ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra radioelektroniky

Algoritmy a využití fúze multispektrálních obrazových dat

Algorithms and Application of Multispectral Image Fusion

Diplomová práce

Studijní program:	Elektronika a komunikace
Studijní obor:	Audiovizuální technika a zpracování signálů
Vedoucí práce:	Ing. Stanislav Vítek, Ph.D.

Bc. Jana Kolmašová

Praha 2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kolmašová** Jméno: **Jana** Osobní číslo: **434657**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Elektronika a komunikace**

Studijní obor: **Audiovizuální technika a zpracování signálů**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Algoritmy a využití fúze multispektrálních obrazových dat

Název diplomové práce anglicky:

Algorithms and Application of Multispectral Image Fusion

Pokyny pro vypracování:

Cílem práce je zvýšení informačního obsahu v datech z bezpečnostních systémů pomocí metod obrazové fůze multispektrálnách dat.
1) Proveďte rešerši metod fúze multispektrálních obrazových dat
2) Navrhněte a implementujte algoritmus registrace obrazu z termokamery a CMOS kamery pracující v blízkém IR pásmu, včetně metody vyhodnocení kvality registrace.
3) Pro data se zvýšeným informačním obsahem navrhněte a implementujte algoritmy pro vyhledávání a klasifikaci objektů a určení jejich trajektorií.

Seznam doporučené literatury:

[1] STATHAKI, Tania. Image fusion: algorithms and applications. Elsevier, 2011
[2] GONZALEZ, Rafael C.; WOODS, Richard E. Processing. Prentice-Hall, 2002
[3] AGHAJAN, Hamid; CAVALLARO, Andrea (ed.). Multi-camera networks: principles and applications. Academic press, 2009.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Stanislav Vítek, Ph.D., katedra radioelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **08.02.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **20.09.2020**

Ing. Stanislav Vítek, Ph.D.	prof. Mgr. Petr Páta, Ph.D.	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studentky

Čestné prohlášení

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

V Praze dne .....

podpis

Poděkování

Ráda bych poděkovala Ing. Stanislavu Vítkovi, Ph.D. za odborný dohled a vedení diplomové práce. Dále chci poděkovat grantu MPO s evidenčním číslem FV20548 *Videodetekce osob v tunelu v reálném čase* za poskytnutá data a finanční podporu.

Abstrakt

Práce se věnuje implementaci metod fúze obrazu multispektrálních dat. Popisuje předzpracování obrazu, registraci a fúzi dat metodami PCA, DWT a LaP. Následně tyto metody porovnává jak pomocí metrik kvality obrazové fúze (např. SSIM), tak podle úspěšnosti detekce osob v obrazových datech. Detekce objektů je implementována dvěma způsoby. Prvním je vytvoření modelu pozadí, jeho odečtení, detekce hran a prahování nalezených ploch. Druhý způsob je kaskádový klasifikátor na bázi Haarových příznaků, který je natrénován pomocí algoritmu AdaBoost.

Klíčová slova: Fúze obrazu; Multispektrální data; Kaskádový klasifikátor; AdaBoost

Abstract

The present Thesis focuses on implementation of multispectral image fusion methods. It discusses image preprocessing, data registration and PCA, DWT or LaP methods of fusion. Then it compares these methods with metrics of image fusion quality (e.g. SSIM) or by success in person detection in the image data. There are two implementations of object detection. The first is the creation of the background model, its subtraction, edge detection and thresholding of the found areas. The second way is the cascade classifier based on Haar features, that has been trained by the AdaBoost algorithm.

Keywords: Image Fusion; Multispectral Data; Cascade Classifier; AdaBoost

Obsah

Úvod

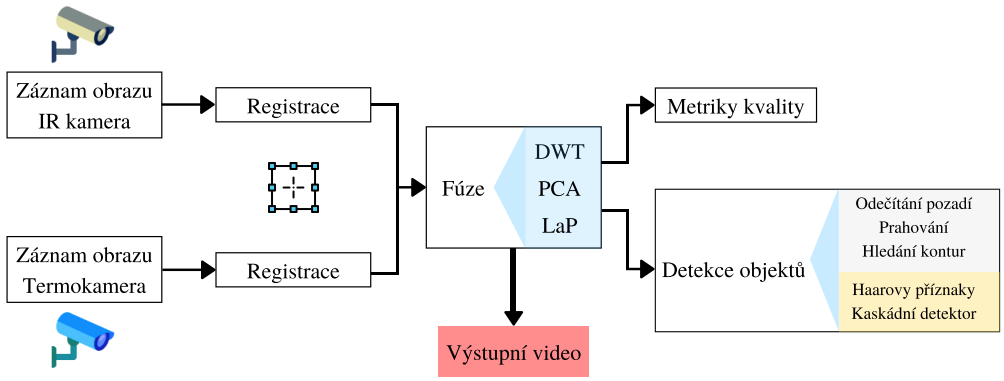
Bezpečnostní kamerové systémy dnes nalezneme ve všech obchodních centrech, na letištích či parkovištích. Ne vždy jsou ale jasové podmínky prostředí ideální pro záznam obrazu běžnou kamerou, pracující ve viditelné části spektra. Proto se používají kamery s rozšířeným spektrem do infračervených hodnot nebo termokamery, které snímají infračervené záření. V případě této práce se jedná o použití dvou zmíněných kamer na stejnou scénu, s cílem omezit působení proměnlivých jasových podmínek a detekovat osoby v záběrech.

Problematika mě zaujala, protože téma detekce osob v bezpečnostních záběrech z kamer je aktuální a v průmyslu hojně využívané. V dnešní době vzniká obrovské množství obrazových dat a je žádoucí z nich vytěžit maximum informací. Často je nedostatečná kvalita nahrávacích zařízení kompenzována pozdějším zpracováním zaznamenaných dat. Fúze obrazu je spojení více snímků do jednoho obrazu, který obsahuje zásadní informace z obou původních snímků. Jedná se o jednu z možností snížení objemu obrazových dat bez ztráty dat užitečných.

Cílem této práce je zvýšit informační obsah v obrazu pomocí fúze multispektrálních obrazových dat. Fúze snímků se často používá pro zvýšení hloubky ostrosti či pro sloučení medicínských obrazových dat. V případě této práce se aplikace vzdáleně podobá postupům používaným v medicíně, kde se ze snímků, zobrazujících různé části spektra, vytváří snímek se zásadními informacemi z obou snímků. Zvolila jsem pro implementaci tři druhy fúze: analýzu hlavních komponent, vlnkovou transformaci a Laplaciánskou pyramidu, protože podle [1], [2] a [3] mají lepší výsledky oproti následujícím druhům fúze: maximum jasu, minimum jasu, průměrování jasu, morfologická pyramida, gradientní pyramida nebo FSD pyramida. Ne vždy se zdroje ohledně kvality metod shodnou, záleží také na použitých metrikách pro hodnocení kvality fúze.

Kvalita fúze se může posuzovat i podle schopnosti správné detekce objektů ve snímcích. Proto práce obsahuje i popis dvou způsobů detekce: odečítání pozadí s prahováním a hledáním kontur a kaskádový klasifikátor na bázi Haarových příznaků, natrénovaný algoritmem AdaBoost. Porovnáním výsledků detekce srovnáme nejen metody fúze mezi sebou, ale i informační přínos fúze vzhledem k originálním datům.

Na obrázku 1 je znázorněn postup fúze obrazu od snímání scény až po získání výsledného sloučeného obrazu.



Obrázek 1: Postup fúze obrazu a detekce objektů

Proces fúze videa se skládá z několika částí, stejně jako tato práce. První kapitola se věnuje předzpracování obrazu, kam patří popis použitých kamer a jejich spektrální citlivost. Následuje korekce zkreslení objektivu a transformace záznamů z kamer. Tyto kroky potřebujeme k docílení co nejlepší registrace obrazů na sebe. Druhá kapitola popisuje samotnou fúzi obrazu, vysvětluje tři často používané druhy fúze: pomocí analýzy hlavních komponent, vlnkové transformace a Laplaciánské pyramidy. Na konci kapitoly je souhrn metrik kvality fúze, pomocí kterých se metody pro fúzi obrazu porovnávají. V poslední kapitole teoretické části je vysvětlen postup detekce objektů v obraze. První popsaný způsob je odečítání pozadí s prahováním a následná detekce kontur. Druhý způsob je klasifikace pomocí Haarových příznaků. V praktické části je popsána implementace veškerých zmíněných kroků, následuje vyhodnocení kvality fúze a porovnání výsledků detektorů.

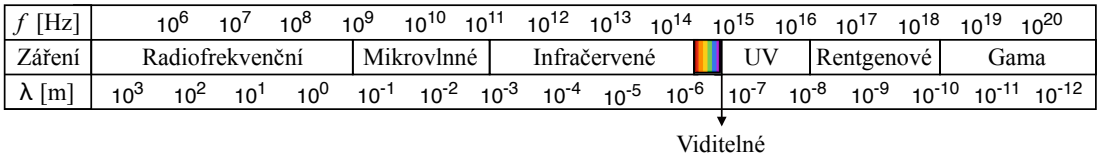
1 Předzpracování obrazu

Před tím, než je možné aplikovat fúzi obrazu, je potřeba dobře porozumět záznamu a zpracování obrazových dat. Začátek kapitoly je zaměřen na základní teorii, kterou je potřeba znát pro porozumění zpracování obrazu. Následuje souhrn úkonů, které předchází fúzi obrazů. Nejprve je potřeba pořídit záznam obrazu, k čemuž se vážou vlastnosti a parametry kamery či fotoaparátu. V našem případě si vysvětlíme principy kamer, ze kterých byl pořízen veškerý materiál, který v této práci používáme. Dále je potřeba znát metody registrace obrazu, aby měla fúze co nejefektivnější výsledky.

1.1 Elektromagnetické spektrum

Elektromagnetické záření se skládá z periodických vln, které se šíří prostorem. Je definováno vlnovou délkou nebo frekvencí. Vlnová délka a frekvence vlnění jsou spojeny vzorcem $\lambda = \frac{c}{f}$, kde λ je vlnová délka, f je frekvence a c je rychlost světla ve vakuu ($3 \cdot 10^8$ ms^{-1}). Energie záření se šíří pomocí fotonů, což jsou kvanta energie s nulovou klidovou hmotností. Vzorec $E = h \cdot f$ vyjadřuje energii fotonu, kde h je Planckova konstanta ($6,626 \cdot 10^{-34}$ Js).

Na obrázku 2 je znázornění části elektromagnetického spektra od 1 MHz. Nejnižší frekvenci má radiofrekvenční záření a na opačném konci spektra s nejvyšší frekvencí vlnění nalezneme ionizující záření (rentgenové a gama). Mezi nimi se na vlnových délkách 380 - 760 nm nachází viditelné světlo, které zabírá jen úzkou část elektromagnetického spektra. Každé těleso, které má vyšší teplotu než absolutní nula, vyzařuje infračervené záření. S vyšší teplotou je spojena i vyšší frekvence záření [4].



Obrázek 2: Spektrum elektromagnetického záření

Kamerové záznamy, které v dalších částech práce používáme, zobrazují trochu větší část spektra než je viditelné světlo. První z kamer snímá viditelné světlo a blízké infračervené záření. Druhá použitá kamera je termokamera, která snímá infračervenou část spektra o vlnové délce v řádu μm.

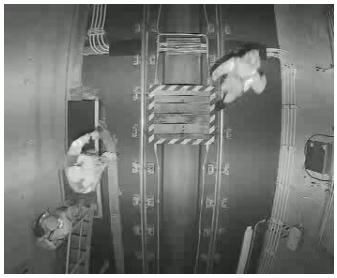
1.2 Záznam obrazu

Účelem používání kamer je záznam trojrozměrné scény do jedné roviny. Obraz vznikne perspektivní transformací reality na senzor kamery. Pro jednodušší matematický popis záznamu obrazu se používají afinní modely projekce [5]. V následujících podkapitolách jsou popsány dvě kamery a jejich výstup, protože s nimi budeme v dalších částech pracovat. Každá z kamer má své výhody a nevýhody, proto budeme záznamy z obou kamer kombinovat.

1.2.1 CMOS kamera pracující v blízkém IR pásmu

Kamera snímá viditelné a blízké infračervené světlo. V práci ji pro jednoduchost nazýváme infrakamerou, i když zaznamenává i viditelnou část spektra. Záznamy, které vzniknou snímáním nedostatečně osvětlené scény, jsou velmi tmavé a nekontrastní. Pro zlepšení kontrastu se někdy používají infračervené reflektory. Jejich nevýhodou je nedostatečný efektivní dosvit a špatná distribuce svitu do stran, proto je často osvětlen jen střed scény. Pohybující se objekt má potom různé hodnoty jasu v různých částech obrazu.

Pokud je scéna osvětlena přímým slunečním světlem, jas záznamu se velmi změní a je náročné rozlišit detaily v původně tmavých částech scény. Na obrázku 3 je příklad snímku zaznamenaného infrakamerou.



Obrázek 3: Příklad snímku z infrakamery

1.2.2 Termokamera

Dalším typem kamery, který budeme používat je termokamera, která zaznamenává pouze infračervenou část spektra. Pokud jsou ve scéně osoby, jsou kontrastnější oproti pozadí v porovnání s záběrem z infrakamery. Se snímáním teploty objektů se vážou i velké nevýhody. Kamera aktualizuje dynamický rozsah podle jasu ve scéně. Pokud srovnáme záznamy pořízené v létě a v zimě, je v dynamickém rozsahu obrovský rozdíl. Na rozdíl od prostředí mají osoby v záznamu v létě a v zimě teplotu podobnou, proto má jejich vstup do scény v létě a v zimě odlišnou odezvu. Ve studeném prostředí se po vstupu osoby do scény musí dynamický rozsah jasu zvětšit několikanásobně více než u teplého prostředí.

Obrázek 4: Příklad snímku z termokamery

Dále pokud je scéna snímána v noci, objekty ve scéně mají většinou vyšší teplotu než vzduch. Při snímání ve dne se podmínky obrátí, vzduch je často teplejší než objekty ve scéně. Na obrázku 4 je příklad snímku zaznamenaného termokamerou v zimě.

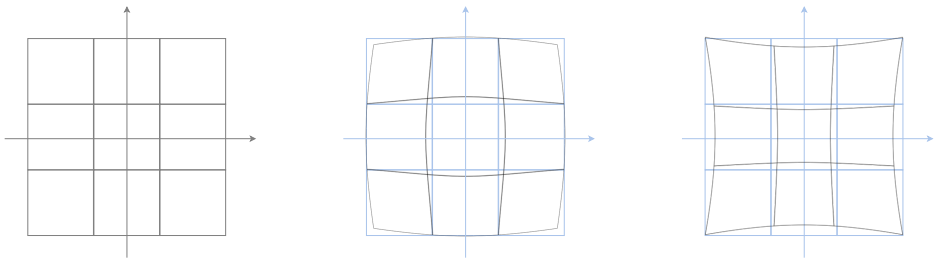
1.3 Registrace obrazu

Registrace obrazu je takový souhrn úprav obrazů, který umožňuje správně aplikovat fúzi. Kapitola vysvětluje dvě úpravy zaznamenaného obrazu, které se nám v dalším postupu budou hodit. První je korekce zkreslení objektivu, které mezi geometrickými vadami čoček převládá. Jedná se o radiální zkreslení, které je způsobeno nepřesností při výrobě čočky. Druhou úpravou je projektivní transformace, díky které se perspektiva zaznamenané scény změní tak, aby na sebe jednotlivé pixely obrazů pasovaly.

1.3.1 Korekce radiálního zkreslení objektivu

Reálné kamery nefungují přesně podle matematických modelů, ale mají různá zkreslení. Častá je radiální vada čočky, která způsobí soudkovité nebo poduškovité zkreslení obrazu, viz obrázek 5. Radiální vada čočky vzniká při její výrobě a způsobuje změnu úhlu vystupujícího paprsku oproti vstupujícímu paprsku. Kvůli tomu nejsou body zobrazeny přesně do stejné polohy jako ve scéně a jejich odchylka od správné polohy se se vzdáleností od středu objektivu mění. Následující rovnice vyjadřuje radiální zkreslení čočky objektivu [5]. Hodnoty x a y jsou výsledkem perspektivního modelu a L je polynom, který modeluje radiální vzdálenosti od středu obrazu.

$$\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = L\left(\sqrt{x^2 + y^2}\right) \begin{bmatrix} x \\ y \end{bmatrix} \tag{1.1}$$



(a) Bez zkreslení (b) Soudkovité (c) Poduškovité

Obrázek 5: Radiální zkreslení objektivu

Korekce vady objektivu se většinou řeší snímáním mřížky s definovaným tvarem a porovnáním se vzniklým obrazem [5].

