ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ

Fakulta elektrotechnická

Vizuální kontrola chodu stroje

Bakalářská práce

Vypracoval: Tomáš Kohout

Vedoucí práce: Ing. Pavel Mezera

Katedra řídící techniky
Praha 2009

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Tomáš Kohout**

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný
Obor: Kybernetika a měření

Název tématu: **Vizuální kontrola chodu stroje**

Pokyny pro vypracování:

1. Navrhněte a realizujte opatření pro potlačení rušivých světelných vlivů na rychloběžnou kameru.
2. Za využití rychloběžné kamery a software pro strojové vidění MvTec Halcon realizujte detekci správného chodu stroje. Výsledek detekce bude reportován přes rozhraní PVI do řídicího panelu.
3. Rozšiřte zpracovávání obrazu o detekci manipulace s kamerou.
4. Připravte úlohu pro výuku řízení polohových aplikací.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Pavel Mezera

Platnost zadání: do konce letního semestru 2009/10

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 13. 3. 2009

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze, dne 10.7.2009 …….

Podpis

Abstrakt

Tématem této bakalářské práce je návrh opatření pro potlačení rušivých vlivů při snímání laboratorního modelu žongléra průmyslovou kamerou tak, aby funkce kamery nebyla závislá na okolních podmínkách.

Práce popisuje implementaci software s využitím knihoven MvTec Halcon pro strojové vidění. Tato aplikace poskytuje zpětnou vazbu o přítomnosti všech tří koulí pro žonglování a komunikuje s řídícím panelem přes PVI rozhraní.

V poslední části je popsána úloha pro řízení polohových aplikací s využitím průmyslové kamery.

Abstract

This bachelor thesis contains informations about elimination of light disturbances which occours by picking up laboratory model of juggler with high speed industry camera. Goal was to design system independent on light conditions around juggler.

In this thesis is described implementation of software with libraries MvTec Halcon for machine vision. This software provides visual feedback if all of three balls is present in juggling cycle and comunicates with control panel via PVI interface.

The last section describes exercise for education of position servoing with usage of industry camera.

Poděkování

Na tomto místě bych chtěl poděkovat svému vedoucímu bakalářské práce Ing. Pavlu Mezerovi za jeho cenné rady a připomínky. Dále patří mé poděkování Pavlu Jarošovi, Lubomíru Prudkovi a Janu Kmentovi, kteří společně se mnou na žonglérovi pracovali, za výbornou spolupráci při dosahování společně vytyčených cílů.

Obsah

1. Úvod ... 1
   1.1. Žonglér ... 1
   1.2. Projekt Žonglér ... 2
   1.3. Cíle práce ... 2
   1.4. Členění práce ... 2
2. Potlačení rušivých vlivů pomocí osvětlení ... 4
   2.1. Strojové vidění ... 4
   2.2. Osvětlení ... 4
      2.2.1. Požadavky na osvětlení scény v průmyslových aplikacích ... 5
      2.2.2. Požadavky na optické vlastnosti: ... 6
      2.2.3. Požadavky na elektrické vlastnosti ... 6
      2.2.4. Požadavky na mechanické vlastnosti ... 6
   2.3. Zdroje osvětlení ... 7
      2.3.1. Teplem buzené zdroje ... 7
      2.3.2. Luminiscenční zdroje ... 8
      2.3.3. Klasické/halogenové žárovky ... 8
      2.3.4. Metal halogenidové výbojky ... 9
      2.3.5. Xenonové výbojky ... 9
      2.3.6. Lineární zářivky ... 10
      2.3.7. LED diody ... 11
      2.3.8. Lasery ... 12
   2.4. Interakce světla a materiálu ... 14
   2.5. Interakce světla na rozhraní dvou prostředí ... 15
   2.6. Žonglér a rušivé vlivy ... 16
      2.6.1. Požadavky na řešení osvětlení žongléra ... 16
      2.6.2. Výsledné řešení osvětlení ... 17
3. Detekce správného chodu stroje ... 19
   3.1. Softwarová implementace aplikací strojového vidění ... 19
   3.2. Metody používané k získávání vlastností ... 21
      3.2.1. Blob analýza ... 21
      3.2.2. Detekce hran ... 22
      3.2.3. Template matching ... 23
   3.3. Objekty určené k rozpoznávání ... 24
   3.4. Algoritmy pro rozpoznávání objektů ... 25
      3.4.1. Použité třídy ... 26
      3.4.2. Rozeznávání vyhazovacích misek ... 26
      3.4.3. Rozeznávání letících koulí ... 30
      3.4.4. Detekce polohy referenčního znaku ... 30
      3.4.5. Detekce přítomnosti koule při inicializaci ... 31
4. Aplikace ... 32
   4.1. Komunikace mezi aplikací a řídícím systémem – PVI ... 32
      4.1.1. Komunikace mezi PC aplikací a PLC ... 33
      4.1.2. PVI Objekty ... 34
      4.1.3. Vytváření objektů ... 35
      4.1.4. Čtení proměnných z PLC ... 35
      4.1.5. Zapisování proměnných do PLC ... 36

4.2. Struktura aplikace ... 36
4.3. Interakce aplikace a řídícího systému ... 38
4.3.1. Zpětná vazba o přítomnosti koulí ... 38
4.3.2. Zpětná vazba o počátečním stavu stroje ... 39
4.3.3. Test provozu aplikace ... 39
4.3.4. Detekce manipulace s kamerou ... 39
4.3.5. Spolehlivost komunikace ... 40
5. Laboratorní úloha ... 44
6. Závěr ... 45

Přílohy:

A. Ukázky zdrojových kódů ... 48
A.1 Detekce vyhazovacích misek ... 48
A.2 Detekce letících koulí ... 48
A.3 Detekce přítomnosti koule ve vyhazovací misce ... 49
A.4 Inicializace PVI ... 50
B. Realizace optické závory průmyslovou kamerou ... 51
B.1 Realizace optické závory ... 51
B.1.1 Nasnímání sekvence snímků pro testování algoritmu ... 51
B.1.2 Vytvoření algoritmu rozpoznávající letící koule ... 52
B.1.3 Použití parametrů v připravené aplikaci+komunikace s PLC ... 56
B.2 Kontrola přítomnosti koule ve vyhazovací misce ... 58
B.2.1 Pořízení snímku s předlohou ... 58
B.2.2 Vytvoření předlohy v HDevelop ... 58
B.2.3 Použití předlohy a parametrů v připravené aplikaci ... 60

Seznam obrázků

Obr. 1.1 Čelní pohled na žonglér ... 1
Obr. 2.1 Základní struktura systému strojového vidění ... 5
Obr. 2.2 Wienův posunovací zákon ... 7
Obr. 2.3 Spektrum vyzařovaných vlnových délek lineární zářivky ... 11
Obr. 2.4 Schnellův zákon ... 15
Obr. 2.5 Vliv okolního světla na zpracování obrazu ... 16
Obr. 2.6 Schéma osvětlení systému ... 18
Obr. 2.7 Porovnání osvětlení před úpravami(vlevo) a po úpravách (vpravo) ... 18
Obr. 3.1 Příklad zpracování obrazu blob analýzou(prahování) ... 22
Obr. 3.2 Příklad zpracování obrazu pomocí detekce hran (Sobelův filtr) ... 22
Obr. 3.3 Příklad zpracování obrazu pomocí template matching ... 23
Obr. 3.4 Detail vyhazovací misky ... 24
Obr. 3.5 Vyhazovací misky ve snímaném obraze ... 24
Obr. 3.6 Referenční znak pro detekci pohybu kamery ... 25
Obr. 3.7 Princip vývoje software s použitím knihoven HALCON a HDevelop ... 25
Obr. 3.8 Jasový profil pozadí při maximálním a minimálním světle z okolí ... 27
Obr. 3.9 Jasové hodnoty odlesku s původníma lambertovským povrchem ... 28
Obr. 3.10 Projevy směrových odlesků původního povrchu ... 28
Obr. 3.11 Histogram časů pro algoritmus prahování ... 29
Obr. 3.12 Histogram časů pro algoritmus template matching ... 29
Obr. 3.13 Oblast ROI pro kontrolu průletu koule ... 30
Obr. 3.14 Oblasti pro vyhodnocování přítomnosti koule při inicializaci ... 31
Obr. 4.1 Struktura PVI ... 33
Obr. 4.2 Hiearchie PVI objektů ... 34
Obr. 4.3 Struktura aplikace ... 36
Obr. 4.4 Screenshot aplikace ... 37
Obr. 4.5 Sekvenční diagram komunikace mezi PLC a aplikací ... 40
Obr. 4.6 Histogram čásů potřebných pro zápis proměnné do PLC ... 41
Obr. 4.7 Histogram časů pořebných pro zápis + čtení proměnné z PLC ... 43

Seznam tabulek

Tab. 2.1 Účinnosti světelnýh zdrojů ... 8
Tab. 2.2 Porovnání zdrojů osvětlení obecně pro strojové vidění ... 13
Tab. 2.3 Porovnání zdrojů osvětlení pro žonglér ... 13
Tab. 4.1 Mezní hodnoty pro zpoždění komunikace ... 40
Tab. 4.2 Změřené odezvy pro různé časové odstupy mezi měřeními ... 42
Tab. 4.3 Reálné časy uspání vlákna funkcí Sleep ... 42

Kapitola 1

1. Úvod

1.1. Žonglér

Žonglér je název automatizovaného zařízení, které bylo postaveno na katedře řídící techniky ČVUT k výuce řízení rychlých servopohonů, které nacházejí široké uplatnění v průmyslových technologiích. Stroj se skládá z dvou žonglujících paží, které jsou každá složená z dvou rychlých třífázových synchronních motorů. Silnější z nich pohání vertikální lineární modul, na kterém je umístěn menší motor, na jehož hřídeli je raménko se záchytnou miskou. Poslední lineární motor je umístěn samostatně mezi oběma žonglujícími moduly a je vybaven taktéž záchytnou miskou. Výsledné trajektorie záchytných misek umožňují vyhození koule z podavače do libovolné paže a přehození koule z jedné paže do druhé.

Obr. 1.1 Čelní pohled na žonglér

1.2. Projekt Žonglér

Tato práce je součástí projektu, který měl za úkol realizovat řídící systém samostatně žonglující se třemi koulemi bez zásahu obsluhy. Současně se mnou na tomto úkolu pracovali další 3 studenti. Pavel Jaroš, který se ve své práci [9] zabýval identifikací stroje, vytvořil základní modely mechaniky a termodynamiky pohonů a nad modely navrhl kinematiku pro přehazování třemi koulemi. Luboš Prudek, který vytvořil řízení celého procesu na úrovni PLC a rozšířil systém o ovládání pátého lineárního modulu pro nadhazování spadlých koulí, a Jan Kment, který se podílel na materiálních úpravách žongléru.

Cílem projektu bylo zůčastnit se mezinárodní soutěže European Industrial Ethernet Award pořádanou společností B&R Automation. Soutěž byla zaměřena na kreativní a inovativní projekty, které využívají Ethernet Powerlink.

1.3. Cíle práce

Úkolem mé práce bylo realizovat zpětnou vazbu o správném chodu stroje, tudíž poskytnout informaci o přítomnosti všech tří koulí. V případě, že by jedna nebo více koulí byly špatně zachyceny bude zajištěno, aby se žonglér dostal do správného stavu. Systém navíc kontroluje počáteční stav žongléra po zapnutí, zdali jsou nějaké koule již přítomny ve vyhazovacích ručkách a informuje o tom řídící PLC, aby mohlo následně vykonat potřebné akce k inicializaci žonglovacího cyklu.

Dalším cílem bylo navrhnout a realizovat opatření pro potlačení rušivých světelných vlivů na rychloběžnou kameru, kterou je stroj snímán. To nejen z důvodu spolehlivého provozu kontroly přítomnosti koulí, ale i za účelem analyzování možnosti zabezpečit stroj pomocí kamery a následného vyhodnocování pozic a rychlostí částí stroje. Za tímto účelem bylo nutné výtvořit takové světelné podmínky, které umožní rozeznávat pozice částí stroje v co možná největší ploše.

1.4. Členění práce

Jako první bylo nutné pro splnění cílů navrhnout odpovídající osvětlení. Řešením osvětlení žongléru se zabývá druhá kapitola, ve které jsou popsány různé druhy osvětlení používané v aplikacích strojového vidění a interakce světla s materiály. Z nich je na konci kapitoly vybráno finální řešení.

V další, třetí kapitole, je rozebrána část strojového vidění týkající se softwarové implementace. Jsou v ní zmíněny metody rozpoznávání obrazu, ze kterých jsem poté vybral použitelné pro moji aplikaci a pro rozeznávání poloh vyhazovacích misek při analýze zabezpečení.

Čtvrtá kapitola je věnována finální aplikaci pro detekci správného chodu stroje. Je v ní popsána komunikace mezi aplikací a řídícím PLC pomocí PVI a zvážena spolehlivost systému.

Poslední kapitola popisuje úlohu připravenou pro výuku.

Kapitola 2

2. Potlačení rušivých vlivů pomocí osvětlení

2.1. Strojové vidění

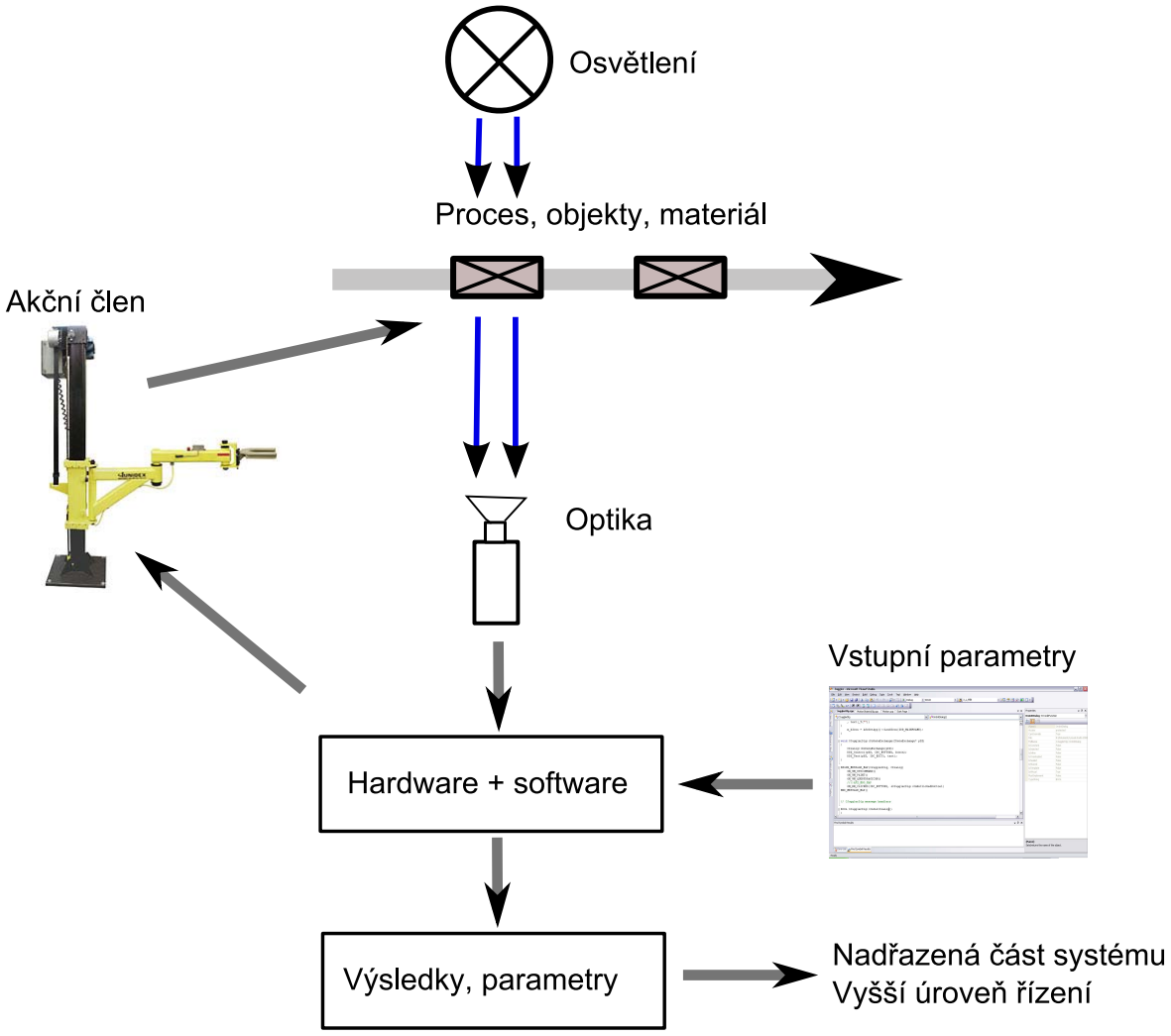
Systémy pro strojové vnímání se skládají z mnoha částí, z nichž každá má v systému jako celku svoji významnou roli. Volba těchto částí a jejich vlastností poté ovlivňuje celý proces zpracování obrazu od vzniku nového snímku až po konečnou interpretaci obrazu jako celku. Celý systém pro strojové vnímání bychom mohli rozdělit na tyto základní části:

- Osvětlení
- Optika
- Použitý hardware
- Softwarová implementace

Návrh těchto částí se skládá z různých technických odvětví a jejich kooperací se dosahuje požadovaných výsledků. Základní struktura systému pro strojové vnímání je zobrazena na obrázku 2.1.

2.2. Osvětlení

Osvětlení v aplikacích pro strojové vnímání hraje jednu z klíčových rolí. Světlo jak je znázorněno v obrázku 2.1 můžeme považovat za informační kanál mezi reálným světem a systémem pro zpracování obrazu. Cílem osvětlení tedy je vytvořit takové podmínky, aby se kanálem přenášelo co nejvíce užitečných informací, které lze v dalších částí oddělit od nedůležitých a správně je interpretovat. Jako informaci v obraze můžeme považovat například kontrast, jas, tmavost, stíny, struktury, odlesky a mnoho dalších. Na všechny tyto případy má vliv právě osvětlení daného objektu. Kromě užitečných informací se však při snímání scény mohou v obraze objevit rušivé elementy, které mohou být následně špatně interpretovány daným algoritmem. Požadavkem na osvětlení je pokud možno co nejvíce tyto rušivé vlivy eliminovat. V dalších odstavcích budou rozebrány světelné zdroje, které se používají v aplikacích strojového vidění a jejichž použití jsem zvažoval. Podrobnější informace o jednotlivých zdrojích lze nalézt v [4].



Obr. 2.1 Základní struktura systému strojového vidění

2.2.1. Požadavky na osvětlení scény v průmyslových aplikacích

Osvětlení jako celek je složen z více částí, které zpravidla zahrnují:

- Světelné zdroje
- Mechanicky nastavitelné elementy
- Světlovody
- Stabilizační a řídící elektroniku
- Ochranné prostředky
- Robustní kabeláž

Průmyslové využití systémů strojového vidění klade na tyto části specifické nároky. Lze je rozdělit do oblastí, které uvedu v následujících odstavcích

2.2.2. Požadavky na optické vlastnosti:

- Vlnová délka – barva světla
- Stálost světelného zdroje
- Nezávislost zdroje na okolní teplotě
- Identické zdroje jednoho druhu
- Homogenní nebo definovaný jasový profil
- Světelné zařízení umožňující osvětlovat trvale nebo záblesky

2.2.3. Požadavky na elektrické vlastnosti

- Nízké bezpečné pracovní napětí
- Pracovní napětí typické pro průmyslové aplikace
- Ochrana proti špatnému připojení – přepólování
- Standardní půmyslová rozhraní – PLC vstupy/výstupy, USB, Ethernet apod.
- Řídící obvody uvnitř zařízení
- Zabezpečení proti přehřátí
- Robustní kabeláž odolávající provozním podmínkám

2.2.4. Požadavky na mechanické vlastnosti

- Robustní konstrukce, odpovídající průmyslovým standardům
- Jednoduchá montáž
- Ochrana proti vibracím(zrychlení součástí)
- Ochrana proti vodě, prachu, vlhkosti a dalším vlivům prostředí

Ze všech výše uvedených vlastností se obecně nejvíce dbá na vlastnosti související s jednoduchostí instalace – integrované řešení bez nutnosti přídavných zařízení. Dále jsou kladeny vysoké požadavky na životnost světelných zdrojů. Běžně je požadována životnost delší než 10 000 hodin, což je déle než rok provozu (rok = 8760h). Nakonec bych zmínil také nutnost, aby byly všechny komponenty standardní. V případě poruchy nebo havárie lze poté nefunkční části vyměnit v nejkratším možném čase, kterým je pouze čas doručení od dodavatele.

