VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# ROZŠÍŘENÍ KNIHOVNY PRO OVLÁDÁNÍ IP KAMER

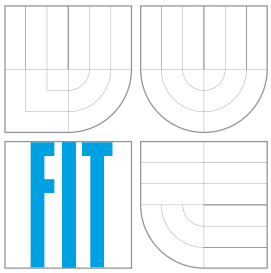
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL ŠEPTUN

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# ROZŠÍŘENÍ KNIHOVNY PRO OVLÁDÁNÍ IP KAMER

EXTENSION OF IP CAMERAS CONTROL LIBRARY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL ŠEPTUN
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADIM LUŽA
SUPERVISOR

BRNO 2013

# Abstrakt

Práce se zabývá návrhem a implementací rozšíření stávající knihovny pro ovládání IP kamer, hlavně podporou detekce pohybu, synchronizací pohybu více kamer a zaostřením pohledu na zadaný bod. Součástí jsou také experimenty zaměřené na rychlost pozičního systému při změně pozorovaného bodu synchronizovaných kamer.



# Abstract

This thesis deals with the design and implementation of the extension of the existing library to control ip cameras, mainly by supporting the motion detection, multiple cameras motion synchronization and focus view at the specific point. There are also experiments performed on the speed of the system when changing the observed point of synchronized cameras.



# Klíčová slova

IP kamery, VAPIX API, pohled více kamer, detekce pohybu, HTTP

# Keywords

IP cameras, VAPIX API, multiple cameras view, motion detection, HTTP


# Citace

Michal Šeptun: Rozšíření knihovny pro ovládání IP kamer, bakalářská práce, Brno, FIT VUT v Brně, 2013

# Rozšíření knihovny pro ovládání IP kamer

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Radima Lužy.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Šeptun
13. května 2013

## Poděkování

Tímto bych rád poděkoval svému vedoucímu Ing. Radimu Lužovi za vedení a pomoc při psaní této práce a také za zpřístupnění potřebného vybavení.

# Obsah

# Kapitola 1

# Úvod

IP kamery je v dnešní době možno vidět téměř na každém kroku a využívají se ke všem možným účelům sledování, pozorování nebo zabezpečení. Jejich výhodou je jednoduchost použití, stačí připojit do sítě, nastavit IP adresu a lze začít sledovat požadovaný objekt. Kamery také mohou nabízet mnoho doprovodných funkcí, jako je například detekce pohybu nebo zvuku, zabezpečení proti krádeži, apod.

Tato práce rozšiřuje stávající knihovnu pro ovládání IP kamer o základní podporu událostí, zvláště pak detekci pohybu, která je právě pro zabezpečení velice užitečná.

Jelikož jsou kamery použity pro sledování robotického ramene, je vhodné zavést podporu pro sledování bodu v trojrozměrném prostoru pomocí souřadnic *[x, y, z]* a více kamer současně.

Nejdříve si popíšeme jak se kamery u robotického pracoviště dají ovládat pomocí VAPIX API, podíváme se také na stávající knihovnu a na další použité knihovny. Navrhneme a implementujeme funkce pro podporu událostí, detekce pohybu a možnost zadávat pro pohyb kamer souřadnice v prostoru místo jednotlivých úhlů naklonění a natočení. A nakonec provedeme experimenty zaměřené na rychlost reakce kamerového systému při změně polohy středového bodu pro synchronizaci pohledů kamer.

# Kapitola 2

# Rozhraní IP kamer

Výrobců IP kamer je mnoho a tak i jejich rozhraní jsou rozdílná. Mezi nejpoužívanější patří PSIA (Physical Security Interoperability Alliance), vyvynuto společnostmi IBM, Cisco a několika dalšími v roce 2008, pro komunikaci používá HTTP protokol, ONVIF (Open Network Video Interface Forum), standard společností AXIS, BOSH a Sony pocházející také z roku 2008, komunikace je zprostředkována pomocí SOAP[1] zpráv, které mají ovšem nevýhodu vetšího objemu dat. V roce 2003 vytvořila firma AXIS pro svoje kamery rozhraní založené na HTTP protokolu VAPIX.

V této práci se zaměříme na kamery AXIS, jelikož jsou umístěny u robotického ramene. Pro ovládání IP kamer tedy použijeme síťový protokol a to HTTP[8]. K přenosu větších dat (např. zachycených obrazů kamery, získání seznamu parametrů) se využívá rozšíření MIME s různým kódováním (multipart/form-data, multipart/x-mixed-replace) nebo formát XML.

## 2.1 VAPIX®

Všechny síťové video produkty Axis podporují aplikační rozhraní založené na HTTP, umožňující zasílat požadavky na obrázky, funkce pro ovládání kamer (PTZ[2]), nastavovat/získávat interní hodnoty parametrů a mnoho dalšího[1].

VAPIX obsahuje:

- Axis HTTP API,
- Axis Parameter Specification,
- Axis RTSP[3] API (pro kontrolu MPEG-4 streamů).

### 2.1.1 Axis HTTP API

HTTP API[2] poskytuje rozhraní pro ovládání, požadavky na data z kamer a nastavení kamer. K jednoduchému ovládání kamery můžeme například použít jakýkoli webový prohlížeč, kde do pole adresy zadáme náš dotaz. Například pro získání snímku kamery zašleme požadavek `http://<jménoserveru>/axis-cgi/jpg/image.cgi`.

Na základní ovládání a nastavení nám bude stačit toto HTTP API, ovšem velké množství vlastností a pokročilých funkcí (pro nás zajímavá Detekce pohybu 2.1.3) se spravuje přes

[1] Simple Object Access Protocol - výměna zpráv založených na XML
[2] Pan/Tilt/Zoom (otočení/naklonění/zoom)
[3] Real Time Streaming Protocol

parametry, které jsou definovány v Axis Parameter Specification 2.1.2. Většinu parametrů můžeme přidat, aktualizovat, odebrat či vypsat jejich hodnoty a mají různé bezpečnostní úrovně.

### 2.1.2 Axis Parameter Specification

Většina vlastností se muže nastavovat parametry, které jsou umístěny v odlišných konfiguračních souborech ve flash paměti. „Vzhledem k tomu, že flash paměť má jen omezenou životnost, není vhodné zapisovat parametry často. Výrobce odhaduje maximální počet flash zápisů asi na 100.000.“[5]

Každý parametr má uvedenou jednu z následujících úrovní zabezpečení (Security level) pro získání/nastavení hodnot:

- 0: nezabezpečené, ovšem přístup povolen minimálně s právi pro zobrazení;
- 1: tyto funkce mohou použít uživatelé s oprávněním `viewer`;
- 4: jsou požadována přístupová práva `operator`;
- 6: `admin`;
- 7: interní parametry, které mohou být modifikovány pouze firmware aplikacemi nebo rootem, editováním konfiguračních souborů přímo

Vytvoření nového uživatele s právy[4] `viewer`, `operator` a `admin`:

```
http://<jménoserveru>/axis-cgi/pwdgrp.cgi?action=add
&user=michal&pwd=sep&grp=users&sgrp=viewer:operator:admin&comment=Septun
```

Obecný tvar požadavku s parametry vypadá takto:

```
http://<jménoserveru>/axis-cgi/<práva>/param.cgi?
<parametr>=<hodnota>[&<parametr>=<hodnota>...]
```

Důležité parametry:

- `action`: Specifikuje jakou operaci chceme provést, hodnoty mohou být: `add, remove, update` nebo `list`.
- `group`: Jakou skupinu nastavení editujeme/zobrazujeme, pokud nebude uveden jsou vráceny všechny parametry zařízení.

Obvykle následují další parametry, uvedené v *Axis Parameter Specification*[3]

Například vytvoření události ze šablony, pojmenované MyEvent a parametrem Enabled s hodnotou yes

```
http://<jménoserveru>/axis-cgi/operator/param.cgi?action=add&group=Event
&template=event&Event.E.Name=MyEvent&Event.E.Enabled=yes
```

### 2.1.3 Detekce pohybu

Vestavěná inteligence kamer umožňuje generovat upozornění na různé události. Tímto můžeme například snížit množství přenášených dat a tak šetřit šířku pásma a nároky na paměť, když místo stálého přenášení videa budeme zasílat obrazy nebo upozornění, jen když se vyskytne nějaká událost, v našem případě detekován pohyb.