1.3.2 Projektivní transformace

Projektivní transformace zaznamenává, jak se mění scéna se změnou umístění pozorovatele a umožňuje simulovat perspektivní zkreslení obrazu. Transformace se vyjadřuje pomocí následující matice

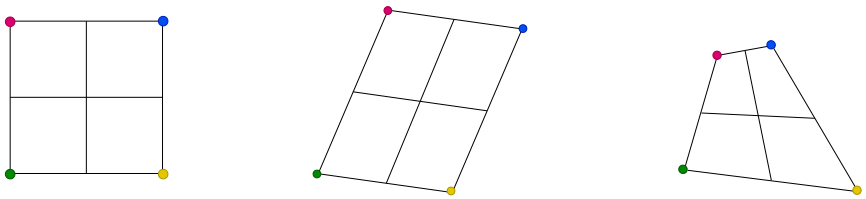
$$\begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ c_1 & c_2 & 1 \end{bmatrix}, \tag{1.2}$$

kde $\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$ je rotační matice, $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ je translační vektor a $\begin{bmatrix} c_1 & c_2 \end{bmatrix}$ je projekční vektor. Transformační matice se násobí se souřadnicemi pixelu, a tím vzniknou

souřadnice nového bodu. Projektivní transformace nezachovává rovnoběžnost, vzdálenosti a úhly.

Afinní transformace je speciálním případem projektivní transformace, kdy $c_1 = c_2 = 0$. Podmínky pro afinní transformaci nenastávají často. Roviny obrazu obou kamer musí být rovnoběžné s rovinou xy. Dále můžeme tuto transformaci použít, pokud je scéna od objektivu dostatečně vzdálená a rotační úhly α a β jsou velmi malé [5]. Afinní transformace na rozdíl od projektivní zachovává rovnoběžnost.

Na obrázku 6 je schématicky znázorněn rozdíl mezi originálním a transformovaným obrazem pomocí afinní a projektivní transformace. Na obrázku 7 jsou ve stejném pořadí zobrazeny transformace na reálné scéně.



(a) Bez transformace (b) Afinní (c) Projektivní

Obrázek 6: Druhy transformace obrazu

(a) Bez transformace (b) Afinní (c) Projektivní

Obrázek 7: Příklady transformace obrazu

2 Fúze obrazu

Fúze dat je znázornění scény, které by nebylo možné získat pomocí jednoho senzoru, vytvořené pomocí kombinace informací o dané scéně z více senzorů. Důvodem pro využívání fúze obrazu je vytvoření obsáhlejšího obrazu s více informacemi, než je možné získat z jednotlivých senzorů [6]. Vylepšená obrazová data se dále používají pro lepší vizuální vnímání nebo pro počítačové zpracování [5].

Vybrané důvody pro použití různých senzorů pro fúzi obrazu jsou podle [7] následující:

- Zvýšení rozsahu podmínek, při kterých je možné sledovat scénu
- Zvýšení rozsahu časových a prostorových vlastností obrazu
- Snížení nepřesností a chyb v obraze
- Redundance dat, která je užitečná při selhání jednoho ze senzorů
- Informačně obsáhlejší reprezentace dat

Většina technik fúze obrazu je kompromisem mezi dostačujícím prostorovým rozlišením a spojitostí spektra [8]. Často se používají následující techniky: analýza hlavních komponent (PCA) a vlnková transformace. Z [7] vyplývá, že zmíněné techniky jsou mezi metodami fúze obrazu nejpoužívanější. Nedá se říct, která z metod je nejlepší, protože záleží na cíli aplikace. Metoda PCA zajistí zvýšení informačního obsahu obrazu bez změny prostorových a spektrálních detailů, kdežto použití vlnkové transformace zachová původní RGB hodnoty a navíc zmenší zkreslení obrazu.

2.1 Analýza hlavních komponent

Metoda analýzy hlavních komponent (PCA) slouží k dekorelaci spektrálních pásem signálu. Získáme obraz s vysokým rozlišením z jednoho snímku a informací o spektru z druhého snímku [9]. Výhodou PCA je, že umožňuje snížení počtu dimenzí dat. Protože se při využití PCA neztrácí velké množství informací, je využívána pro kompresi obrazu.

Postup PCA je následující:

- Z původních obrazů (I_1 a I_2) se vytvoří sloupcové vektory, obsahující hodnoty jasu.
- Aby analýza dobře fungovala, musí mít vektory střední hodnotu 0. To docílíme odečtením průměru vektoru od každé hodnoty tohoto vektoru.
- Pak se spočítá kovarianční matice těchto sloupcových vektorů. Kovarianční matice obsahuje rozptyl hodnot jasu na hlavní diagonále a kovarianci jasu snímků na vedlejší diagonále.

$$C = \begin{bmatrix} \sigma^2_{I_1} & cov_{I_1 I_2} \\ cov_{I_1 I_2} & \sigma^2_{I_2} \end{bmatrix} \tag{2.1}$$

$$cov_{I_1 I_2} = \frac{\sum_{i=1}^{n} (I_{1i} - \bar{I_1})(I_{2i} - \bar{I_2})}{n-1} \tag{2.2}$$

- Z matice získáme vlastní čísla (λ_1 a λ_2) a vlastní vektory ($\mathbf{u}_1$ a $\mathbf{u}_2$).

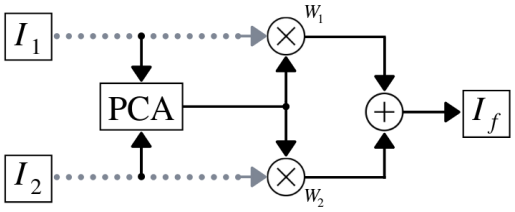
- Porovnáme vlastní čísla mezi sebou a vektor, který odpovídá vyššímu vlastnímu číslu, použijeme jako vážené koeficienty pro fúzi obrazů.

$$\mathbf{u} = \begin{cases} \mathbf{u}_1 & \text{pro } \lambda_1 > \lambda_2 \\ \mathbf{u}_2 & \text{jinde} \end{cases} \tag{2.3}$$

- Vážené obrazy sečteme a dostaneme výsledek (viz vzorec 2.4), kde W_1 a W_2 jsou prvky vlastního vektoru $\mathbf{u}$ [1][10].

$$I_f(x,y) = W_1(x,y)I_1(x,y) + W_2(x,y)I_2(x,y) \tag{2.4}$$

Na obrázku 8 je vidět postup aplikace PCA při fúzi obrazu. Jedná se o váženou superpozici snímků na sebe.



Obrázek 8: Fúze pomocí PCA, překresleno z [11]

2.2 Vlnková transformace

Vlnková transformace rozkládá signál do vlnek, u kterých známe jak frekvenci, tak měřítko. Na rozdíl od Fourierovy transformace získáme kromě dobrého frekvenčního rozlišení i časové rozlišení. Mateřskou vlnku (ψ) upravujeme pomocí roztažení (a) a posunutí (b) a provádíme po úsecích konvoluci se signálem ($f(x)$). Tím zjistíme, které frekvence byly v jaké části signálu použity. Při rozkladu signálu pomocí vlnkové transformace data dekorelujeme [12]. Následující vzorec popisuje základní 1D vlnkovou transformaci:

$$W_{a,b}(f(x)) = \int_{x=-\infty}^{\infty} f(x)\psi_{a,b}(x)dx \tag{2.5}$$

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right) \tag{2.6}$$

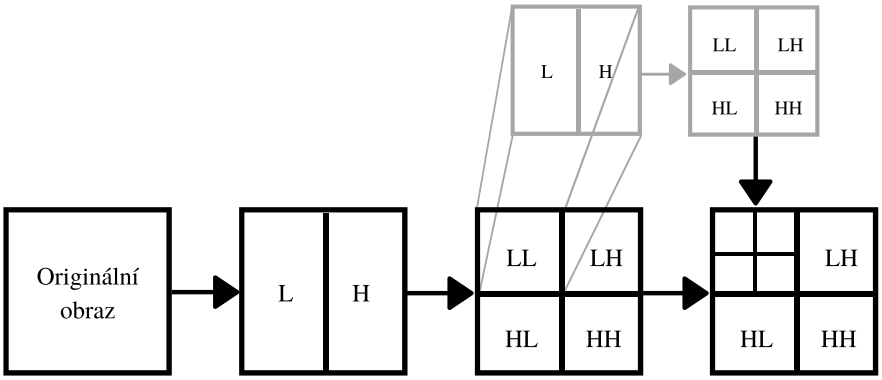
Inverzní vlnková transformace se počítá z vlnkových koeficientů následujícím vzorcem:

$$f(x) = \sum_{a,b} W_{a,b}\psi_{a,b}(x) \tag{2.7}$$

Pro snížení objemu dat se používá diskrétní vlnková transformace (DWT). Počítá se jen s koeficienty $a = 2^m$ a $b = n2^m$. Nejrychlejším způsobem, jak spočítat DWT, je rychlá vlnková transformace. Protože se vlnková funkce chová jako pásmová propust kolem centrálního kmitočtu, může být rovnice 2.5 nahrazena filtrací. Signál se filtruje

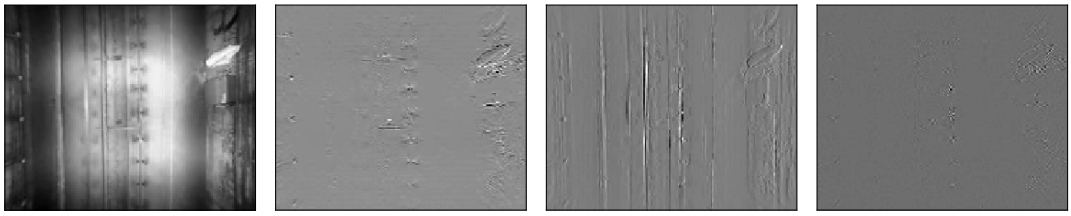
dvojicí kvadraturně zrcadlových filtrů, dolní a horní propustí. Nízké frekvence jsou pak stejnými propustmi znovu filtrovány, dokud nedocílíme požadované úrovně detailů [13].

Pro dekompozici obrazu používáme 2D vlnkovou transformaci místo 1D. Celý proces funguje následovně (postup je znázorněn obrázkem 9):



Obrázek 9: Dyadická dekompozice u 2D vlnkové transformace, inspirováno [14]

- Aplikace 1D vlnkové transformace nejdříve na řádky a pak na sloupce signálu.
- Obraz je filtrován dolní propustí (L) a horní propustí (H) v horizontálním směru.
- Následuje vzorkování v horizontálním směru, je ponechán každý druhý sloupec. Podvzorkování je možné udělat, protože i po zmenšení šířky pásma stále splníme Nyquistův teorém.
- Výsledné obrazy jsou znovu filtrovány pomocí L a H, ale ve vertikálním směru.
- Následuje vzorkování ve vertikálním směru, je ponechán každý druhý řádek. Vstupní signál se tak rozdělí do 4 podpásem, které vznikly kombinací kvadraturně zrcadlových filtrů (L a H). Tato 4 podpásma jsou: podpásmo s nejvyšší energií (LL), horizontální (HL), vertikální (LH) a diagonální (HH) (příklad na obrázku 10).



Obrázek 10: Příklad dělení do 4 podpásem, zleva LL, HL, LH, HH

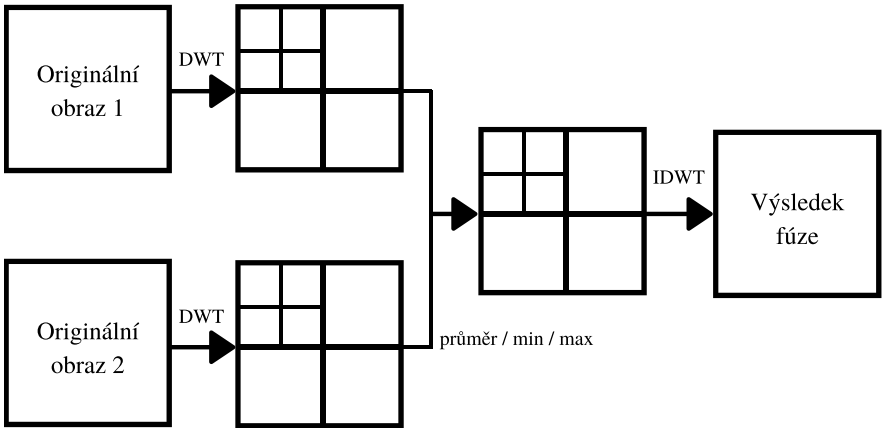
- Na podpásmo s nejvyšší energií (LL) se poté znovu aplikuje dělení do 4 podpásem.
- Postup se opakuje, dokud nedosáhneme požadované úrovně dekompozice [15].

Pro rekonstrukci obrazu se používá inverzní 2D vlnková transformace (IDWT), kde je postup přesně opačný, než výše uvedený. Nejprve se vloží nulové řádky do vyfiltrovaných obrazů, pak se filtruje ve vertikálním směru. Do výsledků se vloží nulové sloupce a pak se filtruje v horizontálním směru. Filtrace probíhá pomocí kvadraturně zrcadlových filtrů, pro které platí buď rovnice 2.8 nebo 2.9, aby byla rekonstrukce věrná. Součtem všech obrazů vznikne rekonstruovaný obraz [16].

$$F'_L(z) = F_H(-z), F'_H(z) = -F_L(-z) \tag{2.8}$$

$$F'_L(z) = -F_H(-z), F'_H(z) = F_L(-z), \tag{2.9}$$

kde F je filtr použitý při přímé vlnkové transformaci a F' je filtr použitý při inverzní vlnkové transformaci.



Obrázek 11: Schéma fúze obrazu pomocí 2D vlnkové transformace

Vlnkovou dekompozici aplikujeme na obrazy, na které chceme použít fúzi (obrázek 11). Koeficienty vlnkové transformace obou signálů na každé úrovni můžeme zprůměrovat nebo vzít minimum nebo maximum z nich. Tím spojíme obrazy na každé úrovni transformace. Po fúzi koeficientů se aplikuje inverzní diskrétní vlnková transformace. Tím získáme obraz se zvýšeným informačním obsahem oproti samostatným originálním snímkům [17].

2.3 Laplaciánská pyramidová fúze

Laplaciánská pyramida (LaP) je odvozena od Gaussovské pyramidy (GaP). Původní obraz je spodní úrovní pyramidy, z něj se odvozují všechny následující úrovně. První úroveň GaP vznikne filtrací obrazu předchozí úrovně pomocí váhovací funkce, která hodnoty zprůměruje. Druhá úroveň vznikne aplikací stejného modelu na první úroveň pyramidy. S každou úrovní pyramidy se sníží rozlišení obrazu na polovinu v řádcích i ve sloupcích [18]. Příklad GaP je na obrázku 12. Princip rozkladu je také popsán následujícím vzorcem:

$$g_l(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) g_{l-1}(2i+m, 2j+n), \tag{2.10}$$

kde $w(m,n)$ je váhovací funkce a l je úroveň pyramidy. Váhovací funkce má normální rozdělení pravděpodobnosti.

$$w = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{2.11}$$

Obrázek 12: Úrovně Gaussovské pyramidy

Pro rekonstrukci obrazu z pyramidy se používá interpolace. Po aplikaci následujícího vzorce na určitou úroveň pyramidy zvětšíme obraz na velikost obrazu o jednu úroveň níže.

$$g_{l,k}(i,j) = 4\sum_{m=-2}^{2}\sum_{n=-2}^{2} w(m,n)g_{l,k-1}\left(\frac{i-m}{2}, \frac{j-n}{2}\right), \tag{2.12}$$

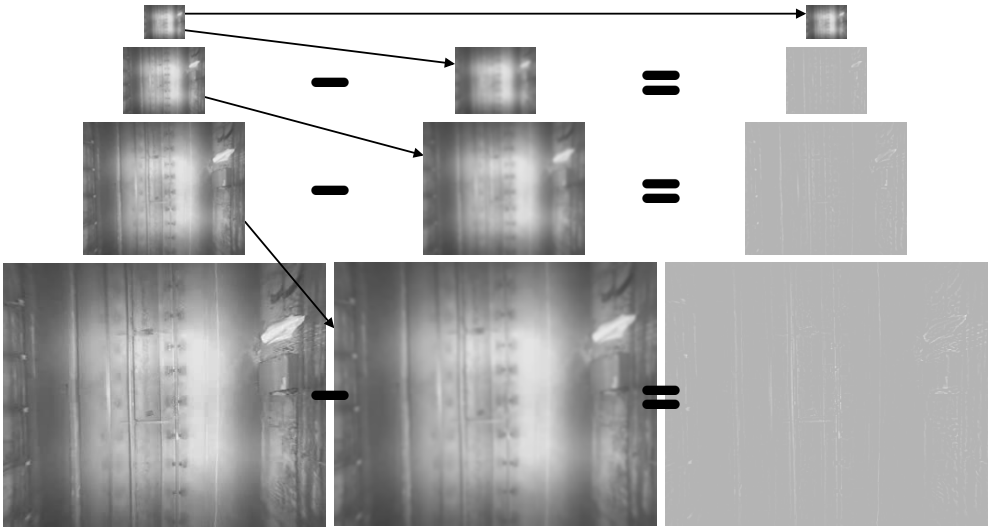
Pro zmenšení objemu dat, potřebného k dalšímu zpracování, se využívá Laplaciánská pyramida. Místo komprimované verze obrazu z předchozí úrovně využíváme pouze rozdíl mezi úrovněmi GaP. Pro získání spodní úrovně LaP využijeme zvětšení obrazu z GaP z 1. úrovně (podle vzorce 2.12), tento zvětšený obraz poté odečteme od spodní úrovně GaP. Tento proces je pro více úrovní pochopitelný z obrázku 13. Pro nejvyšší úroveň pyramidy se LaP rovná GaP [18].