2.3. Zdroje osvětlení

Zdroje světla používané v aplikacích strojového vidění bychom mohli rozdělit podle způsobu přeměny energie na světlo.

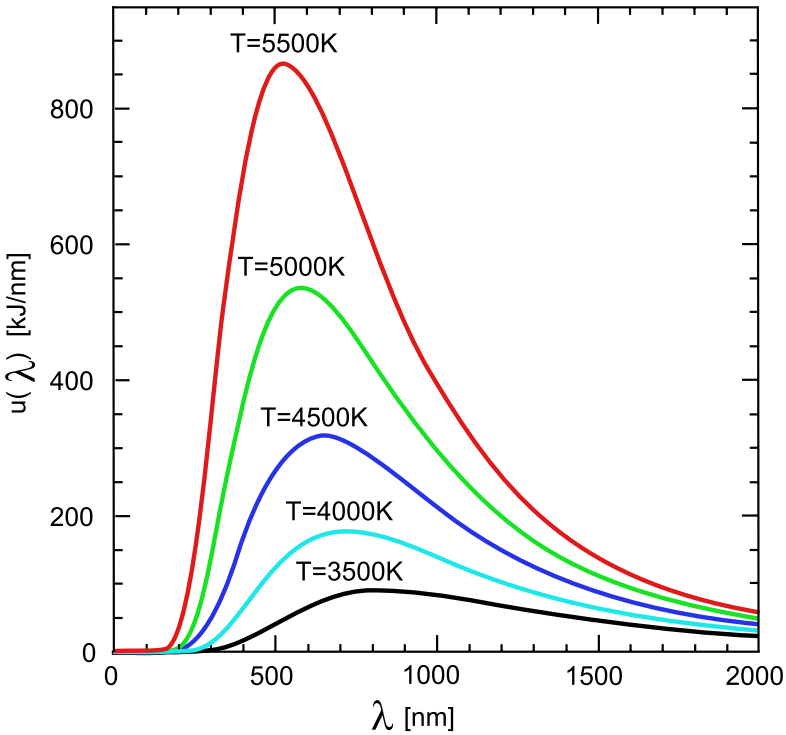
- Zdroje využívající teplotního buzení
- Zdroje pracující na principu luminiscence

2.3.1. Teplem buzené zdroje

Světlo vyzařované teplotními zdroji se skládá z mnoha vlnových délek, mají tudíž široké spojité spektrum. Při použití obrazového snímače CCD nebo CMOS je využita jen malá část emitovaného spektra. Teplotní zdroje se v tomto kontextu vyznačují malou účinností. Efektivita zdroje roste s teplotou daného zdroje na základě Wienova posunovacího zákona:

$$\lambda_{\max} = \frac{b}{T} \tag{1}$$

Kde λ_{max} je vlnová délka maxima vyzařování. S rostoucí teplotou se maximum posunuje ke kratším vlnovým délkám a světlo se jeví jako bělejší.



Obr. 2.2 Wienův posunovací zákon

2.3.2. Luminiscenční zdroje

Luminiscenční zdroje vyzařují pouze úzké spektrum vlnových délek, které je určeno například chemickým složením polovodiče u LED diod. Díky selektivnosti použitého snímacího senzoru je možné dosáhnout vysoké účinnosti oproti tepelně buzeným zdrojům. Následující tabulka srovnává různé typy světelných zdrojů na základě jejich účinnosti, která je určena poměrem výstupního světelného toku a eletrického výkonu daného zdroje, přičemž teoretická maximální účinnost pro bílé světlo je 225 lmW^{-1}.

Typ zdroje	**Účinnost [lm W^{-1}]**
Klasická žárovka 100W	10-20
Halogenové žárovky	max 35
Lineární zářivky	40 - 100
Metal halogenidová výbojka	100
Xenonová výbojka	max 60
LED	přes 100
Nízkotlaká sodíková výbojka	max 200

Tab. 2.1 Účinnosti světelnýh zdrojů

V aplikacích strojového vidění však musíme brát v úvahu spektrální charakteristiku daného zdroje v souvislosti s odrazivými vlastnostmi osvětlovaných povrchů a vlastnostmi snímacího prvku.

2.3.3. Klasické/halogenové žárovky

Jak už bylo uvedenno v části 2.3.1, teplotně buzené světelné zdroje se vyznačují velmi malou účinností. Široké pásmo vlnových délek, které tyto zdroje vyzařují (od UV až po IR) způsobuje, že pouze 7 procent energie je přeměněno na viditelné světlo. Tuto účinnost lze vylepšovat zvyšováním pracovní teploty světelného zdroje a posouváním maxima vyzařování směrem ke kratším vlnovým délkám, což je ovšem na úkor jeho životnosti.

Výhody klasických/halogenových žárovek

- Spojité spektrum (barevná teplota bílého světla 3000 až 3500K)
- Nízké pracovní napětí
- Schopné provozu v teplotách přesahujících 300 °C

Nevýhody:

- Krátká životnost
- Změna jasu v průběhu životnosti
- Použitelné pouze pro statické osvětlení (bez záblesků)
- Výkyv napájecího napětí způsobí výkyv světelného toku
- Produkce velkého množství tepla

2.3.4. Metal halogenidové výbojky

Tyto výbojky pracují s vysokou účinností až 100 lmW^{-1}. Produkují velmi jasné studené světlo. Pro správnou funkci musí pracovat uvnitř za vysokého tlaku, který se vytváří počátečním zahřátím kovových par na pracovní teplotu.

Výhody:

- Vysoká intenzita světla
- Studené světlo

Nevýhody:

- Relativně krátká životnost
- Pracují s vysokým napětím – problémy s EMC
- Jednotlivá spektrální pásma mohou způsobit špatné podání barev
- Teplotní omezení
- Produkují teplo

2.3.5. Xenonové výbojky

Xenonové výbojky produkují světlo s vysokou intenzitou a výborným barevným podáním. Spektrum světla se ve viditelné oblasti přibližuje spojitému spektru. Jejich nespornou výhodou je možnost produkovat velmi silné záblesky, což ještě více zvyšuje jejich využitelnost. V aplikacích strojového vidění se používají hlavně při snímání rychlých procesů, které vyžadují vysokou intenzitu osvětlení.

Výhody:

- Vysoká intenzita světla
- Velmi dobré barevné podání (barevná teplota 5500 – 12000K)
- Možnost stálého/zábleskového osvětlení
- Možnost produkovat více než 200 záblesků za sekundu

Nevýhody:

- Pracují s vysokým napětím – možnost ovlivnění okolních měření
- Velmi drahé – řídící elektronika
- Intenzita záblesků se mění v rozmezí 10%
- Intenzita záblesků se snižuje s jejich počtem

2.3.6. Lineární zářivky

Lineární zářivky jsou nízkotlaké výbojky, jejichž světlo má většinu spektra v UV oblasti (obrázek 2.3). Povrch zářivky je zevnitř pokryt luminoforem, který konvertuje záření z UV do viditelné oblasti spektra. Složení luminoforu je při výrobě voleno na základě požadovaného spektra výstupního světla. Lze si tedy ze sortimentu vybrat zářivku s požadovanou teplotou barev.

Zářivky jsou napájeny střídavým napětím, což způsobuje jejich blikání o dvojnásobné frekvenci napájecího napětí (dosvit luminoforu se pohybuje okolo 1/1000 s). To způsobuje stroboskopický efekt a rovněž může způsobit problémy při použití v aplikacích strojového vidění. Tyto jevy eliminuje použití elektronického předřadníku s dostatečně vysokou frekvencí.

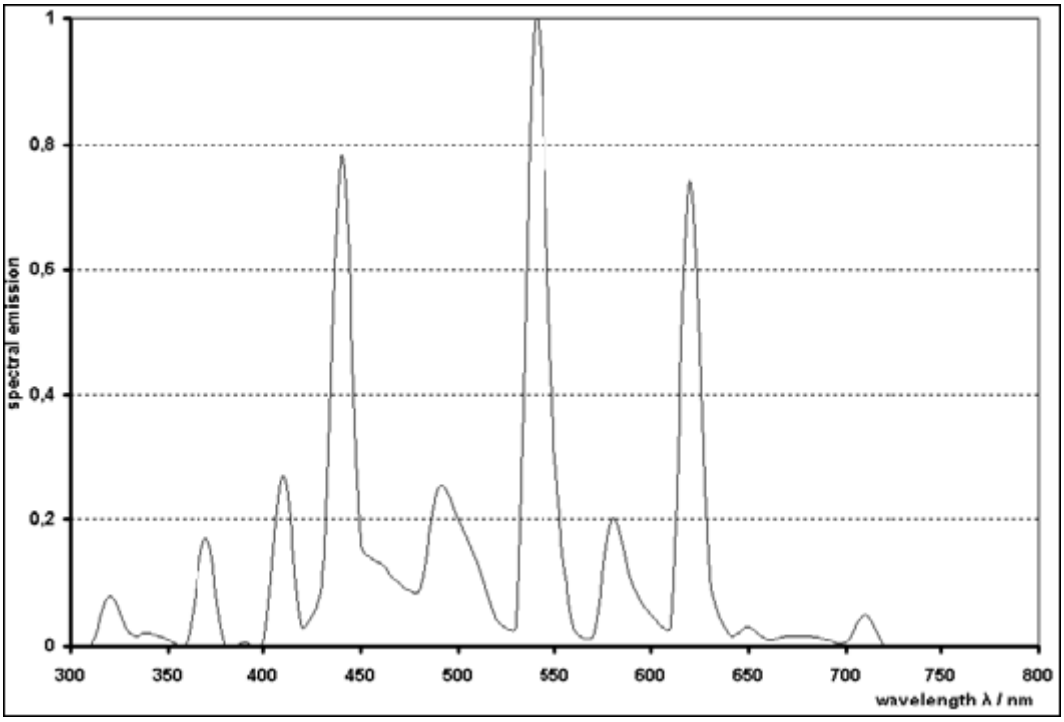
Výhody:

- Velmi levné
- Volitelná teplota barev
- Vhodné pro osvětlení velkých oblastí

Nevýhody

- Malá tvarová flexibilita
- Omezená životnost, stárnutí cca 90% světelného toku po 10 000 hodinách
- Při použití elektronického předřadníku možné problémy s EMC
- Použitelné pouze pro statické osvětlení bez záblesků

Obrázek převzat z [4]



Obr. 2.3 Spektrum vyzařovaných vlnových délek lineární zářivky

2.3.7. LED diody

Vývoj LED diod v posledních několika letech zaznamenal dramatický vývoj směrem kupředu, čímž se staly LED diody nejvhodnějším a nejpoužívanějším světelným zdrojem pro aplikace strojového vidění.

Jejich výhoda tkví v možnosti produkovat téměř monochromatické světlo s velmi úzkým spektrem vlnových délek. Navíc se vyznačují dobrou účinností. Vlastnosti LED diod se díky novým technologiím neustále vylepšují a stávají se využitelnějšími v čím dál více aplikacích.

LED diody mohou pracovat v jak v režimu statického osvětlení, tak mohou produkovat pouze záblesky. Navíc mohou být krátkodobě přetěžovány pro zvýšení intenzity osvětlení bez následného poškození.

Výhody:

- Malé rozměry, tvarová flexibilita, odolnost proti vibracím
- Vysoká životnost - desítky tisíc hodin
- Produkce téměř monochromatického nebo bílého světla
- Jednoduchá řídící elektronika
- Nízká produkce tepla
- Po dobu životnosti prakticky bezúdržbové
- Bezpečné pracovní napětí

Nevýhody:

- Omezené použití ve vysokých teplotách
- Stárnutí v důsledku vysokých teplot
- Stárnutí bílých LED diod způsobuje změnu v teplotě barev

2.3.8. Lasery

Lasery nacházejí v aplikacích strojového vidění své uplatnění hlavně v úlohách pro 3D vidění. Jejich průmyslové použití však naráží na požadavek bezpečnosti pro lidi aby při kontaktu nemohlo dojít například ke zranění oka.

Jako příklad využití laseru bych uvedl měření tloušťky skla, kde se využívá odrazu na obou přechodech prostředí vzduch-sklo a sklo-vzduch

Výhody:

- Vysoká intenzita osvětlení
- Produkce téměř paralelního svazku světla
- Svazek světla může mít různý tvar

Nevýhody:

- Problémy s EMC
- Použití jen v úzkém okruhu aplikací

	Klasické/halogenové žárovky	Metal halogenidové výbojky	Xenonové výbojky	Lineární zářivky	LED diody
Vlnová délka (barva světla)	++	+	+	+	++
Stárnutí	-	-	-	--	++
Závislost na teplotě	+	-	-	-	-
Osvětlení záblesky, pulsy	-	-	++	-	+
Homogenní jasový profil	+	-	+	++	+
Bezpečné pracovní napětí	+	-	--	-	++
Nenáročná řídící elektronika	+	+	--	+	+
Variabilní konstrukce	-	-	-	--	++
Produkce tepla	-	-	+	+	+
Mechanická odolnost - vibrace atd.	-	-	-	-	++

Tab. 2.2 Porovnání zdrojů osvětlení obecně pro strojové vidění

	Klasické/halogenové žárovky	Metal halogenidové výbojky	Xenonové výbojky	Lineární zářivky	LED diody
Vlnová délka (barva světla)	+	+	+	+	+
Stárnutí	-	-	-	--	++
Produkce tepla	-	-	+	+	+
Složitost vhodné konstrukce	-	-	-	+	-
Řídící elektronika	+	+	--	+	+
Homogenní osvětlení	-	-	-	++	+
Dostupnost/náročnost realizace	-	-	--	++	-

Tab. 2.3 Porovnání zdrojů osvětlení pro žonglér

2.4. Interakce světla a materiálu

Současně s parametry zdrojů osvětlení je nutné mít představu o interakci světla a povrchu objektu. Bohužel v úlohách strojového vidění nastává velmi často situace, kdy se vlastnosti povrchů mění, nebo nejsou specifikovány.

Při dopadu světla na daný materiál nastává kombinace odrazu, pohlcení a prostupu světla.Platí při tom:

$$R + T + A = 1 \qquad R = \frac{\Phi_r}{\Phi_0} \tag{2}$$

kde R je odrazivost povrchu, Φ_r je odražený světelný tok a Φ_0 je dopadající světelný tok. Analogicky to platí pro absorbci A a prostup světla T. Množství odraženého světla se však mění na základě úhlu zdroje světla a pozorovatele vzhledem k normále povrchu materiálu. Tento vztah popisuje BRDF (Bidirectional Reflectance Distribution Function)

$$f(\Theta_i, \varphi_i, \Theta_v, \varphi_v) = \frac{\delta L(\Theta_i, \varphi_i)}{\delta E(\Theta_v, \varphi_v)} [sr^{-1}] \tag{3}$$

- L je změřená zář [$Wm^{-2}sr^{-1}$]odražená od povrchu ve směru Θ_v, φ_v
- E je ozáření [Wm^{-2}]dané směrem Θ_i, φ_i
- Θ, φ je azimut a polární úhlel v kulových souřadnicích

Významné zjednodušení nastává pro materiály, které odrážejí světelnou energii do všech směrů rovnoměrně– Lambertovské povrchy.Potom se BRDF dá vyjádřit jako:

$$f(\Theta_i, \varphi_i, \Theta_v, \varphi_v) = \frac{\rho(\lambda)}{\pi} [sr^{-1}] \tag{4}$$

- $\rho(\lambda)$ je koeficient odrazivosti – Albedo

Materiál	Odrazivost	Absorbce	Prostup materiálem
Bílá barva	0,7 - 0,85	0,15 - 0,3	0
Žlutá barva	0,48 - 0,52	0,48 - 0,52	0
Zelená barva	0,12 - 0,2	0,8 - 0,88	0
Modrá barva	0,07 - 0,1	0,93 - 0,9	0
Bílý papír	0,6 - 0,85	0,05 - 0,39	0,05 - 0,41
Bílý plast	0,2 - 0,4	0,1 - 0,2	0,4 - 0,6
Hliník leštěný	0,6 - 0,72	0,28 - 0,4	0
Ocel leštěná	0,55 - 0,6	0,4 - 0,45	0

Tab. 2.4 Orientační hodnoty odrazivosti, absorbce a prostupnosti vybraných materiálů

2.5. Interakce světla na rozhraní dvou prostředí

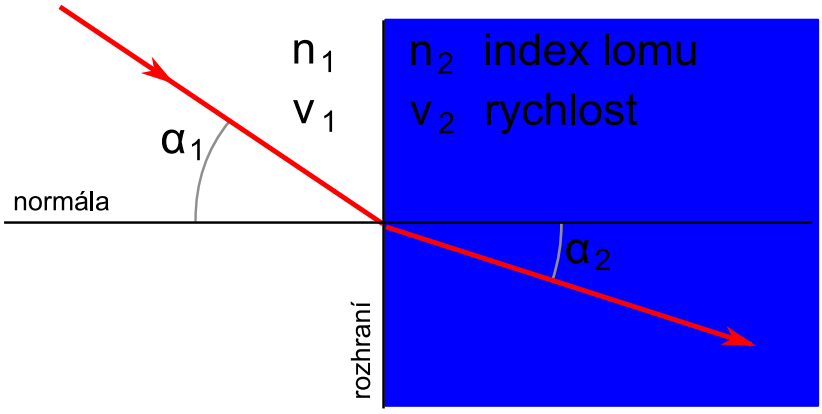
Při přechodu světla z jednoho optického prostředí do druhého nastává jev nazývaný lom světla. Lom světla závisí na rychlosti šíření vlnění v daných prostředích. Proto je zaveden absolutní index lomu, který je vyjádřen jako:

$$n = \frac{c}{v} \tag{5}$$

- c je rychlost světla ve vakuu
- v je rychlost světla v daném prostředí

Chování světla na rovinném rozhraní dvou prostředí popisuje Schnellův zákon, který je odvozen na základě Huygensova principu šíření vlnění, který úvádí, že každý bod na čele vlnoplochy lze brát jako elementární zdroj nového vlnění. Výsledný tvar Schnellova zákona je vyjádřen jako:

$$\frac{\sin\alpha_1}{\sin\alpha_2} = \frac{n_2}{n_1} = \frac{v_1}{v_2} \tag{6}$$



Obr. 2.4 Schnellův zákon

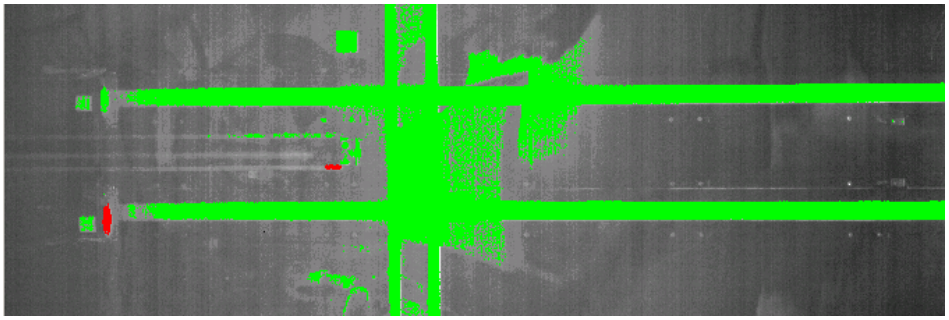
Důsledkem je, že se světlo při přechodu z opticky řidšího prostředí (nízký index lomu) do opticky hustšího lámě směrem ke kolmici. Naopak při přechodu z hustšího do řidšího prostředí se láme od kolmice. V druhém případě může nastat takzvaný totální odraz, kdy je úhel α2 větší nebo roven π/2. V tomto případě je veškeré světlo odraženo zpět do prvního prostředí. Mezní úhel se vypočítá dosazením za $\alpha_2 = \pi/2$ a vyjádřením:

$$\alpha_m = \arcsin\frac{n_2}{n_1} \tag{7}$$

2.6. Žonglér a rušivé vlivy

Celý proces žonglování se z důvodu bezpečnosti odehrává v prostoru uzavřeném ochranným plexisklem. Původní osvětlení se skládalo z lineárních zářivek umístěných pod stropem žongléra. Celkové světlo se tedy skládálo ze světla z okolí a světla produkovaného zářivkami.

