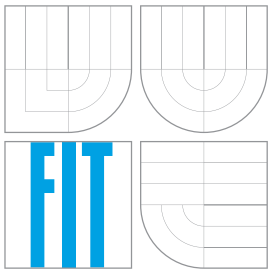
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

STEREOVIZNÍ SYSTÉM PRO POČÍTÁNÍ CESTUJÍCÍCH V HROMADNÝCH DOPRAVNÍCH PROSTŘEDCÍCH

PASSENGER COUNTING SYSTEM BASED ON STEREOVISION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADEK VRZAL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV ROZMAN, Ph.D.

BRNO 2016

Zadání diplomové práce/18934/2015/xvrzal06

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Vrzal Radek, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Stereovizní systém pro počítání cestujících v hromadných dopravních prostředcích**
Passenger Counting System Based on Stereovision
Kategorie: Zpracování obrazu

Pokyny:

1. Nastudujte princip stereovize a sledovacích algoritmů. Zaměřte se na algoritmy pro získání tzv. hustých disparitních map a sledovací algoritmy schopné sledovat více hypotéz.
2. Seznamte se s hardwarovým a programovým vybavením stereovizní jednotky UPC (Unit of People Counting), a to zejména s osazenými kamerovými moduly a výpočetním výkonem instalovaného ARM procesoru iMX6.
3. Navrhněte systém pro počítání cestujících v hromadných dopravních prostředcích, který poběží na jednotce UPC.
4. Navržený systém implementujte a testujte na poskytnuté sadě reálných záznamů.
5. Zhodnoťte přesnost a robustnost navrženého systému. Uveďte možnosti dalšího vylepšení.

Literatura:

- STONE, Lawrence D., Roy L. STREIT, Thomas L. CORWIN a Kristine L. BELL. *Bayesian Multiple Target*

Při obhajobě semestrální části projektu je požadováno:

- První tři body zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Rozman Jaroslav, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato práce pojednává o návrhu systému pro automatické počítání cestujících v dopravních prostředcích. Jednotky počítající cestující jsou umístěny na stopě dveřního prostoru vozidla. Cestující jsou detekováni na disparitní mapě vypočtené ze snímků stereo-kamery. Sledování cestujícího probíhá pomocí sledovacího algoritmu Global nearest neighbor a Multiple hypothesis tracking. Systém je používán pro získávaní dat pro přepravní průzkumy veřejné dopravy.

Abstract

This thesis deals with a concept of system for automatic passenger counting in different modes of transport. Counting units are placed in top of the door area in the vehicle. Passengers are detected at the disparity map counted from the stereo-camera images. Object tracking is achieved with Global nearest neighbor and Multiple hypothesis tracking algorithm. This system is used for public transportation surveys.

Klíčová slova

automatické počítání cestujících, stereo vidění, disparita, sledování cíle, sledování více cílů

Keywords

automatic passenger counting, stereo vision, target tracking, multi target tracking

Citace

VRZAL, Radek. *Stereovizní systém pro počítání cestujících v hromadných dopravních prostředcích*. Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Rozman Jaroslav.

Stereovizní systém pro počítání cestujících v hromadných dopravních prostředcích

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D., další informace mi poskytl Ing. David Herman. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Radek Vrzal
25. května 2016

Poděkování

Chtěl bych poděkovat Ing. Jaroslavu Rozmanovi, Ph. D. za vedení mojí diplomové práce. Dále bych chtěl poděkovat Ing. Davidu Hermanovi za konzultace a možnost pracovat na takovém projektu. Data pro implementaci a testování této práce byla poskytnuta firmou RCE systems s.r.o.

Obsah

Kapitola 1

Úvod

Přepravní průzkumy se provádí v dopravních podnicích všech větších měst a nabyly na důležitosti po vzniku integrovaných dopravních systémů. Dopravní podniky je organizují z důvodu zkvalitňování služeb cestujícím a zefektivnění provozu dopravního podniku. Výsledná data se poté použijí například pro změny intervalů odjezdů vozů na trasách, změny kapacity vozů, popřípadě přepracování tras. Pokud je v integrovaném systému více společností používají se tyto průzkumy také pro rozdělování tržeb mezi tyto společnosti.

Pro provedení přepravního průzkumu existuje několik metod, ty mají různý výstup. Zadavatel tedy musí zvolit metodu, která poskytne co nejvíce požadovaných informací. Mezi tyto metody patří ústní nebo písemné dotazníky. V nich ze zjišťují informace o trase, na které cestující jede, o druhu jízdného, které si kupuje, a o obecné spokojenosti. Další metodou je tzv. lístková metoda. Cestující si před nástupem do dopravního prostředku vezme lístek a v cílové stanici jej odevzdá. Na lístku je označena výchozí stanice a při odevzdání je zaznamenán čas.

Mezi nejpoužívanější metody přepravních průzkumů patří počítání cestujících pomocí jednorázově zaměstnaných pracovníků, kteří jsou umístěni ve vozech nebo na zastávkách. Ti dokážou zadavateli dát informaci o výstupech a nástupech na jednotlivých zastávkách, časech příjezdu na zastávku, odjezdu ze zastávky a vytíženosti vozu na jednotlivých úsecích trasy. Nevýhodou pracovníků na zastávkách je jejich nepřesnost při větších počtech cestujících vyskytujících se na zastávce. Pro počítání cestujících ve vozech je nutné zaměstnat velký počet pracovníků, jelikož vozů je mnohem více než zastávek. Právě v této metodě dokáže pracovníky nahradit systém automatického počítání cestujících, kterým se zabývá tato práce.

1.1 Cíl práce

Cílem práce je navrhnout, implementovat a otestovat systém pro počítání osob v hromadných prostředcích, který poběží na jednotce UCP-01. Každá tato jednotka bude umístěna nad dveřmi vozu hromadné dopravy (autobus, trolejbus, tramvaj) a bude v reálném čase detekovat vstupy a výstupy cestujících danými dveřmi. Tyto informace bude jednotka zasílat palubnímu počítači ve voze, který je poté bude dále zpracovávat.

1.2 Přehled kapitol

V kapitole 2 se práce zabývá analýzou problému, který je předmětem této práce. Jsou stanoveny základní parametry scény, která bude zachytávána, systému na kterém aplikace poběží a prostředí, kde se bude systém nacházet. Kapitola 3 se zabývá návrhem problému. Je zde naznačen způsob řešení jednotlivých podproblémů jako získání disparitní mapy, detekce cestujících na ní a jejich následné sledování. Kapitola 4 je popisem implementace jednotlivých kroků z návrhu, také je zde popsána kalibrace a instalace systému do vozidla. Metody použité při testování a vyhodnocení funkčnosti a použitelnosti celé aplikace se nachází v kapitole 5. V kapitole 6 se nachází shrnutí výsledků, možnosti vylepšení aplikace a také možnosti jak systém v budoucnu vylepšit.

Kapitola 2

Analýza problému

Systém bude použit pro počítání cestujících, kteří prošli dveřmi ve voze hromadné dopravy. Výstupem bude pouze číselná hodnota počtu lidí, kteří vystoupili a nastoupili. V rámci rozšíření může být systém použit jako bezpečnostní kamera sledující dveřní prostor. Obraz by se pak odesílal do palubního počítače vozu.

2.1 Kompozice scény

Jednotka UCP-01 je umístěna na stropě ve dveřním prostoru vozu na stejné úrovni jako střed dveří. Při instalaci je možno s ní otáčet ve směrem ven a dovnitř vozidla. Jednotka je směrována dolů, to znamená že, cestující zabírá shora. Maximální šířka dveřního prostoru autobusů a tramvají je asi 1,2 m. To je dostatečný prostor na to, aby prošli až 3 cestující zároveň. Jednotka je umístěna ve výšce 2,1 - 2,5 m nad podlahou. Jednotka by měla být schopna zaznamenat cestující vyšší než 1,6 m. Maximální výšku cestujícího budeme předpokládat 2 m. To znamená, že nejvyšší přesnost by měla být do vzdálenosti 1 m od kamery. Na obrázku 2.1 je ukázána jednotka UCP-01 instalovaná ve voze. Samotná jednotka je uložena v plechové krabičce.

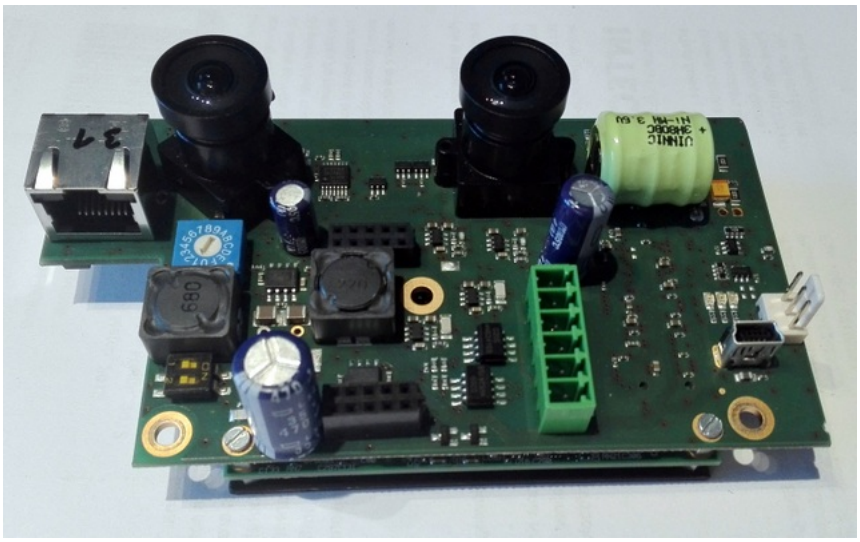
Obrázek 2.1: Umístění jednotky ve dveřním prostoru vozidla.

Vzhledem k technickým omezením jednotka nemůže být umístěna přesně nad nástupní hranou vozidla. Vzdálenost jednotky od kolmice k nástupní hraně je asi 25 - 60 cm podle typu vozidla. To bude mít negativní vliv na správné fungování systému, protože nebude možné přesně určit, kdy cestující prošel přes nástupní hranu.

Při návrhu řešení bude třeba brát ohled na několik faktorů, které mohou ovlivnit kvalitu výsledku. Dveře jsou prosklené, totéž platí o ohraničení dveřní oblasti ve většině vozů. Na skleněných plochách vznikají odrazy, které mohou mít negativní vliv na tvorbu hloubkové mapy. Také je nutno brát ohled na různé světelné podmínky ve vozech. Vozy jezdí ve dne i v noci a uvnitř se střídá přirozené osvětlení s umělým. Dalším faktorem jsou cestující, kteří mohou poškodit sklíčko před kamerami nebo pohnout s krabičkou, ve které je jednotka umístěna.

2.2 Jednotka UCP-01

Jednotka UCP-01 má rozměry 120 x 92 x 50 mm a tvořena dvěma propojenými plošnými spoji. Jeden je system-on-module CM-FX6. Druhý obsahuje kamery a periferie jako ethernetový konektor, slot na SD kartu a další. Tyto plošné spoje jsou propojeny pomocí dvou 140-pinových konektorů. Jednotka je napájena po síťovém kabelu pomocí technologie Power over Ethernet. Jednotka je umístěna v plechové krabičce, která zároveň zajišťuje chlazení. Před objektivy je umístěno ochranné plexisklo.



Obrázek 2.2: Jednotka UCP-01.

2.2.1 Vybavení

Hardwarová specifikace jednotky:

- procesor: ARM Cortex-A9 dual core 1 GHz Freescale i.MX6 family
- grafický akcelerátor: Vivante GC2000 Series, (4 x Vec-4, 16 x Vec-1 Shader Cores), čip podporuje OpenGL ES 2.0 a OpenCL EP
- operační paměť: 1 GB DDR3-1066

- paměť: NAND flash 512 MB
- kamerové čipy: CMOS, 1280 x 960, 45fps, šedotónové, velikost pixelu 3.75 µm
- objektivy: ohnisková vzdálenost 2.1 mm, horizontální zorný úhel 97°

Na jednotce běží operační systém Linux verze jádra 3.0.35, který je dodáván společně s použitou deskou. Do jádra jsou přidána rozšíření pro nastavení diod na deskách, přepínačů a dalších vstupně výstupních zařízení. Také je rozšířeno o ovladače pro kamerové čipy. Kamery se ovládají přes rozhraní Video 4 Linux 2. K jednotce je možné se připojit pomocí sériového kabelu nebo po síti přes protokol Telnet.

Na jednotce UCP-01 je grafický akcelerátor podporující OpenCL EP. Ten může značně urychlit výpočty související s real-time zpracováním obrazu. Byly provedeny testy rychlosti různých algoritmů zpracování obrazu a porovnání rychlostí provedení na procesoru a grafickém akcelerátoru. Všechny testované algoritmy by se mohly klasifikovat jako per-pixel operace (úkon prováděný s každým pixelem obrazu). U jednodušších operací je rozdíl v časech zpracování malý, ale u složitějších je implementace v OpenCL rychlejší. Důvodem je nutnost nahrávat data na grafický čip a zpět což zabere nějaký čas a u jednodušších algoritmů mohou tyto přesuny zabrat více času, než samotné zpracování. Výsledky těchto testů jsou uvedeny v kapitole 5.

2.2.2 Stereo kamery

Výstupem z kamer jsou dva šedotónové obrazy, které se následně zpracovávají. Díky znalosti umístění kamer, typu grafického čipu a objektivu lze spočítat parametry celého systému. Ty jsou poté důležité pro správné umístění zařízení a interpretaci výstupů.

Zorný úhel

Na obrázku 2.3 je případ scény s polovinou dveří. Pro správné vytvoření hloubkové mapy je třeba, aby se objekt nacházel v zorném poli obou kamer. Jelikož je možno s jednotkou otáčet ve směru dovnitř-ven, budeme zkoumat pouze horizontální zorný úhel. Ten je možno spočítat z ohniskové vzdálenosti f a šířky senzoru h pomocí vzorce:

$$FOV = 2\tan^{-1}\left(\frac{h}{2f}\right) \tag{2.1}$$

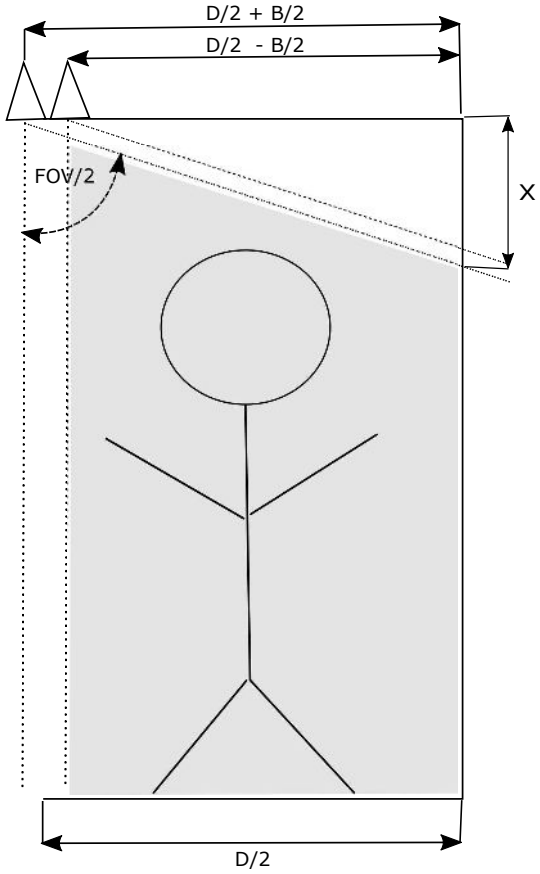
Kamery instalované na jednotce UCP-01 mají horizontální zorný úhel 96°.

Dále je možné spočítat od jaké vzdálenosti dokážou kamery zachytit celou šíři dveřního prostoru. Tuto vzdálenost X lze spočítat pomocí:

$$X = \tan\left(90° - \frac{FOV}{2}\right) * \left(\frac{D}{2} + \frac{B}{2}\right), \tag{2.2}$$

kde D je šířka dveřního prostoru a B je báze stereo-kamery (vdálenost kamer od sebe).

Šířka dveřního prostoru je nejčastěji maximálně 1,2 m a velikost báze 50 mm. Po dosazení vyjde v tomto případě vzdálenost 55 cm. Pokud bude zařízení ve výšce 2,1 m, nebudou vidět objekty, které se nachází na kraji dveřního prostoru ve výšce vyšší než 155 cm. To ale nebude mít negativní vliv na funkci systému, protože cestující ve dveřním prostoru zabere třetinu jeho šířky, tak bude viditelný alespoň částečně.



Obrázek 2.3: Pokrytí pravé poloviny dveřního prostoru kamerami.

Hloubková přesnost

Pro zjištění vzdálenosti bodu na kameře je možné vypočítat disparitu daného bodu. Vztah disparity a vzdálenosti však není lineární. Na obrázku 2.4 je vidět, že blíže kameře je systém přesnější a se zvyšující se vzdáleností se přesnost snižuje.