Originální obraz získáme z LaP tak, že obraz určité úrovně zvětšíme (podle vzorce 2.12), a poté všechny úrovně sečteme. Efektivnějším způsobem je zvětšit nejvyšší úroveň a sečíst obraz s nižší úrovní, poté sečtený obraz zvětšit a sečíst s následující úrovní, dokud nedostaneme originální obraz.

LaP využíváme pro fúzi obrazu tak, že z obou originálních obrazů vytvoříme GaP a z ní LaP. Porovnáme obrazy na jedné úrovni a pro fúzi vytvoříme takový obraz, který se skládá z maximálních hodnot pixelů z obou obrazů (viz vzorec 2.13 a 2.14). Takto postupujeme na všech úrovních až dostaneme jednu LaP, ze které rekonstruujeme obraz po fúzi [19].

$$g_l(i,j) = L_l(i,j) + g_{l+1}(i,j), \tag{2.13}$$

$$L_l(i,j) = \begin{cases} L_l^A(i,j) & \text{pro } \left|L_l^A(i,j)\right| > \left|L_l^B(i,j)\right| \\ L_l^B(i,j) & \text{jinde} \end{cases} \tag{2.14}$$



Obrázek 13: Vznik Laplaciánské pyramidy - pravý sloupec má kvůli přehlednosti zvýšený jas

2.4 Vyhodnocení kvality fúze obrazu

Metriky pro vyhodnocení kvality fúze obrazu se dělí do dvou kategorií. Testování kvality s referenčním obrázkem a bez referenčního obrázku. Měření kvality s referencí se provádí pomocí následujících matematických metod: odmocnina střední kvadratické chyby (RMSE), maximální odstup signál-šum (PSNR), SSIM index. Pro měření kvality fúze bez reference se používá například: křížová entropie (CE) a vzájemná informace fúze (FMI) [20]. V tabulce 1 jsou vzorce, jak jednotlivé metriky vypočítat.

2.4.1 Metriky kvality fúze s referencí

- RMSE používá k výpočtu hodnoty jasu referenčního obrazu a obrazu po fúzi. Popisuje kvalitu obrazu jako celku, kdyby měl obraz po fúzi zkreslení na malé ploše, hodnota RMSE by mohla stále vyjít přijatelně. Čím menší RMSE, tím vyšší kvalita obrazu [21].

- PSNR je běžně užívaná metrika pro zjištění kvality fúze obrazu, která považuje obraz za speciální druh signálu. V tomto případě se místo energie šumu používá rozdíl mezi referencí a obrazem po fúzi. Vyšší hodnoty PSNR značí nízkou energii šumu, tedy vyšší kvalitu obrazu [21].

- SSIM na rozdíl od předchozích metrik nezkoumá chyby v obraze, ale strukturální změny obrazu. Díky tomu mají obrazy s téměř identickými hodnotami RMSE odlišné hodnoty SSIM. Při výpočtu se porovnávají tři vlastnosti: jas, kontrast a struktura. Po porovnání jasu obrazů je průměrný jas odečten a následuje porovnání kontrastu. Obraz normalizujeme vydělením jeho směrodatnou odchylkou (kontrastem) a porovnáváme strukturu. Tato tři porovnání se následně zkombinují a vytvoří vzorec pro SSIM. Čím vyšší je hodnota SSIM, tím větší podobnost struktur obrazů [22].

Metriky kvality fúze s referencí	
Odmocnina střední kvadratické chyby	$RMSE = \sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(I_r(i,j) - I_f(i,j)\right)^2}$
Maximální odstup signál-šum	$PSNR = 10log_{10}\left(\frac{L^2}{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(I_r(i,j)-I_f(i,j))^2}\right)$, kde L je počet stupňů šedé
SSIM index	$SSIM = \frac{(2\mu_{I_r}\mu_{I_f}+C_1)(2\sigma_{I_rI_f}+C_2)}{(\mu_{I_r}^2+\mu_{I_f}^2+C_1)(\sigma_{I_r}^2+\sigma_{I_f}^2+C_2)}$, kde $C_1 = 2,55$ a $C_2 = 7,65$ pro 255 stupňů šedé
Metriky kvality fúze bez reference	
Křížová entropie	$CE(I_1, I_2; I_f) = \frac{CE(I_1;I_f)+CE(I_2;I_f)}{2}$, kde $CE(I_x; I_f) = \sum_{i=0}^{L} h_{I_x}(i)log\left(\frac{h_{I_x}(i)}{h_{I_f}(i)}\right)$
Vzájemná informace fúze	$FMI = MI_{I_1I_f} + MI_{I_2I_f}$, kde $MI_{I_xI_f} = \sum_{i=1}^{M}\sum_{j=1}^{N} h_{I_xI_f}(i,j)ln\left(\frac{h_{I_xI_f}(i,j)}{h_{I_x}(i,j)h_{I_f}(i,j)}\right)$

Tabulka 1: Metriky kvality fúze obrazu

2.4.2 Metriky kvality fúze bez reference

- CE vyjadřuje podobnost informačního obsahu obou zdrojových obrazů a obrazu po fúzi. Čím více je společného obsahu v obrazech, tím nižší je hodnota křížové entropie [23].
- FMI měří závislost zdrojových obrazů a obrazu po fúzi. Z histogramů samotných obrazů a vzájemných histogramů spočítáme vzájemnou informaci mezi zdrojovým a výsledným obrazem. Součtem těchto výsledků získáme vzájemnou informaci fúze. Vyšší hodnota FMI značí větší kvalitu fúze [23].

3 Detekce v obraze

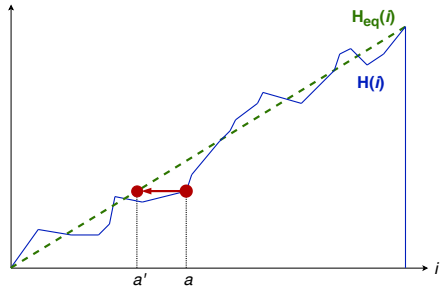
Po registraci a fúzi videa se budeme věnovat krokům, které předchází detekci objektů ve videu. Nejprve vysvětlím, jak vyrovnat histogram a poté jaké jsou varianty modelování pozadí. Vytvořené pozadí se od snímku odečte a zbylé objekty rozřadíme na důležité a nedůležité pomocí prahování. Pak následuje vyhledávání hran nebo kontur v obraze. Další část této kapitoly se věnuje popisu kaskádového detektoru. Používané metody se liší podle druhu videa a natáčeného prostředí. Tato kapitola shrnuje základní postupy, které se hodí pro úpravu a detekci ve videích, použitých v této práci.

3.1 Vyrovnání histogramu

Pro vyvážení hodnot jasu v obraze se využívá vyrovnání histogramu. Snaha je dosáhnout přibližně rovnoměrného rozdělení jasu [24]. Díky tomu se zvýší kontrast snímku a využije se celá škála hodnot jasu pro popis snímku. Proces se provádí v kumulativním histogramu, kde se posouvají linie histogramu tak, aby byl výsledný kumulativní histogram přibližně lineární, viz. obrázek 14. Každý bod histogramu i je upraven pomocí následující rovnice:

$$H_{eq}(i) = \left(H(i) \cdot \frac{K-1}{MN} \right), \tag{3.1}$$

kde $H(i)$ je kumulativní histogram, M a N jsou rozměry obrázku a K je maximální hodnota jasu pixelů.



Obrázek 14: Princip vyrovnání histogramu, překresleno z [24]

(a) před ekvalizací (b) po ekvalizaci

Obrázek 15: Příklad snímků před a po ekvalizaci

V takto upraveném snímku videa je jednodušší nastavení prahování či odečítání pozadí, protože se video stane odolnějším vůči rychlým změnám osvětlení.

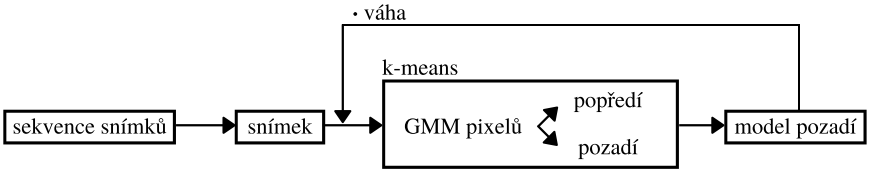
3.2 Odečítání pozadí

V aplikacích se statickou kamerou není náročné vytvořit model pozadí. Pokud máme statistický model pozadí, cizí objekty jsou detekovány jako části obrazu, které nejsou součástí statistického modelu.

Nejjednodušším způsobem modelování pozadí je zprůměrování všech snímků videa. Pokud se ve scéně objevuje velké množství objektů nebo se pohybují pomalu, je tento způsob málo robustní. Změny osvětlení scény představují v těchto aplikacích zásadní problém, protože po celou délku videa se používá jen jedna úroveň prahu [25].

3.2.1 Modelování pozadí pomocí GMM

Když se začaly používat modely typu GMM (Gaussian Mixture Model), modelem pozadí byl jen jeden Gaussián na každý pixel. Později [26] se každý pixel modeloval pomocí GMM. Podle podobností mezi Gaussiány napříč celým obrazem je možné odvodit, který Gaussián odpovídá pixelům pozadí a které hodnoty pixelů nesedí do modelovaného pozadí. Tyto pixely jsou vyhodnoceny jako popředí. S každou novou hodnotou pixelu se používá algoritmus k-means pro zjištění, jaké Gaussiány tento pixel definují. Nevytváří se nový GMM, pokud algoritmus vyhodnotí podobnost mezi předchozím a současným histogramem. Tím se sníží výpočetní náročnost a zajistí se stálost pozadí i při měnícím se osvětlení. Pro definování pozadí hledáme takové Gaussiány, které mají nejmenší rozptyl a častý výskyt. Gaussiány jsou následně srovnány podle pravděpodobnosti, že definují pozadí.



Obrázek 16: Schéma modelování pozadí pomocí GMM

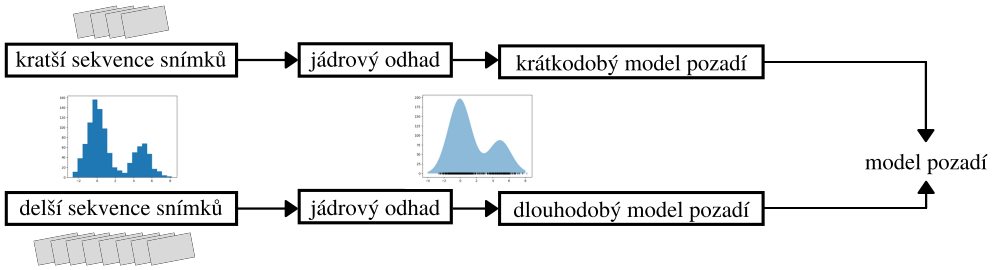
Podle [25] je navíc upraven dopad minulých GMM na modelování modelu současného snímku tak, že časově bližší minulé snímky mají větší váhu při modelování než časově vzdálenější minulé snímky. Váha modelů pozadí předchozích snímků exponenciálně klesá s časem. Schéma GMM modelování pozadí je na obrázku 16.

3.2.2 Modelování pozadí pomocí jádrového odhadu

Využití jádrového odhadu [27] je způsob modelování pozadí, který vznikl pro definování pozadí reálné scény. Při sledování přírody není pozadí statické a hodnota jeho pixelů se v čase velmi mění. Takto měnící se pixely nelze jednoduše popsat, metoda GMM v tomto ohledu selhává. Pokud se ale stane, že bude model pozadí zabírat příliš velkou šíři histogramu, nebude jednoduché detekovat objekty v popředí.

Jádrový odhad (KNN) je zobecněním metody GMM, pokud použijeme jako jádrovou funkci Gaussián. Tento Gaussián je váhovou funkcí, která je použita několikrát a sčítána, dokud nedostaneme vyhovující popis pozadí obrazu. Šířka pásma jádrové funkce je volena na základě odchylky jasu mezi následujícími snímky.

Používají se dva modely pozadí, krátkodobý a dlouhodobý. Krátkodobý model se rychle adaptuje na změny a je velmi citlivý na detekci popředí. Jeho nevýhodou je klasifikace změn v pozadí jako popředí. Dlouhodobý model se na změny adaptuje pomaleji, je vytvořen z delšího časového úseku videa. Průnik těchto modelů je výsledkem modelu pozadí. Navíc k průniku se počítají i pixely z krátkodobého modelu, které sousedí s pixely v dlouhodobém modelu. Jednoduché schéma modelu pozadí KNN je na obrázku 17.



Obrázek 17: Schéma modelování pozadí pomocí KNN

V [25] je navržena obdoba předešlé metody, kdy se mění šířka jádrové funkce tak, aby byl objem dat, který obsahuje, konstantní. Díky tomu není potřeba odhadovat ideální šířku jádrové funkce, mění se podle objemu dat.

Také by podle [25] měla být upravena aktualizace modelu pozadí. Na rozhodování o zařazení pixelu z popředí do modelu pozadí by se měl podílet jak krátkodobý, tak dlouhodobý model pozadí. Ne jako v [27], kde rozhoduje jen dlouhodobý model.

3.3 Prahování a detekce

Speciálním druhem kvantování obrazu je prahování. Rozdělí hodnoty jasu pixelů do dvou skupin podle úrovně prahu. Každé ze skupin je pak přiřazena hodnota pixelu, vznikne obraz, který obsahuje pouze dvě hodnoty jasu [24].

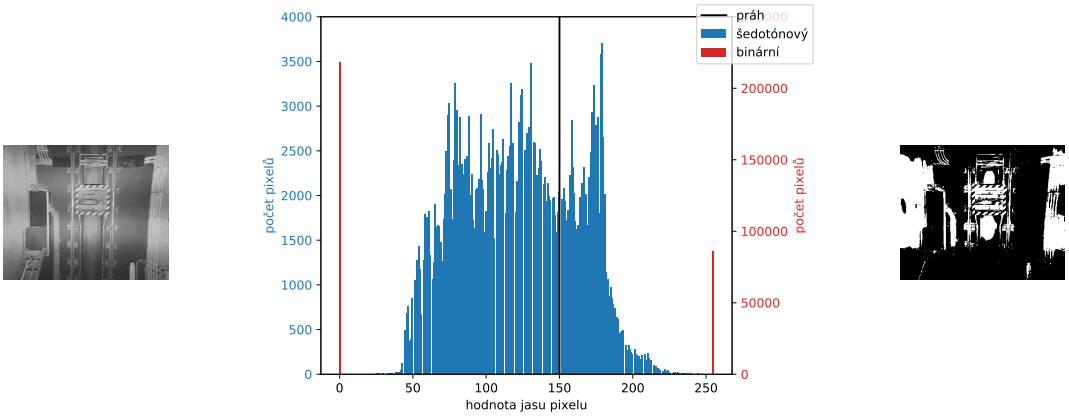
$$f_{prahovaci}(a) = \begin{cases} a_0 & \text{pro } a < a_{prah} \\ a_1 & \text{pro } a \geq a_{prah} \end{cases} \tag{3.2}$$

Pro detekci objektů v obraze se nejvíce hodí binarizace, speciální případ prahování, kdy skupinám pixelů přiřazujeme hodnoty 0 a 255. Tím vznikne černobílý obraz. Příklad prahování histogramu je na obrázku 18. Vlevo je šedotónový snímek před prahováním a vpravo je snímek po binárním prahování. Černá svislá čára je prahovací hodnota jasu. Všechny pixely nad touto hodnotou mají přiřazenou hodnotu 255 a všechny pixely pod hodnotou prahu mají hodnotu 0.

3.3.1 Morfologické operace

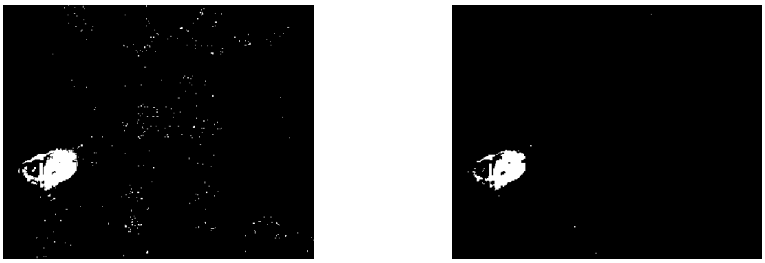
Po odečtení pozadí a prahování nemusí objekty v popředí vypadat ideálně. Často mají nesouvislou plochu a je náročné je rozpoznat od šumu. Základní morfologické operace jsou dilatace a eroze.

Dilatace je vektorový součet obrazu a strukturního elementu přes různé vzájemné pozice. Díky tomu objekty v obraze po obvodu zvětší svou plochu. Eroze je vektorový



Obrázek 18: Příklad prahování histogramu

rozdíl obrazu a pohybujícího se strukturního elementu. Výsledkem je smazání objektů s menší plochou než je element z obrazu a mírné zmenšení plochy velkých objektů. Díky tomu zůstanou v obraze jen objekty s větší plochou a malé objekty způsobené šumem zmizí. Příklad eroze je na obrázku 19. Kombinací eroze a dilatace získáme operace morfologické otevření a uzavření. Otevření je eroze a následně dilatace výsledku. Tím se odstraní malé plochy šumu a následně se zvětší velké objekty na původní plochu [28].