Při snímání scény s touto konfigurací osvětlení se výrazně měnila spolehlivost rozeznávání koulí. Spolehlivost se měnila v závislosti na denní době a na světle pronikajícím okny do místnosti, které způsobovalo odrazem od předmětů a následně od ochranného plexiskla světlé oblasti ve výsledném snímku.



Obr. 2.5 Vliv okolního světla na zpracování obrazu

Na obrázku 2.5 jsou tyto světlé oblasti znázorněny. Pokud byla prahovací úroveň nastavena tak aby byl rozeznán čtvercový symbol na levé straně, vyskytly se v obraze oblasti s jasem vyšším než byl daný práh (zelené oblasti). V těchto místech by poté objekt nemohl být rozeznán.

Dalším nepříznivým faktorem byla nehomogenita osvětlení způsobená umístěním lineární zářivky pod stropem žongléra. Při této konfiguraci se značně lišila intenzita osvětlení mezi horní a dolní částí žongléra. Navíc při osvětlení letící koule z vrchu byla spodní část ve stínu, tudíž se při prahování měla koule tendenci se jevit jako půlkruh.

Dále se při snímání vyhazovacích ruček projevovala dlouhá doba expozice, kdy při vysokých rychlostech byly dané objekty v obraze rozmazané.

2.6.1. Požadavky na řešení osvětlení žongléra

V našem případě požadavky na osvětlení vycházejí hlavně ze způsobu segmentace obrazu pomocí prahování. Cílem osvětlení je tedy zajistit co největší rozdíl mezi jasem rozeznávaných objektů a jasem okolí, což bylo rušeno odrazy od plexiskla, které tento rozdíl zmenšovalo a mnohdy i převyšovalo.

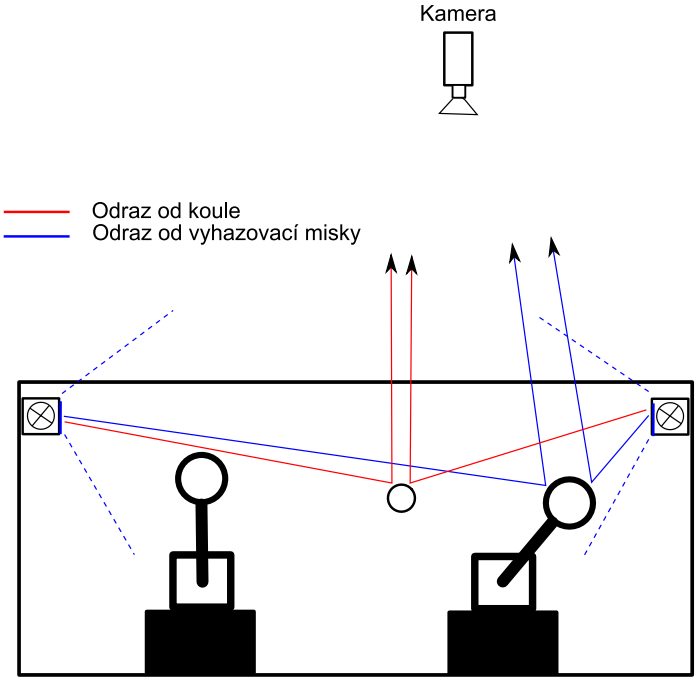
Dále by osvětlení mělo mít stejnou intenzitu ve všech místech, kde budou objekty rozeznávány. Intenzita osvětlení by také měla být ze všech směrů stejná a osvětleny by měly být jen potřebné části, aby nedocházelo k odrazům od jiných objektů, které by se mohly projevit při segmentaci.

Aby mohly být snímány objekty i ve vysoké rychlosti je nutné snížit dobu expozice tak, aby obraz nebyl rozmazaný. To klade požadavky ne celkovou intenzitu osvětlení.

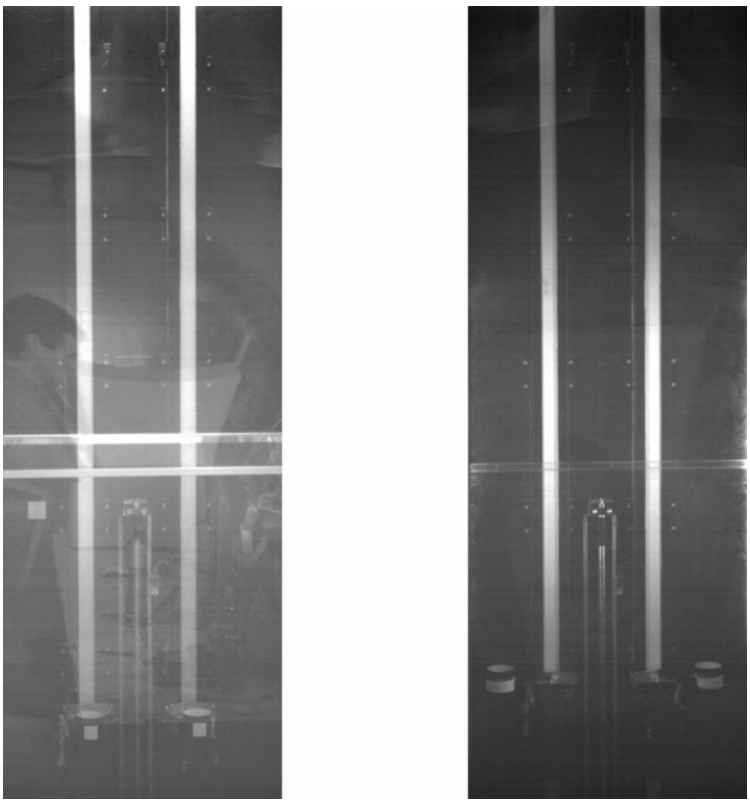
Výsledné řešení musí být samozřejmě mechanicky odolné vůči případnému střetu se špatně vyhozenou či odraženou koulí.

2.6.2. Výsledné řešení osvětlení

V návaznosti na požadavky bylo vybráno osvětlení pomocí lineárních zářivek, které poskytují možnost rovnoměrně osvětlit širší oblasti, představují jednoduché a cenově přijatelné řešení(tabuka 2.3). Zářivka (28W) je umístěna v ochranném hliníkovém U profilu chránícím svítidlo před zásahem koulí. Důležitým faktorem je rozmístění svítidel uvniř pracovního prostoru aby byly rozeznávané objekty pokud možno nasvíceny co nejvíce a pozadí co nejméně. Tomu napomáhá také hliníkový U profil, díky kterému je světlo dopadající na pozadí omezováno. V souvislosti s prostorovými možnostmi jsem zvolil rozmístění zobrazené na obrázku 2.6. Tento způsob vychází z techniky osvětlení v temném poli, kdy se uplatňuje pouze světlo odražené od objektu, který je pak světlý na tmavém pozadí. Při volbě výkonu zářivky bylo přihlédnuto ke zkušenostem s již použitými zářivkami pod stropem žongléru.



Obr. 2.6 Schéma osvětlení systému



Obr. 2.7 Porovnání osvětlení před úpravami(vlevo) a po úpravách (vpravo)

Kapitola 3

3. Detekce správného chodu stroje

V předchozí kapitole byl popsán výběr a konfigurace osvětlení pro aplikace strojového vidění jakožto jedné ze základních komponent systému. Další základní části zmíněné v 2.1 optika a použitý hardware jsou v mém případě dány již použitými prostředky. Systém je snímán monochromatickou průmyslovou kamerou Mikrotron MC1310, ze které je obraz přenášen do PC přes rozhraní Camera Link používané v průmyslových aplikacích strojového vidění. PC je vybaven grabovací kartou Inspecta 5. Díky použité PCI sběrnici a její maximální přenosové rychlosti 133 MB/s pracuje Camera Link v „Base“ režimu se sníženou maximální rychlostí. Z tohoto důvodu je rychlost snímání nastavena na 100 snímků za sekundu. PC je vybaven operačním systémem Windows XP z důvodu nedostupnosti ovladačů pro grabovací kartu pro vhodnější prostředí například RT Linux.

3.1. Softwarová implementace aplikací strojového vidění

Posledním zásadním článkem systému pro strojové vidění je softwarová implementace. Řešení této části se významně liší na základě dané úlohy. Základní koncepci však lze shrnout do následujících kroků:

- Pořízení snímku
- Předzpracování
- Lokalizace oblasti
- Získávání vlastností
- Interpretace vlastností
- Formování výstupního výsledku
- Ošetření vstupních/výstupních operací

Pořízení snímku

V této části je ošetřeno vše spojené s vytvořením nového snímku, což může představovat samotné spuštění akce pro pořízení například externím spouštěcím signálem nebo ošetření akcí spojených s osvětlením jako zapnutí, vypnutí a aktivace osvětlení či blesku.

Předzpracování

Předzpracování může být realizováno jak softwarově tak v případě podpory dané komponenty i hardwarově v kameře nebo grabovací kartě. Nejčastější akce spadající do předzpracování obrazu jsou jasové transformace, aplikace filtrů a korekce stínu v případě nehomogenního osvětlení. Některé z těchto akcí jsou náročné na výpočetní výkon, tudíž je jejich použití omezeno u rychlých procesů.

Lokalizace oblasti

V mnoha případech není nutné vyhodnocovat celý snímek, proto se vymezují v obraze takzvané ROI (regions of interest), což jsou oblasti snímku vybrané tak, aby pokryly požadovanou oblast spolu s případnými tolerančními pásmy pozice daného objektu. Vymezení ROI zvyšuje robustnost dané úlohy a přináší úsporu výpočetního času u rychlých procesů.

Získávání vlastností

Do této stěžejní části spadá aplikace algoritmů pro získávání znalostí. Základní techniky pro získávání vlastností jsou:

- Blob analýza (Blob analysis)
- Detekce hran
- Detekce kontur
- Template matching
- Morfologické operace
- Analýza textur
- Geometrické transformace
- OCR

Interpretace vlastností

V této části probíhá měření, porovnávání a verifikace výsledků z předchozích kroků. Interpretace představuje zobrazení

$$f : R^n \rightarrow \Omega$$

- R^n je množina vlastností
- Ω je množina objektů

Formování výstupního výsledku

Data získaná z obrazu jsou v tomto kroku připravována pro další použití, případně jsou ošetřeny chybné stavy.

Ošetření vstupně výstupních operací

Tato část zajišťuje komunikaci s ostatními částmi systému, kterými mohou být například řídící jednotky, nadřazené stupně řízení, datová úložiště nebo zařízení pro vizualizaci výsledků.

3.2. Metody používané k získávání vlastností

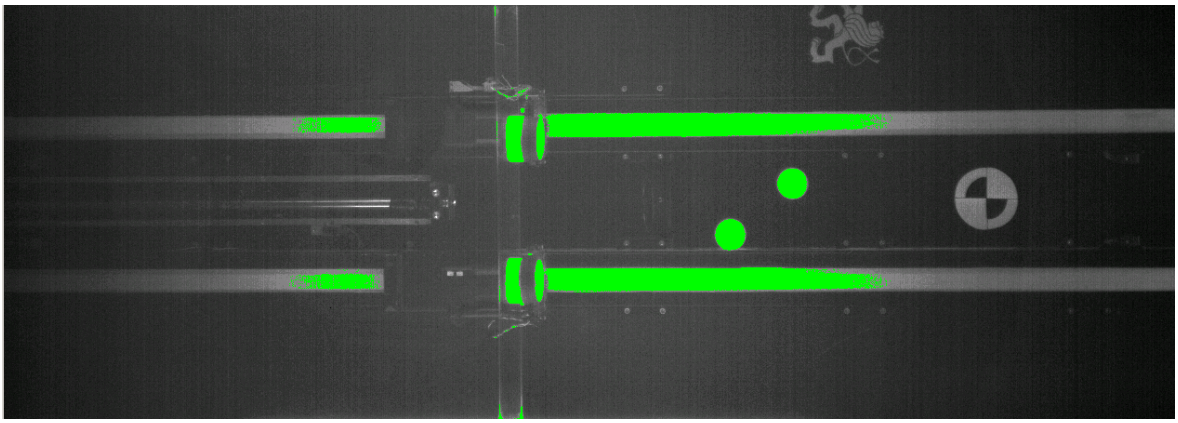
3.2.1. Blob analýza

Blob(binary large objects) analýza využívá k získávání vlastností o objektech jejich jasové hodnoty ve snímku. Základním principem je rozdělení snímku na popředí, které představuje hledané objekty a pozadí, na kterém se objekty vyskytují. Aby mohlo být odděleno popředí od pozadí, musí se lišit v jasových hodnotách. Oddělení popředí a pozadí se obecně nazývá segmentace snímku. Výsledkem segmentace jsou oblasti obrazu (regiony), které spadají do popředí snímku.

Pro segmentaci snímku se nejčastěji používá prahování, avšak jeho nevýhodou je citlivost na stálé světelné podmínky při pořizování obrazu. V některých případech je možné použít dynamické prahování, kde je prahovací úroveň určována pro každý snímek zvlášť, například automatickou analýzou histogramu nebo aplikací lokálního filtru vhodné velikosti.

Výsledné obalsti snímku se dají poté analyzovat mnoha metodami a určit tak jejich vlastnosti(počet pixelů, těžiště oblasti, orientace, podobnost tvaru atd.) nebo je použít k dalšímu zpracování například pro vymezení oblasti ROI.

Blob analýza se vyznačuje velikou variabilitou a vysokou výkonností s malými nároky na výpočetní výkon. Její nevýhodou je závislost na proměnlivých jasových podmínkách. V případě použití prahování pro detekci letících koulí jsem byl omezen bílými pásy lineárních modulů, před kterými nemohou být koule spolehlivě rozpoznány. Pro kontrolu přítomnosi koulí v cyklu žonglování však postačila část obrazu mezi pásy, kde je tmavé pozadí(Obr. 3.13). Tento problém s bílým pozadím se v případě rozpoznávání vyhazovacích misek nevyskytl.



Obr. 3.1 Příklad zpracování obrazu blob analýzou(prahování)

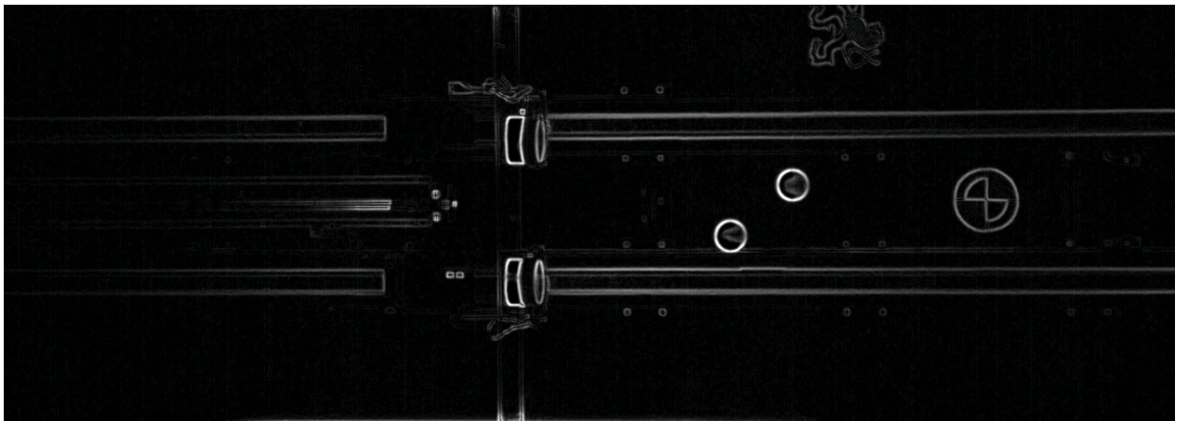
3.2.2. Detekce hran

Hranou rozumíme místo v obraze, kde se jasové hodnoty v závislosti na poloze v obraze rychle mění. Hledání hrany v obraze se tudíž převádí na úlohu hledání míst s vysokým modulem gradientu. Matematicky lze úlohu formulovat jako aplikaci filtru, který určuje maximum první derivace(filtry Roberts, Prewittová, Sobel, Canny aj.) nebo průchod druhé derivace nulou(Marr-Hilderth).

Principielně se úloha nalezení hran rozkládá na vhodné filtrování obrazu z hlediska odstranění šumu, určení místních gradientů a vybrání míst s vysokým gradientem například pomocí prahování.

Výhodou metody detekce hran je její (do určité míry) nezávislost na změnách osvětlení. V případě žongléra má tato metoda hlavní výhodu oproti blob analýze prahováním(Obr. 3.1), že je schopna detekovat kouli i v místech, kde se koule kryje s bílým pásem lineárního modulu. Proto vidím tuto metodu jako nejvhodnější pro detekci pozice koule v celém obraze, ne jen v části, kde se nachází na černém pozadí.

Nevýhodou je ve většině případů vysoká výpočetní náročnost dílčích algoritmů.



Obr. 3.2 Příklad zpracování obrazu pomocí detekce hran (Sobelův filtr)

3.2.3. Template matching

Základní myšlenkou template matching je hledání předem určeného vzoru v obraze. Hledaný vzor můžeme hledat na základě více vlastností, podle kterých se template matching dělí na následující metody.

Hledání na základě odstínů šedi

Toto je nejzákladnější metoda template matching. Může být použita za předpokladu, že se nemění příliš osvětlení scény a rozpoznávané části jsou celé viditelné. Touto metodou lze najít v obraze jednotlivé instance hledaného objektu, které mohou být natočené.

