

Optoelektronické polohování nástroje vůči rovině ve více stupních volnosti s využitím v oblasti bezpečnostních technologií

Optoelectronic Positioning of Tool to Plane Surface in Multiple
Degrees of Freedom with Application in Safety Technologies

Bc. Tomáš Gavenda

Diplomová práce
2011



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Tomáš GAVENDA**
Osobní číslo: **A09360**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Bezpečnostní technologie, systémy a management**

Téma práce: **Optoelektronické polohování nástroje vůči rovině ve více stupních volnosti s využitím v oblasti bezpečnostních technologií**

Zásady pro vypracování:

1. Popište problematiku optoelektronického polohování nástroje vůči rovině ve více stupních volnosti.
2. Analyzujte možnosti implementace na šestiosém robotu STÄUBLI UNIMATION TX40.
3. Realizujte vlastní implementaci.
4. Demonstrujte ukázkovou aplikaci.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. FORSYTH, David A.; PONCE, Jean. Computer Vision : A Modern Approach. Upper Saddle River : Pearson Education, Inc., 2003. 693 s. ISBN 0-13-085198-1.
2. ŽÁRA, Jiří, et al. Moderní počítačová grafika. Brno : Computer Press, 2004. 612 s. ISBN 80-251-0454-0.
3. DUDEK, Gregory; JENKIN, Michael. Computational Principles of Mobile Robotics. New York : Cambridge University Press, 2000. 280 s. ISBN 0-521-56876-5.
4. CHOSET, Howie, et al. Principles of Robot Motion : Theory, Algorithms, and Implementation. Cambridge : The MIT Press, 2005. 603 s. ISBN 0-262-03327-5.
5. Arm – TX series 40 family : Instruction manual. Faverges : Stäubli Faverges SCA, 2007. 84 s.
6. VAL3 Reference Manual : Version 6. Faverges : Stäubli Faverges SCA, 2008. 186 s.
7. CMUcam [online]. 2007, poslední změna 2.8.2010 [cit. 2011-01-31]. Documentation – CMUcam. Dostupné z WWW: [<http://cmucam.org/wiki/Documentation>].

Vedoucí diplomové práce:

Ing. Erik Král

Ústav bezpečnostního inženýrství

Datum zadání diplomové práce:

25. února 2011

Termín odevzdání diplomové práce:

27. května 2011

Ve Zlíně dne 25. února 2011

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. RNDr. Vojtěch Křesálek, CSc.
ředitel ústavu

ABSTRAKT

Účelem práce bylo navržení vlastního řešení optoelektronického polohování nástroje vůči rovině ve více stupních volnosti a návrh využití tohoto řešení v rámci bezpečnostních technologií. Teoretická část podává základní informace o zpracování obrazu kamerou a rozebírá technické specifikace použité inteligentní kamery a šestiosého robota. V praktické části je celé vlastní řešení daného tématu. Byl sestrojen vhodný nástroj, který je robotem polohován. Dále byly vyvinuty a naprogramovány jednotlivé metody nalezení bodů a sestrojení přímek, ty byly testovány a analyzovány, poté došlo k výběru vhodné metody zpracování obrazové informace. Z matematického modelu a vypočtených vztahů byla odvozena metoda polohování, která je implementována na šestiosém robotu a na závěr práce je navržena a demonstrována ukázková aplikace v bezpečnostním průmyslu.

Klíčová slova: optoelektronické polohování, inteligentní kamera CMUcam3, zpracování obrazu, perspektivní projekce, šestiosý robot Stäubli Unimation TX40, servisní robot

ABSTRACT

The purpose of the thesis was to propose own solution to optoelectronic positioning of tool to the plane surface in multiple degrees of freedom and design application of this solution in the scope of safety technologies. The theoretical part gives basic information about the image processing in camera and discusses technical specifications for smart cameras and six-axis robot. The practical part is all own solution to the issue. A suitable tool, which is positioned by the robot, was constructed. The various methods to find points and construct lines have been developed and programmed, they were tested and analyzed, then was selected the appropriate method for image information processing. From the mathematical model and calculated relationships were derived positioning method that is implemented in the six-axis robot and in the ending part of the thesis is proposed and demonstrated exemplary application in the safety industry.

Keywords: optoelectronic positioning, smart camera CMUcam3, image processing, perspective projection, six-axis robot Stäubli Unimation TX40, service robot

Rád bych poděkoval především svému vedoucímu diplomové práce, panu Ing. et Ing. Eriku Královi, za neutuchávající podporu a motivaci, poskytnutí materiálů a zdrojů informací a za celkové vedení při práci. Dále děkuji panu Ing. Petru Navrátilovi, Ph.D. za cenné rady ohledně průmyslového robota Stäubli Unimation TX40 a odbornou pomoc při práci. V neposlední řadě bych rád, in memoriam, poděkoval panu doc. Ing. Mgr. Milanu Kvasnicovi, CSc. za přivedení do oboru robotiky a obrovskou počáteční motivaci pro práci s roboty.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 PROSTŘEDKY PRO ŘEŠENÍ PROBLEMATIKY POLOHOVÁNÍ	11
1.1 BAREVNÉ PROSTORY	11
1.1.1 Monochromatický barevný prostor	11
1.1.2 Prostor RGB	11
1.1.3 Prostor $Y C_B C_R$	12
1.1.4 Prostor HSV	13
1.2 GEOMETRICKÝ MODEL KAMERY	14
1.2.1 Perspektivní projekce	15
1.2.2 Vztahy analytické geometrie.....	16
1.3 ALGORITMY PRO VYHLEDÁNÍ PŘÍMEK.....	17
1.3.1 Split and Merge.....	17
1.3.2 Inkrementální algoritmus	17
1.3.3 RANSAC.....	18
1.3.4 Houghova transformace	18
2 INTELIGENTNÍ KAMERA CMUCAM3	19
2.1 TECHNICKÝ POPIS INTELIGENTNÍ KAMERY CMUCAM3	19
2.2 MOŽNOSTI KAMERY CMUCAM3	20
2.2.1 Emulace CMUcam2	20
3 PRŮMYSLOVÝ ROBOT STÄUBLI UNIMATION TX40	21
3.1 ROBOTICKÁ PAŽE ARM TX40	21
3.2 ŘÍDICÍ JEDNOTKA CS8C	22
3.2.1 Ovládací panel MCP	23
3.2.2 Přepínač pracovního módu.....	23
3.3 PROGRAMOVACÍ JAZYK VAL3	24
4 VYUŽITÍ SERVISNÍCH ROBOTŮ V BEZPEČNOSTNÍM PRŮMYSLU	26
4.1 SOUČASNÉ VYUŽITÍ SERVISNÍCH ROBOTŮ V BEZPEČNOSTNÍM PRŮMYSLU.....	26
II PRAKTICKÁ ČÁST	28
5 NÁSTROJ ROBOTICKÉ PAŽE	29
5.1 LASIRIS SNF LASER.....	30
6 MATEMATICKÝ MODEL	31
7 ZPRACOVÁNÍ OBRAZOVÉ INFORMACE	34
7.1 NASTAVENÍ INTELIGENTNÍ KAMERY	34
7.2 TVORBA ZÁKLADNÍCH STRUKTUR	36
7.3 VYHLEDÁNÍ BODŮ.....	37
7.3.1 Metoda nalezení okrajových bodů.....	38
7.3.2 Metoda nalezení všech bodů	39
7.3.3 Naprogramování metod nalezení bodů světelného obrazce v jazyce C	40
7.4 SESTROJENÍ PŘÍMEK	44
7.4.1 Metoda s použitím lineární regrese	44
7.4.2 Metoda založená na algoritmu RANSAC	47

7.4.3	Metoda pracující na bázi upraveného inkrementálního algoritmu	48
7.4.4	Metoda využívající všech bodů a algoritmu RANSAC	49
7.5	NALEZENÍ ČTYŘÚHELNÍKU	49
7.6	ODVOZENÍ POLOHY NÁSTROJE.....	50
7.6.1	Měření pro polohování nástroje.....	50
7.6.2	Zvolená metoda polohování.....	53
7.7	KOMUNIKACE KAMERY S ROBOTEM.....	54
8	PROGRAMOVÁNÍ ROBOTA STÄUBLI UNIMATION TX40.....	55
8.1	NASTAVENÍ NÁSTROJE A SOUŘADNÉHO SYSTÉMU	55
8.2	PŘÍJEM DAT Z KAMERY	56
8.3	APLIKACE PRO POLOHOVÁNÍ NÁSTROJE.....	56
8.3.1	Hlavní program aplikace	57
9	VYUŽITÍ ŘEŠENÍ V BEZPEČNOSTNÍM PRŮMYSLU	60
9.1	MODELOVÁ APLIKACE ŘEŠENÍ.....	60
	ZÁVĚR	61
	ZÁVĚR V ANGLIČTINĚ.....	63
	SEZNAM POUŽITÉ LITERATURY	65
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	68
	SEZNAM OBRÁZKŮ	70
	SEZNAM TABULEK.....	71
	SEZNAM PŘÍLOH.....	72

ÚVOD

Úvodní část práce má za úkol seznámit čtenáře s důvody, proč bylo dané téma vybráno k řešení a uvést cíle, které byly předsevzaty ke splnění, a které byly považovány při řešení daných úloh za důležité. V úvodní části práce je také představeno členění celé práce – teoretické a praktické části.

Nejvýznamnějším důvodem ke zvolení tématu byla možnost kompletní vlastní realizace řešení, od jeho počátku, až po konec. Pro studenta oboru Bezpečnostních technologií, systému a managementu, je zajímavá a důležitá také možnost implementace řešení do oblasti bezpečnosti, což byl další důvod k výběru.

Optoelektronické polohování nástroje je v rámci práce prováděno s pomocí dostupných komponent, které již obsahoval inventář školy, a vyznačují se svou kompaktností, dostupností a cenou. To sice přineslo řadu problémů, ale bylo výzvou se těmto problémům postavit a řešit je. V práci je kladen důraz na jednoduchá, funkční, rychlá a zejména přenositelná řešení.

Hlavním cílem práce bylo vytvořit funkční soustavu s optoelektronickou senzorickou částí na jedné straně a akčním členem – průmyslovým robotem vykonávajícím příkazy, na straně druhé. Ke splnění tohoto cíle vedla cesta od zvolení vhodného teoretického základu, přes volbu fungujících algoritmů a vytvoření na nich založených metod, až po komunikaci senzorické části a průmyslového robota, který je naprogramován k vykonání obdržených příkazů. Dalším cílem je pak využití daného tématu v oblasti bezpečnostního průmyslu.

Celá diplomová práce se dělí na teoretickou a praktickou část. V teoretické části jsou uvedeny všechny potřebné podklady pro řešení celé problematiky a v praktické části pak samotné řešení, doplněné zdrojovými kódy a obrazovou dokumentací. Praktická část je rozdělena do kapitol tak, jak následovaly jednotlivé úseky řešení časově za sebou. Součástí praktické části a závěru práce je pak naplnění posledního cíle – tedy možnosti zapracování tématu do oblasti bezpečnostních technologií.

Nyní následuje první kapitola teoretické části, která začíná popisem problematiky optoelektronického polohování nástroje ve více stupních volnosti.

I. TEORETICKÁ ČÁST

1 PROSTŘEDKY PRO ŘEŠENÍ PROBLEMATIKY POLOHOVÁNÍ

Polohování může být řešeno například pomocí ultrazvukových senzorů či infračervených senzorů, díky nimž je získána vzdálenost bodů, ve kterých jsou umístěny, od okolního světa. Další možností polohování je polohování pomocí stereovize, kdy je dvěma kamerami sledována a vyhodnocována daná scéna a jsou určeny vzdálenosti předmětů v ní. Polohování lze rovněž vyřešit díky použití hloubkoměrného principu se strukturovaným světlem [1]. Tato práce se však zabývá řešením pomocí zpracování viditelného světelného obrazce kamerou, tedy optoelektronickým polohováním.

Aby bylo možné řešit optoelektronické polohování nástroje vůči rovině ve více stupních volnosti, je nutné seznámit se s touto problematikou hned z několika různých hledisek. K samotnému polohování je třeba snímat prostor, ve kterém se budeme pohybovat a určovat v něm polohu. Tomuto účelu slouží kamera, která snímá laserovou stopu, promítanou na rovinu, vůči které polohujeme nástroj. Následující řádky, se věnují teorii pro práci s obrazovou informací, jež je zpracovávána kamerou, resp. jejím mikropočítačem, vybranou inteligentní kamerou a informacemi o šestiosém kloubovém robotu Stäubli Unimation TX40 [2].

1.1 Barevné prostory

Každé zařízení zpracovávající obrazovou informaci pracuje v určitém barevném prostoru. Pro další postup, budeme předpokládat, že používáme digitální inteligentní kameru s CMOS snímačem. Volba vhodného barevného prostoru je velmi důležitá, protože se od ní pak odvíjí další zpracování obrazových informací.

1.1.1 Monochromatický barevný prostor

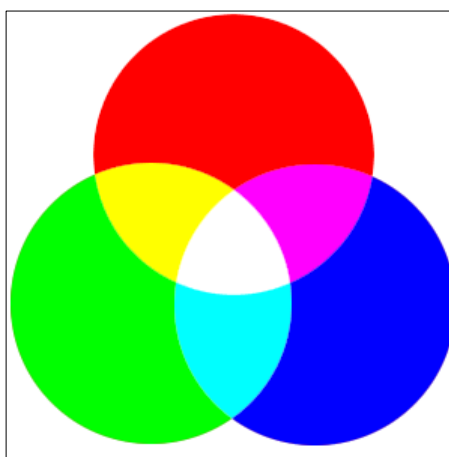
Monochromatický, neboli černobílý, barevný prostor podává informaci pouze o jasové složce obrazu. Barvy se pak zobrazují jako různé stupně šedi od bílé po černou. Tento barevný prostor není pro zpracovávání obrazu v rámci tohoto tématu vhodný a nebudeme se jím dále zabývat.

1.1.2 Prostor RGB

Barevný prostor RGB využívá pro zobrazení různých barev kombinaci tří odlišných složek – R – červené, G – zelené a B – modré. Jednotlivé barvy lze vyjádřit trojicí, či barevným vektorem, jehož složky nabývají hodnot z intervalu $\langle 0,1 \rangle$. Jednotlivé složky mohou být

uváděny také v celočíselném rozsahu od 0 až po 255, což odpovídá kódování každé ze složek RGB v jednom bytu. Hodnota 0 znamená, že složka není zastoupena, maximální hodnota 255 indikuje, že složka nabývá své nejvyšší intenzity. Vyjádření barevných složek pomocí 3 bytů se v současnosti používá nejběžněji [3].

Základní vlastností RGB prostoru je tzv. aditivní (součtové) skládání barev. To znamená, že chceme-li dosáhnout světlejšího odstínu, tím více barev musíme sečíst. Bílá barva je vyjádřena součtem nejvyšších hodnot všech tří složek barevného vektoru – (255, 255, 255), černá pak přesně naopak – (0, 0, 0).



Obr. 1. Aditivní skládání barev [4].

1.1.3 Prostor $YC_B C_R$

Pro barevný prostor $YC_B C_R$ je typické oddělení jasové a barevné složky (luminance a chrominance), kdy složka Y představuje jas a C_B společně s C_R informace o barvě. Podobně jako u RGB modelu nabývají jednotlivé složky určitých hodnot z daného intervalu. Složka Y spadá do intervalu $\langle 0,1 \rangle$, C_B a C_R do intervalu $\langle -1/2, 1/2 \rangle$. Pro zpracování těchto složek v počítači je vhodnější jejich převod do celočíselného rozmezí 0 až 255. Převod mezi prostory RGB a $YC_B C_R$ lze uskutečnit podle následujícího vzorce (1) [3]:

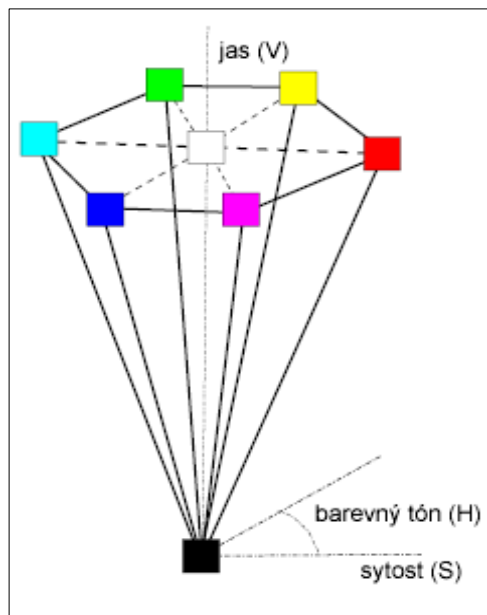
$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 0,299 & 0,587 & 0,114 \\ -0,1687 & -0,3313 & 0,5 \\ 0,5 & -0,4187 & -0,0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (1)$$

Složky C_B a C_R lze definovat zkráceným zápisem (2) [3]:

$$C_B = 0,5643 (B - Y) \quad C_R = 0,7133 (R - Y) \quad (2)$$

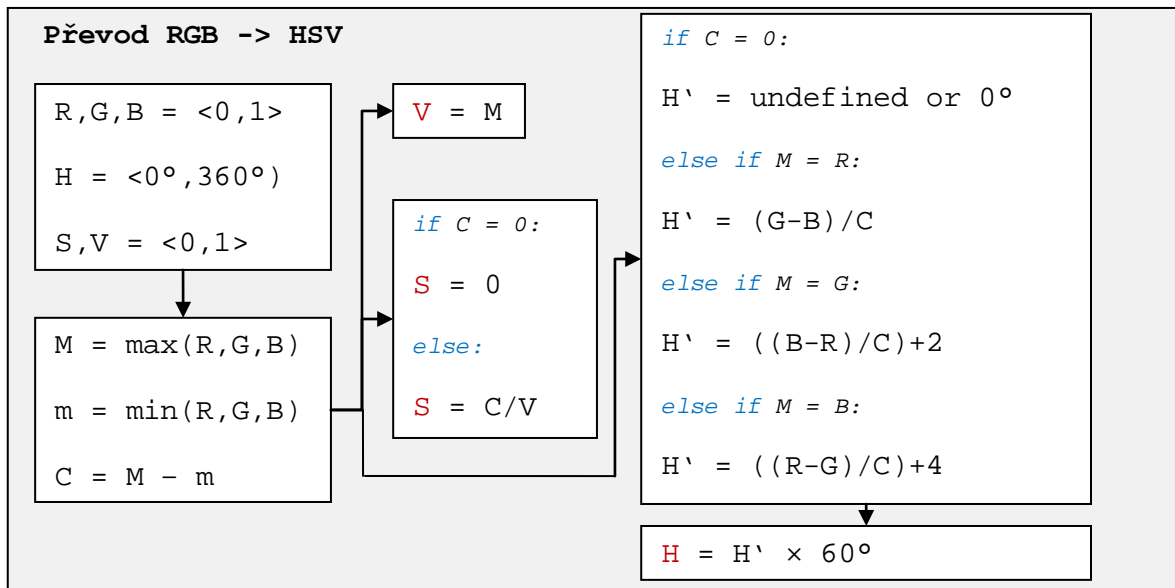
1.1.4 Prostor HSV

Barevný prostor HSV (podobně jako barevný prostor HSL) definuje barvu trojicí složek, které však tentokrát nepředstavují základní barvy. Třemi hlavními parametry prostoru HSV jsou barevný tón (H , hue), sytost (S , saturation) a jasová hodnota (V , value). Barevný tón H označuje převládající spektrální barvu, sytost S určuje příměs jiných barev a jas V je dán množstvím bílého (bezbarvého) světla [3]. Hodnoty jednotlivých složek se nejlépe reprezentují geometricky, na šestibokém jehlanu. Vrchol tohoto jehlanu leží v počátku soustavy souřadnic HSV a představuje černou barvu, při čemž souřadnice (složky) S a V nabývají hodnot z intervalu $\langle 0,1 \rangle$ a H nabývá hodnotu úhlu z intervalu $\langle 0^\circ, 360^\circ \rangle$. Jasová složka V roste směrem ke středu podstavy, který představuje barvu bílou. Jasnější představu přináší obrázek (Obr. 2).



Obr. 2. Znárodnění prostoty HSV [5].

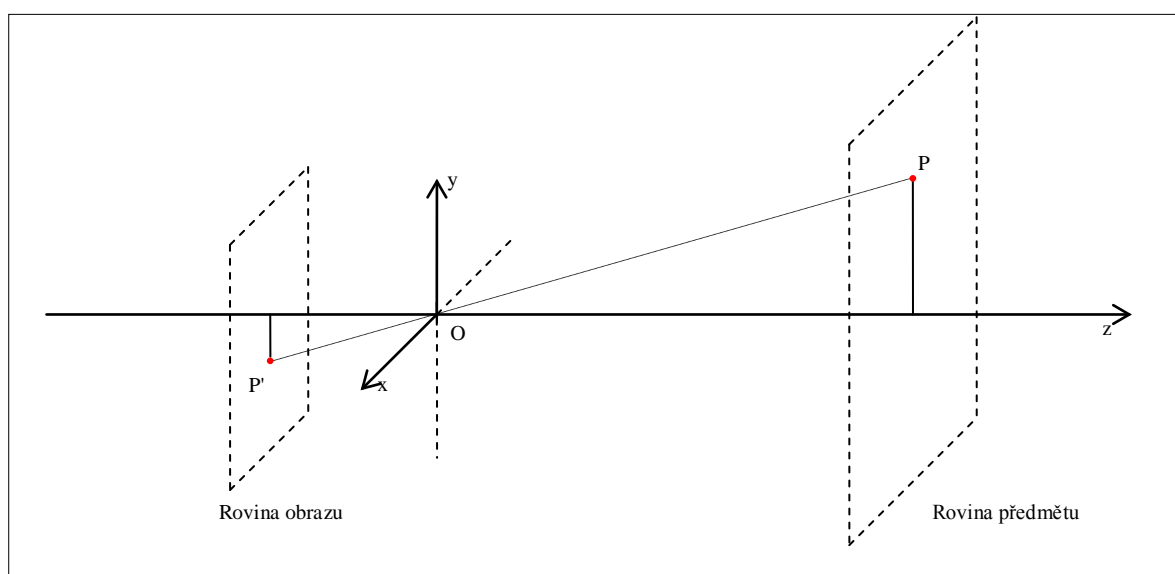
Mezi barevnými prostory RGB a HSV (případně $Y_C R C_B$ a HSV) lze barvy převádět pomocí následujícího algoritmu (Obr. 3) [6].



Obr. 3. Algoritmus převodu barevných prostorů RGB a HSV [6].

1.2 Geometrický model kamery

Pro dané téma můžeme předpokládat vzdálenosti v mnohem větší vzdálenosti, než je ohnisková vzdálenost kamery, stejně jako malý průměr čoček v objektivu, a proto si můžeme dovolit zanedbat zkreslení, ke kterým dochází až už vinou vad čoček či difrakce světla. Finální verze programů předpokládá řešení tohoto zkreslení, ale jeho odstranění je nad rámec této diplomové práce. Geometrický model kamery je tedy možno znázornit jako model dírkové komory – tzv. pinhole kamery (Obr. 4).



Obr. 4. Model dírkové komory.

