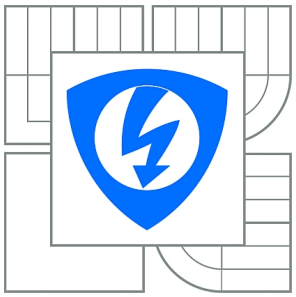
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

SLEDOVÁNÍ CÍLE V PROSTORU

TARGET TRACKING

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

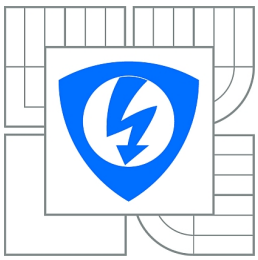
AUTOR PRÁCE — JAN ČERNÍN

AUTHOR

VEDOUCÍ PRÁCE — Ing. ILONA KALOVÁ, Ph.D.

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Jan Černín	***ID:***	109641
Ročník:	3	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Sledování cíle v prostoru

POKYNY PRO VYPRACOVÁNÍ:

Úkolem studenta je sledování vybraného cíle v prostoru s pomocí analýzy obrazu z kamery umístěné na polohovací hlavici (dva motorky ovládané přes sériovou linku). Součástí práce je i vhodné nastavení parametrů objektivu (zoom, fokus, clona).

DOPORUČENÁ LITERATURA:

Hlaváč, V., Šonka, M.: Počítačové vidění. Praha: Grada, 1992.
Haußecker H., Geißler P.: Handbook of Computer Vision and Applications. San Diego: Academic press, 1999.
a další dle pokynů vedoucího.

Termín zadání:	8.2.2010	***Termín odevzdání:***	31.5.2010

Vedoucí práce: Ing. Ilona Kalová, Ph.D.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Předmětem této bakalářské práce bylo naprogramovat algoritmus, který by byl schopen zpracovat obrazová data z připojené kamery. Program pracuje s daty v reálném čase a na základě vyhodnocené pozice objektu zajistí natočení polohovací hlavice tak, aby kamera sledovala vybraný cíl. Celý algoritmus implementuje metody sledování psané v programovacím jazyce C/C++ využívající funkcí knihovny OpenCV.

KLÍČOVÁ SLOVA

Digitální obraz, Sledování cíle, Histogram, MeanShift, Optický tok, Významné body

ABSTRACT

The subject of this bachelor thesis was to program an algorithm, which would be able to process image data from connected camera. Real-time application is working with data in basic of evaluated position of the object and control robotic head that turns and follows selected target. Whole algorithm implements two methods of tracking which are written in program language C/C++ using functions of OpenCV library.

KEYWORDS

Digital image, Target tracking, Histogram, MeanShift, Optical Flow, Interest points

ČERNÍN, J. *Sledování cíle v prostoru*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 60 s. Vedoucí bakalářské práce Ing. Ilona Kalová, Ph.D.

PROHLÁŠENÍ

“Prohlašuji, že svou bakalářskou práci na téma Sledování cíle v prostoru jsem vypracoval samostatně pod vedením vedoucího semestrální práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této semestrální práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Iloně Kalové, Ph.D, za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Předmětem této semestrální práce bylo seznámení se s metodami zpracování obrazu, především práce s obrazovými daty získávaných z USB kamery připojené k PC. Pro práci s obrazem byly využity vlastnosti programového prostředí Matlab s nainstalovaným Image Processing Toolboxem, avšak větší část práce byla zpracovávána s využitím programovacího jazyka C/C++ a knihovny OpenCV 2.0. V prostředí Matlab byla osvojena práce s objektivem kamery a následné vhodné nastavení parametrů objektivu, jako je clona, focus a zoom. Po zpracování této části zadání se pokračovalo k hlavnímu a nejvíce podstatnému bodu týkající se této bakalářské práce a to sledování pohyblivého cíle v prostoru. K splnění tohoto bodu bylo využito vlastností a funkcí knihovny OpenCV 2.0, která v kombinaci s připojenou barevnou USB kamerou značným způsobem zjednodušila a urychlila následnou práci s obrazem. O pohyb kamery se starají dva servomotory Dynamixel RX64, které jsou připojeny k PC pomoci sériové linky. V této bakalářské práci bylo použito dvou metod sledování cíle, které s jistým omezením umožňují vybrat libovolně složitý terčík nebo cíl (např. obličej) ve sledované scéně. Výpočetní algoritmus vyhodnocuje polohu těchto zvolených cílů a následně provede natočení kamery na zjištěné souřadnice. Cílem bakalářské práce bylo učinit tento proces z pohledu uživatele částečně automatizovaným a schopným cíl z předem definovaných parametrů v prostoru vyhledat a natáčet kamerou ve směru pohybu cíle.

První bod zadání se zabývá problémem automatického ostření objektivu. Jde tedy o to, aby programem ovládaný mechanismus objektivu zaostřil sledovanou scénu tak, že bude pro lidské oko ostrá. Toho nejčastěji využívají digitální fotoaparáty s funkcí automatického ostření tzv. autofocus.

V druhém bodě zadání bylo nejprve nutné seznámit se s principem a funkcí servomotorů Dynamixel RX64 a jejich ovládání po sériové lince. Dále již zbývalo detekovat pohyblivý cíl a vypočítat nutný posun kamery tak, aby došlo k posunu jak horizontálním tak vertikálním směrem a objekt byl ve středu snímané scény.

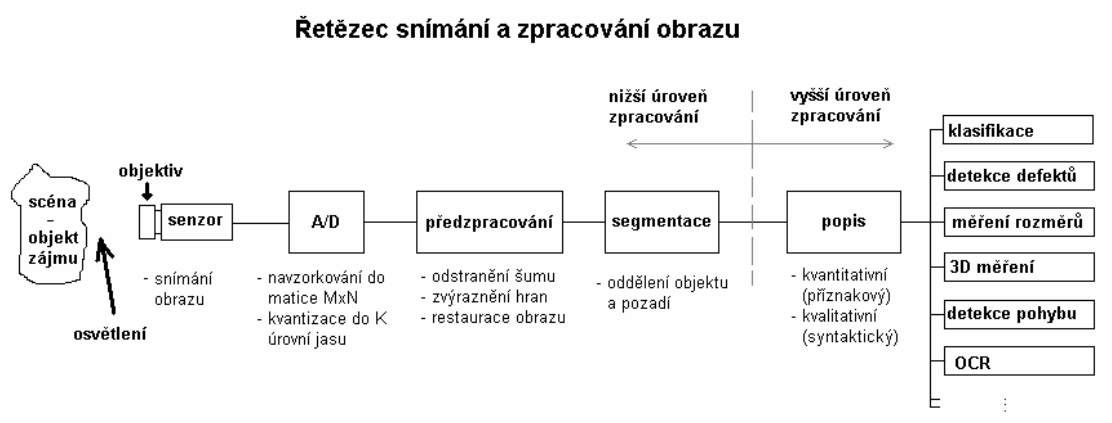
1. POJMY

1.1 DIGITÁLNÍ OBRAZ

Digitální obraz je reprezentací dvojrozměrného obrazu v podobě jedniček a nul. Dále se rozlišují dva typy obrázků, vektorový a rastrový tzv. bitmapový. V této bakalářské práci se pracuje pouze s rastrovou grafikou. Nejmenší částí rastrového obrazu je pixel. Tyto body jsou uspořádány do mřížky. Každý bod má jednoznačně určenou pozici a barevnou hloubku. Tento způsob reprezentace obrazových dat používá např. televize nebo fotoaparát. Kvalitu digitálního obrazu určuje rozlišení a barevná hloubka. U obrazu se setkáváme s barevnými modely, jako je třeba RGB nebo HSV. Jedná se o to, jak je reprezentována barva jednotlivého pixelu.

1.2 ZPRACOVÁNÍ OBRAZU

Zpracování obrazu se sestává z několika části a funkčních bloků, které dohromady tvoří řetězec snímání a zpracování obrazu. V bakalářské práci je kladen důraz především na segmentaci obrazu, kde došlo k oddělení objektu od pozadí a poté k identifikaci objektu a určení jeho pozice tak, aby byl výpočetní algoritmus co nejrychlejší a dokázal reagovat na rychlé změny pozice sledovaného cíle.



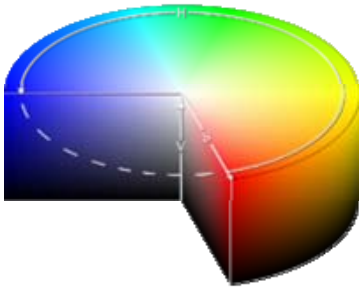
Obrázek 1.1 Řetězce snímání a zpracování obrazu (převzato [1])

1.3 RGB

Barevný model RGB (red,green,blue), je aditivní způsob míchání barev. Každá barva je udána mohutností tří základních barev. Výsledná barva závisí na tom, jaká složka z těchto tří má nejsilnější odstín. Tento způsob reprezentace barvy používají všechny projektory a monitory. Smícháním všech barev RGB získáme barvu bílou, opakem je barva černá. Každá barva je dána mohutností. Mohutnost jednotlivých barev také stanoví barevná hloubka, která určuje kolik bitů je vymezeno pro barevnou komponentu. Může být udávána také v procentech. Máme-li barevnou hloubku 8 bitů, bude červená barva určena 256 způsoby tedy od 0 do 255 [2].

1.4 HSV

HSV barevný model nejvíce odpovídá lidskému vnímání barev, kde je barva určena třemi složkami Hue, Saturation, Value (Odstín,Sytost,Jas). Jakým způsobem barva prochází nebo je odražena od objektu určuje tzv. odstín. Pod sytostí se rozumí čistota barvy tedy množství šedi v odstínu. Hodnota jasu reprezentuje množství bílého světla. Jas vyjadřuje, kolik světla barva odráží. Jednotlivé složky jsou vidět na barevném modelu válce, kde obvod válce představuje hue (odstín), vzdálenost od středu saturation (sytost) a výška válce value (jas) [2].

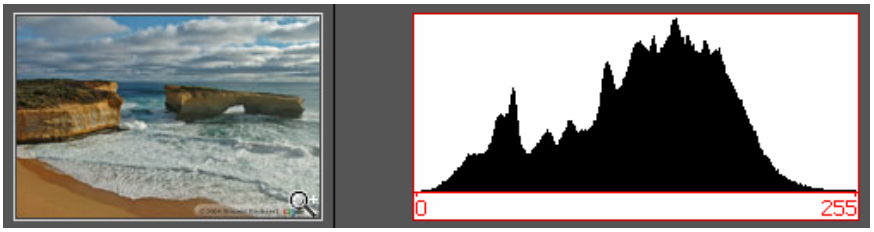


Obrázek 1.2 Grafické zobrazení HSV (převzato z [2])

1.5 HISTOGRAM

Pod pojmem histogram se rozumí graf, který zobrazuje rozložení barevných kanálů - červeného, zeleného, modrého (RGB), nebo šedé stupnice a některých dalších veličin (např. barvy, sytosti barvy, světlosti barvy) v obrázku. Histogram se používá k analýze obrazu před následující úpravou. Horizontální osa nabývá hodnot od 0 do

255 a vertikální osa udává počet pixelů příslušející hodnotě na horizontální ose. Histogram na následujícím obrázku byl vytvořen ze složky jasu snímku.



Obrázek 1.3 Ukázka histogramu obrázku (převzato [3])

1.6 POČÍTAČOVÉ VIDĚNÍ

Počítačové vidění je velmi rozsáhlá oblast, která se zabývá zachytáváním, zpracováním a reprezentací obrazových dat. Snahou této disciplíny je, se v co nejvyšší míře přiblížit a napodobit lidské vidění. Svým využitím najde uplatnění v mnoha vědních a technických oborech, od navádění robota na cíl po rozpoznávání nádorových buněk. Podstatou počítačového vidění je porozumění snímané 3D scéně, například tak, jak ji vidíme při pohledu z okna na ulici. Postupy počítačového vidění jsou značně složité, s kladeným důrazem na interpretaci obrazových dat, která jsou nejčastěji reprezentována symbolicky. V obecném případě chceme porozumět získaným obrazovým datům, o jejichž vlastnostech není nic známo.

Ačkoliv můžou být obrazová data perfektní, postupy pro rozpoznávání objektů jsou poměrně náročné a vyžadují znalost hledaného objektu. Mezi časté nevýhody, které ve značné míře přispívají ke složitosti zpracování dat, patří šum a ztráta informace třetího rozměru vlivem projekce. Tyto nedostatky je možné ignorovat či omezit různými korekčními metodami. Vše se odvíjí od návrhu aplikace a také účelu jejího použití.

Výhodou počítačového vidění je fakt, že není omezeno pouze na zpracování obrazových dat námi viditelného spektra. Objekty v reálném světě odrážejí a vyzařují energii i jiných vlnových délek. Za pomoci moderních technologií jsme schopni snímat např. ultrafialové, infračervené nebo tepelné spektrum. Díky tomu je počítačové vidění možné aplikovat v mnoha vědních oborech [4].

1.7 ZÁKLADY SLEDOVÁNÍ OBJEKTU

Na rozdíl od statických snímků, se při práci s pohyblivým videem často setkáme s konkrétním objektem, jehož trajektorii pohybu bychom rádi sledovali. K vypořádání se s tímto problémem je nejprve zapotřebí porozumět pohybu tohoto objektu. Tato úloha se skládá ze dvou hlavních komponent: identifikace a modelování.

Pod pojmem identifikace se rozumí nalezení objektu zájmu v sekvenci snímků. Existuje několik technik detekce objektu např. pomocí momentů nebo barevného histogramu. Při sledování neznámého cíle je dobré se zaměřit na důležité aspekty jeho pohybu. Právě pohyb, může tento objekt dělat zajímavým. Techniky sledování často využívají vizuálně významné body. Tedy body, které nesou nějakou zajímavou informaci.

Často je třeba určitým způsobem předpovídat pohyb objektu. Tím se zabývá druhá komponenta a to modelování. V této souvislosti bylo vyvinuto mnoho matematických metod, schopných odhadovat pohyb sledovaného cíle. Metody je možné aplikovat na 2D nebo 3D modely objektů a jejich pozice [4].

Blíže k použitým metodám sledování pohyblivého cíle v rámci této práce viz kapitola 5. Popis použitých metod.