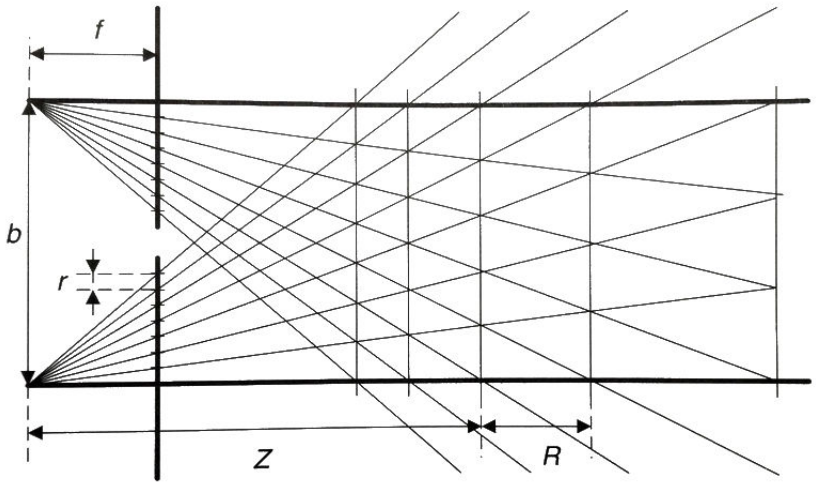
Vztah změny disparity a změny vzdálenosti je dán vzorcem:

$$R = \frac{rZ^2}{fb - rZ}, [6, s.\ 60] \tag{2.3}$$

kde R je hloubkové rozlišení (velikost změny reálné hloubky vzhledem ke změně kroku disparity v dané hloubce Z), r je velikost snímacího elementu senzoru, f je ohnisková vzdálenost.

Z tohoto vzorce lze odvodit, že pro zvýšení přesnosti je možno zvýšit ohniskovou vzdálenost f. To může být nežádoucí, protože se sníží zorný úhel. Další možností je zvětšení báze (vzdálenost kamer), negativním dopadem bude však zvýšení minimální vzdálenosti, od kterých bude možné počítat hloubku. Další možností jak zpřesnit výpočet hloubky je snížit velikost snímacího elementu senzoru (zvýšit rozlišení, vyměnit senzor za menší). Negativním dopadem této možnosti je však velký nárůst časové složitosti algoritmu.

Rovnice 2.3 platí pod podmínkou, že $fb \neq rZ$. Pokud by tato podmínka neplatila, dostane se R do svojí limitní hodnoty. Výsledkem by bylo $R = \infty$. Po úpravě této podmínky lze následně spočítat maximální vzdálenost, u které je stereovizní systém schopen určit hloubku, dle vzorce:



Obrázek 2.4: Ukázka snižující se přesnosti se zvyšující se vzdáleností [6, s. 59].

$$Z = \frac{bf}{r} \tag{2.4}$$

V tabulce 2.1 jsou vypsány hloubková rozlišení pro jednotku UCP-01 s různými rozlišeními obrazu. Do vzdálenosti 1 m jsou hodnoty uspokojivé pro všechna zkoumaná rozlišení obrazu. Pro větší vzdálenosti je už rozlišení 160x120 nedostatečné protože změna disparity o 1 ve vzdálenosti 1,5 m v tomto případě znamená rozdíl 1,12 m. Tak by bylo složité cestujícího zachytit, protože rozdíl disparit osoby, která by byla ve vzdálenosti 1,5 m od kamery, a podlahy ve vzdálenosti 3 m od kamery, by byl velice nízký. Dále se v tabulce nachází vzdálenostní limity jednotlivých konfigurací. Od této vzdálenosti již není možné odhadnout vzdálenost objektu.

	Rozlišení			
Vzdálenost Z[m]	1280x960	640x480	320x240	160x120
0,1	0,0004	0,0007	0,0015	0,0029
0,2	0,0014	0,0029	0,0059	0,0121
0,3	0,0032	0,0066	0,0134	0,0281
0,5	0,0091	0,0185	0,0385	0,0833
1,0	0,0370	0,0769	0,1677	0,4000
1,5	0,0849	0,1800	0,4091	1,1250
Limit	28	14	7	3,5

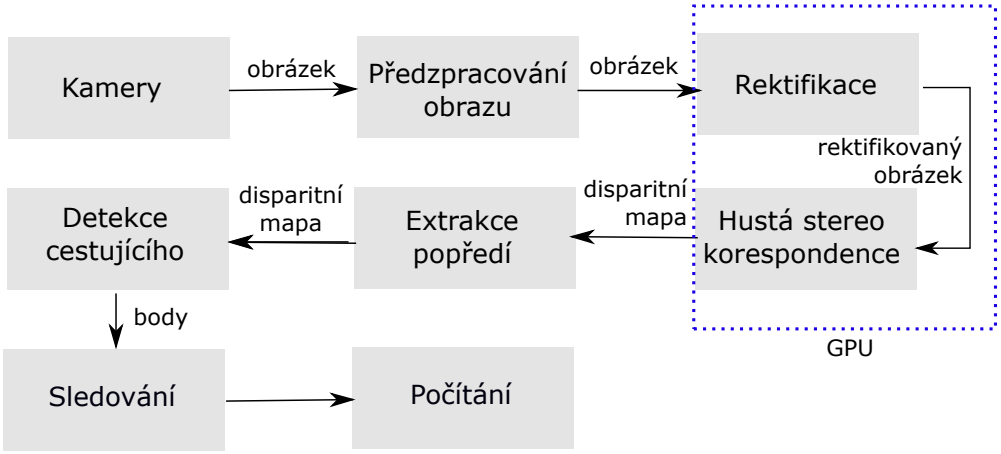
Tabulka 2.1: Hloubkové rozlišení kamer R[m] na zařízení UCP-01 ve vztahu ke vzdálenosti

Pro zvýšení hloubkového rozlišení stereo-systému je možné počítat disparitu sub-pixelově. Jedná se však jen o interpolační zpřesnění, které není tak účinné jako zvýšení rozlišení senzoru.

Kapitola 3

Návrh systému

Na obrázku 3.1 je zobrazena koncepce systému. Po sejmutí snímků kamerami dochází k jejich předzpracování, konkrétně ekvalizaci histogramu. Následně je provedena rektifikace snímků. Jedná se o transformaci snímků, při níž dochází k řádkovému zarovnání. Následuje výpočet disparitní mapy, ze které se následně vyextrahuje popředí, tj. odstraní se objekty, které se nachází v prázdném autobuse (madla, znehodnocovače lístků, ...). Dále se na ni provádí detekce cestujících. Výstupem jsou jejich souřadnice. Ty jsou zpracovány sledovacím algoritmem. Ten přiřadí nalezené cestující k již existujícím záznamům, popřípadě vytvoří nový, pokud do prostoru vstoupí nový cestující. Pokud cestující projde přes pomyslné brány ve dveřním prostoru, algoritmus inkrementuje hodnotu příchodů nebo odchodů.



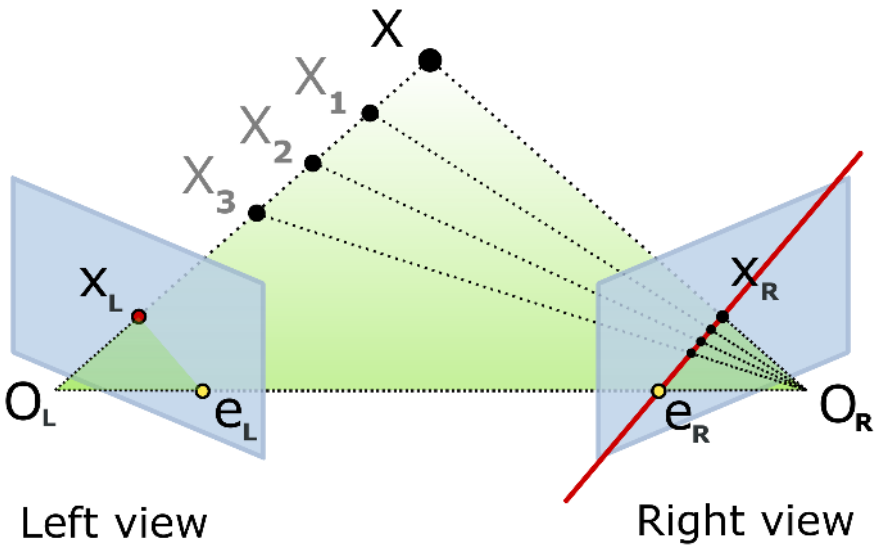
Obrázek 3.1: Diagram navrhovaného systému.

3.1 Stereo vidění

Princip 3D počítačového vidění je postaven na epipolární geometrii. Jedná se o geometrii dvou pohledů na jednu scénu. Bod v prostoru se zobrazí na jeden bod v rovině jedné kamery a jeden bod v rovině druhé kamery. Na obrázku 3.2[1] je příklad scény se dvěma kamerami. Bod X v prostoru se na levé kameře zobrazí jako bod X_L. Je to průsečík přímky $O_L X$ vedené od středu promítání levé kamery k bodu X v prostoru. Na pravé kameře se bod zobrazí na

[1] Zdroj: https://en.wikipedia.org/wiki/Epipolar_geometry| licence: CC BY-SA 3.0

pozici X_R. Z obrázku je patrné, že se na bod X_l na levé kameře může zobrazit více bodů. Jsou to ty, které leží na stejné přímce jako bod X. Všechny tyto body se pak na pravé kameře zobrazí pouze na jedné přímce, která je průmětem přímky $O_L X$ do obrazové roviny pravé kamery. Tato přímka se nazývá epipolára. Přímka, která prochází středy promítání O_L a O_R se nazývá báze. Body ve kterých báze protíná jednotlivé obrazové roviny se nazývají epipóly (e_L, e_R). Všechny epipolární přímky na obrazové rovině procházejí epipólem.



Obrázek 3.2: Ukázka epipolární geometrie.

Pro zjednodušení hledání korespondencí se používá fundamentální matice. Ta mapuje polohu bodu levé obrazové roviny na přímku (epipoláru) pravé obrazové roviny. Pro hledání korespondencí následně stačí vyhledávat tento bod na epipoláře, a ne na celém snímku. Na obrázku je vyznačena rovina $O_L O_R X_L$, na které leží optické středy obou kamer a promítnutý bod na levé obrazové rovině. Průsečík této roviny s pravou obrazovou rovinou tvoří hledanou epipoláru [11].

3.1.1 Rektifikace snímků

Rektifikace je transformace páru stereo-snímků do společné obrazové roviny. Tato transformace způsobí posunutí a otočení původních obrazových rovin a také posunutí epipólů do nekonečna. Epipolární přímky se tak stanou rovnoběžnými a zároveň jsou zarovnány korespondující epipoláry z levého a pravého snímku. Korespondující body z levého snímku se tedy vyskytují na stejném řádku pravého snímku. To snižuje náročnost vyhledávání korespondencí, protože z dvourozměrného vyhledávání se stává jednorozměrné. Transformační matice pro rektifikaci jsou vypočteny z parametrů kamer získaných při kalibraci [15].

3.1.2 Hledání korespondencí

Algoritmy pro nalezení disparity u dvojice snímků se dělí na dvě velké podskupiny podle typu výstupní disparitní mapy. Jedná se o tzv. hustou a řídkou stereo korespondenci. Výstupem husté stereo korespondence je disparitní mapa, kde pro téměř každý pixel vstupního obrazu je vypočtena hodnota disparity. Řídká stereo korespondence naopak vypočítává

disparitu pouze pro některé význačné body v obraze. V tomto řešení je použita hustá stereo korespondence.

Na nalezení korespondujících bodů existuje mnoho algoritmů, ty se dělí na lokální a globální. Lokální pracují pouze s malou oblastí obrazu a vyhledávají korespondence pro každý bod samostatně. Globální používají informace o korespondencích v celém obraze. Kvůli možnosti paralelního zpracování na grafickém akcelerátoru je výhodnější použít lokální metodu.

Výpočet disparity je založen na metodě Area-based matching. Její princip spočívá ve vyhledávání podobných oblastí z levého snímku na snímku pravém. Díky předchozí rektifikaci snímků probíhá prohledávání pouze na řádku. Na základě parametrů systému a geometrie scény jsou stanovena omezení na vzdálenost vyhledávání. Jedná se o minimální a maximální hodnotu disparity. Pro každý bod na levém snímku dochází k hledání korespondujícího bodu na stejném řádku pravého snímku. Rozdíl jejich horizontální souřadnice je pak hledanou hodnotou disparity. Pokud by se však porovnávaly pouze hodnoty jednotlivých bodů na dvou obrázcích, byla by tato metoda náchylná na chyby. Proto se porovnávají čtvercové oblasti v okolí zkoumaných bodů.

Pro porovnávání oblastí je třeba zvolit metriku, která dokáže vyčíslit podobnost těchto oblastí. V tomto algoritmu se používá suma absolutních diferencí z okolí zkoumaného bodu (anglicky Sum of Absolute Differences - SAD), která se vypočítá pomocí vzorce:

$$D_{SAD} = \sum_{(i,j)\in U} \left| I_l(x+i, y+j) - I_r(x+d_x+i, y+j) \right|, \tag{3.1}$$

kde I_l a I_r jsou pravý a levý snímek, x, y jsou souřadnice bodu, pro který hledáme disparitu, U definuje velikost porovnávaného okolí bodu a d_x je aktuální zkoumaná hodnota disparity. Je to součet absolutních hodnot rozdílů porovnávaných pixelů [6, s. 238].

Řešení použité v této práci je úpravou výše zmíněného algoritmu prezentovanou v [8], která využívá vlastnosti integrálního obrazu. Integrální obraz je způsob reprezentace obrazu, který je optimalizovaný pro výpočty součtů hodnot jeho oblastí. Hodnota jednoho bodu je definována jako součet hodnot všech bodů ve sloupci nad ním, všech bodů v řádku nalevo a jeho samotné hodnoty.

$$ii(x, y) = \sum_{\substack{x' \leq x \\ y' \leq y}} i(x', y') \tag{3.2}$$

Standardní obraz je možné převést na integrální jedním průchodem. Hodnotu bodu $ii(x, y)$ v integrálním obraze lze spočítat z bodu $i(x, y)$ původního obrazu pomocí vzorce:

$$ii(x, y) = i(x, y) + ii(x-1, y) + ii(x, y-1) - ii(x-1, y-1), \tag{3.3}$$

kde $ii(-1, y) = 0, ii(y, -1) = 0$.

Součet hodnot S jakékoli čtvercové oblasti integrálního obrazu ii definované polohou levého horního rohu čtverce $[x, y]$, šířkou w a výškou h je pak možno vypočítat podle vzorce:

$$S = ii(x, y) + ii(x+w, y+h) - ii(x+w, y) - ii(x, y+h)[1] \tag{3.4}$$

3.1.3 Extrakce popředí

Ve dveřním prostoru se nachází plno předmětů. Ty mohou mít vliv na funkci sledovacího algoritmu. Našim cílem je odstranit z disparitní mapy veškeré předměty, které se nachází v prázdném dveřním prostoru (pozadí).

Během instalace zařízení do vozu bude provedeno učení modelu pozadí. Jedná se o zachycení a uložení disparitní mapy prázdného dveřního prostoru. Po výpočtu disparitní mapy dojde k jejímu porovnání s modelem pozadí a odstranění objektů, které se nachází na modelu pozadí. Výsledkem bude disparitní mapa zachycující pouze změny ve scéně, což budou cestující.

3.2 Detekce a sledování cestujících

Detekce se provádí na disparitní mapě. Cestující je vidět shora, vytváří tak lokální maximum. Je nutné brát ohled na výšku cestujícího a také na jeho chování. Cestující opírající se během průchodu dveřním prostorem o panel nad dveřmi nebo cestující nesoucí velký batoh by mohli být detekování špatně. Cestující je zde cílem detekce a sledování.

Cíl je, v kontextu detekce a sledování, objekt zájmu. Algoritmus sleduje jeho stav v průběhu času.

Detektor generuje takzvaná **měření**. Jedná se o detekované výskyty nějakého cíle obsahující informace o tomto cíli. V našem případě je to poloha detekovaného cíle na disparitní mapě. Tyto měření mohou pocházet ze třech různých zdrojů:

- Měření patří cíli, který již byl sledován v dřívějších detekcích
- Měření patří novému cíli
- Měření patří falešnému cíli - vznikl v senzoru nebo detektoru, nepatří cestujícímu a nemusí nás zajímat

Vstupem pro sledování cestujících jsou měření obsahující souřadnice v obraze. Úkolem sledovacího algoritmu (tracker) je jejich správné zpracování. Tracker přiřazuje měření do tras.

Trasa je sekvence měření, u kterých sledovací algoritmus vyhodnotil, že pochází z jednoho zdroje. V našem případě to je sekvence bodů na disparitní mapě, kterými se cestující podle sledovacího algoritmu pohyboval v čase. Nejsložitějším problémem je přiřazení správných bodů do správné trasy.

Funkci sledovacího algoritmu je možné rozdělit na 3 části:

- Přiřazení bodů do tras
- Výpočet pravděpodobností a predikcí
- Úprava tras

3.2.1 Přiřazení měření do tras

Ke každé trase, která je právě sledována, je předpočítána očekávaná poloha cíle. Kolem této polohy se nachází oblast přiřazení (anglicky gate). Pouze pokud jsou nalezené body v této oblasti, mohou být přiřazeny do dané trasy [3]. Pravidla pro přidělování nalezených bodů do jednotlivých tras závisí na zvoleném sledovacím algoritmu a budou zmíněny dále.

