

Bakalářská práce



České  
vysoké  
učení technické  
v Praze

**F3**

Fakulta elektrotechnická  
Katedra kybernetiky

# Zpracování obrazu mikrořadiči pro mnohokanálové měření polohy

**Tomáš Novák**

Kybernetika a robotika – obor Robotika

24.5.2013

Vedoucí práce: doc. Ing. Jan Fischer, CSc. (Katedra měření)



České vysoké učení technické v Praze  
Fakulta elektrotechnická

Katedra kybernetiky

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

**Student:** Tomáš Novák  
**Studijní program:** Kybernetika a robotika (bakalářský)  
**Obor:** Robotika  
**Název tématu:** Zpracování obrazu mikrořadiči pro mnohokanálové měření polohy

### Pokyny pro vypracování:

1. Navrhněte metody zpracování obrazu pro mnohokanálové synchronní měření polohy pomocí obrazových senzorů, které vystačí s omezenou snímkovou pamětí a které bude možno implementovat do mikrořadičů STM32F4xx.
2. Vytvořte potřebné programové vybavení pro mikrořadiče STM32F4XX i pro nadřazené PC, které s využitím skupiny kamerových senzorů zapojených do sítě umožní synchronní mnohokanálové měření polohy ve 2D po zvolených liniích obrazu a dále určení polohy kontrastní stopy ve 3D.

### Seznam odborné literatury:

- [1] ARM Limited: Cortex-M4 Revision r0p1, Technical Reference Manual, ARM DDI 0439C
- [2] STMicroelectronics: Reference Manual, RM0090, Doc ID 018909 Rev3
- [3] Yiu J.: The definitive Guide to the ARM Cortex- M3.Elsevier, 2007

**Vedoucí bakalářské práce:** doc. Ing. Jan Fischer, CSc.

**Platnost zadání:** do konce zimního semestru 2013/2014

  
prof. Ing. Vladimír Mařík, DrSc.  
vedoucí katedry



  
prof. Ing. Pavel Ripka, CSc.  
děkan

V Praze dne 10. 1. 2013



## Poděkování / Prohlášení

Rád bych poděkoval svým rodičům za podporu v průběhu studia a psaní bakalářské práce. Dále bych chtěl poděkovat svému vedoucímu, doc. Ing. Janu Fischerovi, CSc. za podporu a čas, který mi věnoval.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Tato práce vznikla v laboratoři videometrie katedry měření ČVUT – FEL v Praze pod vedením doc. Ing. Jana Fischera, CSc. Navazuje též na výzkum v rámci MSM6840770015 – „Výzkum metod a systémů pro měření fyzikálních veličin a zpracování naměřených dat“ a další práce řešené v laboratoři videometrie, jejichž některé poznatky a výstupy v oblasti aplikace optoelektronických senzorů také využívá.

V Praze dne 24. 5. 2013

*Tomáš Novák*  
.....



## Abstrakt / Abstract

Tato práce se zabývá systémem synchronizovaných obrazových senzorů využívajících rozhraní Ethernet pro měření polohy. Jsou v ní navrženy metody pro rychlé a paměťově nenáročné určení polohy těžiště objektu v kontrastním obraze, je navržena metoda určení polohy v prostoru z měření polohy těžiště v obraze více kamerami a je navržen systém pro kalibraci polohy kamer usnadňující instalaci systému.

Celý systém využívá mikrokontroléry řady STM32F4 a obrazových senzorů MT9V034. Software pro mikrokontroléry je napsán v jazyce C, nadstavba a demonstrační grafické rozhraní je napsáno pro Matlab. Chování systému je demonstrováno na pokusech. Systém je schopen přesně určovat polohu kontrastního objektu v prostoru a provádět synchronní měření po liniích.

**Klíčová slova:** měření polohy; zpracování obrazu; mikrořadič

System for position measurement using image sensors synchronized using Ethernet is shown in this work. Memory undemanding methods for object center calculation are proposed. Determining object position in space from camera measurements is described. Calibration algorithm for camera global position is proposed to make installation easier.

Whole system is based on microcontrollers, family line STM32F4, and image CMOS sensors MT9V034. Software for microcontrollers is written in C, measurement coordination and space position calculation is written for Matlab. Behavior of system is demonstrated on experiments. System is capable of precise measurement of position in space. Synchronous measurement on selected lines is also implemented.

**Keywords:** position measurement; image processing; microcontroller

**Title translation:** Image processing using microcontrollers for multichannel position measurement

# Obsah /

<b>1 Úvod</b> .....	1
<b>2 Úkol práce</b> .....	2
2.1 Požadavky na zpracování obrazu .....	2
2.2 Požadavky na nadstavbu pro prostorové měření .....	2
2.2.1 Synchronizace modulů .....	2
2.2.2 Koordinace měření .....	4
2.2.3 Výpočet polohy .....	4
<b>3 Rozbor situace</b> .....	5
3.1 Použitý hardware .....	5
3.1.1 Procesorová deska .....	5
3.1.2 Deska CMOS senzoru .....	7
3.1.3 Programové vybavení k hardwarovému řešení .....	7
3.2 Možnosti odlišení objektu v obraze .....	7
3.3 Možnosti určení polohy objektu v obraze .....	8
3.4 3D měření s použitím kamerových senzorů .....	8
3.4.1 Způsob řešení přeurčeného systému .....	8
<b>4 Použité přístupy</b> .....	9
4.1 Zpracování obrazu na MCU .....	9
4.1.1 Rozlišení objektu v obraze .....	9
4.1.2 Určení těžiště pomocí aritmetické řady .....	9
4.1.3 Určení těžiště s pomocí integrálního obrazu .....	10
4.2 Určení polohy objektu v prostoru z měření těžiště v jednotlivých obrazech .....	11
4.2.1 Souřadnicové systémy použité pro výpočet polohy .....	12
4.2.2 Zobrazení objektu objektivem na senzor .....	13
4.2.3 Vztah globálních a kamerových souřadnicových systémů .....	15
4.2.4 Řešení přeurčené soustavy rovnic pro polohu .....	16
4.3 Kalibrace polohy obrazových senzorů .....	16
4.4 Teoretická dosažitelná přesnost měření .....	17
<b>5 Implementace</b> .....	18
5.1 Programové vybavení pro řídicí MCU .....	18
5.1.1 Firmware pro přístup k modulu přes USB .....	18
5.1.2 Firmware pro komunikaci přes Ethernet .....	20
5.2 Skripty a funkce pro Matlab ..	21
5.2.1 Přístup k externímu hardwaru .....	21
5.2.2 Knihovna pro ovládní polohovací jednotky MARS .....	21
5.2.3 Skupina skriptů pro testování zpracování obrazu .....	22
5.2.4 Skripty pro 3D výpočet polohy .....	22
<b>6 Dosažené výsledky</b> .....	23
6.1 Chování algoritmů určení těžiště v obraze .....	23
6.1.1 Test opakovatelnosti měření .....	24
6.1.2 Určení rozlišovací schopnosti .....	25
6.1.3 Chyba geometrického zobrazení .....	27
6.2 Prostorové měření ve 3D .....	27
6.2.1 Ověření funkčnosti kalibrace polohy .....	30
6.2.2 Určování polohy pohybující se LED .....	30
6.2.3 Vliv měření jednotlivými kamerami v přeurčeném systému .....	31
6.2.4 Experimentální určení rozlišovací schopnosti 3D měření .....	32
6.2.5 Nejistoty při prostorovém měření .....	33
6.3 Měření po liniích ve 2D .....	34
6.3.1 Určení nulové polohy .....	35



6.3.2	Měření odchylek od základní polohy při malé rychlosti .....	35
6.3.3	Měření odchylek od základní polohy při velké rychlosti .....	35
6.3.4	Ověření synchronizace kamer.....	37
<b>7</b>	<b>Závěr .....</b>	<b>38</b>
7.1	Určení těžiště v obraze .....	38
7.2	Systém pro 3D měření polohy .	39
7.3	Systém pro 2D měření polohy po liniích .....	39
7.4	Zhodnocení práce .....	40
	<b>Literatura .....</b>	<b>41</b>
<b>A</b>	<b>Použité zkratky.....</b>	<b>43</b>
<b>B</b>	<b>Oživení hardwaru .....</b>	<b>44</b>
B.1	Osazení hardwaru.....	44
B.1.1	Procesorová deska .....	44
B.1.2	Deska CMOS senzoru ...	44
B.2	Kontrola funkčnosti USB .....	44
B.3	Kontrola funkčnosti CMOS senzoru .....	45
B.4	Kontrola funkčnosti Ethernetu.....	45
<b>C</b>	<b>Opakování pokusů k testování algoritmů.....</b>	<b>46</b>
C.1	Test opakovatelnosti měření ...	46
C.2	Test rozlišovací schopnosti.....	46
C.3	Chyba geometrického zobrazení .....	46
<b>D</b>	<b>Opakování pokusů k testování 3D měření polohy .....</b>	<b>47</b>
D.1	Kalibrace systému .....	47
D.2	Měření polohy .....	47
D.3	Měření rozlišovací schopnosti..	48
<b>E</b>	<b>Opakování pokusů k testování 2D měření po liniích .....</b>	<b>49</b>
<b>F</b>	<b>Knihovna pro ovládání jednotky MARS pro řízení motorů .....</b>	<b>50</b>
<b>G</b>	<b>Obsah přiloženého DVD .....</b>	<b>51</b>
G.1	Firmware .....	51
G.2	Matlab.....	51

## Obrázky /

<b>2.1.</b>	Schema 3D měření polohy .....	3
<b>2.2.</b>	Schema 2D měření polohy .....	3
<b>3.1.</b>	Procesorová deska .....	6
<b>3.2.</b>	Senzorová deska .....	6
<b>3.3.</b>	Test synchronizace přes PTP ....	7
<b>4.1.</b>	Výpočet těžiště pomocí aritmické řady .....	10
<b>4.2.</b>	Úkol prostorového měření .....	11
<b>4.3.</b>	Globální a kamerové souřadnice .....	12
<b>4.4.</b>	Obrazové souřadnice .....	13
<b>4.5.</b>	Zobrazení čočkou .....	14
<b>4.6.</b>	Zjednodušené zobrazení čočkou .....	14
<b>5.1.</b>	Hlavní vlákno progra- mu přes USB .....	19
<b>5.2.</b>	Hlavní vlákno progra- mu přes Ethernet .....	20
<b>6.2.</b>	Redukované rozlišení obrazu a výřez z plného rozlišení .....	23
<b>6.1.</b>	Sestava pokusu pro test algoritmů .....	24
<b>6.3.</b>	Opakovatelnost měření těžiště .....	25
<b>6.4.</b>	Opakovatelnost měření těžiště ve výhodné poloze .....	26
<b>6.5.</b>	Test přesnosti měření těžiště v obraze .....	26
<b>6.6.</b>	Chyby linearit výpo- čtu těžiště .....	27
<b>6.7.</b>	Geometrické chyby zob- razení objektivem .....	28
<b>6.8.</b>	3D měření polohy objektu .....	28
<b>6.9.</b>	Přípravek pro kalibraci kamer .....	29
<b>6.10.</b>	Průběh kalibrace polo- hy kamer .....	30
<b>6.11.</b>	Výsledek 3D měření .....	31
<b>6.12.</b>	Chyby přeúčtenosti systému ...	32
<b>6.13.</b>	Chyba kalibrace obje- vená přeúčteností systému .....	33
<b>6.14.</b>	Chyba určení polohy v prostoru při lineárním pohybu .....	33
<b>6.15.</b>	Nejistoty při lineárním pohybu v 3D měření .....	34
<b>6.16.</b>	Pokus pro 2D měření po liniích .....	34
<b>6.17.</b>	Obrázky tyče v klidu .....	35
<b>6.18.</b>	Naměřené polohy při pomalém pohybu přípravku ...	36
<b>6.19.</b>	Naměřené polohy při rychlém pohybu přípravku .....	36
<b>6.20.</b>	Svícení LED při syn- chronizačním pokusu .....	37

# Kapitola 1

## Úvod

Cílem této práce je vytvořit systém obrazových senzorů pro mnohokanálové měření polohy v prostoru a synchronizované 2D měření polohy po liniích. Příkladem využití měření polohy v prostoru může být kromě určování polohy objektu sledování deformací, např. průhybu mostu, nebo detekce nerovností na ploše s použitím rozmiřaného laseru. Aplikací pro synchronizované měření po liniích může být např. detekce synchronnosti funkce rychlých strojů nebo sledování kmitání výrobní linky.

Tato práce se bude zabývat návrhem algoritmů pro zpracování obrazu, spojením senzorů do sítě a zpracováním naměřených dat pro určování polohy.

Zpracování obrazu bude probíhat na mikrokontroléru řady STM32F4 s jádrem ARM Cortex-M4 a jeho cílem bude určit polohu kontrastního objektu. Mikrokontrolér má omezenou velikost paměti SRAM, což musí použité algoritmy reflektovat.

Moduly budou synchronizovány, k tomu se předpokládá použití Ethernetu a PTP (IEEE 1588). Díky tomu bude systém schopen určovat polohu nejen statických, ale i pohybujících se objektů.

Pro výpočet polohy budou použity alespoň 2 kamery, které budou umístěny v obecné poloze v prostoru. Použitý způsob výpočtu polohy bude tento požadavek respektovat.

Systém bude postaven na hardwaru navrženém v rámci dřívější práce [4]. Tento hardware je uzpůsoben k připojení obrazového senzoru a komunikaci přes rozhraní Ethernet. Pro účely testování umožňuje i připojení k počítači přes rozhraní USB.

Dosažené výsledky budou demonstrovány praktickými pokusy – pro 3D měření polohy je důležitá schopnost systému měřit ve všech směrech, pro 2D měření polohy po liniích bude důležité, zda nedochází k desynchronizaci kamer a následným chybám tímto způsobeným.

# Kapitola 2

## Úkol práce

Tato práce si klade za cíl vytvořit systém pro mnohokanálové distribuované synchronní měření polohy objektu s použitím obrazových senzorů. Měření polohy může mít dvojí podobu:

- 3D měření polohy
- 2D měření po liniích

Příklad 3D měření je vyobrazen na Obrázku 2.1, příklad pro 2D měření na Obrázku 2.2. Cílem 3D měření může být např. určování polohy pohybujícího se robota nebo určení průhybu mostu z pozorování význačného bodu, cílem 2D měření může být např. sledování kmitání výrobní linky.

Vytvoření systému se bude skládat ze dvou úkolů:

- navržení algoritmů pro zpracování obrazu
- vytvoření nadstavby pro synchronní měření a výpočet polohy

**Zpracování obrazu** bude probíhat na mikrokontroléru, kvůli tomu musí být použité algoritmy nenáročné na paměť a procesorový čas. Algoritmy zpracování obrazu mají za úkol určit polohu kontrastního objektu v obraze a musí si vystačit s omezenou pamětí.

**Nadstavba** pro měření a výpočet polohy bude mít za úkol zajistit synchronizaci jednotlivých senzorů, komunikaci mezi nimi, koordinaci měření a výpočty pro nepřímé měření polohy.

## 2.1 Požadavky na zpracování obrazu

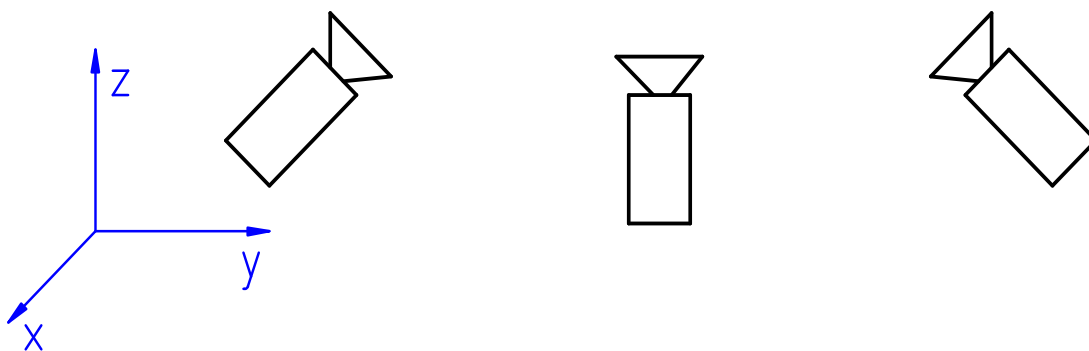
Zpracování obrazu bude probíhat na mikrokontrolérech rodiny STM32F4 bez použití externí paměti, metody tedy musí být nenáročné na paměť. Tento požadavek se zakládá na skutečnosti, že existují mikrokontroléry a signálové procesory, které neumožňují připojení externí paměti a pro funkci si tak musejí vystačit s vnitřní pamětí.

Hledaný objekt bude kontrastní, proto spíše než univerzálnost algoritmu bude kritériem výběru rychlost a přesnost, resp. dosažitelné rozlišení měření. Očekává se, že přesnost určení polohy objektu v obraze bude subpixelová.

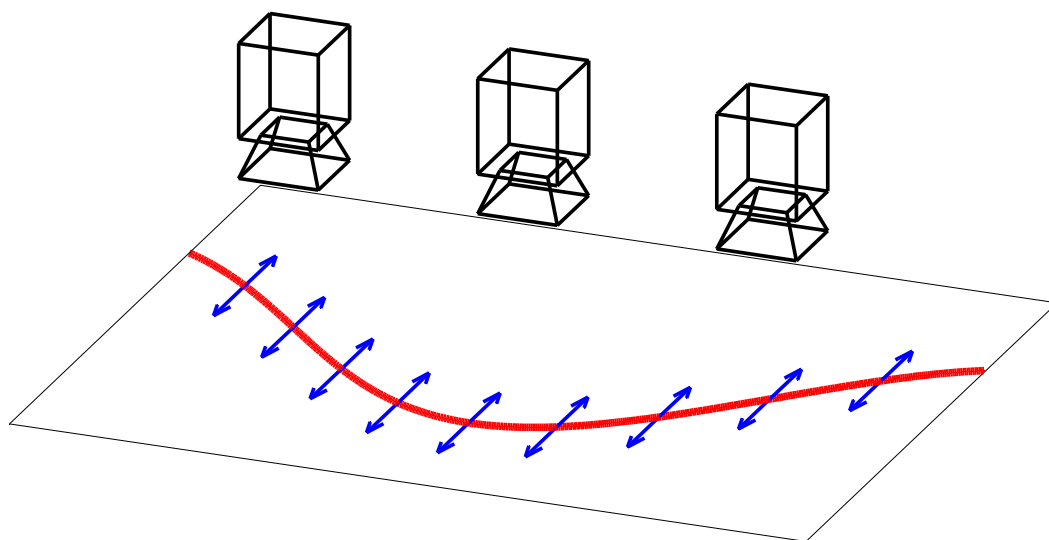
## 2.2 Požadavky na nadstavbu pro prostorové měření

### 2.2.1 Synchronizace modulů

Časová synchronizace jednotlivých senzorů musí být tak přesná, aby pohybující se objekt byl zaznamenán všemi senzory ve stejném okamžiku, tedy ve stejném místě. Vzhledem k použité délce expozice senzoru  $t_e$  musí být přesnost synchronizace  $t_s$  výrazně lepší, tedy  $t_s \ll t_e$ . Pro synchronizaci se předpokládá použití protokolu IEEE 1588 (dále odkazován jako PTP) přes rozhraní Ethernet.