2. POUŽITÁ APARATURA

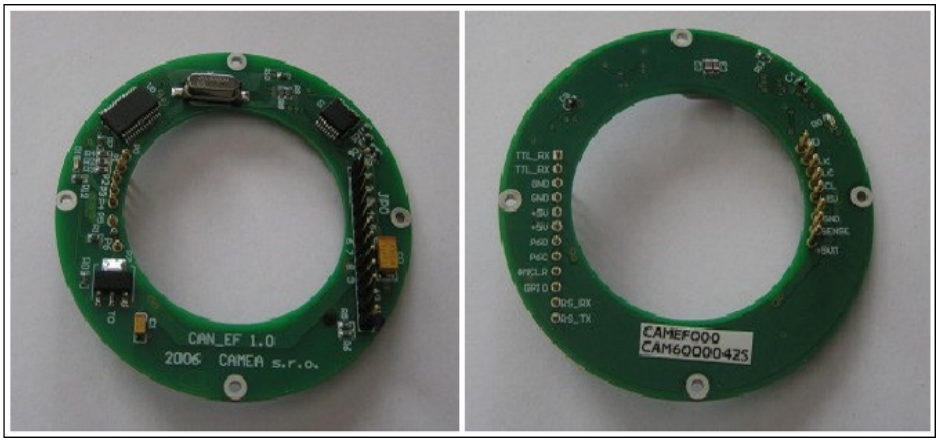
Pro získávání obrazu scény byla použita digitální USB kamera DFK 21BU04 firmy IMAGING SOURCE s CCD čipem o velikosti 1/4" a rozlišením obrazu 640x480 pixelů. Na kameru byly dále nasazeny dva objektivy. První z nich, TAMRON 28-80 f/3,5-5,6 s bajonetem CANON, byl využit ke splnění části zadání nastavování vhodných parametrů objektivu. Pro sledování pohyblivého cíle v prostoru byl zvolen objektiv s pevnou ohniskovou vzdáleností TV LENS F1.2 8 mm.

Pozice kamery je ovládána pomocí dvou motorů Dynamixel RX-64 sestavených tak, aby umožňovaly pohyb ve směru horizontálním tak i vertikálním. Oba motory jsou napájeny stejnosměrným napětím o velikosti 18V a komunikace s nimi probíhá po sériové lince protokolem RS-485. Především z nutnosti zajistit rychlou odezvu na změnu polohy, se tyto pohony jevily jako dobře zvolené vzhledem k charakteru aplikace.

Obrázek 2.1 Použitá aparatura, rameno s kamerou

2.1 REDUKCE CANON EF

CANON EF je modul vytvořený firmou CAMEA spol. s r.o. umožňující řízení objektivu CANON pomocí RS-232 nebo TTL-232. Je vložen do redukce mezi kamerovým závitem C mount a závitem Canon. Podrobnější popis modulu včetně jeho řízení viz [5].



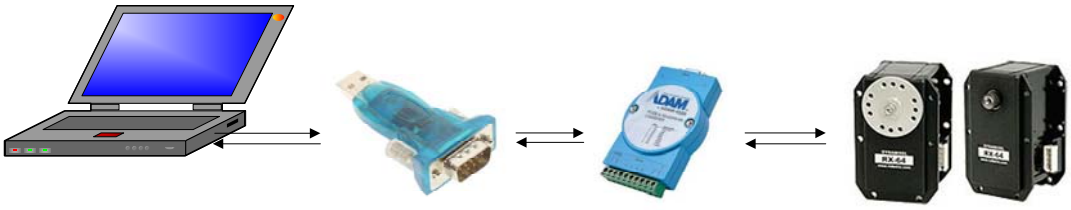
Obrázek 2.2 Mezikroužek Canon EF (CAMEA spol. s r.o.)

2.2 OBJEKTIVY

Obrázek 2.3 Použité objektivy

2.3 PŘEVODNÍKY

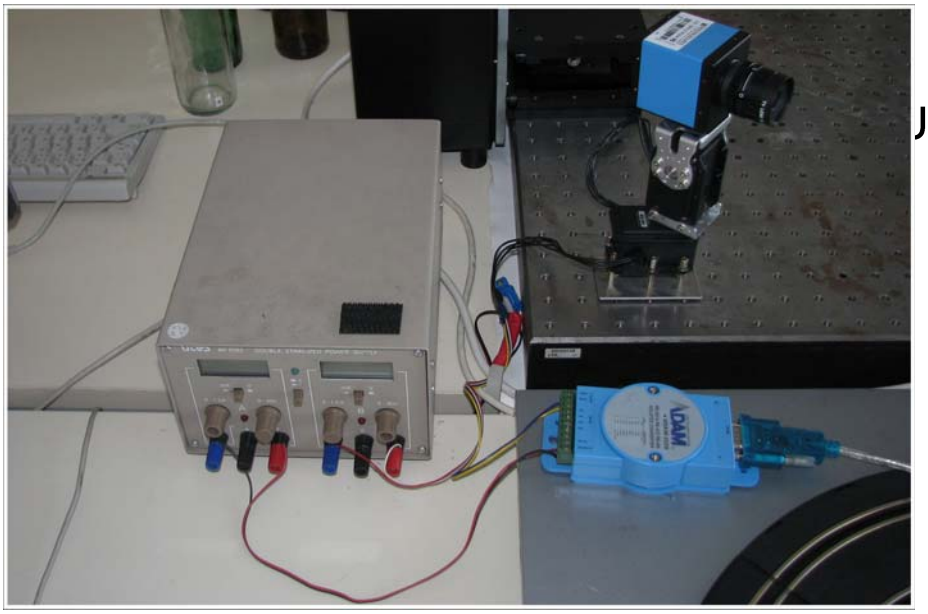
Aby bylo možné motory ovládat z PC, který není opatřen rozhraním sériové linky RS-232, bylo zapotřebí použít následující převodníky. První z nich (USB2RS232 firmy Wiretek) zajišťuje převod z USB portu počítače na protokol komunikace RS232. Druhý převodník (ADAM-4520) již tvoří most mezi protokoly RS232 a RS485.



Obrázek 2.4 Komunikační proces

2.4 NAPÁJENÍ

Napájení celého pracoviště zajišťuje zdroj stabilizovaného stejnosměrného napětí s výstupem 2x 0-30V. První napěťový výstup (8V) napájí převodník ADAM-4520. Motory Dynamixel RX-64 jsou napájeny 18V z druhého napěťového výstupu. Uspořádání celého pracoviště viz Obrázek 2.5.



Obrázek 2.5 Pracoviště, použitá aparatura

2.5 DYNAMIXEL RX-64

Jedná se o inteligentní pohon nové generace, v jehož funkčním bloku je integrován mikročip, ovladač a síťové rozhraní. Výhodou těchto pohonů je možnost sestavení a propojení jednotlivých modulů do libovolných tvarů.

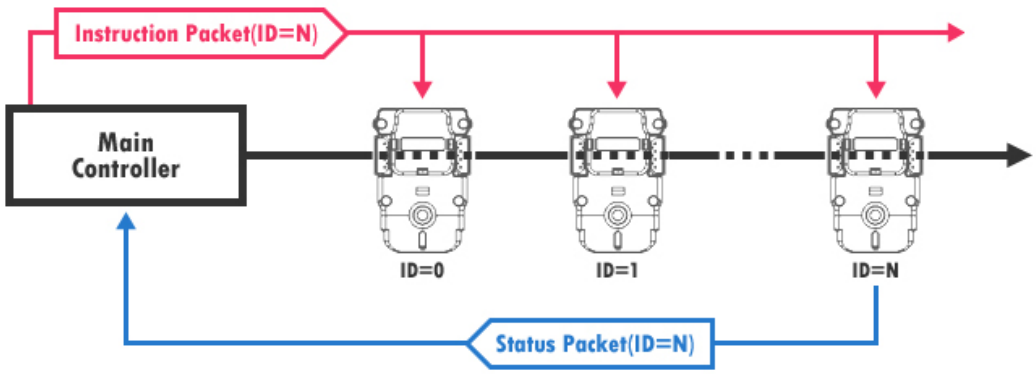
2.5.1 Specifikace

	RX-64	
Váha [g]	125	
Rozměry [mm]	40,2 x 61,1 x 41,0	
Nastavitelnost momentu	1/200	
Napájecí napětí [V]	15	18
Stop moment [Kg.f/cm]	64,4	77,2
Rychlost [sec/60°]	0,188	0,152
Rozlišitelnost [°]	0,29	
Pracovní rozsah [°]	300, nekonečná smyčka	
Pracovní napětí [V]	12~21	
Maximální proud [mA]	1200	
Pracovní teplota [C°]	-5 ~ +85	
Protokol	Asynchronní RS-485	
ID	254 ID (0-253)	
Komunikační rychlost	7343 bps ~ 1Mbps	
Motor	Maxon RE-MAX	

Tabulka 2.1 Specifikace pohonu RX-64 (převzato [6])

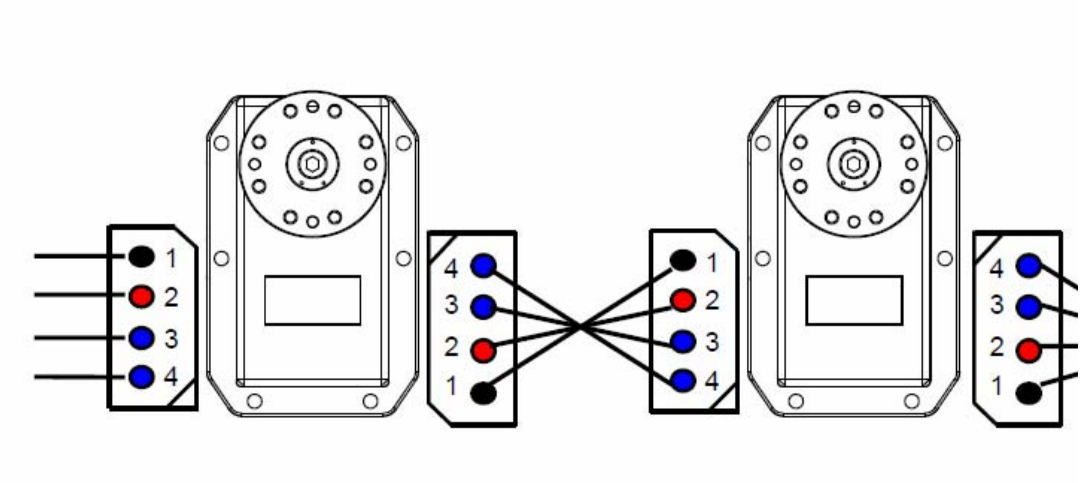
2.5.2 Zapojení pohonů

Každý modul vlastní svoje nastavitelné identifikační číslo ID zařízení, umožňující spojení až 254 zařízení na jedné sériové lince.



Obrázek 2.6 Propojení a komunikace zařízení (převzato [6])

2.5.3 Propojení



Obrázek 2.7 Propojení motorů (převzato [6])

Ke komunikaci se používají pouze dva vodiče (Data+ a Data-). Další dva jsou vodiče napájecí viz. Tabulka 2.2. Pokud je vše správně zapojeno, dojde k potvrzení probliknutím led diody pohonů.

Přiřazení pinů:	
PIN 1	GND
PIN 2	VDD
PIN 3	Data +
PIN 4	Data -

Tabulka 2.2 Přiřazení pinů (převzato [6])

3. VHODNÉ NASTAVENÍ PARAMETRŮ OBJEKTIVU

V této části bylo využito vlastností programového prostředí Matlab, které svými funkcemi a metodami umožňuje snadný přístup k periferiím počítače. Aby bylo možné ovládat parametry objektivu TAMRON s bajonetem CANON, musel být použit mezikroužek CANON EF, který je vložen mezi objektiv a kameru. Komunikace s tímto modulem probíhá po sériové lince RS-232.

3.1 PŘIPOJENÍ KAMERY A POŘIZOVÁNÍ DAT

Pro zpracování dat pořízených kamerou bylo nejprve nutné správně nakonfigurovat vstupní zařízení. Jak již bylo řečeno, kamera je digitální s možností připojení přes rozhraní USB. Nutností bylo doinstalování ovladačů zařízení, bez kterého by nebylo možné se zařízením správně komunikovat. Vlastnosti a parametry připojeného hardwaru byly zjištěny funkcí `imaqhwinfo()`. Tato funkce vrací datový typ typu struktura, v níž data obsahují informace jako je podporované rozlišení kamery, barevný model a identifikační číslo ID připojené kamery.

Použitím funkce `videoinput()`, byl vytvořen datový objekt vstupního proudu videa. Tento objekt zprostředkovává styk s prostředím Matlab a vstupním zařízením pro získávání videa.

```
kamera=videoinput('winvideo',1);
```

Prvním vstupním parametrem této funkce je název adaptéru, který zajišťuje správný přístup k hardwaru a druhým parametrem je ID připojeného zařízení. Po vytvoření spojení již je možné získat live video náhled funkcí `preview()`, jejímž vstupním parametrem je objekt získaný funkcí `videoinput()`.

```
Preview(kamera);
```

K pořízení obrazových dat slouží funkce `getsnapshot()`. Výstupem je matice hodnot odpovídající podporovanému barevnému modelu kamery. Pokud

připojená kamera podporuje snímání v modelu RGB, bude výsledná matice hodnot trojrozměrné pole o velikosti [X,Y,3]. Kde X, Y je rozlišení kamery a třetí rozměr tohoto pole tvoří matice hodnot zastoupení jednotlivých barev (červená, zelená, modrá). V případě, že kamera podporuje pouze černobílé snímání obrazu, bude návratová matice pouze dvourozměrná.

3.2 ZOOM, FOKUS A CLONA

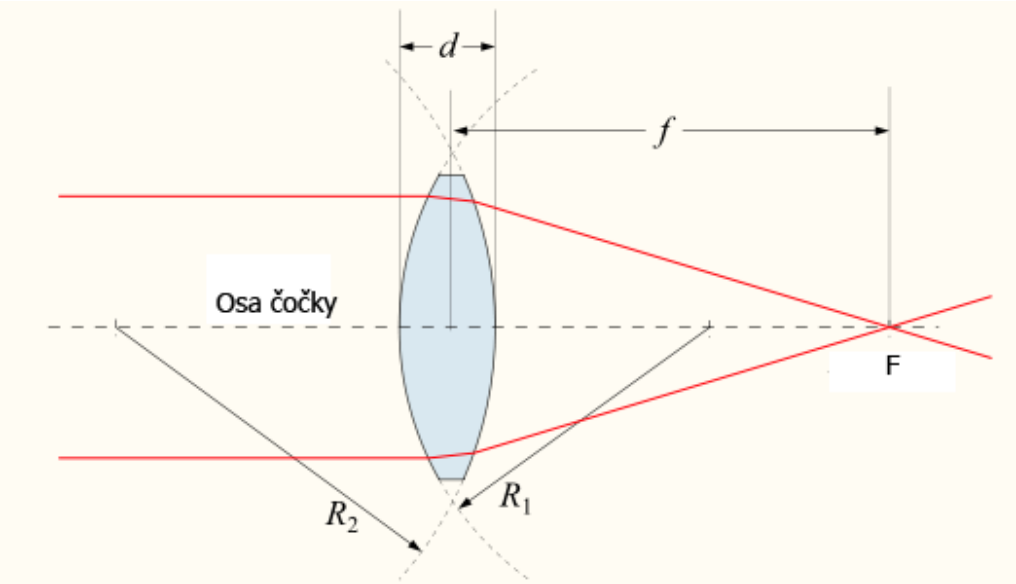
Na kameře je připevněn objektiv s redukcí, umožňující nastavení parametrů jako je ohnisková vzdálenost, světelnost a ostrost. Redukce Canon EF slouží k ovládání objektivů Canon pomocí RS-232 nebo TTL-232. Tento kroužek se vkládá mezi objektiv a kamerový závit C mount.

3.3 OSTŘENÍ

Objektivy lze rozdělit na objektivy s manuálním ostřením MF (manual focus) nebo s ostřením automatickým AF (automatic focus). Dále je lze rozdělit podle typu uspořádání vnitřních členů ostření (zadní, přední, vnitřní) nebo podle použitého systému motorků (ultrazvukové, klasické) a podle rychlosti ostření.

