VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTELIGENTNÍ BEZPEČNOSTNÍ KAMERA ZALOŽENÁ NA RASPBERRY PI

INTELLIGENT SECURITY CAMERA BASED ON RASPBERRY PI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PETER VINARČÍK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ZDENĚK MATERNA, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Vinarčík Peter**
Program: Informační technologie
Název: **Inteligentní bezpečnostní kamera založená na Raspberry Pi**
Intelligent Security Camera Based on Raspberry Pi
Kategorie: Vestavěné systémy
Zadání:

1. Proveďte rešerši existujících řešení se zaměřením na využití Raspberry Pi, prvků počítačového vidění a standardů pro bezpečnostní kamery (např. ONVIF).
2. Vyberte vhodný kamerový modul.
3. Navrhněte koncepci kamery, ověřte základní funkčnost.
4. Implementujte navrženou koncepci.
5. Proveďte testování.
6. Zdrojové kódy a dokumentaci publikujte na GitHubu.
7. Vytvořte stručný plakát nebo video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Dle doporučení vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Materna Zdeněk, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Cieľom práce je vytvoriť Pan-Tilt-Zoom bezpečnostnú kameru skonštruovanú pomocou Raspberry Pi a kamerového modulu. Výsledná kamera podporuje podstatnú časť ONVIF štandardu. Softvér kamery je implementovaný v Pythone a čiastočne v Node.js. Kamerový modul zachytáva zorné pole o veľkosti 200 stupňov. Zachytený obraz je pomocou prvkov počítačového videnia - knižnica OpenCV pre Python prevedený na simuláciu Pan-Tilt-Zoom kamery. Výsledný obraz bude možné ovládať a pozorovať v sieťovom nahrávacom zariadení Shinobi.

Abstract

The aim of this thesis is to create Pan-Tilt-Zoom security camera which will be constructed from Raspberry Pi and camera module. Camera tries to be compliant with ONVIF standard. Software of this camera is implemented in Python and partially in Node.js. Camera module captures 200 degrees field of view. Captured image is transferred as simulation of Pan-Tilt-Zoom camera with the help of computer vision - library OpenCV for Python. The resulting image will be possible to control and watch in network video recorder called Shinobi.

Kľúčové slová

raspberryPi, bezpečnosť, kamera, AI, domáce zabezpečenie, python, raspbian, OpenCV, GStreamer, počítačové videnie

Keywords

raspberryPi, security, camera, AI, homesecurity, python, raspbian, OpenCV, GStreamer, computer vision

Citácia

VINARČÍK, Peter. *Inteligentní bezpečnostní kamera založená na Raspberry Pi.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Zdeněk Materna, Ph.D.

Inteligentní bezpečnostní kamera založená na Raspberry Pi

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Zdeňka Materny, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Peter Vinarčík
11. mája 2021

Poďakovanie

Rád by som sa poďakoval vedúcemu mojej bakalárskej práce, pánovi Ing. Zdeňkovi Maternovi, Ph.D. za vedenie práce, odbornú konzultáciu, užitočné rady, pripomienky a za čas, ktorý mi venoval pri vypracovávaní práce. Tiež by som chcel poďakovať svojej priateľke, mame a starej mame, ktoré ma po celý čas podporovali.

Obsah

Kapitola 1

Úvod

Nasledujúca bakalárska práca sa zaoberá vytvorením a naprogramovaním kamery, respektíve kamerového softvéru, ktorý pri použití Raspberry Pi spolu s pripojeným kamerovým modulom bude schopný simulovať pan-tilt-zoom kameru. Použitý kamerový modul pritom bude typu rybie oko, pričom veľkosť zorného pola bude 200 stupňov.

Takýto obraz je pre ľudské oko ťažšie čitateľný, než je tomu pri bežných kamerových moduloch pre Raspberry Pi, respektíve pri bežných kamerách vo všeobecnosti, ktorých zorné pole je najčastejšie o veľkosti približne 60 až 75 stupňov - pre porovnanie, hardvérová špecializácia Camera Module v2 pre Raspberry Pi [15], čo je vo všeobecnosti najrozšírenejší kamerový modul, uvádza, že má horizontálne zorné pole o veľkosti 62.2 stupňov a vertikálne zorné pole o veľkosti 48.8 stupňov. Takéto zorné pole je pre zdravé ľudské oko, ktoré je schopné zachytávať okolie o veľkosti 120 stupňov (v prípade binokulárneho videnia) [5] prijateľnejšie než je tomu pri fish-eye obraze. Výsledná kamera preto bude obsahovať softvér, ktorý zachytávaný obraz prevedie pomocou prvkov počítačového videnia na vyššie zmienenú simuláciu pan-tilt-zoom kamery. V praxi to bude možné pozorovať tak, že obraz z rybieho oka bude narovnaný a premietaná bude z neho iba časť, ktorá bude vyzerať ako zachytená bežnou kamerou, pričom k ostatným častiam obrazu sa bude možné dostať práve pomocou simulovaného pohybu s kamerou.

Ďalším cieľom práce je, že výsledný produkt sa snaží čo najpresnejšie implementovať ONVIF štandard[1] a priblížiť sa tak štandardu a funkciám, ktoré poskytujú bezpečnostné kamery.

Bezpečnostné systémy sú dnes úplne bežnou súčasťou vo všetkých odvetviach kdekoľvek na svete. V posledných rokoch sa tak dostávajú do popredia aj bezpečnostné systémy vhodné pre inštaláciu a použitie v domácnosti. Vymoženosti súčastnosti tiež ľuďom umožňujú hlbšie skúmať rôzne technické sféry a čoraz viac ľudí sa snaží v domácnostiach preferovať kutilstvo a namiesto toho aby daný produkt kúpili od komerčnej značky, snažia sa produkt zhotoviť doma. Tento koncept je nazývaný DIY (do it yourself). Je však tento koncept možný aplikovať aj pri vytváraní bezpečnostného systému?

Odpoveď je určite áno, avšak pri vytváraní takéhoto systému môže človek bez znalosti programovania naraziť na zásadný problém. Mimo hardvéru totiž kamera musí jednoznačne obsahovať aj funkčný softvér, ktorý Raspberry Pi spolu so zapojeným kamerovým modulom premení na funkčný celok. Výsledok riešenia tejto práce by mal poskytnúť užívateľovi oproti nateraz dostupným riešeniam inováciu v podobe spomínaného pan-tilt-zoom softvéru a čiastočnú kompatibilitu s ONVIF štandardom.

[1] ONVIF: https://www.onvif.org/

Kapitola 2

Existujúce riešenia

Práve kamerové systémy sú prvkom bezpečnosti, po ktorom sa primárne siaha pri budovaní bezpečnostného systému. Pomocou kamier je v dnešnej dobe pomerne jednoduché detekovať akýkoľvek neočakávaný pohyb či pokus o vniknutie. Kamery tiež môžu byť využívané k ochrane súkromného majetku a pri prípadných škodách na ňom, môžu byť záznamy poskytnuté pre najrôznejšie účely.

V tejto kapitole budú predstavené niektoré riešenia, ktoré ponúkajú bezpečnostné kamery v dnešnej dobe.

2.1 Komerčne dostupné riešenia

V tejto kapitole budú predstavené typy, technológie a porovnanie dostupných komerčných kamier.

2.1.1 CCTV Kamery

Kamery, ktoré sú označované ako CCTV alebo celým ńazov Closed Circuit Television, sú označované ako systém analógových kamier.

K záznamovému zariadeniu je pripojená kamera, respektíve sada kamier. Kamery sú pripojené pomocou koaxiálneho káblu a celý systém je následne závislý práve na záznamovom zariadení. V prípade jeho nefunkčnosti je celý systém taktiež nefunkčný.

Vzhľadom k dnešným kamerovým systémom, je tento typ kamerového systému považovaný za zastaralý a to hneď z niekoľkých vážnych dôvodov. Nakoľko je prenos závislý na prenose signálu cez kábel, môže byť signál na väčšie vzdialenosti prerušený. Ďalším dôležitým faktorom je nemožnosť prenášať zvuk. Naopak výhodou systému môžu byť nižšie náklady.

Aj napriek nevýhodám, ktoré tento systém prináša je stále vo veľkom využívaný, predovšetkým kvôli nízkym nákladom na celý kamerový systém.

Funkčnosť takejto kamery je založená na zachytávaní analógového signálu, ktorý je následne prevedený do formy digitálnej. Konverziu vykonáva DVR, na ktorom je následne možné zachytený snímok prehliadať pomocou pripojeného monitoru.

2.1.2 IP Kamery

K samotnej práci je však bližší novší typ kamerových zariadení. Ide o IP kamery, ktoré fungujú na sieťovom protokole IP. Vďaka tejto možnosti jednotlivé kamery nie sú závislé na

centrálnom zariadení (ako napríklad DVR pri kamerách analógových) a tak sa významne zvyšuje percento ochrany proti nefunkčnosti systému. Kamery navyše vedia pracovať aj samostatne vzhľadom na ostatné kamery.

Keďže sa jedná o kamery fungujúce na sieťovom protokole, je pochopiteľne možné samotný systém nastavovať cez prístup na internet. Okrem nastavovania kamier je možné určité kamery cez internet aj ovládať.

Ďalšou významnou zmenou oproti kamerám analógovým je kvalita prenášaného obrazu. Zatiaľ čo pri analógových kamerách je kvalita obmedzená na 0.4 megapixelov a počet snímkov za sekundu je obmedzený na 25, pri IP kamerách je možné nájsť kamery s omnoho vyššou kvalitou (pre príklad až 8 megapixelov čo je v prepočte až 20 násobne vyššie). Obraz je možné pozorovať aj pri 60 snímkoch za sekundu, čo má za následok plynulý obraz bez trhania, ktorý je možné nájsť pri systémoch analógových.

Tabuľka porovnania analógových kamier a IP kamier		
Vlastnosť	Analógový systém kamier	IP systém kamier
Rozlíšenie kamier	0.4 MPix	Štandartne 1.3 - 2 MPix
Citlivosť kamier	Vyššia	Nižšia
Snímková frekvencia	25 FPS	6 - 60 FPS
Detekcia pohybu	Áno (často len pri použití záznamového zariadenia)	Áno
Inteligentná analýza	Nie	Áno
Možnosť sledovať cez internet a na mobilných zariadeniach	Áno	Áno
Nároky na diskovú kapacitu	Nižšia - jedna kamera pri plnej snímkovej frekvencii spotrebuje cca 20GB denne.	Vyššia - Jedna kamera v rozlíšení 2MPix pri plnej snímkovej frekvencii vyžaduje cca 100GB denne
Kabeláž	Vyhradená - káble nemožno využiť na prenos iných informácií, k jednej kamere niekedy musí viesť niekoľko káblov	Zdieľaná - Káble možno využiť aj na iné účely (napríklad pre pripojenie počítačov). Jeden kábel často prenáša niekoľko rôznych typov dát a môže slúžiť aj k prenosu napájania (Power over Ethernet).
Úroveň zabezpečenia	Nižšia	Vyššia
Štandardizácia	Vyššia	Nižšia
Finančné nároky	Nižšie	Vyššie

Tabuľka 2.1: **Tabuľka porovnania analógových kamier a IP kamier** - Tabuľka prebratá z [1]

2.2 Bezpečnostné kamery s využitím Raspberry Pi

Ako bolo spomínané v kapitole 2, v dnešnej dobe existuje viacero typov kamier. Vo vyššie zmieňovanej kapitole boli načrtnuté predovšetkým ich typy. V tejto práci je typ kamery zameraný na kameru, ktorá je vložená do CSI (Camera Serial Interface) portu - kapitola 3.2.1 a kapitola 3.3. Práve preto budú v nadchádzajúcej kapitole predstavené existujúce riešenia spojené s týmto typom kamery a Raspberry Pi.

2.2.1 motionEyeOS bezpečnostná kamera

Najčastejším riešením pre takýto kamerový systém založený na Raspberry Pi je prebratý z oficiálneho webu[1]. Obrovskou výhodou tohto riešenia je, že ponúka užívateľovi vytvoriť kamerový zabezpečovací systém bez nutnosti akejkoľvek znalosti programovacieho jazyka a tak jedinou časťou ostáva spojazdniť hardvérovú časť.

Prvoradé je teda zabezpečiť jednotlivé komponenty (kamerový modul a Raspberry Pi). V prípade, že kamera ktorá sa bude v tomto projekte používať nie je zakončená USB Portom ako tomu je aj v prípade tejto práce, musí užívateľ pripojiť kameru do Raspberry Pi, do takzvaného CSI slotu[2]. Detailnejší popis kamerového modulu a CSI slotu je možné nájsť v kapitole 3.

Po tom, čo je hardvérová časť dokončená, musí prebehnúť stiahnutie motionEyeOS[3] na SD kartu vloženú do samotného Raspberry Pi a všetko je pripravené na používanie.

Po načítaní motionEyeOS v prehliadači pomocou Raspberry Pi IP má užívateľ k dispozícii zachytený obraz z kamery. Samotný operačný systém poskytuje viacero rozšírení v podobe emailových notifikácií, nahrávania obrazu na cloudové služby, určenie pracovného času kamery či zachytávanie snímkov. Zmienený návod však tieto rozšírenia nijak nespomína a nepopisuje, takže užívateľ nemá bližšie informácie o práci s týmito rozšíreniami.

Toto riešenie je vzhľadom na výsledky vyhľadávania najpopulárnejšou možnosťou pre vytvorenie bezpečnostnej kamery a pri vyhľadávaní pojmov súvisiacich s Raspberry Pi a kamerovým systémom, je ako prvá odporúčaná práve vyššie popísaná možnosť.

2.2.2 Inteligentná motionEyeOS bezpečnostná kamera

Ďalšou populárnou voľbou, ku ktorej je možné sa dostať, je opäť, ako aj mnohé ostatné dostupné riešenia, založená na motionEyeOS.

Toto riešenie je však na rozdiel od riešenia v podkapitole 2.2.1 rozšírené o ďalšie nastavenia, ktoré výslednej kamere pridávajú inteligentné prvky ako je detekcia objektov v obraze, zasielanie emailov pri zachytení pohybu a podobne.

Ďalšou výhodou je tak ako aj pri prvom riešení fakt, že užívateľ stále nemusí využívať žiadny programovací jazyk, ani po ňom nie je požadované kopírovanie predpripravených skriptov a podobne[4].

[1] RPi bezpečnostná kamera z oficiálneho webu pre RPi: https://www.raspberrypi.org/blog/raspberry-pi-high-quality-security-camera/

[2] Zapojenie kamery do CSI slotu: https://www.raspberrypi.org/documentation/configuration/camera.md

[3] Oficiálny GitHub motionEyeOS: https://github.com/ccrisan/motioneyeos/wiki

[4] RPi kamera s detekciou pohybu: https://pimylifeup.com/raspberry-pi-security-camera/

2.2.3 Power over Ethernet kamera

Posledné riešenie je rozdielne od predošlých dvoch. Nie je postavené na operačnom systéme motionEyeOS, ale je vybavený Raspberry Pi OS, čo sa zhoduje s plánovanou implementáciou.

Toto riešenie popisuje Raspberry Pi Cloud IP kameru[5], ktorá podporuje PoE (Power over Ethernet). Výsledná kamera je prezentovaná ako možná lacná alternatíva ku komerčným IP kamerám, nakoľko bola údajne testovaná v maloobchodnom reťazci pričom jej čas behu bol 100 percent bez výpadkov. Nevýhodou oproti predchádzajúcim riešeniam je, že k jej zhotoveniu nestačí len samotné Raspberry Pi s kamerovým modulom, ale je potrebné zakúpenie PoE prepínaču, ktorý bude slúžiť ako zdroj kamery.

Celkový odhad ceny všetkých komponentov potrebných k zhotoveniu takéhoto typu kamery, je na úrovni 3400 českých korún, čo značne predražuje náklady v porovnaní s vyššie zmienenými riešeniami respektíve s cenou komponentov použitých v tejto práci.

Riešenie navyše obsahuje časti, v ktorých musí užívateľ či už aktualizovať určité balíčky, alebo konfigurovať sieťové súbory (napríklad `/etc/network/interfaces` a pod.). Nezanedbateľným faktom je tiež manuálna stránka, kedy nestačí len kamerový modul jednoducho pripojiť do CSI slotu, ale je nutná určitá manuálna modifikácia. Týka sa to predovšetkým prepojenia Raspberry Pi so spomínaným PoE prepínačom pomocou UTP kábla, ktorý je nutné vo väčšine prípadov skrátiť aby sa zmestil do skrinky pre kameru a podobne.

Výsledkom je teda kamera podporujúca PoE, avšak v porovnaní s predchádzajúcimi riešeniami zaostáva v čase prípravy, môže dôjsť k nesprávnemu mechanickému zachádzaniu s komponentami a trvalému poškodeniu (zbytočné výdavky na kúpu náhradnej súčiastky) a vlastne celková cena spojená s kúpou jednotlivých komponentov.

2.3 Nedostatky dostupných riešení

Po bližšom preskúmaní a testovaní dostupných riešení, aj napriek ich vysokej popularite a jednoduchosti používania či inštalácie, som narazil na niekoľko nedostatkov respektíve chýbajúcich rozšírení, ktoré by sa častokrát bežnému používateľovi mohli zísť.