[4] přístupová práva jsou nastavena parametrem `sgrp`(secondary group)

### Okna detekce

Pro detekci pohybu musíme nejprve specifikovat okna v obrazu, kde se pohyb bude zaznamenávat. Můžeme také vytvořit okno, kde se pohyb vůbec detekovat nebude (např. část místnosti, kde se pohyb předpokládá). Ovšem počet takto definovaných oken by neměl být příliš vysoký, neboť se snižuje výkonnost dané kamery. Okna přidaná na stejnou pozici a se stejnými parametry by měli být eliminovány.

```
http://<jménoserveru>/axis-cgi/operator/param.cgi?action=add&group=Motion
&template=motion&Motion.M.Name=Entrance&Motion.M.WindowType=include
&Motion.M.Top=500&Motion.M.Bottom=7000&Motion.M.Left=5000&Motion.M.Right=8500
```

### Vytvoření události

Dalším krokem je vytvořit událost[4] skupiny Event užitím dynamických parametrů[5]:

```
http://<jménoserveru>/axis-cgi/admin/param.cgi?action=add
&group=Event&template=event
```

Ke kontrole vytvořených událostí si můžeme vypsat všechny události, popřípadě detail jedné určité.

```
http://<jménoserveru>/axis-cgi/admin/param.cgi?action=list
&group=Event.*.Name
```

```
http://<jménoserveru>/axis-cgi/admin/param.cgi?action=list
&group=Event.E2
```

Podporované HW události:

- Vstupní porty: Změna stavu na vstupním portu.
- PIR senzor: Detekce infračerveným senzorem.

Podporované SW události:

- Ruční spuštění: Virtuální port pro uživatelské události.
- Detekce pohybu: Kamera detekovala pohyb v kontrolovaném okně.
- Detekce zvuku: Hladinu hluku v okolí přesáhla nastavenou mez.
- Manipulace s kamerou: Pokud o krádež kamery.
- Teplota: Teplota vzrostla/klesla mimo rozsah pracovních teplot.
- Ztracení videa: Ztráta analogového signálu (pouze enkodér).
- Bootování: Například po výpadku napájení.
- Předvolená PTZ pozice: Kamera dosáhla požadované pozice.
- Síťové připojení: Kamera má aktivní síťové připojení.

[5] událost vytvořená za běhu

- Překročení čáry: Speciální případ detekce pohybu.

Následně také specifikovat další parametry události například, jméno, spouštění při detekování pohybu[6], čas trvání, buffer zahrnující snímky před/po spuštění a mnoho dalších (viz Specifikace parametrů[3] kapitola 2.11.1 Events.E#).

```
http://<jménoserveru>/axis-cgi/admin/param.cgi?action=add
&group=Event&template=event
&Event.E.Name=MotionDetectionEvent
&Event.E.SWInput=M<motion window no>:<trigger>
&Event.E.Duration=24:00
&Event.E.IncludePreTrigger=yes
&Event.E.PreTriggerSize=10
&Event.E.PreTriggerInterval=1000
```

### Vytvoření Event serveru

Před přiřazením akce k události musíme definovat server/y, které uvedeme v akci pro odeslaní dat nebo upozornění.

- FTP

```
http://<jménoserveru>/axis-cgi/admin/param.cgi?action=add
  &group=EventServers.FTP&template=ftp_config
&EventServers.FTP.F.Name=MyFTPServer
  &EventServers.FTP.F.Address=192.168.254.30
  &EventServers.FTP.F.Login=user
  &EventServers.FTP.F.Password=pass
  &EventServers.FTP.F.UploadPath=upload
  &EventServers.FTP.F.Passive=yes
```

- HTTP

```
http://<jménoserveru>/axis-cgi/admin/param.cgi?action=add
  &group=EventServers.HTTP&template=http_config
  &EventServers.HTTP.H.Name=MyHTTPServer
  &EventServers.HTTP.H.Address=http://192.168.254.10/cgi-bin/upload.cgi
  &EventServers.HTTP.H.Login=user
  &EventServers.HTTP.H.Password=pass
```

- TCP

```
http://<jménoserveru>/axis-cgi/admin/param.cgi?action=add
  &group=EventServers.TCP&template=tcp_config
  &EventServers.TCP.T.Name=MyTCPServer
  &EventServers.TCP.T.Address=192.168.254.20
```

[6] trigger:/=start pohybu,\=konec pohybu,x=start nebo konec.

#### Přidání akce k události

Aby vytvořená událost byla plnohodnotná a měla požadovanou funkci musíme nakonec vytvořit akci, kterou událost spustí.Máme možnost těchto akcí:

- Event FTP Actions
  Akce posílá soubory na FTP server[7].
- Event HTTP Actions
  Tato akce zasílá buď soubory nebo definovanou zprávu na HTTP server[7]. K URL také můžeme přidat vlastní parametry.
- Event HW Actions
  Ovládání digitálních výstupů.
- Event SMTP Actions
  Zaslání e-mailu na zadanou adresu.
- Event TCP Actions
  Upozornění na TCP/IP server[7].
- Event PTZ Actions
  Otočení kamery na přednastavenou pozici P0...Pn.
- Event Guard Tour Actions

Akce pro odeslání zprávy a dat na e-mailovou adresu:

```
http://<jménoserveru>/axis-cgi/admin/param.cgi?action=add
&group=Event.E2.Actions&template=smtpaction
&Event.E2.Actions.A.Type=U
&Event.E2.Actions.A.EmailTo=name@server.cz
&Event.E2.Actions.A.Subject="Detekce kamer"
&Event.E2.Actions.A.Message="Kamera 1 detekovala pohyb"
```

### 2.1.4 Změna pohledu kamery

Kamery také disponují rozhraním pro změnu pohledu kamery a přiblížení, zkráceně PTZ[8]. Rozhraní zahrnuje také nastavení clony, jasu a ostření. Kamerami lze pohybovat v několika režimech.

Relativní posun změní pohled kamery vůči aktuální pozici o zvolený počet stupňů, tedy například o 5stupňů vlevo a o 20stupňů dolů:

```
http://<servername>/axis-cgi/com/ptz.cgi?rpan=-5&rtilt=20
```

Změnu pohledu od počáteční polohy (0,0) způsobíme zadáním úhlů v absolutním režimu. Natočení na 20stupňů a naklonění na 60stupňů:

```
http://<servername>/axis-cgi/com/ptz.cgi?pan=20&tilt=60
```

[7] server musí být specifikován dynamickým parametrem Event servers viz 2.1.3
[8] PTZ - Pan/Tilt/Zoom

Některým aplikacím může také vyhovovat kontinuální změna pohledu konstantní rychlostí[9], kterou si zvolíme. Pohyb vpravo rychlostí 8 a dolů rychlostí 2:

```
http://<servername>/axis-cgi/com/ptz.cgi?continuouspantiltmove=8,2
```

K jednoduššímu ovládání mohou pomoci i některé simulované režimy, jako například automatické ostření a nastavení clony popřípadě digitální zoom, který umožní přiblížit obraz více než optika kamery. Zvolenou polohu PTZ je také možno uložit a místo zadávání úhlů natočení a naklonění zadat uložený název pozice.

## 2.2 Stávající knihovna

Jelikož se jedná o *rozšíření knihovny pro ovládání IP kamer* podíváme se také na již existující knihovnu vytvořenou Janem Schmiedem[9]

Nynější implementace poskytuje aplikační rozhraní pro jazyk C++ a je rozdělena do těchto hlavních modulů, které umožňují další rozšíření:

- Ovládání pozičního systému IP kamery a získáni obrazu, ...;
- AV Streamy a kodeky (MJPEG, RTSP, FFmpeg, ... );
- Události kamer (není implementováno, částečně v této práci);
- Utility (pomocné třídy např. Recorder, CameraPool, ... )

### 2.2.1 Ovládání IP kamer

Modul obsahuje funkce pro inicializaci proměnných –ne kamery, u ní není inicializace nutná– potřebných k navázání spojení s kamerou, změnu pohledu kamery, získávání logů a snímků, nastavení parametrů. Nejdůležitější je třída `CameraInit` a její metoda `Connect` obstarávající připojení ke kameře. Následně můžeme získat ukazatel na instanci třídy `BaseCamera` a kameru začít ovládat, popřípadě ji přidat do skupiny a ovládat více kamer najednou.