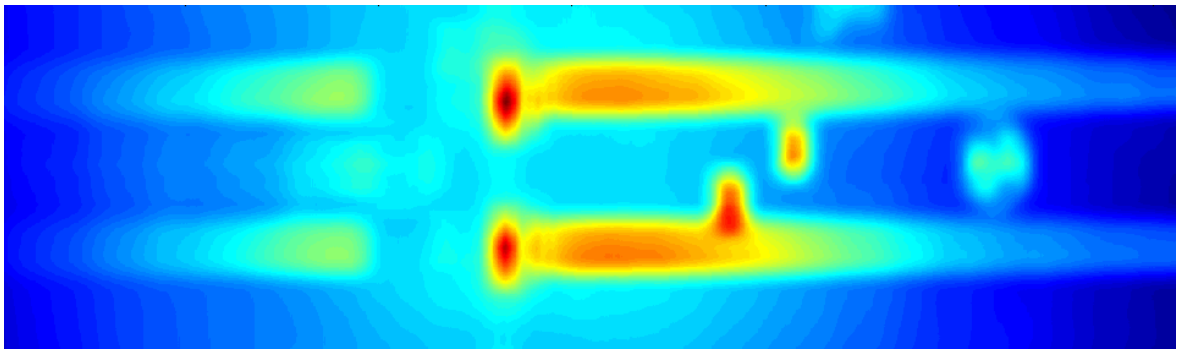
Hledání na základě tvaru

U hledání na základě tvaru jsou při vytváření vzoru mimo jasových hodnot vytvářeny také kontury objektu. Ty jsou uloženy ve vzoru a jsou hledány v obraze určeném ke zpracování. Využitím kontur pro hledání objektu se dosahuje nezávislosti hledání na změně osvětlení, natočení objektu, měřítku a navíc mohou být některé části objektu schovány.

Bodové hledání

U bodového hledání jsou ze vzoru extrahovány důležité body, které jsou poté použity pro hledání v obrazu. Tyto body mohou být přitom natáčeny, může se měnit jejich měřítko, či mohou být jinak transformovány. Výhodou bodového hledání je možnost najít objekty, které jsou zakřiveny v důsledku perspektivy.

Obecnou výhodou metody template matching je její robustnost a flexibilita. Nevýhodou je její vysoká náročnost na výpočetní čas.

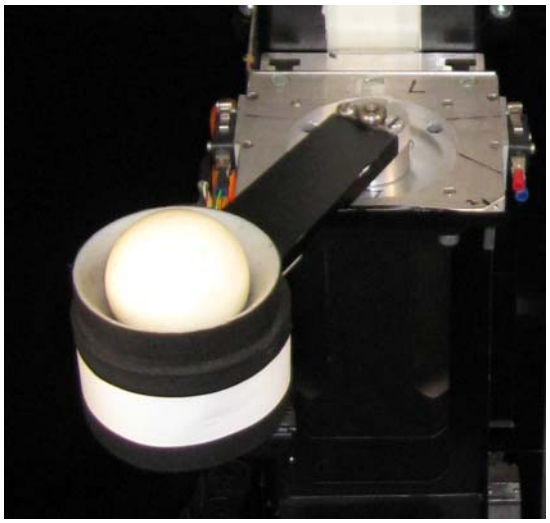


Obr. 3.3 Příklad zpracování obrazu pomocí template matching (hledání na základě odstínů šedi).

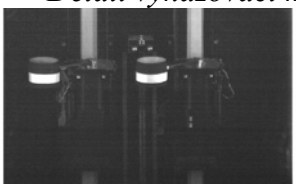
3.3. Objekty určené k rozpoznávání

Cílem mé aplikace je kontrola správného chodu stroje společně s detekcí případného pohybu s kamerou. Dále bylo cílem zvážit možnost případného zabezpečení systému proti kolizím pohyblivých částí. Pro splnění těchto úkolů je tedy nutné v obraze rozeznávat žonglující koule a vyhazovací misky. V případě koulí je tvar, materiál a jejich celkový obraz ve výsledném snímku pevně dán. Jedinou variabilní vlastností je jejich barva, která je logicky zvolená bílá pro zajištění největšího jasu na tmavém pozadí.

U vyhazovacích misek bylo nutné zvolit jejich povrch tak, aby je bylo možné ve snímku spolehlivě lokalizovat. Tento úkol ztěžoval fakt, že se vyhazovací miska otáčí kolem osy, tudíž bylo nutné zajistit její jednotný obraz ve výsledném snímku bez závislosti na poloze vůči ose otáční. Z těchto důvodů jsem zvolil řešení zobrazené na obrázku 3.4. Tím jsem docílil toho, že se bílý pruh ve výsledném obraze vždy zobrazí jako bílý obdélník mírně deformovaný díky pohledu kamery mírně shora.

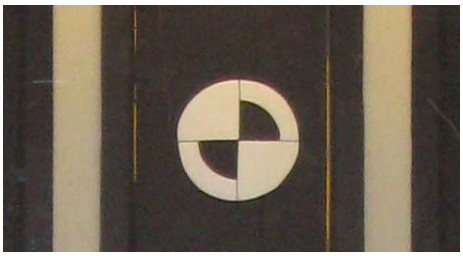


Obr. 3.4 Detail vyhazovací misky



Obr. 3.5 Vyhazovací misky ve snímaném obraze

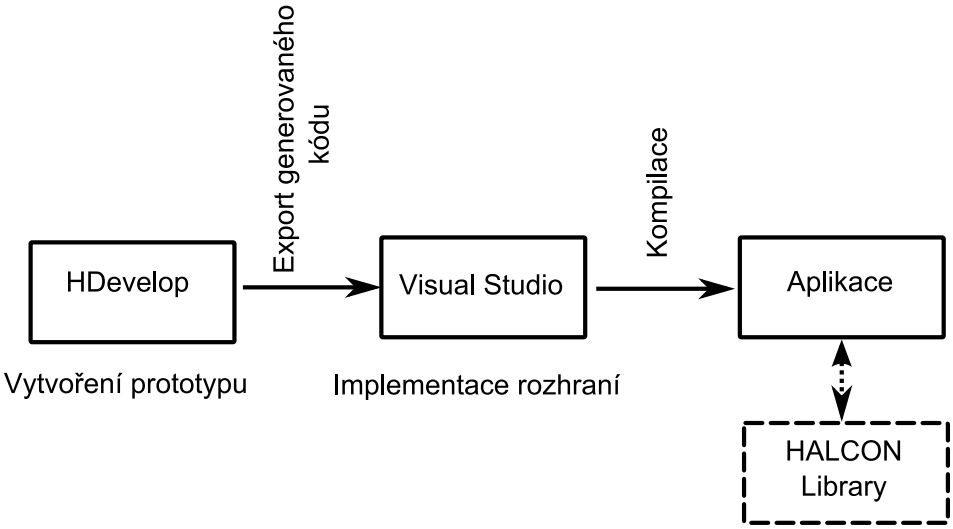
Pro detekci pohybu kamery jsem pevně umístil na zadní stěnu žongléra referenční znak(Obr. 3.6), jehož pozice v obraze se při pohybu kamery změní a bude signalizována chyba.



Obr. 3.6 Referenční znak pro detekci pohybu kamery

3.4. Algoritmy pro rozpoznávání objektů

Pro softwarové zpracování obrazu jsem použil knihovny HALCON, které poskytují komplexní soubor funkcí pro aplikace strojového vidění. Společně s knihovnami jsem využil i vývojový program HDevelop umožňující flexibilní návrh algoritmu s možností rychlé změny parametrů u používaných funkcí. Vytvořený program v HDevelop je možné poté exportovat do zdrojového kódu C, C++ nebo COM. Ten lze využít v dalších vývojových nástrojích jak je znázorněno na obr 3.7.



Obr. 3.7 Princip vývoje software s použitím knihoven HALCON a HDevelop

V úloze rozpoznávání poloh vyhazovacích misek a koulí je úkolem najít v obraze světlou oblast blížící se svým tvarem obdélníku nebo kruhu. Rychlost snímání kamery 100 snímků za sekundu klade na celý algoritmus také požadavek rychlosti, protože je nutné každých 10ms vyhodnotit nový snímek.

3.4.1. Použité třídy

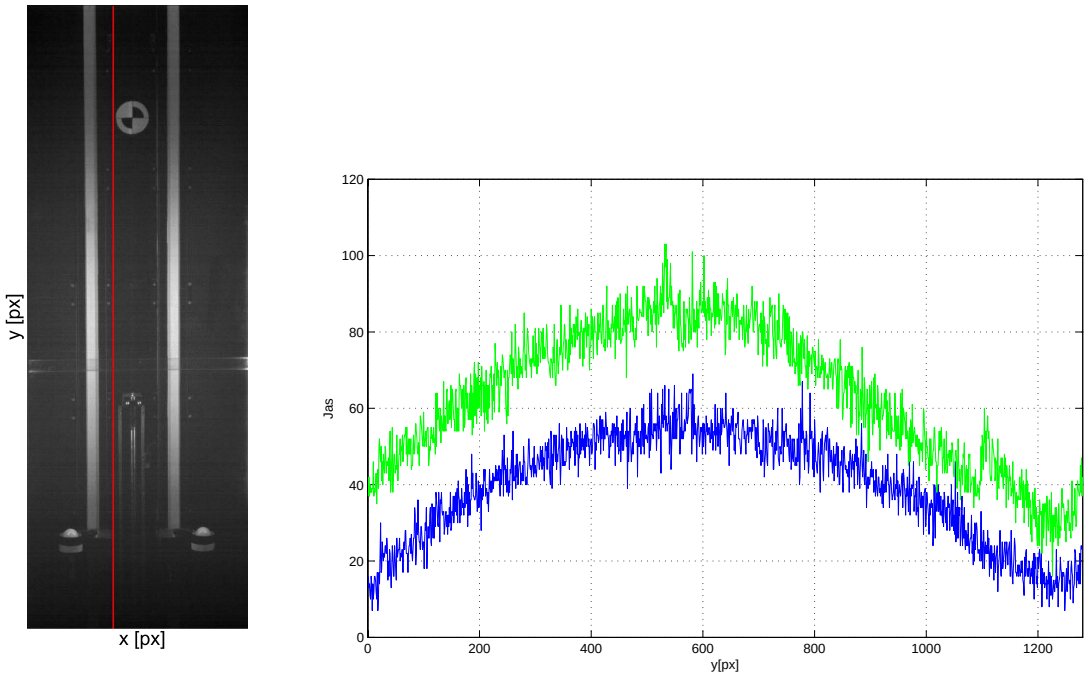
Knihovny HALCON obsahují sadu tříd, které zapouzdřují různé objekty vyskytující se v úlohách strojového vidění. Pomocí jejich metod lze dosáhnout požadovaných výsledků. V mé aplikaci byly použity tyto třídy:

- HImage – třída reprezentující obraz
- HFramegrabber – třída zprostředkovávající funkce grabovací karty a kamery
- HRegionArray - třída reprezentující pole regionů v obraze
- HTemplate – třída, která představuje předlohu při template matching
- HTuple – sada objektů libovolné třídy

3.4.2. Rozeznávání vyhazovacích misek

Pro úlohu identifikace vyhazovacích misek se ukázala jako nejpoužitelnější technika prahování snímku(blob analysis). Výsledkem aplikace funkce prahování jsou oblasti snímku, které mají hodnoty jasu v rozmezí daném parametry funkce. Tyto oblasti jsou v programu objekty typu HregionArray – pole regionů.

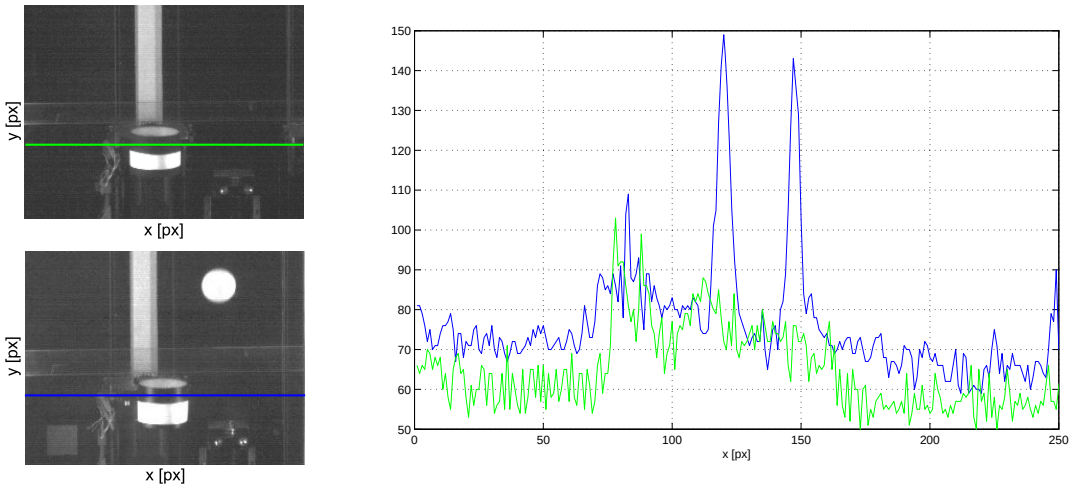
Spolehlivé použití této metody je značně závislé na stálosti parametrů pořizovaných snímků. V případě žongléra je stálost parametrů rušena denním světem, které má během dne různou intenzitu a tím pádem ovlivňuje jas pozadí, na kterém jsou objekty rozeznávány, ale i jas samotných rozeznávaných objektů. Abych mohl tuto metodu použít, otestoval jsem, jak se mění jas pozadí a objektů při krajních podmínkách – při maximálním a při minimálním rušení okolním světlem



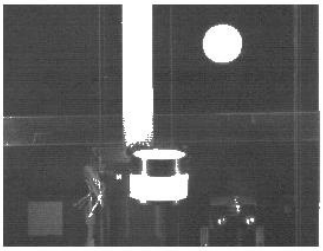
Obr. 3.8 Jasový profil pozadí při maximálním(zelená) a minimálním(modrá) světle z okolí

K maximální hodnotě jasu pozadí jsem navíc přičetl rezervu a prahovací úroveň stanovil na 130 (odstín šedi, 0 odpovídá černé, 255 bílé). Společně s vyhazovací miskou mají však jas vyšší než prahovací úroveň i bílé pásy, které se objeví také ve výsledných regionech. Proto jsou všechny regiony následně filtrovány a vybrány jsou jen ty, které mají velikost odpovídající vyhazovací misce, obdélníkový tvar a horizontální orientaci. Touto metodou lze spolehlivě rozeznávat v obraze vyhazovací misky v rozsahu jejich natočení 0° až 150° měřeno od středu žongléra. V krajních polohách je jejich rozeznání nejisté. Zdrojový kód je k nahlédnutí v příloze A.1.

Po nalezení vhodné prahovací hodnoty se však vyskytly problémy s odlesky od černých okrajů misek(viz Obr. 3.9 a 3.10), který byl stejný jako materiál pozadí-černá fólie. Ta má však i přes vysokou hodnotu absorbce světla vlastnost zrcadlící, tudíž byly v určitých místech světlá místa odpovídající odrazu světelných zdrojů. Tato světlá místa měla často hodnotu jasu vyšší než prahovací úroveň a výsledný obdélníkový region deformovala. Řešením bylo tedy najít takový materiál, jehož vlastnosti se budou blížit co nejvíce lambertovskému povrchu. V našem případě jsem jako povrch použil černou textilní pásku, která splňuje požadavky.

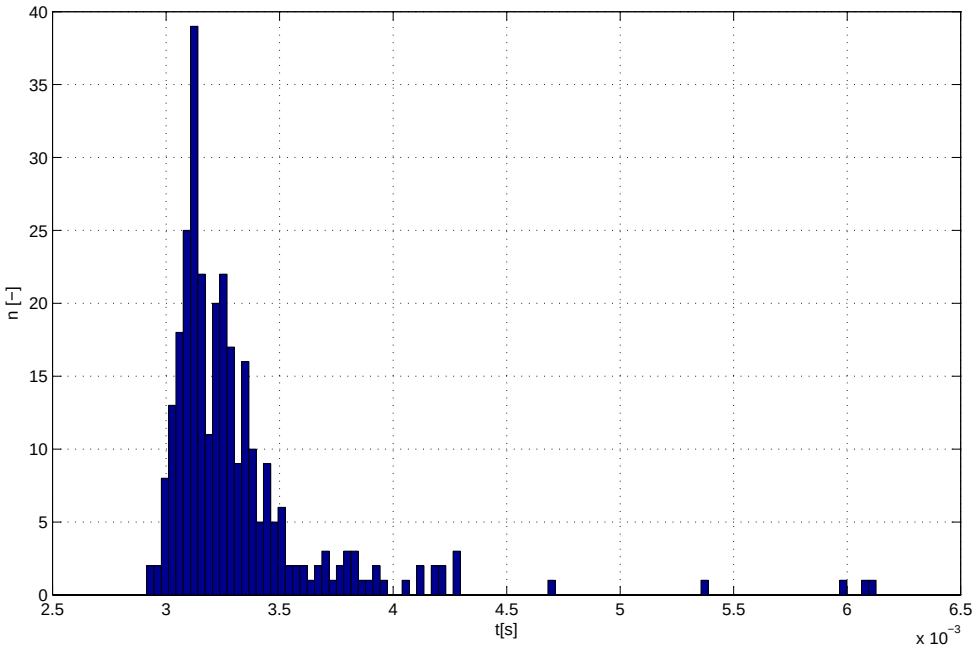


Obr. 3.9 Jasové hodnoty odlesku s původním(modrá) a lambertovským povrchem(zelená)

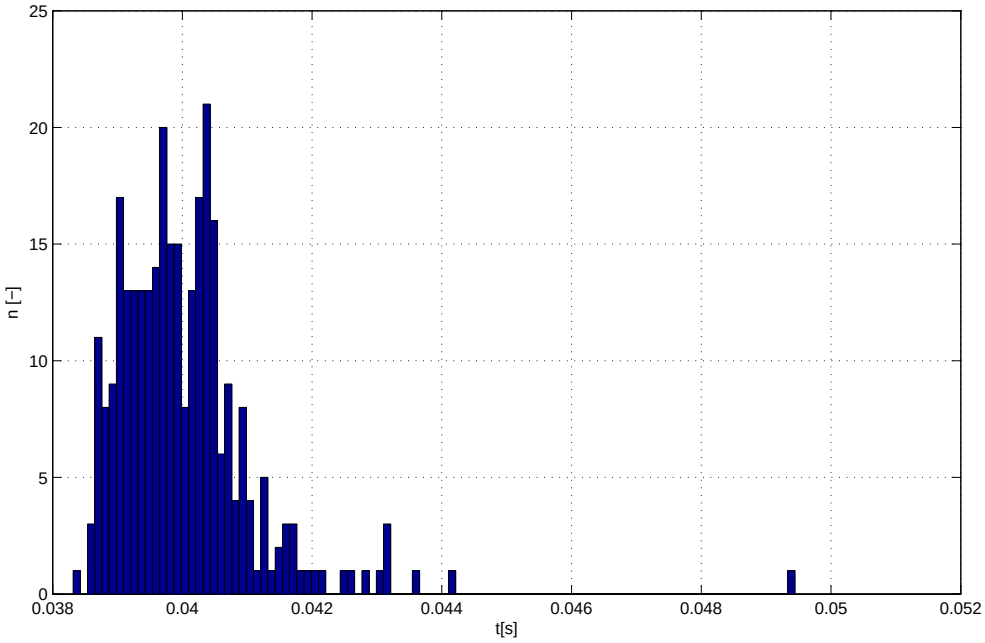


Obr. 3.10 Projevy směrových odlesků původního povrchu

Jako jiný přístup k řešení problému detekce vyhazovacích misek jsem vyzkoušel metodu template matching, kdy jsem po obvodu misky nalepil texturu podobnou čárovému kódu. Tím jsem zajistil opět stejnou podobu v obraze nezávislou na úhlu natočení. Algoritmus hledá v obraze místo, kde se jasové hodnoty co nejméně liší od jasových hodnot předlohy, která je hledána. Tímto přístupem se oproti prahování eliminují rušivé vlivy bílých pásů, avšak rapidně narůstá potřebný výpočetní čas, který s dostupným hardwarem nesplňuje potřebný limit 10ms. Navíc díky operačnímu systému, který nemá deterministický plánovač, je výpočetní čas náhodnou veličinou. Její rozložení jsem znázornil pomocí histogramů časů potřebných pro vykonání jednoho cyklu algoritmu. Průměrná délka zpracování obrazu u prahování je 3,3ms oproti 40ms u template matching.