Veľkým nedostatkom je, že ani jedna dostupná implementácia nepočíta s možnosťou ovládať kameru. Samozrejme je možné zakúpenie kamerového modulu podporujúcu pan-tilt-zoom systém[6], avšak možnosť ovládania kamery bude previazaná na softvér dodaný výrobcom a nebude takúto kameru možné ovládať z nahrávacieho zariadenia spolu s ostatnými kamerami. Odstránenie takéhoto nedostatku sa významne približuje výsledku, ktorý je cieľom tejto práce. Práve výsledná implementácia má poskytnúť možnosť používať bezpečnostnú kameru s rôznymi objektívmi. Možným objektívom bude môcť byť klasický objektív, ako tomu je aj pri vyššie spomenutých riešeniach, ktorý bude pôsobiť ako statický a bude zameraný na jeden statický bod.

Čo však inovatívne prinesie implementácia oproti predošlým dostupným riešeniam, je možnosť podpory pre Fish-eye objektív, alebo teda objektív, ktorý zaberá priestor s rozsahom až 200 stupňov. Pomocou tohto objektívu bude simulovaná pan-tilt-zoom kamera a tak užívateľ bude môcť zaberať väčšie množstvo priestoru pomocou jednej kamery. Táto

[5] RPi kamera s podporou PoE: https://www.instructables.com/Raspberry-Pi-Cloud-IP-Camera-with-POE/

[6] Pan-Tilt-Zoom kamerový modul: https://rpishop.cz/kamery/2917-arducam-8mp-ptz-kamera-se-zakladnou-pro-raspberry-pi-43b3.html

inovácia sa tiež môže podieľať na šetrení výdavkov v závislosti na množstve kamier, ktoré je nutné zakúpiť, nakoľko jednou kamerou bude možné pokryť viacero uhlov.

Ďalšou nevýhodou je, že ani jedno zo spomenutých (aj nespomenutých) dostupných riešení sa nezmieňuje o žiadnom kamerovom štandarde. Riešenia sú postavené na jednoduchom zložení kamery a jednoduchom spojazdnení celkového softvéru a nejestvuje náznak, že by sa akékoľvek riešenie pokúsilo priblížiť presadzovaným kamerovým štandardom a bolo ju možné používať s ostatnými bezpečnostnými kamerami iných výrobcov. Ako bolo spomenuté v kapitole 1, cieľová implementácia sa snaží túto nevýhodu dostupných riešení odstrániť tým, že sa snaží čo najlepšie naimplementovať ONVIF štandard vo svojom softvéri.

Na druhú stranu je výsledná implementácia práce inšpirovaná výhodami, ktoré dostupné riešenia ponúkajú a to v podobe toho, že kamera nemá očakávať žiadne väčšie technické znalosti ako je programovanie. Ďalšou inšpiráciou je koncept "plug and play", kedy užívateľ len pripojí kameru k zdroju a okamžite k nej môže pristúpiť v zariadení v sieti pomocou sieťového nahrávacieho zariadenia (motionEye). Tento koncept by mal byť v implementácii rozšírený o určitú formu bezpečnosti - minimálne v podobe autorizačných údajov pri prístupe do sieťového nahrávacieho zariadenia, nakoľko ku kamere bežiacej na motionEyeOS je v konfigurácii uvedenej v návodoch vyššie možné pristúpiť z ľubovoľného zariadenia v sieti pomocou url `http://<RasberryPiIP>:<port>` čo má značný vplyv na celkovú bezpečnosť.

Kapitola 3

Raspberry Pi a komponenty

V nasledujúcej kapitole budú detailnejšie rozobraté jednotlivé použité komponenty zmienené z úvodu.

Konkrétne ide o komponenty nasledujúce:

- Raspberry Pi vo verzii 4B[1]
- Waveshare RPi kamera[2]

3.1 Raspberry Pi

Počítač Raspberry Pi je malý počítač, veľkostne prirovnávaný ku veľkosti kreditnej karty. Hlavným a významným faktom však predovšetkým ostáva, že poriaďovacia cena tohto malého počítača je extrémne nízka. Raspberry Pi je dostupný v štyroch (respektíve piatich) generáciách[3]. Každá generácia je označená číslom 1 (najstaršia) až 4 (najnovšia) pričom každá generácia môže byť rozdelená na modely A, B, A+, B+.

Model, ktorý je použitý v tejto práci je model B 4 generácie, s veľkosťou RAM 2GB. Tento model je tiež dostupný v konfigurácii so 4GB RAM, prípadne až s 8GB RAM.

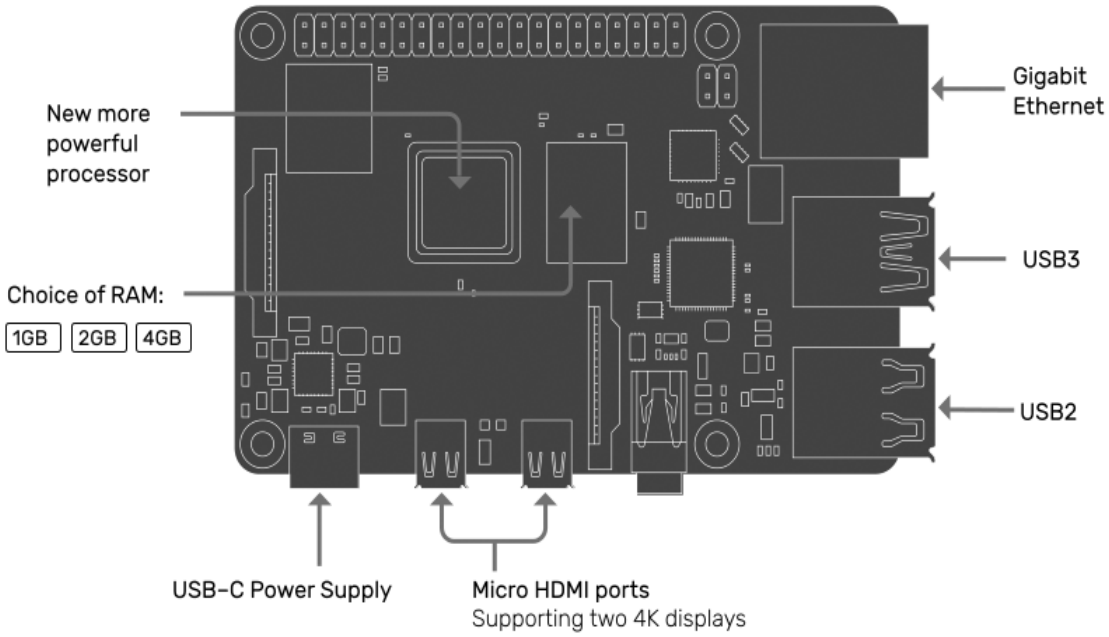
Pre prácu bol zvolený model s veľkostou 2GB v dôsledku:

- Cena, za ktorú je dostupná 2GB verzia, je až o 500 českých korún nižšia ako tomu je pri 4GB verzii, respektíve o 1300 českých korún nižšia v prípade verzie 8GB.
- Nakoľko je cieľom práce nie len priniesť inováciu v podobe inteligentnej kamery, je tiež žiadúce, aby bola dostupná za cenu nižšiu, než je cena produktov priemyselných bezpečnostných kamier.

3.1.1 Operačný systém

Najčastejšie využívaným operačným systémom pre Raspberry Pi je jednoznačne Raspberry Pi OS, ktorý je využitý aj v aktuálnej práci.

[1] Raspberry Pi 4B: https://rpishop.cz/518-raspberry-pi-4b
[2] Waveshare RPi kamera: https://www.waveshare.com/rpi-camera-m.htm/
[3] Raspberry Pi - prehľad: https://www.raspberrypi.org/products/



Obr. 3.1: **Jednoduchá schéma Raspberry Pi 4B** - Obrázok je prevzatý z oficiálnej stránky pre Raspberry Pi 4B

Je vyvíjaný a odporúčaný výrobcom Raspberry Pi, nakoľko pre bežné projekty a bežné používanie je tento operačný systém najlepšie optimalizovaný vzhľadom na hardvérovú časť Raspberry Pi.

Raspberry Pi OS je založený na Debiane [16] a tak je ho možné nájsť aj pod pojmom Raspbian. Samotný operačný systém prichádza s množstvom predkompilovaného softvéru vrátane podpory programovacích jazykov ako Python, C, C++, Java či Ruby.

3.2 Pi Camera Module

Pi Camera Module sú v jednoduchosti povedané také kamery, ktoré sú ľahko prenosné a hlavne sú schopné podporovať Raspberry Pi. Kamery tohto typu komunikujú s Raspberry Pi pomocou MIPI Camera Serial Interface protokolu - kapitola 3.3. Bežne sa tento kamerový modul používa pre strojové učenie, je však tiež nevyhnutnou súčasťou projektov týkajúcich sa bezpečnosti, konkrétne napríklad bezpečnostných kamier. Mimo tohto odvetvia sa s takýmito kamerami možno stretnúť napríklad v projektoch týkajúcich sa dronov[4].

3.2.1 Waveshare RPi kamera

Táto kamera je určená pre všetky typy Raspberry Pi. Zvolená však bola predovšetkým kvôli pomeru ceny a výkonu, keďže za cenovku približne 800 korún je k dispozícii kvalitná kamera, ktorá je schopná nahrávať 30 snímkov za sekundu pri kvalite 1080p a 60 snímkov pri kvalite 720p. Táto kvalita a plynulosť obrazu je viac než dostačujúca pre účely domácej bezpečnostnej kamery.

[4]Využiteľnosť CSI kamery na dronoch pomocou FlytOS: https://flytbase.com/flytos/

Obr. 3.2: **Schéma kamerového modulu pre Raspberry Pi**

Ďalšou dôležitou súčasťou tejto kamery je jeho objektív, ktorý je typu rybie oko. Dostupné (a dokonca drahšie) fish-eye kamery pre Raspberry Pi poskytujú zorné pole 130 až približne 170 stupňov. Avšak Waveshare RPi kamera typu M poskytuje zorné pole vo veľkosti až 200 stupňov čo má za následok, že kamera je schopná zachytávať nezanedbateľne väčší priestor než by tomu bolo pri iných verziách kamier takéhoto typu.

Typy kamier			
Názov	Zorné pole (v °)	Typ zapojenia	Cena (v Kč)
Waveshare RPi kamera (B)	60,6	CSI	535
Raspberry Pi kamera V2	72	CSI	659
RPi 4 Fish-eye kamera s nočným videním	130	CSI	865
Waveshare RPi kamera (I)	170	CSI	825
ELP Fisheye RPi Security Camera	180	USB	1 174
Waveshare RPi kamera (M)	200	CSI	799

Tabuľka 3.1: Tabuľka popisujúca typy jednotlivých kamier, vrátane ich zorného poľa, ceny a typu zapojenia

Do Raspberry Pi je kamera pripojená pomocou CSI slotu, respektíve pomocou plochého 15 pinového káblu smerujúceho z kamery do pripraveného slotu.

Taktiež nutno podotknúť, že v prípade tejto kamery je nemožnosť automatického zaostrovania, nakoľko kamera disponuje manuálnym zaostrovaním pomocou otáčania objektívu do strán. V praxi to znamená, že pred upevnením kamery na miesto nahrávania bude potrebné previesť manuálnu konfiguráciu v podobe otestovania zaostrenia.

3.3 Camera Serial Interface

Samotné Raspberry Pi, má pre pripojenie kamery dostupné nie len USB porty, ale taktiež CSI (Camera Serial Interface) port[5].

[5]CSI-2 - popis z oficiálnej stránky pre RPi: https://www.raspberrypi.org/documentation/linux/software/libcamera/csi-2-usage.md/

Obr. 3.3: **Waveshare kamera (M) pripojená do RPi**

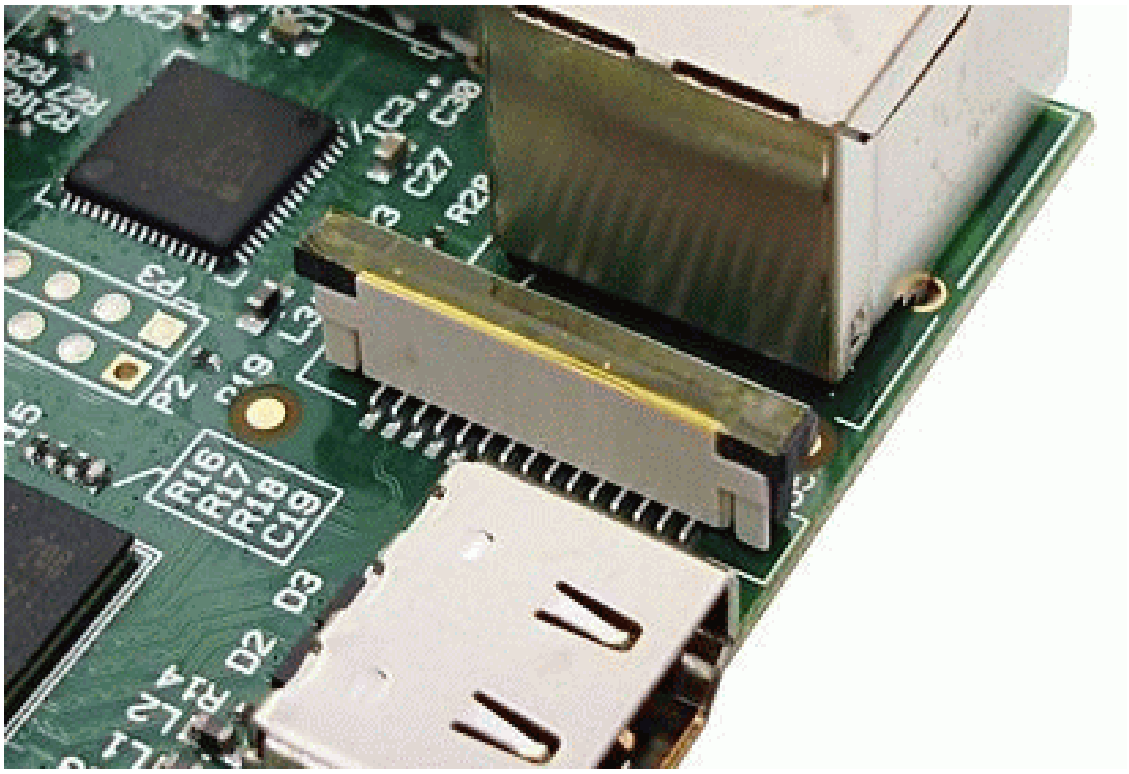
Tento port je kamerový port, ktorý poskytuje zbernicu, ktorá slúži k prepojeniu dvoch zariadení. V aktuálnom prípade ide o spomínaný kamerový modul Waveshare RPi Camera (M) a Raspberry Pi.

Cieľom tohto rozhrania bolo, respektíve je, štandardizovanie zapájania kamerových modulov do procesorov. Verzia CSI-2 tohto rozhrania bola významná a populárna, čo malo za následok, že bola používaná na takmer každom mobilnom zariadení, ktoré bolo možné nájsť. Významným faktom pri tomto rozhraní ostáva, že je schopné dosahovať vysokých prenosových rýchlostí aj napriek tomu, že veľkosť tohto rozhrania je len 24 x 25 x 9 milimetrov. Práve preto je hojne využívaný práve v mobilnom priemysle, kde rozmery a v neposlednom rade váha hrajú významnú rolu.

CSI-2 využíva D-PHY (technológia pre prácu s fyzickou vrstvou L1). Detailnejšia špecifikácia pre CSI-2 rozhranie je rozdelená na viacero vrstiev.

CSI-2 rozdelenie vrstiev:

- Fyzická vrstva - D-PHY
- Lane Merger Layer
- Nízkoúrovňová vrstva protokolu
- Prevodná vrstva z pixelu na byte
- Aplikačná vrstva



Obr. 3.4: **CSI-2 Port na Raspberry Pi**

Kapitola 4

ONVIF

Skratka ONVIF[1] znamená - Open Network Video Interface Forum a jeho úlohou je tvoriť štandard pre komunikáciu bezpečnostných systémov založených práve na IP. Tento štandard využívajú IP kamery a záznamové jednotky. Tým, že jednotlivé komponenty bezpečnostného systému používajú štandard ONVIF je zaistená vzájomná kompatibilita týchto komponentov a významnou vecou v tomto štandarde ostáva, že vďaka nemu je umožnené komunikovať komponentom rôznych značiek od rôznych výrobcov.

V rámci štandardu ONVIF je špecifikovaných viacero profilov, pričom každý profil má svoju jedinečnú úlohu.

Profily:

- Profil A - slúži pre prístup k nastaveniam konfigurácie,
- Profil C - slúži ako správca udalostí,
- Profil G - slúži pre správu úložiska,
- Profil Q - slúži pre rýchlu a jednoduchú inštaláciu,
- Profil S - slúži na základné streamovanie videa z kamery,
- Profil T - slúži na pokročilé streamovanie videa z kamery.