Obrázek 2.1. Schematické znázornění 3D měření polohy



Obrázek 2.2. Schematické znázornění 2D měření polohy po liniích

### ■ 2.2.2 Koordinace měření

Pro zajištění funkčnosti systému bude nutné, aby se systém choval deterministicky. Kvůli výpočtu polohy bude třeba časová identifikace jednotlivých snímků (měření).

### ■ 2.2.3 Výpočet polohy

Pro určování polohy se bude používat 2 a více kamer, výpočet polohy proto bude muset řešit přeürčenost systému. Pro umožnění snadné instalace se předpokládá obecné umístění kamer v prostoru.

# Kapitola 3

## Rozbor situace

V této práci bude navrhován systém kamerových senzorů pro mnohokanálové měření polohy. Vytváření tohoto měřicího systému se skládá z následujících podproblémů:

1. návrh a realizace hardware
2. časová synchronizace
3. zpracování obrazu pro rozlišení objektů a určení jejich polohy
4. výpočty pro 3D měření

Tato práce využije hardware navržený v rámci dřívější diplomové práce [4]. Na tomto hardwaru je již nainstalováno lwIP pro komunikaci přes TCP/UDP a PTPd pro synchronizaci pomocí PTP. Mým úkolem bude desky osadit a ověřit jejich funkčnost.

Dále bude třeba navrhnout a implementovat algoritmy pro zpracování obrazu a vytvořit nadstavbu pro 3D měření polohy. Pro výběr algoritmů zpracování obrazu bude určující především výpočetní náročnost a jejich přesnost, výpočet polohy bude muset brát v potaz obecné uspořádání kamer a přeuročeness systému.

### 3.1 Použitý hardware

Hardware se skládá ze dvou desek – senzorové desky s CMOS senzorem a procesorové desky s MCU. Použitý mikrokontrolér je STM32F407, použitý obrazový senzor je MT9V034.

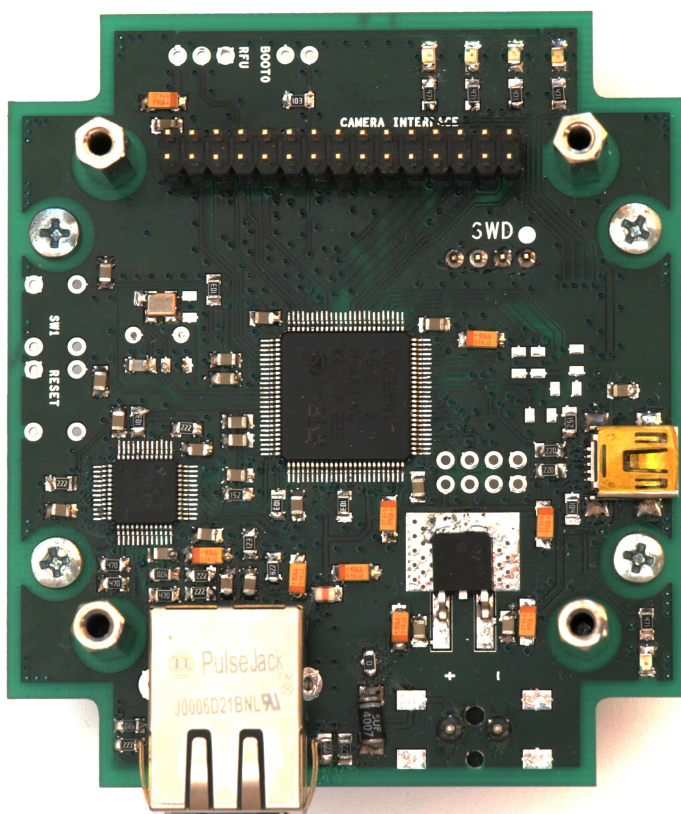
#### 3.1.1 Procesorová deska

Procesorová deska umožňuje:

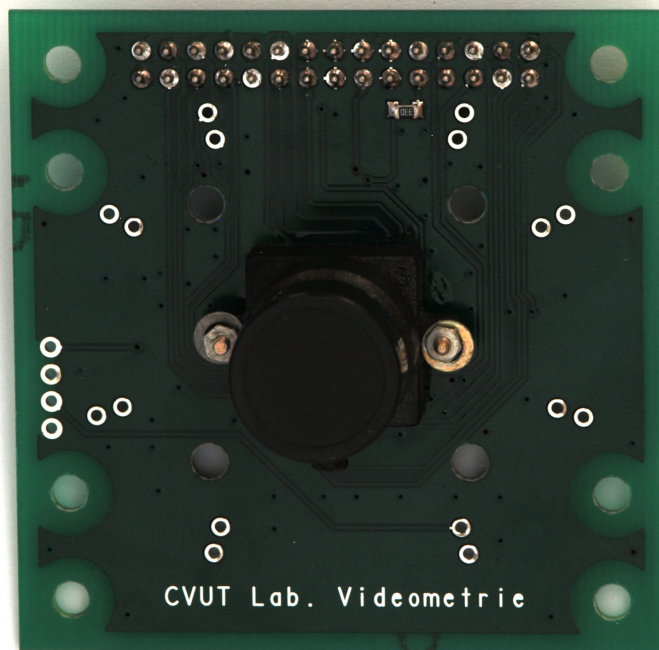
- připojení na Ethernet
- připojení kamery přes DCMI rozhraní
- připojení USB v režimu full-speed
- 4 indikační LED diody
- připojení SD karty

Deska je založena na mikrokontroléru STM32F407, je ale hardwarově kompatibilní s procesory řady STM32F4 a STM32F2. Kvůli využití Ethernetu je nutné jako zdroj hodin použít externí oscilátor 50 MHz<sup>1)</sup>. Osazená procesorová deska je vidět na Obrázku 3.1.

<sup>1)</sup> Generování hodin pro fyzickou vrstvu přes vnitřní PLL nelze kvůli jitteru. Komunikace přes Ethernet pak nefunguje.



Obrázek 3.1. Procesorová deska (pohled shora)



Obrázek 3.2. Senzorová deska s objektivem (pohled shora)



### 3.1.2 Deska CMOS senzoru

Deska CMOS senzoru zajišťuje především:

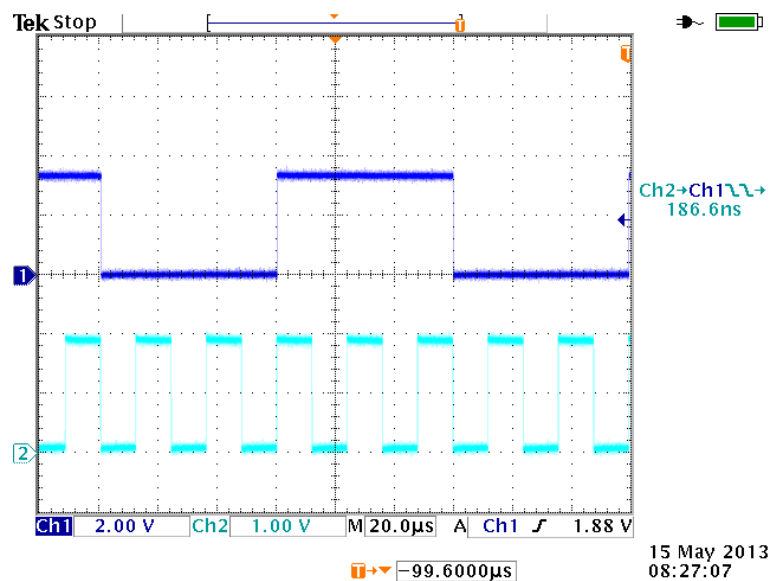
- možnost připevnění objektivu
- vyvedení DCMI na konektor
- nastavení adresy pro I2C ovládací sběrnici

Deska obsahuje senzor MT9V034 a k němu potřebné pasivní součástky. Umožňuje sice připojení LED pro blesk, toho ale aplikace využívat nebude. Osazená deska je vyobrazena na Obrázku 3.2.

### 3.1.3 Programové vybavení k hardwarovému řešení

K použitému hardwaru je vytvořen software, který obsahuje instalaci lwIP a PTPd. Pro ověření funkčnosti synchronizace jsem nechal jednu desku generovat signál 10 kHz a druhou 50kHz. Oba tyto signály byly generovány s využitím časových značek uvnitř procesoru a měly by mít jednu hranu na celé sekundě.

Pokud by časy obou modulů nebyly shodné, projevil by se to vzájemným posuvem signálů. Pokud by moduly měly jinak rychle běžící hodiny pro určení času, projevil by se to vzájemným posouváním signálů vůči sobě. Oba tyto jevy bylo možné pozorovat po startu systému, než došlo ke stabilizaci časů protokolem PTP. Průběhy po ustálení jsou zobrazeny na Obrázku 3.3. Ve skutečnosti naměřené frekvence byly o něco nižší, ale signály zůstávají synchronizované, což je požadovaná vlastnost.



Obrázek 3.3. Test synchronizace přes PTP

## 3.2 Možnosti odlišení objektu v obraze

Pro určení polohy objektu je nejprve nutné odlišit objekt od pozadí. Jelikož objekty budou kontrastní, nabízejí se v zásadě dvě možnosti:

- prahování
- hledání hran

Hledání hran v obraze funguje lépe u obrazů, kde může docházet k pozvolným změnám jasu v pozadí a u obrazů s nestálým nebo nedefinovaným osvětlením.

Prahování je výrazně jednodušší a méně náročné na výpočetní čas, lze předpokládat, že jeho nečnosti se neprojeví kvůli kontrastnímu charakteru scény.

### 3.3 Možnosti určení polohy objektu v obraze

Pro měření polohy objektu je nejvhodnější určení jeho obrazového těžiště. Můžeme totiž předpokládat, že obrazové těžiště bude rozumným odhadem polohy hmotného středu objektu.

Výpočet těžiště podle definice je výpočetně náročný, proto je výhodné využít algoritmů, které zlepšují výpočetní náročnost.

Pro zjednodušení lze použít např. tyto dva přístupy:

- výpočet s využitím aritmetických řad
- výpočet s využitím integrálního obrazu

Obě tyto metody snižují nutný počet násobení. První vychází ze vzorce pro výpočet součtu aritmetické řady a lze ji využít v binárním obraze nebo při zpracování dat z analogové kamery s využitím komparátoru.

Metoda využívající integrálního obrazu rozděluje výpočet těžiště na rychlé předzpracování, kdy se spočte integrální obraz, a následný výpočet těžiště z posledního řádku a sloupce integrálního obrazu.

### 3.4 3D měření s použitím kamerových senzorů

Prostorové měření kamerami v závislosti na požadavcích na jejich umístění lze rozdělit na dva druhy:

- stereovize
- obecné uspořádání kamer

**Stereovize** předpokládá umístění kamer paralelně vedle sebe. Stereovize je jednoduchá, ale pro mnohokanálové měření polohy nevhodná, kvůli příliš svazujícím požadavkům na umístění kamer a jejich počet.

**Obecné uspořádání** kamer předpokládá, že kamery budou umístěny libovolně v prostoru. Výpočet polohy proto bude složitější, ale umožní současně i kalibraci polohy kamer pro zvýšení přesnosti měření.

#### 3.4.1 Způsob řešení přeúřčeného systému

Měřením těžiště objektu z  $n$  kamer změříme celkem  $2n$  údajů. Těleso v prostoru má polohu určenou třemi souřadnicemi, při použití dvou a více kamer bude  $2n > 3$ , což znamená, že měřicí systém bude přeúřčený.

Přeúřčenost systému lze řešit např. metodou nejmenších čtverců, která bude použita v této práci.

# Kapitola 4

## Použité přístupy

Při řešení této práce je třeba se zabývat dvěma hlavními částmi:

1. nalezení kontrastního objektu v obraze
2. zpracování pro mnohokanálové měření polohy

Pro algoritmy nalezení kontrastního bodu a určení jeho polohy v obraze bylo třeba brát v potaz omezení plynoucí z použitého hardwaru, zejm. nároky na složitost. Druhým faktorem byla přesnost určení polohy, kterou daný algoritmus umožní.

Pro zpracování polohy jsem se snažil, aby výpočty nebyly závislé na umístění kamer a aby vše bylo co nejrobustnější. Očekává se, že se použije více než dvou kamer, takže použité řešení musí reflektovat přeuročenost systému.

### 4.1 Zpracování obrazu na MCU

#### 4.1.1 Rozlišení objektu v obraze

Pro rozlišení objektu v obraze bylo zvoleno prahování. Práce očekává, že se v obraze nachází jen jeden objekt a ten je velmi kontrastní, proto je prahování dostatečné. Kvůli přesnosti následujících výpočtů ale bylo vhodné vytvořit dvě možnosti prahování, jejichž výstupem byl buď *binární* obraz, nebo obraz ve *stupních šedi*.

Je-li  $p(i, j)$  vstupní obraz, *binární* obraz  $b(i, j)$  bude určen podle vzorce:

$$b(i, j) = \begin{cases} 0 & \text{pro } p(i, j) < t \\ 1 & \text{jinak} \end{cases} \quad (1)$$

kde  $t$  je práh. Pro obraz ve *stupních šedi*  $g(i, j)$  bude platit:

$$g(i, j) = \begin{cases} 0 & \text{pro } p(i, j) < t \\ p(i, j) & \text{jinak} \end{cases} \quad (2)$$

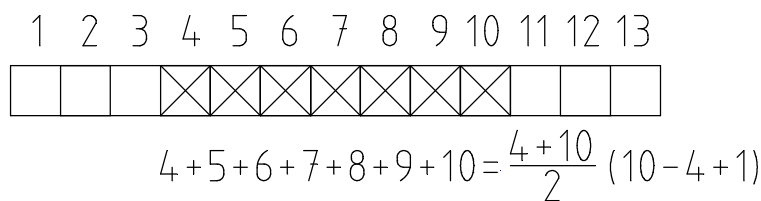
#### 4.1.2 Určení těžiště pomocí aritmetické řady

Výpočet těžiště s využitím aritmetických řad předpokládá využití binárního obrazu. Vzorec pro výpočet těžiště podle definice je:

$$T_i = \frac{\sum_j \sum_i i \cdot b(i, j)}{\sum_j \sum_i b(i, j)} \quad (3)$$

V případě binárního obrazu půjde vyjádřit část výpočtu jako součet aritmetické řady:

$$\sum_i i \cdot b(i, j) = \frac{i_{max} + i_{min}}{2} (i_{max} - i_{min} + 1) \quad (4)$$



$$4 + 5 + 6 + 7 + 8 + 9 + 10 = \frac{4 + 10}{2} (10 - 4 + 1)$$

**Obrázek 4.1.** Výpočet těžiště pomocí aritmetické řady

kde  $i_{min}$  je index na řádku  $j$ , kde je poprvé  $b(i, j) = 1$  a  $i_{max}$  je index, kde je naposled  $b(i, j) = 1$ , tedy vzestupná a sestupná hrana objektu. Tohoto zjednodušení lze využít při výpočtu těžiště v řádcích. Tento postup je znázorněn na Obrázku 4.1.

Při výpočtu těžiště ve sloupcích, tedy:

$$T_j = \frac{\sum_j \sum_i j \cdot b(i, j)}{\sum_j \sum_i b(i, j)} \quad (5)$$

půjde přejít k vyjádření čitatele jako:

$$\sum_j j \cdot (i_{max} - i_{min} + 1) \quad (6)$$

kde již ale nedochází k významnému zjednodušení v počtu násobení. Hlavní výhoda této metody je při zpracování těžiště v řádkovém směru obrazu, proto bude použita pro 2D měření po liniích, kde se bude měřit těžiště ve vybraných řádcích.

### ■ 4.1.3 Určení těžiště s pomocí integrálního obrazu

Je-li  $p(i, j)$  obraz, pak integrální obraz  $I$  obrazu  $p$  je  $I(i, j)$ , kde:

$$I(i, j) = \sum_{k=1}^i \sum_{l=1}^j p(k, l) \quad (7)$$

Pro výpočet na mikrokontroléru bude výhodnější použít rekurentní vzorec:

$$I(i, j) = I(i - 1, j) + I(i, j - 1) - I(i - 1, j - 1) + p(i, j) \quad (8)$$

Předpokládejme, že v obraze se nachází právě jeden objekt a těžiště obrazu je shodné s těžištěm objektu, pak těžiště ve směru osy  $i$  bude:

$$T_i = \frac{\sum_j \sum_i i \cdot p(i, j)}{\sum_j \sum_i p(i, j)} = \frac{\sum_i i \cdot \sum_j p(i, j)}{\sum_j \sum_i p(i, j)} \quad (9)$$

kde  $\sum_j p(i, j)$  můžeme získat z integrálního obrazu:

$$\sum_j p(i, j) = I(i, j_{max}) - I(i - 1, j_{max}) \quad (10)$$

zde  $j_{max}$  je maximální přípustná hodnota parametru  $j$ . Podobně můžeme vyjádřit i  $\sum_j \sum_i p(i, j)$ :

$$\sum_j \sum_i p(i, j) = I(i_{max}, j_{max}) \quad (11)$$

kde  $i_{max}$  je maximální přípustná hodnota parametru  $i$ .

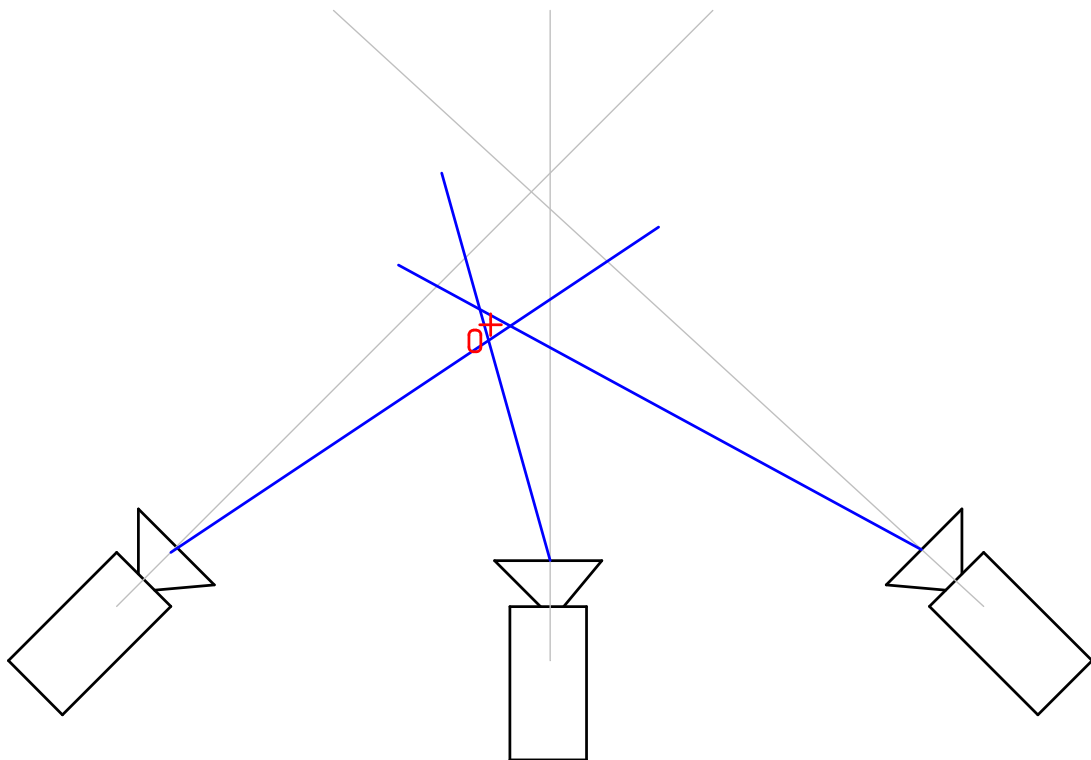
Význam tohoto zjednodušení je, že nejprve vypočteme součty jednotlivých řádků a sloupců a teprve poté provedeme násobení.