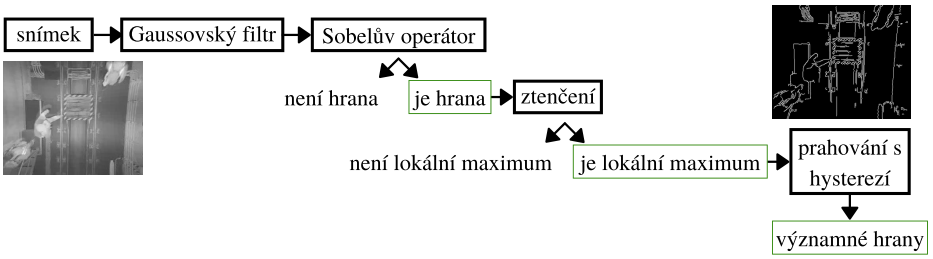


Obrázek 19: Příklad eroze obrazu

3.3.2 Cannyho hranový detektor

Místo prahování se pro binarizaci obrazu může použít detekce hran. Používá se velké množství hranových operátorů, ale nejvíce komplexním hranovým detektorem je Cannyho hranový detektor [29]. Tento detektor se skládá z několika po sobě jdoucích kroků. Nejprve se sníží šum obrazu použitím Gaussovského filtru, poté se hledá velikost a směr gradientu, na to se nejčastěji používá Sobelův operátor. Je aplikován v horizontálním i vertikálním směru. Dalším krokem je ztenčení, kdy se kontroluje každý detekovaný pixel vzhledem ke svému okolí ve směru gradientu, a pokud se nejedná o lokální maximum, je pixel odebrán z detekovaných hranových pixelů. Tím zajistíme detekci pixelů v místě největšího gradientu. Posledním krokem je prahování s hysterezí. Jsou zvoleny dvě prahovací úrovně gradientu jasu. Pokud má pixel vyšší hodnotu než oba prahy, je automaticky považován za hranu. Pokud pixel leží mezi prahy, je

detekován jako hranový, jen pokud sousedí s dříve určeným hranovým pixelem. Hodnoty pod oběma prahy nejsou hranami. Na obrázku 20 je znázorněno schéma postupu detekce hran.

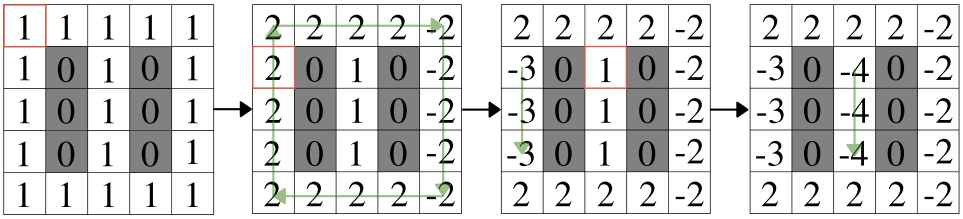


Obrázek 20: Schéma Cannyho hranového detektoru

3.3.3 Detekce kontur

Kontury jsou propojené okrajové body se stejnou úrovní jasu. Jejich vyhledávání v obraze umožňuje detekci objektů a zjištění tvaru objektů. Detekce kontur a stanovení jejich hierarchie se podle [30] provádí následovně (také viz obrázek 21):

1. Předpokládáme bílé objekty (hodnota pixelů 1) na černém pozadí (hodnota 0).
2. Po jednotlivých pixelech procházíme obraz a hledáme hranici mezi hodnotami 0 a 1, resp. 1 a 0. Pixel s hodnotou 1 na této hranici označíme jako začátek vnější hranice, resp. začátek vnitřní hranice.
3. Nalezené hranici přiřadíme jedinečné identifikační číslo.
4. Sledujeme nalezenou hranici a označujeme hraniční pixely. Podle algoritmu popsaném v [31] prohledáme okolní pixely začátku hranice. Pokud žádný z nich nemá hodnotu 1, hranice se nezaznamenává, jedná se o osamocený bílý pixel. Pokud nalezneme pixel s hodnotou 1, stává se následujícím bodem hranice a znovu prohledáváme jeho okolí.
5. Po označení celé hranice pokračujeme následujícím pixelem, kde jsme při hledání začátků hranic přestali.
6. Výsledkem jsou označené vnitřní a vnější hranice všech objektů obrazu.



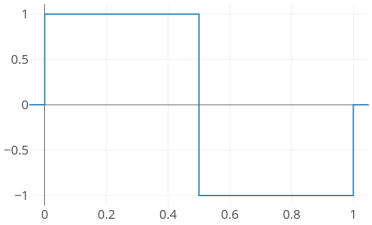
Obrázek 21: Postup detekce kontur, inspirováno [30]

3.4 Haarovy příznaky

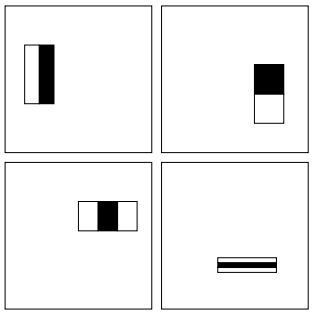
Pro zjednodušení reprezentace obrazu můžeme popsat části obrazu pomocí Haarových příznaků. Příznaky se používají kvůli snížení objemu dat oproti popisu pomocí pixelů.

Nejjednodušší funkce, používaná pro extrakci příznaků, je Haarova vlnka. Mateřská vlnka je vyjádřena vzorcem 3.3 a obrázkem 22a. Obdélníkové 2D vlnky, které budeme dále používat, jsou na obrázku 22b. Příznaky dělíme na dva základní druhy:

- První jsou složené ze dvou obdélníků o stejných rozměrech, které spolu jednou stranou sousedí. Tyto příznaky pomáhají detekovat hrany.
- Druhé jsou složeny ze tří stejných obdélníků, které spolu sousedí tak, že je černý obdélník obklopen bílými obdélníky. Tyto příznaky slouží k detekci linií.



(a) Mateřská vlnka



(b) Obdélníkové příznaky

Obrázek 22: Haarovy vlnky

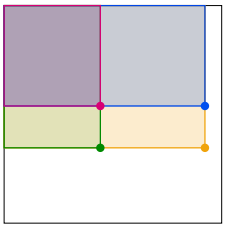
$$\psi(t) = \begin{cases} 1 & \text{pro } 0 \leq t < \frac{1}{2} \\ -1 & \text{pro } \frac{1}{2} \leq t < 1 \\ 0 & \text{jinde} \end{cases} \tag{3.3}$$

Příznak aplikujeme na určitou oblast tak, že sečteme hodnoty pixelů v bílých oblastech a černých oblastech a sumu z bílých oblastí odečteme od sumy z černé oblasti. Pomocí změny velikosti použitých obdélníků můžeme pro extrakci informace z obrazu vytvořit velké množství Haarových příznaků [32].

Kromě vodorovných a svislých obdélníků se můžou Haarovy příznaky vyskytovat také pootočené o 45°. Jejich použití podle [33] vylepšuje výsledky kaskádového klasifikátoru (viz kapitolu 3.5).

Hodnoty příznaků ze sumy pixelů v obdélnících lze spočítat velmi rychle pomocí integrálního obrazu. Integrální obraz se spočítá pomocí rovnice 3.4, jedná se o součet všech pixelů nalevo a nahoru od bodu na daných souřadnicích. Součet pixelů pod obdélníkem pak spočítáme pomocí kombinace integrálních obrazů pixelů v jeho vrcholech. Znázornění sumy pixelů obdélníku pomocí integrálního obrazu vrcholů je na obrázku 23.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{3.4}$$



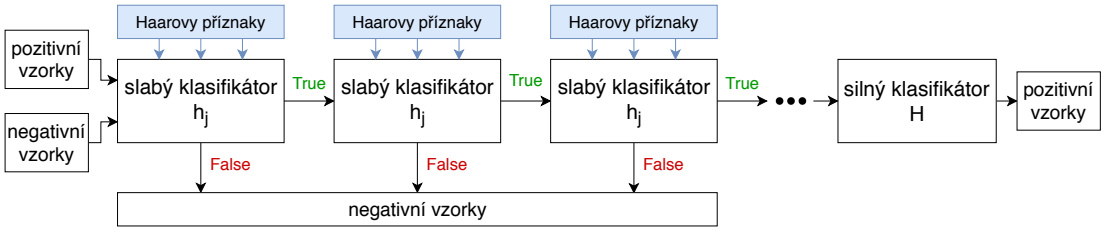
Obrázek 23: Integrální obrazy vrcholů obdélníku

3.5 Kaskádový klasifikátor

Kaskádový klasifikátor se používá pro klasifikaci objektů v obraze proto, že se pomocí kombinace malého počtu extrahovaných příznaků dosáhne efektivní klasifikace. Protože nepotřebujeme rozlišovat objekty mezi sebou, ale zajímá nás pouze existence osoby ve snímku, dělíme snímky do dvou skupin. Pozitivní vzorky jsou snímky, které obsahují osobu v záběru a negativní vzorky jsou snímky, které neobsahují osobu v záběru. Na každé úrovni kaskády je vybrán takový příznak, který nejlépe separuje pozitivní a negativní vzorky. Po nalezení takového příznaku je vytvořen první slabý klasifikátor h_j, který je definován svým příznakem f_j a prahem θ_j (3.5). Okno, se kterým procházíme snímek, je x. Často se používá rozměr okna 24x24, jak najdeme v [32].

$$h_j(x) = \begin{cases} 1 & \text{pro } f_j \leq \theta_j \\ 0 & \text{jinde} \end{cases} \tag{3.5}$$

Za sebe do kaskády postavíme několik slabých klasifikátorů, které mají úspěšnost detekce negativních vzorků kolem 50 % a úspěšnost detekce pozitivních vzorků blížící se 100 %. S větším počtem klasifikátorů se chybovost velkého klasifikátoru blíží k nule. Schéma zapojení je na obrázku 24, tato metoda se nazývá AdaBoost. Každý ze slabých klasifikátorů má vzhledem ke vstupním vzorkům určitou váhu, na základě chybovosti je váha klasifikátoru upravena, aby vznikl co nejsilnější klasifikátor.



Obrázek 24: Kaskádový klasifikátor AdaBoost

Pro zlepšení výsledků klasifikátoru se používá rozšíření knihovny negativních vzorků. To znamená, že se klasifikátor aplikuje na trénovací data a falešně pozitivní podokna ze snímků se použijí pro další trénování klasifikátoru. Tento silnější klasifikátor by měl méně chybovat a snížit hodnotu falešně pozitivně detekovaných vzorků. Trénování klasifikátoru může zabrat velké množství času, ale jeho aplikace na reální data je pak velice rychlá. Proto je kaskádový klasifikátor velmi vhodný pro detekci v reálném čase [32]. Výsledky klasifikátoru se zjistí jeho aplikací na testovací data, což je odlišná skupina obrázků oproti trénovací sadě.

3.6 Metriky úspěšnosti detekce

Nemůžeme pouze porovnávat počet detekovaných oken k celkovému počtu osob ve snímcích, protože nevíme, kdy se model trefil do správné polohy osoby a kdy se spletl. Detekci vyhodnocujeme porovnáním detekovaných objektů v obraze a jejich skutečnou polohou. Testovací snímky jsou rozřazené na pozitivní a negativní vzorky a díky tomu zjistíme, jaké jsou hodnoty pravdivě pozitivních (TP), falešně pozitivních (FP) a falešně negativních (FN) vzorků.

- TP = detekované okno odpovídá pozici osoby v testovacím snímku
- FP = detekované okno neodpovídá pozici osoby v testovacím snímku
- FN = pozice osoby v testovacím snímku není detekovaná

Z těchto metrik můžeme vypočítat další hodnoty, které vypovídají o kvalitě detekce. Jedná se o přesnost (Precision) a výtěžnost (Recall). Z nich získáme i F1 score, které kombinuje předešlé ukazatele. Pokud hledáme vyváženost hodnot Precision a Recall, hodnota F1 score je dobrým ukazatelem.

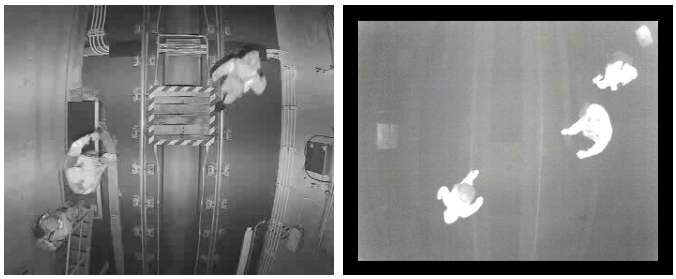
- $Precision = \frac{TP}{TP+FP}$ = jak se model trefuje do reálných osob, když detekuje
- $Recall = \frac{TP}{TP+FN}$ = jakou část reálných osob model dokáže detekovat
- $F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$ = rovnováha mezi Precision a Recall

4 Návrh a implementace algoritmu fúze

V této kapitole jsou vysvětleny všechny kroky algoritmu, které zajistí zvýšení informačního přínosu videa pomocí fúze. Nejprve se seznámíme s daty ze dvou kamer, které sledují odlišnou část spektra. Poté upravíme snímkové frekvence tak, aby si odpovídaly, pak následují kroky předzpracování, registrace a fúze obrazu.

4.1 Zpracování dat

Více kamerové systémy jsou používané v různých prostředích z bezpečnostních důvodů. Jedním z často používaných prostředí jsou i stanice metra [34]. Data, se kterými pracujeme, jsou pořízena na dvě zařízení: CMOS kameru pracující v blízkém IR pásmu (infrakameru) a termokameru. Na obrázku 25 jsou snímky z originálních záznamů.



(a) Infrakamera (b) Termokamera

Obrázek 25: Originální záznamy

Parametry záznamu z kamer jsou v tabulce 2. Protože se snímkové frekvence použitých záznamů velmi liší, prvním krokem je převzorkování videa na stejnou snímkovou frekvenci.

Zařízení	Rozlišení videa [px]	Snímková frekvence [fps]
Infrakamera	352 x 288	$< 10,01; 12,01 >$
Termokamera	352 x 288	$< 12,39; 25,00 >$

Tabulka 2: Parametry záznamu

4.1.1 Převzorkování videa

Aby měly záznamy z obou kamer stejnou snímkovou frekvenci, je potřeba je převzorkovat. K převzorkování jsem použila nástroj `FFmpeg`[1]. Tato platforma dokáže aplikovat základní úpravy na video, jako je změna velikosti, ořez, rotace nebo filtrace. Pomocí filtru `fps` je možné změnit vzorkovací frekvenci videa, ale jen tak, že se změní počet snímků ve videu. Není změněný obsah videa, proto na sebe i po převzorkování nemusí snímky z infrakamery a termokamery přesně obsahově sedět. Následující kód aplikujeme na všechna videa, která potřebujeme převzorkovat, zde je příklad pro infra video:

[1] Dostupné zde: http://ffmpeg.org/

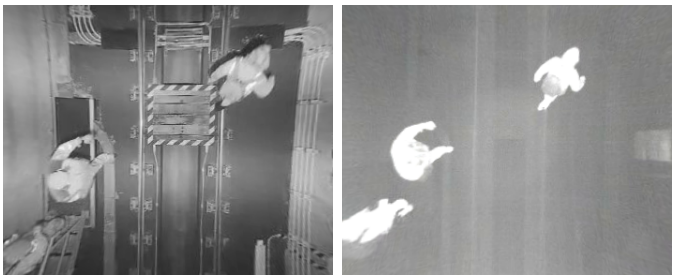
```
import ffmpeg
(
    ffmpeg
        .input('INFRA.avi')
        .filter('fps', fps=25, round='up')
        .output('INFRA-novy.avi')
        .run()
)
```

Používáme standardní vzorkovací frekvenci 25 snímků/s jako výslednou pro záznamy z obou kamer.

4.2 Registrace obrazu

Zásadní pro fúzi videí je, aby na sebe snímky z kamer seděly. Procesu napasování obrazů na sebe se říká registrace obrazu. Nejprve je potřeba videa časově srovnat, aby si snímky z obou kamer odpovídaly. Poté se řeší vada objektivu a transformace snímků. Časové zarovnání videí jsem provedla ručně, některá z použitých videí byla časově srovnána správně a bylo možné tento krok přeskočit. Dále jsem před řešením radiálního zkreslení provedla rotaci záznamu z termokamery o 180° pomocí funkce `imutils.rotate_bound`, která zároveň vyplnila plochu obrazu důležitým obsahem a odstranila černý rámeček. Tato transformace byla na první pohled jednoznačná a ulehčila porovnávání záznamů mezi sebou.

