Software pro Vision Based Control

Profesionální řízení robotických manipulátorů na základě obrazové informace se běžně realizuje pomocí průmyslových PLC systémů a proprietárních softwarových nástrojů výrobce. Variantně lze roboty programovat pomocí různých programovacích jazyků. Nejčastěji jsou zde využívány nízkoúrovňové jazyky, jako je jazyk C či FORTRAN. Vybrané kritické aplikace bývají realizovány dokonce přímo v jazyce symbolických adres (tzv. assembleru).

Při nízkonákladových hardwarových řešeních se využívá široké spektrum programovacích nástrojů, počínaje jednoúčelovými jazyky vytvořenými pro konkrétní mikrokontrolery (např. Wiring pro Arduino), tak i standardní vysokoúrovňové jazyky, respektive jejich verze určené pro spouštění na mikrokontrolerech (MicroPython, JavaScript s frameworkem Cylon.js).

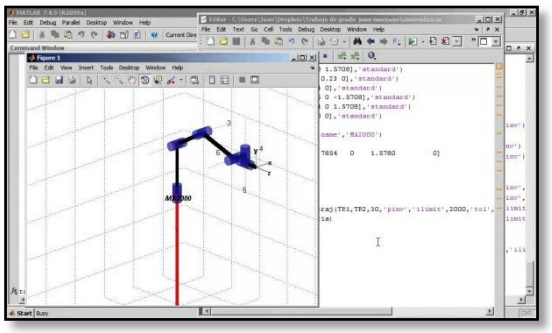
Zvláštní skupinou jsou takzvané grafické programovací jazyky, které obvykle nejsou vyvíjeny s přímým záměrem tvorby klasických aplikací. Využívají se převážně při matematickém modelování různých fyzikálních systémů, popřípadě jako nástroj digitální instrumentace pro zpracování signálů. Tyto jazyky využívají již připravených bloků reprezentující škálu od jednoduchých matematických operací až po složité modely fyzikálních systémů. Některé z nich umožňují vytváření a programování vlastních bloků.

Programy vytvořené v těchto jazycích mohou v zásadě pracovat ve dvou režimech. V prvním program převezme signál od senzoru, v PC je proveden příslušný matematický výpočet (zpracování vstupního signálu) a příslušný výstupní signál pošle do mikrokontroleru například přes sériovou linku. Druhou variantou je převod matematického modelu do zdrojového kódu v příslušném programovacím jazyku, který je následně do mikrokontroleru nahrán.

V dalších podkapitolách budou představeny tři zástupci programovacích jazyků používaných pro vědeckotechnické výpočty a návrh a obsluhu specializovaných mechatronických systémů. Jedná se o jazyk MATLAB s možností nástavby grafickým programovacím jazykem Simulink, dále o jazyk SciLab s možností nástavby modulem XCos a o čistě grafický programovací jazyk LabVIEW. V této bakalářské práci je využit programovací jazyk MATLAB.

4.1. MATLAB/Simulink

MATLAB je vysokoúrovňový programovací jazyk a programovací prostředí pro technické výpočty. Jeho název vznikl ze zkrácení slov Matrix Laboratory. Základní výpočetní modul lze rozšířit o tzv. toolboxy, což jsou nástroje umožňující například symbolické výpočty, zpracování signálů, generování kódu pro vestavěné (embedded) systémy a řadu dalších. Pro tuto práci jsou nejdůležitější Robotics Toolbox a Machine Vision Toolbox od Petera Corkea. [11]

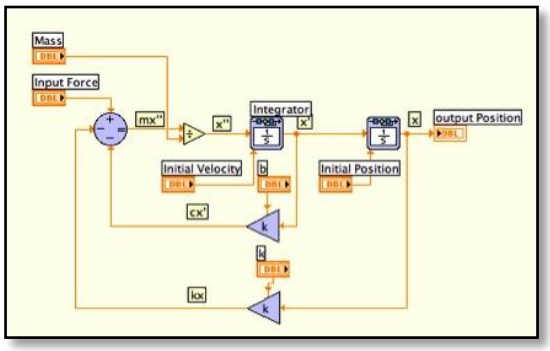


Obrázek 22 - Model robotu v prostředí MATLAB a Robotic Tbx [12]

Simulink je nástavba MATLABu a grafický programovací jazyk. Je určen primárně k simulaci chování dynamických systémů. Obsahuje bloky pro standardní matematické operace, bloky pro zobrazování dat, ale i pro připojení nejrůznějších hardwarových periferií.

4.2.LabVIEW

LabVIEW je grafický programovací jazyk a vývojové prostředí od firmy National Instruments (NI). Jeho název je zkratkou pro laboratorní pracoviště virtuálních strojů. Podobně jako MATLAB/Simulink tento programovací jazyk využívá množství rozšiřujících bloků, z nichž lze sestavit i velice komplexní modely systémů. Má též velké množství modulů pro senzorické systémy. LabVIEW proto využívají například i firmy vyrábějící přístroje pro MRI (Magnetic Resonance Imaging) či CERN (Evropská organizace pro jaderný výzkum). [29]

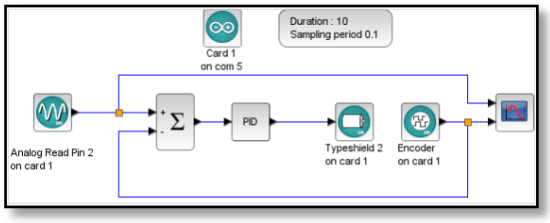


Obrázek 23 - Simulace dynamického systému v LabVIEW [30]

4.3.Scilab

Scilab je software pro numerické výpočty, vyvíjený francouzskou vědeckou institucí INRIA. Ta patří pod společnost ESI group, zabývající se virtuálním prototypingem (t. j. počítačově podporovaným návrhem produktů). Ze zde zmíněného softwaru je jako jediný šířený jako Open Source.

Mimo standardních knihoven pro numerické i analytické výpočty, zpracování signálu, teorii řízení a další procesy obsahuje Scilab i modul Xcos, což je grafický programovací jazyk, podobný grafickým programovacím jazykům Simulink či LabVIEW. Francouzská aerokosmická společnost CNES například Scilab využila pro modelování letové dynamiky sondy Rosetta, jejíž modul Philae bylo první člověkem vytvořené těleso, které přistálo na kometě (Churyumov - Gerasimenko 67P dne 12. 11. 2014). [45, 46]



Obrázek 24 - Čtení teploty z Arduina v Xcos modulu Scilab

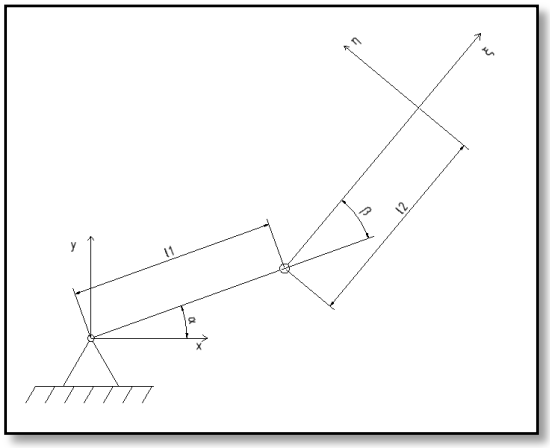
5. Mechanika průmyslových robotů a manipulátorů

Robot je z mechanického hlediska vázaný kinematický systém s N stupni volnosti. Nejčastěji využívaní průmysloví roboti se sférickým zápěstím mají 6 stupňů volnosti, ale vyskytují se i roboti s jiným množstvím stupňů volnosti, v závislosti na jejich zaměření.

Pro sériové manipulátory platí, že systém musí mít tolik hnacích členů, kolik má stupňů volnosti. Hnací členy jsou u robotů realizovány nejčastěji pomocí servomotorů, které jsou buď rotační, nebo lineární. Lineární hnací členy mohou být realizovány i pneumatickými či hydraulickými písty.

5.1. Přímá úloha

V přímé úloze kinematiky jsou známy délky všech členů tzv. kinematického řetězce a počet úhlů natočení v kloubech či servomotorech, který musí být minimálně počtu stupňů volnosti manipulátoru. Z těchto zadaných veličin pak lze spočítat polohu koncového efektoru robotického manipulátoru. Vzorový výpočet bude proveden na ukázce rovinného (planárního) manipulátoru s dvěma stupni volnosti.



Obrázek 25 - Schéma rovinného manipulátoru s 2 stupni volnosti

Za počátek souřadného systému *xy* je v tomto případě považován bod základny, kolem něhož se otáčí rameno l_1. Jsou-li zadány délky ramen l_1a l_2 a úhly natočení α a β, pak souřadnice koncového bodu manipulátoru lze stanovit ze vzorce [33]:

$$(1)\ \begin{bmatrix} x \\ y \end{bmatrix} = l_1 \begin{bmatrix} \cos\alpha \\ \sin\alpha \end{bmatrix} + l_2 \begin{bmatrix} \cos(\alpha - \beta) \\ sin(\alpha - \beta) \end{bmatrix}$$

Pro některé aplikace je důležité znát nejen koncovou polohu efektoru, ale i natočení jeho souřadného systému vůči souřadnému systému základny. Známe-li polohu uchopovaného objektu v souřadnicovém systému robotického manipulátoru *xy*, jeho polohu v souřadnicovém systému koncového efektoru $\xi\eta$ získáme pomocí geometrických transformací.

V tomto případě je rotace souřadného systému o úhel α definována rotační maticí:

$$(2)\ R(\alpha) = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Posunutí souřadného systému podél ramene l_1 je definováno translační maticí:

$$(3)\ T_x(l1) = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rotace souřadného systému o úhel β je určena rotační maticí:

$$(4)\ R(\beta) = \begin{bmatrix} cos\beta & -sin\beta & 0 \\ sin\beta & cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Posunutí souřadného systému podél ramene l_2 je definováno translační maticí:

$$(5)\ T_x(l2) = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

Výsledná transformace je určena maticovým vynásobením všech dílčích transformací:

$$(6)\ T_{\vartheta\zeta} = R(\alpha)T_x(l1)R(\beta)T_x(l2)$$

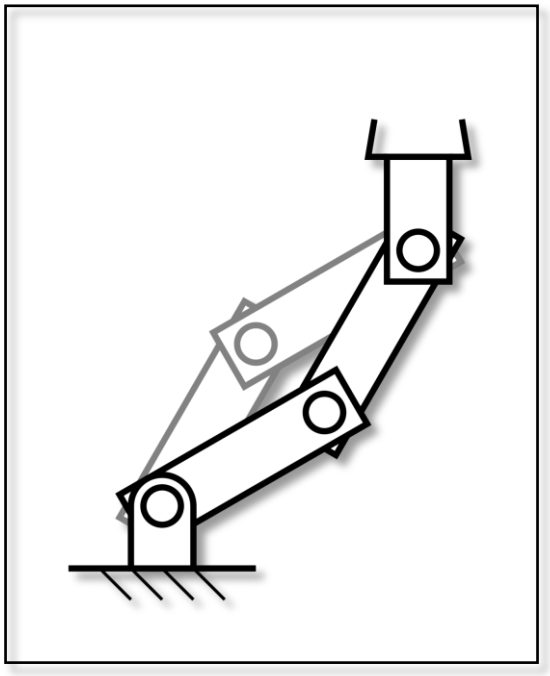
$$T_{\vartheta\zeta} = \begin{bmatrix} c(\alpha)\cdot c(\beta) - s(\alpha)\cdot s(\beta) & -c(\alpha)\cdot s(\beta) - c(\beta)\cdot s(\alpha) & l_2 * (c(\alpha)\cdot c(\beta) - s(\alpha)\cdot s(\beta)) + l_1 \cdot c(\alpha) \\ c(\alpha)\cdot s(\beta) + c(\beta)\cdot s(\alpha) & c(\alpha)\cdot c(\beta) - s(\alpha)\cdot s(\beta) & l_2 * (c(\alpha)\cdot s(\beta) + c(\beta)\cdot s(\alpha)) + l_1 \cdot s(\alpha) \\ 0 & 0 & 1 \end{bmatrix},$$

V této matici je výjimečně nahrazena funkce cosinus zkratkou „c“ a funkce sinus zkratkou „s“ z důvodu, aby se výsledná transformační matice vešla na jeden řádek. [47]

5.2. Inverzní úloha

Inverzní úloha se od přímé liší tím, že jsou známy poloha a orientace souřadného systému efektoru vůči počátku souřadného systému a hledanými veličinami jsou u rotačních vazeb úhly natočení jednotlivých servomotorů, případně u prizmatických vazeb délka vysunutí.

U manipulátorů s více stupni volnosti obsahuje rotační matice $T_{\vartheta\zeta}$ nelineární členy. To znamená, že výsledných řešení může být více než jedno. Často se stává, že řešení nemůže být vyjádřeno analyticky a je třeba přistoupit k numerickému řešení rovnic. [47]



Obrázek 26 - Dosažení stejné transformace dvěma cestami [24]

Výpočet inverzní kinematiky pro manipulátor na obrázku 25 lze provést analyticky a je následující: Jsou zadány souřadnice koncového bodu manipulátoru x a y. Taktéž je zadána transformační matice (6). Z ní lze určit analyticky koordináty koncového bodu manipulátoru, které reprezentuje poslední sloupec matice. Vyjádření koncové polohy je následující:

$$(7)\ x = l_2 * (cos(\alpha) \cdot cos(\beta) - sin(\alpha) \cdot sin(\beta)) + l_1 \cdot cos(\alpha)$$

$$(8)\ y = l_2 * (cos(\alpha) \cdot sin(\beta) + cos(\beta) \cdot sin(\alpha)) + l_1 \cdot sin(\alpha)$$

Při využití vzájemných vztahů mezi goniometrickými funkcemi lze tyto rovnice vyjádřit ve zjednodušeném tvaru:

$$x = l_2 \cdot cos(\alpha + \beta) + l_1 \cdot cos(\alpha)$$

$$y = l_2 \cdot sin(\alpha + \beta) + l_1 \cdot sin(\alpha)$$

Při sečtení těchto dvou rovnic je získána rovnice:

$$(9)\ x^2 + y^2 = {l_1}^2 + {l_2}^2 + 2l_1 l_2 cos(\beta)$$

Z této rovnice se vypočítá úhel beta:

$$(10)\ \beta = \arccos\left(\frac{x^2 + y^2 - {l_1}^2 - {l_2}^2}{2l_1 l_2}\right)$$

Provedeme – li substituci $cos(\beta) \rightarrow C_2$ a $sin(\beta) \rightarrow S_2$, lze rovnice (7) a (8) vyjádřit následovně:

$$(11)\ x = (l_1 + l_2 C_2) \cdot cos(\alpha) - l_2 S_2 \cdot sin(\alpha)$$

$$(12)\ y = (l_1 + l_2 C_2) \cdot sin(\alpha) + l_2 S_2 \cdot cos(\alpha)$$

Poloha kloubové souřadnice α se vypočítá vydělením rovnic (11) a (12) a substitucí podle rovnice (9):

$$(13)\ \alpha = \arctan\left(\frac{y}{x}\right) - \arctan\left(\frac{l_2 S_2}{l_1 + l_2 C_2}\right)$$

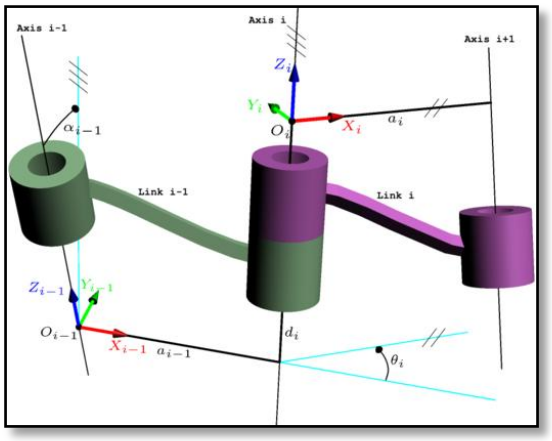
[47] str. 28

5.3.Denavit – Hartenbergova notace

Denavit – Hartenbergova notace je způsob, který postuloval Jacques Denavit s Richardem Hartenbergem v roce 1955 a kterým lze jednoduše popsat sériové robotické manipulátory definováním geometrických parametrů jejich vazeb. Oproti klasické geometrické reprezentaci robota je potřeba zadání pouze čtyř parametrů namísto šesti. Denavit – Hartenbergovými parametry jsou běžně definovány manipulátory v softwarových nástrojích, mezi které patří i níže uvedený MATLAB Robotics Toolbox.