Vytvoření integrálního obrazu by mělo být rychlé. K výpočtu těžiště je třeba jen posledního sloupce a posledního řádku integrálního obrazu. Díky tomu je paměťová náročnost výpočtu těžiště pomocí integrálního obrazu pouze dva řádky obrazu.

Výhodou oproti určení těžiště pomocí aritmetické řady je, že tento způsob je možné provést i s ne-binárním obrazem. Díky tomu lze dosáhnout vyšší přesnosti.

## 4.2 Určení polohy objektu v prostoru z měření těžiště v jednotlivých obrazech

Při měření polohy objektu vidí každý senzor objekt ve svém obraze. Tím jsou určeny dvě roviny, na kterých daný objekt leží. Průnik těchto dvou rovin tvoří přímku, která je znázorněna na Obrázku 4.2. Na obrázku jsou šedou barvou zobrazeny optické osy kamer, modrou určené přímky a červenou barvou bod  $O$ , což je měřený objekt.



Obrázek 4.2. Úkol prostorového měření

Určení polohy objektu v prostoru se skládá z následujících částí:

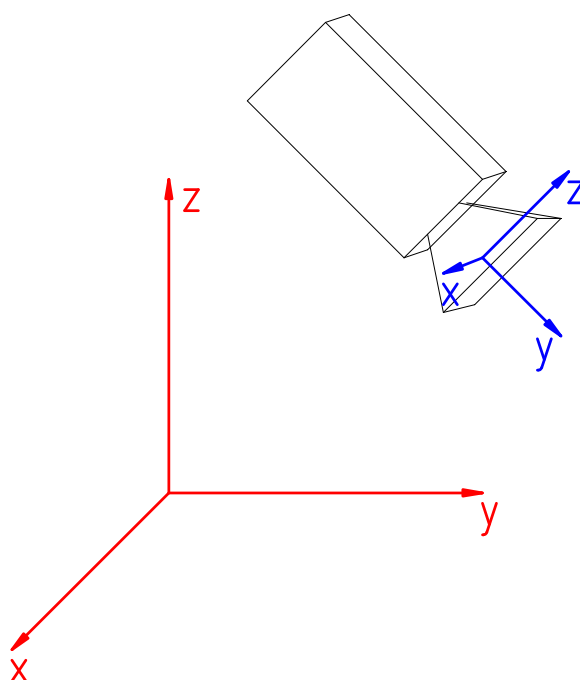
1. popis přímky v souřadnicích kamery (zobrazení objektivem na senzor)
2. převod do globálních souřadnic
3. řešení přeurtčené soustavy rovnic

### 4.2.1 Souřadnicové systémy použité pro výpočet polohy

Pro vyjádření jednotlivých rovnic je vhodné kvůli složitosti vztahů zavést několik souřadných systémů.

**Globální souřadnice** jsou společné pro všechny kamery a budou je značit  $(x, y, z)$ . Měření polohy budou probíhat v globálních souřadnicích.

**Kamerové souřadnice** jsou pevně spojeny s konkrétní kamerou, budou značeny  $(x_k, y_k, z_k)$ . Vůči kameře budou umístěny tak, že osa  $y_k$  bude spojena s optickou osou kamery a počátek bude v hlavním bodě objektivu (viz. bod H na Obrázku 4.5). Vztah globálních a kamerových souřadnic je zobrazen na Obrázku 4.3 .



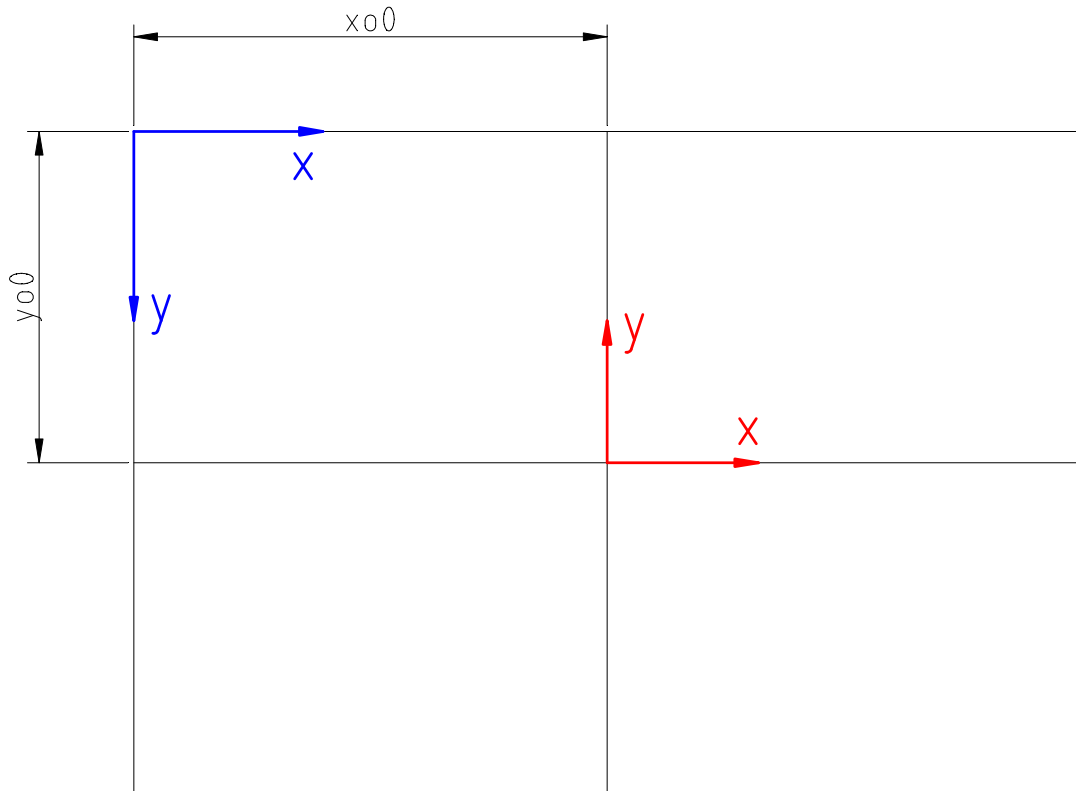
**Obrázek 4.3.** Globální (červené) a kamerové (modré) souřadnice

**Měřenou polohu** budu značit  $(x_p, y_p)$ . Tyto souřadnice budou vycházet z levého horního rohu obrazu.

**Obrazové souřadnice** budu značit  $(x_o, y_o)$ . Obrazové souřadnice budou na rozdíl od měřené polohy pravotočivé a jejich počátek bude ve středu obrazu, resp. na průsečíku obrazu s optickou osou. Vztah obrazových souřadnic s měřenou polohou je na Obrázku 4.4. Přepočítání mezi nimi je:

$$\begin{aligned} x_o &= x_p - x_{o0} \\ y_o &= y_{pmax} - y_p - y_{o0} \end{aligned} \quad (12)$$

kde  $y_{pmax}$  odpovídá výšce obrázku a je maximální hodnota souřadnice  $x_p$ ,  $x_{o0}$  a  $y_{o0}$  jsou souřadnice optické osy vůči levému hornímu rohu obrazu.



**Obrázek 4.4.** Obrazové (červené) a měřené (modré) souřadnice

### ■ 4.2.2 Zobrazení objektu objektivem na senzor

Kamera se skládá z optického senzoru a objektivu. Objektiv lze pro zjednodušení považovat za tenkou spojnou čočku. Zobrazení touto čočkou je nakresleno na Obrázku 4.5.

Z tohoto nákresu vychází zobrazovací rovnice čočky:

$$zz' = ff' \quad (13)$$

Při použití objektivu o ohniskové vzdálenosti 8 mm a minimální vzdálenosti sledovaného objektu 200 mm od hlavního bodu objektivu bude:

$$z' < \frac{ff'}{z} = \frac{64}{200} \text{ mm} = 0.32 \text{ mm} \ll 8 \text{ mm} = f \quad (14)$$

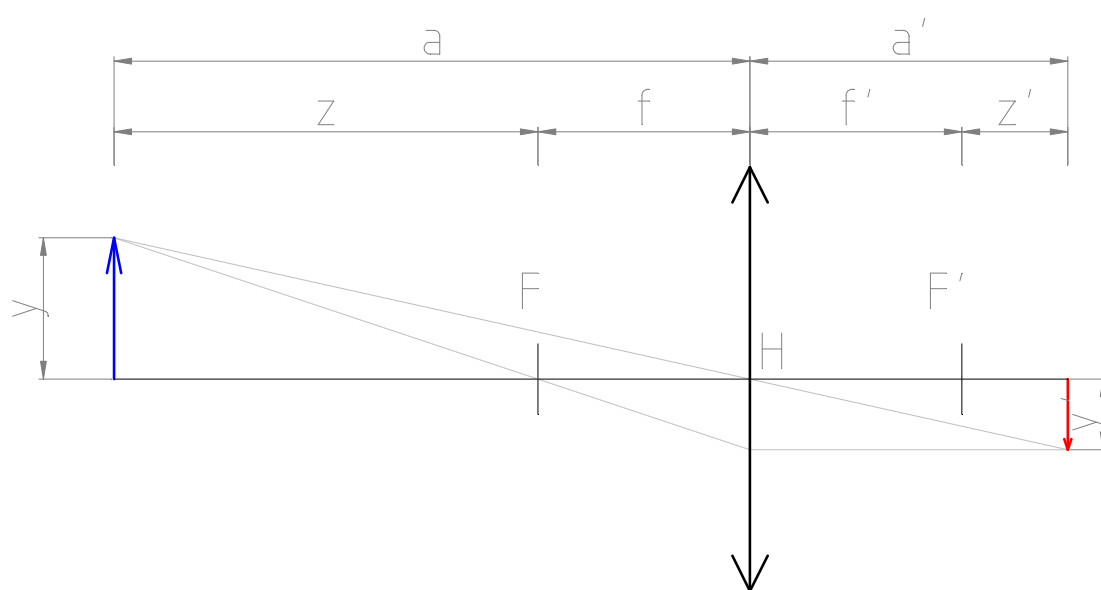
a tím pádem lze zavést zjednodušení – objekt bude zobrazen do ohniska kamery. Pro velikost zvětšení objektu bude platit:

$$\beta = \left| \frac{y'}{y} \right| = \frac{f}{a - f} \quad (15)$$

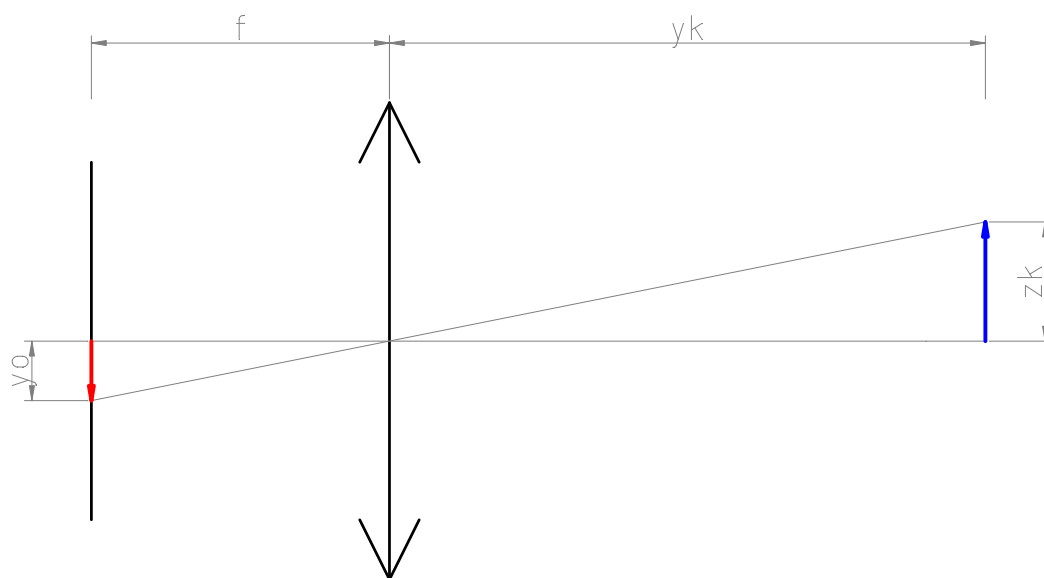
Jelikož všechny kamery budou zaostřeny na nekonečno, lze tyto vztahy zjednodušit. Dále budu předpokládat rovnost  $f = f'$  a umístění stínítka (CMOS senzoru) v ohnisku spojně čočky (objektivu). Tímto zjednodušením se dopustím chyby, protože objekt nebude nikdy ideálně zobrazen do ohniska. Vyjdeme-li z (14), bude maximální chyba zvětšení způsobená tímto zjednodušením 4%. Nákres této situace je na Obrázku 4.6.

Při použití značení z tohoto obrázku bude platit:

$$\frac{y_o}{f} = \frac{z_k}{y_k} \quad (16)$$



Obrázek 4.5. Zobrazení čočkou



Obrázek 4.6. Zjednodušené zobrazení čočkou



Vztah pro  $x_k$  bude analogický. Pro každou kameru tedy o objektu můžeme vytvořit dvě rovnice (pro  $x_k$  a  $y_k$ ):

$$\begin{pmatrix} 0 & y_o & -f \\ -f & x_o & 0 \end{pmatrix} \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (17)$$

Pro další výpočty označím:

$$\mathbf{Z} = \mathbf{Z}(x_o, y_o, f) = \begin{pmatrix} 0 & y_o & -f \\ -f & x_o & 0 \end{pmatrix} \quad (18)$$

Pro finální výpočet polohy se použijí tyto rovnice a za  $(x_k, y_k, z_k)$  se dosadí transformační vztahy do globálních souřadnic. (Viz. sekce 4.2.3.)

### 4.2.3 Vztah globálních a kamerových souřadnicových systémů

Polohu  $i$ -té kamery v prostoru můžeme popsat 6 souřadnicemi:

$$x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i \quad (19)$$

Z globálních do kamerových souřadnic se dostaneme takto:

1. posunem o  $(x_i, y_i, z_i)$
2. otočením o  $\alpha_i$  podle osy  $z$
3. otočením o  $\beta_i$  podle nové osy  $x$
4. otočením o  $\gamma_i$  podle nové osy  $y$

Tuto transformaci lze zapsat v homogenních souřadnicích jako maticový vztah:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \mathbf{T}\mathbf{R}_2\mathbf{R}_3\mathbf{R}_4 \begin{pmatrix} x_k \\ y_k \\ z_k \\ 1 \end{pmatrix} \quad (20)$$

Pro dosazení do rovnice (17) je třeba inverzní vztah, který bude ve tvaru:

$$\begin{pmatrix} x_k \\ y_k \\ z_k \\ 1 \end{pmatrix} = \mathbf{R}_4^{-1}\mathbf{R}_3^{-1}\mathbf{R}_2^{-1}\mathbf{T}^{-1} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (21)$$

Pro další výpočty budu  $\mathbf{R}$  značit matici, která bude tvořena prvními třemi řádky matice vzniklé součinem  $\mathbf{R}_4^{-1}\mathbf{R}_3^{-1}\mathbf{R}_2^{-1}\mathbf{T}^{-1}$ . Matice  $\mathbf{R}$  bude splňovat vztah:

$$\begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} = \mathbf{R} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (22)$$

Matice  $\mathbf{T}, \mathbf{R}_2, \mathbf{R}_3$  a  $\mathbf{R}_4$  budou:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & x_i \\ 0 & 1 & 0 & y_i \\ 0 & 0 & 1 & z_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{R}_2 = \begin{pmatrix} \cos(\alpha_i) & -\sin(\alpha_i) & 0 & 0 \\ \sin(\alpha_i) & \cos(\alpha_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (23)$$

$$\mathbf{R}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\beta_i) & -\sin(\beta_i) & 0 \\ 0 & \sin(\beta_i) & \cos(\beta_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{R}_4 = \begin{pmatrix} \cos(\gamma_i) & 0 & \sin(\gamma_i) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\gamma_i) & 0 & \cos(\gamma_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

#### ■ 4.2.4 Řešení přeúřčené soustavy rovnic pro polohu

Z každé kamery získáme dvojici rovnic, kterou lze zapsat:

$$\mathbf{Z}\mathbf{R} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \mathbf{0} \quad (24)$$

Pro  $n$  kamer získám  $2n$  rovnic, které upravím do tvaru:

$$\mathbf{A} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{b} \quad (25)$$

Tím jsem získal  $2n$  rovnic pro 3 neznámé. Každá dvojice rovnic reprezentuje jednu přímku v prostoru. Hledaný bod budu volit tak, aby co nejlépe odpovídal všem těmto rovnicím. Obecně se bude jednat o přeúřčený systém, tj.  $2n > 3$ . Tuto soustavu vyřeším Newton-Eulerovou metodou, která vypočte výsledek ve smyslu nejmenších čtverců:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} \quad (26)$$

### ■ 4.3 Kalibrace polohy obrazových senzorů

Poloha každé kamery je určena šesti parametry (viz. (19)). Přesnost výpočtu polohy závisí na přesnosti těchto parametrů. V mnohých případech je obtížné přesně změřit polohu kamer nebo by bylo přesné umístění příliš náročné. Proto je vhodné po umístění senzorů provést kalibraci, ve které se s použitím vlastního senzoru a kalibračního přípravku (konkrétní realizace viz. Obrázek 6.9) zpřesní změřené nebo odhadnuté parametry.

Při kalibraci budeme měřit polohu objektů známé polohy. Poté budeme optimalizovat parametry ve výpočtu (24) tak, aby pro všechny měřené body byla chyba co nejmenší. Zadefinujeme-li funkci  $\mathbf{f}$  jako:

$$\mathbf{f} = \mathbf{Z}\mathbf{R} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (27)$$

bude funkce  $\mathbf{f}$  funkcí polohy měřeného bodu, polohy kamery a dalších parametrů (např. ohniskové vzdálenosti). Pokud  $\mathbf{F}$  je vektor funkcí  $\mathbf{f}$  pro všechny kalibrační body, pak v ideálním případě by měla platit rovnost:

$$\mathbf{F} = \mathbf{0} \quad (28)$$

Tato rovnost platí ale jen v ideálním případě, kdy nedochází k chybám měření a všechny parametry jsou přesné, což v realitě nikdy nenastane. Předpokládejme, že tato chyba je způsobena pouze chybou parametrů. Potom bychom mohli lineárně aproximovat:

$$\mathbf{F} + \mathbf{J}_d \delta \mathbf{d} = \mathbf{0} \quad (29)$$

kde  $\mathbf{d} = (x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i)$  je vektor parametrů a  $\mathbf{J}_d$  je Jacobiho matice vektorové funkce  $\mathbf{F}$  podle vektoru  $\mathbf{d}$ . Řešením (29) ve smyslu nejmenších čtverců dostaneme změnu parametrů  $\delta \mathbf{d}$ , o kterou upravíme odhadnuté či změřené parametry. S takto upravenými parametry vypočteme znovu  $\mathbf{F}$  a  $\mathbf{J}_d$  a můžeme provést další iteraci zpřesnění parametrů polohy  $\mathbf{d}$ . Tento postup opakujeme tak dlouho, dokud se chyba zlepšuje.