3.2.2 Predikce polohy objektu a výpočet pravděpodobností tras

Na predikci polohy následujícího bodu trasy se nejčastěji používá Kalmanův filtr. Ten na základě modelu systému, ve kterém je uchována poloha, rychlost a důvěryhodnost odhadu dokáže predikovat následující polohu bodu. Pracuje v diskrétním čase, což vyhovuje našemu užití.

Na základě stavu modelu je vypočtena očekávaná pozice, na základě aktuální důvěryhodnosti je vypočtena velikost oblasti přiřazení. Následně se nalezne bod, ve kterém se objekt nachází (výstup z detekce). Poloha predikovaná a nalezená se následně porovnají a vypočítá se hodnota pro polohu objektu. Rozdíl mezi predikovanou a naměřenou hodnotou se použije pro korekci důvěryhodnosti predikce. Pokud je systém důvěryhodný, pak je větší váha predikovaných souřadnic. Kalmanův filtr tedy provádí korekci i pro nepřesně naměřenou polohu objektů [14].

Pokud je to pro funkci algoritmu potřeba, jsou vypočteny pravděpodobnosti tras. Ty jsou závislé na počtu úspěšných přiřazení nalezených objektů do trasy, dále na diferencích mezi očekávanými polohami objektu a skutečnými měřeními a také na dalších parametrech modelu.

3.2.3 Úprava tras

V tomto kroku jsou upraveny informace o trasách.

Trasy se ve sledovacích algoritmech mohou nacházet ve třech různých stavech:

- Prozatimní (anglicky tentatitve) - Trasa, která vznikla z měření, které nebylo přiřazeno žádné trase. V dalším snímku se rozhodne, jestli se opravdu jedná o trasu odpovídající nějakému cíli, nebo se jednalo o chybnou detekci.
- Potvrzená - U této trasy je potvrzeno, že odpovídá existujícímu cíli. Existuje tedy posloupnost měření přiřazených k této trase.
- Smazaná - Tento typ odpovídá měřením, které byly detekovány jen v jednom snímku a byly klasifikovány jako falešný cíl, nebo potvrzeným trasám, které již není možné detekovat, protože cíle zmizely z dosahu.

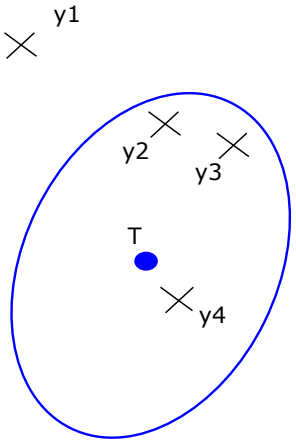
Každé měření, které není přiřazeno do extistující trasy, vytvoří prozatimní trasu. Pokud budou v následujících snímcích do této trasy přiřazeny další měření, bude potvrzena. U existujících tras se vyhodnocuje, jestli se ještě cíl nachází ve sledovaném prostoru. Pokud do trasy není několikrát po sobě přiřazeno měření, bude smazána.

3.2.4 Algoritmy pro sledování objektů

Existuje několik typů algoritmů pro sledování objektů, ty se klasifikují do různých skupin podle několika parametrů. Prvním je typ rozhodování, které provádí. Algoritmy s tvrdým rozhodováním přiřazují trase pouze jedno měření v oblasti předpovídané polohy objektu. Ostatní jsou klasifikovány jako nové trasy, nebo šum. Naopak algoritmy poskytující měkké rozhodnutí vrací pravděpodobnost, s jakou detekované měření patří do dané trasy. Dalším aspektem je počet tras, které dokážou tyto algoritmy sledovat. Dělí se na „jednocílové“ (single-target), a „vícecílové“ (multi-target). Posledním parametrem je způsob práce s přiřazovacími hypotézami. Ty se dělí na algoritmy používající jednu hypotézu a více hypotéz.

Nejbližší soused - Nearest neighbor

Metoda nejbližšího souseda je nejjednodušší způsob pro přiřazení detekovaných objektů k trase. Jedná se o algoritmus s tvrdým rozhodováním schopný sledovat jen jednu trasu.



Obrázek 3.3: Přiřazování objektů k trase u algoritmů pro sledování jednoho cíle.

Na obrázku 3.3 se nachází příklad asociace detekovaných objektů do trasy T. Do trasy se přiřazuje nejbližší objekt k predikované poloze sledovaného cíle. V našem případě je to bod $y4$.

Pravděpodobnostní přiřazení dat - Probabilistic data association

Tento algoritmus je také orientovaný na sledování jednoho cíle, ale výstupem je měkké rozhodnutí. Nejprve je vypočítána predikce výskytu cíle, který trasa sleduje. Následně je vypočtena pravděpodobnost pro všechny možnosti přiřazení měření do dané trasy. Ty by se v situaci, která je na obrázku 3.3, počítaly pro objekty $y2, y3, y4$ a pro situaci, že všechny tyto měření jsou falešné cíle [9].

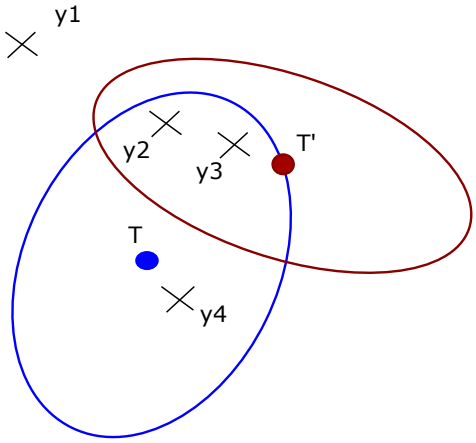
Algoritmy pro sledování více cílů

Jedná se o rozšíření dvou výše zmíněných algoritmů o schopnost sledovat více cílů. Jsou to algoritmy Global Nearest Neighbor (GNN) a Joint Probabilistic Data Association (JPDA). Na obrázku 3.4 se nachází ukázka situace přidělování objektů k trasám u algoritmů pro sledování více cílů. Nejprve jsou vypočteny pravděpodobnosti všech přiřazení a následně je vybrána kombinace s největším součtem pravděpodobností. Je nutné zajistit konzistenci řešení. To znamená, že jeden objekt nesmí být přidělen více trasám.

Rozdíl mezi GNN a JPDA je pouze ve výstupní informaci. GNN poskytuje jedno výsledné řešení. V případě uvedeném na obrázku 3.4 by to bylo přiřazení $(y4, T), (y3, T')$. JPDA vrací kompatibilní kombinace přiřazení a jejich pravděpodobnosti.

Multi hypothesis tracking

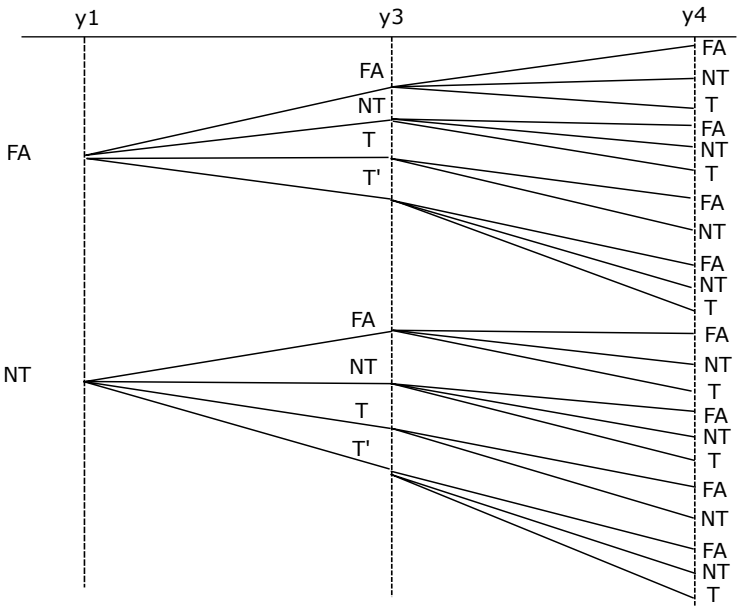
Tento algoritmus je rozšířením výše zmíněných algoritmů pro sledování více cílů. Narozdíl od nich si uchovává všechny kombinace přiřazení a jejich pravděpodobností a s dalšími měřeními je rozvíjí. Výstupem je tvrdé rozhodnutí [9][2].



Obrázek 3.4: Přiřazování objektů k trasám tras u algoritmů pro sledování více cílů. Trasy jsou označeny jako T, T', měření z detektoru jako y1-4.

Každá uchovaná hypotéza teoreticky obsahuje všechna dosavadní přiřazení detekovaných objektů do tří skupin:

- Chybná dektekce (False alarm - FA)
- Nová trasa (NT)
- Přiřazení ke stávajícímu cíli



Obrázek 3.5: Ukázka formování hypotéz pro situaci na obrázku 3.4. Pro zjednodušení je vypuštěno měření y2. Každý sloupec odpovídá jednomu měření. Každý průchod zleva doprava jedné hypotéze. Znaky v průsečíku hypotézy a měření znamenají způsob klasifikace daného měření v hypotéze. FA - falešný cíl, NT - nová trasa, T, T' - přiřazení měření do trasy T resp. T'.

Formování jednotlivých hypotéz je naznačeno na obrázku 3.5. Přiřazení odpovídají obrázku 3.4. Pro zjednodušení nebereme v potaz objekt $y2$. Každý průchod tímto grafem zleva doprava odpovídá jedné vygenerované hypotéze. Ke každé hypotéze je možné vypočítat její pravděpodobnost. Je to součet pravděpodobností přiřazení detekovaných objektů k cílům a pravděpodobností chybných detekcí. Hypotéza s největší pravděpodobností je poté označena za správnou. Z obrázku 3.5 je patrné, že s každým dalším krokem algoritmu roste počet hypotéz. V případě uvedeném na obrázku se jich v jednom kroku vygenerovalo 22.

Pro snížení složitosti algoritmu se aplikuje několik technik pro odstraňování nadbytečných hypotéz:

- Shlukování - Pokud se oblasti predikovaného výskytu objektu u více tras nepřekrývají, je možné zpracovávat je samostatně. Pro každou takovouto samostatnou oblast je možno vytvořit shluk (cluster), ve kterém probíhá algoritmus odděleně. Výsledná hypotéza se pak tvoří kombinací hypotéz jednotlivých clusterů. V případě obrázku 3.5 by byl vytvořen jeden cluster pro $y1$ a jeden pro $y3, y4$. Počet vytvořených hypotéz by potom byl $2 + 11$ oproti původním 22.

- Mazání nepravděpodobných hypotéz - Hypotézy s nízkou pravděpodobností je možné odstranit. To se dá udělat dvěma možnostmi. První je zvolit práh a hypotézy s nižší pravděpodobností budou mazány. Druhá možnost je zvolit si počet hypotéz, které chce mít uživatel uchovaných, a pokud jich bude více, budou smazány ty s nejnižší pravděpodobností.

- Mazání starších přiřazení - Některé hypotézy se mohou lišit až ve přiřazeních, které jsou několik měření (snímků) staré. Ty je možno sloučit dohromady a stará přiřazení smazat. Pravděpodobnost této sloučené hypotézy pak bude součtem pravděpodobností původních hypotéz.

Sledování za pomoci více hypotéz je výpočetně nejnáročnější z výše zmíněných algoritmů. Oproti GNN je však mnohem odolnější vůči chybám detektoru. Při správném použití redukcí hypotéz je možno tuto náročnost snížit a algoritmus využít v praxi.

Kapitola 4

Implementace

Tato kapitola popisuje způsob implementace jednotlivých částí aplikace `app_upc`, z nichž nejvýznamnějšími jsou výpočet disparitní mapy, detekce a sledování. Postup zpracování odpovídá blokovému diagramu 3.1, který vznikl při návrhu systému. Během implementace bylo nutno přihlédnout k několika omezením a podmínkám, které by mělo řešení splňovat. Největší omezující podmínkou byl výkon jednotky. Realtime zpracování obrazu je výpočetně náročný problém. Bylo třeba zvolit správné rozlišení vstupního obrazu a počet sejmutých snímků za sekundu tak, aby aplikace dávala co nejlepší výsledky a zároveň zpracovávala snímky v reálném čase.

Zachytávání snímku ve vyšším rozlišení by generovalo podrobnější disparitní mapu a to nejen díky vyššímu rozlišení disparitní mapy, ale také díky vyšší hloubkové přesnosti. Problémem je razantní nárůst času, za který by se zpracoval vstupní snímek. Při použití dvojnásobného rozlišení vstupních snímků se čas na zpracování jednoho snímku, při zachování stejných parametrů disparitní mapy, zvětší více než osminásobně.

Větší počet zpracovaných snímků za sekundu zase zlepšuje funkci sledovacího algoritmu, protože pohyb ve scéně je zachycen s menšími časovými odstupy. To pak umožňuje lépe sledovat rychle se pohybující cíle. Zvětšování počtu zachycených snímků za sekundu zvyšuje požadavky na výkon jednotky.

Při implementaci byl kladen důraz na výhody a nevýhody jednotky UCP-01. Jednotka má velkou operační paměť 1 GB. Při výběru algoritmů pro řešení problému byly častěji použity paměťově náročnější s nižší zátěží na výpočetní čas. Jednotka má také grafický čip s podporou OpenCL EP, což je využito k výpočtu disparitní mapy.

4.1 Použité technologie

Aplikace `app_upc` je implementovaná v programovacím jazyce C++. Části aplikace, které jsou zpracované na grafické kartě, jsou napsány v jazyce OpenCL C. Pro vytvoření struktury aplikace, síťovou komunikaci, grafická uživatelská rozhraní pro podpůrné aplikace, vícevláknové zpracování a přenositelnost mezi architekturami byla použita knihovna Qt verze 5.5.1. Pro metody na práci s obrazem a jeho zpracování byla použita knihovna OpenCV verze 3.0. Aplikace je spustitelná na jednotce UCP-01 a také na počítačích s operačními systémy Windows a Linux.

4.2 Zachytávání videa

Pro zachytávání vstupních snímků byla implementována třída `VideoCaptureDevice`, která je inspirovaná třídou `VideoCapture` z OpenCV [5]. Třída umožňuje zachytávání videa ze stereo-kamery na jednotce UCP-01 a také ze vstupního souboru.

Kamery jsou ovládány pomocí `ioctl` volání ovladačů, které mají aplikační rozhraní `Video4Linux2`. I samotné ovladače bylo nutné upravit pro použití dvou kamer jako stereo-kamery. Samotným výstupem z kamer je 16 bit obraz, kde prvních 8 bitů je hodnota pixelu z levé kamery a dalších 8 bitů je hodnota pixelu z pravé kamery. Třída je také zodpovědná za hlídání času sejmutí snímku tak, aby odpovídal požadovenému počtu snímků za sekundu.

Třída také umožňuje přehrát video z nahrávky, která byla pořízena dříve. Vše je implementováno tak, aby bylo rozhraní třídy pro vstup z kamery a ze souboru stejné. Možnosti vytvoření záznamu z jednotky jsou popsány v kapitole 4.10.

Pořízené snímky jsou ukládány do kruhového bufferu, který obsahuje frontu obrázků čekajících na další zpracování. Tím je zajištěno zpracování všech pořízených snímků i při nutnosti provést nějakou náročnou operaci, která by zpozdila zpracování snímků v reálném čase.

4.3 Rektifikace obrazu a výpočet disparitní mapy

Rektifikaci obrazu a následný výpočet disparitní mapy provádí třída `FrameProcWorker`, instance této třídy běží na jednotce UCP-01 v samostatném vlákně. Výpočet je částečně prováděn na grafickém čipu pomocí OpenCL EP, ten je naimplementován ve třídě `AIP`.

4.3.1 Ekvalizace histogramu

Před samotnou rektifikací je ještě provedena ekvalizace histogramu u obou snímků. Kamery na jednotce UCP-01 jsou schopny automaticky nastavovat čas expozice. Kamery ale nesnímají scénu z jedné pozice. Při nevhodných světelných podmínkách, například při ostrém světle nebo u málo osvětlené scény, dochází k případům, kdy jsou snímky různě exponované. I při zobrazení takových snímků je tento jev patrný.

Obrázek 4.1: Výstup ekvalizace histogramu. Nalevo vstupní snímek, napravo ekvalizovaný.

Jedním z řešení by bylo, aby expozici řídila pouze jedna kamera. Tato funkce ale není podporována. Další možností je provádět řízení expozice programově. Tuto možnost kamery

nabízí, ale bylo by nutné vytvořit algoritmus, který by expozici řídil. Také by došlo ke zvýšení času zpracování jednoho snímku.

Řešením tohoto problému v popisované metodě je provedení ekvalizace histogramu. Bylo experimentálně zjištěno, že disparitní mapa, která je vypočtena z ekvalizovaných snímků je kvalitnější, než bez ekvalizace. Ekvalizace způsobí rovnoměrnější rozložení hodnot pixelů v obraze. Na obrázku 4.1 je vidět obraz před a po ekvalizaci. Nevýraznější změnou je zvýšení kontrastu v obraze.

