ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

DIPLOMOVÁ PRÁCE

Zpracování obrazové informace pro kamerový dálkoměrný systém

Autor: Bc. Tomáš Krotil

Praha, 2015

Vedoucí: Ing. Jan Fischer, CSc.

Prohlášení

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.

V Praze, dne 5. 1. 2015 .....

.....

Podpis

Poděkování

Děkuji vedoucímu Ing. Janu Fischerovy za odborné vedení a Ing. Janu Roháčovi za vstřícný přístup. Neméně tak děkuji svým rodičům a blízkým za plnou podporu během studia.

Abstrakt

Tato práce popisuje zpracování obrazu s vysokým rozlišením v reálné aplikaci v reálném čase pomocí výkonného centrálního počítače. Cílem je zjistit vzájemné natočení a pozice jednotlivých kamer, komunikace s řídicím hardware a návrh metod na hledání kontrastních objektů v obraze i za různých světelných podmínek. Tato práce je součást většího projektu, proto vývoj probíhal podle požadavků projektu. K realizaci bylo využito kamer od firmy Allied Vision typu GT1290C a grafických knihoven WPF a Qt. Dále je v práci polemizováno nad různými typy osvětlovačů pro detekci bodů v prostoru, což je také testováno pomocí programu Matlab.

Abstract

This work describes image processing with high resolution pictures in real application and real time using high end central computer. Goal of this work is to find mutual angles of each cameras, position of each cameras, communication with control hardware and design of methods for finding contrast objects in image with not perfect light conditions. This work was part of bigger project, that's why development was driven by project demands. Cameras from Allied Vision type GT1290C was used for realization along with graphical frameworks WPF and Qt. There is polemics about different types of illuminators for detection of points in space, which is tested in Matlab.

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Tomáš Krotil**

Studijní program: Otevřená informatika (magisterský)
Obor: Počítačové inženýrství

Název tématu: **Zpracování obrazové informace pro kamerový dálkoměrný systém**

Pokyny pro vypracování:

1. Pro kamerový dálkoměrný systém vytvořte program pro zpracování obrazu, který určí polohu referenčních značek i polohu sledovaných objektů ve snímané scéně.
2. Navrhněte metodiku nastavení vzájemné úhlové polohy kamer s využitím přesného určení místa pomocného záměrného kříže ve snímaném obrazu.
3. Vytvořte aplikaci vyhodnocení obrazu pro řízení elektronické závěrky kamer a vytvořte program potřebný pro regulaci clony objektivu, nastavení jeho zaostření a zvětšení, který bude spolupracovat s blokem elektroniky řízení tohoto objektivu.

Seznam odborné literatury:

[1] Heijden, F.:Image Based Measurement Systems: Object Recognition and Parameter Estimation, ISBN 10: 0471950629, Wiley, 1995
[2] Hlaváč, V., Sedláček,.M.: Zpracování signálu a obrazu, skriptum, ČVUT- FEL, 2002
[3]Fischer J. Optoelektronické senzory a videometrie, skriptum, ČVUT, FEL, Praha, 2002

Vedoucí: doc.Ing. Jan Fischer, CSc.

Platnost zadání: do konce letního semestru 2014/2015

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 28. 1. 2014

Obsah

1 Úvod

Tato práce je součástí projektu kamerového systému pro dohled nad perimetrem pomocí systému 4 kamer. Tento systém má za cíl sledování objektů na obloze pomocí přehledové kamery a pomocí zbylých tří měřících kamer je měřena vzdálenost od báze. Pro tuto úlohu je nutné zjistit pozice jednotlivých kamer po rozložení měřícího systému a ovládání Hardware prostředků. Tato práce je zaměřena na vytvoření programu pro získání vzájemné polohy jednotlivých kamer, operátorského stanoviště přehledové kamery s ovládatelnou iris clonou, zoomem a ostřením. Cílem této práce je navrhnout metodiku zaměření středu sensorů samotných kamer, ovládání objektivu přehledové kamery, získávání a zpracování snímků ze všech kamer. Dále navrhnout komunikaci s řídicí jednotkou objektivu přes hlavní řídicí jednotku každé kamery, vytvořit aplikaci pro operátorské stanoviště a pro měření úhlu mezi jednotlivými kamerami. Cílem je též zjistit, jaký programovací jazyk a jaká grafická knihovna je nejlepší pro práci s velkým objemem rychle přicházejících dat.

Tato práce je součást většího projektu, na kterém spolupracují studenti Fakulty elektrotechnické s externí firmou. Díky tomu nejsou veškeré informace veřejně přístupné a byly konzultovány s vedoucími a lidmi z této firmy. Tato práce je podstatná část tohoto projektu, jelikož bez určení úhlů kamer od severu a pozice není možné řádně zaměřovat objekty a počítat výslednou pozici. Díky spolupráci více lidí je nutné v průběhu návrhu počítat s případnými změnami na výsledném programu.

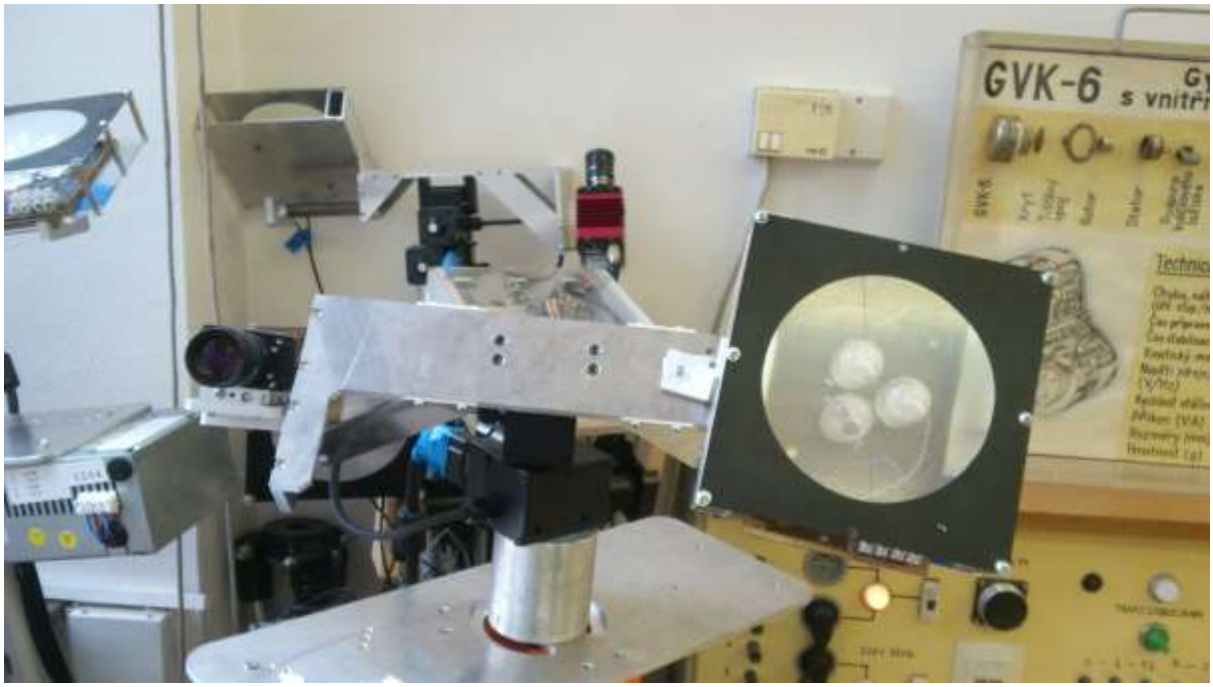
2 Analýza

Celková problematika tohoto projektu se skládá z více částí. Hardwarové vybavení, justáž, zastaničení, operátorské stanoviště, detekce objektů a měření vzdálenosti těchto objektů od základny. Hardwarové vybavení je návrh řídicích jednotek, stojanů, kamer, zdrojového napájení, centrálního počítače a komunikací. Justáž je proces, při kterém jsou odměřeny mechanické odchylky celého systému od hlavních os stojanů, toto je měřeno po výrobě jednotlivých stojanů. Výstupní údaje jsou nutné pro samotné zastaničení. Zastaničení je proces, který je aplikován po rozestavení stojanů na cílovou pozici. Při tomto procesu se zjišťuje pozice celého systému a vzájemná pozice kamer. Operátorské stanoviště je uživatelsky přístupný software, který umožnuje operátorovi ovládat odměr a náměr přehledové kamery, zoom, světlost obrazu a ostření. Měření vzdálenosti objektů od základny je proces, který běží kontinuálně a který vyhodnocuje výslednou vzdálenost objektů od základny pomocí měřících kamer.

Tato práce se zabývá zastaničením, operátorským stanovištěm a částečně hardwarovým vybavením, které je spojeno s hardwarem samotných kamer.

2.1 Hardwarové prostředky

Na Obr. 1 je vidět uchycení výsledného záměrného kříže a kamery na jednotce odměru a náměru. Toto je prototyp jednotky, kde jsou jednotlivé součásti připojeny pomocí kabelů. K této konstrukci je při provozu připojena řídicí jednotka, GPS s anténou, zdroj 12V napětí a inklinometr.



Obr. 1 Uchycení kamerového systému Zdroj: Autor

2.1.1 Specifikace kamer

V projektu jsou použity kamery GT1290C od firmy Allied Vision. Tato kamera je vybavena barevným sensorem Sony ICX445 EXview HAD CCD. Kamera komunikuje přes Gigabitový ethernet. Kvůli propustnosti přepínače jsou připojeny kamery přes optickou linku a každá optická linka je zapojena do vlastní síťové karty centrálního počítače. Na Obr. 2 je rozložení jednotlivých pinů kamery. Zajímavý pro tuto práci je hlavně pin číslo 5, který slouží jako opticky izolovaný výstup pro různé funkce. Dále kamera podporuje protokol IEEE 1588, který slouží k synchronizaci hodin.

Pin	Signal	Direction	Level	Description
1	Camera GND	In	GND for RS232 and ext. power	Ground for camera power supply, and RS-232
2	Camera Power	In	7–25 VDC	Camera power supply
3	Out 4	Out	Open emitter max. 20 mA	Output 4 opto-isolated (SyncOut4)
4	In 1	In	LVTTL max. 3.3 V	Input 1 non-isolated (SyncIn1)
5	Out 3	Out	Open emitter max. 20 mA	Output 3 opto-isolated (SyncOut3)
6	Out 1	Out	3.3 V LVTTL max. 50 µA	Output 1 non-isolated (SyncOut1)
7	Isolated IO GND	In/Out	Common GND for In/Out	Isolated input and output signal ground
8	RxD RS-232	In	RS-232	Terminal receive data
9	TxD RS-232	Out	RS-232	Terminal transmit data
10	Isolated Out Power	In	Common VCC for outputs 5–24 VDC	Power input for opto-isolated outputs
11	In 2	In	U_{in}(high) = 5–24 V U_{in}(low) = 0–0.8 V	Input 2 opto-isolated (SyncIn2)
12	Out 2	Out	3.3 V LVTTL max. 50 µA	Output 2 non-isolated (SyncOut2)

Obr. 2 Zapojení pinů kamery Zdroj: Allied Vision

2.1.2 CCD sensor

CCD sensor v kameře používá závěrku typu Global shutter, u které nevznikají při pohybu nepřesnosti v obraze způsobené například změnou osvětlení, na rozdíl od závěrky typu Rolling shutter. Tento sensor snímá obrázky v rozlišení 1280 x 960 s maximální rychlostí 33 snímků za vteřinu. Závěrka typu Global shutter funguje díky dodatečným pamětem u každého pixelu, kde se informace o jasu sensoru uloží a poté se postupně vyčítá z pamětí, na rozdíl od Rolling shutter, kde se údaje vyčítají přímo ze sensorů a díky tomu se liší čas vyčítání z dvou různých buněk.

2.1.3 IEEE 1588

Protokol pro časování a synchronizaci. Tento protokol běží na Ethernetu a jeho architektura je typu master-slave. Pro běh protokolu se nastaví jedna kamera, nebo jiné zařízení na stejné

Ethernet síti, jako časovací master. Tento master pak posílá multicastem jednotlivé synchronizační zpřavy. Tento mechanizmus je důležitý pro ostatní fáze projektu.

2.2 Operátorské stanoviště

Operátorské stanoviště je program, který využívá mechanicko-elektricky ovládané optické soustavy. K této práci je využita řídicí jednotka objektivu z bakalářské práce Bc. Andreje Čižmára, jejíž vývoj byl paralelní s touto prací, kvůli synchronizaci řídicího protokolu a testování.

2.2.1 Řídicí jednotka objektivu

Řídící jednotka objektivu je připojena k hlavní řídicí jednotce přehledové kamery přes RS232. Přiblížení, ostření a iris clona jsou v tomto objektivu realizovány pomocí motorků. Jejich řízení tedy není kompletně přesné a mělo by být řízeno pomocí obrazové informace. Objektiv je nutné řídit se zpětnou vazbou na dorazy objektivu pro možnost řídit relativní posuny a zároveň na přibližnou přesnou polohu, odpovídající vždy stejné pozici. Navržený komunikační protokol by tudíž měl mít možnost nastavení pozice absolutní i relativní.

2.2.2 Uživatelská přístupnost a možné ovládání

Operátorské stanoviště by mělo být vytvořeno tak, aby bylo jednoduše ovladatelné pomocí jedné ruky. Jako nejpřímější možnost ovládání připadá v úvahu klasické ovládání pomocí posuvníků a tlačítek plus a mínus. Tento přístup je dobrý pro neznalého uživatele, který aplikaci používá poprvé. Tento uživatel se nemusí učit specifický přístup k aplikaci. Nevýhodou je nepříjemné dlouhodobé ovládání, díky nutnosti neustálého klikání a přepínání režimů. Další možný přístup je použití klávesových zkratek a ovládání pomocí kolečka myši. Tento přístup je výhodný z hlediska rychlosti ovládání, avšak má problém v nepřehlednosti. Poslední možnost je využití USB joysticku. Tato možnost má velikou výhodu v přirozenosti ovládání a velký počet ovládacích prvků zajišťuje velkou možnost variability ovládání.

Nejlepší možnost se jeví kombinace těchto možností. Díky ovládání pomocí tlačítek je ovládání uživatelsky přístupné. Ovládání pomocí joysticku dává rychlost a přirozenost ovládání pokročilejším uživatelům. A ovládání pomocí kláves zajišťuje jemnost ovládání.

2.3 Proces zastaničení

Pro přesné měření pozic a vzdáleností objektů pomocí více kamer je nutné zjistit co nejpřesněji odchylky nulové polohy odměru a náměru od severu, vzájemné vzdálenosti jednotlivých kamer a celkovou polohu daného stativu. Pro tento projekt byl napsán algoritmus v Matlabu, který je nutné zavést do výsledné aplikace. Tento algoritmus ze vstupních parametrů

určených z justáže a zastaničení vrátí požadované hodnoty pro další fázi aplikace, která není cílem této práce, ale je to hlavní zaměření hlavního projektu.

2.4 Záměr kamery v obraze

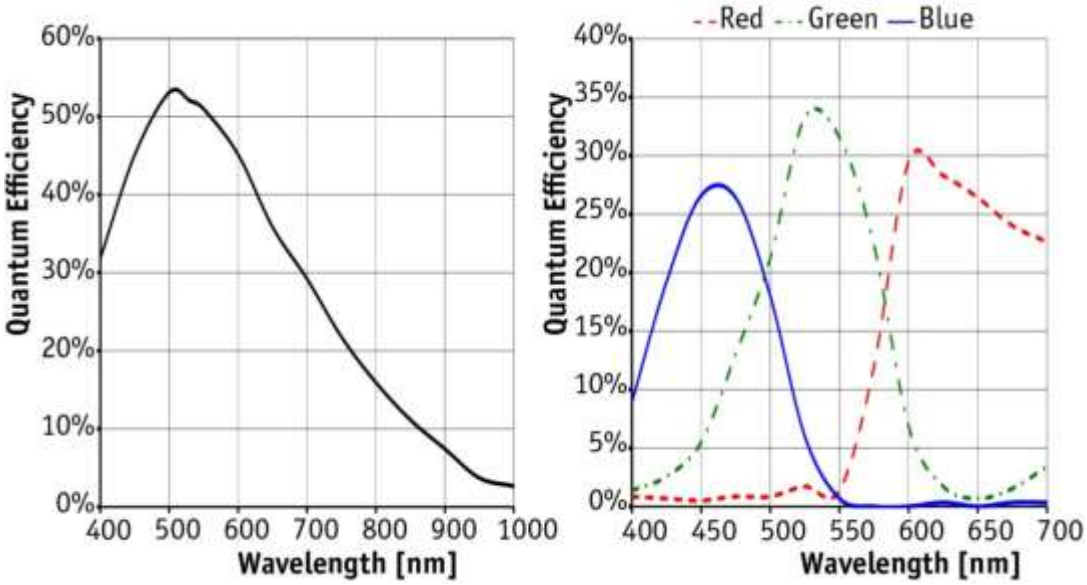
Pro zjištění vzájemného úhlu kamer je nutné zjistit jejich polohu v obraze každé z nich. Pro jejich přesné určení není vhodné používat detekci objektu kamery jako takové, jelikož by bylo vždy možný výběr z tří dalších kamer a nebylo by jednoznačně jasné, jaká kamera má být aktuálně zaměřena. Proto je nutné vytvořit systém záměrných značek, který bude možné vypínat a zapínat dle programové potřeby. Nabízí se použít programově řiditelný zdroj světla. Řiditelný zdroj světla má výhodu před detekcí například určitého obrazce v rychlejším a jednodušším zpracování a přesnější určení středu.