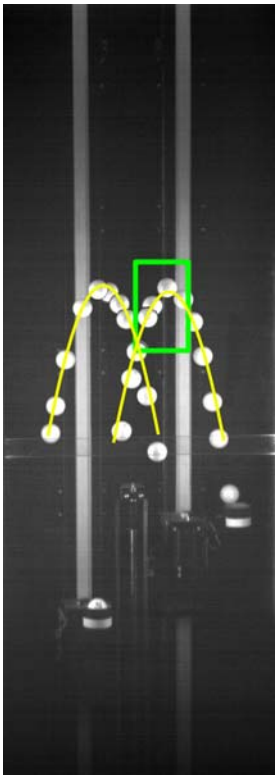
Obr. 3.11 Histogram časů pro algoritmus prahování



Obr. 3.12 Histogram časů pro algoritmus template matching

3.4.3. Rozeznávání letících koulí

Protože na rychlost rozeznávání letících koulí jsou kladeny stejné nároky jako na detekci vyhazovacích misek, použil jsem opět prahování s tím rozdílem, že výsledné regiony jsou filtrovány na základě podobnosti kruhu. Kontrola letící koule probíhá pouze v určité části obrazu aby bylo zajištěno, že místo požadované koule nebude rozpoznána jiná. Tato oblast byla definována na základě nasnímaných trajektorií přehazovaných koulí viz obrázek 3.13. Aplikace v tomto případě nahrazuje funkci optické závory. Zdrojový kód je k nalezní v příloze A.2.



Obr. 3.13 Oblast ROI pro kontrolu průletu koule

3.4.4. Detekce polohy referenčního znaku

Za účelem zjišťování pohybu kamery je potřeba určovat pozici referenčního znaku v obraze. Pokud se jeho obrazové souřadnice změní, znamená to, že se kamera pohnula. Nalezení znaku není časově omezeno rychlostí pořizování nových snímků jako u hledání misek nebo koulí, tudíž jsem použil template matching s hledáním vzoru na základě odstínů šedi. Konkrétně funkci `BestMatch`, jejíž výstupem je obrazová souřadnice nalezeného znaku.

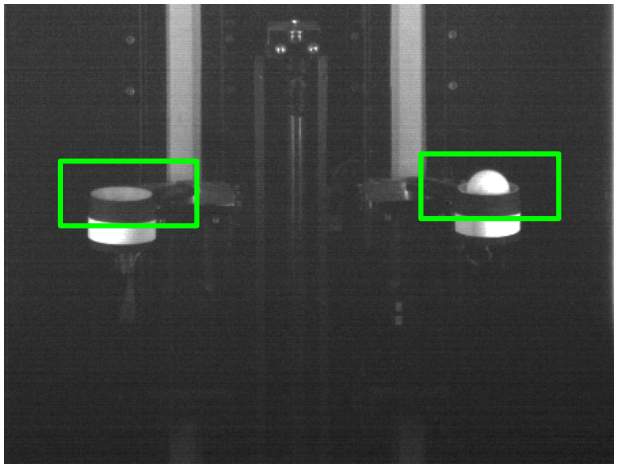
3.4.5. Detekce přítomnosti koule při inicializaci

Další funkcí aplikace je zjištění počátečního stavu systému. Jedná se o ověření, zdali jsou ve vyhazovacích miskách koule již přítomny nebo nikoliv. Tato kontrola probíhá vždy v pevně definovaných pozicích vyhazovacích ruček a není časově kritická jako kontrola koulí. Proto jsem zvolil opět template matching s pevně vymezenými oblastmi (ROI), kde se nacházejí misky ke kontrole. Stejně jako u detekce referenčního znaku využívám i zde funkci `BestMatch`, ale nadále pracuji s její výstupním parametrem `error`. Ten udává průměrný rozdíl jasových hodnot mezi předlohou a místem kde je určena jeho pozice v obraze.

Vzorec pro výpočet parametru lze nalézt v [1] a má tvar:

```
                    SUMu,v {ABS { Image[row-u,col-v] - Template[u,v] }}
error[row,col] =  -----------------------------------------------------
                                  AREA ( Template )
```

Pomocí průměrného rozdílu lze určit, zdali koule v misce je nebo není. Při přítomnosti koule se průměrný rozdíl pohybuje v závislosti na světelných podmínkách od 0 do 40. Algoritmus vyhodnocuje jako přítomnost koule hodnoty menší než 50. Zdrojový kód detekce přítomnosti koule je v příloze A.3.



Obr. 3.14 Oblasti pro vyhodnocování přítomnosti koule při inicializaci

Kapitola 4

4. Aplikace

4.1. Komunikace mezi aplikací a řídícím systémem – PVI

PVI (Process Visualization Interface) je rozhraní, které umožňuje přistupovat k síti B&R Automation Net™ z operačních systémů Windows™ 95/98/ME, NT/2000/XP. Každá stanice připojená k síti může přijímat či odesílat programové a procesní objekty nezávisle na fyzickém médiu, přes které je připojena, stejně jako na protokolu, který je pro komunikaci použit.PVI rozhraní se skládá ze čtyř základních částí:

- PVI Manager
- PVI Monitor
- PVICOM Interface
- PVI Lines

PVI Manager

PVI Manager je základní komponenta PVI. Může běžet na pozadí buď jako proces nebo jako služba. PVI Manager zprostředkovává veškerou komunikaci se zařízeními v B&R Automation Net.

PVI Monitor

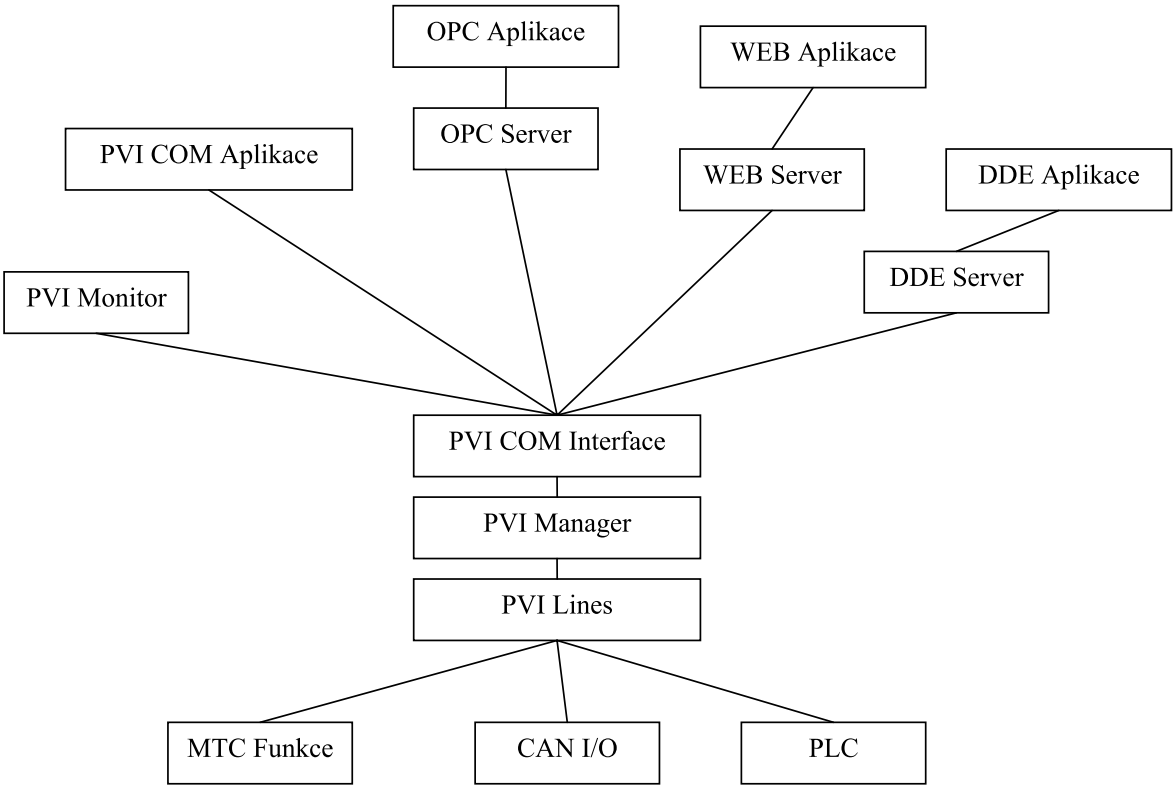
PVI Monitor je přídavný nástroj, pomocí něhož lze sledovat komunikaci v síti, případně zajistit logování komunikace.

PVICOM Interface

PVICOM Interface zajišťuje přístup do PVI na nejnižší vrstvě. Toto rozhraní je využíváno všemi aplikacemi ve Windows, které chtějí využívat PVI.

PVI Lines

PVI Line je odpovědná za komunikaci mezi kontroléry B&R a PVI Managerem a určuje protokol, který bude pro komunikaci použit.



Obr. 4.1 Struktura PVI

4.1.1. Komunikace mezi PC aplikací a PLC

Navázání komunikace mezi PC a PLC vyžaduje vytvoření spojení mezi danou aplikací a PVI Managerem a vytvoření objektů, pomocí kterých bude komunikace probíhat. Na začátku je potřeba inicializovat komunikační instanci. Ta zajišťuje spojení mezi aplikací a PVI Managerem. Vytváří se pomocí funkce `PviInitialize` z knihovny `PviCom`, která obsahuje veškeré funkce potřebné pro komunikaci přes PVI.

Komunikace mezi PVI Managerem a Aplikací probíhá pomocí knihovních funkcí a definovaných zpráv systému Windows (Windows Messages).

Po vytvoření komunikační instance funkcí `PviInitialize` je nutné nastavit tři druhy globálních událostí `POBJ_EVENT_PVI_CONNECT`, `POBJ_EVENT_PVI_ARRANGE` a `POBJ_EVENT_PVI_DISCONNECT` pomocí knihovní funkce `PviSetGlobEventMsg`.

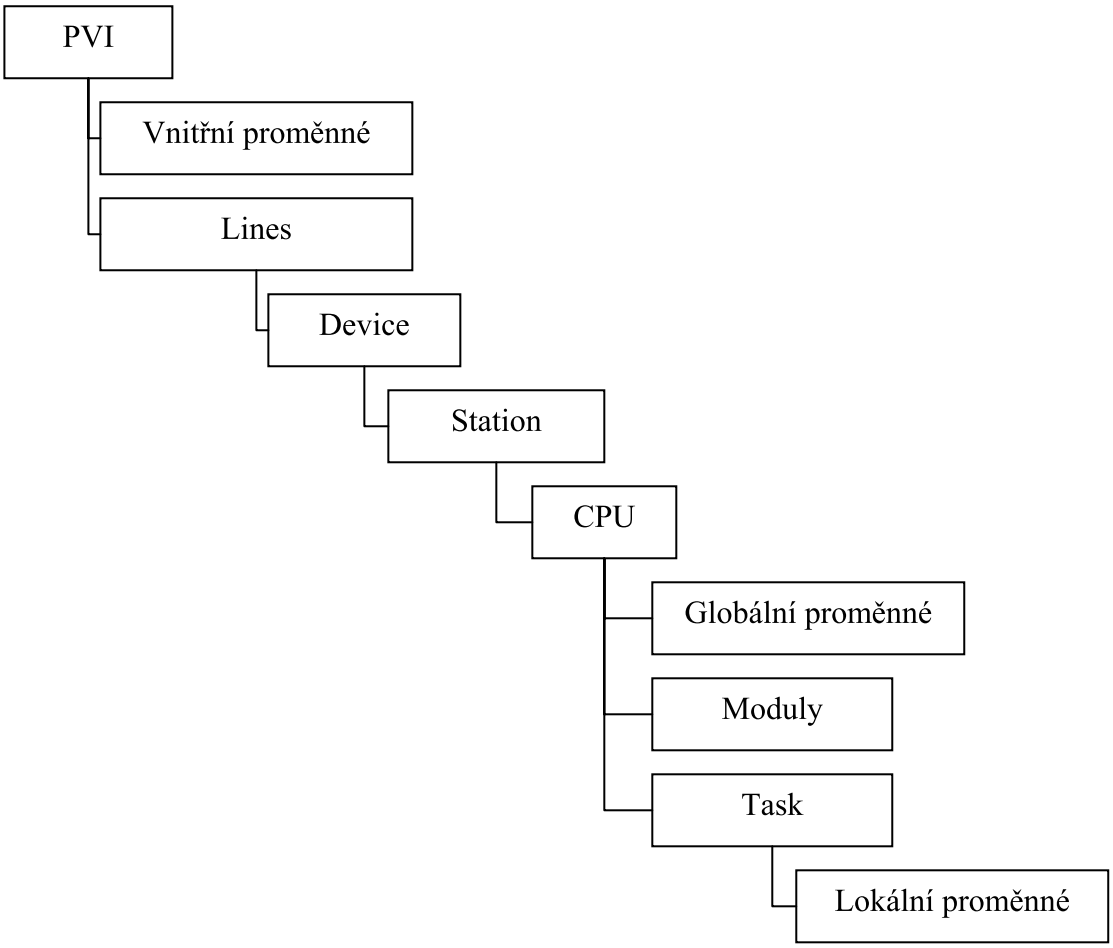
Význam globálních událostí je následující:

- `POBJ_EVENT_PVI_CONNECT` – Bylo navázáno spojení s Pvi Managerem
- `POBJ_EVENT_PVI_DISCONN` – Komunikace s PVI Managerem bylo přerušeno
- `POBJ_EVENT_PVI_ARRANGE` – Komunikační instance byla PVI Managerem zaregistrována a je potřeba vytvořit znovu dočasné objekty

Příklad inicializace je v příloze A.4

4.1.2. PVI Objekty

Po vytvoření spojení s PviManagerem a vyvolání globální události `POBJ_EVENT_PVI_ARRANGE` je nutné vytvořit v obsluze události objekty potřebné pro komunikaci s PLC. Objekty mají pevně danou hiearchii znázorněnou na následujícím schématu.



Obr. 4.2 Hiearchie PVI objektů

Line object

Line object definuje použitý protokol. Pro komunikaci s automaty B&R SG3 a SG4 se používá protokol INA2000.

Device object

Device object určuje, které zařízení bude použito pro komunikaci respektive po kterém fyzickém médiu bude komunikace probíhat. PVI umožňuje přenos po sériové lince, ethernetu, CAN, modemu, Profibusu, a LS251.

Station object

Station object představuje uzel v síti pro komunikaci. Pokud není potřeba, může být vynechán.

CPU object

CPU object reprezentuje PLC v síti. Pro komunikaci s daným CPU přes ethernet je nutné při vytváření objektu nastavit IP adresu a číslo PLC v síti.

Module object

Module object reprezentuje modul uvnitř PLC.

Task object

Task object reprezentuje task v PLC

Variable object

Variable object představuje proměnnou uvnitř PLC. Může být vytvořen ve stromové struktuře buď pod objektem typu CPU object, potom se jedná o globální proměnnou uvnitř PLC, nebo může být umístěn pod objektem typu Task object. V tomto případě se bude jednat o lokální proměnnou.

4.1.3. Vytváření objektů

Vytváření objektů se dělá pomocí knihovní funkce `PviCreate`. Při vytváření se specifikuje pozice objektu ve struktuře, která je znázorněna na obrázku 4.2. Dále se specifikuje příslušné číslo Windows message, která je generována při nějaké události objektu, a argument LPARAM příslušné události. Chování a vlastnosti objektu jsou určeny parametry Object Description a Link Description, jejichž formát a hodnoty jsou k nalezení v dokumentaci PVI.

Po vytvoření struktury objektů je možné komunikovat přes PVI již standardním způsobem.

4.1.4. Čtení proměnných z PLC

Při cyklické aktualizaci hodnot obrazu proměnné v PVI Manageru (nastavuje se při vytváření objektu) probíhá čtení pomocí událostí (Windows Messages). Událost je

vyvolána vždy, když dojde ke změně dat v obrazu proměnné nebo chybě. V aplikaci se tedy nemusí zajišťovat pravidelné čtení proměnné.

Čtení dat se tím pádem ošetřuje v handleru události, která byla definována při vytváření objektu dané proměnné. Událost obsahuje navíc parametr LPARAM, jenž byl také definován při vytváření objektu proměnné.

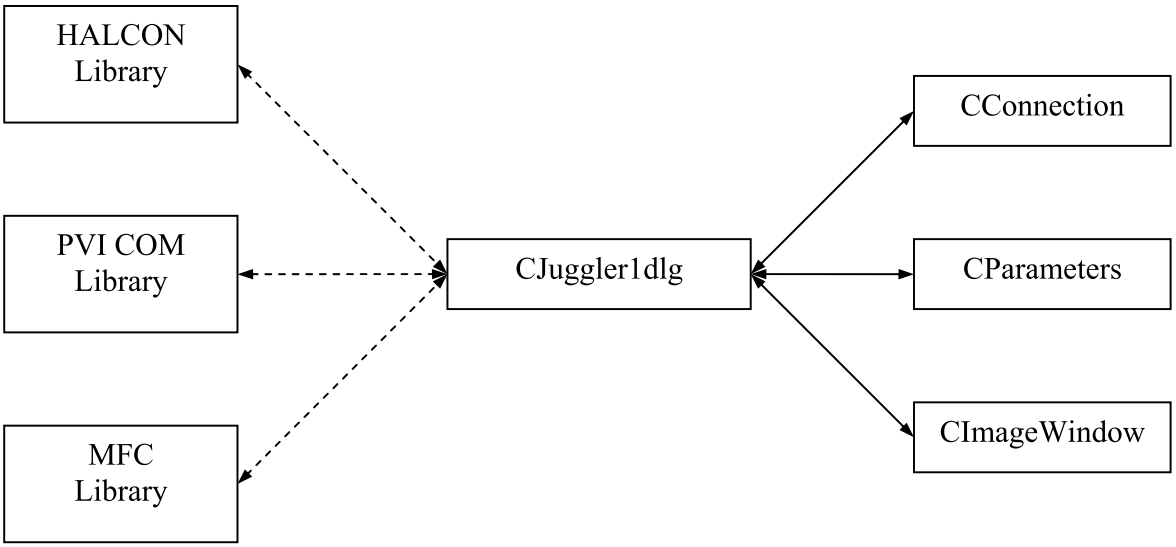
V mém případě se pro všechny objekty generuje stejná událost WM_PVI_EVENT, přičemž pro každý objekt je přiřazen unikátní parametr LPARAM, podle kterého lze rozlišit, který objekt událost vyvolal.

4.1.5. Zapisování proměnných do PLC

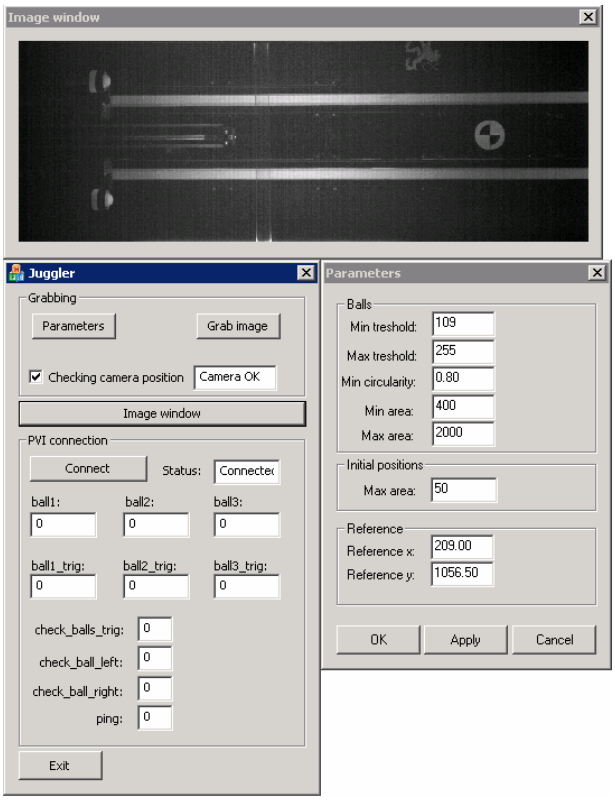
V případě zápisu je volána funkce `PviWriteRequest`, po níž následuje response od PVI Managera opět ve formě události. V handleru události poté funkce `PviWriteResponse` vrací hodnotu, která signalizuje číslo případné chyby (0 signalizuje úspěšný zápis).