Pro ekvalizaci je nejprve nutné vypočítat histogram. Histogram obsahuje informaci o distribuci hodnot v obraze. Pro každou intenzitu, která může být obsažena v obraze, je uložen počet pixelů, které tuto intenzitu mají. Následně je z histogramu vypočtena tzv. kumulativní distribuční funkce (CDF). Pro histogram $H(i)$ se CDF $H'(i)$ spočítá jako:

$$H'(i) = \sum_{0 \leq j < i} H(j) \tag{4.1}$$

Následuje výpočet ekvalizovaného obrazu pomocí vzorce:

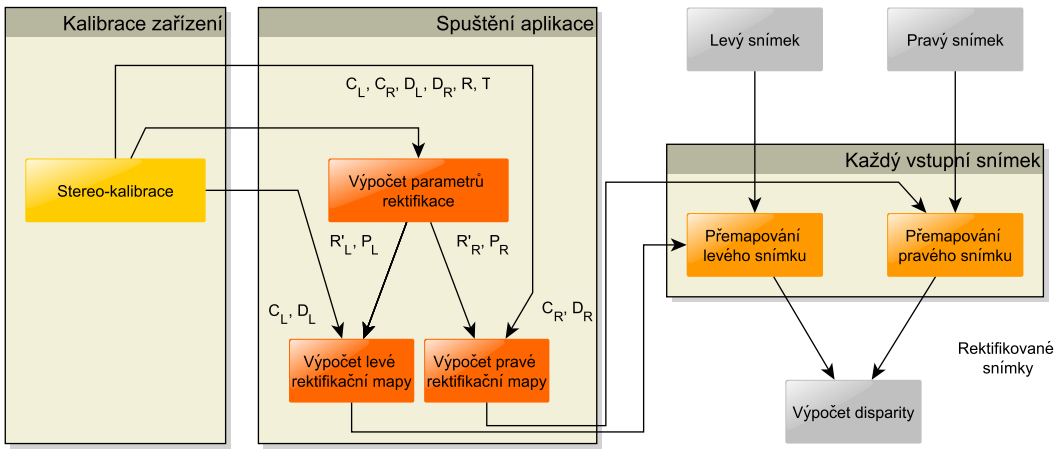
$$eq(x, y) = H'\left(src(x, y)\right), \tag{4.2}$$

kde eq je ekvalizovaný obrázek a src je vstupní obraz [7].

Kvůli urychlení výpočtu histogramu je procházena jen čtvrtina vstupního obrazu. Tento řídký histogram je však podobný histogramu z celého vstupního obrázku. Kvůli absenci atomických operací v omezené verzi OpenCL na jednotce UCP-01 je nutné vypočítat CDF na procesoru. Výpočet ekvalizovaného obrazu už probíhá na grafickém čipu pro každý bod výstupního obrazu paralelně.

4.3.2 Rektifikace snímků

Postup rektifikace snímků je ilustrován obrázkem 4.2. Jednotlivé kroky jsou rozdělené do skupin, podle toho, kdy se provádí. Implementace se nachází ve třídě `StereoRectifier` a je napsána podle knihy [4]. Nejprve se provede stereo-kalibrace systému, která je popsána v kapitole 4.7. Kalibrace se na každé jednotce provádí pouze jednou a její výsledky jsou uloženy v konfiguračním souboru aplikace.

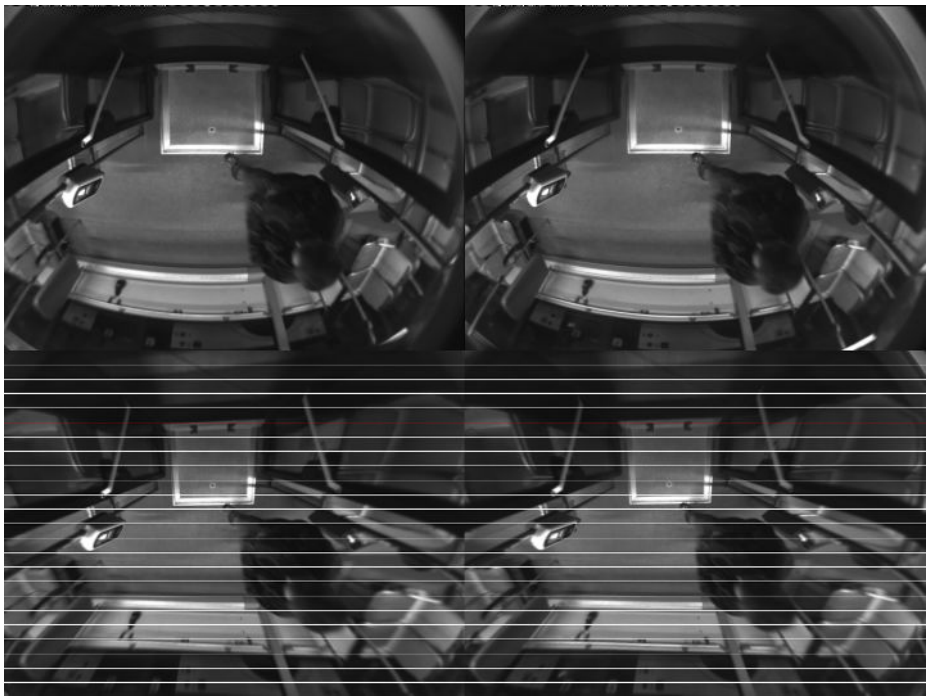


Obrázek 4.2: Diagram jednotlivých kroků rektifikace rozdělených podle četnosti provádění.

Při každem spuštění aplikace se provede výpočet parametrů rektifikace pomocí funkce `stereoRectify` z OpenCV. Vstupem výpočtu jsou data získaná při kalibraci systému. Jsou to kalibrační matice kamer C_L, C_R a jejich distorzní koeficienty D_L, D_R, rotační matice R a translační vektor mezi koordináty první a druhé kamery T. Výstupem jsou rotační matice pro rektifikaci kamer R'_L, R'_R a také nové projekční matice pro rektifikovaný souřadný systém kamery P_L, P_R.

Následuje výpočet mapování vstupních snímků z kamery na snímky rektifikované a zároveň zbavené distorze. Výstupem jsou mapy, kde je pro každý výstupní pixel uložena souřadnice bodu v původním obraze.

Poslední částí je samotná rektifikace vstupních snímků. Ta je prováděna paralelně na grafickém čipu. Za pomocí mapování vypočtených při spuštění aplikace je provedena bilineární interpolace vstupních snímků na snímky rektifikované.



Obrázek 4.3: Výstup výpočtu rektifikace. Nahoře vstupní snímky, dole snímky rektifikované.

Na obrázku 4.3 je ukázka výstupu z rektifikace. Došlo k zarovnání řádků a také k potlačení zkreslení objektivů. Korekce zkreslení je nejlépe vidět při porovnání rovných linií mezi obrazy před a po rektifikaci.

4.3.3 Disparitní mapa

Výpočet disparitní mapy je celý proveden na grafickém čipu. Jeho implementace se nachází v třídě `StereoBM`. Řešení je inspirováno [5] s využitím integrálního obrazu z článku [8].

Nejprve je provedena filtrace obrazu pomocí Sobelova operátoru. Filtrace se provádí pouze v ose X, protože na rektifikovaných snímcích probíhá hledání disparity pouze v jednom řádku. Tento filtr počítá derivaci vstupního obrazu (detekci hran) v horizontálním směru, a tak se snižuje vliv rozdílného osvětlení snímků [13].

Samotný algoritmus Block Matching je pozměněn pro použití na grafickém čipu. Parametry pro výpočet disparitní mapy jsou `min_disparity`, což je hodnota minimální disparity

od které se má začít hledat, a `num_of_disparities`, který udává kolik hodnot disparity je prohledáváno. Parametry byly nastaveny takto $min\_disparity = 0, num\_of\_disparities = 16$, to znamená že algoritmus bude vyhledávat disparitu v rozmezí $\langle 0, 15 \rangle$.

Prvním krokem je výpočet diferenčních obrazů. Pro každou hodnotu disparity d se spočítá obraz absolutních diferencí $diff_d$ tak, že se hodnoty v pravém obrazu posunou o hodnotu disparity a následně se pro každý pixel spočítá hodnota diference podle vzorce:

$$diff_d(x, y) = |I_L(x, y) - I_R(x + d, y)| \tag{4.3}$$

Pokud tedy hledáme disparitu o hodnotě 0 - 15 na rozlišení 160 x 120, Bude vytvořeno 16 obrazů absolutních diferencí s velikostí 160 x 120. Výpočet je spuštěn paralelně pro každý pixel diferenciálních obrazů.

Následuje převedení obrazů absolutních diferencí na integrální obrazy. Integrální obraz byl definován podle vzorce 3.2. Hodnota pixelu je dána jako suma součtu pixelů v řádku nalevo, součtu pixelů ve sloupci nad a vlastní hodoty bodu.

Na GPU je výpočet integrálního obrazu implementován více paralelními průchody obrazu. Nejprve se vypočte částečný integrální obraz, výpočet bude spuštěn paralelně pro každý řádek. Hodnota pixelu bude rovna součtu hodnot pixelů nalevo a sebe samého. Výsledkem bude obrázek definovaný takto:

$$ii(x, y) = \sum_{x' \leq x} i(x', y) \tag{4.4}$$

Následně se výpočet spustí paralelně pro všechny sloupce. Hodnota pixelu bude rovna součtu hodnot pixelů se sloupci nad ním a hodnotě sebe samého. Výsledkem je tedy integrální obraz odpovídající vzorci 3.2.

Postup výpočtu disparity byl naznačen v 3.1.2. Vzhledem k použití předpočítaných integrálních obrazů absolutních diferencí bylo nutno upravit algoritmus výpočtu disparity. Největší změnou je úprava výpočtu SAD metriky. Ta se nyní počítá pomocí vzorce:

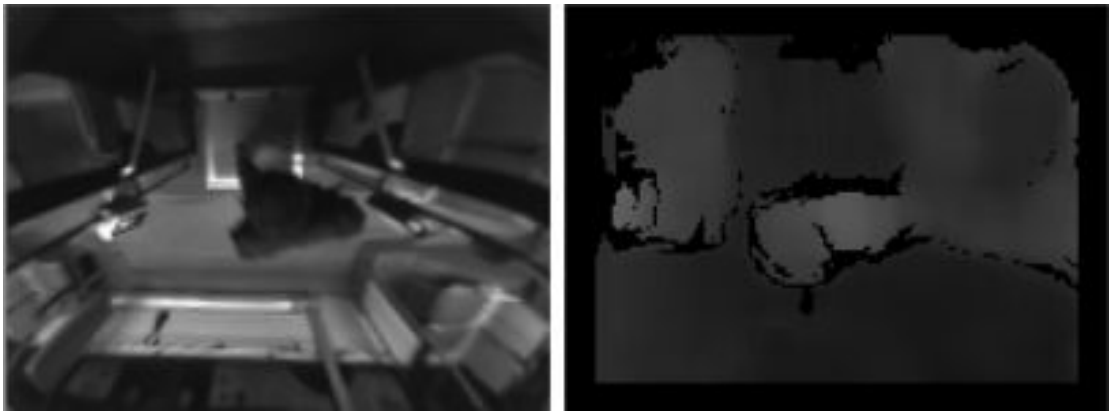
$$SAD_d(x, y) = diff_d(x-k, y-k)+diff_d(x+k, y+k)-diff_d(x+k, y-k)-diff_d(x-k, y+k), \tag{4.5}$$

kde x, y jsou souřadnice bodu, pro který se hledá jeho disparita, d je hodnota disparity u které zjišťujeme její SAD hodnocení, $diff_d$ je integrální obraz absolutních diferencí, který vznikl při posunutí pravého obrazu o d pixelů a k je velikost porovnávaného okolí.

Při vyhledávání korespondencí se pak prochází všechny integrální obrazy absolutních diferencí, kde se počítá SAD. Hodnota disparity s nejnižším SAD je výsledná hodnota. Na obrázku 4.4 je ukázka touto metodou vypočtené disparitní mapy.

Tento přístup je náročný na paměť. Pro vyšší hodnoty maximální disparity nebo vyšší rozlišení mohou být paměťové nároky neúnosné. Dochází ale k úspoře procesorového času a to několikanásobné. Absolutní diference dvou pixelů se v původním algoritmu počítá tolikrát, kolik pixelů je v porovnávaném okolí bodu. Pokud je tedy okolí 1, jeden a ten stejný výpočet je proveden devětkrát. Pokud je okolí velikosti 3, jeden výpočet je proveden devětačtyřicetkrát. V metodě používající integrální obraz je tento výpočet proveden vždy jednou.

Další výhodou je urychlení součtu těchto diferencí. V původním algoritmu se sčítaly vypočtené diference z celého okolí bodu, to znamená, že bylo nutné sečíst tolik čísel kolik



Obrázek 4.4: Ukázka výsledku výpočtu disparity. Vlevo rektifikovaný snímek, vpravo vypočtená disparitní mapa.

pixelů je v porovnávané oblasti. V metodě využívající integrální obraz jde vždy o součet čtyř čísel viz rovnice 4.5. Lze tedy říct, že čas výpočtu metody využívající integrální obraz se nemění v závislosti na změně velikosti porovnávaného okolí bodu. Efektivita tohoto přístupu roste se zvětšující se velikostí porovnávaného okolí.

Výpočet disparity je ovlivněn dvěma parametry. Prvním je `SAD_window_size`, což představuje velikost provnávacího okna pro sumu absolutních diferencí. U řešení použitém v této práci parametr neovlivňuje čas pro výpočet disparitní mapy. Zvětšování tohoto parametru zvyšuje odolnost vůči šumu na vstupních snímcích. Dochází ale také k odstranění ostrých přechodů na výsledné disparitní mapě (rozmazání). Dalším parametrem je `uniques_ratio`. Tím se ovlivňuje, jak moc uníkátní musí výsledné řešení být oproti ostatním. Pokud je nejlepší hodnota SAD jen nepatrně lepší než u druhé nejlepší nalezené disparity, je řešení nejednoznačné a výsledná hodnota disparity je označena jako nevypočtena.

Nejnižší hodnota SAD se většinou nenachází na celé hodnotě disparity, ale někde mezi nejlepší a druhou nejlepší hodnotou. Pro zpřesnění výsledné disparity je možné spočítat její subpixelovou hodnotu. Řešení je inspirováno článkem [10] a implementací v OpenCV. Jakmile je nalezena nejlepší hodnota disparity, uloží se také hodnoty SAD u sousedních disparit. Z těchto tří hodnot se pak interpoluje výsledná hodnota disparity podle vzorce:

$$d_{sub} = d - \frac{SAD_{d+1} - SAD_{d-1}}{SAD_{d+1} + SAD_{d-1} - 2 * SAD_d + |SAD_{d+1} - SAD_{d-1}|}, \tag{4.6}$$

kde d je nejlepší nalezená hodnota disparity (nejmenší SAD), SAD_x je hodnota sumy absolutních diferencí pro disparitu x.

Díky výpočtu subpixelové disparity je možné zvýšit hloubkové rozlišení disparitní mapy i při nízkém rozlišení vstupního obrazu. Jedná se však o interpolaci a její výsledky nebudou tak dobré jako při reálném zvýšení rozlišení vstupního obrazu.

4.4 Extrakce popředí

V kapitole 3.1.3 byl popsán způsob extrakce popředí z disparitní mapy. Model pozadí je v aplikaci implementován ve třídě `BGModel`. Při výpočtu popředí dochází k odstranění oblastí disparitní mapy, kde je hodnota disparity na modelu pozadí i na disparitní mapě stejná

nebo nižší. V závislosti na osvětlení interiéru vozidla se lehce mění i vypočtená disparitní mapa. Řešením je přidat podmínku, že vypočtená disparita musí být větší než disparitní mapa v modelu pozadí o nějaký malý práh, aby byla klasifikována jako popředí. Zmíněný práh je parametrem modelu pozadí s názvem `disparity_tolerance_for_BG`.

Obrázek 4.5: Výpočet popředí. Vlevo model pozadí, uprostřed disparitní mapa, vpravo vypočtené popředí.

4.5 Detekce cestujících

Detektor je implementován v metodě `measurements` třídy `Counter`. Detekce se provádí opakovaným hledáním maximalních hodnot v disparitní mapě s popředím. Pro každé nalezené maximum se následně provede ověření, zda se jedná o cestujícího.

Na základě hodnoty nalezeného maxima (předpokládaná výška cestujícího) je vypočtena očekávaná oblast, ve které by se cestující měl nacházet. Její velikost je závislá na hodnotě disparity nalezeného maxima. Vyšší cestující totiž zaberou na disparitní mapě větší prostor. Pokud se v prohledávané oblasti nachází dostatečný počet bodů vyhodnocených jako popředí, je nalezené maximum klasifikováno jako cestující. Klasifikační práh pro detekci cestujícího má název `min_disp_non_zero_ratio`. Nakonec je z popředí odstraněna oblast s cestujícím a algoritmus pokračuje opětovným hledáním maxima na zbytku disparitní mapy kvůli hledání dalších cestujících.