3.4 OHNISKOVÁ VZDÁLENOST

Použitý objektiv umožňuje nastavení ohniskové vzdálenosti od 28mm do 80mm. Protože se zoom u objektivu nastavuje manuálně, je zapotřebí použít vnější motorek s řemenovým převodem, který přenáší točivý moment na mechaniku objektivu. Na následujícím obrázku jsou zobrazeny parametry čočky (tloušťka d a poloměry R_1, R_2) spolu s ohniskovou vzdáleností f a ohniskem v bodě F (viz Obrázek 3.1).



Obrázek 3.1 Spojná čočka, ohnisková vzdálenost a ohnisko

3.5 SVĚTELNOST

Světelnost je dána podílem ohniskové vzdálenosti *f* a velikosti *d* vstupního otvoru

$$z = \frac{f}{d}.$$

Prvek umožňující změnu velikosti vstupního otvoru se nazývá clona. Právě tento otvor udává, jaké množství světla dopadne na snímací chip. Nejmenší možná nastavitelná clona udává světelnost objektivu. Čím menší clonové číslo tím lepší použitelnost při zhoršených světelných podmínkách.

3.6 ŘÍZENÍ OBJEKTIVU

Aby bylo možné komunikovat s redukcí Canon EF, je nejprve nutné navázat spojení po sériové lince RS-232.

V matlabu byl vytvořen objekt asociovaný s COM portem počítače pomocí příkazu `serial()`.

```
S=serial('COM1');
fopen(S);
fprintf(S, '\002x%s\003x', 'lfp0200');
A = fread(S, S.BytesAvailable);
```

Příkazem `fopen()` byla otevřena komunikace s příslušným COM portem počítače. Funkce `fprintf()` umožňuje zapsaní dat na daný port. Redukce se ovládá zápisem řetězce znaků na sériovou linku, kde \002 a \003 je start a stop bit následovaný ID jednotky x (libovolný znak) a třetí parametr je formát pro zápis string řetězce [5].

Příkaz `fread()` vyčte z portu daný počet dat.

3.7 ALGORITMUS PRO AUTOMATICKÉ OSTŘENÍ

Tento algoritmus byl napsán tak, aby ze sekvence snímků vybral obrázek, který byl při daném nastavením ostřící čočky ten nejostřejší. Nejprve dojde k nastavení nejmenší clony a zoom byl přizpůsoben tak, aby byl fokus přibližně v polovině. V následujícím cyklu dojde ke změně fokusu od prvního krajního bodu až po konec rozsahu. V každém kroku jsou získávána obrazová data pomocí příkazu `getsnapshot()`. Pořízené obrázky jsou dále pomocí funkce `im2bw()` převedeny z indexovaného černobílého obrazu na binární, na němž je aplikován hranový detektor. Funkce `edge()` ze vstupního obrazu vytvoří stejně velký výstupní binární obraz, ve kterém jsou nalezeny hrany.

Pro nalezení nejostřejšího snímku je v cyklu zaznamenáno množství bílé barvy, které bylo spočteno z obrazu hran pro aktuální nastavení fokusu objektivu. Ten obrázek, který obsahuje nejvíce hran, je potom nejostřejší a tím pádem obsahuje nejvíce bílé barvy. Na konci algoritmu dojde k vyhodnocení, pro které nastavení fokusu byl obrázek nejostřejší a dojde k jeho nastavení.

4. INTERFACE PROGRAMU SLEDOVÁNÍ POHYBLIVÉHO CÍLE

V této kapitole je popsán interface vytvořeného programu určený k sledování pohyblivého cíle v prostoru. Vše je realizováno na laboratorním přípravku se snímací barevnou USB kamerou. Touto kamerou je vyhodnocován pohyb zvoleného cíle s následným nastavením otočné hlavy sestavené tak, aby bylo možné kameru natáčet jak v horizontálním tak vertikálním směru. Jak již bylo řečeno, otočná hlava se skládá ze dvou servomotorů Dynamixel RX-64. Celý proces je řízen programem napsaným v programovacím jazyce C++ s využitím knihovny OpenCV 2.0. Program byl napsán ve vývojovém prostředí Microsoft® Visual Studio 2008 s využitím standardního kompilátoru MinGW, který je jeho součástí.

4.1 KNIHOVNA OPENCV 2.0

OpenCV 2.0 je svobodná, otevřená a multiplatfomní knihovna určená pro práci s obrazem. Je zaměřena především na počítačové vidění a na práci s videem v reálném čase. Je psána v jazyce C a umožňuje efektivně využít vlastností multiprocesorového jádra počítače. Knihovna obsahuje přes 500 funkcí, které pokrývají rozsáhlou oblast počítačového vidění od průmyslu, medicíny, robotiky až například k bezpečnostním systémům. Hlavní výhoda OpenCV je možnost použití všech částí knihovny ke komerčnímu využití s velice rozsáhlou podporou ze strany uživatelů i programátorů.

V této bakalářské práci bylo ve značné míře čerpáno z příkladů, které jsou součástí distribuce OpenCV. Tento kód byl dále upraven pro potřeby navrhované aplikace.

4.2 KNIHOVNA CSERIAL PORT

Tato freeware knihovna CSerialPort v1.27 zabaluje části Windows API (Application Programming Interface) do ucelených C++ tříd, které zajišťují použití a plnou kompatibilitu s většinou platforem operačních systémů Windows. Knihovna

zprostředkovává styk mezi motory Dynamixel RX-64 a PC po sériové lince a je volně stažitelná z webových stránek autora [7].

CSerialPort slouží jako rozhraní, z kterého se volají programové funkce Win32. Umožňuje tedy i odchytávání výjimek, vzniklých při nesprávném přístupu k funkci nebo portu. Díky tomu má programátor vetší kontrolu nad kódem, z kterého se funkce CSerialPort volají. Tím se program stává robustnější.

4.3 KOMUNIKACE S RX-64

Protože pohon RX-64 podporuje komunikační protokol RS-485 a PC, které bylo použito pro výpočty algoritmů a ovládání je vybaven sériovou linkou RS-232, bylo nejprve zapotřebí použít převodník ADAM-4520, který zajistí správnou konverzi mezi těmito komunikačními protokoly. Dále je ještě nutné zajistit správnou přenosovou rychlost převodníku. Ta musí být stejná jako přenosová rychlost pohonů RX-64, defaultně 57600bps. Pokud není port pro sériovou linku v PC k dispozici, je možné použít libovolný převodník USB2RS-232.

RX-64 je ovládán binárním daty zapouzdřenými v paketu. Použity jsou dva druhy paketu. Instrukční paket (Instruction paket) sloužící ke komunikaci s pohonem a status paket, který je odesílán z RX-64 jako odpověď. Pro komunikaci je použita asynchronní sériová komunikace bez parity, složena z 8 bitů s 1 Stop bitem operující na defaultně nastavené rychlosti 57600 bps.

Jedinečný ID zařízení v instrukčním paketu určuje, který RX-64 bude ovládán v případě, je-li propojeno několik zařízení najednou. Pokud má více jak jedno zařízení stejné ID, bude instrukční paket v síti kolidovat.

4.3.1 Ovládání motorů

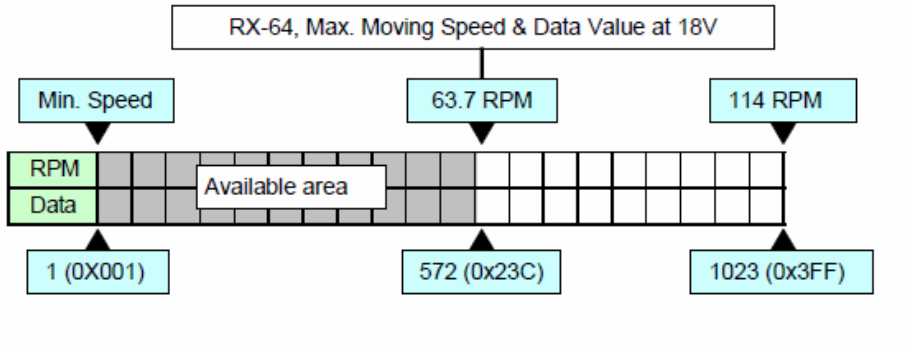
Aby bylo možné pohony dynamixel RX-64 ovládat, je nejprve nutné otevřít komunikaci s daným portem. K tomu byla použita již zmiňovaná knihovna CSerialPort. Pro objekt asociovaný s COM portem se nejprve dynamicky alokuje paměť, přičemž je volán defaultní konstruktor třídy CSerialPort. Spojení je poté vytvořeno voláním příslušné metody třídy s příslušnými parametry.

```
CSerialPort *COM = new CSerialPort;
COM->Open(13,57600,CSerialPort::NoParity,8,
          CSerialPort::OneStopBit,CSerialPort::NoFlowContro);
```

Prvním vstupním parametrem metody `Open()` je číslo portu, ke kterému je zařízení připojeno. Další parametry upřesňují nastavení portu.

Motor je schopen dosáhnout maximálního natočení 300°, s minimálním krokem 0,3°. Pozici 300°, tedy odpovídá hodnota 1023. Tato hodnota je předána jako parametr instrukčního paketu v hexadecimálním tvaru. Z toho vyplívá, že do kontrolní tabulky motoru jsou zapsány 2 bajty (0x3FF). Tento parametr se tedy skládá z hodnoty Low (1 bajt) a High (1 bajt) hexadecimálního čísla 0x3FF.

Druhým parametrem je rychlost pohybu motorů. Dosažitelná rychlost při napajení 18V viz Obrázek 4.1.



Obrázek 4.1 Rychlost při napájení 18 V (převzato z [6])

Instrukční paket

Instrukční paket je paket, v němž jsou zapouzdřena příkazová data v následujícím formátu.

0xFF	0xFF	ID	Length	Instruction	Param1	…	Param2	Check Sum

Tabulka 4.1 Instrukční paket (převzato z [6])

0xFF	Znamená začátek paketu.
ID	ID jednotky.
Length	Délka paketu, tzn. počet Parametrů (N) + 2.
Instruction	Pro ovládání pohonu je použita instrukce WRITE_DATA 0x03.
Check Sum	Výsledná suma odeslaných dat. Slouží k ověření správnosti přijetí dat.

Status paket

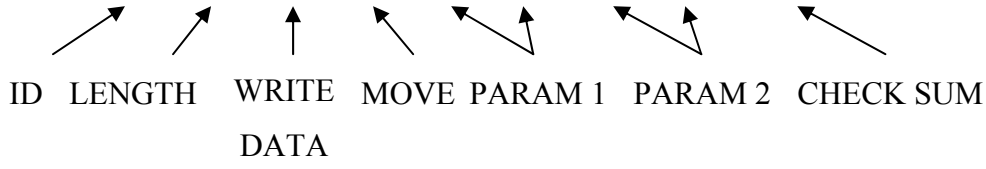
0xFF	0xFF	ID	Length	ERROR	Param1	…	Param2	Check Sum

Tabulka 4.2 Status paket (převzato z [6])

Příklad Instrukčního paketu pro pohyb na pozici 180° rychlostí 63.7 ot/min pro RX-64 s ID 1: Pohyb na pozici 0x1FF (Param 1) rychlostí 0x23C (Param 2).

Tabulka 4.3 Příklad instrukčního paketu

0xFF	0xFF	0x01	0x07	0x03	0x1E	0xFF	0x01	0x0C	0x23	0xA7



Takto vytvořený instrukční paket, sestavený jako pole hodnot typu unsigned char, zapíšeme na port voláním metody `Write()` třídy CSerialPort.

```
unsigned char paket[]={255,255,01,07,03,30,255,1,12,35};
port->Write(paket, sizeof(paket));
```

Prvním vstupním parametrem funkce je předávaný instrukční paket a druhým počet bytů zapisovaných na port.

Takto vytvořený instrukční paket otočí pouze jedním pohonem. Pohony dynamixel jsou schopny pohybu ve stejný čas. Tím se dosáhne plynulejšího a rychlejšího najetí na cíl. Zároveň došlo ke zkrácení doby nutné k čekání na provedení akce a je možné rychleji pokračovat ve výpočetním algoritmu. Instrukční paket bez vypočteného kontrolního součtu pak může vypadat následovně.

```
unsigned char paket[]={255,255,254,14,131,30,04,01,L1,H1,94,
                       01,02,L2,H2,94,01};
```

Kde L1, H1 je parametr motoru s ID1 a L2, H2 je parametr motoru s ID2.

4.3.2 Checksum instrukčního paketu

Hodnoty parametrů instrukčního paketu se za běhu programu neustále mění. Je tedy třeba pro každý paket dopočítat odpovídající kontrolní součet, který slouží ke

kontrole správnosti přenesených dat. Součet se přidá na konec každého instrukčního paketu. Výpočet checksumu se provádí podle následujícího vztahu

Check Sum = ~ (ID + Length + Instruction + Param1 + … Param N) (1).

4.4 ZÍSKÁNÍ SNÍMKU Z PŘIPOJENÉ KAMERY

Spojení s připojenou USB kamerou je realizováno prostřednictvím funkce `cvCaptureFromCAM()`. Tato funkce alokuje a inicializuje strukturu typu `CvCapture` pro čtení vstupního video streamu. Vstupem funkce je ID připojené kamery. Tato struktura tvoří pouze interface mezi programem a kamerou. Proto je nutné pro další práci s obrazovými daty pořídit snímek pomocí funkce `cvQuerryFrame()`, jejímž vstupem je právě vytvořená struktura `CvCapture`. Výstupem je pořízený snímek určený k dalšímu zpracování.

```
CvCapture capture = cvCaptureFromCAM(0);
IplImage *frame = cvQueryFrame( capture );
```

4.5 VYKRESLENÍ NAVIGACE

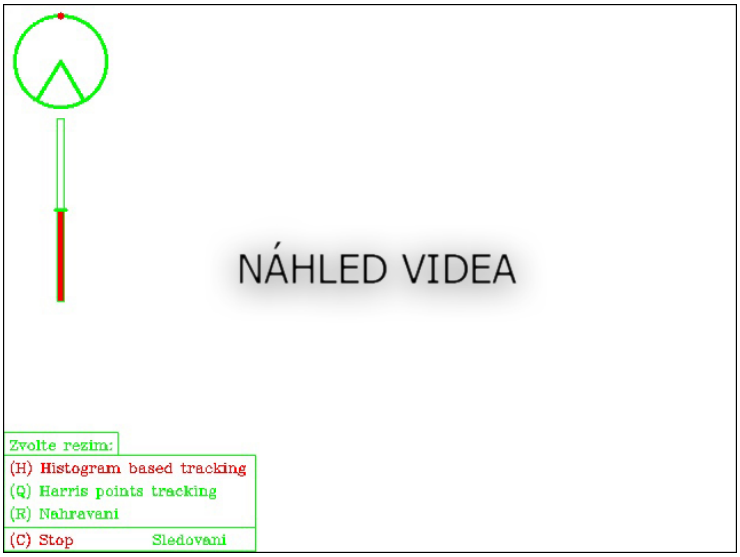
Vykreslení navigace má význam v případě, nachází-li se obsluha v jiném místě, než je umístěno rameno s kamerou. Pro orientaci v prostoru je dobré vědět, jakým směrem je kamera natočená. Poloha jednotlivých kloubů ramene se zobrazuje v levém horním rohu zobrazovaného videa. Indikace polohy motoru zajišťující horizontální posun je zobrazována na kružnici. Pracovní meze motoru zobrazují, při jakém natočení již motory nejsou schopny dalšího pohybu. Polohu vertikálního motoru zobrazuje ukazatel umístěný pod kružnicí. Pokud kamera snímá scénu vodorovně, nachází se ukazatel přímo uprostřed stupnice.