2.4.1 Záměrný kříž

Záměrný kříž je světelná značka, která označuje střed sensoru kamery. Její umístění záleží na pozici kamery na stanovišti.

2.4.2 Zdrojové světlo

Zdrojové světlo použité na záměrný kříž je z hlediska jednoduchosti ovládání a omezeném zdrojovém napětí nejlepší použít LED diody. Mělo by být vzato v úvahu, že záměrný kříž by měl být viditelný na vzdálenost 50 až 100 metrů, jelikož v tomto rozmezí by měly být rozestaveny samotné kamery. Dále je nutné vzít v potaz barevné jasové rozlišení kamery. V Obr. 3 je zobrazena charakteristika kvantové účinnosti sensoru pro jednotlivé barvy. Z obrázku je dobře vidět, že nejlepší účinnost je pro vlnové délky okolo 500nm, což je vlnová délka pro zelenou barvu. Nevýhodou zelených LED diod je fakt, že nemají dostatečnou účinnost na delší vzdálenosti. Dále je nutné brát v úvahu vyzařovací úhel LED diod. Pro tuto aplikaci jsou nejlepší LED diody s úzkou vyzařovací charakteristikou.



Obr. 3 Kvantová účinnost sensoru kamery Zdroj: Allied Vision

2.5 Komunikace s řídicí jednotkou

Řídicí jednotka každé kamery je napojena na RS422 a napojena do řídicího počítače a přeposílá požadavky do objektivu, zařízení odměru a náměru, GPS přijímače a zařizuje synchronizaci snímků z kamery a zařízení odměru a náměru.

2.5.1 Komunikační protokol

Komunikační protokol s jednotkami je založen na binárních příkazech posílaných přes RS 422. Každá kamera je připojena k vlastní řídicí jednotce. Všechny řídicí jednotky jsou připojeny do centrálního počítače, který obstarává komunikaci s jednotlivými kamerami.

2.5.1.1 Objektiv

Tab. 1 zobrazuje jednotlivé příkazy pro komunikaci do objektivu. Tab. 2 zobrazuje jednotlivé odpovědi z objektivu. Tab. 3 zobrazuje asynchronní zprávy z objektivu.

Byte Hlavička	**Byte param 1**	**Byte param 2**	**Byte param 3**	**Byte param 4**	**popis**
0x01	h	r	x	x	změna zoomu na hodnotu h (h jest rozsah zoomu dělený 255) a rychlost zoomu r, kde 1 je rychle, 2 pomalu
0x02	h	x	x	x	změna ostření na hodnotu h (h jest rozsah ostření dělený 255)
0x03	h	x	x	x	změna clony na hodnotu h (h jest rozsah clony dělený 255)

0x04	r	s	x	x	nastavení rychlosti pohybu clony r (0x00 až 0xFF) a směr otáčení clony s - 0xF0 otáčení po směru a 0x0F protisměru
0x05	x	x	x	x	všechny parametry se vrátí do původní polohy
0x06	p	x	x	x	parametr p se vrátí do výchozí polohy, p = 1 zoom; p = 2 ostření; p = 3 clona

Tab. 1 Příkazy do objektivu

Byte Hlavička	Byte param	popis
0x01	p	p označuje, na který parametr bylo odpovězeno

Tab. 2 Odpovědi z objektivu

Byte Hlavička	Byte param	popis
0x02	p	p určuje parametr, který aktuálně dojel do zábran

Tab. 3 Asynchronní zprávy z objektivu

2.5.1.2 Zařízení odměru a náměru

Tab. 4 zobrazuje jednotlivé příkazy na zařízení náměru a odměru Tab. 5 zobrazuje jednotlivé odpovědi ze zařízení náměru a odměru. Tab. 6 zobrazuje asynchronní zprávy chodící ze zařízení odměru a náměru. Zde je jedna zpráva, která je jediná nezávislá na příkazu z řídicího počítače. Tato zpráva je posílána vždy, když je započata expozice snímků z připojené kamery k řídicí jednotce. Toto může být použito na automatické přiřazení sériovým portům jednotlivým kamerám.

Byte Hlavička	2Byte param1	popis
0x13	x	Požadavek na inicializaci
0x83	p	Nastav relativní pozici odměru o hodnotu p
0x84	p	Nastav relativní pozici náměru o hodnotu p
0x91	x	Požadavek na odměr
0x92	x	Požadavek na náměr
0xA5	x	Požadavek na přesnost odměru
0xA6	x	Požadavek na přesnost náměru

Tab. 4 Příkazy do zařízení odměru a náměru

Byte Hlavička	param1	popis
0x13	s – 1B	Zařízení náměru a odměru je připravené po inicializaci, s udává status
0x83	s – 1B	s je status pohybu p relativní pozici odměru
0x84	s – 1B	s je status pohybu o relativní pozici náměru
0x91	o – 2B	o - pozice odměru v int16
0x92	n – 2B	n – pozice náměru v int16

0xA5	o – 4B	o – přesnost odměru v double
0xA6	n – 4B	n – přesnost náměru v double

Tab. 5 Odpovědi ze zařízení odměru a náměru

Byte Hlavička	param1	param2	popis
0x1F	n	o	n – náměr aktuálního snímku, o – odměr aktuálního snímku

Tab. 6 Asynchronní zprávy ze zařízení odměru a náměru

2.5.1.3 GPS

Tab. 7 zobrazuje příkazy pro GPS získávání dat. Zde se zavolá příkaz 0x0D s parametrem 1 a začne získávání dat, které přichází v asynchronní odpovědi vis Tab. 9. Odpovědi na vyžádané příkazy jsou v Tab. 8.

Byte Hlavička	param1	popis
0x0D	s – 1B	s = 1 – začít sběr dat z GPS, s = 0 – skončit sběr dat z GPS

Tab. 7 Příkazy pro GPS

Byte Hlavička	param1	popis
0x0D	s -1B	s – status začátku měření GPS

Tab. 8 Odpovědi z GPS

Byte Hlavička	param1	popis
0x0D	d = 64B	d – přijatá čistá data v NMEA formátu pro další zpracování knihovnou RTKLib

Tab. 9 Asynchronní zprávy z GPS

2.5.1.4 Příčný a podélný úhel základny

Podélný a příčný úhel základny je dotazován z paměti řídicích jednotek podle Tab. 10 a dopovědi přichází podle Tab. 11.

Byte Hlavička	Byte param	popis
0x22	x	Požadavek na poslední změřenou hodnotu náklonu základny

Tab. 10 Požadavky na příčný a podélný úhel základny

Byte Hlavička	param1	param2	popis
0x22	b – 4B	p – 4B	b – boční náklon v double32b, p – podélný náklon v double32b

Tab. 11 Odpovědi na příčný a podélný úhel základny

2.5.1.5 LED

Řídicí jednotka řídí rozsvěcení a zhasínání daných záměrných křížů podle Tab. 12 a dostává odpovědi podle Tab. 13.

Byte Hlavička	Byte param	popis
0x0A	s	s = 1 – požadavek na rozsvícení LED diody dané jednotky

Tab. 12 Požadavky na LED kříž

Byte Hlavička	Byte param	popis
0x0A	s	s – status rozsvícení

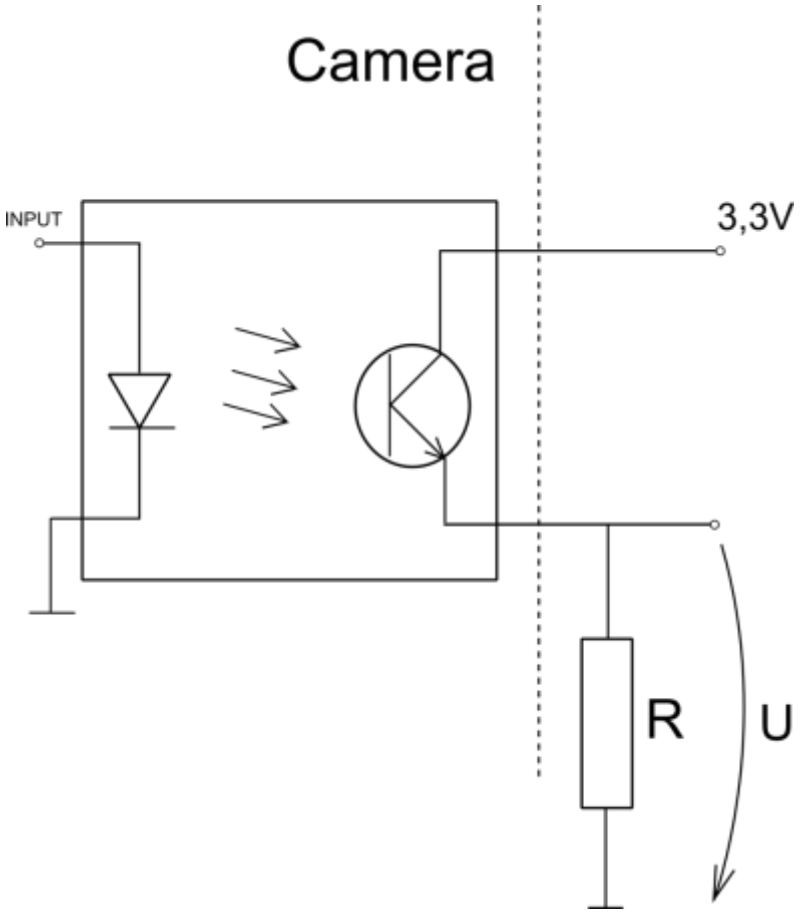
Tab. 13 Odpovědi od kříže

2.5.2 Synchronizace obrazu a pozice odměru a náměru kamery

V kapitole 2.1.1 Specifikace kamer je zmíněn pin číslo 5. Tento pin je vhodné použít pro synchronizaci obrazových dat a dat z jednotky odměru a náměru. V kameře je možné nastavit, aby na tomto pinu byla zvednuta logická úroveň na jedna, pokud je obraz exponován. Díky tomuto je možné zažádat na začátku expozice obrazu o data z jednotky odměru a náměru a následné poslání těchto údajů do centrálního počítače, který tuto informaci může nezávisle zobrazovat a případně párovat s obrazy.

Na Obr. 4 je schéma zapojení opticky odděleného členu. Toto schéma bylo navrženo pro vyvolávání přerušení na pinu procesoru. Při tomto přerušení jsou vyžádána data z jednotky odměru a náměru. Na schématu je vidět „input“, což je vnitřní pin procesoru, který řídí CCD sensor a jeho expozici a dává požadovanou hodnotu na výstup. Tato hodnota na výstupu rozsvěcí LED diodu. LED dioda osvětluje fototranzistor, který spíná 3,3V proud do zátěže R. Měřením napětí U se zjistí, zda je na „input“ logická 1 nebo logická 0. Pokud je dobře zvolen odpor R, pak je možné zapojit toto zapojení přímo na pin procesoru a snímat přerušení.

Optické oddělení se v tomto případě používá pro ochranu čipu v kameře. Při špatném zapojení, nebo zkratu bez galvanického oddělení může dojít k trvalému poškození čipu v kameře, jejíž cena může být značně vysoká.



Obr. 4 Schéma zapojení opticky odděleného členu Zdroj: Autor

2.6 C# nebo C++

Tento problém také požaduje hlubší zamyšlení a testování, jaký programovací jazyk použít na zpracování velkého množství dat v reálném čase. Jelikož je požadována rychlost snímání alespoň 20 snímků za vteřinu. Každý snímek je v rozlišení 1280 x 960, to znamená, že ze čtyř kamer přichází dohromady (1280*960*4*20*3)B, což je 294,912MB/s a pokud by nebyla obrazová informace dostatečně rychle čištěna z paměti, mohlo by rychle nastat k pádu samotného programu.

2.6.1 WPF C#

Výhodou použití C# je vysoká rychlost vývoje a pokročilá možnost objektově orientovaného návrhu. C# nabízí vývojový Framework WPF, který umožnuje velmi příjemný návrh grafického rozhraní s pomocí xaml a Visual studia 2012, které v sobě integruje Expression Blend. S tímto nástrojem je možné navrhnout grafické uživatelské rozhraní rychle za pomoci vzorů a vlastností, tak aby byl oddělen grafický design a funkční část programu. C# řeší návratové funkce přes takzvané delegáty, které slouží jako datový typ pro návratové funkce.

2.6.2 Qt Framework

Z C++ frameworků se jako nejlepší jeví Qt Framework, který staví dosti propracovanou nástavbu nad klasické meze C++. Qt je multiplatformní Open Source projekt. Qt má vlastní SDK, které se jmenuje Qt Creator, avšak je možné ho také vyvíjet pod Visual Studiem. Mezi přednosti patří systém signálů a slotů, díky kterému je možné velmi dobře navrhovat více vláknové aplikace pouze s použitím tohoto systému. Dále díky tomu že je Qt založeno na C++, má výhodu vůči C#, že si programátor může sám spravovat paměťový prostor, kterého může v jazycích pouze s garbage collectorem velmi rychle docházet.

2.7 Vimba API

Kamerový systém využívá takzvané Vimba API. Toto API slouží ke komunikaci s kamerovým systémem přes síť Ethernet.

2.8 Zpracování obrazové informace

Záměrný kříž na kameře musí být rozpoznán v obrazovém záznamu druhé kamery. Tohoto je dosaženo pomocí algoritmů na rozpoznávání obrazu. Obraz je rozpoznáván na černobílém obraze, kvůli potřebě rozpoznávat pouze jas. Existují také algoritmy pro rozpoznávání v barevném obraze, ale ty nejsou pro tuto aplikaci vhodné. Černobílý obraz je vlastně pouze matice s hodnotami od 0 do 255, kde 0 je absolutně černá a 255 určuje bílou jasovou barvu.

2.8.1 Lokalizace záměrného kříže

Když se rozpoznává záměrný kříž, tak se využívá algoritmů na prahování obrazu od určité hodnoty. Když se takováto matice rozpozná, tak je nutné s tímto prostorem dále pracovat, jelikož prahování nám pouze dává osvětlenou plochu. Pro další zpracování této plochy se používá algoritmu, který se jmenuje hledání těžiště obrazu. Tento algoritmus je stejný jako hledání těžiště deterministického objektu.

Prahování používá algoritmus, kde se zvolí prahová hodnota a hodnoty větší než tato hodnota jsou označeny jako 1 a hodnoty menší jsou označeny jako 0. Těžiště se hledá jejich průměrem v ose X a Y. $X = \frac{\sum x_i * val_i}{N}$ a $Y = \frac{\sum y_i * val_i}{N}$, kde x_i je poloha pixelu v ose x, y_i je hodnota v ose y, val_i je 1 pro buňku nad úrovní a N je počet pixelů s hodnotou 1. Tyto hodnoty jsou zaokrouhleny na celé pixely. Tímto se nalezne střed tohoto obrazce.

