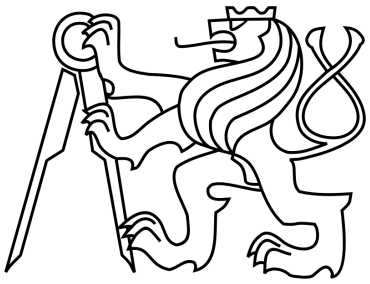
České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra kybernetiky



BAKALÁŘSKÁ PRÁCE

Zpracování obrazu pro vzdálenou vizualizaci

Autor: Miroslav Strob

Vedoucí práce: Ing. Jan Chudoba

Praha, 2012

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Miroslav S t r o b

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Zpracování obrazu pro vzdálenou vizualizaci

Pokyny pro vypracování:

1. Prostudujte metody přenosu obrazu v reálném čase prostřednictvím počítačové sítě.
2. S využitím standardních dostupných open-source nástrojů implementujte programové moduly realizující přenos obrazu z kamer určených pro vizualizaci robotického pracoviště cílovým uživatelům připojeným sítí Internet.
3. Implementujte modul pro odstranění zkreslení a převzorkování obrazu.
4. Realizujte podporu ukládání záznamu z kamer a jejich zpětné prohlížení uživatelem.
5. K projektu vypracujte odpovídající dokumentaci.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí bakalářské práce: Ing. Jan Chudoba

Platnost zadání: do konce zimního semestru 2012/2013

L.S.

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 1. 2012

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

V Praze dne 22. 5. 2012

Miroslav Strob

Název práce: Zpracování obrazu pro vzdálenou vizualizaci

Autor: Miroslav Strob

Katedra: Katedra kybernetiky

Vedoucí bakalářské práce: Ing. Jan Chudoba

e-mail vedoucího: chudoba@labe.felk.cvut.cz

Abstrakt Práce se zabývá částí zpracování obrazu a vizualizace v projektu SyRoTek s použitím síťových kamer ELPHEL. Vizualizaci tvoří živý přenos obrazu sítí Internet společně s dalšími funkcemi jako je pořizování video záznamů a rozhraní pro poskytování jednotlivých snímků protokolem HTTP. Pozornost je věnována korekci soudkovitého zkreslení, které nastává u širokoúhlých objektivů použitých kamer. Vada je odstraněna obrazovou transformací. Programové řešení projektu je založené na open-source frameworku GStreamer. Výsledné navržené řešení umožňuje uživatelům sledovat živý obraz přes síť Internet a pořídit si jeho záznam na serveru, obraz z kamer je možné sledovat i na webových stránkách projektu SyRoTek.

Klíčová slova: zpracování obrazu, kamera ELPHEL, GStreamer

Title: Image Processing for Remote Visualization

Author: Miroslav Strob

Department: Department of Cybernetics

Supervisor: Ing. Jan Chudoba

Supervisor's e-mail address: chudoba@labe.felk.cvut.cz

Abstract This work deals with part of the image processing and visualization in the project SyRoTek using network cameras ELPHEL. Visualization of live video transmission through the Internet together with other functions such as making videos and providing an interface for each snapshot using HTTP. Attention is given to correct the barrel distortion that occurs at wide-angle lenses used cameras. The defect is removed by image transformations. The solution is based on open-source framework GStreamer. The proposed solution allows users to view live stream over the Internet and take his record on the server, the image from the cameras can be monitored on the project website.

Keywords: image procesing, camera ELPHEL, GStreamer

Obsah

Zadání práce ii

Abstrakt iv

1. Úvod 1

2. Použité prostředky 2
2.1. Kamera ELPHEL . . . 2
2.2. GStreamer . . . 4

3. Kalibrace kamery 7
3.1. Teoretický úvod . . . 7
3.2. Camera Calibration Toolbox . . . 8
3.3. Implementace kalibrace v programu . . . 10

4. Návrh řešení 14
4.1. Přenos dat po síti . . . 14
4.2. Ukládání videí . . . 18
4.3. Příjem snímků z kamery a odstranění soudkovité vady . . . 19
4.4. Návrh struktury projektu . . . 19

5. Realizace 20
5.1. Komunikace mezi jednotlivými programy . . . 21
5.2. Image server . . . 22
5.3. RTSP server . . . 26
5.4. Ukládací aplikace . . . 28
5.5. Ovládání aplikace image_server a stream_storage . . . 30
5.6. Spouštění programů . . . 30
5.7. Zdrojové soubory . . . 31

6. Závěr 32
6.1. Velikosti datových přenosů a ukládaných dat . . . 32

Literatura 34

Přílohy I

A. Zdrojový kód RTSP serveru II

B. Adresářová struktura CD přílohy IV

C. Komunikace s GST-RTSP serverem V

1. Úvod

Práce se věnuje zpracování obrazu a vzdálené vizualizaci projektu SyRoTek (System for a robotic e-learning) zabývající se výukou mobilní robotiky a umělé inteligence na katedře kybernetiky FEL. Základem projektu je hřiště o rozměrech 3,5 × 3,8 m vybavené pohyblivými překážkami, na kterém se pohybují roboty. Systém je přizpůsobený pro vzdálený přístup přes Internet. Uživatel (student) může vzdáleně používat skutečné roboty pro řešení úloh z mobilní robotiky a umělé inteligence, což přináší oproti simulacím výhodu skutečných podmínek jako je šum snímačů, neznalost všech parametrů okolního prostředí, nejistotu apod. Výhoda projektu SyRoTek spočívá v možnosti zaznamenávat data ze snímačů a polohy robotů. Záznamy je možno následně použít jak ke kontrole splnění úlohy učitelem, tak i k následnému posouzení postupu řešení úlohy. Další informace o funkcích a poslání projektu lze nalézt v [8] a v [16].

Každý robot je vybaven akumulátorem, pohonem, modulem pro komunikaci a snímači. Dobíjení akumulátorů probíhá přímo na hřišti na určených místech s vyvedenými kontakty. Roboty jsou vybaveny bezdrátovým Wi-Fi komunikačním modulem pro komunikaci s hlavním počítačem, který umožňuje roboty ovládat na dálku přes Internet. Podrobnější technická specifikace robotů je uvedena v [9]. Hřiště, na kterém se pohybují je snímané několika kamerami, jedna kamera umístěná přímo nad hřištěm slouží k lokalizaci jednotlivých robotů. Tato kamera snímá hřiště a určuje polohu jednotlivých robotů na základě značky umístěné na každém robotu. Další kamery jsou určené pro vizualizaci.

Pro realizaci úloh budou uživatelé využívat senzorová data z robotů a jejich polohu určenou lokalizací. Tyto data je vhodné doplnit vizualizací reálné scény s roboty, aby byl umožněn přímý náhled na dění na hřišti. To znamená zajistit obrazový proud z kamer poskytovaný přes Internet s využitím vhodného přenosového protokolu a komprese. Podmínka živého přenosu videa u vizualizace s sebou samozřejmě přináší požadavky především na minimální dobu zpoždění, nízký datový tok a spolehlivost přenosu. Musí být zvolena nejen metoda komprese obrazu, ale i způsob přenosu dat sítí Internet. Hřiště je rozsáhlé a uživatel by měl mít možnost detailně sledovat jen část hřiště. Dále je kladen požadavek na pořizování záznamů z kamer při probíhající práci s roboty, tyto záznamy dále archivovat pro pozdější potřeby. Rezervační systém přiděluje uživatelům na práci bloky o délce 30 minut, to je třeba respektovat při realizaci ukládání. Jelikož se hřiště nachází ve školní učebně je zapotřebí vyřešit, z důvodu dohledu, i snímání hřiště během celého dne mimo samotných prací na úlohách. Je řešená i otázka prezentace projektu SyRoTek, ta by měla spočívat v možnosti sledovat formou webové kamery dění na hřišti na webových stránkách projektu [1].

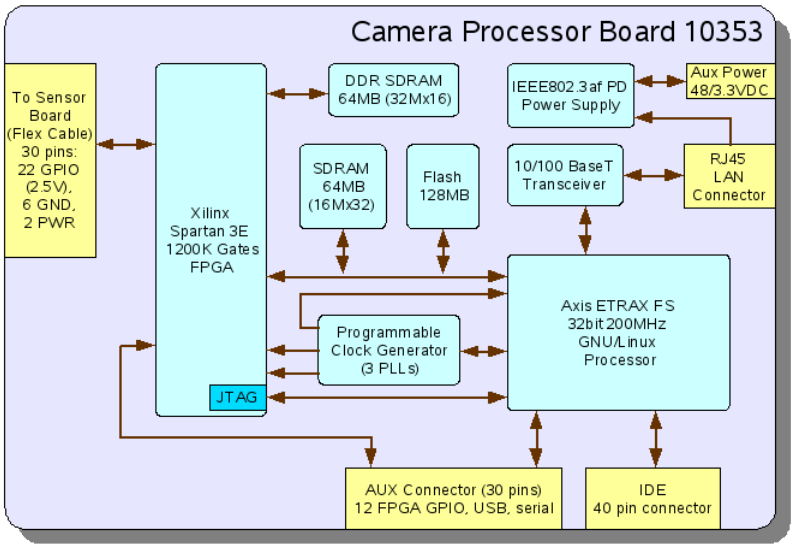
Řešení těchto dílčích úloh by mělo tvořit kompaktní programový celek s možností snadného ovládání a přizpůsobení jednotlivých parametrů jako jsou jednotlivé obrazové formáty, volba umístění nahrávaných záznamů apod. Předpokládá se navržení jednotného řešení, které může být použito duplicitně pro více než jednu kameru pouze s obměnou výše uvedených parametrů.

V jednotlivých kapitolách bude postupně probírán postup vývoje projektu včetně teoretických informací o problematice datových přenosů, zpracování obrazu a obrazových transformací. Ve druhé kapitole jsou popsány použité prostředky k řešení projektu – kamery ELPHEL a multimediální open-source framework GStreamer, na kterém je založena většina programového řešení projektu. Následující třetí kapitola je věnována problematice zkreslení objektivů kamer a jeho odstranění, výsledkem je navržení obrazové transformace, která je implementována v programech. Čtvrtá a pátá kapitola se věnuje návrhu a řešení jednotlivých částí vyplývajících ze zadání, tyto části jsou seskupeny do jednotného celku tvořící výslednou podobu projektu.

2. Použité prostředky

2.1. Kamera ELPHEL

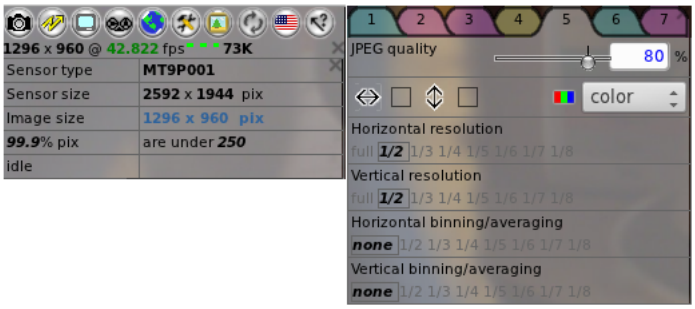
Kamera ELPHEL 353L je kamera snímající ve vysokém rozlišení až 2592×1944 px (projekt pracuje s rozlišením zhruba poloviční – 1296×960 px). Specifikace kamery, návody a další jsou uvedeny na internetových stránkách výrobce [3]. Hlavní část zpracování obrazu probíhá na hradlovém poli FPGA Spartan 1200K připojeném přímo ke snímacímu prvku kamery. Na hradlovém poli probíhá komprese do formátu JPEG. Dále je v hradlovém poli implementován komunikační modul pro vlastní snímač. Dále obsahuje počítačovou část obsahující procesor Axis s vestavěným Linuxem. V systému kamery je spuštěno několik programů, v projektu se pracuje s aplikací `imgsrv`, která slouží jako webový server poskytující snímky z kamery a program `str` jako RTSP server. Počítač obsahuje též vestavěný webový server včetně podpory PHP. To dovoluje přistupovat k nastavení kamery pomocí internetového prohlížeče a snadno měnit nastavení kamery, popřípadě rozšířit o vlastní webovou aplikaci. Kamera je připojena do sítě LAN, je jí přidělena vlastní IP adresa a spojení s kamerou je možné přes protokol SSH. V kamerách jsou v současné době firmware verzí 8.1.1 a 8.0.9.



Obrázek 2.1.: Blokové schéma kamery, převzato z [3]

2.1.1. Nastavení kamery

Před použitím je třeba kameru vhodně nastavit. K tomuto účelu slouží připravené webové rozhraní, které je přístupné internetovému prohlížeči na IP adrese kamery. Hlavní části jsou – *Parameter Editor* a *Camera Control Interface*. Editor nastavení obsahuje nastavení všech parametrů kamery. Pro účely streamování RTSP protokolem postačí nastavit hodnoty týkající se streameru – záložka `streamer` v editoru.



Obrázek 2.2.: Nastavení hodnot streameru v kameře – CCI

Aby kamera vysílala RTSP unicast stream na její IP adrese a portu 554 je třeba zakázat multicast `STROP_MCAST_EN=0` a povolit samočinné spuštění streameru po restartu kamery hodnotou `DAEMON_EN_STREAMER=1` ; nastavení dalších hodnot je zřejmé z krátké nápovědy k těmto hodnotám připojené, vztahují se především na nastavení multicast streamovaní a přenos zvuku. Hodnoty nastavení v *Parameter Editor* je možné uložit – platné nastavení kamery pro projekt je uloženo jako „Výchozí nastavení“.

Další nastavení je třeba provést v *Camera Control Interface*. Jedná se o zcela grafické rozhraní umožňující intuitivně nastavit parametry výstupního obrazu. Současné nastavení je na obrázku 2.2. Základní rozlišení je hodnotami *Horizontal* a *Vertical resolution* zmenšené na polovinu – 1296×960 px. Souhrnné informace o stavu kamery jsou v horní části obrázku 2.2 – nastavené rozlišení a snímková rychlost. Zelený obrázek vedle snímkové rychlosti indikuje stav snímání kamery. V rozhraní je dále možné manipulovat s vyvážením barev a dalšími filtry.