Menu programu je vykreslováno v levém dolním rohu okna s náhledem videa. Volba režimu je popsána slovně, spolu s údajem jakou klávesu je nutné stisknout, aby došlo k přepnutí metody nebo funkce programu. Vykresluje se zde i informace o průběhu sledování. Sledování cíle je indikováno zčervenáním položky „Sledování“.

K vykreslování čar byly použity funkce OpenCv. Rovné čáry jsou vykresleny funkcí `cvLine()`, jejímž vstupem je souřadnice počátku a souřadnice konce čáry. Volitelným parametrem může být i tloušťka vykreslované čáry. Kružnice je do obrazu vepsána funkcí `cvCircle()`. Vstupem této funkce je střed a obvod kružnice. K vykreslení čtyřúhelníku byla využita funkce `cvRectangle()`. Text menu byl do obrazu vykreslen pomocí funkce `cvPutText()`. K použití této funkce bylo ještě nutné definovat font textu. Zvolený font Hersey_complex je vstupem funkce `cvPutText()`, spolu se souřadnicí počátku textu a pole znaků obsahující samotný text.

4.5.1 Nahrávání sledované scény

Program umožňuje i nahrávání celého procesu sledování. Nahrávání se spustí stiskem klávesy „R“. Video stopa je uložena v avi kontejneru a je komprimováno pomocí kodeku MPEG-4. Celé video je zapsáno do kořenového adresáře, z kterého je program spouštěn. Počet snímků za sekundu závisí na rychlosti algoritmu, který je používán ke sledování. Pokud je při inicializaci struktury `CvVideoWriter()` použitý framerate (počet zpracovávaných snímků za sec.) menší než je framerate výpočetního algoritmu, bude se uložené video jevit zpomaleně. Naopak s vyšším frameratem bude video zrychlené. Vše závisí na hardwarovém vybavení použitého PC.



Obrázek 4.2 Ukázka vykreslované navigace

5. POPIS POUŽITÝCH METOD

5.1 NALEZENÍ VÝZNAMNÝCH BODŮ

V obraze lze nalézt mnoho výrazných bodů, vhodných pro sledování. Je ale důležité uvědomit si, co opravdu stojí za sledování. Pokud je například vybrán bod na čistém papíře, bude velmi obtížné najít ten samý bod na dalším snímku videa. Na tomto papíře se totiž spolu s vybraným bodem nacházejí další, velmi podobné tomu vybranému. Sledování tohoto bodu bude velmi obtížné. Naopak pokud bude vybrán bod, který je mezi ostatními unikátní, šance najít tento bod v dalším snímku se zvyšují.

Pod pojmem významný bod se tedy rozumí bod, který má matematicky dobře podloženou definici a jeho okolí je bohaté na informace vhodné pro pozdější zpracování vizuálním systémem. Tím se myslí body, které jsou co nejméně podobné svému okolí např. hranice objektů nebo vrcholy. Příklad nalezených bodů je vidět na následujícím obrázku (viz Obrázek 5.1).

Obrázek 5.1 Nalezené významné body (Shi and Tomasi)

Nejvíce používaným detektorem významných bodů je Harrisův detektor. Tento detektor se stal populárním, díky své invariantnosti k rotaci, změně měřítka,

osvětlení a šumu v obraze. Pro každý bod snímku je vypočítána autokorelační matice druhých derivací intenzit obrazu, v malém okolí zkoumaného bodu. Podle Harrise se roh v obraze nachází v tom místě, kde má autokorelační matice dvě velká vlastní čísla. To znamená, že nalezená hrana v místě bodu rozděluje prostor nejméně do dvou oddělených oblastí, stejně tak jako roh obecně, má nejméně dvě hrany.

Z této metody hledání významných bodů podle Harrise vycházejí další algoritmy jako je Shi and Tomasi, podle nichž nese významnou informaci pouze menší z vlastních čísel autokorelační matice. Pokud je toto číslo větší než prahová hodnota, jedná se o významný bod [4].

Shi and Tomasi algoritmus je implementován přímo ve funkci OpenCV a to `cvGoodFeaturesToTrack()`. Funkce počítá druhé derivace obrazu, z nichž jsou dále počítána vlastní čísla matic. Výstupem funkce je poté seznam bodů vhodných pro sledování.

```
void cvGoodFeaturesToTrack(
const CvArr* image,
CvArr* eigImage,
CvArr* tempImage,
CvPoint2D32f* corners,
int* corner_count,
double quality_level,
double min_distance,
const CvArr* mask = NULL,
int block_size = 3,
int use_harris = 0,
double k = 0.4);
```

Prvním vstupním parametrem je oblast zájmu v černobílém obraze. Jedná se tedy o uživatelem vybraný objekt. Další dva parametry jsou určeny pro ukládání pomocných dat (vlastní čísla autokorelačních matic). Souřadnice nalezených bodů jsou ukládány do pole struktur typu `CvPoint2D32f`. Každý prvek pole tedy obsahuje hodnotu souřadnice x a y. Toto pole musí být alokováno před voláním funkce. Proměnná `corner_count` určuje maximální počet hledaných bodů. Po dokončení funkce je tato hodnota přepsána množstvím nalezených bodů. Parametr `quality_level` je minimální hodnota vlastního čísla autokorelační matice. Pokud je vlastní číslo matice větší jak tato hodnota, bod je považován za významný. Typické hodnoty prahu jsou 0.10 nebo 0.01 a neměly by být větší než 1. Vstupní

hodnota `min_distance` určuje minimální vzdálenost dvou vedle sebe nalezených bodů.

Parametr maska je typu `CvArr` a určuje, které body mají být zpracovány. Pokud je například v masce pro příslušný bod log. 0, autokorelační matice pro tento bod nebude počítána. Velikost autokorelační matice určuje parametr `block_size`.

Předposlední vstupní hodnota funkce je proměnná `use_harris`. Pokud je hodnota nulová, k výpočtu je použita metoda Shi&Tomasi. Každá nenulová hodnota určuje metodu výpočtu významných bodů podle Harrise [4].

Body nalezené pomocí funkce `cvGoodFeaturesToTrack()` viz Obrázek 5.1 Nalezené významné body (Shi and Tomasi).

5.2 OPTICKÝ TOK

Optický tok reprezentuje směr a velikost pohybu v každém bodě obrazu v časové sekvenci vzhledem k následujícímu snímku. Každému bodu v obraze je přiřazena určitá rychlost pohybu. Bod tedy mezi aktuálním a předchozím snímkem urazí určitou vzdálenost. Výpočet optického toku je prováděn z intenzit obrazu vyvíjejících se v čase. Ideálními jsou tedy nalezené významné body (viz kapitola 5.1). V těchto bodech je výpočet gradientu intenzit nenulový [8]. Vývoj směru a velikosti pohybu vektorového pole při přesouvání předmětu ve snímané scéně viz Obrázek 5.2.

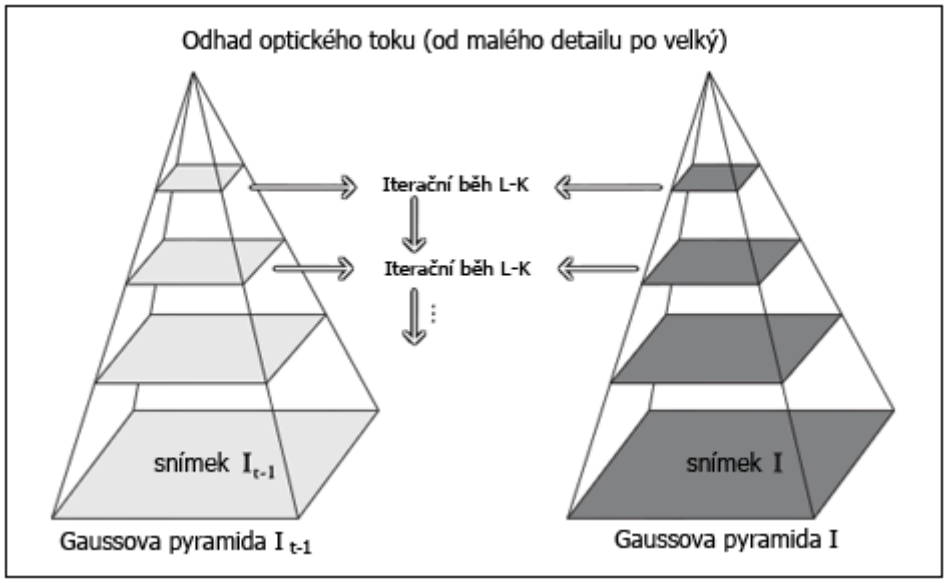
K problémům při výpočtu optického toku dochází z důvodu neschopnosti snímače zachytit tolik prostorových rozměrů, jež obsahuje snímaná scéna. Důsledkem je, že body pohybující se směrem k pozorovateli, zůstávají ve scéně skryty. Ztráta rozměru také způsobí rozbíhavost vektoru při přibližování objektu.

Výpočet optického toku je prováděn pomocí funkce `cvCalcOpticalFlowPyrLk()`, která implementuje pyramidový výpočet toku metodou Lucas-Kanade. Blíže k této funkci viz [4]. Výstupem funkce je pole souřadnic nově nalezených bodů. Tyto body jsou hledány podle podobnosti s body z předchozího snímku. Pole těchto souřadnic v dalším kroku tvoří vstup funkce trasující významné body.

Obrázek 5.2 Optický tok – velikost a směr (pohyb zleva doprava)

Pyramidová metoda (Lucas-Kanade) je založena na výpočtu optického toku pouze v malém okolí bodu zájmu. Nevýhodou však je omezení právě malého okolí, protože při větších pohybech se bod může dostat mimo tuto oblast, což znamená, že ho algoritmus nebude schopen zpět vyhledat.

To vedlo k přetvoření algoritmu na tzv. „pyramidový“ výpočet optického toku (viz Obrázek 5.3), který okolí bodu sleduje od nejmenšího detailu postupně k největšímu. Každý pixel v obraze na úrovni $k+1$ je svázán nějakým funkčním vztahem s $n \times n$ pixely na úrovni k. Takto reprezentovaná vstupní data lze pak procházet shora dolů s postupným zjemňováním výsledku. Algoritmus tedy umožňuje vyhledání bodu v následujícím snímku i při rozsáhlejších pohybech. Metoda se ukázala jako velice důležitou a efektivní technikou při výpočtu optického toku [4].



Obrázek 5.3 Gaussova pyramida výpočtu optického toku (převzato z [4])

5.3 MEAN-SHIFT

Mean-shift segmentace shlukuje body obrazu (pixely) na základě podobnosti jejich vzhledu a blízkosti jejich pozice pomocí konvergence do lokálních maxim spojeného souřadnicového a intenzitního prostoru. Tento algoritmus je velice robustní, protože ignoruje data ležící daleko od počítaného maxima [9]. Hledání maxima je aplikováno pouze na body ležící uvnitř ohraničené oblasti.

5.3.1 Shlukování bodů stejné intenzity

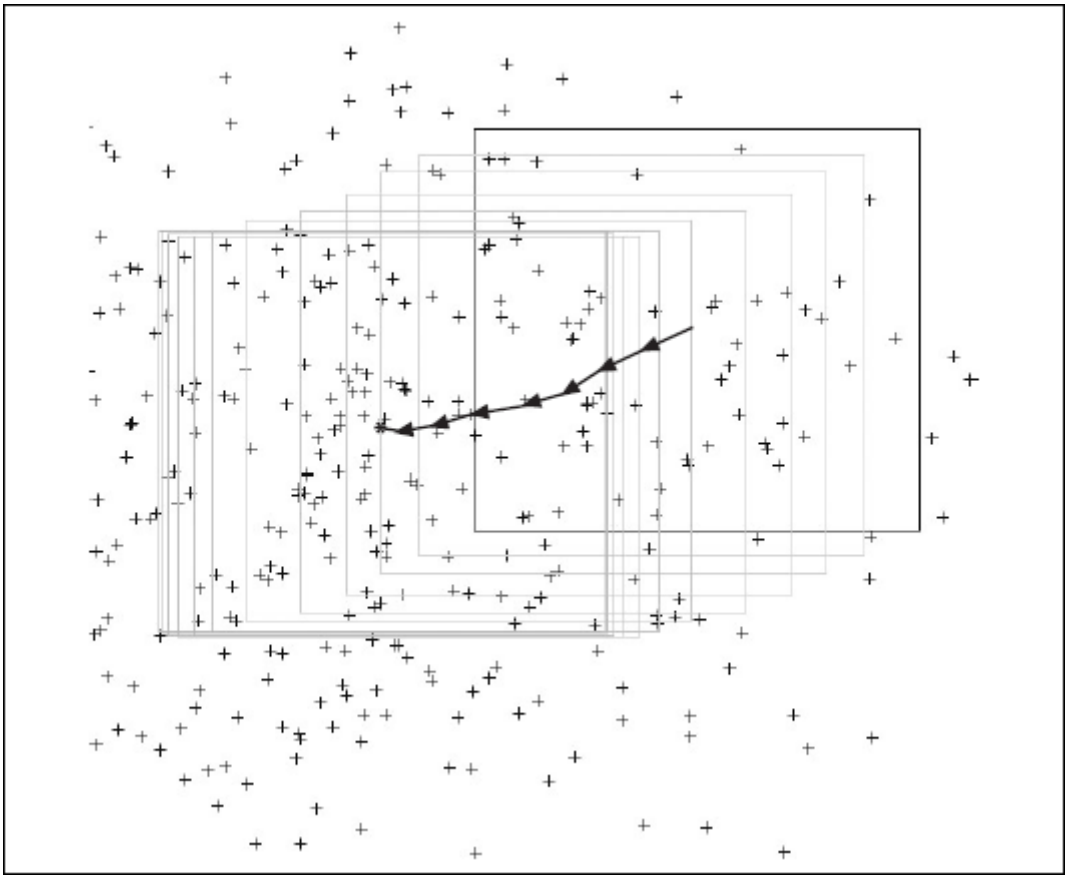
Každý bod *[u,v]* šedotónového obrazu má danou intenzitu jasu *i*. V třírozměrném prostoru jsou body popsány jasovou maticí *[u,v,i]*, kde první dvě souřadnice udávají pozici bodu a třetí určuje právě jeho intenzitu. Blízké pixely náležející nějakému objektu budou mít podobnou intenzitu a v prostoru budou poté vytvářet shluky. Naopak je to s pixely, které spolu sousedí a jejichž hodnota jasu je rozdílná. U těchto bodů obrazu se předpokládá, že náleží různým objektům a v prostoru *[u,v,i]* budou vzdálené. Tímto je úloha segmentace převáděna na shlukování bodů podobných intenzit.