## 4.4 Teoretická dosažitelná přesnost měření

Předpokládejme, že měřená veličina  $x$  se dá vyjádřit jako funkce  $F$  proměnných  $s_1, s_2, \dots, s_n$ . Standardní nejistota měření  $u_x$  veličiny  $x$  se určí ze vztahu:

$$u_x^2 = \sum_i^n \left( \frac{\partial F}{\partial s_i} \right)^2 u_{s_i}^2 \quad (30)$$

kde  $u_{s_i}$  jsou standardní nejistoty měřených veličin.

V případě určení polohy z měření třemi kamerami by funkce  $F$  byla funkcí 27 proměnných, která by se dala vyjádřit ze vztahu (26). Kvůli složitosti tohoto vztahu není snadné vyjádřit analytický tvar výpočtu nejistot v plném rozsahu. O vytvoření tohoto vzorce jsem se pokusil s využitím Matlabu. Tento pokus skončil po dlouhém běhu neúspěchem kvůli nedostatku paměti. Ruční výpočet by byl zdlouhavý a pravděpodobně chybný v důsledku lidského faktoru.

Proto při stanovení nejistot budu postupovat takto:

1. Budu předpokládat, že určená poloha kamer je správná a neměnná.
2. Pak budou transformační vztahy mezi souřadnými systémy konstantní.
3. Funkci  $F$  vyjádřím pomocí konkrétních (konstantních) transformačních vztahů mezi souřadnicemi.
4. Výpočet nejistot provedu pouze z 6 měřených veličin.

# Kapitola 5

## Implementace

Pro testování a funkci systému jsem vytvořil několik druhů programů:

- programy pro MCU komunikující přes USB
- programy pro MCU komunikující přes Ethernet
- skripty a funkce pro Matlab

Pro komunikaci mezi počítačem a jednotlivými moduly jsem používal Matlab, a to ze dvou důvodů: umožňuje relativně snadný přístup k sériové lince (a při importu Javy i k socketům a TCP/IP) a lze v něm velmi snadno vykreslovat grafy a vytvářet jednoduché animace, které jsou potřeba k ověření funkčnosti systému. Díky příkazové řádce navíc není nutné kvůli jednoduchým instrukcím psát celý program.

Komunikace probíhala na dvou úrovních – přes USB a pomocí TCP/IP. Přes USB se komunikovalo s programem pro testování algoritmů zpracování obrazu, nastavování parametrů senzoru a vyčítání obrazu do PC. Druhý program se používal k synchronizovanému sběru dat, tedy k měření polohy.

Obrazový senzor je ovládán pomocí knihovny, kterou jsem vytvořil v předcházejícím individuálním projektu [10]. Knihovnu jsem dále upravoval a rozšiřoval její funkčnost, aby vyhovovala požadavkům aplikace.

### 5.1 Programové vybavení pro řídicí MCU

Dva vytvořené projekty vycházejí z potřeb při vývoji systému. Pro základní vyzkoušení funkčnosti jednotlivých sensorových modulů a vyčítání obrazu z kamer jsem vytvořil a využíval program komunikující přes USB pomocí Virtual Com Port (VCP).

Pro vlastní funkci systému bylo nutné, aby docházelo k synchronizovanému sběru dat. K synchronizaci byl použit protokol PTP přes rozhraní Ethernet. K tomuto účelu jsem upravil program k použitému hardwaru (viz. sekce 3.1.3).

U všech těchto programů bylo třeba dát pozor na nastavení hodin, které se nachází v souboru `system_stm32f4xx.c`. Nastavení hodin se vůči vzorovým programům lišilo vždy v první předděličce hodin vstupujících do bloku PLL. Vstup má být 1 MHz, čehož bylo s externím 50 MHz oscilátorem docíleno použitím předděličky 50.

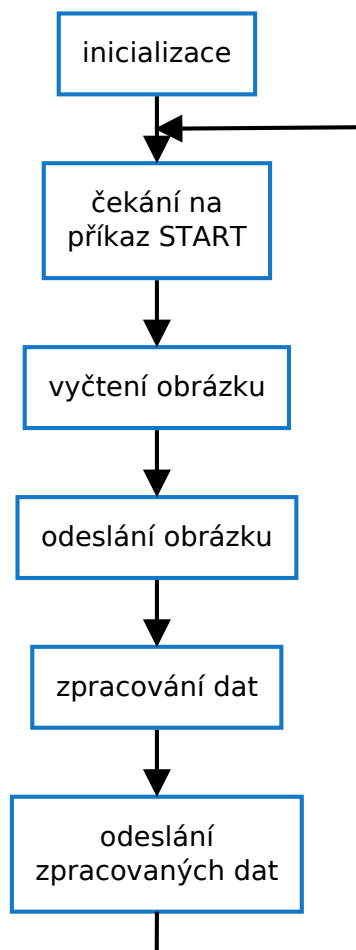
#### 5.1.1 Firmware pro přístup k modulu přes USB

Pro vytvoření programu umožňujícího přístup k modulu přes USB rozhraní jsem vyšel z vzorového příkladu od výrobce mikrokontroléru (STMicroelectronics) – USB VCP example. Pro snadné umožnění komunikace jsem v něm provedl několik úprav:

- úprava funkce `VCP_DataTx`, aby fungovala ve smyslu odesílání bloku dat
- úprava funkce `VCP_DataRx` – přesunutí do hlavního aplikačního souboru `app.c`
- úprava funkcí pro nastavování parametrů COM portu – vymazání funkčnosti<sup>1)</sup>

<sup>1)</sup> Příklad byl určen pro komunikaci s vnějším zařízením přes rozhraní UART. Toho tato aplikace nevyužívala a bylo třeba tuto funkčnost odstranit, aby se nenarušila funkce aplikace používáním výstupních/vstupních pinů, které zabíral UART.

Běh programu je znázorněn na Obrázku 5.1. Během inicializace dojde k základnímu nastavení senzoru a k nastartování a inicializaci USB. Poté vstoupí program do hlavní smyčky, ve které periodicky dochází k vyčítání obrazu (iniciované příkazem přes USB), odesílání obrazu a výpočtu.



**Obrázek 5.1.** Hlavní vlákno programu přes USB

Nastavování parametrů je zajištěno pomocí funkce `VCP_DataRx`, která je volaná v rámci přerušení metod implementace USB. Tato funkce nastavuje přepínače, podle kterých se vykonávají jednotlivé části hlavní smyčky programu. Takto lze:

- vypnout/zapnout odesílání obrázku
- vypnout/zapnout zpracování dat
- přepínat mezi zpracováním po liniích a určením těžiště ve 2D
- vypnout/zapnout využívání stupňů šedi
- přímo číst a zapisovat registry senzoru
- nastavovat *binning* obrazu nebo výřez jeho části

Rychlost jedné programové smyčky je cca 0.2 s při odesílání obrazu do počítače a cca 0.08 s, když se obraz do počítače neodesílá. Všechna zpracování obrazu probíhají s obrazem o velikosti 188 na 120 obrazových bodů.

### ■ 5.1.2 Firmware pro komunikaci přes Ethernet

Tato část softwaru vychází ze softwaru k použitému hardwaru. Důležitá byla především proto, že zajišťuje synchronizaci modulů. Tento software zajišťuje dvě základní funkce:

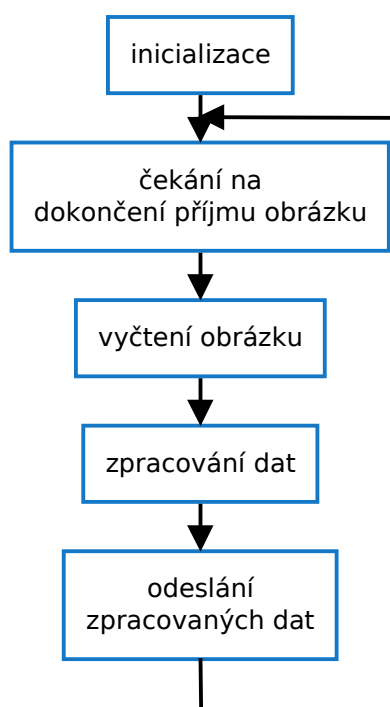
- synchronizace a přijímání a odesílání dat přes Ethernet
- periodické měření po 0.2 s

Zpracování obrazu (měření) probíhá ve dvou variantách (byly vytvořeny dva rozdílné projekty):

- určení těžiště objektu v obraze ve 2D (pro měření 3D polohy)
- určení polohy kontrastní stopy ve vybraných liniích (pro 2D měření)

Příchozí komunikace a synchronizace pomocí PTP byla řešena pomocí pollingu dat – v průběhu jakéhokoli čekání program kontroluje, zda se přes Ethernet přijala data, pokud ano, tak je zpracuje. Synchronizované měření je zajištěno pomocí přerušení PTP jednotky v procesoru – generuje se přerušení, které zahájí expozici.

Běh hlavní smyčky programu je podobný USB variantě a je zobrazen na Obrázku 5.2. Hlavní rozdíl je v absenci odesílání obrázku do počítače a použití jiného přenosového kanálu do počítače. Možnost parametrizace sice zůstala zachována, nebyla ale využívána.



**Obrázek 5.2.** Hlavní vlákno programu přes Ethernet

Po naměření dat vždy dojde k odeslání aktuálních hodnot. Každý takto odeslaný packet je opatřen časovou značkou, podle které je pak možné poskládat jednotlivá měření z různých modulů k sobě.

Pro správnou funkci komunikačního rozhraní bylo nutné tři použité moduly naprogramovat s rozdílnou IP a MAC adresou, které se nastavují v souboru `main.h`. Pokud se nastaví dvě stejné adresy, systém vůbec nefunguje.

Během rozběhávání komunikace jsem řešil problém velmi snížené rychlosti odesílání dat z modulu do počítače. Nakonec se ukázalo, že to bylo způsobené zapnutým Nagle algoritmem na straně modulu. Modul neodesílal další data přes protokol TCP, dokud nedostal potvrzení (ACK), že došla předchozí data v pořádku. Počítač ale odesílal potvrzení až po cca 200 ms, čímž byla komunikace výrazně zpomalena, protože aplikace vyžadovala odesílání spíše většího množství malých packetů. Toto se opravilo zavoláním funkce `tcp_nagle_disable` při sestavení TCP spojení.

## 5.2 Skripty a funkce pro Matlab

Během práce jsem vytvořil velké množství skriptů a funkcí pro Matlab. Tyto skripty a funkce řešily několik problémových oblastí:

- komunikace
- procesy vyčítání dat
- zobrazování naměřených dat
- zpracování dat
- ovládání lineárního posuvu
- principiální ověření algoritmů

Tyto funkce jsem rozdělil do složek a vytvořil si skript pro inicializaci složkové struktury (resp. přidání těchto složek do pracovních cest Matlabu). Tím jsem docílil lepší přehlednosti, než jakou by mělo ponechání všech funkcí a skriptů v jednom adresáři.

### 5.2.1 Přístup k externímu hardwaru

V Matlabu lze snadno přistupovat k externímu hardwaru. Výhodou přístupu přímo z Matlabu je okamžitý přístup k datům pro skripty použité na jejich zpracování. K externímu hardwaru jsem přistupoval dvěma způsoby:

- přes sériovou linku
- přes TCP protokol a Ethernet

Pro sériovou linku existují přímo funkce napsané pro Matlab, které jsem s výhodou využil. Pro spojení pomocí TCP protokolu bylo nutné využít možnosti načtení Javy do prostředí Matlabu. Nevýhodou tohoto řešení je, že třída `Socket`, která se pro komunikaci přes síť používá, není serializovatelná, a tedy nelze uložit do souboru. To mi občas způsobilo problémy, protože Matlab při pokusu o uložení naměřených dat selhal a měření jsem musel provádět znovu.

### 5.2.2 Knihovna pro ovládání polohovací jednotky MARS

V pokusech (viz. Kapitola 6) jsem používal na přesné polohování lineární posun spolu s jednotkou pro řízení motorů MARS. Tato jednotka může být ovládána přes sériovou linku. Abych mohl ovládat jednotku snadno z počítače, vytvořil jsem knihovnu několika funkcí pro Matlab, které umožňují základní ovládání jednotky, zejména:

- připojení a nastavení parametrů linky
- homing jednotlivých motorů
- pohyb do relativní a absolutní pozice
- zjišťování, zda je dokončen pohyb

Podrobnější popis knihovny je v příloze F.

### ■ 5.2.3 Skupina skriptů pro testování zpracování obrazu

Všechny algoritmy zpracování obrazu jsem nejprve implementoval v Matlabu a odkrokoval, abych si ověřil jejich funkčnost. Díky tomu pak nebylo nutné ověřovat jejich funkčnost přímo na mikrokontroléru, kde by tento proces byl výrazně složitější, protože by data šla hůře vizualizovat.

### ■ 5.2.4 Skripty pro 3D výpočet polohy

Výpočet polohy ve 3D jsem vytvořil v Matlabu. Matlab jsem použil jak pro odvození vzorců (využití Symbolic toolboxu), tak pro následné výpočty při určování polohy.

Odvození vzorců pomocí počítače jsem zvolil zejména kvůli eliminaci chyb při elementárních úpravách. Většina vzorců byla odvozena pomocí matic na papíře a následně matice přepsány do Matlabu a vyřešeny počítačem.

Výpočty kvůli rychlosti nepoužívaly symbolické proměnné a byly ze spočítaných vzorců přepsány do funkcí. Tyto funkce byly integrovány do skriptů, které zajišťovaly průběh měření.



# Kapitola 6

## Dosažené výsledky

V rámci této práce vznikl systém pro mnohokanálové měření polohy. Pro tento systém byly navrženy algoritmy určení těžiště objektu v obraze a vytvořena programová nadstavba do řídicího PC pro prostorové měření.

Pro určení vlastností vytvořeného systému jsem provedl pokusy, které testovaly:

- určení polohy těžiště
- chování 3D systému měření polohy
- chování systému měření po liniích ve 2D

Všechna následující měření byla provedena se senzorem v manuálním režimu a nastavenou expoziční dobou 0.47 ms.

### 6.1 Chování algoritmů určení těžiště v obraze

Pro zhodnocení vlastností metod určení těžiště jsem provedl dva druhy pokusů:

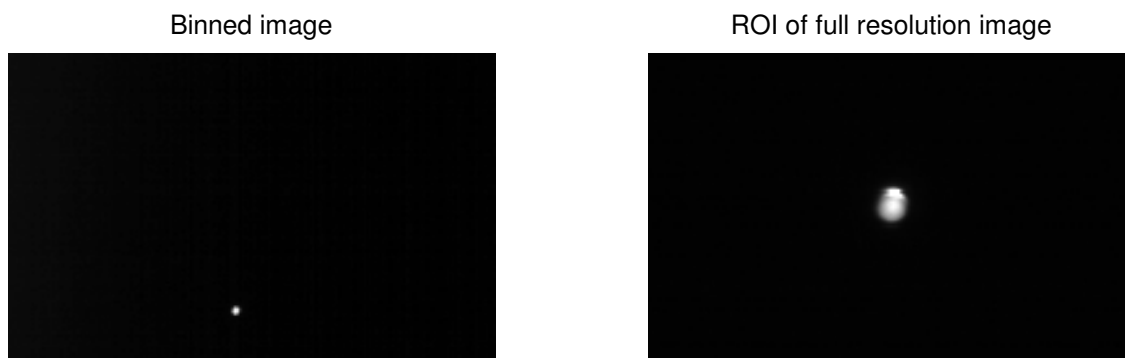
- test opakovatelnosti měření
- test rozlišovací schopnosti určení polohy těžiště

Pro představu o přesnosti měření jsem při stejném rozestavení pokusu naměřil i geometrickou vadu objektivu.

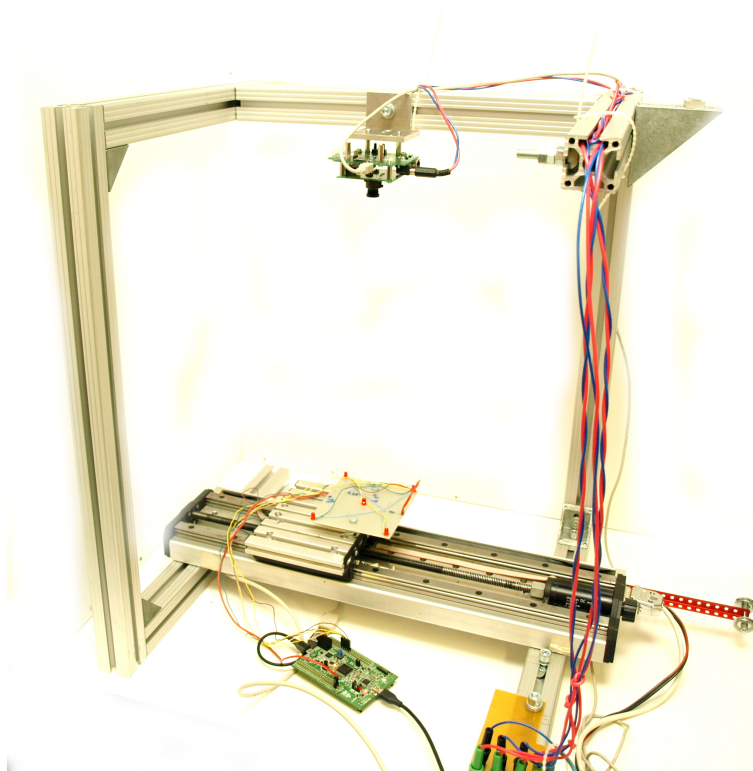
Pro tyto pokusy jsem umístil kameru 360 mm nad stolek lineárního posuvu. Na stolku byla pevně připevněna LED. Obrazový senzor s objektivem s 8 mm ohniskovou vzdáleností byl umístěn tak, že LED na stolku byla ve středu obrazu kamery a mohla se posouvat lineárním posuvem po ose  $y$  obrazu. Sestavený pokus je vidět na Obrázku 6.1.

V následujících testech jsem pracoval se 4 vypočítanými polohami těžiště, které rozlišovalo především použité rozlišení a využití stupňů šedi z obrazových dat.

Z obrazového senzoru jsem vyčítal buď výřez obrazu v plném rozlišení (dále jako obraz v *plném rozlišení*, resp. *full resolution ROI*), nebo plný obraz v redukovaném rozlišení (dále jako *binnigovaný* obraz, resp. *binned image*). Příklad obou obrazů viz. Obrázek 6.2.



Obrázek 6.2. Redukované rozlišení obrazu a výřez z plného rozlišení



**Obrázek 6.1.** Sestava pokusu pro test algoritmů

Plné rozlišení použitého senzoru je 752 na 480 obrazových bodů. *Binningovaný* obraz byl získán ze senzoru s využitím funkce čtyřnásobného *binningu* v řádcích i sloupcích (popis viz. [7])<sup>1)</sup>, měl tedy rozlišení 188 na 120 obrazových bodů.

Obraz v *plném rozlišení* byl vyčítán pouze jako výřez celého obrazu o velikosti 188 na 120 obrazových bodů.

Při výpočtu těžiště se využíval buď *binární* obraz (viz.(1)), nebo obraz ve *stupních šedi* (viz. (2)).

Při následujících pokusech bude jednotka 1 px znamenat 1 obrazový bod *plného rozlišení* senzoru. Použitý senzor má rozlišení WVGA, tedy 752 na 480 obrazových bodů, 1 px má na senzoru velikost 6  $\mu\text{m}$ .

Varianty určení polohy těžiště byly:

- výpočtem z *binningovaného binárního* obrazu
- výpočtem z *binningovaného* obrazu ve *stupních šedi*
- výpočtem z *binárního* obrazu v *plném rozlišení*
- výpočtem z obrazu ve *stupních šedi* v *plném rozlišení*

Posuv byl zajištěn lineárním posuvem, který umožňoval polohování s přesností na 0.001 mm.