2.1.2. Programy spuštěné v kameře

RTSP streamer

V kameře se jedná o program `str`, který je po nastavení kamery samočinně spouštěn po jejím zapnutí. Jedná se o RTSP streamovací server, který čte JPEG snímky z vnitřního bufferu kamery a streamuje je přes RTSP protokol jako MJPEG[1]. Může pracovat v unicast i multicast režimu, kamera je nastavena na unicast stream, to znamená, že streamer může v jeden čas odesílat data pouze jednomu přijemci. Infomace o multicastu lze získat např v [4] nebo v části 4.1.1. RTSP protokol bude popsán v kapitole 4.

Image server

Image server je HTTP server, který naslouchá na portu 8081 a slouží k přenosu jednotlivých JPEG snímků z vnitřního bufferu kamery přes HTTP protokol. V dokumentaci [3] jsou veškeré informace pod položkou *imgsrv*.

Základní přístup ke snímku je na URL `http://<camera_ip>:8081/bimg/`. Přenese se nejnovější snímek z pomocného bufferu – nehrozí smíchání snímků při pomalejším přenosu. Při nezdaru vrací server obrázek GIF o velikosti 1×1 px. Komunikace se serverem protokolem HTTP je na výpisu 2.1.

[1] Motion JPEG – sekvence JPEG obrázků.

Výpis 2.1: Navázání komunikace se serverem protokolem HTTP

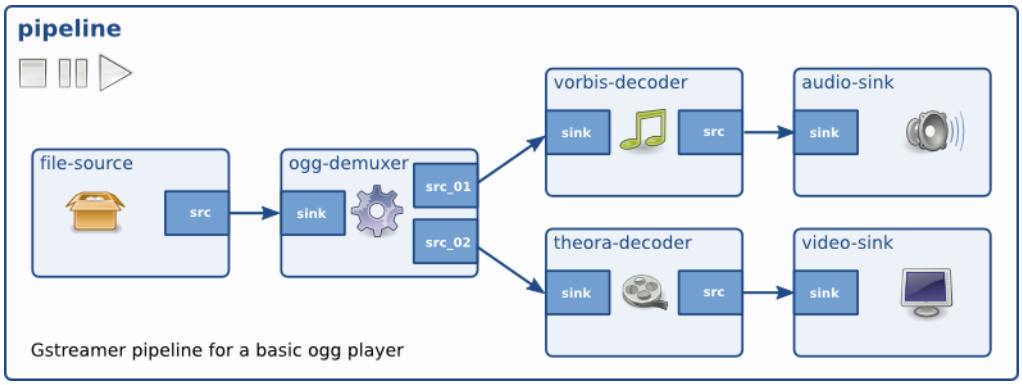
```
GET /bimg/


HTTP/1.0 200 OK
Server: Elphel Imgsrv
Expires: 0
Pragma: no-cache
Content-Type: image/jpeg
Content-Disposition: inline; filename="elphelimg_79327.jpeg"
Content-Length: 80369

JPEG data ...
```

Obrázek 2.3.: Kamera ELPHEL 353L, převzato z [3]

2.2. GStreamer

GStreamer [2] je volný multimediální framework[2] vyvíjený pro systémy GNU/Linux využívající knihoven GLib 2.0. GStreamer spolu s jeho pluginy je k dispozici pro Linux ve standardních balíčcích. V současné době je využíván GStreamer ve verzi 0.10.

Struktura GStreameru je založena na tzv. „pipeline“ – pojem představuje myšlenku, že data ať obrazová nebo zvuková „procházejí“ jakýmsi „potrubím“. Jedná se o řetězec navzájem propojitelných prvků „Elements“, místo spojení je nazýváno jako „Pads“. Příklad principu pipeline je na obrázku 2.4, kde je realizován videopřehrávač.



Obrázek 2.4.: Příklad pipeline jako OGG přehrávače, převzato z [2]

[2]Balík programového vybavení sloužící jako podpora při vývoji aplikací, obsahuje pomocné programy, knihovny apod.

2.2.1. GST Elements

Jednotlivé elementy jsou základním stavebním prvkem každé pipeline. V obrázku 2.4 se jedná o modré rámečky. Na základě možností datových vstupů/výstupů elementů je možné je rozdělit na:

- Zdrojové elementy – Source elements
- Filtry, převodníky, multiplexory, demultiplexory a kodeky
- Výstupní elementy – Sink elements

Zdrojové elementy

Zdrojové elementy tvoří zdroj signálu pro následují prvky pipeline. Umožňují číst ze souboru, TCP, UDP a RTSP protokolů apod. Zdrojový element nepřijímá žádná data, je pouze jejich zdrojem.

Filtry, převodníky, multiplexory, demultiplexory a kodeky

Filtry, převodníky, multiplexory, demultiplexory a kodeky zpracovávají vstupní data, které obdrží na „Sink Pads“ a výstupní data poskytují na „Source Pads“. Nejčasteji se jedná o filtry a převodníky. Mohou však mít i více vstupů/výstupů, pak se jedná o multiplexory/demultiplexory nebo kodeky.

Výstupní elementy

Výstupní elementy jsou konečnou částí pipeline. Mohou sloužit k přímé vizualizaci dat na monitoru, ukládat do souboru nebo je streamovat do sítě.

2.2.2. GST Pads

Jedná se o místa napojení dílčích elementů pipeline. Každý „Pad“ pracuje s definovaným formátem dat. V případě obrazové informace se jedná především o RGB a YUV model. Dále již obraz může být zakódován komopresí JPEG, MPEG, Theora apod. Možnosti vstupu a výstupu elementů je možné zjistit příkazem `gst-inspect`. Příklad výpisu části týkající se možností vstupu/výstupu elementu je na obrázku 2.2. Příkazem `gst-inspect` je dále možné zjistit parametry, kterými se dá ovlivnit chování elementu apod. Podrobná dokumentace všech elementů je k dispozici v [2]. Element může podporovat více formátů, o správný výběr a propojení se stará vnitřní mechanismus GStreameru.

Výpis 2.2: Výpis informací o GST elementu

```
Pad Templates:
  SRC template: 'src'
    Availability: Always
    Capabilities:
      video/x-raw-yuv
        format: { UYVY }
        width: [ 1, 2147483647 ]
        height: [ 1, 2147483647 ]
        framerate: [ 0/1, 2147483647/1 ]
  SINK template: 'sink'
    Availability: Always
    Capabilities:
      video/x-raw-rgb
```

```
      bpp: 24
      depth: 24
      endianness: 4321
      red_mask: 16711680
      green_mask: 65280
      blue_mask: 255
      width: [ 1, 2147483647 ]
      height: [ 1, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
```

Standardní formáty jako RGB a YUV mohou mít další parametry kromě šířky, výšky a snímkové rychlosti. U YUV formátu se jedná především o řazení jasových a barvonosných složek, u RGB o počet bitů na barvu. Popis a vysvětlení spolu s příslušnou značkou *FOURCC* (na výpisu 2.2 je u YUV uveden např. UYVY) nalezneme na `http://www.fourcc.org/`. Správná znalost těchto formátů je důležitá pro naprogramování vlastního elementu.

2.2.3. Návrh GST aplikací

Aplikace `gst-launch` umožňuje spustit vlastní pipeline, aplikace je součástí standardní instalace GStreameru. Do programu vstupuje pipeline jako parametr, jednotlivé elementy jsou od sebe odděleny vykřičníky. Základní zápis představuje následující příklad:

```
gst-launch videotestsrc ! video/x-raw-yuv,width=640,height=480,framerate=15/1
! textoverlay text="Zkouska \!" ! ffmpegcolorspace ! ximagesink
```

Element `videotestsrc` je zdrojem zkušebního obrazce – monoskop; část
`video/x-raw-yuv,width=640,height=480,framerate=15/1`
je tzv. „CapsFilter“, ten vynutí konkrétní formát dat na spoji elementů; `ffmpegcolorspace` slouží k převodu mezi jednotlivými obrazovými modely – dokáže převádět mezi RGB a YUV včetně jejich různých podob podle *FOURCC*.

Například pro příjem unicast RTSP streamu z kamery vypadá pipeline následovně:

```
gst-launch-0.10 rtspsrc location=rtsp://10.10.20.13:554
! rtpjpegdepay ! jpegdec ! ffmpegcolorspace ! xvimagesink
```

Ne všechny elementy jsou obsaženy v základní instalaci GStreameru, existují mnohé další, které jsou součastí GST pluginů. Jsou rozděleny do následujících balíčků `gst-plugins-base`, `gst-plugins-good`, `gst-plugins-ugly`, `gst-plugins-bad`. Jak již názvy napovídají rozdělené jsou podle kompatibility a spolehlivosti při chodu v aplikaci. Informace o elementech obsažených v jednotlivých pluginech nalezneme v [2].

Další možností je naprogramování vlastní aplikace nebo vlastního GST elementu na základě předpřipravených kódů. Jedná se o šablonu dostupnou na stránkách projektu GStreameru [2], konkrétně `git://anongit.freedesktop.org/gstreamer/gst-template.git`.

Aplikace umožňuje sestavit GST pipeline, řídit její chod, měnit parametry apod. Aplikace je psána v jazyce C, dá se jednoduše doplnit vlastním kódem a vytvořit tak vlastní program – na GStreameru je založeno několik přehrávačů v OS Linux. Šablona obsahuje také prostředky k vytvoření vlastního elementu. Na webu GStreameru je k dispozici podrobný návod popisující tvorbu jednotlivých programů a elementů, výsledku je možné dosáhnout i modifikací obsažených příkladů.

3. Kalibrace kamery

Kamery jsou vybaveny širokoúhlými objektivy, u nich dochází k tzv. soudkovité vadě, která je způsobena nestejnoměrným zvětšením čočky objektivu. Zkreslení v tomto případě můžeme rozdělit na dvě hlavní části, a to na zkreslení radiální – symetrické vůči hlavnímu bodu kamery (optická osa objektivu) a zkreslení tangenciální – kolmé na radiální. Největší význam má zkreslení radiální dané měnícím se zvětšením od optického středu objektivu. Zkreslení se mění v závislosti na ohniskové vzdálenosti a zaostření obrazu.

Tuto vadu je nutné odstranit, aby se obraz jevil jako rovinný. K tomu je třeba navrhnout vhodnou obrazovou transformaci a identifikovat její parametry. Pro identifikaci parametrů a navržení transformace slouží plugin programu MATLAB *Camera Calibration Toolbox for Matlab*.

3.1. Teoretický úvod

Obrázek 3.1 znázorňuje zjednodušené schéma kamery. Podle [14] je prostor rozdělený na dvě části – kamerový prostor, popsán souřadnicemi $\mathbf{x}_c = x_c,\ y_c,\ z_c$, a obrazovou rovinu, jako místem projekce obrazu na snímací prvek popsán souřadnicemi $\mathbf{u}_c$. Ohnisková vzdálenost je značena jako f. Projekci na obrazovou rovinu můžeme popsat v homogenních souřadnicích následovně:

$$\mathbf{u}_c = \left[-\frac{fx_c}{z_c};\ -\frac{fy_c}{z_c};\ 1\right]$$

S ohledem na odstranění radiálního zkreslení je podle [15] správná pozice pixelu v kamerových souřadnicích dána vztahem 3.1 s radiální vzdáleností od středu $r = \sqrt{x_c^2 + y_c^2}$.

$$\begin{aligned} x'_c &= x_c\left(1 + K_1 r^2 + K_2 r^4 + K_3 r^6\right) \\ y'_c &= y_c\left(1 + K_1 r^2 + K_2 r^4 + K_3 r^6\right) \end{aligned} \tag{3.1}$$

Pro tangenciální zkreslení platí vzorec 3.2.

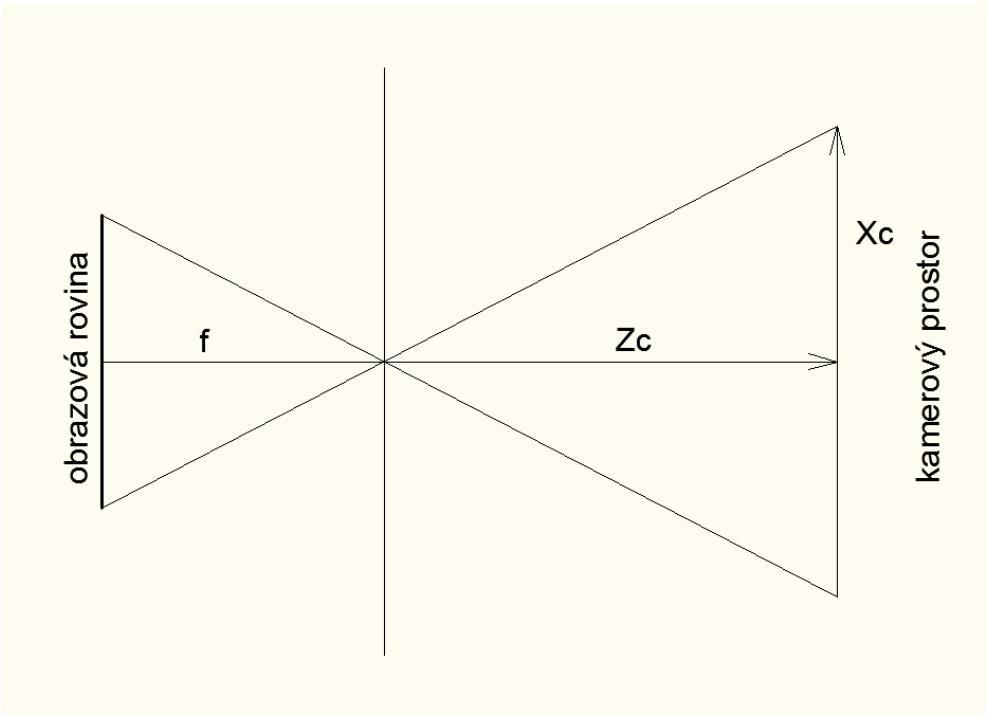
$$\begin{aligned} x'_c &= x_c + K_4\left(r^2 + 2x_c^2\right) + 2K_5 x_c y_c \\ y'_c &= y_c + K_5\left(r^2 + 2y_c^2\right) + 2K_4 x_c y_c \end{aligned} \tag{3.2}$$

Konstanty K_1, K_2, K_3, K_4, K_5 je třeba získat *Camera Calibration Toolboxem*.

3.1.1. Extrinsické a intrinsické parametry kamery

Extrinsické (vnější) parametry popisují vztah mezi světovou souřadnou soustavou a soustavou kamerových souřadnic např. pomocí rotační matice a vektoru posunutí. V *Camera Calibration Toolboxu* jsou extrinsické parametry obsaženy ve výstupu pro každý kalibrovaný snímek (složky x, y světové souřadné soustavy jsou vyznačeny při kalibraci snímku, příkladem je obrázek 3.2c). Určují tedy jen umístění kamery ve zvolené soustavě světových souřadnic a nebudou již dále zmiňovány.