Druhý přístup vyhodnocuje prahování pouze jako redukci šumu po rozdílu obrazu s rozsvícenou LED a se zhaslou LED. Hodnoty větší než práh se zachovají stejné a hodnoty menší

než práh se vynulují. Poté se počítá těžiště průchodem celé matice v ose X a v ose Y pro nalezení součtu v obou osách. Součet se poté vydělí součtem všech hodnot v prahované matici.

$$D = obr_{br} - obr_{dk}$$

V této rovnici je D matice rozdílu, obr_{br} je obraz s rozsvícenou LED a obr_{dk} je obraz se zhaslou LED.

$$M = (\, if\ D(x,y) > tr)\ D(x,y)\ else\ 0\,)$$

V této rovnice je M výsledná matice po prahování, D je matice rozdílu. Tato rovnice vyjadřuje ořízntutou matici o šumové hodnoty vzniklé rozdílem dvou obrazů, které jsou pořízené rychle za sebou.

$$sum_x = \sum_{i}^{x_{max}} \sum_{j}^{y_{max}} M(i,j) * i$$

Tato rovnice vyjadřuje výpočet součtu jednotlivých složek x násobených vzdáleností od bodu 0,0. Matice M se prochází prvek po prvku.

$$sum_y = \sum_{i}^{x_{max}} \sum_{j}^{y_{max}} M(i,j) * j$$

Tato rovnice vyjadřuje výpočet součtu jednotlivých složek y násobených vzdáleností od bodu 0,0. Matice M se prochází prvek po prvku.

$$Z = \sum\sum M(i,j)$$

Tato rovnice vyjadřuje součet všech prvků.

$$res_x = round(\frac{sum_x}{Z})$$

Tato rovnice vyjadřuje výsledný pixel v ose x.

$$res_y = round(\frac{sum_y}{Z})$$

Tato rovnice vyjadřuje výsledný pixel v ose y.

První postup je rychlejší, ale je velmi náchylný na více osvícené plochy v záběru kamery a je u něj potřeba správně nastavit práh, jinak dochází k nepřesnostem. Má však výhodu v použité paměti a rychlosti. Druhý postup je pomalejší, ale je navržen tak, aby nedocházelo k nepřesnostem díky vnějšímu osvětlení. Počítá s tím, že mezi dvěma záběry se změní světlost pouze u LED, která je vypnuta a zapnuta pro jednotlivé obrazy.

Jelikož obě metody mají své problémy, proto je nejlepší použít první metodu a při zjištění moc velké světelné plochy po prahování použít metodu druhou. Výsledek měření bude výsledně potvrzen operátorem, a pokud nebude výsledek vyhovovat, bude provedeno nové měření.

2.8.2 Automatická regulace elektronické závěrky

Problém s přeexponovaným obrazem nastane, když na sensor dopadá za dobu expozice větší množství fotonů, než je citlivost sensoru. Kamera sama o sobě má metodu na auto expozici, avšak u ní může nastat problém s kontrolovanou regulací a hlavně při výpočtu těžiště diferenční metodou. Proto je možné použít vlastní metodu na regulaci expozice. Při každém měření se změří maximální hodnota osvětlení, a pokud je na maximální hodnotě sensoru, pak se sníží doba expozice a další obrázek bude mít za stejných světelných podmínek pravděpodobně menší maximum než limit sensoru, pokud ne, tak se proces opakuje, dokud není dosaženo menších maxim. Naopak pokud je maximum moc malé, tak se doba expozice zvýší, dokud není maximum v dostatečných hodnotách.

2.8.3 Přesnost měření úhlů kamer ve vodorovném a svislém směru

Přesnost měření zařízení odměru a náměru je $\frac{1}{1000}$ ° v náměru a $\frac{1}{100}$ ° v odměru.

Přesnost sensoru na pixel na 100m je dána vzorcem

$$\frac{a-f}{f} * y' = y$$

kde a je vzdálenost objektu ke kameře, f je ohnisková vzdálenost, y’ je velikost jednoho pixelu na sensoru a y je velikost objektu ve skutečnosti. Velikost jednoho pixelu na sensoru je 3,75um, po dosazení do rovnice je vzdálenost jednoho pixelu ve skutečnosti na vzdálenost 100m rovna 7,496mm. Pomocí goniometrické rovnice je výsledný úhel na pixel na vzdálenost 100m roven

$$\operatorname{atan}\left(\frac{y}{a}\right)$$

což je v tomto případě 0,004295°. Z tohoto dostaneme přesnost ve vodorovném směru 0,0052949° a ve svislém 0.0142949°.

2.8.4 Regulace ostření při zoomu

Při testování objektivu se projevil problém při změně zoomu. Obraz se rozostřuje při změně zoomu. Proto je vhodné navrhnout algoritmus pro pomocné ostření při změně zoomu. Takovýto algoritmus využívá takzvaný Sobel filtr pro kvantifikování hran v obraze. Pomocí Sobel filtru se vypočítá míra strmosti hran ve vodorovném a svislém směru v každém bodě, kromě krajních bodů pomocí konvoluce s maticí $\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$ pro osu x a s maticí $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$ pro osu y. Poté se jednotlivé body matice sečtou a výsledek udává míru ostrosti, která určuje míru strmosti v obraze. A pokud předpokládáme stále stejnou velikost obrazů, můžeme tento údaj použít pro regulaci ostrosti.

Tato metoda samozřejmě není perfektní. Její nevýhoda je, že počítá s ostrostí mezi každými pixely. Proto se pro hrany, které by měli být bližší, používá decimace obrazu, čímž se počítá s více průměrovanými pixely. Pokud se ale uvažuje, že objekty na které má býti ostřeno jsou ve větší vzdálenosti, rozlišení hrany na jeden pixel je užitečnější. To by měl být i případ této práce.

2.8.5 Detekce objektů na obloze

Pro detekci kontrastních objektů na obloze je vhodné použít podobný algoritmus jako pro detekci záměrného kříže. Jelikož obloha je vesměs jednolitá, je možné použít následující vzorce.

$$M = (\, if\ O(x,y) < tr)\ 1\ else\ 0\,)$$

Tento vzorec z obrazové matice O vybere pouze hodnoty menší než určitý práh tr. V matici M jsou poté hodnoty 1 pouze na pozicích, kde byl kontrastní tmavý objekt. Tato metoda má však úskalí ve zvolení daného prahu. Pokud předpokládáme detekci objektů na obloze, je možné použít následující vzorec pro vybrání prahu tr.

$$tr = \min(O) + (\max(O) - \min(O)) * 0{,}1$$

Toto řešení vybere deset procent minimálního rozdílu. Avšak tímto se dostal do řešení problém, pokud je maximum a minimum velmi blízké hodnoty. Proto je dobré detekovat tento případ a vyhodnotit ho, že žádný objekt není detekován. To už není problém, protože když v obraze je maximum a minimum skoro stejné, kontrastní objekt by se v obraze vyskytovat neměl.

3 Realizace

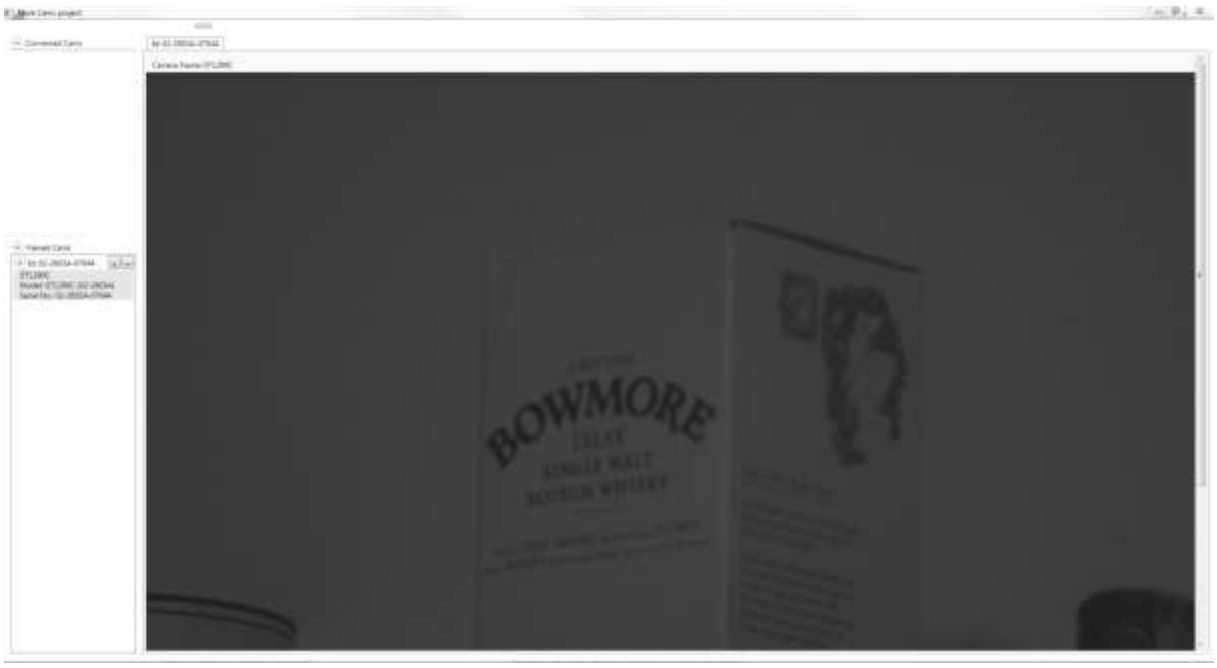
Realizace se skládá z vytvoření programu v C# pro ověření jestli je možné obraz bezproblémově zpracovávat v tomto programovacím jazyku. Navržení záměrného kříže a jeho zapojení do systému. Vytvoření programu v C++ s knihovnou Qt, který zajistí operátorské stanoviště, proces zastaničení a základní detekci objektů.

3.1 Testování rychlosti různých programovacích jazyků a grafických knihoven

V kapitole 2.6 byly rozebrány výhody a nevýhody použití programovacích jazyků C++ a C# s jejich grafickými knihovnami WPF a Qt. V této kapitole je rozebráno samotné testování těchto možností a je přidána i možnost použití importované dll knihovny do C# vygenerované z C++ aplikace.

3.1.1 Test zpracování a zobrazení obrazu v C#

Za pomoci dokumentace k Vimba API byl vytvořen program v C# ve WPF. Tento program je pouze testovací, ale na Obr. 5 je vidět, že byl schopen zaregistrovat připojení nové kamery a po přidání kamery do listu běžících kamer byla přidána záložka s obrazem této kamery.



Obr. 5 Vzhled testovacího software v C#

3.1.1.1 Struktura

Program je strukturován na několik základních objektů. Hlavní grafický objekt je StreamWindow, který obsahuje práci s grafickým prostředím a obsahuje ostatní objekty pro práci s Vimba API. Grafické prostředí bylo navrženo pomocí Extension Blend, který je součástí Visual Studia 2012. Toto prostředí se používá díky větší škále různých grafických prvků a možností jak s grafickým rozhraním pracovat. Grafické prostředí ve WPF se navrhuje v jazyce XAML, který je založen na XML a jedná se o značkovací jazyk, pomocí kterého se vytváří stromové struktury grafických prvků. Takto vytvořený soubor se za běhu aplikace načítá a grafické prvky se rozloží podle tohoto návrhu. Extension Blend se právě používá pro tvorbu XAML souborů, pro jednodušší a komplexnější návrh. Každý grafická třída ve WPF obsahuje soubory s koncovkou .xaml a .xaml.cs. První soubor obsahuje hierarchické rozložení a druhý obsahuje funkční logiku.

Vláknové operace se v C# nejlépe řeší přes takzvaný invoke. Jelikož se volá uvnitř funkce, může se odkazovat na proměnné funkce, či objektu.

```
Application.Current.Dispatcher.BeginInvoke((Action)(() =>
{
  try
  {
    if (cameraImage == null) cameraImage = new WriteableBitmap(width, height, 96.0,
96.0, PixelFormats.Gray8, pal);
    Int32Rect rect = new Int32Rect(0, 0, width, height);
    cameraImage.WritePixels(rect, fr.Buffer, width, 0);
  }
  catch (Exception e)
  {
    this.close();
    MessageBox.Show("Error: " + e.ToString());
  }
}));
```

Toto je příklad použití invoke, která je zavolaná uvnitř metody přijímající Frame fr.

Vimba API bylo v C# zaobaleno do několika pomocných tříd. Třída CameraFactory se stará o správu spuštěných kamer, jejich detekci a spuštění celého API. Dává k dispozici návratovou funkci, při změně listu připojených kamer. Dále pracuje se třídou wCamera, která zaštiťuje veškerou práci s kamerami samotnými. Převádí přijaté rámce na bitmapové obrazy pro jejich lepší zpracování. Dává k dispozici zapínání a vypínání jednotlivých kamer. A poskytuje návratové funkce s převedenými rámci na bitmapy, které jsou zobrazovány v grafické třídě.

3.1.1.2 Testování

Byla otestována rychlost aplikace na základním zobrazování obrázků ze dvou připojených kamer. Při zapojení dvou kamer přes komerční přepínač Ethernetu se projevoval jev, při kterém se ztrácelo až 70% snímků z druhé kamery. Tento jev byl viditelný pouhým okem.

Pro vyřešení tohoto problému byly kamery připojeny přes vlastní optické vlákno do centrálního počítače. Každá kamera je připojena do dedikované síťové karty a mezi síťovými kartami je vytvořen virtuální síťový most, aby mohl fungovat protokol IEEE 1588, který zajišťuje synchronizaci času jednotlivých kamer vis kapitola 2.1.3. Po vyřešení tohoto problému bylo zjištěno, že aplikaci dochází paměťový prostor. Důvod pro docházení tohoto prostoru byl ve zpracování a zobrazování obrazu, kde Vimba Api jakožto C# zaobalovač C++ knihovny se stará o mazání svých ukazatelů na jednotlivé snímky. Avšak pro zobrazení v aplikaci bylo nutné obraz zkopírovat do zobrazitelného objektu, aby nebyl zahozen, dokud není vytvořen nový. Díky tomu docházelo ke hromadění obrazů v paměti, které nebylo možné manuálně smazat. Jediné řešení bylo přepisovat stále stejný objekt, ale nevýhoda toho byla v nemožnosti posílání těchto objektů do vlastních vláken pro asynchronní zpracovávání obrazu. Navíc problém s rychlostí zpracování stále přetrvával.

3.1.2 Test generování C++ knihovny dll a import do C#

Tato možnost byla zvažována díky již naimplementované grafické knihovně v C#. Pro toto testování byl vytvořen projekt pro C++ generování dll knihovny. Knihovna samotná byla testována v předchozí C# aplikaci. Import dll knihoven generovaných z C++ do C# je časově relativně nároční, jelikož je nutné v samotné C# aplikaci nutné dll knihovnu zabalit proměnou po proměnné. Byl problém s předáváním větších struktur a objektů, což vedlo k nutnosti znovu generovat obraz v hostitelské aplikaci , čímž docházelo k dvojitému generování obrazu a ve výsledku byla tato možnost ještě pomalejší, proto se od ní záhy opustilo.

3.1.3 Test zpracování a zobrazení v C++ Qt knihovně

Jelikož z těchto kamer byl velký tok dat a C# neposkytoval dostatečnou kontrolu nad pamětí, bylo následně testováno, jestli je možné tuto aplikaci navrhnout v C++. Pro podobnost implementovaných objektů se C# bylo rozhodnuto pro Qt mezi MFC a Qt. Rozhodnutí bylo právě mezi těmito dvěma C++ knihovnami kvůli platformě Windows a poskytnutým příkladům práce s Vimba API pro obě tyto knihovny.

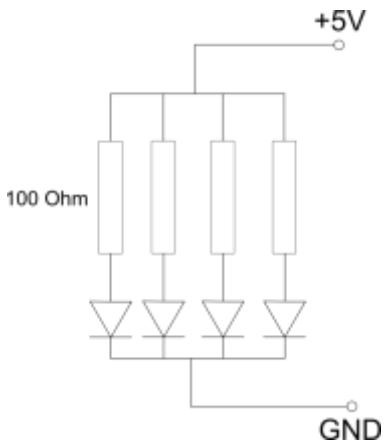
Byla napsána jednoduchá aplikace, se stejnými vlastnostmi jako aplikace v C#. Vyskytl se však znovu problém s přetékáním paměti, i když tento problém byl po mnohem delší době, než u C#

aplikace, stále k němu nastávalo. Naštěstí tento problém byl vyřešen pomocí použití inteligentních ukazatelů, které počítají reference na svůj objekt a pokud objekt není dále referencován, je obsah tohoto ukazatele smazán. V Qt se tento inteligentní ukazatel jmenuje QSharedPointer.

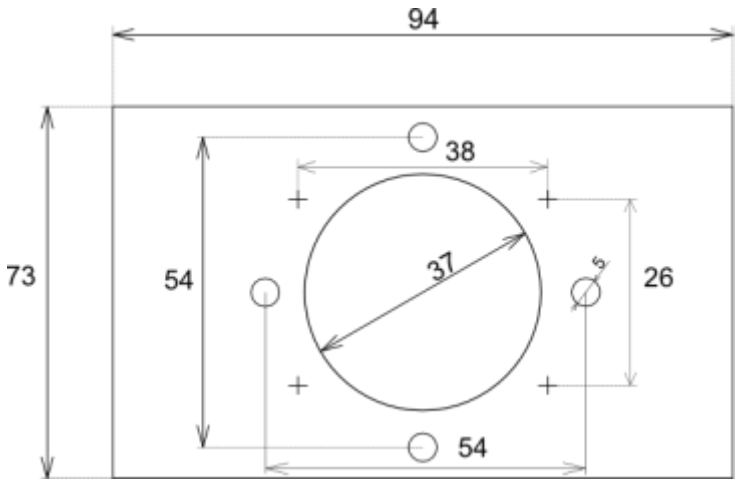
Po vyřešení těchto problémů byla další aplikace vyvíjena pouze v Qt Framework. Podrobnější popis dalších částí projektu následuje v příštích kapitolách.