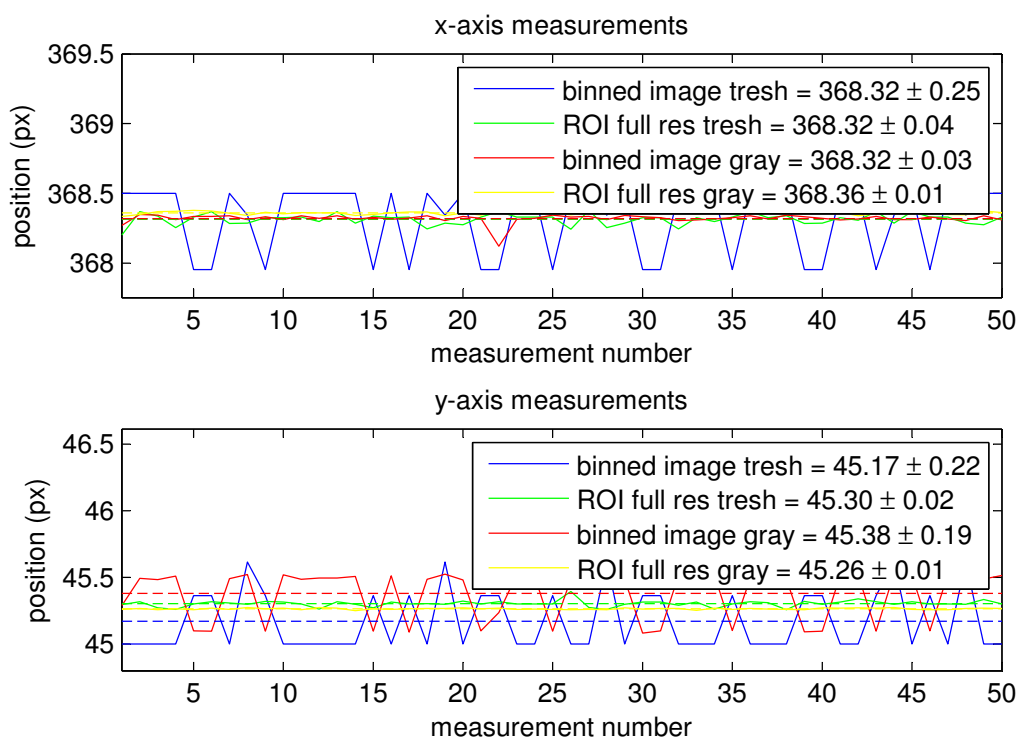
### ■ 6.1.1 Test opakovatelnosti měření

Při testech opakovatelnosti měření bylo padesátkrát určeno těžiště staticky umístěné LED. Cílem tohoto pokusu bylo vyhodnotit stabilitu měření, tedy vliv šumu.

<sup>1)</sup> Každý pixel nově vzniklého obrazu byl získán z 16 sousedních pixelů, jak přesně je popsáno v data-sheetu [7].

Bylo provedeno několik měření a po jejich provedení se ukázalo, že vycházejí dva extrémní typy výsledků, které reflektují chování obrazového senzoru při redukci rozlišení pomocí *binningu*.

První význačný příklad je, když se měřená LED nacházela na hranici *binningovaného* pixelu, takže šum způsobil přechody jednoho obrazového bodu nad a pod hranici prahování. V tomto případě jsou významné rozdíly mezi standardními odchylkami jednotlivých použitých metod. Příklad tohoto chování je na grafu na Obrázku 6.3.



Obrázek 6.3. Opakovatelnost měření těžiště

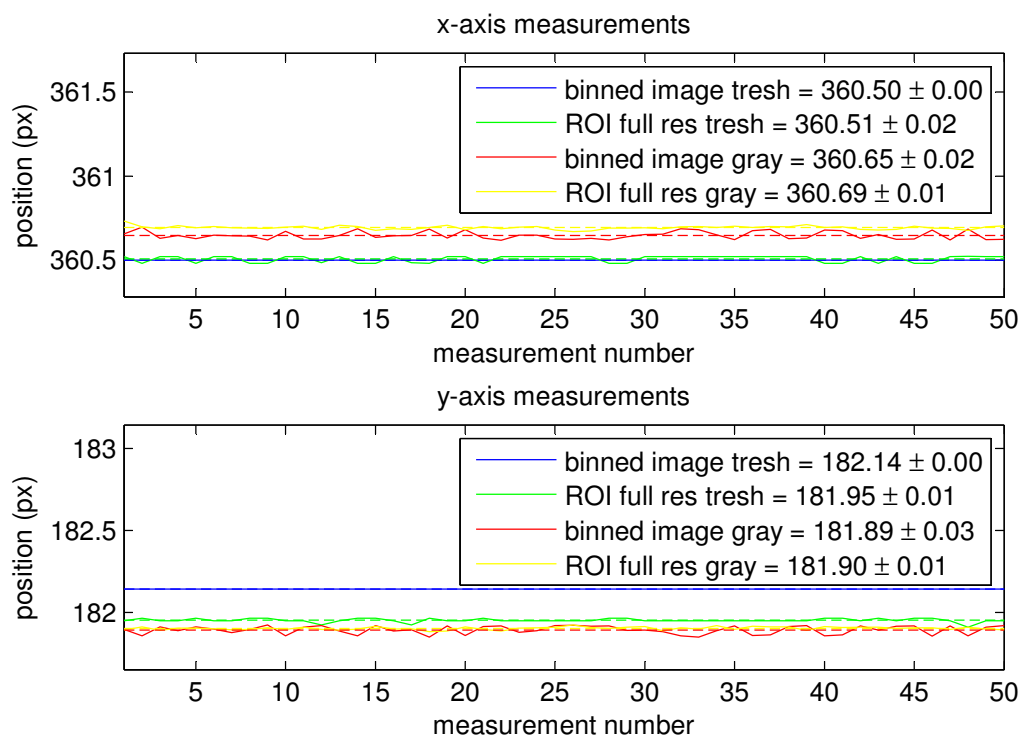
Druhá možnost je, že umístění diody bylo v místě, ve kterém se veškerý šum ztratil při *binningu*. V tomto případě vycházely standardní odchylky výpočtu těžiště v *binningovaném* obraze nulové, a tedy výrazně lepší než z obrazu s plným rozlišením. Toto chování je vidět na Obrázku 6.4.

### 6.1.2 Určení rozlišovací schopnosti

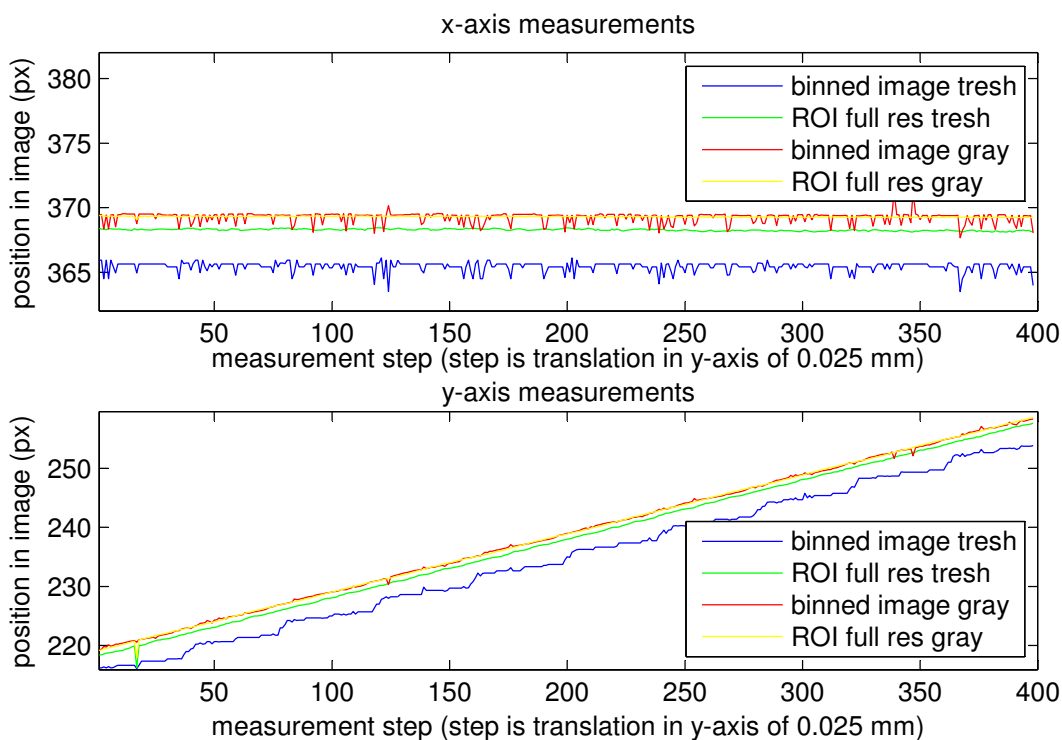
Cílem tohoto pokusu bylo zjistit, jakou rozlišovací schopnost mají různé přístupy k měření těžiště v obraze a určit, zda je smysluplné snažit se o co nejvyšší rozlišení zpracovaného obrazu.

Při tomto testu kamera sledovala LED diodu umístěnou na posuvném stolku, který se posouval s krokem 0.025 mm ve směru osy y obrazu. Toto měření bylo provedeno v okolí optické osy objektivu, aby bylo možno bez velkých chyb zanedbat geometrické vady použitého objektivu. Celkem bylo provedeno 400 měření, což odpovídá posuvu stolku o 10 mm.

Pro měření bylo použito připojení modulu přes USB rozhraní, obrázky byly přenášeny do počítače a následně zpracovávány na počítači.



Obrázek 6.4. Opakovatelnost měření těžiště ve výhodné poloze pro *binning*

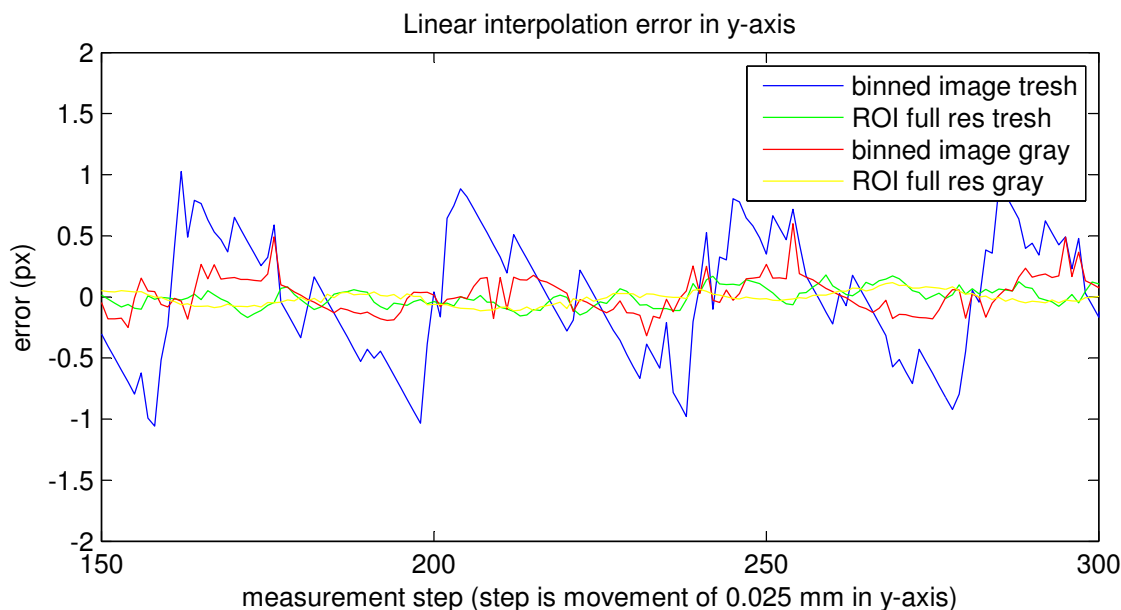


Obrázek 6.5. Test přesnosti měření těžiště v obraze

Naměřené hodnoty jsou zobrazeny na Obrázku 6.5. Pozice naměřené v ose  $x$  se nemění, což odpovídá pohybu pouze po ose  $y$ . Zpracování pro určení rozlišovací schopnosti určení polohy těžiště jsem proto provedl pouze z dat naměřených v ose  $y$ .

K vyhodnocení přesnosti měření jsem celé měření proložil přímkou a zobrazil graf odchylky naměřené hodnoty od této přímky. Tento graf je vyobrazen na Obrázku 6.6. Z grafu je patrné, že vyšší rozlišení senzoru pozitivně ovlivňuje přesnost určení polohy těžiště. Využití stupňů šedi navíc přináší další zlepšení.

Jako optimální metoda zpracování se na základě tohoto pokusu jeví zpracování v *plném rozlišení* při využití *stupňů šedi*. Rozdíl mezi maximálními chybami při výpočtu z *binárního* obrazu a obrazu se *stupni šedi* je sice minimální, nicméně průběh poloh vypočtený pomocí *stupňů šedi* je hladší, což svědčí o menším vlivu šumu. Pokud by se zvlnění korigovalo, šlo by dosáhnout ještě lepší rozlišovací schopnosti.



Obrázek 6.6. Chyby linearity výpočtu těžiště

### 6.1.3 Chyba geometrického zobrazení

Při využití obrazových dat pro měření je vhodné mít představu o chybě, kterou do měření přinesou geometrické chyby objektivu. K určení těchto chyb jsem provedl měření polohy LED s využitím obrazu v *plném rozlišení* a výpočtu se *stupni šedi*. Provedl jsem měření přes celý rozsah měření v ose *y* obrazu.

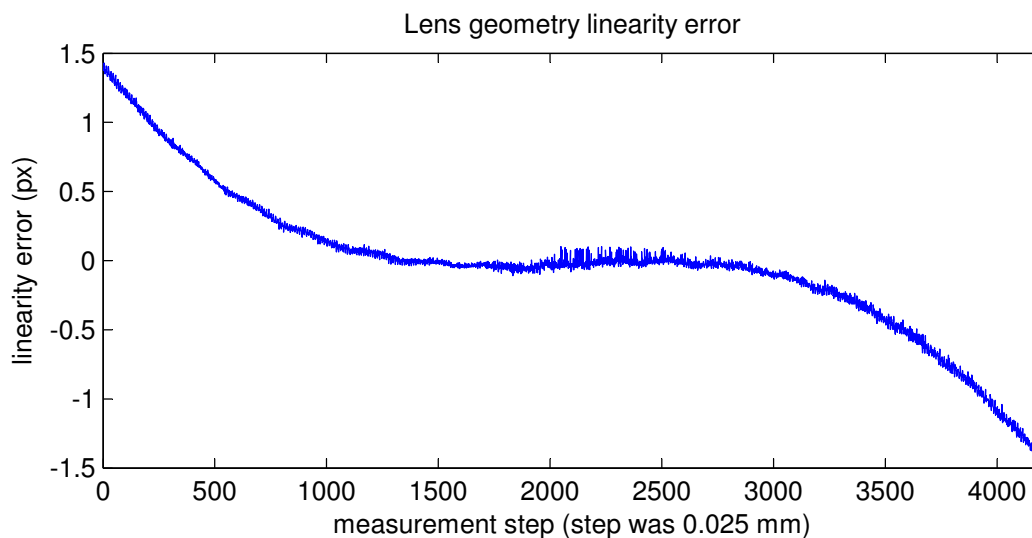
Naměřená závislost by teoreticky měla být lineární (pokud by v objektivu byla ideální čočka). Ve skutečnosti však dochází k chybám. Tyto chyby jsou v okolí optické osy velmi malé a se vzdáleností od ní rostou. Toho jsem využil při prokládání přímkou – proložení jsem provedl pouze v okolí optické osy (metodou nejmenších čtverců).

Chyby způsobené geometrickými vadami čočky zobrazeny na Obrázku 6.7. Jedná se o rozdíl mezi naměřenými daty a proloženou přímkou, tedy o chybu zobrazení objektivem při linearizaci v optickém středu objektivu.

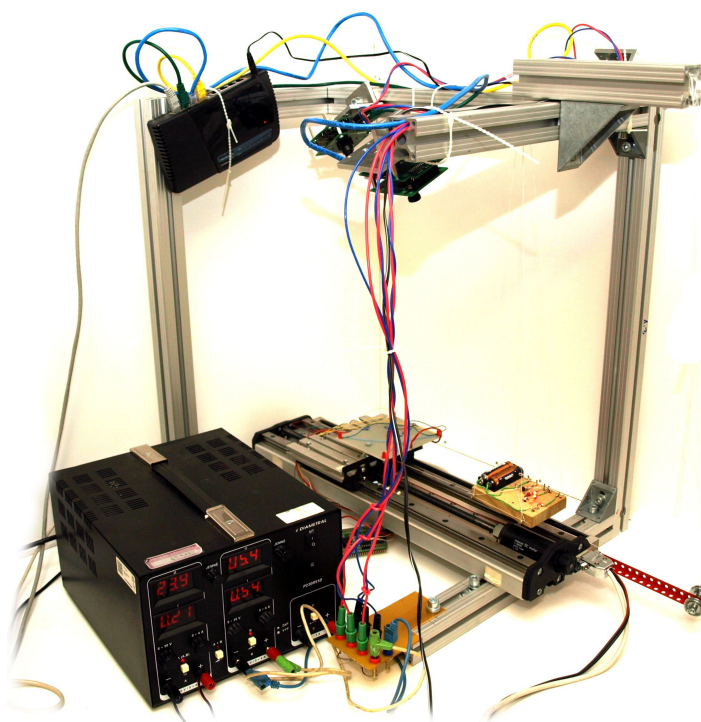
## 6.2 Prostorové měření ve 3D

Pro měření v prostoru se využily 3 kamery propojené mezi sebou a s počítačem přes Ethernet. Kamery byly umístěny shora nad lineárním posuvem. Na lineárním posuvu byl umístěn kalibrační přípravek. Celé rozmístění pokusu je vyobrazeno na Obrázku 6.8.

Pro měření jsem zavedl globální souřadný systém, jehož počátek byl v jedné LED na kalibračním přípravku při zajištění vozíku lineárního posuvu na nulovou pozici (využití



**Obrázek 6.7.** Geometrické chyby zobrazení objektivem (měřené přes osu  $y$  senzoru, hodnoty  $y$  od 12 do 433 px, na jeden krok měření by měl při zobrazení bez chyb připadat posun v obraze o 0.093 px)



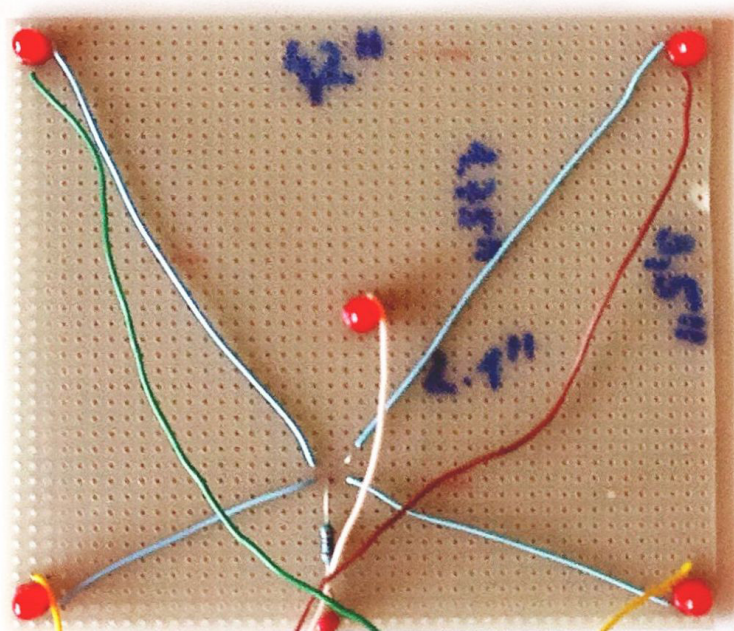
**Obrázek 6.8.** Pokus pro 3D měření polohy objektu

funkce *home*, vozík byl přibližně na pozici visícího dřevěného přípravku s LED – viz. Obrázek 6.8).