Mezi intrinsické (vnitřní) parametry patří ohnisková vzdálenost, hlavní bod (místo protnutí optické osy objektivu se zobrazovací rovinou) a velikost snímku. Pomocí těchto parametrů je



Obrázek 3.1.: Zjednodušené schéma optiky kamery

možné převést bod z kamerového prostoru na obrazovou rovinu za pomoci kalibrační matice následovně:

$$z_c \mathbf{u}_c = \begin{bmatrix} -fa & -fc & u_0 \\ 0 & -fb & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Parametry u_0, v_0 odpovídají hlavnímu bodu kamery, dále parametr c odpovídá koeficientu zkosení, parametry a, b modifikují ohniskovou vzdálenost f. Poslední 3 parametry se uplatní např. při kalibraci kamery s *Camera Calibration Toolbox* v případě, že snímací prvek není pravoúhlý nebo jednotlivé pixely nemají přesně čtvercový tvar.

3.2. Camera Calibration Toolbox

Camera Calibration Toolbox je plugin pro program MATLAB určený ke kalibraci kamer. Návod na instalaci pluginu, jeho používání a další informace jsou uvedeny na webových stránkách `http://www.vision.caltech.edu/bouguetj/calib_doc/`. Vlastní kalibrace se provádí nasnímáním dostatečného počtu snímků (10 – 20) se šachovnicí snímanou z různých úhlů tak, aby se zřetelně projevila optická vada. Rozměry šachovnice a velikosti políček nejsou předepsány, vstupují jako parametry do kalibrace. Snímky musejí být kontrastní, nerozmazané, aby algoritmus toolboxu mohl spolehlivě detekovat vrcholy čtverců. Snímky musí mít stejný rozměr jako velikost snímku se kterou se pracuje v projektu (1296×960 px). Ukázku snímků představují obrázky 3.2. Pro provedení kalibrace je možné exportovat průběh kalibrace tlačítkem `Save`, dále se ke kalibraci vrátit volbou `Load` a exportovat kalibrační data do souboru `Calib_Results.m` volbou `Export calib results`.

Soubor `Calib_Results.m` obsahuje extrinsické data pro každý snímek, ale hlavně parametry intrinsické. Část souboru obsahující intrinsické parametry je na výpisu 3.1. Hodnoty vektoru `kc` odpovídají koeficientům radiálního a tangenciálního zkreslení podle vztahů 3.1 a 3.2. Podrobný popis transformace a význam jednotlivých parametrů je uveden v [5].

Výpis 3.1: Část souboru `Calib_Results.m`

```
%— Focal length:
fc = [ 849.276117068398090 ; 857.029873076937860 ];
%— Principal point:
cc = [ 647.500000000000000 ; 479.500000000000000 ];
%— Skew coefficient:
alpha_c = 0.000000000000000;
%— Distortion coefficients:
kc = [ -0.398662913858983 ; 0.144189155960651 ;
       -0.003418160123165 ; -0.004032085725116 ;
       0.000000000000000 ];
```

Tyto vnitřní parametry je nutné přepsat do souboru `KC.mat`, odkud se budou načítat do jednotlivých programů. Pořadí hodnot v souboru `KC.mat` je podle následujícího výpisu (3.2).

Výpis 3.2: Struktura souboru KC.mat

```
cc(1)
cc(2)
fc(1)
fc(2)
kc(1)
kc(2)
kc(3)
kc(4)
kc(5)
```

Poznámka k výstupu kalibrace

Z porovnání obrázků 3.2a a 3.2b je zřejmé, že po kalibraci s parametry z toolboxu je obraz oříznut, aby neobsahoval prázdné plochy a vlivem silného zkreslení chybí podstatná část obrazové informace na krajích hřiště. Z tohoto důvodu je v kódu programu předřazena ještě afinní obrazová transformace, která umožňuje měnit měřítko, natočení, posunutí a zkosení obrazu.

Při práci v homogenních souřadnicích má transformace tvar $P' = \mathbf{A}P$, kde $P = [x, y, 1]$ je transformovaný bod a $\mathbf{A}$ je transformační matice 3×3.

Pro posunutí o vektor $[x_0, y_0]$ je matice ve tvaru:

$$\mathbf{A}_T = \begin{pmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotaci o úhel α:

$$\mathbf{A}_R = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Změnu měřítka:

$$\mathbf{A}_S = \begin{pmatrix} \beta & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Transformace je možné skládat postupným násobením transformačních matic, záleží však na jejich pořadí. V projektu je afinní transformace představena souborem `KK.mat`, který obsahuje inverzní matici k navržené transformaci. Příklad souboru `KK.mat` a jeho formát je na následujícím výpisu (3.3).

Výpis 3.3: Soubor `KK.mat`

```
 1.2261000e+00   -0.0088000e+00   -1.3497000e+00
 0.0000000e+00    1.2069000e+00   16.8963000e+00
-0.0001000e+00    0.0000000e+00    1.0006000e+00
```

Návrh afinní transformace

Obrázky 3.3a a 3.3b znázorňují návrh afinní transformace volně šiřitelným grafickým editorem GIMP. Metoda návrhu a spočívá nejprve ve zmenšení snímku – soubor `KK.mat` je ve tvaru transformační matice změny měřítka s koeficientem přibližně 0,8; po inverzi má pak matice na prvních dvou prvcích úhlopříčky hodnotu $0,8^{-1} = 1,25$. Takto zmenšený snímek je upraven nástrojem „Perspektiva“ v GIMPu podle obrázku 3.3a. Program přímo zobrazuje transformační matici. Označíme-li tuto matici $\mathbf{A}_G$, tak výslednou transformační matici, kterou by měl obsahovat soubor `KK.mat`, dostaneme podle vztahu 3.3.

$$\mathbf{A}_{výsledná} = \left(\begin{pmatrix} 0,8 & 0 & 0 \\ 0 & 0,8 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{A}_G \right)^{-1} \tag{3.3}$$

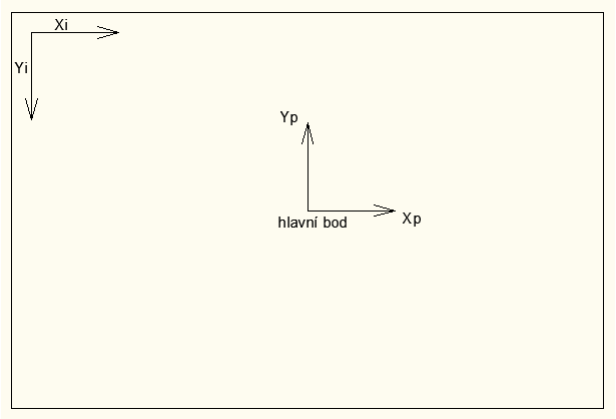
3.3. Implementace kalibrace v programu

Každý snímek je v napsaných programech reprezentován jako pole osmibitových čísel, kde na jeden obrazový bod připadá trojice čísel reprezentující RGB složky (24 b), při používaném rozlišení se jedná o celkovou délku pole $3 \times 1296 \times 960 = 3\,732\,480$ B. Princip obrazové transformace je, že každému bodu přisuzuje jiný bod ze stejného snímku, jelikož 3 barvonosné složky nemá smysl od sebe oddělovat, je počet zpracovávaných bodů roven $1296 \times 960 = 1\,244\,160$.

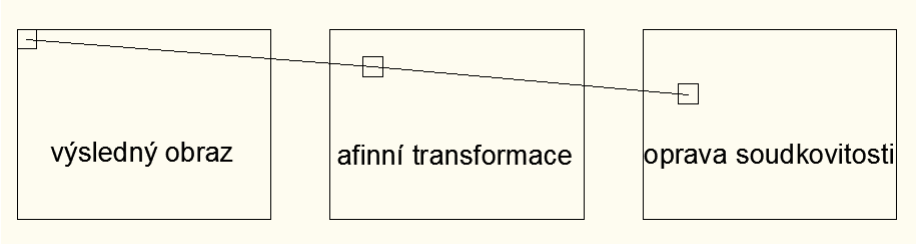
V programu je hleděno především na efektivitu a rychlost provádění obrazové transformace. Při tomto rozlišení a rychlosti přibližně 10 snímků za sekundu není vhodné pro každý obrazový bod snímku počítat jakoukoli složitější operaci. Proto je transformace předem spočítána do podoby transformační tabulky, která každému bodu přisuzuje jeho správnou pozici.

Tabulka má též podobu pole o stejné délce, její prvky musejí být datového typu, který dokáže pokrýt rozsah hodnot od 0 do 1 244 159 – v kódu je použit `integer`, protože každý prvek obsahuje pozici pixelu ve vstupním obraze – pozicí se rozumí pořadí pixelu od levého horního rohu přes řádky až k danému pixelu počínaje od 0. Zde je třeba upozornit na různé přístupy k určení umístění pixelu. Podle obrázku 3.4 se musí rozlišovat souřadný systém kamerových/obrazových souřadnic s počátkem v hlavním bodě a pixelovými souřadnicemi s počátkem v levém horním rohu snímku. Označíme-li obrazové souřadnice x_i, y_i a pixelové x_p, y_p, tak s hodnotou hlavního bodu určenou při kalibraci hodnotami vektoru `cc` platí převod 3.4.

$$\begin{aligned} x_i &= x_p - cc\,(1) \\ y_i &= -\ (y_p - cc\,(2)) \end{aligned} \tag{3.4}$$



Obrázek 3.4.: Souřadnice užité při snímku



Obrázek 3.5.: Pořadí jednotlivých transformací

Výpočet tabulky je dán složením afinní transformace s maticí podle souboru `KK.mat` a transformace navržené kalibrační toolboxem, obrázek 3.5. Smyslem je, aby hodnoty transformační tabulky odkazovali na pozice odpovídajících pixelů ve vstupním obrazem a mohl se pro složení výstupního obrazu použít nasledující úsek C kódu:

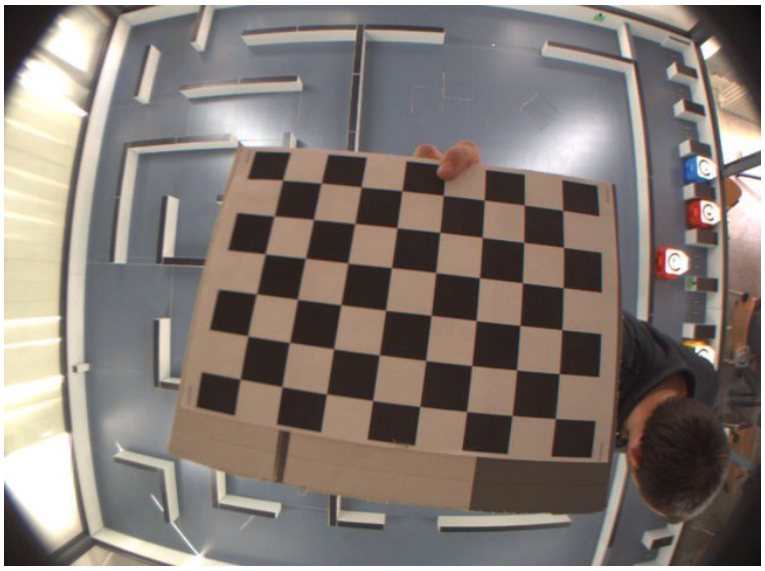
```
vystupni_snimek[index_pixelu] = vstupni_snimek[tabulka[index_pixelu]]
```

Transformace implementovaná v kódu podle kalibračního toolboxu je popsána a vysvětlana v dokumentaci k toolboxu [5] a odpovídá vztahům 3.1 a 3.2. Transformace přiřazuje pixelu pixel, který by měl být na jeho místě nebýt zkreslení soudkovitou vadou. Z důvodů popsaných výše je před tuto transformaci vložena ještě afinní transformace.

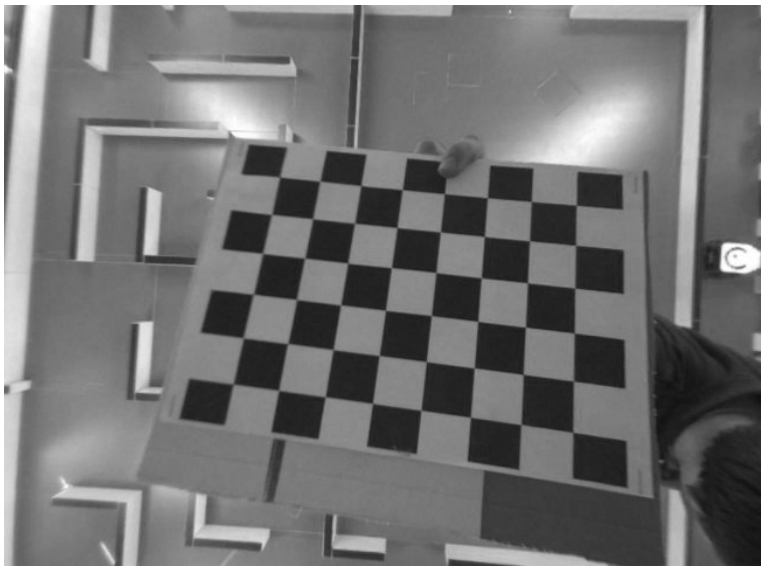
Bilineární interpolace

Pokud by byl obraz složen podle výše uvedeného principu s tabulkou tvořenou celými čísly, docházelo by ke ztrátě přesnosti, protože hledaný pixel obecně nebude ležet přesně na pixelu ve vstupním obraze, ale podle výpočtu bude ležet někde mezi čtyřmi okolními. Proto je v kódu ještě implementována bilineární transformace. Ta vytvoří výsledný pixel váhováním hodnot okolních pixelů a obraz se poté jeví kvalitnější. Princip užité bilineární transformace je naznačen na obrázku 3.6 a pro jednotlivé barvonosné složky pixelu platí vztah 3.5.