Ke komunikaci mezi kamerou a knihovnou slouží komunikátory odvozené od třídy `BaseCommunicator`, v případě VAPIX je to `HTTPCommunicator`, který využívá pro zasílání požadavků a přijímání odpovědí externí knihovnu cURL viz 2.3.1

### 2.2.2 AV streamy a kodeky

Zpracování a uchování MJpeg nebo RTSP streamovaných dat probíhá pomocí několika tříd. Základní interface pro implementaci streamů je obsažen v `BaseAVStream`, obsahuje audio a video buffery. Od základní třídy jsou odvozeny konkrétní implementace, které mohou použít i kodeky pro převod mezi jednotlivými formáty.

### 2.2.3 Utility

Jistým zjednodušením aplikačního programování jsou třídy `Recorder`, `GroupCamera` a `CameraPool`, které nejsou nezbytné pro činnost knihovny, ale poskytují další prostředek pro správu kamer.

`Recorder` použijeme pro nahrávání proudu snímků s případnou možností dekomprese, u které je nutné správné nastavení kodeků.

[9] jednotka rychlosti není specifikována, lze však předpokládat stupně za sekundu

`GroupCamera` lze použít pro ovládání více kamer, můžeme všem kamerám ve skupině zaslat stejný příkaz, například pro změnu pohledu kamery jinými PTZ parametry.

V `CameraPool` mohou být kamery pojmenovány a zařazený do jednotlivých skupin.

## 2.3 Další použité knihovny

Jelikož knihovna neimplementuje vlastní systém pro komunikaci po síti a pro jednodušší práci s multimediálním obsahem, jsou využity následující externí knihovny obstarávající potřebnou funkčnost.

### 2.3.1 cURL

*cURL* představuje softwarový projekt zahrnující knihovnu a nástroj příkazové řádky pro přenášení dat po síti za použití různých protokolů jako jsou například FTP, HTTP, HTTPS, Telnet, POP3, SMTP a RTSP. Umožňuje také použít HTTPS certifikáty, cookies, user-password ověřování a mnoho dalšího.

Pro komunikaci s kamerami použijeme funkce *libcurl* knihovny pro generování HTTP dotazů a pro ověřování identity jednoduché ověření přístupu (Basic access authentication), které server zabudovaný v kamerách podporuje.

Knihovna je volně k dispozici pod licencí MIT, lze ji kompilovat a spouštět pod mnoha operačními systémy. K dispozici jsou ke stažení jak zdrojové kódy, tak binární soubory, které je možno stáhnout na webu projektu [10].

### 2.3.2 FFmpeg

Soubor knihoven a utilit umožňující nahrávání, konverzi a streamování obrazu a zvuku, obsahuje významnou knihovnu s audio/video dekodéry a enkodéry *libavcodec*, která je využita pro kompresi a dekompresi videa

*FFmpeg* obsahuje tyto knihovny:

- libavcodec – knihovna s kodeky;
- libavfilter – zvukové a obrazové filtry;
- libavformat – demuxery a muxery[10] pro audio/video kontejnerové formáty;
- libavutil – funkce pro usnadnění programování;
- libavdevice – práce s vstupními a výstupními zařízeními;
- libaswscale – změna velikosti, barvy a formátu snímků;
- libswresample – převzorkování a konverzi formátu vzorků;

a různé nástroje:

- ffmpeg - nástroj příkazové řádky pro převod formátů multimedialních souborů;
- ffserver - server podporující živé vysílání multimédií;

[10] Demultiplexer, demuxer dokáže extrahovat jednotlivé datové toky (video, audio, titulky) z multimediálního kontejneru, opakem je multiplexer neboli muxer

- ffplay - jednoduchý přehrávač založený na FFmpeg knihovnách;
- ffprobe - analyzátor multimediálních streamů;

Knihovna je vyvíjena pod operačním systémem Linux, ale podobně jako u cURL ji lze přeložit na různých systémech, podrobnosti k překladu knihoven viz 6. Opět se jedná o otevřený software pod LGPL nebo GPL licencí – závisí na konfiguraci – a lze ji stáhnout na webu FFmpeg [7]

# Kapitola 3

# Návrh

V návrhu bylo nutné vytvořit třídy, které budou dále použitelné a rozšiřitelné, pro vytvoření události a s ní spojené akce a dále funkce použité při pohledu více kamer na jeden bod. Také je důležité zohlednit již hotové třídy pro komunikaci, výjimky,..

## 3.1 Události

K událostem nejprve vytvoříme základní abstraktní třídu, neboli interface `BaseEvent` a pro každou událost vytvoříme samostatnou třídu, která musí implementovat obecné metody té základní, a to:

- výpis již vytvořených Event serverů,
- výpis podobných událostí,
- výpis akcí k určité události,
- vytvoření Event serveru,
- vytvoření vlastní události,
- vytvoření akce, která je na událost navázána

Každá vytvořená třída bude mít také konstruktor s parametrem, který bude obsahovat instanci třídy pro komunikování po síti.

## 3.2 Detekce

Jak je uvedeno již dříve v kapitole 2.1.3 pro korektní zprovoznění detekce pohybu kamerou, je nutné provést několik kroků a to v určeném pořadí:

- specifikace rozměrů okna detekce,
- vytvoření události (tedy v tomto případě detekce),
- Event server (kam se budou následně zasílat data nebo upozornění),
- vytvořit akci k dané události

Každý krok reprezentuje jedna či více funkcí s nezbytnými parametry.

### 3.2.1 Okno detekce

Vytvoření okna detekce bude možné jednou ze dvou funkcí s rozdílným počtem parametrů. Základní funkce bude obsahovat tyto parametry: *jméno okna, vrchní okraj, spodní okraj, levý okraj, pravý okraj*. Dále bude možno zadat parametry pro zvolení *zdroje obrazu, citlivosti, historie a velikosti detekovaného objektu*.

*Jméno okna* může být jakýkoliv námi zvolený textový řetězec. *Vrchní okraj, spodní okraj* se udává v rozsahu 0-9999, kde maximální hodnota znamená maximální výšku a nula odpovídá vrchnímu okraji obrazu. *Levý okraj, pravý okraj* se udává ve stejných jednotkách jako vrchní a spodní okraj, s rozdílem že se nula bere od pravého okraje. V podstatě můžeme říci, že 9999 je 100% dané velikosti obrazu a když budete chtít například pravý okraj do půlky obrazu zadáme hodnotu 5000. *Zdroj obrazu* specifikuje v jakém zdroji se okno detekce nachází. *Citlivost* určuje, jak moc se musí objekt lišit od pozadí, maximální citlivost (100) může způsobit i nežádoucí jev, že se detekce aktivuje i při vetším šumu v obrazu a naopak při malé citlivosti musí být rozdíl obrovský, například černá na bílém pozadí. Pomocí *historie* můžeme upravovat citlivost na rychlý pohyb. A *velikostí obrazu* udáváme, jak moc se obraz musí procentuálně změnit.

### 3.2.2 Vytvoření události

Událost detekce pohybu vytvoříme zadáním tří základních parametru *jméno události, číslo okna, spuštěni*. Dále je také možno specifikovat detailnější parametry jako *trvání, dny v týdnu, čas startu, pre/post trigger*.

*Jméno události* může být opět libovolný řetězec, *číslo okna* získáme z okna vytvořeného v předchozím kroku, například M0 nebo M2, jedná se o identifikátor vytvořený dynamicky systémem kamery vždy sestávající z písmene M a pořadového čísla okna. Parametrem *spuštění* určíme, jestli se má událost spustit při vzniku detekce pohybu, na konci nebo v obou případech. Jak dlouho může být událost spuštěna definujeme časem *trvání* ve formátu „hodiny:minuty“ s maximální dobou 168:00 hodin, dále je také možno zvolit v jaké dny se má detekovat změna v obrazu, *dny v týdnu* pomocí vektoru čísel určí tyto dny začínající nedělí, například „0111110“ znamená detekce od pondělí do pátku a „0000010“ pouze v pátek. *Čas startu* určuje dobu započetí detekce ve zvolené dny, udávající ve 24 hodinovém formátu. Pomocí *pre/post triggeru* můžeme definovat buffer obrázků zaznamenaných před nebo po spuštění události, které se zašlou společně s událostí, pokud je akce nastavena na zasílání obrázků.