5.4 MEAN-SHIFT TRACKER

Algoritmus mean-shift lze využít i pro řešení úlohy sledování objektu. Jak již bylo řečeno, vychází se z předpokladu, že body patřící objektu mají podobnou hodnotu jasu. Běh algoritmu je poté následující:

1. Výběr oblasti zájmu:
   - Pozice okna
   - Typ (jednotný, polynomiální, exponenciální nebo Gaussovský)
   - Tvar (symetrický, asymetrický, možnost rotace, kulatý nebo obdélníkový)
   - Velikost
2. Výpočet těžiště shluků uvnitř výběru
3. Posun okna do těžiště
4. Návrat ke kroku 2., dokud se okno nepřestane hýbat (nalezeno těžiště shluků)

Následující Obrázek 5.4 znázorňuje běh algoritmu na 2D distribuci dat se startem v ohraničené čtvercové oblasti. Šipky naznačují směr pohybu a zároveň tedy i konvergenci metody. Konvergence není ovlivněna body, které leží mimo znázorněnou oblast. Knihovna OpenCV implementuje algoritmus Mean-Shift v jedné ze svých funkcí hojně využívaných pro sledování. Předpokládá se, že vstupem funkce bude obraz reprezentující hustotu distribuce dat [4].

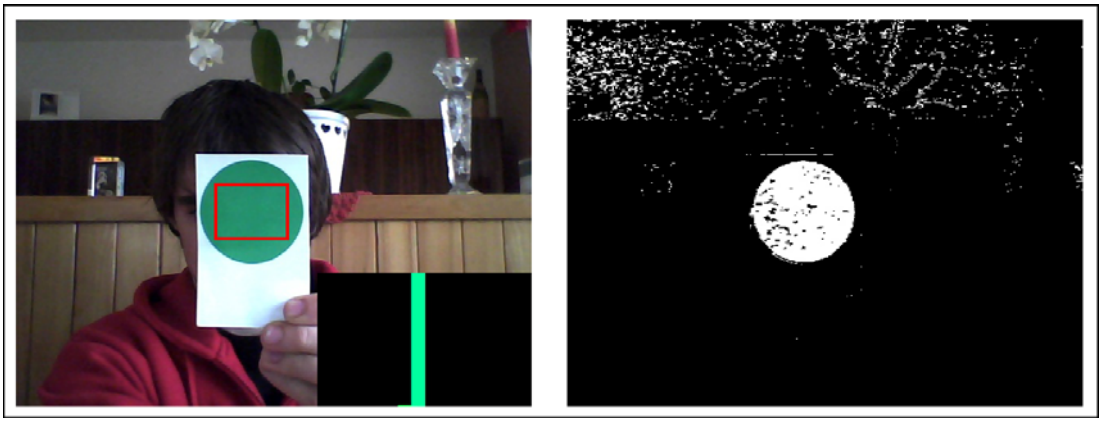


Obrázek 5.4 Mean-Shift – konvergence metody (převzato z [4])

Nastává však problém, jak vytvořit takovouto distribuci dat, kde budou shluky tvořit právě ty intenzity, patřící objektu zájmu. K tomu slouží funkce `cvCalcBackProjection()`, která z histogramu sledovaného objektu vytvoří tzv. zadní projekci (z anglického „Back Projection“). Výstup této funkce je použit jako vstup sledovacího algoritmu `cvMeanShift()`.

5.4.1 Zadní projekce „Back Projection“

Jak již bylo řečeno k vytvoření tzv. zadní projekce je zapotřebí znát histogram sledovaného objektu. Pokud je histogram například vytvořen z odstínu lidské kůže, metoda zadní projekce nalezne oblasti odpovídající právě histogramu tohoto odstínu. Obraz zadní projekce viz Obrázek 5.5. Histogram objektu byl vytvořen z červeně ohraničené oblasti.



Obrázek 5.5 Objekt a jeho histogram (vlevo), „Back Projection“ (vpravo)

5.5 CAM-SHIFT TRACKER

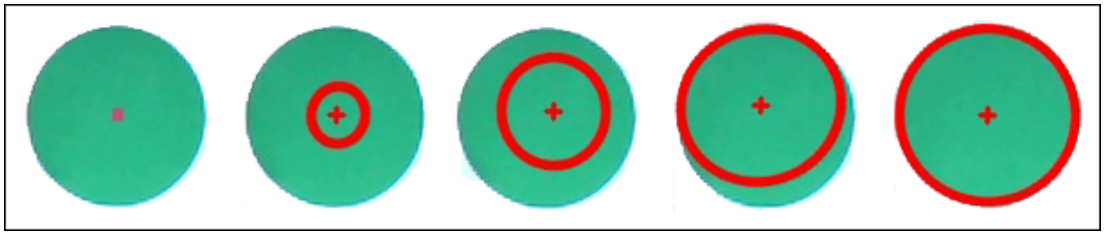
Tento algoritmus se výpočtem od Mean-Shiftu nijak neliší. Je však schopen měnit rozměry okna, které vymezuje oblast výpočtů. Pokud tedy sledujeme objekt, který se ve scéně přibližuje a oddaluje, algoritmus Camshift přizpůsobuje rozměr oblasti podle velikosti cíle. Funkce `cvCamShift()`, která tento algoritmus implementuje, je pouze vylepšené volání funkce `cvMeanShift()`.

```
int cvCamShift(
const CvArr* prob_image,
CvRect window,
CvTermCriteria criteria,
CvConnectedComp* comp,
CvBox2D* track_box = NULL
);
```

Prvním vstupním parametrem funkce je obraz `prob_image`. Parametr `prob_image` reprezentuje obraz zadní projekce vytvořený funkcí `cvCalcBackProjection()` (viz Obrázek 6.5). Další parametr `window` je počáteční lokace a velikost okna vymezující oblast výpočtů. Velikost tohoto okna se mění v průběhu vykonávání programu. Vstupní parametr kriterium určuje maximální množství iterací metody Camshift a především minimální posun okna v místě konvergence algoritmu (nastaveno defaultně). `Track_box` parametr, pokud je přítomen, obsahuje velikost nově zvětšeného resp. zmenšeného okna. Po ukončení funkce musí dojít ke zkopírování obsahu komponenty typu `CvBox2D` do proměnné

`window`, typu `CvRect`. Při dalším volání funkce nese proměnná `window` nové souřadnice a velikost okna vymezující oblast výpočtu [4].

Obrázek 5.6 ukazuje konvergenci algoritmu se zvětšující se oblastí zájmu. Pro zjednodušení byl vybrán homogenní objekt jedné barvy. Obrázky na sekvenci snímků byly pořízeny po každém programovém volání funkce `cvCamShift()`. S časem se tedy oblast zvětšuje, až do doby kdy dosáhne plného pokrytí objektu.



Obrázek 5.6 Camshift tracker, přizpůsobení oblasti

6. HLAVNÍ PROGRAM SLEDOVÁNÍ

V této části bude popsán celý výpočetní algoritmus, který se skládá z několika částí a funkcí. Jak již bylo řečeno, sledování pohyblivého cíle je řešeno pomocí dvou metod. Mezi těmito metodami výpočtu se uživatel může libovolně přepínat a měnit některé parametry. Z hlavního programu jsou volány funkce popsány v kapitole 6.2 Popis použitých funkcí.

První metoda využívá funkci `cvCamShift()` (viz kapitola 6.3 Metoda sledování využívající Camshift), jejímž vstupem je `prob_image` (pravděpodobný výskyt objektu) vytvořený z histogramu cíle. Druhá metoda ke sledování využívá nalezení významných bodů (Harris, Shi-Tomasi) a jejich následné sledování v pohyblivé scéně pomocí techniky optického toku (viz kapitola 6.4 Metoda sledování využívající významné body.).

Na začátku programu je řešen import halvičkových souborů, potřebných pro správný chod a volání funkcí. Importují se hlavičky souborů OpenCV, CSerialPort a standardní knihovny jazyka C/C++. Název hlavičkového souboru za direktivou `#include` je nahrazen kódem příslušné knihovny. Tato část kódu je určena pro lexikální preprocesor kompileru jazyka C.

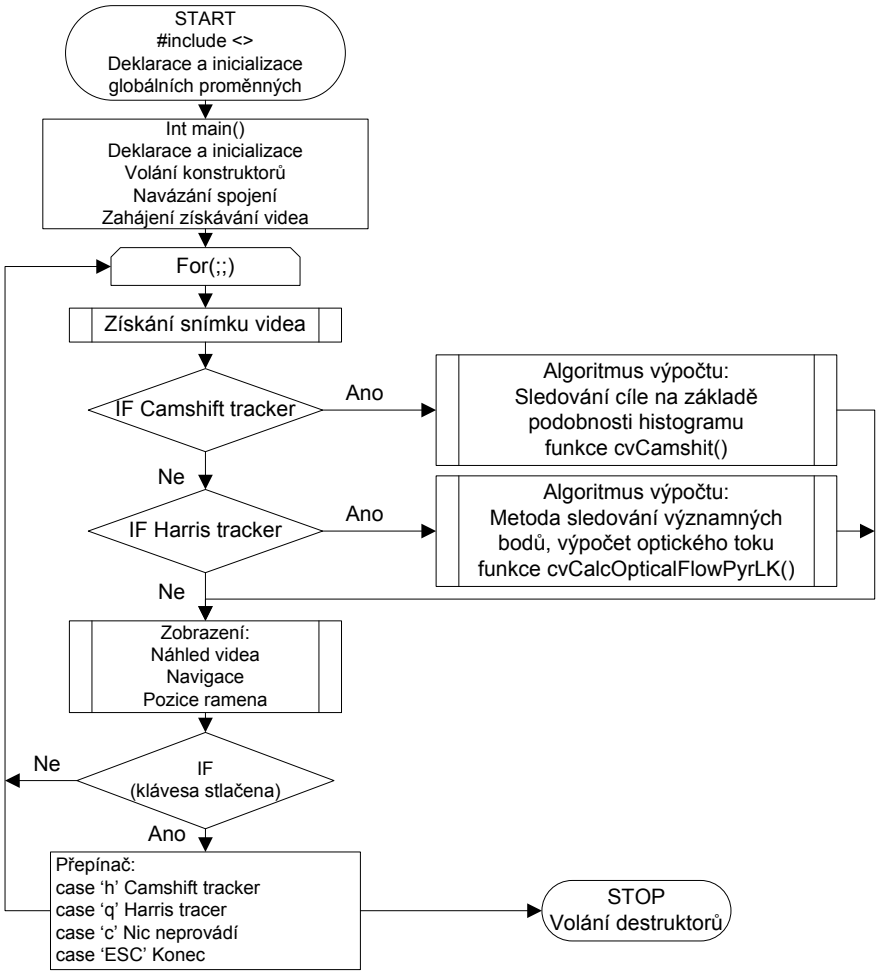
```
#define _AFXDLL
#ifndef _EiC

#include "stdafx.h"
//knihovna zajistujici komunikaci RS-232
#include "SerialPort.h"
//knihovny OpenCV
#include <cv.h>          //hlavni funkce knihovny OpenCV
#include <highgui.h>     //graficke rozhrani OpenCV
#include <cxcore.h>      //datove struktury, makra
//standartni knihovny
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <iostream>

#endif
```

Následují deklarace a inicializace globálních proměnných, využívaných funkcemi vně hlavního programu main().

Po deklaraci a inicializaci všech proměnných, úspěšném navázání spojení po sériové lince a vytvoření vstupního streamu videa z příslušné web kamery, program vstoupí do nekonečné smyčky. V té zůstává po dobu, dokud není přerušena uživatelem.

6.1 VÝVOJOVÝ DIAGRAM, VĚTVENÍ PROGRAMU

Za běhu programu lze libovolně přepínat mezi metodami, kterými bude program provádět trasování a výpočet středu sledovaného objektu. To je řešeno pomocí přepínače na konci programu. Funkce switch() přepíná mezi volbami pouze tehdy, je-li stlačena příslušná ovládací klávesa. Větvení programu je ukázáno na následujícím obrázku.



Obrázek 6.1 Vývojový diagram, větvení programu

Z vývojového diagramu programu je vidět, že se algoritmy výpočtu provádějí za předpokladu, je-li splněna podmínka pro větvení. Získávání video snímku a zobrazení náhledu s vykreslením navigace, probíhá v každém kroku programu.

Přehled událostí vzniklých při stisku klávesy je popsán v následující tabulce (viz Tabulka 6.1).

Klávesa	Událost
"h"	- Nastaven příznak Camshift tracker - V prvním kroku dojde k odečtení histogramu objektu - Sledování cíle na základě podobnosti histogramu, vytvoření backproject obrazu, který je vstupem funkce camshift()
"q"	- Nastaven příznak Harris tracker - V prvním kroku vyhledání významných bodů - Sledování cíle metodou významných bodů, výpočet optického toku, body využity k sledování objektu funkcí meanshift()
"c"	Objekt nesledován, vymazání významných bodu a histogramu
"r"	Start/Stop nahrávání videa
"ESC"	EXIT programu, volání destruktorů

Tabulka 6.1 Ovládání programu, přehled událostí

6.2 POPIS POUŽITÝCH FUNKCÍ

V následujících podkapitolách jsou popsány vlastnosti jednotlivých funkcí volaných z hlavní části programu. Ve funkcích jsou řešeny programové rutiny ovládání motoru, výpočet kontrolního součtu, výběr objektu a výpočet úhlu, o který se motory posunují.

6.2.1 Výběr počáteční oblasti

Tato funkce `on_mouse()` je volána v případě, že dojde ke zmačknutí tlačítka myši uvnitř okna s náhledem videa. Je tedy nutné, aby byl nastaven callback ze strany periferie (myš). Uvnitř této funkce dojde k výběru počáteční oblasti s výskytem objektu, určeného pro sledování. Oblast je vybrána táhnutím myši. Veškeré souřadnice a velikost vymezené oblasti jsou ukládány do proměnné typu struktura `CvRect`. Struktura obsahuje souřadnice počátku, výšku a šířku vybrané oblasti

obdélníka. Tato oblast slouží jako počáteční vstup pro funkce, které mají za úkol vyhledat hledaný cíl.

6.2.2 Převod z barevného modelu HSV do RGB

Histogram je počítán ze složky odstínu modelu HSV označeného objektu, pomocí funkce `on_mouse()`. Pro zobrazení histogramu v barevném modelu RGB, bylo zapotřebí provést výpočet, který složku odstínu (H) přepočítá. To provádí popisovaná funkce `hsv2rgb()`, jejíž vstupem je hodnota odstínu H a výstupem struktura `CvScalar` obsahující hodnoty zastoupení jednotlivých barev v modelu BGR. Funkce byla součástí vzorového příkladu Camshiftdemo, který je distribuován v balíčku knihovny OpenCV.