3.2 Realizace záměrného kříže

Záměrný kříž byl navržen nejprve v základním provedení s běžnými LED diodami. V Obr. 6 je vidět zapojení tohoto záměrného kříže. Byly použity červené LED diody s napětím 3,2V a proudem 20mA, za těchto parametrů produkují světelný tok o síle 20000-35000mcd v 15° rozpětí. Bylo uvažováno napájeni z 5V zdroje. Na Obr. 7 je nákres zaměření děr pro objektiv kamery a děr pro LED. Křížem jsou zobrazeny úchyty, kruhem díry pro LED a objektiv. Samotné zpracování je vidět na Obr. 33 a Obr. 34.



Obr. 6 Základní zapojení LED



Obr. 7 Zaměřený záměrného kříže pro umístění na kameře

Tento záměrný kříž byl poté testován v chodbě na vzdálenost přibližně 100m. Obraz byl pořizován černobílý, pro zpracování v Matlab skriptech (vis 3.3Skripty Matlab), které byli navrženy pro testování navržených algoritmů a pro testování světelných zdrojů.

Výsledek testu záměrného kříže je vidět na Obr. 8. Už z tohoto obrázku je vidět, že LED není moc výrazná. Po otestování v Matlabu bylo zjištěno, že horní část obrazu je více světlá, než samotná LED. Proto bylo nutné udělat další sadu testů s LED různé barvy a svítivosti.

Obr. 8 Záměrný kříž první test

Při dalším testu se zaostřily optiky kamer na nekonečno, jelikož by to měl být jejich výchozí stav a měřicí kamery nemají měnitelnou optiku. Byly testovány 4 LED, červená, modrá, zelená a bílá. Testovány byly nejvíce zářivé LED, které byli běžně dostupné. Výsledné obrázky z jednotlivých pokusů jsou vidět na Obr. 35 až Obr. 38.

Barva	**Maximální Intenzita**
Červená	198
Modrá	231
Zelená	223
Bílá	255

Tab. 14 Intenzita jednotlivých LED

Z Tab. 14 je vidět, že nevětší intenzitu má LED bílá, dokonce takovou, že sensor byl saturován. Zároveň je tímto skriptem vyhodnoceno základní prahování, pro určení představy, jaká plocha by asi byla výsledně prahována. Následují Obr. 9 až Obr. 12, které zobrazují jednotlivé diody po prahování.

Obr. 9 Červená LED po prahování

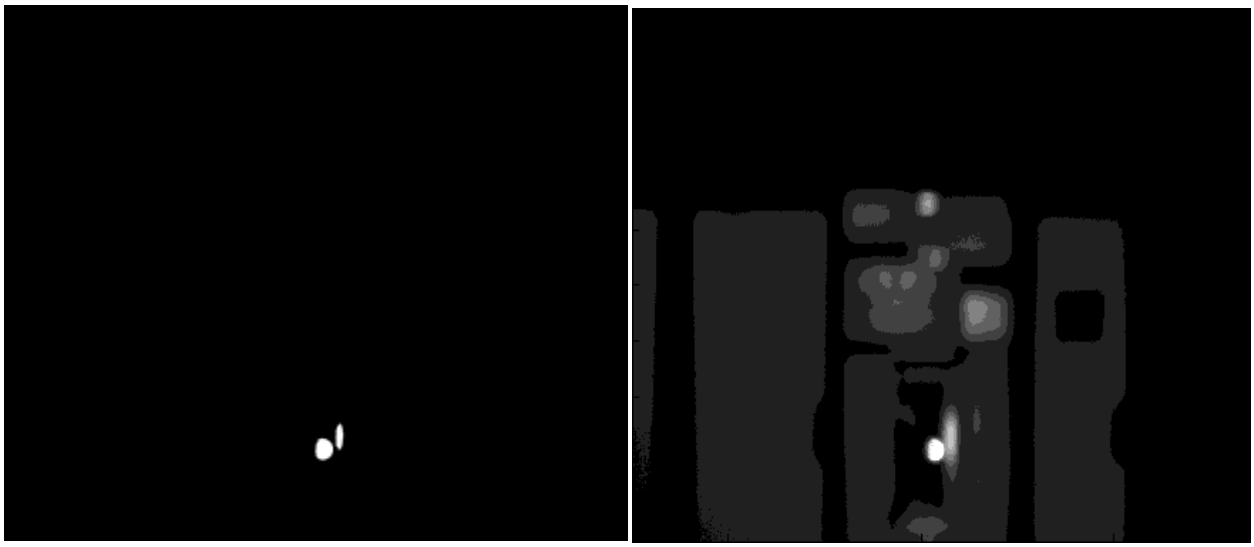
Na Obr. 9 je vidět, že červená LED po prahování má velmi nedostačující světlost. Na pravém obrázku je vidět, že při změně prahu se začínají projevovat ostatní osvětlené plochy, včetně ozářené plochy vlastní diodou.

Obr. 10 Modrá LED po prahování

Na Obr. 10 je vidět, že modrá dioda má dostatečné prahované pole a mohlo by v tomto případě dobře použito pro lokalizaci středu.

Obr. 11 Zelená LED po prahování

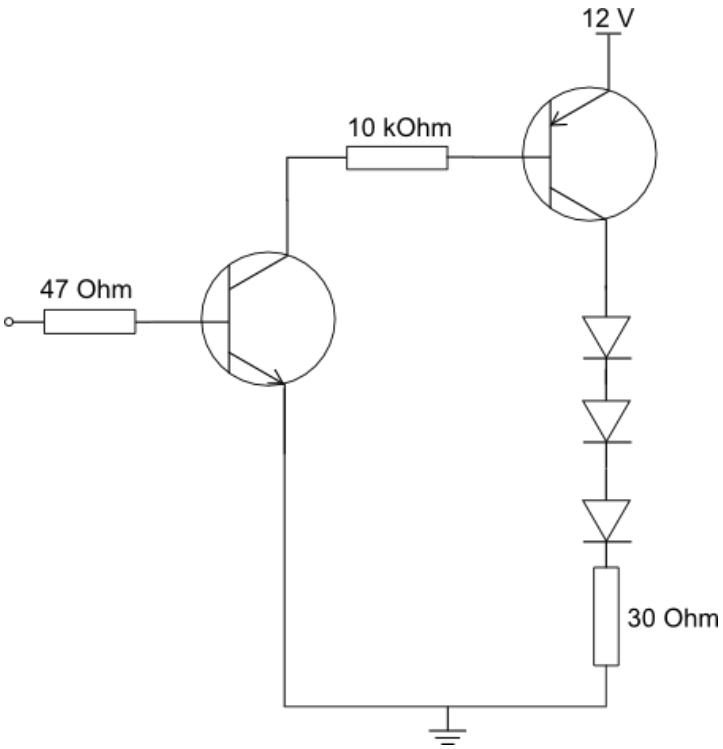
U zelené LED je vidět trochu horší charakteristika po prahování.



Obr. 12 Bílá LED po prahování

Bílá LED je nejlepší v intenzitě. Avšak je vidět, že dochází k odrazům v kraji obrazu.

Po těchto testech je vidět, že v tmavém prostředí jsou tyto diody dostatečné, avšak když se záměrný systém dostává do vnějších prostor, nastává k přesvětlení ostatními zdroji světla, proto bylo nutné vybrat jiný zdroj světla. Proto byly vybrány specifické LED pole s kolimátorem, který zaručuje 10° vyzařovací charakteristiku a 163lm. Více specifikace v [5]. Problém s těmito LED je takový, že odběr těchto LED je 700mA, což znamená, že by nebylo možné řídit obyčejným transistorem. Proto bylo nutné navrhnout zapojení s omezeným proudem, takové, aby bylo možné tyto záměrné kříže řídit. Dále bylo nutné toto zapojení vytvořit a zapojit do řídicí jednotky.

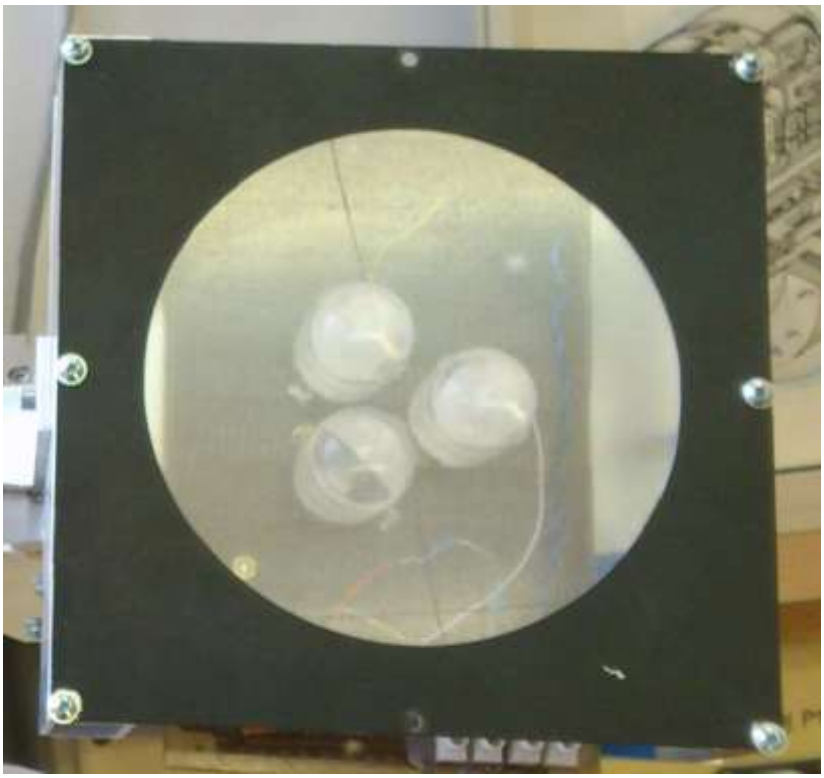


Obr. 13 Schéma finálního záměrného kříže

Na schématu Obr. 13 je zobrazeno výsledné zapojení 4 LED polí s kolimátorem. Každé LED pole obsahuje 16 malých plošných LED, nad kterými je kolimátor, usměrňující do 10°. Pro jejich úspěšné zapojení je nutné omezit proud, jelikož by nebylo možné, aby PNP transistorem procházelo 700mA. Proto se proud zapojením odporu 30 Ohm omezí na 110mA, což by měl PNP transistor typu KD140 zvládnout. Tento transistor je však výkonový, proto nemůže být spínán přímo pinem procesoru. Z tohoto důvodu je na jeho bázi zapojen transistor NPN, který zajišťuje spínání transistoru PNP, který spíná celou soustavu LED.

Vyzařovací charakteristika těchto LED je čtvercová, díky čtvercovému poli malých LED diod. Proto je nutné správně nasměrovat jednotlivé kolimátory, jinak by nevyzařovali na stejný obrazec. Jelikož se napájení, zem a spínací pin připojují kabely k řídicí jednotce, tak se při otáčení záměrného kříže musí dávat pozor na zamotání kabelů.

Tento záměrný kříž je vhodný i pro první metodu, jelikož jeho svítivost je mnohonásobně větší než předem testovaných LED. Výsledný záměrný kříž je vidět na Obr. 14.



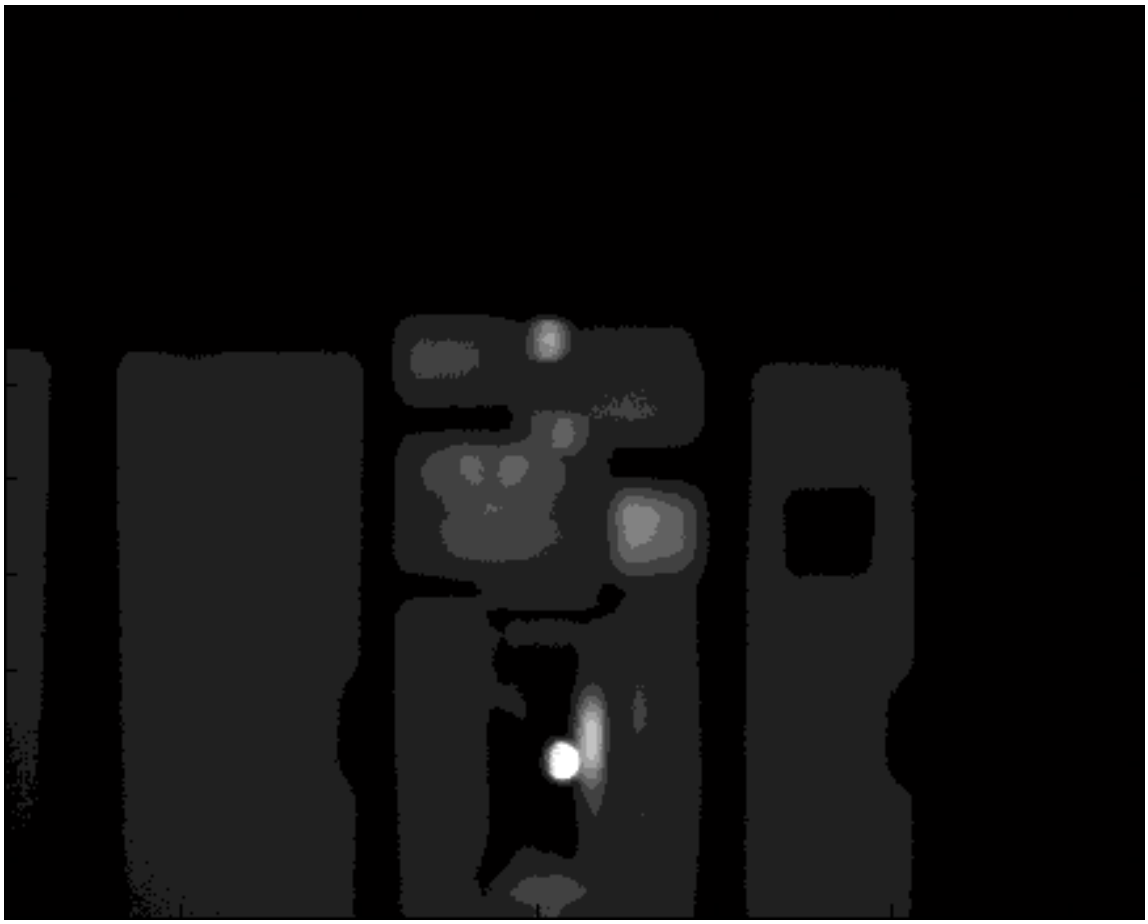
Obr. 14 Finální verze záměrného kříže

Tab. 14 Intenzita jednotlivých LED

3.3 Skripty Matlab

Pomocí programu byly testovány navržené algoritmy, díky dobré podpoře práce s obrazem a díky rychlejšímu psaní samotných algoritmů. Příklady algoritmů jsou v příloze 5.3.

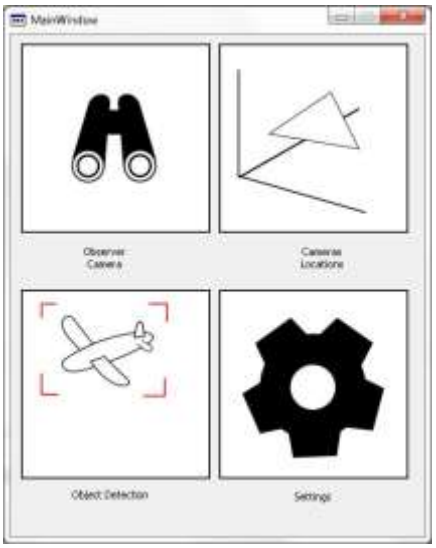
Na Obr. 15 je vidět výsledek následující algoritmu, který je zde uveden pro ukázku. Tento algoritmus byl navržen pouze v Matlabu, jelikož slouží pouze pro názornost. Cílem tohoto obrázku je vidět, jak je přibližně rozložená světlost. Algoritmus implementován v kapitole 5.3.4. Nejprve jsou načteny obrázky. Ty jsou poté prahovány několika iteracemi. Při každé iteraci jsou hodnoty překračující hodnotu prahu uloženy do pomocné matice. Pokud ji hodnoty nepřekračují, nejsou řešeny. Díky tomu vzniká vrstvená mapa různých prahů. Dalo by se to přirovnat k vrstevnicím na mapě, akorát vrstevnice je udána velikostí světlosti.