### 3.2.3 Event server

Prvotní návrh obsahoval myšlenku, že vytvoření jakéhokoliv *event serveru* bude zajišťovat jedna funkce. U všech musíme určit jméno a adresu, buď IP nebo doménové jméno, ovšem jednotlivé servery mají v některých případech až moc odlišností a tak bude pro vytvoření každého serveru jednotlivá funkce s potřebnými argumenty.

Funkce pro vytvoření FTP, HTTP a TCP/IP tedy budou všechny obsahovat *jméno*, tvořené libovolným řetězcem a *adresu* tvořenou buď IP adresou nebo doménovým jménem. Dále pak každá funkce bude mít své potřebné parametry.

#### FTP

Protokol pro přenos souborů a s ním spojený FTP server mohou vyžadovat ověření uživatele, proto také při vytváření tohoto typu event serveru můžeme zadat *jméno* a *heslo*. Dále pak lze specifikovat cestu pro ukládání souborů parametrem *cesta uložení*, zvolit *port* na kterém daný server naslouchá a popřípadě zvolit pasivní režim přenosu.

#### HTTP

*HTTP server* jako jediný umožňuje zasílat jak upozornění, tak data na zvolenou adresu, která specifikujeme i dokument či skript na zpracování požadavku, například: *„http://192.168.254.10/cgi-bin/upload.cgi“*, na ověření identity http protokolem je možné zadat *jméno* a *heslo*. Lze také definovat proxy server a to *proxy adresou*, *portem* a uživatelským *jménem* a *heslem* pro přihlášení na daný server.

#### TCP/IP

Na *TCP/IP server* lze zasílat pouze upozornění, nikoli obrazová data. A kromě *jména* a *adresy* lze nastavuje také *port*, na který se má upozornění zaslat.

#### SMTP

Akce využívající SMTP protokol, tedy pracující s e-mailem, potřebují pro svoji funkčnost definovat mail server. I když se nejedná a dynamický parametr jako je Event server, ale o nastavení již existujících proměnných, kvůli podobnosti je i toto nastavení v této části.

V základním nastavení definujeme *odesilatele mailu*, zvolená e-mailová adresa zobrazovaná v poli „Od“, *hlavní* a *záložní mail server*, IP adresa nebo doménové jméno SMTP serveru pro odeslání pošty.

K jednotlivým mailserverům lze také přidat ověření identity pomocí *jména*, *hesla*, zvolit je možno i nejslabší metodu ověřování. Na výběr je jedna ze čtyř možností Login, Plain, CRAM-MD5, DIGEST-MD5, popřípadě se můžeme přihlásit také přes POP server, u kterého zadáme adresu, opět buď IP nebo doménové jméno.

### 3.2.4 Akce k události

Na spuštění události musíme také zaregistrovat akci, která se provede. *Akce k události* můžeme rozdělit na dvě základní:

- Upload souborů,
- Upozornění formou zprávy

Soubory lze odesílat na servery FTP, HTTP, SMTP, upozornění pak na TCP/IP, HTTP, SMTP a zvláštním druhem upozornění je PTZ akce.

K jednotlivým akcím budeme zadávat následující:

- FTP: označení již vytvořeného FTP Event serveru,
  označení záložního serveru;
- HTTP: označení HTTP Event serveru,
  obsah odesílané zprávy,
  volitelné parametry pro přidání k URL;

- TCP/IP: označení TCP/IP Event serveru,
  obsah zprávy upozornění;
- SMTP: adresáta, jakákoliv e-mailová adresa,
  předmět zprávy,
  textový obsah e-mailu,
  maximální počet obrázku v jedné zprávě;
- PTZ: označení uložené pozice kamery, P0, P1,...

### 3.2.5 Jednodušší detekce

Pro aplikace, které nemusí přesněji nastavat detekce, by bylo vhodné vytvořit metodu, která vše potřebné obstará. Například využít specializaci třídy `V2MotionDetect`. K oznámení, že se před kamerou něco pohnulo je nejprve nutno v děděné třídě přepsat jednu z jejich metod, ve které specifikujeme co se má v případě zaznamenání pohybu stát a následně využít metodu, která provede všechny potřebné kroky, tedy:

vytvoří socket, na kterém bude přijímat případnou zprávu o detekování pohybu,
nastaví kameru:

- velikost okna pro detekci nastaví přes celý obraz,
- vytvoří událost s tímto oknem,
- event server nastaví na http a nasměruje na vlastní síťové rozhraní,
- vytvoří akci k události, která při detekování pohybu pošle upozornění aplikaci,

čeká na přijetí zprávy od kamery a v případě detekce invokuje přepsanou metodu s požadovaným chováním.

## 3.3 Synchronizace pohledu více kamer

Pro otočení více kamer do jednoho bodu budeme potřebovat funkce 4.2 pro vypočítání úhlů natočení a naklonění, které se provedou při každé změně souřadnic pro každou kameru zvlášť a systém PTZ následně nasměruje pohled určeným směrem viz 2.1.4.

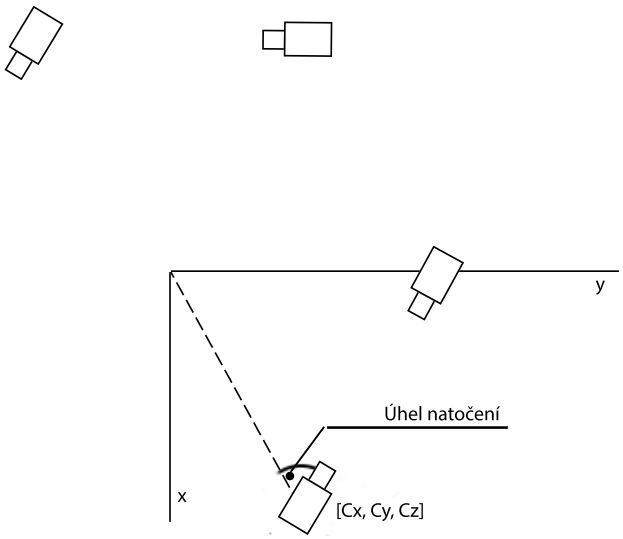
### 3.3.1 Inicializace

Správná změna pohledu je také závislé na inicializaci proměnných potřebných pro výpočet úhlů. Před zahájením práce je nutné zadat souřadnice umístění jednotlivých kamer v souřadném systému a natočení od středu, viz obrázek základního natočení z půdorysu 3.1.

#### Pomocný bod pro výpočet natočení

Ze zadaných souřadnic vypočítáme pomocný bod ležící na přímce, která prochází horizontální osou pohledu kamery. Pokud směřuje kamera do středu souřadného systému ([0, 0, 0]), tak můžeme za pomocný bod zvolit souřadnice [0,0,z] a je hotovo, ovšem pokud je kamera natočena budeme muset provést následující odvození souřadnic.

Bod bude vrchol nad přeponou pomyslného trojúhelníka, další dva vrcholy budou *pozice kamery* a *počátek souřadného systému* a poslední potřebný údaj je *základní úhel* natočení kamery, což bude jeden z úhlů u přepony.