Parametr `min_disp_non_zero_ratio` má nezanedbatelný vliv na funkci detektoru. Při jeho snížení dochází k detekci šumu nebo malých objektů jako cestujících. Naopak při jeho zvýšení se nedetekují opravdoví cestující zvláště při snížené kvalitě vstupní disparitní mapy (špatné osvětlení scény nebo špatná kalibrace stereo-kamer).

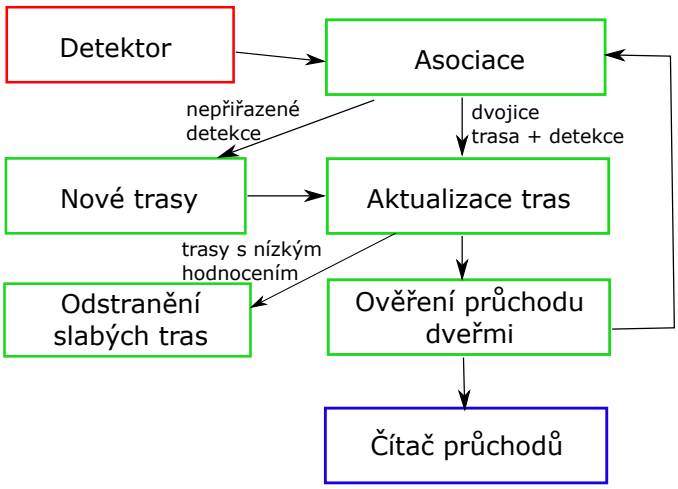
Velikost oblasti, ve které je očekáván cestující, je možné ovlivnit parametrem `head_safety_coeff`. Zvětšování tohoto parametru zvětšuje oblast, kde je očekáván výskyt cestujícího. Pokud by byl parametr moc velký, docházelo by k detekci a klasifikaci dvou cestujících blízko u sebe jako jednoho. Pokud by byl parametr moc malý, docházelo by k detekci jednoho cestujícího vícekrát, protože by z disparitní mapy nebyl odstraněn celý.

4.6 Sledování cestujících

4.6.1 Global Nearest Neighbor

Pro sledování cestujících byl původně implementován algoritmus Global nearest neighbor popsaný v kapitole 3.2.4. Sledovací algoritmus je implementován ve třídě `Counter`. Vstu-

pem jsou pozice detekovaných cestujících. Jeden detekovaný cestující je instancí třídy `Measurement`. Funkce sledovacího algoritmu je naznačena na obrázku 4.6.



Obrázek 4.6: Diagram funkce trackovacího algoritmu.

Vzhledem k rozličnému způsobu pohybu cestujících ve dveřním prostoru nebylo možné použít predikci výskytu cestujícího založenou na znalosti jeho předchozí polohy. Někteří cestující dveřním prostorem jen projdou a vyskytují se jen na několika snímcích, jiní stojí ve dveřním prostoru dlouhou dobu a pak rychle odejdou. Detektor také nezaručuje, že detekovaný bod na cestujícím bude pořád stejný (například střed jeho hlavy). I když cestující stojí na místě, jeho detekovaná poloha se neustále mění.

Nejprve se provede přiřazení měření (detekovaných bodů) do existujících tras. Vypočítá se vzdálenost mezi měřeními a posledními pozicemi na kterých se vyskytovaly trasy. Poté jsou nejbližší body přiřazeny jednotlivým trasám (sledovaným cestujícím), pokud je vzdálenost menší, než je parametr `gate_radius`. Tento parametr by měl udávat, jakou maximální vzdálenost může urazit cestující mezi dvěma snímky. Pokud by byl moc malý, nedařilo by se sledovat cestující, kteří se rychle pohybují. Pokud by byl moc velký, mohlo by docházet ke špatnému přiřazování meření do tras především u okrajů dveřního prostoru.

Informace o jedné trase je uložena ve třídě `Track`. Jsou zde uloženy všechny body, kde byl cestující detekován, pravděpodobnost trasy a další informace potřebné pro sledovací algoritmus. Jedno přiřazení mezi detekcí a trasou je instancí třídy `Association`.

Následuje aktualizace tras pomocí metody `update`. U každé trasy je aktualizováno její hodnocení. S každým úspěšným přiřazením detekce do trasy se její hodnocení zvyšuje dokud nedosáhne stanoveného stropu. Při dalších úspešných detekcích se hodnocení udržuje na stanovením maximu. Aktualizaci hodnocení trasy je v případě úspěšného přiřazení vyjádřit pomocí vzorce:

$$Li_n = Li_{n-1} + log\left(\frac{P_d}{P_{fa}}\right), \tag{4.7}$$

kde Li je hodnocení trasy, t je krok algoritmu, P_d je pravděpodobnost správné detekce a P_{fa} je pravděpodobnost falešné detekce.

V případě že do trasy nebyl přiřazen detekovaný bod, její hodnocení se snižuje. Úpravu hodnocení takové trasy je možno vyjádřit jako:

$$Li_n = Li_{n-1} + log\left(1 - P_d\right) \tag{4.8}$$

Po výpočtu hodnocení tras dochází k odstranění těch, které mají nízké hodnocení. Parametry použité při výpočtu hodnocení a práh pro odstranění trasy jsou nastaveny tak, aby trasa zůstala v systému, i když k ní nebude několikrát přiřazena detekce. To zohledňuje případné chyby, které by mohl generovat detektor.

Dalším krokem úpravy tras je přidání nových (prozatimních) tras pro meření, které nebyly asociovány s nějakou stávající trasou. Jejich hodnocení je inicializováno na nulu. Pokud v příští detekci bude nalezeno přiřazení do této trasy, zůstane trasa v systému. V opačném případě bude trasa smazána.

Posledním krokem sledovacího algoritmu je ověření, zda některý sledovaný cestující nesplnil podmínky pro započítání jeho průchodu přes dveřní prostor, to je implemenováno v metodě `checkConfirmedTracks`. Detekční prostor je rozdělen na tři části: vnější, střední a vnitřní. Aby byl průchod cestujícího započítán musí první bod jeho trasy ležet ve vnitřní oblasti a poslední bod jeho trasy ležet ve vnější oblasti, nebo naopak. Zároveň také musí urazit určitou vzdálenost ve směru z nebo do vozidla. Tato podmínka pomáhá odstranit chyby v počítání způsobené cestujícími, kteří se dlouhou dobu zdržují a pohybují ve dveřním prostoru. Parametr, který nastavuje tuto vzdálenost se nazývá `trajectory_length_ratio`.

4.6.2 Multiple Hypothesis Tracking

Vzhledem k nedostatečnému výkonu jednotky UCP-01 nebylo možné implementovat metodu Multiple Hypothesis Tracking, která je popsána v kapitole 3.2.4, takovým způsobem, jaký je popsaný například v [2]. Již při použití výše popsaného algoritmu Global Nearest Neighbor se jednotka dostává na hranici svého výkonu. Byl však implementován sledovací algoritmus, který tuto metodu aproximuje.

Palubní počítač posílá jednotce informace o stavu dveří. Tak je možné rozdělit funkci jednotky na část, kdy jsou dveře otevřené a zavřené. Ve stavu otevřených dveří si aplikace pouze zaznamenává pozice detekovaných cestujících. Během doby, kdy jsou dveře otevřené, se neprovádí žádné sledování ani počítání cestujících.

Po zavření dveří se spustí samotné sledování cestujících na zaznamenaných datech. Algoritmus postupně generuje jednotlivé hypotézy, které se následně hodnotí. Nejlepší hypotéza je pak použita jako správná. Po nalezení nejlepší hypotézy jsou pak palubnímu počítači zaslána data o počtu průchodů dveřmi.

Narozdíl od řešení v kapitole 3.2.4 nejsou hypotézy generovány paralelně a nejsou v systému drženy zároveň. Generování a vyhodnocení hypotéz se provádí postupně, ta nejlepší je vždy uchovávána. Také nedochází k žádnému mazání hypotéz, které mají během generování slabé hodnocení.

Generování hypotéz je prováděno metodou Random walks. Veliká část algoritmu je stejná jako u GNN jedinný rozdíl je u přiřazování detekovaných bodů do tras. V GNN byl do trasy přiřazen vždy nejbližší detekovaný bod, který je zároveň v nějaké maximální vzdálenosti. U tohoto přístupu je bod přiřazen náhodně.

Při dostatečném počtu vygenerovaných hypotéz je pokryta velká část všech možných hypotéz, a algoritmus tak může poskytovat velice podobné výsledky jako implementace MHT popsaná v [2]. Výpočet probíhá iterativně a během něj se postupně zlepšuje nalezená hypotéza. Tak je možné ukončit tento výpočet dříve (například v případě opětovného otevření dveří) a klasifikovat dosud nejlepší nalezenou hypotézu za správnou.

Aby bylo možné vybrat nejlepší hypotézu, je nutné vytvořit hodnotící funkci, která bude co nejlépe odpovídat reálnému stavu ve scéně. Byly implementovány dva přístupy,

prvním je hodnocení hypotéz podle počtu vygenerovaných průchodů přes dveřní prostor. Čím větší byl počet cestujících, kteří podle hypotézy prošli dveřním prostorem, tím bylo lepší hodnocení hypotézy.

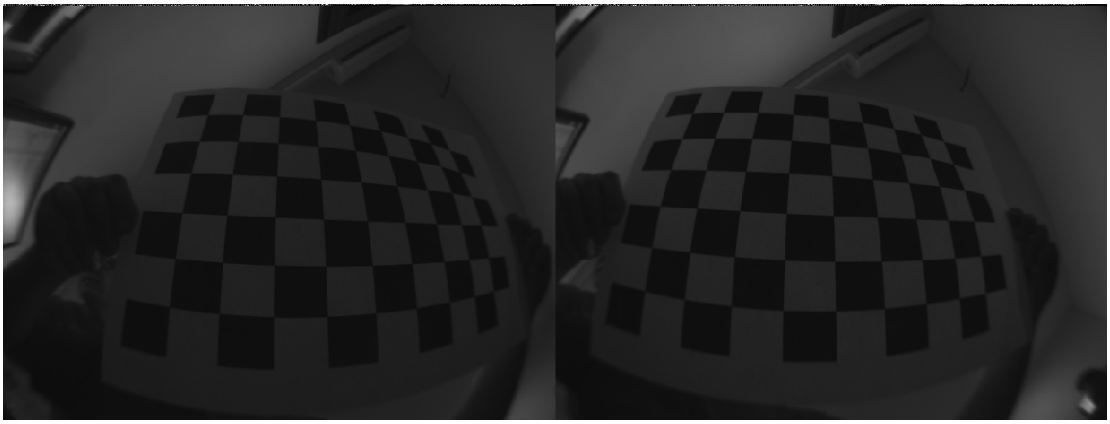
Druhá hodnotící funkce upřednostňovala hypotézy, kde průměrná vzdálenost mezi jednotlivými body tras byla nejnižší. Vliv hodnotící funkce na výsledky sledovacího algoritmu a jejich provnání jsou podrobně popsány v kapitole 5.

4.7 Kalibrace zařízení a jeho instalace do vozidla

Pro kalibraci a instalaci jednotky byla implementována aplikace `app_config`. Příprava jednotky UCP-01 k provozu sestává ze dvou částí.

Nejprve je potřeba provést kalibraci stereo-kamery. Stereo-kalibrace je postup, při kterém se zjišťují parametry kamer a také jejich vzájemná poloha v systému. Výstupem jsou rotační matice a translační vektor mezi souřadnými systémy kamer, vnitřní matice kamer a jejich distorzní koeficienty. Tyto parametry jsou nutné pro výpočet rektifikace snímků popsaný v kapitole 4.3.2. Implementace stereo-kalibrace byla provedena podle [4] použitím knihovny OpenCV.

Nejprve je několikrát nasnímán nějaký známý planární vzor (v našem případě šachovnice 9 x 6 polí) tak, aby byl celý viditelný na obou kamerách. Šachovnice je na snímcích detekována a jsou tak známy korespondující body na obou snímcích. Z polohy těchto korespondencí jsou vypočteny parametry stereo-kalibrace. Ukázku z kalibrace je možné vidět na obrázku 4.7. Pro správnou kalibraci je vhodné nasnímat šachovnici z různých úhlů a pohybovat s ní tak, aby byla nasnímána na všech místech vstupního obrazu.



Obrázek 4.7: Snímek pořízený při kalibraci stereo-kamery.

Druhou částí je kalibrace jednotky během instalace do vozidla. Tento proces je velmi důležitý pro správné fungování aplikace. Nejprve je v konfigurační aplikaci provedena volba typu vozidla. Každý typ vozidla má přednastavenou sadu parametrů pro správné fungování aplikace. Jednotlivé typy vozů se mezi sebou liší umístěním jednotek ve dveřním prostoru, rozdílnou výškou nad podlahou a také různou velikostí dveřního prostoru.

Následně je jednotka nasměrována tak, aby hrana dveří byla zarovnána s 25. řádkem obrazu. Obraz správně zarovnané jednotky je vidět na obrázku 4.3. Jednotka je naklopena mírně do vozidla aby vstupní obraz tolik neovlivňovalo světlo z vnějšku vozidla. Dalším krokem je uložení modelu pozadí. Ve dvou krocích jsou uloženy disparitní mapy scény postupně s otevřenými a zavřenými dveřmi. Model pozadí je třeba sejmout u otevřených a

zavřených dveří z toho důvodu, že u některých vozidel se dveře otevírají dovnitř vozu. Při otevření dveří by tak ve scéně přibyly další objekty, které by byly detekovány jako cestující. Během zachycení modelu pozadí jsou na dveře nasazeny pruhy kartonu, které zakrývají sklo ve dveřích, které je průhledné a disparita se u něj normálně nevypočítá. Výsledný model pozadí je pak vypočten jako maximum z modelu pozadí při otevřených a zavřených dveřích. Během instalačního procesu jsou také zaznamenány snímky scény, které jsou pak použity pro detekci posunutí jednotky. Ta je popsána v následující kapitole.

Posledním krokem procesu instalace jednotky do vozidla je ověření správnosti počítání. Obsluhující personál několikrát projde dveřmi a následně zjišťuje, jestli byly průchody správně započítány.

Aplikace `app_config` zaznamená během instalačního procesu všechna vstupní data. Pokud se následně stane, že jednotka nepočítá cestující správně, je možné si správnost kalibrace zkontrolovat dodatečně.

Během připojení simuluje konfigurační aplikace veškerou funkcionalitu jednotky UCP-01. Je možné zkontrolovat rektifikovaný obraz, vypočtenou disparitní mapu, model pozadí, sledovací algoritmus a také detekovat posunutí jednotky, které je popsáno v následujícím textu.

4.8 Detekce posunutí jednotky

Vzhledem k umístění a konstrukci krabičky, ve které je jednotka UCP-01 umístěna, se může stát, že se jednotka pohne. Může to být zaviněno vibracemi ve voze nebo také zapříčiněním zvědavého cestujícího. Pokud se s jednotkou pohne dostatečně, zhorší se výsledky sledovacího algoritmu. Model pozadí po posunutí nebude odpovídat scéně, kterou jednotka sleduje. Pro zjištění této situace byla ve třídě `AlignmentChecker` implementována detekce posunutí jednotky.

Během instalace jednotky do vozidla jsou pořízeny snímky prázdné scény. Ty jsou uloženy v konfiguračním souboru aplikace a palubní počítač může na vyžádání v jednotce vyvolat kontrolu posunutí jednotky.

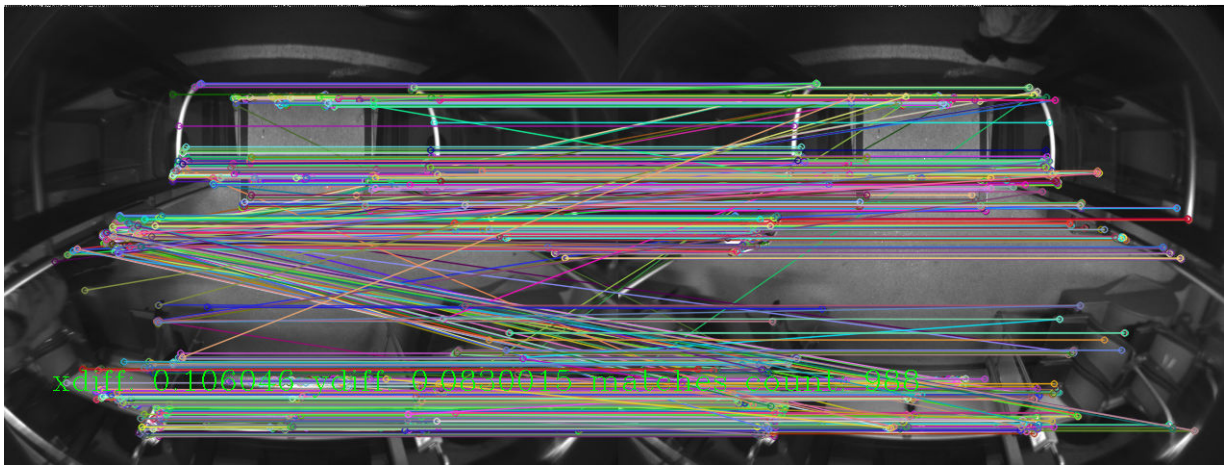
Bylo nutné vytvořit algoritmus, který je nezávislý na osvětlení scény, takže nebylo možné pouze porovnat dva obrazy. Ověření se provádí pomocí porovnání význačných bodů v obraze uloženém během instalace a aktuální scénou.