4.2.1 Korekce radiálního zkreslení



(a) Infrakamera (b) Termokamera

Obrázek 26: Snímky po korekci radiálního zkreslení

Jako první byly srovnány vady objektivů. Obě kamery mají soudkovité zkreslení obrazu, které bylo korigováno funkcí `cv2.undistort` [35]. Do této funkce vstupují parametry objektivu a koeficienty zkreslení. V matici vstupní kamery $A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ jsou hodnoty ohniskové vzdálenosti (f_x, f_y) a optického středu (c_x, c_y). Koeficienty radiálního zkreslení jsou $(k_1, k_2, \ldots, k_n)$, které získáme z rovnice zkreslených souřadnic:

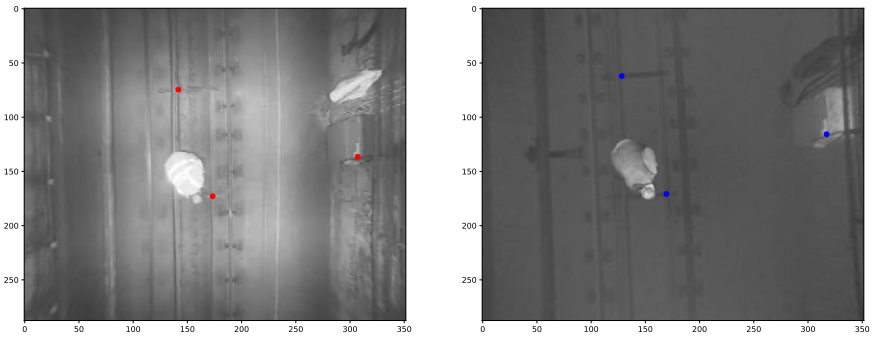
$$\begin{aligned} x_{zkresleny} &= x(1 + k_1 r^2 + k_2 r^4 + \ldots + k_n r^{2n}) \\ y_{zkresleny} &= y(1 + k_1 r^2 + k_2 r^4 + \ldots + k_n r^{2n}). \end{aligned} \tag{4.1}$$

Pro infrakameru jsem zvolila korekční koeficient $k_4 = 7 \cdot 10^{-4}$ a pro termokameru jsem zvolila korekční koeficient $k_4 = 8 \cdot 10^{-4}$. Výsledné obrazy po korekci jsou na

obrázku 26. Ve všech záběrech jsou použité stejné kamery, proto je tato korekce použita na všechna videa, se kterými pracujeme.

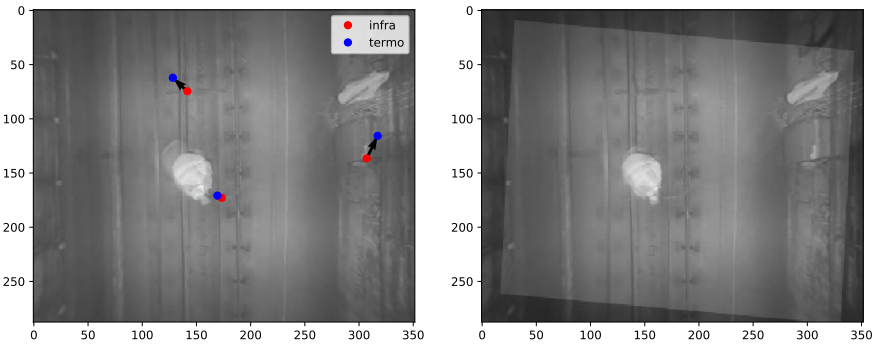
4.2.2 Afinní transformace

Dalším krokem je zarovnat obrazy kamer na sebe, k tomu použijeme afinní transformaci na základě odpovídajících si bodů v obrazech. Bylo by možné použít i projektivní transformaci, ale po vyřešení radiálního zkreslení je pro zarovnání obrazů potřeba jen translace a rotace, není potřeba měnit projekci scény. Nejprve se ve snímku z infra kamery zvolí 3 body a poté se stejné body označí v termo snímku.



Obrázek 27: Snímky se zvolenými body pro transformaci

Na základě těchto 3 dvojic bodů funkce `cv2.getAffineTransform` [35] vytvoří transformační matici. Na obrázku 27 jsou vidět dva pohledy na stejnou scénu a červené a modré body jsou výchozími body pro transformaci. Na obrázku 28 je na 28a jejich posun při překrytí původních obrázků a na 28b překrytí obrázků po transformaci.



(a) před transformací (b) po transformaci

Obrázek 28: Překrytí snímků a posun bodů transformace

Výsledek na obrázku 28b je překrytí snímků takovým způsobem, že si pixely z obou snímků odpovídají. Na takto zkalibrovaná videa se aplikuje fúze obrazu. Uložené hodnoty transformace se aplikují na všechna videa ze stejného prostředí.

4.3 Fúze obrazu

Na správně registrované obrazy je ideální použít fúzi. Při fúzi obrazu si ceníme zachování co největšího množství informace z obou původních obrazů. V této kapitole budou popsány algoritmy tří druhů fúze obrazu. Srovnání jejich účinnosti pak naleznete v tabulkách 5 a 6, na straně 47. Prvním krokem u všech druhů fúze je převod videí z barevného prostoru do odstínů šedi. Abychom vytvořili video záznam pomocí fúze dvou záznamů, aplikujeme následující algoritmy na každý snímek videa.

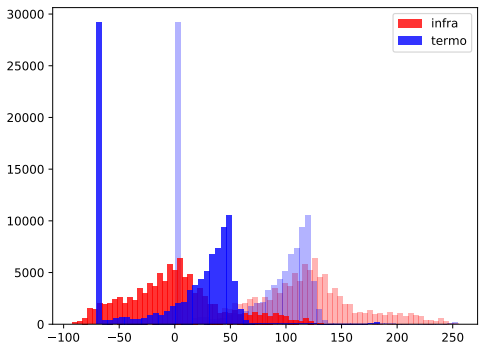
4.3.1 Analýza hlavních komponent

Abychom získali obraz vytvořený fúzí na základě analýzy hlavních komponent (PCA), vložíme dva šedotónové obrázky do následující funkce:

```
def fusion_pca(im1,im2):
    im1flat = im1.flatten('F')
    im2flat = im2.flatten('F')
    im1flat = im1flat - np.mean(im1flat)
    im2flat = im2flat - np.mean(im2flat)
    C = np.cov([im1flat,im2flat])
    d,v = np.linalg.eig(C)

    if (d[0] >= d[1]):
        pca = v[:,0]/sum(v[:,0])
    else:
        pca = v[:,1]/sum(v[:,1])
    imf = pca[0]*im1 + pca[1]*im2
    fusedImage = imf.astype(np.uint8)
    return fusedImage
```

Nejdřív ze vstupních obrazů vytvoříme sloupcové vektory a odečteme průměrnou hodnotu snímku. Tím zařídíme, aby byly hodnoty jasu kolem hodnoty 0. Histogramy obou snímků před a po normalizaci jsou na obrázku 29. Veliké množství černých pixelů se nachází v termo snímku proto, že při afinní transformaci se černými pixely doplňují místa, která se nachází mezi transformovaným snímkem a rozměry referenčního snímku.



Obrázek 29: Histogramy infra a termo snímků před normalizací (méně syté) a po normalizaci

(a) (b) (c)

Obrázek 30: Vstupní snímky a výstupní obraz PCA fúze

Dalším krokem algoritmu je výpočet kovarianční matice `C`, vlastních vektorů `v` a vlastních čísel `d`. Porovnáme velikost vlastních čísel a vlastní vektor, který patří k většímu vlastnímu číslu, použijeme jako váhovací koeficienty. Koeficienty vynásobíme původní 2D obrazy a ty pak sečteme. Na obrázku 30c je výsledek fúze.

4.3.2 Vlnková transformace

Pro získání obrazu obrazu pomocí vlnkové transformace (DWT) je nejdříve potřeba se rozhodnout, kterou funkci použijeme jako mateřskou vlnku. Nejjednodušší vlnkou, která se pro výpočet DWT používá je Haarova vlnka.

Fúzi vlnkových koeficientů je možné provádět pomocí tří variant. Buď to z koeficientů vezmeme ten menší, ten větší nebo jejich průměr. K tomu slouží následující funkce, kterou použijeme později:

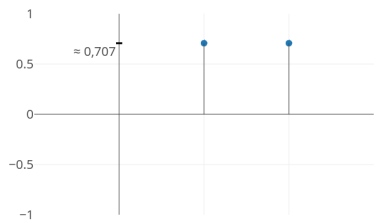
```
def fuseCoeff(cooef1, cooef2, method):
    if (method == 'mean'):
        cooef = (cooef1 + cooef2) / 2
    elif (method == 'min'):
        cooef = np.minimum(cooef1,cooef2)
    elif (method == 'max'):
        cooef = np.maximum(cooef1,cooef2)
    else:
        cooef = []
    return cooef
```

Algoritmus fúze obrazu pomocí DWT začíná tím, že vložíme dva šedotónové obrazy do následující funkce:

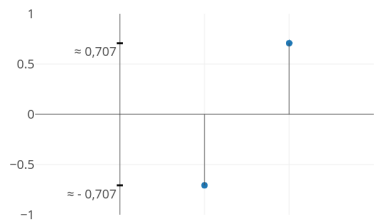
```
def fusion_wavelet(im1,im2):
    FUSION_METHOD = 'mean' # 'min' / 'max'
    cooef1 = pywt.wavedec2(im1[:,:], 'db1')
    cooef2 = pywt.wavedec2(im2[:,:], 'db1')

    fusedCooef = []
    for i in range(len(cooef1)):
        if(i == 0):
            fusedCooef.append(fuseCoeff(cooef1[0],cooef2[0], FUSION_METHOD))
        else:
            c1 = fuseCoeff(cooef1[i][0],cooef2[i][0], FUSION_METHOD)
            c2 = fuseCoeff(cooef1[i][1], cooef2[i][1], FUSION_METHOD)
            c3 = fuseCoeff(cooef1[i][2], cooef2[i][2], FUSION_METHOD)
            fusedCooef.append((c1,c2,c3))
    fusedImage = pywt.waverec2(fusedCooef, 'db1')
    fusedImage = np.multiply(np.divide(fusedImage - np.min(fusedImage),
    (np.max(fusedImage) - np.min(fusedImage))),255)
    fusedImage = fusedImage.astype(np.uint8)
    return fusedImage
```

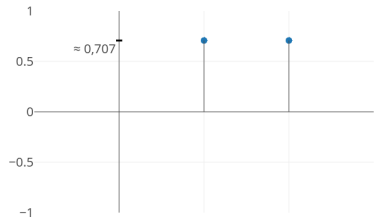
Na začátku napíšeme druh volby koeficientů do proměnné `FUSION_METHOD`, a pak následuje rozklad obrazu `im1` a `im2` pomocí 2D DWT s Haarovou mateřskou vlnkou. K rozkladu používáme balíček `PyWavelet` [36], který má funkci pro víceúrovňovou 2D dekompozici. V první úrovni dekompozice se obraz dělí do 4 podpásem pomocí kombinace filtrů H a L, znázorněných na obrázku 31a a 31b.



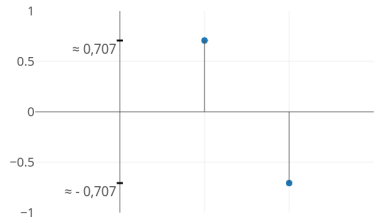
(a) Dolní propust pro dekompozici



(b) Horní propust pro dekompozici



(c) Dolní propust pro rekonstrukci



(d) Horní propust pro rekonstrukci

Obrázek 31: Kvadraturně zrcadlové filtry

Dále v cyklu přes všechny koeficienty provádíme fúzi, tím kombinujeme obrazy na stejných úrovních dekompozice. Pomocí koeficientů po fúzi provádíme zpětnou rekonstrukci obrazu funkcí `waverec2`, která aplikuje 2D inverzní DWT na všechny úrovně

dyadické dekompozice a poté obrazy zkombinuje. Pro rekonstrukci se každá úroveň filtruje pomocí kvadraturně zrcadlových filtrů (viz obrázky 31c a 31d). Posledním krokem je normalizace výsledného obrazu.

Na obrázku 32 jsou obrazy po fúzi při různé volbě fúze koeficientů ve funkci `fuseCoeff`. Je vidět, že při volbě nejmenších (32a) nebo největších (32b) koeficientů pro fúzi na výsledném obrazu vzniknou nežádoucí artefakty. Proto pro další využití DWT fúze obrazu používáme průměr koeficientů (32c). Originální snímky pro fúzi jsou 30a a 30b.

(a) minimum (b) maximum (c) průměr

Obrázek 32: Výstupní obrazy DWT fúze

4.3.3 Laplaciánská pyramidová fúze

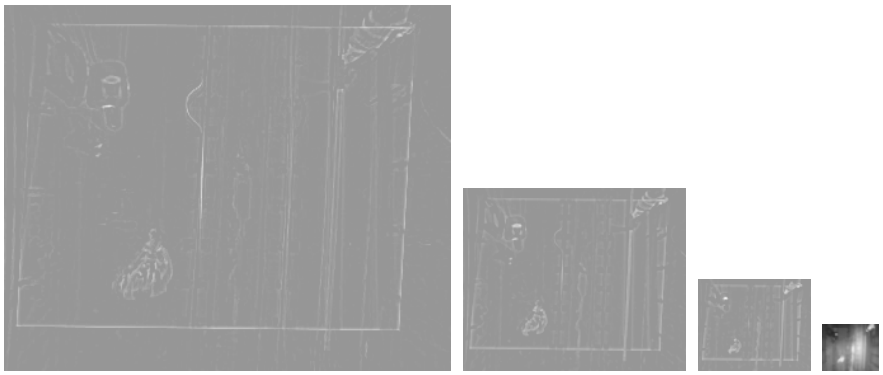
Posledním algoritmem fúze, který jsem implementovala, je Laplaciánská pyramidová fúze (LaP). Následující kód přesně odpovídá krokům, popsaným v kapitole 2.3:

```
def fusion_lap(im1,im2):
    G = im1
    gpA = [G]
    for i in range(6):
        G = cv2.pyrDown(G)
        gpA.append(G)
    G = im2
    gpB = [G]
    for i in range(6):
        G = cv2.pyrDown(G)
        gpB.append(G)
    lpA = [gpA[5]]
    for i in range(5,0,-1):
        GE = cv2.pyrUp(gpA[i])
        L = cv2.subtract(gpA[i-1],GE)
        lpA.append(L)
    lpB = [gpB[5]]
    for i in range(5,0,-1):
        GE = cv2.pyrUp(gpB[i])
        L = cv2.subtract(gpB[i-1],GE)
        lpB.append(L)
    LS = []
    for la,lb in zip(lpA,lpB):
        ls = np.maximum(la,lb)
        LS.append(ls)
    ls_ = LS[0]
    for i in range(1,6):
        ls_ = cv2.pyrUp(ls_)
        ls_ = cv2.add(ls_, LS[i])

    fusedImage = ls_
    fusedImage = np.multiply(np.divide(fusedImage -
        np.min(fusedImage),(np.max(fusedImage) -
        np.min(fusedImage))),255)
    fusedImage = fusedImage.astype(np.uint8)
    return fusedImage
```

Do funkce vstupují dva šedotónové snímky, ze kterých pak pomocí funkce `cv2.pyrDown` [35] vytvoříme rozklad do Gaussovské pyramidy. Následuje vytvoření Laplaciánské pyramidy pomocí funkcí `cv2.pyrUp` a `cv2.subtract`, které snímek určité úrovně zvětší a odečtou od snímku předchozí úrovně. Dále se snímky na všech úrovních porovnávají a maximální hodnoty vytvoří nové snímky. Příklad Laplaciánské pyramidy po fúzi je

na obrázku 33, kde je pro lepší přehlednost zvýšený jas a pouhé 4 úrovně pyramidy. V reálu jsem aplikovala rozklad do 6 úrovní.



Obrázek 33: Laplaciánská pyramida po fúzi před rekonstrukcí

Pro rekonstrukci je potřeba použít funkci `cv2.pyrUp` a postupně úrovně sčítat. Následuje jen normalizace výsledného obrazu a fúze je hotová. Na obrázku 34c je výsledek LaP fúze z originálních obrázků 34a a 34b.

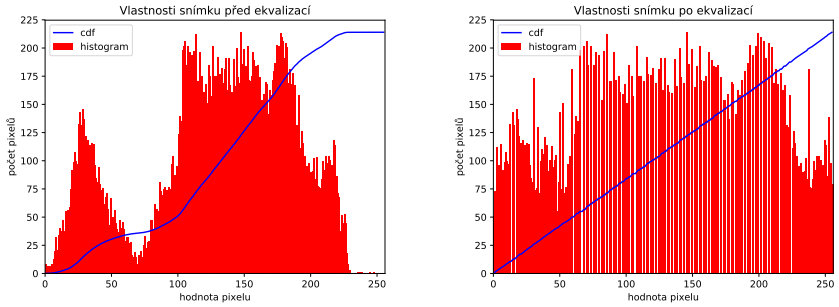
(a) (b) (c)

Obrázek 34: Vstupní snímky a výstupní obraz LaP fúze

5 Návrh a implementace algoritmu detekce

Pro detekování objektů v obraze existuje velké množství algoritmů, které se velmi liší svým využitím a výpočetní náročností. V této kapitole je nejdříve popsána nejjednodušší detekce pomocí odečítání pozadí a v další části je vysvětlena extrakce Haarových příznaků a aplikace kaskádních filtrů.

Vstupní videa mají často problém s šumem nebo s nestabilním osvětlením. Dopady měnícího se osvětlení se dají zmírnit pomocí vyrovnání histogramu, ideální je funkce `cv2.equalizeHist()` [35]. Na obrázku 35 je příklad histogramu reálné scény před a po vyrovnání. Odečítání pozadí a všechny další kroky jsem aplikovala na videa bez a s vyrovnáním histogramu. Pomocí aplikace detekce na problematické video z hlediska jasu jsem se ujistila, že videa s vyrovnanými histogramy jsou pro detekci spolehlivější (viz obrázek 40).