d	Vzdálenost os x_j a x_{j-1}, kde osa z je osou rotace každé vazby a x_j je rovnoběžná se společnou normálou os z_j a z_{j-1}, osa y vždy doplňuje pravoúhlý souřadný systém
a	Vzdálenost os z_j a z_{j-1}, kde osa z je osou rotace každé vazby
θ	Úhel vzájemného natočení os x_j a x_{j-1}
α	Úhel vzájemného natočení os z_j a z_{j-1}

Tabulka 1 - Definice Denavit - Hartenbergových parametrů [11]

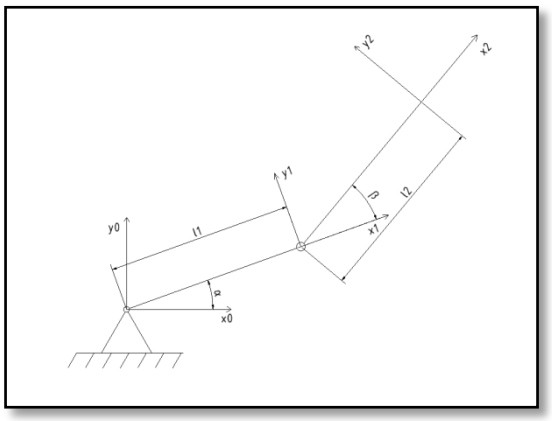


Obrázek 27 – Demonstrace Denavit - Hartenbergerových parametrů [13]

Stanovení Denavit – Hartenbergových parametrů pro manipulátor na obrázku 28 je následující: Manipulátor má dva články. Vzhledem k tomu, že osy z_1 a z_2 jsou rovnoběžné a zároveň jsou obě rovnoběžné s osou z_0 rotace kolem základny, parametr α je proto 0°. Osa x_1 je vůči ose x_0 souřadného systému základny v tomto případě natočena o úhel α a osa x_2 je vůči ose x_1 natočena o úhel β. Vzdálenost os z_2 a z_1 je l_2, čemuž je roven parametr a_2 a vzdálenost os z_1 a z_0 je l_1, čemuž je roven parametr a_1. Osa x_2 se protíná s osou x_1, proto je parametr d_2 roven 0 a osa x_1 se protíná s osou x_0, proto je parametr d_1 taktéž roven 0.

vazba	d	a	θ	α
1	0	l_1	α	0
2	0	l_2	β	0

Tabulka 2 - DH parametry 2D planárního manipulátoru [11]



Obrázek 28 - Stanovení D - H parametrů 2 DoF planárního manipulátoru

5.4. MATLAB Robotics Toolbox

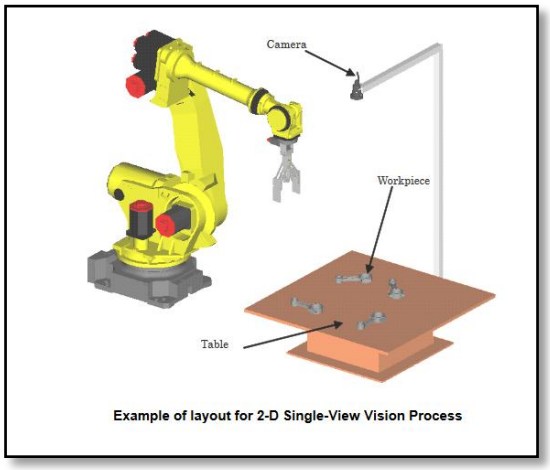
MATLAB Robotics Toolbox je softwarový nástroj, vytvořený Peterem Corkem, profesorem robotiky na Queensland University of Technology. Toolbox ulehčuje práci při studiu a simulaci širokého spektra mobilních i průmyslových robotů díky předdefinovaným funkcím pro výpočty klasické i inverzní kinematiky, dynamiky, pro výpočty trajektorií, geometrické transformace ve 2D i 3D a pro modelování mobilních robotů a sériových i paralelních manipulátorů. [44]

6. Robotické vidění

Základním senzorem pro interakci robota s prostředím je v případě této práce kamera. Jejím úkolem je zachytit obraz pracoviště robota a výstupem je matice hodnot jasu jednotlivých pixelů. Ta sama o sobě je pouhým souborem dat bez vyšší přidané hodnoty. Následnou analýzou pomocí vhodných matematických a softwarových prostředků umožní rozpoznání předmětů ve scéně, jejich identifikaci a další informace. Lze tedy říci, že především softwarovým zpracováním vstupních dat lze zajistit robotickému systému vizuální interakci s okolím.

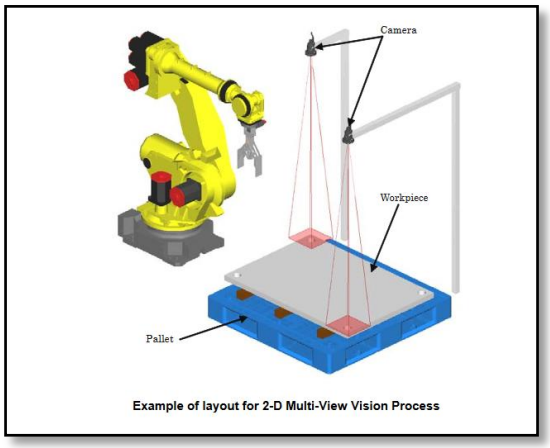
6.1. 2D vidění

V případě 2D vidění pomocí jediné kamery systém robotického vidění není schopen získat prostorové geometrické charakteristiky objektu. Jediné informace, které je systém schopen v tomto případě o předmětu získat, jsou tvar, barva, souřadnice objektu v prostoru robota a jeho natočení vůči souřadnému systému pracovní plochy robota. [18]



Obrázek 29 - Využití kamery k získání 2D obrazu pracovního prostoru [18]

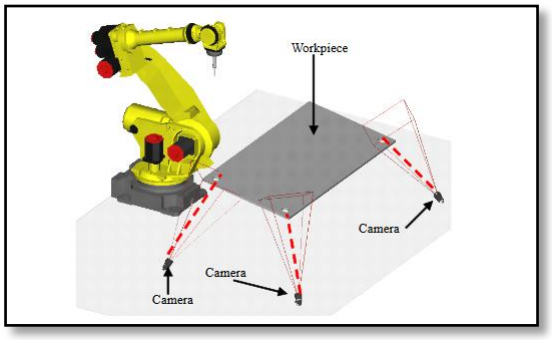
V případě potřeby vizuální percepce velkého prostoru, například pro kontrolu rozměrných dílů či montáž velkého množství předmětů na rozměrný základní kus, je třeba využít více kamer bez získání 3D modelu. [18]



Obrázek 30 - Využití více kamer bez získání 3D obrazu [18]

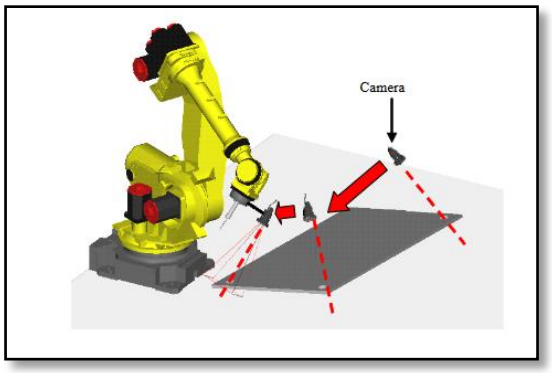
6.2. 3D vidění

Pro 3D vidění existují v zásadě dva přístupy. První z nich využívá statického umístění kamery a využívá obrazu z dvou a více kamer. Obraz složený z jednotlivých kamer při znalosti jejich prostorové vzdálenosti, popřípadě natočení, lze interpretovat a reprodukovat z něj 3D obraz. [18]



Obrázek 31 - 3D vidění s využitím více kamer [18]

Další možností je rekonstrukce 3D obrazu z jediné kamery, která se nachází na rameni robota. Během procesu fotografování robot najede do více poloh pro nasnímání obrazu a složením obrazů se reprodukuje výsledná 3D scéna. Tento druh získávání 3D obrazu se nehodí pro aplikace, ve kterých je kladen důraz na rychlost, z důvodu omezené rychlosti pořízení všech snímků. [18]



Obrázek 32 - 3D vidění s využitím jediné kamery [18]

6.3.MATLAB Machine Vision Toolbox

Tento toolbox je opět nástrojem od Petera Corkea a obsahuje přes 100 funkcí pro získávání, zobrazování, filtrování, pro segmentaci obrazu a dalších. Vzhledem k tomu, že byl přímo navržen pro Vision - Based Control v kombinaci s MATLAB Robotic Toolbox, umožňuje analyzovat jednoduché scény v téměř reálném čase. U složitých scén analýzu dat v reálném čase nelze použít vzhledem k tomu, že je MATLAB interpretovaným jazykem, jehož výpočetní rychlost je v tomto případě nedostatečnou.

Toolbox využívá toho, že MATLAB nativně pracuje se všemi daty jako s maticemi. Kvůli tomu lze pracovat s rastrovým obrazem jako s maticí bodů, které mají různou intenzitu RGB, popřípadě různé stupně šedi (grayscale). Některé základní funkce zpracování obrazu, jakou je například prahování (tresholding), mohou být implementovány i bez Machine Vision

Toolboxu. Machine Vision Toolbox obsahuje navíc například funkce pro zpracování obrazu formou pokročilé segmentace, jako je například MSER (Minimum Stable Extreme Regions). Machine Vision Toolbox obsahuje také podporu vstupních dat z mnoha typů kamer, například kamer typu rybí oko (fisheye), katadioptických systémů (systémů využívajících kombinace lomu a odrazu světla, využívaných v astronomii), sférických kamer a dalších. Obsahuje také Simulink bloky pro Vision - Based Control základních typů robotů. [33]

7. Návrh robotického pracoviště

Úloha, kterou bude robot realizovat v tomto případě, bude třídění předmětů na základě jejich tvaru. Celkové uspořádání pracoviště je třeba přizpůsobit povaze tříděných předmětů, jako je tvar, materiál a jeho vlastnosti fyzikální (např. zda je feromagnetický).

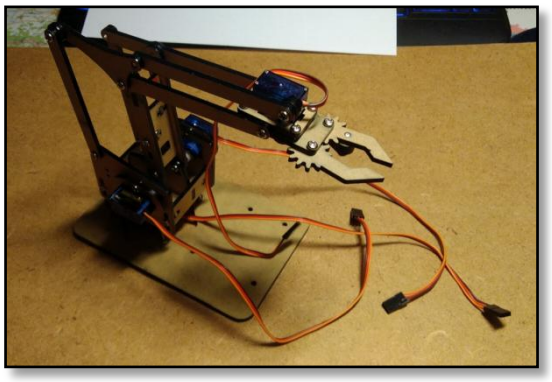
Na základě těchto vlastností je třeba stanovit nejen nutný počet stupňů volnosti použitého robotického manipulátoru a koncový efektor, ale i to, zda pro řešenou úlohu postačí 2D vidění, realizované jednou kamerou, či zda je třeba využít 3D vidění při nasazení 2 a více kamer.

7.1. Robotický manipulátor

V této práci byly zpracovány tři různé testovací úlohy. Při první byl ověřen řídící algoritmus na jednoduchém modelu robotického manipulátoru se čtyřmi stupni volnosti. Ve druhé úloze byl vytvořený software testován na modelu robotického manipulátoru se šesti stupni volnosti. Třetí úlohou byla aplikace softwaru pro řízení reálného robota Mitsubishi MELFA RV – 2SD, taktéž se šesti stupni volnosti.

7.1.1. Robotická paže MeArm

Prvním použitým robotickým manipulátorem je robotická paže MeArm. Tento robot je vyroben z plastové překližky ABS a je určen k ručnímu hobby sestavení.



Obrázek 33 – Sestavená robotická paže MeArm

Manipulátor má 4 DoF a je vybaven 9 g modelářskými servomotory SG90. Tyto servomotory mohou být napájeny přímo z desky Arduina Uno bez nutnosti použití externího napájení. Jejich nevýhodou je omezený krouticí moment a z toho se odvíjející nízká zatížitelnost robotické paže.

Celková konstrukce odpovídá nízké ceně a není příliš robustní. Výhodou je poměrně snadné sestavení robotické paže, robotická paže není na ose servomotoru základny nijak adjustována a při prudkých změnách pohybu má tendenci z této vazby vypadávat. Tento problém byl svépomocí částečně vyřešen vložením silikonové distanční podložky.

7.1.2. Robotická paže SainSmart

Robotická paže SainSmart je hobby model robotického manipulátoru se 6 DoF. V porovnání s paží MeArm je však mnohem robustnější a svými vlastnostmi se přibližuje reálnému robotickému manipulátoru, ačkoliv se stále jedná pouze o model. Výhodou je také možnost použití vlastního koncového efektoru, který lze pomocí na rameno přišroubovat k standardizované přírubě.

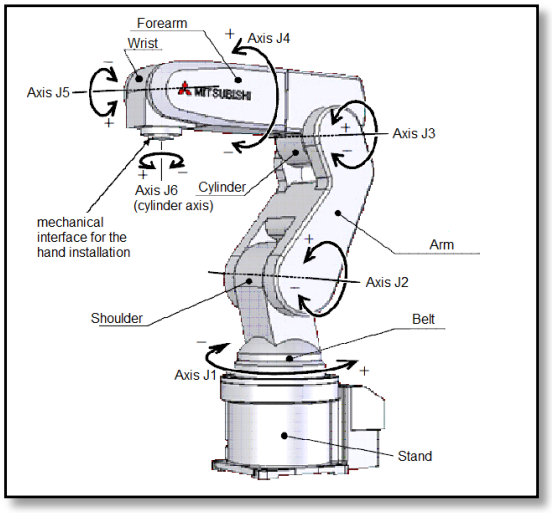
Obrázek 34 - Robotická paže SainSmart [1]

Použitým materiálem je PVC a celá konstrukce je dostatečně tuhá i pro vyšší zatížení. Tomu odpovídají i zvolené pohony prvních čtyř os, které jsou tvořeny čtyřmi elektrickými servomotory MG996R, pohon posledních dvou os je zajištěn dvěma již výše zmíněnými servomotory SG90. Nevýhodou servomotoru MG996R je vyšší napěťový a proudový odběr a s ním související nutnost použití externí napájení desky (shieldu). Naopak výhodou je krouticí moment 9,4 – 13 kg/cm, což je asi 7x více, než u servomotoru SG90. [35]

Obrázek 35 - Servomotor MG996R [35]

7.1.3. Robot Mitsubishi

Robot MELFA RV – 2SD je robotická paže s 6 DoF od společnosti Mitsubishi Electric. Tento robot váží 19 kg, přičemž maximální nosnost je 3 kg. Pro zajištění maximální přesnosti opakovaného pohybu na danou souřadnici doporučuje výrobce maximální zatížení omezit na 2 kg. Lineární rychlost manipulátoru je až 4400 mm/s. Přesnost opakovaného úchopu je 0,02 mm. [37]



Obrázek 36 - Manipulátor MELFA RV - 2SD [38]

7.2. Použitá kamera

V tomto experimentu je použita webkamera Logitech C170. S cenou okolo 500 Kč se jedná o webkameru střední třídy. Kamera se vybavena rozhraním USB 2.0 a senzorem typu CMOS s maximálním rozlišením 1024 x 768 Px, přičemž může pořizovat fotografie s rozlišením až do 5MPx. [31]

Obrázek 37 - Webkamera Logitech C170 [31]

7.3. Tříděné předměty

Při volbě předmětů, které mohou být tříděny, je třeba respektovat několik omezení. Je to z důvodů, že výstupem této práce je technologický demonstrátor, jehož schopnosti pracovat v reálných podmínkách, ve kterých běžně pracují robotické třídící linky, jsou omezené.