Obr. 15 Postupné prahování bílé LED

3.4 Obecná programová realizace

Program je rozdělen na 4 části, Observer Camera, Camera Locations, Object Detection a Settings. Na Obr. 16 jsou vidět jednotlivé obrazovky. Observer Camera je obrazovka s operátorským stanovištěm a vším ovládáním pro operátora. Camera Locations je obrazovka se samotným zastaničením a vším potřebným pro proces zastaničení. Object Detection je obrazovka pro algoritmus detekce objektů na obloze. Settings je obrazovka nutná pro počáteční nastavení sériových linek, barevného režimu, počtu rámců za vteřinu a nastavení označení kamer.



Obr. 16 Rozcestník - obrazovka SW

3.4.1 Struktura projektu

Programová struktura je navržena jako MVC, což je zkratka pro Model, View, Controller. Tato struktura je popsána na Obr. 17. Zde je tento model využit hlavně pro větší přehlednost a možnost jednoduše přidávat další obrazovky (View). Z obecného pohledu blok View zajišťuje pouze grafickou prezentaci výpočtů z modelu a dat z bloku Controller. Blok Controller zajišťuje komunikaci s jednotlivými rozhraními, regulaci a řízení datových toků. Blok Model běží v samostatném vláknu, které komunikuje s Controllerem pomocí mechanizmu signálů a slotů, který je typický pro Qt Framework. Tímto mechanizmem se komunikuje mezi jednotlivými bloky.

Základní myšlenka MVC je v rozdělení programu na části, které jsou každá nezávislá, a jen spolu s ostatními částmi komunikuje. V praxi to funguje tak, že hlavní controller je vytvořen při startu aplikace ve statické části programu a poté předáván do všech Views referencí nebo ukazatelem. Model sídlí v hlavním kontroléru, jen je přesunut do vlastního vlákna, které je také umístěno v kontroléru.

3.4.1.1 Controllers

V controllers jsou třídy pro ovládání Vimba API, Joysticku, sériových linek, logiky nastavení a hlavního kontroléru.

Třídy pro ovládání Vimba API:

- ApiController – v této třídě se řeší celkové startování Vimba API, management kamer. Tato třída podává přístup k připojeným kamerám, správě jejích listů, získávání rámců podle jejích indexů určených v nastavení.

- CameraController – individuální kontrolér jednotlivých kamer. Stará se získání obrazu z rámce přijatého z kamery, o jeho zpracování a převedení do QSharedPointer<QImage> a o jeho distribuci do Modelu a View. Dále se stará také o regulaci světlosti obrazu, přes regulaci elektronické závěrky.
- CameraObserver – malá pomocná třída která se stará o detekci změn listu kamer.
- FrameObserver – třída pro detekci vlastních rámců z Vimba API. Také se stará o bezpečnost vláken pomocí mutexů.
- JoystickController – Třída starající se o připojený Joystick
- SerialController – Tato třída se stará o běh jedné sériové linky, veškeré příkazy sériové komunikace jsou definovány v této třídě. Je to mu tak proto, aby byla komunikace s řídicí jednotkou jen na jednom místě kódu a bylo to lépe
- SettingsData – Tato třída je kontejner na celkové nastavení, stará se o ukládání a nahrávání ze statického souboru uloženého v instalační složce.
- ZastaniceniExtMatlab – třída zaobalující veškeré parametry nutné pro výpočet hlavních parametrů zastaničení. Tato třída také volá samotné skripty vygenerované do dll knihovny.

3.4.1.2 Models

V Models jsou pouze specifické třídy pro paralelní zpracování jednotlivých výpočtů. Třída modelu obsahuje pouze jednu třídu a tou je CamSharpController, který pracuje se signály picIntensity, který vrací maximální intenzitu, která se používá pro regulaci světlosti a signálem equilibriumGot, který vrací pozici v obraze nalezeného těžiště.

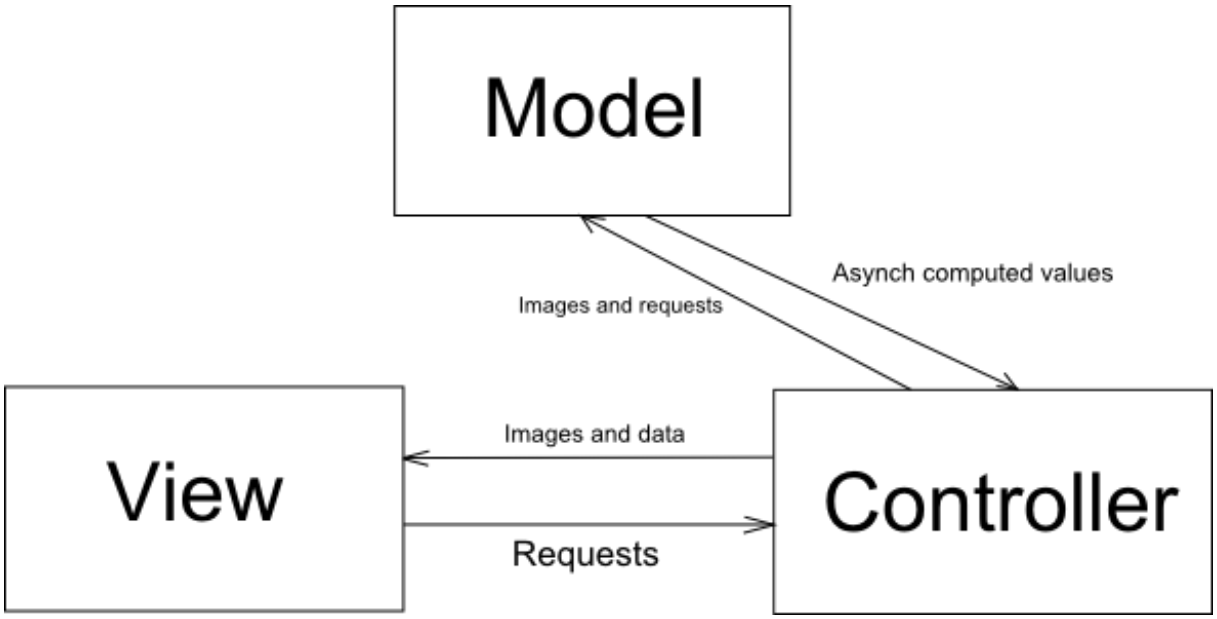
3.4.1.3 Views

Ve Views jsou dva typy tříd, třídy podpůrné a třídy grafické. Grafické třídy obsahují vlastní ui soubor, který definuje rozmístění grafických prvků. Pro toto rozmístění se používá Qt Creator, nebo se může definovat ručně. Avšak Qt Creator má řadu výhod jako například přiřazování signálů a slotů vybraným grafickým prvkům, okamžité vidění pozicování grafických prvků. Podpůrné třídy jsou třídy, které jsou použity bez ui a mají podpůrnou funkci v grafických třídách. Podpůrná třída je v tomto projektu pouze jedna a tou je GenericOverlay. Tato třída se stará o detekci klikání a vykreslování přes obraz. Tato třída dostává v parametru konstruktoru ukazatel na QLabel, který se používá pro zobrazování obrazu z kamer.

Grafické Views:

- CameraLocations – Třída, starající se o zobrazení zastaničení a o stavový automat ovládání vpřed a zpět. Zde je hlavní logika určující jak se bude procházet procesem zastaničení.

- ModeSelection – tato třída zajišťuje hlavní obrazovku vis Obr. 16, která slouží pro výběr různých režimů běhu.
- ObserverCameraView – tato obrazovka slouží pro správu operátorského stanoviště. Spojuje grafické prvky s prvky z kontroléru.
- SettingsView – tato třída se stará o zobrazení jednotlivých nastavení pro jednotlivé kamery.
- Justazdataview – tato třída se stará o uživatelské zadání parametrů z justáže na začátku měření zastaničení a předává data do ZastaniceniExtMatlab



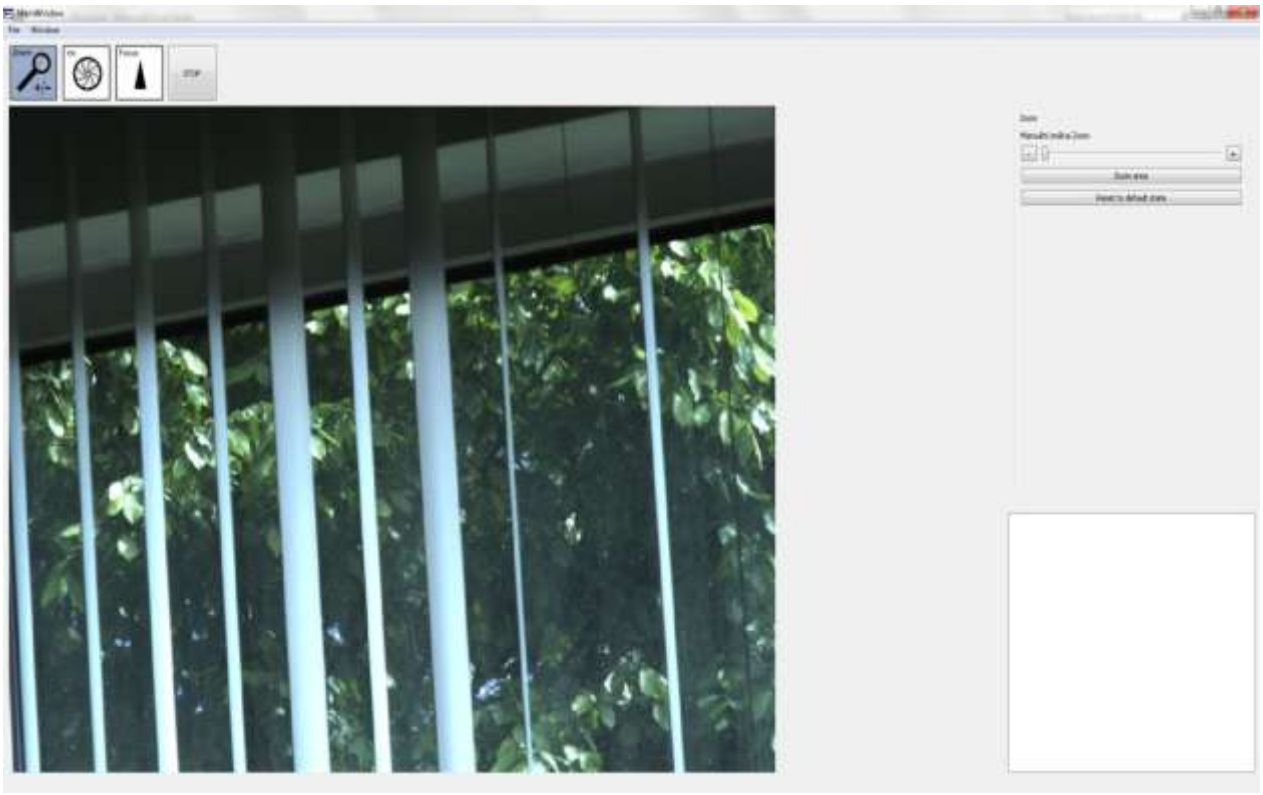
Obr. 17 Popis MVC

3.4.2 Joystick

Pro použití joysticku je nutné zahrnout knihovnu windows.h, která zahrnuje funkce pro komunikaci s joystickem. Komunikace s USB joystickem je synchronní, proto se volá aktuální stav joysticku s periodou 50ms. Pokaždé, když vyprší perioda 50ms, je zavolána funkce joyGetPosEx. Pokud tato funkce vrátí JOYERR_NOERROR, pokračuje se vyčítáním tlačítek ze struktury, která byla předána parametrem do joyGetPosEx. Tato struktura obsahuje stav tlačítek, pozici v ose x a y a polohu POI. Po přijetí těchto parametrů a při změně je emitován signál do slotů, které jsou napojeny na signály pohybu kniplu, stisku tlačítek a změny POI. Tato knihovna je v programovém vybavení Windows.

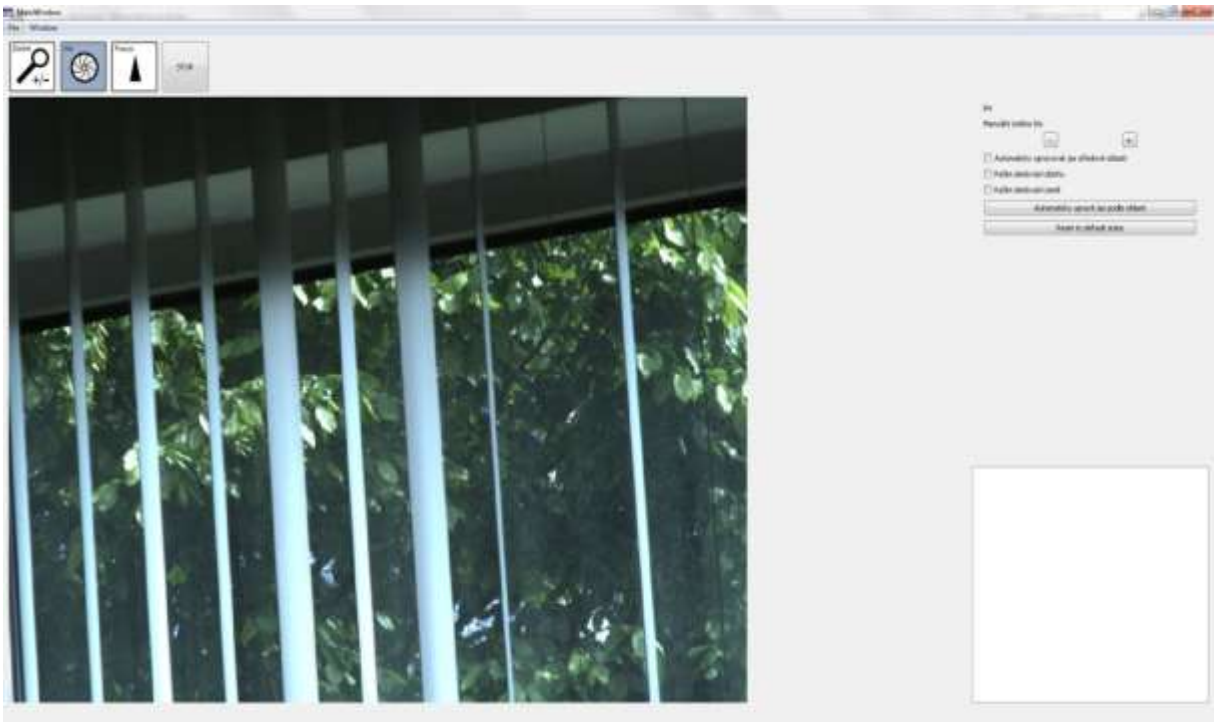
3.4.3 Operátorské stanoviště

Operátorské stanoviště slouží pro ovládání přehledové kamery pomocí joysticku, kláves a myši. Následuje popis jednotlivých obrazovek systému.