1.2.1 Perspektivní projekce

Matematický popis modelu dírkové komory (pinhole kamery) se nazývá perspektivní projekcí [7]. Pro perspektivní projekci platí [8], že je-li bod P v předmětové rovině vyjádřen souřadnicemi v prostoru x, y, z a bod P' ležící v obrazové rovině vyjádřen souřadnicemi x', y', z' , při čemž z' je rovno ohniskové vzdálenosti kamery f , lze pro body P, O, P' , které leží na jedné přímce (jsou kolineární), vyjádřit rovnost

$$\overrightarrow{OP'} = \lambda \overrightarrow{OP}, \quad (3)$$

kde λ vyjadřuje konstantu, pro kterou platí

$$\left. \begin{array}{l} x' = \lambda x \\ y' = \lambda y \\ f = \lambda z \end{array} \right\} \Leftrightarrow \lambda = \frac{x'}{x} = \frac{y'}{y} = \frac{f}{z}. \quad (4)$$

Ze vztahu (4) pak vyplývá vztah pro výpočet souřadnic bodu v obrazové rovině dírkové komory ze souřadnic libovolného bodu v prostoru

$$\begin{aligned} x' &= f \frac{x}{z}, \\ y' &= f \frac{y}{z}. \end{aligned} \quad (5)$$

Jelikož může být kamera v prostoru libovolně posunutá a rotovaná, je třeba uvést jednotlivé transformace [8], které je nutné v případě posunu či rotace vykonat před samotnou perspektivní projekcí.

Posunutí (translace) v prostoru lze reprezentovat přičtením translačního vektoru k vektoru souřadnic bodu. Pomocí maticového zápisu se translace vyjadřuje součinem translační matice a vektoru souřadnic bodu v homogenních souřadnicích

$$\begin{bmatrix} 1 & 0 & 0 & v_1 \\ 0 & 1 & 0 & v_2 \\ 0 & 0 & 1 & v_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + v_1 \\ y + v_2 \\ z + v_3 \\ 1 \end{bmatrix}, \quad (6)$$

kde vektor posunutí $\mathbf{v} = (v_1, v_2, v_3)$.

Rotace v prostoru okolo jednotlivých os souřadného systému x, y a z o úhly α, β a γ lze maticově vyjádřit pomocí vztahů

$$Rot(x, \alpha) = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 & 0 \\ \sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (7)$$

$$Rot(y, \beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \beta & 0 & \cos \beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ a} \quad (8)$$

$$Rot(z, \gamma) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \gamma & -\sin \gamma & 0 \\ 0 & \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (9)$$

Chceme-li vyjádřit rotaci obecně kolem všech os najednou o úhly α , β a γ , je třeba jednotlivé matice rotací mezi sebou vynásobit, což vede k výsledné transformační matici

$$T_R = \begin{bmatrix} \cos \alpha \cos \beta & -\cos \gamma \sin \alpha + \cos \alpha \sin \beta \sin \gamma & \cos \alpha \cos \gamma \sin \beta + \sin \alpha \sin \gamma & 0 \\ \cos \beta \sin \alpha & \cos \alpha \cos \gamma + \sin \alpha \sin \beta \sin \gamma & \cos \gamma \sin \alpha \sin \beta - \cos \alpha \sin \gamma & 0 \\ -\sin \beta & \cos \beta \sin \gamma & \cos \beta \cos \gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (10)$$

1.2.2 Vztahy analytické geometrie

V praktické části práce se řada funkcí metod zpracování obrazové informace prolíná s analytickou geometrií. Proto jsou nezbytnou teorií také základní vztahy bodů a přímek [9].

Dva různé body $A = (a_1, a_2)$ a $B = (b_1, b_2)$ v rovině mají vzdálenost

$$|AB| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}. \quad (11)$$

Analytické vyjádření přímky v rovině, její obecná rovnice, má tvar

$$a'x + b'y + c' = 0, \quad (12)$$

chceme-li z ní vyjádřit, pro účely této práce výhodnější, směrnicový tvar, musíme rovnici (12) upravit na vztah

$$\left. \begin{array}{l} a' = a \\ b' = -1 \\ c' = b \end{array} \right\} \Rightarrow y = ax + b. \quad (13)$$

Vzdálenost v bodu $C = (C_x, C_y)$ od přímky p je dána rovnicí

$$v = \frac{|a'C_x + b'C_y + c'|}{\sqrt{a'^2 + b'^2}} \text{ pro obecnou rovnici přímky a} \quad (14)$$

$$v = \frac{|aC_x - C_y + b|}{\sqrt{a^2 + 1}} \text{ pro směrnicový tvar přímky.} \quad (15)$$

1.3 Algoritmy pro vyhledání přímek

Pro určení roviny v prostoru, vůči které je nástroj polohován, je nutno tuto rovinu popsat, pomocí bodů či přímek v této rovině ležících. Existuje řada vhodných algoritmů [10][7][11], které slouží k získání přímek z množiny bodů. V následujících podkapitolách jsou jednotlivé vybrané algoritmy teoreticky popsány.

1.3.1 Split and Merge

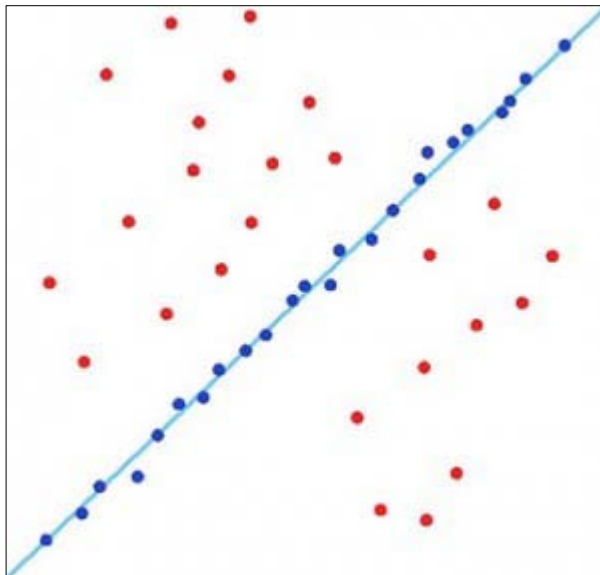
Algoritmus Split and Merge [10] je dle literatury [10] jedním z nejoblíbenějších algoritmů. Jeho principem je dělení a shlukování množin linií a bodů (jak napovídá název). Na počátku je dán seznam S obsahující množiny m_i , kde $i = \{1, 2, 3, \dots, n\}$, shluklých bodů. Mezi množinou m_l a další množinou m_i v S proložíme přímku a hledáme bod B z množiny m_i , který má největší vzdálenost od proložené přímky. Pokud je tato vzdálenost pod určenou mezí, pokračujeme s další množinou v S . Pokud se ale nachází nad mezní hodnotou, rozdělíme množinu m_i na dvě podmnožiny m_{i1} a m_{i2} v bodě B . Po ověření všech množin v S dochází ke sloučení těch množin, které leží na jedné přímce.

1.3.2 Inkrementální algoritmus

Hlavní výhodou tohoto postupu je jednoduchost jeho naprogramování. Postup při vyhledávání přímek je následující. Prvními dvěma body B_1 a B_2 proložíme přímku P_1 . U dalšího bodu B_i ověříme, zda leží na přímce P_1 (popřípadě v její těsné blízkosti). Vyhovuje-li, postupujeme k dalšímu bodu až po B_n . Nevyhovuje-li, sestavíme další přímku P_i a pokračujeme obdobně s dalšími body [10].

1.3.3 RANSAC

Algoritmus RANSAC [10] přináší velkou výhodu v tom, že dokáže „očistit“ výsledek od bodů, které nespádají do hledané množiny. V praxi to znamená, že pokud je znám hledaný tvar, model, je možno pomocí dostatečného počtu opakování náhodného přístupu k datům, tento hledaný tvar nalézt s vysokou přesností a bez nadbytečných dat. RANSAC funguje následovně. Na počátku je dána množina bodů a maximální počet opakování. Náhodně se vyberou dva body z množiny a proloží se jimi přímka. Poté je vypočtena vzdálenost každého bodu od této přímky a vytvoří se množina takových bodů, které na této přímce leží (nebo leží v její těsné blízkosti). Je-li těchto bodů dostatek, daná přímka se uloží a všechny k ní náležející body se odeberou s celku všech bodů a algoritmus se opakuje až po maximální počet opakování, či odebrání většiny bodů.



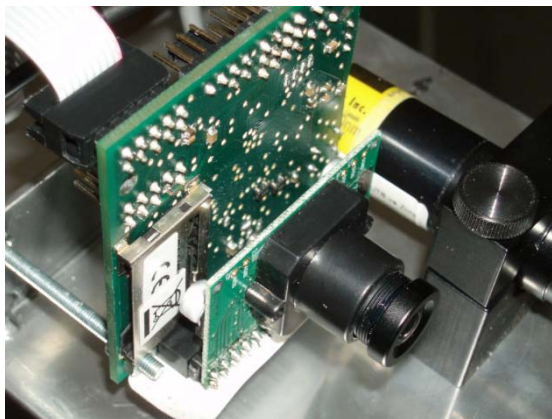
Obr. 5. Výsledek algoritmu RANSAC [12].

1.3.4 Houghova transformace

Houghova transformace [13] se používá především u obrazců, u kterých je známo analytické vyjádření. Nevýhodou tohoto algoritmu se stává jeho výpočetní náročnost. Zjednodušený princip vyhledání přímky spočívá ve vytvoření všech možných variant přímek v každém bodě, který je dán na počátku, a následovné zvolení takové přímky, která je řešením pro co nejvíce bodů. V praxi se algoritmus provádí v diskretizovaném prostoru, protože reálně nelze vytvořit spojitou množinu všech řešení v každém bodě.

2 INTELIGENTNÍ KAMERA CMUCAM3

Pro zpracování obrazové informace a realizaci jednotlivých metod určení polohy nástroje vůči rovině byla vybrána inteligentní digitální kamera CMUcam3 [14]. Její popis a možnosti jsou popsány níže.



Obr. 6. Fotografie kamery CMUcam3.

2.1 Technický popis inteligentní kamery CMUcam3

CMUcam3 je plně programovatelná inteligentní kamera (dále jen kamera). Skládá se z hlavního procesoru Philips LPC2106 [15], postaveném na jádru ARM7TDMI [16] a spojeném s CMOS sensorovým modulem Omnivision OV7620 [17]. Kameru je možno napájet pomocí stejnosměrného adaptéru s pracovním napětím v rozmezí 6 až 15 voltů, při čemž je požadován proud alespoň 150 miliampér. Kamera má pro komunikaci k dispozici sériové rozhraní. Charakteristika hardwaru kamery je uvedena v tabulce (Tab. 1).

Tab. 1. Charakteristika hardwaru inteligentní kamery CMUcam3 [18].

Hardware	Vlastnosti	Hardware	Vlastnosti
CPU		CMOS snímač	
<i>RAM</i>	64 KB	<i>max. rozlišení</i>	352×288 pixelů
<i>ROM</i>	128 KB	<i>barevná hloubka</i>	8 bit na pixel
<i>frekvence</i>	(14 – 60) MHz	Různé	
Paměť FIFO		<i>max. rychlost přenosu</i>	115200 bitů za sekundu
<i>kapacita</i>	1 MB		

2.2 Možnosti kamery CMUcam3

Uvedený seznam obsahuje výčet možností inteligentní kamery CMUcam3, které je možné uplatnit v rámci práce [18]:

- vývojové prostředí pro OS Windows a Linux,
- načítání snímků do paměti s rychlostí až 26 snímků za sekundu,
- softwarová JPEG komprese,
- převzorkování snímků,
- prahovací a konvoluční funkce,
- možnost nastavení barevných prostorů RGB, $Y C_R C_B$ a HSV,
- emulace CMUcam2.

2.2.1 Emulace CMUcam2

Mezi základní možnosti kamery spadá emulace předchozí verze CMUcam2, která je zjednodušenou verzí CMUcam3 pro nenáročného uživatele. Pomocí volně šiřitelného software CMUcam3 Frame Grabber [19] lze pak kameru díky jednoduchým příkazům ovládat, a tak například testovat nastavení jasu, kontrastu či barevného prostoru. Nástroj CMUcam3 Frame Grabber společně s emulátorem CMUcam2 slouží také k pořizování snímků přímo z kamery a to jak manuálně po jednom snímku, tak ve smyčce (snímky mohou být snímány v barevném prostoru RGB a $Y C_R C_B$). Pomocí tohoto nástroje lze také zobrazovat výsledky ze všech uživatelských programů v kameře.

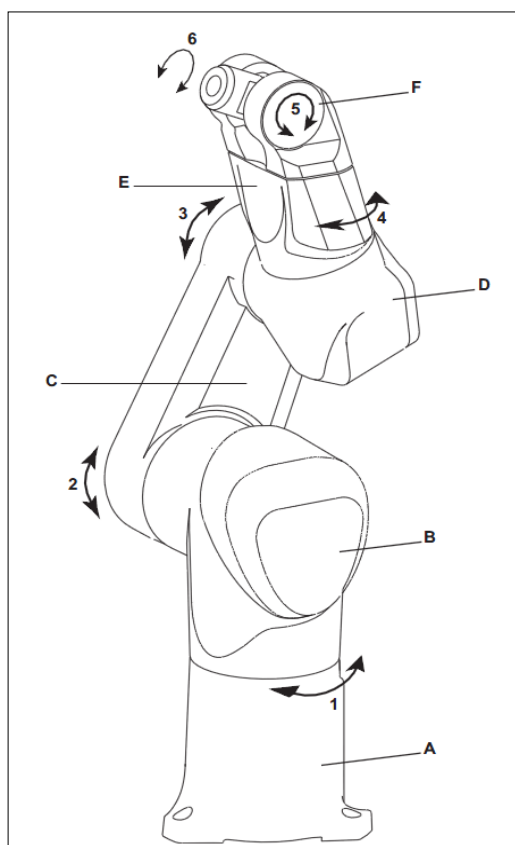
Emulace CMUcam2 ovšem pro náročnější aplikace nestačí, je nutno naprogramovat vlastní programy, přesně podle požadavků aplikace. Pro snadnější práci s kamerou je na webových stránkách výrobce [14] možnost stáhnout již vytvořené knihovny a programy, či procházet fórum, na kterém uživatelé rozebírají případné nedostatky, chyby či možná řešení.

3 PRŮMYSLOVÝ ROBOT STÄUBLI UNIMATION TX40

Jako prostředek, pro polohování nástroje vůči rovině a demonstraci řešení, byl vybrán průmyslový robot Stäubli Unimation TX40 [2]. V následujících podkapitolách je popsána robotická paže ARM TX40 [20], řídicí jednotka CS8C [21] a programovací jazyk VAL3 [22] jímž se robot programuje a ovládá. Rozvedeny jsou také možnosti komunikace a spojení robota s ostatními zařízeními.

3.1 Robotická paže ARM TX40

Pro pohyb v prostoru a mechanickou interakci s okolním světem slouží robotická paže ARM TX40 [20]. Jedná se o kloubovou paži, která umožňuje pohyb až v šesti stupních volnosti. Na Obr. 7 jsou popsány jednotlivé úseky této paže, počínaje základnou (označena písmenem „A“), následuje rameno (ozn. „B“), paže (ozn. „C“), loket (ozn. „D“), předloktí (ozn. „E“) a zápěstí (ozn. „F“). Tyto úseky jsou spojeny klouby (označeny číslicí), ve kterých se nachází servomotory a čítače pro určení absolutní polohy paže v prostoru. Krytí robotické paže splňuje požadavky označení IP65, v části zápěstí IP67. Celková váha paže činí 27 kg.

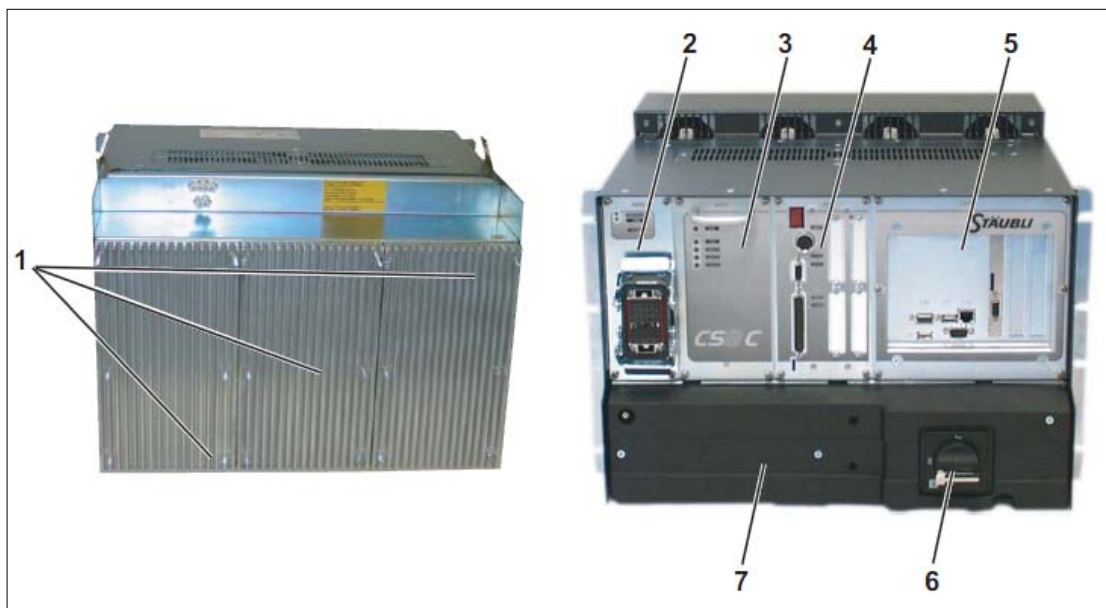


Obr. 7. Robotická paže ARM TX40 [20].

Pracovní prostor robotické paže je v horizontální rovině vyčleněn mezikružím o poloměrech $R_{\text{vnitřní}} = 15,1$ cm a $R_{\text{vnější}} = 45$ cm, se středem ve výšce kloubu „2“. Ve vertikální rovině je $R_{\text{vnitřní}} = 16,2$ cm, $R_{\text{vnější}}$ zůstává stejný. Maximální rychlost pohybu paže v tomto prostoru je 8,2 metru za sekundu při zachování opakované přesnosti na $\pm 0,02$ milimetru. Maximální nosnost robotické paže při nominální rychlosti je 1,7 kilogramu, při redukované rychlosti 2 kilogramy, při zachování určitých omezení (viz [20]), lze nosnost navýšit až na 2,3 kilogramu. K přírubě robotické paže na konci zápěstí lze přichytit libovolný nástroj splňující konstrukční požadavky uvedené v manuálech [20], a tak pracovní prostor ještě rozšířit.

3.2 Řídicí jednotka CS8C

Tato součást průmyslového robota Staubli Unimation TX40 s označením CS8C [21] slouží k provádění veškerých výpočtů a pokynů pro řízení pohybu robotické paže. Skládá se z několika modulů. Na Obr. 8 je řídicí jednotka zobrazena a popsána.



Obr. 8. Řídicí jednotka CS8C [21].

Číslem „1“ jsou označeny výkonové zesilovače řídicí jednotlivé klouby robotické paže. Každý ze tří výkonových zesilovačů řídí jeden pár kloubů. I když jsou mechanicky stejné, jsou nakonfigurovány každý zvlášť, a proto není možná vzájemná výměna. Pod čísly „2“ a „3“ jsou na obrázku označeny napájecí zdroje, mající na starost zásobování jednotlivých součástí řídicí jednotky stejnosměrným napětím, které je ze střídavého síťového napětí transformováno v modulu „7“, který rovněž obsahuje pojistky. Napájení robotické paže je

zajištěno RSI deskou (ozn. „4“), která robotickou paži také chrání před poškozením při poruše v elektrické síti. K RSI desce je rovněž připojen ovládací panel robota a přepínač pracovního módu. Procesní jednotka řídicí jednotky (ozn. „5“) spravuje aplikace a řídí vstupy a výstupy. Její součástí jsou ethernetové, USB a sériové vstupy a výstupy, základní deska, procesor a paměť pro VAL3 aplikace a programy. Je možné připojit společností Stäubli podporované PCI desky. Pod touto jednotkou se nachází hlavní vypínač řídicí jednotky (ozn. „6“).

3.2.1 Ovládací panel MCP

Pomocí ovládacího panelu [21] připojeného k řídicí jednotce CS8C lze robota Stäubli Unimation TX40 uvádět do provozu, přímo ovládat, programovat a konfigurovat. Nalézá se zde rovněž několik pojistek, aby byla vždy dodržena bezpečnost práce s robotem. Ovládací panel je vybaven grafickým monochromatickým LCD displejem a klávesnicí pro zadávání vstupních informací s LED diodami pro znázornění aktivních kláves.



Obr. 9. Ovládací panel MCP.

3.2.2 Přepínač pracovního módu

Posledním připojeným zařízením k řídicí jednotce je přepínač pracovního módu. Slouží pro přepínání tří pracovních režimů – manuálního, místního a vzdáleného a je možné pomocí nouzového tlačítka zastavit činnost robota. V manuálním režimu lze robota ovládat pomocí ovládacího panelu, tento režim slouží pro učení robota a ladění aplikací. Maximální rychlost pohybu robotické paže je 25 centimetrů za sekundu a aplikace může spustit pohyb pouze při stisku tlačítka Move/Hold na ovládacím panelu. Místní režim slouží ke spouštění aplikací, které v něm mohou zcela řídit pohyby robotické paže. V pracovním prostoru

robota se v tomto režimu nesmí nacházet žádné osoby, protože robot se již pohybuje nominální rychlostí, která může být rovná maximální. Rychlost pohybu a spouštění aplikací se řídí ovládacím panelem. Vzdálený režim se od místního liší pouze možností spustit aplikace ze vzdáleného stanoviště, propojeného s robotem.



Obr. 10. Přepínač pracovního módu [21].

3.3 Programovací jazyk VAL3

Jazyk VAL3 [22] je vysokoúrovňový programovací jazyk vyvinutý společností Stäubli pro vytváření aplikací pro řízení robotů. Tento programovací jazyk kombinuje základní vlastnosti standardních vysokoúrovňových programovacích jazyků (syntaxe, způsob práce) s funkcemi, které jsou specifické pro ovládání průmyslových robotů. Pro účely ovládání robotů obsahuje VAL3 nástroje pro přímé řízení robotů, pro geometrické modelování a také nástroje pro řízení vstupů a výstupů.

VAL3 se sestává z několika základních prvků. Jsou to:

- aplikace,
- programy,
- knihovny,
- datové typy,
- konstanty,
- proměnné (globální, lokální a parametry),
- úlohy.

Aplikace se skládají z programů, které obsahují instrukce k vykonání, globálních proměnných dat a knihoven. Když je aplikace spuštěna, obsahuje také sadu úloh. Výchozím obsahem každé aplikace jsou programy **start()** a **stop()**, které jsou vždy volány při spuštění a ukončení aplikace a nemohou mít žádné parametry, a globální proměnné **world** (typu *frame*) a **flange** (typu *tool*). Mimo základní datové typy jako jsou číselný *num*,

písmenný *string*, logický *bool* a vstupní *dio*, *sio*, *aio*, obsahuje programovací jazyk VAL3 ještě strukturované datové typy *trsf* (transformace v kartézském souřadnicovém systému), *frame* (kartézský souřadnicový systém (dále jen kart. s. s.)), *tool* (nástroj upevněný na přírubě paže), *point* (bod v kart. s. s.), *joint* (pozice jednotlivých kloubů), *config* (konfigurace robotické paže) a datový typ *mdesc* (parametr rychlosti pohybu robotické paže).