Profil týkajúci sa tejto práce je predovšetkým profil S. Tento profil ako jeho popis vyššie napovedá, slúži pre základný prenos videa z kamery užívateľovi. Ako užívateľ v tomto profile vystupuje video softvér, v aktuálnom prípade to bude ZoneMinder a Shinobi, ktorých popis bude uvedený v neskoršej fáze. Ako zariadenie je pre účely práce zvolená práve Waveshare RPi Camera (M) s objektívom typu rybie oko z kapitoly 3.2.1, ktorá bude pripojená do Raspberry Pi.

Profil S taktiež podľa definície špecifikovanej na oficiálnej stránke ONVIF štandardu[2] zahŕňa podporu pre pan-tilt-zoom kameru, ktorej simulácia je cieľom tejto bakalárskej práce.

Vhodným profilom pre takúto kameru je aj profil T, ktorý sa radí medzi pokročilejšie profily práve vďaka svojej rozšírenej funkcionalite, avšak pre potreby tejto práce nebol braný v úvahu, nakoľko inovácie, ktoré poskytuje oproti profilu S nie sú nutným faktorom

[1] Oficiálna stránka ONVIF štandardu: https://www.onvif.org/
[2] ONVIF Profil S: https://www.onvif.org/profiles/profile-s/

pre zhotovenie finálneho produktu. Do budúcna môže byť plánované rozšírenie, respektíve úprava kamery spĺňajúcej profil S na kameru spĺňajúcu profil T. Na nasledujúcej tabuľke je vyzobrazené porovnanie jednotlivých častí profilu S a profilu T.

Vlastnosť	Profil S	Profil T
H265	Nie	Áno
Kontrola zaostrovania	Nie	Áno
Obojsmerný audio prenos	Nie	Áno
Alarm systém na základe pohybu	Nie	Áno
Streaming metadát	Nie	Áno
Snapshot[3]	Nie	Áno

Tabuľka 4.1: Popis rozdielov medzi ONVIF Profil S a Profil T

4.1 Profil S

Ako bolo spomenuté v úvode kapitoly 4, profil S je v ONVIF štandarde špecifikovaný ako najzákladnejšia položka pre streaming videa z kamery na klientske zariadenie. Tento profil však obsahuje viac potrebnej konfigurácie než je tomu na prvý pohľad zrejmé.

Zariadenie profilu S (ďalej len ako kamera), musí obsahovať systémové nastavenia. Rovnako je tomu aj pre klientske zariadenie profilu S (ďalej len ako NVR). S týmito systémovými nastaveniami úzko súvisí ďalšia nevyhnutná položka, ktorú neobsahuje žiadny ďalší ONVIF profil a tým je **User Authentication pomocou WS-Username Token a Digest Authentication**.

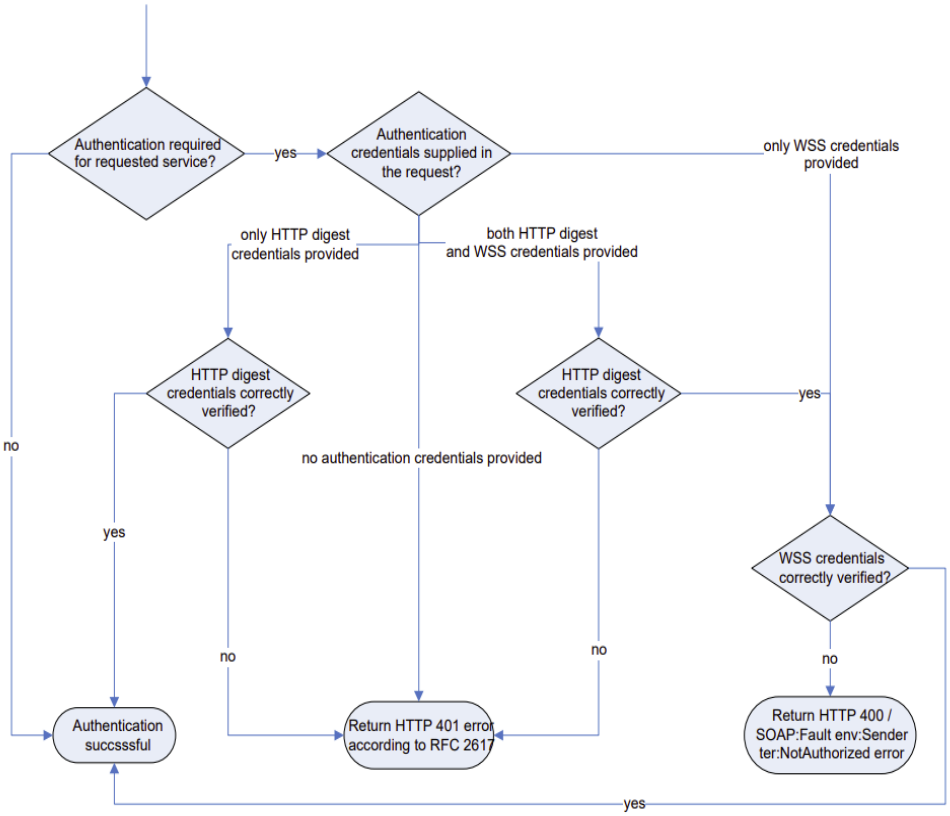
4.1.1 Základy zabezpečenia

WS-Username Token je definovaný v časti týkajúcej sa autentizácie a v ONVIF štandarde má spĺňať zabezpečenie služieb, ktoré sa spomínajú v tejto, ale aj v nasledujúcich kapitolách (konkrétne media, discovery, imaging, device, PTZ service), pričom využíva **Digest Authentication** podľa RFC 2617 [3], ktorý definuje HTTP autentifikáciu a tým zvyšuje zabezpečenie samotnej kamery.

Server tiež môže podporovať ako Digest Authentication z RFC 2617 tak aj **username token profile**, ktorý je špecifikovaný vo WS-Security [7]. Tieto možnosti prinášajú dynamickosť pri autentifikácii, nakoľko môže prebehnúť autentifikácia na HTTP leveli pomocou digest authentication alebo až na webovej službe pomocou WS-Security frameworku.

V praxi sa vyššie zmienené odstavce využijú následovne. V prípade, že z klienta putuje požiadavka na hociktorú z vyššie zmienených služieb a v danej požiadavke klient neposkytne v hlavičke autentifikačné údaje vo forme používateľského mena a hesla, automaticky dostáva chybovú odpoveď HTTP 401 - unauthorized.

Na záver treba spomenúť, že implementácia podľa vyššie zvolenej špecifikácie poskytuje len veľmi základnú formu zabezpečenia a samotný ONVIF upozorňuje, že v systémoch kde je bezpečnosť dôležitá - respektíve kde sa očakáva hackerský útok je stále vysoko odporúčané využívať prístup založený na TLS. Táto časť je popísaná v ONVIF Core špecifikácii pod Advanced Security Service avšak pre potreby tejto práce nie je nijak významná, je však nutné na túto skutočnosť upozorniť.

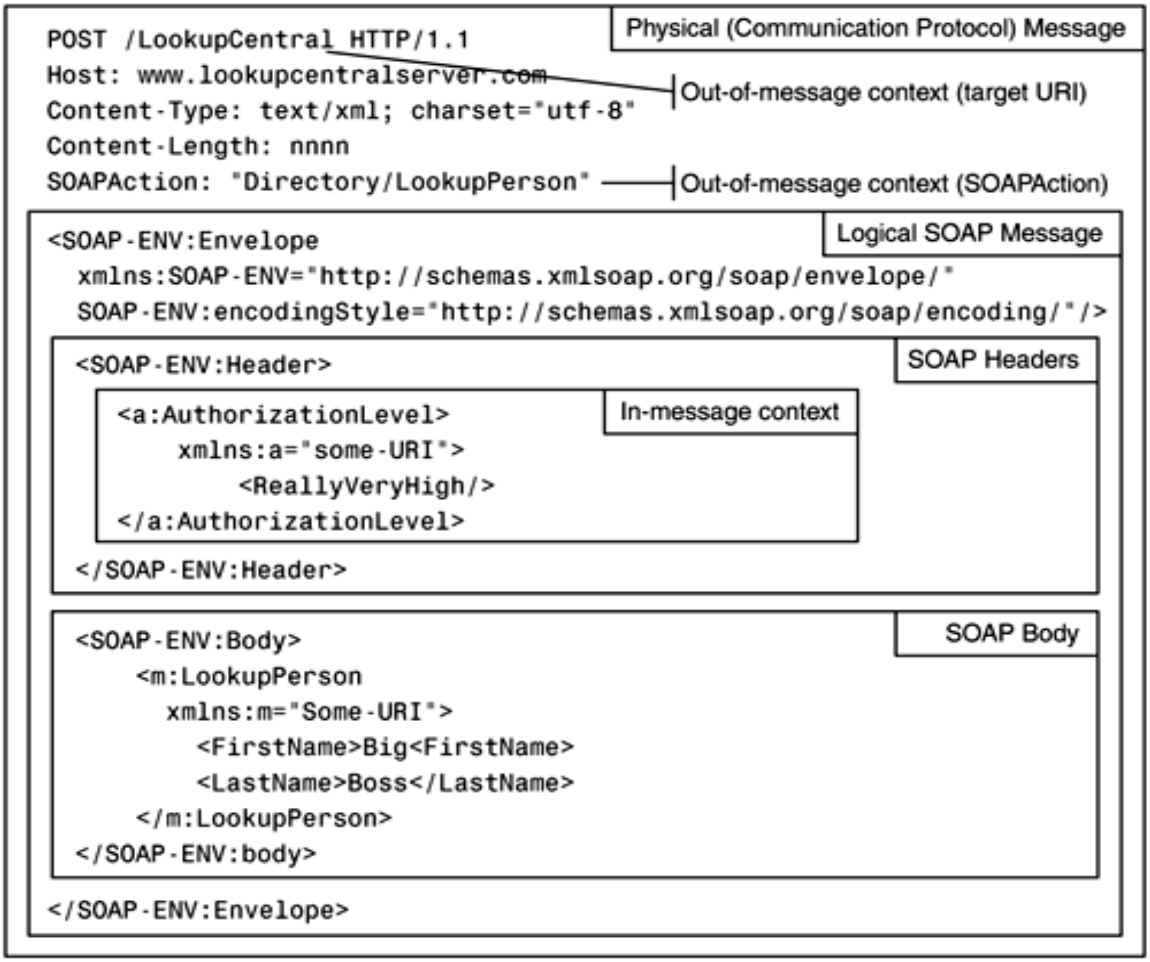


Obr. 4.1: **Diagram pre autentifikáciu zo strany servera na klienta** - diagram prebratý z ONVIF Core Specification [8]

4.1.2 ONVIF Služby

Device Management - je súčasťou **Device Service**, čiže služby, ktorá je v ONVIF štandarde považovaná za vstupný bod do zariadenia a teda aj do všetkých ostatných služieb, ktoré boli spomenuté pri autentifikácii klient - server. Jednotlivé služby majú svoje preddefinované WSDL (Web Service Definition Language - formát XML) súbory. Konkrétne Device Management je rozdelený na podkategórie - Capabilities, Network, System, Security.

SOAP a WSDL. WSDL je jazyk pre popis funkcií, ktoré sú k dispozícii z webovej služby. Zapisuje sa v značkovacom jazyku XML. V tomto prípade slúži pre opis SOAP komunikácie. SOAP, alebo celým názvom Simple Object Access Protocol je protokol slúžiaci k výmene XML správ. Komunikuje predovšetkým cez HTTP. Tvorí základnú vrstvu pre komunikáciu s webovým rozhraním. Najčastejšie využívanou architektúrou pre takúto komunikáciu je niekoľkokrát zmienený klient - server.



Obr. 4.2: **Štruktúra pre SOAP komunikáciu** - prebraté zo SOAP Protocol Binding [4]

Device discovery - je služba, ktorá ako samotný názov vypovedá, slúži pre objavenie kamery v sieti pomocou NVR. Moderné ONVIF kamery navyše túto službu využívajú na automatické vyhľadanie a pridanie kamery v NVR zariadeniach. Implementácia pre kameru má tento servis podporovať. Naopak je tomu však pri NVR, ktoré nemusí nutne podporovať device discovery, čo znamená, že nie každé NVR, predovšetkým opensourceové varianty, podporuje automatické vyhľadávanie a konfiguráciu kamery.

Funguje na princípe, kedy klient prehľadáva sieť a vyhľadáva všetky dostupné zariadenia, pričom používa protokol pre webovú službu pre vyhľadavánie zariadení - presnejšie WS-Discovery.

V prípade, že kamera implementuje možnosť jej vyhľadávania, vysiela pomocou multicastu Hello správy, alebo posiela správy o zmene jej statusu.

Kamera môže byť v stave Discoverable alebo Non-discoverable, teda vyhľadateľná a nevyhľadateľná. Druhá možnosť sa používa pre zamedzenie DoS útokov. V takomto prípade je kameru možné do klientskeho zariadenia pridať manuálne.

Media Transport - je súčasťou Media service, ktorý (respektíve ktoré) majú za úlohu spracúvať napríklad HTTP GET (prípadne POST, PUT) requesty smerujúce na Media Service a predovšetkým majú za úlohu **prenášať dáta pomocou RTSP protokolu**. Pre prenášanie dát medzi kamerou a NVR je potrebné, aby obaja implementovali Media Transport service.

Profil S pochopiteľne podporuje implementáciu aj pre **Audio Streaming** a aj keď implementácia tejto práce nepočíta s použitím audia, je prípadne možné implementáciu rozšíriť o zachytávanie zvukovej stopy z kamery.

Pan-Tilt-Zoom - jeho implementácia v Profile S je podmienená v závislosti od typu kamery. V prípade PTZ kamier je pochopiteľne očakávané, že kamera bude schopná komunikovať s NVR pomocou ONVIF (prípadne GET) requestov. V praxi to znamená, že pri stlačení tlačidla pohybu (či priblíženia, oddialenia, centrovanie, home pozície), vyšle NVR request smerom na kameru a tá zareaguje pohybom v závislosti od stlačeného tlačidla.

4.1.3 ONVIF a dostupná implementácia

Záverom kapitoly treba podotknúť, že ONVIF protokol je komplexný a niekoľko rokov presadzovaný štandard, ktorý ako bolo spomenuté skúša zjednotiť všetky kamery. Aj keď primárnou snahou tejto práce je vytvorenie kamery, ktorá bude schopná spracúvať obraz z fish-eye kamery a simulovať tým pan-tilt-zoom kameru, je tu tiež snaha o premenenie tejto kamery na kameru aspoň čiastočne spĺňajúcu ONVIF štandard.

Pre tieto účely je na oficiálnych ONVIF stránkach [6] dostupný odkaz na repozitár open sourcového projektu, ktorý má za úlohu premeniť Raspberry Pi spolu s kamerou na ONVIF kompatibilné zariadenie.

Implementácia ONVIF štandardu v tejto práci je úzko spojená práve s týmto dostupným riešením - kapitola 7.4, pričom v nej došlo k modifikácii niektorých súborov pre kompatibilitu s pan-tilt-zoom službou, ktorá je implementovaná odlišne od ONVIF štandardu. Avšak aj napriek tomuto faktu je kamera schopná fungovať s ostatnými kamerami v ľubovoľnom nahrávacom zariadení čím je zachovaný hlavný cieľ ONVIF štandardu.

Kapitola 5

Počítačové videnie

Človek vníma svet ako trojdimenzionálnu štruktúru. Ľudským okom je možné vnímať tvary, farby, jas či napríklad spočítať počet ľudí na obraze. Pre bežný život sú tieto fakty triviálnými záležitosťami avšak v prípade počítačového videnia to až taká banalita nie je. V nasledujúcej kapitole budú popísané základné princípy počítačového videnia.

5.1 Úvod do počítačového videnia

Počítačové videnie je inšpirované ľudským okom a jeho začiatky siahajú do roku 1960 a 1970. V minulosti bolo toto odvetvie považované za relatívne jednoduché, nakoľko pre človeka je spracovanie obrazu reálneho sveta triviálnou záležitosťou. Pravdou však je, že aj z biologického hľadiska je ľudské oko zložitou časťou, čo sa premieta aj do zložitosti týkajúcej sa technickej stránky počítačového videnia. Cieľom tejto disciplíny je odpočiatku čo najvernejšie napodobniť ľudské videnie. Nahrávanie videa, zhotovovanie fotiek, to všetko je úzko prepojené.

V dnešnej dobe je však počítačové videnie využívané na viac než len zachytávanie obrazu ako tomu bolo v minulosti.

Je možné pozorovať ho napríklad:

- Detekcia v obraze - rozpoznanie tváre, pohybu, klasifikácia zvierat na základe druhu a pod.
- Cestná doprava - monitorovanie dopravných uzlov, meranie rýchlosti
- Ovládanie procesov - autonómne vozidlá, výrobné linky
- Interakcia - rozpoznávanie vstupu na základe giest
- Bezpečnostné systémy - monitorovanie, alarmy
- Medicína - obrazové dáta z RTG, ultrazvuku

Samozrejme vyššie zmienené body sú len zlomkom z toho, čo všetko je schopné odvetvie počítačového videnia pokryť.