Obrázek 3.1: Základní úhel

Nejprve vypočítáme délku strany mezi kamerou a počátkem souřadného systému:

$$stranaPK = \sqrt{C_x^2 + C_y^2} \tag{3.1}$$

Pro zjednodušení bude délka strany od kamery vždy polovina strany mezi počátkem a kamerou:

$$stranaK = \frac{stranaPK}{2} \tag{3.2}$$

Délku strany od počátku získáme Kosinovou větou[6]:

$$a^2 = b^2 + c^2 - 2bc\cos\alpha \tag{3.3}$$

po dosazeni:

$$stranaP = \sqrt{stranaPK^2 + stranaK^2 - (2 * stranaPK * stranaK * \cos(zaklad.uhel))} \tag{3.4}$$

Nyní známe délky stran, pomocný bod získáme vypočítáním průsečíku dvou kružnic, jedné se středem v počátku, poloměrem odpovídajícímu délce strany od počátku *stranaP* a druhé se středem v pozici kamery a poloměrem délky strany od kamery *stranaK*. Obecná rovnice kružnice:

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{3.5}$$

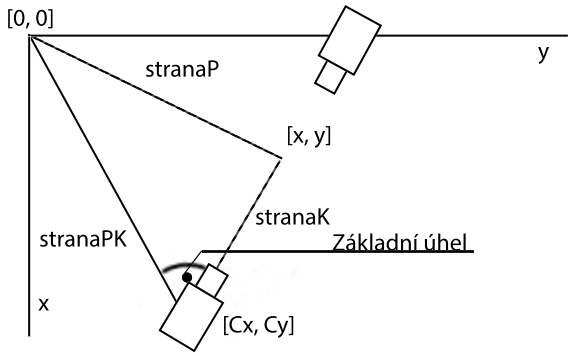
První a druhá rovnice po dosazení:

$$(x)^2 + (y)^2 - stranaP^2 = 0 \tag{3.6}$$

$$(x - C_x)^2 + (y - C_y)^2 - stranaK^2 = 0 \tag{3.7}$$

Nyní odečteme od první rovnice druhou a vyjádříme y:

$$2 * x * C_x - C_x^2 + 2 * y * C_y - Cy^2 - stranaP^2 + stranaK^2 = 0 \tag{3.8}$$




Obrázek 3.2: Pomocný trojúhelník

$$y = \frac{(-2 * C_x) * x + (C_x^2 + Cy^2 + stranaP^2 - stranaK^2)}{2 * C_y} \tag{3.9}$$

Pro jednodušší práci si konstanty, tam kde to jde, nahradíme jednou neznámou: $i = -2 * C_x$
$j = C_x^2 + Cy^2 + stranaP^2 - stranaK^2$
$k = 2 * C_y$

$$y = \frac{i * x + j}{k} \tag{3.10}$$

Nyní dosadíme y3.10 do první obecné rovnice kružnice3.6:

$$x^2 + \frac{i^2 * x^2 + 2 * i * j * x + j^2}{k^2} - stranaP^2 = 0 \tag{3.11}$$

z této rovnice potřebujeme dostat kvadratickou rovnici ve tvaru

$$a * x^2 + b * x + c = 0 \tag{3.12}$$

Mějme na paměti, že vše kromě $x$ a $y$ jsou konstanty. Naše upravená rovnice bude tedy vypadat takto:

$$(1 + \frac{i^2}{k^2}) * x^2 + \frac{2 * i * j}{k^2} * x + \frac{j^2}{k^2} - stranaP^2 = 0 \tag{3.13}$$

Konstanty $i$, $j$, $k$ lze expandovat a získáme konstanty kvadratické rovnice $a$, $b$, $c$:

$$a = 1 + \frac{(-2 * C_x)^2}{(2 * C_y)^2} \tag{3.14}$$

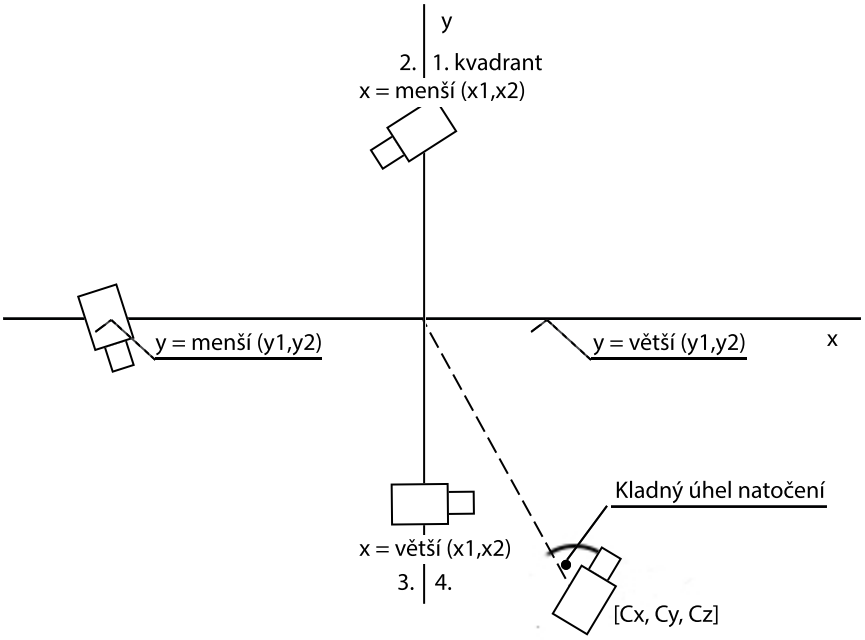
$$b = \frac{-4 * C_x * (C_x^2 + C_y^2 + stranaP^2 - stranaK^2)}{(2 * C_y)^2} \tag{3.15}$$

$$c = \frac{(C_x^2 + C_y^2 + stranaP^2 - stranaK^2)^2}{(2 * C_y)^2} - stranaP^2 \tag{3.16}$$

Pomocí diskriminantu vypočítáme $x_1$ a $x_2$, dosadíme do první rovnice 3.6 a získáme i $y_1$ a $y_2$.

Úpravami rovnic a tím, že se vetšinou souřadnice kamery v rovnici umocňují na druhou, ztrácíme přesný obraz o znaménku jednotlivých souřadnic, proto musíme provést korekci tím, že se podíváme na skutečné souřadnice x a y kamery a pokud je některá záporná, tak musíme vynásobit -1 také vypočítanou souřadnici pomocného bodu. Jinak by nám mohl vyjít pomocný bod špatně.

Dále také musíme určit tu správnou dvojici souřadnic x, y. Nejlépe si to ukážeme na obrázku3.3, ve kterém jsou zobrazeny osy x, y a kladný počáteční úhel natočení kamery.




Obrázek 3.3: Výběr souřadnic x, y

Můžeme vidět několik možných případů, ty odpovídají počátečnímu kladnému úhlu, při záporném úhlu se vezmou druhé dvě souřadnice:

- 1. a 2. kvadrant ($y > 0$): pokud se kamera nachází v této oblasti, požadovaný pomocný bod bude ten s menší $x$ souřadnicí,
- 3. a 4. kvadrant ($y < 0$): v tomto případě bereme bod naopak s větší $x$ souřadnicí

Zvláštní případy jsou také hraniční polopřímky mezi oběma polorovinami (y=0):

- hranice mezi 1. a 4. kvadrantem ($x > 0$): pomocný bod kamery umístěné na této polopřímce bude takový, kde je $y$ souřadnice větší,
- hranice mezi 2. a 3. kvadrantem ($x < 0$): opět opačný případ, $y$ souřadnice menší

Tímto jsme získali pomocný bod v ose pohledu kamery, který použijeme při vypočtu úhlu natočení kamery, pro zadané souřadnice v trojrozměrném prostoru.

#### Obecná přímka osy pohledu kamery

Protože se kamera může otáčet jak doprava, kladný úhel, tak doleva, záporný úhel, musíme také určit toto znaménko u úhlu. To určíme tak, že zjistíme v jaké pozici vůči ose kamery bude zadaný bod. Pro tento účel budeme potřebovat obecnou rovnici přímky a po dosazení bodu do této rovnice požadovanou informaci dostaneme.

Obecný tvar přímky tedy vypadá takto:

$$a * x + b * y + c = 0 \tag{3.17}$$

K určení obecné rovnice osy použijeme souřadnice kamery $K$ a vypočítaný pomocný bod $P$ ležící na této ose, viz výše 3.3.1. Nejprve musíme určit směrový vektor:

$$\vec{u} = P - K \Rightarrow \vec{u} = (P_x - K_x; P_y - K_y) \tag{3.18}$$

Přehozením souřadnic směrového vektoru a změnou jednoho znaménka získáme normálový vektor.

$$\vec{n} = (u_2; -u_1) \tag{3.19}$$

Do obecné rovnice 3.17 dosadíme za $a$, $b$ první dvě souřadnice normálového vektoru

$$n_1 * x + n_2 * y + c = 0 \tag{3.20}$$

Konstantu $c$ získáme tak, že za $x$, $y$ dosadíme souřadnice kamery

$$c = -1 * (n_1 * K_x + n_2 * K_y) \tag{3.21}$$

Pomocný bod ležící v ose pohledu kamery, normálový vektor a konstantu obecné rovnice osy tedy už známe a nyní můžeme přistoupit k výpočtu natočení a naklonění kamery ze zadaného bodu.