4.2. Struktura aplikace

Jako vývojové prostředí jsem použil Microsoft Visual Studio 2005. K vytvoření grafického uživatelského rozhraní jsem využil MFC (Microsoft Foundation Class library), které zprostředkovávají objektově orientovaný přístup k programování v prostředí Windows a zapouzdřují Windows API. Díky MFC lze jednoduše vytvářet dialogová okna pro ovládání aplikace a pro interakci s uživatelem.



Obr. 4.3 Struktura aplikace



Obr. 4.4 Screenshot aplikace

CJuggler1dlg

Tato třída představuje základní dialogové okno aplikace pro inerakci s uživatelem. Kromě grafického uživatelského rozhraní obsahuje vlastní implementaci aplikace. Zajišťuje počáteční inicializaci a veškerou komunikaci přes PVI. Při startu aplikace inicializuje kontinuální pořizování snímků grabovací kartou a načtení vzorů pro template matching z uložených souborů. Zajišťuje také výstup aplikace do textového souboru, kam se zaznamenávají výsledky zpracování obrazu pro případné ladění aplikace a identifikaci chyb.

CConnection

Tento dialog slouží k zadávání parametrů pro spojení aplikace a PVI Manageru při inicializaci komunikace s řídícím PLC. Při ponechání volby Local connection je PVI Manager automaticky spuštěn jako proces. Pokud je PVI Manager spuštěn na jiném místě v síti, je možné se s ním spojit specifikováním IP adresy a portu, kde se PVI Manager v síti nachází.

CParameters

Dialog CParameters umožňuje měnit parametry algoritmu pro detekci průletu koule. Lze nastavit prahovací úrovně, parametr circularity, mezní hodnoty velikosti pro detekovanou kouli a obrazové souřadnice detekovaného referenčního obrazce.

CImageWindow

Tento dialog slouží pouze ke zobrazení snímků a výsledků zpracování obrazů. Při zobrazeném okně Image Window klade zobrazování výsledků vyšší nároky na výpočetní čas. Proto nemusí být stihnut zpracován každý snímek a některý může být vynechán.

4.3. Interakce aplikace a řídícího systému

Celý proces žonglování se odehrává ve velmi rychlém sledu akcí, které na sebe navazují v přesně definovaných časových okamžicích. To klade vysoké nároky nejen na synchronizaci všech servopohonů, ale i na synchronizaci s aplikací strojového vidění pro detekci správného chodu.

V řídícím programu PLC jsou za účelem synchronizace a komunikace vyhrazeny proměnné, k nimž aplikace přistupuje přes PVI rozhraní. Konkrétně se jedná o následující lokální proměnné umístěné v tasku komun.

```
ball1_trig
ball2_trig
ball3_trig
ball1
ball2
ball3
check_balls_trig
check_ball_left
check_ball_right
ping
```

Proměnné, které mají v názvu na konci `trig` jsou spouštěcí proměnné pro danou akci.

4.3.1. Zpětná vazba o přítomnosti koulí

`ball1_trig`, `ball2_trig` a `ball3_trig` spouští svou náběžnou hranou v aplikaci samostatné vlákno vyhodnocující obraz z kamery. Připomenu, že hodnota proměnných je PVI Managerem aktualizovaná každých 10ms. Tyto jednotlivé proměnné jsou nastavovány do „1“ vždy společně s aktivací vačkového profilu pro vyhození příslušné koule. Od tohoto okamžiku do průletu koule místem, kde je obraz vyhodnocován, uplyne přibližně 0.4

sekundy. Dobu běhu vlákna jsem tudíž nastavil na 0.6 sekundy. Za tuto dobu koule prolétne celou vyhodnocovací oblastí.

Pokud je koule v obraze nalezena, je vlákno vyhodnocující obraz ukončeno a zpět řídícímu programu není poskytována zpětná vazba o průletu koule. Důvod tohoto řešení spočívá v možnosti provozovat žonglování i bez zpětné vazby z aplikace, kdy se předpokládá, že žádná koule nespadla a tudíž není zařazován cyklus na nahození koule zpět. V případě, že koule není v obraze rozpoznána do 0.6 sekundy od spuštění vlákna, je aplikací nastavena příslušná proměnná v PLC (`ball1, ball2, ball3`) do logické „1“. Od této proměnné se pak v řídícím programu odvíjí zařazení sekvence vačkových profilů pro nahození nové koule. Po navrácení koule do cyklu je proměnná pochopitelně zpět řídícím programem nastavena do logické „0“.

Sekvenční diagram komunikace je znázorněn na obrázku 4.5.

4.3.2. Zpětná vazba o počátečním stavu stroje

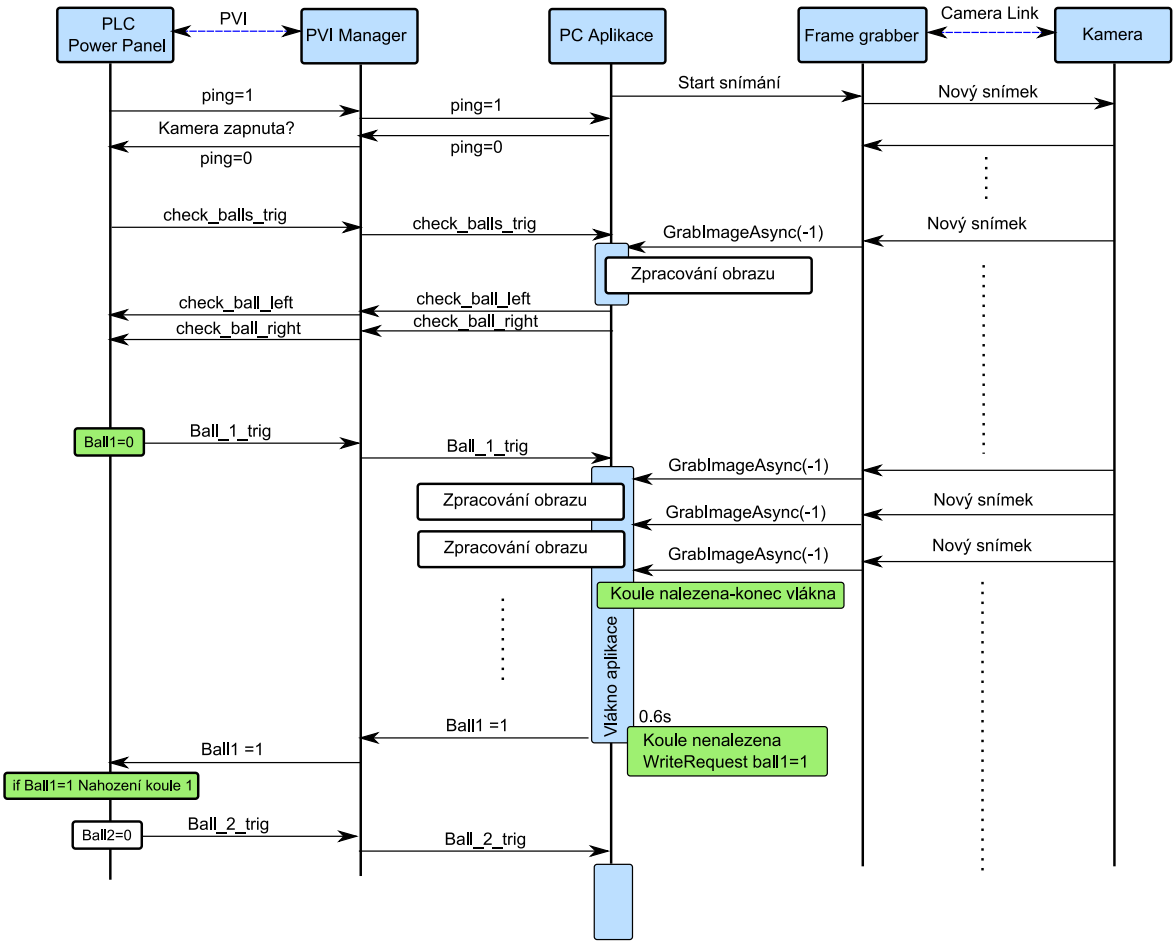
Ke spouštění algoritmu pro kontrolu počátečního stavu je použita proměnná `check_balls_trig`, respektive její náběžná hrana. V řídícím programu je tato proměnná nastavena do „1“ po dosažení pozic, odpovídajícím oblastem, kde se přítomnost koulí vyhodnocuje (Obr. 3.14). Současně jsou do „1“ nastaveny i proměnné `check_ball_left` a `check_ball_right` signalizující nepřítomnost koulí. To opět z důvodu možnosti provozu systému i bez zpětné vazby aplikace. Při vyhodnocení přítomnosti koule je zapsána do příslušné proměnné „0“ a cyklus nahození dané koule bude vynechán.

4.3.3. Test provozu aplikace

Poslední proměnná `ping` slouží pouze pro testování zdali aplikace pro zpracování obrazu je zapnuta, aby mohl být stav signalizován na řídícím panelu. Na náběžnou hranu proměnné odpovídá aplikace zápisem „0“.

4.3.4. Detekce manipulace s kamerou

Algoritmus rozpoznávání manipulace s kamerou byl popsán v 3.4.4. Pokud je detekován pohyb s kamerou, přestává aplikace posílat informace o přítomnosti koulí a na test provozu aplikace (`ping`) nereaguje. Na řídícím panelu se pak kamera zobrazí jako vypnutá.



Obr. 4.5 Sekvenční diagram komunikace mezi PLC a aplikací

4.3.5. Spolehlivost komunikace

Jak už bylo v předchozích odstavcích zmíněno, pro určité operace existují časové limity, které musejí být pro správnou funkci aplikace dodrženy. Konkrétně se jedná o následující časové intervaly:

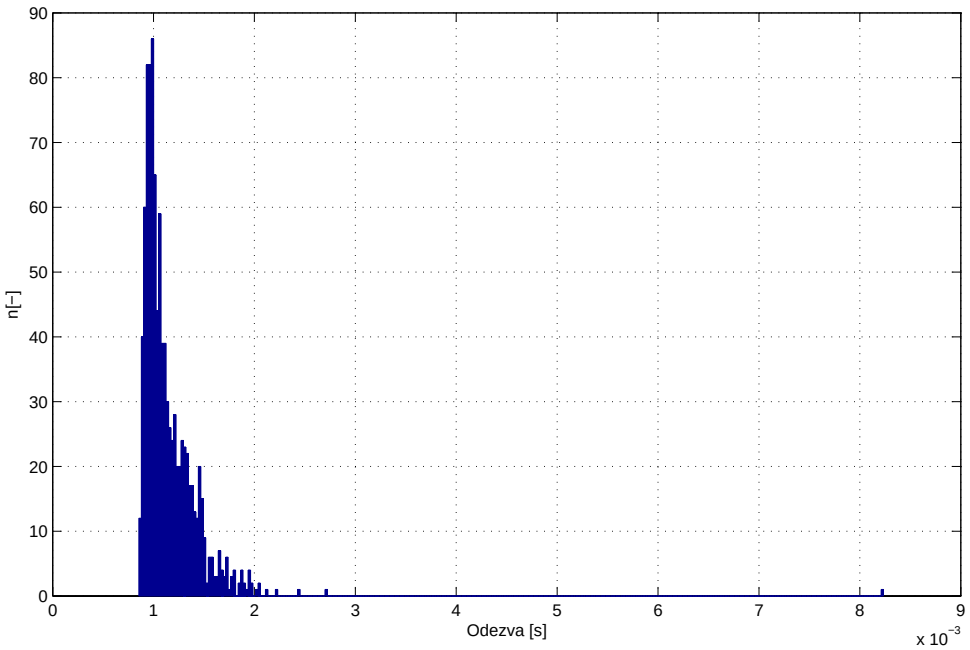
Časový bod 1	Časový bod 2	Maximální interval
Nastavení `ballx_trig` do "1"	Start vlákna rozpoznávání koulí	t1 = 0,4s
Konec vlákna rozpoznávání koulí	Vyhodnocení přítmnosti v PLC (kontrola proměnné `ballx`)	t2 = 0,2s*
Nastavení `check_balls_trig` do "1"	Start algoritmu testujícího přítomnost koulí v miskách	t3 = 2s
Vyhodnocení počátečních pozic algoritmem aplikace	Vyhodnocení počátečních pozic v PLC (kontrola `check_ball_left/right`)	t4 = 3s

Tab. 4.1 Mezní hodnoty pro zpoždění komunikace

** Tento interval je zmenšován reálnou hodnotou t1*

Spolehlivost systému jako celku může být narušena použitím komponent, jejichž řízení není deterministicky časově určeno. V mém případě se jedná o spojení počítače a PLC přes síť Ethernet s použitím PVI/UDP protokolu a použití operačního systému Windows XP SP2. V případě použití PVI/UDP protokolu je případná ztráta datagramu řešena vyšší vrstvou, která očekává potvrzení přijetí datagramu druhou stranou. Pokud není potvrzení přijato v časovém limitu je datagram odeslán znovu. Časový limit, po kterém je datagram znovu odesílán, se specifikuje při vytváření CPU objektu v PVI struktuře. Konkrétně se jedná o parametr RT (Response Time) v Link Description. Pokud není potvrzení o přijetí obdrženo třikrát po sobě v daném intervalu, je spojení považováno za ztracené. Na straně aplikace se to projeví vyvoláním události `WM_PVI_EVENT`, která v obsluze události po volání funkce `PviReadResponse` vrátí číslo chyby. Doba, za kterou je chyba spojení detekována, je rovna 4*Response Time.

Pevně definované časové rozmezí, ve kterém bude zpráva doručena není možné s jistotou určit. Rychlost zápisu/čtení proměnných je tedy do jisté míry náhodnou veličinou. Abych získal přehled s jakou rychlostí je prováděn zápis z aplikace do PLC, změřil jsem čas od volání funkce `PviWriteRequest` (zápis proměnné) do vyvolání události, která signalizuje, úspěšný zápis. Průměrná hodnota tohoto času byla 1,1ms a maximum 8,2ms. Histogram z naměřených hodnot je na obrázku 4.6.



Obr. 4.6 Histogram časů potřebných pro zápis proměnné do PLC

Pro zápis proměnné do PLC jsou tedy potřebné časy na velmi nízké úrovni. Současně je třeba si uvědomit, že naměřené časy jsou ovlivněny přidělováním procesorového času vyhodnocujícímu procesu.

Čtení proměnných je popsáno v 4.1.4. Pro otestování času potřebného k přečtení proměnné jsem zapsal do proměnné v PLC „1“, přičemž v PLC je proměnná při prvním cyklu vrácena do „0“. Měřena byla doba od vyslání požadavku na zápis „1“ do přijetí „0“. Po přijetí „0“ byl znovu vysílán požadavek na zápis „1“. Tento cyklus měření byl opakován tisíckrát. V tomto případě se hodnota času rovnala až na několik málo výjimek hodnotě blízké 62ms. Při vložení funkce `Sleep(10)` (uspání vlákna na 10ms) mezi přijetí „0“ a vyslání požadavku na „1“ se hodnota času snížila na 47ms. Při prodlužování doby před dalším požadavkem na „1“ byly časy následující: (Tab. 4.2)

`Sleep(x)`	
x	Změřený čas
10	47
20	32
30	32
40	16
50	63
60	63
70	47
80	32

Tab. 4.2 Změřené odezvy pro různé časové odstupy mezi měřeními

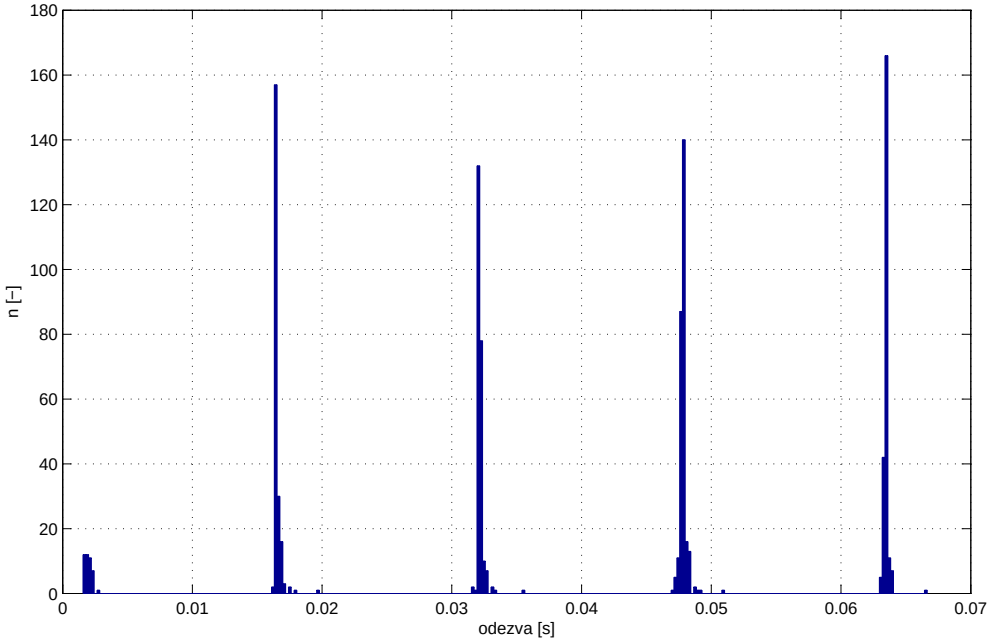
Poté jsem otestoval jak se ve skutečnosti chová funkce `Sleep(x)`která uspí vlákno na x milisekund. Reálné hodnoty uspání vlákna jsou v Tab. 4.3. Tyto časy jsou stabilní pouze když je funkce volána po sobě například v cyklu for.

`Sleep(x)`	
x	Změřený čas
10	15
20	31
30	31
40	46
50	62
60	62
70	78

Tab. 4.3 Reálné časy uspání vlákna funkcí Sleep

Další měření jsem provedl tak, že jsem před každý požadavek vložil náhodně generovanou časovou prodlevu v rozsahu 0-200ms. Reálné prodlevy měly však hodnoty odpovídající těm v Tab. 4.3. Výsledné změřené časy jsem pak zobrazil pomocí histogramu na Obr. 4.7.

Z výsledků měření se dá odhadnout, že doba cyklu, se kterým je hodnota proměnné z PLC vyčítána, je blízká 60ms. Doba čtení proměnné je tedy závislá na stavu cyklu PLC a také na plánovači operačního systému.



Obr. 4.7 Histogram časů pořebných pro zápis + čtení proměnné z PLC

Závěrem bych doplnil, že kritický čas 0,2 sekundy z tabulky 4.1 nebyl při měření nikdy dosažen. V mé aplikaci jsem navíc nastavil parametr RT na hodnotu 66 milisekund, aby se daný rámec v nejdelším možném intervalu 0,2s stihl při případné chybě třikrát odeslat. Jako spolehlivé řešení by bylo použití sítě s deterministickým protokolem současně s operačním systémem reálného času.