Souřadný systém byl orientován následovně (popis odpovídá obrázku):

- osa  $x$  byla ve směru od kalibračního přípravku směrem doleva dolů
- osa  $y$  byla ve směru od kalibračního přípravku směrem doprava dolů
- osa  $z$  směřovala směrem vzhůru





**Obrázek 6.9.** Přípravek pro kalibraci kamer

Kalibrační přípravek s LED byl vyroben s použitím univerzálního vrtaného plošného spoje (viz. Obrázek 6.9). Odhadovaná přesnost umístění LED na kalibračním přípravku je v řádu desetin milimetru.

Globální souřadný systém byl pevně spojen s kalibračním přípravkem. Polohy kalibračních diod byly:

$$X_1 = \begin{pmatrix} 0.00 \\ 0.00 \\ 0.00 \end{pmatrix}, X_2 = \begin{pmatrix} 106.68 \\ 0.00 \\ 0.00 \end{pmatrix}, X_3 = \begin{pmatrix} 53.34 \\ 44.45 \\ 0.00 \end{pmatrix}, X_4 = \begin{pmatrix} 0.00 \\ 88.90 \\ 0.00 \end{pmatrix}, X_5 = \begin{pmatrix} 106.68 \\ 88.90 \\ 0.00 \end{pmatrix}$$

kde všechny jednotky jsou v milimetrech.

Polohu kamer určenou šesticí parametrů (viz. (19)) bylo nutné přibližně změřit či odhadnout kvůli vstupu do kalibračního algoritmu. Polohu  $(x, y, z)$  kamer jsem odhadl hrubým měřením s použitím pravítka, úhly natočení pouze náhledem, důležité bylo, aby natočení kamer bylo přibližně správným směrem.

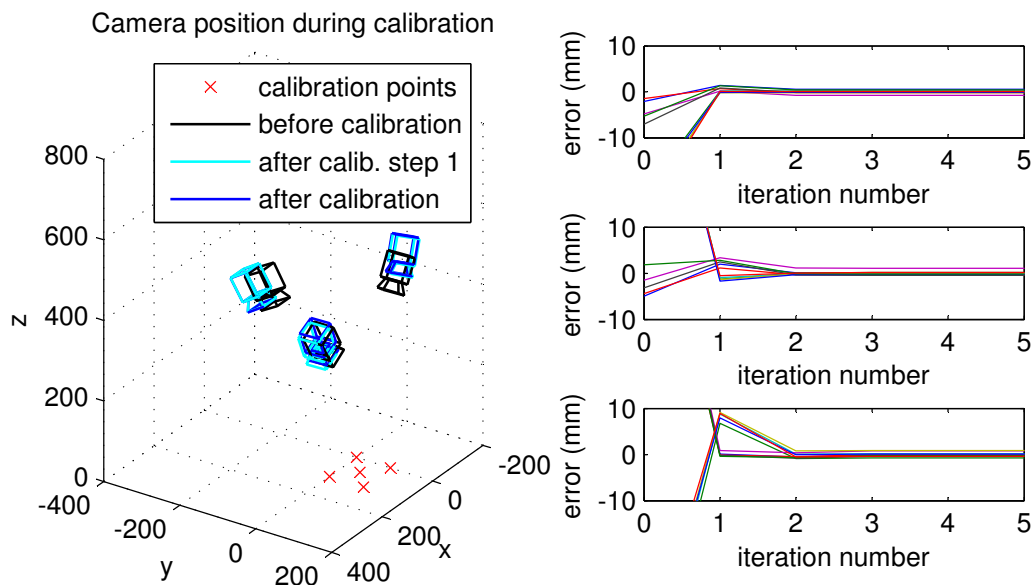
Odhadnuté polohy kamer byly:

$$\mathbf{d}_1 = \begin{pmatrix} 50 \\ -180 \\ 350 \\ 0 \\ -\frac{\pi}{3} \\ 0 \end{pmatrix}, \mathbf{d}_2 = \begin{pmatrix} 170 \\ 40 \\ 340 \\ \frac{\pi}{2} \\ -\frac{\pi}{3} \\ -\frac{\pi}{2} \end{pmatrix}, \mathbf{d}_3 = \begin{pmatrix} -60 \\ 50 \\ 400 \\ -\frac{\pi}{2} \\ -\frac{\pi}{3} \\ \frac{\pi}{4} \end{pmatrix}$$

kde všechny pozice jsou v mm, všechny úhly jsou v radiánech. Kamera 1 je na Obrázku 6.8 nejvíce vlevo, kamera 2 je nejvíce vpředu a kamera 3 je vzadu vpravo.

Pro ověření funkčnosti měření polohy pohybujícího se objektu byl v pokusu zavěšený přípravek s LED, jehož výšku šlo ovlivňovat přes kladkový mechanismus pomocí lineárního posuvu. Při pokusu se hýbalo stolkem o 50 mm, což odpovídalo vertikálnímu posuvu měřeného objektu o 40 mm.<sup>1)</sup>

<sup>1)</sup> Kvůli geometrické složitosti kladkového mechanismu byl tento údaj přímo změřen pomocí značek nakreslených na uchycujícím provázku.



**Obrázek 6.10.** Průběh kalibrace polohy kamer (napravo jsou průběhy chyb jednotlivých kamer, chyba je vzdálenost umístění reálného bodu od roviny určené měřením kamerou)

### 6.2.1 Ověření funkčnosti kalibrace polohy

Pro kalibraci byla provedena měření pěti kalibračních LED. Z těchto měření se následně iterativně vylepšoval odhad pozice kamer v prostoru. Kalibrace probíhala podle postupu popsáno v sekci 4.3. Průběh kalibrace naznačuje Obrázek 6.10.

Pro orientační zjištění přesnosti kalibrace jsem vypočítal vzdálenosti jednotlivých kalibračních bodů od rovin určených měřeními jednotlivými kamerami. Z každé kamery jsou pro každý bod určeny dvě roviny, dohromady pro 5 kalibračních bodů se jedná o 10 rovin a tedy 10 chyb pro každou kameru.

Na Obrázku 6.10 vpravo je vidět závislost vzdálenosti kalibračních bodů od změřených rovin na iteračním kroku kalibračního algoritmu.

Polohy kamer po kalibraci:

$$\mathbf{d}_1 = \begin{pmatrix} 73.0635 \\ -198.6333 \\ 352.7151 \\ -0.0074 \\ -1.0074 \\ 0.0052 \end{pmatrix}, \mathbf{d}_2 = \begin{pmatrix} 184.0280 \\ 30.3350 \\ 345.9959 \\ 1.5211 \\ -1.1982 \\ -1.6491 \end{pmatrix}, \mathbf{d}_3 = \begin{pmatrix} -136.6858 \\ 21.1397 \\ 402.3702 \\ -1.3515 \\ -1.0869 \\ 1.0354 \end{pmatrix}$$

Po kalibraci kamer byly kalibrační body vzdálené od změřených rovin, na kterých se měly nacházet, v řádu desetin milimetrů. To odpovídá předpokládané přesnosti kalibračního výrobku.

Pro zlepšení vlastností kalibrace by mohlo pomoci:

- zlepšit přesnost kalibračního přípravku (umístění LED)
- umístit kalibrační body do prostoru a ne jen v rovině
- zvýšit počet kalibračních bodů

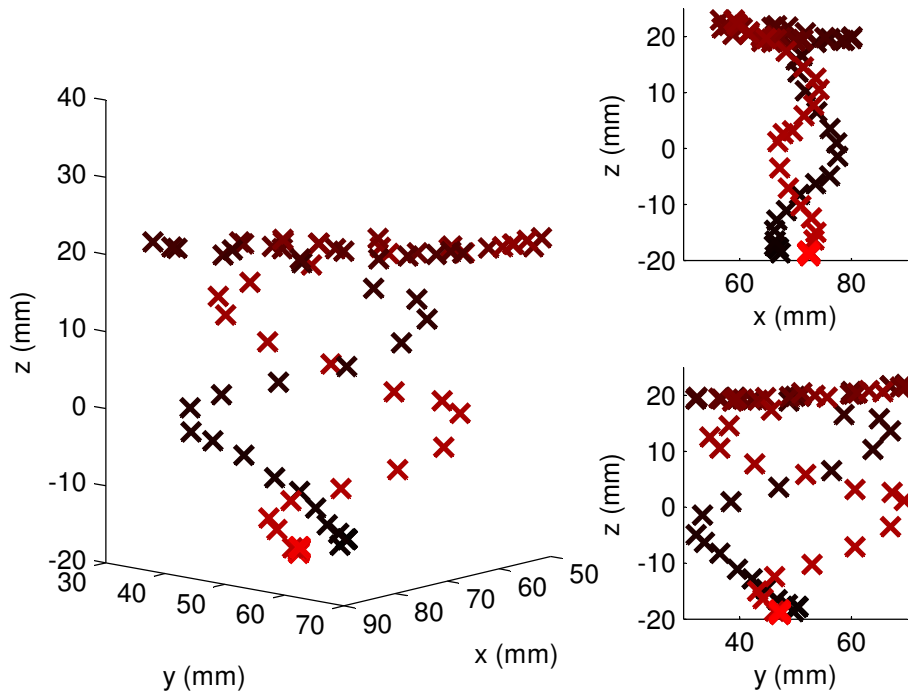
### 6.2.2 Určování polohy pohybující se LED

Pro demonstraci funkčnosti systému je nutné ukázat, že systém měří ve všech osách současně. Pro tento pokus byl použit referenční objekt vybavený svítící LED diodou,



kteřý byl zavěšen a přes kladku napojen na lineární posuv, který s ním pohyboval nahoru a dolů. Pro pohyb v osách  $x$  a  $y$  bylo využito chování celého závěsu jako matematického kyvadla.

Graf průběhu měření je vyobrazen na Obrázku 6.11. Bylo provedeno celkem 100 měření se vzorkovací frekvencí 5 Hz. Jedno takovéto měření tedy zabralo dobu 20 s.



**Obrázek 6.11.** Výsledek 3D měření (intenzita červené barvy odpovídá času)

Pro správnou funkci měření pohybujícího se objektu je nutná synchronizace kamer. Pokud by synchronizace nebyla v pořádku, jednotlivé kamery by vyfotily objekt na různých místech a projevilo by se to na výsledném určení polohy. Naměřený pohyb objektu by nebyl hladký a výrazně by se zhoršila přesnost měření.

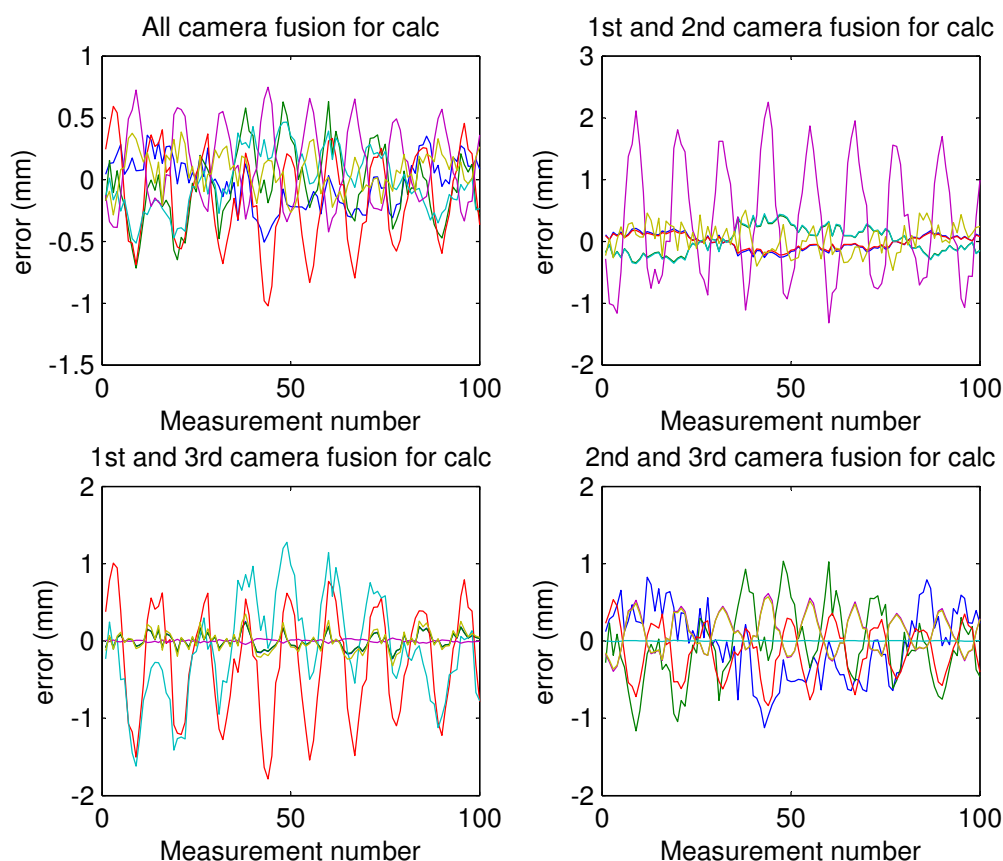
### ■ 6.2.3 Vliv měření jednotlivými kamerami v přeúčném systému

Pro relevanci měření více kamerami, které vytváří přeúčný systém, je nutné vědět, zda některá kamera do měření nezanáší spíše chybu než přínos.

Fúze dat naměřených jednotlivými kamerami se provádí metodou nejmenších čtverců. Pokud by jedna kamera měřila špatně, např. kvůli špatně provedené kalibraci polohy, zkreslovala by měření polohy.

Pro vyobrazení správnosti výpočtu jsem použil podobný postup jako při zobrazení chyb kalibrace polohy kamer. Pro každou kameru jsem určil dvě chyby, kterými je vzdálenost změřené polohy objektu od rovin určených z měření těžiště objektu kamerou. Pro tři kamery jsem tak získal 6 signálů.

Poloha objektu se získává metodou nejmenších čtverců z měření třemi kamerami. Pro ověření, že některá z kamer nezanáší výraznou chybu, jsem provedl výpočty i pouze jednotlivými dvojicemi kamer. Následně jsem zobrazil chyby i pro tyto výpočty pro všechny kamery (pro výsledky z měření zobrazeného na Obrázku 6.11 viz. Obrázek 6.12).



**Obrázek 6.12.** Chyby způsobené přeúčteností systému

Tento způsob zobrazení chyb neřeší chyby měření výsledné polohy objektu, je to však vhodný způsob, jak odhalit chybu např. ve vzájemném umístění kamer nebo nevhodném umístění jednotlivé kamery. Po zobrazení výsledků tohoto šetření se mi podařilo při jednom z měření odhalit chybu kalibrace polohy jedné z kamer, která se projevila výrazně odlišnými chybami této kamery při výpočtu polohy z druhých dvou kamer. Graf zobrazující tuto situaci je na Obrázku 6.13.

#### ■ 6.2.4 Experimentální určení rozlišovací schopnosti 3D měření

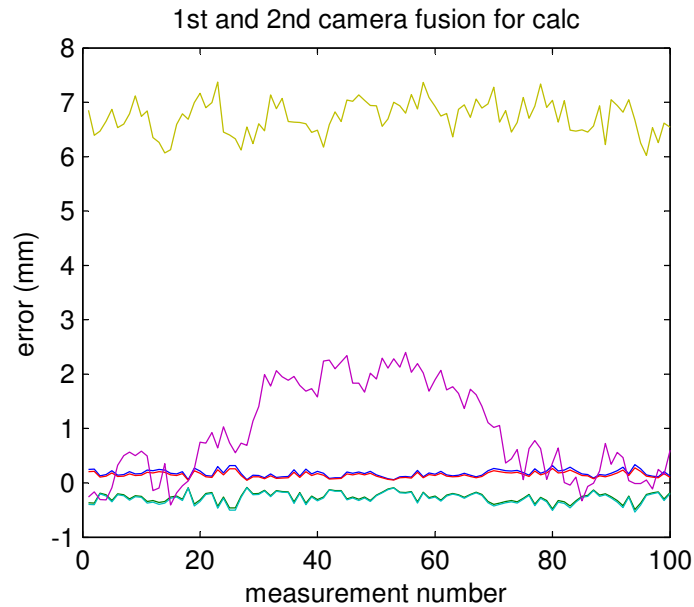
Podobně jako u určování těžiště jsem provedl pokus, při kterém jsem se pokusil určit rozlišovací schopnost určení polohy objektu v prostoru. Při tomto testu jsem měřil polohu LED pevně umístěné na vozíku lineárního posuvu. Tímto vozíkem jsem posouval po 0.025 mm v 500 krocích, celkový posun po přímce byl 12.5 mm.

Vozík se posouval ve směru osy  $y$ , ve směru ostatních os nebyl proveden žádný pohyb. V osách  $x$  a  $z$  byla naměřená hodnota:

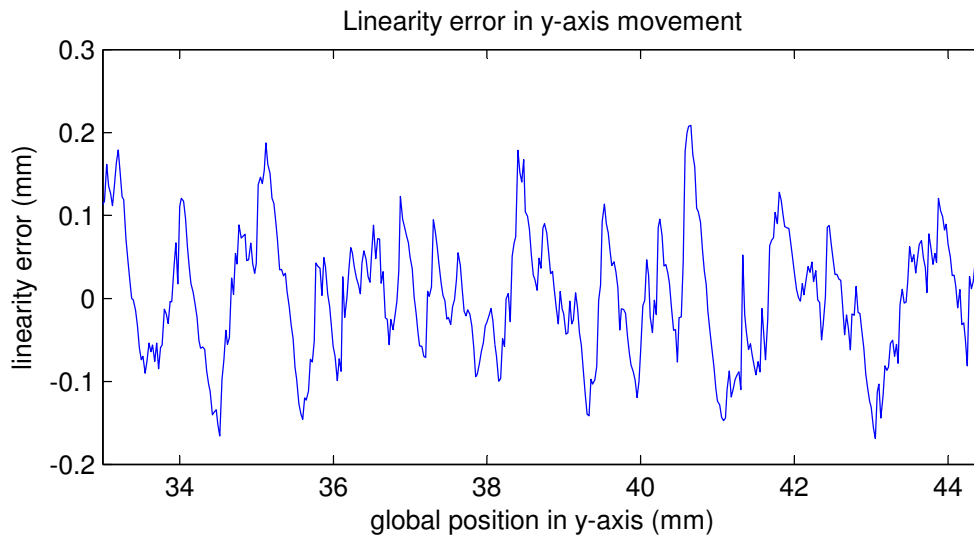
$$x = 53.20 \pm 0.06 \text{ mm}, z = -0.04 \pm 0.18 \text{ mm}$$

V ose  $z$  je větší standardní odchylka pravděpodobně způsobena dvěma vlivy – všechny kamery sledují objekty shora a kalibrační přípravek je planární v rovině určené osami  $x$  a  $y$ .