Obrázek 35: Příklad vyrovnání histogramu, zobrazení kumulativní distribuční funkce

5.1 Detekce pomocí odečítání pozadí

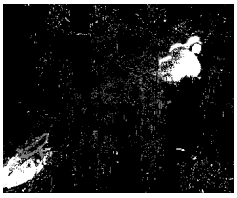
Prvním způsobem detekce objektů v obraze je odečítání pozadí, prahování a vyhledávání hran či kontur. Pro odečítání pozadí z obrazu je možné použít několik algoritmů. Zde popíšu a porovnám dvě funkce, které se často používají. První funkcí je `cv2.createBackgroundSubtractorMOG2()` [35], která pro modelování pozadí využívá GMM (viz kapitolu 3.2.1). Druhá je funkce `cv2.createBackgroundSubtractorKNN()` [35], která pro modelování pozadí využívá jádrový odhad (KNN) (viz kapitolu 3.2.2). Funkce nejdříve vytvoří soubor, který obsahuje 3 proměnné:

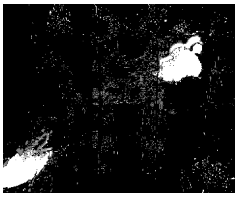
- počet snímků, ze kterých se bude model pozadí vytvářet
- prahovací úroveň pro vzdálenost pixelu od modelu pozadí
- možnost detekovat stíny.

Funkce odečítání pozadí se aplikuje na snímek videa v cyklu. Výstupem je prahovaný černobílý obraz pro variantu bez detekce stínů a při detekci stínů navíc s šedou barvou. Na obrázku 36 je vidět originální snímek a výsledky po odečtení pozadí a prahování obou zmíněných funkcí.

Pokud porovnáme výsledek odečtení modelu pozadí GMM a KNN, vidíme, že jádrový odhad je o trochu robustnější, co se týče neustálých změn osvětlení. V následujících odstavcích jsou ukázány aplikace hledání hran a kontur, také vysvětlím, proč je hledání


(a) Originál


(b) Modelování GMM

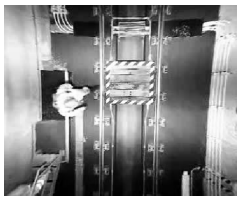

(c) Modelování KNN

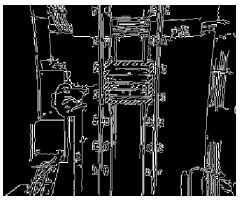
Obrázek 36: Vstupní snímek a obrazy po odečtení pozadí

hran méně vhodnou variantou detekce v našich obrazových datech oproti hledání kontur.

Hledání hran

Cannyho hranový operátor použijeme pomocí funkce `cv2.Canny()` [35]. Výsledek hledání hran je na obrázku 37. Výstupem funkce je obrázek, jehož pixely jsou prahované, černé značí pozadí a bílé značí hrany. Operátor funguje spolehlivě, ale detekuje i šum v pozadí. Protože nemáme o ohraničených objektech žádné další informace, nemůžeme s výsledným obrazem dále manipulovat. Obrázek 37c vznikl při použití modelování pozadí pomocí jádrového odhadu a po dalších krocích pro eliminaci šumu. Tyto kroky jsou na obrázku 38 před hledáním kontur.


(a) Originální snímek


(b) Detekce hran


(c) Detekce osob

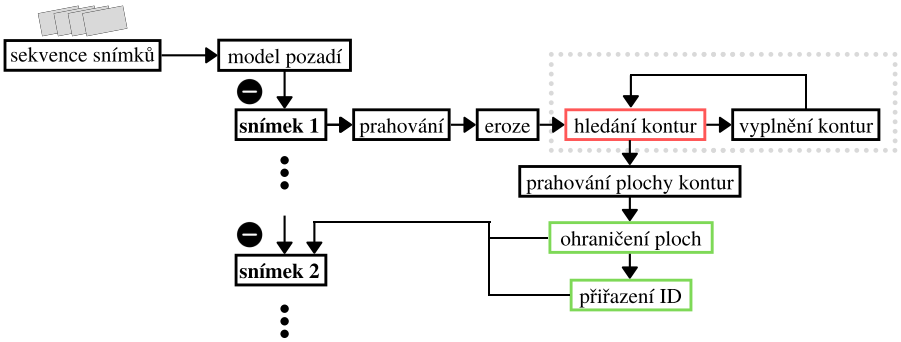
Obrázek 37: Porovnání detekce bez a s vyrovnáním histogramu

Při použití hranového operátoru na originální obraz ztratíme informaci o jasu objektu a nelze poznat, která část je popředí a která pozadí. Také není možné vytvořit po detekci hran model pozadí, protože se změnou osvětlení většina detekovaných hran změní svojí polohu.

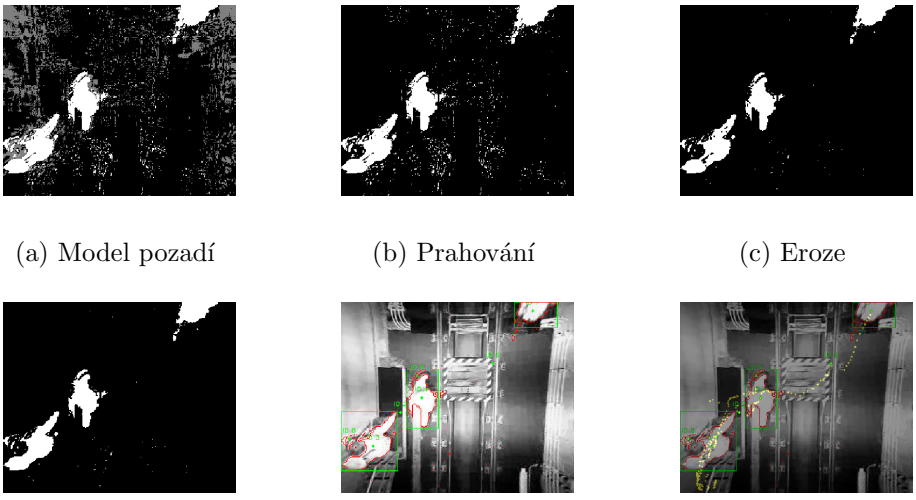
Další postup je znázorněn schématem na obrázku 38 a příkladem snímků na obrázku 39. Abychom z předchozích snímků odečetli stíny, použijeme takové prahování, že se ze šedých pixelů stanou černé. Pro binarizaci snímku se hodí funkce `cv2.threshold()` [35]. I po odstranění stínů je scéna zašumělá, proto použijeme erozi pomocí strukturního elementu. Zvolíme si např. element o velikosti (2x2) a pokud v obraze narazíme na místo, kde se pod celým jádrem nenachází světlé pixely, odstraníme je. Tím se proměnlivého šumu zbavíme.

Hledání kontur

Hledání kontur je pro detekci osob výhodnější, protože jsou obvody objektů spojité a dají se pro další potřeby vyplnit. Další výhodou je, že výstupem z funkce pro



Obrázek 38: Schéma odečítání pozadí, detekce a sledování objektů



(a) Model pozadí (b) Prahování (c) Eroze

(d) Vyplnění kontur (e) Detekce (f) Trajektorie

Obrázek 39: Příklady snímků z jednotlivých kroků detekce objektů

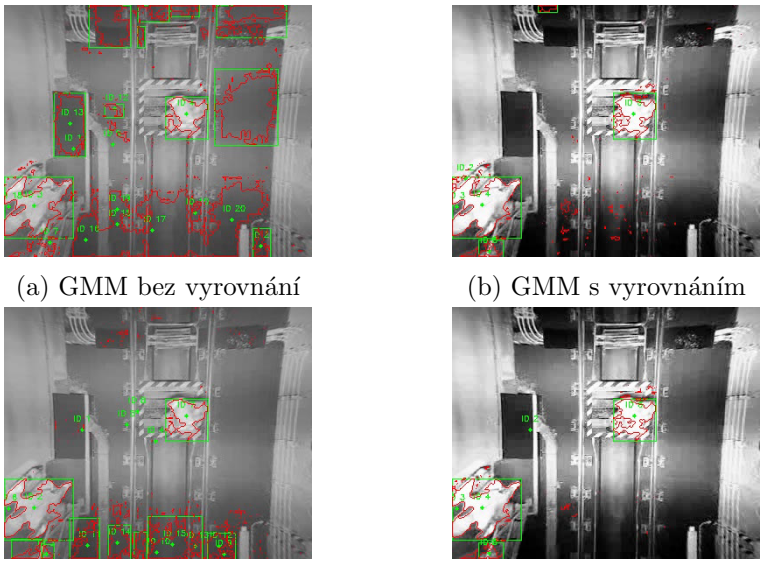
hledání kontur jsou souřadnice těchto kontur a lze k nim přistupovat odděleně, nebo zjistit jejich plochu a polohu. Pro vysvětlení dalších kroků detekce využijeme hledání kontur.

Binární obraz po erozi obsahuje objekty s nevyplněnými konturami, proto je pomocí kombinace příkazů `cv2.findContours` a `cv2.drawContours` najdeme a vyplníme. Ve výsledném obraze pak vyhledáme kontury o ploše, která odpovídá velikosti osoby v záběru. Veškeré nalezené kontury jsou v obraze zaznamenány červeně a ty o vhodné ploše ohraničíme zeleným obdélníkem.

Každý obdélník sledujeme ze snímku na snímek pomocí funkce `CentroidTracker()` [37]. Díky této funkci můžeme přiřadit každému objektu vlastní ID a sledovat ho v průběhu videa. Trajektorie objektu je zaznamenávání středu těchto objektů, v obraze 39f žlutě. Sledování geometrických středů objektů (centroidů) má několik kroků:

- definování obdélníků kolem detekovaných objektů a jejich centroidů
- přiřazení ID

- přenos informací do dalšího snímku
- definování obdélníků kolem detekovaných objektů a jejich centroidů v novém snímku
- výpočet vzdálenosti mezi novými a starými centroidy
- přiřazení nejbližších centroidů k původním ID
- identifikace nových centroidů



(a) GMM bez vyrovnání (b) GMM s vyrovnáním

(c) KNN bez vyrovnání (d) KNN s vyrovnáním

Obrázek 40: Porovnání detekce bez a s vyrovnáním histogramu

Na obrázku 40 je srovnání detekce osob ve videu po fúzi bez vyrovnání histogramu a s vyrovnáním. Jedná se o snímek se skokovou změnou osvětlení. Je vidět, že ve snímku před vyrovnáním je chybně detekováno i osvětlení, proto všechny detekce s modelováním pozadí provádíme na videích s vyrovnanými histogramy.

5.2 Detekce pomocí kaskádového klasifikátoru

Kaskádový klasifikátor není tak složité implementovat a používat, ale použitá data můžou velmi ovlivnit výsledek detekce. Abychom mohli později porovnávat mezi zvolenými parametry, vytvoříme více klasifikátorů natrénovaných na různých datech. V další části kapitoly vysvětlím trénování a testování klasifikátorů.

5.2.1 Vytvoření trénovacích a testovacích dat

Pro soubor dat, používaných u učícího se algoritmu, je zásadní jejich počet. K práci jsem měla dostupnou celou databázi videí, proto nebyl problém s počtem snímků. Celkový počet dat jsem zvolila s ohledem na časovou náročnost trénování a testování klasifikátoru. Pro trénování jsem použila vždy 1000 pozitivních vzorků a přibližně 1100

negativních vzorků. Pro testování jsem použila 628/728 snímků, které obsahují alespoň 1 osobu k detekci.

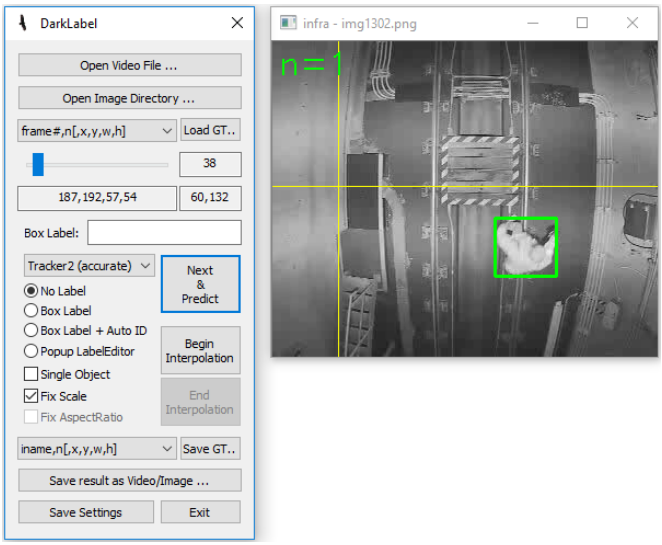
Trénovací data

První databází, kterou je potřeba vytvořit, je skupina trénovacích dat. Jedná se o snímky či výstřižky snímků, pomocí kterých naučíme klasifikátor rozdíly mezi osobami a pozadím. Protože budeme vytvářet 5 klasifikátorů, potřebujeme 5 skupin trénovacích dat. Tyto skupiny jsou:

- originální záznamy z infrakamery
- originální záznamy z termokamery
- videa po DWT fúzi
- videa po PCA fúzi
- videa po LaP fúzi

Nejprve z videozáznamů vytvoříme snímky a ty roztřídíme na pozadí a na snímky s osobami. Snímky s částí postavy jsem z trénovací sady vyřadila. Snímky pozadí jsou připravené a tvoří skupinu negativních vzorků pro trénování. Ze snímků s postavami potřebujeme vytvořit databázi pozitivních vzorků, která obsahuje jen ořezy snímků, ohraničující osobu.

Pro vytvoření výřezů snímků s postavami jsem použila software DarkLabel[1], který umožňuje označit a sledovat objekty v sérii snímků (nebo i ve videu). Následně je možné uložit několika způsoby zápis zaznamenaných hodnot do `.txt` souboru. Zvolila jsem variantu, která je celkem kompatibilní s opencv funkcí pro vytváření vzorků ke klasifikaci. Program uloží název snímku, počet detekovaných objektů, hodnotu pixelu v levém horním rohu ohraničujícího obdélníku a jeho šířku a výšku. Ukázka rozhraní je na obrázku 41.



Obrázek 41: Rozhraní programu DarkLabel

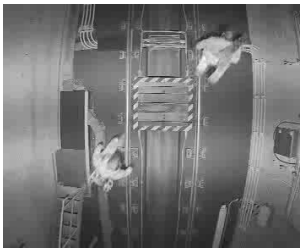
[1]Dostupné zde: https://darkpgmr.tistory.com/16

Soubor s těmito informacemi vložíme do funkce pro vytvoření databáze pozitivních vzorků `opencv_createsamples` [35]. Kromě informací o snímcích do této funkce vkládáme i další parametry: umístění výstupního `.vec` souboru, počet snímků, který má funkce vytvořit, a rozměry výstupního snímku. Zvolila jsem 1000 pozitivních snímků o výšce 24 pixelů a šířce 24 pixelů.

Testovací data

Dále je potřeba vytvořit databázi testovacích dat. Skupina trénovacích a testovacích dat se nesmí překrývat, protože by nám vyšly nepravdivé výsledky. Tuto databázi použijeme k porovnání výsledků detekce všech klasifikátorů. Vytvoříme dvě skupiny testovacích dat, originální záznamy z infrakamery a z termokamery.

Znovu vytvoříme z videí seznam snímků a vybereme ty snímky, které obsahují v záznamu osoby. Pomocí programu DarkLabel označíme polohu osob ve snímku a soubor s názvy obrázků a polohou detekce uložíme. Nemáme tedy negativní a pozitivní vzorky, ale v každém snímku alespoň jeden pozitivní vzorek. Na obrázku 42 je ukázka snímku z testovací databáze infra (42a) a termo (42b).



(a) infra

(b) termo

Obrázek 42: Ukázka snímků z testovací databáze

5.2.2 Trénování klasifikátoru

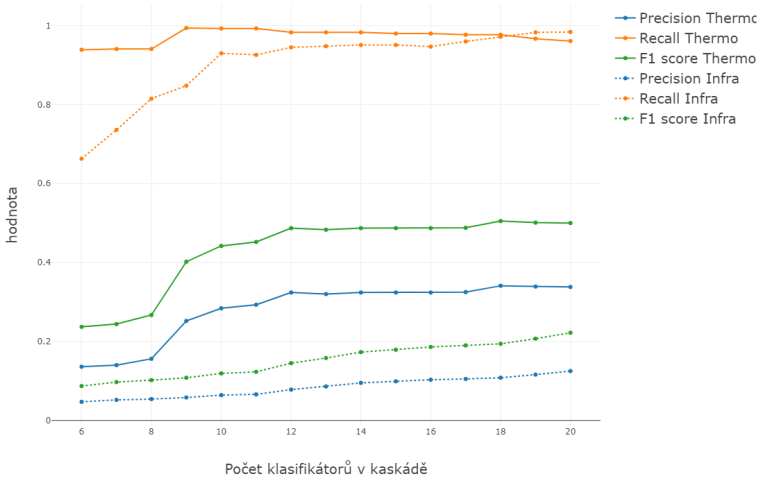
Pomocí sady pozitivních a negativních vzorků se trénuje kaskáda klasifikátoru. Použila jsem funkci `opencv_traincascade` [35]. Použité parametry jsou v tabulce 3. Ne všechny ze zmíněných parametrů jsou povinné, např. parametr -mode ALL umožňuje použít Haarovy vlnky pootočené o 45°, jak jsem se zmiňovala v kapitole 3.4.