První věcí, která byla vzata v potaz, byl použitý aktuátor, viz kapitoly 7.4.2, 7.4.3 a kapitola 8. Uchopení manipulovaného předmětu zajišťuje magnet, proto je potřeba, aby byly předměty magnetické. Kvůli omezené magnetické síle aktuátoru nesmí být manipulované předměty příliš těžké, jinak by nemohla být zaručena potřebná adjustace během manipulačního procesu.

Robotická paže MeArm nevyžaduje feromagnetický materiál z důvodu využití klešťového mechanismu, viz kapitola 7.4.1. Je taktéž potřeba, aby tříděné předměty byly kontrastní vůči pozadí pracovní plochy. Dále je potřeba, aby měly dostatečnou výšku, protože aktuátor umožňuje pouze uchopení z boku. Poslední požadavkem je velice nízká hmotnost manipulovaných předmětů vzhledem ke konstrukci manipulátoru.

Posledním společným znakem manipulovaných předmětů musí být dostatečný barevný kontrast vůči pozadí manipulační plochy, zajišťující správnou identifikaci předmětů. Dále je třeba respektovat, že software na vizuální rozpoznávání předmětů, který vznikl v rámci této práce, umí rozpoznávat pouze tři druhy tvarů, kterými jsou kruhy, čtverce a obdélníky.

7.4. Koncový efektor

U každého z robotů použitých v této práci je k uchopování tříděných předmětů použitý jiný systém. Ten byl odvozen především z konstrukce jednotlivých manipulátorů, jejich rozměrů a od jejich maximální možné zatížitelnosti.

7.4.1. MeArm

Robotická ruka MeArm je již v základu vybavena vlastním chapadlem, které tvoří symetricky otevírané kleště spřažené ozubeným soukolím a ovládané servomotorem SG90. Kvůli jejich konstrukčnímu uspořádání je třeba, aby byl tříděný předmět uchopen z boku. Materiál manipulovaných předmětů nemá žádná omezení.

7.4.2. Mitsubishi

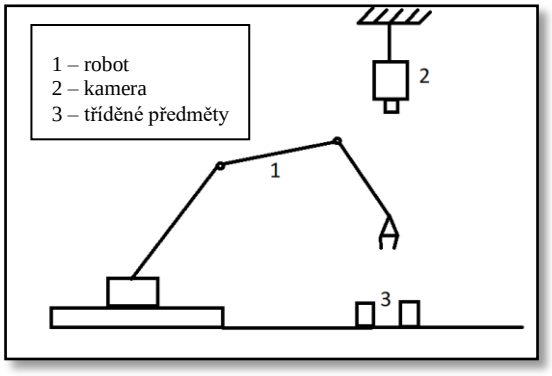
Robotický manipulátor MELFA RV - 2SD je vybaven magnetickým uchopovačem na bázi lineárního elektromotoru. Vzhledem k nepřiměřeně vysoké finanční náročnosti pořízení takového uchopovače byl po domluvě s vedoucím bakalářské práce zařazen do práce jeho návrh, kterému se bude věnovat kapitola 8 a uchopovač byl následně vyroben za využití 3D tisku.

7.4.3. SainSmart

Robotická paže SainSmart byla pro třídění vybavena taktéž magnetickým uchopovačem na bázi lineárního elektromotoru. Ten je přišroubován na přírubě robota a lze jej využít pouze k uchopování předmětů z feromagnetických materiálů. Jeho konstrukce je stejná, jako v případě uchopovače použitého na robotu Mitsubishi MELFA, vzhledem k menším rozměrům byl zvolen přiměřeně velký aktuátor a tomu byla uzpůsobena konstrukce. Konstrukce musela být taktéž přizpůsobena rozměrům příruby robota SainSmart.

7.5. Jednotlivá pracoviště

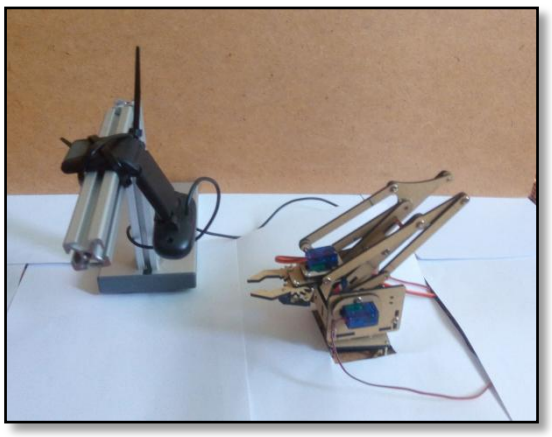
Všechna pracoviště byla navržena podle podobného vzoru, který je uveden na obrázku 39, především z důvodu co největší přenositelnosti softwaru. Společným znakem je napevno umístěná kamera nad plochou, na které jsou umístěny tříděné předměty.



Obrázek 38 - Návrh robotického pracoviště

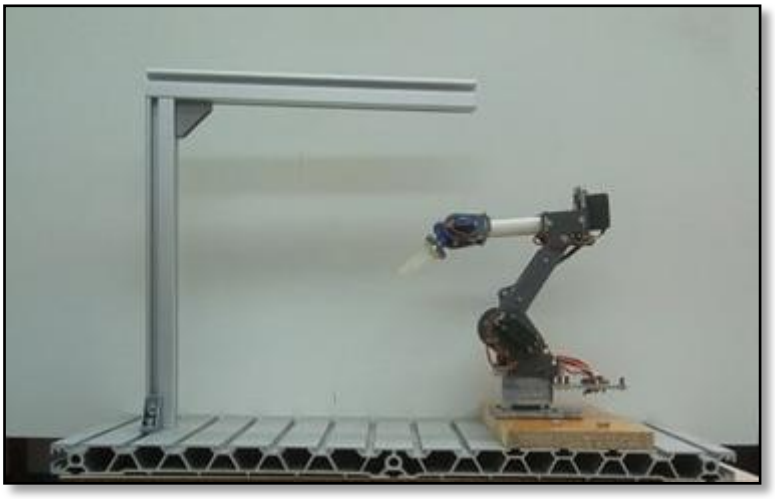
Kamera je umístěna vně pracovního prostoru robota, ale tak, aby byla snímána celá pracovní plocha. Zároveň je robot umístěn tak, aby se mohl během snímání scény celým ramenem nacházet mimo snímanou oblast a neznemožnil identifikaci některých tříděných předmětů v pracovním prostoru.

Pracoviště robotické paže MeArm bylo koncipováno především jako experimentální platforma na testování jednotlivých částí softwaru, jenž byl vyvinut v rámci této bakalářské práce. Základ tvoří samotný manipulátor, který je pomocí zdrhovací elektrikářské pásky připevněn k dřevěné desce, čímž je mu poskytnuta dostatečná fixace. Pro vytvoření dostatečného kontrastu pro rozpoznání předmětů v pracovním prostoru je místo podloženo bílým papírem. Kamera byla připevněna na hliníkovém profilu od firmy FESTO opět pomocí zdrhovací elektrikářské pásky. Zásobníky na ukládání tříděných předmětů byly tvořeny plastovými krabičkami a z důvodu omezeného rozsahu robota byla tato úloha zjednodušena na třídění pouhých dvou druhů předmětů, přičemž po každé straně se nacházel jeden zásobník.



Obrázek 39 - Pracoviště robota MeArm

Pracoviště robota SainSmart tvoří samotná robotická paže, která je pomocí vrutů přišroubována k desce z dřevotřísky. Toto deska je dále čtyřmi šrouby připevněna k hliníkové desce od firmy FESTO, která svým profilem umožňuje snadnou fixaci. Pomocí tvarových prvků od téhož výrobce byl vytvořen prvek pro uchycení kamery. Ta je opět k tomuto prvku připojena pomocí zdrhovací elektrikářské pásky. Pracovní prostor byl stejně jako v předchozím případě vybaven bílým pozadím pro vytvoření kontrastu předmětů na manipulační ploše.



Obrázek 40 - Pracoviště manipulátoru SainSmart

Pracoviště robota Mitsubishi MELFA RV – 2SD je tvořeno samotným robotem připevněným opět k hliníkovému profilu FESTO, který je připevněn k manipulačnímu vozíku, na němž je možno robota přemisťovat. Opět je součástí tvarový prvek, který slouží k uchycení kamery. Pracoviště robota nebylo prozatím vybaveno třídícími zásobníky a počítá se s tímto do budoucna.

U robotů SainSmart a Mitsubishi MELFA RV – 2SD prozatím nebyly zásobníky na tříděné předměty nijak realizovány.

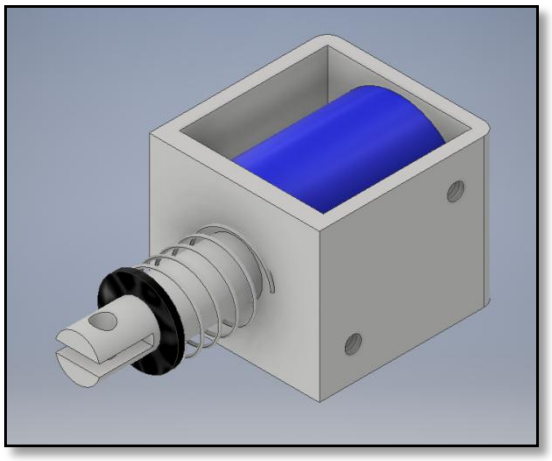
Obrázek 41 - Pracoviště manipulátoru MELFA

8. Návrh a konstrukce magnetického uchopovače

Tato kapitola se bude věnovat konstrukci jednoduchého magnetického uchopovače, který lze využít na robotickém manipulátoru Mitsubishi MELFA RV-2SD. Vlastní konstrukce byla zvolena především z důvodu finanční nákladnosti pořízení podobného zařízení od komerčního poskytovatele.

Zařízení je tvořeno válcem, v němž je umístěn elektromagnetický lineární aktuátor. Při maximálním vysunutí pístu se magnet přiblíží dostatečně k okraji válce a dostatečnou silou přitáhne uchopovaný předmět. Při zasunutí pístu se magnet vzdálí od předmětu natolik, že síla magnetického pole magnetu již nestačí k udržení předmětu a ten od okraje válce odpadne.

Centrálním prvkem celého návrhu je použitý aktuační člen, kterým je elektromagnetický lineární aktuátor JF – 0826B. Po přivedení elektrického proudu do vinutí cívky je jádro vtaženo dovnitř a magnet se vzdálí od uchopovaného předmětu. Po vypnutí proudu se jádro díky pružině opět vysune ven. Na konci jádra aktuátor je přišroubován neodymový magnet. Výška magnetu je 5 mm.



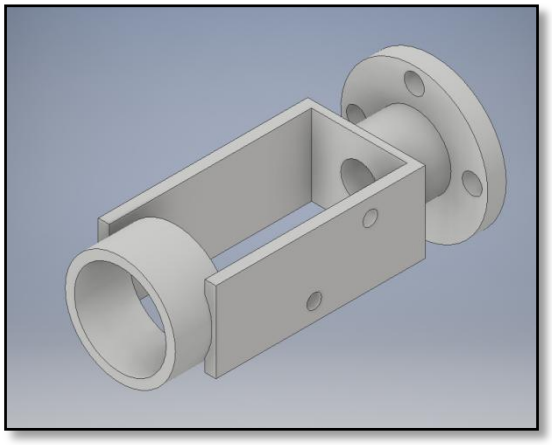
Obrázek 42 - 3D model použitého aktuátoru

Tomuto komponentu bylo nutné přizpůsobit vnitřní rozměry uchopovače, aby bylo možné aktuátor upevnit uvnitř. Dalšími rozměry, které je nutné při návrhu respektovat, jsou rozměry příruby robota, pro který je uchopovač navrhován (výkres manipulátoru Mitsubishi MELFA RV – 2 SD je přiložen v příloze č.3.

Výroba uchopovače byla realizována pomocí metody 3D tisku, jehož přesnosti je omezena. Vzhledem k tomu, že absence přesného tolerování rozměrů neovlivní nijak funkčnost celého zařízení, jsou tolerance omezeny na minimum.

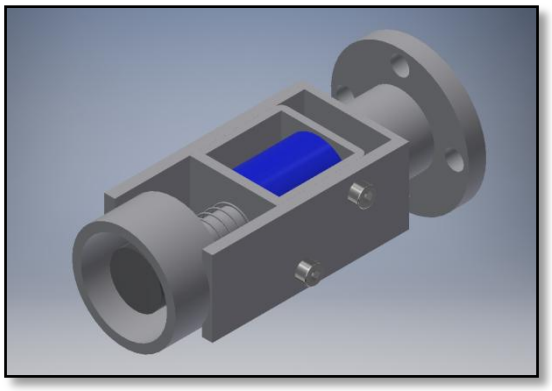
Příruba je standardní pro manipulátor Mitsubishi MELFA RV - 2SD. Po obou stranách má osazení. Jedno slouží k vystředění příruby uchopovače vůči přírubě manipulátoru. Druhé osazení slouží k tomu, aby byl vytvořen prostor pro fixaci příruby pomocí šroubů. Vzhledem k velikosti použitého aktuátoru by jinak nebylo možné přírubu k manipulátoru přišroubovat. V osazení i přírubě je v ose po celé délce otvor poskytující prostor pro vtažené jádro uchopovače, jehož osa se při přivedení proudu do cívky vysune zadní stranou do vzdálenosti cca 10 mm.

Následuje prostor pro samotný aktuátor. Rozměry odpovídá rozměru aktuátoru a umožňuje jeho snadnou montáž do těla uchopovače.



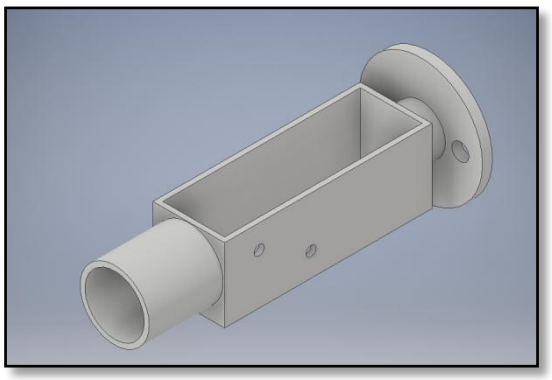
Obrázek 43 - Tělo uchopovače

Poslední částí aktuátoru je obruba, která slouží k vymezení vzdálenosti mezi manipulovaným předmětem a magnetem připevněným k aktuátoru. Tím předchází možnému poškození magnetu v případě, že by došlo k prudkému kontaktu manipulovaného předmětu s magnetem. Při vtažení jádra do cívky udržuje vzdálenost mezi manipulovaným předmětem a magnetem, čímž je umožněno uvolnění předmětu.



Obrázek 44 - Sestava uchopovače

Konstrukce uchopovače pro robotický manipulátor SainSmart je téměř identická. Liší se pouze rozměry, jež odpovídají jednak menšímu aktuátoru a jednak menšímu průměru příruby robota.



Obrázek 45 - Uchopovač pro robota SainSmart

9. Tvorba softwaru pro robotické manipulátory

V této kapitole bude představen celkový návrh softwaru pro řízení robotického manipulátoru. Software byl implementován ve skriptovacím jazyku MATLAB s rozšířením o toolboxy Machine Vision Toolbox a Robotics Toolbox a knihovny Image Acquisition Toolbox a ArduinoIO. Pro programování robotického manipulátoru MELFA RV – 2SD byl využit Mitsubishi Melfa Robot Control Toolbox.