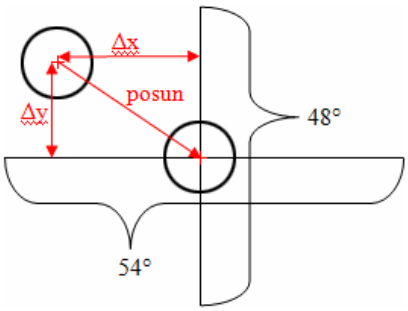
6.2.3 Výpočet kontrolního součtu

Jak již název napovídá, jedná se o funkci počítající kontrolní součet instrukčního paketu, který je jejím vstupem. Výstup je řešen pomocí pointru na paměťové místo proměnné typu `int check_sum`. Hlavička funkce vypadá následovně.

```
void Checksum(unsigned char *paket,int velikost,int *check_sum);
```

6.2.4 Výpočet úhlů natočení

Vstupem funkce `uhly_natoceni()` jsou souřadnice středu nalezeného objektu. Nejprve bylo nutné experimentálně vypočíst zobrazovací úhly kamery ve směru osy x i y. Tyto úhly byly potom dány do poměru se vzdáleností cíle od středu obrazu v pixelech. Výstupem funkce jsou úhly, o které se pohony posunou ve směru osy x a y tak, aby byl sledovaný cíl uprostřed scény.



Obrázek 6.2 Zjištění vzdálenosti objektu od středu obrazu, rozlišení 640x480

6.2.5 Funkce median

Tato funkce počítá střed shluků nalezených bodů. Je volána pouze při provádění algoritmu sledování významných bodů nalezených v obraze. Metoda výpočtu je podrobně popsána v kapitole 6.4.4 Souřadnice středu a otočení pohonů. Vstupem funkce je pole hodnot se souřadnicemi významných bodů. Protože se některý bod může nacházet mimo objekt zájmu, není možné střed počítat z průměru hodnot. Lepším řešením je výpočet mediánu, který je výstupem funkce.

6.2.6 Funkce zajišťující komunikaci s pohony

Následující funkce obstarávají komunikaci a správný chod pohonů Dynamixel RX-64. Vstupem funkcí je vždy objekt typu `CSerialCom`, asociovaný s COM portem počítače. Tento objekt je vytvořen během inicializace proměnných, za použití již zmíněné knihovny CSerialCom v1.27 viz. 4.2 Knihovna CSerial Port.

6.2.6.1 Is_moving()

Motory RX-64 programátorovi nabízejí možnost, mít plnou kontrolu nad prováděným pohybem. Disponují tedy množstvím vnitřních registrů, ať už v paměti RAM nebo EEPROM. Z paměťového místa s adresou 0x2E v kontrolní tabulce pohonů lze vyčíst přítomnost pohybu v daný okamžik. To přineslo výhodu v podobě zrychlení výpočetního algoritmu a ošetření chyb vzniklých při komunikaci po sériové lince. Docházelo totiž ke kolizím, objevujících se při zápisu do kontrolní tabulky pohonů.

Volání funkce následuje po nalezení středu hledaného cíle v obraze a výstup ovlivní provádění pohybu. Pokud se motory hýbou, není nutné zpomalovat výpočetní algoritmus zapisováním instrukčního paketu na port v každém běhu programové smyčky.

Výstup funkce je typu BOOL. Přítomnost pohybu tedy nese hodnotu true, nepřítomnost hodnotu false. Instrukční paket vypadá následovně.

```
unsigned char paket1[]={255, 255, 1, 4, 2, 46, 1, 0};
```

Motor s ID 1 na obdržený instrukční paket odpoví status paketem, obsahující hodnotu paměťového místa s adresou 0x2E (Moving). Ve funkci se obdržený status

paket vyhodnotí a podle potřeb nastaví výstup. K čtení vstupního bufferu dochází okamžitě po zápisu na port.

```
port->WriteEx(paket1, sizeof(paket1));
port->Read(buffer1,sizeof(buffer1));
```

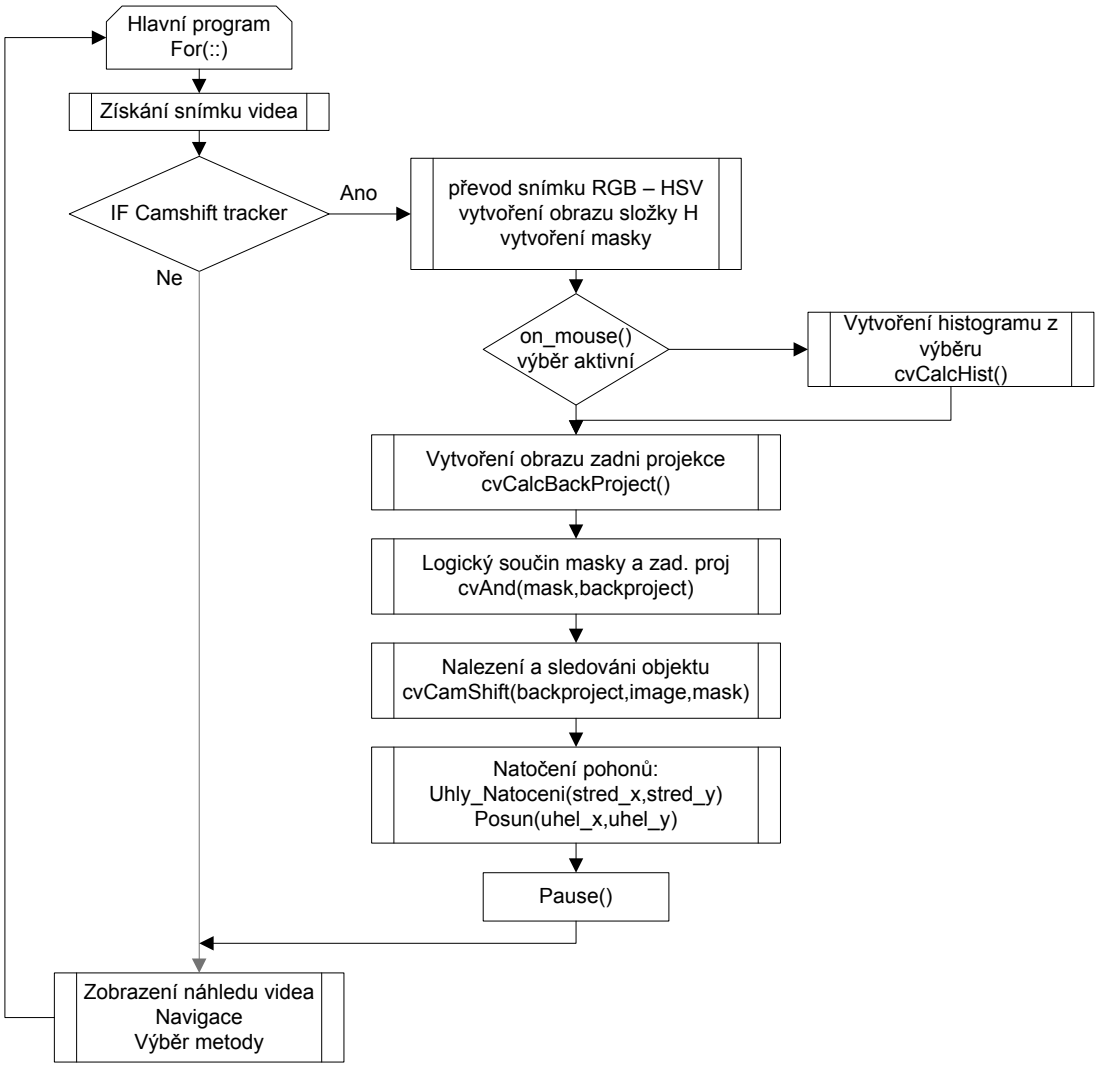
Vstupní buffer portu COM se uloží do proměnné typu `unsigned char buffer1`. Toto pole hodnot obsahuje status paket, jehož šestý byte obsahuje informaci o pohybu pohonů.

6.2.6.2 Posun()

Tato funkce obstarává pohyb pohonů Dynamixel. Zároveň zde dochází ke správnému přepočtu hodnot úhlu ve stupních, na low a high hodnoty o velikosti jeden byte, určené pro zápis do kontrolní tabulky pohonů. Vstupem jsou úhly natočení ve směru osy x a y. Po zkompletování se výsledný instrukční paket zapíše na port COM.

6.3 METODA SLEDOVÁNÍ VYUŽÍVAJÍCÍ CAMSHIFT

6.3.1 Vývojový diagram volaných funkcí



Obrázek 6.3 Vývojový diagram pro metodu Camshift

6.3.2 Vytvoření masky obrazu

Při každém běhu programu je pořízen snímek, z kterého je vytvořena maska. Maska je binární obraz určující, které oblasti jsou pro následné výpočty zajímavé. Jedná se o část předzpracování obrazových dat zvanou segmentace. Využita je v případech, kdy je možné při zpracovávaných algoritmech vynechat některé části obrazu, které pro programátorem navrhovanou aplikaci nesou jen méně významnou informaci.

Postup tvorby masky je následující. Nejprve se pořízený obraz převede do barevného modelu HSV. Z něj je vytvořena maska tak, že jsou ignorovány body, jejichž hodnoty neleží v uživatelem zadaném intervalu[1] složek S (S_{min};S_{max}) a V (V_{min};V_{max}). Body mající hodnoty S a V mimo zadaný interval jsou v masce reprezentovány hodnotami 0. Naopak body patřící do rozsahu sytosti a jasu mají hodnotu 1. Interval hodnot je volen tak, aby byly eliminovány ty body obrazu, které jsou buď velmi tmavé, nebo naopak velmi světlé. Tyto body jsou ve scéně spíše rušivým elementem a vyplívá z toho, že je není možné sledovat.

Binární obraz masky je výstupem funkce `cvInRangeS()`. Vstupem funkce jsou intervaly hodnot složek Hue, Saturation a Value, z kterých je zmíněný binární obraz vytvořen.

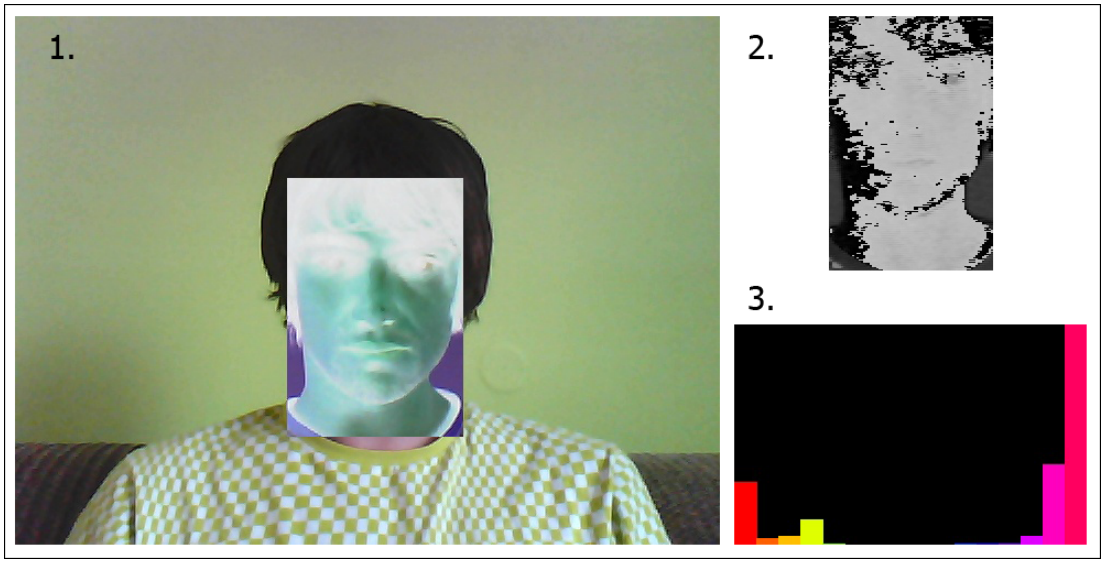
6.3.3 Vytvoření histogramu

Tato část programu se provede pouze jednou a to při výběru objektu. Histogram je počítán z barevného modelu HSV. K výpočtu je použita pouze složka odstínu. Z již převedeného snímku (HSV) se pomocí funkce `cvSplit()` oddělí pouze kanál odstínu (barevný tón Hue) a uloží se do pomocné proměnné typu `IplImage`.

Výpočet histogramu je prováděn pouze z té časti obrazu, kde se nachází uživatelem vybraný objekt. Právě tato vymezená oblast zájmu je vstupem pro výpočet histogramu funkcí `cvCalcHist()`. Vstupem funkce je objekt zájmu a maska sledovaného obrazu. Histogram objektu je zobrazován spolu s náhledem zpracovávaného video streamu. Pro barevné zobrazení histogramu se hodnoty odstínu zastoupených v obraze přepočítávají z hodnot H na RGB. K tomu je volána funkce `hsv2rgb()` popsána v kapitole 6.2.2. Na následujícím obrázku je vytvořen barevný histogram z výběru. Histogram udává, kolik pixelů ve výběru obsahuje danou barvu (odstín).

---

[1] Interval hodnot (S a V) je možno měnit za běhu programu pomocí sliderů umístěných ve vrchní částí hlavního okna s náhledem.



Obrázek 6.4 Histogram vytvořený z výběru
(1. originální obraz, 2. výběr HSV, 3. barevný histogram)

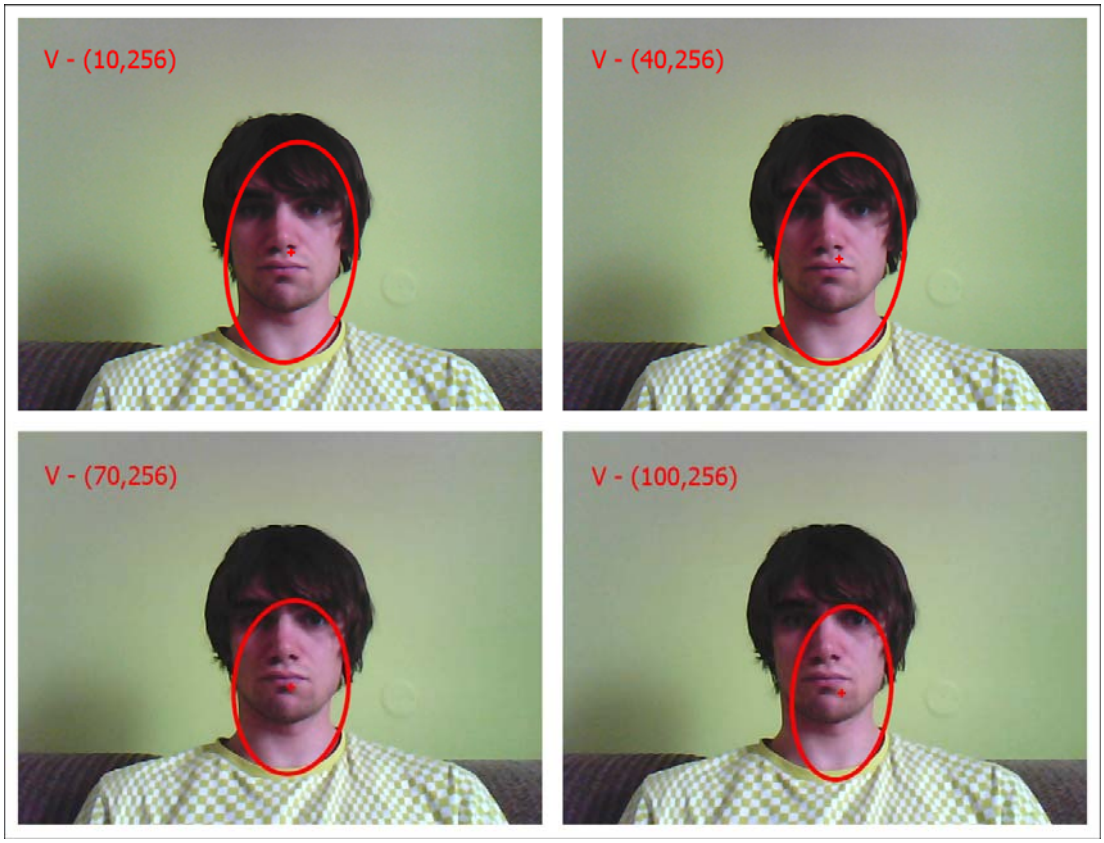
6.3.4 Backprojection a CamShift

Histogram je dále využit k vytvoření zpětné projekce (viz. kapitola 5.4.1 Zadní projekce „Back Projection“) pomocí funkce `cvCalcBackproject()`. Vstupem funkce jsou dva parametry a to obraz obsahující pouze složku odstínu snímané scény a zjištěný histogram objektu.