Kapitola 5

5. Laboratorní úloha

Vytvoření optické závory pomocí průmyslové kamery

V návaznosti na zamýšlené budoucí použití žongléra pro vzdálenou výuku řízení polohových aplikací jsem se snažil vytvořit úlohu, jejíž realizací by se dosáhlo variabilní kontroly správného chodu stroje při přehazování kulečníkových koulí. Cílem úlohy je tedy vytvořit prostředky, pomocí kterých bude možné detekci realizovat vzdáleně bez nutnosti přítomnosti studenta v laboratoři. Ke splnění tohoto cíle bude využita průmyslová kamera, software pro zpracování obrazu HALCON, vývojové prostředí HDevelop a aplikace, které jsem vytvořil, aby bylo možné úlohu realizovat v kratším časovém rozpětí. Tato úloha je studentem realizována poté, co úspěšně implementoval předchozí úlohy a má smysl vytvářet kontrolu pomocí průmyslové kamery.

Úloha se skládá z dvou základních bodů. Prvním bodem je vytvoření programu v prostředí HDevelop, který bude rozpoznávat letící koule, a následná konfigurace předpřipravené aplikace, která daný algoritmus bude vykonávat a rozšiřuje ho o komunikaci s řídícím PLC. Druhým bodem je kontrola přítomnosti koule ve vyhazovací misce. Tento úkol zahrnuje vytvoření předlohy vyhazovací misky obsahující kouli, která bude použita v připravené aplikaci pro vyhodnocení, zdali koule v misce je, nebo nikoliv. Součástí druhého úkolu je také samozřejmě konfigurace parametrů v aplikaci.

Tato úloha může být použita například v předmětu X35PRA Programovatelné automaty. Po úspěšném dokončení úlohy by měl student získat základní zkušenost s vývojem aplikace pro strojové vidění pomocí vývojového nástroje HDevelop. Také si prakticky vyzkouší aplikaci jednotlivých technik rozpoznávání obrazu na konkrétní úloze.

Podrobný postup s dplňujícími informacemi je v příloze B.

Kapitola 6

6. Závěr

V závěru této práce bych chtěl po jednotlivých částech rekapitulovat výsledky, kterých jsem při práci na žonglérovi dosáhl. Také bych zde rád zmínil další práci, která přímo navazovala na výsledky popsané v tomto dokumentu, jejíž realizace byla nad rámec zadání mé bakalářské práce.

První kapitola osahuje základní informace o žongléru a definování cílů, kterých jsem chtěl dosáhnout. V druhé kapitole jsem se zabýval problematikou osvětlení v aplikacích strojového vidění a popsal, jakým způsobem se aplikace zapojuje do automatizačního procesu(Obr. 2.1). Analyzoval jsem použitelné zdroje osvětlení z obecného hlediska strojového vnímání (Tab. 2.2) a poté z hlediska vhodnosti pro žonglér (Tab. 2.3). Na základě tohoto rozboru jsem zvolil výsledné osvětlení pomocí lineárních zářivek umístěných svisle na každé straně žongléru(Obr. 2.6). Tímto způsobem jsem docílil osvětlení, které splňuje kladené požadavky vycházející ze způsobu zpracování obrazu. Tyto požadavky jsou splněny v pracovním prostoru vymezeném délkou zářivek. Na osvětlení celého prostoru žongléra by bylo nutné pořídit další světelné zdroje. Tato konfigurace však pro realizaci mé úlohy a pro provedení analýzy zabezpečení žongléru byla dostatečná.

Třetí kapitolu jsem věnoval rozboru metod a algoritmů používaných v úlohách strojového vidění. Z nich jsem poté vybral vhodné metody k řešení mých cílů. Pro zpracování obrazu, které je náročné na dobu zpracování obrazu(každý snímek musí být stihnut zpracován) jsem použil prahování s následnou klasifikací výsledných regionů. Pro jednotlivé úlohy nenáročné na rychlost zpracování jsem použil metodu template matching.

Ve čtvrté kapitole jsem vytvořil stručný popis rozhraní PVI pro komunikaci mezi aplikací a řídícím PLC. Také jsem popsal výslednou naprogramovanou aplikaci z hlediska logické funkčnosti a komunikace s PLC. V poslední části této kapitoly jsem popsal měření, která jsem provedl, abych získal představu o spolehlivosti aplikace. To z důvodu použití nedetrministického komunikačního protokolu a operačního systému. Z výsledků měření i praktického zkoušení mohu potvrdit dobrou spolehlivost aplikace. Vzhledem k povaze úlohy není případné selhání pro systém kritické.

Poslední kapitola popisuje úlohu pro studenty předmětu X35PRA, kterou jsem připravil za účelem využití průmyslové kamery v řídícím systému.

Kromě splnění cílů, které jsem měl v zadání bakalářské práce, jsem se věnoval i spolupráci na analýze zabezpečení žongléru proti kolizím mechanických částí pomocí průmyslové kamery. Za tímto účelem bylo nutné rozeznávat v obraze vyhazovací misky, což jsem popsal v části 3.4.2. Pro analýzu jsem vytvořil program, jehož výstupem byly souřadnice vyhazovacích misek získané pomocí transformačních údajů z kalibrace kamery. Také jsem pomocí tohoto programu vytvořil sadu snímků obsahující známé pozice misek. Na základě těchto dat mohla být následně provedena hlubší rozvaha nad zabezpečením pomocí kamery.

Při realizaci cílů této práce jsem získal mnoho zkušeností z oblasti strojového vidění, ale i z oblasti vývoje C++ aplikací v prostředí Visual Studio. Také jsem se naučil používat PVI rozhraní pro komunikaci s řídícím PLC. V neposlední řadě bych také zmínil zkušenost se spoluprací na projektu s více studenty při dosahování společně vytyčených cílů.

Seznam použité literatury

[1] MVTec HALCON 7.1, *Application guide*, MVTec Software GmbH, München, Germany, 2005.

[2] NĚMEC TOMÁŠ, *Diplomová práce: Positional visual feedback control*, Katedra řídící techniky ČVUT, Praha 2007

[3] PŠENIČKA JAN, *Diplomová práce: Synchronizace servopohonů*, Katedra řídící techniky ČVUT, Praha 2007

[4] HORNBERG ALEXANDER, *Handbook of machine visio*n, Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim 2006, ISBN – 978-3-527-40584-8

[5] *Philips produktový katalog*, 2008/2009, Philips GmbH, Hamburk

[6] HORTON IVOR, *Beginning Visual C++*, Wiley Publishing Inc, Indianapolis 2008, ISBN: 978-0-470-22590-5

[7] Mikrotron MC13xx manual, Mikrotron GmbH 2006

[8] *Wikipedia* [online], http://en.wikipedia.org/

[9] JAROŠ PAVEL, *Bakalářská práce: Dynamika rychlých servopohonů,* Katedra řídící techniky ČVUT, Praha 2009

Přílohy

A. Ukázky zdrojových kódů

A.1 Detekce vyhazovacích misek

```
finalRegions.Reset();
try{
      image=FG.GrabImageAsync(-1);
}catch(HException &exception){
      FG.GrabImageStart(-1);
}


pWindow->SetColor("green");
pWindow->ClearWindow();
pWindow->Display(image);

regionArray=image.FastThreshold(m_iMinTreshold,m_iMaxtreshold,20);
connectedRegions=regionArray.Connection();
selectedRegions=connectedRegions.SelectShape(
      HTuple("area").Concat("rectangularity")
      ,"and"
      ,HTuple(m_iMinArea).Concat(m_DminRect)
      ,HTuple(m_iMaxArea).Concat(1));

double orientation=0;
for(int i=0;i<selectedRegions.Num();i++){
      orientation = selectedRegions[i].OrientationRegion();
      if(
            (orientation>1.4)&&(orientation<1.7) ||
            (orientation>-1.7})&&(orientation<-1.4)
         )
                  finalRegions.Append(selectedRegions[i]);
}

pWindow->Display(connectedRegions);
pWindow->SetColor("red");
pWindow->Display(finalRegions);

finalRegions.AreaCenter(&ix,&iy);
```

A.2 Detekce letících koulí

```
rect=rect.GenRectangle1(220,710,300,900);
...
try{
      image=jugglerDlg->FG.GrabImageAsync(-1);
}catch(HException &exception){
      jugglerDlg->FG.GrabImageStart(-1);
}

reducedImage=image.ReduceDomain(rect);
regionArray=reducedImage.FastThreshold(minTreshold,maxTreshold,20);
connectedRegions=regionArray.Connection();
```

```
selectedRegions=connectedRegions.SelectShape(
      HTuple("area").Concat("circularity")
      ,"and"
      ,HTuple(minArea).Concat(minCirc)
      ,HTuple(maxArea).Concat(1));
```

A.3 Detekce přítomnosti koule ve vyhazovací misce

```
int ErrCode;
BYTE left=0;
BYTE right=0;
...
try{
      image=FG.GrabImageAsync(-1);
}catch(HException &exception){
      FG.GrabImageStart(-1);
}

pWindow->ClearWindow();
pWindow->Display(image);

rectLeft=rectLeft.GenRectangle1(50,345,120,375);
rectRight=rectRight.GenRectangle1(320,345,390,375);

reducedImage=image.ReduceDomain(rectLeft);
dix=reducedImage.BestMatch(templateLeft,255,"false",&diy,&Error);
pWindow->SetColor("red");
ix=(HTuple)dix;
iy=(HTuple)diy;
pWindow->DispCircle(ix,iy,10);
if(Error<50.0){
      ErrCode = PviWriteRequest (hCheckLeft
                                 , POBJ_ACC_DATA
                                 ,&left
                                 , 1
                                 ,CWnd::m_hWnd
                                 , NULL
                                 , IDC_CHECK_LEFT);

if (ErrCode != NULL){
   ErrorMessage ("Error on write request checkLeft (%u)", ErrCode);
                    }
}
reducedImage=image.ReduceDomain(rectRight);
dix=reducedImage.BestMatch(templateRight,255,"false",&diy,&Error);
ix=(HTuple)dix;
iy=(HTuple)diy;
pWindow->DispCircle(ix,iy,10);
if(Error<50.0){ //right is correct
      ErrCode = PviWriteRequest (hCheckRight
                                          , POBJ_ACC_DATA
                                          ,&right
                                          , 1
                                          ,CWnd::m_hWnd
                                          , NULL
                                          , IDC_CHECK_RIGHT);

if (ErrCode != NULL){
  ErrorMessage ("Error on write request checkRight (%u)", ErrCode);
                      }
```

A.4 Inicializace PVI

```
int         ErrCode;
DWORD Timeout;
CString     InitParam;

if(connection->m_bLocal)InitParam="";
else{
      InitParam.Format ("%s=%s %s=%s",
            KWINIT_IPADDR, connection->m_strIPManager,
            KWINIT_IPPORT, connection->m_strPortManager);
}

ErrCode = PviInitialize (3, INFINITE, InitParam, NULL);
if (ErrCode != NULL)
{
      ErrorMessage ("Can't initialize PVI (%u)", ErrCode);
      return;
}

PviSetGlobEventMsg (POBJ_EVENT_PVI_CONNECT,
                  CWnd::m_hWnd, WM_PVI_GLOBAL_EVENT,
POBJ_EVENT_PVI_CONNECT);
      PviSetGlobEventMsg (POBJ_EVENT_PVI_DISCONN,
                  CWnd::m_hWnd, WM_PVI_GLOBAL_EVENT,
POBJ_EVENT_PVI_DISCONN);
      PviSetGlobEventMsg (POBJ_EVENT_PVI_ARRANGE,
                  CWnd::m_hWnd, WM_PVI_GLOBAL_EVENT,
POBJ_EVENT_PVI_ARRANGE);
```

Úloha pro studenty

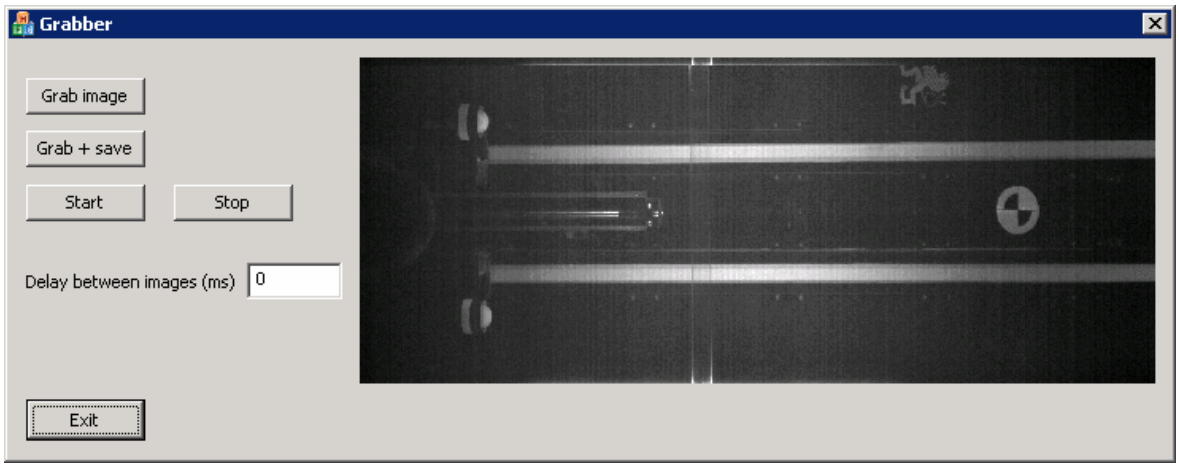
B. Realizace optické závory průmyslovou kamerou

Cílem úlohy je realizovat detekci správného chodu stroje, což představuje kontrolu přítomnosti koule ve vyhazovací misce na začátku cyklu a kontrolu přítomnosti po vyhození koule. K detekci koule po vyhození bude použita technika prahování a k rozpoznání přítomnosti koule v misce metoda template matching. Pro obě úlohy bude sestaven program ve vývojovém prostředí HDevelop. Ze sestaveného programu budou poté použity parametry pro zpracování v připravené aplikaci, která komunikuje s řídícím PLC.

B.1 Realizace optické závory

B.1.1 Nasnímání sekvence snímků pro testování algoritmu

Prvním krokem je vytvoření sekvence snímků, které využijeme pro testování rozpoznávání letících koulí. K tomuto účelu je připravena aplikace Grabber, která se nachází ve složce úloha. Aplikace obsahuje jedno dialogové okno s ovládacími prvky. Viz obrázek B.1. Letící koule budou detekovány jen v určité části letu, nasnímejte proto sekvenci snímků obsahující průlet koule právě inkriminovaným místem.



Obr. B.1 Aplikace grabber

Význam ovládacích prvků:

Grab image – Pořízení jednotlivého snímku a zobrazení v okně

Grab+save – Pořízení jednotlivého snímku a jeho uložení. Snímky se ukládají do složky images ve složce aplikace ve formátu `pic_rrrrmmdd_hhmmss.tif`

Start – Spuštění snímání sekvence snímků. Snímky jsou ukládány do složky images a jsou číslovány. Současně je ve složce vytvořen soubor sekvence.seq, který obsahuje názvy a cestu ke všem snímkům.Pozor, při každém spuštění nové sekvence jsou staré snímky ponechané ve složce přepsány novými!

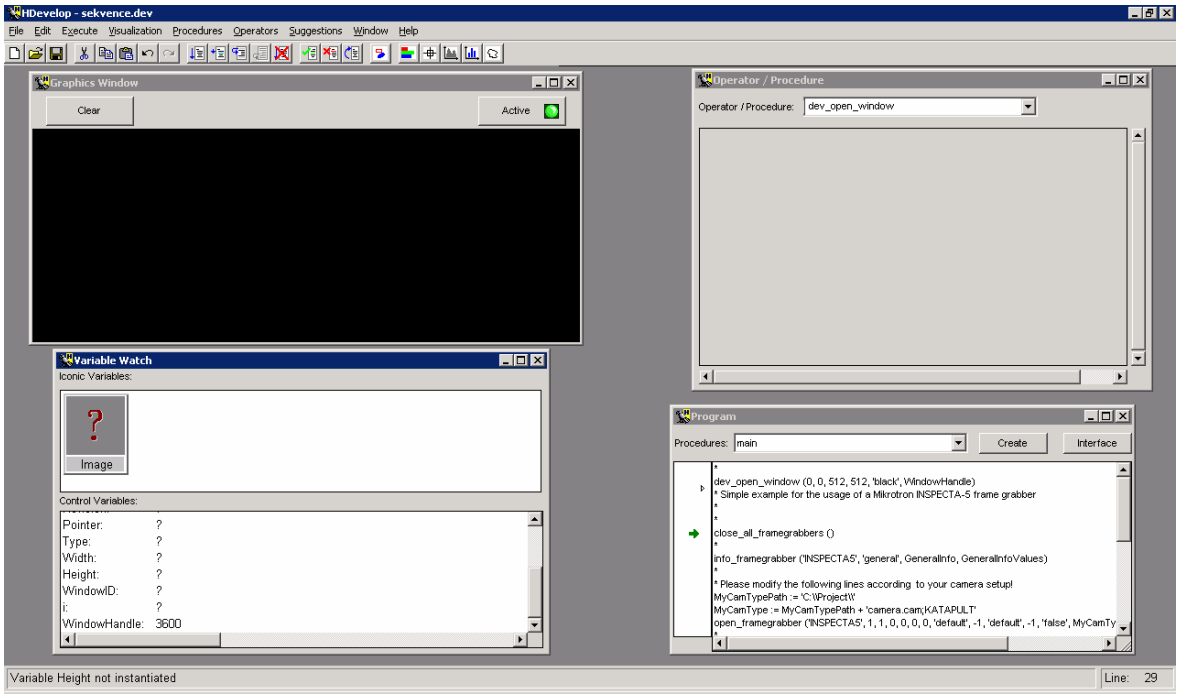
Stop – zastavení snímání sekvence snímků.

Delay between images – zpoždění mezi pořizovanými snímky. Při hodnotě 0 je snímkováno maximální možnou frekvencí 100fps. Tato frekvence se po čase zmenší z důvodu zahlcení vyrovnávací paměti. Při hodnotě vyšší než 0 je po každém snímku snímání zastaveno na příslušný počet milisekund a k zahlcení vyrovnávací paměti nedojde tak rychle.

B.1.2 Vytvoření algoritmu rozpoznávající letící koule

HDevelop

HDevelop je vývojový software pro vytváření algoritmů pro zpracování obrazu. Obsahuje veliké množství funkcí, pomocí kterých je možné zpracovávat obraz nejrůznějšími technikami používanými ve strojovém vidění. Screenshot okna HDevelopu je zobrazen na obrázku B.2. Aplikace obsahuje čtyři základní okna.



Obr. B.2 Aplikace HDevelop

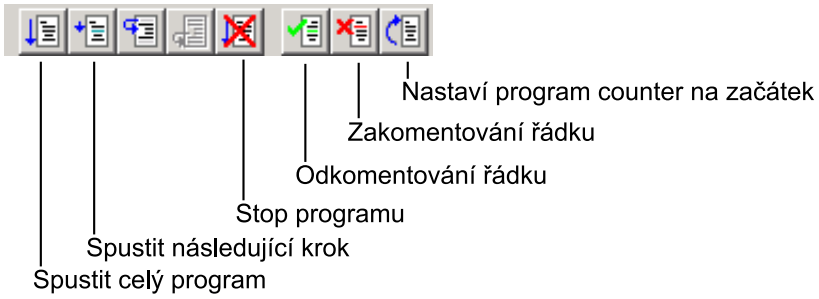
Graphics Window – Okno sloužící pro zobrazení obrazu a vizualizaci výsledků aplikace funkcí.

Variable Watch – Toto okno je rozděleno na dvě části a slouží ke správě proměnných, které se v sestavovaném programu nacházejí. V horní části jsou zobrazeny náhledy proměnných, které je možné zobrazit v Graphics Window (obrázky, regiony atd…). V dolní části je zobrazen přehled proměnných, které nelze vizualizovat (řetězce, čísla atd…). U proměnných je zobrazena také jejich aktuální hodnota.