Návrh algoritmu spočívá v tom, že robot se nejprve přemístí do polohy pro fotografování. Následně kamera sejme pracovní prostor robota. Dalším krokem je identifikace předmětů v tomto prostoru, jejich druh a poloha. Poté jsou softwarově vypočítány prostřednictvím funkce pro inverzní kinematiku potřebné kloubové souřadnice pro najetí efektorem k tříděným předmětům za účelem jejich uchopení a stanovení potřebných trajektorií jejich přesunu do připravených boxů v závislosti na druhu manipulovaného předmětu. Poslední fází je vykonání pohybů robotem, přičemž vždy po najetí efektoru do bodu pro úchop je Arduinem vyslán příkaz k uchopení předmětu, k jeho přenesení nad příslušný box a k jeho uvolnění.

Vývojový diagram algoritmu pro ovládání robotického manipulátoru je přidán jako samostatná příloha.

9.1. Matematický model

Robotický manipulátor je nutné matematicky popsat, t. j. je nutno stanovit a zadat příslušné Denavit – Hartenbergovy parametry a odpovídající geometrické transformace jednotlivých manipulátorů. Analýzou výše uvedeného lze získat použitím funkcí pro výpočty inverzní kinematiky příslušné kloubové souřadnice. Více o této problematice je popsáno v kapitolách 5.2 a 5.3.

Každému manipulátoru, který je testován v této práci, přísluší jiný matematický model.

Robotická paže MeArm má 4 stupně volnosti, z nichž tři zajišťují pohyby ramen robota, a jeden znázorňuje míru rozevření čelistí. Vzhledem k její konstrukci, jsou osy souřadného systému efektoru $\xi\eta\zeta$ rovnoběžné s osami *xyz*. Při sestavování matematického modelu tudíž není třeba brát v úvahu natočení aktuátoru. Denavit – Hartenbergovy parametry jsou pro tuto paži následující, přičemž vzhledem k sérioparalelní struktuře manipulátoru je úhel q_3 závisí na úhlu q_2:

Vazba	$\boldsymbol{\alpha}$	**a**	$\boldsymbol{\theta}$	**d**
1	$\frac{\pi}{2}$	15	q_1	55
2	0	80	q_2	0
3	0	88	$q_2 + q_3$	0

Tabulka 3 – DH parametry paže MeArm

Příslušné transformace pro popis transformace mezi základnou robotu a koncovým efektorem jsou následující:

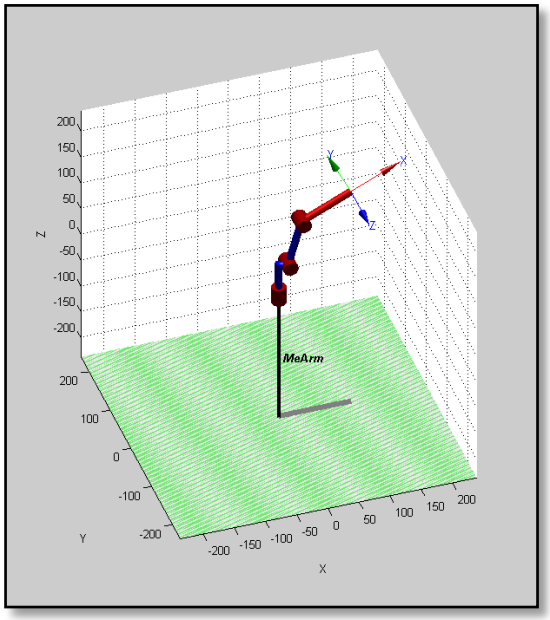
$$T_{10} = R_z(\theta_1) \cdot T(d_1) \cdot T(a_1) \cdot R_x(\alpha_1)$$

$$T_{21} = R_x(\theta_2) \cdot T(a_2)$$

$$T_{32} = R_z(\theta_3) \cdot R_x(\alpha_3) \cdot T(a_3)$$

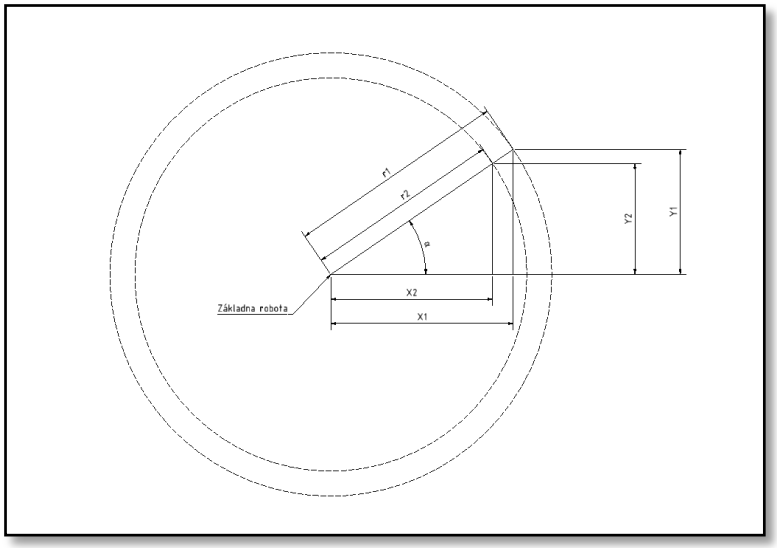
Celková transformace je popsána vztahem:

$$T = T_{10} * T_{21} * T_{32}$$



Obrázek 46 - Model robotické paže MeArm v Robotics Toolboxu

Je potřeba brát v úvahu fakt, že střed čelistí má od konce posledního článku jisté odsazení. Toto odsazení je 18 mm. Při programování funkce na přepočet souřadnic detekovaných předmětů v obraze na souřadnice v souřadném systému robota musela být tato korekce respektována.



Obrázek 47 - Úprava cílových souřadnic manipulátoru

Na obrázku 30 jsou vyznačeny souřadnice uchopovaného předmětu x_1 a y_1 a potřebné souřadnice x_2 a y_2 koncového bodu manipulátoru. Cílové souřadnice lze získat určením délky přepony trojúhelníku $x_1 y_1 r_1$:

$$r_1 = \sqrt{{x_1}^2 + {y_1}^2}$$

Dále je třeba určit úhel α, což je úhel natočení ramene robotu v souřadnicovém systému základny *xy*:

$$\alpha = \arctan \frac{y_1}{x_1}$$

Délka přepony r_2, t. j. se započtenou výše zmíněnou korekcí, se vypočítá následovně:

$$r_2 = r_1 - 18\,mm$$

Souřadnice koncového bodu efektoru jsou:

$$x_2 = r_2 \cdot \cos\alpha$$

$$y_2 = r_2 \cdot \sin\alpha$$

Robotická paže SainSmart má 6 stupňů volnosti, realizované rotačními vazbami, přičemž stejně jako v případě paže MeArm, se jedná o sériově – paralelní manipulátor. Příslušné Denavit – Hartenbergovy parametry jsou následující:

Vazba	$\boldsymbol{\alpha}$	**a**	$\boldsymbol{\theta}$	**d**
1	$-\frac{\pi}{2}$	40	q_1	110
2	0	127	q_2	0
3	$-\frac{\pi}{2}$	26	$q_2 + q_3$	0
4	$\frac{\pi}{2}$	0	q_4	133
5	$-\frac{\pi}{2}$	0	q_5	0
6	0	0	q_6	25

Tabulka 4 - DH parametry paže SainSmart

Příslušné transformace pro popis transformace mezi základnou robotu a koncovým efektorem jsou následující:

$$T_{10} = R_z(\theta_1) \cdot T(d_1) \cdot T(a_1) \cdot R_x(\alpha_1)$$

$$T_{21} = R_z(\theta_2) \cdot T(a_2)$$

$$T_{32} = R_z(\theta_3) \cdot R_x(\alpha_3)$$

$$T_{43} = R_z(\theta_4) \cdot T(d_4) \cdot R_x(\alpha_4)$$

$$T_{54} = R_z(\theta_5) \cdot T(a_5) \cdot R_x(\alpha_5)$$

$$T_{65} = R_z(\theta_6) \cdot R_x(\alpha_6)$$

Celková transformace je popsána vztahem:

$$T = T_{10} * T_{21} * T_{32} * T_{43} * T_{54} * T_{65}$$

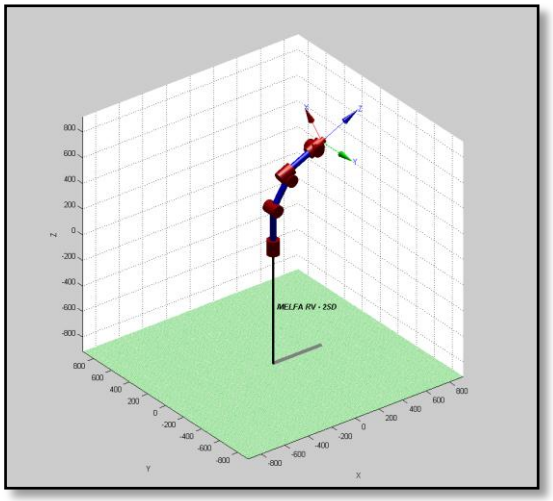
Manipulátor MELFA RV – 2SD má stejně jako robotická paže SainSmart 6 stupňů volnosti. Její kinematické schéma je tedy stejné jako u této robotické paže, liší se pouze délky jednotlivých spojovacích článků mezi jednotlivými rotačními vazbami a limitní polohy natočení v rotačních vazbách.

Vazba	$\boldsymbol{\alpha}$	**a**	$\boldsymbol{\theta}$	**d**
1	$-\frac{\pi}{2}$	0	q_1	295
2	0	230	q_2	0
3	$-\frac{\pi}{2}$	50	q_3	0
4	$\frac{\pi}{2}$	0	q_4	270
5	$-\frac{\pi}{2}$	0	q_5	0
6	0	0	q_6	70

Tabulka 5 - DH parametry robotu MELFA RV-2SD [38]

$$T_{10} = R_z(\theta_1) \cdot T(d_1) \cdot T(a_1) \cdot R_x(\alpha_1)$$
$$T_{21} = R_z(\theta_2) \cdot T(a_2)$$
$$T_{32} = R_z(\theta_3) \cdot R_x(\alpha_3)$$
$$T_{43} = R_z(\theta_4) \cdot T(d_4) \cdot R_x(\alpha_4)$$
$$T_{54} = R_z(\theta_5) \cdot T(a_5) \cdot R_x(\alpha_5)$$
$$T_{65} = R_z(\theta_6) \cdot R_x(\alpha_6)$$

$$T = T_{10} * T_{21} * T_{32} * T_{43} * T_{54} * T_{65}$$



Obrázek 48 - Kinematický model robotu MELFA v Robotics Toolbox

9.2. Zpracování obrazu

Při tvorbě softwaru pro zpracování obrazu bylo pro účely této práce uvažováno několik základních zjednodušujících předpokladů. Prvním z nich je přítomnost tzv. ideálního osvětlení, eliminujícího vznik stínů. Pro správnou funkci prahování obrazu je dále zapotřebí, aby hledané předměty byly vůči pozadí dostatečně kontrastní. Pro účely této práce proto bylo zvoleno bílé pozadí a s ním dostatečně kontrastující manipulované předměty.

Dále uvažujeme pouze statickou scénu. To znamená, že se tříděné objekty nebudou po celou dobu třídícího procesu pohybovat po pracovní ploše. Nasnímaná scéna se po pořízení snímku již nebude nijak měnit.

Pro zpracování obrazu z kamery byl využit dříve zmíněný MATLAB Machine Vision Toolbox. Prvním krokem je načtení obrazu z kamery ve formě matice hodnot RGB jednotlivých pixelů. Scéna je statická, proto je analyzován pouze jediný snímek. Kamera se iniciuje funkcí:

```
cam = VideoCamera();
```

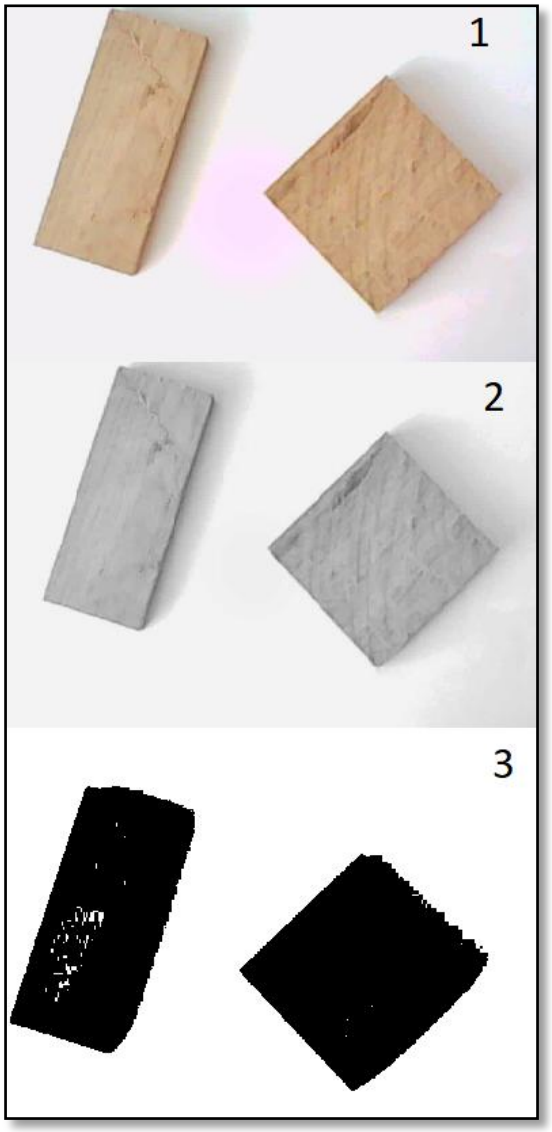
Obraz byl načten do proměnné *im* pomocí příkazu:

```
im = cam.grab();
```

Snímek je dále převeden z barevného obrazu na stupně šedi (grayscale). Výsledek je patrný z obrázku 49 – 2. Obraz ve stupních šedi byl převeden do proměnné *x*:

```
x = imono(im, 'r601');
```

Obraz ve stupních šedi se dále tzv. prahuje. Prahování (tresholding) je princip, kdy se obraz ve stupních šedi transformuje na černobílý binárně logický obraz. Pixelům od určité hodnoty jasu přiřadí hodnota jasu na logickou 1 („bílá“) a všem ostatním na logickou 0 („černá“), jak je vidět na obrázku 49 - 3.



Obrázek 49 - Proces prahování obrazu

Správná volba intenzity prahu je klíčovým krokem celé operace a její zjištění není triviální. V praxi bývá užíváno mnoha metod, přičemž každá se využívá pro specifické aplikace. V příloze č. 4 je popsáno několik často používaných metod pro prahování. V tomto případě byla volba prahu provedena ručně zadáním konstantní hodnoty prahu z důvodu dostatečného kontrastu tříděných předmětů vůči pozadí.

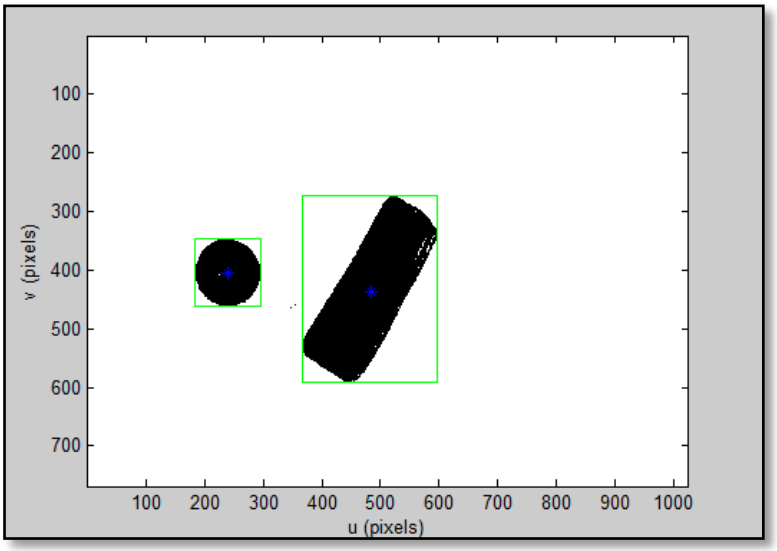
Dalším krokem je vyhledání tříděných předmětů v nasnímaném pracovním prostoru robota. Docílíme toho vyhledáním tzv. blobů, což jsou homogenní oblasti v prahovaném obrazu, které reprezentují předměty v pracovním prostoru robota.