Obraz zpětné projekce je následně ještě vynásoben s obrazem masky pomocí funkce `cvAnd()`, která realizuje logický součin dvou snímků. Výsledný obraz zadní projekce a masky (viz Obrázek 6.5). Maska tedy propustí (binární 1) pouze ty body, jejichž hodnota spadá do pevně daného intervalu hodnot složky odstínu (0;128) a sytosti (30;256). Interval pro složku jasu byl nastavován tak, jak ukazuje obrázek. Je zřejmé, že maska plní funkci jakéhosi filtru, který odstraní nežádoucí šum z originálního obrazu zadní projekce vytvořeného z histogramu.

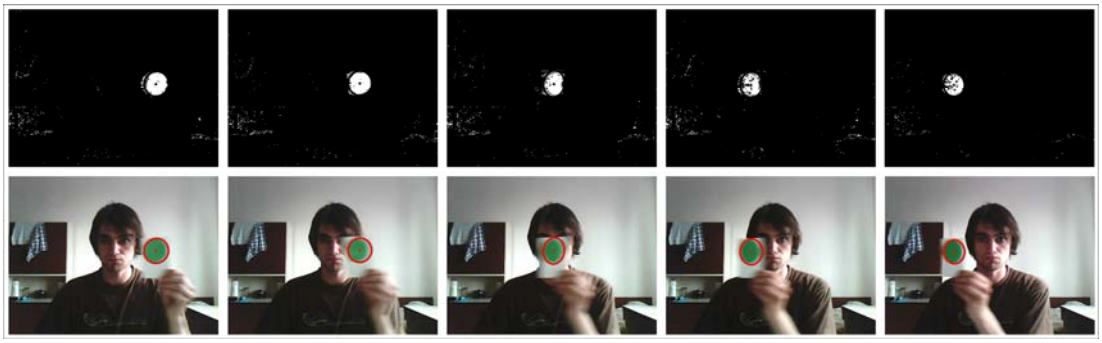
Obrázek 6.5 Backprojection pro různou propustnost masky

Takto vytvořený obraz je již připraven ke zpracování funkcí `cvCamShift()`. Výstupy funkce `cvCamShift()`, pro různé vstupní obrazy zadní projekce podle obrázku výše (Obrázek 6.5), jsou vidět na následujícím snímku. Jak je vidět, oblast s pravděpodobným výskytem objektu se mění s nastavením propustnosti masky.



Obrázek 6.6 Výstup funkce cvCamShift()

Pro real-time vykreslení oblasti nálezu objektu do náhledu videa byla využita funkce OpenCV `cvEllipseBox()`, která pracuje s hodnotami komponenty CvBox2D. Tato komponenta je výstupem funkce `cvCamshift()` a obsahuje souřadnice středu, velikost a natočení nalezené oblasti.



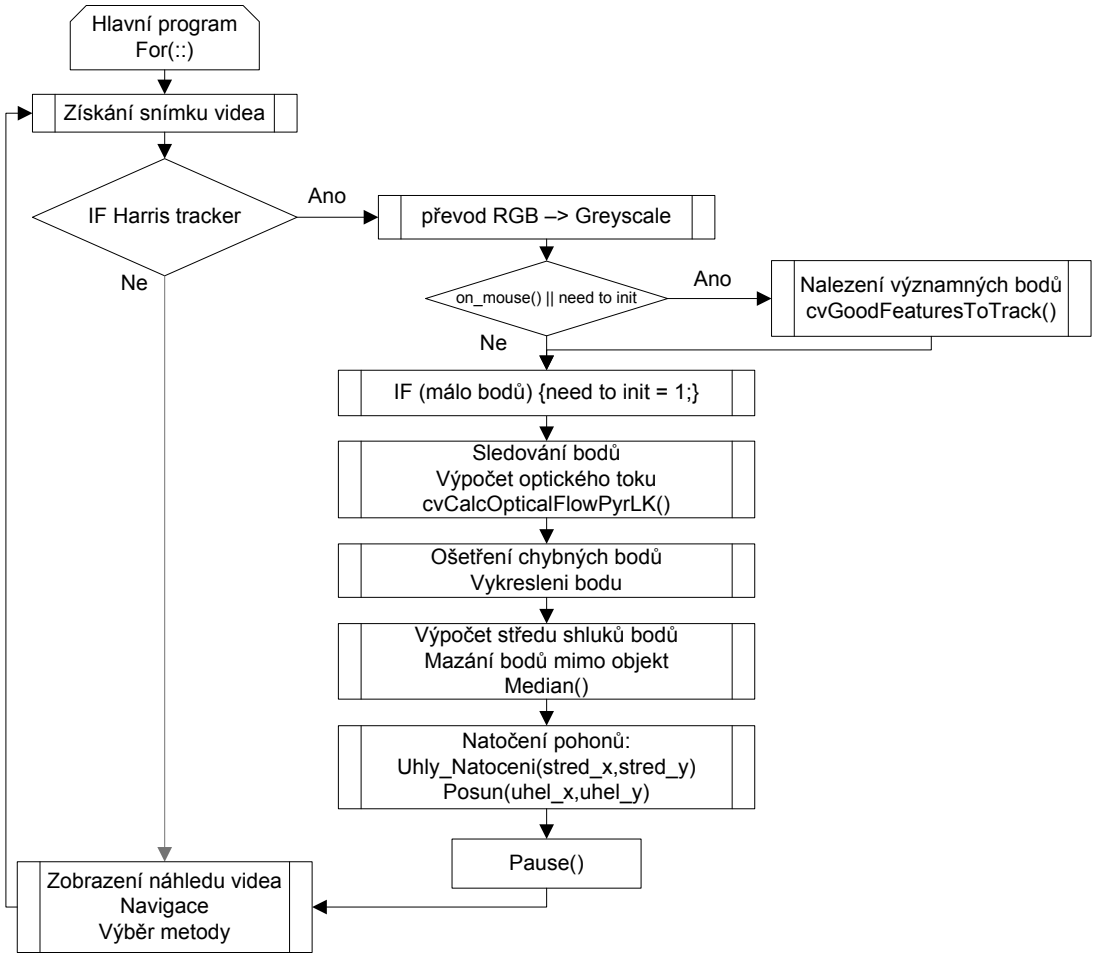
Obrázek 6.7 Sekvence snímků zpracovaná metodou Camshift

6.3.5 Souřadnice středu a otočení pohonů

Po provedení funkce `cvCamShift()` následuje volání rutiny `Uhly_natoceni()` (viz. 6.2.4 Výpočet úhlů natočení). Výstupy funkce (uhel_x a uhel_y) jsou použity pro volání rutiny `Pohyb()`, která již zajistí otočení pohonu tak, aby byl sledovaný cíl uprostřed scény. Funkce `Pohyb()` je volána pouze v případě, že se motory nehýbou. Přítomnost pohybu je kontrolována voláním funkce `Is_moving()`. Tato funkce vrací log. 0 resp. log. 1 pokud motor provádí resp. neprovádí pohyb.

6.4 METODA SLEDOVÁNÍ VYUŽÍVAJÍCÍ VÝZNAMNÉ BODY

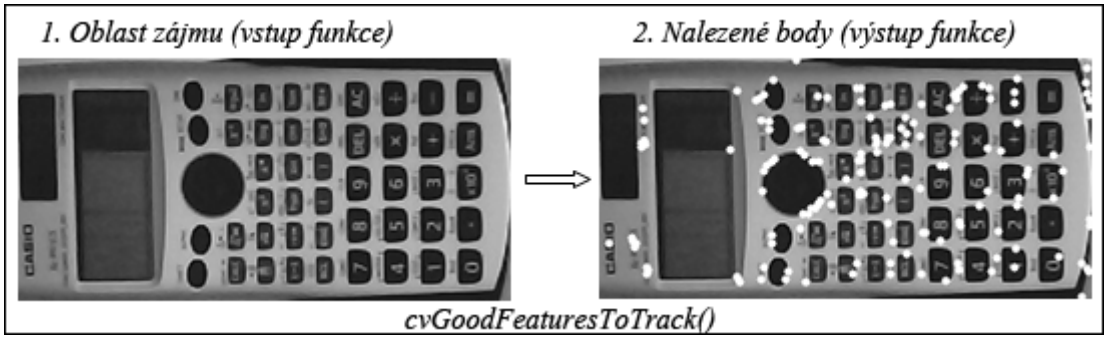
6.4.1 Vývojový diagram volaných funkcí



Obrázek 6.8 Vývojový diagram pro metodu Významné body

6.4.2 Nalezení významných bodů

Tato část programu se provede pouze jednou a to při volání callback funkce `on_mouse()`. Uživatel tedy specifikuje oblast výskytu objektu, v níž jsou významné body hledány. Algoritmus starající se o nalezení významných bodů je implementován ve funkci `cvGoodFeaturesToTrack()`. Funkce pracuje s černobílým obrazem. Blíže k metodě významných bodů v kapitole 5.1 Nalezení významných bodů.



Obrázek 6.9 Oblast zájmu a nalezené významné body

Funkce vrací souřadnice nalezených bodů v poli struktur `CvPoint2D32f`. Protože byly body hledány v oblasti zájmu vytvořené funkcí `cvSetImageROI()` (Rectangle of interest), souřadnice těchto bodů jsou v obraze posunuty o velikost počáteční pozice této oblasti. Ke všem bodům je tedy přičtena hodnota počátku oblasti. Nalezené body v oblasti zájmu znázorňuje Obrázek 6.9. Množství nalezených bodů bylo definováno na 150.

6.4.3 Výpočet optického toku

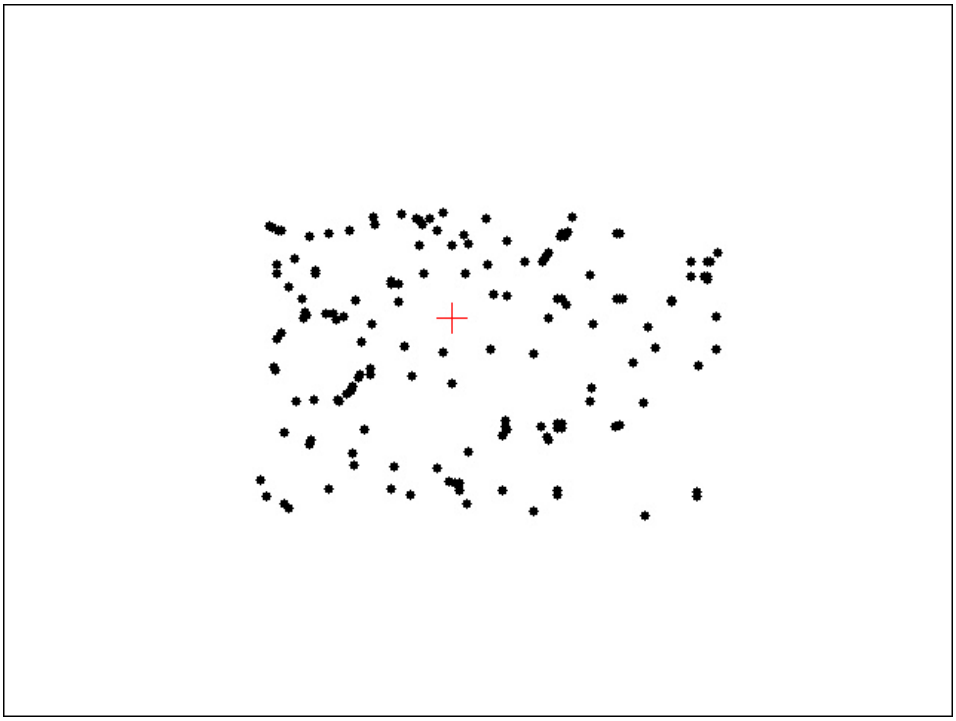
Výše nalezené body jsou vstupem funkce `cvCalcOpticalFlowPyrLK()`, která realizuje výpočet optického toku pyramidovou metodou Lukas-Kanade (viz kapitola 5.2 Optický tok). Výstupem funkce jsou nově nalezené body, které v co největší míře odpovídají významným bodům.

6.4.4 Souřadnice středu a otočení pohonů

Významné body v obraze tvoří shluky. Protože se ve snímku nacházejí pouze body patřící objektu, střed je možné vypočítat jako medián tohoto shluku bodů. Výpočet

mediánu provádí funkce `Median()`. Výstupem této funkce je medián z hodnot zvlášť pro souřadnice x a y. Pokud by body byly rovnoměrně rozprostřeny po celém objektu, výpočet středu pomocí mediánu by byl velice přesný. V navrhované aplikaci však nejde o nalezení absolutního středu objektu, avšak o jeho sledování. Proto se na přesnost neklade příliš velký důraz.

Výpočet mediánu byl zvolen z důvodu ztráty bodů v průběhu pohybu, které se ustálí v nové pozici mimo sledovaný objekt. Tyto body by pak v případě průměrování mohly dostat střed mimo cíl. Na následujícím obrázku je znázorněna distribuce dat v podobě nalezených bodů a střed tohoto shluku (viz Obrázek 6.10).