Program – V oknu Program je zobrazen aktuální program jak je vytvořen pomocí sekvence funkcí, které HDevelop nabízí. Černá šipka na levé straně označuje místo v programu kam bude vložena vybraná funkce. Pozici šipky lze měnit poklepáním pravým tlačítkem myši na místo kam šipku chceme umístit a vybráním možnosti Set Insert Cursor. Zelená šipka označuje následující funkci, která bude při vykonávání programu spuštěna (program counter).

Operator/Procedure - Toto okno slouží k výběru funkce, kterou chceme zařadit na příslušné místo do programu a nastavení jejích parametrů. Funkci lze vybrat napsáním názvu do řádku Operator/Procedure (stačí napsat část názvu funkce a vybrat z možností, které se objeví po rozbalení možností). Druhým způsobem je vybrat funkci z menu Operators nebo Procedures v horním panelu. Pro každou vybranou funkci je možné zobrazit dokumentaci kliknutím na tlačítko Help v pravé dolní části okna. Po kliknutí na OK se funkce s definovanými parametry vloží na místo v programu označené černou šipkou.

Pro ovládání programu jsou důležitá následující tlačítka umístěná v horní liště:



Obr. B.3 Důležitá tlačítka v HDevelop

Algoritmus pro rozpoznávání koulí bude obsahovat následující úkony:

- zavření všech instancí Frame Graber (instance komunikující s grabovací kartou – pořizování nových snímků)
- zavření starého okna Graphics Window
- otevření nového okna Graphics Window ve velikosti 1280x450
- zavření všech instancí Frame graber
- vytvoření cesty ke konfiguračnímu souboru kamery
- vytvoření instance Frame grabber. Pomocí této instance se snímají nové snímky z kamery. Pro testování použijte snímky nasnímané v předchozím kroku
- vytvoření obdélníku, který bude určovat místo v obraze, kde bude obraz zpracováván
- nekonečnou smyčku while=1 obsahující:
  - Pořízení nového snímku
  - Zobrazení snímku v Graphics Window
  - redukci smínku na oblast, určenou obdélníkem vytvořeným v předešlých bodech
  - prahování snímku
  - spojení výsledných regionů
  - výběr regionů, které mají požadovaný tvar a velikost

Cílem algoritmu je, aby na konci smyčky byl jeden region reprezentující letící kouli.
Připravený projekt HDevelop naleznete ve složce Uloha s názvem treshold.dev

V programu budou použity tyto funkce(operátory):

`dev_close_window`
Zavře aktivní okno Graphics Window

`dev_open_window`
Otevře nové okno Graphics Window
Parametry:
Row1(Input): Vertikální souřadnice levého horního rohu okna
Col1(Input): Horizontální souřadnice levého horního rohu okna
Width(Input): Šířka okna
Height(Input): Výška okna
Background(Input): Barva pozadí okna
Window Handle(Output): identifikátor okna (název okna)

`close_all_framegrabbers`
Zavře všecny instance Frame grabber

`assign`

Přiřadí proměnné hodnotu. Pokud proměnná zatím neexistuje, vytvoří ji.

Input(Input): Hodnota proměnné. Pro vytvoření cesty k souboru sekvence.seq bude parametr mít například tvar: 'C:\\Uloha\\Grabber\\images\\sekvence.seq'
Pro cestu ke konfiguračnímu souboru kamery: 'C:\\Uloha\\camera.cam;CONFIG'

Result(Output):Výsledná proměnná

`open_framegrabber`

Vytváří novou instanci Frame grabber, pomocí které jsou pořizovány nové snímky z kamery/grabovací karty. Dle specifikovaných parametrů je možné snímky pořizovat nejen z kamery, ale i z již nasnímaných snímků uložených na pevném disku. Pro pořizování snímků ze souboru zadejte do parametru camera type cestu k souboru, který má příponu .seq a obsahuje názvy souborů.(aplikace Grabber tento soubor generuje). Pro pořizování snímků z kamery je potřeba zadat cestu ke konfiguračnímu souboru kamery. Oba případy jsou popsány u funkce `assign`.

Vybrané parametry, které je potřeba vyplnit:
Name(Input): Jméno Frame grabberu. Pro snímání z kamery použijte ‘INSPECTA5‘, pro snímání ze souborů ‘File‘
Cameratype(Input): zadejte proměnnou cesty vytvořenou v předchozím kroku
FGHandle(Output): identifikátor Frame grabberu

`grab_image`

Pořízení nového snímku
Parametry:
Image(Output): Jméno pořízeného snímku
FGHandle(Input): Identifikátor Frame grabberu ze kterého je snímek pořízen

`dev_display`

Zobrazí objekt(obrázek, region atd…) v aktivním okne Graphics Window
Parametry:
Object(Input): Objekt, který se má zobrazit

`gen_rectangle1`

Vytvoří objekt typu Region, který má tvar obdélníku. Region reprezentuje množinu bodů v obraze.
Parametry:
Rectangle(Output): Název výsledného regionu
Row1(Input): Vertikální souřadnice levého horního rohu
Column1(Input): Horizontální souřadnice levého horního rohu
Row2(Input): Vertikální souřadnice pravého dolního rohu
Column2(Input): Horizontální souřadnice pravého dolního rohu

`while`

Cyklus while. Podmínka je true pokud je různá od 0.
Parametry:
Condition(Input): Podmínka.

`reduce_domain`
Redukuje snímek na oblast určenou regionem. Výsledný obraz je výsledkem průniku regionu a vstupního obrazu.
Parametry:
Image(Input): Vstupní obraz
Region(Input): Region určující vybranou oblast obrazu
ImageReduced(Output): Výstupní redukovaný obraz

`fast_treshold`
Prahování snímku. Funkce vybere body s odstínem šedi *g* pro které platí:

$$MinGray < g < MaxGray$$

Výsledkem prahování je pole regionů.
Parametry:
Image(Input): Vstupní obraz
Region(Output): Výstupní pole regionů
MinGray(Input): Spodní hranice odstínu šedi
MaxGray(Input): Horní hranice odstínu šedi
MinSize(Input): Tento parametr slouží ke zrychlení prahování. V prvním kroku jsou prahovány pouze body s rozestupy MinSize. V druhém kroku jsou prahovány okolí bodů(MinSize x MinSize), které splnily podmínku v prvním kroku.

`connection`
Spojí okolní regiony do jednoho regionu. Při spojování je v základním nastavení použito 8-okolí.
Paramtery:
Region(Input): Vstupní regiony
ConnectedRegions(Output): Výstupní spojené regiony

`select_shape`
Vybere z regionů pouze ty, co vyhovují definovaným podmínkám.
Parametry:
Regions(Input): Vstupní regiony
SelectedRegions(Output): Výstupní vybrané regiony
Features(Input): Požadované vlastnosti regionů. Celý seznam možných vlastností je možné najít v dokumentaci.(tlačítko Help). Pro úlohu jsou důležité vlastnosti area-velikost regionu a circularity – podobnost kruhu.Při použití více vlastností najednou je nutné je zadat jako pole. Syntaxe je následující: ['area','circularity']
Operation(Input): Operátor mezi vlastnostni. Možné hodnoty: ‘and’,’or’
Min(Input): Minimální hodnota pro vlastnost. Při použití více vlastností jsou hodnoty zadány jako pole hodnot se stejným pořadím jako pole vlastností. Například [150,0.5]
Max(Input): Maximální hodnota pro vlastnost. Například [1000,1]

B.1.3 Použití parametrů v připravené aplikaci+komunikace s PLC

Po nastavení parametrů algoritmu tak, aby fungoval správně, je možné tyto parametry použít v aplikaci OBarrier, která je umístěna ve složce úloha. Po zmáčknutí tlačítka Start je

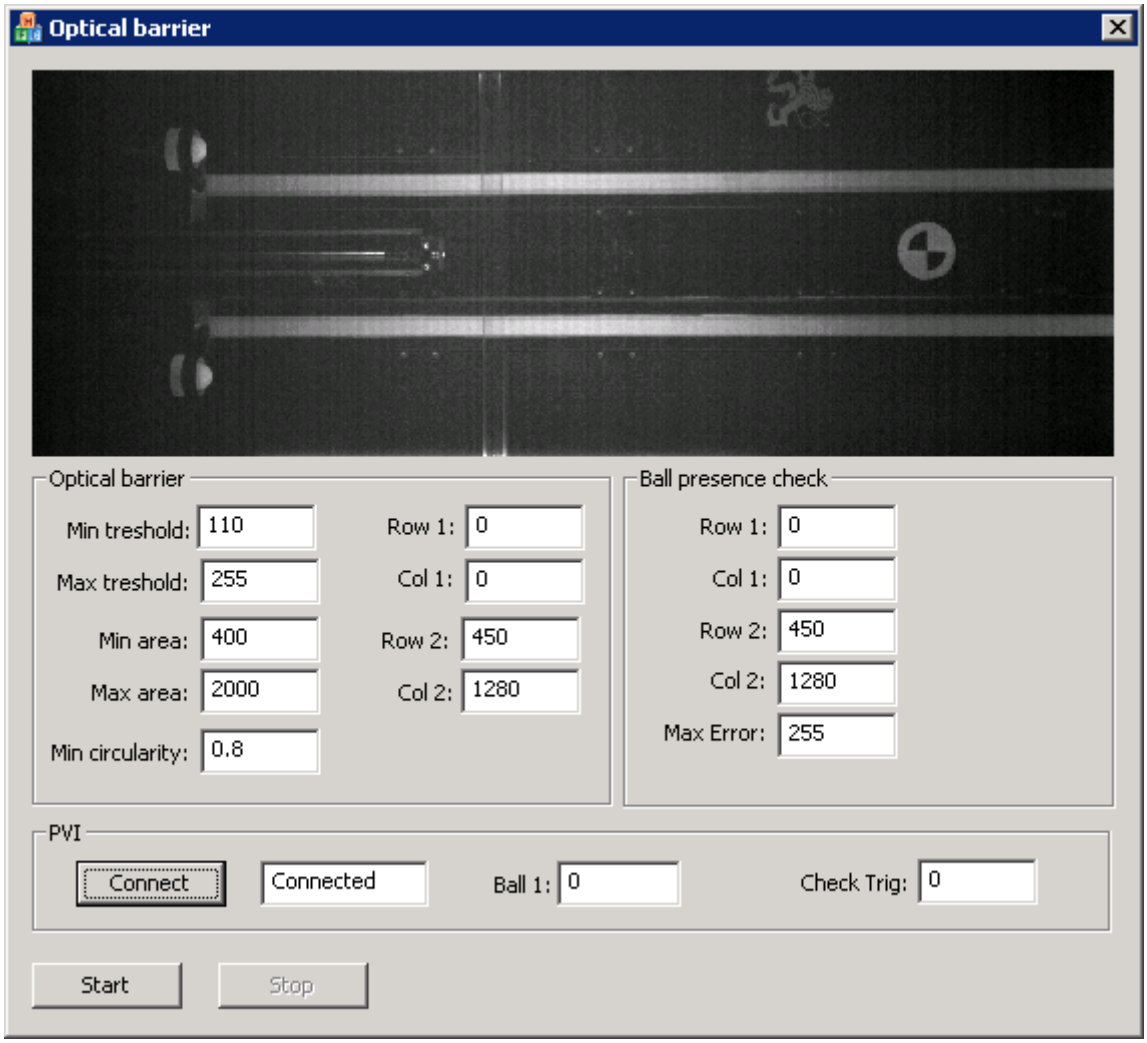
spuštěno zpracování obrazu, které funguje na principu zadání kroku 2 s parametry, které jsou do aplikace zadány.

Komunikace s řídícím PLC

Aplikace komunikuje s PLC pomocí proměnných umístěných v PLC v tasku komun. Konkrétně se jedná o tyto proměnné:

`Ball1` – tato proměnná je nastavena do „1“ kdykoliv je ve snímaném obrazu nalezena letící koule

`Check_balls_trig` – na náběžnou hranu této proměnné je spuštěn algoritmus vyhodnucijící přítomnost koule v misce. V případě, že je koule přítomna, je proměnná v PLC nastavena aplikací zpět do „0“. Pokud koule chybí, není provedena žádná akce.



Obr. B.4 Aplikace OBarrier

B.2 Kontrola přítomnosti koule ve vyhazovací misce

Pro kontrolu koule na začátku cyklu bude použita metoda template matching. V prvním kroku bude vytvořena předloha(template) vyhazovací misky obsahující kouli. Tato předloha pak bude hledána v oblasti snímku, kde bude kontrola přítmnosti probíhat. Na základě odlišnosti předlohy a snímku potom bude vyhodnoceno, zdali je potřeba kouli nahodit, nebo může cyklus začít.

B.2.1 Pořízení snímku s předlohou

V tomto kroku využijeme opět aplikace Grabber. Tentokrát funkci pořízení jednotlivého snímku (tlačítko Grab+save). Necháme najet vyhazovací misku žongléra (s koulí)do kontrolního bodu a po dosažení požadované pozice uložíme snímek z kamery. Snímek je ukládán do složky images ve složce Grabber ve formátu `pic_rrrrmmdd_hhmmss.tif`. Stejným způsobem vytvoříme snímek bez přítomné koule.

B.2.2 Vytvoření předlohy v HDevelop

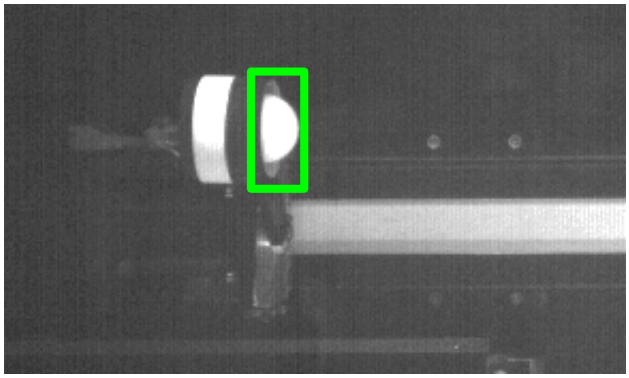
Cílem algoritmu je vytvořit předlohu(template) a uložit ji do souboru template.tmp, který bude použit v připravené aplikaci. Dále je cílem zjistit hodnotu výstupního parametru Error funkce best_match při přítomnosti/nepřítomnosti koule. Na základě těchto hodnot určete hranici, pro které hodnoty Error bude program vyhodnocovat přítomnost koule. Připravený projekt HDevelop naleznete ve složce Uloha s názvem template.dev.

Algoritmus bude obsahovat následující úkony:

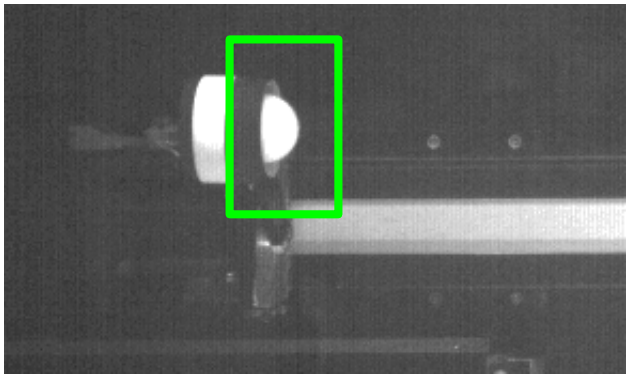
- zavření starého okna Graphics Window
- otevření nového okna Graphics Window ve velikosti 1280x450
- načtení snímku pořízeného v předchozím kroku s přítomnou koulí
- zobrazení snímku
- označení oblasti s vyhazovací miskou. Oblast volte tak aby v ní byla pouze koule a bílá část vyhazovací misky. viz obr. B.5
- vytvoření obdélníkového regionu s parametry získanými z předchozí funkce
- redukce snímku na označenou oblast
- vytvoření předlohy
- uložení předlohy do souboru template.tmp

- vytvoření obdélníku, který bude určovat místo v obraze, kde bude předloha hledána(oblast hledání musí být větší než předloha). Oblast volte tak, aby neobsahovala bílý pruh na vyhazovací misce. viz obr B.6.

- načtení snímku bez přítomné koule
- redukce snímku na oblast hledání
- hledání předlohy v redukovaném obraze



Obr. B.5 Příklad oblasti vyhazovací misky s koulí



Obr. B.6 Příklad oblasti pro detekci přítomnosti koule

Pro sestavení programu druhé části úlohy budou potřeba navíc tyto funkce(operátory):

`read_image`
Načte obraz ze souboru
Parametry:
Image(Output): Název načteného obrazu
FileName(Input): Cesta k souboru. Například: 'C:\\Uloha\\img1.tiff'. V případě, že je obrázek umístěn ve stejné složce jako projekt HDevelop stačí pouze jméno souboru: 'img1.tiff'

`draw_rectangle1`
Nakreslení obdélníku do okna Graphics Window. Stisknutím levého tlačítka myši a táhnutím je možno vytvořit obdélník na libovolném místě v obraze. Po stisknutí pravého tlačítka myši je nakreslený obdélník potvrzen a souřadnice horního levého a pravého dolního rohu jsou zapsány do výstupních parametrů.
Parametry:
WindowHandle(Input): Identifikátor okna pro kreslení
Row1(Output): Vertikální souřadnice levého horního rohu
Column1(Output): Horizontální souřadnice levého horního rohu

Row2(Output): Vertikální souřadnice pravého dolního rohu
Column2(Output): Horizontální souřadnice pravého dolního rohu

`create_template`
Vytváří předlohu pro template matching.
Parametry:
Template(Input): Obraz, který má být použit jako předloha
FirstError: v Halcon 7.1 není implementováno, ponechte hodnotu 255
NumLevel(Input): Specifikuje počet úrovní šedi, ve kterých má být předloha hledána. Hodnota 1 znamená, že bude hledáno pouze v originálních odstínech šedi, což je pro tuto úlohu vhodné.
Optimize(Input): Optimalizace pro zrychlení hledání. Možné hodnoty: 'none','sort'
GrayValues(Input): Typ filtru, který má být použit. Pro úlohu zvolte 'original' (bez filtru)
templateID(Output): Identifikátor vytvořené předlohy.

`write_template`
Vyvoří předlohu(template) pro template matching.
Parametry:
TemplateID(Input): Identifikátor předlohy
FileName(Input): Jméno souboru. Pro uložení použijte název 'template.tmp', soubor se uloží do stejné složky, kde je uložen aktuální projekt HDevelop.

`best_match`
Hledá místo nejlepší shody předlohy a snímku.
Parametry:
Image(Input): Obraz, ve kterém je předloha hledána.
TemplateID(Input): Identifikátor předlohy
MaxError(Input): Parametr pro zrychlení hledání. Pokud průměrná chyba přesáhne tuto hodnotu, je předloha hledána na dalším místě. Pro úlohu zvolte 255.
SubPixel(Input): Udává, jestli výsledná pozice může být s přesností na desetinná místa.
Row(Output): Vertikální souřadnice rozeznané předlohy v obraze
Column(Output): Horizontální souřadnice rozeznané předlohy v obraze
Error(Output): Průměrný rozdíl odstínů šedi mezi předlohou a místem, kde byla předloha nalezena.

B.2.3 Použití předlohy a parametrů v připravené aplikaci

Soubor template.tmp, který byl vytvořen v předchozím kroku zkopírujte do složky OBarrier. V aplikaci vyplňte parametry získané v předchozím kroku. Aplikace komunikuje s řídícím PLC jak bylo popsáno v části B.1.3. Na základě této komunikace můžete do řídícího programu zařadit signalizaci chyby semaforem, nebo zajistit nahození nové koule.