Nejprve je provedena detekce význačných bodů na obou obrazech a výpočet jejich ORB (Oriented FAST and rotated BRIEF) deskritptorů. ORB byl použit hlavně proto, že narozdíl od SIFT nebo SURF deskriptorů není jeho komerční využití zpoplatněno. Tyto deskriptory dokážou popsat lokální oblast kolem nějakého význačného bodu jsou invariantní vůči transformaci a také změně světelných podmínek. Následně se vyhledají korespondující body na obou snímcích.

Z nalezených korespondencí je možné vypočítat transformační matici mezi obrazovými rovinami obou snímků. Výpočet transformace se provádí metodou RANSAC implementovanou v OpenCV. Pomocí transformační matice je vypočten posun krajních bodů. Pokud je tento posun větší než stanovený práh, jednotka zahlásí chybu.

Na obrázku je zobrazena vizualizace výpočtu detekce posunutí jednotky. Je patrné, že některé korespondence nebyly určeny správně, ale je jich výrazná menšina. Algoritmus RANSAC je považuje na tzv. outliers a tak se nezhorší výpočet transformace.

Výpočet deskriptorů, hledání korespondujících bodů a výpočet trasnsformace mezi obrazovými rovinami byl implementován pomocí knihovny OpenCV. Kontrola posunutí jed-



Obrázek 4.8: Vizualizace detekce posunutí. Na obrázku jsou zobrazeny nalezené korespondence ORB deskritporů. A vypočtená hodnota posunu v pixelech. Vlevo je obrázek pořízený při kalibraci, vpravo snímek pořízený při kontrole.

notky je výpočetně náročný algoritmus, na jednotce UCP-01 trvá spuštění přes 20 sekund. Není možné jej provádět periodicky během spuštěného sledování cestujících. Je také nutné zajistit prázdnost dveřního prostoru. Proto je možné kontrolu vyvolat pouze zadáním požadavku do palubního počítače. Během výpočtu kontroly posunutí se také neprovádí detekce a sledování cestujících, pořízené snímky jsou zahozeny.

4.9 Síťová komunikace

Jednotky UCP-01 jsou do systému připojeny přes lokální síť vytvořenou ve voze. Jednotka primárně komunikuje s palubním počítačem přes textový protokol.

Pro konfiguraci a instalaci jednotky do vozidla byl vytvořen komunikační protokol konfigurační aplikace `app_config`, která je popsána v kapitole 4.7. Pro hromadné nahrávání záznamů byl vytvořen komunikační protokol pro komunikaci s aplikací `app_recorder`, která je popsána v kapitole 4.10.

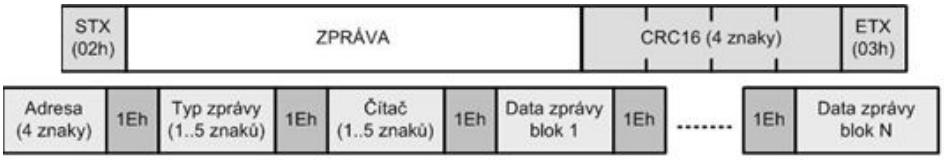
Samotná jednotka se vždy chová jako server a ostatní aplikace se k ní připojují. Jednotlivé jednotky ve voze jsou rozlišeny koncovým číslem jejich ip adresy.

4.9.1 Komunikace s palubním počítačem

Komunikace mezi palubním počítačem a APC je realizována v síti využívajících rodiny protokolů TCP/IP. Jako protokol transportní vrstvy je použit bez stavový, nespojovaný, nespolehlivý protokol UDP (User Datagram Protocol), který v případě ztráty paketu nezajišťuje spolehlivost opakovaným odesláním. Předpokládá se, že palubní počítač bude současně komunikovat s několika jednotkami UCP.

Implementace komunikace s palubním počítačem se nachází ve třídě `EPISServer`. Pro komunikaci byla využita třída `QUDPSocket` z knihovny QT. Komunikace je vždy inicializována palubním počítačem, který odesílá dotazy a jednotka na ně odpovídá.

Na obrázku 4.9 je naznačena struktura rámce, pomocí kterého jsou přenášeny zprávy v komunikačním protokolu. Z něj je vidět, že je rámec ohraničen speciálními znaky STX



Obrázek 4.9: Struktura přenášeného rámce (nahoře) a struktura samotné přenášené zprávy (dole).

(start of text) a ETX (end of text). Za zprávou se ještě nachází kontrolní součet pro ověření správného přenosu rámce.

Samotná zpráva se skládá z bloků, které jsou od sebe odděleny dalším speciálním znakem RS (record separator). První tři bloky zprávy jsou vždy povinné. V prvním bloku je uložena adresa. Ta už není používána a zůstala v protokolu pouze z důvodů udržení zpětné kompatibility. Další blok obsahuje identifikátor služby, o kterou palubní počítač žádá. Posledním povinným blokem je čítač zpráv. Čítač slouží pro správné párování příkazů a odpovědí. Je inkrementován s každou novou zprávou dané služby. V potvrzení se zopakuje číslo čítače ze žádosti/příkazu.

Zde je seznam důležitých služeb implementovaných v komunikačním protokolu mezi palubním počítačem a jednotkou:

Start/stop počítání Příkazy pro započetí a ukončení počítání. Tyto příkazy jsou odeslány ve chvíli kdy se otevřou nebo zavřou dveře.

Reset Služba pro smazání počítadel. Tento příkaz se jednotce posílá na konci každé trasy, kterou vůz absolvuje.

Stav Žádost palubního počítače pro získání informací o stavu jednotky. V odpovědi se nachází počet cestujících, kteří prošli dveřním prostorem a stav počítání jednotky.

Update Příkaz pro aktualizaci počítací aplikace. Po přijetí tohoto příkazu je z palubního počítače stažena nová verze a následně je na jednotku instalována.

Config Pomocí této služby nastaví palubní počítač na jednotce číslo vozidla, správné datum a čas. Datum a čas se pak používají při debug výpisech pro jednodušší odstraňování chyb v aplikaci.

Kontrola posunutí Příkaz na vyvolání kontroly posunutí jednotky. Postup této kontroly je popsán v kapitole 4.8.

Přenos obrazu Palubní počítač si může na vyžádání nechat zaslat snímek z kamery jednotky. Jednotky by potom mohly sloužit také jako bezpečnostní kamery snímající dveřní prostor. Jelikož je však protokol textový je nutné binární obrazová data převést na textovou reprezentaci. Provádí se tedy převod do kódování Base64, které je popsáno v [12]. Převod a odesílání zaznamenaných snímků také zabere určitý procesorový čas. Na kontinuální odesílání vstupních snímků jednotka nemá dostatečný výkon. Po určité době by se pořízené snímky začaly zahazovat a zhoršila by se úspěšnost počítání. Proto si palubní počítač může vyžádat maximálně jeden snímek scény za sekundu.

4.9.2 Komunikace s konfigurační aplikací

Komunikace s konfigurační aplikací probíhá přes protokol TCP. Její implementace se nachází ve třídě `CFGServer`. Pro komunikaci byla využita třída `QTCPSocket` z knihovny QT. Samotný protokol je velice jednoduchý a poskytuje pouze tři služby.

První službou je přenos konfiguračního souboru z jednotky UCP-01 do konfigurační aplikace. Další službou je přenos upraveného konfiguračního souboru z konfigurační aplikace do jednotky. Poslední službou implementovanou v tomto komunikačním protokolu je přenos pořízeného obrazu z jednotky do konfigurační aplikace.

Po připojení konfigurační aplikace k jednotce se nejprve přenese konfigurační soubor a následně jsou do konfigurační aplikace kontinuálně zasílány pořízené snímky. Pokud konfigurační aplikace změní nastavení jednotky, je upravený konfigurační soubor ihned zaslán do jednotky. Během kalibrace jednotky a její instalace do vozidla jsou všechny výpočty a změny prováděny v konfigurační aplikaci a po ukončení kalibrace/instalace jsou změny přeneseny do jednotky.

4.9.3 Komunikace s nahrávací aplikací

Pro účely nahrávání záznamů z jednotek byla implementována aplikace `app_recorder`, která je popsána v kapitole 4.10. Její komunikační protokol je pozměněnou verzí komunikace s konfigurační aplikací. Chybí zde služby pro přenos konfiguračního souboru. Navíc jsou implementovány zprávy o započetí a ukončení počítání, které jednotce přicházejí od palubního počítače. Komunikace s nahrávací aplikací je implementována v třídě `RecServer`.

4.10 Vytváření záznamů z provozu a jejich anotace

Pro pořizování záznamů kvůli testování aplikace je možné použít konfigurační aplikaci. Záznamy jsou nahrávány v rozlišení 640 x 480 bez komprese, přestože sledovací algoritmus pracuje na rozlišení 160 x 120. Záznamy byly nahrávány ve vyšším rozlišení, aby bylo možné vyzkoušet jak pracuje sledovací algoritmus ve vyšším rozlišení. V budoucnu by se mohl změnit hardware jednotky a tyto záznamy by mohly být využitelné. Během pořizování záznamů bylo nutné být fyzicky ve voze s notebookem, který byl připojen k LAN ve vozu. Proces nahrávání záznamů byl časově velice náročný. Pomocí konfigurační aplikace bylo možné nahrávat záznam pouze z jednoho zařízení. Navíc bylo nahrávání časově omezené z důvodu omezené výdrže baterie notebooku, na který se záznam nahrával.

Pro vylepšení procesu nahrávání byla implementována aplikace `app_recorder`. S touto aplikací je postup nahrávání podobný jako s konfigurační aplikací. Rozlišení přenášené disparity je však stejné, jaké se používá pro počítání cestujících. Bylo však možné nahrávat záznamy ze všech jednotek ve voze najednou. Dochází také k záznamu příkazů o začátku a konci počítání od palubního počítače. Tato aplikace zefektivnila proces nahrávání záznamů. Stále však zůstala nutnost být u záznamu fyzicky přítomen, a také přetrvával problém s výdrží baterie.

Popsané nevýhody nahrávacího systému vyřešilo až použití jednodeskového počítače Raspberry Pi 3. Jedná se o malý počítač se čtyřjádrovým procesorem Arm, 4 USB porty, LAN konektorem, WiFi a dalšími periferiemi. Ten je připojen do sítě ve vozidle a pořizuje záznamy disparitní mapy během provozu. Pro nahrávání na Raspberry Pi byla také upravena aplikace `app_recorder`. Nahrávání je teď možné ovládat automaticky pomocí palubního počítače, pomocí přepínače připojeného ke GPIO portům počítače nebo bezdrátově

přes WiFi síť vytvořenou na Raspberry Pi.

Počítač je napájen pomocí externí baterie s kapacitou 10000 mAh. S úspornými opatřeními, které byly na počítači nastaveny, vydrží baterie více než 12 hodin provozu, což je pro pořizování záznamu dostačující. Nahrávání záznamů disparity tímto způsobem odstraňuje oba problémy, které se vyskytovaly u předchozích metod.

Pořízené záznamy je nutné dále zpracovat. K tomu byla implementována aplikace `app_records_parser`. Pomocí této aplikace se rozdělí záznamy celých jízd na jednotlivé zastávky nebo kratší úseky. Sledování cestujících v mezizastávkovém prostoru by zpomalovalo testování. Zároveň uživatel u každého záznamu spočítá průchody cestujících dovnitř a ven z vozu. Výstupem jsou pak zkrácené záznamy a také soubor s počty průchodů cestujících, který se použije pro testování.

Kapitola 5

Testování

Během testování byly určeny hodnoty parametrů celého systému pro jednotlivé typy vozů. Jejich změnou je možné výrazně ovlivnit funkci jednotlivých částí systému. Tato kapitola se bude zabývat vlivem těchto parametrů na funkci jednotlivých částí a také celého systému. V úvodu zde také bude popsána výpočetní náročnost jednotlivých kroků řešení.

Kvůli ochraně soukromí cestujících není možné nahrávat a uchovávat videozáznamy z kamer ve voze při jeho normálním provozu. Z tohoto důvodu byla zaznamenávána pouze vypočtená disparitní mapa. Pro testování funkčnosti výpočtu disparitní mapy však byly pořízeny videozáznamy z kamer v prázdném voze popřípadě záznamy z instalace jednotek do vozidla. Pro ladění výpočtu disparitní mapy to bylo dostačující.

Celkově bylo vytvořeno a ručně anotováno přes 4200 záznamů ze zastávek z 19 různých vozů nejčastěji autobusů, také trolejbusů a tramvají. Záznamy se pořizovaly za různého počasí a v různou denní dobu včetně nočních hodin. Taková testovací sada je velká, vyhodnocení by trvalo dlouhou dobu. Pro potřeby diplomové práce byly vybrány 3 datové sady (každá z jiného typu vozu).

Testy vlivu jednotlivých parametrů probíhaly následujícím způsobem. Pro každý typ vozu byly v konfiguračním souboru nastaveny hodnoty parametrů, které mají nejnižší chybovost. S hodnotou testovaného parametru se postupně pohybovalo a zkoumal se jeho vliv jak na výslednou hodnotou chybovosti, tak jeho reálný důsledek na funkci systému. Testy jednotlivých parametrů jsou rozděleny podle části systému, kterou ovlivňují.

K testování byla implementována aplikace `app_eval`. Vstupem pro test jsou zkrácené záznamy disparitních map ze zastávek, soubor s anotací záznamů a konfigurační soubor jednotky, na které byl záznam pořízen. Testovací aplikace simuluje funkci aplikace spuštěné na jednotce UCP-01. Výstupem je tabulka aplikace Microsoft Excel, ve které se nachází přehled výsledků testu. U každého vstupního záznamu je vypsána informace o napočítaném a reálném počtu vstupů, výstupů a jejich diference. V souboru s výsledky se také nachází sumarizační informace o celém testu. Ukazatelem úspěšnsosti fungování aplikace je hodnota `Total error rate`, která ukazuje procento chybovosti algoritmu na dané sadě záznamů. Jeho hodnota je vypočtena podle vzorce:

$$Err = \frac{\sum_{i=0}^{n} DIFFin_i + DIFFout_i}{\sum_{i=0}^{n} GTin_i + GTout_i} * 100\%, \tag{5.1}$$

kde $DIFFin_i, DIFFout_i$ je rozdíl spočtených vstupů/výstupů u záznamu i vůči pravdě , $GTin_i, GTout_i$ je reálný počet vstupů/výstupů u záznamu i a n je počet záznamů v testu.

Je to poměr počtu nezapočítaných a falešně započítaných cestujících vůči součtu průchodů přes dveřní prostor.

Současně je také možné v aplikaci zapnout vizualizaci algoritmu, která zobrazuje mezivýsledky jednotlivých kroků běhu algoritmu. Je tak možné vidět jak byla vypočtena disparitní mapa, jak vypadá model pozadí a jakým způsobem jsou přiřazovány detekce do tras.

Jedním významným faktorem, který negativně ovlivňoval výslednou chybovost algoritmu, je výskyt dětí a kočárků ve vozech MHD. Jejich výška je totiž na hranici výšky stanovené pro klasifikaci cíle jako cestujícího. V případě průchodu dětí a průjezdu kočárků dveřním prostorem nebylo možné zaručit chování aplikace, totiž jestli budou inkrementovány čítače průchodů přes dveřní prostor, nebo ne. Při ruční anotaci zaznamenaných disparitních map také nebylo možné určit jak vysoké dítě je, respektive jestli jej započítat, či nikoliv. Během anotace se k takovým záznamů přidával komentář o jejich výskytu a během testů se tyto záznamy nepoužívaly.

5.1 Rychlost výpočtu

Ještě před implementací aplikace byly provedeny testy rychlosti různých jednoduchých algoritmů pro zpracování obrazu na jednotce UCP-01. Na základě těchto testů byl zvolen způsob implementace jednotlivých částí aplikace. Výsledky těchto testů se nachází v tabulce 5.1. Z dat je patrné, že pro jednoduché operace, jako je součet dvou matic, není na nízkém rozlišení velký rozdíl v čase zpracování implementace na procesoru, OpenGL nebo OpenCL.