Programování ve VAL3 programovacím jazyku se provádí buď přímo v řídicí jednotce přes ovládací panel, či přes s řídicí jednotkou spojený počítač se softwarem VAL3 Studio a hardwarovým klíčem. VAL3 Studio oproti programování ovládacím panelem nabízí standardní funkce vývojového prostředí jako je ladění aplikací či editovací funkce a nápovědu.

```

28  toNum(mid(vstupstr,1,0),pod_x_pls,prev)
29  toNum(mid(vstupstr,1,1),pod_x_min,prev)
30  toNum(mid(vstupstr,1,2),pod_y_pls,prev)
31  toNum(mid(vstupstr,1,3),pod_y_min,prev)
32  if pod_x_pls==1
33      x_rot=x_rot-rot_hod_x
34  endif
35  if pod_x_min==1
36      x_rot=x_rot+rot_hod_x
37  endif
38  if pod_y_pls==1
39      y_rot=y_rot-rot_hod
40  endif
41  if pod_y_min==1
42      y_rot=y_rot+rot_hod
43  endif
44  put("data ")
45  putln(vstupstr)
46  put("x rotace ")
47  putln(x_rot)
48  put("y rotace ")

```

Obr. 11. Ukázka programu ve VAL3 Studiu.

4 VYUŽITÍ SERVISNÍCH ROBOTŮ V BEZPEČNOSTNÍM PRŮMYSLU

Řešení problému polohování nástroje vůči rovině ve více stupních volnosti lze aplikovat také do bezpečnostního průmyslu. Robotika v oblasti průmyslu komerční bezpečnosti je v rozkvětu, a v blízké budoucnosti se předpokládá, že servisní roboti budou vykonávat čím dál tím více úkolů.

4.1 Současné využití servisních robotů v bezpečnostním průmyslu

Celkového odhadované množství profesionálně využívaných servisních robotů v nevýrobních oblastech použití v roce 2008 bylo 63000, z čehož 32% bylo využito v oblasti obrany a bezpečnosti [27]. Jejich nasazení se předpokládá především tam, kde je pro příliš nebezpečné okolní prostředí popř. tam, kde se člověk může jen obtížně dostat. Jejich výzkumem se zabývají odborníci především v Japonsku a Německu a do roku 2012 se předpokládá nárůst profesionálně používaných servisních robotů až o 49000 kusů [27]. Zdokonalování vlastností servisních robotů se týká především oblasti jejich sensoriky a zvýšení samostatnosti, inspirací pro vylepšení jsou především robotické systémy využívané v kosmickém průmyslu. V oblasti ochrany, záchranných prací a bezpečnosti je zájem o servisní roboty nejvyšší.

Nasazování servisních robotů v oblasti bezpečnosti probíhá v rámci vyprošťovacích prací, odstraňování výbušnin, hlídání objektů a také v bojových situacích. Americká armáda testovala nasazení bojových robotů již v roce 2000, v roce 2007 byly nasazeny tři robotické jednotky typu SWORDS vybavené automatickými zbraněmi v Iráku [28]. Známé jsou také bezpilotní letouny sloužící k průzkumu a bombardování.

Jedním z prvních použití servisních robotů v průmyslu komerční bezpečnosti bylo nasazení 11 robotických jednotek ke střežení parkoviště a stanového městečka na mistrovství světa v kopané v Berlíně roku 2006. Tyto robotické jednotky byly vybaveny především kamerami a termovizí [29].

Mezi vybavení Armády ČR patří také robot typu tEODor [30], používaný například 15. ženijní brigádou, sloužící k průzkumu a zneškodňování výbušných látek a výbušných zařízení. Ve výzbroji Armády ČR je tento robot již od roku 2005, současně je tento typ robota také vybavením pyrotechnické jednotky Policie ČR a v nedávné době byl nasazen například při dopravní nehodě na dálnici D5 2. května 2011 [31].



Obr. 12. Vojenský robot typu SWORDS [28].

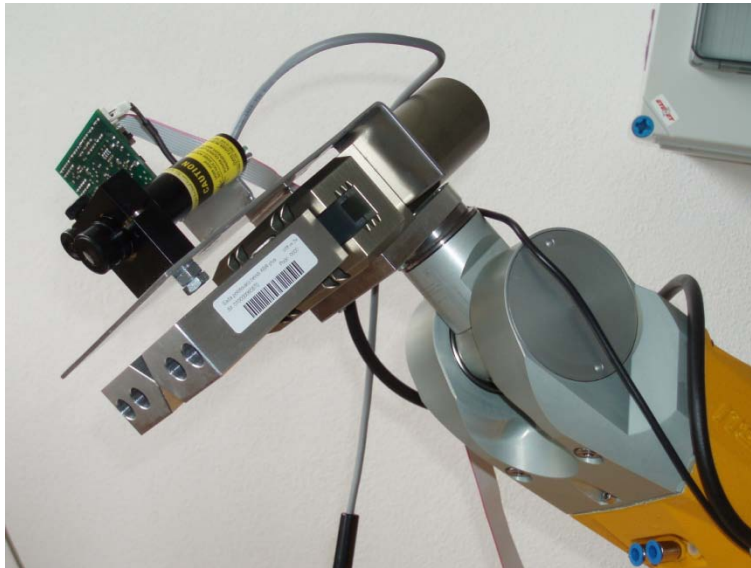


Obr. 13. Servisní robot typu tEODor Armády ČR [30].

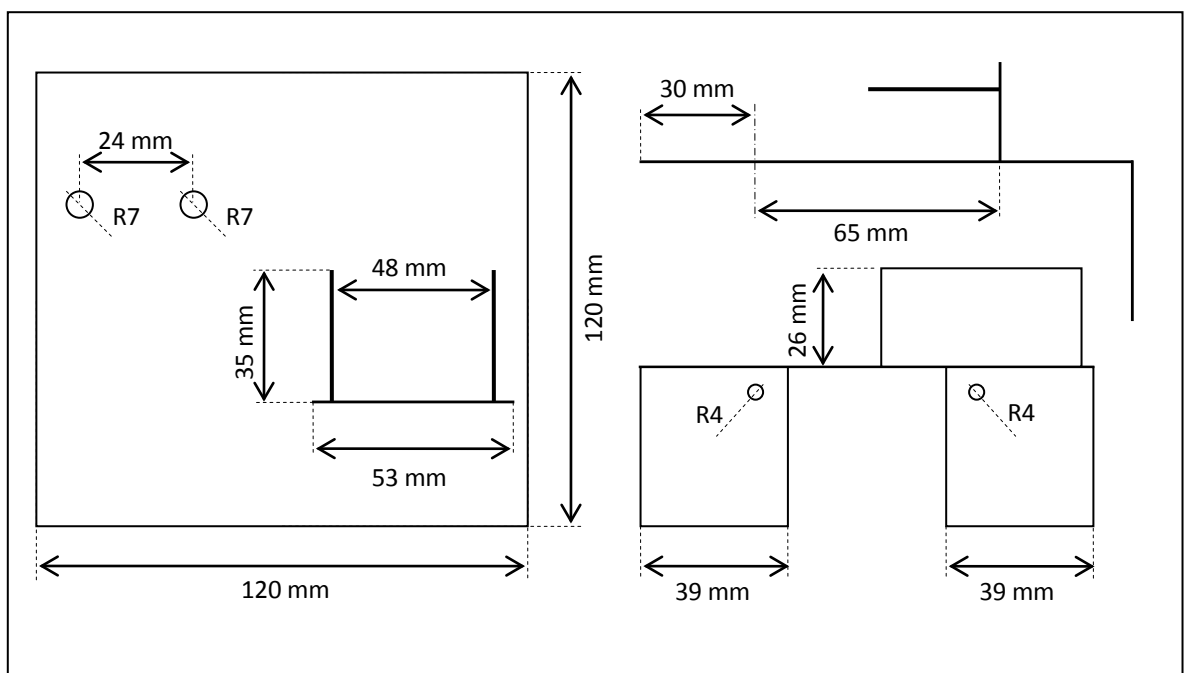
II. PRAKTICKÁ ČÁST

5 NÁSTROJ ROBOTICKÉ PAŽE

Prvním úkolem a částí implementace řešení na průmyslového robota Stäubli Unimation TX40 bylo sestavení vhodného nástroje umístitelného na přírubu robotické paže. Bylo využito již namontovaného nástroje s čelistmi, na který byl vytvořen železný nástavec s otvory pro přišroubování laseru a inteligentní kamery. Fotografie tohoto nástroje se nachází na Obr. 14, jeho konstrukční výkres pak na Obr. 15.



Obr. 14. Fotografie nástroje robotické paže.



Obr. 15. Zjednodušený konstrukční výkres nástavce nástroje.

Střed laseru je na nástroji umístěn co nejbližší středu kamery, aby bylo dosaženo přesných výsledků i bez nutnosti použití transformace posunu.

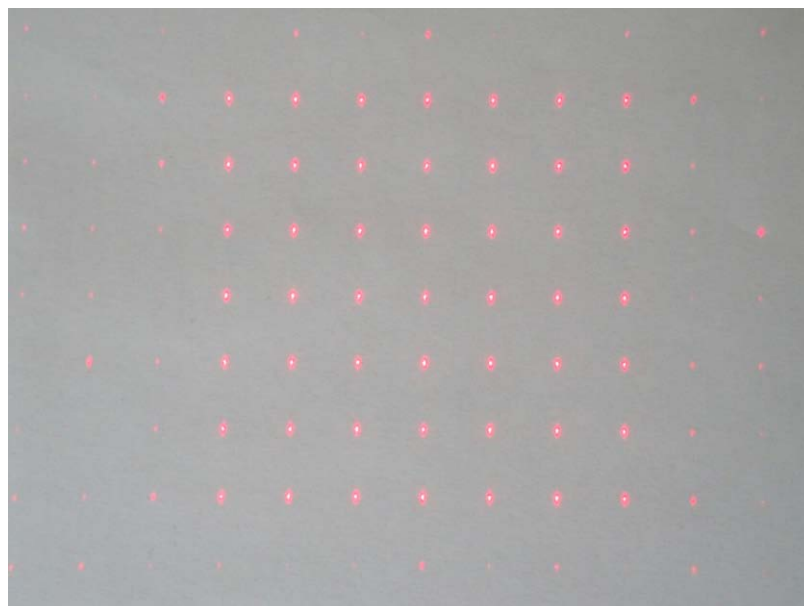
5.1 Lasiris SNF Laser

Jako zdroj koherentního světla byl zvolen Lasiris SNF Laser [23] od společnosti StockerYale. Specifikace tohoto laseru jsou uvedeny v tabulce (Tab. 2).

Tab. 2. Specifikace Lasiris SNF Laseru [23].

Veličina	Hodnota
Váha	65 gramů
Výkon	20 mW
Vlnová délka	660 nm
Provozní napětí	(4,8 – 6,5) voltů

Laser je možno osadit optikou, která vytváří strukturované světelné obrazce. Byla vybrána optika s označením 507X tvořící strukturovaný světelný obrazec ve formě matice 7×7 bodů. Úhel mezi jednotlivými paprsky, které tvoří tento obrazec, závisí na vlnové délce laserové diody [23]. Pro vlnovou délku 660 nm svírají paprsky mezi sebou úhel $1,87^\circ$.

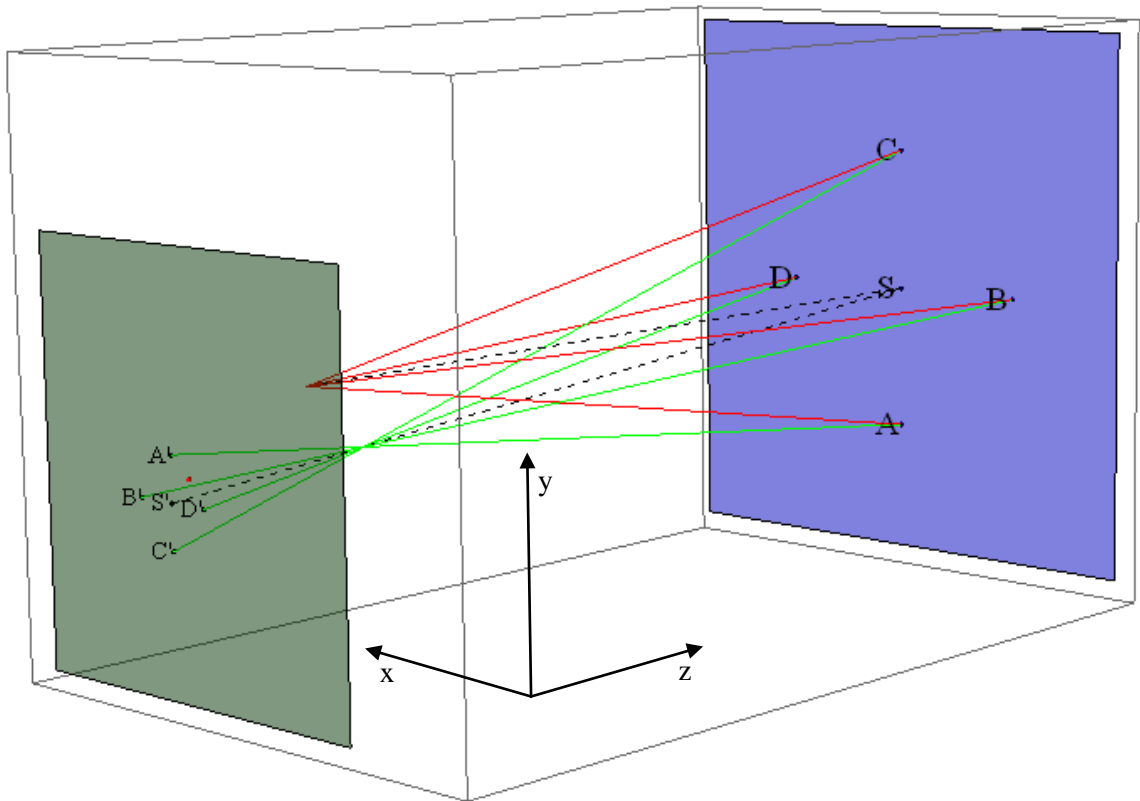


Obr. 16. Fotografie světelného obrazce ve formě matice bodů.

Upevnění laseru na nástroji řeší nastavitelná objímka, která dovoluje přesné nasměrování paprsku v prostoru. Tato objímka je umístěna na nastavci nástroje (viz Obr. 15).

6 MATEMATICKÝ MODEL

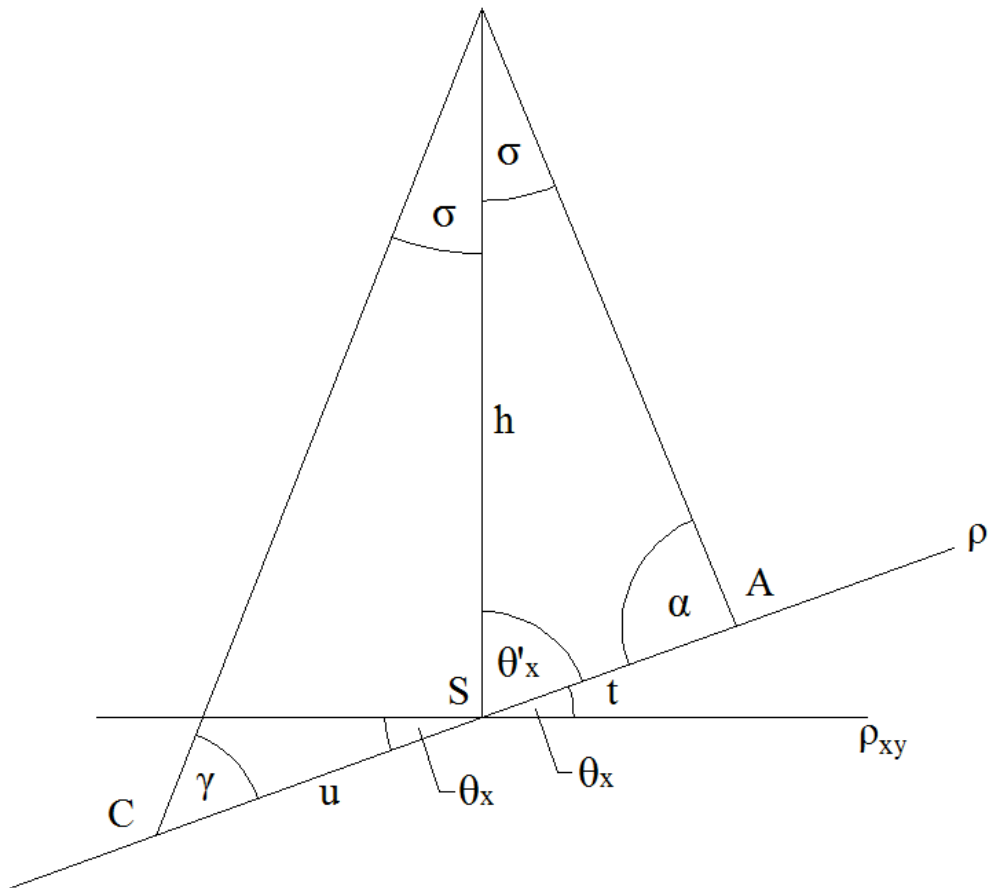
Pro účely polohování byl odvozen matematický model celé situace. V prostředí Wolfram Mathematica [24] byla situace vymodelována pomocí 3D znázornění uvedeném na Obr. 17.



Obr. 17. 3D znázornění matematického modelu.

Ve strukturovaném světelném obrazi tvořeném laserem lze nalézt právě čtyři body, shodné s body A, B, C a D na Obr. 17. Tyto body tvoří vrcholy čtyřúhelníku, jenž se promítá do obrazové roviny inteligentní kamery. Promítání má charakter perspektivní projekce popsané vztahy (5), (6) a (10). Původní čtyřúhelník ABCD tak lze získat ze čtyřúhelníku A'B'C'D', který je zpracováván inteligentní kamerou.

Na obrázku (Obr. 18) je znázorněna výchozí situace, ze které jsou odvozeny vztahy pro výpočet úhlu rotace roviny ρ okolo osy x, ze známých údajů. Tento úhel pak svírá rovina ρ s rovinou ρ_{xy} tvořenou osami x a y.



Obr. 18. Závislost rotace roviny obrazce a poměru úseček.

Jelikož známe polohu bodů A, C, S a úhel σ (úhel σ svírají krajní paprsky tvořící světelný obrazec se středovým paprskem laseru – viz kapitola 5.1), můžeme odvodit následující vztahy:

$$\frac{t}{\sin \sigma} = \frac{h}{\sin \alpha} \Rightarrow t = \frac{h \sin \sigma}{\sin \alpha}, \text{ obdobně pak} \quad (16)$$

$$u = \frac{h \sin \sigma}{\sin \gamma}. \quad (17)$$

Pro úhly α a γ (viz Obr. 18) platí rovnosti

$$\alpha = 90^\circ - \sigma + \theta_x \text{ a } \gamma = 90^\circ - \sigma - \theta_x. \quad (18)$$

Dosazením (18) do vztahů (16) a (17) a provedením základní goniometrické úpravy získáme vztahy

$$t = \frac{h \sin \sigma}{\cos(\theta_x - \sigma)} \text{ a } u = \frac{h \sin \sigma}{\cos(\theta_x + \sigma)}. \quad (19)$$

Pro poměr délek t a u , což jsou velikosti úseček AS a CS, vychází výsledná rovnost

$$\frac{t}{u} = \frac{\cos(\theta_x + \sigma)}{\cos(\theta_x - \sigma)}. \quad (20)$$

Jediná neznámá v této rovnici je úhel θ_x , který je možno vyjádřit rovnicí

$$\theta_x = \arccos\left(\frac{(t+u)\sin\sigma}{\sqrt{t^2 + u^2 - 2tu\cos 2\sigma}}\right), \quad (21)$$

pokud je úhel $0^\circ \leq \sigma < 90^\circ$ a $t \leq u$.

Díky určení úhlu rotace roviny okolo osy x , lze vypočítat vzdálenost středu čtyřúhelníku od zdroje světelných paprsků, tedy laseru,

$$h = \frac{t \cos(\theta_x - \sigma)}{\sin \sigma}. \quad (22)$$

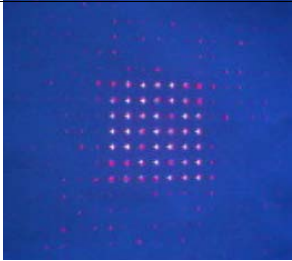
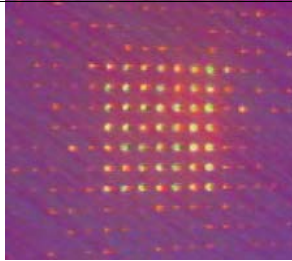
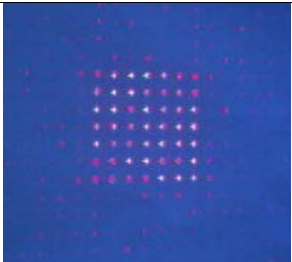
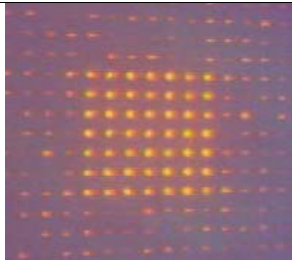

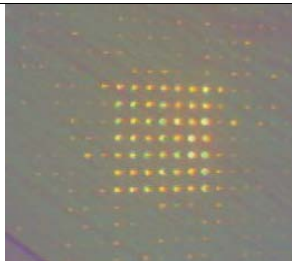
7 ZPRACOVÁNÍ OBRAZOVÉ INFORMACE

Tato kapitola rozvádí zpracování obrazové informace inteligentní kamerou CMUcam3, respektive vyhledání strukturovaného světelného obrazce, jeho převedení na do vhodných souřadnic a určení čtyřúhelníku, podle kterého je možno polohovat nástroj robotické paže (viz kapitola 6). Řešení tohoto zpracování informací a vyvíjení vlastních metod a algoritmů bylo nejobsáhlejší a nejrozsáhlejší částí praktické části práce jak časově, tak obtížnostně a výsledky tohoto řešení již byly prezentovány v rámci mezinárodní soutěže STOČ 2011 v Ostravě [25].

7.1 Nastavení inteligentní kamery

Aby byla zaručena správná funkčnost následujících metod a algoritmů, bylo nutné nastavit inteligentní kameru CMUcam3 (dále jen kameru) do vhodného barevného režimu a nastavit jas, kontrast, automatické vyvážení bílé, automatickou expozici a rozlišení. Testování různých nastavení probíhalo pomocí emulátoru CMUcam2, spuštěného v kameře, a nástroje CMUcam3 Frame Grabber, jehož pomocí byly sejmuty testovací snímky (viz Tab. 3).

Tab. 3. Testovací snímky pořízené pomocí nástroje CMUcam3 Frame Grabber.

Jas: 0 Kontrast: 0 Prostor: RGB		Jas: 0 Kontrast: 0 Prostor: $Y C_R C_B$	
Jas: 50 Kontrast: 50 Prostor: RGB		Jas: 100 Kontrast: 100 Prostor: $Y C_R C_B$	
Jas: 255 Kontrast: 100 Prostor: RGB		Jas: 255 Kontrast: 100 Prostor: $Y C_R C_B$	

Z testování nakonec vyplynulo, že kamera nedokáže správně reprezentovat příliš jasné červené světlo produkované laserovou diodou jak v RGB barevném prostoru, tak v $Y C_R C_B$ barevném prostoru, a proto byl zvolen barevný prostor HSV, respektive jeho jasová složka V, pro další zpracování.