Parametr	-data	-vec	-bg	-numPos	-numNeg
Hodnota	G:/DP/cascade_Infra	outinfra.vec	bg.txt	1000	1153
Parametr	-w	-h	-mode	-minHitRate	-numStages
Hodnota	24	24	ALL	0.999	20

Tabulka 3: Parametry funkce `opencv_traincascade`

Udělala jsem test, jehož výsledkem je informace o tom, jaký počet klasifikátorů v kaskádě nám dává nejlepší výsledky. Varianty od 6 do 20 klasifikátorů jsem podrobila testu na testovacích datech. Zajímaly mě hodnoty přesnosti (Precision) a výtěžnosti (Recall). Z nich pak získáme i hodnotu F1 score. V grafu 43 je porovnání Infra klasifikátoru a Termo klasifikátoru a jejich výsledků na testovacích vzorcích. Protože je

našim cílem detekovat co největší počet osob, je zásadnější vysoká hodnota Recall. Hodnotu Precision se pokusím snížit pomocí přetrénování.



Obrázek 43: Porovnání počtu klasifikátorů v kaskádě

Vylepšení klasifikátoru se dělá pomocí přidání falešně pozitivních vzorků do databáze negativních vzorků. Falešně pozitivní vzorky získáme z aplikace klasifikátoru na trénovací data. Špatně detekovaná místa v obraze oříznememe a vložíme do databáze negativních vzorků. Příklady falešně pozitivních vzorků jsou na obrázku 44. V tabulce 4 jsou parametry pro přetrénování klasifikátoru. Přestože nám z grafu 43 vyšlo, že kolem hodnoty 9 klasifikátorů probíhá skokové zvýšení hodnoty Recall, budeme nový model trénovat na 18 slabých klasifikátorů. I když to vypadá, že nepřichází zásadní zlepšení žádné z hodnot, při tomto počtu je nejvyšší hodnota F1-skóre u Termo klasifikátoru a zároveň hodnota Recall u Infra klasifikátoru je nad hodnotou 97%.

Obrázek 44: Falešně pozitivní snímky po prvním trénování

Parametr	-data	-vec	-bg	-numPos	-numNeg
Hodnota	G:/DP/cascade_Infra2	outinfra.vec	bg2.txt	1000	1195
Parametr	-w	-h	-mode	-minHitRate	-numStages
Hodnota	24	24	ALL	0.999	18

Tabulka 4: Parametry funkce `opencv_traincascade` pro přetrénování

5.2.3 Testování klasifikátoru

Klasifikátory se testují aplikováním na testovací data. V následujícím kódu je ukázka, jak aplikovat klasifikátor. Kaskáda se načte pomocí funkce `cv2.CascadeClassifier` [35] a následně se použije na snímek funkcí `.detectMultiScale`. Parametry této funkce jsem experimentálně nastavila podle velikosti osob v záběru.

```
full_cascade = cv2.CascadeClassifier('G:/DP/cascade.xml')
f = open("G:/DP/testovaci/gt.txt", "r")
for line in f:
        words = line.split()
        img = cv2.imread(words[0])
        full = full_cascade.detectMultiScale(img, scaleFactor = 1.05,
        minNeighbors = 20, minSize = (30,30), maxSize = (70,70))
```

V pokračování kódu je zobrazení modrých čtverců okolo všech detekovaných oblastí klasifikátorem a zobrazení červených čtverců okolo všech reálných osob v testovacích datech. Pomocí euklidovské vzdálenosti porovnáme vzdálenosti středů čtverců detekovaných a reálných a pro vzdálenost menší než 20 px označíme detekci za pravdivou. V obrázku 45 je vidět i označení správné detekce pomocí zeleného středu.

Obrázek 45: Ukázka testování klasifikátoru

Porovnáním detekovaných osob, celkového počtu osob na snímku a falešných detekcí získáme hodnoty TP, FP a FN, které použijeme pro výpočet Precision, Recall a F1 score. Pomocí těchto metrik porovnáme 5 klasifikátorů a 2 databáze testovacích snímků.

6 Výsledky

V následující části práce jsou veškeré výsledky a porovnání jednotlivých metod fúze obrazu. V další části jsou výsledky týkající se detekce, kdy ukážeme výsledky detektorů a srovnáme výsledky detekce pomocí hledání kontur v obraze a pomocí kaskádového klasifikátoru. Následně porovnáme na výsledcích detekce všechny druhy fúze. V závěru se zaměříme na porovnání klasifikátorů, které jsou trénované na datech po fúzi, bude nás zajímat jejich výkon při detekci v surových datech z infra a termo kamery.

6.1 Porovnání výsledků fúze

Kvalitu fúze porovnáme podle metrik, vysvětlených v kapitole 2.4. Tabulka 5 ukazuje výsledky metrik kvality fúze pro příklad snímku bez osob v záběru.

Druh fúze	RMSE	PSNR	SSIM	CE	FMI
PCA	$30,83$	$18,35$	$\mathbf{8,61 \cdot 10^{-1}}$	$1,03 \cdot 10^{-3}$	$1,54$
DWT	$\mathbf{28,95}$	$\mathbf{18,90}$	$8,59 \cdot 10^{-1}$	$\mathbf{2,62 \cdot 10^{-4}}$	$\mathbf{1,83}$
LaP	$37,60$	$16,63$	$7,44 \cdot 10^{-1}$	$4,58 \cdot 10^{-4}$	$1,48$

Tabulka 5: Porovnání kvality fúze obrazu bez osob

Hodnoty všech metrik jsou celkem vyrovnané, nemůžeme říct, že by se jedna z metod velmi odlišovala od ostatních. V dalším porovnávání se zaměříme pouze na index SSIM, protože zachování struktury obrazu je pro nás nejdůležitější metrikou. Na obrázku 47a je srovnání druhů fúze na základě 218 snímků bez osob. Je vidět, že LaP fúze má u všech snímků horší hodnoty SSIM než metody DWT a PCA.

(a) PCA (b) DWT (c) LaP

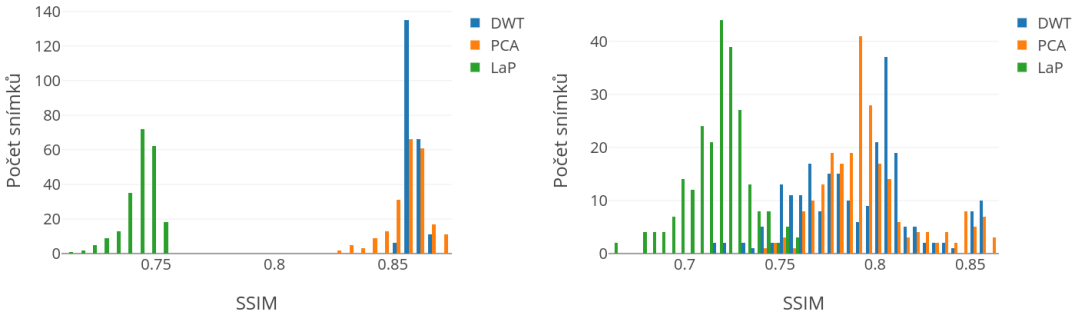
Obrázek 46: Příklad scény s osobou

Protože se výsledky registrace a fúze lépe porovnávají se snímku s objekty, dále jsem aplikovala metriky na vzorek snímku s osobou v záběru. Tabulka 6 ukazuje výsledky metrik kvality fúze pro příklad snímku s osobou v záběru. Použité snímky jsou na obrázku 46.

Druh fúze	RMSE	PSNR	SSIM	CE	FMI
PCA	$\mathbf{30,66}$	$\mathbf{18,40}$	$7,94 \cdot 10^{-1}$	$\mathbf{1,22 \cdot 10^{-5}}$	$1,40$
DWT	$36,20$	$16,96$	$\mathbf{8,05 \cdot 10^{-1}}$	$6,84 \cdot 10^{-5}$	$1,51$
LaP	$39,67$	$16,16$	$7,20 \cdot 10^{-1}$	$6,25 \cdot 10^{-5}$	$\mathbf{1,55}$

Tabulka 6: Porovnání kvality fúze obrazu v přítomnosti osoby v záběru

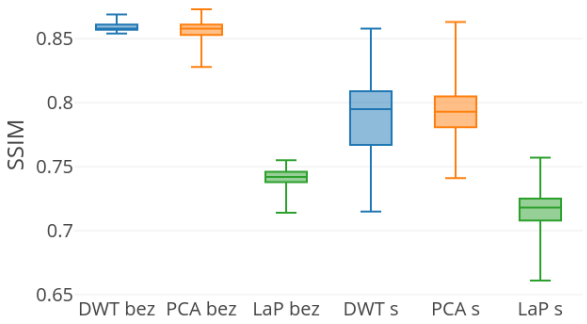
Výsledky jednoho snímku nejsou dostatečně vypovídající, proto jsem se zaměřila na SSIM index u 241 snímků, které mají v záběru osobu. Výsledky jsou v histogramu na obrázku 47b. Oproti snímkům bez osob jsou zde hodnoty SSIM průměrně nižší pro všechny metody fúze. Z toho usuzuji, že s přítomností pohybujícího se objektu ve videu se mohla projevit špatná časová synchronizace originálních záznamů. Osoby ve snímku mají vždy vyšší hodnoty jasu než okolí, což se ve fúzi projeví tak, že když registrace obrazů není dokonalá, plocha osoby se oproti originálním snímkům zvětší.



(a) pro scény bez osob (b) pro scény s osobami

Obrázek 47: Histogram indexu SSIM

Na obrázku 48 je krabicový diagram pro porovnání mediánů a rozptylů hodnot indexu SSIM pro snímky bez osob a s osobami v záběru. Zajímavým zjištěním je, že fúze pomocí LaP nemá tak výrazné zhoršení SSIM při porovnání snímků s osobami a bez osob oproti DWT a PCA, i když má stále nejnižší hodnoty. Rozptyly u se u všech metod fúze rozšířily, což se dalo vzhledem k velké variabilitě snímků s osobami předpokládat.



Obrázek 48: Rozdělení hodnot fúzí **bez** nebo **s** osobami v záběru

6.2 Porovnání výsledků detekce

Obsahem této kapitoly je popsání úspěšnosti detekce dvou druhů vyhledávání objektů v obraze. Důležitou součástí je i aplikace vytvořených algoritmů na původní data, abychom vyhodnotili přínos fúze multispektrálních dat a neporovnávali jen druhy fúze mezi sebou.

6.2.1 Výsledky hledání kontur

Pro algoritmus hledání kontur, kterému předchází odečítání pozadí a prahování, používám snímky s vyrovnaným histogramem. V kapitole 5.1 je vysvětleno na příkladu, proč nepoužíváme původní data. Tato úprava pomůže zmírnit proměnlivost osvětlení, která je v použitých videích vysoká.

Algoritmus odečítání modelu pozadí, prahování, hledání kontur a sledování centroidů jsem aplikovala na 5 sérií vzorků. Na snímky stejné scény původních záběrů z infrakamery a termokamery jsem aplikovala 3 druhy fúze: PCA, DWT a LaP. Snímkům po fúzi jsem vyrovnala histogramy. Původní snímky a vyrovnané obrázky po fúzi tvoří testovací databáze pro porovnání úspěšnosti detekce osob na různých datech. Testovací databáze mají cca 600 osob k detekci, v jednom snímku jsou najednou maximálně 3.

V tabulce 7 jsou výsledné hodnoty Precision, Recall a F1 score pro modelování pozadí metodou GMM. Výsledky jsou velmi podobné variantě s jádrovým odhadem, jen s o trochu lepšími hodnotami Recall. O žádné z úprav obrazu nemůžeme říct, že by byla výrazně lepší k detekci než ostatní. V tabulce 8 jsou výsledné hodnoty Precision, Recall a F1 score pro modelování pozadí jádrovým odhadem. Celkově jsou výsledky dost neuspokojivé. Na všech databázích je celkem slušná hodnota Recall, ale v aplikacích, jako je videozáznam osob z bezpečnostních kamer, je zásadní detekovat téměř všechny osoby, které se v záběru pohybují. Proto často ani Recall = 90% není uspokojivá hodnota. Když k tomu navíc přidáme extrémně nízkou hodnotu Precision, což hodnoty pod 1% určitě jsou, závěrem je velmi špatný detektor.

Testovací data	Infra	Termo	DWT	PCA	LaP
Precision	0,057	0,044	0,086	0,074	0,094
Recall	0,953	0,862	0,909	0,923	0,898
F1	0,107	0,084	0,156	0,137	0,171

Tabulka 7: Výsledky detekce na testovacích databázích, metoda GMM

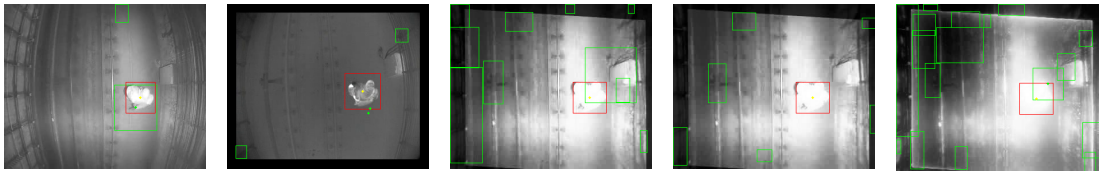
Testovací data	Infra	Termo	DWT	PCA	LaP
Precision	0,076	0,044	0,091	0,084	0,092
Recall	0,932	0,835	0,871	0,838	0,812
F1	0,142	0,083	0,165	0,153	0,166

Tabulka 8: Výsledky detekce na testovacích databázích, metoda jádrového odhadu

Když navíc přihlédneme k faktu, že je zohledněna velikost plochy objektu, kterou hledáme, je detekce opravdu špatná. V tomto způsobu detekce nevidím žádný přínos dělat fúzi multispektrálních dat, protože se musí řešit artefakty vzniklé různým

osvětlením v obou snímcích. Následné vyrovnání histogramu sice pomůže se změnami jasu, ale zavede do obrazu rušivé kompresní artefakty.

Na obrázku 49 je ukázka detekce na snímku ze stejného okamžiku zachyceného na infra a termokameru, ze kterých je třemi druhy fúze vytvořen nový obraz. Červeně jsou ohrazena pravdivá místa osob v testovacích datech a zelené obdélníky ohraničují detekci pomocí odečítání modelu pozadí GMM. Je vidět, že fúze vysokou úroveň jasu v infra snímku ještě umocní a detekce je náročnější.



Obrázek 49: Příklad detekce na databázích, zleva Infra, Termo, DWT, PCA, LaP

6.2.2 Výsledky kaskádového klasifikátoru

Vytvořila jsem 5 klasifikátorů, které jsem trénovala na 5 databázích. Databáze obsahují přibližně stejné snímky, 2 sady obrázků jsou z původních záznamů z infra a termokamery a 3 sady trénovacích obrázků jsou z dat po fúzi.

Testovací sady dat jsou dvě: původní snímky z infrakamery a původní snímky z termokamery. Na snímky z infrakamery vyzkoušíme detekci pomocí klasifikátoru natrénovaného na infra snímcích a a porovnáme výsledky s klasifikátory trénovanými na datech po fúzi. Tím zjistíme, jestli sjednocení snímků do multispektrálních dat pomáhá lepší detekci či nikoli. Stejným způsobem porovnáme fúzní klasifikátory s termo klasifikátorem na datech z termokamery.

První trénování

V tabulkách 9 a 10 jsou výsledky hodnot Precision, Recall a F1 score pro klasifikátory po prvním natrénování, aplikované na původní záznamy z infra a termokamery. V prvním řádku je název sady trénovacích dat pro daný klasifikátor. Tato označení budu dále v textu používat pro zjednodušení. Protože pro naši aplikaci je zásadní rozpoznat co nejvyšší počet osob v záběru, není nejvyšší hodnota F1 score to, co hledáme. Pokud bychom např. nastavili jako hraniční hodnotu Recall 90%, LaP klasifikátor by z variant úplně vypadl.

Klasifikátor	Infra	DWT	PCA	LaP
Precision	0,108	**0,245**	0,181	0,229
Recall	0,972	0,964	**0,998**	0,691
F1	0,194	**0,391**	0,306	0,344

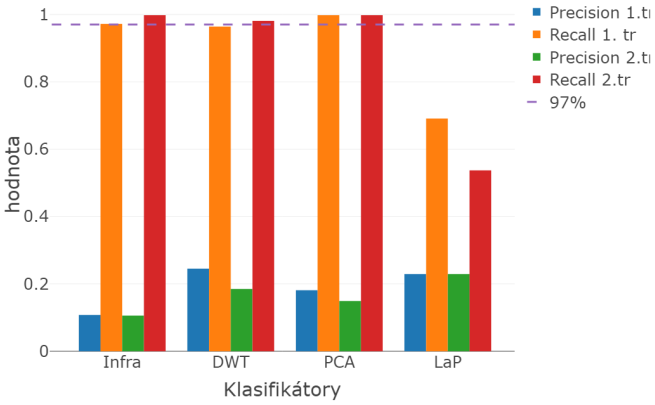
Tabulka 9: Porovnání klasifikátorů aplikovaných na Infra snímky

Z hodnot je vidět, že už po prvním trénování má PCA klasifikátor na infra datech lepší hodnoty všech ukazatelů než samotný Infra klasifikátor. To je pro nás zásadní zjištění, protože fúze obrazových dat přináší novou informaci do původních dat. Podobné výsledky jsou i na termo datech, ale ne s tak velkým rozdílem. Na

Klasifikátor	Termo	DWT	PCA	LaP
Precision	0, 341	**0, 363**	0, 293	0, 312
Recall	0, 977	0, 937	**0, 978**	0, 582
F1	0, 505	**0, 523**	0, 451	0, 407

Tabulka 10: Porovnání klasifikátorů aplikovaných na Termo snímky

obrázku 50 je grafické znázornění zmíněných parametrů a zvýrazněná hranice 97%, jako pomyslné zadání, které musí klasifikátor splnit.