Pro zpracování pohybu po ose  $y$  jsem naměřená data proložil přímkou ve smyslu nejmenších čtverců a zobrazil chybu linearity (viz. Obrázek 6.14). Standardní odchylka této chyby je 0.08 mm.



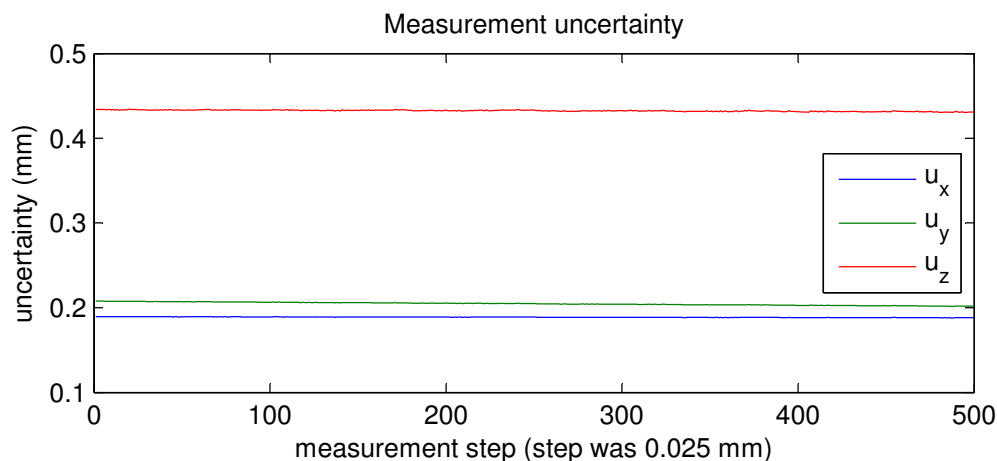
**Obrázek 6.13.** Chyba kalibrace objevená přeřčeností systému (fialová a žlutá křivka odpovídají chybám 3. kamery, výrazně nenulová střední hodnota svědčí o špatně provedené kalibraci této kamery)



**Obrázek 6.14.** Chyba určení polohy v prostoru při lineárním pohybu

### ■ 6.2.5 Nejistoty při prostorovém měření

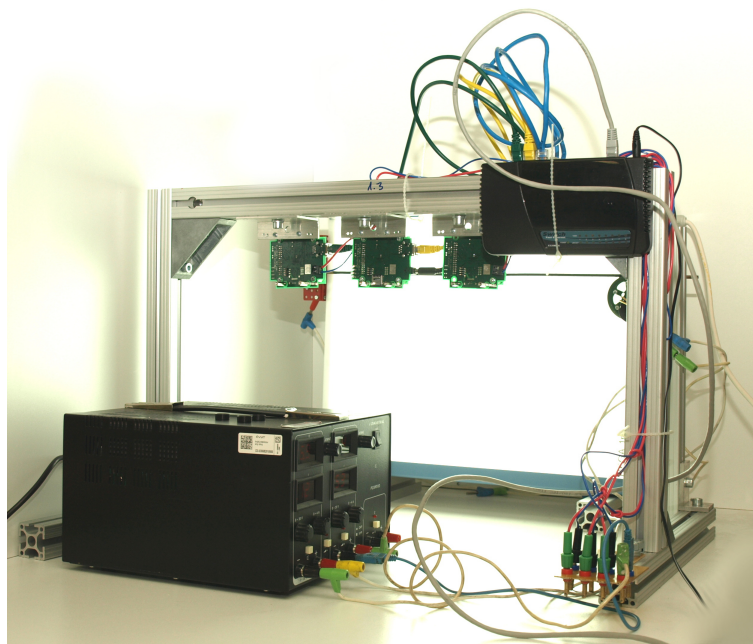
Pro měření popsané v sekci 6.2.4 jsem vypočetl nejistoty měření podle postupu popsaného v sekci 4.4. Výpočet těchto nejistot probíhal v Matlabu s využitím symbolického toolboxu a byl velmi pomalý, což bylo způsobeno i složitostí vzorců. V tomto výpočtu se zanedbávají veškeré chyby v umístění kamer, jediné započítané chyby jsou chyby určení polohy těžiště v obraze. Nejistoty těchto poloh jsem určil na 0.25 px, ze kterého se vypočítává poloha. Průběhy jsou vyobrazeny na Obrázku 6.15.



Obrázek 6.15. Nejistoty při lineárním pohybu v 3D měření

### 6.3 Měření po liniích ve 2D

Měření 2D polohy po liniích má za úkol např. sledovat klepající se výrobní linku. Mnou navržený systém pro 2D měření určuje polohu černé stopy vůči bílému pozadí. Pro sestavení pokusu jsem využil zpětné osvětlení, před kterým byl upevněn přípravek na simulaci klepání linky. Na tuto sestavu byly namířeny kamerové senzory. Sestavený pokus je vidět na Obrázku 6.16.



Obrázek 6.16. Pokus pro 2D měření po liniích

Kamery byly umístěné 122 mm od sebe a byly vzdálené 220 mm od přípravku simulujícího klepání linky. Byly postavené na výšku, jejich zorné pole tak zabíralo 79 mm široké části přípravku, mezi jednotlivými částmi sledovanými kamerami byla mezera 43 mm. Poloha byla určována s přesností na 0.5 obrazového bodu *binnigovaného* obrazu, což odpovídalo vzdálenosti 1.32 mm.

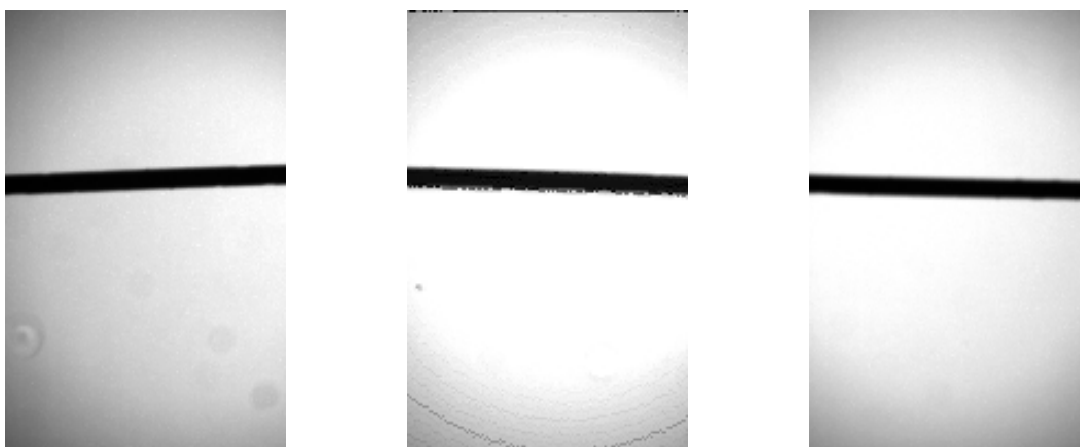
Simulační přípravek byl tvořen dvěma motory a tyčí, která byla na oba motory excentricky připojena. Motory se od sebe lišily převodovým poměrem převodovky, proto

se pohyb na první pohled jevil jako nepravidelný. Motory byly napájeny z laboratorního zdroje s možností regulace napětí, kterou jsem využíval k nastavení rychlosti běhu přípravku.

Pro demonstraci funkčnosti systému jsem provedl dva pokusy rozdílné v rychlosti pohybující se referenční tyče.

### 6.3.1 Určení nulové polohy

Po umístění modulů jsem ze všech kamer vyčetl snímek zastaveného simulačního přípravku. Tyto snímky jsou zobrazeny na Obrázku 6.17. V ideálním případě by na sebe měly navazovat, to ale není splněno. Důvodem je nepřesné umístění kamer, zejména jejich úhlové natočení podle optické osy objektivu.



Obrázek 6.17. Obrázky tyče v klidu

Tuto chybu je třeba korigovat. Každý senzor měří polohu linie v 11 místech, pro každé toto místo jsem určil nulovou polohu, kterou jsem získal měřením tyče v klidu. Vůči této poloze pak budu určovat polohu přípravku.

Tímto se dopustím chyby, ale ta bude zanedbatelná, protože rozdíly v úhlech natočení jsou malé. Správná vzdálenost vůči této poloze by se vypočetla přenásobením získané hodnoty výrazem  $\cos \alpha$ , kde  $\alpha$  je úhel natočení vůči rovině, vůči které chceme měřit. Je-li  $\alpha$  malá, můžeme použít aproximaci  $\cos \alpha \approx 1$ .

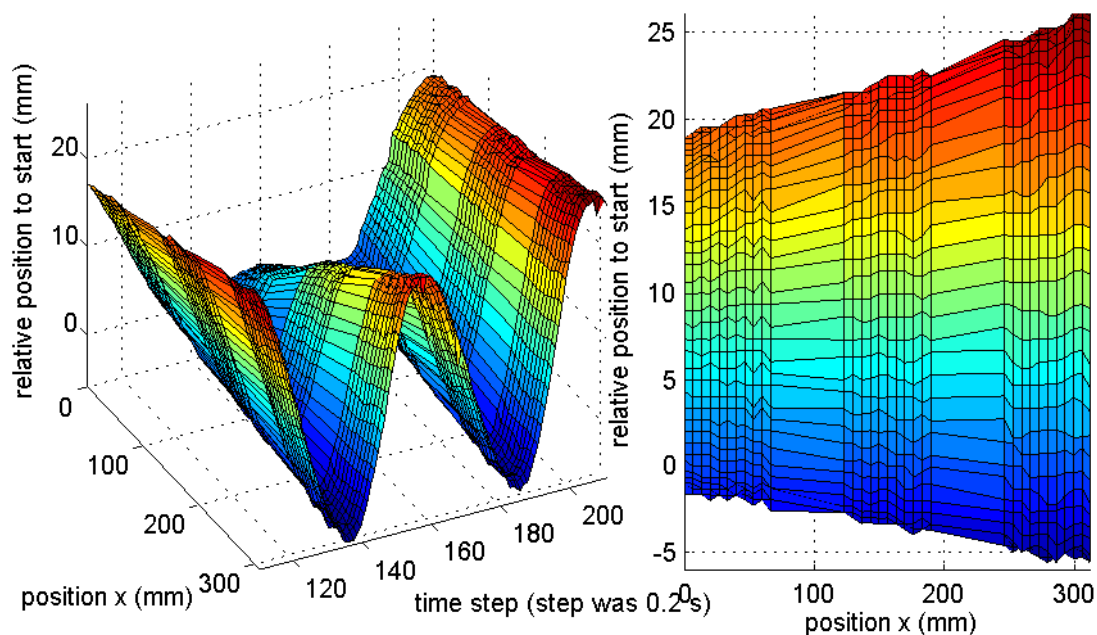
Pro kalibraci jsem provedl 100 měření nehýbající se tyče, nulové body jsem určil jako průměr změřených hodnot.

### 6.3.2 Měření odchylek od základní polohy při malé rychlosti

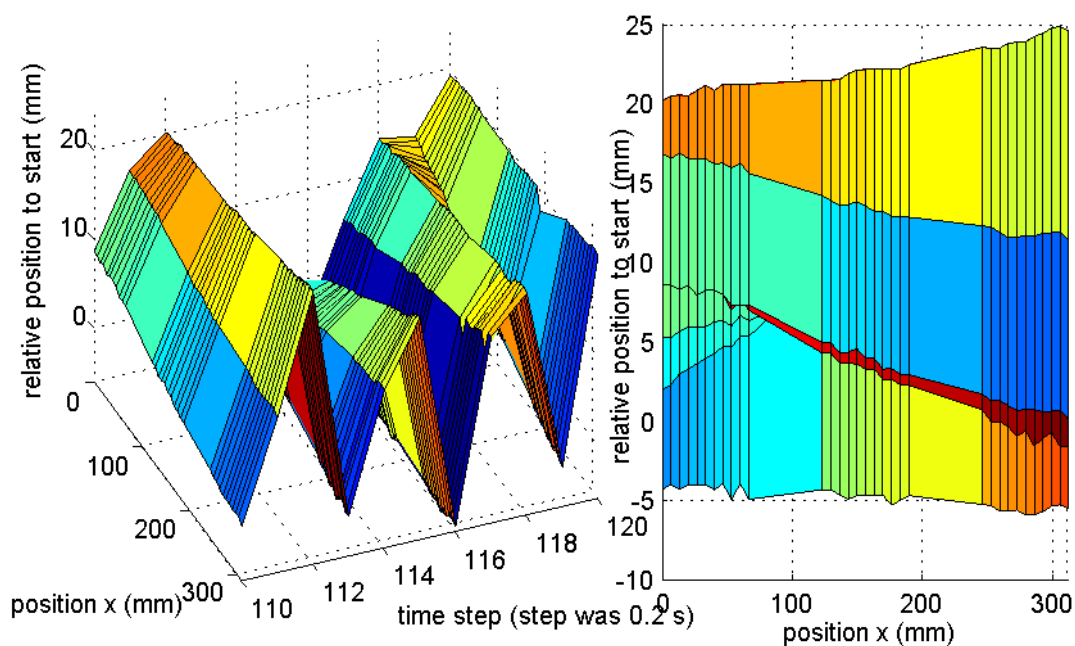
Při malé testovací rychlosti jsem nastavil rychlost pohybu přípravku na nejnižší, jakou použité motory umožňovaly. Pro zobrazení naměřených hodnot jsem využil vytvoření 3D plochy, kde ve směru osy  $z$  je naměřená hodnota, směr  $x$  určuje místo, ve kterém bylo měření provedeno a třetí směr je využit pro vyjádření časového průběhu. Tento graf je vyobrazen na Obrázku 6.18. Tento pokus sloužil především k principiální demonstraci funkčnosti systému.

### 6.3.3 Měření odchylek od základní polohy při velké rychlosti

Další měření bylo provedeno při výrazně vyšší rychlosti pohybu simulačního zařízení. Cílem tohoto měření bylo ukázat, že kamery jsou synchronní a naměřené hodnoty



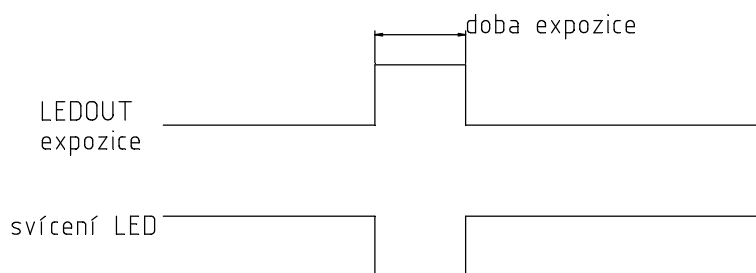
**Obrázek 6.18.** Naměřené polohy při pomalém pohybu přípravku (vpravo pohled na časoplošnou plochu z boku)



**Obrázek 6.19.** Naměřené polohy při rychlém pohybu přípravku

z více kamer určují přímkou. Naměřené hodnoty byly zobrazeny ve stejném smyslu jako v sekci 6.3.2 a jsou vyobrazeny na Obrázku 6.19.

Výsledkem tohoto měření je, že nelze vyloučit správnou synchronizaci kamer. Naměřené body při zohlednění nejistot měření tvoří přímkou. Rychlost pohybu referenčního



**Obrázek 6.20.** Svícení LED při synchronizačním pokusu

přípravku ale nebyla dost velká, aby se případná špatná synchronizace projevila<sup>1)</sup>). Z tohoto důvodu byl navržen nový pokus, kterým by se měla přesnost synchronizace ověřit (viz. sekce 6.3.4).

#### ■ 6.3.4 Ověření synchronizace kamer

Pro ověření synchronizace kamer kamery sledovaly LED, které byly ovládané pomocí signálu z jedné kamery tímto způsobem: pokud neprobíhala expozice, LED svítily, pokud expozice probíhala, LED nesvítily. Průběh je vidět na Obrázku 6.20.

Myšlenkou tohoto pokusu bylo, že v obraze kamery, kterou se ovládají, nebude LED vidět. Pokud se ostatní kamery synchronizují správně, nebude LED vidět ani v jejich obrazech.

Ukázalo se, že LED v obraze vidět jsou, nicméně jsou vidět ve všech, a to přibližně stejně intenzivně. Pro kontrolu jsem provedl měření ještě s LED trvale zapnutými. Trvale svítící LED jsou v obrazech vidět pětikrát intenzivněji než LED řízené podle schematu na Obrázku 6.20. Vzhledem k podobnosti naměřených dat z jednotlivých kamer bych řekl, že kamery nejsou nesynchronizované.

<sup>1)</sup> Změna polohy způsobená chybou synchronizace o 0.5 ms by byla menší, než chyby měření.



# Kapitola 7

## Závěr

V rámci této práce byl vytvořen mnohokanálový systém pro měření polohy s využitím obrazových senzorů založený na mikrokontrolérech rodiny STM32F4 a obrazovém senzoru MT9V032. Systém používá pro komunikaci a synchronizaci rozhraní Ethernet. Při práci na tomto systému jsem:

- vytvořil programy pro použité MCU
- navrhl průběh komunikace v senzorové síti
- vytvořil ovládací funkce pro Matlab

Při návrhu zpracování obrazu jsem bral v potaz omezení plynoucí z použitého hardwaru. Navržené algoritmy určují těžiště kontrastního objektu v obraze. Při návrhu algoritmů jsem provedl:

- ověření jejich funkčnosti v Matlabu
- implementaci do mikrokontrolérů

Pro mnohokanálový systém měření polohy jsem odvodil a navrhl:

- výpočet polohy ze změřených poloh objektu v obrazech
- kalibraci poloh kamer pomocí měření referenčních objektů

Funkčnost vytvořeného systému mnohokanálového měření polohy jsem demonstroval na pokusech. K provedení těchto pokusů jsem:

- navrhl a sestavil mechanické uspořádání
- vytvořil knihovnu pro Matlab pro ovládání lineárního posuvu
- připravil programy pro Matlab pro řízení průběhu jednotlivých pokusů

Pro ověření vlastností vytvořeného systému jsem provedl 9 pokusů, v rámci kterých byly dohromady provedeny tisíce měření. Pokusy byly rozděleny do tří částí:

- určování těžiště v obraze
- určování polohy objektu ve 3D
- měření polohy objektu ve 2D

### 7.1 Určení těžiště v obraze

V rámci testování určení těžiště v obraze jsem provedl tato měření:

- test opakovatelnosti měření
- test rozlišovací schopnosti určení těžiště
- měření geometrické chyby objektivu

Při testu opakovatelnosti měření se ukázalo, že šum polohy těžiště objektu v obraze je výrazně závislý na poloze, což je způsobeno relativním umístěním objektu vůči jednotlivým pixelům senzoru (viz. sekce 6.1.1).



Test rozlišovací schopnosti (viz. sekce 6.1.2) ukázal, že přesnost určení těžiště použitými kamerami je nejhůře  $\pm 1$  px plného rozlišení senzoru (tj. WVGA, 752x480 px).

Geometrická chyba objektivu nebyla ve výpočtech korigována, a proto bylo provedeno její měření k získání představy o její velikosti. Naměřená relativní velikost chyby při linearizaci vůči středu objektivu je menší než 1%, což odpovídá 1.5 px plného rozlišení senzoru přes jeho užší stranu (viz. sekce 6.1.3).