		Čas T[ms]		
Operace	Rozlišení	OpenCV (cpu)	OpenGL	OpenCL
Součet (char)	128x128	1,190	1,453	1,866
	640x480	19,377	7,501	12,661
Součet (float)	128x128	1,395	-	2,219
	640x480	28,332	-	26,797
Eroze (char)	128x128	3,753	1,987	0,945
	640x480	51,691	38.813	11,940
Kopie (char)	128x128	0,038	1,453	1,222
	640x480	7,192	19,429	23,447

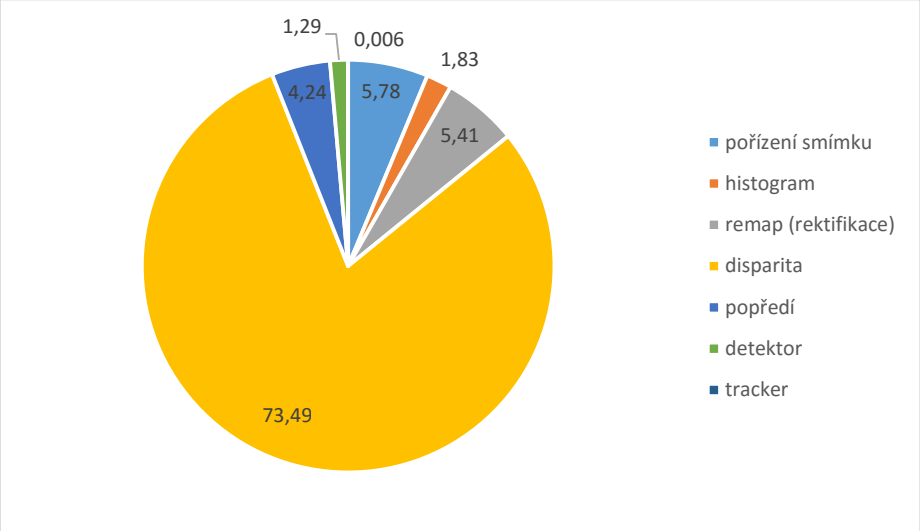
Tabulka 5.1: Výsledky původního testu rychlosti.

U jednoduchého kopírování jedné matice do druhé je implementace na CPU rychlejší, než na grafickém čipu. Naproti tomu u výpočtu eroze je už poznat velký rozdíl. Jedná se o složitější operaci, kterou je zároveň možno provádět paralelně samostatně pro každý pixel výsledného obrazu.

Pro provádění operací na GPU je totiž nutné nejprve nahrát data na grafický čip. Tento čas je konstantní pro jedno rozlišení, nezáleží na prováděném algoritmu. Další čas je stráven spouštěním tzv. kernelu, který v sobě obsahuje samotný výpočet. Z těchto dvou důvodů není výhodné používat grafický čip pro jednoduché operace. Pro složitější výpočty se zpracování na grafickém čipu naopak hodí. Je vhodné přesouvat data z čipu a na něj v co nejmenší míře a provádět co nejvíce výpočtů v jednom kernelu. Tím se ušetří čas při kopírování dat a při spouštění kernelů. Samotná implementace na GPU se tak stane efektivnější.

Pro zpracování obrazu bylo nakonec použito OpenCL, které je pro takové výpočty programátorsky přívětivější a také umožňuje používat datový typ `float` narozdíl od verze OpenGL, který podporuje jednotka UCP.

Pro aplikaci `app_upc` je důležité, aby snímky zpracovávala v reálném čase. Bylo tedy nutné zaručit určitý čas výpočtu pro jeden snímek. Pro vstupní snímky bylo zvoleno rozlišení 2 x 160 x 120. Po implementaci byly provedeny testy rychlosti jednotlivých kroků algoritmu. První část zpracování vstupních snímků se provádí v konstantním čase. Čas běhu detektoru a sledovací části se odvíjí od počtu cestujících ve scéně.



Obrázek 5.1: Rozdělení průměrného času (v milisekundách) zpracování jednoho snímku jednotlivých částí algoritmu. Na prázdné scéně je průměrný čas zpracování 92 ms.

Pokud se ve scéně nachází cestující, trvá výpočet detekce a sledování déle. Takový nárůst je v řádu desetin, maximálně jednotek milisekund. Celkový čas se tak pohybuje kolem 93 ms, tak byla zvolena rychlost snímání jako 10 snímků za sekundu. Na obrázku 5.1 je zobrazeno rozložení času zpracování jednotlivých částí algoritmu. Nejnáročnější je výpočet disparitní mapy. To odpovídá očekávanému stavu. Při výpočtu disparity je provedeno mnoho per-pixel operací:

- 16x obraz absolutních diferencí
- 16x integrální obraz
- suma absolutních diferencí + subpixelová interpolace

Ty jsou samozřejmě na celém algoritmu časově nejnáročnější. Pro zlepšení výpočtu disparitní mapy byla přidána ekvalizace histogramu vstupních snímků. Samotná ekvalizace neprodloužila čas zpracování jednoho snímku, výpočet histogramu ho však prodloužil na 99 ms, což už je velice blízko hranice 100 ms na snímek při zachování zvoleného počtu snímků za sekundu. Histogram se tedy počítá jen ze čtvrtiny obrazu. Testy ukázaly, že vliv ekvalizace pomocí takto vypočteného histogramu je na fungování algoritmu podobný jako standardní ekvalizace.

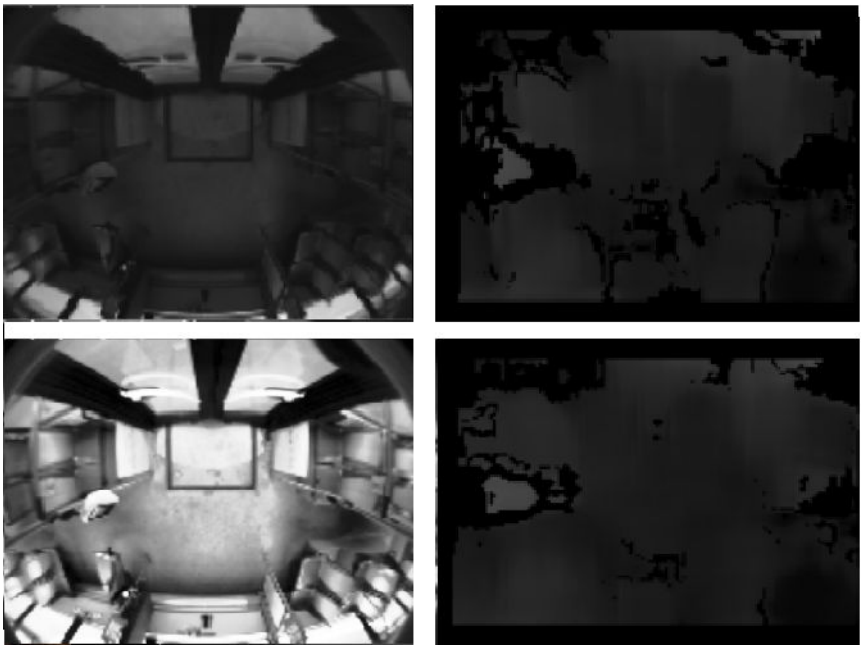
Při provozu aplikace jsou spuštěné ještě další služby, například server pro komunikaci s palubním počítačem. Jejich vliv na rychlost zpracování obrazu na jednotce UCP-01 je však

zanedbatelný. Jedinou náročnější operací je přenos obrazu z kamery do palubního počítače, kde je nutné obraz překódovat do textového kódování. U této služby je však omezena četnost odeslaných snímků, tak aby to negativně neovlivňovalo hlavní funkci aplikace. Během nahrávání záznamů a instalace jednotky do vozidla je sledovací algoritmus vypnut.

5.2 Disparitní mapa

Testování výpočtu disparitní mapy bylo nejprve prováděno v jiných prostorách než ve vozech. Její hodnocení se provádělo opticky porovnáním se vstupním obrazem a znalostí scény v obraze. Po implementaci celé aplikace byly také provedeny testy pomocí evaluační aplikace popsané na začátku této kapitoly, kde je hodnocením úspěšnosti položka `Total error rate` v souboru s výsledkem testu.

Prvním zlepšujícím faktorem bylo zavedení předzpracování obrazu před samotným výpočtem disparity. Jde konkrétně o ekvalizaci histogramu. V kapitole 4.3.1 byla popsána funkce a důvod použití tohoto algoritmu v našem systému. Jde o to unifikovat světlost obou vstupních obrazů tak, aby v nejlepším případě odpovídající body v obou obrazech měly stejnou hodnotu.

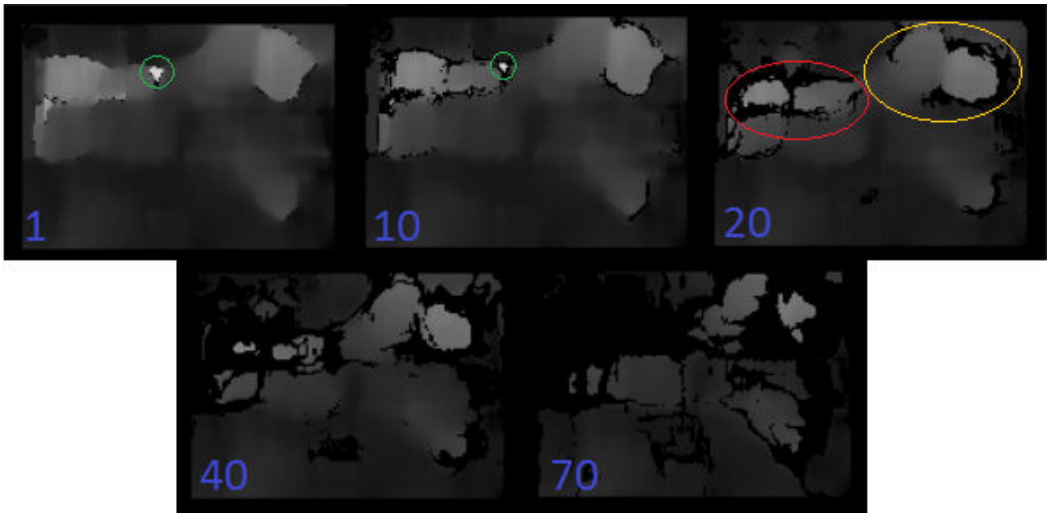


Obrázek 5.2: Rozdíl mezi disparitní mapou vypočtenou z neupraveného (nahoře) a ekvalizovaného (dole) obrazu. Vlevo vstupní obraz, napravo disparitní mapa.

Rozdíl mezi disparitou vypočtenou z ekvalizovaného obrazu a z obrazu bez předzpracování je patrný z obrázku 5.2. Černá místa představují nevypočtenou disparitu. Je vidět, že druhá disparitní mapa obsahuje méně bodů s neznámou hodnotou disparity. Obecně lze říci, že v čím horší kvalitě jsou vstupní snímky (tmavé snímky, ostré světlo jen na několika místech obrazu), tím více ekvalizace zlepšuje vypočtenou disparitní mapu.

Popis parametrů a způsob jak ovlivňují výpočet disparitní mapy se nachází v kapitole 4.3.3. Prvním parametrem je `uniquess_ratio`. Ten udává o kolik lepší musí být hodnocení

nalezené disparity, než je druhé nejlepší. Jde o určitou unikátnost řešení. Pokud by hodnocení různých hodnot disparity pro jeden bod bylo podobné, je pravděpodobé, že žádná z těchto hodnot nebude správná.



Obrázek 5.3: Vypočtená disparitní mapa s různými nastaveními parametru uniquess ratio. Červeně a oranžově jsou označeni cestující. Zeleně chybně vypočtené hodnoty disparity.

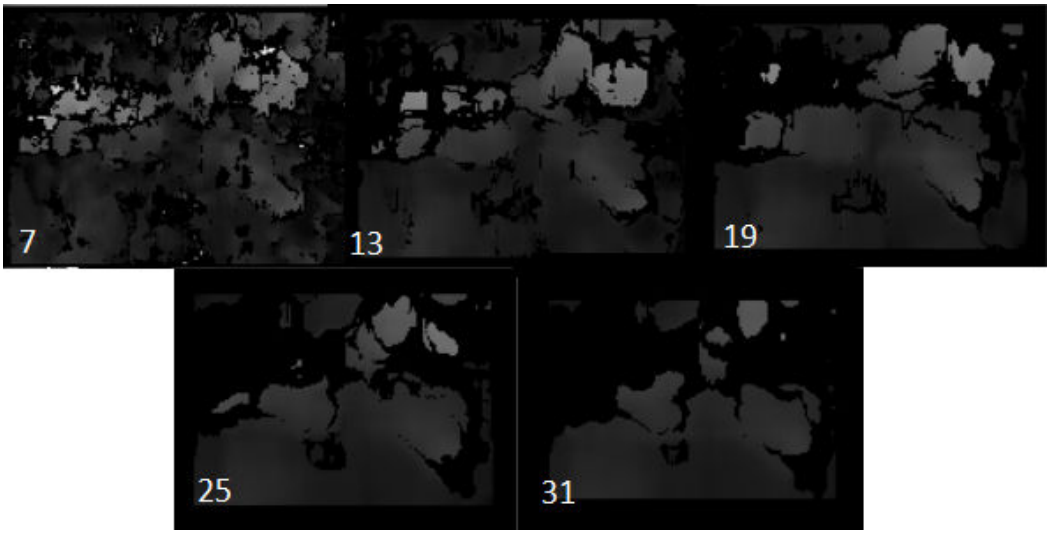
Na obrázku 5.3 jsou zobrazeny vypočtené disparitní mapy s různou hodnotou tohoto parametru. Je vidět, že nižší hodnoty vytvářejí jemnější přechody mezi hodnotami na disparitní mapě. Vyšší hodnoty způsobují nevypočtení disparity pro místa, kde dochází k prudké změně hloubky (v našem případě na okraji siluety cestujícího). Pro vyšší hodnoty už někteří cestující nejsou vidět vůbec. Naopak nízké hodnoty mohou způsobovat špatně vypočtenou hodnotu disparity pro nejednoznačná místa. V případě, který je na obrázku by tak oblast označená zeleně byla detekována jako další cestující.

		uniquess_ratio				
sada	best	1	10	20	40	70
bus	6,29	18,18	17,48	6,99	7,69	12,58
tram	3,85	15,38	7,69	5,76	4,49	3,85
trol	5,42	8,43	5,42	4,82	5,42	6,62

Tabulka 5.2: Vliv parametru uniquess ratio na chybovost algoritmu. Chybovost je vyjádřena jako Total error rate popsaný na začátku kapitoly. Čím nižší číslo, tím lepší.

V tabulce 5.2 se nachází výsledky testu provedeného s různými hodnotami tohoto parametru. Ve sloupci `best` je výsledek s hodnotou parametru, který se ve voze reálně používá. Nejlepší hodnota `uniquess_ratio` souvisí s pozicí jednotky. V tramvajích je jednotka výš, než v ostatních typech vozu a také není nakloněna ven z vozu. Pozice jednotky v autobuse a trolejbuse je podobná. Hodnota použitá na jednotkách se pohybuje mezi 20-50 dle typu vozu.

Další parametr, který ovlivňuje funkci výpočtu disparity je velikost okna pro Sumu absolutních diferencí. Jak již bylo řečeno, díky použití integrálního obrazu nemá tento parametr vliv na rycholst výpočtu disparitní mapy. Na obrázku 5.4 je vidět vypočtená disparitní



Obrázek 5.4: Disparitní mapa vypočtená s různými velikostmi okna pro sumu absolutních diferencí.

mapa s různými nastaveními tohoto parametru. Scéna je stejná, jako na předchozím obrázku. Nižší hodnota tohoto parametru poskytuje členitější disparitní mapu, která však obsahuje mnoho nesouvislých částí. Na rozhraní různých hloubek se často vyskytuje šum, který by v detektoru generoval falešné cíle. Zvyšování hodnoty způsobuje snížení výskytu takového šumu. Disparitní mapa obsahuje větší celistvější plochy, ale ostré přechody mezi hloubkami se zjemňují, nebo na místech přechodu není vypočtena disparita vůbec. Také se zmenšuje velikost vypočtené disparitní mapy.

		sad window				
sada	best	7	13	19	25	31
tram	3,84	22,44	8,33	3,84	5,12	11,54
trol	5,42	16,27	6,02	5,42	10,84	30,12

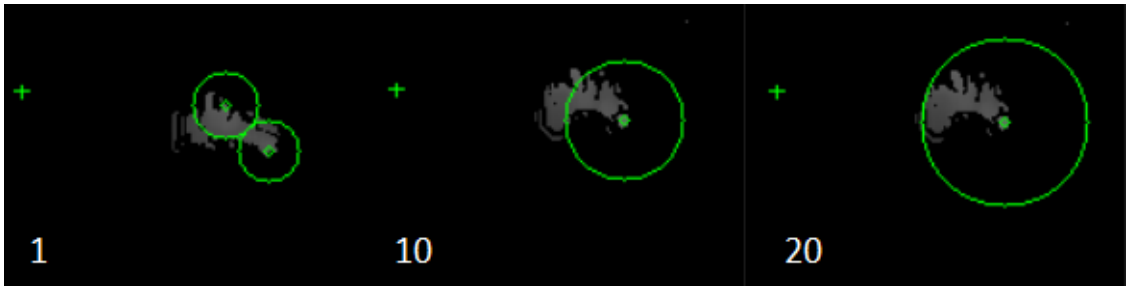
Tabulka 5.3: Vliv velikosti vyhledávacího okna SAD na chybovost algoritmu.

V tabulce 5.3 se nachází výsledky testů provedených s různou hodnotou velikosti okna SAD. U tohoto parametru nemá žádný vliv pozice jednotky ani vzhled scény. Šlo jen o to najít takový parametr, který by dostatečně eliminoval šum a zároveň nedegradoval rozdíly v disparitě. Hodnota velikosti okna SAD je na jednotkách UCP-01 nastavena na 19.