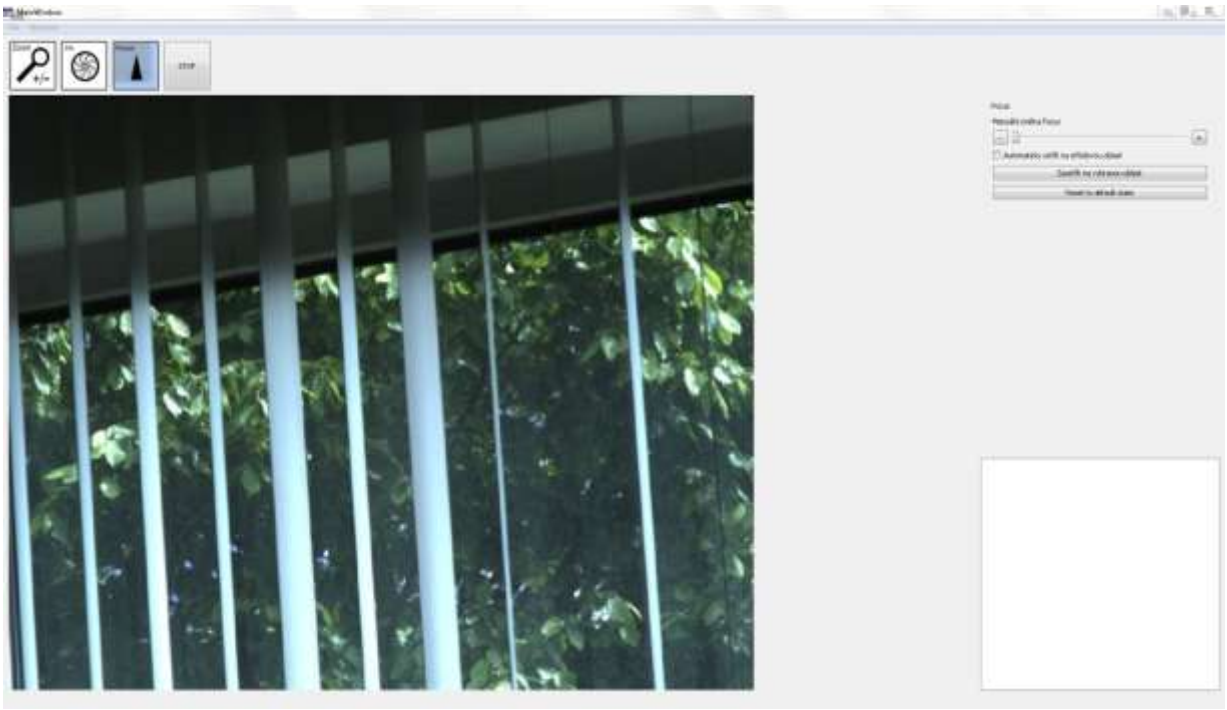
Obr. 18 Zoom na operátorském stanovišti

Zoom je zde realizován pomocí posuvníku, jelikož se nastavuje přes nastavení přímé hodnoty. Pod posuvníkem se nachází možnost resetu zoomu na základní hodnotu, čímž dojede zoom v řídicí jednotce objektivu do základní polohy. Bílé pole v pravém dolním rohu slouží pro zobrazování chybových hlášek. Na záložku zoomu je možné se přepnout stiskem klávesy „z“ pro rychlejší přepínání mezi požadovanými režimy. Dále je možné ovládat zoom pomocí posuvníku na joysticku. Tento posuvní odpovídá posuvníku na této obrazovce, a pokud se s ním hne, obrazovka se automaticky přepne na zoom.



Obr. 19 Clona na operátorském stanovišti

Clona je zde realizována pouze pomocí tlačítek, jelikož se clona pouze otevírá o určitý počet kroků. Řízení clony neumožnuje přesné nastavení hodnoty a není ani nutné, jelikož clona je řízena pomocí obrazu a zpětné vazby. Dále jsou zde položky na sledování země a oblohy, pro různé hodnoty požadovaných jasů. Jako poslední je možnost manuálně resetovat do základní polohy. Bílý obdélník slouží k zobrazování chybových hlášek. Na záložku clony je možné se přepnout stiskem klávesy „x“ pro rychlejší přepnutí na obrazovku clony. Clona se také ovládá pomocí joysticku, ale jelikož zde není posuvník možný, je využito kruhového palcového ovladače na vrchní části joysticku, díky čemuž je možné ovládat bezproblémově jednou rukou.



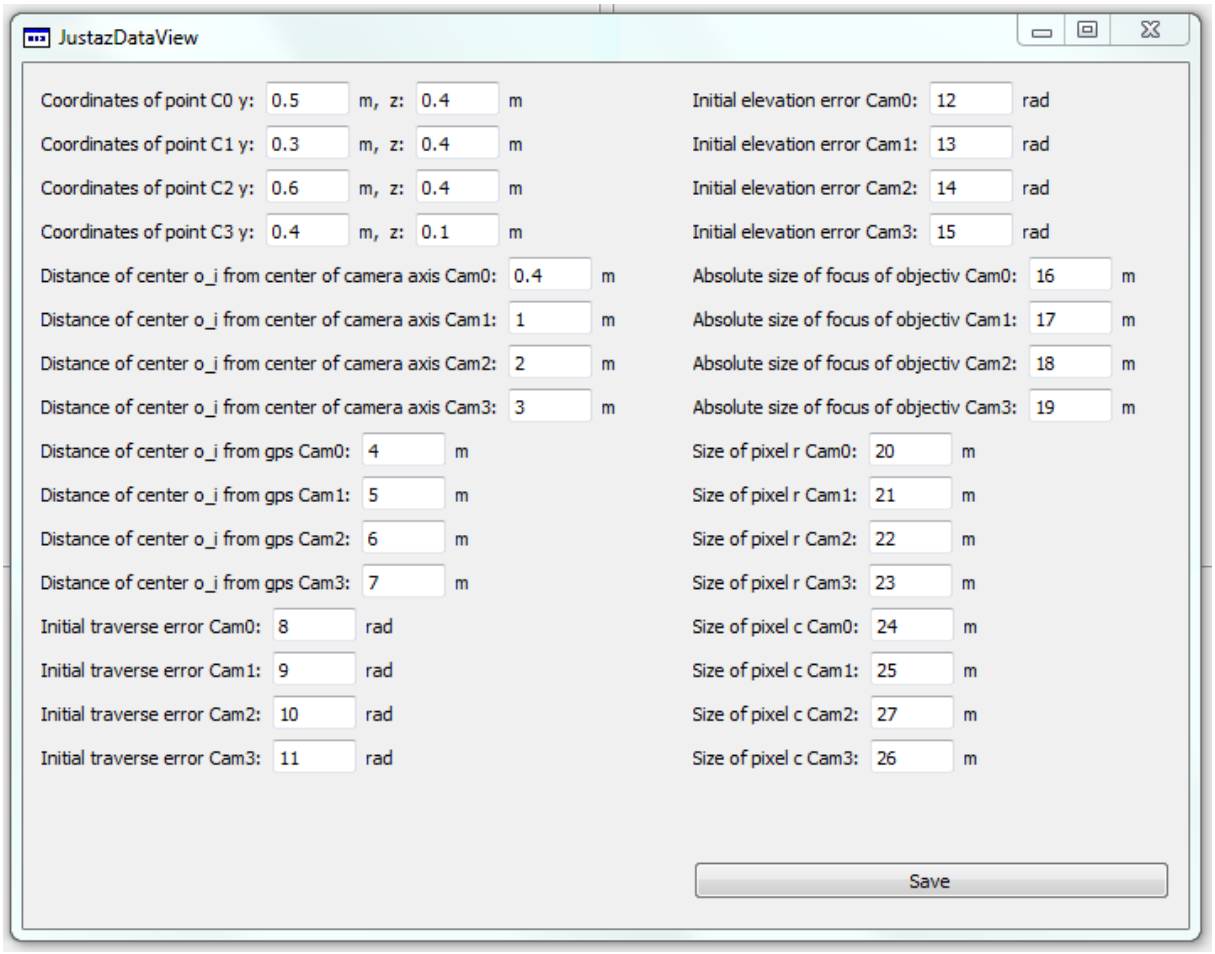
Obr. 20 Ostření na operátorském stanovišti

Ostření je zde zajištěno pomocí posuvníku, jelikož se ostření ovládá na přesnou hodnotu. Pod posuvníkem je možnost resetování ostření na určitou hodnotu. Bílý obdélník slouží pro zobrazení chybových hlášek. Do režimu ostření je možné se přepnout klávesou „c“ pro rychlejší přepínání. Je také možné ovládat ostření pomocí palcového ovladače na vrchní části joysticku pro ovládání jednou rukou. Palcový ovladač je dvouosý, díky čemuž je možné jím ovládat jak clonu, tak ostření

Ovládání zařízení náměru a odměru se ovládá kniplem joysticku, náměr se ovládá od sebe a k sobě a odměr se ovládá zleva doprava.

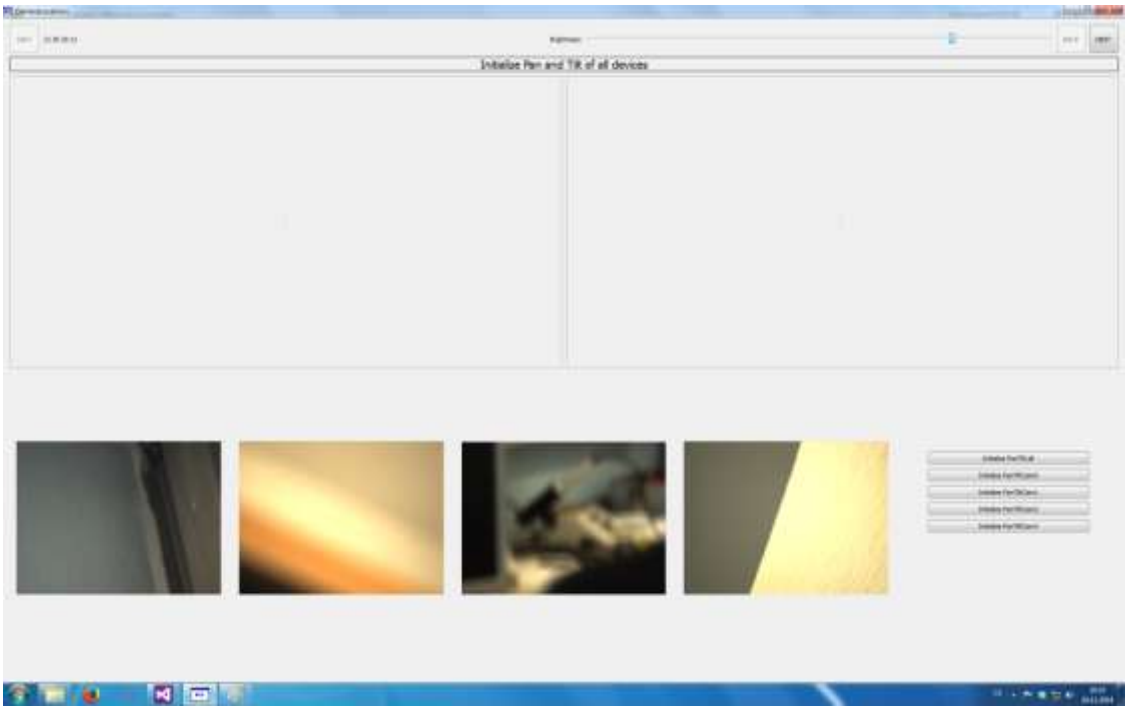
3.4.4 Realizace zastaničení

Zastaničení je složitější úkon než operátorské stanoviště. Proces zastaničení má více kroků, které musí být vykonány, aby bylo možné vypočítat odchylky od severu a polohy jednotlivých kamer. Následují jednotlivé obrazovky zastaničení s vysvětlením jednotlivých kroků. Zastaničení je koncipováno jako pomocník, který v horní liště operátoru dává vyzvání pro jednotlivé kroky. Zastaničení bylo navrženo, aby mohlo být ovládáno pouze poučenou osobou, bez hlubších znalostí problému.



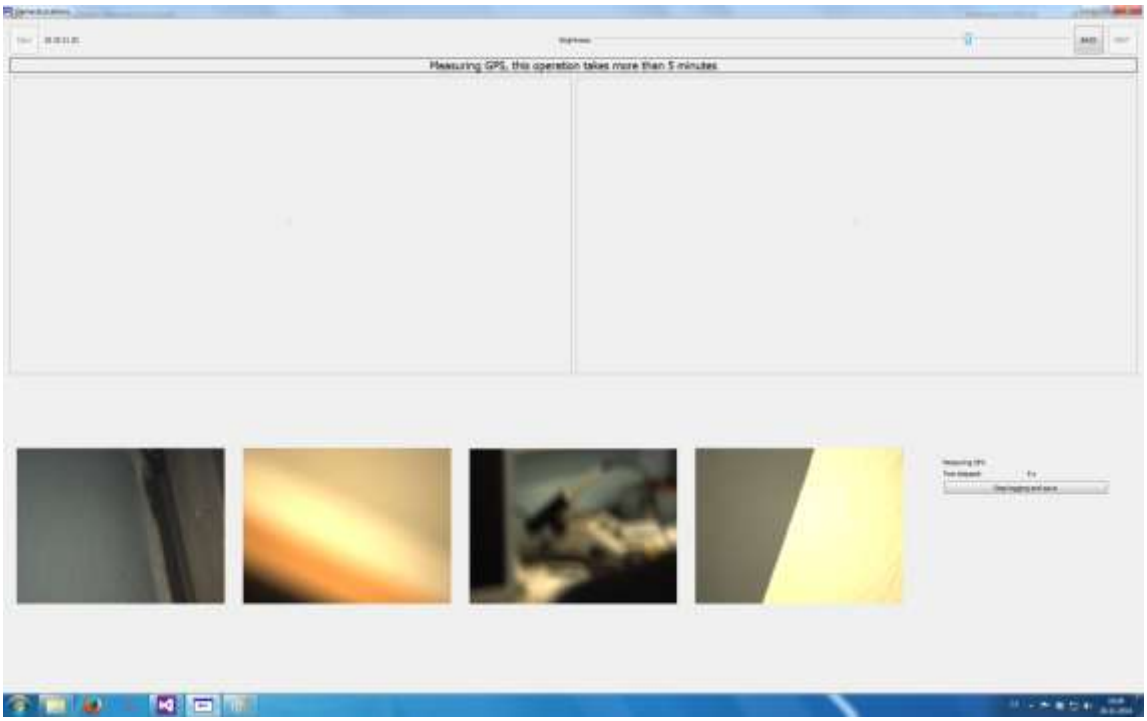
Obr. 21 Zastaničení - Vyplnění justáže

V první obrazovce se nastavují naměřené hodnoty z justáže. Tyto hodnoty určují převážně odchylky stojanů a úchytů, vzdálenosti středů kamery a záměrného kříže a nastavení objektivu kamer.



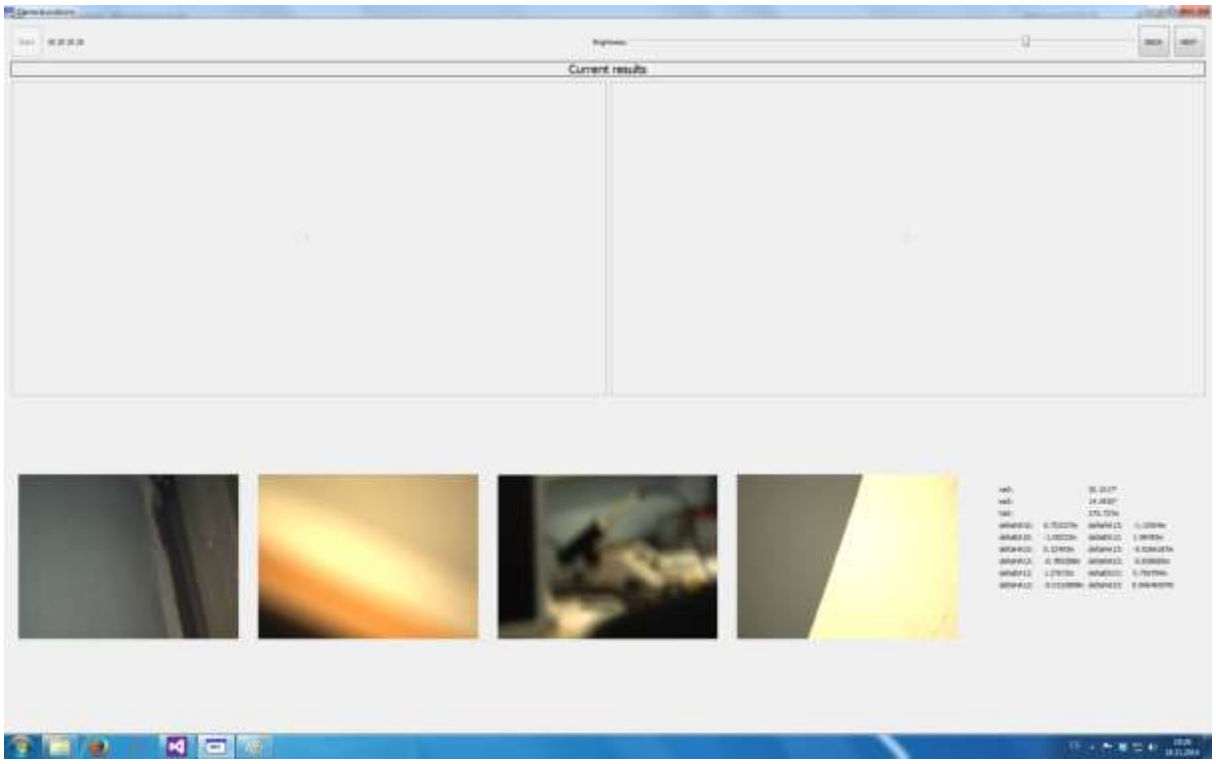
Obr. 22 Zastaničení - inicializace zařízení náměru a odměru

Při spuštění zastaničení je nutné zařízení náměru a odměru resetovat do nulového stavu, aby bylo zajištěno, že jsou kamery v základní pozici kvůli GPS anténám, které musí být namířeny na oblohu. Možnost resetovat všechny najednou, nebo po jedné kameře bylo vytvořeno kvůli problému s kabely na stojanech kamer, které mají tendenci se na prototypu zasekávat.