### 3.3.2 Natočení kamery

Ke správnému *natočení kamery* v horizontální rovině použijeme již zmíněnou Kosinovu větu 3.3, ze které si vyjádříme úhel:

$$\cos\alpha = \frac{b^2 + c^2 - a^2}{2bc} \tag{3.22}$$

Pomocí 2 známých(kamera a pomocný bod) a jednoho zadaného požadovaného bodu vypočítáme délky jednotlivých stran, které následně dosadíme do vzorce.

Délka strany:

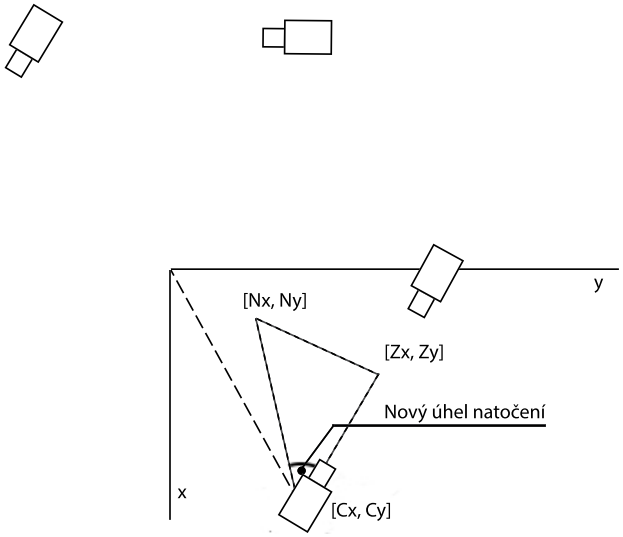
$$b = \sqrt{(C_x - Z_x)^2 + (C_y - Z_y)^2} \tag{3.23}$$

Vzorec s dosazením délky stran:

$$\cos\alpha = \frac{(C_x - Z_x)^2 + (C_y - Z_y)^2 + (N_x - C_x)^2 + (N_y - C_y)^2 - ((Z_x - N_x)^2 + (Z_y - N_y)^2)}{2\sqrt{(C_x - Z_x)^2 + (C_y - Z_y)^2}\sqrt{(N_x - C_x)^2 + (N_y - C_y)^2}} \tag{3.24}$$

Všechny požadované hodnoty tedy již známe, dosadíme souřadnice bodů z obr.3.4 do rovnice 3.24 a získáme úhel.

Protože jsme úhel získali z délek stran, nevíme na jakou stranu tento úhel má směřovat, proto musíme ještě zadaný bod dosadil do obecné rovnice přímky 3.3.1. Získáme kladné nebo záporné číslo, toto znaménko odpovídá i znaménku úhlu, pokud tedy po dosazení získáme záporné, tak musíme výsledný úhel natočení vynásobit -1. Nyní již tento vypočítaný úhel odešleme kameře ke zpracování.

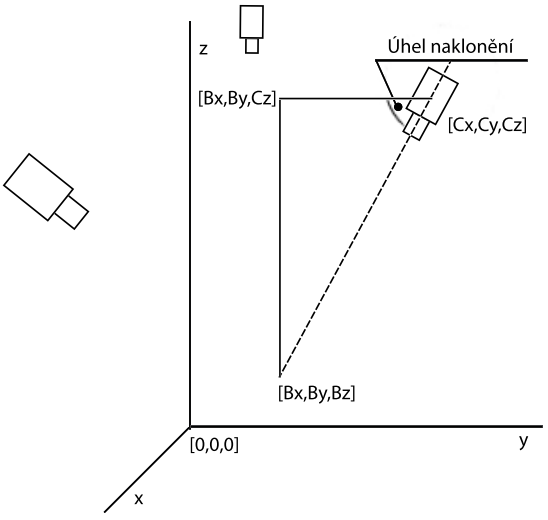



Obrázek 3.4: Úhel natočení

### 3.3.3 Naklonění kamery

*Úhel naklonění* vypočítáme pomocí kombinace Pythagorovi věty, na vypočítání délky spodní strany pomocného trojúhelníka a goniometrické funkce tangens, ze které dostaneme vlastní úhel naklonění.

$$\tan \beta = \frac{\sqrt{(C_x - B_x)^2 + (C_y - B_y)^2}}{C_z - B_z} \tag{3.25}$$




Obrázek 3.5: Úhel naklonění

Úhel naklonění počítáme od kamery dolů, tam má kamera záporný úhel, proto musíme otočit znaménko u výsledné hodnoty před odesláním k vykonání změny.

### 3.3.4 Zaostření

Na zaostření můžeme použít funkci automatického zaostření nebo zaostřit na základě vzdálenosti kamery od objektu. Pro tento účel budou provedeny experimenty, jaké se má použít zaostření na určenou délku.

Vzdálenost $v$ požadovaného bodu $B$ od kamery $C$ vypočítáme podle vzorce:

$$v = \sqrt{(C_x - B_x)^2 + (C_y - B_y)^2 + (C_z - B_z)^2} \tag{3.26}$$

Podle dokumentace má funkce pro manuální zaostření kamery rozsah hodnot 0-9999 a zadaná hodnota znamená „posunutí zaostření n-krát“, ovšem nikde se neobjevuje jakou mají hodnoty jednotku.

# Kapitola 4

# Implementace

Knihovna pro ovládání kamery je implementována objektově orientovaným způsobem v jazyce C++, jak již bylo zmíněno využívá dvě externí knihovny cURL a FFmpeg. S použitím první žádný problém nebyl, ale po úspěšném překladu druhé a pokusech přeložit stávající *knihovnu pro ovládání IP kamer* se vyskytl problém. Tento problém byl způsoben absencí makra, které vytvářelo typu „`AVCodecID`“ alias „`CodecID`“, které se v knihovně vyskytovalo.

Dále v návrhu nebyla uvažována potřeba specifikovat DNS server, ale jelikož adresy serverů se mohou zadávat i doménovým jménem, tak pokud této možnosti chceme využít musíme nastavit odpovídající parametr kamery „`root.Network.DNSServer1`“ popřípadě i záložní server „`root.Network.DNSServer2`“.

Protože události i pohyb kamer jsou implementovány v samostatných třídách bylo také nutné vytvořit korespondující metody ve třídě pro obsluhování kamer `Vapix2Camera`.

## 4.1 Události

Podpora událostí zahrnující detekci pohybu byla implementována za pomoci následujících tříd a funkcí.

### 4.1.1 Základní třída událostí

Podle návrhu 3.1 byla vytvořená třída `BaseEvent` se základními funkcemi, které jsou pro všechny události stejné. A to vytvoření různých Event serveru, výpis existujících serverů, událostí, akcí k událostem a vytvoření akce k události.

Pro komunikaci s kamerou je obsažena proměnná referencující instanci třídy `HTTPCommunicator`, která obsahuje informace pro spojení s kamerou a potřebné funkce pro odeslání požadavků a příjem odpovědí. Tuto proměnnou musí každá děděná třída nastavit, jinak nebude možno s událostmi pracovat. Pro tento účel byla ve třídě pro obsluhu Vapix 2 kamer vytvořena funkce `getCommunicator()`, která instanci komunikátoru získá.

V této třídě jsou všechny metody virtuální a to z důvodu možné změny chování v děděných třídách.