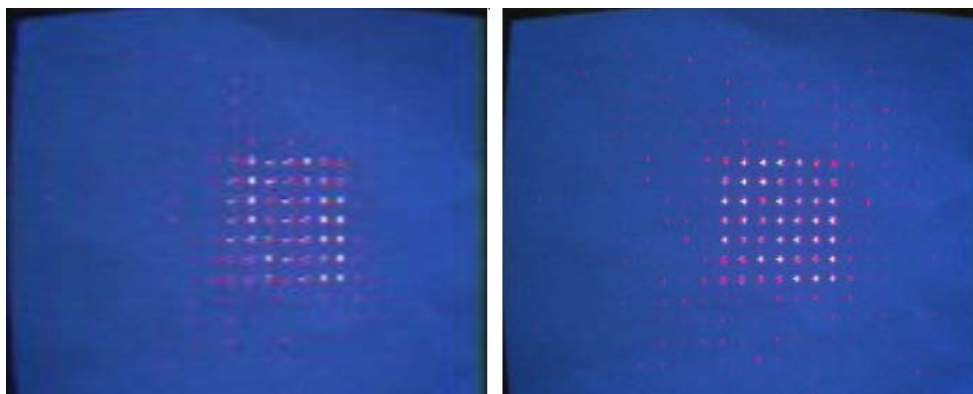
Po zvolení barevného prostoru bylo nutné zvolit, zda si má kamera sama řídit nastavení jasu a kontrastu pomocí automatického vyvážení bílé a automatické expozice. Při testování kamery v různých světelných podmínkách se automatická nastavení prokázala jako nevhodná, jas je proto snížen vždy na minimální hodnotu 0 (z rozsahu 0 až 255) a kontrast naopak na hodnotu maximální, tedy 255, a automatické vyvážení bílé, spolu s automatickou expozicí, bylo vypnuto.

Příkazy sloužící k nastavení v emulátoru jsou uvedeny v tabulce (Tab. 4).

Tab. 4. Příkazy pro nastavení emulátoru CMUcam2 [26].

Příkaz	Efekt
CR 5 (0-255)	Nastavení hodnoty kontrastu
CR 6 (0-255)	Nastavení hodnoty jasu
CR 18 36	Barevný prostor $Y C_R C_B$, automatické vyvážení bílé zapnuto
CR 18 32	Barevný prostor $Y C_R C_B$, automatické vyvážení bílé vypnuto
CR 18 44	Barevný prostor RGB, automatické vyvážení bílé zapnuto
CR 18 40	Barevný prostor RGB, automatické vyvážení bílé vypnuto
CR 19 32	Automatická expozice vypnuta
CR 19 33	Automatická expozice zapnuta

Rozlišení kamery lze zvolit ze dvou variant – nízké (255×160 pixelů) a vysoké (352×288 pixelů). Na následujícím obrázku (Obr. 19) jsou dva stejné snímky v odlišných rozlišeních. Nižší rozlišení poskytuje vyšší rychlost, ale počet chyb (především ztráta některých z bodů) se zvyšuje, proto bylo zvoleno rozlišení vysoké.



Obr. 19. Totožné snímky s rozdílným rozlišením.

Ve vlastních programech, běžících v kameře, je nutné kameru nastavit pomocí funkcí z knihovny `cc3.h`. Jasnou představu o způsobu nastavení přináší následující ukázka kódu:

```
cc3_camera_set_colorspace (CC3_COLORSPACE_HSV); //HSV prostor
//cc3_camera_set_colorspace (CC3_COLORSPACE_RGB); //RGB prostor

cc3_camera_set_resolution (CC3_CAMERA_RESOLUTION_HIGH);
cc3_camera_set_auto_white_balance (false);
cc3_camera_set_auto_exposure (false);
cc3_camera_set_brightness (0);
cc3_camera_set_contrast (255);
```

7.2 Tvorba základních struktur

Základem všech programů, které byly napsány, bylo definování vhodných struktur, které jsou pro tyto programy společné, či se jen nepatrně liší. Jedná se o struktury `bod`, `primka` a `vysledek`. Tyto struktury jsou naprogramovány v kameře následovně:

```
typedef struct
{
uint16_t xx;
uint16_t yy;
} bod;

typedef struct
{
float a;
float b;
bod vlastni[50];
int posl;
float NAD;
float POD;
} primka;

typedef struct
{
primka vys1;
primka vys2;
primka vys3;
primka vys4;
int navrat;
} vysledek;
```

Struktura `bod` se skládá ze dvou souřadnic celočíselných x a y . Jedná se o základní strukturu bodu, který leží v rovině. Struktura `primka` vyjadřuje přímku zadanou rovnicí ve směrníkovém tvaru $y = ax + b$ (viz vzorce (12) a (13)), proměnné a a b jsou první částí struktury, následuje statické pole bodů ležících na této přímce, proměnná `posl` uchovává

informaci o posledním volném indexu v poli vlastních bodů, reálná čísla NAD a POD vyjadřují procentuální zastoupení všech bodů nad přímkou a pod přímkou. Poslední struktura `vysledek` obsahuje čtyři výsledné přímky a proměnnou `navrat`, která poskytuje informaci o stavu naplnění těchto výsledných přímek. Ne v každém programu jsou struktury úplné, to znamená, že když není určitá část struktur nutná pro práci metod a algoritmů, je vynechána, aby bylo k dispozici větší množství dostupné paměti kamery.

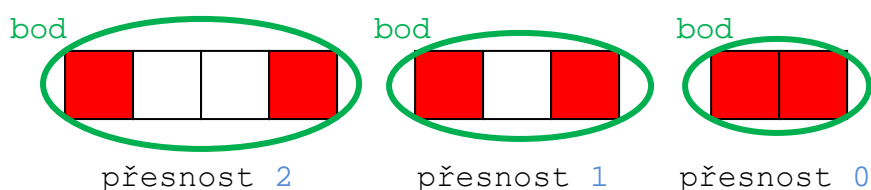
7.3 Vyhledání bodů

Vyhledávání bodů strukturovaného světelného obrazce je založeno na nalezení bodů s hodnotou jasové složky vyšší, než je hraniční hodnota. Hledání bodů probíhá postupným průchodem snímku pixel po pixelu, řádek po řádku. Mezi již vytvořenými programy pro inteligentní kameru CMUcam3, se nalézá jednoduchý program `simple-track-color.c` [14] pro sledování barvy ve snímku, resp. její vyhledání. Je v něm znázorněno, jak procházet snímky a porovnávat hodnoty jejich jednotlivých složek. Na tomto algoritmu procházení byly pak založeny algoritmy pro vyhledávání bodů, ty jsou však kompletně přepracovány a po všech vlastních úpravách jsou od jednoduchého programu velmi odlišné.

Pro urychlení vyhledávání jsou vyhodnocovány pouze jasové složky jednotlivých pixelů – jejich složka *V*, pohybujeme-li se v barevném prostoru HSV.

Aby bylo dosaženo nalezení všech potřebných bodů světelného obrazce (tedy matice 7×7 bodů) za všech světelných podmínek, mění se hraniční hodnota jasu dynamicky podle toho, kolik bylo vhodných bodů nalezeno, a to v obou směrech.

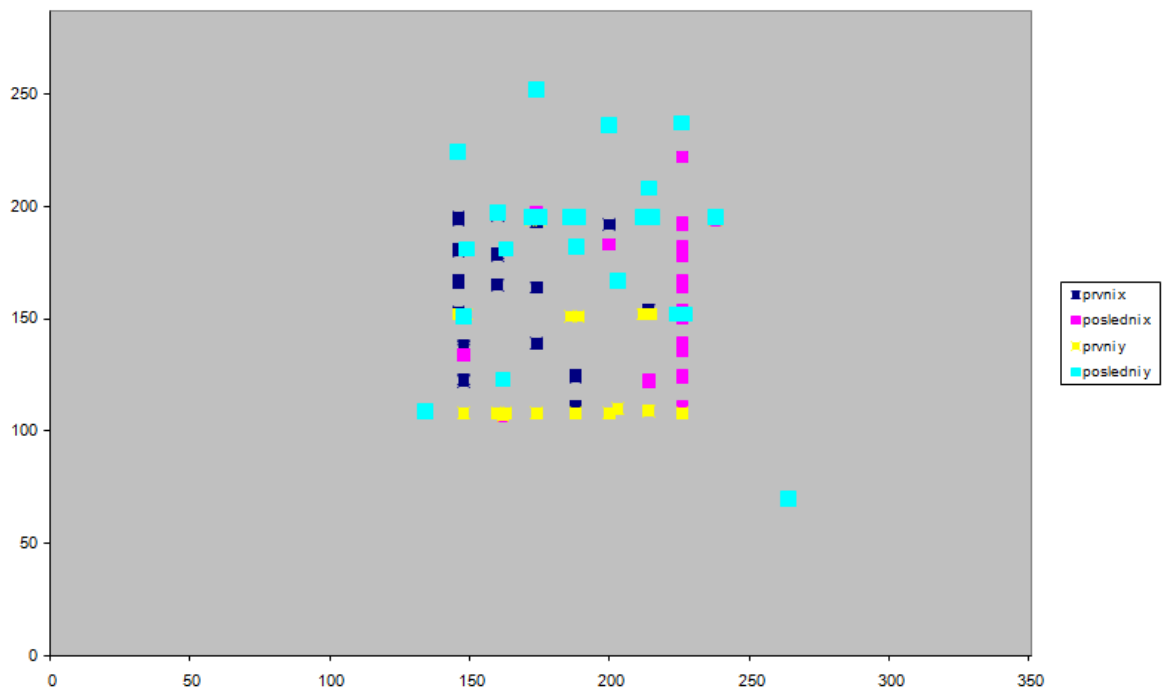
Pro následné sestrojení přímek byly vyvinuty dvě následující metody vyhledání bodů. Obě metody mají společný základ ve vyhledání bodů s určenou přesností. Přesností je myšlena vzájemná vzdálenost pixelů jednoho bodu v řádku. Pokud je nalezen určitý počet pixelů vedle sebe, je do množiny bodů uložen tzv. potenciální bod, který je dále vyhodnocen.



Obr. 20. Znázornění přesnosti vyhledávání bodů.

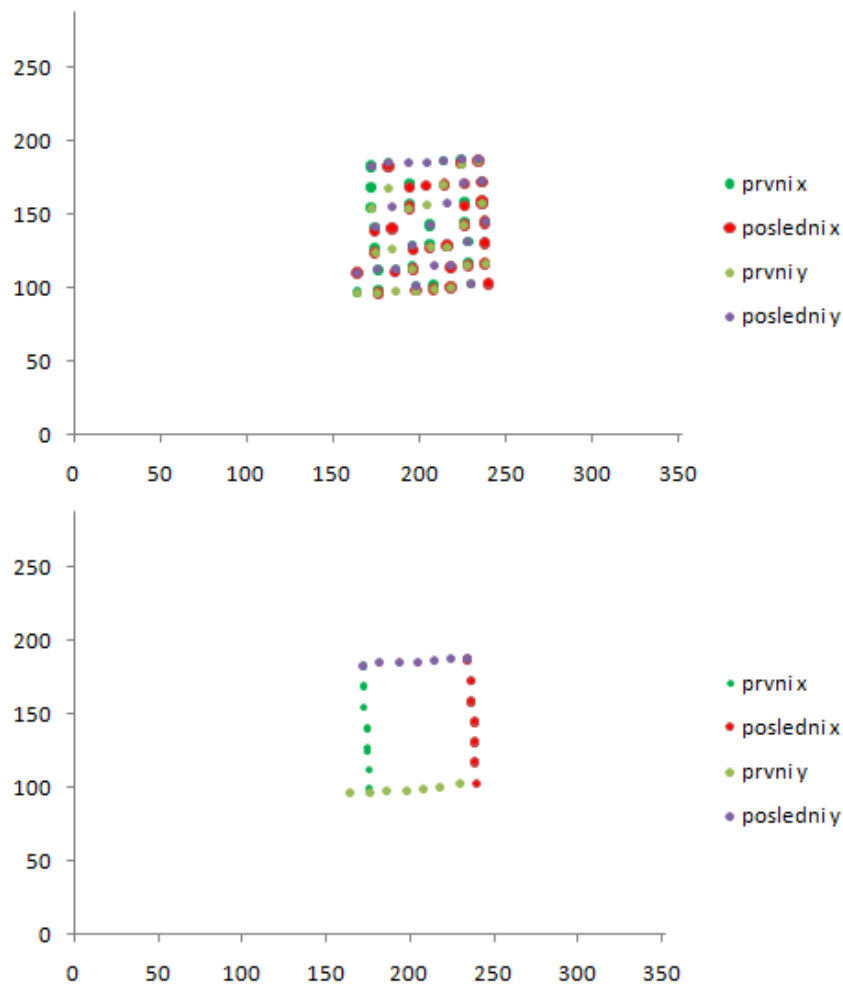
7.3.1 Metoda nalezení okrajových bodů

Tato metoda spočívá v nalezení pouze okrajových bodů světelného obrazce, tedy sedmi bodů z každého směru. Jsou vytvořeny čtyři skupiny bodů, které se utvářejí s množiny potenciálních bodů. Skupiny bodů jsou rozděleny jako první body ve směru osy x, poslední body ve směru osy x, první body ve směru osy y a poslední body ve směru osy y. Pokud není v metodě použito očištění od tzv. falešných bodů, tedy bodů, které nejsou na okraji obrazce, ale jsou do skupin bodů přiřazeny, je tato metoda nepřesná, ale při použití správných algoritmů sestavení přímek použitelná a velmi rychlá (nejlepší výsledky doby trvání algoritmů založených na této metodě dosahovaly hodnot okolo 250 milisekund). Výsledné nalezené body lze zobrazit například pomocí grafů v programu Microsoft Excel, ve kterém byly tvořeny také analýzy výsledných dat, která se podařilo z kamery získat.



Obr. 21. Skupiny okrajových bodů bez očištění od falešných bodů.

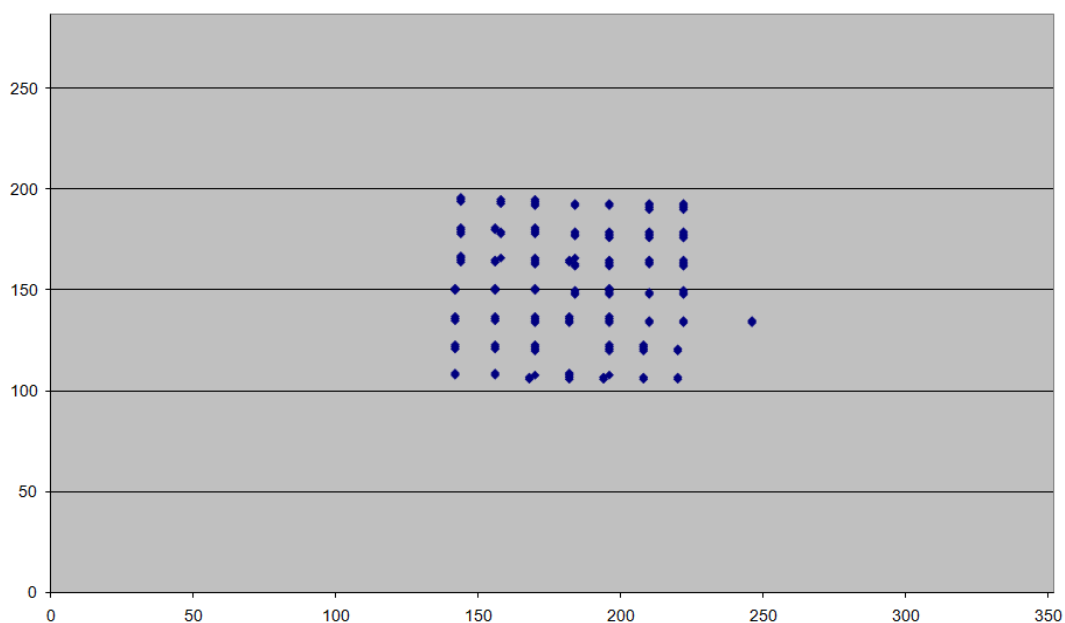
Zpřesnění výsledků této metody bylo dosaženo pomocí čištění výsledných skupin bodů od falešných bodů na základě četnosti výskytu souřadnic x, či y. Srovnání výsledků před a po očištění je uvedeno na Obr. 22.



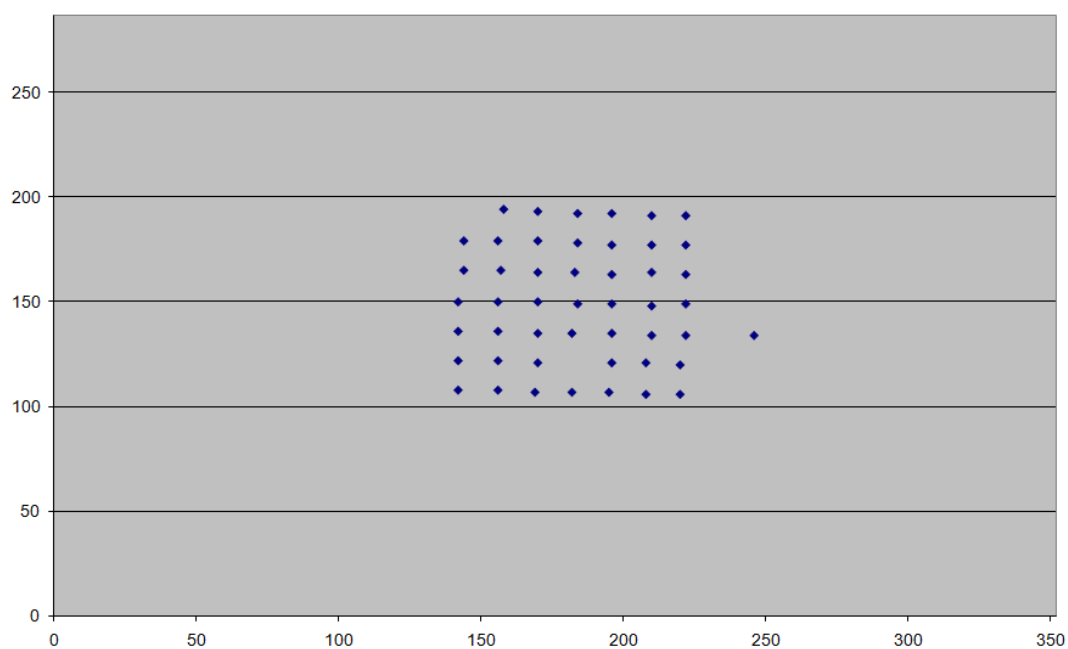
Obr. 22. Srovnání výsledků před a po očištění podle četnosti.

7.3.2 Metoda nalezení všech bodů

Princip nalezení všech bodů je podobný jako nalezení okrajových bodů, ale tentokrát existuje pouze jedna výsledná skupina bodů, která obsahuje všechny vyhledané body, kterých by v ideálním případě mělo být 49 (matice 7×7). U předchozí metody se velmi blízké body neslučovaly v jeden, jak je tomu ve skutečnosti, ale byly brány jako několik bodů vedle sebe. Tato nevýhoda první metody je způsobena samotnou metodou, kdy se ukládaly pouze první a poslední body a nemohlo tak dojít ke správnému sloučení. Jelikož u druhé metody dochází k uložení všech bodů ve všech směrech, je možné provést očištění množiny od falešných bodů pomocí sloučení blízkých bodů, při čemž je stanovena podmínka počtu blízkých bodů tak, aby byly vyloučeny osamocené falešné body.



Obr. 23. Všechny nalezené body před sloučením.



Obr. 24. Všechny nalezené body po sloučení.

Na výše uvedených obrázcích Obr. 23 a Obr. 24 lze pozorovat zredukování výsledné množiny na konečnou matici 7×7 bodů.

7.3.3 Naprogramování metod nalezení bodů světelného obrazce v jazyce C

Programování výše popsaných metod je nejlepší rozvést s pomocí ukázky zdrojového kódu programů. Základem pro obě metody je již zmiňovaný průchod snímkem pixel po pixelu.

Průchodu předchází načtení snímku do paměti a určení šířky řádku, což je zároveň počet sloupců snímku. V podkapitole 7.3 je poznamenáno, že se pro vyhodnocení jednotlivých pixelů používá pouze jejich jasová složka, což je při načítání snímku do paměti také nutno nastavit.

```
cc3_pixbuf_load ();
img.channels = 1; //nastaveni jednoho kanalu - jasove slozky
img.width = cc3_g_pixbuf_frame.width;
img.height = 1;
img.pix = cc3_malloc_rows (1);

...

cc3_pixbuf_frame_set_coi(CC3_CHANNEL_VAL); //jasova slozka HSV
```

Po tomto načtení snímku a jeho nastavení následuje průchod snímku. Tři tečky zastupují část kódu, která je specifická pro každou metodu.

```
while (cc3_pixbuf_read_rows (img.pix, 1))
{
    for (uint16_t x = 0; x < img.width; x++)
    {
        uint8_t val = ((uint8_t *) img.pix)[x];

        ...
    }

    ...

    y++;
}
```

Cyklus `while` se opakuje tak dlouho, dokud snímek obsahuje řádky, které nebyly přečteny, s každým řádkem narůstá hodnota `y`. Cyklus `for` prochází řádek po jednom pixelu, narůstá proměnná `x` a ukládá se hodnota jasové složky pixelu do proměnné `val`.

Kompletní zdrojové kódy metod vyhledání skupin okrajových bodů a metody vyhledání všech bodů jsou uvedeny v příloze P I, pro vysvětlení principu ukládání výsledných bodů postačí vysvětlit podmínky ukládání.

U metody nalezení okrajových bodů jsou vytvořeny čtyři statická pole `prvnib[287]`, `poslednib[287]`, `prvnib_y[287]` a `poslednib_y[287]`. Velikost statických polí je omezena maximální šířkou řádku, což je při nejvyšším rozlišení kamery 287 pixelů, hodnoty prvků polí jsou před jejich použitím vynulovány. Do těchto statických polí jsou

pak nahrávají body, které byly vyhodnoceny jako platné. Je-li platný bod nalezen, testuje se následující podmínka:

```
if (prvniOK==0)
{
    prvniib[radkyspbod].xx=pbod.xx;
    prvniib[radkyspbod].yy=pbod.yy;
    prvniOK=1;
    radkyspbod++;
}
```

Proměnná `prvniOK` udává informaci o naplnění prvku pole `prvniib` pro aktuální řádek, ve kterém se vyhodnocují body. Je-li tato proměnná nulová, první nalezený bod je uložen a zvýší se proměnná `radkyspbod`, která udává jak množství prvků pole `prvniib`, tak počet řádků, ve kterých byly nalezeny platné body. Po vyhodnocení této podmínky je vždy uložen platný bod do pole `poslednib`. Tímto je zaručeno, že body v tomto poli jsou vždy posledními platnými body řádku. Pro ukládání do polí `prvniib_y` a `poslednib_y` platí tyto podmínky:

```
if (prvniib_y[pbod.xx].yy==0)
{
    prvniib_y[pbod.xx].xx=pbod.xx;
    prvniib_y[pbod.xx].yy=pbod.yy;
    citacy1++;
}

if (poslednib_y[pbod.xx].yy<=pbod.yy)
{
    poslednib_y[pbod.xx].xx=pbod.xx;
    poslednib_y[pbod.xx].yy=pbod.yy;
    citacy2++;
}
```

Jestliže má prvek pole `prvniib_y` s indexem `pbod.xx` hodnotu souřadnice `y` nulovou, to znamená, že ve sloupci `pbod.xx` zatím není žádný bod, pak je bod `pbod` uložen jako prvek pole `prvniib_y`. Zároveň je navýšena hodnota proměnné `citacy1`, která určuje počet nenulových prvků v poli `prvniib_y`. Druhá uvedená podmínka funguje obdobně pro pole `poslednib_y` s tím rozdílem, že se netestuje, zda-li je souřadnice `y` prvku s indexem `pbod.xx` nulová, ale menší než souřadnice `y` aktuálního platného bodu `pbod`.