Celkovo je možné povedať, že počítačové videnie je relatívne nové a mladé odvetvie a mnoho problémov a metód k riešeniu určitých problémov sú stále v stave skúmania. Za zmienku stojí napríklad skúmanie využitia počítačového videnia v umelej inteligencii,

kde je častokrát očakávané, že systém počítačového videnia bude slúžiť ako zrak, respektíve zrakový senzor a bude danému prvku umelej inteligencie (napríklad AI robotovi) odovzdávať informácie o tom kde a v akom prostredí sa daný subjekt nachádza

5.2 Kamera a obraz

Obraz z kamery je v prípade bezpečnostnej kamery hlavným zdrojom, ktorý poskytuje dáta, ktoré je možné pomocou prvkov počítačového videnia, konkrétne knižnice OpenCV, spracovať. V tomto prípade je pomocou počítačového videnia možné obraz z kamery využiť pre simuláciu pohybu kamery, či prípadne rozšíriť implementáciu o formu detekcie pohybu alebo detekcie tváre v obraze.

5.3 Obraz

Obraz - je vlastne obrázok, ktorý obyčajne predstavuje 2D projekciu z 3D sceńy zachytenej pomocou senzoru, v tomto prípade bezpečnostnej kamery. Je to spojité zobrazenie z dvoch koordinátov i,j.

V prípade OpenCV sú obrázky reprezentované pomocou datovej štruktúry Mat, pričom v danej matici môžu byť uložené reálne vektory, komplexné vektory, matice, šedotónové obrázky, farebné obrázky či tenzory [13].

5.3.1 Vzorkovanie

Digitálne obrázky zachytené pomocou kamery sú vytvárané pomocou vzorkovania zo spojitého zobrazenia na diskrétne elementy. Toto vzorkovanie prebieha ako vzorkovanie spojitej funkcie f(i,j) do matice s M riadkami a N stĺpcami. Pri spracovaní pomocou OpenCV je možné nastaviť vzorkovanie (respektíve počet pixelov v obraze M x N) pomocou resize funkcie [2].

Obr. 5.1: **Výsledok rozdielneho vzorkovania**

Ako je zrejmé z obrázku 5.1, výsledkom funkcie sú 4 rôzne obrázky, ktoré vznikli odlišným vzorkovaním. Konkrétne ľavý horný roh: 256x192, pravý horný 128x96 - je možné pozorovať jemné rozostrenie nakoľko počet pixelov, ktoré sa nachádzajú na obrázku sa znížili o polovicu, vľavo dole ide o vzorky 64x48 a vpravo dole je možné pozorovať 32x24.

Výsledok daného pokusu ukazuje, že počet vzorkov v obrázku je úzko spojený s objektami, ktoré sa na obrázku nachádzajú. V daných 4 obrázkoch to je možné pozorovať na fakte, že ľudí v obraze vidno iba na horných dvoch snímkach. Samozrejme vyšší počet vzorkov má za následok náročnejší výpočetný čas čo má za následok predĺženie spracovania obrazu.

5.3.2 Geometrické transformácie

Nakoľko je cieľom práce spracovanie 200 stupňového obrazu zachytávaného fish-eye objektívom, implementácia sa nezaobíde bez transformácie obrazu.

Geometrické transformácie slúžia k odstráneniu skreslení v obraze, ktoré môžu byť zachytené počas zachytávania obrazu.

Geometrická transformácia je obyčajne tvorená dvoma základnými krokmi.

- Mapovanie súradníc pixelov vstupného obrazu na bod výstupného obrazu
- Jasová interpolácia - jas je vypočítaný pomocou interpolácie jasu vo viacerých bodoch svojho okolia

5.3.2.1 Rotácia

Rotácia sa radí do kategórie geometrických transformácií. Nasledujúca rotácia funguje na princípe otočenia o uhol ϕ.

$$\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} cos\phi & sin\phi & 0 \\ -sin\phi & cos\phi & 0 \end{bmatrix} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix} \tag{5.1}$$

5.4 OpenCV

OpenCV - celým názvom Open Source Vision Library je ako z názvu vyplýva open source-ová knižnica, ktorá sa zameriava špeciálne na prvky počítačového videnia. Taktiež je úzko prepojená so strojovým učením.

Knižnica OpenCV je široko využívaná po celom svete nie len na malé projekty ako je tomu aj v tomto prípade, ale jeho oficiálny repozitár[1] je používaný tými najväčšími spoločnosťami z rôznych odvetví. Napríklad Google, Yahoo!, Microsoft, Intel či dokonca Honda a Toyota.

Obsahuje okolo 2500 optimalizovaných algoritmov, ktoré môžu byť využité ako prvky počítačového videnia. Viacero z nich už bolo spomínaných - rozpoznávanie tvárí, identifikácia objektov, identifikácia zvierat, pohyb kamery, sledovanie pohybujúcich sa objektov a mnoho ďalších. Kvôli práci sú zámerne spomínané predovšetkým časti OpenCV, ktoré budú využívané pri implementácii praktickej časti práce.

[1] Oficiálny repozitár OpenCV: https://github.com/opencv/opencv

K práci bola zvolená Pythonovská varianta OpenCV - avšak samotná knižnica disponuje rozhraním okrem Pythonu aj pre C++, Javu a MATLAB. Je tiež plne kompatibilný s operačnými systémami Windows, Android, Mac OS a Linux.

Optimalizované algoritmy majú tiež vysoký podiel na tom, že je vhodné využiť OpenCV aj na zariadeniach s nie vysokou výpočetnou silou.

5.5 Vyrovnávanie obrazu

V predchádzajúcich kapitolách bol niekoľkokrát spomenutý fakt, že cieľom práce je dosiahnutie spracovania obrazu z kamery s objektívom typu rybie oko. Práve odvetvie počítačového videnia umožňuje spracovávať zachytávaný obraz v takejto forme. Spôsob spracovania základného obrazu - z kamery do Raspberry Pi (respektíve prípadne iného dostupného zariadenia) je popísaný v podkapitolách 5.2 a 5.3.

Kapitola 5.5.1 je súčasťou technológie, ktorá poskytuje práve spomínanú možnosť spracovania fish eye obrazu.

5.5.1 Kalibrácia kamery

Kalibrácia kamery je nutná v prípade, že obraz obsahuje určité skreslenie oproti skutočnému pohľadu. Takéto skreslenie je možné pozorovať častokrát na lacných pinhole kamerách, ale treba zdôrazniť, že ani kamery, ktoré sa radia do vyššej cenovej relácie nie sú dokonalé a aj tu je možné detekovať náznak skreslenia.

Za skreslenie sa dá považovať taktiež obraz zachytený fish eye kamerou, ktorý je pre ľudské oko ťažšie čitateľný a rozhodne sa nedá považovať za prirodzený.

Na obrázku 5.2 je možné pozorovať porovnanie medzi skreslením bežnej kamery a fish eye kamery. Snímok zachytený z bežnej kamery sa na prvý pohľad zdá rovný, nakoľko pre ľudské oko je zložitejšie nájsť nepatrné skreslenie. Keď sa však do obrázku vložili úsečky naznačené červenou farbou je možné bez pochyby určiť, že obraz je drobne skreslený. Na spodnom obrázku je toto skreslenie možné spozorovať ihneď nakoľko kamera zachytáva 200 stupňové rozpätie.

Toto skreslenie vzniká kvôli takzvanému radiálnemu skresleniu, kedy sa rovné čiary javia ako zahnuté. Toto skreslenie sa zväčšuje smerom od stredu. Na fish eye obraze 5.2 je tento efekt skutočne pozorovateľný - v strede snímku sú odfotené čiary takmer rovné (nábytok v strede), zatiaľ čo blízko pri kraji je toto skreslenie značné, v podobe zahnutých hrán (pozorovateľné napríklad na monitore vľavo).

Riešením takéhoto skreslenia je korekcia skreslenia. Docieliť je to možné pomocou vzorca

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{5.2}$$

Pre skreslený pixel, ktorý sa nachádza na pozícii $\mathbf{x}$ a $\mathbf{y}$ je teda jeho nová výsledná pozícia vypočítaná vzorcom (5.2) vyššie v tvare $x_{distorted}, y_{distorted}$ [11].

Ďalšou dôležitou súčasťou pri vyrovnávaní obrazu je nájsť ďalšie dôležité informácie, ktorými sú vnútorné a vonkajšie parametre kamery.

Medzi vnútorné parametre sa radia veci ako ohnisková vzdialenosť (f_x, f_y) či optické stredy - označované ako (c_x, c_y). Spomenuté parametre sú uložené v matici a tým tvoria maticu kamery. Táto matica je viazaná výhradne pre potreby jednej jedinej kamery, čo znamená, že po tom, ako sú tieto hodnoty raz vypočítané, je možné ich uložiť pre budúce

použitie. V implementácii sú tieto parametre nahrávané z JSON súboru, ktorý je vytvorený pri prvom spustení kamery na zariadení. Táto matica je reprezentovaná ako 3x3 matica nasledovne:

$$\begin{bmatrix} f(x) & 0 & c(x) \\ 0 & f(y) & c(y) \\ 0 & 0 & 1 \end{bmatrix}$$

Vonkajšie parametre súvisia s rotačnými a translačnými vektormi, ktoré transformujú koordináty 3D bodu do koordinátov systému.

Pre nájdenie všetkých vyššie zmienených parametrov je nutné vyhotoviť kamerou fotky, ktoré obsahujú predom definovaný vzor. Vo všeobecnosti platí, že čím viac fotiek s daným vzorom bude vyhotovených, tým presnejšie budú vypočítané parametre, čo sa v konečnom dôsledku pretaví do presnejšieho a kvalitne spracovanejšieho obrazu.

Najčastejšie využívaným vzorom pre kalibráciu a následnú korekciu skreslenia obrazu je šachovnicový vzor, ktorý je možné nájsť na oficiálnom repozitári OpenCV[2].

5.6 Rozpoznávanie tváre v obraze

Nakoľko implementácia počíta aj s rozpoznávaním tváre v obraze, nasledujúca kapitola načrtne využitú techniku rozpoznávania tváre.

5.6.1 Haar cascade detekcia

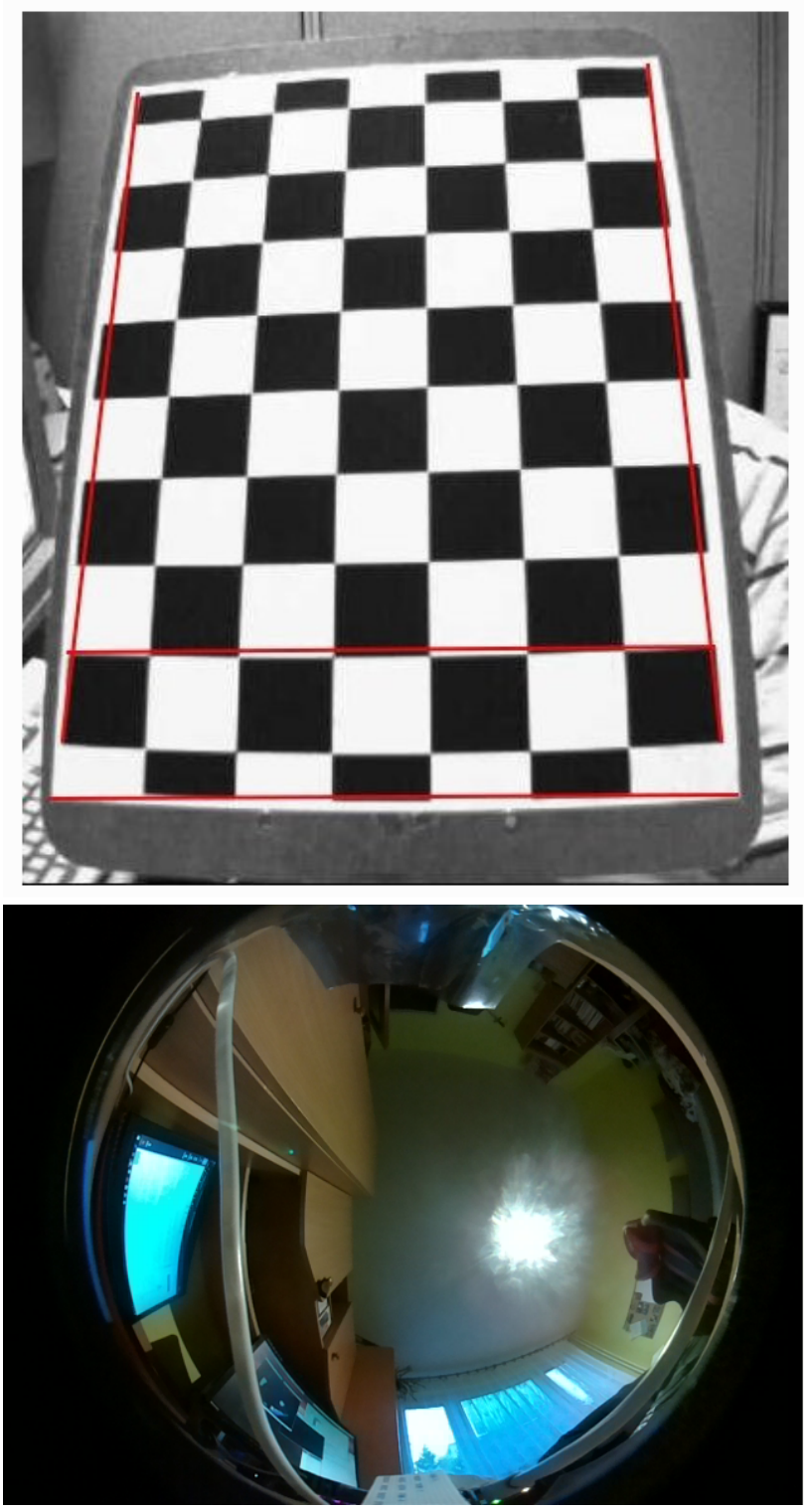
Jednou z najrozšírenejších techník detekcie tváre v obraze je **Haar cascade detekcia**.

Ako uvádza oficiálna dokumentácia OpenCV [12], detekcia nie len tvárí, ale aj objektov pri používaní kaskádovitých klasifikátorov je považovaná za veľmi efektívnu metódu. Je založená na strojovom učení, kde kaskádová funkcia je trénovaná množstvom obrazového materiálu. Následne je použitá na spomínanú detekciu v iných obrázkoch.

Počítačové videnie prepojené s OpenCV poskytuje už predom trénovanú metódu respektíve predtrénované modely, ktoré môžu byť načítané za pomoci modulov OpenCV. Najjednoduchším takýmto modelom je model pre detekciu prednej časti tváre[3].

[2] Kalibrovací vzor - šachovnicový: https://github.com/opencv/opencv/blob/master/doc/pattern.png

[3] Predtrénovaný model: https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml



Obr. 5.2: **Porovnanie skreslenia z bežnej kamery a fish eye kamery**

Kapitola 6

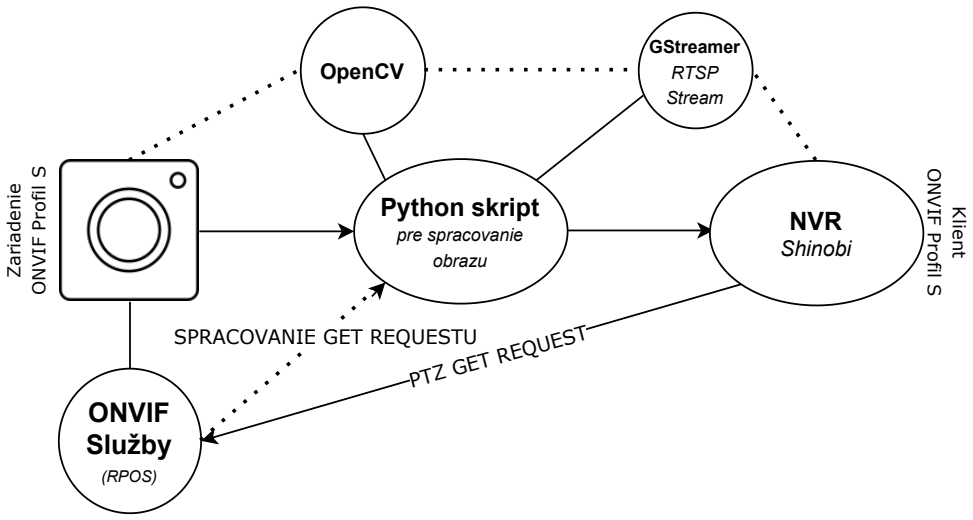
Návrh riešenia

Dôležitou súčasťou prípravy implementácie bola voľba vhodných technológií a následné prepojenie týchto technológií do fungujúceho celku. Na obrázku 6.1 je možné pozorovať projektovú pipeline, ktorá naznačuje vzťahy medzi jednotlivými komponentami výsledného bezpečnostného systému.

Obraz zachytávaný kamerou je prvotne spracovaný pomocou Pythonovského skriptu, ktorý obraz narovná, pomocou korekcie skreslenia na základe kalibrácie kamery 5.5. Skript následne zvolí špecifickú časť spracovaného obrazu, na základe udaných parametrov rotačnej matice, čím bude docielená lepšia prehľadnosť spracovaného obrazu a možnosť simulácie pan-tilt-zoomu.