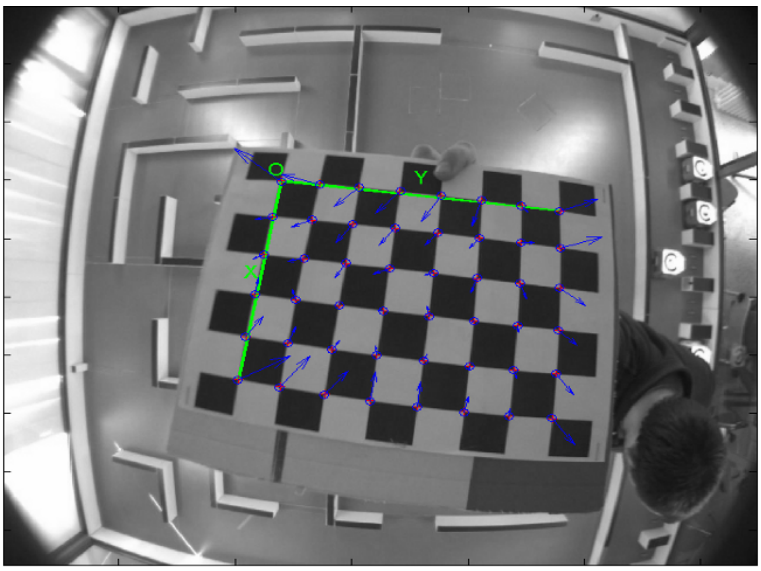
$$P_{R,G,B} = ((1-h_x)(1-h_y))\,P_1 + (h_x(1-h_y))\,P_2 + ((1-h_x)\,h_y)\,P_3 + (h_x h_y)\,P_4 \qquad (3.5)$$



(a) Snímek použitý pro kalibraci

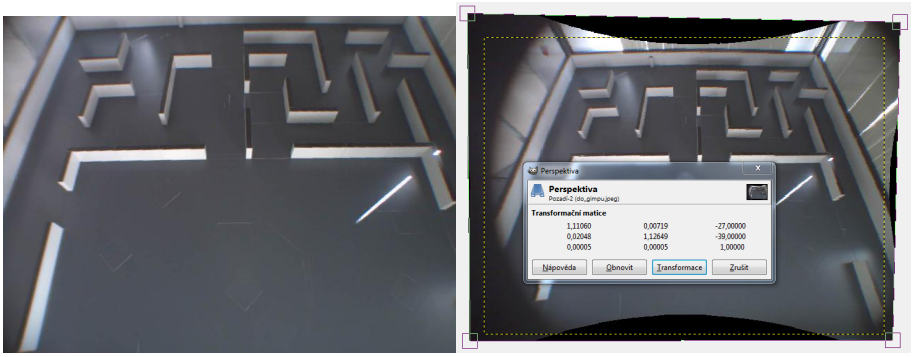


(b) Po odstranění vady kalibračním toolboxem

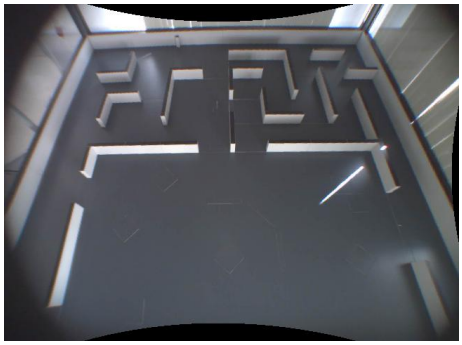


(c) Detekce vrcholů

Obrázek 3.2.: Ukázka kalibračních snímků

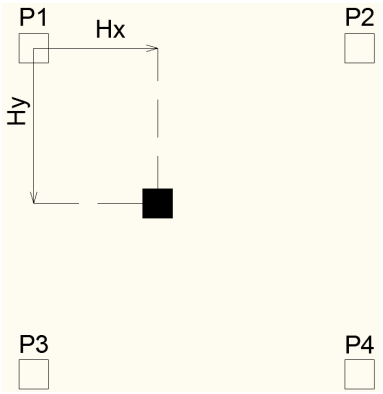


(a) Původní snímek a následná úprava v programu GIMP



(b) Výsledný snímek

Obrázek 3.3.: Návrh parametrů pro afinní transformaci



Obrázek 3.6.: Bilineární transformace

4. Návrh řešení

Ze zadání vyplývají následující požadavky na funkci celého projektu:

- Odstranění vady soudkovitosti obrazu
- Poskytovat živý video stream pro uživatele s možností výřezů obrazu
- Ukládání videí po 30 minutových úsecích
- Možnost sledovat hřiště na webových stránkách projektu
- Ukládání snímků pro dohled nad hřištěm

Jako hlavní prostředek pro realizaci projektu je předpokládán framework GStreamer, ten ale nenabízí přímočaré řešení všech jednotlivých částí, proto je nutné prodiskutovat jednotlivé úkoly, k nim navrhnout možná řešení (optimálně za co největší účasti GStreameru), dále navrhnout vhodný přenos dat přes síť Internet. Z návrhu jednotlivých částí by měl vzniknout kompaktní celek splňující všechny zadané vlastnosti.

Z celkového hlediska jsou nejproblematičtější tyto části:

- Přenos dat mezi jednotlivými částmi projektu a přes síť Internet, především jak vyřešit přenášení živého datového toku k uživatelům.
- Ukládání souborů po 30 minutových celcích. Samotné ukládání je realizovatelné GStreamerem, ale je nutné dořešit dělení souboru.
- Příjem snímků z kamery. Je nutné zvolit přenosový protokol pro příjem dat z kamery, programově ho implementovat a uvážit do jaké části po příjmu začlenit algoritmus odstranění soudkovitého zkreslení.

4.1. Přenos dat po síti

Důležitou částí projektu je vyřešení přenosu obrazových dat nejen přes síť Internet, ale i mezi jednotlivými bloky projektu. Přenos mezi jednotlivými bloky musí být rychlý, spolehlivý a především by měl vyžadovat minimum systémových prostředků. Přenos obrazu k uživatelům by měl splňovat parametry živého přenosu – tedy minimální dobu zpoždění a nízký datový tok dosažený složitější kompresí videa.

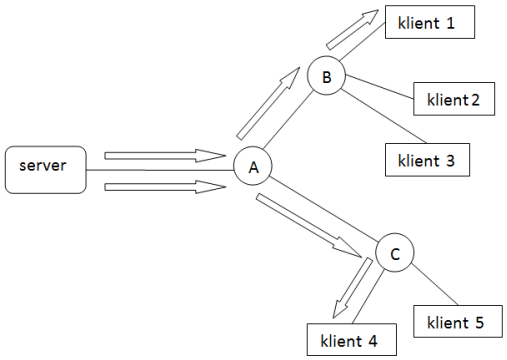
4.1.1. Unicast / Broadcast / Multicast

Unicast označuje metodu zasílání paketů mezi dvěma body v síti – odesílatelem a příjemcem. Paket je směřován na IP adresu příjemce náležícím buď k lokální nebo veřejné síti. V případě potřeby zasílání dat více příjemcům od stejného odesílatele jsou pakety duplikovány a ke každému příjemci vede samostaný proud dat, síť je pak nadbytečně zatěžována. Situace je naznačena na obrázku 4.1.

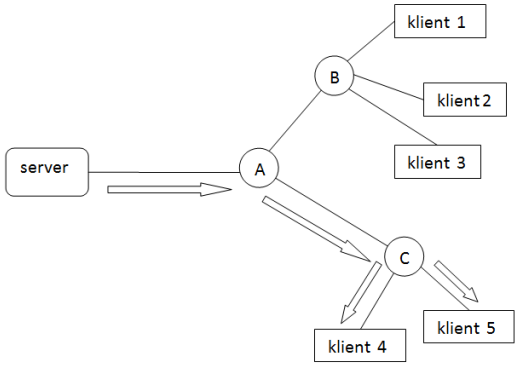
Broadcastový paket je stejný jako unicastový, ale směrovaný na broadcastovou IP adresu sítě. Takto zaslaný paket je routerem zaslán všem počítačům náležících dané podsíti. Ethernetová adresa se zkládá ze 48 bitů (fyzická MAC adresa). Broadcastová MAC adresa vznikne nastavením všech bitů adresy na 1. Každý počítač rozezná paket směrovaný na jeho MAC adresu a též

akceptuje adresu broadcastovou. Broadcast zabírá značnou šířku pásma, pokud nejsou vhodně voleny podsítě omezující šíření broadcastových paketů. Na obrázku 4.2 jsou data posílána na multicastovou adresu routeru C – daty je poté zaplněna celá podsíť tvořená tímto routerem.

Multicast využívá skupinového adresování. Každé zařízení má možnost se přihlásit do multicastové skupiny. Následně jsou data přenášena sítí optimální trasou směrem k příjemcům. Na místech větvení jsou pakety duplikovány a nedochází k přehlcení sítě. V Ethernetu je multicastová IPv4 adresa skupiny v rozsahu 224.0.0.0 – 239.255.255.255. K přihlašováním do multicast skupin se používá protokol IGMP (Internet Group Management Procol). Routery zapojené do multicastu periodicky zjišťují protokolem IGMP zdali v lokálních sítích stále existují příjemci připojeni do multicastových skupin, pokud na dotaz neodpoví žádný příjemce, router se odhlásí z příjmu. Z principu fungování multicastu je možné provozovat jen protokol UDP, který nezaručuje správné přijetí paketů. Na obrázku 4.3 server odesílá data do multicastové skupiny, ke které se přihlásily klienti 3 a 4. V síti je nalezena optimální trasa a data směřují jen k příjemcům bez žádných duplicit.



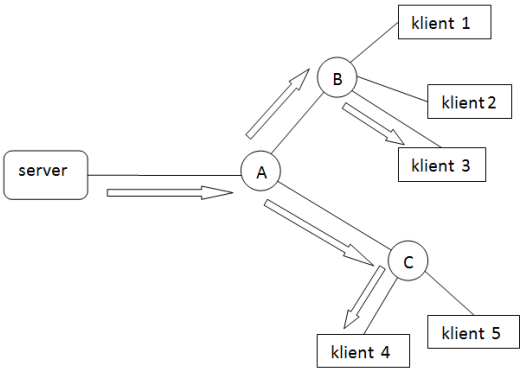
Obrázek 4.1.: Přenos unicast



Obrázek 4.2.: Přenos broadcast

4.1.2. Realizace datových přenosů GStreamerem

Všechny výše uvedené možnosti komunikace je možno realizovat GStreamerem. Příklady provedení příslušných pipeline jsou uvedeny níže, ve všech ukázkových pipeline je použit formát MJPEG (`jpegenc ! multipartmux` / `multipartdemux ! jpegdec`) s testovacím obrazcem. První pipeline je vysílací, druhá příjmová. Příklad komunikace probíhá mezi počítači v lokální síti LAN se síťovou adresou 192.168.1.0 a maskou podsítě 255.255.255.0 – z výpočtů IP adres vyplývá, že připojené počítače mohou mít přidělenou IP adresu v rozsahu 192.168.1.1 – 192.168.1.254, broadcastová IP je 192.168.1.255. Vysílací počítač má adresu 192.168.1.102 přijímací 192.168.1.107.



Obrázek 4.3.: Přenos multicast

Unicastová komunikace přes protokol UDP:

```
videotestsrc ! ... ! udpsink host=192.168.1.107 port=40000
udpsrc port=40000 ! ... ! xvimagesink
```

Unicastová komunikace přes protokol TCP:

```
videotestsrc ! ... ! tcpserversink host=192.168.1.107 port=40000
tcpclientsrc host=192.168.1.102 port=40000 ! ... ! xvimagesink
```

Broadcast (UDP):

```
videotestsrc ! ... ! udpsink host=192.168.1.255 port=40000
udpsrc port=40000 ! ... ! xvimagesink
```

Multicast (UDP):

```
videotestsrc ! ... ! udpsink host=224.0.0.100 auto-multicast=true port=40000
udpsrc multicast-group=224.0.0.100 auto-multicast=true port=40000
! ... ! xvimagesink
```

Při unicastové komunikaci UDP protokolem jsou pakety ihned po spuštění vysílací pipeline odesílány na cílouvou IP adresu. V případě TCP spojení nastane přenos dat až po připojení příjemce. TCP spojení jde zachytit i z počítače s neveřejnou IP (pokud se spojení realizuje přes Internet a serverová pipeline je spuštěna na PC s veřejnou IP) a může být zachyceno i více příjemci. U UDP spojení pakety směřují jen na jedinou konkrétní IP – přes Internet jen na veřejnou IP.

V případě broadcastu lze použít jen UDP pakety – ty jsou zasílány na broadcastovou adresu sítě a šíří se na všechny počítače.

Multicast opět jen UDP – na straně vysílání se zaregistruje multicastová skupina na IP 224.0.0.100 na portu 40000, příjemce se k této skupině připojí a router začne směrovat UDP pakety na tento počítač na základě komunikace IGMP protokolem. Po odpojení příjemce ze skupiny přestanou být UDP pakety na tento počítač zasílány. Multicast má vyhrazeno několik IP adres jako např. 224.0.0.1 – do multicast skupiny se připojí všechny multicastové prvky v LAN ; 224.0.0.2 – všechny multicastové routery v LAN; zbylý rozsah adres do 224.0.0.255 slouží pro lokální LAN s TTL=1. V Internetu musí být přidělena veřejná multicastová adresa.

Použitelnost přenosů

V lokálních sítích LAN (např. školní síť ČVUT) by měli být všechny výše uvedené možnosti proveditelné. Obraz je ale třeba přenášet přes Internet, kde ne každý počítač má k dispozici veřejnou IP nebo možnost přesměrování portů apod. V tomto případě nelze použít paketů UDP, zbývá TCP unicastové spojení a multicast.

Multicast je určen právě pro multimediální streamování – data jsou efektivně přenášena velkému počtu příjemců. Multicast využívá protokolu UDP, který ačkoli nezaručuje správné doručení paketů, nedochází u něj ke zpětnému dotazování jako u protokolu TCP, které při živém vysílání je spíše zdržujícím prvkem a nemá smysl lpět na části obrazové informace, která se při přenosu opozdí (nebo dokonce ztratí). Multicast ale není podporován všemi síťovými prvky a v Internetu je zatím nepoužitelný.

Zbývá jediná možnost, a to TCP unicastové spojení. Zasílat data přímo přes navázané TCP spojení, jak bylo uvedeno výše se nejevilo po zkouškách optimální. Docházelo k narůstání časového zpoždění mezi zobrazeným obrazem a časem pořízení snímku. Bylo třeba použít jiný způsob přenosu dat nad protokolem TCP.