5.3 Detektor

Funkce detektoru byla popsána v kapitole 4.5. Parametr `head_safety_coeff` upravuje odhad velikosti cestujícího na disparitní mapě. Velikost je vypočtena na základě hodnoty disparity. Jelikož jsou však jednotky v různých vozech různě vysoko, používá se pro korekci tento parametr. Vypočtená velikost je také použita na odstranění detekovaného cestujícího z disparitní mapy a následnou detekci dalších cestujících.

Na obrázku 5.5 jsou zobrazeny detekce na vyextrahovaném popředí a také oblast očekávaného výskytu cestujícího. Ve scéně se nachází jeden cestující. Při použití nízké hodnoty je sice cestující detekován správně, ale není dostatečně odstraněn z popředí. Tak dochází k



Obrázek 5.5: Detekce na vyextrahovaném popředí s různým nastavením head_safety_coeff.

jeho opětovné detekci na špatné pozici. U velkých hodnot je zase více cestujících detekováno jako jeden.

Vliv hodnoty tohoto parametru na chybovost aplikace je zobrazen v tabulce 5.4. Z tabulky je vidět, že se v provozu vždy nepoužívají nejlépe hodnocené parametry. V těchto konkrétních případech by sice měla změna hodnoty pozitvní vliv na snížení chybovosti, při použití větší datové sady by se však nejlepší hodnocení blížilo zvoleným hodnotám.

		head_safety_coeff				
sada	best	1	5	10	15	20
tram	3,85	19,23	11,54	8,97	3,85	12,18
bus	6,29	9,09	8,39	6,29	3,50	4,19
trol	5,42	4,88	4,82	4,21	5,42	6,63

Tabulka 5.4: Vliv parametru head_safety_coeff na chybovost algoritmu.

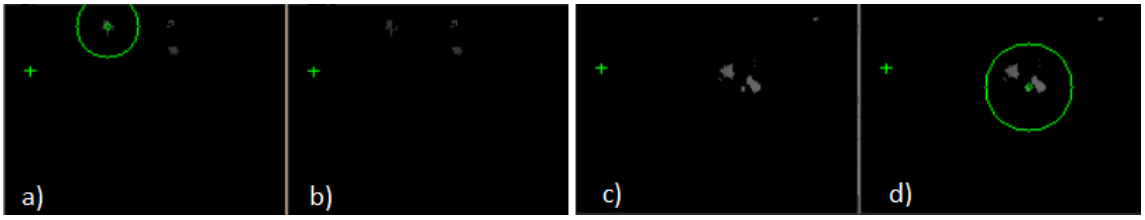
Obecně by se dalo říct, že čím blíže k podlaze vozidla je jednotka umístěna, tím větší by měla být hodnota koeficientu pro velikost cestujícího.

Parametr `min_disp_non_zero_ratio` udává jaký podíl z odhadnuté velikosti cestujícího musí být viditelný na vypočteném popředí. Velmi úzce souvisí s nastavením výpočtu disparitní mapy. Na obrázku 5.5 bylo vidět, že většina odhadnutého prostoru popředí neobsahovala. Prvním důvodem je nastavení disparitní mapy, kde jsou parametry nastaveny tak, aby nevznikal šum. Důsledkem toho se na rozhraních různých hodnot disparit často hodnota vůbec nevypočítá a cestující tak zabírají menší plochu.

Dalším důvodem je fakt, že střed detekované oblasti se nachází na místě s nejvyšší disparitou. Tento bod však není totožný s nějakým středem nebo těžištěm bodů, které danému cestujícímu patří. Hodnota tohoto parametru je plošně nastavena na 15%, ale kdyby se změnilo nastavení výpočtu disparitní mapy, je nutné změnit i tento parametr.

Na obrázku 5.6 je ukázka situací, které jsou ovlivněny hodnotou `min_disp_non_zero_ratio`. První situace je naznačena na obrázcích a, b, jejichž vstupem je stejná disparitní mapa. Vlevo je hodnota 5%, vpravo 15%. Na levém snímku je detekována velice malá oblast, která je ve skutečnosti šum. Kdyby se v popředí vyskytoval tento šum častěji, mohlo by to negativně ovlivnit sledovací algoritmus. Cestující procházející touto oblastí by byli sledováni špatně.

Obrázky c, d vznikly také ze stejné disparitní mapy. Vlevo je parametr nastaven na 20% vpravo je 15%. Cestující byl v této situaci přímo pod jednotkou a byl tak vysoký, že pro velkou část prostoru, který zabíral ve scéně, nebyla vypočtena disparita. Pro tyto situace je naopak výhodnější hodnotu parametru snížit. Tyto dvě výše popsané situace jsou však



Obrázek 5.6: Detekce na vyextrahovaném popředí s různým nastavením min_disp_non_zero.

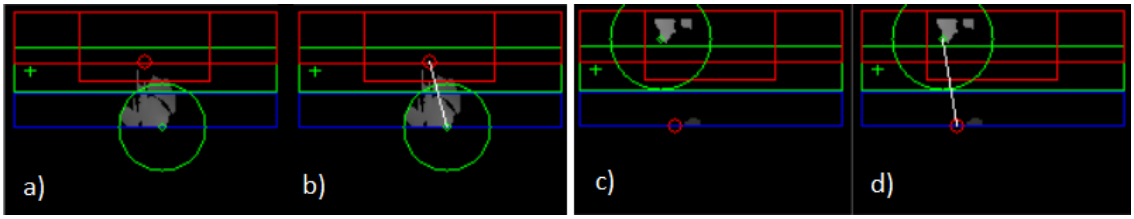
velice vzácné a změna hodnoty o jednotky procent mění chybovost výsledku o desetiny procenta.

5.4 Sledovací algoritmus

Pro sledování cestujících byly implementovány algoritmy na bázi hledání nejbližšího souseda a také algoritmus generující více hypotéz. V kapitole 4.6 se nachází popis jejich implementace. Všechny dříve zmíněné testy byly provedeny s algoritmem hledání nejbližšího souseda, protože poskytuje lepší výsledky. Funkci obou těchto algoritmů ovlivňuje hodnota parametru `gate_radius`. Parametr představuje maximální vzdálenost (v pixelech) přiřazení měření do trasy, nebo také maximální vzdálenost po sobě jdoucích bodů v jedné trase.

Hodnota tohoto parametru souvisí se vzdáleností jednotky od země a také s počtem snímků za sekundu. S rostoucí vzdáleností jednotky od podlahy se snižuje délka dráhy, kterou může cestující urazit mezi dvěma pořízenými snímky. Taktéž vyšší počet snímků za sekundu způsobí zkrácení této dráhy. Je také nutno vzít v potaz rychlost jakou se cestující pohybují. Pro zjištění hodnoty byly provedeny testy ve voze, hodnota se dále upravovala během testování na datové sadě.

Na obrázku 5.7 je ukázka nejčastějších chyb vznikajících vlivem špatného nastavení velikosti `gate_radius` pro trasu. Na obrázcích je velkým obdélníkem ohraničena oblast detekce a sledování. Červené body jsou poslední měření přiřazená trasám, které jsou v systému. Zelené body jsou nová měření. Přímka mezi červeným a zeleným bodem znamená asociaci měření s trasou. V situaci na obrázcích a, b se cestující pohybuje přes dveřní prostor velice rychle a vzdálenost, kterou urazil, je větší než hodnota gate. V situaci na obrázku a) sledovací algoritmus vytvoří novou trasu v zeleném bodě. Jelikož trasa končící červeným bodem nekončí v oblasti uvnitř vozu (modrý obdélník), cestující nebyl započítán. Na obrázku b) je stejná situace s větší hodnotou brány.



Obrázek 5.7: Chyby vzinkající při špatném nastavení parametru gate_radius.

Naopak častá chyba vznikající nastavením vysoké hodnoty gate je vidět na obrázcích c, d. Ve scéně se nachází dva cestující. První je uvnitř vozu, druhý teprve vchází. Na obrázku

d) je kvůli vysoké hodnotě gate asociována trasa sledující prvního cestujícího s měřením, jehož původcem je druhý cestující. Průchod těchto dvou cestujících je poté započítán jako jeden vstup. Na obrázku c) je správné chování sledovacího algoritmu. Trasa sledující prvního cestujícího zanikne a vzniká nová trasa původem od druhého cestujícího.

Fungování algoritmu sledování na základě více hypotéz velmi ovlivňuje funkce hodnotící jednotlivé hypotézy. Byly implementovány dvě verze hodnotící funkce, první (H_1) je založena na počtu průchodů, které hypotéza vygenerovala. Čím více jich bylo vypočteno, tím lepší bylo hodnocení hypotézy. Algoritmus s touto hodnotící funkcí často generoval větší počet průchodů přes dveřní prostor, než byla pravdivá hodnota.

Druhá hodnotící funkce (H_2), která byla implementována, je založena na výpočtu průměrné vzdálenosti mezi jednotlivými body v trasách. Idea byla použít ty hypotézy, ve kterých jsou po sobě jdoucí měření co nejblíže. Tak by měly být trasy plynulejší. V úvahu se braly pouze trasy, které vygenerovaly změnu počtu průchodů dveřmi. Tato hodnotící funkce má menší chybovost, než předchozí založená na počtu průchodů. Výsledky testů provedených na algoritmu s více hypotézami a porovnání s algoritmem GNN se nachází v tabulce 5.5.

	algoritmus		
sada	GNN	MHT H_1	MHT H_2
tram	3,85	37,18	36,54
bus	6,29	16,78	13,27
trol	5,42	20,48	16,87

Tabulka 5.5: Provnání chybovosti použitých sledovacích algoritmů.

Testy sledovacího algoritmu generujícího více hypotéz dopadly hůře, než pomocí přiřazení nejbližšího souseda. Nejedná se o chybu v samotném generování hypotéz, ale o špatnou volbu hodnotící funkce. Hypotéza odpovídající té, kterou vygeneruje algoritmus GNN, je vytvořena, ale její hodnocení je vetšinou horší, než u jiných hypotéz. Řešením by bylo vytvořit hodnotící funkci, která by lépe stanovovala nejlepší hypotézu, taková zatím nebyla nalezena.

Jelikož byly v testovacích sadách použity i záznamy s dětmi a kočárky, u kterých není fungování aplikace jasně definováno, je úspěšnost přístupu s algoritmem GNN dostatečná. Během testu, ve kterém zaměstnanci dopravního podniku porovnávali skutečný počet průchodů dveřmi a hodnoty vypočtené systémem, byla chybovost kolem 5%, což je pro potřeby dopravních průzkumů akceptovatelné.

Vlivem vibrací a možná i tepelných změn ve vozidle nastávaly situace, kdy se v objektivu posunuly polohy čoček a tím i parametry kamery, rektifikované snímky pak už nebyly řádkově zarovnané. Často docházelo k posunu obrazu ve vertikálním směru o jednotky pixelů. V těchto situacích se zhoršila chybovost o 5-10%. Tento problém byl však technologicky vyřešen a parametry kamer by se již neměly samovolně měnit.

Kapitola 6

Závěr

V této práci je popsaná metoda detekce, sledování a počítání cestujících ve vozech hromadné dopravy. Aplikace měla fungovat na jednotce UCP-01 a měla pracovat v reálném čase, tomu byl podřízen návrh a implementace. Vzhledem k podmínkám a možnostem byly zvoleny metody pro výpočet disparitní mapy, detekci cílů a jejich sledování. Část práce se také zabývala urychlením některých výpočtů na grafickém čipu. Pro sledování cílů byly otestovány algoritmy Global nearest neighbor a Multiple hypothesis tracking. Nakonec byl použit první algoritmus, protože dosahoval lepších výsledků. Pokud by byl v budoucnu nalezen lepší způsob hodnocení hypotéz, než momentálně používaný, mohl by algoritmus s více hypotézami poskytovat lepší výsledky.

Pomocí testů na různých datových sadách zaznamenaných disparitních map byla zjištěna úspěšnost průměrně 95%, což je pro použití aplikace dostačující. Chyby v počítání nejčastěji vznikají při průchodu dětí nebo nízkých cestujících dveřním prostorem. Systém je implementován tak, aby byl co nejodolnější vůči špatným světelným podmínkám, které mohou ve vozech vznikat. Ostré světlo svítící do vozu může zhoršit úspěšnost o jednotky procent.

Pro zlepšení aplikace by mohlo pomoci zvýšení rozlišení disparitní mapy. Ta by nebyla jen podrobnější, ale také by měla větší hloubkovou přesnost. Do detektoru by poté přicházely přesnější informace a bylo by možné lépe rozlišit minimální výšku cestujícího pro jeho započítání. Další možností vylepšení by bylo zvýšit počet snímků pořízených za sekundu. Scéna by tak byla snímána v menších časových úsecích a sledovací algoritmus by měl více informací o trajektorii cestujících. Také by docházelo k úspěšnější asociaci měření a tras. Oba tyto návrhy na zlepšení by ovšem vyžadovaly výkonnější hardware a byla by tak nutná výměna jednotky UCP-01.

Úprava funkce aplikace se provádí pomocí změn hodnot parametrů aplikace. Tyto parametry ovlivňují funkci jednotlivých výpočetních částí a pro každý typ vozu se hodí jiná sada těchto parametrů. Vzhledem k velké testovací sadě trvá hledání vhodné kombinace parametrů velmi dlouho a je pravděpodobné, že existuje taková kombinace parametrů, která má menší chybovost, než ta současně používaná. Bylo by možné, ať pomocí brute-force nebo například genetického programování, najít vhodnější kombinace parametrů, se kterými má aplikace vyšší úspěšnost. Musela by se ale zvolit vhodná datová sada, nebo urychlit vyhodnocení testů.

V současnosti je systém automatické počítání cestujících (APC) s jednotkami UCP-01 nasazen asi v padesátce vozů Dopravního podniku Ostrava. Jedná se o autobusy, tramvaje i trolejbusy. Jednotky by po malé úpravě aplikace mohly být použity pro počítání procházejících lidí v budovách nebo na jiných místech.

Literatura

[1] Computer Vision – The Integral Image [Online]. 2010, [cit. 2016-05-09].
URL https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/

[2] Angelos, A.; Thomaidis, G.; Maroudis, P.; aj.: *Multiple Hypothesis Tracking Implementation, Laser Scanner Technology*. INTECH Open Access Publisher, 2012.

[3] Blackman, S. S.; Popoli, R.: *Design and analysis of modern tracking systems*. Artech House, 1999.

[4] Bradski, D. G. R.; Kaehler, A.: *Learning Opencv, 1st Edition*. O'Reilly Media, Inc., první vydání, 2008, ISBN 9780596516130.

[5] Bradski, G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] Cyganek, B.; Siebert, J. P.: *An introduction to 3D computer vision techniques and algorithms*. John Wiley and Sons, 2009.

[7] docs.opencv.org: Histogram Equalization - OpenCV 2.4.13.0 documentation [Online]. 2014, [cit. 2016-05-09].
URL http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html

[8] Facciolo, G.; Limare, N.; Meinhardt-Llopis, E.: Integral Images for Block Matching. *Image Processing On Line*, ročník 4, 2014: s. 344–369, doi:10.5201/ipol.2014.57.

[9] Hall, D. L.: *Handbook of multisensor data fusion*. CRC Press, 2001.

[10] Haller, I.; Nedevschi, S.: Design of Interpolation Functions for Subpixel-Accuracy Stereo-Vision Systems. *IEEE Trans. Image Processing*, ročník 21, č. 2, 2012: s. 889–898, doi:10.1109/TIP.2011.2163163.
URL http://dx.doi.org/10.1109/TIP.2011.2163163

[11] Hartley, R.; Zisserman, A.: *Multiple view geometry in computer vision*. Cambridge University Press, druhé vydání, 2003.

[12] Josefsson, S.: RFC 4648 - The Base16, Base32, and Base64 Data Encodings. 2006.
URL https://tools.ietf.org/html/rfc4648

[13] Mroz, F.; Breckon, T. P.: An empirical comparison of real-time dense stereo approaches for use in the automotive environment. *EURASIP Journal on Image and Video Processing*, ročník 2012, č. 1, 2012: str. 13, doi:10.1186/1687-5281-2012-13.
URL http://dx.doi.org/10.1186/1687-5281-2012-13

[14] Richter, M.: Kalmanův Filtr [Online]. [cit. 2016-05-09].
URL http://www.uamt.feec.vutbr.cz/~richter/vyuka/0910_mpov/tmp/kalman_filter.html

[15] Wöhler, C.: *3D computer vision efficient methods and applications*. Springer, druhé vydání, 2013.

Přílohy

Seznam příloh

Příloha A

Obsah DVD

- `/src` - Zdrojové soubory aplikace app_upc
- `/app_upc` - Binární aplikace spustitelná na systému Windows, 32bit
- `/datasets` - Záznamy z insalace některých jednotek a odpovídající konfigurační soubory
- `/test_results` - Soubory aplikace Microsoft Excel s výsledky některých testů
- `/prace/pdf` - Tato práce ve formátu PDF
- `/prace/tex` - Zdrojové soubory této práce
- `readme.txt` - Návod na spuštění aplikace app_upc