Následne je spracovaný obraz vysielaný pomocou RTSP protokolu na bežiacom streame vytvorenom pomocou GStreameru. Stream je potom možné pridať a zobraziť do ľubovoľného nahrávacieho zariadenia (v tomto prípade Shinobi - kapitola 6.2.1).

Na opačnú stranu je zase možné zasielať GET požiadavky pre ovládanie kamery. Kamera, respektíve kamerový softvér musí zabezpečiť posúvanie zachytávaného obrazu na základe požiadavkov prichádzajúcich z nahrávacieho zariadenia.



Obr. 6.1: **Produktová pipeline**

6.1 Programovacie jazyky

6.1.1 Python

Pre veľkú časť implementácie bol zvolený interpretovaný programovací jazyk Python.

Pôvodná implementácia počítala s programovacím jazykom C++, kde bola pôvodná implementácia aj vyhotovená, avšak kvôli dynamickosti a predovšetkým možnej spolupráci s Node.js (kapitola 6.1.2) bol kód prepísaný do Pythonovského prostredia.

Výhodou predchádzajúcej implementácie v C++ bol fakt, že jazyk C++ je v konečnom dôsledku jednoznačne rýchlejší nakoľko C++ je jazyk kompilovaný a Python interpretovaný.

Podstatne jednoduchšie je však využiť komunikáciu medzi bežiacimi procesmi od Node.js a pythonu, než by tomu bolo pri dvojici C++ a Node.js. Bolo výhodnejšie obetovať rýchlosť na úkor prehľadnejšej a jednoduchšej implementácie, respektíve komunikácie medzi procesmi. Navyše OpenCV ponúka veľmi optimalizovaný modul (kapitola 7.2.1) Omnidir, ktorý zabezpečil, že aj napriek použitiu Pythonovskej verzii OpenCV sa využitie procesoru nepriblížilo k viac než 40 percentám a taktiež sa táto zmena nepreukázala ani v prípade premietania obrazu na klientske zariadenie, kedy obraz bežal bezproblémovo aj pre 30 snímkov za sekundu, čo je pre bezpečnostnú kameru viac než dostačujúce.

6.1.2 Node.js

Použitie Node.js bolo podmienené použitím voľne dostupnej implementácie pre Raspberry Pi - konkrétne Raspberry Pi ONVIF Server (kapitola 7.4), ktorý má za úlohu previesť kamerový modul pripojený do Raspberry Pi, respektíve celý tento systém na ONVIF kompatibilnú kameru.

Podobne ako iné dostupné riešenia, ani Raspberry Pi ONVIF Server nepočíta s možnosťou, že pan-tilt-zoom kamera bude simulovaná z objektívu typu Fish-eye, čo malo za následok, že nebolo možné celý kód prebrať a využiť ho pre účely bezpečnostnej kamery, ale bolo nutné vykonať častokrát malé, avšak významné zmeny pre účely bezpečnostnej kamery a zaistenie kompatibility s vlastným pan-tilt-zoom driverom, nakoľko ako bolo spomenuté - dostupné konfigurácie nepočítajú so simuláciou pan-tilt-zoomu.

Samotné Node.js je v aktuálnej implementácii využité pre generovanie jednotlivých služieb spomínaných v kapitole 4.1.2.

6.2 Výber nahrávacieho zariadenia

Dôležitou časťou pri navrhovaní riešenia bolo vybrať vhodné nahrávacie zariadenie. Softvérov, ktoré podporujú ONVIF štandard a zjednocujú v ňom viaceré kamery rôznych výrobcov je hneď niekoľko. Medzi najčastejšie používané a kvalitne hodnotené sa radia NVR typu iSpy Video Surveillance Software[1], EyeLine Video Management Software[2] či Blue Iris[3]. Nevýhodou však je, že dané systémy sú licencované pričom licenciu je potrebné si zakúpiť na stránkach jednotlivých výrobcov. Pri zhotovovaní tejto bezpečnostnej kamery, ktorá je vnímaná v zmysle DIY je žiadúce, aby neboli výtvárané zbytočné náklady na kúpu použiteľného softvéru.

[1] iSpy Video Surveillance Software: https://www.ispyconnect.com/default.aspx
[2] EyeLine Video Management Software: https://www.nchsoftware.com/surveillance/index.html
[3] Blue Iris: https://blueirissoftware.com/

6.2.1 Shinobi

Práve vyššie zmienený dôvod naskytá príležitosť využiť voľne dostupný open sourceový projekt Shinobi. Softvér Shinobi je vhodnou alternatívou v porovnaní s platenými NVR softvérmi z dôvodu, že Shinobi je schopné udržať aj veľké množstvo kamier bežiacich ako RTSP stream s vysokou kvalitou prenosu - H.264.

Ďalšou výhodou je, že množstvo pridaných kamier do Shinobi nie je ničím limitované (len používateľom samotným, kedy je možné maximálny počet kamier nastaviť v administrácii super používateľa). Táto výhoda je nezanedbateľná, nakoľko aj platený NVR softvér má častokrát limitovaný počet kamier udaný výrobcom daného NVR. Pre potreby tejto práce je tento faktor zanedbateľný, avšak v prípade rozširovania kamier na väčší a komplexnejší bezpečnostný systém, softvér Shinobi nebude pri danom rozšírení limitovať počet použitých kamier.

Názov	Open-Source	Počet kamier
iSpy	Nie	Neobmedzené
EyeLine	Nie	100
Blue Iris	Nie	64
Blue Iris LE	Nie	1
Shinobi	Áno	Neobmedzené

Tabuľka 6.1: Porovnanie jednotlivých NVR softvérov

V tabuľke vyššie je možné pozorovať porovnanie jednotlivých NVR softvérov, pričom varianty, ktoré sú označené ako neopen-sourceové je nutné si zakúpiť. Výnimkou je Blue Iris Light Edition (LE), ktorú je možné využívať aj ako neplatenú verziu avšak jej limit kamier je stanovený len pre jednu jedinú kameru.

6.3 Streamovanie obrazu

Popri spracovaní obrazu pomocou Pythonu, respektíve pomocou OpenCV (kapitola 6.1.1 a 5.4) je taktiež nutné daný obraz vysielať, aby bolo možné ku kamere pristúpiť z ľubovoľného zariadenia v sieti (po úspešnom prihlásení do NVR).

6.3.1 GStreamer

Vhodným prostriedkom pre vytvorenie RTSP vysielania sa ukazuje použitie GStreameru. GStreamer je framework vytvorený pre potreby vysielania médií z rôznych aplikácií. Je prenositeľný na majoritnú časť operačných systémov ako Windows, macOS, Linux či Android. Zároveň je poskytovaný pod open-sourceovou licenciou. GStreamer je hojne využívaný práve pri práci nielen s knižnicou OpenCV, ale celkovo s Pythonom nakoľko poskytuje možnosť vytvoriť streamovaciu pipeline priamo v kóde pomocou gi.repository[4], ktorá vytvorí proces streamovania automaticky bez nutnosti explicitného štartovania procesu mimo hlavného programu, čo zvyšuje kontrolu nad streamom priamo v danom programe.

GStreamer je využívaný a kompatibilný s modernými aplikáciami, jeho využitie je možné nájsť napríklad v open-sourceovom programe PulseEffects[5] alebo SoundConverter[6].

[4] gi.repository dokumentácia: https://wiki.gnome.org/Projects/PyGObject
[5] PulseEffects: https://awesomeopensource.com/project/wwmm/pulseeffects
[6] SoundConverter: https://soundconverter.org/

Alternatívnym riešením je použitie FFmpeg, ale po porovnaní týchto dvoch streamovacích prostriedkov vyšiel ako vhodnejším riešením GStreamer. Hlavné výhody sú v tabuľke nižšie naznačené zelenou farbou.

Porovnanie GStreamer a FFmpeg		
Vlastnosť	GStreamer	FFmpeg
Open-Source	Áno	Áno
Integrácia vrámci pythonu	**Áno**	Nie
Rozšírenie funkcií pomocou programu (spracovanie obrazu a až následné vysielanie)	**Áno**	Nie
Prenositeľnosť	Áno	Možné problémy s prenositeľnosťou

Tabuľka 6.2: Porovnanie GStreameru a FFmpeg z hľadiska výberu pre implementáciu

Kapitola 7

Implementácia

V nasledujúcej kapitole bude detailne popísaná implementácia týkajúca sa zhotovenia bezpečnostnej kamery od úplného počiatku až po hotový projekt.

7.1 Inštalácia OpenCV

Nutnosťou pre spracovanie obrazu z kamery pomocou Pythonu je doinštalovanie knižnice OpenCV (kapitola 5.4), nakoľko táto knižnica nie je základnou súčasťou Pythonu, predovšetkým kvôli jej veľkosti.

Pretože v systéme pre správu balíkov pythonu (pip) sa nachádza iba neoficiálna predkompilovaná verzia OpenCV, je nutné nainštalovať OpenCV priamo z oficiálneho repozitára.

Pre potreby práce je však pri inštalácii OpenCV nutné zdeliť inštalačnému skriptu, že pri inštalácii je potrebné počítať aj s predinštalovaným programom GStreamer (kapitola 6.3.1).

Potrebné prekrekvizity k inštalácii OpenCV s GStreamerom

- Python 3.7
- OpenCV 4.1.0 (a vyššie)
- GStreamer 1.8.3
- CMake 3.5.0 (a vyššie)

7.1.1 Inštalácia

Ako prvé je nutné zabezpečiť závislosti, ktoré sú nutné pre bezchybné spustenie samotného OpenCV.

V prípade, že závislosti z obrázka 7.1 sú už nainštalované (respektíve prebehlo ich doinštalovanie), je nutné stiahnuť samotné oficiálne zdrojové kódy OpenCV. Najnovšiu verziu je možné zistiť z oficiálnej webovej stránky OpenCV [9], prípadne ju je možné dohľadať aj na oficiálnom repozitári.

Pri vytváraní konfiguračných súborov, slúžiacich k nainštalovaniu OpenCV, pomocou CMAKE je nutné k bežne dostupným príkazom, ktoré je možné nájsť vo všetkých bežne dostupných návodoch k sprevádzkovaniu OpenCV knižnice doložiť po nainštalovaní GStreameru argument `-D WITH_GSTREAMER=ON`.

```
sudo apt-get update && sudo apt-get upgrade -y
sudo apt-get install build-essential cmake unzip pkg-config -y
sudo apt-get install libjpeg-dev libpng-dev libtiff-dev -y
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev -y
sudo apt-get install libxvidcore-dev libx264-dev -y
sudo apt-get install libgtk-3-dev -y
sudo apt-get install libcanberra-gtk* -y
sudo apt-get install libatlas-base-dev gfortran -y
sudo apt-get install python3-dev -y
```

Obr. 7.1: **Predinštalované závislosti**

Finálny príkaz pre vytvorenie konfiguračných súborov pred nainštalovaním samotného OpenCV bude vyzerať následovne:

cmake -D CMAKE_BUILD_TYPE=RELEASE
-D INSTALL_PYTHON_EXAMPLES=ON
-D INSTALL_C_EXAMPLES=OFF
-D PYTHON_EXECUTABLE=$(which python3)
-D BUILD_opencv_python2=OFF
-D CMAKE_INSTALL_PREFIX=$(python3 -c “import sys; print(sys.prefix)”)
-D PYTHON3_EXECUTABLE=$(which python3)
-D PYTHON3_INCLUDE_DIR=$(python3 -c “from distutils.sysconfig import get_python_inc; print(get_python_inc())”)
-D PYTHON3_PACKAGES_PATH=$(python3 -c “from distutils.sysconfig import get_python_lib; print(get_python_lib())”)
-D WITH_GSTREAMER=ON
-D BUILD_EXAMPLES=ON ..

Po prebehnutí cmake príkazu a vytvorení konfiguračných súborov je vhodné skontrolovať výstup daného príkazu kedy v sekcii **Video I/O** (nachádzajúcej sa v spodnej časti výstupu) musí byť pozorovaný nasledujúci výstup:

Video I/O	
DC1394	YES
FFMPEG	YES
avcodec	YES
avformat	YES
avutil	YES
swscale	YES
avresample	NO
avcodec	YES
GStreamer	YES
v4l/v4l2	YES

Tabuľka 7.1: Výstup cmake príkazu pre kontrolu inštalácie s GStreamerom

7.2 Spracovanie obrazu

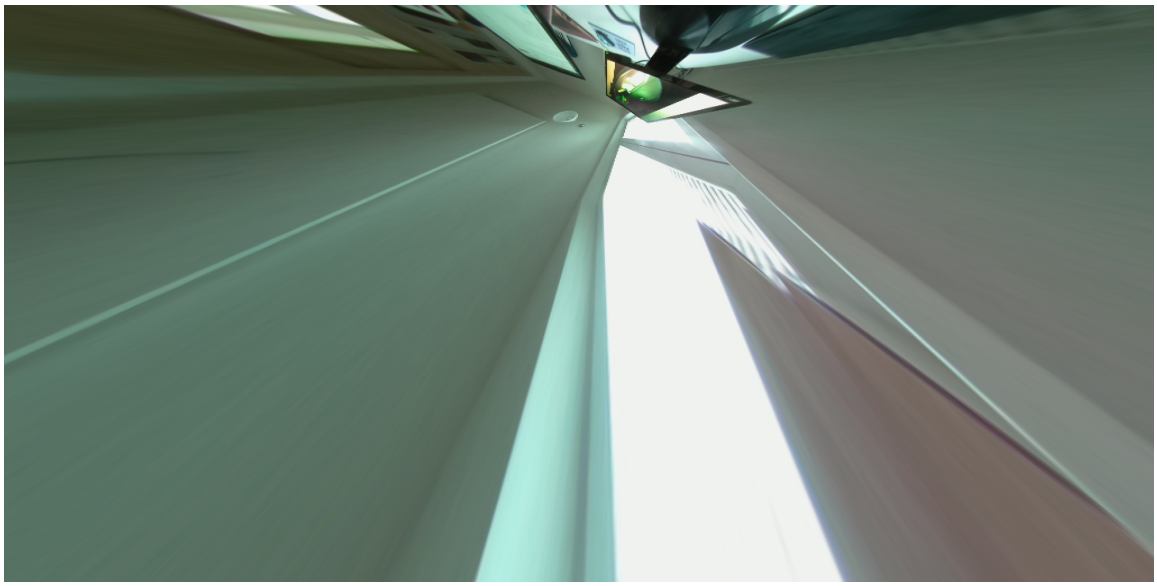
V teoretickej časti, konkrétne v kapitolách 5.3 a 5.5 je možné vidieť techniky využívané k spracovaniu obrazu. Pre prevedenie daných znalostí do praxe, ktoré budú viesť k úspešnému výsledku je nutné zvážiť viacero faktorov.

7.2.1 Výber vhodného modulu z OpenCV

Nakoľko knižnica OpenCV ponúka viacero modulov pre spracovanie fish-eye obrazu, bolo nutné po počiatočnom rešerši vybrať ten najvhodnejší.

7.2.1.1 CV::FISHEYE modul

Po preskúmaní dostupných technológií sa pre implementáciu najlepšie naskytala možnosť využitia Fisheye modulu z knižnice OpenCV (kapitola 5.4) [14]. Po tom, čo prebehla implementácia pomocou využitia `cv::fisheye::calibrate`, `cv::fisheye::undistortImage` a ďalších častí fisheye modulu, bol výsledkom obraz bez pôvodného skreslenia, avšak výsledná korekcia skreslenia nepriniesla o nič lepší prehľad než tomu bolo pred danou korekciou. Na obrázku 7.2 je možné pozorovať výsledok tohto implementačného pokusu.



Obr. 7.2: **Neprehľadná korekcia skreslenia** - zo zachyteného obrazu

Po preskúmaní webových stránok OpenCV a príslušných repozitárov, je možné naraziť na problém[1] týkajúci sa práve nedostatočnej podpory kamier typu rybie oko s veľkosťou zorného pola viac než 180 stupňov.

Problém, ktorý vznikol použitím modulu `cv::fisheye` je možné eliminovať nepoužitím danej knižnice, alebo použitím kamerového modulu, ktorý zachytáva zorné pole o veľkosti do 180 stupňov.

[1] Fisheye modul: https://github.com/opencv/opencv/issues/15923

7.2.2 CV::OMNIDIR modul

Riešenie problému z podkapitoly 7.2.1.1 - nepoužitie daného modulu, naskytlo možnosť využitia modulu `cv::omnidir`. Ten ponúka predimplementované funkcie podobné tým z `cv::fisheye`, avšak `cv::omnidir` počíta aj s možným použitím katadioptrických kamier s veľkosťou zorného pola viac než 180 stupňov.

7.2.3 Kalibrácia kamery

Úspešnou časťou vyššie zmienenej implementácie z podkapitoly 7.2.1.1 je však korektné vytvorenie kódu pre kalibráciu. Teoretický spôsob kalibrácie je popísaný v kapitole 5.5.1.

Prvým krokom pri korekcii skreslenia je skalibrovanie kamery a získanie matice kamery, respektíve vnútorných parametrov.