Vzhledem k tomu, že vyhledáváme na pracovní ploše více než jeden předmět, využijeme metodu *ilabel*, jejímž parametrem je prahovaný obraz. Tato metoda vyhledá v obraze všechny homogenní oblasti, které jsou uzavřené a obklopené pixely opačné úrovně jasu. Jednotlivým blobům jsou přiřazeny tzv. štítky (labels), které ukazují index přidělený konkrétní homogenní oblasti.

K určení středu jednotlivých blobů je využita funkce *iblobs*. V základním nastavení vrací osm návratových hodnot, z nichž nejdůležitějšími pro tuto práci jsou hodnoty plochy blobu

(area), poměru *b/a*, což je poměr poloos tzv. ekvivalentní elipsy (Equivalent Ellipse) souřadnice geometrického středu tzv. hraničního rámečku (Bounding Box).

Hraniční rámeček je obdélník obklopující daný blob, jehož rozměry se určují z rozdílu hodnot pozice krajních pixelů *umax* a *umin* pro horizontální souřadnice (*x*) a *vmax* a *vmin* pro vertikální souřadnice (*y*).



Obrázek 50 - Objekty ohraničené ramečkem (Bounding Box)

Největší blob tvoří samotné pozadí scény. Vzhledem k faktu, že tento blob není pro vyhledávání manipulovaných předmětů zapotřebí, lze omezit vyhledávání objektů pouze na tmavé bloby zadáním funkce s těmito parametry, kde byly jednotlové bloby uloženy do proměnné *ib*:

```
ib = iblobs(b,'class',0);
```

Vzhledem k tomu, že hodnota prahu je zadána ručně a je konstantní, může dojít k detekci blobů, které jsou způsobeny stíny a jinými vadami v prahovaném obrazu. Jejich velikost je většinou v řádu pixelů. Selekce blobů, které reprezentují manipulované změny, bylo docíleno vytvořením nového pole prvků, do kterého byly vyselektovány pouze bloby, které obsahem přesáhly určitou hodnotu.

Dalším parametrem, který bylo třeba určit, bylo rozpoznání, o který objekt se jedná. Tříděnými objekty jsou předměty tvaru válce a šestibokých hranolů s podstavou tvaru čtverce a obdélníku. Při analýze obrazu tak software třídí tři druhy tvarů: kruhy, obdélníky a čtverce.

Obsah kruhu je definován vzorcem:

$$S = \frac{\pi \cdot d^2}{4}$$

Obsah čtvercového blobu obklopujícího kruh je definován:

$$S = a \cdot b$$

Čtvercový blob má obsah limitně se blížící k:

$$S = d^2$$

Poměr obsahu kruhu a blobu jej obklopujícího je:

$$i = \frac{\frac{\pi \cdot d^2}{4}}{d^2} = \frac{\pi}{4} \approx 0{,}79$$

Pro případ například nechtěné identifikace stínu a podobně je zadáno, aby software identifikoval veškeré předměty, které mají poměr obsahu k obsahu blobu větší než 0,79 a menší než 0,85 jako kruhy.

Dalším užitečným prvkem pro identifikaci prvků je parametr *b/a*, který znázorňuje poměr poloos ekvivalentní elipsy. Pomocí nich lze vyjádřit poměr stran tělesa samotného. U čtverce či kruhu se tato hodnota blíží k 1, s narůstajícím rozdílem stran klesá teoreticky až k nule pro přímku. V tomto případě byla všechna tělesa, která nejsou kruhem a mají poměr *b/a* větší, než 0,9, klasifikována jako čtverce. Veškerá ostatní tělesa jsou klasifikována jako obdélníky.

Všechny vyselektované předměty byly přidány do nové proměnné, kterou tvoří vícerozměrné pole obsahující souřadnice středů jednotlivých blobů a tvar jednotlivých předmětů a se kterým následně pracuje program pro výpočet trajektorie.

9.3. Výpočet trajektorie

Pro vypočtení trajektorie je nejprve nutné znát skutečnou polohu tříděných předmětů. Skript pro zpracování obrazu vrací jako výstup polohy středů tříděných předmětů v souřadném systému *xy* v pixelech. Aby byla získána jejich poloha v pracovním prostoru robota, musí být poloha v pixelech vůči skutečné vzdálenosti stanovena poměrem.

Funkce zajišťující přepočet souřadnic v pixelech na skutečně rozměry se odvíjí od výšky umístění kamery nad pracovní plochou. Za tímto účelem byly vytvořeny tři konstantní funkce pro přepočet souřadnic (je uvažováno fixní umístění kamery), které byly vypočítány z výšky umístění kamery u jednotlivých robotů. Vzhledem k tomu, že diagonální zorné pole kamery je 58°, vypočítá se vzdálenost krajních bodů diagonály zorného úhlu:

$$l = 2 \cdot h \cdot \tan\left(\frac{58°}{2}\right)$$

kde l je délka přepony, h je výška umístění kamery nad plochou a gama je zorný úhel kamery. Z poměru rozlišení stran pak lze dopočítat rozsah v souřadnicích x a y následovně, přičemž k souřadnici y je třeba připočítat odsazení středu zorného pole od základny robota:

$$x = l \cdot \cos(arctan\left(\frac{768}{1024}\right))$$

$$y = l \cdot \sin(arctan\left(\frac{768}{1024}\right))$$

Dále bylo potřeba nadefinovat několik významných bodů v prostoru. Prvním z nich je bod, do něhož robot najede před pořízením snímku pracovního prostoru. Dalšími významnými body jsou body pro najetí robota nad zásobník pro upuštění předmětu. Posledním významným bodem je „přejezdový“ bod, kterým prochází trajektorie robota při každém pohybu. Tím je zajištěno, že robot při všech vypočítaných trajektoriích nebude kolidovat se zásobníky a během transportu nedojde ke kolizi manipulovaných předmětů.

Pro cílový bod je třeba vypočítat translační matici. Ta se získá pomocí funkce:

```
p1=transl(x,y,z);
```

V případě významných bodů jsou matice vygenerovány předem, v případě souřadnic objektů získaných ze zpracovaného obrazu z kamery se získané souřadnice dosadí do této funkce. Výsledkem je transformační matice:

$$p1 = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pro výpočet potřebných kloubových natočení se používá funkce pro výpočet inverzní kinematiky, která může mít několik tvarů. Pokud má robot sférické zápěstí, jako je v případě robotů SainSmart či Mitsubishi MELFA, lze použít funkci:

```
uhly = promenna_robota.ikine6s(bod,'f');
```

V případě použití robota s méně stupni volnosti, jako v případě paže MeArm, lze použít funkci pro numerický výpočet kloubových souřadnic:

```
uhly = promenna_robota.ikine(cilovy_bod,vychozi_bod,maska);
```

Tuto funkce lze ale použít i pro výpočet inverzní kinematiky robota s 6 stupni volnosti. Maska, neboli Mask Matrix je matice, která určuje počet požadovaných změn v souřadnicovém systému při využití funkce *ikine*.

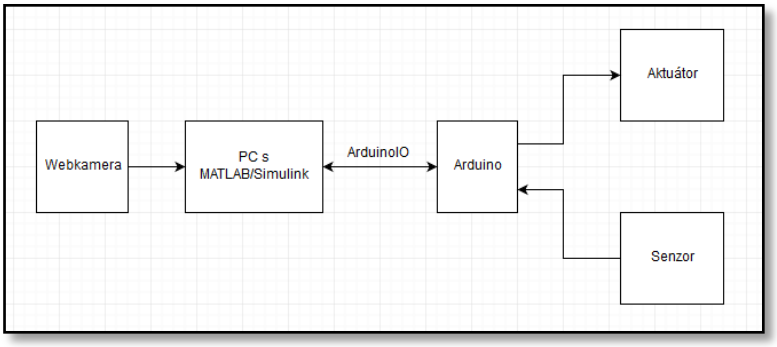
9.4. ArduinoIO

ArduinoIO je rozšiřující knihovna pro MATLAB/Simulink, umožňující přímou komunikaci s Arduinem, které v tomto případě plní funkci převodníkové karty. V paměti Arduina není nahrán celý program pro ovládání robota, ale pouze skript, který zpracovává signál z MATLABu, komunikující s Arduinem po sériové lince.

V závislosi na použitém shieldu existují 3 různé soubory. V případě nepoužití žádného shieldu se do Arduina nahraje soubor *srv.pde*. V případě použití motor shieldu pro ovládání stejnosměrných elektromotorů se do Arduina nahraje soubor *motorsrv.pde*. V případě použití servomotor shieldu pro PWM řízení servomotorů se použije soubor *adiosrv.pde*.

V případě vstupních periferií připojených na GPIO piny Arduina Uno zprostředkovává odesílání dat z těchto periferií do počítače, kde mohou být dále zpracovány, například pomocí MATLABu. [4]

Využití této knihovny značně rozšiřuje možnosti ovládání robotů pomocí arduina, jelikož umožňuje provádět hardwarově náročné úkoly, jakým je v našem případě například zpracování obrazu, je výpočet proveden na externím PC v MATLABu a Arduino vykonává pouze jednoduché instrukce pro práci s GPIO.



Obrázek 51 - Blokové schéma využití ArduinoIO

Vytvoření spojení s Arduinem přes virtuální sériový COM port se provede pomocí volání funkce *arduino*, jejíž vstupní hodnotou je COM port, ke kterému je Arduino připojeno. Tato funkce vytvoří proměnnou (instanci) obsahující veškeré informace o připojené desce, které jsou potřebné pro komunikaci MATLABu s Arduinem.

Dále je třeba definovat, které piny budou použity pro PWM řízení servomotorů. K tomuto účelu slouží metoda objektu Arduina *servoAttach*, jejímž vstupním parametrem je číslo pinu, který bude k řízení servomotoru použit. Pokud již dále není pin k ovládání servomotoru zapotřebí, lze ho deaktivovat pomocí metody *servoDetach*, jejímž vstupním parametrem je opět číslo pinu.

K řízení jednotlivých servomotorů slouží metoda *servoWrite*. Tato funkce má dva vstupní parametry. Prvním je číslo pinu, k němuž je připojený řízený servomotor. Druhým parametrem je úhel natočení servomotoru ve stupních.

V případě, že již není potřebná další komunikace MATLABu s Arduinem, ukončí se komunikace pomocí funkce *delete*, jejíž vstupní hodnotou je název proměnné s údaji pro sériovou komunikaci.

Knihovna ArduinoIO obsahuje množství dalších funkcí, které ale nebyly v bakalářské práci využity, a proto v této kapitole nebyly zmíněny.

10. Závěr

Cílem této bakalářské práce bylo navrhnout a ověřit model řízení robotického manipulátoru na základě obrazové informace s použitím jazyka MATLAB. V první fázi byla provedena rešerše v oblasti historie průmyslové robotiky a zpracování obrazu. Taktéž byla provedena i rešerše v oblasti dostupných komerčních systémů pro řízení průmyslových robotů na základě vizuální informace, dále v oblasti Open Source hardwaru a často využívaného softwaru.

V teoretické části byl představen teoretický základ k řešení základních úloh v dané oblasti, především týkající se úloh mechaniky průmyslových manipulátorů. Taktéž byly představeny využité softwarové nástroje, manipulátory, které byly v rámci práce modelovány/testovány a další použitý hardware.

V praktické části práce byl navrhnut prototyp testovacího pracoviště pro každý testovaný manipulátor. S využitím znalostí představených v teoretické části byly sestaveny matematické modely jednotlivých manipulátorů v prostředí MATLAB s využitím Robotics Toolboxu. Dále byl vytvořen s využitím MATLAB Machine Vision Toolboxu jednoduchý program na zpracování obrazu, který umožnil vyhledat souřadnice předmětů a taktéž byl tvořen program, který s využitím obou výše představených je schopen vypočítat pohyby robota potřebné ke splnění úlohy. Funkčnost celého programu byla ověřena na robotické paži MeArm a částečně na robotické paži SainSmart. Aplikace na robotickém manipulátoru Mitsubishi MELFA je plánována do budoucnosti.

Poslední částí, která byla vypracována dodatečně, je zjednodušený konstrukční návrh magnetického uchopovače, jenž bude v budoucnu realizován pomocí 3D tisku.

11. Vize budoucnosti

Zkušenosti, které jsem získal při psaní této bakalářské práce, bych rád dále zúročil při svém zaměstnání v laboratoři mechatroniky na Katedře konstruování strojů Západočeské univerzity v Plzni. Část kódu, který jsem v rámci své bakalářské práce napsal, bych v budoucnu rád využil k testování provozu univerzitní robotické linky Hirata a k dalším pokusům s průmyslovým manipulátorem Mitsubishi MELFA RV - 2SD.

Část programu, která se věnuje zpracování obrazu z kamery, plánuji rozšířit o implementaci algoritmu pro automatické určení intenzity prahu z histogramu obrazu. Tím by měla rapidně vzrůst adaptivita této části programu i ve ztížených světelných podmínkách. Zároveň bych se chtěl věnovat problematice využití konvolučních neuronových sítí jako náhrady tohoto řešení. V další fázi hodlám upravit algoritmus pro analýzu scény v reálném čase, případně pro využití více než jedné kamery pro 3D vidění.

Seznam použité literatury

[1] *6-Axis Desktop Robotic Arm, Assembled* [online]. [cit. 2018-11-25]. Dostupné z: https://www.sainsmart.com/products/6-axis-desktop-robotic-arm-assembled

[2] *ABB IRC5 Pendant* [online]. [cit. 2019-01-15]. Dostupné z: https://www07.abb.com/images/default-source/robotics/irc5-data.jpg?sfvrsn=0

[3] *ABB Robotics Historical milestones* [online]. [cit. 2019-02-09]. Dostupné z: https://new.abb.com/products/robotics/home/about-us/historical-milestones

[4] *ArduinoIO* [online]. [cit. 2019-02-19]. Dostupné z: https://github.com/bensu/piezo_control/blob/master/ArduinoIO/ArduinoIO/readme.txt

[5] *Arduino Uno Rev3* [online]. [cit. 2018-11-30]. Dostupné z: https://store.arduino.cc/arduino-uno-rev3

[6] *Arduino UNO rev3* [online]. [cit. 2018-12-01]. Dostupné z: https://arduino-shop.cz/photos/produkty_gal/d/3/3050.jpg?m=1502871436

[7] *Arduino Yún* [online]. [cit. 2018-12-15]. Dostupné z: https://store.arduino.cc/arduino-yun

[8] *Arduino Yún*[online]. [cit. 2018-12-13]. Dostupné z: http://www.hobbytronics.co.uk/image/cache/data/arduino/arduino-yun-500x500.jpg

[9] *Arduino Xcos Asservissement Position* [online]. [cit. 2018-11-14]. Dostupné z: https://www.scilab.org/sites/default/files/arduino_xcos_asservissement_position_0.png?itok=CG_UJy0L

[10] *Campanile Reconstruction* [online]. [cit. 2019-01-22]. Dostupné z: https://www.pauldebevec.com/Research/campanile-reconstruction.jpg

[11] CORKE, Peter. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Haideberg: Springer-Verlag Berlin, 2011. ISBN 978-3-642-20143-1.