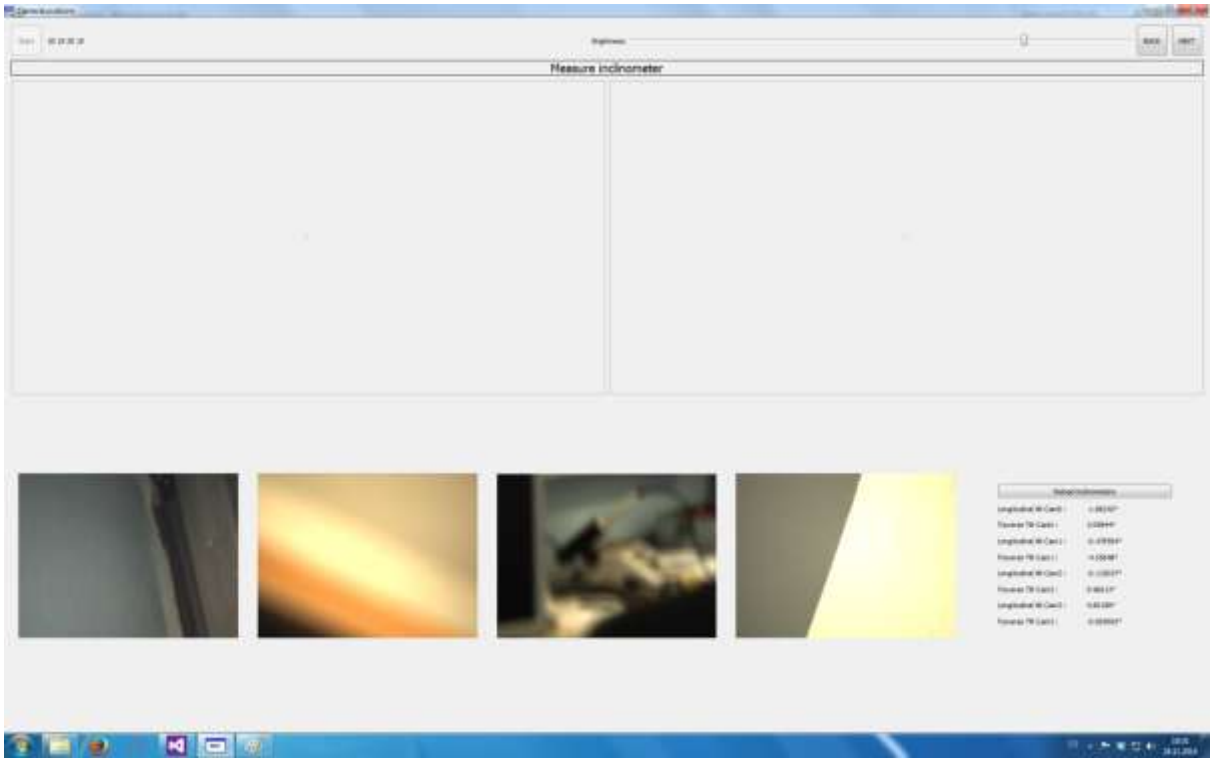
Obr. 23 Zastaničení - měření GPS

Po nastavení kamer do původních stavů je nutné naměřit GPS ze všech stojanů všech kamer. Toto měření není podmíněno časově, pouze se čas zobrazuje a operátor po dostatečné době vypne měření. Po vypnutí měření se spustí zpracování externího skriptu, který počítá pozice pomocí diferenciální GPS pomocí knihovny RTKLib. Když se toto dokončí, spouští se skript z Matlabu, který zpracovává výsledky z RTKLib a vrací je zpět do programu. Tento celý proces probíhá automaticky.



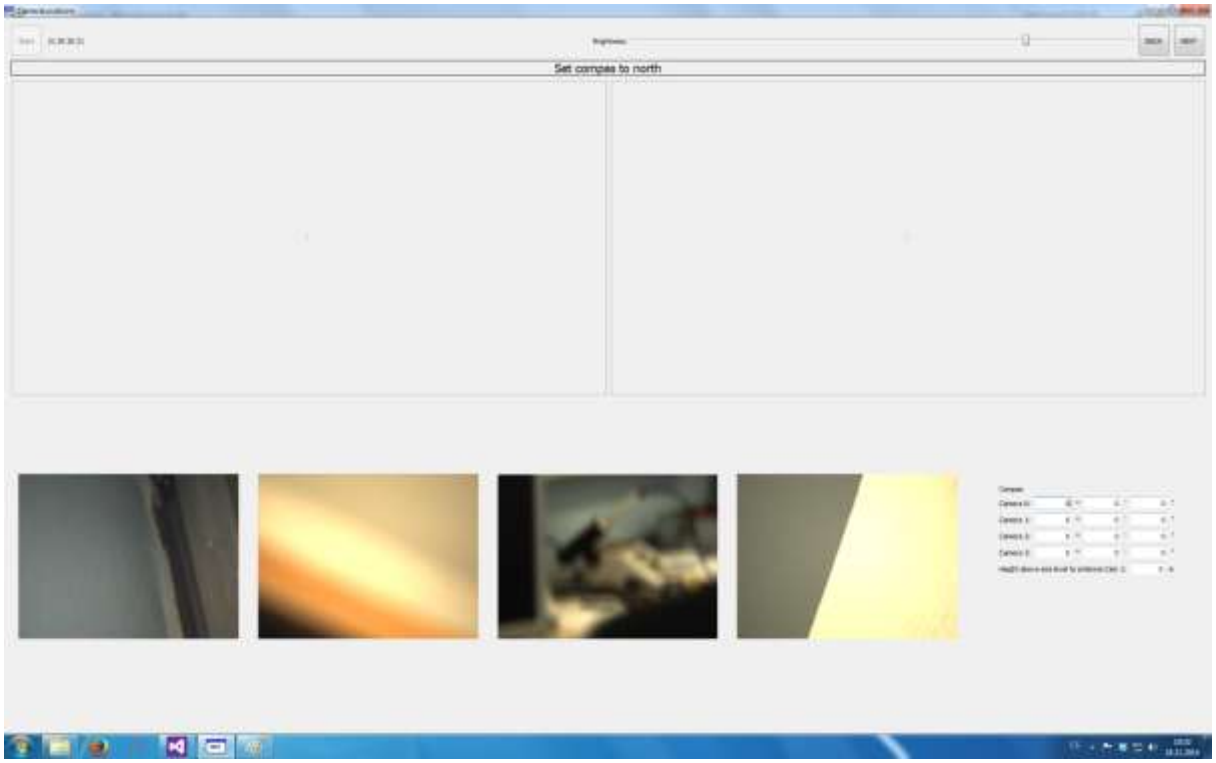
Obr. 24 Zastaničení - zobrazení kontrolních výsledků z GPS

Tato obrazovka slouží ke kontrole výsledků z měření a výpočtu GPS. Tato obrazovka byla vytvořena hlavně pro testování algoritmů RTKLib a jejich spolehlivosti.



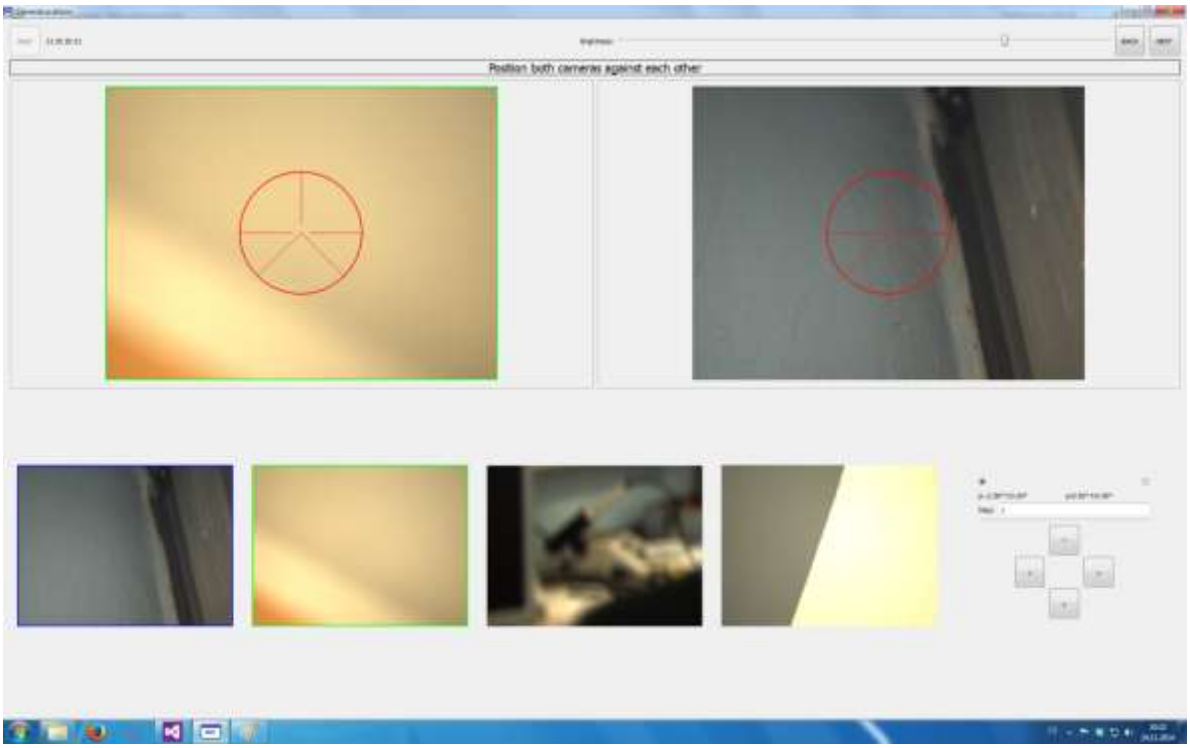
Obr. 25 Zastaničení - měření inklinometrů

Během měření GPS je vhodný čas pro operátora, aby obešel všechny stojany a změřil náklon stojanům vůči zemi. Hodnoty z inklinometru se vyčítají pouze jednou automaticky po připojení k jednotce. Po dokončení měření a výpočtu GPS se hodnoty z inklinometru vyčtou z jednotlivých řídicích jednotek a uloží do procesu zastaničení.



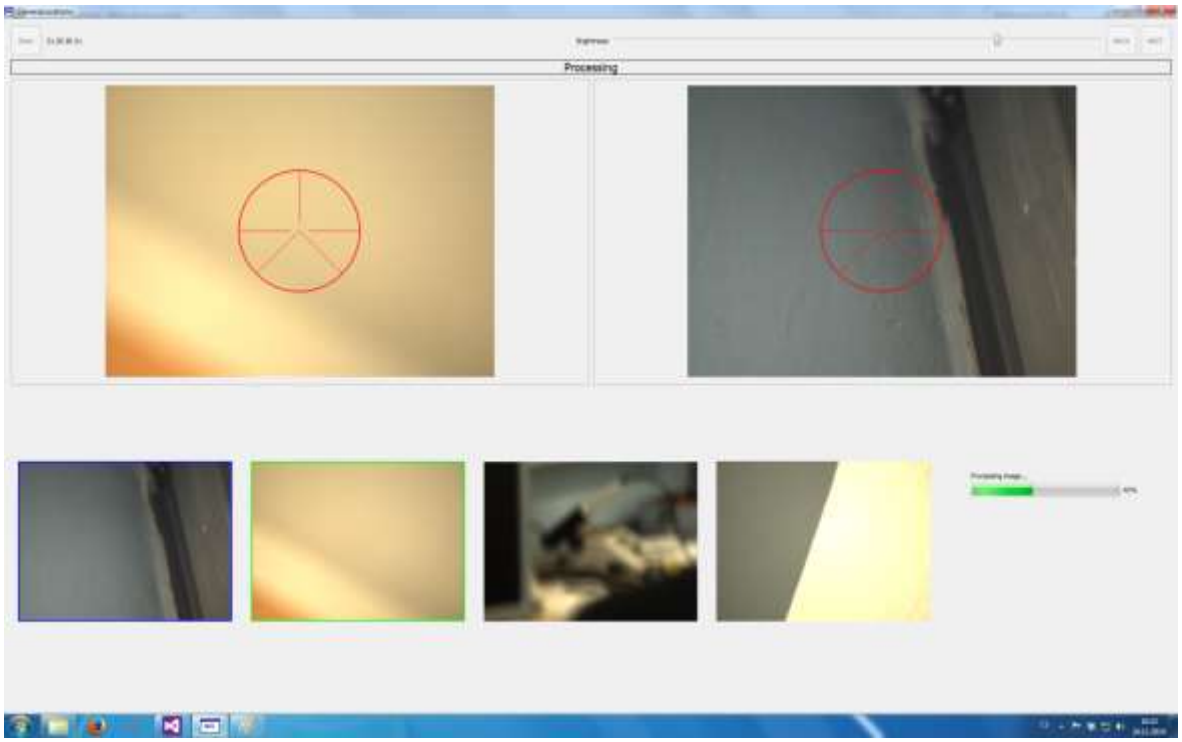
Obr. 26 Zastaničení - zadání kompasu

Odchylky od severu každé kamery změřené kompasem musí být zadány operátorem manuálně. Avšak tyto hodnoty nejsou kritické, pouze kontrolní pro algoritmus výpočtu zastaničení. Také musí operátor vyčíst pro aktuální polohu první kamery nadmořskou výšku z mapy.



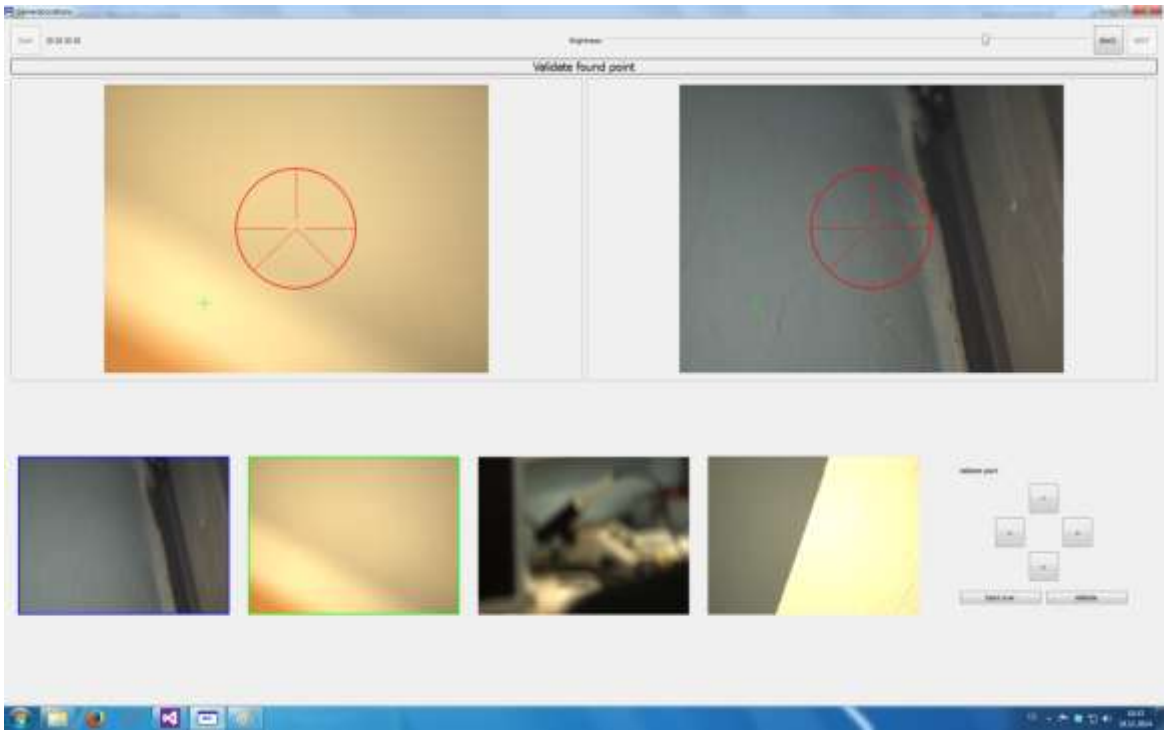
Obr. 27 Zastaničení - vzájemné namíření kamer

Po zadání a naměření těchto parametrů se přichází k procesu, který měří úhly mezi kamerami. V tomto kroku se pomocí joysticku a přesného krokování operátor snaží namířit záměrný kříž obou kamer nasměrovat do centra obrazu označeného červenou záměrnou značkou. Když operátor dostane záměrný kříž obou kamer co nejblíže tohoto bodu, pokračuje se dalším bodem.