RTSP protokol

RTSP (Real Time Streaming Protocol) je protokolem reálného času aplikační vrstvy v OSI modelu, slouží zejména k přenosu multimediálního obsahu v IP sítích. RTSP umožňuje obsluhovat a synchronizovat několik datových proudů (audio i video), nepřenáší je sám, ale přes protokoly RTP. RSTP je syntaxí podobný protokolu HTTP (používá URL a podobné mechanismy komunikace) – každý datový tok je definován vlastní RTSP URL.

Obrázek 4.4 ukazuje rozdělení datových přenosů mezi mediálním serverem a příjemcem. Spojení s mediálním serverem začíná komunikací s vestavěným web serverem obdobně jako při dotazech u HTTP protokolu. Jednotlivé příkazy jsou vypsány v tabulce 4.1 spolu se směrem komunikace (Client × Server).

Každé započaté relaci je přidělen identifikační řetězec `session`, který jednoznačně identifikuje datový tok spojený s konkrétním příjemcem. Po navázání spojení se serverem příjemce obdrží informace o datovém toku – SDP (Session Description Protocol). Záznam počátku komunikace s použitým RTSP serverem je v příloze na výpisu C.1.

příkaz	směr
DESCRIBE	C→S
ANNOUNCE	C→S, S→C
GET_PARAMETER	C→S, S→C
OPTIONS	C→S, S→C
PAUSE	C→S
PLAY	C→S
RECORD	C→S
REDIRECT	S→C
SETUP	C→S
SET_PARAMETER	C→S, S→C
TEARDOWN	C→S

Tabulka 4.1.: Příkazy RTSP protokolu

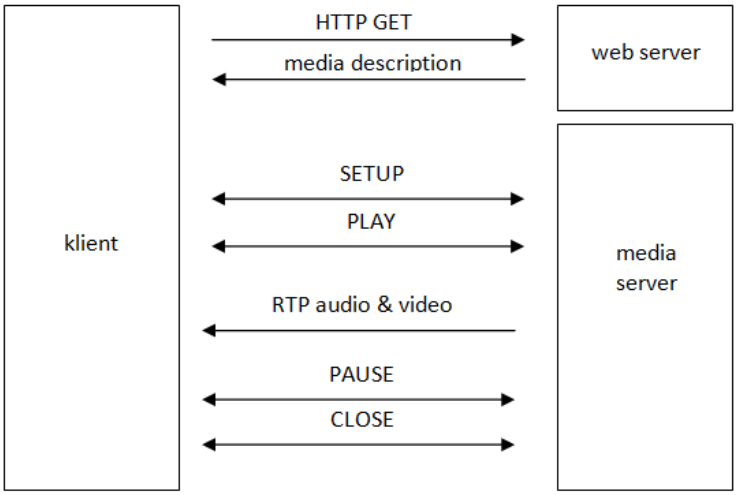
Mezi nejdůležitější příkazy protokolu RTSP patří:

SETUP Klient požaduje po serveru připravit zdroje a zahájit RTSP relaci.

PLAY Klient požaduje přenos dat připravených podle SETUP.

PAUSE Klient dočasně pozastavuje tok bez uvolnění prostředků na serveru.

TEARDOWN Klient požaduje uvolnění prostředků na straně serveru spojených s datovým tokem. Relace je ukončena.



Obrázek 4.4.: Komunikace mezi RTSP serverem a klientem

4.1.3. Datové přenosy použité v projektu

Možnost, jak realizovat RTSP server s výše uvedenými vlastnostmi, je server vytvořený programovým balíkem *gst-rtsp-0.10.8*, což je RTSP server založen na GStreameru – jednotlivé mediální streamy jsou reprezentovány GStreamerovskými pipeline. Na stránkách GStreameru [2] je možné tento balík stáhnout spolu s ukázkovými kódy a popisem. Na základě RTSP serveru je realizována část přenosu obrazu k uživatelům.

TCP unicast realizovaný GStreamerem se ukázal jako vhodný prostředek pro přenos dat mezi jednotlivými bloky aplikace. Z důvodu jednoduché implementace a relativně jednoduché komprese je zvolen formát MJPEG. Tato komunikace použitá na stejném počítači je velmi rychlá a efektivní, je vhodná i pro přenos dat mezi počítači v místní síti LAN.

4.2. Ukládání videí

Ukládání slouží k zaznamenávání dění na hřišti po dobu práce uživatelů s roboty. V současnosti jsou práce na hřišti rozdělovány rezervačním systémem na 30 minutové úseky počínající každou celou hodinu a hodinu a půl. Proto i jednotlivé záznamy by měli být dělené po třicetiminutových úsecích. Dále by měla existovat možnost, jak ukládání spustit a zastavit.

Z hlediska GST pipeline se nabízí element `filesink`, který je výstupní částí pipeline a zapisuje data do souboru předaném parametrem `location`. Koncová část pipeline musí být ve tvaru `! filesink location="soubor"`. Jinou použitelnou možnost pro ukládání GStreamer nenabízí.

Dělení souborů na 30 minutové části musí být řízeno z vnějšku pipeline – změnou parametru `location`, zastavením a opětovným spuštěním pipeline – zde poslouží navržení vlastní GST aplikace podle šablony, viz. část 2.2.3.

Řízení spouštění a ukončování ukládání je vhodné řešit síťovým soketem, jelikož se jedná o jednoduchou a spolehlivou metodu meziprocesní komunikace.

4.3. Příjem snímků z kamery a odstranění soudkovité vady

Získat obraz z kamery je možné dvěma způsoby:

- Jako RTSP stream – v GStreameru využít elementu `rtspsrc` a obraz dále zpracovávat v pipeline.
- Přes HTTP protokol – využít programu `imgsrv` v kameře, který slouží jako zdroj jednotlivých JPEG snímků.

Počáteční úvahy o odstranění soudkovitého zkreslení směřovaly k vytvoření vlastního GST elementu, který by se vřazoval do pipeline a odstranil by soudkovité zkreslení – tedy přijímat RTSP stream z kamery GStreamerem. Po zrealizování elementu se při zkouškách projevila nepříjemná vlastnost, kdy při snímkové rychlosti 10 snímků za sekundu (rozlišení 1296×960 px) a zvýšené zátěži serveru nebyl element schopen provést obrazovou transformaci v rámci trvání jednoho snímků – tedy do 0,1 s. Docházelo ke zřetelnému zasekávání obrazu.

Při druhé možnosti, získávat snímky přes HTTP protokol, GStreamer nenabízí prostředky pro realizaci příjmu snímků přes HTTP protokol a aplikace musí být samostatným programem nezávislým na GStreameru. Po příjmu snímku bude mít tato aplikace přístup ke snímkům v plném rozlišení a může ukládat snímky pro dohled nad hřištěm. Dále je v ní možné implementovat HTTP server sloužící jako zdroj JPEG snímků pro webové stránky projektu v rámci „webkamery“. Algoritmus odstranění soudkovité vady zde může být implementován nepřinášeje komplikace způsobené delší dobou zpracování.

Potřebná komunikace – příjem snímků, HTTP server, ovládání parametrů je možné implementovat přes síťové sokety. Poskytování obrazu dál ke zbylým částem projektu, které jsou založeny na GStreameru, je nutné vyřešit s podmínkou kompatibility přenosu dat s možnostmi vstupu signálu do GST pipeline.

4.4. Návrh struktury projektu

Na obrázku 5.1 v následující kapitole je navržené blokové schéma, které vyplývá z výše probíraných aspektů jednotlivých funkčních bloků. Jádrem projektu je aplikace `image_server`, která jako jediná není založena na GStreameru a slouží k příjmu snímků z kamery (HTTP protokolem), ukládá bezpečnostní snímky a je web-serverem poskytující snímky HTTP protokolem. Algoritmus odstranění soudkovité vady je implementován jen v této aplikaci. Obrazová data jsou z aplikace pro `stream_provider` (RTSP server) a `stream_storage` (ukládací aplikace) přenášena TCP protokolem ve formátu MJPEG. Aplikace simuluje chování GStreameru s konečnou částí pipeline – `! jpegenc ! multipartmux ! tcpserversink`.

Ukládací aplikace `stream_storage` je samostatnou aplikací, jenž zajišťuje nahrávání 30 minutových bloků – aplikace spravuje pipeline, která reprezentuje ukládací řetězec. Aplikace je založená na šabloně GST aplikace popsané v části 2.2.3.

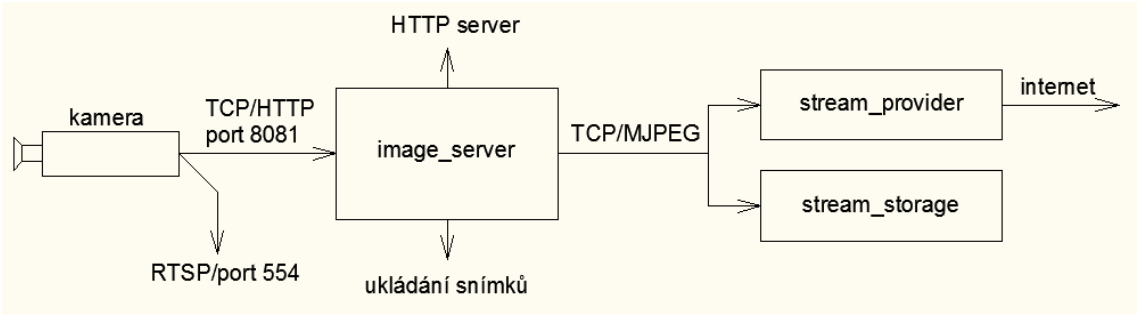
Streamovací aplikace `stream_provider` – RTSP server – přebírá MJPEG stream od `image-server` společně se `stream_provider` a slouží jako RTSP server. GStreamer nabízí RTSP server na svých webových stránkách [2].

5. Realizace

Jednotlivé dílčí celky projektu jsou seskupeny do bloků podle obrázku 5.1. Každý blok je realizován samostatnou aplikací, které dohromady tvoří celý řetězec zpracování obrazového signálu. Jedná se o program `image_server` – ten slouží pro ukládání dohledových snímků, dále jako HTTP server, který poskytuje snímky přes HTTP protokol a především jako zdroj obrazového signálu pro další části řetězce. Snímky získává přímo z kamery přes protokol HTTP viz. 2.1.2 (Image server v kameře).

RTSP server – aplikace `stream_provider` – poskytuje RTSP stream v několika formátech pro prohlížení uživateli přes Internet. Přejímá signál od `image_server` spolu s aplikací `stream-storage` jako MJPEG stream přes TCP port.

Aplikace `stream_storage` ukládá videozáznam po třicetiminutových blocích na disk serveru. Aplikaci `image_server` a `stream_storage` je možné ovládat zasíláním textových zpráv přes TCP soket.



Obrázek 5.1.: Blokové schéma navrženého řešení

Tento způsob realizace a rozložení jednotlivých bloků je zvolen na základě možností GStreameru sloučených s požadavky na chování celého projektu a na základě zkušeností s GStreamerem. Blok `image_server` s uvedenými vlastnostmi nelze GStreamerem vůbec realizovat. S požadavkem na odstranění soudkovitého zkreslení, které je náročnou operací, je vhodné obrazovou transformaci provádět pouze na jednom místě a ne vícekrát pro stejný snímek. Tímto místem je `image_server`, který dále slouží jako zdroj korigovaného obrazu. Bloky `stream_provider` a `stream_storage` jsou založeny na rozdílných programových částech GStreameru. Sloučení `stream_provider` a `stream_storage` do jedné aplikace by mělo být možné, ale oddělení do samostatných aplikací nepřináší zvýšení nároků na systémové prostředky při příjmu MJPEG streamu – oba bloky přijímají stejná data a část příjmu MJPEG proudu by stejně nešlo sdílet.

Další argument pro realizaci bloku `image_server` jako samostatné aplikace nezaložené na GStreameru je fakt, že GST pipeline zakončená TCP elementem `tcpserversink` (viz. část 5.2.4) je neustále spuštěná (i když nejsou připojeni žádní příjemci). Aplikace `image_server` aktivuje potřebné části oproti GStreameru až při připojení příjemce.

Na serveru jsou programy reprezentující jednotlivé bloky rozdělené do adresářové struktury podle výpisu 5.1.

Výpis 5.1: Adresářová struktura a rozmístění programů na serveru

```
bin/
  stream_control - ovládání parametrů pro
                   image_server a~stream_storage
etc/
  k13/
    KC.mat - parametry obrazové transformace - vnitřní
    KK.mat - parametry obrazové transformace - af. trans.
    config - konf. soubor image_serveru
  k14/
    ...
  k15/
    ...
lib/
  libgstrtspserver-0.10.so - knihovna pro RTSP server
sbin/
  image_server
  stream_provider
  stream_storage
var/
  log/ - logovací soubory
  security/ - ukládání jednotlivých snímků
  video/ - místo ukládání nahraných videí
```

`bin/` obsahuje programy určené ke spouštění uživateli – `stream_control` pro ovládání `image_server` a `stream_storage`.

`etc/` pro jednotlivé kamery obsahuje konfigurační soubory `KC.mat` a `KK.mat` k odstranění soudkovité vady a soubor `config` s konfiguračním nastavením pro aplikaci `image_server`.

`lib/` obsahuje knihovny nutné ke spuštění programů.

`sbin/` obsahuje programy, které uživatel nespouští.

`var/` sem se ukládají záznamy ze `stream_storage`, bezpečnostní snímky a logovací soubory.

5.1. Komunikace mezi jednotlivými programy

Komunikace mezi jednotlivými programy je řešena síťovým soketem. Na soketech je založena veškerá síťová komunikace v projektu – HTTP server a TCP MJPEG streamer v `image_server`, příjem snímků z kamery, ovládání `image_server` a `stream_storage`.

Soket je prostředek, který slouží ke komunikaci mezi procesy. Procesy nemusejí být spuštěné na stejném počítači, ale mohou komunikovat po síti prostřednictvím síťového protokolu. Komunikaci tvoří spojový protokol TCP nebo protokol UDP. V projektu se pracuje jen s protokolem TCP, který zajišťuje správný příjem informace. Jmenné prostory (protocol family) se u soketů rozlišují podle verze IP protokolu na IPv4 a IPv6 (v projektu verze IPv4).