[12] *Corke Peter Robotics Tbx* [online]. [cit. 2018-11-26]. Dostupné z: https://i.ytimg.com/vi/hSz09WsLiCk/maxresdefault.jpg

[13] *Denavit–Hartenberg parameters* [online]. [cit. 2019-04-29]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/thumb/d/d8/DHParameter.png/519px-DHParameter.png

[14] *Doosan robotics* [online]. [cit. 2019-03-23]. Dostupné z: https://i.ytimg.com/vi/OLn3UVodtQw/maxresdefault.jpg

[15] *ESP 32 devkit* [online]. [cit. 2018-12-11]. Dostupné z: https://cdn.sos.sk/productdata/90/d5/9dcaac3b/esp32-devkitc.jpg

[16] *ESP32 Series Datasheet* [online]. [cit. 2018-12-15]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[17] *FANUC Robot series R-30iB Plus CONTROLLER: iRVision OPERATOR'S MANUAL* [online]. FANUC, 2019 [cit. 2019-02-17]. Dostupné z: http://rab.ict.pwr.wroc.pl/~malewicz/Fanuc/

[18]*FANUC Robot series R-30iB Plus CONTROLLER: iRVision2D Camera Application OPERATOR'S MANUAL* [online]. FANUC, 2019 [cit. 2019-02-17]. Dostupné z: http://rab.ict.pwr.wroc.pl/~malewicz/Fanuc/

[19]*Hirata SCARA arm* [online]. [cit. 2019-03-02]. Dostupné z: https://www.hirata.co.jp/files/bgeditor/img/401__MTM0MF9BUk0tQkFTReOCt_ODquODvOOCul9BUi1GNjUwSENz.png

[20] *Horse recognition using CNN* [online]. [cit. 2019-01-25]. Dostupné z: https://cdn-images-1.medium.com/max/1600/1*GEyZBs9pDAq2TsDp80f_Pw.png

[21] HUANG, T. S. *Computer Vision: Evolution And Promise* [online]. CERN, 1996 [cit. 2019-05-15]. ISSN 0007-8328. Dostupné z: https://cds.cern.ch/record/400313/files/p21.pdf

[22] *Industry 4.0* [online]. [cit. 2019-02-18]. Dostupné z: https://aethon.com/wp-content/uploads/2015/07/Industry40.jpg

[23] *Integrated Vision* [online]. [cit. 2018-11-25]. Dostupné z: https://new.abb.com/products/robotics/cs/aplikacni-zarizeni-a-prislusenstvi/kamerove-systemy/integrated-vision

[24] *Inverse Kinematics Multiple Solutions* [online]. [cit. 2019-02-06]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/thumb/e/e3/Inverse-kinematics-multiple-solutions.svg/2000px-Inverse-kinematics-multiple-solutions.svg.png

[25] *IRB 360 series ABB Robotics* [online]. [cit. 2019-02-25]. Dostupné z: http://www.directindustry.com/prod/abb-robotics/product-30265-169123.html

[26] *Keyence Vision Systems* [online]. [cit. 2018-11-27]. Dostupné z: https://www.medital.com/products/vision-systems-keyence

[27] *Kliniczne znaczenie poszerzonych przestrzeni okołonaczyniowych Virchowa-Robina* [online]. [cit. 2019-01-25]. Dostupné z: https://journals.viamedica.pl/polski_przeglad_neurologiczny/article/viewFile/41982/29709/76590

[28] Kolektiv autorů. *Iniciativa Průmysl 4.0* [online]. 2016 [cit. 2019-04-14]. Dostupné z: https://www.mpo.cz/assets/dokumenty/53723/64358/658713/priloha001.pdf

[29] KRETSCHMEROVÁ, Lenka a Jaroslav VLACH. *Programování v LabView v příkladech.* [online]. Liberec: Vysokoškolský podnik Liberec, 2014 [cit. 2019-03-12]. ISBN 978-80-7494-167-2. Dostupné z: https://dspace.tul.cz/bitstream/handle/15240/7158/LabVIEW_skripta.pdf?sequence=1

[30] *Labview* [online]. [cit. 2019-04-13]. Dostupné z: http://www.ni.com/cms/images/devzone/tut/image_4.jpg

[31] *Logitech Webcam C170* [online]. [cit. 2019-03-05]. Dostupné z: https://www.qatar.ourshopee.com/details/Logitech-Webcam-C170/499/

[32]MAC NAMEE, Brian. *Digital Image Processing: Introduction* [online]. [cit. 2019-04-29]. Dostupné z: http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/Image%20Processing-Introduction-Bryan-Mac-Namee.pdf

[33] *Machine Vision Toolbox* [online]. [cit. 2019-03-08]. Dostupné z: http://petercorke.com/wordpress/toolboxes/machine-vision-toolbox

[34] *Marr Algorithm* [online]. [cit. 2019-04-15]. Dostupné z: http://www.doc.gold.ac.uk/~mas02fl/MSC101/Vision/Pics/MarrImg2.gif

[35]*MG996R servo* [online]. [cit. 2019-03-07]. Dostupné z: https://www.santy.cz/data/product/216_1703.jpg

[36] *Mitsubishi Electric Robots Overview* [online]. [cit. 2019-02-25]. Dostupné z: https://www.allied-automation.com/partners/mitsubishi-electric/robots/

[37] *Mitsubishi Industrial Robot SQ/SD Series Catalog* [online]. [cit. 2018-12-28]. Dostupné z: https://mitsubishirobotics.com/pdf/MEAU_product_catalog.pdf

[38] *Mitsubishi Industrial Robot RV-2SD* [online]. [cit. 2019-01-19]. Dostupné z: http://robotics.ee.uwa.edu.au/courses/robotics/project/festo/MPS_TD_V2.4_EN/English/06_Robot/RV-2SDB/Mitsubishi%20manuals/bfp-a8790b.pdf

[39] PARKER, J. R. *Algorithms for Image Processing and Computer Vision*. 2 nd ed. IN: Wiley Publishing, 2011. ISBN 978-0-47064385-3.

[40] *Product specification Integrated Vision*. [online]. [cit. 2019-02-06]. Dostupné z: https://search-ext.abb.com/library/Download.aspx?DocumentID=3HAC046868-001&LanguageCode=en&DocumentPartId=&Action=Launch

[41] *Quick History of Machine Vision* [online]. [cit. 2019-02-10]. Dostupné z: https://www.epicsysinc.com/assets/content/images/Riehn%20added%20pictures/larryrobertsblocks.jpg

[42] *Raspberry Pi 3 Model B 64-bit 1GB RAM*[online]. [cit. 2018-12-05]. Dostupné z: http://rpishop.cz/raspberry-pi-3b/283-raspberry-pi-3-model-b-64-bit.html

[43] *Raspberry Pi 3 Model B+* [online]. [cit. 2018-12-05]. Dostupné z: https://cdn-reichelt.de/bilder/web/xxl_ws/A300/RASP_03_01.png

[44] *Robotics Toolbox* [online]. [cit. 2019-03-10]. Dostupné z: https://petercorke.com/wordpress/toolboxes/robotics-toolbox

[45] *Rosetta milion: Landing Philae on comet Churyumov-Gerasimenko* [online]. [cit. 2019-04-05]. Dostupné z: https://www.scilab.org/use-cases/rosetta-mission/

[46] *Scilab* [online]. [cit. 2019-04-05]. Dostupné z: https://www.scilab.org/

[47] SICILIANO, *Bruno. Springer handbook of robotics*. 1st ed. New York, NY: SpringerHeidelberg, 2007. ISBN 978-354-0239-574.

[48] *Stanford Arm* [online]. [cit. 2019-02-14]. Dostupné z: http://autodromo.co.uk/wp-content/uploads/2018/03/stanfordarm1.jpg

[49] SZELISKI, Richard. *Computer Vision: Algorithms and Applications* [online]. 1st ed. Heidelberg: Springer-Verlag Berlin, 2010 [cit. 2019-04-01]. ISBN 1848829345 9781848829343. Dostupné z: http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf#subsection.3.7.2

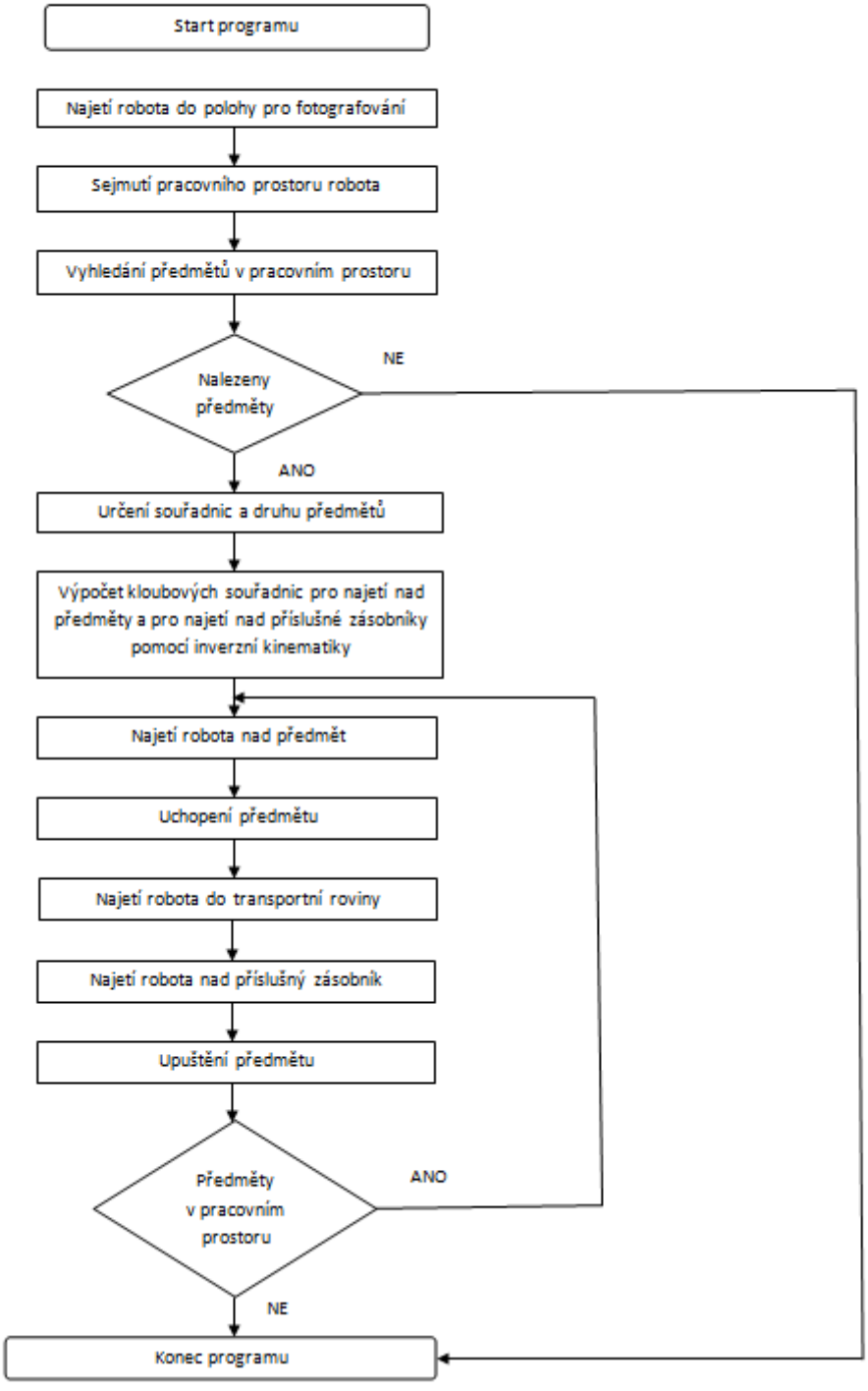
[50] *The Unimate robot was the first modern industrial robot and was used for material handling in factories.* [online]. [cit. 2019-02-12]. Dostupné z: https://www.researchgate.net/profile/Steven_Keating/publication/279815778/figure/fig8/AS:614348808548361@1523483578729/The-Unimate-robot-was-the-first-modern-industrial-robot-and-was-used-for-material.png

[51] VODA, Zbyšek. *Průvodce světem Arduina*. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7

[52] VERSLEEGERS, Wim. *Vision-based control of robotic arm with 6 degrees of freedom* [online]. 2014 [cit. 2019-05-10]. Dostupné z: https://uhdspace.uhasselt.be/dspace/handle/1942/17499. Diplomová práce. KU Leuven.

PŘÍLOHA č. 1

Vývojový diagram programu


Start programu
Najetí robota do polohy pro fotografování
Sejmutí pracovního prostoru robota
Vyhledání předmětů v pracovním prostoru
Nalezeny předměty
NE
ANO
Určení souřadnic a druhu předmětů
Výpočet kloubových souřadnic pro najetí nad předměty a pro najetí nad příslušné zásobníky pomocí inverzní kinematiky
Najetí robota nad předmět
Uchopení předmětu
Najetí robota do transportní roviny
Najetí robota nad příslušný zásobník
Upuštění předmětu
Předměty v pracovním prostoru
ANO
NE
Konec programu

PŘÍLOHA č. 2

Zdrojový kód programu

Zpracování obrazu

```
%zpracovani obrazu z kamery
%načtení obrazu z kamery
cam = VideoCamera();
im = cam.grab();
%převedeni na greyscale
x = imono(im, 'r601');
%prahovani obrazu
b = x > 100;
idisp(b);
%hledani počtu blobu
[L,nb] = ilabel(b);
ib = iblobs(b,'class',0);
%vytvoření pole hledaných prvků [X coordinát/Y coordinát/tvar]
items = [];
%výběr blobů splňujících kritéria hledaných předmětů
for i = 1:(nb)
%omezení velikosti blobů
if ib(i).area>5000 & ib(i).area<200000
ib(i).plot_box('g');
ib(i).plot('b*');
x1 = ib(i).umax;
x2 = ib(i).umin;
y1 = ib(i).vmax;
y2 = ib(i).vmin;
%výpočet plochy blobu
blobArea = ((x1-x2)*(y1-y2));
if ib(i).area > 0.78*blobArea & ib(i).area < 0.9*blobArea &
ib(i).shape>0.9
thisItem = [ib(i).uc,ib(i).vc,1]; %1 - odpovídá kruhu
elseif ib(i).shape < 0.85
thisItem = [ib(i).uc,ib(i).vc,2]; %2 - odpovídá obdélníku
else
thisItem = [ib(i).uc,ib(i).vc,3]; %3 - odpovídá čtverci
```

```
end
%přidání předmětu do seznamu
items = [items; thisItem];
end
end
%vypíše objekty do konzole
disp(items)
```

Model MeArm

```
%definovani clanku MeArm
%link1
alfa1 = pi/3;
a1 = 0.2;
d1 = 0.3;
%link2
a2 = 0.8;
d4 = 0.743;
%link3
alfa3 = pi/3;
a3 = 0.0;

%DH parametry
%           theta   d       a         alpha
L(1) = Link([ 0     d1      a1        pi/2    0], 'standard');
L(2) = Link([ 0     0       a2         0      0], 'standard');
L(3) = Link([ 0     0       a3         0      0], 'standard');

%symbolicke vyjadreni transformaci
syms theta1 theta2 theta3  a1 a2 a3 d1 alfa1 alfa3

T10 =
trotz(theta1)*transl(0,0,d1)*transl(a1,0,0)*round(trotx(alfa1)
);
T21 = trotx(theta2)*transl(a2,0,0);
T32 = trotz(theta3)*round(trotx(alfa3));
```

```
%celkova transformace
T = T10*T21*T32


%limity kloubů
q_lim = [ -pi/2      pi/2;
         -pi*4/9    pi*4/9;
            0       pi*4/9;
                           ]
%vytvoření manipulatoru
mearm =
SerialLink(L,'name','MeArm','manufacturer','Unknown','qlim',q_
lim);
%najeti do pozice pro focení
f = transl(70,0,40);
r=mearm.ikine(f, [0,0,0], [1 1 1 0 0 0] )
a.servoWrite(2,r(1)*(pi/180)-(pi/2));
a.servoWrite(3,r(2)*(pi/180)-(pi/2));
a.servoWrite(4,r(3)*(pi/180)-(pi/2));
```