Obr. 28 Zastaničení - zpracovávání obrazu

Po namíření kamer na sebe se v obrazech hledají záměrné značky pomocí algoritmů navržených v analýze. Tento proces se několikrát opakuje a výsledky se průměrují pro co nejpřesnější výsledek.



Obr. 29 Zastaničení - validace nalezených bodů

Po nalezení těchto bodů jsou operátorovi zobrazeny zeleným křížem. Operátor má možnost zkontrolovat, jestli nejsou nalezené body mimo rozumné meze například kvůli vlivu jiného světelného zdroje. Poté může body buď potvrdit, nebo začít měření znovu. Pokud bylo měření potvrzeno, s měřením se pokračuje u dalších kamer od vzájemného namíření kamer vis Obr. 27.

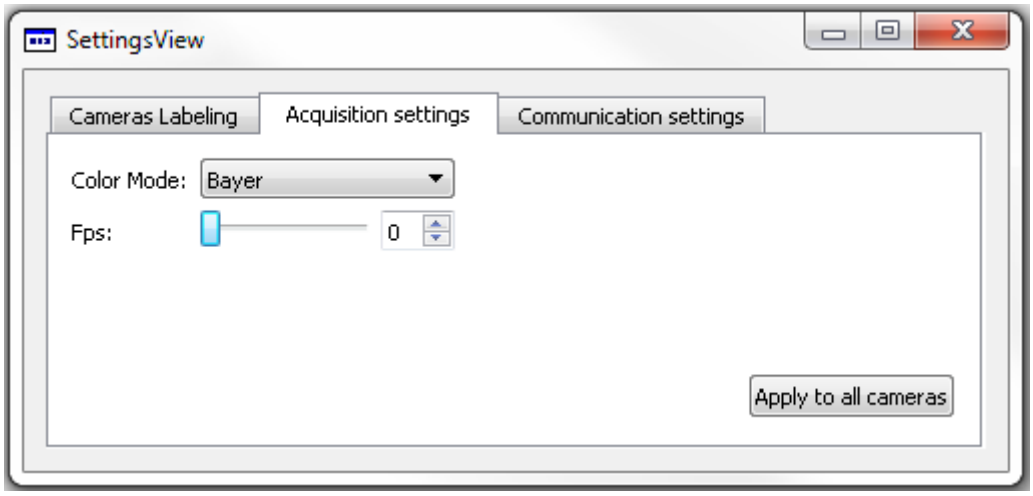
V dolní části obrazovky jsou obrazy ze všech kamer a zeleným obdélníkem je označena kamera vlevo a modrým obdélníkem je označena kamera vpravo. Při míření kamer je aktivní kamera zvýrazněna v hlavní obrazovce barevným obdélníkem. Tyto grafické operace jsou zpravovány právě přes GenericOverlay třídu.

3.4.5 Nastavení

Nastavení slouží k propojení jednotlivých sériových linek, vybrání požadované rámce za vteřinu a přiřazení rolí jednotlivým kamerám. Následuje přehled obrazovek nastavení.

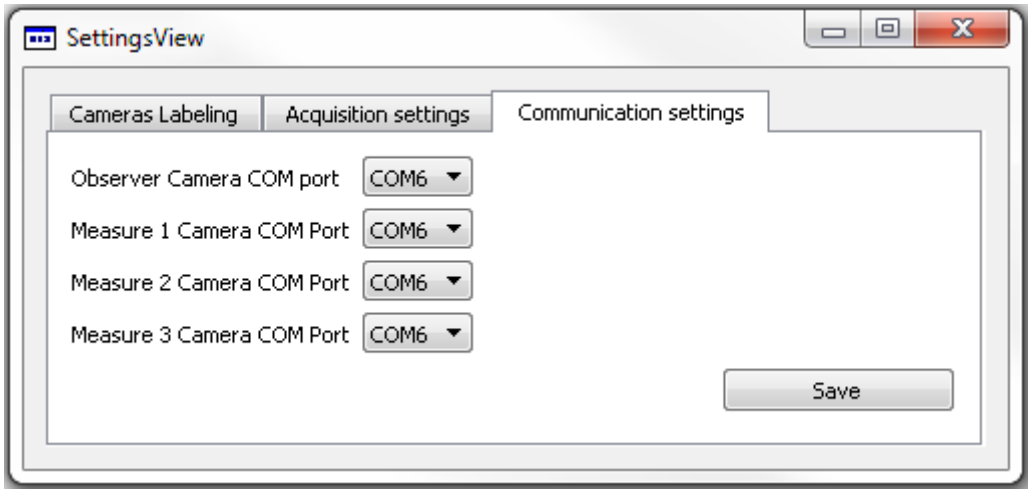
Obr. 30 Nastavení - označení kamer

Nastavení označení jednotlivých kamer slouží pro označení jednotlivých kamer. Toto slouží pro zajištění jednoznačného označení jednotlivých kamer. Toto nastavení je možné zjistit v obrazovce zastaničení, jelikož jsou kamery seřazeny podle tohoto nastavení.



Obr. 31 Nastavení - Barevný režim a rámce za sekundu

Nastavení obrazové akvizice dává možnost nastavit barevný režim na černobílý nebo barevný, kde barevný režim je Bayer. Nastavení rámců za vteřinu je možné pouze od 1 do 33 snímků za vteřinu. Toto nastavení je sdílené pro všechny kamery.



Obr. 32 Nastavení - přiřazení sériových portů

V nastavení komunikačního rozhraní se přiřazují komunikační porty k jednotlivým kamerám. Toto je nutné, aby byly jednotlivé kamery synchronizovány se svými řídicími jednotkami.

Všechna tato nastavení jsou načítána ze souboru config.cfg, který se nachází v instalační složce aplikace. Při uložení nastavení se ukládá do tohoto souboru. Toto bylo vybráno, aby se při znovu spuštění aplikace nemusely tyto hodnoty nastavovat.

4 Závěr

Práce si kladla za cíl vytvořit program pro zpracování obrazu a dalších informací nutných pro měření úhlů kamer k severu a jejich vzájemné pozice. Tato práce si dávala za cíl navrhnout proces, který tato měření zajistí s minimální operátorovou znalostí systému. Dále si dávala za cíl vytvořit snadno ovladatelné operátorské stanoviště pro ovládání zařízení odměru a náměru, zoom, clonu a ostření objektivu přehledové kamery.

Následující úkoly byly zpracovány:

- Byly otestovány různé světelné značky s použitím různých LED. Závěrem z tohoto testování bylo navržení záměrného kříže s použitím světelného pole s kolimátorem. A jeho následné vytvoření a zapojení do řídicí jednotky.
- Byla navržena metoda pro měření vzájemných úhlů, tato metoda vyžaduje spolupráci operátora při vlastním měření, avšak nevyžaduje velkou znalost postupu a problému pro ovládajícího operátora, jelikož poskytuje přehledný systém postup a návod v průběhu celého procesu.
- Byly navrženy algoritmy pro detekci kontrastních objektů a hlavně záměrných značek. Kvůli problémům s přesvětlením jinými objekty byl navržen alternativní algoritmus, který využívá zapínání a vypínání LED křížů a byl vybrán záměrný kříž s nevyšší možnou světelností.
- Byly otestovány a porovnány zpracování obrazů a hlavně jejich rychlost v programovacích jazycích C++ a C#. Na jejímž základě byl vybrán programovací jazyk C++.
- Bylo navrženo operátorské stanoviště pro ovládání přehledové kamery, jejího zoomu, clony, ostření a pohybu zařízením náměru a odměru.

Tato práce by mohla být rozvinuta v budoucnu rozšířením různých detekčních mechanizmů pro detekci záměrných značek a objektů, realizací podobného systému v C#, který by našel řešení problémů nalezených v této práci. Jelikož je tato práce součást a jeden ze základních kamenů hlavního projektu, je jasné její hlavní budoucí využití. Pro její budoucí využití bude muset být zakomponována do hlavního projektu právě jako knihovna dll, což již bylo v této práci započato.

Tato práce mi byla velmi přínosná. Naučil jsem se jak pod platformou Windows vytvářet dll knihovny, WPF grafickou knihovnu pro tvorbu GUI nové generace v C#, Qt Framework pro vysoce pohodlný a přehledný vývoj GUI v C++, práci s velkými obrazovými daty v počítači, zpracovávání obrazu pro různé použití a mnoho dalšího.

5 Přílohy

5.1 Zdroje

[1]http://www.alliedvisiontec.com/emea/products/cameras/gigabit-ethernet/prosilica-gt/gt1290.html

[2]http://measure.feld.cvut.cz/system/files/files/cs/vyuka/predmety/A4B38NVS/A0M38OSE_2014_Pred1_2__CMOS_1.pdf

[3] http://doc.qt.io/qt-5/

[4] http://msdn.microsoft.com/en-us/library/ms754130%28v=vs.110%29.aspx

[5]http://cz.mouser.com/ProductDetail/Illumitex/ARGW9S-100/?qs=sGAEpiMZZMsuj5tdPuAIdzjE%252b%252bSw5C3gHuS1QFQl%2fcI%3d

[6] Heijden, F.: Image Based Measurement Systems: Object Recognition and Parameter Estimation, ISBN 10: 0471950629, Wiley, 1995

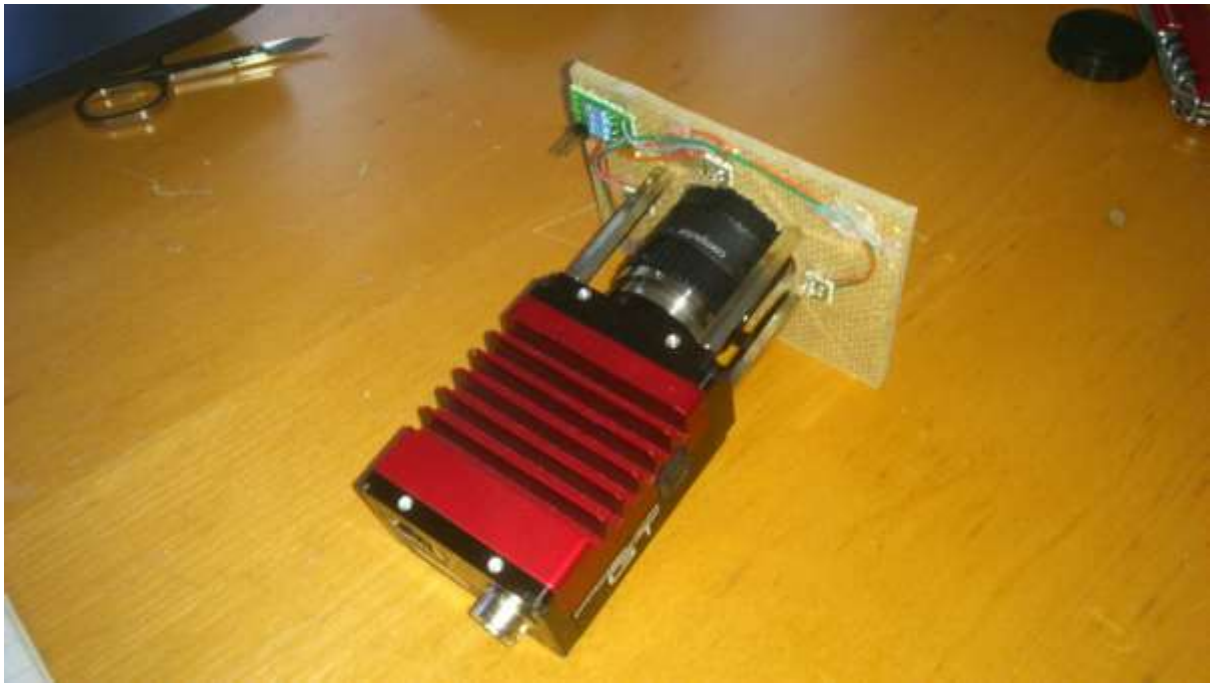
[7] Hlaváč V., Sedláček, M.: Zpracování signálu a obrazu, skriptum, ČVUT-FEL, 2002

[8] Fischer J.: Optoelektronické sensory a videometrie, skriptum, ČVUT-FEL, 2002

[9] Pavlišta D.: *Diplomová práce, 3D Sensor,* 2010

5.2 Obrázky

Obr. 33 Prototyp záměrného kříže



Obr. 34 Prototyp záměrného kříže – zapojení

Obr. 35 Bílá LED zaostřeno na nekonečno

Obr. 36 Červená LED zaostřeno na nekonečno

Obr. 37 Modrá LED zaostřeno na nekonečno

Obr. 38 Zelená LED zaostřeno na nekonečno

5.3 Testovací skripty v Matlab

5.3.1 Metoda výpočtu rozdílového těžiště

```
im1 = double(imread('bila_grey_02.bmp'));
im2 = double(imread('bila_grey_01.bmp'));

im_diff = im1-im2;
```

```
im_diff_calc_x = 0;
im_diff_calc_y = 0;

im_diff = abs(im_diff);

im_diff(im_diff<20) = 0;

disp(max(im_diff(:)));

for i = 1:size(im1,1),
    for j = 1:size(im1,2),
        im_diff_calc_x = im_diff_calc_x + im_diff(i,j)*i;
        im_diff_calc_y = im_diff_calc_y + im_diff(i,j)*j;
    end
end

M = sum(im_diff(:));
res_x = round((im_diff_calc_x/M));
res_y = round((im_diff_calc_y/M));

disp(sprintf('%d',res_x));
disp(sprintf('%d',res_y));

xRealSize = abs(res_x-size(im1,1)/2);
yRealSize = abs(res_y-size(im1,2)/2);

disp(sprintf('%d mm',xRealSize));
disp(sprintf('%d mm',yRealSize));

im = im1;

imshow(im/255);
hold on;
pause
imshow(im_diff);
hold on;
plot(size(im1,2)/2,size(im1,1)/2,'+','MarkerSize',40);
hold on;
plot(res_y,res_x,'+','MarkerSize',40);
```

5.3.2 Metoda základního prahování

```
function [ output_matrix ] = threshold( im, value )
%THRESHOLD Summary of this function goes here
%   Detailed explanation goes here

output_matrix = im;
for i = 1:size(im,1),
   for j = 1:size(im,2),
```

```
        if output_matrix(i,j) < value,
         output_matrix(i,j) = 0;
        else
         output_matrix(i,j) = 255;
        end
   end
end
imshow(output_matrix);
end
```

5.3.3 Testování intenzity barevných LED

```
im_green = double(imread('zelena_inf.bmp'));
im_blue  = double(imread('modra_inf.bmp'));
im_red   = double(imread('cervena_inf.bmp'));
im_white = double(imread('bila_inf.bmp'));

treshold = 180;

disp('Max zelena');
disp(max(im_green(:)));
im_green(im_green < treshold) = 0;
im_green(im_green >= treshold) = 1;
imwrite(im_green,'zelena_inf_tresh.bmp');

pause

disp('Max modra');
disp(max(im_blue(:)));
im_blue(im_blue < treshold) = 0;
im_blue(im_blue >= treshold) = 1;
imwrite(im_blue,'modra_inf_tresh.bmp');

pause

disp('Max bila');
disp(max(im_white(:)));
im_white(im_white < treshold) = 0;
im_white(im_white >= treshold) = 1;
imwrite(im_white,'bila_inf_tresh.bmp');

pause

disp('Max cervena');
disp(max(im_red(:)));
im_red(im_red < treshold) = 0;
im_red(im_red >= treshold) = 1;
imwrite(im_red,'cervena_inf_tresh.bmp');

pause
```

5.3.4 Algoritmus postupného prahování

```
im_green = double(imread('zelena_inf.bmp'));
im_blue  = double(imread('modra_inf.bmp'));
im_red   = double(imread('cervena_inf.bmp'));
im_white = double(imread('bila_inf.bmp'));

for treshold = 1:30:255
    g(im_green >= treshold) = treshold;

    b(im_blue >= treshold) = treshold;

    r(im_red >= treshold) = treshold;

    w(im_white >= treshold) = treshold;
end

figure('Name','Green','NumberTitle','off');
imagesc(g);
colormap(gray);
figure('Name','Blue','NumberTitle','off');
imagesc(b);
colormap(gray);
figure('Name','Red','NumberTitle','off');
imagesc(r);
colormap(gray);
figure('Name','White','NumberTitle','off');
imagesc(w);
colormap(gray);
```