U statických polí `prvniib_y` a `poslednib_y` je po načtení bodů nutné odstranit nulové hodnoty, aby byla tato pole naplněna pouze platnými body. K tomu slouží „čistící“ cykly `for` za hlavním cyklem průchodu snímku.

Metoda nalezení všech bodů je z hlediska programování jednodušší, obsahuje však navíc ještě sloučení bodů, které leží v těsné blízkosti a odstranění falešných bodů z výsledné množiny. Veškeré body se ukládají do prázdného statického pole `body_pole[300]` a následně, po průchodu celého snímku, se provádí sloučení blízkých bodů do jednoho bodu, jehož souřadnice jsou výsledkem průměrné hodnoty všech sloučených bodů.

```
for(i=0;i<mnozstvibod;i++)
{
    x_sum=0;
    y_sum=0;
    k=0;
    if(body_pole[i].xx==0 && body_pole[i].yy==0) continue;
    for(j=0;j<mnozstvibod;j++)
    {
        if((body_pole[j].xx==0 && body_pole[j].yy==0) || i==j)
            continue;
        if(((sqrt((((float)body_pole[i].xx)-
            ((float)body_pole[j].xx))*(((float)body_pole[i].xx)-
            ((float)body_pole[j].xx)))+(((float)body_pole[i].yy)-
            ((float)body_pole[j].yy))*(((float)body_pole[i].yy)-
            ((float)body_pole[j].yy))))<5)
        {
            if(x_sum==0) x_sum=body_pole[i].xx;
            else x_sum=x_sum+body_pole[j].xx;
            if(y_sum==0) y_sum=body_pole[i].yy;
            else y_sum=y_sum+body_pole[j].yy;
            pomocne[k]=j;
            k++;
        }
    }
    body_pole[i].xx=(uint16_t)((float)x_sum/(float)k);
    body_pole[i].yy=(uint16_t)((float)y_sum/(float)k);
    for(j=0;j<k;j++)
    {
        body_pole[pomocne[j]].xx=0;
        body_pole[pomocne[j]].yy=0;
    }
}
```

Proměnná `mnozstvibod` určuje celkovou délku pole `body_pole` před sloučením jeho prvků. Proměnné `x_sum` a `y_sum` ukládají součty slučovaných bodů pro výpočet průměrné hodnoty a `k` určuje jejich počet, rovněž pro výpočet průměru. Celé pole se prochází bod po bodu, limitní vzdálenost pro sloučení byla určena 5 pixelů. Při slučování dochází k uchování indexů prvků, které byly sloučeny, do pomocného statického pole `pomocne[300]`, jako poslední krok jsou pak prvky mající indexy uložené v tomto poli vymazány.

7.4 Sestrojení přímek

V rámci práce byla vyvinuta řada metod pro řešení sestrojení přímek určujících vlastnosti světelného obrazce a tak i roviny, na kterou je promítán. Jedná se především o okrajové přímky obrazce, ze kterých je pak možno sestrojiti čtyři body udávající charakteristický čtyřúhelník. Základem těchto metod jsou pak některé z algoritmů uvedených v podkapitole 1.3. Sestrojení přímek v rámci programů probíhá pouze tehdy, je-li nalezen dostatečný počet bodů, ze kterých je možné přímky konstruovat.

7.4.1 Metoda s použitím lineární regrese

První testovanou metodou sestrojení přímek byla metoda založená na lineární regresi, tedy na metodě nejmenších čtverců. Pro sestrojení přímek pomocí této metody se využívá čtyř polí okrajových bodů. Je-li ovšem použita celé pole bez odebrání bodů, které leží mimo předpokládanou přímku, je výsledná přímka těmito body ovlivněna a klesá tak přesnost metody. Proto bylo přistoupeno k algoritmu, který má za úkol nejprve odebrat z polí okrajových bodů body, které jsou mimo předpokládanou přímku. Odebrání je založeno na nalezení nejčtenějších hodnot souřadnic a následném odstranění bodů, jejichž daná souřadnice překračuje vzdálenost od souřadnice nejčtenější. Zdrojový kód je pro každé ze čtyř polí stejný, pro popis bylo vybráno odstranění falešných bodů podle četnosti v poli bodů `prvnib`.

```
for (j=0;j<287;j++) //naplneni pomocneho pole jenom .xx
{
    if (prvnib[j].xx!=0)
    {
        for(o=0;o<287;o++)
        {
            if (cisticpole[o].xx==0)
            {
                cisticpole[o].xx=prvnib[j].xx;
                o=287;
            }
            if (o<287){
                if (cisticpole[o].xx==prvnib[j].xx)
                {
                    cisticpole[o].yy++; //navyseni cetnosti
                    o=287;
                }
            }
            pocetx++;
        }
    }
}
for (j=0;j<pocetx;j++)
{
```

```

        if(((float)cisticpole[j].yy)>=x_sum)
        {
            x_sum=(float)cisticpole[j].yy;
            x_avg=(float)cisticpole[j].xx;
        }
    }
for (j=0;j<pocetx;j++)
{
    if ((float)(abs(((int)prvnib[j].xx)-((int)x_avg)))>=7)
    {
        prvnib[j].xx=0;
        prvnib[j].yy=0;
    }
}
for (j=0;j<287;j++)
{
    cisticpole[j].xx=0;
    cisticpole[j].yy=0;
}
delka1 = 0;
for (o=0;o<287;o++)
{
    if (prvnib[o].xx!=0 || prvnib[o].yy!=0)
    {
        cisticpole[delka1].xx=prvnib[o].xx;
        cisticpole[delka1].yy=prvnib[o].yy;
        delka1++;
    }
}
for (j=0;j<287;j++)
{
    prvnib[j].xx=0;
    prvnib[j].yy=0;
}
if (delka1>0) delka1--;
for (o=0;o<=delka1;o++)
{
    prvnib[o].xx=cisticpole[o].xx;
    prvnib[o].yy=cisticpole[o].yy;
}

```

Prvním cyklem `for` se řeší naplnění pomocného statického pole struktur o dvou prvcích `cisticpole[287]` hodnotami souřadnic `x` z pole `prvnib`, při čemž je-li hodnota v pomocném poli již zastoupena, navýší se druhý prvek struktury na indexu této hodnoty. Po naplnění pomocného pole následuje jeho průchod, s uložením hodnoty s maximální četností (četnost je uložena v proměnné `x_sum`) do proměnné `x_avg`. Po určení těchto proměnných dojde k vynulování všech prvků v původním poli `prvnib`, které mají vyšší vzdálenost než je 7 pixelů od prvku s nejvyšší četností. Hodnota této vzdálenosti byla určena analýzou výsledků a závisí na vzdálenosti nástroje od roviny, která byla v rámci

testování neměnná. Samotná funkce vytvoření přímek se nazývá `linreg` a probíhá následovně.

```

for (j=0;j<287;j++)
{
    if (vstupy[j].xx==0 && vstupy[j].yy==0) break;
    x_sum=x_sum+(float)vstupy[j].xx;
    y_sum=y_sum+(float)vstupy[j].yy;
    x_pocet++;
    y_pocet++;
}

if(x_pocet==0) x_avg=0; else x_avg=x_sum/x_pocet;
if(y_pocet==0) y_avg=0; else y_avg=y_sum/y_pocet;

x_sum=0;
y_sum=0;

for (j=0;j<x_pocet;j++)
{
    x_sum=x_sum+(((float)vstupy[j].xx-
x_avg)*((float)vstupy[j].yy-y_avg));
    y_sum=y_sum+(((float)vstupy[j].xx-
x_avg)*((float)vstupy[j].xx-x_avg));
}

if (y_sum==0) primka_final.a=0; else primka_final.a=(x_sum/y_sum);
primka_final.b=y_avg-(primka_final.a*x_avg);

return primka_final;

```

Při průchodu polem bodů `vstupy` jsou hodnoty souřadnic `x` a `y` sečteny do proměnných `x_sum` a `y_sum` a je uložen jejich počet. Poté se vypočítá průměr hodnot souřadnic (za podmínky, že jejich počet je nenulový). Následuje vynulování proměnných `x_sum` a `y_sum`, aby mohly být využity v dalším cyklu, který počítá součty mocnin vzdáleností bodů od průměrných hodnot. Následně jsou vypočteny parametry `primka_final.a` a `primka_final.b` výsledné přímky, která je funkcí `linreg` vrácena. Pro případ, že je přímka kolmá, nastává uložení nulové hodnoty do `primka_final.a`. Tento případ by však nastat neměl vzhledem k charakteru vstupní množiny bodů – v případě funkce `linreg` se jedná o pole bodů `prvnib` a `poslednib`. Pro pole bodů `prvnib_y` a `poslednib_y` slouží k vytvoření přímek funkce `linreg_inv`, která je identická s funkcí `linreg`, až na vzájemnou výměnu souřadnic `x` a `y`, aby bylo zaručeno, že nedojde k případu kolmé přímky.

Pro ilustraci celého postupu je jako příloha P II uvedena analýza výsledných dat.

7.4.2 Metoda založená na algoritmu RANSAC

Metoda založená na algoritmu RANSAC byla druhou testovanou metodou sestavení přímk. Tato metoda opět vychází ze skupin okrajových bodů, ale tentokrát již nemusí být řešeno odebrání falešných bodů z množiny, protože algoritmus RANSAC tyto body odstraňuje ve své podstatě. Pro urychlení celého algoritmu byla naprogramována pouze zjednodušená verze algoritmu RANSAC, která nepřečítává přímky podle v jejich blízkosti nalezených bodů, ale pouze vybírá takovou, která má přiřazených bodů nejvíce. Funkci s algoritmem RANSAC je opět nejlépe popsatelná s pomocí zdrojového kódu.

```
for (i=0;i<=280;i++)
{
    prvek1=rand() % delka+1;
    while(1)
    {
        prvek2=rand() % delka+1;
        if (prvek1!=prvek2)
            break;
    }
    x_1=(float)vstupy[prvek1].xx;
    x_2=(float)vstupy[prvek2].xx;
    y_1=(float)vstupy[prvek1].yy;
    y_2=(float)vstupy[prvek2].yy;
    if (x_1==x_2)
        continue;
    else
    {
        pocet_primek++;
        a=(y_1-y_2)/(x_1-x_2);
        b=y_1-(a*x_1);
        sila_primky=0;
        odmocprodeleni=sqrt((a*a)+1);
        for(j=0;j<delka;j++)
        {
            bodx=(float)vstupy[j].xx;
            body=(float)vstupy[j].yy;
            vzdal=(absol((a*bodx)-body+b))/odmocprodeleni;
            if (vzdal<=limit)
                sila_primky++;
            if (sila_max==0 && pocet_primek==0)
                sila_max=sila_primky;
            if (sila_primky>sila_max)
            {
                sila_max=sila_primky;
                final.a=a;
                final.b=b;
            }
        }
    }
}
```

Na začátku funkce pojmenované `simple_RANSAC` je definován počet, udávající kolikrát se má náhodné vyhledání přímky provést. Čím vyšší tento počet je, tím vyšší je pravděpodobnost nalezení nejuvhodnější přímky, ale zároveň se prodlužuje doba trvání funkce. Proto byla zvolena hodnota, která byla v rámci testování vyhodnocena jako optimální. Opakující cyklus má za úkol náhodně nalézt dva prvky (proměnné `prvek1` a `prvek2`), které se sobě nerovnají, respektive dva indexy vstupního pole, ze kterého jsou vybrány dva body o těchto indexech a z nich je spočítána rovnice přímky. Pokud je nalezena kolmá přímka, cyklus pokračuje a kolmá přímka se vyloučí. Poté se nalezená přímka vyhodnocuje. Vyhodnocení probíhá tak, že cyklus `for` projde celé pole vstupních bodů po jednom bodě, při čemž je vždy spočítána vzdálenost bodu od přímky. Je-li tato vzdálenost menší než limitní (uložená v proměnné `limit`), narůstá proměnná `sila_primky`, jež porovnána s maximální hodnotou uloženou v proměnné `sila_max`. Překoná-li tuto maximální hodnotu, jsou parametry nalezené přímky považovány za výsledné a uloženy do struktury `final`, kterou funkce `simple_RANSAC` vrací. Analýza výsledků pro různé limitní hodnoty je uvedena v příloze P III.

7.4.3 Metoda pracující na bázi upraveného inkrementálního algoritmu

Všechny metody sestavení přímek zpracovávající skupiny okrajových bodů dosahovaly při testování chybných výsledků, vycházejících právě z charakteru skupin okrajových bodů. Proto bylo přistoupeno k metodě nalezení všech bodů, a následném vývoji metod sestavení přímek, počítajících se všemi body. Lineární regrese (metoda nejmenších čtverců) již nepřipadala v úvahu, protože jejím výsledkem by byla pouze jedna přímka, nikoliv potřebné čtyři, ale algoritmus jejího výpočtu byl nadále použit pro přepočítávání výsledných přímek (jejich zpřesňování). První vyvinutá metoda, pracující se všemi body, byla založena na upraveném inkrementálním algoritmu.

Principem metody je průchod polem vstupních bodů a výpočtu všech možných přímek sestavených z dvou odlišných bodů, a jejich uložení do pole struktur `primka`. Pokračování metody spočívá v dalším průchodu pole přímek a přiřazením bodů k přímkám, v jejich těsné blízkosti ležících a výpočtu procentuálního zastoupení všech bodů pod a nad přímkou. Přímky s nejvyšším počtem bodů na přímce a nejvyššími počty bodů pod či nad, jsou na závěr vyhodnoceny jako výsledné. Nutno podotknout, že během vývoje této metody bylo zjištěno, že je v rámci mikropočítače inteligentní kamery nerealizovatelná, protože nevystačí velikost paměti kamery. Proto zde nejsou popsány ani zdrojové kódy.

7.4.4 Metoda využívající všech bodů a algoritmu RANSAC

Výsledná používaná metoda se zakládá na zpracování pole všech bodů a algoritmu RANSAC. Algoritmus RANSAC je užit z důvodu jeho dobrých výsledků v metodě popsané v podkapitole 7.4.2 (viz také přílohu P III). Rozsáhlý zdrojový kód programu vytvořený pro aplikaci této metody je v příloze P IV, jeho nejdůležitější funkce (`prepocitej_primku`, `prirad_body`, `urci_NADPOD`, `ziskejprimkyRANSAC`) je ovšem třeba popsat.

Funkce `prepocitej_primku` slouží k přidávání vlastních bodů přímky do statického pole `vlastni[50]` struktury `primka` a zpřesnění konstant směrnicové rovnice přímky metodou nejmenších čtverců. Funkce `prirad_body` ověří polohu všech bodů ze vstupního pole k přímce, s využitím funkce `prepocitej_primku`, pokud je vzájemná vzdálenost bodu a přímky menší než vzdálenost limitní (uložena v globální proměnné `limit_vzdalenosti`). Za pomoci funkce `urci_NADPOD` jsou dané přímce určeny procentuální podíly všech bodů, nepřirazených k přímce, s kladnou orientovanou vzdáleností od přímky a zápornou orientovanou vzdáleností od přímky, pro kterou procentuální podíly určovány. Tyto podíly jsou pak uloženy do prvku `NAD` a `POD` struktury `primka`. Hlavní funkce metody se nazývá `ziskejprimkyRANSAC` a pracuje s generováním náhodných indexů vstupního pole bodů, při čemž pro každou náhodnou dvojici je spočítána přímka, jsou volány funkce popsané výše v tomto odstavci a přímky vyhovující podmínkám jsou vráceny jako výsledné.

Po naprogramování výsledné metody bylo třeba ji otestovat přímo v mikropočítači inteligentní kamery. Výsledky těchto testů jsou uvedeny v příloze P IV společně se zdrojovými kódy metody.

7.5 Nalezení čtyřúhelníku

Čtyři výsledné přímky, tvořící okraje světelného obrazce, jsou dále použity jako výchozí údaje pro výpočet bodů čtyřúhelníku, ze kterého je možno odvodit polohu nástroje. Pro výpočet čtyř bodů se provádí výpočet výsledku soustav rovnic složených ze čtyř párů odlišných přímek – respektive jsou vypočítány průsečíky různoběžných přímek, které tvoří čtyřúhelník. Obecně lze tuto soustavu vyjádřit vztahem (23).

$$\begin{aligned}
 y &= a_1x + b_1 \\
 y &= a_2x + b_2 \\
 x &= \frac{b_1 - b_2}{a_2 - a_1}
 \end{aligned}
 \tag{23}$$

Využitím vztahu (23) pro všechny vzájemné kombinace výsledných přímk získáme již zmiňované vrcholy čtyřúhelníku $A''B''C''D''$. Střed S'' leží na průsečíku úhlopříček $A''C''$ a $B''D''$ a jeho souřadnice S''_x a S''_y jsou odvozeny následovně:

$$S''_x = \frac{b_{A''C''} - b_{B''D''}}{a_{B''D''} - a_{A''C''}}, S''_y = a_{A''C''}S''_x + b_{A''C''}, \text{ kde} \tag{24}$$

$$a_{A''C''} = \frac{A''_y - C''_y}{A''_x - C''_x}, b_{A''C''} = A''_y - a_{A''C''}A''_x \text{ a} \tag{25}$$

$$a_{B''D''} = \frac{B''_y - D''_y}{B''_x - D''_x}, b_{B''D''} = B''_y - a_{B''D''}B''_x, \text{ kde} \tag{26}$$

$$A''_x \neq C''_x \text{ a } B''_x \neq D''_x. \tag{27}$$

Podmínky (27) jsou splněny, pokud body A'' a C'' či B'' a D'' neleží na kolmici či nejsou shodné. Body být shodné nemohou, což vyplývá z použití rozdílných přímk k jejich vytvoření. Na kolmici může ležet vždy jen jedna z dvojic, protože je zaručeno, že úhlopříčky čtyřúhelníka mají různý úhel vůči počátku. V případě, že se rovnají souřadnice A''_x a C''_x , souřadnice S''_x je jim také rovna. Analogicky pak platí stejné pravidlo pro dvojici B''_x a D''_x .

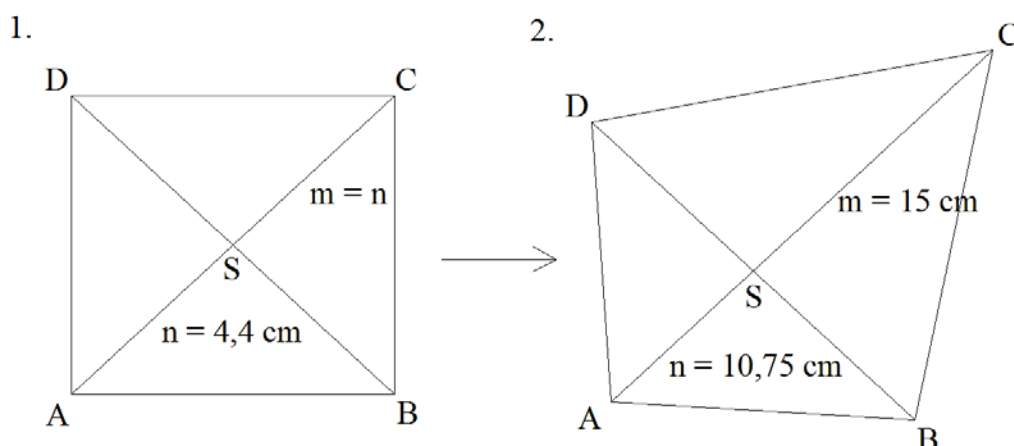
7.6 Odvození polohy nástroje

V rámci vývoje algoritmů pro odvození polohy nástroje byla provedena měření pro ověření matematického modelu. Tato měření se zakládala na vztazích odvozených v kapitole 6 a teorii uvedené v podkapitole 1.2.

7.6.1 Měření pro polohování nástroje

První část měření se zabývala ověřením výpočtu polohy bodů A , B , C a D (jejich vzdálenosti od bodu S) světelného obrazce v prostoru, když jsou známy poloha zdroje světelného obrazce (vzdálenost zdroje od bodu S , značena v_{ZS}), úhly mezi jednotlivými paprsky, které obrazec tvoří (značeno σ) a úhel roviny, na kterou je obrazec promítán vůči

rovině kolmé na středový paprsek (značeno α) – pro znázornění viz Obr. 18. Naměřená data byla graficky znázorněna na Obr. 25.



Obr. 25. Naměřené hodnoty délek (není zachováno měřítko).

Pro vzdálenost zdroje paprsků tvořících body A, B, C, D a S od bodu byla naměřena hodnota $v_{ZS} = 31,5$ cm pro polohu 1., kdy úhel $\alpha = 0^\circ$ a $v_{ZS} = 57,85$ cm pro polohu 2., kdy úhel $\alpha = 50^\circ$. Po dosazení do vztahu (19), kdy $v_{ZS} = h$, $\alpha = \theta_x$ a $m', n' = u, t$ vzniká vztah

$$m' = \frac{v_{ZS} \sin \sigma'}{\cos(\alpha - \sigma')} \text{ a } n' = \frac{v_{ZS} \sin \sigma'}{\cos(\alpha + \sigma')}. \quad (28)$$

Úhel σ mezi paprsky tvořící body A a B musí být přepočítán pro paprsky tvořící body A a C na nový úhel σ'' podle rovnice

$$\frac{\sigma''}{2} = \arctan\left(\sqrt{2} \tan \frac{\sigma}{2}\right). \quad (29)$$

Po dosazení známých hodnot ($\sigma = 6 \cdot 1,87^\circ$) do rovnice (29) získáme $\sigma'' = 15,82^\circ$. Paprsky tvořící bod A a S či S a C mají vzájemný úhel σ' roven polovině σ'' . Nyní již lze dosadit všechny neznámé a vypočítat tak m' a n' .

$$\text{Poloha 1.: } m' = n' = \frac{31,5 \cdot \sin 7,91^\circ}{\cos(0^\circ + 7,91^\circ)} = \underline{\underline{4,38 \text{ cm}}}. \quad (30)$$

$$\text{Poloha 2.: } m' = \frac{57,85 \sin 7,91^\circ}{\cos(50^\circ - 7,91^\circ)} = \underline{\underline{10,73 \text{ cm}}} \text{ a } n' = \frac{57,85 \sin 7,91^\circ}{\cos(50^\circ + 7,91^\circ)} = \underline{\underline{14,98 \text{ cm}}}. \quad (31)$$

Výsledky po dosazení v rovnicích (30) a (31) vycházejí téměř shodně s naměřenými hodnotami, odchylka je způsobena nepřesností prováděného měření (hodnoty vzdáleností nebyly měřeny pomocí přesných přístrojů).

Další měření proběhlo pro vzdálenost $v_{ZS} = 31,5$ cm, $\alpha = 20^\circ$, $m = 4,4$ cm, $n = 4,9$ cm. Po výpočtu pomocí vztahů (28) byly získány hodnoty $m' = 4,43$ cm a $n' = 4,91$ cm, tedy opět vychází $m \approx m'$ a $n \approx n'$.