Model SainSmart

```
%symbolicke vyjadreni
 syms theta1 theta2 theta3 theta4 theta5 theta6 alfa1 alfa2
alfa3 alfa4 alfa5 alfa6 a1 a2 a3 a4 a5 a6 d1 d2 d3 d4 d5 d6
%transformace
T10 =
trotz(theta1)*transl(0,0,d1)*transl(a1,0,0)*round(trotx(alfa1)
);
T21 = trotz(theta2)*transl(a2,0,0);
T32 = trotz(theta3)*round(trotx(alfa3));
T43 = trotz(theta4)*transl(0,0,d4)*round(trotx(alfa4));
T54 = trotz(theta5)*transl(a5,0,0)*round(trotx(alfa5));
T65 = trotz(theta6)*round(trotx(alfa6));
%uplna transformace
T = T10*T21*T32*T43*T54*T65

%DH parametry

%              theta   d       a          alfa    vazba
L(1) = Link([   0     110    40    -pi/2       0],'standard');
L(2) = Link([ -pi/2    0     127     0         0],'standard');
L(3) = Link([ -pi/2    0     26     pi/2       0],'standard');
L(4) = Link([   0     133    0      pi/2       0],'standard');
L(5) = Link([   0      0     0     -pi/2       0],'standard');
L(6) = Link([   pi     25    0       0         0],'standard');
```

```
%limitni souradnice
q_lim = [-pi/2  pi/2;
         -pi/2  pi/2;
         -pi/2  pi/2;
         -pi/2  pi/2;
         -pi/2  pi/2;
         -pi/2  pi/2;
                    ]
%vytvoreni robota
sainsmart =
SerialLink(L,'name','SainSmart','manufacturer','SainSmart','ql
im',q_lim);
%najeti do pozice pro focení
f = transl(120,0,70);
r=sainsmart.ikine(f, [0,0,0,0,0,0], [1 1 1 0 0 0] )
a.servoWrite(2,r(1)*(pi/180)-(pi/2));
a.servoWrite(3,r(2)*(pi/180)-(pi/2));
a.servoWrite(4,r(3)*(pi/180)-(pi/2));
a.servoWrite(5,r(4)*(pi/180)-(pi/2));
a.servoWrite(6,r(5)*(pi/180)-(pi/2));
a.servoWrite(7,r(6)*(pi/180)-(pi/2));
```

Model Melfa

```
%symbolicke vyjadreni
syms theta1 theta2 theta3 theta4 theta5 theta6 alfa1 alfa2
alfa3 alfa4 alfa5 alfa6 a1 a2 a3 a4 a5 a6 d1 d2 d3 d4 d5 d6
%transformace
T10 =
trotz(theta1)*transl(0,0,d1)*transl(a1,0,0)*round(trotx(alfa1)
);
T21 = trotz(theta2)*transl(a2,0,0);
T32 = trotz(theta3)*round(trotx(alfa3));
T43 = trotz(theta4)*transl(0,0,d4)*round(trotx(alfa4));
T54 = trotz(theta5)*transl(a5,0,0)*round(trotx(alfa5));
T65 = trotz(theta6)*round(trotx(alfa6));
%uplna transformace
T = T10*T21*T32*T43*T54*T65
%DH parametry
%             theta   d       a     alfa     vazba
L(1) = Link([   0     295     0    -pi/2        0],'standard');
L(2) = Link([ -pi/2    0      230    0          0],'standard');
L(3) = Link([ -pi/2    0      50   -pi/2        0],'standard');
L(4) = Link([   0     270      0    pi/2        0],'standard');
L(5) = Link([   0      0       0   -pi/2        0],'standard');
L(6) = Link([   pi    70       0     0          0],'standard');
```

```
%limitni souradnice
q_lim = [-4/3*pi 4/3*pi;
         -2/3*pi 2/3*pi;
         0       8/9*pi;
         -10/9*pi 10/9*pi;
         -2/3*pi 2/3*pi;
         -2*pi 2*pi;
            ]
%vytvoreni robota
melfa = SerialLink(L,'name','MELFA RV -
2SD','manufacturer','Mitsubishi','qlim',q_lim);
```

Inverzní kinematika 3 DoF

```
%Definice prejezdového bodu
p1 = transl(70,0,50);
%zasobnik valečky
p2 = transl(0,-70,70);
%zasobnik kostky a hranoly
p3 = transl(0,70,70);
%vyska kamery
h = 200;
%vzdalenost kamery
l=150;
%korekce x
x1=2*h*tan(29*(pi/180))*cos(atan(768/1024));
x2=cos(atan(y1/x1))*(sqrt(x1^2+y1^2)-18);
%korekce y
y1=2*h*tan(29*(pi/180))*sin(atan(768/1024));
y2=sin(atan(y1/x1))*(sqrt(x1^2+y1^2)-18);
for i = 1:(size(items)/3)
%najeti nad cil
x = items((i-1)*3+1)+x2;
y = items((i-1)*3+2)+y2;
z = 30;
pcil = transl(x,y,z);
pinv = mearm.ikine(pcil, p1, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
%sepnout efektor ON
pause(1);
pinv = mearm.ikine(p1, pcil, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
%najeti nad zasobniky
if items((i-1)*3+1) = 1
pinv = mearm.ikine(p2, p1, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
```

```
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
%sepnout efektor OFF
pause(1);
pinv = mrarm.ikine(p1, p2, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
else
pinv = mearm.ikine(p4, p1, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
%sepnout efektor OFF
pause(1);
pinv = mearm.ikine(p1, p4, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
end
end
%najetí do klidové polohy a konec
f = transl(70,20,50);
r=mearm.ikine(f, ptrans, [1 1 1 0 0 0] )
a.servoWrite(2,r(1)*(pi/180)-(pi/2));
a.servoWrite(3,r(2)*(pi/180)-(pi/2));
a.servoWrite(4,r(3)*(pi/180)-(pi/2));
disp('Program dokončen.')
```

Inverzní kinematika 6 DoF (Sainsmart)

```
%Definice prejezdového bodu
p1 = transl(100,0,90);
%zasobnik valečky
p2 = transl(-100,0,70);
%zasobnik kostky
p3 = transl(100,0,70);
%zasobnik hranoly
p4 = transl(50,0,70);
%vyska kamery
h = 300;
%vzdalenost kamery
l=200;
%korekce x
x1=2*h*tan(29*(pi/180))*cos(atan(768/1024));
%korekce y
y1=2*h*tan(29*(pi/180))*sin(atan(768/1024));
%najeti na prejezdovy bod
pinv = sainsmart.ikine(p1, f, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
```

```
a.servoWrite(5,pinv(4)*(pi/180)-(pi/2));
a.servoWrite(6,pinv(5)*(pi/180)-(pi/2));
a.servoWrite(7,pinv(6)*(pi/180)-(pi/2));
for i = 1:(size(items)/3)
%najeti nad cil
x = items((i-1)*3+1)+x1;
y = items((i-1)*3+2)+y1;
z = 30;
pcil = transl(x,y,z);
pinv = sainsmart.ikine(pcil, p1, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
a.servoWrite(5,pinv(4)*(pi/180)-(pi/2));
a.servoWrite(6,pinv(5)*(pi/180)-(pi/2));
a.servoWrite(7,pinv(6)*(pi/180)-(pi/2));
%sepnout efektor ON
pause(1);
pinv = sainsmart.ikine(p1, pcil, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
a.servoWrite(5,pinv(4)*(pi/180)-(pi/2));
a.servoWrite(6,pinv(5)*(pi/180)-(pi/2));
a.servoWrite(7,pinv(6)*(pi/180)-(pi/2));
%najeti nad zasobniky
if items((i-1)*3+1) = 1
pinv = sainsmart.ikine(p2, p1, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
a.servoWrite(5,pinv(4)*(pi/180)-(pi/2));
a.servoWrite(6,pinv(5)*(pi/180)-(pi/2));
a.servoWrite(7,pinv(6)*(pi/180)-(pi/2));
%sepnout efektor OFF
pause(1);
pinv = sainsmart.ikine(p1, p2, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
a.servoWrite(5,pinv(4)*(pi/180)-(pi/2));
a.servoWrite(6,pinv(5)*(pi/180)-(pi/2));
a.servoWrite(7,pinv(6)*(pi/180)-(pi/2));
elseif items((i-1)*3+1) = 2
pinv = sainsmart.ikine(p3, p1, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
a.servoWrite(5,pinv(4)*(pi/180)-(pi/2));
a.servoWrite(6,pinv(5)*(pi/180)-(pi/2));
a.servoWrite(7,pinv(6)*(pi/180)-(pi/2));
```

```
%sepnout efektor OFF
pause(1);
pinv = sainsmart.ikine(p1, p3, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
a.servoWrite(5,pinv(4)*(pi/180)-(pi/2));
a.servoWrite(6,pinv(5)*(pi/180)-(pi/2));
a.servoWrite(7,pinv(6)*(pi/180)-(pi/2));
else
pinv = sainsmart.ikine(p4, p1, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,(pinv(3)-pinv(2))*(pi/180)-(pi/2));
a.servoWrite(5,pinv(4)*(pi/180)-(pi/2));
a.servoWrite(6,pinv(5)*(pi/180)-(pi/2));
a.servoWrite(7,pinv(6)*(pi/180)-(pi/2));
%sepnout efektor OFF
pause(1);
pinv = sainsmart.ikine(p1, p4, [1 1 1 0 0 0] );
a.servoWrite(2,pinv(1)*(pi/180)-(pi/2));
a.servoWrite(3,pinv(2)*(pi/180)-(pi/2));
a.servoWrite(4,pinv(3)*(pi/180)-(pi/2));
a.servoWrite(5,pinv(4)*(pi/180)-(pi/2));
a.servoWrite(6,pinv(5)*(pi/180)-(pi/2));
a.servoWrite(7,pinv(6)*(pi/180)-(pi/2));
end

end

f = transl(70,20,50);
%najetí do klidové polohy a konec
r=sainsmart.ikine(f, ptrans, [1 1 1 0 0 0] )
a.servoWrite(2,r(1)*(pi/180)-(pi/2));
a.servoWrite(3,r(2)*(pi/180)-(pi/2));
a.servoWrite(4,(r(3)-r(2))*(pi/180)-(pi/2));
a.servoWrite(5,r(4)*(pi/180)-(pi/2));
a.servoWrite(6,r(5)*(pi/180)-(pi/2));
a.servoWrite(7,r(6)*(pi/180)-(pi/2));
disp('Program dokončen.')
```

Řídící program

```
clc;clear;
%volba robota
prompt = 'Jaký je testovaný model?'
disp('A - MeArm')
disp('B - SainSmart')
disp('C - Melfa')
```

```
x = input(prompt,'s');
if x == 'a' || x == 'A'
   run mearm_odevzdani;
elseif x == 'b' || x == 'B'
  run sainsmart_odevzdani;
elseif x == 'c' || x == 'C'
    run melfa_odevzdani;
end
%navazani spojeni s arduinem
a = arduino('COM11');
a.servoAttach(2);
a.servoAttach(3);
a.servoAttach(4);
a.servoAttach(5);
a.servoAttach(6);
a.servoAttach(7);
%zpracovani obrazu
run vision_odevzdani;
%spuštění programu pro výpočet trajektorie robota MeArm
if x == 'a' || x == 'A'
    run inverzni_3dof_odevzdani;
end
 %spuštění programu pro výpočet trajektorie robota SainSmart
else
    run inverzni_6dof_odevzdani;
end
```

PŘÍLOHA č. 3

Výkres manipulátoru Mitsubishi MELFA RV – 2SD

2.4 Outside dimensions・Operating range diagram

(1) RV-2SD (standard specification)

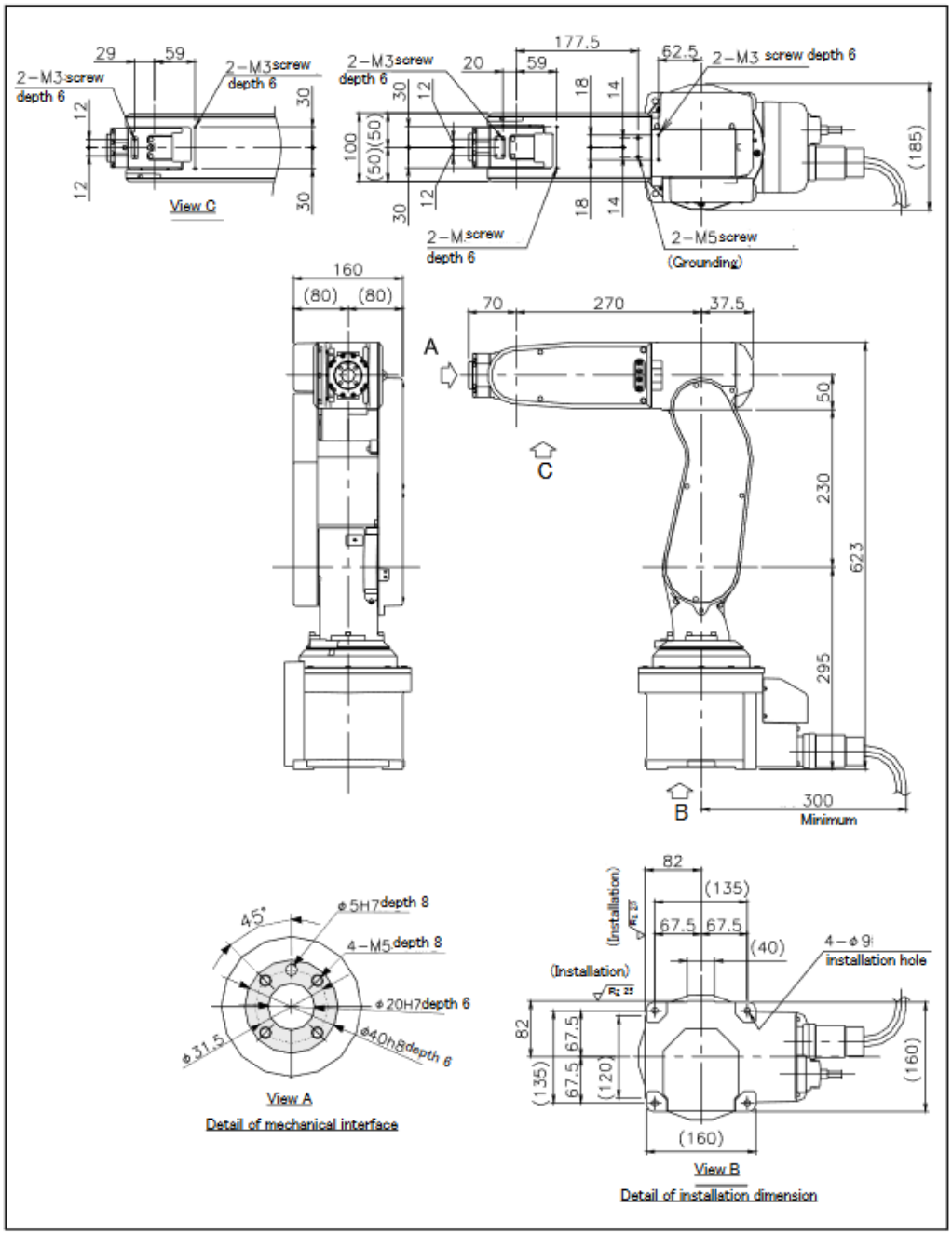


Fig.2-3 : Outside dimensions (Standard specification)

PŘÍLOHA č. 4

Popis vybraných algoritmů pro prahování obrazu (Bylo kompletně převzato z [39])

Using Edge Pixels

An edge pixel must be near to the boundary between an object and the background, or between two objects; that is why it is an edge pixel. As a result, the levels of the edge pixels are likely to be more consistent. Because They will sometimes be inside the object and sometimes be a little outside due to sampling concerns, the histogram of the levels of the edge pixels will be more regular than the overall histogram.