Je potrebné vyhotoviť niekoľko fotografií pomocou kamery - OpenCV odporúča minimálne 10 snímkov [10], pričom na daných snímkoch musí byť jasne viditeľný kalibračný vzor. Pre implementáciu bol zvolený šachovnicový vzor. Vo vseobecnosti platí, že čím viac vyhotovených snímkov má kalibračná funkcia k dispozícii, tým presnejšie budú určené jednotlivé parametre. V `./img` zložke je možné nájsť 18 vyhotovených kalibračných snímkov pre kamerový modul použitý v tejto implementácii (kapitola 3.2.1).

Ďalším krokom v kalibrácii je následná extrakcia rohov jednotlivých políčok z kalibračného vzoru. Za pomoci funkcie `cv::findChessboardCorners`, ktorá všetky zachytené snímky prejde a následne rozhodne, či vstupný obraz obsahuje šachovnicový vzor a ak áno, lokalizuje jednotlivé rohy.

Výsledkom funkcie `cv::findChessboardCorners` sú súradnice pozícií jednotlivých rohov kalibračného vzoru uložené v matici.

Mimo získaných bodov v obraze je nutnosť tiež získať trojdimenzionálne body, predstavujúce body reálneho sveta. Treba teda poznať hodnoty (X,Y,Z). Keďže vzor alebo kamera (v tomto prípade kalibračný vzor) boli na statickom mieste pri vytváraní kalibračných snímkov, hodnotu Z je možné vždy nastaviť na 0. Hodnoty (X,Y) je zas možné nastaviť na hodnoty (0,0), (1,0), (2,0), ..., (5,8), ktoré označujú lokáciu bodov. Výsledkom prípravy týchto 3D bodov je nakoniec numpy pole o veľkosti mierky šachovnice. Objektové body, ako sú tieto 3D body často označované je tiež možné dopočítať svojpomocne v prípade, že je známa fyzická veľkosť použitého vzoru.

Možné definovanie 3D bodov použité v implementácii:

```
objp = np.zeros((1, 6*9, 3), np.float32)
objp[0,:,:2] = np.mgrid[0:6, 0:9].T.reshape(-1, 2)
```

Ak nastala situácia, že sú k dispozícii objektové body spolu s maticou, ktorá bola výsledkom funkcie `cv::findChessboardCorners`, je možné zahájiť samotnú kalibráciu kamery.

Nakoľko výsledky kalibrácie sú zmienené parametre z kapitoly 5.5.1 je nutné pred spustením samotnej kalibrácie preddefinovať výsledné matice.

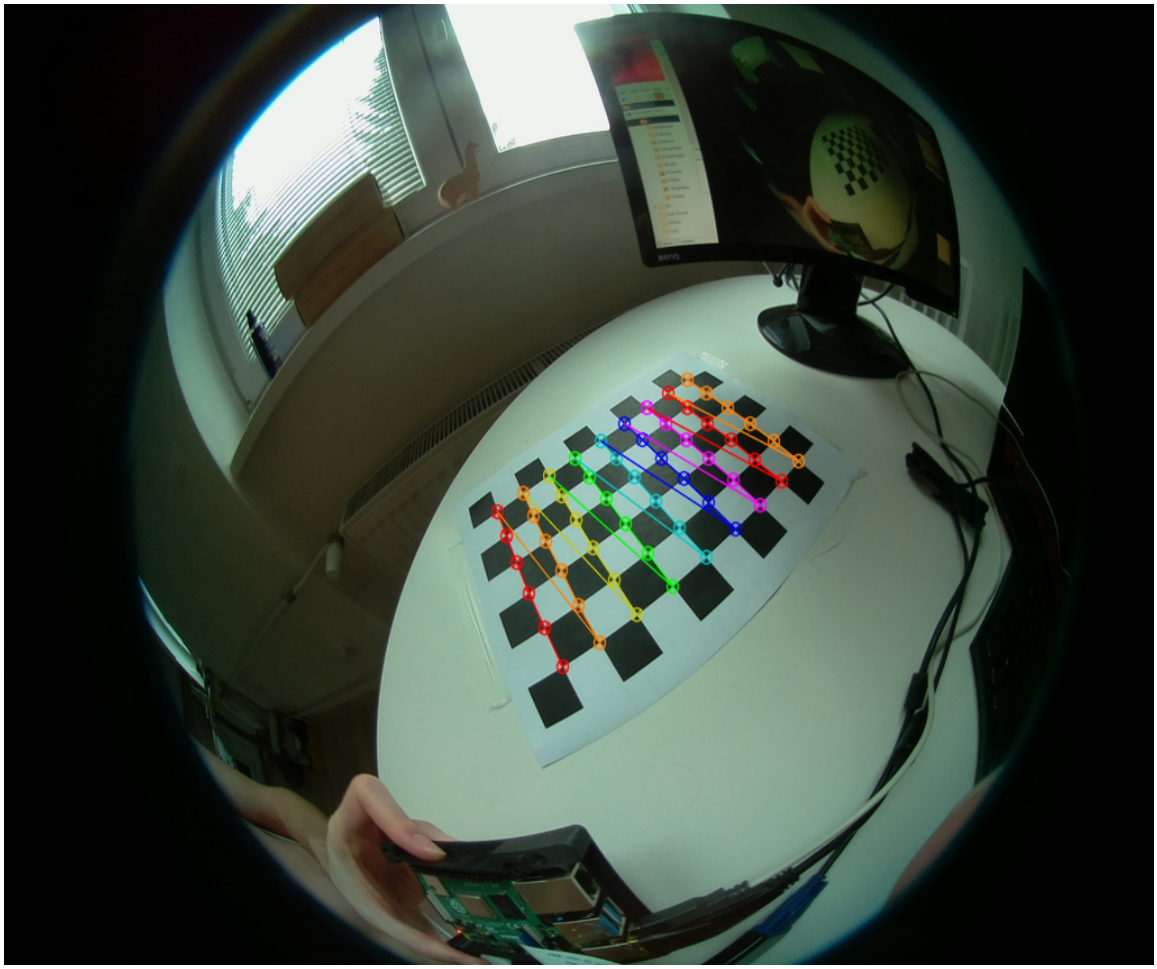
Po preddefinovaní týchto výstupných matíc a vektorov je možné využiť kalibračnú funkciu z modulu `cv::fisheye`, respektíve `cv::omnidir` (kapitola 7.2.2), ktorá zabezpečí vyplnenie vonkajších a vnútorných parametrov kamery, ktoré môžu byť následne využité pri korekcii skreslenia obrazu z kamery.

Správna kalibrácia môže byť overená pomocou funkcie `cv::drawChessboardCorners`, ktorá v prípade, že bol hľadaný vzor na obrázku nájdený, vykreslí na jednotlivé rohy vzoru farebné kolieska, ktoré sú spolu prepojené. Ak vzor nájdený nebol, alebo nebol nájdený

úplne, funkcia vkreslí na niektoré detekované časti červené kruhy, ktoré navzájom nie sú pospájané. Na obrázku 7.3 je možné pozorovať výsledok správnej kalibrácie kamery.

Implementácia počíta s možným overením správnosti kalibrácie, kedy pomocou súboru `calibrationTest.py` sú užívateľovi premietané jednotlivé kalibračné snímky, na ktorých je zobrazený výsledok kalibrácie (na premietanom obraze je možné vidieť spojené farebné kolieska v rohoch kalibračného vzoru).

Samotná kalibrácia prebieha iba pri prvom spustení, nakoľko po prvotnom skalibrovaní je vytvorený JSON súbor `cameraParams.json` s príslušnými parametrami. V čase kedy skript prichádza ku časti pre korekciu skreslenia skontroluje, či je možné dané parametre načítať z dostupného súboru. Parametre sú ukladané predovšetkým z dôvodu šetrenia času spúšťania napríklad po zapnutí prípadne reštartovaní kamery.



Obr. 7.3: **Správne prebehnutá kalibrácia** - jednotlivé rohy sú farebne prepojené

7.2.4 Korekcia skreslenia

Po prebehnutom a overenom výsledku kalibrácie, respektíve načítaniu potrebných parametrov, môže naimplementovaný kamerový softvér pristúpiť k odstráneniu skreslenia, ktoré vzniklo použitím fisheye kamery.

Obraz z kamerového modulu je nahrávaný a ukladaný pomocou funkcie `cv::VideoCapture`. Práve funkcia `cv2::VideoCapture::read` je zodpovedná za uloženie obrazu. Po zachytení jedného snímku je obraz zaslaný do funkcie `undistortImg(K, D, xi, img)`.

Obraz je následne vo funkcii spracovaný pomocou funkcií z omnidir modulu (kapitola 7.2.2).

Ako prvé dôjde k úprave matice kamery, kedy je vytvorená jej kópia (až tá je následne upravovaná). Koeficienty na pozícii (0,0) respektíve (1,1), ktoré v matici kamery zmienenej v kapitole 5.5.1 zastávajú koeficienty definujúce ohniskovú vzdialenosť, je nutné predeliť. K správnemu pomeru predeleniu týchto koeficientov bolo možné dôjsť pomocou testovania, kedy boli dané koeficienty predelenované v rôznych mierkach. Ako najschodnejšie sa ukázalo predelenie 2 pre f(x) (0,0) a 3 pre f(y) (1,1). V prípade, že by nedošlo k predeleniu týchto parametrov by výsledný obraz kamery pôsobil ako príliš priblížený a ťažko čitateľný - práve preto bolo nutné nájsť správne konštanty na predelenie zmienených parametrov.

Toľkokrát spomínaný pan-tilt-zoom servis je súčasťou práve funkcie `undistortImg(K, D, xi, img)`, kde je pre dané potreby vytvorená rotačná matica s niekoľkými parametrami, určenými k pohybu do všetkých smerov, vrátane približovania, oddiaľovania a návratu do domovskej pozície.

Ako je uvedené v návode na oficiálnych stránkach dokumentácie OpenCV [10], je potrebné si uvedomiť, že rotačná matica o veľkosti 3x3 je rozdelená podľa 3D koordinátov reálneho sveta - (X,Y,Z).

Na základe tohto faktu som previedol viacero pokusov k vytvoreniu správnej rotačnej matice, ktorá bola vhodná k použitiu vo výslednej implementácii. Bolo potrebné previesť viacero pokusov s jednotlivými parametrami a postupne pozorovať výsledky pri zmenách týchto parametrov.

Výsledná rotačná matica definovaná pre pan-tilt-zoom servis vyzerá následovne:

$$\begin{bmatrix} 0.5 & 0.0 & RL \\ 0.0 & 0.6 & UD \\ 0.0 & 0.0 & ZOOM \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Parametre **RL** (Rotate Right / Rotate Left) - pohyb vpravo / vľavo, **UD** (Rotate Up / Rotate Down) - pohyb hore / dole a **ZOOM** - priblíženie / oddialenie sú preddefinované na začiatku skriptu na východziu hodnotu 0.0.

Simulovaný pan-tilt-zoom pohyb kamery umožňujú zmeny týchto parametrov v rotačnej matici za behu programovej časti, ktorá zabezpečuje spracovanie obrazu.

Zmeny parametrov sú vždy vykonávané pred volaním funkcie `undistortImg` - táto funkcia je volaná pri zachytení každého jedného snímku, nakoľko na každom zachytenom fisheye obraze je nutné previesť korekciu skreslenia. Parametre sú menené podľa užívateľských požiadaviek tvorených v Sieťovom nahrávacom zariadení (kapitola 7.5 a 6.2.1). Požiadavky sú vytvárané vo forme GET požiadaviek, ktoré smerujú zo Shinobi (alebo iného NVR) na kameru, respektíve na ONVIF služby pôsobiacej na kamere. GET požiadavka konkrétne smeruje na ONVIF media servis kde je spracovaná. V prípade, že je požiadavka smerovaná

na jednu z možných URL pre pohyb kamery (viac o nastavení celého pan-tilt-zoom servisu v podkapitole 7.5), odošle Media servis dáta do Pythonovského skriptu pomocou stdin pipeline so smerom, ktorým sa má kamera pohnúť. Pre príklad - v prípade, že kamera má byť posunutá napríklad do pravej strany, po stlačení tlačidla pohybu v NVR, vyšle NVR GET požiadavku s príslušnou URL na ONVIF Media Servis kde je následne predaná skriptu, ktorý upraví parameter RL pre pohyb do prava (RL += 0.1). V ďalšom volaní funkcie pre korekciu skreslenia je už finálny obraz počítaný s novou rotačnou maticou.

Po pripravení rotačnej matice dochádza k samotnej korekcii skreslenia pomocou `cv::omnidir::undistortImage`. Do tejto funkcie sú vložené všetky potrebné zistené parametre respektíve matice. Funkcia `cv::omnidir::undistortImage` je v skutočnosti zaobalenie volania funkcií `cv::omnidir::initUndistortRectifyMap` a `cv::omnidir::remap`. V tejto časti je vypočítané mapovanie, ktoré povedie ku korekcii skreslenia. Funkcia `cv::omnidir::initUndistortRectifyMap` vytvorí mapy pre inverzný mapovací algoritmus, ktorý je použitý na premapovanie skresleného obrazu. To znamená, že pre každý pixel (u,v) v upravenom obraze, funkcia vypočíta korešpondujúce koordináty nachádzajúce sa v originálnom obraze (kapitola 5.5).

Po vyhotovení mapovania je pomocou mapovacej funkcie `cv::omnidir::remap` obraz transformovaný tak, aby bolo odstránené skreslenie nasledovne:

$$dst(x, y) = src(map_x(x, y), map_y(x, y)) \tag{7.1}$$

Po premapovaní je obraz bez zkreslenia vrátený do cyklu, ktorý vpisuje pomocou modulu Gst (7.3) spracovaný obraz do predpripraveného RTSP vysielania.

7.3 GStreamer

Ako bolo načrtnuté v kapitole 6.3.1, GStreamer slúži ako framework pre vysielanie médií. V implementácii je využívaný práve pre účely vytvorenia a udržiavania RTSP vysielania, pričom je doňho možné vpisovať dáta z Pythonovského skriptu, v ktorom je GStreamer spúšťaný.

Po prebehnutí kalibrácie (alebo načítania potrebných parametrov) je inicializovaný proces pre cyklus, ktorý bude zabezpečovať neustále vysielanie a posielanie dát počas behu programu.

Pre tieto účely je využívaný `GObject`, ktorý zároveň podporuje signalizáciu, ktorá je potrebná pre vyvolanie signálu pre vpisovanie spracovaných obrazových dát do prebiehajúceho RTSP vysielania.

Ďalej sú pre potreby vysielania obrazu využité moduly `Gst` a `GstRtspServer`, pričom Gst slúži k inicializácii GStreameru a k alokácii miesta pre spracované obrazové dáta.

Významnou časťou v tejto časti implementácie je vytvorená trieda `GstServer`, ktorá dedí z `GstRtspServer::RTSPServer`. Táto trieda pripravuje samotné vysielanie pomocou `RTSPServeru`, ktorý pripojí na daný bod vytvorenú a definovanú Factory.

Trieda Factory má za úlohu vysielanie spracovaného obrazu a vloženie spúšťacej pipeline pomocou funkcie `Factory::do_create_element` pri spustení GStreameru.

Pipeline GStreameru je nevyhnutnou súčasťou jeho správneho spustenia. Keďže môže byť GStreamer spustený aj nezávisle z príkazovej riadky, syntax spustenia je následovná: `gst-launch-1.0 <pipeline>`. Obdobným spôsobom je GStreamer spúšťaný aj pomocou

Obr. 7.4: **Vpisované obrazové dáta do RTSP vysielania.** Na prvom obrázku vľavo hore je pôvodný fish-eye obraz, ten je spracovaný a výsledkom je obraz bez skreslenia, ktorý je zobrazovaný vo vysielaní. Pravý horný obrázok je základná poloha kamery a spodný obrázok je výsledok simulovaného pohybu pan-tilt-zoomu do pravého horného rohu pôvodného obrazu.

použitých modulov GStreameru a tak je nutné túto pipeline definovať v spomínanej funkcii do_create_element. V prvom argumente pipeline musí byť definovaný zdroj, odkiaľ má GStreamer čerpať obrazové dáta pre vysielanie. Keďže obraz z kamery nie je vysielaný bez predošlého spracovania, ale musí byť najprv v skripte spracovaný, nie je ako zdroj uvedený `rpicamsrc` (kamerový modul v RPi), ale **`appsrc`**, čo zabezpečí, že GStreamer bude obraz očakávať ako výstup funkcie `undistortImg`. Pre úspešné prepojenie GStreameru a výstupu OpenCV pomocou argumentu appsrc je dôležité uviesť ako ďalší parameter `name=source`, aby bolo možné vo funkcii `do_configure` definovať odkiaľ má appsrc očakávať obrazové dáta. Ďalšou nutnosťou je špecifikácia výšky, šírky a snímkov za sekundu, pričom tieto argumenty by mali byť zhodné s údajmi využívanými pri spracovaní obrazu. **Finálna pipeline použitá v implementácii vyzerá následovne:**

```
'appsrc name=source is-live=true block=true format=GST_FORMAT_TIME'
'caps=video/x-raw,format=BGR,width=640,height=480,framerate=30'
'! videoconvert ! video/x-raw,format=I420'
'! x264enc speed-preset=ultrafast tune=zerolatency'
'! rtph264pay config-interval=1 name=pay0 pt=96'
```

7.4 Raspberry Pi ONVIF Server

Ako bolo spomenuté v kapitole 4, pre vytvorenie ONVIF kompatibilnej kamery bol použitý open-sourceový projekt Raspberry Pi ONVIF Server. Táto implementácia v sebe obsahuje podporu servisov - device, media, imaging, discovery a pan-tilt-zoom. Keďže dostupná implementácia nepočíta so simulovanou pan-tilt-zoom kamerou, bol tento servis odstránený a nahradený vlastnou implementáciou.