### 4.1.2 Detekce pohybu

*Detekce pohybu* je jednou z událostí kamery, proto použijeme základní třídu událostí 4.1.1 a specializujeme ji. Důležitou funkci má parametrický konstruktor, kterým předáme instanci komunikátoru a tím v podstatě určíme s jakou kamerou se má komunikovat. Následně je také

obsažena metoda specifická pro detekci a to vytvoření okna, ve kterém se pohyb detekuje `CreateDetWindow()`. Je také kontrolován rozsah hodnot, který musí být v intervalu od 0 do 9999. Pro ověření slouží metoda vypisující již vytvořená okna.

Vlastní událost detekce se vytvoří metodou `CreateEvent()`, které se zadají parametry obsahující číslo vytvořeného okna a požadovaný spouštěč události START, STOP nebo START_STOP.

## 4.2 Pohyb kamer pomocí 3D souřadnic

Synchronizace pohledu kamer je zajišťována metodami pro pohyb kamer pomocí prostorových souřadnic a již vytvořenou utilitou `GroupCamera`, která pošle požadavek všem kamerám přidaných do skupiny.

### 4.2.1 Natočení a naklonění

Třída pro výpočet *natočení a naklonění* obsahuje pouze dvě veřejné metody a parametrický konstruktor pro inicializaci proměnných, požadující zadat instanci komunikátoru. Podobně jako u událostí, souřadnice udávající pozici kamery a základní úhel natočení kamery. První metoda slouží k zadání změn pozice kamery a druhá pro zadání bodu, do kterého požadujeme kamery směřovat. Tato metoda pak invokuje jednotlivé privátní metody pro výpočty.

#### Inicializace

V konstruktoru také provedeme důležité výpočty 3.3.1 pomocného bodu a obecné rovnice přímky, tím naplníme nezbytné instanční proměnné *souřadnice pomocného bodu*, *normálový vektor* a konstantu $c$, které použijeme při výpočtech úhlů natočení a naklonění.

#### Natočení

Pomocí souřadnic pomocného bodu, pozice kamery a požadovaného bodu vytvoříme rovnici 3.24 pro získání úhlu. Protože funkce `acos()` počítá výsledný úhel v radiánech, musíme jej pomocí vzorce 4.1 převést na stupně, které se zadávají PTZ systému kamery (a je velikost úhlu v radiánech a $\alpha$ ve stupních).

$$\alpha = \frac{a * 180}{\pi} \tag{4.1}$$

A znaménko určíme pomocí výsledku obecné rovnice.

$$obec = normVec.a * x + normVec.b * y + c; \tag{4.2}$$

#### Naklonění

Jelikož se jedná o jednoduché operace v pravoúhlém trojúhelníku, zde žádný problém nenastal. Pouze musíme stejně jako u natočení převést výsledek funkce `atan()`, která dává výsledek v radiánech na stupně. A případně provést malou korekci, a to pokud se úhel nerovná nule, tak musíme otočit znaménko výsledného úhlu z důvodu popsaného v návrhu 3.3.3.

### 4.2.2 Zaostření

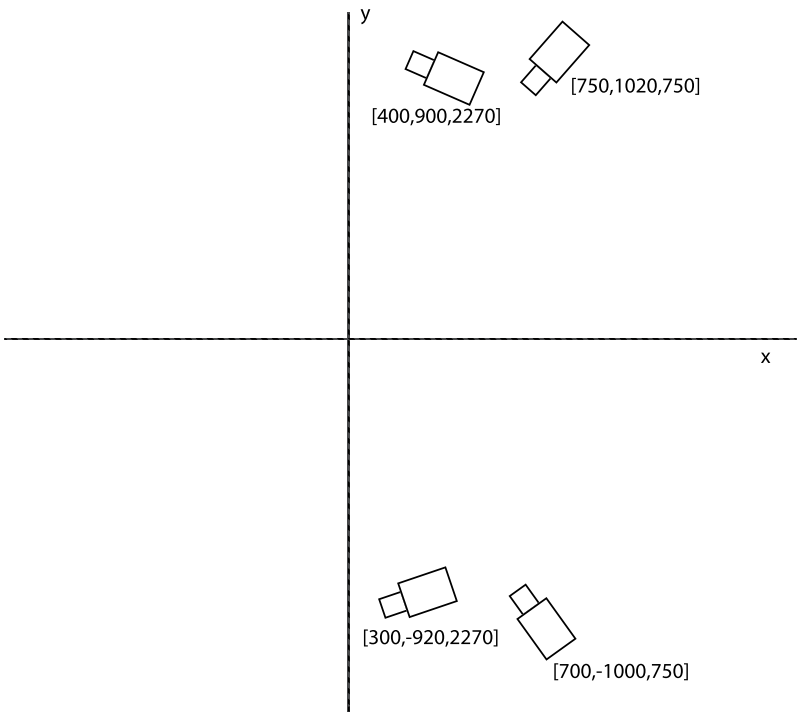
Po provedení několika neúspěšných pokusů se zaostřením, nejspíše způsobených nejasnou definicí nastavování manuálního zaostření a jednotek hodnoty zaostření, bylo upuštěno od zaostřovaní na základě vzdálenosti od objektu. Použijeme pouze automatické zaostření, s možností manuální korekce, kterou kamera poskytuje. I samotný výrobce doporučuje používat automatické zaostření oproti manuálnímu.

# Kapitola 5

# Rychlost reakce PTZ systému

Po dokončení implementace bylo provedeno měření zaměřené na rychlost reakce kamerového systému při změně polohy středového bodu pro synchronizaci pohledů kamer.

## 5.1 Umístění kamer u robotického pracoviště



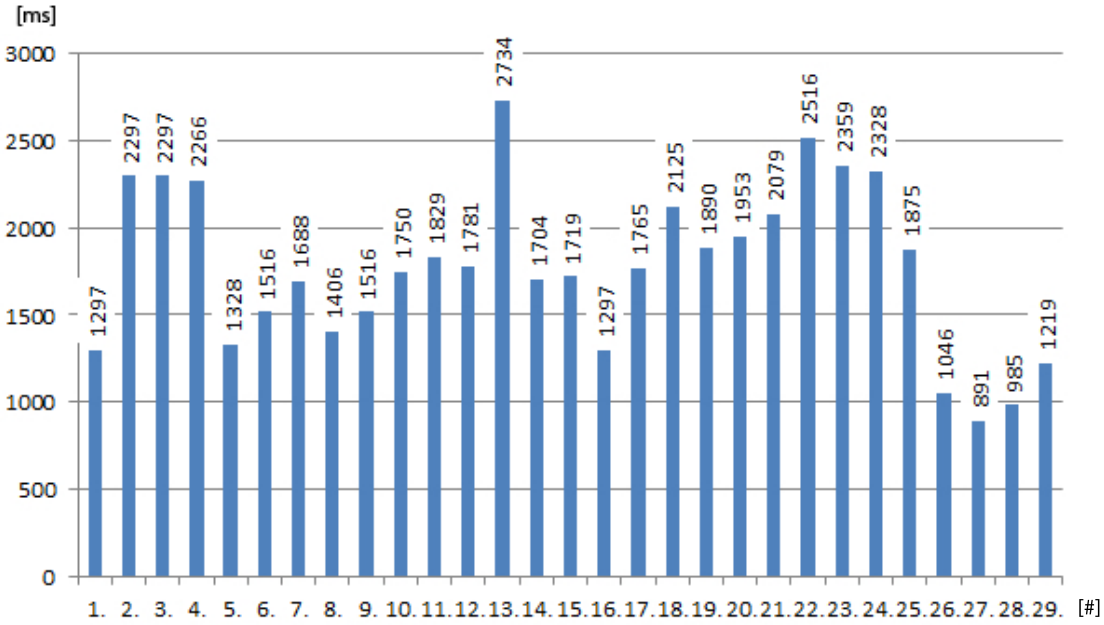

Obrázek 5.1: Umístění kamer

Měření bylo prováděno u pracoviště robotického ramene se 4 nainstalovanými kamerami

AXIS 214 PTZ Network Camera s verzí firmwaru: 4.40 , jejichž umístění v prostoru můžeme vidět na obrázku 5.1. Kamery jsou umístěny na privátní síti a byly ovládány z pracovní stanice u robotického ramene.

## 5.2 Měření

Rychlost reakce byla měřena pomocí zpoždění od zadání pozice po její dosažení. Po zadání souřadnic se zaznamenal v programu čas a po potvrzení dosažení požadovaných pozic se vypočítal uplynulý čas a ten se spolu se souřadnicemi zaznamenal. Tabulku naměřených hodnot lze vidět v příloze B.