This idea was used to produce a thresholding method based on the digital Laplacian, which is a non-directional edge-detection operator [Weszka, 1974]. The threshold is found by first computing the Laplacian of the input image.

There are many ways to do this, but a simple one is to convolve the image with the mask:

$$\begin{matrix} 0 & 1 & 0 \\ 0 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Now a histogram of the original image is found considering only those pixels having large Laplacians; those in the 85th percentile and above will do nicely. Those pixels having a Laplacian greater than 85% of their peers will have thei rgrey level appear in the histogram, whereas all other pixels will not. Now thethreshold is selected using the histogram thus computed. Using a better approximation to the Laplacian should give better results,but in many cases this simple procedure will show an improvement over the previous methods. The C program on the website that computes a threshold using this method is calledthrlap.c, and requires that the user enter the percentage value used to select the Laplacian values.

[39] str. 139 - 140

The Method of Grey-Level Histograms

The thresholding methods based on selecting the low point between two histogram peaks use the concept that object pixels and background pixels have different mean levels, and are random numbers drawn from one of two normal distributions. These distributions also have their own standard deviations andvariances, where variance is the square of the standard deviation. If there are two groups of pixels in the image, as suggested, then it is a simple matter to compute the overall, ortotal, variance of the grey level values in the image, denoted by $\sigma_t{}^2$. For any given threshold t, it is also possible to separately compute the variance of the object pixels and of the background pixels; these represent the within-class variance values, denoted by $\sigma_w{}^2$. Finally, the variation of the mean values for each class from the overall mean of all pixels defines a between-classes variance, which will be denoted by $\sigma_b{}^2$. This is the beginning of a method in statistics called analysis of variance, but we will not go too much further with it here. The important issue is that an optimal (in some respects) threshold can be found by minimizing the ratio of the between-class variance to the total variance [Otsu, 1979]; that is,

$$\eta(t) = \frac{\sigma_b{}^2}{\sigma_t{}^2}$$

defines the needed ratio, and the value of t that gives the smallest value for his the best threshold. Since $\sigma_t{}^2$ is the overall variance it is easy to calculate from the image, as is the overall mean μT. The between class variance is calculated by:

$$\sigma_b{}^2 = \omega_0\omega_1(\mu_0\mu_1)^2$$

where:

$$\omega_0 = \sum_{i=0}^{t} p_i$$

$$\omega_1 = 1 - \omega_0$$

and pi is the probability of grey level i, or the histogram value at i divided by the total number of pixels. Also,

$$\mu_0 = \frac{\mu_t}{\omega_0}$$

$$\mu_1 = \frac{\mu_T - \mu_t}{1 - \omega_0}$$

$$\mu_t = \sum_{i=0}^{t} i \cdot p_i$$

All these values are quite easy to calculate from the histogram h of the image. Then η(t) is computed for all possible values of t, and the t that oves the smallest value of η(t) is the optimal threshold. There is a program called *thrglh.c* on the website that thresholds an image using this method. It is run in exactly the same was as thris. A recent development is a fast version of this algorithm that gives the same results [Dong 2008].

[39] str. 141 - 142

Minimum Error Thresholding

The histogram of the image can be thought of as a measured probability density function of the two distributions (object pixels and background pixels). These are, as has been discussed, usually thought of as normal distributions, so the histogram is an approximation to

$$p(g) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\left(\frac{(g-\mu_1)^2}{2\sigma_1}\right)} + \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\left(\frac{(g-\mu_2)^2}{2\sigma_2}\right)}$$

Where σ_1 and μ_1 are the standard deviation and mean of one of the classes, and σ_2 and μ_2 are the standard deviation and mean of the other. After taking the log of both sides and rearranging, we get a quadratic equation that could be solved for g:

$$\frac{(g-\mu_1)^2}{\sigma_1{}^2} + log\sigma_1 - 2logP_1 = \frac{(g-\mu_2)^2}{\sigma_2{}^2} + log\sigma_2 - 2logP_2$$

However, the values of σ, μ, and P are not known, and they can be estimated only with some difficulty. Instead of that, Kittler and Illingworth [Kittler, 1986] created a new criterion function to be minimized:

$$J(t) = 1 + 2\big(P_1(t)log\sigma_1(t) + P_2(t)log\sigma_2(t)\big) - 2(P_1(t)logP_1(t) + P_2(t)logP_2(t))$$

using formulas that should be starting to look familiar:

$$P_1(t) = \sum_{g=0}^{t} h(g)$$

$$P_2(t) = \sum_{g=t+1}^{255} h(g)$$

$$\mu_1(t) = \frac{\sum_{g=0}^{t} g \cdot h(g)}{P_1(t)}$$

$$\mu_2(t) = \frac{\sum_{g=t+1}^{255} g \cdot h(g)}{P_2(t)}$$

$${\sigma_2}^2(t) = \frac{\sum_{g=0}^{t} h(g) \cdot (g - \mu_1(t))^2}{P_1(t)}$$

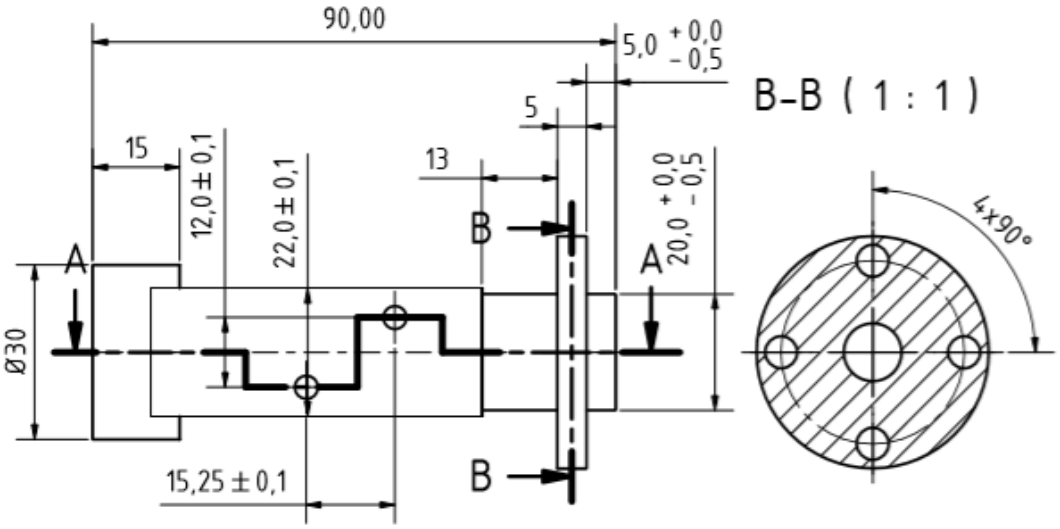
$${\sigma_2}^2(t) = \frac{\sum_{g=t+1}^{255} h(g) \cdot (g - \mu_2(t))^2}{P_2(t)}$$

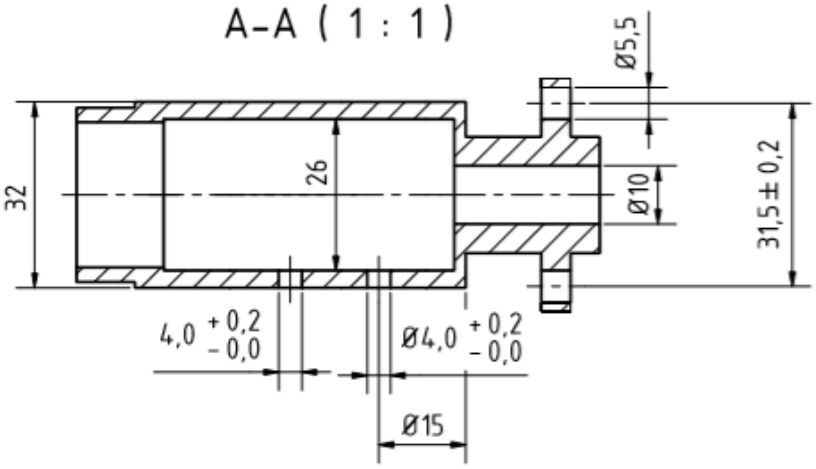
The value of t that minimizes J(t) is the best threshold. This is often referred to as minimum error thresholding.

[39] str. 148 - 149

PŘÍLOHA č. 5

Výkresová dokumentace uchopovačů





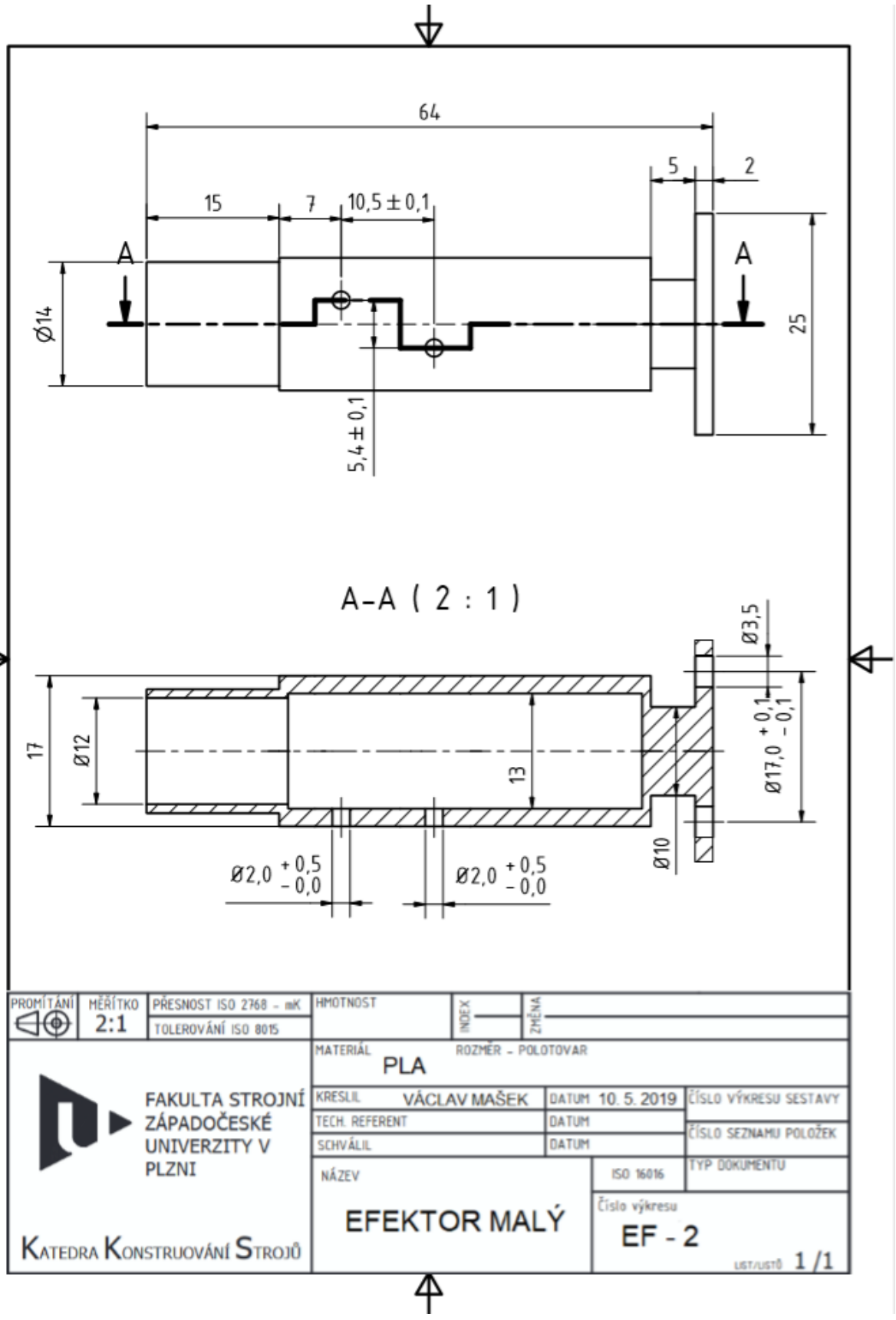
PROMÍTÁNÍ	MĚŘÍTKO	PŘESNOST ISO 2768 – mK	HMOTNOST	INDEX	ZMĚNA
	1:1	TOLEROVÁNÍ ISO 8015			

FAKULTA STROJNÍ
ZÁPADOČESKÉ
UNIVERZITY V
PLZNI

KATEDRA KONSTRUOVÁNÍ STROJŮ

MATERIÁL: PLA — ROZMĚR – POLOTOVAR

KRESLIL: VÁCLAV MAŠEK	DATUM: 10. 5. 2019	ČÍSLO VÝKRESU SESTAVY	
TECH. REFERENT	DATUM	ČÍSLO SEZNAMU POLOŽEK	
SCHVÁLIL	DATUM		
NÁZEV: EFEKTOR VELKÝ	ISO 16016	TYP DOKUMENTU	
	Číslo výkresu: EF- 1	LIST/LISTŮ 1 /1	



PROMÍTÁNÍ	MĚŘÍTKO 2:1	PŘESNOST ISO 2768 - mK TOLEROVÁNÍ ISO 8015	HMOTNOST	INDEX	ZMĚNA

MATERIÁL: PLA — ROZMĚR - POLOTOVAR

FAKULTA STROJNÍ ZÁPADOČESKÉ UNIVERZITY V PLZNI

KATEDRA KONSTRUOVÁNÍ STROJŮ

KRESLIL: VÁCLAV MAŠEK	DATUM: 10. 5. 2019	ČÍSLO VÝKRESU SESTAVY	
TECH. REFERENT	DATUM	ČÍSLO SEZNAMU POLOŽEK	
SCHVÁLIL	DATUM		
NÁZEV: EFEKTOR MALÝ	ISO 16016	TYP DOKUMENTU	
	Číslo výkresu: EF - 2	LIST/LISTŮ 1 /1	

PŘÍLOHA č. 6

Komunikace MATLABu s řídící jednotkou Mitsubishi MELFA RV – 2SD

Pro komunikaci mezi MATLABem a robotem je využit postup publikovaný v práci [52], která byla na Západočeské univerzitě zpracována v roce 2014. Komunikace probíhá přes sériovou linku a zpracování sejmuté scény je prováděno dávkově. Zde je uveden příklad sériové komunikace:

```
%% Otevření seriové komunikace
fopen(s);
% Zahájení komunikace s řídící jednotkou
sendCommand(s,'OPEN=USERTOOL');
% Inicializace
sendCommand(s,'CNTLON');
% Otevreni programu TRAN pro zápis
sendCommand(s,'LOAD=TRAN');
% Vložení řádku - nastavení rychlosti, a zapsání do programu
TRAN
command = 'EDATA1 OVRD 20';
sendCommand(s,command);
% Vložení jednotlivých bodů trajektorie = vypočítaných
konfigurací robota
% v kloubových souřadnicích a zapsání do programu TRAN
for i = 2:length+1 %length+2:length*2+1
b = conformations{i-1};
command = …
sprintf('EDATA%dJ%d=(%0.3f,%0.3f,%0.3f,%0.3f,%0.3f,%0.3f,0.000
,0.000)', …
i,i-1, b(1), b(2), b(3), b(4), b(5), b(6));
sendCommand(s,command);
end
% Vložení příkazů MOV mezi jednotlivými body trajektorie =
konfiguracemi robota
% v kloubových souřadnicích a zapsání do programu TRAN
for i = length+2:length*2+1
command = sprintf('EDATA%d MOV J%d',i,i-length-1);
sendCommand(s,command);
end
% Ukončení programu
command = sprintf('EDATA%d END',length*2 + 2);
sendCommand(s,command);
% Uložení programu TRAN do řídící jednotky
sendCommand(s,'SAVE');
% Zapnutí serv
sendCommand(s,'SRVON');
% Spuštění programu TRAN v řídící jednotce robota
sendCommand(s,'RUNTRAN;1');
% Uzavření seriové komunikace a uvolnění sériového portu
fclose(s);
```