Obrázek 50: Parametry detekce na testovacích datech Infra

Přetrénování

Pro zvýšení hodnoty Precision, která je po prvním trénování celkem nízká, jsem data přetrénovala s přidanými vzorky falešně pozitivních obrázků do databáze negativních vzorků. Takto jsem vylepšila všech 5 klasifikátorů a znovu jsem je aplikovala na původní záznamy z infra a termokamery. V tabulkách 11 a 12 jsou výsledky hodnot Precision, Recall a F1 score.

Klasifikátor	Infra	DWT	PCA	LaP
Precision	0, 106	0, 185	0, 149	**0, 229**
Recall	0, 998	0, 981	**0, 998**	0, 537
F1	0, 192	0, 312	0, 259	**0, 321**

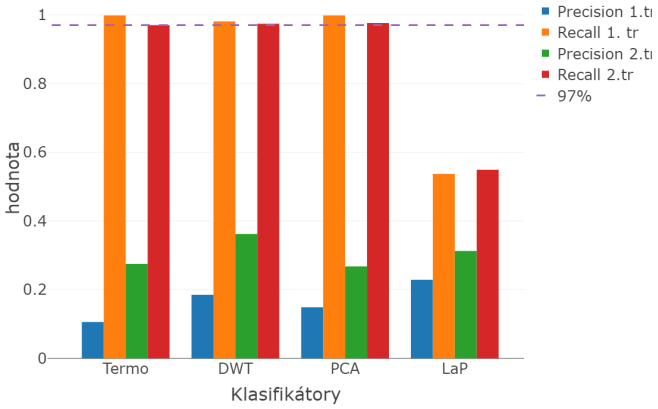
Tabulka 11: Porovnání klasifikátorů aplikovaných na Infra snímky

K mému překvapení se hodnoty Precision vůbec nezlepšily, někde dokonce zhoršily. To ukazuje na náročnou odlišitelnost osob od pozadí v infra snímcích. U všech klasifikací kromě LaP klasifikátoru se hodnoty Recall ještě zlepšily. Hodnoty F1 score zůstaly téměř beze změny kvůli tomu, že se parametry Recall a Precision měnily proti sobě. Tučné označení nejvyšších hodnot u LaP tedy o ničem nevypovídá, protože se hodnoty ostatních klasifikátorů oproti prvnímu trénování zhoršily, jen u LaP se příliš nezměnily.

Klasifikátor	Termo	DWT	PCA	LaP
Precision	0, 275	**0, 362**	0, 268	0, 313
Recall	0, 970	0, 974	**0, 976**	0, 549
F1	0, 429	**0, 528**	0, 421	0, 399

Tabulka 12: Porovnání klasifikátorů aplikovaných na Termo snímky

U termo snímků jsou výsledky podobné, zde setrvaly nebo se snížily hodnoty všech parametrů kromě Recall u DWT klasifikátoru, který po přetrénování překonal hodnotu 97%. Na obrázcích 50 a 51 je porovnání klasifikátorů před (1.tr) a po přetrénování (2.tr).



Obrázek 51: Parametry detekce na testovacích datech Termo

Pokud bych měla zhodnotit výsledky dohromady na obou testovacích databázích, zvolila bych PCA klasifikátor na Infra datech po prvním trénování. Hodnota Recall 99.8% je výborná a počet falešně pozitivních detekcí lze dále snížit. Například by se dalo pracovat s celým videem a nejen se snímky, protože většina falešně pozitivních oken se ve snímku nepohybuje, ale osoby ano. Na obrázku 52 je ukázka stejného snímku z infrakamery s detekovanými osobami různými klasifikátory po prvním trénování. Na tomto konkrétním snímku má nejlepší výsledek DWT klasifikátor, ale přes celou testovací sadu si vede lépe PCA klasifikátor.

(a) Infra klasifikátor (b) DWT klasifikátor (c) PCA klasifikátor (d) LaP klasifikátor

Obrázek 52: Ukázka detekce na Infra testovacím snímku

Závěr

Práce se zabývá možnostmi fúze multispektrálních dat a následnou detekcí osob v obraze. Vstupními daty jsou záběry stejné scény z CMOS kamery pracující ve viditelném a blízkém infračerveném spektru (Infra) a z termokamery (Termo). Povedlo se implementovat tři druhy fúze obrazu: PCA (analýza hlavních komponent), DWT (Vlnková transformace s dyadickou dekompozicí) a LaP (Laplaciánská pyramidová fúze). Pro detekci osob byly použity dva algoritmy: první je odečítání modelu pozadí a vyhledávání kontur a druhý je kaskádový klasifikátor na bázi Haarových vlnek.

Běžně používané metriky pro hodnocení kvality fúze neměly výrazně odlišné výsledky ani pro jeden z druhů fúze obrazu. Po zaměření se na index SSIM vyšly hodnoty u LaP fúze horší oproti PCA a DWT, a to jak u snímků bez osob v záběru, tak u snímků s osobami v záběru. U snímků s osobami v záběru klesl medián indexu SSIM u všech druhů fúze a zvýšil se jeho rozptyl.

Další možností, jak porovnat fúze mezi sebou, je jejich vhodnost pro detekci osob ve snímku. Proto byla vytvořena testovací data s uloženými reálnými pozicemi osob, které umožňují kontrolovat úspěšnost detekce. Dva druhy modelování pozadí byly použity na 5 databází snímků (Infra, Termo, DWT, PCA, LaP) a následně se porovnaly hodnoty Recall a Presicion dané detekce. Hodnoty Recall se vždy dostaly nad 80%, ale extrémně nízké hodnoty Precision ($< 1\%$) dělají z této metody detekce téměř nepoužitelnou aplikaci pro použitá data. Jednotlivé fúze mají mezi sebou podobné výsledky, stejně jako aplikace detekce na původní záznamy. Tyto výsledky jsou zapříčiněny velmi nestabilními jasovými podmínkami prostředí a skokovými změnami jasu při vstupu objektu do záběru, kterým nezabránilo ani vyrovnání histogramu.

Lepší výsledky dává aplikace kaskádového klasifikátoru. Nejprve bylo vytvořeno 5 databází trénovacích snímků (Infra, Termo, DWT, PCA, LaP) a 2 databáze testovacích snímků (Infra a Termo z předchozího testování). Pomocí trénovacích dat jsem vytvořila 5 kaskádových klasifikátorů algoritmem AdaBoost. Na testovacích datech jsem zjistila jejich úspěšnost. Všechny klasifikátory mají celkem nízké hodnoty Precision, což značí obrovský počet falešně pozitivních vzorků. Metoda fúze LaP se ukázala jednoznačně jako horší než PCA a DWT jak při aplikaci na Infra testovací data, tak na Termo testovací data. U detekce osob z bezpečnostních záběrů je zásadní hodnota ukazatele Recall, proto se na ni zaměříme při porovnávání úspěšnosti detekce. Metoda PCA dokonce na zdrojových datech překonala klasifikátor na těchto datech natrénovaný. Má lepší výsledky v porovnání s Infra (resp. Termo) klasifikátorem při aplikaci na Infra (resp. Termo) testovací data (viz. tabulku 13).

Testovací data \ Klasifikátor	Infra	Termo	PCA
Infra	0,998	–	0,998
Termo	–	0,970	0,976

Tabulka 13: Porovnání hodnot Recall u klasifikátorů

V porovnání s Infra klasifikátorem je hodnota Recall u PCA identická, ale hodnota Precision je u PCA vyšší. V porovnání s Termo klasifikátorem je hodnota Recall u PCA klasifikátoru vyšší. To znamená, že klasifikátor natrénovaný na datech po fúzi přidává zásadní informaci k detekci ve zdrojových datech z kamer.

Abych ještě vylepšila výsledky detekce, aplikovala jsem klasifikátor na trénovací data a přidala jsem do databáze negativních trénovacích vzorků falešně pozitivní vzorky detekce. Poté jsem znovu natrénovala 5 klasifikátorů na rozšířených datech a otestovala jejich úspěšnost na testovacích databázích. Na Infra snímcích se vylepšily hodnoty Recall u všech klasifikátorů kromě LaP, ale vůbec se nezlepšily hodnoty Precision, u kterých jsem očekávala zvýšení. Podobné výsledky měla i aplikace na Termo data. V tomto případě přetrénování klasifikátoru nevedlo k lepším výsledkům detekce.

Další rozšíření práce vidím ve vylepšování klasifikátorů:

- Dalo by se pracovat s polohou falešně pozitivních vzorků, které jsou často v rámci video sekvence neměnné oproti pohybujícím se osobám.
- Databáze negativních trénovacích snímků by mohla obsahovat pouze výřezy ze snímků a ne celé snímky pozadí.
- K rozšíření databáze negativních vzorků (např. PCA klasifikátoru) by se mohly použít výřezy z Infra snímků a vznikl by hybridní klasifikátor odolný vůči falešným detekcím v testovacích datech.

Možným vylepšením samotného algoritmu fúze obrazu by mohla být změna parametrů fúze na základě lepšího výsledku indexu SSIM. Např. počet úrovní dyadické dekompozice nebo Laplaciánské pyramidy v závislosti na SSIM.

Seznam použitých zkratek

CE Cross Entropy.

DWT Discrete Wavelet Transform.

FMI Fusion Mutual Information.

FN False Negative.

FP False Positive.

GaP Gaussian Pyramid.

GMM Gaussian Mixture Model.

KNN K-nearest neighbours.

LaP Laplacian Pyramid.

PCA Principal Component Analysis.

PSNR Peak Signal to Noise Ratio.

RMSE Root Mean Square Error.

SSIM Structural Similarity.

TP True Positive.

Seznam obrázků

Seznam tabulek

Literatura

1. KRISHNAMOORTHY, Shivsubramani; SOMAN, KP. Implementation and comparative study of image fusion algorithms. *International Journal of Computer Applications*. 2010, roč. 9, č. 2, s. 25–35.
2. DESHMUKH, Manjusha; BHOSALE, Udhav. Image fusion and image quality assessment of fused images. *International Journal of Image Processing (IJIP)*. 2010, roč. 4, č. 5, s. 484.
3. SUMATHI, M; BARANI, R. Qualitative evaluation of pixel level image fusion algorithms. In: *International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012)*. 2012, s. 312–317.
4. BENEŠ, Jiří; KYMPLOVÁ, Jaroslava; VÍTEK, František. *Základy fyziky pro lékařské a zdravotnické obory: pro studium i praxi*. Grada Publishing as, 2015.
5. GOSHTASBY, A.; NIKOLOV, S. G. Guest editorial: Image fusion: Advances in the state of the art. *Information Fusion*. 2007, roč. 8 (2), s. 114–118. Publisher: Elsevier.
6. POHL, C.; GENDEREN, J. L. Van. Multisensor image fusion in remote sensing: Concepts, methods and applications. *International Journal of Remote Sensing*. 1998, roč. 19, č. 5, s. 823–854.
7. MASOOD, Saleha; SHARIF, Muhammad; YASMIN, Mussarat; SHAHID, Muhammad; REHMAN, Amjad. Image Fusion Methods: A Survey. *Journal of Engineering Science and Technology Review*. 2017, roč. 10, s. 186–195.
8. JIANG, Dong; ZHUANG, Dafang; HUANG, Yaohuan; FU, Jinying. Survey of Multispectral Image Fusion Techniques in Remote Sensing Applications. In: ZHENG, Yufeng (ed.). *Image Fusion and Its Applications*. Rijeka: IntechOpen, 2011, kap. 1.
9. ZHANG, Y. Understanding image fusion. *Photogrammetric Engineering and Remote Sensing*. 2004, roč. 70.
10. LINDSAY, I Smith; SMITH, A. A tutorial on principal components analysis. In: *Cornell University*. 2002, sv. 51, s. 52.
11. CARRASCO-OCHOA, J.A.; MARTINEZ-TRINIDAD, J.F.; RODRIGUEZ, J.S.; BAJA, G.S. di. *Pattern Recognition: 5th Mexican Conference, MCPR 2013, Queretaro, Mexico, June 26-29, 2013. Proceedings*. Springer Berlin Heidelberg, 2013. Lecture Notes in Computer Science. ISBN 9783642389894.
12. NAIDU, VPS; RAOL, Jitendra R. Pixel-level image fusion using wavelets and principal component analysis. *Defence Science Journal*. 2008, roč. 58, č. 3, s. 338–352.
13. ANISIMOVA, E; BEDNÁŘ, J; PÁTA, P. Zpracování obrazu pomocí vlnkové transformace. *Electro revue*. 2013.
14. ČÍKA, Petr. *Multimédia*. Multimédia [online]. Vysoké učení technické v Brně: Vysoké učení technické v Brně, 2014.
15. ADELSON, E. H.; SIMONCELLI, E. P.; HINGORANI, R. Orthogonal pyramid transforms for image coding. In: *Proc. SPIE, Visual Comm. and Image Proc. II*. Cambridge, MA, 1987, sv. 845, s. 50–58.

16. RYŠÁNEK, JAN. Filtrace signálů EKG s využitím vlnkové transformace. *Dipl. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ*. 2012.

17. SABRE, Rachid; WAHYUNI, Ias SRI I. S. Wavelet Decomposition in Laplacian Pyramid for Image Fusion. *International Journal of Signal Processing Systems*. 2016, roč. 4, č. 1, s. 37–44.

18. BURT, Peter J.; ADELSON, Edward H. The Laplacian Pyramid as a Compact Image Code. *IEEE TRANSACTIONS ON COMMUNICATIONS*. 1983, roč. 31, s. 532–540.

19. COLORES, Juan; GARCÍA-VÁZQUEZ, Mireya; RAMIREZ, Alejandro; PEREZ-MEANA, Hector; NAKANO-MIYATAKE, M. Video Images Fusion to Improve Iris Recognition Accuracy in Unconstrained Environments. In: 2013, sv. 7914, s. 114–125. ISBN 9783642389887. Dostupné z DOI: 10.1007/978-3-642-38989-4_12.

20. CVEJIC, Nedeljko; ŁOZA, Artur; BULL, David; CANAGARAJAH, Nishan. A similarity metric for assessment of image fusion algorithms. In: *International Journal of Signal Processing*. 2005.

21. STATHAKI, Tania. *Image Fusion: Algorithms and Applications*. Orlando, FL, USA: Academic Press, Inc., 2008. ISBN 0123725291.

22. WANG, Zhou; BOVIK, Alan C.; SHEIKH, Hamid R.; SIMONCELLI, Eero P. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*. 2004, roč. 13, č. 4, s. 600–612.

23. PATIL, U.; MUDENGUDI, U. Image fusion using hierarchical PCA. In: *2011 International Conference on Image Information Processing*. 2011, s. 1–6.

24. BURGER, W.; BURGE, M.J. *Digital Image Processing: An Algorithmic Introduction Using Java*. Springer London, 2009. Texts in Computer Science. ISBN 9781846289682.

25. Z.ZIVKOVIC; HEIJDEN, F.van der. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*. 2006, roč. 27, č. 7, s. 773–780.

26. STAUFFER, C.; GRIMSON, W. E. L. Adaptive background mixture models for real-time tracking. In: *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. 1999, sv. 2, 246–252 Vol. 2. ISSN 1063-6919.

27. ELGAMMAL, Ahmed; HARWOOD, David; DAVIS, Larry. Non-parametric Model for Background Subtraction. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2000, s. 751–767.

28. GONZALEZ, Rafael C.; WOODS, Richard E. *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN 013168728X.

29. CANNY, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1986, roč. PAMI-8, č. 6, s. 679–698. ISSN 0162-8828.

30. SUZUKI, Satoshi; BE, KeiichiA. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*. 1985, roč. 30, č. 1, s. 32–46. ISSN 0734-189X.

31. ROSENFELD, Azriel. Connectivity in Digital Pictures. *J. ACM*. 1970, roč. 17, č. 1, s. 146–160. ISSN 0004-5411.

32. VIOLA, Paul; JONES, Michael et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*. 2001, roč. 1, s. 511–518.

33. LIENHART, Rainer; MAYDT, Jochen. An extended set of haar-like features for rapid object detection. In: *Proceedings. International Conference on Image Processing*. 2002, sv. 1, s. I–I.

34. AGHAJAN, Hamid; CAVALLARO, Andrea. *Multi-Camera Networks: Principles and Applications*. Orlando, FL, USA: Academic Press, Inc., 2009.

35. BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 2000.

36. LEE, G; GOMMERS, R; WASILEWSKI, F; WOHLFAHRT, K; O'LEARY, A; NAHRSTAEDT, H. *PyWavelets - Wavelet Transforms in Python* [online]. 2006 [cit. 2019-04-20]. Dostupné z: `https://github.com/PyWavelets/pywt/`.

37. ROSEBROCK, Adrian. *Simple object tracking with OpenCV* [online]. 2018 [cit. 2019-05-12]. Dostupné z: `https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/`.