Komunikace začíná založením soketového serveru na TCP portu, server čeká na připojení klienta, jenž pro připojení potřebuje znát IPv4 adresu serveru a příslušný port. Po připojení

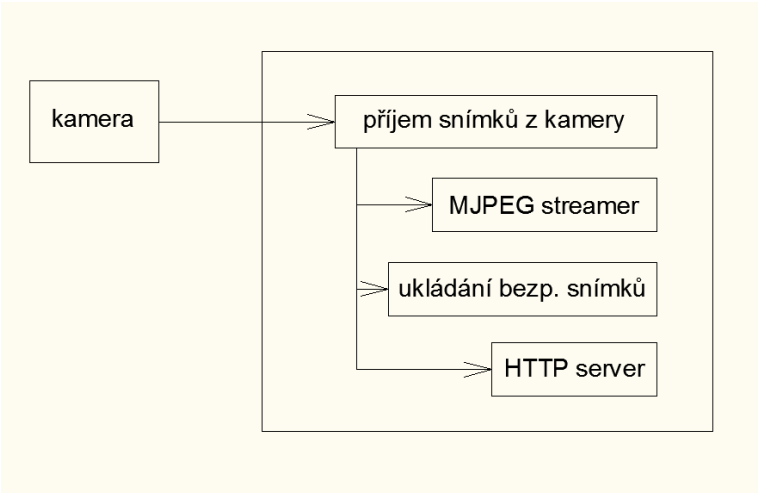
klienta server obsluhuje komunikaci. Server může komunikovat zároveň s více klienty, kteří se připojují stále na stejný port serveru.

Informace o soketech jsou čerpány z [11] spolu s ukázkami zdrojových kódů použitých v projektu. Na základě těchto kódů jsou řešeny všechny programy využívající soketovou komunikaci.

5.2. Image server

`Image_server` slouží jako zdroj signálu pro ostatní části projektu. Aplikace se skládá z několika částí, které jsou do jisté míry samostatné, viz. obrázek 5.2. Jedná se o tyto části:

- *Příjem snímků z kamery*; získává JPEG snímky a dále je poskytuje dalším částem aplikace.
- *MJPEG streamer* poskytuje obraz s korekcí zkreslení programům `stream_storage` a `stream-provider`.
- *Ukládání kontrolních snímků*; ukládá s periodou 1 s snímky přijaté přímo z kamery bez úpravy na disk počítače.
- *HTTP server* poskytuje přes HTTP protokol JPEG snímky o žádaných rozměrech s korigovanou vadou soudkovitosti.



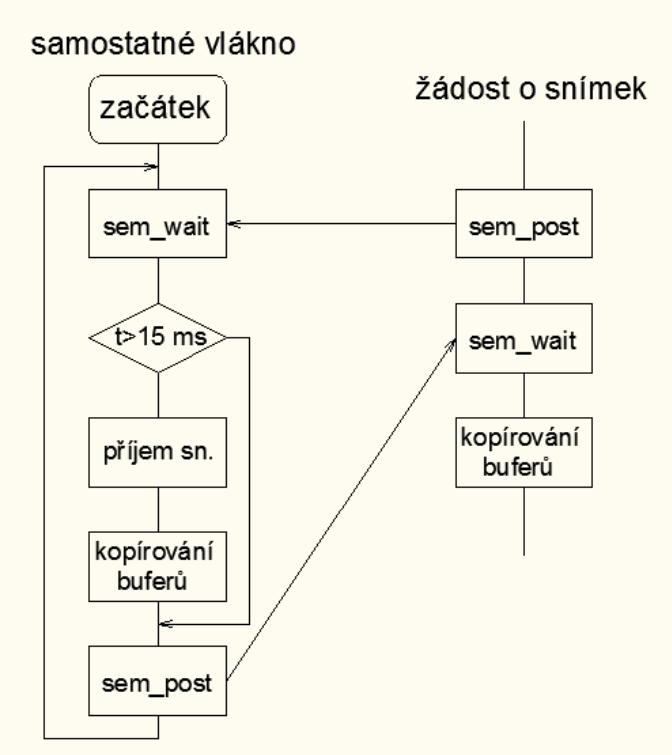
Obrázek 5.2.: Blokové schéma programu `image_server`

Program pracuje s JPEG snímky, k manipulaci s JPEG formátem je použita knihovna *jpeglib-8d* od *The Independent JPEG Group* [7]. Tato volně šiřitelná knihovna obsahuje funkce pro převod mezi obrázky ve formátu JPEG a maticí RGB bodů. Mezi zdrojovými soubory potřebnými ke zkompilování dynamické knihovny jsou zahrnuty i ukázkové programy. Od verze 8 můžou funkce pracovat s JPEG snímky reprezentovanými polem v paměti počítače oproti minulým verzím, kde JPEG snímky byly soubory na disku. V případě snímku JPEG i RGB matice se jedná o pole 8 bitových čísel, u JPEGu je délka pole uvedena zvlášť v proměnné, u RGB je velikost pole dána $3 \times \check{s}\acute{i}\check{r}ka \times v\acute{y}\check{s}ka$.

Program je vícevláknovou aplikací, aby byl umožněn paralelní chod jednotlivých bloků a urychleno zpracování obrazu (převody formátů, odstranění soudkovité vady). Synchronizaci vláken a zabránění přístupu ke sdíleným datům různými vlákny zajišťují prvky *mutex* a *semafor*.

Podle obrázku 5.2 všechny části programu sdílí blok příjmu snímků z kamery. Každý blok může žádat o snímek v jiný čas a musí být zajištěn rychlý přísun snímků každému bloku. Současný přístup k prostředkům příjmu snímku je vyřešen mutexy a semafory. Na obrázku 5.3 je naznačen princip příjmu nového snímku na vyžádání jedním z bloků. Blokům jsou přiřazena

vlákna, která se liší jen semafory. Podle vývojového diagramu (5.3) je blok příjmu snímku umístěn v nekonečném cyklu, kde každý průchod je spouštěn semaforem nastavovaným mimo vlákno. Následuje rozhodovací blok, který určí dobu od příjmu posledního snímku, je-li tato doba kratší než 15 ms použije se starý snímek. Vlastní příjem snímku trvá přibližně 50 ms, aby po tuto dobu nebyly blokovány příjmové buffery jsou využity ještě pomocné buffery, do kterých jsou data nakopírována po příjmu snímku a po celou dobu jsou přístupná ostatním částem programu. Bloky *příjem snímku* a *kopírování bufferů* ve výjovém diagramu jsou v kódu obaleny mutexy. Po přijetí snímku a zkopírování bufferů je uvolněn semafor v místě požadavku snímku a proběhne opětovné kopírování bufferů tentokrát do místa dalšího zpracování, opět z důvodu zamezení blokace pomocných příjmových bufferů po dobu zpracování jednotlivými částmi programu.



Obrázek 5.3.: Princip příjmu novéhu snímku na vyžádání jedním z bloků

Vstupní parametry

Veškeré vstupní parametry jsou sepsány v konfiguračním souboru `config`. Umístění tohoto souboru je jedinným parametrem vstupujícím do programu. Příklad struktury konfiguračního souboru je na výpisu 5.2. Soubory `config` jsou v adresářové struktuře na serveru (výpis 5.1) umístěné ve složce `etc/kN/` (`N` značí číslo kamery – v projektu koresponduje s poslední částí IP adresy kamery).

Výpis 5.2: Příklad konfiguračního souboru aplikace `imgsrv`

```
# IP adresa kamery
  10.10.20.13
# port na kamere k~imgsrvr
  8081
# cast URL za portem
  /bimg/
# port HTTP serveru
  3313
# port pro TCP-MJPEG streamer
  4413
# port ovladani soketem
  5513
# sirka a~vyska vystupniho snimku ve streameru
  648
  480
# vzorkovaci slozka a~povoleni vzorkovani (1/0)
  /opt/streamer/var/security/
  1
# umisteni KC a~KK souboru
  /opt/streamer/etc/k13/
# logovaci soubory
  /opt/streamer/var/log/imgsrv13.log
```

5.2.1. Příjem snímků z kamery

Snímky se přijímají z kamery HTTP požadavky na obrazový server spuštený v kameře. V programu je implementována stejná komunikace naznačená na výpisu 2.1 na straně 4. Naváže se komunikace na IP adresu kamery a port 8081. HTTP požadavek je „GET /bimg/“. Odpověď serveru v případě zdařeného spojení je na výpisu 5.3. Není-li kamera schopna zaslat JPEG snímek signalizuje chybu zasláním obrázku GIF 1×1 px s odpovídající HTTP hlavičkou. Při úspěšném příjmu je zjištěna velikost snímku a ten je uložen do připraveného JPEG bufferu s indikací jeho délky. Poté je JPEG převeden funkcemi z knihovny *jpeglib-8d* do RGB bufferu, s kterým dále pracují ostatní aplikace.

Výpis 5.3: Odpověď `imgsrv` v kameře na HTTP požadavek „GET /bimg/“

```
HTTP/1.0 200 OK
Server: Elphel Imgsrv
Expires: 0
Pragma: no-cache
Content-Type: image/jpeg
Content-Disposition: inline; filename="elphelimg_79327.jpeg"
Content-Length: 80369

JPEG data ...
```

5.2.2. Ukládání kontrolních snímků

Ukládání kontrolních snímků slouží k dohledu nad hřištěm po celou dobu spuštění programu `image_server`. Aplikace každou sekundu stáhne JPEG snímek v plném rozlišení 1296×960 px.

U snímku není korigována vada soudkovitosti a není upravován, aby byl server minimálně zatěžován. Složka pro ukládání snímků je předána parametrem v konfiguračním souboru, dalším parametrem je možné pořizování snímků zakázat. Snímky jsou v adresáři děleny podle dní a jméno souboru odpovídá času pořízení snímku.

5.2.3. HTTP server

Server poskytuje JPEG snímky z kamery přes protokol HTTP. Server naslouchá na portu předaném parametrem a data jsou k dispozici pod URL zakončenou „/“ – HTTP požadavek by byl „GET /“, server odešle potvrzení „HTTP/1.0 200 OK“ a hlavičku JPEG obdobně jako obrázkový server v kameře – výpis 2.1 na straně 4, při ostatních požadavcích zasílá HTML hlavičku „HTTP/1.0 404 NOT FOUND“. Snímky je možné získat přes URL:

```
http://<server>:<port>/
http://<server>:<port>/?width=648
http://<server>:<port>/?height=480
http://<server>:<port>/?height=480&width=700
```

Parametry v URL slouží k nastavení šířky nebo výšky snímku – druhý rozměr je dopočten se zachováním proporcí, při zadání obou rozměru je velikost snímku upravena tak, aby byl celý zobrazený s černými okraji.

Webkamera umístěná na stránkách projektu čerpá snímky z tohoto rozhraní. JavaScriptem je snímek obnovován každých 500 ms.

5.2.4. Zdroj MJPEG streamu

Tato část programu tvoří MJPEG stream dále využívaný aplikacemi `stream_provider` a `stream-storage`. MJPEG je formát videa, kde je každý snímek komprimován zvlášť podle standardu JPEG. Stream je plně kompatibilní se vstupními elementy GStreameru. V projektu tvoří MJPEG proud dat, který probíhá mezi odesílatelem a příjemcem po síťovém protokolu. V projektu se využívá spojový protokol TCP. GStreamer nabízí tyto elementy pro obsluhu TCP protokolu:

- `tcpserversrc`
- `tcpserversink`
- `tcpclientsrc`
- `tcpclientsink`.

Komunikace vyžaduje nejprve vytvoření serveru na TCP portu, ke kterému se následně připojuje klient – tomu odpovídají i názvy elementů. Element se chová z pohledu pipeline jako zdrojový (`src`) nebo jako výstupní (`sink`). Komunikaci lze v GStreameru realizovat takto:

server

```
videotestsrc ! jpegenc ! multipartmux ! tcpserversink port=1789
```

klient

```
tcpclientsrc port=1789 ! multipartdemux ! jpegdec ! ...
```

Po navázání komunikace jsou data přenášena ve tvaru podle výpisu 5.4. Element `multipartmux` tvoří MJPEG stream za pomoci oddělovacích řetězců (není stanoven konkrétní tvar, musí začínat „– –“), následuje HTTP hlavička pro JPEG s velikostí následujícího snímku. Přenos se neřídí přesnou snímkovou rychlostí. V `image_server` je implementována serverová část pipeline – `! jpegenc ! multipartmux ! tcpserversink`.

Výpis 5.4: Formát MJPEG

```
– –oddelovaci_retezec
Content–Type: image/jpeg
Content–Length: 9290

[JPEG DATA]
– –oddelovaci_retezec
Content–Type: image/jpeg
...
```

Aplikace může obsluhovat maximálně 5 příjemců streamu, druhému a dalším příjemcům zasílá kopii streamu. Po odpojení všech příjemců zůstává aplikace nečinná a čeká na další spojení.

Vysílaný obraz má korigovanou vadu soudkovitosti a jeho parametry jako zvětšení a posun výřezu jsou řízené soketovým ovládáním `stream_control`.

5.3. RTSP server

RTSP server je založen na balíku *gst-rtsp-0.10.8* – `http://gstreamer.freedesktop.org/src/gst-rtsp-server/` ; (v distribuci Ubuntu ve správci balíků jako *gst-rtsp-server*). Po zkompilování obsahuje dynamické knihovny, hlavičkové soubory, nápovědu a příklady jednotlivých RTSP serverů.

Výhoda použití RTSP serveru spočívá v optimálním využíváním systémových prostředků. Není-li připojen žádný příjemce je RTSP server zcela nečinný. Po připojení příjemce se spustí pipeline, o kterou příjemce žádá. Při připojení druhého a dalších příjemců na stejnou pipeline dostávají kopii streamu. Po jejich odpojení se pipeline ukončí. Zcela nevhodné je v tomto případě použití TCP elementů samotného GStreameru, protože pipeline zakončená např. elementem `tcpserversink` je neustále spuštěná nehledě na připojené klienty.

Popis kódu

Ukázka kódu ve výpisu A.1 na straně II v příloze představuje nejjednodušší RTSP aplikaci, která vysílá testovací videoobrazec na URL `rtsp://<IP>:8555/test`. Kód je převzatý z ukázkového programu z balíčku *gst-rtsp-0.10.8* a mírně doplněn.

Funkce `gst_rtsp_server_set_service(server, "8555");` slouží ke změně streamovacího portu, implicitně je nastaven port 8554. V návodu k RTSP serveru je zatím uvedená jiná chybná funkce.