Druhá část měření měla za úkol ověřit zobrazení bodů A, C a S v kameře pomocí perspektivní projekce při rotaci roviny, na kterou je světelný obrazec promítán, stejně jako v první části měření. Kamera vůči zdroji světla je posunuta o vzdálenosti $t_x = 3,5$ cm, $t_y = 2,2$ cm a $t_z = 3,9$ cm a není rotována, měření a výpočet proběhl pro vzdálenost $v_{ZS} = 31,5$ cm a úhel $\alpha = 20^\circ$. Pro polohu bodu A a C musí být vypočteny souřadnice A_x , A_y , A_z , C_x , C_y a C_z v prostoru vůči bodu S, který má určeny souřadnice 0, 0, 0. Pro tyto souřadnice platí

$$\begin{array}{l} A_x = 0 \\ A_y = n' \cos \alpha \\ A_z = n' \sin \alpha \end{array} \left\| \begin{array}{l} C_x = 0 \\ C_y = m' \cos \alpha \\ C_z = m' \sin \alpha \end{array} \right. \quad (32)$$

Souřadnice A'_x , A'_y , C'_x , C'_y v kameře jsou získány přičtením vektoru translace (viz (6)) a provedením perspektivní projekce (5). Platí tedy

$$A'_x = f \frac{A_x + t_x}{A_z + t_z} \text{ a } A'_y = f \frac{A_y + t_y}{A_z + t_z}, \quad (33)$$

$$C'_x = f \frac{C_x + t_x}{C_z + t_z} \text{ a } C'_y = f \frac{C_y + t_y}{C_z + t_z}. \quad (34)$$

Dosazením (28) do jednotlivých souřadnic uvedených v (32), a následně (32) do vztahů (33) a (34), vznikají vztahy pro výpočet hodnot souřadnic v kameře

$$A'_x = f \frac{t_x}{\frac{v_{ZS} \sin \sigma}{\cos(\alpha - \sigma)} \sin \alpha + t_z} \text{ a } A'_y = f \frac{\frac{v_{ZS} \sin \sigma}{\cos(\alpha - \sigma)} \cos \alpha + t_y}{\frac{v_{ZS} \sin \sigma}{\cos(\alpha - \sigma)} \sin \alpha + t_z}, \quad (35)$$

$$C'_x = f \frac{t_x}{\frac{v_{ZS} \sin \sigma}{\cos(\alpha + \sigma)} \sin \alpha + t_z} \text{ a } C'_y = f \frac{\frac{v_{ZS} \sin \sigma}{\cos(\alpha + \sigma)} \cos \alpha + t_y}{\frac{v_{ZS} \sin \sigma}{\cos(\alpha + \sigma)} \sin \alpha + t_z}. \quad (36)$$

Po dosazení známých hodnot do vzorců (35) a (36) jsou obdrženy výsledné hodnoty souřadnic A'_x, A'_y, C'_x, C'_y v kameře. Z nich lze pak vypočítat vzdálenosti bodů A' a C' od bodu S' , podle Pythagorovy věty

$$v_{A'S'} = \sqrt{A_x'^2 + A_y'^2} \text{ a } v_{C'S'} = \sqrt{C_x'^2 + C_y'^2}. \quad (37)$$

Tyto vzdálenosti po vyčíslení vyšly

$$v_{A'S'} = \underline{\underline{f \cdot 1,38}} \text{ a } v_{C'S'} = \underline{\underline{f \cdot 1,34}}, \quad (38)$$

jejichž poměr $v_{A'S'}/v_{C'S'}$ je roven 1,03:1. Na reálném snímku z kamery byly naměřeny hodnoty vzdáleností 61 pixel a 59 pixel, které udávají poměr vzdáleností 1,034:1. Bylo tedy ověřeno, že měřené hodnoty odpovídají hodnotám vypočteným a porovnáním s poměrem vzdáleností m a n původních bodů A, C a S , jenž je roven 1,1:1 (pro úhel $\alpha = 20^\circ$ a vzdálenost $v_{ZS} = 31,5$ cm) bylo také prokázáno, že perspektivní projekce nezachovává poměry délek na jedné přímce.

7.6.2 Zvolená metoda polohování

V rámci demonstrace aplikace a jejího testování polohování byla zvolena metoda polohování nástroje robotické paže, mající za úkol vyrovnat nástroj robota do polohy kolmé k rovině, na kterou je promítán světelný obrazec. V kameře jsou po nalezení čtyřúhelníku vypočteny bod S' a jeho vzdálenosti od bodů A', B', C' a D' . Poté jsou tyto vzdálenosti dány do poměrů, ze kterých se odvozuje, v kterém směru a okolo které osy má být nástroj rotován tak, aby byly výsledné poměry 1:1 v obou osách. Zdrojový kód programu spuštěného v kameře je následující:

```
uhloprickaAC=spocitej_primku(bodA,bodB);
uhloprickaBD=spocitej_primku(bodA,bodB);

bodS.xx=(uint16_t)((uhloprickaAC.b-
uhloprickaBD.b)/(uhloprickaBD.a-uhloprickaAC.a));
bodS.yy=(uint16_t)(uhloprickaAC.a*(float)bodS.xx+uhloprickaAC.b);

stranaAS=sqrt(power((float)bodA.xx-(float)bodS.xx,2) +
power((float)bodA.yy-(float)bodS.yy,2));
stranaBS=sqrt(power((float)bodB.xx-(float)bodS.xx,2) +
power((float)bodB.yy-(float)bodS.yy,2));
stranaCS=sqrt(power((float)bodC.xx-(float)bodS.xx,2) +
power((float)bodC.yy-(float)bodS.yy,2));
stranaDS=sqrt(power((float)bodD.xx-(float)bodS.xx,2) +
power((float)bodD.yy-(float)bodS.yy,2));
```

```

if(stranaAS>stranaCS) { naklon_x_pls=1; naklon_x_min=0;}
else if(stranaAS<stranaCS) { naklon_x_pls=0; naklon_x_min=1;}
else { naklon_x_pls=0; naklon_x_min=0;}

if(stranaBS>stranaDS) { naklon_y_pls=1; naklon_y_min=0;}
else if(stranaBS<stranaDS) { naklon_y_pls=0; naklon_y_min=1;}
else { naklon_y_pls=0; naklon_y_min=0;}

```

Pro výpočet bodu S', v kódu zastoupen proměnnou bodS, musí být sestrojeny úhlopříčky uhloprickaAC a uhloprickaBD, v jejichž průsečíku leží (viz (24)). Poté jsou spočítány vzdálenosti od bodů A', B', C' a D' a uloženy do proměnných stranaAS, stranaBS, stranaCS a stranaDS. Program pokračuje porovnáním hodnot vzdáleností A'S', C'S' a B'S', D'S', resp. zjištěním jejich poměrů, z nichž je pak odvozena hodnota 0 či 1 proměnných naklon_x_pls, naklon_x_min, naklon_y_pls a naklon_y_min, které podávají informaci o směru rotace, kterou má robot vykonat.

7.7 Komunikace kamery s robotem

Konečná část programu připravuje data pro odeslání do řídicí jednotky robota přes sériový port a poté je odesílá.

```

char odeslani[6]; //posilani dat do robota

if(naklon_x_pls==1) odeslani[0]='1'; else odeslani[0]='0';
if(naklon_x_min==1) odeslani[1]='1'; else odeslani[1]='0';
if(naklon_y_pls==1) odeslani[2]='1'; else odeslani[2]='0';
if(naklon_y_min==1) odeslani[3]='1'; else odeslani[3]='0';

odeslani[4]='@';
odeslani[5]='\0';

printf("%s", (char*)odeslani);

```

Řídicí jednotka robota Stäubli Unimation TX40 je schopná zpracovávat data přes sériový port ve formátu datového typu *string* či libovolném číselném datovém typu (*int*, *float*, *double*). Byl vybrán datový typ *string*, tedy řetězec znaků (pole *char*), o délce 6 znaků, nazvaný odeslani[6]. První čtyři znaky obsahují informaci od směru rotace, která má být vykonána, předávanou podmínkami z proměnných naklon_x_pls, naklon_x_min, naklon_y_pls a naklon_y_min. Pátý znak je vždy roven znaku „@“, protože to vyžaduje formátování dat odesílaných do robota. Posledním znakem pole je ukončovací znak „\0“. Celý řetězec je pak odeslán příkazem `printf()`. Konfigurace sériového vstupu/výstupu kamery je v rámci její inicializace provedena příkazem `cc3_uart_init(0, CC3_UART_RATE_115200, CC3_UART_MODE_8N1, CC3_UART_BINMODE_TEXT);`.

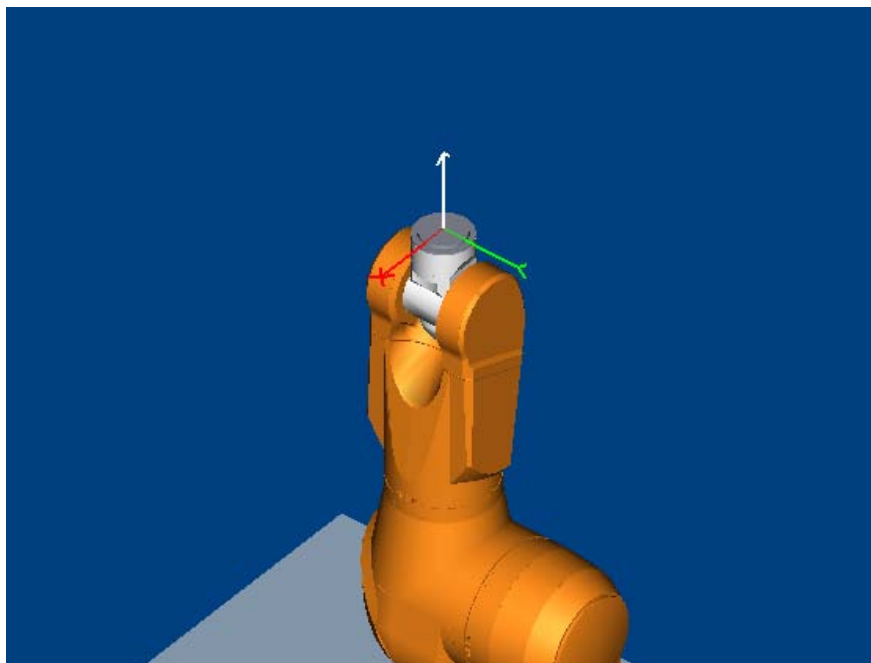
8 PROGRAMOVÁNÍ ROBOTA STÄUBLI UNIMATION TX40

Po zpracování obrazové informace a odvození potřebných dat k provedení polohování robotem v inteligentní kameře CMUcam3 následuje vytvoření aplikace v programovacím jazyce VAL3. Jak bylo popsáno v podkapitole 3.3, aplikace jazyka VAL3 se skládá z několika součástí, které musejí být vhodně naprogramovány.

Vytváření aplikace probíhalo přes ovládací panel, VAL3 Studio nebylo použito. Upřednostnění ovládacího panelu před komfortem vývojového prostředí proběhlo z důvodu potřeby učení robota polohám v prostoru. Nová aplikace byla pojmenována `a_gavenda1` a uložena do řídicí jednotky robota.

8.1 Nastavení nástroje a souřadného systému

Úplně prvním krokem po vytvoření nové aplikace je nadefinování nástroje. Jedná se o datový typ *tool*, který se skládá ze tří hodnot posunu a tří hodnot rotací vůči přírubě robotické paže. Nástroj definovaný v rámci této práce je umístěn 9 centimetrů v kladném směru osy z příruby a je otočen 90° kolem osy y příruby. Pro testovací účely polohování byla zadána pevná vzdálenost nástroje a roviny, vůči níž je polohován. Tato vzdálenost byla zvolena jako 50 centimetrů v kladném směru osy x příruby. Při definování nástroje musí být respektován souřadný systém příruby zobrazený na Obr. 26, kde červená polopřímka je kladný směr osy x, zelená osy y a bílá osy z.



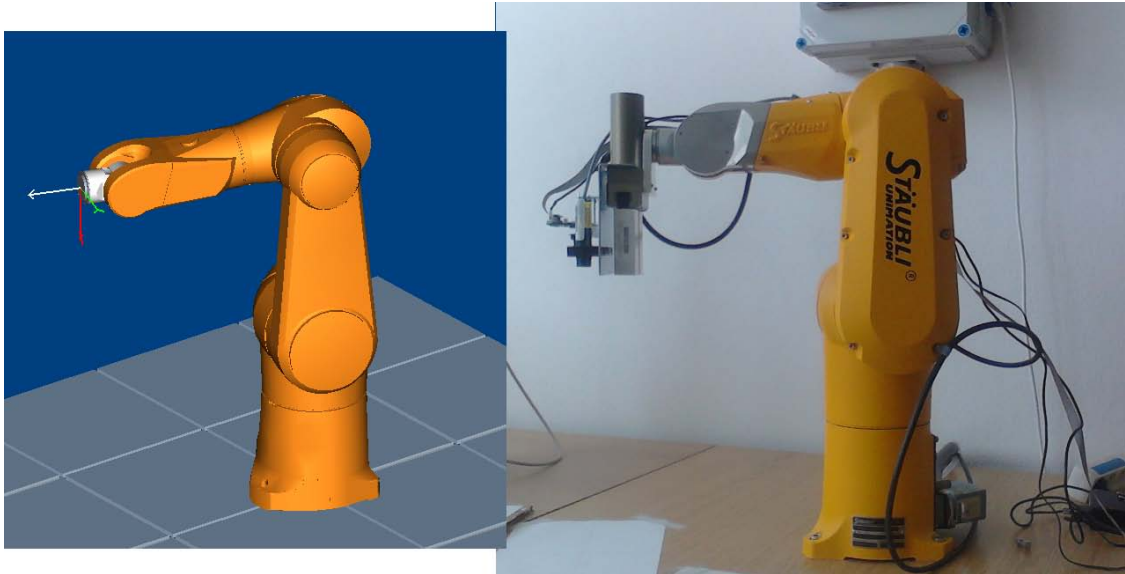
Obr. 26. Souřadný systém příruby ve výchozí pozici robota.

8.2 Příjem dat z kamery

Data z kamery jsou přijímána přes sériový port řídicí jednotky a jsou dále zpracovávána aplikací. Pro správnou komunikaci je nutné nastavit sériový vstup/výstup řídicí jednotky označený jako `portSerial1` tak, aby byl stejně nakonfigurován jako sériový vstup/výstup kamery. Nastavení se provádí pomocí ovládacího panelu řídicí jednotky a je třeba nastavit přenosovou rychlost (115200 baud), počet datových bitů (8 bitů), paritu (není), počet stop bitů (1 bit), čas spojení (zadána nulová hodnota pro nekonečný čas spojení), znakový oddělovač (znak „@“, v ASCII kódování 64), mód (RS232) a kontrolu datového toku (žádná). V rámci programu aplikace VAL3 je na začátku použit příkaz `sioLink(vstup,io:portSerial1)`, který má za úkol propojit proměnnou sériového vstupu/výstupu `sio vstup` s nastaveným vstupem/výstupem. Jelikož jsou data z kamery posílána v nekonečné smyčce a je rezervován čas pro vykonání polohování, musí být chráněn zásobník paměti řídicí jednotky proti přetečení pomocí příkazu `clearBuffer(io:portSerial1)` před každým načtením nových hodnot programem, čímž je také zaručena aktuálnost vstupních dat.

8.3 Aplikace pro polohování nástroje

Na začátku vytvořená aplikace `a_gavenda1` implicitně obsahuje programy `start()` a `stop()` a globální proměnné `world` a `flange` (viz podkapitola 3.3). Dále byly naprogramovány další globální proměnné – `nastroj`, `pocatek` a `rychlost`. Globální proměnná `nastroj` je datového typu `tool` a definuje nástroj robotické paže upevněný na přírubě (definice nástroje je popsána v podkapitole 8.1). Globální proměnná `pocatek` je datového typu `point`, udává bod v pracovním prostoru robota, který je znázorněn na Obr. 27 a natavuje počáteční (výchozí) polohu, ze které je nástroj polohován vůči rovině v pracovním prostoru robotické paže. Při spuštění aplikace, je robotická paže vždy přesunuta do tohoto výchozího bodu. Poslední globální proměnnou aplikace je `rychlost`, datového typu `mdesc`, obsahující informaci o rychlosti, jakou se má robotická paže pohybovat. Její hodnota je nastavena na 25% maximální rychlosti, aby byly zmírněny vibrace způsobené prudkým brzděním robotické paže v polohách, do kterých je přesouvána (robotická paže je ve výukové učebně umístěna na dřevěné ploše stolu, což způsobuje její vibrace při prudkých pohybech – konstrukce robotické paže ARM TX40 předpokládá umístění na pevném základu).



Obr. 27. Počáteční poloha robota na modelu a fotografii.

8.3.1 Hlavní program aplikace

Hlavní částí aplikace `a_gavenda1` je program `start()`, vykonávaný řídicí jednotkou vždy při spuštění aplikace. Zdrojový kód programu je rozdělen do několika částí.

```
sioLink(vstup,io:portSerial1)
x_rot=y_rot=0
rot_hod_x=0.4
rot_hod_y=0.1
cekat=0.1
userPage()
movej(pocatek,nastroj,rychlost)
waitEndMove()
clearBuffer(io:portSerial1)
```

Úvodní část programu lze nazvat částí inicializační. První příkaz propojuje proměnnou sériového vstupu/výstupu `vstup` se sériovým vstupem/výstupem `io:portSerial1` (viz podkapitola 8.2). Do číselných proměnných `x_rot` a `y_rot`, vyjadřujících rotaci nástroje okolo osy `x` a `y`, je nahrána počáteční nulová hodnota. Krok rotace při polohování je nastaven do číselných proměnných `rot_hod_x` a `rot_hod_y`. Proměnná `cekat` udává časovou prodlevu mezi pohyby robota v sekundách. Pro zobrazení přijímaných hodnot a hodnot rotací je v ovládacím panelu řídicí jednotky vyvolána uživatelská obrazovka pomocí funkce `userPage()`. Funkce `movej(pocatek,nastroj,rychlost)` přesouvá nástroj robotické paže do bodu `pocatek`, následuje příkaz pro čekání na dokončení pohybu robotické paže a příkaz pro vyčištění zásobníku paměti sériového vstupu/výstupu, aby byla zaručena aktuální data z kamery v další části programu.

```

while (true)
  delay(cekat)
  clearBuffer(io:portSerial1)
  vstupstr=vstup
  toNum(mid(vstupstr,1,0),pod_x_pls,prev)
  toNum(mid(vstupstr,1,1),pod_x_min,prev)
  toNum(mid(vstupstr,1,2),pod_y_pls,prev)
  toNum(mid(vstupstr,1,3),pod_y_min,prev)

```

Počáteční část nekonečného cyklu, kterým program po úvodní části pokračuje, zpracovává vstupní data z kamery. Jelikož je první funkcí `delay(cekat)`, mající za úkol potlačit vliv vibrací robotické paže na světelný obrazec tvořený laserem, musí být opět vyčištěn zásobník paměti sériového rozhraní. Do proměnné `vstupstr` datového typu *string* je přiřazena hodnota řetězce přijímaného z kamery. První čtyři znaky řetězce, nesoucí informaci o rotacích nástroje, jsou převedeny do číselných proměnných `pod_x_pls`, `pod_x_min`, `pod_y_pls` a `pod_y_min`.

```

  if pod_x_pls==1
    x_rot=x_rot-rot_hod_x
  endIf
  if pod_x_min==1
    x_rot=x_rot+rot_hod_x
  endIf
  if pod_y_pls==1
    y_rot=y_rot-rot_hod_y
  endIf
  if pod_y_min==1
    y_rot=y_rot+rot_hod_y
  endIf

```

Číselné proměnné `pod_x_pls`, `pod_x_min`, `pod_y_pls` a `pod_y_min` jsou vyhodnoceny podmínkami, a podle jejich hodnot je navýšena, snížena či nezměněna hodnota rotace v osách x a y.

```

  put("data ")
  putln(vstupstr)
  put("x rotace ")
  putln(x_rot)
  put("y rotace ")
  putln(y_rot)
  movej(appro(pocatek,{0,0,0,x_rot,y_rot,0}),nastroj,rychlost)
  waitEndMove()
  delay(cekat)
  clearBuffer(io:portSerial1)
endWhile

```

Závěrečná část programu vypisuje data a hodnoty rotací a uživatelskou obrazovku a volá funkci `movej(appro(pocatek, {0,0,0,x_rot,y_rot,0}),nastroj, rychlost)` pro samotný pohyb robotické paže. Robotická paže je přesunuta na pozici rotovanou o hodnoty proměnných `x_rot` a `y_rot` vůči počáteční poloze. Program uzavírá funkce pro čekání na dokončení pohybu robotické paže `waitEndMove()`, čekání proti nechtěným vibracím robotické paže `delay(cekat)` a opětovné odebrání nadbytečných dat ze zásobníku paměti `clearBuffer(io:portSerial1)`.

9 VYUŽITÍ ŘEŠENÍ V BEZPEČNOSTNÍM PRŮMYSLU

Pro předvedení možnosti implementace vyvinutého řešení do průmyslu komerční bezpečnosti byla vytvořena modelová aplikace, která demonstruje využití řešení v praxi.

9.1 Modelová aplikace řešení

Za současného stavu rozmachu použití servisní robotů také v průmyslu komerční bezpečnosti (viz kapitola 4) lze předpokládat, že servisní roboty budou využity také jako prostředky pro usnadnění montáží či oprav ostatních technických prostředků komerční bezpečnosti. Součástí těchto servisních robotů pak budou také senzorické části, které budou mít za úkol například polohování nástrojů pro interakci s okolím servisního robota. Jako příklad využití řešení vyvinutého v rámci této diplomové práce byla navržena modelová aplikace tohoto řešení.

V modelové aplikaci je nastolen problém přesného polohování nástroje mobilního servisního robota, majícího za úkol vyvrtat otvor pro vložení špionážní kamery do rovny dřevěné zdi (popř. umístění detektoru na zeď). Modelový mobilní servisní robot je řízen operátorem, který jej navede tak, aby byla zeď v dosahu nástroje. Pak je spuštěna vyvinutá aplikace, která autonomně polohuje nástroj tak, aby se nacházel v požadované poloze bez zásahu operátora. Při vrtání otvoru můžeme předpokládat požadovanou polohu kolmou k rovině zdi. Po skončení polohování je vydán signál operátorovi výsledku akce, otvor může být vyvrtán a kamera vložena.



Obr. 28. 3D model ukázkové aplikace.

ZÁVĚR

V práci je vyvinuto vlastní řešení problematiky optoelektronického polohování nástroje vůči rovině, které zahrnuje několik částí. Tyto části se dají rozdělit na sestavení vlastní aparatury nástroje s inteligentní kamerou a laserem, implementované na šestiosého kloubového robota Stäubli Unimation TX40, nastavení a naprogramování inteligentní kamery, vlastní matematický model a výpočty související s polohováním v prostoru, nastavení a naprogramování aplikace v řídicí jednotce robota a navržení modelové aplikace v oblasti bezpečnostních technologií.

Časově i obsahově nejdelší se stala část zabývající se zpracováním obrazové informace v inteligentní kameře. Bylo nutné zvolit správné nastavení kamery fungující za všech světelných podmínek a poté vyvinout a navrhnout vlastní algoritmy a programy zpracování obrazové informace. Po naprogramování vlastních metod do inteligentní kamery bylo dosaženo nalezení, pro účely práce, nejvhodnější metody, která dokáže vyhledat charakteristický čtyřúhelník (tvořený okrajovými body světelného obrazce) a pomocí něj určit potřebné informace pro výpočty týkající se polohování. Nalezení této metody bylo dosaženo pomocí testování a analýz dat (uvedených v rámci práce) odlišných způsobů vyhledání bodů laserem tvořeného strukturovaného světelného obrazce a sestrojení přímk. Nejvhodnější metoda se tedy zakládá na určení všech platných bodů světelného obrazce a sestrojení přímk pomocí algoritmu RANSAC.