Vyššie zmienené ONVIF služby sú spustené ako prvé pred samotným skriptom na spracovanie obrazu. Jednotlivé služby bežia na IP adrese pridelenej danej kamere v sieti na porte 8081 a príslušnej url.

Device servis zabezpečí bezproblémové pridanie kamery do nahrávacieho zariadenia v prípade, že boli správne vyplnené prihlasovacie údaje zapisované v tvare `rtsp://<meno>:<heslo>@<RaspberryPiIP>:8554/onvif1`.

Media servis je najpodstatnejšou časťou pre implementovaný pan-tilt-zoom softvér, nakoľko daný servis prijíma a následne spracováva všetky požiadavky prichádzajúce na východziu URL `<RaspberryPiIP>:8081`. V prípade, že z nahrávacieho zariadenia príde príslušná požiadavka, media servis ju vyhodnotí ako požiadavku týkajúcu sa pohybu kamery a pomocou pipeline smerujúcej na štandardný vstup bežiaceho pythonovského skriptu je následne prevedený pohyb kamery pomocou zmeny spomínanej rotačnej matice.

7.5 Sieťové nahrávacie zariadenia

Ako bolo viackrát spomenuté, kamera je vo veľkej miere implementovaná pomocou ONVIF štandardu čo znamená, že je možné jej obsah zobraziť v takmer ľubovoľnom nahrávacom zariadení.

Otestované nahrávacie zariadenia pre kameru boli ZoneMinder a Shinobi, kde kamera fungovala bezproblémovo. Implicitne však implementácia zahrňuje použitie Shinobi NVR a to tak, že samotná kamera neobsahuje iba kamerový softvér, ktorý je nutné pridať do určitého NVR, ale Shinobi je priamo predinštalovanou súčasťou v Raspberry Pi, pričom obsahuje všetky prednastavené nutnosti ako je meno, heslo, port alebo pan-tilt-zoom nastavenie.

Predinštalovanie Shinobi hrá významnú úlohu práve v ohľade, že kameru po zapnutí nie je nutné nijak nastavovať, iba vyhľadať v sieti kameru pre zistenie IP adresy (napríklad program nmap) tejto kamery a následne na ľubovoľnom zariadení v sieti pristúpiť do Shinobi cez prehliadač. Prístupové meno a heslo do Shinobi je zvolené užívateľom v super administrácii. Po prihlásení je možné kameru ihneď ovládať pomocou dostupných tlačidiel ovládania, respektíve sledovať obraz z kamery.

Kameru je taktiež pochopiteľne možné previesť do ľubovoľného iného NVR. Pri pridávaní kamery je nutné vyplniť vysielaciu URL (častokrát možné nájsť pod pojmom Stream URL) v tvare `rtsp://<meno>:<heslo>@<RaspberryPiIP>:8554/onvif1`. Je dôležité vyplniť správne údaje, v opačnom prípade by ONVIF mal zamedziť prístup ku kamere.

7.5.1 Nastavenie pan-tilt-zoom

Ďalším potrebným nastavením pri vkladaní kamery do iného NVR je mimo spustenia samotného vysielania aj nastavenie pan-tilt-zoom servisu. V časti pan-tilt-zoom v nastaveniach kamery priamo v NVR, je nutné zvoliť ovládanie pomocou GET požiadaviek a do jednotlivých polí pre smery vyplniť URL podľa tabuľky uvedenej nižšie.

Pan-Tilt-Zoom nastavenie	
Pohyb vpravo	/right
Pohyb vľavo	/left
Pohyb hore	/up
Pohyb dole	/down
Priblíženie	/zoom-in
Oddialenie	/zoom-out

Aj keď sa nastavovanie podľa vyššie uvedenej tabuľky javí ako jednoduché, je to neopomenuteľná súčasť funkčnosti celého pan-tilt-zoom softvéru.

Detailnejšie nastavenie celej kamery sa nachádza v prílohe A.

7.6 Rozpoznávanie tváre v obraze

Ako bolo zmienené v kapitole 5.6, pre implementáciu bola na základe preskúmaných technológií zvolená Haar cascade detekcia. Súbor `haarcascade_frontalface_default.xml`, obsahuje výsledok predtrénovaného klasifikátoru.

V prípade, že je v súbore `launch.sh` pridaný prepínač `-face` alebo `-f`, je kamera spustená s detekciou tváre. Implicitne je táto funkcia vypnutá.

Klasifikátor je najprv načítaný pomocou funkcie `cv2::CascadeClassifier`. Následne je využitý po zavolaní funkcie `detectFace()` z `undistortImg()` funkcie. Po aplikovaní rotačnej matice na obraz bez skreslenia, je spracovaný obraz zaslaný do funkcie `detectFace()`, ktorá pomocou modulu kaskádového klasifikátora, konkrétne `detectMultiScale`, prehľadá obraz a v prípade, že bola na obraze nájdená tvár, vráti list so súradnicami, na ktorých sa v obraze nachádza. Tieto súradnice je následne možné využiť pre vykreslenie ohraničenia tváre v obraze.

Pred vložením obrazu do funkcie `detectFace()` je nutné obraz previesť na šedotónový, nakoľko spracovanie takéhoto obrazu je z výpočetného hľadiska podstatne jednoduchšie, než by tomu bolo v prípade obrazu farebného.

Kapitola 8

Testovanie

Nasledujúca kapitola poskytuje detailný popis o prebehnutom testovaní finálneho výrobku. Celkový dizajn výslednej kamery je možné vidieť na obrázku 8.1.

Obr. 8.1: **Výsledný dizajn kamery**

Testovanie kamery bolo rozdelené do dvoch separátnych častí. Prvá časť, popísaná v kapitole 8.1, je určená k popisu testovania základných funkcií kamery ako je funkčnosť konceptu plug-and-play, pan-tilt-zoom testovanie, či možnosť pripojiť kameru do nahrávacieho zariadenia ručne a nastaviť pan-tilt-zoom servis podľa návodu v prílohe.

Druhá kapitola testovania slúži k otestovaniu domnienky, že fish-eye obraz je pre človeka ťažšie čitateľný a výsledný spracovaný obraz, ktorý pôsobí ako obraz zachytený bežnou kamerou uľahčí toto čítanie a bude možné v takomto obraze ľahšie detekovať pohyb či jednotlivé objekty.

8.1 Základné testovanie

Ako bolo v úvode tejto kapitoly spomenuté, základné testovanie slúži k nasledujúcemu overeniu funkčnosti:

- Plug-And-Play koncept
- Detekcia tváre v obraze
- Nastavenie pan-tilt-zoom servisu
- Pan-Tilt-Zoom funkčnosť
- Celková funkčnosť kamery

8.1.1 Plug-and-Play

Ako prvé bolo zvolené testovanie plug-and-play konceptu. Dodržanie tohto bodu bolo inšpirované existujúcimi riešeniami, nakoľko som túto vlastnosť pri daných riešeniach hodnotil pozitívne hlavne z dôvodu, že len málo vecí musí byť vykonaných na užívateľskej strane.

Na prvý pohľad sa javilo, že takéto testovanie je nadbytočné, avšak rýchlo sa ukázalo, že test plug-and-play konceptu mal význam, nakoľko výsledok prvotného testovania ukázal neúspech podpory takéhoto konceptu, pretože prvotná konfigurácia, do ktorej som navyše nezapočítal nainštalovanie operačného systému a pripojenie kamerového modulu, vyžaduje okrem naklonovania repozitára aj predinštalovanie OpenCV knižnice, čo v prípade kombinácie s GStreamerom môže aj napriek relatívne ľahkej avšak zdĺhavej inštalácii pôsobiť zmätočne.

Na základe výsledkov tohto testovania bol vytvorený a následne znovu pretestovaný shellovský skript `configure.sh`, ktorý sa nachádza v repozitári respektíve na pamäťovom médiu. Skript po spustení nainštaluje potrebné závislosti a pripraví kameru. Navyše zabezpečí aj implicitnú inštaláciu NVR Shinobi, do ktorého je možné pristúpiť aj z iného zariadenia v sieti.

Finálne testovanie danej časti ukázalo, že kamera je lepšie pripravená pre plug-and-play koncept, avšak treba povedať, že zdokonalenie je možné napríklad formou predpripraveného obrazu, ktorý by stačilo vložiť na SD kartu a kameru doslova len pripojiť k zdroju.

8.1.2 Detekcia tváre

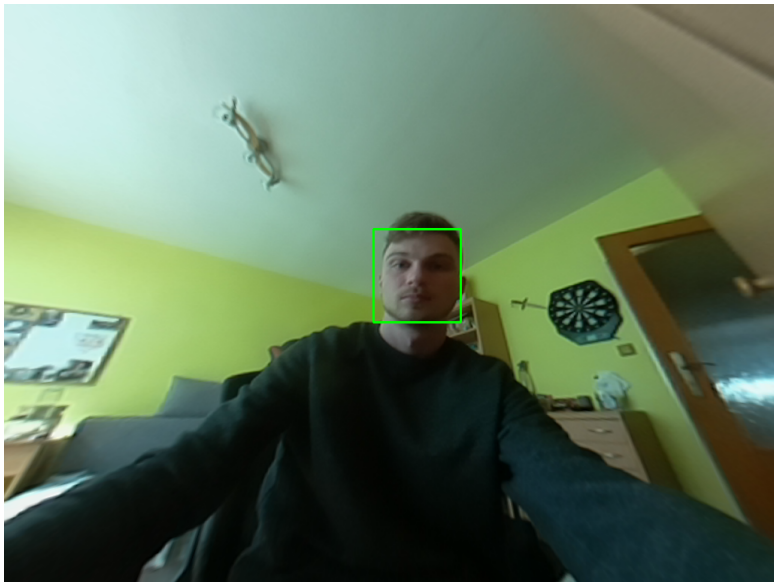
Jednou z inteligentných funkcií kamery je mimo pan-tilt-zoom softvéru aj možná detekcia tváre. Ako bolo spomenuté v kapitolách 5.6 a 7.6, bola využitá haar cascade detekcia.

Už prvotné testovania detekcie tváre, ktoré prebehli lokálne - to znamená, že výsledný obraz ešte nebol zasielaný pomocou GStreameru na RTSP vysielanie ukázali, že detekcia tváre vyžaduje náročný výpočetný proces, pretože kamera, ktorá pôvodne fungovala bezproblémovo pri 30 snímkoch za sekundu a bolo ju možné ovládať pan-tilt-zoom softvérom, začala mať značné pády snímkov za sekundu. Tieto pády sa prvotne nejavili ako väčší problém, nakoľko bezpečnostnú kameru bolo možné bezproblémovo využívať aj pri 10, respektíve 15 snímkoch za sekundu. Problémom však začala byť situácia, kedy sa obraz začal pomocou RTSP, respektíve GStreameru, vysielať aby k nemu bolo možné pristúpiť v sieťovom nahrávacom zariadení. Nakoľko musí prebiehať spracovanie obrazu, ktorý je následne vpisovaný do streamu je každé ďalšie prerušenie tejto rutiny sťažením výpočetného času potrebného pre dané spracovanie a zaslanie obrazu.

Výsledkom bolo, že vysielanie do NVR, ktoré ponúkalo aj automatickú detekciu tváre bolo nestále a častokrát sa stávalo, že vysielanie bolo na pár sekúnd prerušené.

Možným riešením tohto problému bolo pripojenie priamo do Raspberry Pi z iného zariadenia v sieti a spustenie vysielanie iba lokálne bez RTSP vysielania, čím by však došlo k narušeniu ONVIF štandardu a nemožnosti integrácie s inými kamerami.

Preto bolo pristúpené k možnosti, kedy je možné rozpoznávanie tváre zapnúť dodatočne pomocou prepínača `-face` alebo `-f` v spúšťacom skripte `launch.sh`. V prípade zapnutia tejto možnosti však nie je garantovaná stabilita vysielania. Implicitne je detekcia vypnutá čím je stabilita garantovaná.



Obr. 8.2: **Testovanie detekcie tváre** - z obrázku vidieť, že po korekcii skreslenia sú z obrazu vykalkulované súradnice kde sa tvár nachádza a je možné tvár v obraze označiť napríklad pomocou zeleného štvorca.

8.1.3 Pan-Tilt-Zoom

Testovanie pan-tilt-zoom servisu pozostáva z dvoch častí. V prvej časti prebehla priama konfrontácia s užívateľmi, ktorí podľa popisu v kapitole 7.5.1, mali za úlohu nastaviť pan-tilt-zoom servis priamo v Shinobi. Daný servis je nevyhnutnou súčasťou správneho fungovania pohybu kamery a tak bolo potrebné docieliť, aby bolo nastavenie na užívateľskej strane jasné a bezchybné.

Vzhľadom na pandemickú situáciu bola nutnosť testovanie obmedziť na rodinných príslušníkov. Celkovo sa testovania zúčastnilo 5 ľudí. Samotné testovanie prebiehalo priamočiaro. Nikto z testovanej vzorky nemal väčší problém s nastavením kamery a pan-tilt-zoom servis bol v priemere uvedený do pohybu za čas okolo 4 minút.

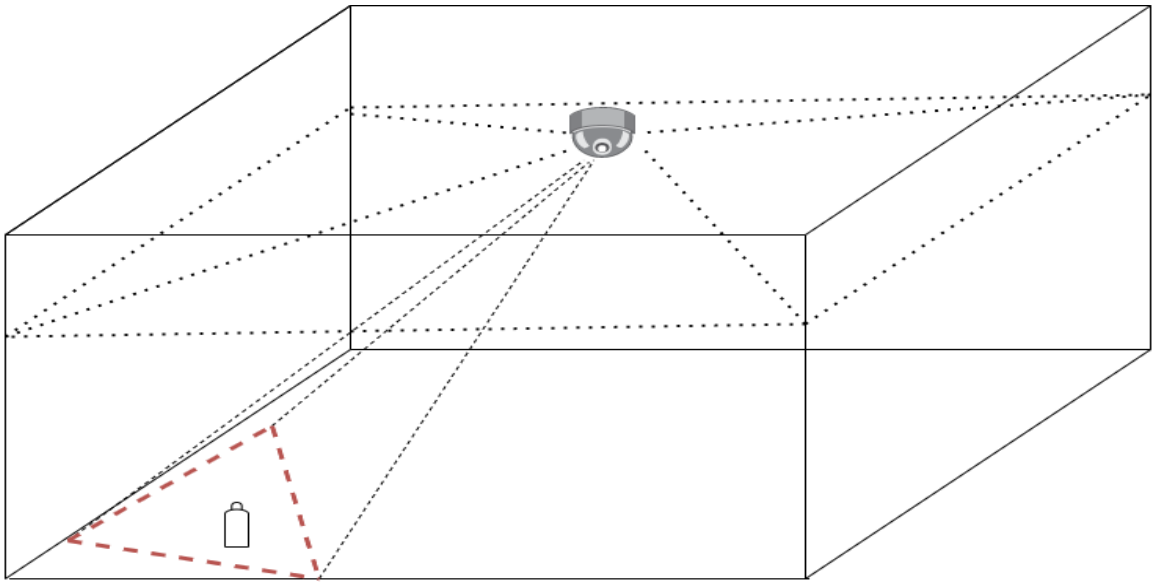
Po úspešnom nastavení bolo potrebné otestovať samotnú funkčnosť pan-tilt-zoom servisu v kamere. Pohyb kamery bolo možné vykonávať pomocou jednotlivých tlačidiel smeru pohybu nachádzajúcich sa priamo na obrazovke kamery v nahrávacom zariadení. Po každom stlačení tlačidla pohybu bolo možné v termináli bežiacom na Raspberry Pi pozorovať úspešne prijatú požiadavku na pohyb, ktorá zároveň obsahovala popis smeru kamery. Následne bolo možné vidieť v nahrávacom zariadení pohyb kamery v zvolenom smere.

Po pretestovaní vyššie zmienených bodov bolo možné vyhodnotiť funkčnosť kamery ako dostatočnú, čo umožňovalo prechod do ďalšej testovacej fázy, ktorá je zameraná na obrazovú časť, ktorá bola spracovávaná pomocou počítačového videnia.

8.2 Orientovanie sa v obraze

V predchádzajúcich kapitolách bol niekoľkokrát zmienený fakt, že podstatnou časťou práce je spracovanie fish-eye obrazu pre lepšiu orientáciu v zachytenom obraze. Nasledujúca kapitola pojednáva o testovaní tejto skutočnosti.