Obrázek 5.2: Jednotlivé časy změny pohledu kamer

Jednotlivé časy zpoždění při změně pohledu můžeme také vidět na obrázku 5.2, kde jdou na horizontální ose uvedena pořadí zadávané změny a na vertikální ose čas trvání této změny.

| **Čas [ms]** | **Četnost** |
|---|---|
| 0-1000 | 2 |
| 1000-2000 | 18 |
| 2000-3000 | 9 |

Tabulka 5.1: Rozložení dob změny pohledu

V tabulce 5.1 můžeme pozorovat, že do jedné sekundy trvaly pouze 2 změny, nad 2 sekundy bylo naměřeno 9 změn a nejčastěji byla doba od zadání až po dosažení polohy v intervalu 1 až 2 sekundy.

Důležitá je také přesnost natočení kamer, ta je velice závislá na přesnosti zadaných počátečních dat, a to souřadnicích pozice kamer a základnímu natočení vůči středu souřadného systému. Přesnost může být ovlivněna i nepřesností PTZ systému kamer, například špatnou kalibrací nebo vůlí v převodech servomotorků.

# Kapitola 6

# Překlad knihovny

Knihovnu lze přeložit na různých operačních systémech. Důležité je pouze, aby byli dostupné použité pomocné knihovny FFmpeg a cURL. Po spuštění příkazu `make` a bezchybném přeložení se vytvoří soubor knihovny *libipcam.dll* pro systémy Windows a pro ostatní *libipcam.so*

Na platformě Windows budeme pro překlad potřebovat MinGW[1]

### Překlad použitých knihoven

Použité pomocné knihovny se překládají standardním způsobem a to kombinací tří kroku.

Nejprve je nutne nakonfigurovat instalaci příkazem `./configure`, následně přeložit knihovnu programem `make` a jako poslední zkopírovat hlavičkové soubory do složky „/include/“ a přeložené soubory knihovny do složky „lib/“, kde se nacházejí i další knihovny, příkazem `make install`.

Pro překlad knihovny FFmpeg je dobré mít také překladač YASM, protože některé části jsou psány v assembleru.

[1] Minimalist GNU for Windows - http://www.mingw.org/

# Kapitola 7

# Závěr

Záměrem této práce bylo vytvořit rozšíření knihovny pro ovládání IP kamer. Pomocí jazyka C++ je implementováno rozhraní VAPIX2 kamer AXIS. Knihovna nyní umožňuje aktivaci detekce pohybu a poskytuje základ pro rozšíření dalších podporovaných událostí, dále je možno kamery ovládat nejen zadáním úhlů, ale také pomocí prostorových souřadnic, které knihovna na požadované úhly převede. Z provedených experimentů vyplynulo, že pro zaostření na pozorovaný bod bude nejlepší využít automatické zaostření kamer, které má dobré výsledky.

Z provedených měření také dostáváme obraz o rychlosti změny pohledu, nejčastěji trvá změna od jedné do dvou sekund. Čas potřebný pro nastavení pohledu kamery je také závislý na parametru rychlosti pohybu kamery. Přesnost změny se pak odvíjí od počátečního nastavení pozice kamery a je také ovlivněna nepřesnostmi v pozičním systému kamery.

Knihovnu je možno dále rozvíjet v mnoha směrech. Ke zjednodušení práce se souřadným systémem by například pomohla jednodušší inicializace pozic kamer. Lze zavést větší podporu událostí, například spuštění události při výpadku a znovu zpřístupnění sítě, detekce zvuku, případně definování GuardTour. Popřípadě dodělat podporu pro Vapix 3 nebo kamery s úplně jiným rozhraním jako je PSIA[1] nebo ONVIF[2].

[1] Physical Security Interoperability Alliance
[2] Open Network Video Interface Forum

# Literatura


[1] Axis Communications: Dokumentace VAPIX®. [online], 2007-10-17 [cit. 2013-01-06].
URL http://www.axis.com/techsup/cam_servers/dev/cam_http_api_index.php

[2] Axis Communications: VAPIX®, HTTP API Specification. [online], 2007-10-17 [cit. 2013-01-06].
URL http://www.axis.com/techsup/cam_servers/dev/cam_http_api_2.php

[3] Axis Communications: VAPIX®, Parameter Specification. [online], 2007-10-17 [cit. 2013-01-06].
URL http://www.axis.com/techsup/cam_servers/dev/cam_param_2.php

[4] Axis Communications: VAPIX® version 2, Event Handling API. [online], 2007-10-17 [cit. 2013-01-17].
URL http://www.axis.com/files/manuals/eventHandling100326.pdf

[5] Axis Communications: Axis Development Guidelines. [online], 2010-4-21 [cit. 2013-01-21].
URL
http://www.axis.com/files/tech_notes/development_guidelines_1_01.pdf

[6] Řezka, P.: Přehled středoškolské matematiky. [online], 1999 [cit. 2013-01-06].
URL http://www.sgo.cz/stranky_predmetu/mat/Download/STUDIJNI_MATERIALY/Weinlich/matematika/Prehled_stredoskolske_matematiky_1.pdf

[7] FFmpeg - leading multimedia framework: FFmpeg. [online], [cit. 2013-04-04].
URL http://ffmpeg.org/

[8] Fielding R., Gettys J. a kolektiv: RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1. [online], 1999-06 [cit. 2013-01-12].
URL http://www.ietf.org/rfc/rfc2616.txt

[9] Schmied, J.: *Knihovna pro ovládání IP kamer*. bakalářská práce, FIT VUT v Brně, 2012.

[10] WWW stránky: cURL and libcurl. [online], [cit. 2013-04-04].
URL http://curl.haxx.se/

# Příloha A

# Obsah CD

- bin/ - Adresář se spustitelnými soubory aplikace
- doc/ - Dokumentace k aplikaca
- examples/ – složka s ukázkovými kódy
- libs/ – Adresář se zdrojovými kódy potřebných knihoven
- src/ - Adresář se zdrojovými kódy knihovny
- text/ - Adresář se zdrojovými soubory technické zprávy
- LICENSE – text licence LGPL
- Makefile – soubor makefile pro přeložení knihovny
- README – stručný návod na překlad knihovny
- rozsireni_knihovny_pro_ovladani_IP_kamer.pdf - Technická zpráva ve formátu pdf

# Příloha B

# Naměřené hodnoty rychlosti změny pohledu kamer

| | **Na souřadnice [x, y, z]** | **Doba přesunu [ms]** |
|---|---|---|
| | 0, 0, 0 | |
| 1. | 100, 0, 0 | 1297 |
| 2. | -100, 0, 0 | 2297 |
| 3. | 100, 100, 0 | 2297 |
| 4. | 100, 50, 50 | 2266 |
| 5. | 0, 0, 100 | 1328 |
| 6. | 50, 0, 200 | 1516 |
| 7. | -50, 50, 50 | 1688 |
| 8. | 50, 50, 50 | 1406 |
| 9. | 50, -50, 50 | 1516 |
| 10. | -50, -50, 50 | 1750 |
| 11. | 100, 100, 0 | 1829 |
| 12. | 100, -100, 0 | 1781 |
| 13. | -100, 100, 0 | 2734 |
| 14. | -100, -100, 0 | 1704 |
| 15. | 0, 0, 200 | 1719 |
| 16. | 0, 100, 100 | 1297 |
| 17. | 0, -100, 100 | 1765 |
| 18. | 100, 0, 100 | 2125 |
| 19. | -100, 0, 100 | 1890 |
| 20. | 100, 100, 200 | 1953 |
| 21. | -100, -100, 0 | 2079 |
| 22. | 100, -100, 100 | 2516 |
| 23. | -100, 100, 0 | 2359 |
| 24. | 100, -100, 200 | 2328 |
| 25. | 0, 0, 0 | 1875 |
| 26. | 0, 0, 50 | 1046 |
| 27. | 0, 0, 100 | 891 |
| 28. | 0, 0, 200 | 985 |
| 29. | 0, 0, 0 | 1219 |