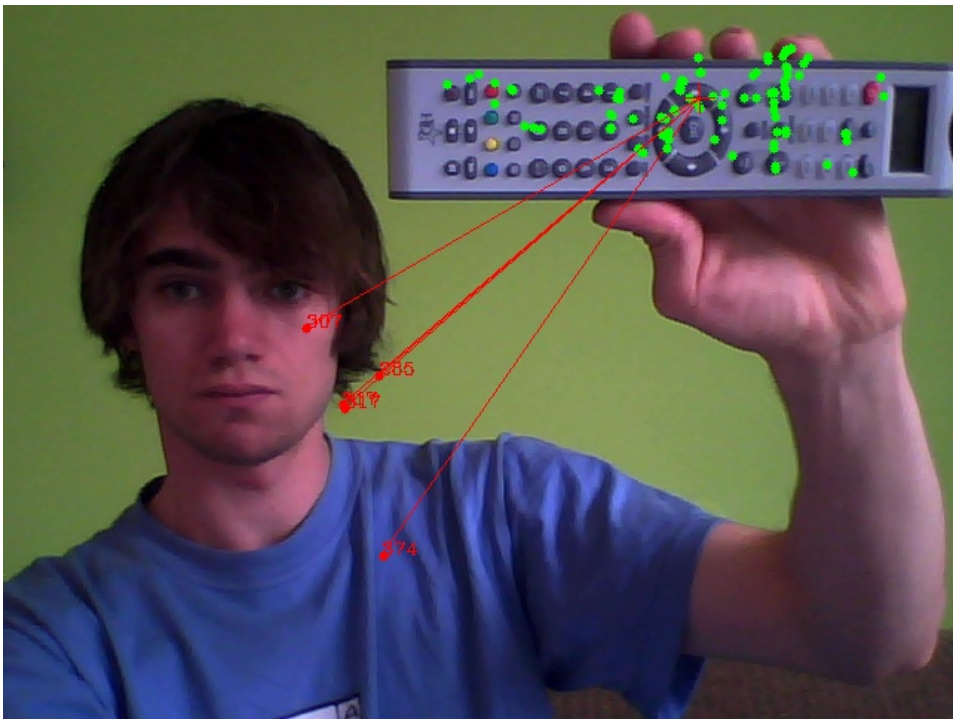
Obrázek 6.10 Shluk bodů a nalezený střed

Jakmile je střed shluků bodů spočítán, hodnoty se použijí k posunu pohonů viz kapitola 6.3.5 Souřadnice středu a otočení pohonů.

6.4.5 Ošetření bodů mimo objekt

Jak již bylo řečeno, některé body nacházející se na okrajích (hranách) sledovaného cíle se mohou ustálit v nové pozici mimo objekt. Tento problém byl vyřešen počítáním vzdálenosti každého bodu od středu shluku bodů. Body, jejichž vzdálenost je k × větší než medián vzdálenosti všech bodů, jsou smazány. Konstanta k byla

volena jako pětinásobek hodnoty mediánu vzdálenosti metodou pokus-omyl. Na následujícím obrázku jsou znázorněny ztracené body a sledovaný objekt (viz Obrázek 6.11).



Obrázek 6.11 Ztracené body a jejich vzdálenost od středu

6.5 SROVNÁNÍ POUŽITÝCH METOD SLEDOVÁNÍ

Metody užité ke sledování mají své výhody a nevýhody. Snaha vytvořit program, který by byl schopen se naučit jakýkoliv objekt, vyžadovala užití více metod sledování.

První metoda využívající histogram cíle, který je vstupem funkce Camshift se ukázala jako velice spolehlivá a rychlá. Tento algoritmus se však hodí ke sledování objektů, jejichž povrch je homogenní. Znamená to tedy, že je tvořen z podobných odstínů barev a zároveň se tyto odstíny ve sledované scéně vyskytují jen velmi málo. Protože je histogram počítán pouze ze složky odstínu barevného modelu HSV, není možno sledovat předměty, které jsou příliš světlé nebo tmavé. Tato metoda sledování ukazovala velmi dobré výsledky při sledování obličeje člověka nebo objektu, jehož barva byla zvolená tak, aby se co nejméně vyskytovala ve zkoumaném prostředí.

Algoritmus dokázal dobře pracovat i na větší vzdálenosti při rozlišení kamery 640x480. Naopak horší výsledky byly spojeny se zpracováním obrazu ve špatných světelných podmínkách. Při sledování objektu ve ztemnělém prostředí docházelo k vzniku velkého šumu z důvodu neschopnosti kamery věrně zobrazit barvy vyskytující se v prostředí. Algoritmus tedy selhával.

Výpočetní algoritmus založen na metodě sledování významných bodů vykazoval dobré výsledky při trasování těles, jejichž povrch byl rozmanitý na barvy a obsahoval velké množství hran. Problémy nastaly v případě rychlých změn pozice sledovaného objektu. Docházelo ke ztrátě významných bodů, tím pádem i k postupné ztrátě pozorovaného objektu. U tohoto algoritmu nezáleží na barevných odstínech obsažených ve scéně. Vykazoval lepší výsledky i ve zhoršených světelných podmínkách. Tato metoda není vhodná pro sledování objektů, jejichž poloha se rychle mění. Ve větších vzdálenostech objektu od kamery algoritmus selhával z důvodu ztráty informace způsobené menším rozlišením kamery.

Kombinací těchto dvou metod by se dalo dosáhnout větší spolehlivost algoritmu a možnost sledování libovolně složitého objektu.

6.6 RYCHLOST ALGORITMU

Rychlost zpracování pořízených obrazových dat vytvořeným programem byla měřena na notebooku ASUS s procesorem Intel Core2Duo 1.66 GHz, 2GB RAM, grafická karta NVIDIA GeForce 8400 M. Sledování cíle bylo prováděno po dobu 60sec. Z množství pořízených snímku za tuto dobu byl stanoven počet zpracovaných snímků za sekundu (viz Tabulka 6.2).

		počet snímků za sec [fps]	čas zpracování jednoho snímku [ms]
Histogram tracker	s pohony	26	38,5
	bez pohonů	29	34,5
Harris tracker	s pohony	23	43,5
	bez pohonů	27	37,0

Tabulka 6.2 Počet zpracovaných snímků za sec [fps]

ZÁVĚR

Část této bakalářské práce se týká vhodného nastavení parametrů objektivu, jako je clona, zoom a fokus. Pro ovládání tohoto objektivu byl naprogramován algoritmus, který zajistí automatické zaostření sledované scény tak, jak ho známe z běžných digitálních fotoaparátů. Bohužel nebylo možné objektiv použít i při zpracování druhého úkolu sledování pohyblivého cíle z důvodu, že se ohnisko objektivu nedalo vhodně přizpůsobit velikosti snímacího čipu kamery. Jedná se totiž o objektiv, který je určen pro digitální zrcadlovku s větším snímacím čipem než má použitá kamera. Objektiv tedy zabírá pouze část snímané scény.

Pro splnění druhého úkolu byl naprogramován řídící a výpočetní algoritmus využívající funkcí knihovny OpenCV. Tento algoritmus ovládá rameno sestavené ze dvou servo motorů na jejichž konci je připevněna snímací kamera. Pro rozpoznání a sledování cíle bylo užito dvou metod sledování. První z nich je metoda, která hledá objekt ve scéně pomocí podobnosti histogramu. Tato metoda sledování je vhodná pro objekty, jejichž povrch je tvořen podobnými odstíny jedné barvy. Druhá použitá metoda využívá ke sledování významné body, které byly nalezeny v oblasti výskytu objektu. Ze shluků významných bodů je vypočítán střed a ten je poté vstupem funkce, která ovládá robotické rameno s připevněnou kamerou. Program zpracovává video v reálném čase a průběh sledování je zobrazován jako náhled videa, v kterém se mimo jiné vykreslují i údaje o aktuálním natočení pohonů.

Pro efektivnější sledování by se v budoucnu mohl využít i objektiv, který byl použit při zpracování 1. úkolu. Použité metody by bylo vhodné zkombinovat tak, aby byl program schopen sledovat libovolně složitý cíl.

LITERATURA

[1] HLAVÁČ, Václav., ŠONKA, Milan. *Image processing, analysis and machine vision.*. Toronto: Thomson, 2008. 829 s. ISBN 978-0-495-08252-1

[2] WIKIPEDIA. *HSV, kategorie Barevné prostory* [online]. březen 2010, [cit. 2.4.2010] <http://cs.wikipedia.org/wiki/HSV>

[3] BOCKAERT Vincent, V. *The 123 of digital imaging interactive learning suite.* Vydavatel: 123di, 2009. ver 6.2, ISBN: 978-981-08-4271-0

[4] BRADSKI, Gary R., KAEHLER, Adrian. *Learning OpneCV*. Sebastopol: O'Reilly Media, c2008, 555 s. ISBN 978-0-596-51613-0

[5] *Manual CANON EF* [CD-ROM]. Brno: Camea spol. s r.o. 13.6.2007, verze 1.0, 14 s.

[6] ROBOTIS CO., LTD. *User's Manual Dynamixel RX-64* [online]. Nov. 2007, 49 s. [cit. 10.1.2010], ver. 1.10. <http://www.robotis.com/zbxe/?mid=software_en&category=7471&document_srl=5438>

[7] NAUGHTER, *Knihovna CSerialPort v1.27* [online], Last revision 4th of July 2008 [cit. 1.5.2010]. <http://www.naughter.com/serialport.html>

[8] HRNČÍŘ, Z. *Optický tok v obrazových datech živých buněk ,* Brno: Masyrykova univerzita. Fakulta informatiky, 2006. 56 s. Vedoucí diplomové práce Mgr. Vladimír Ulman. <http://is.muni.cz/th/60514/fi_m/dp.pdf>

[9] DOUBEK, P. (2007). *Mean-Shift segmentace* [online], Praha: České vysoké učení technické, Fakulta elektrotechniky, 29. října 2007, 4 s. [cit. 1.5.2010] <http://cmp.felk.cvut.cz/cmp/courses/ZS1/Cviceni/cv4/meanshift.pdf>

[10] HÝNA, P. *Detekce rohů v obraze.* Brno: Vysoké učení technické, fakulta informačních technologií, 2007. 54 s. Vedoucí bakalářské práce Ing. Michal Španěl, <http://www.fit.vutbr.cz/study/DP/rpfile.php?id=4930>

SEZNAM ZKRATEK A SYMBOLŮ

RGB – Red, Green, Blue

HSV – Hue, Saturation, Value

USB – Universal Serial Bus

CCD – Charge-Coupled Device

EF – Electro-Focus

API – Application Programming Interface

AVI – Audio Video Interleave

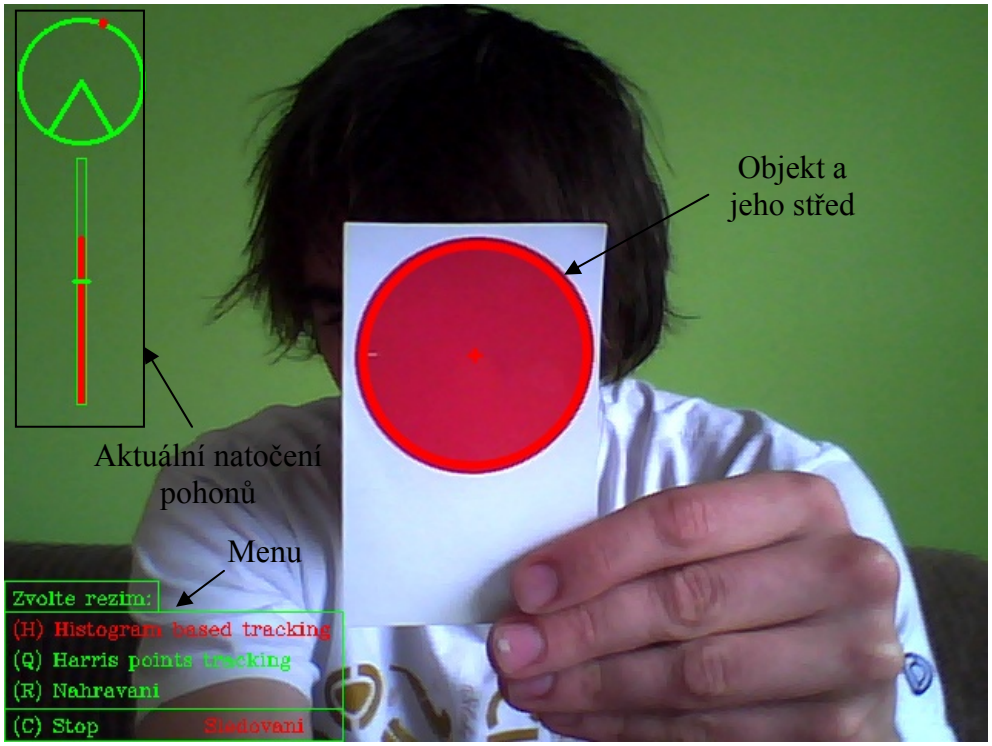
MPEG – Moving Picture Experts Group

C/C++

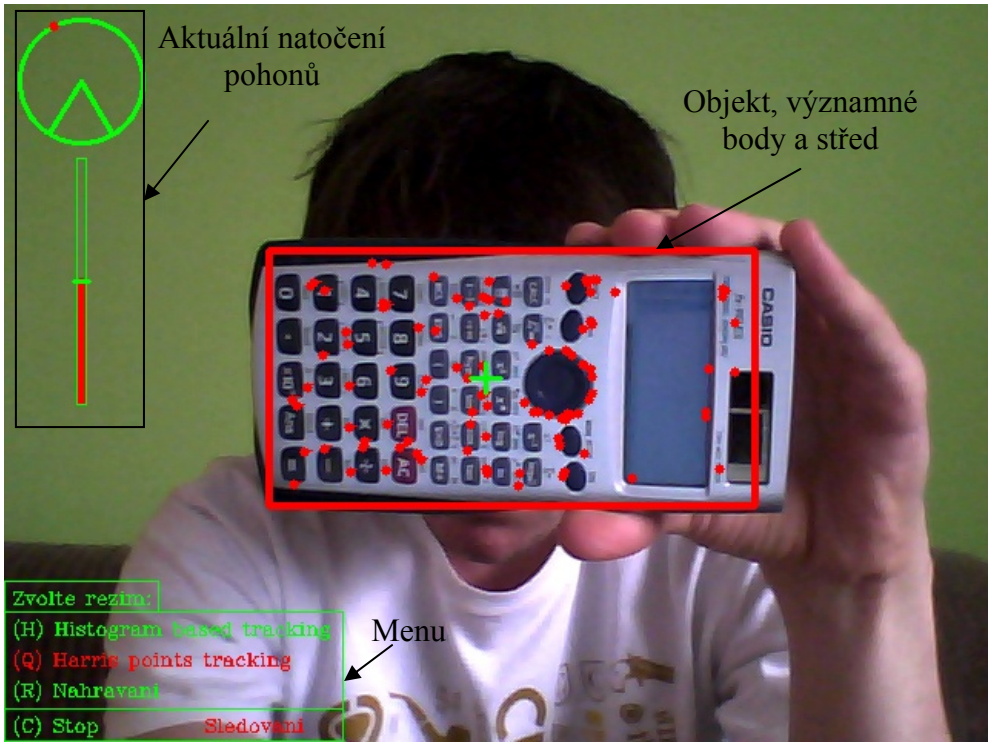
SEZNAM PŘÍLOH

A UKÁZKA ZOBRAZENÍ A SLEDOVÁNÍ

A.1 Sledování na základě podobnosti histogramu



A.2 Sledování pomocí významných bodů



B PŘÍLOHA CD-ROM

V kořenovém adresáři přílohy CD se nachází text bakalářské práce cerninBP.pdf. Vypracovaný projekt a zdrojové kódy obsahuje adresář Vypracování. Veškeré použité dokumenty jako jsou datasheety a návody se nachází v adresáři Manuály. Ve složce OpenCV je instalační balíček knihovny s manuálem pro instalaci v prostředí Windows 7. Dále se na CD v adresáři Video nachází i video ukázka procesu sledování.