Testovanie orientovania sa v obraze spočíval v tom, že kamera bola umiestnená v rohu miestnosti a v tejto miestnosti bol ukrytý predmet, ktorý bolo potrebné vyhľadať. Testovanie však neprebiehalo iba na spracovanom obraze, ale taktiež na pôvodnom fish-eye obraze aby bolo možné porovnať výsledky vyhľadávania a teda či došlo k zlepšeniu celkového prehľadu obrazu kamery alebo naopak.



Obr. 8.3: **Testovanie obrazu** - z obrázku vidieť, že fish-eye obraz naznačený bodkovanou čiarou by mal v zábere obsahovať hľadaný objekt a rovnako tomu je aj v prípade spracovaného obrazu (naznačeného čiarkovanou červenou čiarou). Cieľom testovania je zistiť, či aj napriek tomu, že sa objekt v obraze nachádza, či je ho možné spozorovať v daných typoch obrazu.

Ako prvý hľadaný objekt bola zvolená sklenená váza. Váza bola umiestnená na relatívne viditeľnom mieste. Každý zo vzorky piatich ľudí bol schopný na obraze vyhľadať hľadaný objekt ako na fish-eye obraze tak aj na spracovanom obraze pomocou pan-tilt-zoomu.

Výsledky prvého testovania			
Účastník	Fish-eye obraz	Spracovaný obraz	Čas (MM:SS)
1. Účastník	Áno	Áno	01:53
2. Účastník	Áno	Áno	00:40
3. Účastník	Áno	Áno	00:15
4. Účastník	Áno	Áno	00:54
5. Účastník	Áno	Áno	01:00

Tabuľka 8.1: Stĺpce Fish-eye obraz a Spracovaný obraz určujú, či bol účastník schopný vyhľadať hľadaný predmet v obraze, pričom v stĺpci čas je odmeraná potrebná doba pre vyhľadanie predmetu.

Na obrázku 8.4 je tento test možné pozorovať. Na stole je umiestnená váza, ktorú je po chvíľke možné bezproblémovo zahliadnuť.

Obr. 8.4: **Prvé testovanie orientovania sa v obraze.** Na prvom obrázku vľavo je pôvodný fish-eye obraz, na ktorom je možné sklenenú vázu bezproblémovo zahliadnuť. Spracovaný obraz - obrázok vpravo, je kvôli miernemu priblíženiu mierne rozostrený, ale aj napriek tomu je možné vázu na stole bezproblémovo detekovať.

V ďalšej fáze bol však test upravený na náročnejšiu detekciu. Bolo nutné vyhľadať zlatú obálku, ktorá bola ukrytá a z pohľadu kamery sa nenachádzala úplne v strede obrazu, ale v spodnejšej časti zorného poľa čo vyhľadávanie značne sťažilo.

Iba traja z piatich respondentov boli schopní vyhľadať ukrytý predmet. Čo však považujem za úspech je, že dvaja z troch úspešných respondentov, ktorí boli schopní nápis vyhľadať tak učinili pomocou pan-tilt-zoomu. Až po nájdení objektu pomocou pan-tilt-zoomu boli schopní rozoznať objekt aj na fish-eye obraze.

Druhá fáza testovania potvrdila hypotézu, že pomocou spracovaného obrazu je možné lepšie detekovať jednotlivé objekty, nakoľko je z obrazu odstránené skreslenie, ktoré značne sťažuje detekciu v pôvodnom obraze.

Čo však testovanie tiež ukázalo je, že krajné body pri veľkom pan-tilt-zoome nie sú v kvalite v akej by boli očakávané a tak sa stále ponúka možnosť vylepšenia celkového spracovania obrazu. To, že pri spracovaní je stále priestor k zlepšeniu utvrdzuje testovanú hypotézu ešte viac, nakoľko aj bez tohto vylepšenia bolo možné objekty lepšie detekovať zo spracovaného obrazu.

Výsledky druhého testovania			
Účastník	Fish-eye obraz	Spracovaný obraz	Čas (MM:SS)
1. Účastník	Nie	Nie	–:–
2. Účastník	Áno	Nie	02:53
3. Účastník	Nie	Áno	04:15
4. Účastník	Nie	Áno	02:51
5. Účastník	Nie	Nie	–:–

Tabuľka 8.2: Stĺpce Fish-eye obraz a Spracovaný obraz určujú, či bol účastník schopný vyhľadať hľadaný predmet v obraze, pričom v stĺpci čas je odmeraná potrebná doba pre vyhľadanie predmetu.

Obr. 8.5: **Finálne testovanie orientovania sa v obraze.** Na prvom obrázku je pôvodný fish-eye obraz, na ktorom je nutné vyhľadať obálku. Tá sa nachádza medzi operadlom a sedacou časťou kancelárskej stoličky. Obálka je viditeľná aj na tomto obraze, avšak tým, že je zaberané veľké množstvo priestoru, veľa elementov pôsobí rušivo čo môže zabrániť všimnutiu si hľadaného predmetu. Na dolnom spracovanom obraze je možné vidieť, že po využití pan-tilt-zoom softvéru bolo možné po priblížení na stoličku jasnejšie spozorovať hľadaný predmet.

Kapitola 9

Záver

Cieľom tejto práce bolo vytvorenie bezpečnostnej kamery založenej na Raspberry Pi, ktorú je možné ovládať pan-tilt-zoom softvérom a to za použitia kamerového modulu s objektívom typu rybie oko, ktorý zachytáva obraz o veľkosti 200 stupňov. K úspešnému vypracovaniu práce bolo nutné oboznámiť sa so základmi Raspberry Pi a príslušným kamerovým modulom, prvkami počítačového videnia a ONVIF štandardom.

Pred pristúpením k implementácii bolo nutné vyhotoviť návrh riešenia, ktorý dopomohol k detailnejšiemu naštudovaniu zmieňovaných technológií a lepšiemu prehľadu pri ich spoločnom prepojení. Ukázalo sa, že aj na prvý pohľad správne vytvorený návrh riešenia môže byť chybný, keďže ako prvotný modul z knižnice OpenCV bol použitý Fisheye modul, ktorý sa neskôr pri implementácii ukázal ako nedostatočný pre kamery s veľkým zorným polom a musel byť nahradený Omnidir modulom. Ako tiež bolo spomenuté, niektoré technológie boli z môjho pohľadu úplne nové, ako tomu bolo predovšetkým v prípade ONVIF štandardu, s ktorého technológiami a implementáciou som sa zoznámil až počas vypracovávania tejto práce. Navyše preštudovanie danej problematiky ukázalo, že nielen ONVIF štandard, ale kamerové štandardy vo všeobecnosti nemajú obrovskú popularitu pri DIY systémoch, pretože pre ľudí, ktorí s danými štandardmi nikdy nepracovali neponúkajú detailnejší prehľad pre implementáciu, ale len relatívne zložito písanú špecifikáciu jednotlivých profilov. Ako dôležitou súčasťou sa tiež ukázalo preštudovanie dostupných riešení, nakoľko na základe tejto štúdie bolo možné sledovať prednosti, ktoré dané implementácie ponúkajú, ale taktiež nevýhody a nedostatky.

Pri implementácii bola následne využitá predovšetkým knižnica OpenCV, ktorá pomocou modulu Omnidir a základných funkcií ponúkala možnosť prevedenia obrazu s veľkým skreslením na obraz bez skreslenia pomocou korekcie. Na základe preštudovaných a následne naimplementovaných technológií bolo možné sa po spracovanom obraze presúvať do všetkých strán, čo simuluje požadovaný pan-tilt-zoom. Vyhotovená implementácia spracovania obrazu bola následne pomocou Raspberry Pi ONVIF Serveru (RPOS) prepojená so základnými prvkami ONVIF štandardu. Kamera je po transformácii na ONVIF kameru prenesená do nahrávacieho zariadenia Shinobi, odkiaľ je možné ju ovládať. Implementovaná kamera je schopná zachytávať 200 stupňový obraz, ktorý následne spracuje a ponúka pan-tilt-zoom pohyb. Kamera taktiež obsahuje ONVIF štandard podľa profilu S, s výnimkou vlastnej implementácie pan-tilt-zoom servisu. Priestor pre zlepšenie sa určite naskytá pri spracovaní obrazu, predovšetkým krajných častí obrazu, kedy pri veľkom pan-tilt-zoome je obraz mierne perspektívne skreslený čo však neprekáža plnohodnotnému využitiu kamery.

Literatúra

[1] BezpečnostnéKamery.sk. *IP vs. analógové kamery a základné pojmy* [online]. [cit. 2021-03-02]. Dostupné z: https://www.bezpecnostnekamery.sk/clanky/ip-vs-analogove-kamery-a-zakladne-pojmy.

[2] Dawson Howe, K. *A Practical Introduction to Computer Vision with OpenCV*. 1. vyd. John Wiley and Sons Ltd, 2014. ISBN 978-1-118-84845-6.

[3] Franks, J., Hallam Baker, P., hostetler, J., Lawrence, S., Leach, P. et al. *HTTP Authentication: Basic and Digest Access Authentication* [online]. The Internet Society, jún 1999 [cit. 2021-03-18]. Dostupné z: https://tools.ietf.org/html/rfc2617.

[4] Gigatux.nl. *SOAP Protocol Bindings* [online]. [cit. 2021-03-30]. Dostupné z: http://books.gigatux.nl/mirror/buildingwebserviceswithjava/0672321815_ch03lev1sec12.html.

[5] Henson, D. B. *Visual Fields*. 1. vyd. Oxford University Press, 1993. ISBN 978-0192623614.

[6] ONVIF. *Resources* [online]. [cit. 2021-04-02]. Dostupné z: https://www.onvif.org/resources/.

[7] ONVIF. *ONVIF Profile S Specification* [online]. ONVIF, november 2019 [cit. 2021-03-27]. Dostupné z: https://www.onvif.org/wp-content/uploads/2019/12/ONVIF_Profile_-S_Specification_v1-3.pdf?26d877&26d877/.

[8] ONVIF. *ONVIF Core Specification* [online]. ONVIF, december 2020 [cit. 2021-03-27]. Dostupné z: https://www.onvif.org/specs/core/ONVIF-Core-Specification.pdf/.

[9] OpenCV. *Releases* [online]. [cit. 2021-04-10]. Dostupné z: https://opencv.org/releases/.

[10] OpenCV. *Camera Calibration* [online]. OpenCV, apríl 2021 [cit. 2021-04-08]. Dostupné z: https://docs.opencv.org/3.4/dc/dbb/tutorial_py_calibration.html.

[11] OpenCV. *Camera calibration With OpenCV* [online]. OpenCV, marec 2021 [cit. 2021-03-18]. Dostupné z: https://docs.opencv.org/master/d4/d94/tutorial_camera_calibration.html.

[12] OpenCV. *Cascade Classifier* [online]. OpenCV, marec 2021 [cit. 2021-04-10]. Dostupné z: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.

[13] OpenCV. *Cv::Mat Class Reference* [online]. OpenCV, marec 2021 [cit. 2021-03-15]. Dostupné z: https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#details.

[14] OpenCV. *Fisheye camera model* [online]. OpenCV, apríl 2021 [cit. 2021-04-12]. Dostupné z: https://docs.opencv.org/3.4/db/d58/group__calib3d__fisheye.html.

[15] RaspberryPi.org. *Camera Module* [online]. [cit. 2021-02-17]. Dostupné z: https://www.raspberrypi.org/documentation/hardware/camera/.

[16] RaspberryPi.org. *Raspberry Pi OS* [online]. [cit. 2021-02-17]. Dostupné z: https://www.raspberrypi.org/documentation/raspbian/.

[17] Alam, M. *TypeError: s.getConfigWithBranding is not a function* [online]. 2021 [cit. 2021-04-28]. Dostupné z: https://hub.shinobi.video/articles/view/sIuhLW2A0E8A7K3.

Príloha A

Návod pre inštaláciu kamery

Potrebná inštalácia závislostí pre korektné fungovanie pythonovského skriptu pre spracovanie obrazu bola popísané v kapitole 7.1.

Ďalšou nutnou súčasťou je správne nakonfigurovanie samotnej kamery formou spustenia pribalených zdrojových kódov z pamäťového média, prípadne alternatívne je možné zdrojové súbory naklonovať z github repozitára pomocou príkazu
`git clone https://www.github.com/bixorko/Bakalarska-praca`.

Je potrebné spustiť pripravený shellovský skript `configure.sh`, ktorý najprv predinštaluje chýbajúce závisloti a následne spustí Raspberry Pi ONVIF Server, pričom počas spúšťania nastaví IP adresu v súbore
`rposConfig.json` na IP adresu, ktoré má Raspberry Pi priradené v sieti. Následne je spustený samotný skript pre spracovanie obrazu.

Skript `configure.sh` navyše pri spustení skontroluje, či je na Raspberry Pi predinštalovaný program Shinobi. V prípade že nie (najčastejšia situácia pri prvom spustení), nainštaluje Shinobi priamo na Raspberry Pi.

Následne je možné pristúpiť do Shinobi z ľubovoľného zariadenia v sieti pomocou
`<RaspberryPiIP>:8080`. *Pri prvom spustení po inštalácii je nutné prihlásiť sa ako superuser cez url* ***`<RaspberryPiIP>:8080/super`*** *kde implicitné používateľské meno je* ***admin@shinobi.video*** *a heslo* ***admin****. Následne je v tejto administrácii nutné vytvoriť používateľské konto, do ktorého je sa možné prihlásiť zo spomínanej url* ***`<RaspberryPiIP>:8080`***.
Po prihlásení do užívateľského konta je nutné pridať kameru.

V prípade, že pre kameru bude využívaný predinštalovaný softvér Shinobi, je po prihlásení nutnosť kliknúť na tlačidlo **Add Camera**, následne **Options** v ľavom dolnom rohu, **Import** a **Upload File**.
Ako súbor zvolíme predom pripravený JSON súbor `shinobiCameraSettings.json` a potvrdíme. Tento predpripravený JSON súbor umožňuje bezproblémové pridanie kamery bez nutnosti špecifikovať jednotlivé polia pre vysielanie a čo je najdôležitejšie, nie je nutná manuálna konfigurácia pan-tilt-zoom softvéru.

Alternatívne je možné po stlačení tlačidla **Add Camera** vyplniť jednotlivé polia manuálne a to následovne:

Nastavenie Shinobi v Add Camera	
Nastavenie	Hodnota
Full URL Path	rtsp://<meno>:<heslo>@<RPiIP>:8554/onvif1
ONVIF	Yes
Non-Standard ONVIF	Yes
ONVIF Port	8081
Monitor Capture Rate	10
Controllable	Yes
Custom Base URL	http://<RPiIP>:8081
Call Method	GET (Default)
Center	/center
Left	/left
Right	/right
Up	/up
Down	/down
Zoom Out	/zoom-out
Zoom In	/zoom-in

Tabuľka A.1: Tabuľka popisujúca manuálne nastavenie kamery v nahrávacom zariadení Shinobi

Obdobná manuálna konfigurácia je taktiež nutná v ľubovoľnom nahrávacom zariadení, ktoré bolo zvolené ako alternatíva k predinštalovanému Shinobi. Nespomenuté polia treba ponechať v implicitnom nastavení.

A.1 Možné problémy pri inštalácii Shinobi

Ako uvádza oficiálny web Shinobi [17], je možné, že po inštalácii samotného Shinobi NVR nebude možné pristúpiť do užívateľského rozhrania na URL `<RaspberryPiIP>:8080` respektíve `<RaspberryPiIP>:8080/super`, nakoľko Node.js verzia na zariadení môže byť príliš zastaralá.

V prípade, že sa takýto problém vyskytol je nutné nasledovať kroky popísané nižšie. :

1. Po otvorení terminálu `sudo su`
2. curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
3. `export NVM_DIR="$HOME/.nvm"`
4. `[ -s "$NVM_DIR/nvm.sh"] && ̇"$NVM_DIR/nvm.sh"`
5. `[ -s "$NVM_DIR/bash_completion"] && ̇"$NVM_DIR/bash_completion"`[1]
6. nvm install 12
7. nvm use 12
8. npm i npm -g
9. npm i pm2 -g
10. pm2 update
11. cd /home/Shinobi - respektíve do adresára kde je Shinobi nainštalované
12. sh UPDATE.sh
13. pm2 restart all
14. npm install forever -g && pm2 kill && pm2 unstartup && pm2 save && forever start camera.js

[1] Tretí až piaty bod je nutné zadať ako jeden príkaz

Príloha B

Návod pre overenie správnosti kalibrácie

V prípade, že po spustení kamerového systému je obraz posunutý, respektíve nie je úplne odstránená korekcia skreslenia z pôvodného obrazu, je možné overiť výsledok kalibrácie pomocou pythonovského skriptu `calibrationTest.py`.

Tento súbor môže byť spustený aj mimo Raspberry Pi, pričom však musí platiť, že skript musí mať k dispozícii vyhotovené snímky s kalibračným vzorom. Po spustení programu vo formáte `python3 calibrationTest.py` sú postupne vyzobrazované jednotlivé kalibračné snímky, pričom musia obsahovať farebné krúžky pospájané v rohoch kalibračného vzoru rovnako, ako je tomu na obrázku 7.3. V prípade, že program nebol schopný detekovať vzor na obrázku, je nutné vyhotoviť sadu snímkov ešte raz v rôznych uhloch a vzdialenostiach kamery a kalibračného vzoru.