Po výběru metody zpracování obrazové informace, pokračuje práce částí zabývající se matematickým modelem a výpočty z oblasti analytické geometrie a goniometrie. Odvozené rovnice a výsledky byly ověřeny pomocí měření skutečných dat. Jejich zpracování v rámci inteligentní kamery a implementace do řešení polohování je však nad rámec této práce. Pro polohování byla vyvinuta výpočetně a programově jednodušší varianta metody, která se zakládá na vypočítaných matematických vztazích.

Následně byla zrealizována vlastní implementace řešení na šestiosého kloubového robota Stäubli Unimation TX40. V rámci této implementace bylo dosaženo funkčního komunikačního spojení přes sériové rozhraní mezi inteligentní kamerou a robotem a byla naprogramována aplikace v řídicí jednotce robota, zpracovávající data odesílaná kamerou a přijímaná na straně robota.

Poslední částí práce je navržená aplikace řešení v bezpečnostním průmyslu s ohledem na současné směry v použití a vývoji servisních robotů v bezpečnostním odvětví. Pro účely

demonstrace této aplikace byl vytvořen 3D model servisního mobilního robota, jež má za úkol vizuálně dokreslit vlastní modelovou aplikaci řešení.

Dalším pokračováním práce bude vývoj metod pro určení polohy až v šesti stupních volnosti s umístěním inteligentní kamery mimo nástroj robotické paže. Dosavadní výsledky a řešení tvoří základ pro další práce v tématu optoelektronického polohování a přináší řadu vztahů, metod a algoritmů použitelných například v rámci doktorského studia.

ZÁVĚR V ANGLIČTINĚ

In the thesis is developed a custom made solution of optoelectronic positioning of tool to the plane surface, which consists of several parts. These parts can be divided to building an own tool apparatus with intelligent camera and laser, implemented on an six-axis joint-type robot Stäubli Unimation TX40, adjusting and programming of the smart camera, the mathematical model and calculations related to the positioning in space, setting and programming application in robot control unit and designing model applications in the scope of safety technologies.

The longest part in matter of time and content is the part dealing with processing of image information in the smart camera. It was necessary to choose the right camera settings, operating under all lighting conditions and then develop and design own algorithms and programs for image processing. After programming own methods to the smart camera was achieved to find, for the purpose of work, the best method that can locate the characteristic quadrilateral (formed by edges of a light pattern) and use it to identify the necessary information for calculations related to positioning. Finding this method was achieved by testing and data analysis (listed in the work) of different methods for locating points of the laser-made structured light pattern and construction of lines. The best method is based on identifying all valid points of light pattern and constructing lines using the RANSAC algorithm.

After selecting the image information processing method, the thesis continues with a part dealing with the mathematical model and calculations from the field of the analytical geometry and goniometry. Derived equations and the results of calculations were verified by measuring the actual data. Their processing within the smart camera and the implementation of the positioning solution is beyond this thesis. For positioning was developed and programmed computationally simpler alternative method that is based on the calculated mathematical relationships.

Consequently, own implementation of the solution on the six-axis robot Stäubli Unimation TX40 was realized. In this implementation the functional communication connection via a serial interface between the smart cameras and the robot was achieved and application in robot control unit processing the data sent by the camera and received by the robot was programmed.

The last section of the thesis is designed application of the solution in the safety industry with regard to current trends in the use and development of service robots in the safety industry. For demonstration purposes was created a 3D model of mobile service robot, which is supposed to visually illustrate model application of the solution.

The continue of the thesis will be the development of methods for identifying the position in up to six degrees of freedom with placing the smart camera outside the robotic arm tool. Actual results and solutions form the basis for further work on the topic of optoelectronic positioning and deliver a range of relations, methods and algorithms applicable for example in the doctoral studies.

SEZNAM POUŽITÉ LITERATURY

- [1] SURÝNEK, Tomáš. *Určení vzdálenosti cíle hloubkoměrným principem se strukturovaným světlem*. Zlín, 2008. 63 s. Diplomová práce. Univerzita Tomáše Bati, FAI.
- [2] *Robotics - SCARA and 6 axis industrial robots & software solutions - Staubli* : [online]. 2010 [cit. 2011-05-02]. Dostupné z WWW: <<http://www.staubli.com/en/robotics/>>.
- [3] ŽÁRA, Jiří, et al. *Moderní počítačová grafika*. Brno : Computer Press, 2004. 612 s. ISBN 80-251-0454-0.
- [4] *JISC Digital Media* [online]. 2011 [cit. 2011-05-03]. Dostupné z WWW: <http://www.jiscdigitalmedia.ac.uk/images/rgb_01.gif>.
- [5] *Geomatika na ZČU v Plzni* [online]. 2008 [cit. 2011-05-03]. Dostupné z WWW: <<http://gis.zcu.cz/studium/pok/Materialy/Book/resources/hsv.PNG>>.
- [6] *Wikipedia, the free encyclopedia : HSL and HSV* [online]. 2005 [cit. 2011-05-03]. Dostupné z WWW: <http://en.wikipedia.org/wiki/HSL_and_HSV>.
- [7] DUDEK, Gregory; JENKIN, Michael. *Computational Principles of Mobile Robotics*. New York : Cambridge University Press, 2000. 280 s. ISBN 0-521-56876-5.
- [8] FORSYTH, David A.; PONCE, Jean. *Computer Vision: A Modern Approach*. Upper Saddle River : Pearson Education, Inc., 2003. 693 s. ISBN 0-13-085198-1.
- [9] *Analytická geometrie - Geometrie v rovině - Parametrické vyjádření přímky* [online]. 2009 [cit. 2011-05-04]. Dostupné z WWW: <http://www.karlin.mff.cuni.cz/katedry/kdm/diplomky/jan_koncel/rovina.php>.
- [10] NGUYEN, Viet, et al. *A Comparison of Line Extraction Algorithms using 2D Laser Rangefinder for Indoor Mobile Robotics*. Edmonton : IROS, 2005. 6 s.
- [11] CHOSET, Howie, et al. *Principles of Robot Motion : Theory, Algorithms and Implementation*. Cambridge : The MIT Press, 2005. 603 s. ISBN 0-262-03327-5.
- [12] *Wikipedia, the free encyclopedia : RANSAC* [online]. 2007 [cit. 2011-05-05]. Dostupné z WWW: <http://en.wikipedia.org/wiki/File:Fitted_line.svg>.

- [13] *Center for Machine Perception* [online]. 2008 [cit. 2011-05-06]. Houghova transformace. Dostupné z WWW: <<http://cmp.felk.cvut.cz/cmp/courses/ZS1/Cviceni/cv5/hough.htm>>.
- [14] *CMUcam* [online]. 2006 [cit. 2011-05-12]. Dostupné z WWW: <<http://cmucam.org/>>.
- [15] *NXP Semiconductors - Microcontrollers* [online]. 2006 [cit. 2011-05-19]. Dostupné z WWW: <<http://ics.nxp.com/microcontrollers/>>.
- [16] *ARM7 Processor Family - ARM* [online]. 2011 [cit. 2011-05-19]. Dostupné z WWW: <<http://www.arm.com/products/processors/classic/arm7/index.php>>.
- [17] *OmniVision* [online]. 2011 [cit. 2011-05-19]. Dostupné z WWW: <<http://www.ovt.com/>>.
- [18] *CMUcam3 Datasheet*. Pittsburgh : Carnegie Mellon University, 2007. 34 s.
- [19] *Software – CMUcam* [online]. 2007 [cit. 2011-05-03]. Dostupné z WWW: <<http://www.cmucam.org/wiki/Software>>.
- [20] *Arm – TX series 40 family : Instruction Manual*. Faverges : Stäubli Faverges SCA, 2007. 95 s.
- [21] *CS8C Controller : Instruction Manual*. Faverges : Stäubli Faverges SCA, 2007. 259 s.
- [22] *VAL3 Reference Manual : Version 6*. Faverges : Stäubli Faverges SCA, 2008. 186 s.
- [23] *Lasiris SNF Laser*. Montreal : StockerYale, 2008. 4 s. Dostupné z WWW: <<http://www.photonic-sourcing.com/public/photonics/lasiris-snf-lasers-292.pdf>>.
- [24] *Wolfram Mathematica* [online]. 2010 [cit. 2011-04-15]. Dostupné z WWW: <<http://www.wolfram.com/mathematica/>>.
- [25] GAVENDA, Tomáš. *Řízení polohování nástroje průmyslového robota Stäubli Unimation TX-40 pomocí snímání světelného obrazce*. Ostrava, 2011 (ke dni 23. 5. 2011 zatím nevydáno). 10 s. Soutěžní práce. STOČ 2011 v Ostravě.
- [26] *CMUcam* [online]. 2007, poslední změna 2.8.2010 [cit. 2011-01-31]. Documentation – CMUcam. Dostupné z WWW: <<http://cmucam.org/wiki/Documentation>>.

- [27] Servisní roboty dobývají svět. *Automa* [online]. 2010, č. 5, [cit. 2011-05-10]. Dostupný z WWW: <http://www.odbornecasopisy.cz/index.php?id_document=41046>.
- [28] *Wikipedia, the free encyclopedia : Foster-Miller TALON* [online]. 2007 [cit. 2011-05-20]. Dostupné z WWW: <<http://en.wikipedia.org/wiki/SWORDS>>.
- [29] *Msnbc.com : Surveillance robots patrol World Cup* [online]. 2006 [cit. 2011-05-20]. Dostupné z WWW: <http://www.msnbc.msn.com/id/13485246/ns/technology_and_science-innovation/t/surveillance-robots-patrol-world-cup/>.
- [30] *Ministerstvo obrany : Robot EOD tEODor* [online]. 2004 [cit. 2011-05-20]. Dostupné z WWW: <<http://www.army.cz/scripts/detail.php?id=6343>>.
- [31] *Policie České republiky : Při nehodě na D5 zasahovali i pyrotechnici* [online]. 2011 [cit. 2011-05-20]. Dostupné z WWW: <<http://www.policie.cz/clanek/pri-nehode-na-d5-zasahovali-i-pyrotechnici.aspx>>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CMOS	<i>Complementary Metal-Oxide Semiconductor</i> – technologie pro výrobu integrovaných obvodů.
RGB	<i>Red Green Blue</i> – označení barevného prostoru.
Y _C B _B C _R	Označení barevného prostoru.
HSV, HSL	Označení barevných prostorů.
RANSAC	<i>RANdom SAmples Consensus</i> – název algoritmu pro vyhledávání křivek.
CMUcam3	Obchodní označení inteligentní kamery.
CMUcam2	Obchodní označení inteligentní kamery.
LPC2106	Název typu procesoru.
ARM7TDMI	Označení jádra procesoru.
OV7620	Název typu kamerového modulu.
CPU	<i>Central Processing Unit</i> – procesor, základní součást počítače.
RAM	<i>Random Access Memory</i> – paměť s přímým přístupem.
KB, MB	Kilobajt, megabajt – jednotka velikosti paměti.
ROM	<i>Read Only Memory</i> – paměť s pevně daným obsahem z výroby.
MHz	Megahertz – jednotka frekvence.
FIFO	<i>First In First Out</i> – abstraktní datový typ fronty.
OS	Operační systém.
JPEG	Metoda ztrátové komprese pro ukládání obrázků.
ARM TX40	Název typu robotické paže.
CS8C	Firemní označení řídicí jednotky robota.
VAL3	Název programovacího jazyku robota.
IP65, IP67	Typ krytí proti vodě a prachu.
kg	Kilogram – jednotka hmotnosti.
nm, mm, cm	Nanometr, milimetr, centimetr – jednotky vzdálenosti.

RSI	Firemní označení ochranné a napájecí základní desky.
USB	<i>Universal Serial Bus</i> – univerzální sériová sběrnice.
PCI	<i>Peripheral Component Interconnect</i> – sběrnice pro připojení periférií k základní desce.
LCD	<i>Liquid Crystal Display</i> – displej z tekutých krystalů.
LED	<i>Light-Emitting Diode</i> – dioda emitující světlo.
kart. s. s.	Kartézský souřadnicový systém.
SWORDS	<i>Special Weapons Observation Reconnaissance Detection System</i> – typ bojového robota.
tEODor	Označení typu pyrotechnického servisního robota.
ČR	Česká republika
D5	Označení dálnice v České republice.
SNF Laser	Obchodní označení laseru společnosti StockeYale
mW	Miliwatt – jednotka výkonu.
507X	Obchodní označení optiky laseru.
°	Stupeň – jednotka rovinného úhlu.
3D	Trojrozměrný prostor.
STOČ	Studentská tvůrčí a odborná činnost – mezinárodní studentská soutěž.
pixel	Obrazový bod.
Jazyk C	Název programovacího jazyka.
ASCII	<i>American Standard Code for Information Interchange</i> – kódová tabulka definující znaky anglické abecedy.
RS232	Komunikační rozhraní – také sériová linka.

SEZNAM OBRÁZKŮ

Obr. 1. Aditivní skládání barev [4].	12
Obr. 2. Znázornění prostoty HSV [5].	13
Obr. 3. Algoritmus převodu barevných prostorů RGB a HSV [6].	14
Obr. 4. Model dírkové komory.	14
Obr. 5. Výsledek algoritmu RANSAC [12].	18
Obr. 6. Fotografie kamery CMUcam3.	19
Obr. 7. Robotická paže ARM TX40 [20].	21
Obr. 8. Řídicí jednotka CS8C [21].	22
Obr. 9. Ovládací panel MCP.	23
Obr. 10. Přepínač pracovního módu [21].	24
Obr. 11. Ukázka programu ve VAL3 Studiu.	25
Obr. 12. Vojenský robot typu SWORDS [28].	27
Obr. 13. Servisní robot typu tEODor Armády ČR [30].	27
Obr. 14. Fotografie nástroje robotické paže.	29
Obr. 15. Zjednodušený konstrukční výkres nástavce nástroje.	29
Obr. 16. Fotografie světelného obrazce ve formě matice bodů.	30
Obr. 17. 3D znázornění matematického modelu.	31
Obr. 18. Závislost rotace roviny obrazce a poměru úseček.	32
Obr. 19. Totožné snímky s rozdílným rozlišením.	35
Obr. 20. Znázornění přesnosti vyhledávání bodů.	37
Obr. 21. Skupiny okrajových bodů bez očištění od falešných bodů.	38
Obr. 22. Srovnání výsledků před a po očištění podle četnosti.	39
Obr. 23. Všechny nalezené body před sloučením.	40
Obr. 24. Všechny nalezené body po sloučení.	40
Obr. 25. Naměřené hodnoty délek (není zachováno měřítko).	51
Obr. 26. Souřadný systém příruby ve výchozí pozici robota.	55
Obr. 27. Počáteční poloha robota na modelu a fotografii.	57
Obr. 28. 3D model ukázkové aplikace.	60

SEZNAM TABULEK

Tab. 1. Charakteristika hardwaru inteligentní kamery CMUcam3 [18].	19
Tab. 2. Specifikace Lasiris SNF Laseru [23].	30
Tab. 3. Testovací snímky pořízené pomocí nástroje CMUcam3 Frame Grabber.	34
Tab. 4. Příkazy pro nastavení emulátoru CMUcam2 [26].	35

SEZNAM PŘÍLOH

- P I Zdrojové kódy metod pro nalezení bodů.
- P II Analýza výsledných dat metody založené na lineární regresi.
- P III Analýza výsledných dat metody založené na algoritmu RANSAC.
- P IV Zdrojový kód výsledné metody.

PŘÍLOHA P I: ZDROJOVÉ KÓDY METOD PRO NALEZENÍ BODŮ

Metoda nalezení okrajových bodů:

```
while (cc3_pixbuf_read_rows (img.pix, 1))
{
    mnozstvipred = mnozstvi;

    for (uint16_t x = 0; x < img.width; x++)
    {
        uint8_t val = ((uint8_t *) img.pix)[x];
        if (val > mezjasu) //hodnota jasove slozky
        {
            if (mnozstvi==0)
            {
                pbod.xx=x;
                pbod.yy=y;
            }
            else
            {
                if (abs(pbod.xx-x)<=presnost)
                {
                    pbodsila++;
                    if (pbodsila>=1)
                    {
                        mnozstvipbod++;
                        findpbod=1;
                    }
                }
                else
                {
                    pbod.xx=x;
                    pbod.yy=y;
                    pbodsila=0;
                    findpbod=0;
                }
            }
        }
        if (findpbod==1)
        {
            if (prvniOK==0)
            {
                prvnib[radkyspbod].xx=pbod.xx;
                prvnib[radkyspbod].yy=pbod.yy;
                prvniOK=1;
                radkyspbod++;
            }
            poslednib[radkyspbod-1].xx=pbod.xx;
            poslednib[radkyspbod-1].yy=pbod.yy;

            if (prvnib_y[pbod.xx].yy==0)
            {
                prvnib_y[pbod.xx].xx=pbod.xx;
                prvnib_y[pbod.xx].yy=pbod.yy;
                citacyl1++;
            }
        }
    }
}
```

```

        if (poslednib_y[pbod.xx].yy<=pbod.yy)
        {
            poslednib_y[pbod.xx].xx=pbod.xx;
            poslednib_y[pbod.xx].yy=pbod.yy;
            citacy2++;
        }
        findpbod=0;
    }
    mnozstvi++;
}
}

if (mnozstvipred!=mnozstvi) radky++;
y++;
prvniOK=0;
}

```

```

if (mnozstvipbod>600) mezjasu++;
if (mnozstvipbod<=80 && mnozstvipbod!=0) mezjasu--;

```

Metoda nalezení všech bodů:

```

while (cc3_pixbuf_read_rows (img.pix, 1))
{
    for (uint16_t x = 0; x < img.width; x++)
    {
        uint8_t val = ((uint8_t *) img.pix)[x];

        if (val > mezjasu) //hodnota jasove slozky
        {
            if (mnozstvi==0)
            {
                pbod.xx=x;
                pbod.yy=y;
            }
            else
            {
                if (abs(pbod.xx-x)<=presnost_x)
                {
                    pbodsila++;
                    if (pbodsila>=1)
                    {
                        if(findpbod==0) mnozstvipbod++;
                        findpbod=1;
                    }
                }
                else
                {
                    pbod.xx=x;
                    pbod.yy=y;
                    pbodsila=0;
                    findpbod=0;
                }
            }
        }
    }
}

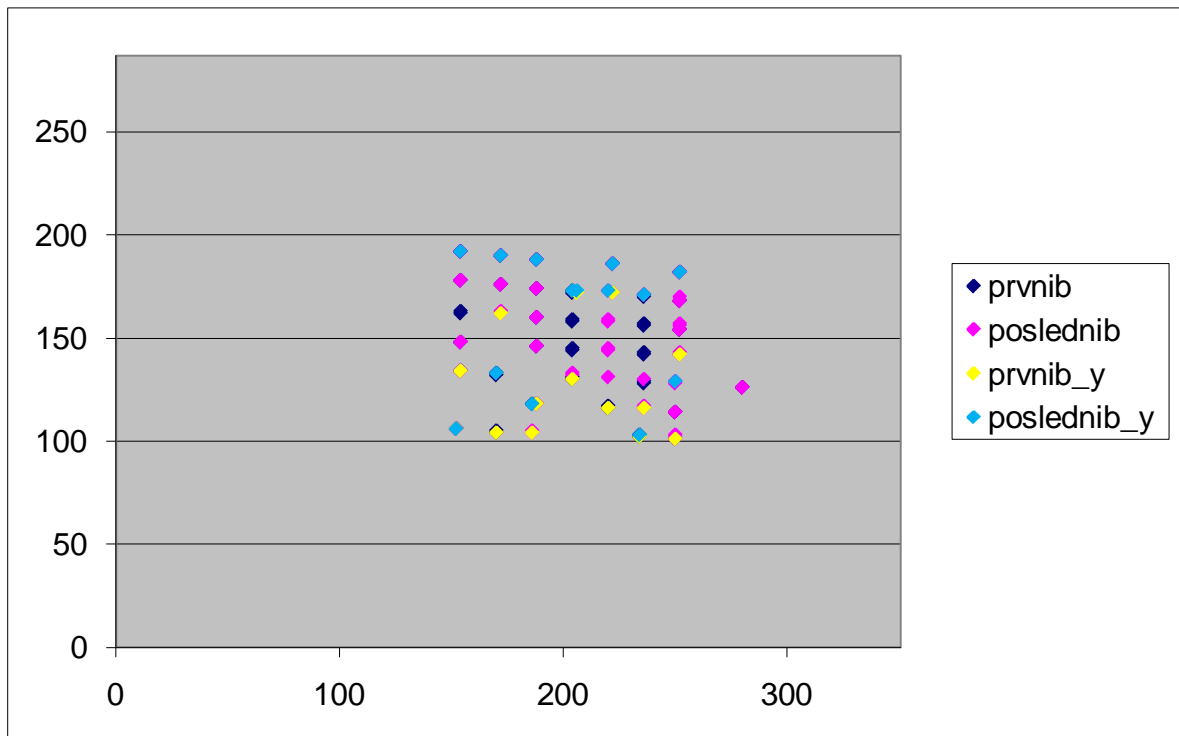
```

```
        if (findpbod==1 && mnozstvipbod<301)
        {
                body_pole[mnozstvipbod-1].xx=pbod.xx;
                body_pole[mnozstvipbod-1].yy=pbod.yy;
        }
        mnozstvi++;
    }
}
y++;
}

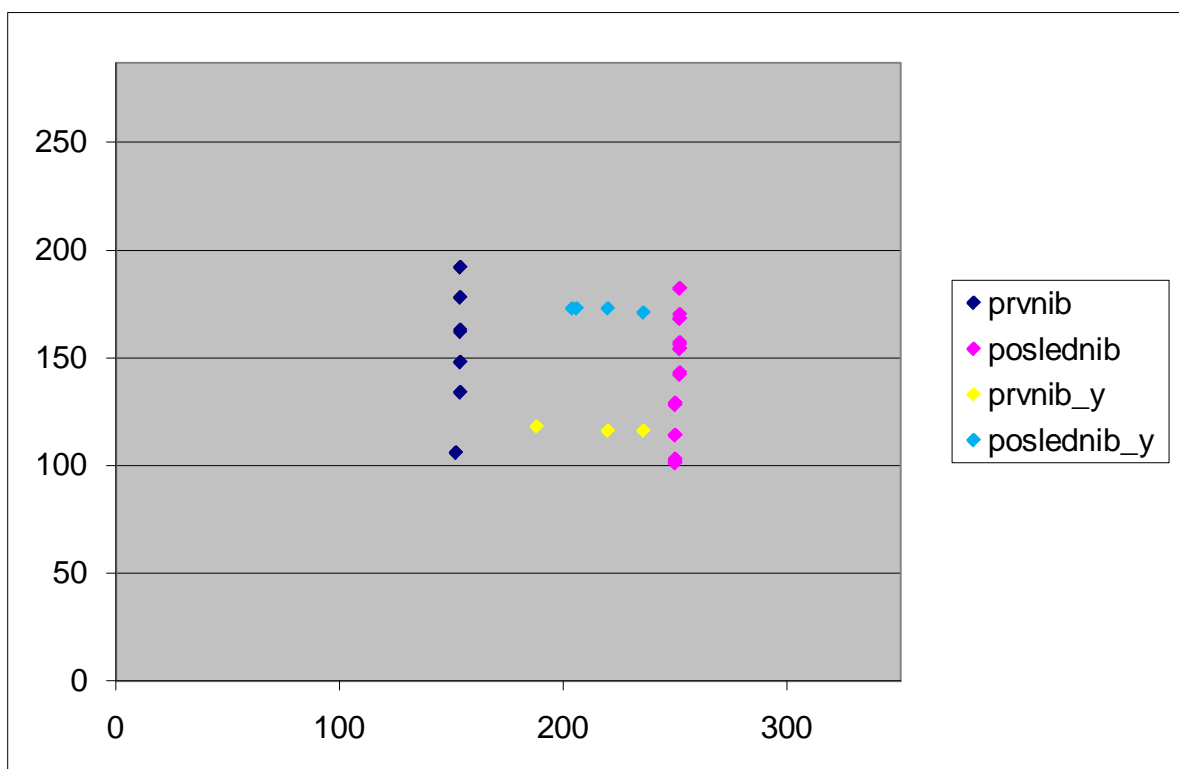
if (mnozstvipbod>300) mezjasu++;
if (mnozstvipbod<=150 && mnozstvipbod!=0) mezjasu--;
```