## 7.2 Systém pro 3D měření polohy

Pro 3D měření polohy jsem provedl se sestavou pokusu popsanou v sekci 6.2:

- kalibraci pozic kamer
- ověření funkčnosti 3D měření
- experimentální zjištění přesnosti měření
- výpočet nejistot měření

Při kalibraci kamer (viz. sekce 6.2.1) došlo k odhadnutí jejich pozice a následnému iterativnímu zlepšování tohoto odhadu. Po provedené kalibraci byla nejhorší chyba (tj. vzdálenost bodu kalibračního přípravku od roviny určené z měření kamerou) 1.01 mm, což představuje více než 50-násobné zlepšení oproti provedenému odhadu. Tato byla pravděpodobně způsobena z velké části nepřesností kalibračního přípravku.

Ověřil jsem funkčnost 3D měření pomocí matematického kyvadla s nastavitelnou délkou závěsu. Tím byla otestována funkčnost systému pro obecný pohyb v prostoru (viz. sekce 6.2.2).

Pro experimentální zjištění přesnosti určení polohy jsem provedl měření lineárně se posouvající LED. Pro změřená data jsem zobrazil chybu linearity, která byla ve směru pohybu lepší než  $\pm 0.2$  mm (viz. sekce 6.2.4).

Pro lineární pohyb jsem vypočetl nejistoty měření (viz. sekce 6.2.5). V průběhu pohybu byly nejhorší hodnoty nejistot  $\pm 0.21$  mm ve směru osy  $x$ ,  $\pm 0.19$  mm ve směru osy  $y$  a  $\pm 0.43$  mm ve směru osy  $z$ . Vysoká nejistota ve směru osy  $z$  byla způsobena umístěním kamer – objekt byl sledován „shora“ a díky tomu byly kamery citlivější na pohyb ve směru os  $x$  a  $y$ .

## 7.3 Systém pro 2D měření polohy po liniích

Při 2D měření polohy po vybraných liniích se ukázala nutnost korekce nepřesného úhlového umístění kamer (viz. sekce 6.3.1). Po provedení této korekce jsem provedl 2 pokusy:

- měření s pomalu se pohybujícím simulačním přípravkem
- měření s rychle se pohybujícím simulačním přípravkem

V obou pokusech (viz. sekce 6.3.2 a 6.3.3) byla měřena výchylka od počáteční polohy s rozlišovací schopností 1.32 mm. Nižší rozlišovací schopnost byla způsobena měřením s využitím pouze jednoho řádku obrazu, tedy nižšího množství vstupních dat. U obou pokusů by bylo možné naměřenými polohami proložit přímkou, která by ležela uvnitř v rámci nejistot těchto měření.

Pro ověření synchronizace kamer jsem provedl pokus se snímáním LED řízené jako inverzní blesk (viz. sekce 6.3.4). Výsledky tohoto pokusu odpovídají tomu, že kamery snímají snímky synchronně.

## **7.4 Zhodnocení práce**

V rámci této práce byl vytvořen mnohokanálový systém pro měření polohy s využitím obrazových senzorů založený na mikrokontrolérech rodiny STM32F4 a obrazovém senzoru MT9V032.

Výsledky této práce potvrdily, že použitím kamerových senzorů zapojených do sítě a synchronizovaných pomocí protokolu PTP lze přesně měřit polohu kontrastního objektu.

## Literatura

- [1] ARM Limited: *Cortex-M4 Revision t0p1, Technical Reference Manual*, ARM DDI 0439C.
- [2] STMicroelectronics: *Reference Manual, RM0090*, Doc ID 018909 Rev 3.
- [3] Yiu J.: *The definitive Guide to the ARM Cortex-M3*, Elsevier, 2007.
- [4] Řípa R.: *Synchronizované obrazové snímáče*, diplomová práce ČVUT FEL, 2013.
- [5] Dunkels A., Wöstenberg L., Simons Ch.: *Raw TCPIP interface for lwIP*, dokumentace k lwIP\_v1.3.2.
- [6] National Semiconductor: *DP83848C PHYTER®*, May 2008.
- [7] Aptina: *MT9V034: 1/3-Inch Wide-VGA Digital Image Sensor*, Rev. A, 2008.
- [8] Třeštík J.: *Měřicí modul s binárním zpracováním videosignálu z CCD kamery*, diplomová práce ČVUT FEL, 2005.
- [9] Coufal V.: *Přesné měření polohy kamerami CCD a CMOS*, diplomová práce ČVUT FEL, 2011
- [10] Novák T.: *Síť obrazových senzorů s rozhraním Ethernet se synchronizací s využitím protokolu PTP. Programové vybavení pro použité řídicí procesory – STM32F2xx*, individuální projekt ČVUT FEL, 2013
- [11] Saro J.: *Spolupráce STM32F4xx s optoelektronickými senzory*, bakalářská práce ČVUT FEL, 2012.
- [12] Píša P.: *Uživatelský manuál k jednotce MARS 2*, 2004



# Příloha A

## Použité zkratky

SRAM	Static Random Access Memory (zde ve smyslu operační paměti mikro-kontroléru)
MCU	Micro-controller unit
DCMI	Digital Camera Interface
LED	Light Emmiting Diode
lwIP	Light-weight IP stack
PTPd	PTP device – knihovna implementující PTP v2 protokol
PTP	Precision Time Protocol
IEEE 1588	PTP
TCP	Transfer Control Protocol
UDP	User Datagram Protocol
PLL	Phase-Lock Loop – fázový závěs (zde využívaný pro násobičku hodinové frekvence)
ROI	Region of Interest — výřez z obrazu
VCP	Virtual Com Port
WVGA	Wide VGA
VGA	Video Graphics Array
ACK	Acknowledgement

# Příloha B

## Oživení hardwaru

Pro orientační kontrolu po naprogramování hardwaru může posloužit měření spotřeby celého zařízení. Při využití obrazového senzoru a zapojeném Ethernetu s připojeným kabelem má jednotka spotřebu přibližně 200 mA.

Všechny další informace se vztahují k procesorové desce verze 3, desce senzoru verze 2.

### B.1 Osazení hardwaru

#### B.1.1 Procesorová deska

V případě použití **Ethernetu** hardware nefunguje s krystalovým oscilátorem a je nutné použít **externí oscilátor** 50 MHz. Používaný oscilátor má značení ASE-50.000MHz-LR-T. V návrhu desky je ale chyba v pouzdře, je třeba plošný spoj upravit, aby mohl být oscilátor použitý. Druhou chybou desky je zapojení R14 a R13 (jsou zapojeny do země, měly by být do napájení). Přes tuto chybu ale vše funguje.

Pro funkci programů popsaných v této práci není nutné osazovat rezistory R21 až R25, konektor na SD kartu, konektory J1 a J2 a eventuelně tlačítka SW1 a SW2.

#### B.1.2 Deska CMOS senzoru

Desku CMOS senzoru je třeba osadit podle schematu. Odpory R9 až R12 určují adresu senzoru na I2C sběrnici. Všechn software, který jsem k těmto deskám vytvořil, předpokládá adresu senzoru 0xB8. Odpor R4 musí být osazen.

### B.2 Kontrola funkčnosti USB

Pro kontrolu funkčnosti nahrajte do procesorové desky projekt **STM32\_USB-Host-Device\_Lib\_V2.1.0\_usbtest** v konfiguraci STM324xG-EVAL\_USBD-FS. Program očekává použití externího oscilátoru 50 MHz. Pokud tomu tak není, je třeba upravit v souboru `system_stm32f4xx.c` nastavení, aby odpovídalo použitému oscilátoru nebo externímu krystalu.

Pro správnou funkci je třeba mít nainstalovány drivery pro VCP od ST (jsou na DVD v **firmware/vcp.zip**). Po zapojení naprogramované desky přes USB do počítače by se měla deska připojit na COM port. Deska by měla zprostředkovávat funkci echa. Nastavení možností sériového portu nemá na funkci vliv.

Pokud tato funkce nefunguje, velmi pravděpodobně je chyba v nastavení hodin procesoru.

## B.3 Kontrola funkčnosti CMOS senzoru

Pro kontrolu funkčnosti CMOS senzoru budeme využívat projekt **STM32\_USB-Host-Device\_Lib\_V2.1.0** se stejnými úpravami a konfigurací, jaká byla třeba k rozběhání USB.

Nejprve je vhodné zkontrolovat, zda program projde do hlavní smyčky (pomocí debuggeru). V případě, že by se běh programu někde zasekl, může to znamenat:

- neúspěch při nastavení hodin senzoru
- neúspěch při komunikaci se senzorem přes I2C (většinou způsoben špatnou adresou senzoru – nastavuje se v `cmos_hw.h`)

Poté je vhodné se připojit k desce přes USB z Matlabu. Využijte příkaz `seropen`, upravte ho tak, aby odpovídal použitý COM port. Tento skript vytvoří proměnnou `s`, přes kterou se přistupuje na VCP. Pro vyčtení 10 obrázků využijte příkaz:

```
readimages(s,10,true)
```

V případě, že Matlab nezobrazuje obrázky a zobrazuje pouze chybu komunikace, pravděpodobně nedošlo k vyčtení dat z CMOS senzoru v procesoru. Je vhodné tento postup opakovat s připojeným debuggerem a podívat se, kde je proces zaseklý. Častým problémem by mohlo být zaseknutí ve funkci `CMOS_StartExposure`, což by indikovalo chybu čtení pinu `LED_OUT` senzoru. Zkontrolujte, že je zapájený odpor `0R` (reference `R4`).

## B.4 Kontrola funkčnosti Ethernetu

Ethernet musí používat 50 MHz externí oscilátor, pájecí můstek naspodu desky musí být propojen tak, aby byl propojen `OSC_IN` a `PHY_SCLK`. Program pro využívání Ethernetu komunikuje s CMOS senzorem, proto je nutné mít CMOS senzor připojen k desce.

Do desky nahrajte projekt **STM32F2\_Ethernet\_87\_ptpTest1**, nezapomeňte zkontrolovat nastavení IP adresy a MAC adresy (v souboru `main.h`). Po připojení počítače k desce přes Ethernet je třeba zkontrolovat nastavení síťové karty počítače, aby odpovídalo nastavení v procesoru.

První kontrolou je využití funkce `ping` z příkazové řádky. Neměly by se ztrácet žádné `packety` a odezva by měla být stabilně nižší, než 1 ms.

V druhé fázi je možné připojit se k desce přes TCP a vyzkoušet obsažený echoserver na portu 49345. V Matlabu jsou na toto implementovány skripty `tcp_connect` a `tcp_echotest`, které jsou přesunuty do složky `obsolete`. Soubor `tcp_connect` je třeba upravit, aby měl správnou IP adresu. Následně zavolejte skripty v tomto pořadí:

```
tcp_connect
tcp_echotest
```

Skript `tcp_echotest` odešle desce řetězec `ahoj` a následně přijme to, co mu deska odpoví. Pokud jsou přijatá data shodná, vše funguje jak má.

## Příloha C

### Opakování pokusů k testování algoritmů

Pro opakování pokusů k testování algoritmů je třeba nahrát do procesorové desky projekt **STM32\_USB-Host-Device\_Lib\_V2.1.0**. Před začátkem práce s Matlabem je vhodné vyčistit workspace pomocí příkazu `clear all`. Pak připojit k desce přes USB a v Matlabu otevřít spojení s deskou pomocí `seropen` (případně zde upravit port, na kterém se nachází deska). Nakonec je třeba připojit se k jednotce MARS a vytvořit proměnnou `mars`, kterou se bude ovládat motor zapojený na pozici A.

```
clear all
seropen
mars=tn_MARS_connect('COM4')
```

#### C.1 Test opakovatelnosti měření

Pro test opakovatelnosti měření slouží skript `save_static_test` ve složce **processes**. Tento proces nasbírá data a uloží je do složky označené datem a číslem ve složce **data/static\_tests**. Dojde i k zobrazení naměřených hodnot v grafu. Zpracování obrazů probíhá přímo na počítači, je třeba nastavit `threshol` ve funkci `show_positions`.

#### C.2 Test rozlišovací schopnosti

Pro test rozlišovací schopnosti je použit skript `resolution_test`. Na začátku tohoto skriptu je vhodné nastavit parametry, které jsou popsány na začátku tohoto souboru. Stejně jako v minulém případě nastavte požadovaný `threshol` ve funkci `show_positions`, data budou uložena do složky označené datem a číslem ve složce **data/resolution\_tests**.

Pro zobrazení lineárních odchylek v ose y zavolejte skript `process_resol_test`.

#### C.3 Chyba geometrického zobrazení

Pro naměření chyby zavolejte skript `resol_test_grayfull`. Je vhodné nastavit parametry na začátku tohoto skriptu. Naměřená data se uloží do složky podobně jako v předchozím případě.

Generování lineární odchylky probíhalo pomocí skriptu `process_lens_test` ve složce **plots**.



## Příloha D

# Opakování pokusů k testování 3D měření polohy

Pro tyto pokusy byl v jednotlivých modulech software **STM32F2\_Ethernet\_87\_for3D**. Pro jednotlivé moduly bylo třeba překonfigurovat MAC a IP adresu. Požadavek na MAC je jen, aby byla různá, IP adresy byly nastaveny od 192.168.2.9 do 192.168.2.11 (šlo by použít i jiné, ale bude třeba upravit kódy pro otevření TCP spojení).

Před začátkem jakéhokoli měření je nutné se připojit k senzorovému systému, skript pro připojení 3 kamer je:

```
start_tcp_3
```

### D.1 Kalibrace systému

Pro kalibraci je třeba provést měření kalibračního přípravku, který musí být umístěn tak, aby všechny kamery viděly všechny kalibrační body. Pro ujištění, že tomu tak je, doporučuji nahrát USB aplikaci a vyčíst z ní obrazová data pomocí `readimages`.

Dále je třeba inicializovat parametry – nastavení odpovídající pokusu provedenému během této práce je uloženo ve skriptu `initParameters`. V tomto skriptu změňte umístění kamer tak, aby odpovídalo vašemu pokusu.

Dále je třeba mít připojen kalibrační přípravek s ovládacím softwarem přes sériovou linku, předpokládá se, že rozsvícení  $n$ -té LED se provede pomocí odeslání čísla  $n$  do kalibračního přípravku. Kód pro STM32F4 Discovery Kit je ve složce **calib-f4-disc**.

Celá kalibrace by se dala zapsat takto:

```
seropen %pripojeni ke kalibracnimu pripravku
start_tcp_3 %otevreni TCP spojeni s jednotlivymi sensory
initParameters %nacteni defaultniho nataveni parametru

%provedeni kalibrace
[cam_params, calib_data]=calibrate_cameras(s,...
    tcp1_input, tcp2_input, tcp3_input, cam_params,global_positions)
```

V proměnné `cam_params` budou uloženy zkalibrované hodnoty kamer, v proměnné `calib_data` budou naměřené hodnoty použité pro výpočet kalibrace.

### D.2 Měření polohy

K vyčtení naměřených dat ze senzorů slouží funkce `read_all_tcp`, která zajišťuje i konzistenci dat (všechna data jsou ze stejného času). Výpočet polohy zajišťuje funkce `calculcate_position`. Celý proces vyčtení dat a následného výpočtu vykonává funkce `perform_measurement`. Po zkalibrování polohy lze provést měření tímto způsobem:

```
empty_all_tcp %vymazani jiz prijatych dat
%provedeme 100 mereni polohy, zobrazime na novou figuru
[X,Xo]=perform_measurement(cam_params,..
    tcp1_input, tcp2_input, tcp3_input, 100, figure())
```

V proměnné **X** budou uloženy naměřené polohy, v proměnné **Xo** budou uloženy výchozí hodnoty, ze kterých se polohy počítaly.

### D.3 Měření rozlišovací schopnosti

Měření rozlišovací schopnosti systému pro 3D měření zajišťuje skript `linear_mesure`, který provede měření v 500 krocích po 0.025 mm. Před spuštěním je třeba mít provedenou kalibraci a otevřenou komunikaci s jednotkou MARS a s kalibračním přípravkem. Je-li kalibrace hotová průběh tohoto měření by se spustil takto:

```
mars=tn_MARS_connect('COM4')
linear_mesure
```

Po dokončení budou naměřená data uložena v proměnných **Xs** a **Xos**. Tento skript nikam neukládá, uložení proměnných je třeba provést ručně.

## Příloha E

### Opakování pokusů k testování 2D měření po liniích

Pro funkci systému na 2D měření po liniích je třeba nahrát do modulů program **STM32F2\_Ethernet.87\_for2D** ve stejném smyslu jako při měření polohy ve 3D. Funkce umožňující vyčítání dat ze systému jsou velmi podobné jako při 3D měření polohy. Vyčtení dat se provede takto:

```
start_tcp_3
```

```
save_500_2D_measure
```

Po dokončení běhu budou naměřená data uložena v proměnné **Xs**.

## Příloha F

### Knihovna pro ovládání jednotky MARS pro řízení motorů

Knihovna pro ovládání jednotky MARS byla vytvořena pro Matlab. Nachází se na DVD ve složce **matlab/tn\_MARSLIB**. Pro použití knihovny doporučuji vložit celou složku do kořenového adresáře Matlabu a zavolat následující příkazy:

```
addpath tn_MARSLIB
addpath tn_MARSLIB/motor_movements
```

Příklad – připojení k jednotce, homing motoru A a následně jeho pohyb na danou polohu, poté relativní pohyb:

```
mars = tn_MARS_connect('COMX') %za COMX dosadte port na kterém
                                %je připojena jednotka MARS
tn_MARS_home(mars,'A') %iniciuje homing motoru A
while tn_MARS_motorFinished(mars,'A') ~= 1 %počkáme, až motor dokončí
end                                         %pohyb
tn_MARS_goAbsolute(mars,'A',200)%popojeď na pozici 200
while tn_MARS_motorFinished(mars,'A') ~= 1 %počkáme, až motor dokončí
end                                         %pohyb
tn_MARS_goRelative(mars,'A',0.025) %posuň se ještě o 0.025 kroku
```

Jednotka MARS umožňuje ještě další možnosti ovládání, které nejsou v knihovně implementovány. Pro přímé posílání příkazů mohou posloužit funkce **tn\_MARS\_set\_cmd** a **tn\_MARS\_get\_cmd**.

# Příloha G

## Obsah přiloženého DVD

V kořenovém adresáři DVD se nacházejí tyto složky:

- **bp** – obsahuje zdrojové kódy k vygenerování tohoto souboru
- **firmware** – obsahuje zdrojové kódy pro mikrokontrolér
- **literature** – použitá literatura
- **matlab** – skripty a funkce pro Matlab
- **multimedia** – obrázky a videa, která vznikla v průběhu práce

Data k naměřeným pokusům jsou přístupná ve složce **matlab** ve formátu **.mat**.

### G.1 Firmware

Pro použitý hardware jsou určeny projekty **STM32F2\_Ethernet\_87\_\***<sup>1)</sup> (pro komunikaci přes Ethernet) a projekty **STM32\_USB-Host-Device\_Lib\_V2.1.0\***. Zbylé dva projekty jsou určeny pro STM32F4 Discovery Kit.

### G.2 Matlab

Tato složka obsahuje skripty a funkce používané v prostředí programu Matlab. Ve složce **data** se nachází naměřená data z jednotlivých pokusů. Složka **obsolete** obsahuje zastaralé a nepoužívané skripty.

Pro možnost využívání funkcí ze všech složek je ve složce skript **initPath**, který slouží k inicializaci složkové struktury a měl by být volán, aby se daly používat všechny potřebné funkce.

---

<sup>1)</sup> Software pro Ethernet vznikl na základě práce v Laboratoři měřicích systémů Katedry měření FEL ČVUT v Praze. Využívá výsledky výzkumného týmu ve složení Ing. J. Breuer, doc. Ing. J. Fischer, CSc., doc. Ing. J. Roztočil, CSc. a Ing. V. Vigner.