`gst_rtsp_media_mapping_add_factory(mapping, "/test", factory);` přidává konečnou část URL adresy.

Dále je důležitá funkce `gst_rtsp_media_factory_set_shared(factory, TRUE);` ta povoluje sdílení streamu pro další pozorovatele – druhý a další příjemci dostávají kopii streamu.

`gst_rtsp_media_factory_set_launch(factory, "( videotestsrc ! ffenc_mpeg4 ! rtpmp4vpay name=pay0 pt=96 )");` přidává vlastní pipeline, tím volíme zdroj, zpracování a konečnou kompresi signálu. V příkladu je použitá komprese MPEG4, elementem `rtpmp4vpay` dochází k zakódování do RTP paketů. Důležitou volbou je `name=pay0` – každý zdroj videa zakončený „payloaderem“ se stane streamem. Číslování probíhá od `pay0` dále `pay1` atd. Takto je možno do streamu přiřadit např. zvukovou stopu.

Aplikace může obsahovat několik zdrojů RTSP videa snadno rozlišených částí URL adresy za lomítkem. K tomu stačí vytvořit více proměnných typu `GstRTSPMediaFactory` a funkcemi `gst_rtsp_media_factory_set_launch`, `gst_rtsp_media_mapping_add_factory` přiřadit potřebný konec URL adresy a samotnou pipeline.

Použití RTSP serveru

RTSP server `stream_provider` přebírá vstupní MJPEG stream z TCP portu a poskytuje celkem čtyři RTSP streamy v jednom RTSP serveru založeném na portu zadaném parametrem. Formát obrazu je 648×480 px a 432×320 px v kompresích MPEG4, Theora a MJPEG při snímkové rychlosti 10 nebo 5 snímků za sekundu. Rozlišit mezi jednotlivými streamy umožňuje koncová část URL. Bude-li port serveru `8554` a koncová část URL `kam`, budou k dispozici následující streamy s příslušnými pipeline:

`rtsp://<server>:8554/kamT` – MJPEG testovací obrazec, 648×480 px
`videotestsrc ... ! ffenc_mjpeg bitrate=100000 ! rtpjpegpay name=pay0 pt=96`

`rtsp://<server>:8554/kamA` – MPEG4 stream, 648×480 px, 10 snímků za sekundu
`tcpclientsrc ... ! ffenc_mpeg4 ! rtpmp4vpay name=pay0 pt=96`

`rtsp://<server>:8554/kamB` – Theora stream, 648×480 px, 10 snímků za sekundu
`tcpclientsrc ... ! theoraenc ! rtptheorapay name=pay0 pt=96`

`rtsp://<server>:8554/kamC` – MPEG4 stream, 432×320 px, 5 snímků za sekundu
`tcpclientsrc ... ! ffenc_mpeg4 ! rtpmp4vpay name=pay0 pt=96`

Testovací stream dovoluje vyzkoušet funkčnost samotného RTSP serveru, další slouží ke streamování v různých kvalitách nebo formátech.

Kompilace a spuštění

Kompilaci lze provést příkazem (5.5). Na základě tohoto příkazu je vytvořen soubor `Makefile` sloužící ke kompilaci RTSP serveru. Program pro spuštění vyžaduje dynamickou knihovnu `libgstrtspserver-0.10.so` použitou při kompilaci. Po spuštění programu začne být RTSP server aktivní až po připojení klienta, v době nečinnosti nezatěžuje počítač.

Výpis 5.5: Kompilace RTSP serveru

```
gcc -I../gst-rtsp-0.10.8/ -I/usr/local/include/glib-2.0/
  -I/usr/local/include/gstreamer-0.10/
  -I/usr/local/include/libxml2/ -O0 -g3 -Wall -c
  -fmessage-length=0
  -MMD -MP -MF "stream_provider.d" -MT "stream_provider.d"
  -o "stream_provider.o" "stream_provider.c" -I/usr/local/include/

gcc  -o"stream_provider"  ./stream_provider.o  -lgstrtspserver-0.10
  -lgstreamer-0.10 -lglib-2.0
  -L../gst-rtsp-0.10.8/gst/rtsp-server/.libs/
  -L/usr/local/lib/ -Wall -g
```

Prohlížení streamů

GStreamer umožňuje prohlížet RTSP streamy elementem pro příjem RTSP videa `rtspsrc`. Spuštění může vypadat následovně:

```
gst-launch-0.10 rtspsrc location=rtsp://syrotek.felk.cvut.cz:8554/kamA
                                latency=100
              ! decodebin ! ffmpegcolorspace ! ximagesink
```

Parametrem `latency` elementu `rtspsrc` volíme velikost bufferu RTSP streamu v ms. Při navázání prvotního spojení s RTSP serverem se `rtspsrc` pokouší nejdříve přijímat stream přes UDP pakety, pokud po dobu 5 s neobdrží žádný paket přejde na TCP pakety. Spojení UDP

pakety je možné v lokální síti, mělo by fungovat i na počítačích s veřejnou IP, jinak je třeba zvolit spojení TCP. Vynutit TCP spojení se serverem je možné parametrem `protocols=0x00000004` elementu `rtspsrc`. Element `decodebin` sám zvolí potřebnou metodu dekódování a dá se použít pro všechny výše uvedené streamy.

5.4. Ukládací aplikace

Hlavní úkol aplikace `stream_storage` je zaznamenávat obraz z kamer sledujících hřiště. Ze zadaných požadavků plyne potřeba ukládat video ve 30 minutových blocích. Aplikace je založena na programovém balíku z repozitáře GIT – `git://anongit.freedesktop.org/gstreamer/gst-template.git` – obsahuje vše potřebné pro tvorbu buď vlastního elementu GStreameru nebo vyvinout GST aplikaci – spustitelný program, který je schopen složit vlastní pipeline a řídit její spouštění, nastavovat její parametry apod.; v repozitáři SVN a na příloze je tento balík zkomprimován v souboru `gst-template.tar.bz2`. Balík je nejprve nutné sestavit spuštěním skriptu `./autogen.sh` v adresáři `gst-app/`. Veškeré zdrojové soubory se nacházejí v adresáři `gst-template/gst-app/src/`. Kromě zdrojových a hlavičkových souborů obsahuje adresář soubor `Makefile.am`, který slouží pro vytvoření výsledného `Makefile`. Struktura souboru `Makefile.am` je intuitivní a snadnou úpravou můžeme přidávat další zdrojové soubory, případně další aplikace. Při každé úpravě tohoto souboru je třeba nechat znovu vygenerovat nový soubor `Makefile` spuštěním konfiguračního skriptu `./configure` v nadřazeném adresáři.

Zdrojový kód ukládací aplikace je v souboru `gst-app/src/main.c`. Po zkompilování se spustitelný soubor nazývá `stream_storage`, spouští se s parametry uvedených v následujícím pořadí:

- Cesta, kam se budou ukládat nahrávaná videa.
- TCP port sloužící pro ovládání.
- TCP port MJPEG streamu.
- Příznak okamžitého spuštění nahrávání po spuštění programu (`1` nebo `0`).
- Cesta k logovacímu souboru.

Popis GST aplikace

Základním prvkem GST aplikace je objekt `GstElement`, tím jsou jednak elementy, ale i vlastní pipeline. Objekty se odlišují inicializací:

```
GstElement *pipeline, *element[];
...
pipeline = gst_pipeline_new ("video-ukladac");
element[0] = gst_element_factory_make ("tcpclientsrc", "TCP_zdroj");
```

Další důležitou částí je přiřazení parametrů jednotlivým elementům:

```
g_object_set (G_OBJECT (element[0]), "host", "127.0.0.1", NULL);
g_object_set (G_OBJECT (element[0]), "port", atoi(argv[3]), NULL);
```

Elementy se přidavají do pipeline:

```
gst_bin_add (GST_BIN (pipeline), element);
```

Elementy se spojují do řetězce:

```
gst_element_link ( element[i], element[i+1] );
```

Po sestavení celé pipeline se spustí:

```
gst_element_set_state (pipeline, GST_STATE_PLAYING);
```

Aplikace čeká ve smyčce.

```
g_main_loop_run (loop);
```

Nyní je pipeline spuštěná stejně, jako by se spustila aplikací `gst-launch`. K pipeline je též připojena obsluha zpráv, které přicházejí z pipeline, zde se odchytávají především chybové zprávy a značka GST_MESSAGE_EOS (End of Stream) – konec proudu, poté může být smyčka přerušena a program ukončen.

```
g_main_loop_quit (loop);
```

Pipeline složená v aplikaci má následující zápis:

```
tcpclientsrc host=127.0.0.1 port=... do-timestamp=TRUE is-live=TRUE
! multipartdemux ! jpegdec ! ffmpegcolorspace ! videorate
! video/x-raw-yuv, framerate=10/1 ! clockoverlay ! ffmpegcolorspace
! ffenc_mpeg4 bitrate=... ! mp4mux ! filesink location="nahravka.mp4"
```

Pipeline přijímá MJPEG stream z místního TCP portu, parametrem `do-timestamp=TRUE` se stream opatří časovými značkami, na jejich základě pak element `videorate` zajistí přesnou snímkovou rychlost 10 snímků za sekundu. `Multipartdemux` a `jpegdec` slouží k dekódování streamu, `clockoverlay` přidává do obrazu časovou značku s rozlišením na sekundy. Videoobraz je kódován kompresí MPEG4 a elementem `filesink` ukládán do souboru.

Pipeline a GST elementy se mohou nacházet v různých stavech, kterých je dále využito pro řízení nahrávání po 30 minutových blocích. Mezi stavy patří:

GST_STATE_NULL Je výchozím stavem. Nejsou přiřazené žádné prostředky. Elementům je dovoleno v tomto stavu měnit parametry.

GST_STATE_READY V tomto stavu jsou elementům přiřazeny prostředky (alokování paměti, bufferů, otevření souboru apod.), datový proud není otevřen a nachází se na začátku.

GST_STATE_PAUSED V tomto stavu je datový proud otevřen, ale není aktivní. Je možné měnit pozici přehrávání ve streamu. Prakticky se jedná o stejný stav jako PLAYING s tím rozdílem, že je pozastavený zdroj hodin v pipeline.

GST_STATE_PLAYING Datový proud se přehrává.

Podle požadavku ukládat video po 30 minutových blocích musí zahajovací sekvence splňovat následující:

1. Dostat pipeline do stavu NULL.
2. Nastavit parametr `location` elementu `filesink` na nový soubor.
3. Nastavit pipeline do stavu PLAYING.

Situace při ukončování bloků je obtížnější, protože nejde jednoduše pipeline násilně přerušit stavem NULL, jelikož zůstane otevřený soubor v elementu `filesink` a nastavení tohoto elementu též na NULL situaci neřeší z důvodu komprese MPEG4. Soubor záznamu MPEG4 videa obsahuje na konci část informací zvanou „Moov“. Ta by v tomto případě chyběla a soubor by byl pro většinu přehrávačů nepoužitelný.

Ukončení ukládání je vyřešené nastavením zdrojového elementu na NULL. To vygeneruje událost EOS (End of Stream), na kterou zareaguje `mp4mux` vytvořením bloku „Moov“ a filesink poté uzavře soubor. Událost EOS je zachyna programem a v obsluze se nastaví celá pipeline do stavu NULL a čeká se na spuštění dalšího bloku. Tímto přístupem je zaručeno bezpečné uzavření souboru a neporušení struktury MPEG4.

Ukládání je možné spustit a ukončit zasláním řídící zprávy na soketový port. Pokud je nahrávání spuštěné ukládají se soubory po třicetiminutových blocích. Blok začíná každou celou hodinu nebo hodinu a půl. V adresáři předaném parametrem vznikne adresářová struktura, kde na každý den bude připadat jedna složka s pojmenováním podle datumu `RRRR-MM-DD`. V těchto adresářích jsou již samotné `*.mp4` soubory pod názvem časové značky zahájení nahrávání – `HH-MM-SS.mp4`.

5.5. Ovládání aplikace image_server a stream_storage

Programy `image_server` a `stream_storage` mohou být ovládány přes TCP soket. K tomu slouží aplikace `stream_control`. Aplikace se připojí na TCP port IP adresy serveru, kde jsou spuštěné řízené programy. Poté zašle ASCII zprávu zakončenou znakem *nochar* – '\0' (hodnota 0x00). Příjetí zprávy je potvrzeno krátkou odpovědí. Jelikož jsou porty na straně serveru otevřené všem, a aby nedošlo k mylnému příjmu zprávy je každá platná zpráva opatřena krátkým textovým řetězcem, který slouží jako heslo. Veškerá komunikace, která neobsahuje před vlastní zprávou toto heslo, bude odmítnuta. Vložení hesla do zprávy řídí sama aplikace `stream_control`. Tvar zprávy je – `HesloZpráva\0`. Heslo je uložené do proměnné na obou stranách ve zdrojových kódech.

Volání programu pro odeslání zprávy na port `17890` serveru `syrotek.felk.cvut.cz` je následující:

```
./stream_control zpráva 17890 147.32.84.212
```

Řídící zprávy pro aplikaci „image_server“

V aplikaci `image_server` je možné nastavovat parametry výstupního obrazu, který je dále ve formátu MJPEG zasílán na TCP port pro zpracování dalšími aplikacemi, částí ukládání kontrolních snímků a snímků zpracovávaných HTTP serverem se ovládání netýká. První písmeno značí příkaz, následující číslo pak parametr. Zpráva začínající na *„z“* – *zoom* (zvětšení) – 1 odpovídá zobrazení celého snímku, hodnoty větší než 1 jsou pak zvětšením. Zprávy začínající *„l“* – *left* nebo *„t“* – *top* při hodnotě zvětšení větší jak 1 umožňují posouvat zvětšený výřez po vstupním snímku. Podle prvního písmene buď od levé nebo horní strany snímku. Mohou nabývat hodnot 0,0 – 1,0.

Příklad řídících zpráv:
(v desetinných číslech musí být desetinná tečka)

l0.5 Okno výstupu se přesune na střed vstupního obrazu ve vodorovném směru.

t0.5 Okno výstupu se přesune na střed vstupního obrazu ve svislém směru.

z1.2 Nastaví 1,2 násobné zvětšení.

Řídící zprávy pro aplikaci „stream_storage“

Řídící zprávy v aplikaci `stream_storage` umožňují spustit a zastavit nahrávání, jejich tvary jsou:

start Spustí nahrávání, založí nový soubor s aktuální časovou značkou. Dále nahrává po 30 minutových blocích.

stop Zastaví nahrávání, uzavře stávající soubor. Dále nepokračuje v nahrávání ani v dalším bloku.

5.6. Spouštění programů

Jednotlivé aplikace tvoří jednotný celek, a aby celý řetězec správně fungoval, musí být programy spuštěné v daném pořadí. Musí být také ošetřena možnost, kdy z důvodu chyby dojde k předčasnému ukončení jedné z aplikací.

Programy `stream_storage` a `stream_provider` jsou závislé na programu `image_server` – ten musí být spuštěn jako první. Obrazový signál se mezi aplikacemi předává po protokolu TCP

(spojení klient – server), takže při ukončení aplikace `image_server` dojde k přerušení komunikace a následnému pádu aplikací přijímající data a celý řetězec je nutné ukončit a znovu spustit. GStreamer též reaguje na přerušení TCP spojení ukončením pipeline, což s sebou přináší uzavření nahrávaného souboru v `stream_storage` a přerušení RTSP streamu ze `stream_provider`.

Pro správné spouštění a hlídání stavů jednotlivých aplikací jsou použity BASH skripty, které také obsahují vstupní parametry k jednotlivým programům. Na serveru jsou skripty umístěné ve složce `bin/` (výpis 5.1).

5.7. Zdrojové soubory

Veškeré zdrojové soubory jsou obsaženy v SVN repozitáři `https://syrotek.felk.cvut.cz/svn/strobmir` a na CD příloze. Výpis adresářové struktury je na výpisu B.1 v příloze. Názvy složek korespondují s názvy jednotlivých aplikací.

6. Závěr

Projekt byl úspěšně nasazen na 3 kamery a splňuje zadané požadavky na svou funkci. Streamování splňuje parametry živého přenosu obrazu k uživatelům protokolem RTSP, který zajišťuje optimální přenos dat přes síť Internet s minimálním zpožděním. Uživatelům je umožněno měnit za chodu výřez obrazu, ten sledovat zvětšený a pohybovat s ním po celé ploše hřiště a zároveň obraz, který sledují (nebo by mohli sledovat) je ukládán na server v plném rozlišení 1296×960 px. Na serveru jsou záznamy uživatelů ponechávány 14 dní, odtud si je mohou vyzvednout a prohlížet. Po celou dobu spuštění aplikací jsou z jedné kamery pořizovány snímky pro dohled nad hřištěm. Na webových stránkách projektu je umožněno návštěvníkům sledovat dění na hřišti ze všech tří kamer.

Programové řešení je stabilní a nevykazuje žádné závažné chyby. Všechny programy zaznamenávají do logovacích souborů informace o své činnosti, ze kterých je možné odhalit případné chyby.

6.1. Velikosti datových přenosů a ukládaných dat

V tabulce 6.1 jsou uvedeny velikosti datových toků probíhající mezi kamerou a serverem, aplikací `image_server` a `stream_storage/stream_provider` a velikosti RTSP streamů. Měření probíhalo po dobu 1 minuty a při nevytíženém serveru – snímková rychlost TCP streamu dosahuje maxima, tedy 22 snímků za sekundu. Při velikém zatížení serveru se snímková rychlost začíná snižovat a není pravidelná – to ale nevadí při MJPEG streamu, který se neřídí přesnou snímkovou rychlostí a ve `stream_storage/stream_provider` element `videorate` upraví snímkovou rychlost na konstantní hodnotu duplikováním nebo odstraňováním snímků.

Hodnoty datových toků RTSP streamů jsou ve všech případech přijatelné pro přenos Internetem.

	datový tok [kB/s]	snímková rychlost [s^{-1}]	rozlišení [px]
TCP stream z `image_server`	460,8	22,6	1296×960
snímky z kamery přes HTTP	1 400	22,6	1296×960
RTSP MPEG4 – vyšší kvalita	40,6	10	648×480
RTSP Theora	28,4	10	648×480
RTSP MPEG4 – nižší kvalita	19,7	5	324×240

Tabulka 6.1.: Datové toky

V tabulce 6.2 jsou uvedeny velikosti záznámů – bezpečnostní JPEG snímky a 30 minutové videozáznamy. Videozáznamy dosahují velikosti 150 – 180 MB/30 min., při rozlišení 1296×960 px. Velikost souborů je možné ovlivnit parametrem `bitrate` u elementu `ffenc_mpeg4` v aplikaci `stream_storage`. Soubory jsou uchovávány po dobu 14 dní na serveru, poté jsou automaticky smazány. Při 100% využití 24 hodin denně by záznamy z jedné kamery za 14 dní zabíraly přibližně 100 GB.

	velikost [MB]	rozlišení [px]
bezpečnostní snímek JPEG	0,08 – 0,1	1296×960
30 min. záznam MPEG4	cca 150 – 180	1296×960 (10 snímků/s)

Tabulka 6.2.: Velikosti záznamů

Z porovnání komprese MPEG4 a Theora v tabulce 6.1 je patrné, že komprese Theora dosahuje lepších kompresních poměrů. Při zkouškách různých kompresí v aplikaci `stream_storage` se ale Theora neosvědčila, zřejmě z důvodu, že vyžaduje přesně časovaný datový tok, který není zaručen. Po uložení záznamu v kompresi Theora soubor neodpovídal svou časovou délkou době nahrávání. Komprese MPEG4 se jeví v současné konfiguraci jako nejvhodnější.

Literatura

[1] *Webové stránky projektu SyRoTek.* `http://syrotek.felk.cvut.cz`

[2] *Dokumentace programu GStreamer* [online]. `http://gstreamer.freedesktop.org/`

[3] *Dokumentace kamer ELPHEL* [online]. `http://wiki.elphel.com/`

[4] *IP multicast* [online]. `http://cs.wikipedia.org/wiki/IP_multicast`

[5] *Camera Calibration Toolbox for Matlab* [online]. `http://www.vision.caltech.edu/bouguetj/calib_doc/`

[6] Boyanique C., Pinard E. *Rééchantillonnage d'images, Traitement Numérique du Signal* [online]. `http://www.raceme.org/imac/redim/`

[7] Independent JPEG Group. *Webové stránky projektu.* `http://www.ijg.org/`

[8] Faigl J., Chudoba J., Košnar K., Saska M, Kulich M. et al. *SyRoTek – A Robotic System for Education* In: AT&P journal PLUS 2. 2010, p. 31-36. ISSN 1336-5010.

[9] Chudoba J., Faigl J., Kulich M., Košnar K., Krajník T. et al. *A TECHNICAL SOLUTION OF A ROBOTIC E-LEARNING SYSTEM IN THE SYROTEK PROJECT* In: 3rd International Conference on Computer Supported Education. Setúbal: INSTICC Press, 2011, p. 412-417. ISBN 978-989-8425-49-2.

[10] Kulich M., Faigl J., Chudoba, J., Košnar K. *A Visualization System for Teaching Intelligent Mobile Robotics in SyRoTek* [online]. `http://syrotek.felk.cvut.cz/publications`

[11] Katedra řídící techniky FEL ČVUT. *Učební materiál k předmětu POS* [online]. `http://dce.felk.cvut.cz/pos/`

[12] Outrata J. Synchronizace v OS Linux. *Učební materiál, Katedra počítačů UP Olomouc* [online]. `http://phoenix.inf.upol.cz/~outrata/courses/os2/texts/synch1_l.html`

[13] Université Paul Sabatier, Toulouse. *Učební materiál – Architecture TCP/IP* [online]. `http://www.httr.ups-tlse.fr/pedagogie/cours/tcp-ip/`

[14] Dlouhý, M. *Úvod do mobilní robotiky – předmět AIL028* [online]. `http://robotika.cz/guide/umor07/cs`

[15] Hanzl, T. (2005). *3D model mostu.* VUT Brno, Fakulta stavební, Ústav geodézie.

[16] Faigl J., Chudoba J., Kulich M., Mázl R., Košnar K., Přeučil L., Štěpán P. (2008). *Detailní koncepce systému SyRoTek.* Technická zpráva. Gerstner Laboratory, FEL ČVUT Praha.

Přílohy

A. Zdrojový kód RTSP serveru

Výpis A.1: Příklad jednoduchého RTSP serveru

```
#include <gst/gst.h>
#include <gst/rtsp-server/rtsp-server.h>

static gboolean timeout(GstRTSPServer *server, gboolean ignored) {
   GstRTSPSessionPool *pool;
   pool = gst_rtsp_server_get_session_pool ( server );
   gst_rtsp_session_pool_cleanup ( pool );
   g_object_unref ( pool );
   return TRUE;
}

int main ( int argc, char *argv[] ) {
   gchar *retezec;
   GMainLoop *loop;
   GstRTSPServer *server;
   GstRTSPMediaMapping *mapping;
   GstRTSPMediaFactory *factory;

   gst_init ( &argc, &argv );

   loop = g_main_loop_new ( NULL, FALSE );
   server = gst_rtsp_server_new ();

   gst_rtsp_server_set_service (server, "8555");

   mapping = gst_rtsp_server_get_media_mapping ( server );
   factory = gst_rtsp_media_factory_new ();

   retezec = g_strdup_printf ( "( videotestsrc ! ffmpegcolorspace \
                  ! ffenc_mpeg4 ! rtpmp4vpay name=pay0 pt=96 )" );

   gst_rtsp_media_factory_set_shared ( factory, TRUE );
   gst_rtsp_media_factory_set_launch ( factory, retezec );
   g_free ( retezec );

   gst_rtsp_media_mapping_add_factory ( mapping, "/test", factory );
   g_object_unref ( mapping );

   if ( gst_rtsp_server_attach ( server, NULL ) == 0 )
      goto failed;

   g_timeout_add_seconds ( 2, ( GSourceFunc ) timeout, server );
```

```
    g_main_loop_run ( loop );
    return 0;
    /* ERRORS */
    failed: {
        g_print ( "failed to attach the server\n" );
        return -1;
    }
}
```

B. Adresářová struktura CD přílohy

Výpis B.1: Výpis adresáře se zdrojovými soubory

```
gst-rtsp-0.10.8.tar.bz2
gst-template.tar.bz2
gst-rtsp-0.10.8/
gst-template/
  gst-app/
    /src/
  gst-plugin/
stream_storage/ (-> gst-template/gst-app/src/)
RTSP_server/
image_server/
  src/
    include/
    httpd/
    jpeg-8d/
    jpegsrc.v8d.tar.gz
  build/
stream_control/
```

C. Komunikace s GST-RTSP serverem

Výpis C.1: Počátek komunikace s RTSP serverem

```
Opening connection to 147.32.84.212, port 8013...
...remote connection opened

Sending request:
OPTIONS rtsp://syrotek.felk.cvut.cz:8013/k013T RTSP/1.0
CSeq: 2
User-Agent:
LibVLC/1.1.12 (LIVE555 Streaming Media v2011.07.21)
Received 183 new bytes of response data.

Received a complete OPTIONS response:
RTSP/1.0 200 OK
CSeq: 2
Public: OPTIONS, DESCRIBE, GET_PARAMETER, PAUSE,
        PLAY, SETUP, SET_PARAMETER, TEARDOWN
Server: GStreamer RTSP server
Date: Tue, 01 May 2012 14:54:15 GMT


Sending request:
DESCRIBE rtsp://syrotek.felk.cvut.cz:8013/k013T RTSP/1.0
CSeq: 3
User-Agent:
LibVLC/1.1.12 (LIVE555 Streaming Media v2011.07.21)
Accept: application/sdp
Received 475 new bytes of response data.

Received a complete DESCRIBE response:
RTSP/1.0 200 OK
CSeq: 3
Content-Type: application/sdp
Content-Base: rtsp://syrotek.felk.cvut.cz:8013/k013T/
Server: GStreamer RTSP server
Date: Tue, 01 May 2012 14:54:15 GMT
Content-Length: 272
v=0
o=- 1188340656180883 1 IN IP4 147.32.84.212
s=Session streamed with GStreamer
i=rtsp-server
e=NONE
t=0 0
a=tool:GStreamer
a=type:broadcast
```

```
a=control:*
a=range:npt=now-
m=video 0 RTP/AVP 96
c=IN IP4 147.32.84.212
a=rtpmap:96 JPEG/90000
a=control:stream=0


Sending request:
SETUP rtsp://syrotek.felk.cvut.cz:8013/k013T/stream=0 RTSP/1.0
CSeq: 4
User-Agent:
LibVLC/1.1.12 (LIVE555 Streaming Media v2011.07.21)
Transport: RTP/AVP/TCP;unicast;interleaved=0-1
Received 197 new bytes of response data.

Received a complete SETUP response:
RTSP/1.0 200 OK
CSeq: 4
Transport: RTP/AVP/TCP;unicast;interleaved=0-1;
           ssrc=ED6EB16B;mode="PLAY"
Server: GStreamer RTSP server
Session: ydetumcuxtgyjbjq
Date: Tue, 01 May 2012 14:54:15 GMT


Sending request:
PLAY rtsp://syrotek.felk.cvut.cz:8013/k013T/ RTSP/1.0
CSeq: 5
User-Agent:
LibVLC/1.1.12 (LIVE555 Streaming Media v2011.07.21)
Session: ydetumcuxtgyjbjq
Range: npt=0.000-

Received a complete PLAY response:
RTSP/1.0 200 OK
CSeq: 5
RTP-Info: url=rtsp://syrotek.felk.cvut.cz:8013/k013T/stream=0;
          seq=65135;rtptime=1832178079 Range: npt=0.000000-
Server: GStreamer RTSP server
Session: ydetumcuxtgyjbjq Date: Tue, 01 May 2012 14:54:16 GMT


Sending request:
GET_PARAMETER rtsp://syrotek.felk.cvut.cz:8013/k013T/ RTSP/1.0
CSeq: 6
User-Agent:
LibVLC/1.1.12 (LIVE555 Streaming Media v2011.07.21)
Session: ydetumcuxtgyjbjq
Received a complete GET_PARAMETER response:
RTSP/1.0 200 OK
```

```
CSeq: 6
Server: GStreamer RTSP server
Date: Tue, 01 May 2012 14:54:16 GMT


Sending request:
TEARDOWN rtsp://syrotek.felk.cvut.cz:8013/k013T/ RTSP/1.0
CSeq: 7
User-Agent:
LibVLC/1.1.12 (LIVE555 Streaming Media v2011.07.21)
Session: ydetumcuxtgyjbjq
```