PŘÍLOHA P II: ANALÝZA VÝSLEDNÝCH DAT METODY ZALOŽENÉ NA LINEÁRNÍ REGRESI

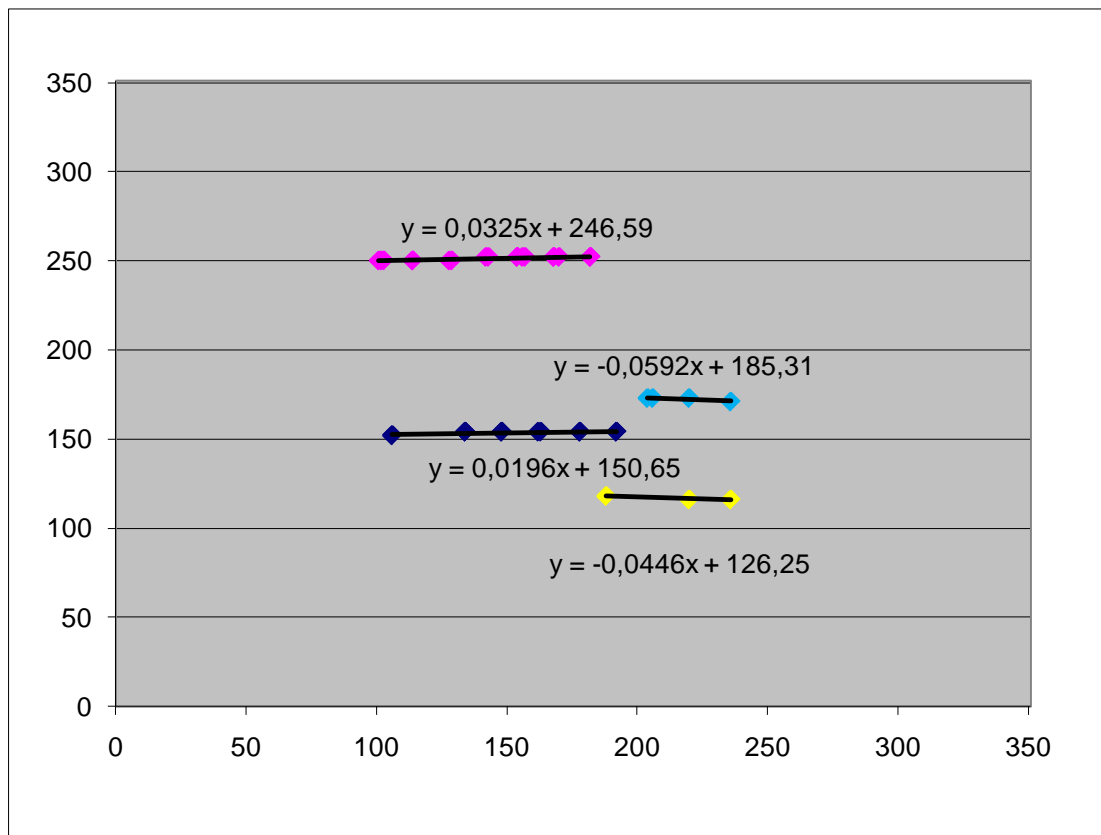
Původní data:



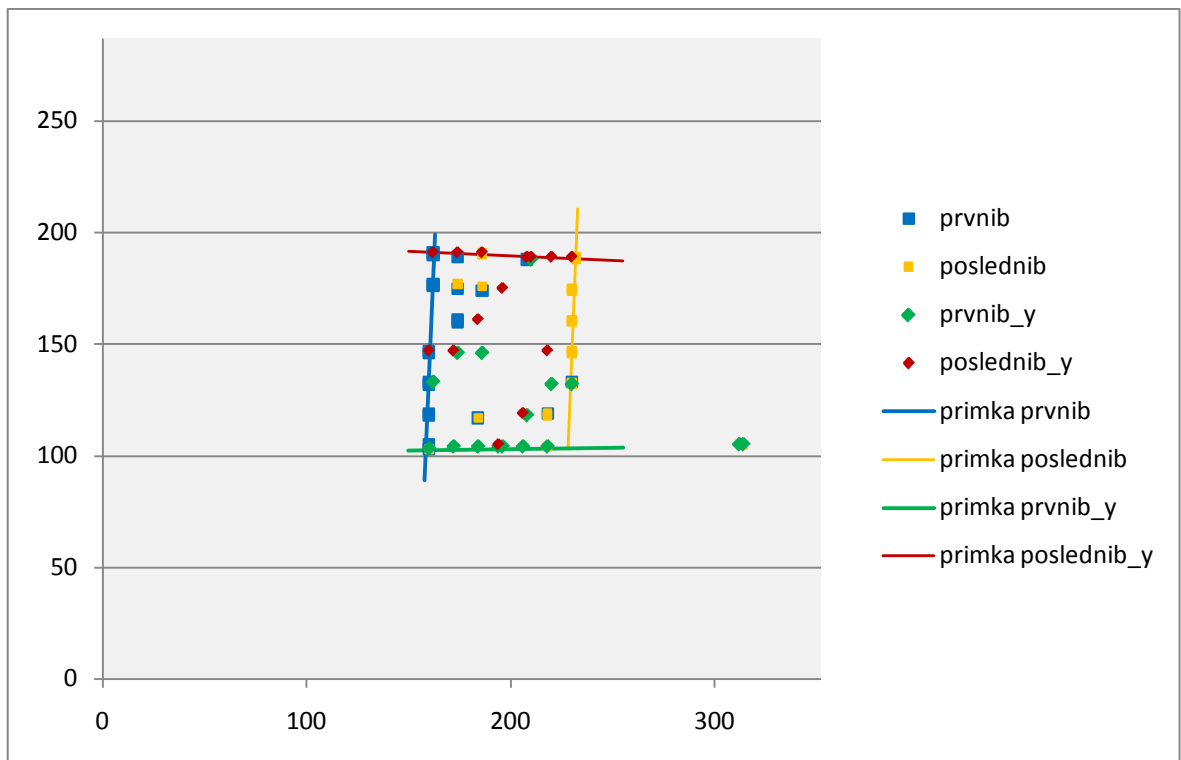
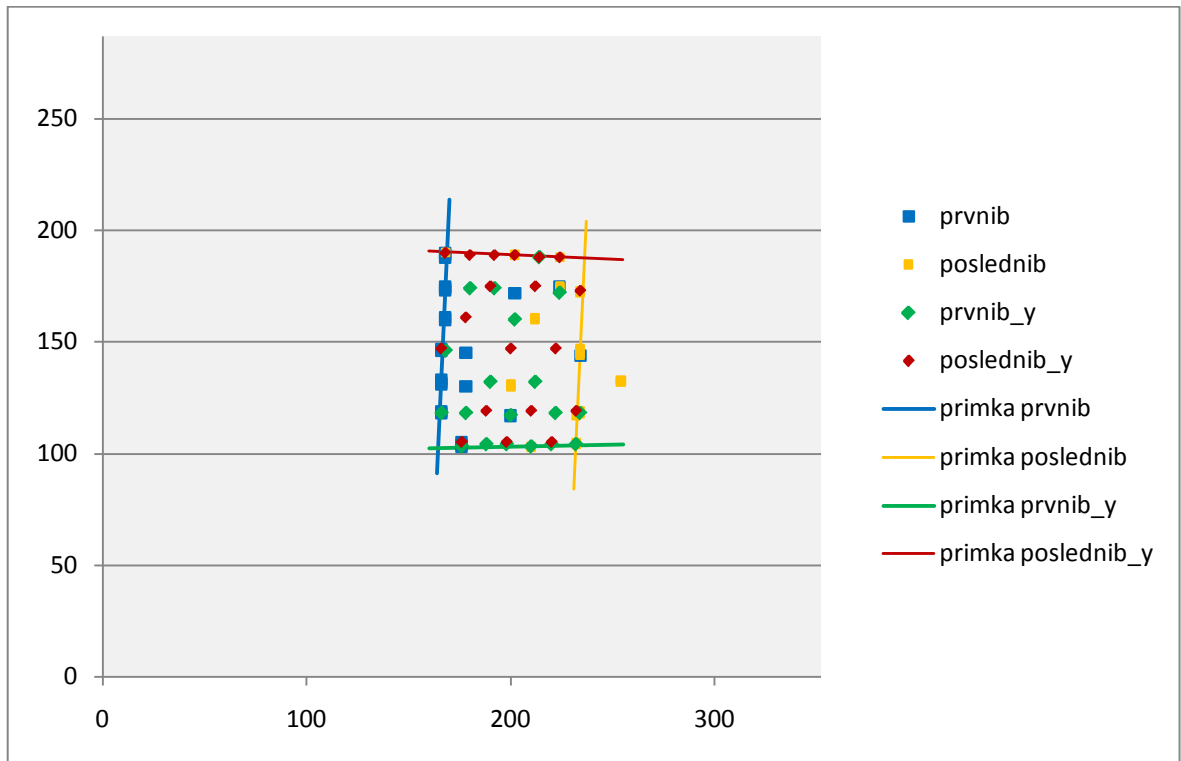
Výsledek odstranění falešných bodů podle četnosti:

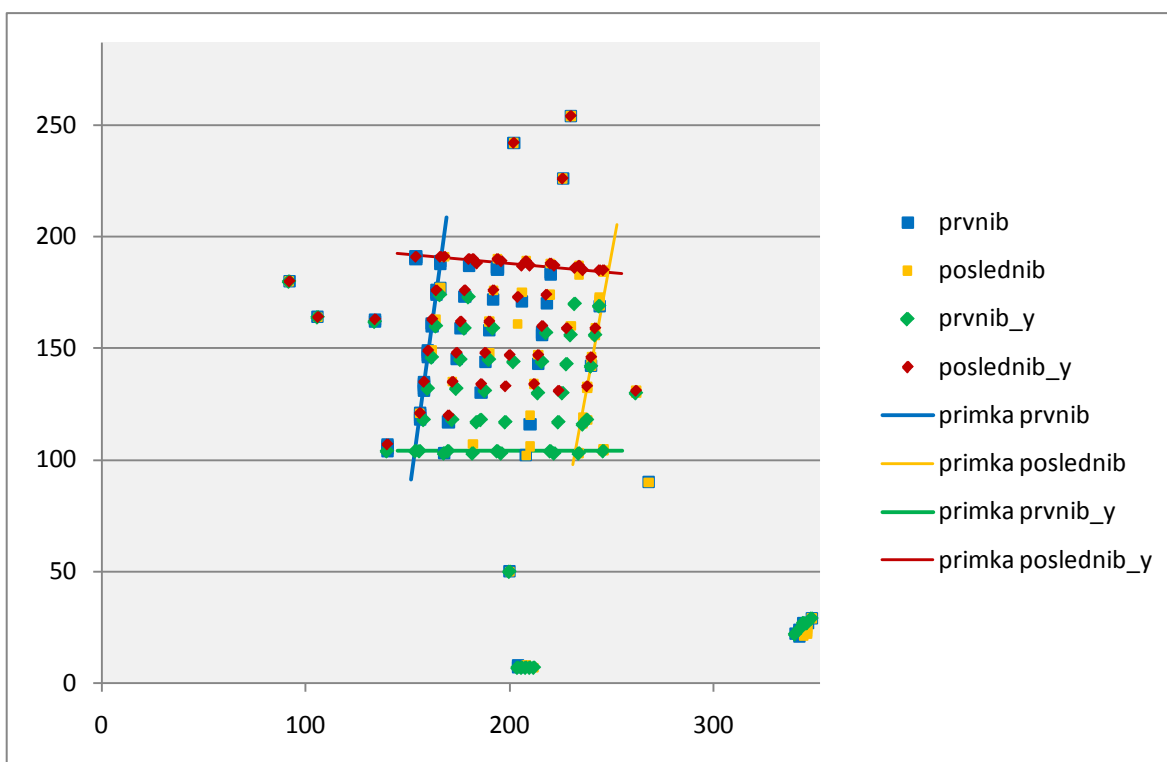
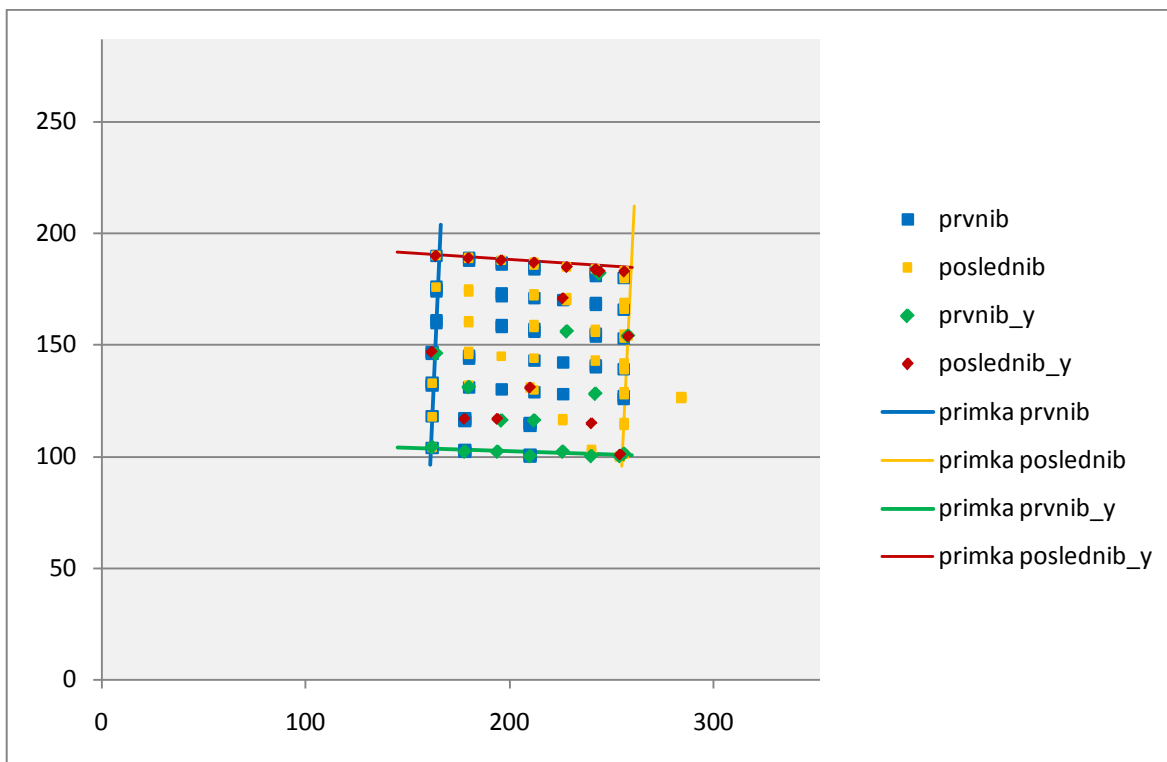


Výsledné přímky vypočítané pomocí funkcí linreg a linreg_inv:



PŘÍLOHA P III: ANALÝZA VÝSLEDNÝCH DAT METODY ZALOŽENÉ NA ALGORITMU RANSAC





PŘÍLOHA P IV: ZDROJOVÝ KÓD VÝSLEDNÉ METODY

Funkce pro výpočet vzdálenosti bodu a přímky:

```
float vzdalenostbp(bod C, primka AB) //vzdalenost bodu o primky
{
    float vzdal,body,bodx,acko,bcko;

    acko=AB.a;
    bcko=AB.b;
    bodx=(float)C.xx;
    body=(float)C.yy;
    vzdal=(absol((acko*bodx)-
body+bcko))/((float)sqrt((acko*acko)+1));

    return vzdal;
}
```

Funkce pro výpočet orientované vzdálenosti bodu a přímky:

```
float orient_vzdalenostbp(bod C, primka AB) //vzdalenost bodu od
primky se znamenkem
{
    float vzdal,body,bodx,acko,bcko;

    acko=AB.a;
    bcko=AB.b;
    bodx=(float)C.xx;
    body=(float)C.yy;
    vzdal=((acko*bodx)-body+bcko)/((float)sqrt((acko*acko)+1));

    return vzdal; //kdyz minusova, tak NAD primkou, kdyz plusova,
tak POD primkou (alespon zatim to tak bylo na papire)
}
```

Funkce pro sestrojení přímky ze dvou bodů:

```
primka spocitej_primku(bod A, bod B)
{
    primka spocitanapr;

    spocitanapr.posl=2;
    spocitanapr.NAD=0;
    spocitanapr.POD=0;

    spocitanapr.vlastni[0].xx=A.xx;
    spocitanapr.vlastni[0].yy=A.yy;

    spocitanapr.vlastni[1].xx=B.xx;
    spocitanapr.vlastni[1].yy=B.yy;

    if (A.xx==B.xx) spocitanapr.a=1000;
    else spocitanapr.a=((float)A.yy-(float)B.yy)/((float)A.xx-
(float)B.xx);
}
```

```

        spocitanapr.b=(float)A.yy-(spocitanapr.a*(float)A.xx);

    return spocitanapr;
}

```

Funkce přiřazení bodu k přímce a její přepočítání pomocí metody nejmenších čtverců:

```

primka prepocitej_primku(bod C, primka AB)
{
    primka prepoc;
    int i;
    float x_avg,y_avg,x_sum,y_sum;

    x_avg=y_avg=x_sum=y_sum=0;

    prepoc=AB;

    for(i=0;i<prepoc.posl;i++)
    {
        if(preproc.vlastni[i].xx==C.xx &&
prepoc.vlastni[i].yy==C.yy)
            {
                return prepoc;
            }
    }

    prepoc.vlastni[prepoc.posl].xx=C.xx;
    prepoc.vlastni[prepoc.posl].yy=C.yy;
    prepoc.posl++;

    for(i=0;i<prepoc.posl;i++)
    {
        x_sum=x_sum+(float)prepoc.vlastni[i].xx;
        y_sum=y_sum+(float)prepoc.vlastni[i].yy;
    }

    x_avg=x_sum/(float)prepoc.posl;
    y_avg=y_sum/(float)prepoc.posl;

    x_sum=0;
    y_sum=0;

    for(i=0;i<prepoc.posl;i++)
    {
        x_sum=x_sum+(((float)prepoc.vlastni[i].xx-
x_avg)*((float)prepoc.vlastni[i].yy-y_avg));
        y_sum=y_sum+(((float)prepoc.vlastni[i].xx-
x_avg)*((float)prepoc.vlastni[i].xx-x_avg));
    }

    if(y_sum==0) prepoc.a=1000; else prepoc.a=x_sum/y_sum;
    prepoc.b=y_avg-(prepoc.a*x_avg);

    return prepoc;
}

```

Funkce určující procento bodů pod a nad přímkou:

```
primka urci_NADPOD(primka AB, bod* poleb, int kolik)
{
    primka vyslednap;
    int i;

    vyslednap=AB;

    vyslednap.NAD=vyslednap.POD=0;

    for(i=0;i<kolik;i++)
    {
        if(orient_vzdalenostbp(poleb[i],vyslednap)>=limit_vzdalenosti
)
            vyslednap.POD++;
        else if(orient_vzdalenostbp(poleb[i],vyslednap)<=(0-
limit_vzdalenosti))
            vyslednap.NAD++;
    }

    vyslednap.NAD=(vyslednap.NAD/(vyslednap.NAD+vyslednap.POD)*10
0); //uprava na procenta
    vyslednap.POD=(vyslednap.POD/(vyslednap.NAD+vyslednap.POD)*10
0);

    return vyslednap;
}
```

Funkce pro přiřazení bodů k přímce z pole bodů:

```
primka prirad_body(bod* poleb, primka AB, int kolik)
{
    primka sbody;
    int i;

    sbody=AB;

    for(i=0;i<kolik;i++)
    {
        if(vzdalenostbp(poleb[i],sbody)<limit_vzdalenosti)
        {
            sbody=prepocitej_primku(poleb[i],sbody);
        }
    }

    return sbody;
}
```

Výsledná funkce metody s použitím algoritmu RANSAC:

```
vysledek ziskejprimkyRANSAC(bod* vstupni, int pocetbodu)
{
    vysledek hotovy_vysledek;
    primka pomocna;
    int i,j,k;
    int vraci,vraci1,vraci2,vraci3,vraci4;

    vraci=vraci1=vraci2=vraci3=vraci4=0;
    hotovy_vysledek.vys1.posl=hotovy_vysledek.vys2.posl=hotovy_vy
sledek.vys3.posl=hotovy_vysledek.vys4.posl=0;

    srand((unsigned)time(NULL));

    for(i=0;i<1000;i++)
    {
        j=rand() % pocetbodu+1;
        k=rand() % pocetbodu+1;
        if(j==k)
        {
            i--;
            continue;
        }
        pomocna=spocitej_primku(vstupni[j],vstupni[k]);
        pomocna=urci_NADPOD(pomocna,vstupni,pocetbodu);
        if(pomocna.NAD>90 || pomocna.POD>90)
        {

            pomocna=prirad_body((bod*)vstupni,pomocna,pocetbodu);

            if(pomocna.posl>3)
            {
                if(pomocna.NAD>90 && pomocna.a<1 &&
pomocna.posl>hotovy_vysledek.vys1.posl)
                {
                    hotovy_vysledek.vys1=pomocna;
                    vraci1=1;
                }
                else if(pomocna.NAD>90 && pomocna.a>4 &&
pomocna.posl>hotovy_vysledek.vys2.posl)
                {
                    hotovy_vysledek.vys2=pomocna;
                    vraci2=1;
                }
                else if(pomocna.POD>90 && pomocna.a<1 &&
pomocna.posl>hotovy_vysledek.vys3.posl)
                {
                    hotovy_vysledek.vys3=pomocna;
                    vraci3=1;
                }

                else if(pomocna.POD>90 && pomocna.a>4 &&
pomocna.posl>hotovy_vysledek.vys4.posl)
```

```

        {
            hotovy_vysledek.vys4=pomocna;
            vraci4=1;
        }
    }}
}
if(vraci1==1 && vraci2==1 && vraci3==1 && vraci4==1)
    vraci=1;

if(vraci==1)
    hotovy_vysledek.navrat=1;
else
    hotovy_vysledek.navrat=0;

return hotovy_vysledek;
}

```

Snímky z inteligentní kamery a jejich následné zpracování:

