České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra kybernetiky

DIPLOMOVÁ PRÁCE

Aktivní detekce obětí s RGB-D-T senzorem

Praha, Květen 2016 **Autor: Bc. Jiří Těžký**

Vedoucí práce: Ing. Karel Zimmermann, Ph.D.

Prohlášení autora práce

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

V Praze dne _____ _____

podpis

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Jiří T ě ž k ý

Studijní program: Kybernetika a robotika

Obor: Robotika

Název tématu: Aktivní detekce obětí s RGB-D-T senzorem

Pokyny pro vypracování:

V Centru strojového vnímaní (CMP) na katedře kybernetiky je v rámci Evropského projektu EU FP7 TRADR vyvíjen robot záchranář (http://cmp.felk.cvut.cz/demos). Robot je (mimo jiné) vybaven (i) všesměrovou RGB kamerou LadyBug III, (ii) rotačním laserovým scannerem SICK LMS-15 poskytující hloubková data a (iii) termo kamerou thermoIMAGER TIM 160 s malým zorným polem, která je umístěna na pohyblivé "pan-tilt" jednotce. Jednou z důležitých funkcionalit je autonomní detekce potencionálních obětí v místech, kam záchranáři nemají přístup.

1. Nastudujte implementaci [1] pro učení hlubokých konvolučních sítí a její aplikaci [2] pro segmentaci z RGB dat.
2. Navrhněte, naučte a vyhodnoťte síť pro segmentaci obětí z:
   a. RGB dat (i),
   b. RGBD dat (i)-(ii),
   c. a RGBDT dat (i)-(iii).
3. Navrhněte algoritmus řízení pohyblivé "pan-tilt" jednotky, zajišťující teplotní měření v místech kde segmentace z RGBD dat selhává.
4. Navržené algoritmy zaintegrujte do prostředí ROS (www.ros.org) na platformu robota.

Seznam odborné literatury:

[1] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks, NIPS, 2012.
[2] Jonathan Long, Evan Shelhamer, Trevor Darrell: Fully Convolutional Networks for Semantic Segmentation, CVPR, 2015

Vedoucí diplomové práce: Ing. Karel Zimmermann, Ph.D.

Platnost zadání: do konce letního semestru 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 16. 12. 2015

Poděkování

Chtěl bych poděkovat vedoucímu své diplomové práce Ing. Karlu Zimmermannovi, Ph.D. za vedení práce, trpělivost, cenné rady a odborný dohled. Dále chci poděkovat za cenné rady také Ing. Tomáši Petříčkovi.

Abstrakt

Diplomová práce se zabývá aktivní detekcí potenciálních lidských obětí na základě kamerových (RGB), hloubkových (D) a termovizních (T) dat, a to jejich klasifikací (segmentací) pomocí umělé neuronové sítě. Věnuje se učení hlubokých neuronových sítí pro segmentaci RGB-D-T snímků. Využívá se framework *Caffe* pro hluboké učení a práci s umělými neuronovými sítěmi. Výsledná segmentace vstupního RGB-D-T snímku je použita k detekci oběti a následně také k řízení pohledu termovizní kamery, která je umístěna na pohyblivé jednotce, a to k upřesnění segmentace v místech, kde nemůžeme rozhodnout a zároveň zde chybí informace o teplotě z důvodu menšího zorného pole termokamery. Pro tento účel byl přeučen model konvoluční neuronové RGB sítě a vytvořena nástavba pro hloubkovou a teplotní složku. Bylo sestaveno několik modelových architektur sítí, které byly vyhodnoceny. Model s nejlepšími výsledky byl vybrán k použití segmentace na robotovi se systémem ROS. K tomu je navržen a implementován algoritmus pro řízení pohledu termokamery k upřesnění konfidenční hodnoty segmentace. Na závěr byl proveden a vyhodnocen experiment, který demonstruje celkovou funkčnost systému.

Klíčová slova

Počítačové vidění, Strojové učení, Segmentace, Hluboké učení, Neuronová síť, Konvoluční neuronové sítě, Caffe, ROS

Abstract

The diploma thesis deals with active detection of the potential human victims based on camera (RGB), depth (D) and thermal (T) data. Detection uses this data for classification (segmentation) by the artificial neural network. The thesis dedicated to deep learning and working with artificial neural networks. Output of the RGB-D-T segmentation is used to victim detection and subsequently for segmentation clarification at indecisive locations assuming the missing temperature information due to the small field of thermal camera view. Clarification is done by view control of the thermal imaging camera which is placed on the movable unit. For this purpose was fine-tuned RGB model of convolutional neural network and designed extension for depth and thermal component. Several models were created and evaluated. The RGB-D-T model with the best results was chosen for segmentation on the robot running on ROS system. Then the camera view control algorithm was designed for clarification of segmentation confidence value. On a proof of concept was performed experiment with the robot.

Keywords

Computer Vision, Machine Learning, Image Segmentation, Deep Learning, Neural Networks, Convolutional Neural Networks, Caffe, ROS

Obsah

Seznam obrázků

Kapitola 1

Úvod

V Centru strojového vnímaní (CMP) na katedře kybernetiky je v rámci Evropského projektu EU FP7 TRADR vyvíjen robot záchranář. Jednou z důležitých funkcí je autonomní detekce potenciálních lidských obětí v místech přírodních katastrof, havárií a všeobecně v oblastech, kde je předpoklad újmy člověka na zdraví a kam záchranáři nemají snadný přístup nebo se jedná pro ně o příliš nebezpečné prostředí. Naším cílem je takovéto oběti identifikovat a lokalizovat v daném prostoru pomocí robota. Robot je proto mimo jiné vybaven všesměrovou RGB kamerou, rotačním laserovým scannerem a termokamerou s malým zorným úhlem, která je umístěna na polohovací "pan-tilt"jednotce. Tyto periferie nám poskytují kamerová, hloubková a teplotní (RGB-D-T) data. Prostředky robota mohou sloužit k případnému nalezení oběti i za velmi špatných okolních podmínek, jako například špatná viditelnost.

Cílem této práce je zpracovat RGB-D-T data a využít je ke správné identifikaci a lokalizaci již dříve zmíněných potenciálních lidských obětí pomocí segmentace obrazu. Tato technika se často úspěšně využívá pro detekci přesných kontur objektů na klasických obrázcích a dokonce se pořádají v tomto odvětví i soutěže. V teto oblasti jsou v poslední době dosahovány velmi dobré výsledky při použití hlubokých konvolučních neuronových sítí. Proto se přímo nabízí tuto techniku využít pro detekci specifického objektu a tedy naší potenciální oběti a přitom využít i dalších složek jako je hloubka a teplota.

Ke splnění cíle práce je nutné se seznámit s hlubokými konvolučními sítěmi a navrhnout vhodnou strukturu pro segmentaci RGB, RGB-D a RGB-D-T dat. Práce se také věnuje využití segmentace k řízení pohledu termokamery sloužící k upřesnění detekce, jelikož ta nepokrývá stejnou oblast jako RGB-D snímek, za předpokladu, že teplotní informace může pomoci. Pro demonstraci funkčnosti jsou navržené postupy a algoritmy zaintegrovány do prostředí ROS na platformu robota. Na závěr je proveden a vyhodnocen jednoduchý experiment dokazující funkčnost konceptu.

1.1 Struktura textu

Krátký popis jednotlivých kapitol této práce.

Úvod Tato kapitola obsahuje motivaci a rozbor zadání.

Teoretický úvod V této části se nachází úvod do problematiky segmentace. Jsou zde zmíněny pořádané segmentační soutěže, a které segmentační přístupy se umisťují na předních pozicích. Následuje popis robota, se kterým se bude pracovat a návrh řešení a architektury neuronové sítě pro segmentaci.

Učení Kapitola rozebírá teorii učení neuronových sítí. Následně je zde uveden postup přípravy dat. Konec kapitoly se věnuje učení zvolených modelů neuronových sítí a jejich vyhodnocení.

Implementace na robotovi Zde je popsána struktura celé implementace na platformě robota.

Experiment V závěrečné části práce je popsán a vyhodnocen provedený experiment s robotem.

Závěr Na závěr jsou shrnuty a zhodnoceny dosažené výsledky. Uvedeny jsou i náměty na zlepšení a pokračování v dané problematice.

1.2 Použité programy a pomůcky

V práci se využívá framework hlubokého učení *Caffe*, program *Matlab*, programovací jazyk Python, C++ a robotický operační systém ROS. Nejsou zde uváděny žádné zdrojové kódy nebo skripty. Ty a veškeré další soubory v práci zmíněné, se nacházejí na přiloženém DVD formou přílohy.

Kapitola 2

Teoretický úvod

2.1 Segmentace obrazu

Segmentace digitálního dvourozměrného obrazu, jenž je dán funkcí $f(x,\ y)$, je dělení na podobrazy $R_1,\ R_2,\ \ldots\ R_n$ takové, že splňují [4]:

1. $\bigcup_{i=1}^{n} R_i = f(x, y)$,
2. $R_i \cap R_j = \emptyset$ pro $i \neq j$,
3. všechny pixely v R_i mají alespoň jednu společnou vlastnost.

2.1.1 Segmentační techniky obrazu

Definice říká, že segmentací hledáme seskupení pixelů, které spolu sousedí a mají stejné určité vlastnosti.

Existuje mnoho segmentačních technik založených na různých principech a s různými výsledky. Jejich výčet a základní princip je možné nalézt v [4]. Nejvýznamnější z nich jsou:

- Detekce hran - jedná se o pouhé detekování hran objektů, například pomocí detekce změny jasu, barev. Mužeme využít první nebo druhou derivaci obrazové funkce $f(x,\ y)$ a nalézt extrém nebo průchod nulou.
- Detekce oblastí - zde už se jedná o vytyčení homogenního regionu v obraze. Příklad metody Region growin.
- Statistické metody - Markov Random Fields (MRF) se shlukováním.
- Znalostní metody - Active Appearance Models (AAM).
- Hybridní metody - neuronové sítě.

Právě poslední metoda segmentace s využitím umělých neuronových sítí s rozvojem výpočetního výkonu a možnost provádění výpočtů na čipech grafických karet se dostává do popředí a v segmentačních soutěžích se umisťuje na prvních místech.

2.1.2 PASCAL VOC Challenge

The PASCAL[1] Visual Object Classes Challenge je soutěž v rozpoznávání rozpoznávání sady objektů v realistické obrazové scéně [5]. Probíhala v letech 2005 - 2012 a měla několik kategorií [6]: Klasifikace, detekce, segmentace objektů, klasifikace akce a rozpoznávání polohy člověka. Kromě každoroční soutěže projekt poskytuje veřejně obrazový dataset. Ve výsledcích se tehdy téměř ani neobjevilo využití neuronových sítí.

2.1.3 ImageNet Challenge

O rozšíření umělých neuronových sítí v oblasti segmentace se zasloužila implementace *AlexNet* [7] nasazená v ILSVRC 2012.

ILSVRC - Large Scale Visual Recognition Challenge [8] je podobná soutěž doplňující VOC a je pořádána každoročně od roku 2010. Cílem je klasifikovat obsah na fotografiích. Trénovací dataset je založen na obrovské databázi ImageNet a obsahuje zhruba 10 000 000 obrázků s více než 10 000 kategoriemi [8]. Model sítě *AlexNet* využíval hluboké konvoluční vrstvy neuronové sítě. Několik konvolučních vrstev bylo poskládáno za sebou následovanách pool vrstvou (viz. 2.4.1.3). Narozdíl třeba od populární sítě *LeNet* [9] z devadesátých let pro rozpoznávání psaného písma, kdy pool vrstva následovala po každé konvoluční vrstvě zvlášť.

Architektura *AlexNet* se skládala z 5-ti skupin konvolučních vrstev zakončeným poolingem a nakonec tři plně propojených vrstev sloužící k finální klasifikaci. Implementace v soutěži výrazně překonala druhého v pořadí v TOP 5 chybě. TOP 5 chybovost znamená, že správný label označující správnou třídu není v prvních 5 nejpravděpodobnějších výsledcích klasifikátoru. V další ročnících se pak objevovali na vítězných pozicích převážně konvoluční neuronové sítě. Například *ZFNet*, *GoogLeNet*, *VGGNet* a další implementace [1].

Zdá se, že dostatečná hloubka sítě je pro tento úspěch klíčová. Autoři modelu AlexNet zkoušeli odebrat vrstvy a model sítě zjednodušit, vždy však nastalo zhoršení. Limitující byla pro trénování velikost paměti na tehdy dostupných grafických kartách [7].

Z výsledků soutěží v posledních letech vyplývá, že využití hlubokých konvolučních sítí má velký potenciál. Proto tuto metodu použijeme na segmentaci RGB-D-T dat za účelem hledání potenciálních obětí.

[1] Pattern Analysis, Statistical Modeling and Computational Learning

2.2 Active vision

Oproti klasickému počítačovému vidění, kdy je kamera většinou pevně umístěná na statickém nebo pohyblivém objektu (automobil), se odvětví aktivního počítačového vidění zabývá možností optického nebo mechanického řízení pohledu kamery. Toto řízení vede k optimalizaci cíle daného úkolu nebo zjednodušení strojového zpracování, neboť kamerou se nelze dívat všude [10]. Příklad takového systému je právě náš úkol, s využitím dalších komponent, řídit pohled termovizní kamery do míst, kde je to dle určitého kritéria nejužitečnější.

2.3 Robot

Práce se bude implementovat na robotu v rámci projektu EU FP7 TRADR. Ten je mimo jiné vybaven všemi potřebnými senzory k získání RGB-D-T dat. Na robotu se nachází komplexní zařízení na snímání pohybu Asus Xtion obsahující RGB kameru a hloubkový senzor. Snímání hloubkových dat toto zařízení provádí na základě principu infračervené laserové projekce a analýzy nasnímaných patternů monochromatickou kamerou. Nevýhodou tohoto řešení je nemožnost využití v přesvětlených venkovních podmínkách, kdy ostré denní světlo narušuje rozeznání promítaných vzorů. Přesto se jedná o vývojáři často využívané a levné řešení k tvorbě prostorových dat. S využitím tohoto zařízení byl proveden první test implementace na robotovi. Dodaná trénovací data byla také vytvořena na tomto zařízení.

Pro finální implementaci však bude využita všesměrová RGB kamera Ladybug 3, kterou je robot také vybaven. Obsahující šest $2\,Mpx$ kamer dovolující zachytit snímky z 80% celého prostoru [11]. Tato kamera nám bude poskytovat RGB data. Hloubková data budou pocházet z rotačního laserového LIDAR skeneru SICK LMS-15, měřící vzdálenost až do $50\,m$ [12].

Posledním senzorem je vysokorychlostní $120\,Hz$ infra kamera thermoIMAGER TIM160 poskytující teplotní data v rozsahu $-20\,°\mathrm{C}$ až $900\,°\mathrm{C}$ [13]. Snímky mají rozlišení $160 \times 120\ px$. Kamera je umístěna na řiditelné dvouosé pan-tilt jednotce.

2.4 Návrh řešení

Pro práci s neuronovými sítěmi byl vybrán framework *Caffe*[2] [14] vyvinutý v komunitě BVLC[3]. Jedná se o C++ knihovnu s BSD-licenci poskytující interface pro Python a Matlab pro trénování a efektivní práci s neuronovými sítěmi [15]. Kód frameworku je také psán s využitím CUDA,

[2]Convolutional Architecture for Fast Feature Embedding
[3]Berkeley Vision and Learning Center

a je tedy možné k výpočtům využít kromě CPU i GPU. Díky tomu je možné několikanásobně zrychlit proces učení i klasický dopředný průchod sítí.

Pro různé účely byly početnou komunitou vytvořeny a naučeny *Caffe* modely s širokým spektrem použití. Například modely pro klasifikace nebo segmentace objektů na snímku, rozpoznání scény, sportu prováděném člověkem, stanovení polohy člověka, určení věku, pohlaví a mnoho dalších aplikací. Tyto modely dali vývojáři k dispozici v rámci BVLC model license ke stažení z *Caffe* GitHub[4]. Jsou ve většině případů naučeny z neporovnatelně většího množství trénovacích dat a to například z obsáhlé databáze ImageNet (2.1.3) tvořící milióny obrázků. Proto se naskytla myšlenka využít jeden z těchto modelů pro účel segmentace RGB složek a přeučit ho na nových datech, které odpovídají našemu použití, tedy detekci potenciálních obětí.

2.4.1 Architektura modelů

Framework *Caffe* nabízí pro architekturu sítí několik typů vrstev. Nyní si popíšeme ty použité v této práci.

2.4.1.1 Konvoluční vrstva

Jedná se o vrstvu typu "Vision Layers", je tedy navržena pro práci prostorovou strukturou obrázků s daným počtem kanálů, výškou a šířkou ($c \times h \times w$). Na rozdíl od ostatních vrstev, které berou vstupní data pro zjednodušení práce jako jednorozměrný vektor a neberou v úvahu prostorové spojitosti.

Zde si uvedeme parametry vrstvy používané v této práci.

- Počet konvolučních filtrů neboli hloubka sítě num_output,
- velikost konvolučního jádra $kernel\_size$ nebo $kernel\_h$, $kernel\_w$ pro každý rozměr zvlášť,
- počet pixelů přidaných ke každé straně výstupu pad,
- velikost kroku posouvání konvolučního filtru $stride$ nebo zvlášť $stride\_h$, $stride\_w$.

Framework nabízí nastavení dalších volitelných parametrů, jejich výčet je uveden v tutoriálu [15].

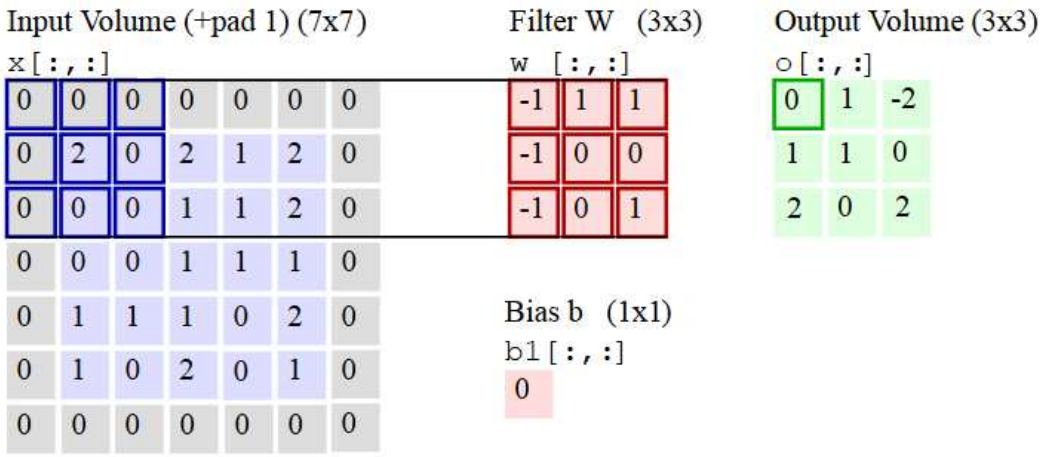
Jako vstup lze považovat datovou dávku o rozměrech $n \cdot c \cdot h \cdot w$, kde n je počet obrázků v jedné dávce. Výstup je pak dán zvolenými parametry $n \cdot c\_o \cdot h\_o \cdot w\_o$, kde

$$h_o = (h + 2 \cdot pad\_h - kernel\_h)/stride\_h + 1$$

a podobně pak w_o. Parametry je nutné volit tak, aby h_o, w_o byly celé čísla.

[4] https://github.com/BVLC/caffe/wiki/Model-Zoo

Princip Při použití klastické plně propojené neuronové sítě bychom měli mnoho parametrů, což by vedlo nejen k velké paměťově náročnosti, ale síť by měla sklony k přeučení. Při dopředném průchodu konvoluční sítí každý filtr ve vrstvě konvoluje vstupní obrázek s celou hloubkou (se všemi kanály) konvolučním jádrem přes výšku a šířku. Propojení neuronů je tedy lokální v prostoru výšky, šířky vstupních dat a plné propojení přes hloubku (počet kanálů). Při konvoluci se vytváří 2D výstupní aktivační mapa a každý prvek výstupu lze interpretovat jako výstup jednoho neuronu, který je propojen s lokální oblastí danou velikostí konvolučního jádra [1]. Avšak díky tomu, že sdílejí váhy s ostatnímu neurony ve stejné aktivační mapě (konvoluce se provádí se stejným jádrem), se jedná o diskrétní konvoluci. Proces je znázorněn na obrázku 2.1.



Obrázek 2.1: Konvoluce (převzato a upraveno z [1])

Díky tomuto sdílení vah snížíme počet parametrů. Například uvažujme konvoluční vrstvu s hloubku $num\_output = 2$, konvolučním jádrem $kernel\_size = 3$, posunem $stride = 2$ a za vstup obrázek se 3 kanály s rozměry $h \times w \times c = 5 \times 5 \times 3$ doplněný o $pad = 1$. Výstup bude mít velikost $h\_o \times w\_o \times num\_output$, kde

$$w\_o = h\_o = (h + 2 \cdot pad - kernel)/stride + 1 = (5 + 2 \cdot 1 - 3)/2 + 1 = 3$$

Bez sdílení vah bychom měli $w\_o \cdot h\_o \cdot num\_output = 3 \cdot 3 \cdot 2 = 18$ neuronů a každý $kernel\_size \cdot kernel\_size \cdot c = 3 \cdot 3 \cdot 3 = 27$ vah, což je $18 \cdot 27 = 486$ parametrů. Při konvoluci však neurony ve stejné aktivační mapě mají stejné váhy a parametrů je jen $num\_output \cdot kernel\_size \cdot kernel\_size \cdot c = 2 \cdot 3 \cdot 3 \cdot 3 = 54$, což je značná úspora paměťová a také při učení časová. Znázornění tohoto příkladu lze vidět v příloze B.1.

2.4.1.2 ReLu vrstva

Vrstva ReLU (Rectified-Linear and Leaky-ReLU) je s plným propojením a elementy mající aktivační funkci:

$$g(x) = max(0,\ x) \tag{2.1}$$

a v případě definovaného parametru *negative_slope*:

$$g(x) = \begin{cases} x, & pro\ x > 0, \\ negative\_slope \cdot x, & jinak. \end{cases} \tag{2.2}$$

Zavádí nesaturovanou nelinearitu a což je při trénování algoritmem SGD časově rychlejší než u aktivačních funkcí typu *tanh* [7]. Podporuje in-place výpočet, pro úsporu paměti a tedy vstupní a výstupní vrstva může být stejná, jak je znázorněno na celé architektuře FCN-32s modelu sítě příloze A.1.

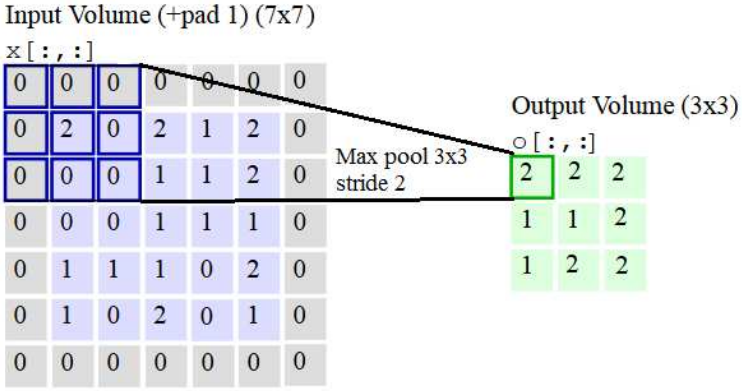
2.4.1.3 Pool vrstva

Vrstva zjišťuje v podstatě downsampling při zachování hloubky. Často se vkládá za několik konvolučních vrstev k omezení velikosti a počtu parametrů.

Parametry vrstvy jsou podobné těm u konvoluční vrstvy. Použité v modelu jsou [15]:

- Velikost jádra *kernel_size* nebo *kernel_h*, *kernel_w* pro každý rozměr zvlášť,
- velikost kroku posunutí *stride* nebo *stride_h*, *stride_w*.

Podobně jako u konvoluční vrstvy se jádro (okno) o velikosti *kernel_size* posouvá o krok *stride*. Je možné nastavit *pad* a pooling metodu. Nejčastější a také defaultní metoda je MAX, která z každého překryvu předá na výstup maximum ze všech elementu. Je možní taky použít průměrování nebo výběr náhodné hodnoty. Jelikož princip a parametry jsou podobné jako u konvoluční vrstvy (viz. 2.4.1.1) velikost vstupních a výstupních dat se řídí stejnými pravidly.



Obrázek 2.2: Princip pool vrstvy [1]

K ořezání počtu parametrů je možné použít také samotnou konvoluční vrstvu s velkým krokem (*stride*) jak je uvedeno v [16].

2.4.1.4 Dropout vrstva

Vrstva využívaná k prevenci proti přeučení. Jedná se o plně propojenou vrstvu a podobně jako u ReLu vrstvy umožňuje in-place výpočet. Funkce vrstvy je při učení náhodně odstranit elementy v dané vrstvě včetně jejich propojení nastavením výstupu do nuly [17]. Většinou se používá s pravděpodobností 0.5. To znamená, že pro každý neuron je poloviční pravděpodobnost, že se nebude podílet a jeho výstup bude nastaven na nulu.

2.4.1.5 Dekonvoluční vrstva

V podstatě se jedná o obrácenou funkci konvoluční vrstvy. Více méně se může zaměnit dopředný průchod za zpětný. Je možné ji využít pro upsampling. Při velikosti jádra $kernel\_size = 1$ se bude jednat o plně propojenou vrstvu s předchozí (kovoluční jádro pokrývá celou vstupní oblast). Parametr *stride* udává výsledný posun jádra ve výstupu a tedy upsampling poměr. I váhy této vrstvy lze upravovat učením. Pokud je necháme fixní, jedná se čistě o upsampling například bilineární [2].

2.4.1.6 Crop vrstva

Upravuje souřadnice mezi vrstvami aby výstupní pixely odpovídaly vstupním.

2.4.1.7 Ztrátové vrstvy

Řadí se mezi vrstvy prezentující ztrátovou funkci, která se využívá k řízení adaptačního procesu učení sítě, jehož cílem je minimalizace této ztráty. Framework *Caffe* nabízí několik typů těchto vrstev. Během práce budeme využívat *SoftmaxWithLoss*, která počítá multinomickou logistickou ztrátu 2.4 (multinomial logistic loss) funkce softmax:

$$\hat{p}_{nk} = \frac{e^{x_{nk}}}{\sum_{k'} e^{x_{nk'}}}. \tag{2.3}$$

Vstupem je zpravidla končená vrstva sítě obsahující predikci x ve tvaru $N \times C \times H \times W$ a datová vrstva obsahující správnou masku (ground truth) s K třídami $l_n \in [0, 1, \ldots K-1]$, kde N je dávka při trénování, C je hloubka poslední vrstvy, H a W výška, šířka [15].

$$loss = -\frac{1}{N} \sum_{n=1}^{N} \log(\hat{p}_{nl_n}), \tag{2.4}$$

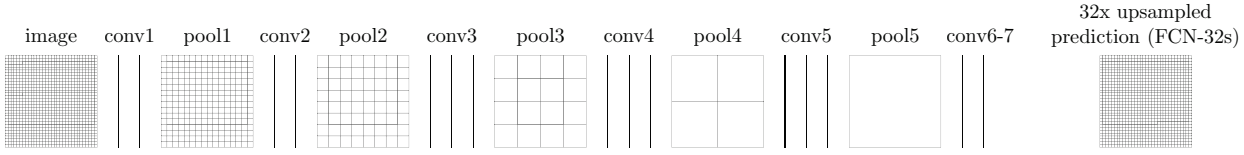
2.4.1.8 Datová vrstva

Nejčastější využívá k definici vstupních trénovacích dat při adaptaci sítě. Jedná se o způsob jak přivést přivést externí data. Nejefektivnější je použít dat ve formátu databáze (LevelDB, LMDB), možné je nahrát data přímo z paměti nebo méně efektivně přímo z jednotlivých souborů ve formátu běžných obrázků nebo struktury HDF5 [14]. Do skupiny parametrů vrstvy *data_param* je nutné zadat cestu k souborům *source* a velikost dávky zpracovávané v jeden čas *batch_size*. Dále je možné specifikovat typ vstupních dat *backend*.

V této práci budeme pracovat s LMDB databázi vstupních dat. Pokud budeme využívat při trénování i validační data, je nutné specifikovat i fázi. Ve skupině *include* se jedná o parametr *phase*.

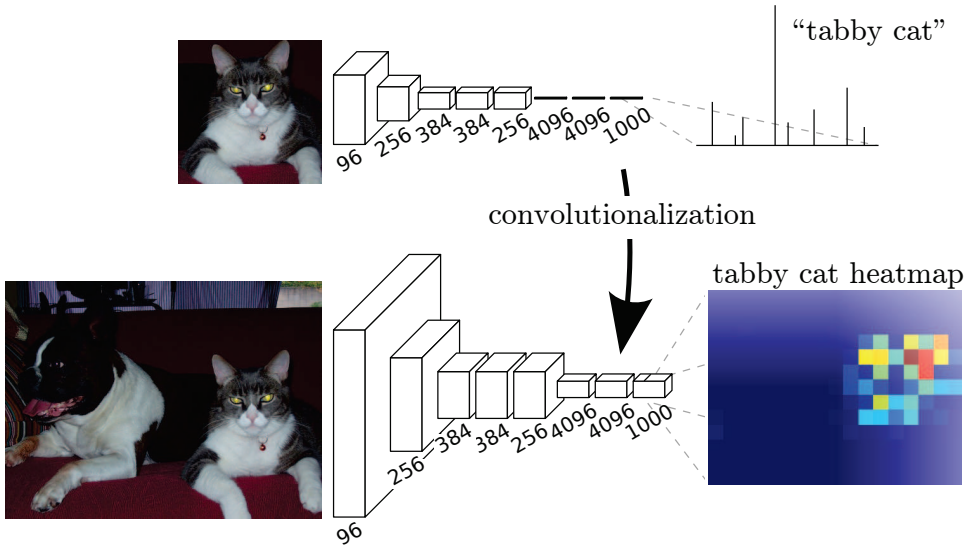
2.4.2 RGB model sítě

K segmentaci RGB snímků byl vybrán model FCN-32s PASCAL [2]. Jedná se o model plně konvoluční neuronové sítě pro sémantickou segmentaci, jeho strukturu lze vidět na obrázku 2.3. Autoři model naučili na segmentačním datasetu PASCAL VOC 2012[5] zmíněném v 2.1.2. Ten obsahoval 5623 trénovacích dat s 20 třídami a pozadím. Jejich cílem bylo vytvořit plně konvoluční síť pro segmentaci obrázku libovolných rozměrů, která bude produkovat segmentační výstup se stejnými rozměry. Jinak řečeno, cílem není klasifikace obrazu jako tomu je například u sítě *AlexNet*, ale cílem je získat predici pro každý jednotlivý pixel. Síť byla vytvořena na základě známých naučených sítích *AlexNet*, *VGGNet* a *GoogLeNet* a následný fine-tuning parametrů pro segmentaci. Autoři provedli transformaci zmíněných modelů na plně konvoluční síť schopnou predikce pro každý pixel zvlášť (obrázek 2.4).



Obrázek 2.3: FCN-32s PASCAL [2]

[5]Semantic Boundaries Dataset and Benchmark:
http://www.cs.berkeley.edu/ bharath2/codes/SBD/download.html



Obrázek 2.4: Konvolucionalizace sítě určené pro klasifikaci [2]

Díky této modifikaci dosáhli relativního zlepšení téměř o 20% [2] (62% mean IU[6], přičemž nejlepší výsledek z VOC2012 byl 47.3% [5]) na datech Pascal VOC 2012.

Označení 32s se týká poslední upsampling vrstvy s parametrem $stride = 32$, která převzorkuje předposlední vrstvu 32× pro docílení stejné velikosti výstupní predikce jako vstupního obrazu. Ve článku [2] byly prezentovány další modely FCN-16s a FCN-8s s větším rozlišením výstupní predikce využívající přeskočení několika vrstev sítě a součtem posledních pool vrstev dohromady. Pro RGB segmentaci byl vybrán model FCN-32s a konečným 32× zvětšením. Neboť naším cílem není přesnost zachycení kontury člověka, ale pouze přesnost detekce obětí samotné.

Cílem je naučit tento vybraný model na nových datech s inicializací vah z [2]. Využita byla skutečnost, že původní model obsahoval mimo jiné kategorii pro pro člověka a pozadí. Trénovací masky (segmentační ground truth[7]) byly upravena tak aby oběť odpovídala kategorii člověka a zbytek pozadí. Tím nemusíme zasahovat do struktury sítě FCN-32s PASCAL.

2.4.3 Zakomponování D-T složek

Předpoklad této práce spočívá v myšlence informačního přínosu hloubkových a teplotních dat k upřesnění segmentace obrazu. Z toho plyne otázka jakým způsobem rozšířit zvolený RGB model 2.4.2 o D-T složky. Jednou z možností je rozšířit celou síť FCN-32s o další dva vstupní kanály. Implementačně se nejedná o nic složitého, část vah bychom mohli zkopírovat z naučeného modelu

[6] mean Intersection over Union $= \frac{1}{n} \frac{\sum_i n_{ii}}{t_i + \sum_j (n_{ji} - n_{ii})}$, kde n je počet tříd, n_{ij} je počet pixelů třídy i patřící třídě j a $t_i = \sum_j n_{ij}$ [2].

[7] Většinou ručně označená skutečná oblast daného objektu na snímku.

a zbytek vah by muselo být doučeno. Učení téměř od nuly s takto velkou sítí by nemuselo vést k dobrému výsledku. Dokonce i autoři sítě FCN-32s prováděli přeučení z již vytvořených sítí. Proto byl zvolen jiný postup, a to opět do FCN-32s sítě, ale využít jeho výstup k vytvoření konfidenčního obrázku. Hodnoty jednotlivých pixelů budou představovat míru přesvědčení v rozsahu 0 až 1, že daný pixel patří oběti.

Tuto segmentační konfidence z RGB dat (označíme C_{rgb}) vytvoříme z výstupu poslední datové vrstvy s názvem *upscore* viz. A.1. Jak název vrstvy napovídá, je zde uloženo skóre pro každou třídu ve formě blobu s rozměry $h \times w \times num\_output$, kde h, w jsou rozměry vstupních RGB dat a $num\_output = 21$, což je hloubka poslední dekonvoluční vrstvy *upsample* a odpovídá počtu segmentovaných tříd $0 - 20$. Třída 15 patří člověku/oběti a 0 představuje pozadí.

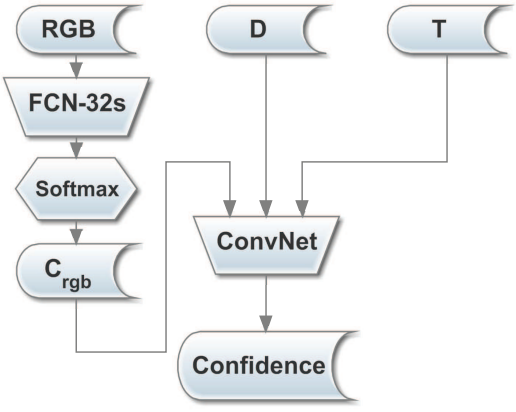
Jelikož nás však zajímá pouze zmíněná třída člověka/oběť, budeme považovat za pozadí i ostatní třídy. Vytvoříme dvě matice velikosti vstupního snímku, které budou obsahovat skóre pro oběť $score_v$ a pozadí $score_{bg}$ jehož každý pixel je tvořen maximem ze stejného pixelu ostatních tříd:

$$\begin{aligned} score_v &= upscore[:, :, 15], \\ score_{bg} &= max_{dim=3}\left(upscore[:, :, 0:14-16:20]\right). \end{aligned} \tag{2.5}$$

C_{rgb} představuje softmax $score_v$ a $score_{bg}$:

$$C_{rgb} = \frac{e^{score_v}}{e^{score_v} + e^{score_{bg}}}. \tag{2.6}$$

Vytvořili jsme tak novou konfidenční složku C_{rgb}, kterou využijeme spolu s D a T jako vstup do další konvoluční sítě s označením *ConvNet*. Návrh celé struktury je možné vidět na obrázku 2.5. *ConvNet* vytvoříme a naučíme od počátku pouze z našich trénovacích dat. Při návrhu bylo vyzkoušeno několik různých struktur, které jsou popsány v následující kapitole 3 věnující se učení neuronových sítí. Struktura s nejlepšími výsledky bude použita k implementaci na robotovi.



Obrázek 2.5: Návrh přidání D-T složek

2.4.4 Řízení kamery

Jak už bylo řečeno, díky malému zornému poli termokamery nezle získat teplotní informace v celém rozsahu všesměrové RGB kamery, dokonce ani najednou pokrýt oblast laserového scanneru dodávající hloubková data. Proto je nutné směrovat záběr termokamery pomocí pan-tilt jednotky do míst, kde je to nejužitečnější. Například do míst, kde teplotní informace pomůže upřesnit výskyt potenciální oběti.

Idea je ve využití, k řízení pohledu termokamery, výše rozebranou segmentaci obrazu. Na robotu bude neustále prováděna segmentace snímků RGB-D a v místech, kde bude k dispozici i termální informace se provede segmentace RGB-D-T. Výsledky se ukládají do prostorové mapy a místa, kde chybí teplota se analyzují. Díky tomu budeme moci rozhodnout kam je výhodné se s termokamerou podívat. Implementace tohoto algoritmu je popsána v kapitole 4.

Kapitola 3

Učení

Po návrhu architektury umělé neuronové sítě následuje proces učení, jehož cílem je nalezení vah a propojení jednotlivých neuronů mezi sebou. Pří učení s učitelem je požadovaná funkce sítě dána trénovací množinou dat. Proces učení je adaptací vah.

Caffe k učení poskytuje metodu *solver* řešící obecný optimalizační problém nad trénovací datovou množinou D. Cílem je minimalizace průměrné ztráty přes instance této množiny [14]:

$$L(W) = \frac{1}{|D|} \sum_{i}^{|D|} f_W(X^{(i)}) + \lambda r(W), \tag{3.1}$$

kde W je množina parametrů neuronové sítě, $f_W(X^{(i)})$ je hodnota ztrátové funkce na datové instanci $X^{(i)} \in D$ a $r(W)$ je regularizační term s váhou λ. V některých publikacích je $r(w)$ nazýván weight decay term jenž má za následek pokles velikosti vah a pomáhá předejít přeučení [18]. Velikost $|D|$ je často velmi velká a proto se v praxi rozdělují data do menších dávek m, kde $m << |D|$, pro každou iteraci metody *solver* [14]:

$$L(W) \approx \frac{1}{m} \sum_{i}^{m} f_W(X^{(i)}) + \lambda r(W). \tag{3.2}$$

Funkce $L(W)$ je pak aproximována empirickým riskem [19]. Cílem je v každé iteraci aktualizovat parametry sítě W. Změna ΔW je odvozena od gradientů ∇f_W, $\nabla r(W)$ a dalších parametrů závislých na použité metodě optimalizace.

Caffe má implementovaných několik optimalizačních metod: Stochastic gradient descent dále jen SGD, Adaptive gradient, Adam, Nesterov's accelerated gradient, RMSprop. Defaultně Caffe využívá optimalizační metodu SGD.

3.1 Stochastic gradient descent

Jedná se o nejrozšířenější algoritmus učení ve většině aplikací neuronových sítí [20]. Algoritmus vybere vždy náhodnou dávku z tréninkové množiny dat D, přičemž se předpokládá, že náhodný výběr dávky se nebude vlastnostmi příliš lišit od ostatních instancí množiny D. Váhy W jsou nejprve inicializovány náhodně malou hodnotou blízkou nule a následně aktualizovány ve směru negativního gradientu empirického risku $\nabla L(W)$ definovaného rovnicí 3.2.

$$\begin{aligned} V_{t+1} &= \mu V_t - \alpha \nabla L(W_t), \\ W_{t+1} &= W_t + V_{t+1}, \end{aligned} \tag{3.3}$$

kde V_t je aktualizace váhy v iteraci t. μ moment a α koeficient učení jsou parametry. Pokud $L(W)$ není konvexní funkce může SGD skončit v lokálním minimu, v praxi ale většinou pracuje velmi dobře [18].

Předpokládejme, že máme trénovací množinu

$$D = \{X^{(1)}, \ldots, X^{(m)}\} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\} \tag{3.4}$$

o m příkladech. Pak jeden trénovací příklad $X^{(i)}$ je dvojice $(x^{(i)}, y^{(i)})$, kde x je vstupní hodnota a y výstupní požadovaná. Pak můžeme definovat ztrátovou funkci $f_W(X^{(i)})$ jako:

$$f_W(X^{(i)}) = \|h_W(x^{(i)}) - y^{(i)}\|^2, \tag{3.5}$$

kde $h_W(x^{(i)})$ je při dané množině vah W a vstupu $x^{(i)}$ sítě její výstupní hypotéza [18], kterou dostaneme dopředným průchodem sítí - forwardpropagation.

3.1.1 Forwardpropagation

Pro zjednodušení zápisu budeme uvažovat pouze jeden trénovací příklad $X = (x, y)$ se vstupním vektorem x a očekávaným výstupem y. Máme-li více vrstvou acyklickou síť s počtem vrstev n_l, kde l je index vrstvy přičemž $l = 1$ je vstupní a $l = n_l$ výstupní vrstva. Rozdělíme li množinu W na podmnožiny vah $w^{(l)}$ mezi jednotlivými neurony ve vrstvě l a $l + 1$ a na podmnožiny biasu $b^{(l)}$ spojenou s neurony ve vrstvě $l + 1$, pak můžeme psát [18]:

$$\begin{aligned} z^{(l+1)} &= w^{(l)} a^{(l)} + b^{(l)}, \\ a^{(l+1)} &= f(z^{(l+1)}), \\ h_W(x) &= a^{(n_l)} = f(z^{(n_l)}), \end{aligned} \tag{3.6}$$

kde $a^{(l)}$ je aktivace (výstup) neuronů ve vrstvě l (pro vstupní vrstvu je $a^{(1)} = x$), $z^{(l)}$ jsou celkové vážené sumy vstupu do neuronů ve vrstvě l včetně biasů. Je zřejmé, že $a^{(l)} = g(z^{(l)})$, kde g je aktivační funkce neuronů.

Takto lze pomocí dopředného průchodu sítí spočítat $h_W(x)$.

3.1.2 Backpropagation

Rozepíšeme 3.2 [18]:

$$\begin{aligned} L(W) &= \frac{1}{m}\sum_{i=1}^{m} f_{(w,\,b)}(x^{(i)},\ y^{(i)}) + \lambda \sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}} \left(w_{ji}^{(l)}\right)^2 \\ &= \frac{1}{m}\sum_{i=1}^{m}\left(\left\|h_{w,b}(x^{(i)}) - y^{(i)}\right\|^2\right) + \lambda \sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}} \left(w_{ji}^{(l)}\right)^2, \end{aligned} \tag{3.7}$$

kde s_l je počet jednotek neuronů ve vrstvě l a $w_{ji}^{(l)}$ je váha na spojení neuronu j ve vrstvě l s neuronem i ve vrstvě $l+1$, pak gradient :

$$\nabla L(W) = \nabla L(w,b) = \left(\frac{\partial L}{\partial w},\ \frac{\partial L}{\partial b}\right), \tag{3.8}$$

a rozepsané složky [18]:

$$\begin{aligned} \frac{\partial}{\partial w_{ij}^{(l)}} L(w,b) &= \left[\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial w_{ij}^{(l)}} f_{(w,\,b)}(x^{(i)},\ y^{(i)})\right] + \lambda w_{ij}^{(l)}, \\ \frac{\partial}{\partial b_i^{(l)}} L(w,b) &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial b_i^{(l)}} f_{(w,\,b)}(x^{(i)},\ y^{(i)}). \end{aligned} \tag{3.9}$$

K tomu abychom mohli spočíst tyto parciální derivace je třeba zavést zpětné šíření chyby - backpropagation.

Pro jeden trénovací $X = (x,\ y)$ je nejprve nutné provést dopředný průchod sítí (Forwardpropagation) 3.1.2 čímž dostaneme hypotézu $h_{w,\,b}(x)$. Poté pro každý neuron i ve vrstvě l spočítat jeho přírůstek chyby $\delta_i^{(l)}$. Pro poslední (výstupní) vrstvu je jednoduché tento přírůstek $\delta_i^{(n_l)}$ určit z rozdílu $h_{w,\,b}(x)$ a požadovaného výstupu y [18].

$\delta_i^{(l)}$ spočteme na základě váženého průměru chybových přírůstků těch neuronů, které mají vstup $a_i^{(l)}$ [18]:

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \left\|y - h_{w,b}(x)\right\|^2 = -(y_i - a_i^{(n_l)}) \cdot g'(z_i^{(n_l)}), \tag{3.10}$$

a dále pro $l = n_l - 1, n_l - 2, n_l - 3, \ldots, 2$:

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} w_{ji}^{(l)} \delta_j^{(l+1)}\right) g'(z_i^{(l)}). \tag{3.11}$$

Hledané parciální derivace jsou [18]:

$$\begin{aligned} \frac{\partial}{\partial w_{ij}^{(l)}} f_{(w,\,b)}(x,\ y) &= a_j^{(l)} \delta_i^{(l+1)} \\ \frac{\partial}{\partial b_i^{(l)}} f_{(w,\,b)}(x,\ y) &= \delta_i^{(l+1)}. \end{aligned} \tag{3.12}$$

Aktivační funkce g s adaptační metodou SGD využívající zpětné šíření chyby musí být spojitá, diferencovatelná a monotónně neklesající [20].

3.1.2.1 Backpropagation u jednotlivých vrstev

Konvoluční vrstva Využívá se výše zmíněného řetězového pravidla. Při zpětné propagaci se jedná opět o konvoluci s převrácenými filtry [1].

Pooling vrstva Zpětná propagace této vrstvy při funkci *max* je předání gradientu na vstup, který měl největší hodnotu při dopředném průchodu [1].

3.2 Příprava dat

Pro trénování sítě byla vedoucím práce dodána RGB-D-T data. Jednalo se o nasnímaná data fiktivních lidských obětí v různém prostředí. Větší část dat byla vygenerována a menší část byla reálná data z kancelářského prostředí. Rozlišení všech dodaných snímků je 640×480 px. Díky užšímu zornému poli termokamery, jsou teplotní data validní pouze v malém výřezu o stanoveném rozlišení 381×281 px. Vzhledem k tomu, že paměťová náročnost roste s rozlišením a objety zájmu jsou v pohledu termokamery, byly pro učení využity ty části obrazové RGB a hloubkové D, ve kterých byla validní termovizní data. Všechny snímky jsou dle toho ořezány s výsledným rozlišení 381×281 px.

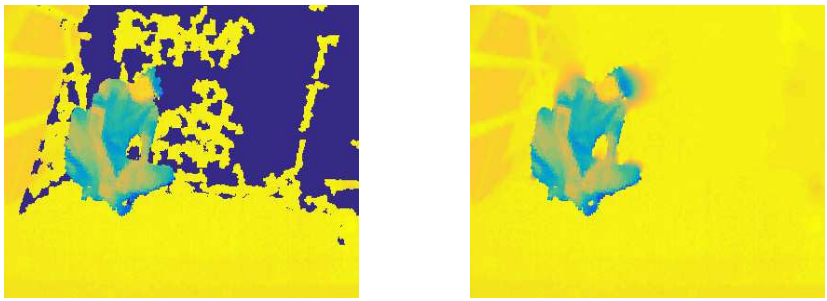
Obrázek 3.1: Příklad generovaného RGB-D-T snímku

Obrázek 3.2: Příklad reálného RGB-D-T snímku

Data jsou rozdělena na trénovací, testovací, validační a reálnou množinu, s počty snímků 1445, 1524, 1453, 108. Trénovací sada dat je určena k adaptaci parametrů sítě při procesu učení a zároveň při tomto procesu využijeme i validační data ke kontrole během učení. Testovací a reálná sada snímků byla využita k vyhodnocení modelu sítě.

3.2.0.1 Interpolace chybějících dat

I po výřezu validní části snímané termokamerou, se mohou naskytnout v obraze části, kde teplota není definovaná. Aby byly trénovací příklady konzistentní je využita k interpolaci funkce *inpaint_nans* ze stejnojmeného balíčku [21] pro Matlab. Tato funkce implementuje několik metod, ze kterých je pro doplnění využívána metoda nejbližších osmi sousedů.



Obrázek 3.3: Ukázka interpolace chybějících teplotních dat

Na obrázku 3.3 můžeme vidět demonstraci na vybraném snímku, kde chybí větší množství dat. Stejný postup je nutné aplikovat i na hloubková data.

3.2.0.2 Převod do databáze

Pro efektivní práci s tady v *Caffe*, je třeba převést sady RGB-D-T snímků na do jednotlivých databází LMDB[1]. Databáze založená na B+ stromové struktuře je výkonná a paměťově efek-

[1]Lightning Memory-Mapped Database

tivní [22]. Navržená je pro mapování klíč-hodnota, což se hodí například pro klasifikaci, kdy se ukládá obrázek a hodnota třídy, do které patří. U segmentace tato vlastnost není využívána, neboť maska (ground truth) a obraz jsou ukládány zvlášť do jednotlivých databází. Před převodem byla teplotní a hloubková data byla převzorkována do stejného rozsahu jako u RGB složek. Pro převod lze využít Python balíček *lmdb* nebo využít wrapper pro Matlab *matlab-lmdb* [23].

3.3 Učení a experimenty s architekturou sítě

Účení neuronových sítí probíhalo na školním výpočetním serveru HALMOS paralelně na 3x GPU GeForce GTX TITAN. *Caffe* využívá standardní stochastický gradient descent algoritmus [15].

Pro práci se sítí byly vytvořený tyto soubory:

- `*_deploy.prototxt` architektura sítě pro vybavení,
- `*_train.prototxt` architektura sítě pro trénování. Od předchozího popisu se liší vrstvami navíc. Jedná se o vstupní datové a finální ztrátovou vrstvu, která řídí trénování.
- `*_solver.prototxt` sloužící k nastavení metody solve,
- `*.caffemodel` soubor s parametry sítě.

Definice modelů jsou psány jako konfigurační soubory pomocí jazyka Protocol Buffer a jednotlivé modely parametry sítě (váhy) jsou uloženy ve formě Google Protocol Buffers[2].

3.3.1 Adaptační parametry sítě

Jedná se o parametry, se kterými bylo v průběhu učení experimentováno a které tento proces ovlivňují. Přímo v testovací definici sítě (`*_train.prototxt`) můžeme pro každou konvoluční vrstvu zadat násobič learning rate (*lr_mult*) a decay (*decay_mult*) zvlášť pro váhy a bias zvášť. Tímto nastavením můžeme určit, které vrstvy se mají učit rychleji a nebo učení zcela vypnout. Například u poslední dekonvoluční vrstvy učení modelu FCN-32s jsou parametry nulové a vrstva zajišťuje pouze upsampling. Při definici popisu datových vrstev zadáme cestu k LMDB databázím obsahující trénovací a validační data a jejich ground truth. U sítě FCN-32s zadáme ještě střední hodnoty jednotlivých RGB složek.

3.3.2 Solver

Metoda *solver* frameworku *Caffe* má vlastní konfigurační soubor (`*_solver.prototxt`). Umožňuje nastavit parametry metody, průběh učení a je nutný k samotnému spuštění adaptace sítě pokud nechceme všechny parametry zadávat z příkazového řádku. Soubor obsahuje například cestu

[2] https://code.google.com/p/protobuf/

k definici architektury sítě pro trénování (`*_train.prototxt`). Užitečná je i možnost nastavit četnost a cestu k ukládání snapshot souborů, ze kterých můžeme vždy pokračovat v případě pádů nebo přerušení učení z důvodu úpravy parametrů. Z dalších důležitých parametrů uvedeme následující, se kterými se experimentovalo:

- *test_interval*: jedná se o interval ve kterém se spouští testování pokud jsme v `*_train.prototxt` uvedli cestu i k validační sadě a s tímto parametrem souvisí:
- *test_iter*, který udává kolik iterací má být provedeno při testovacím průchodu.
- *lr_policy* je důležitý parametr ovlivňující koeficient učení (learning rate) α v průběhu iterací a je možné volit metodu útlumu koeficientu učení. Spolu se zvolenou metodou musejí být definovány i výše zmíněné parametry:
  (*base_lr*, *gamma*, *step*, *stepvalue*, *power*). Metody jsou [14]:
  - $fixed$: při této hodnotě zůstává koeficient učení na hodnotě zadané parametrem $base\_lr$.
  - inv: $base\_lr(1 + gamma \cdot iter)^{-power}$.
  - $step$: $base\_lr \cdot gamma^{\lfloor \frac{iter}{step} \rfloor}$
  - exp: $base\_lr \cdot gamma^{itern}$
  - dalšími jsou *multistep*, *poly*, *sigmoid*. S těmito pravidly útlumu koeficientu učení se neexperimentovalo a jejich popis funkce lze nalézt v soubotu `caffe.proto` nacházející se ve zdrojových kódech frameworku.
- *weight_decay* regularizační váhový úbytek.
- *momentum*: váha μ předchozí aktualizace vah, viz. 3.1.
- *max_iter*: celkový počet iterací,
- *iter_size*: počet volání funkce `ForwardBackward()`, ztráta se akumuluje a až poté se provede adaptace vah.

3.3.3 Hodnotící metodika

Prvotní zhodnocení použitých architektur probíhalo již při procesu učení a to z charakteristiky jeho průběhu, zejména byl sledován výstup hodnot ztrátové vrstvy dané vztahem 2.4. U poslední ztrátové vrstvy typu *SoftmaxWithLoss* byl z tohoto důvodu nastaven parametr *normalize* na hodnotu `false`, díky tomu je možné alespoň orientačně porovnávat průběh učení mezi jednotlivými modely síti.

Výstup procesu segmentace po softmax normalizaci dává hodnotu konfidence v rozsahu $0 - 1$, což je míra přesvědčení, zda se na daném pixelu nachází oběť (1) nebo, zda se jedná o pixel náležící pozadí (0). Rozhodnutí může být dáno určitou prahovou hodnotou θ. Volbou prahu ovlivňujeme citlivost detekce.

Pro porovnání jednotlivých modelů sítí budeme využívat charakteristiky ROC[3] a Recall - Precision křivek, které se hodí k hodnocení binární (oběť/pozadí) klasifikace. Binární masku vytvoříme porovnání s prahem θ pohybující se v rozsahu $\langle 0; 1\rangle$.

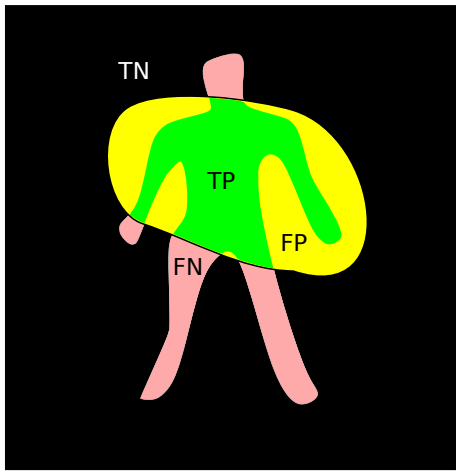
Recall udává úspěšnost detekce oběti a precision porovnává počet falešně pozitivních pixelů vůči pravdivě pozitivních. Specificita pak v našem případě udává úspěšnost klasifikace pozadí.

$$\begin{aligned} \text{Recall (Sensitivity, True Positive Rate)} &= \frac{TP}{TP+FN}, \\ \text{Precision (Positive Predictive Value)} &= \frac{TP}{TP+FP}, \\ \text{Specificity (True Negative Rate)} &= \frac{TN}{FP+TN}, \end{aligned} \tag{3.13}$$

kde TP: true positive, TN: true negative, FP: false positive a FN: false positive jsou definované jako:

$$\begin{aligned} TP &= GT_V \cap OUT_V, \quad & FP &= GT_{Bg} \cap OUT_V, \\ TN &= GT_{Bg} \cap OUT_{Bg}, \quad & FN &= GT_V \cap OUT_{Bg}, \end{aligned} \tag{3.14}$$

přičemž GT_V/GT_{Bg} je oblast v ground truth patřící oběti V respektive pozadí Bg. Podobně OUT_V/OUT_{Bg} je segmentovaný výstup a oblasti pro oběť a pozadí. Dané oblasti jsou kvantifikovány počtem pixelů.



Obrázek 3.4: Příklad ground truth a výstupu segmentace.

ROC křivka ukazuje počet správně klasifikovaných (True Positive Rate, Recall, Sensitivity) vůči počtu nesprávně klasifikovaných negativních (False Positive Rate, $1-$Specificity), v našem případě ,pixelů. Tato charakteristika může dávat velmi optimistické pohledy na klasifikační algoritmy, a to například v případě kdy počet negativních příkladů (počet pixelů pozadí) značně převyšuje počet těch pozitivních (počet pixelů náležící oběti). Velká změna počtu falešně pozitivních klasifikací pak vede k malé změně ve False Positive Rate v ROC křivce [24]. Proto

[3]Receiver Operating Characteristic

v hodnocení byla použita také Recall - Precision charakteristika, která v našem případě poskytuje o něco lepší hodnocení, protože podchycuje efekt nepoměru pozadí a oběti ve snímcích.

Pro klasifikátor je cílový tvar křivky u Recall - Precision v pravém horním rohu. To představuje co možná nejmenší falešně negativních (FN) a falešně pozitivních (FP) případů. A naopak u ROC je ideální průběh co nejblíže levému hornímu rohu. To zajistíme velkým počtem TN případů a malým počtem falešně negativních (FN).

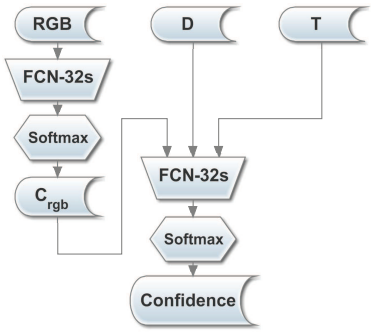
3.3.4 Segmentační architektury a adaptace parametrů

3.3.4.1 Přeučení FCN-32s

Nejprve bylo nutné přeučit převzatou síť FCN-32s na nových RGB datech 3.2. K vybavení sítě při trénování byl vytvořen definiční soubor `RGB_train.prototxt`, rychlost učení byla volena pro všechny vrstvy stejná a pro upsampling vrstvu nulová. Metoda solver byla nastavena pomocí `RGB_solver.prototxt` s *lr_policy* : "*inv*". S parametry *base_lr*, *gamma* a *power* bylo během učení manipulováno pro postupné snížení koeficientu učení a jemnému doladění parametrů. Celkový počet iterací byl 80000 a učení trvalo tři dny. Samotná segmentace pouze z RGB kanálů dosahuje velmi dobrých výsledků. Výsledné charakteristiky na testovacích a reálných datech jsou vykresleny například na obrázcích 3.8 a 3.9 v podobě modrého průběhu. Tento průběh Recall-Precision a ROC charakteristik byl použit jako srovnávací reference pro všechny ostatní sítě.

3.3.4.2 Přídání D-T složek

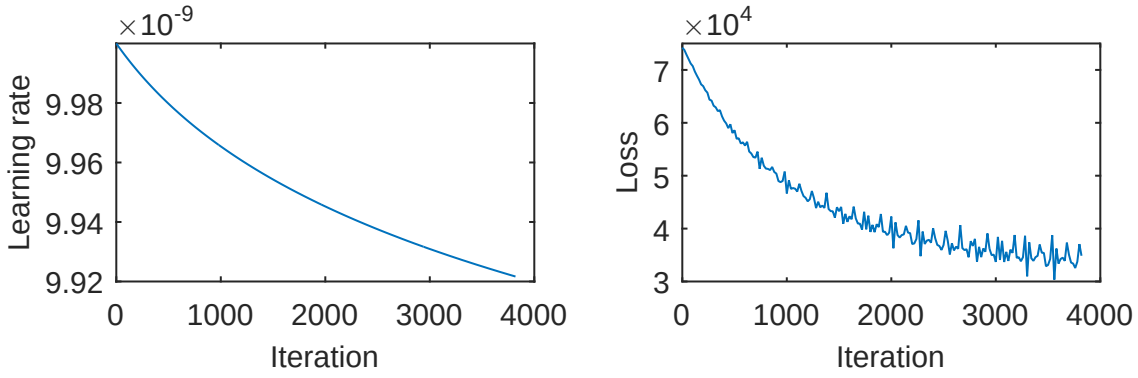
K zakomponování dalších D-T kanálů se nabízí opětovné použití FCN-32s a její přeučení na vstupních datech s kanály C_{RGB}-D-T, kde C_{RGB} je segmentační konfidence získána pomocí vztahu 2.6. Struktura je zobrazena na obrázku 3.5:



Obrázek 3.5: Příklad kaskádního zapojení dvou FCN-32s sítí

Při učení však hodnota ztrátové funkce začala fluktuovat jak lze vidět na grafu 3.6 okolo

relativně vysoké hodnoty 30000, což je více než u ostatních vyzkoušených struktur dále. Na vině může být nedostatek trénovacích dat a celková složitost sítě.

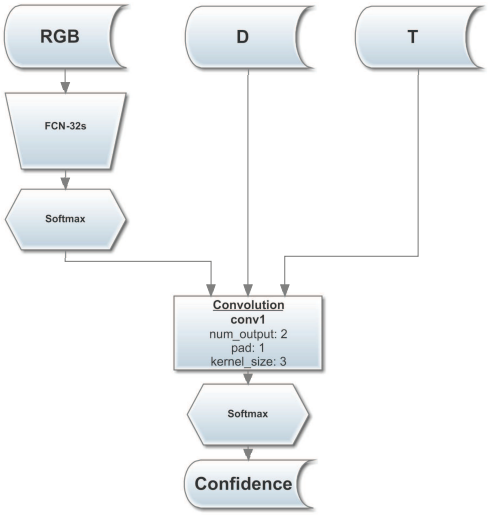


Obrázek 3.6: Průběh učení

Vyzkoušeny byly také struktury vytvořené na základě FCN-32s odebráním několika posledních konvolučních vrstev. Vždy byla odebrána jedna struktura konvolučních vrstev následovaných pooling vrstvou. Podíváme-li se na podobu FCN-32s v příloze A.1, tak první odebraná část byla struktura od `conv5_1` až po `pool5` včetně. Všechny následující vrstvy počínaje `fc6` pak změnili strukturu a bylo nutné je učit od počátku znova na nových datech. Z tohoto důvodu bylo učení rozděleno do dvou fází.

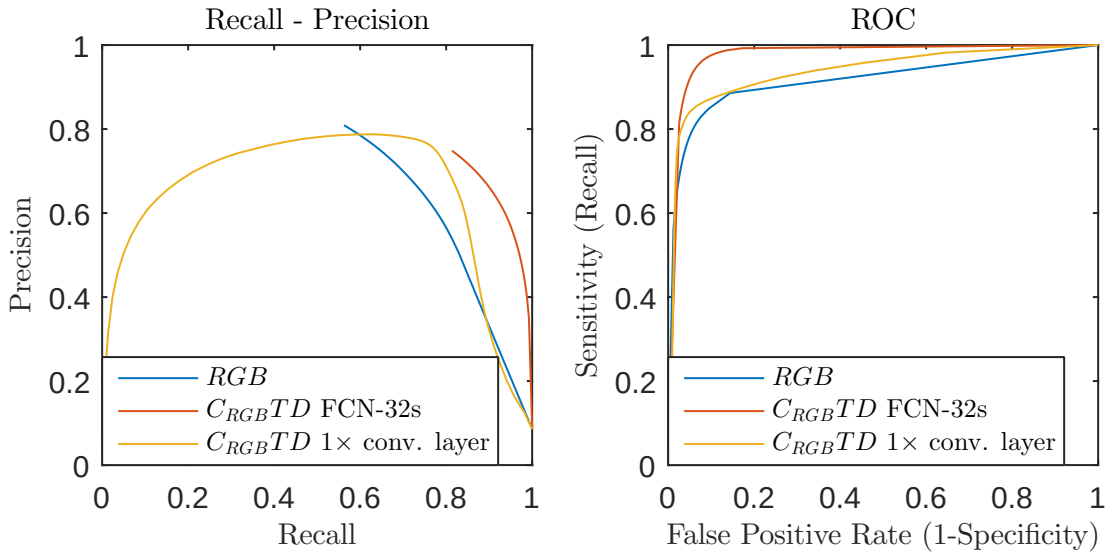
Nejprve byly použity pro nepozměněné vrstvy původní naučené váhy z [2] a ty byly zafixovány. Vrstvy, které se díky zásahu změnily byly inicializovány vahami s malou hodnotou kolem nuly Gaussovým rozdělením. Pro tento účel byl nastaven $weight\_filler$ na typ $gaussian$ se směrodatnou odchylkou $\sigma = 0.1$. U poslední upsampling dekonvoluční vrstvy pak byly nastaveny konstantní váhy a nenulové. Po ustálení hodnoty ztrátové funkce během učení bylo učení spuštěno na všech vrstvách. Ani s postupným snižováním parametru $weight_decay$, nebylo dosaženo lepšího výsledku než hodnoty ztrátové funkce pohybující se kolem 20000. Ani další postupné zjednodušování sítě včetně odebrání dropout vrstev u `fc6` a `fc7` nevedlo ke zlepšení výsledků.

Další pokus směřoval v postavení vlastní jednodušší architektury konvoluční sítě s využitím pouze jedné konvoluční vrstvy:



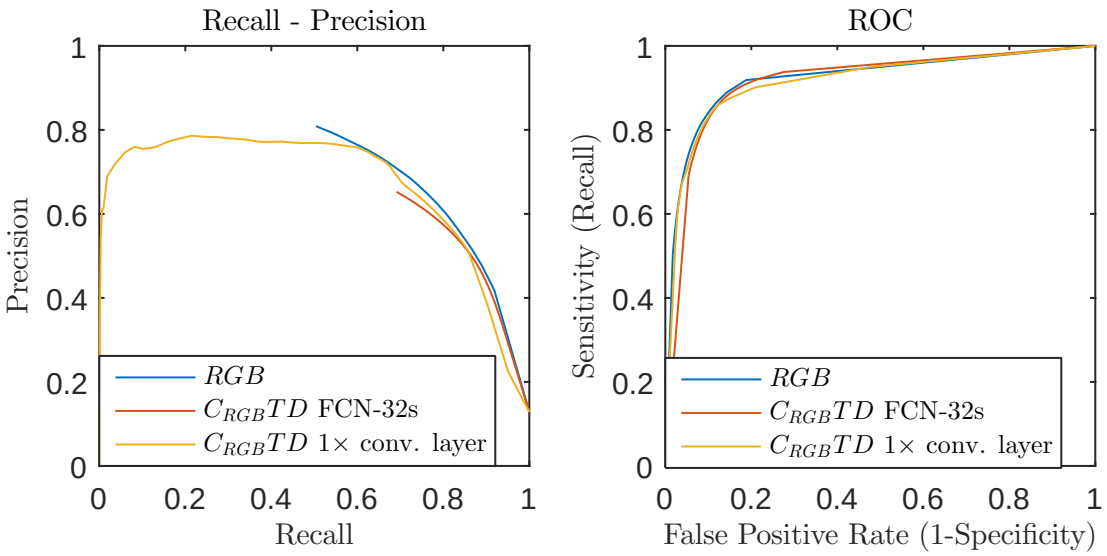
Obrázek 3.7: Spojení s jednou konvoluční vrstvou

Porovnání jednotlivách modelů na testovacích a reálných datech můžeme vidět na následujících grafech 3.8 a 3.9. U testovacích dat bylo vidět na křivkách zlepšení po přidání D-T složek.



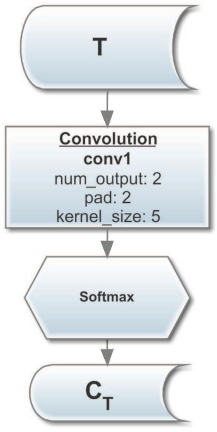
Obrázek 3.8: Testovací data: RGB, C_{RGB}TD FCN-32s, C_{RGB}TD Conv1

U reálných dat však křivky víceméně splývají se křivkou RGB segmentace.

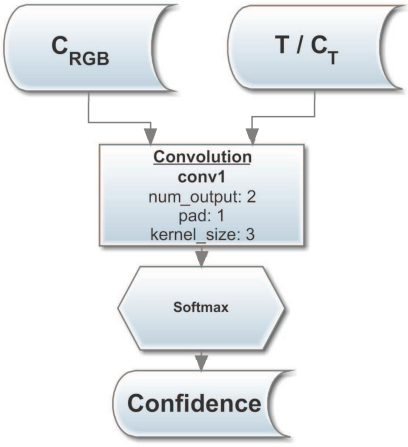


Obrázek 3.9: Reálná data: RGB, $C_{RGB}TD$ FCN-32s, $C_{RGB}TD$ Conv1

K nalezení finální segmentační struktury sítě pomohlo zhodnotit přínos jednotlivých D-T složek. K tomuto účelu byla provedena segmentace pouze z nich 3.10. Opět pomocí jednovrstvé konvoluční sítě. Pro velikost konvolučního jádra byla zvolena velikost 5. Opět jsme získali pomocí softmax hodnotu konfidence C_T (respektive C_D).



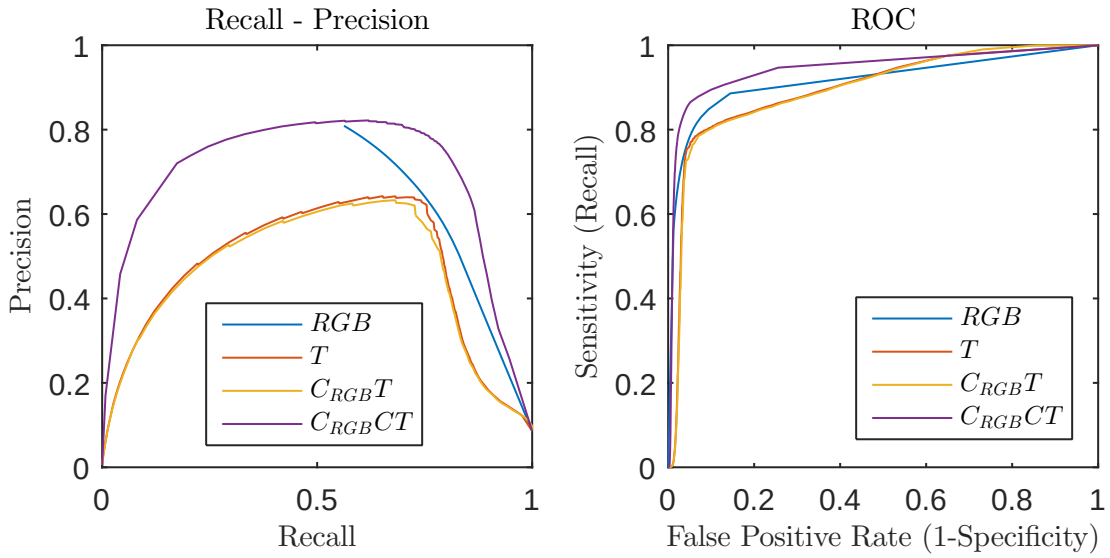
Obrázek 3.10: Jednovrstvá konvoluční síť pro segmentaci pomocí teploty



Obrázek 3.11: Síť $C_{RGB}T$ a $C_{RGB}C_T$

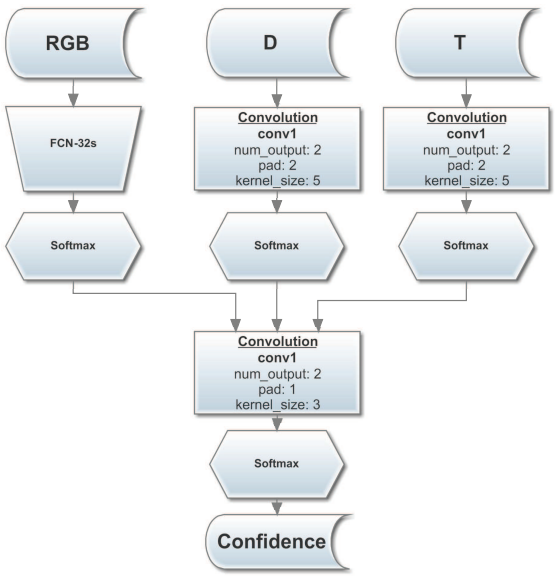
Ke zhodnocení přínosu k RGB byla vytvořená dvoukanálová struktura 3.11 k segmentaci pomocí C_{RGB} a samotné teploty T popřípadě již segmentované konfidence C_T. Při pohledu na hodnotící křivky 3.12 testovacích dat, přesněji na Recall-Precision charakteristiku, je vidět, že segmentace pouze z teploty T nedosahuje kvalit RGB sítě. A překvapivě se nekonal ani

žádný nárůst po přidání C_{RGB} složky. Značný posun k lepšímu nastal ve spojení C_{RGB} a C_T. Z největší pravděpodobností je to dané neseparovatelnou segmentační úlohou pouze pomocí jedné konvoluční vrstvy. Funkce softmax a přidání další vrstvy výsledek značně vylepšilo.

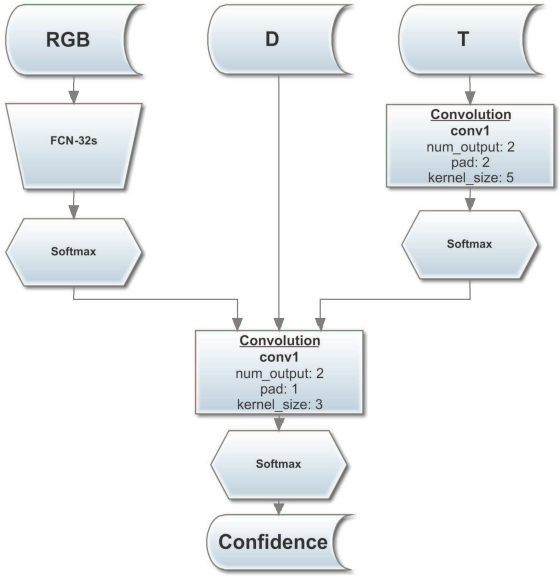


Obrázek 3.12: Testovací data: RGB, $C_{RGB}T$, $C_{RGB}C_T$

Stejný experiment byl proveden i v případě hloubkové informace. Následné spojení jednotlivých verzí dohromady viz. obrázky 3.13 a 3.14.



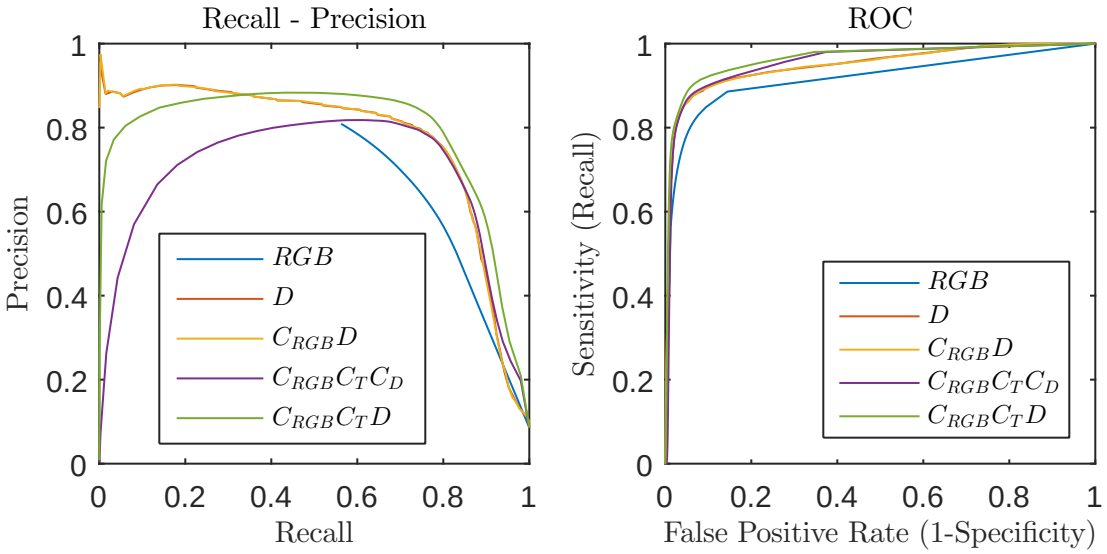
Obrázek 3.13: Struktura sítě $C_{RGB}C_TC_D$



Obrázek 3.14: Struktura sítě $C_{RGB}C_TD$

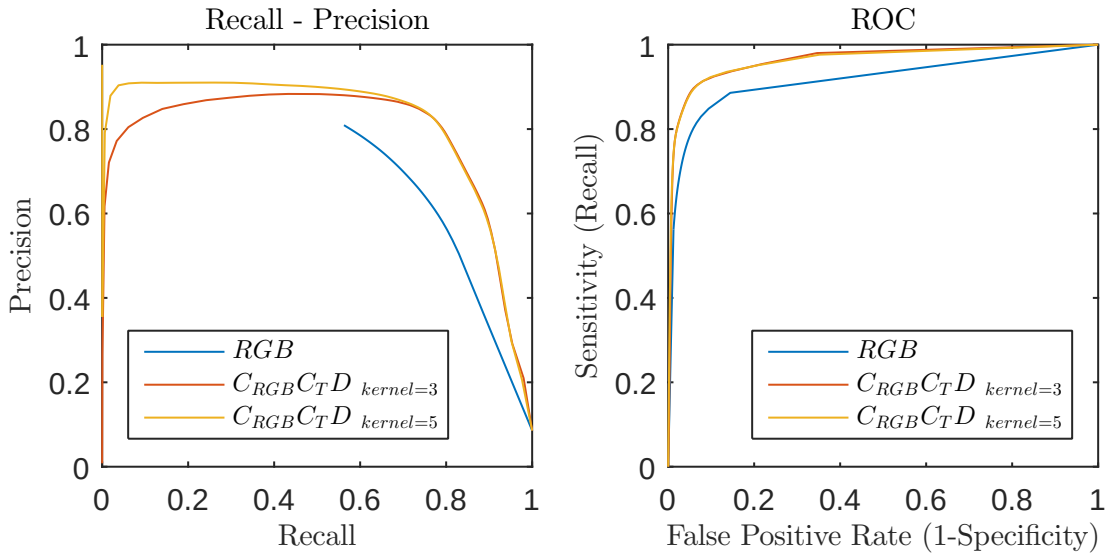
Vyhodnocení těchto spojení na testovacích datech lze vidět na grafech 3.15. Překvapivý výsledek udává segmentace pouze z hloubkové složky, který u Recall-Precision na testovacích

předčil i složitou sít pro RGB segmentaci. Přidání její konfidence C_{RGB} nepomohla stejně, jako tomu bylo u teploty. Dalším krokem byl pokus o přidání složky D nebo konfidence C_D k síti $C_{RGB}C_T$. Výsledek v případě připojení konfidence C_D zůstal téměř shodný jako samotná segmentace dle $C_{RGB}C_T$. Po přidání samotné hloubky D nastalo mírné zlepšení. Tato skutečnost může být dána charakteristikou dat. Je velká pravděpodobnost, že oběť bude na hloubkových datech blíže než pozadí a je možné využít velmi jednoduchého klasifikátoru s velmi dobrými výsledky. U teploty tato skutečnost nemusí platit, jelikož teplota oběti nemusí být vždy větší nebo menší než teplota pozadí.



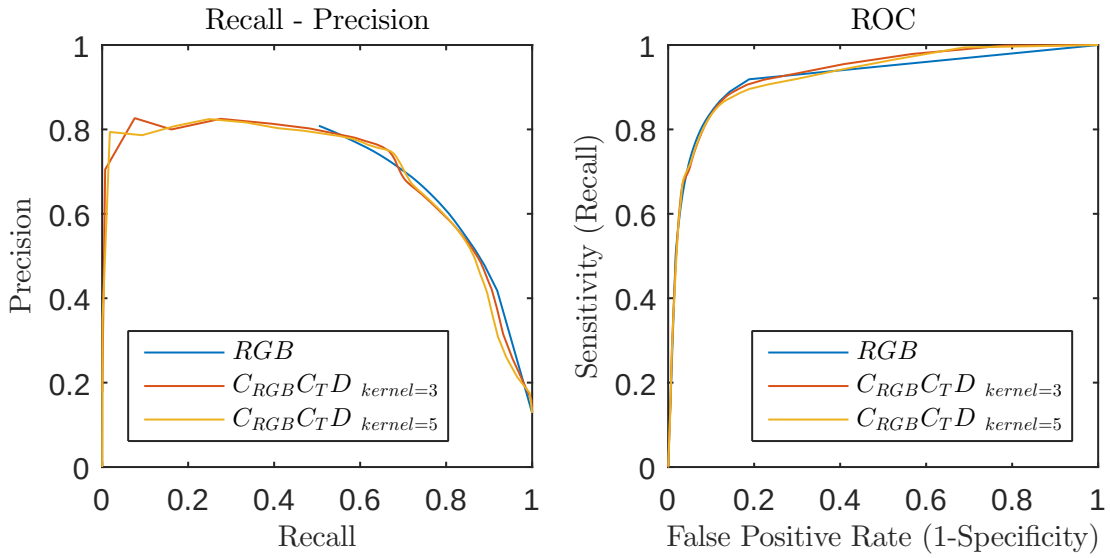
Obrázek 3.15: Testovací data: RGB, D, $C_{RGB}D$, $C_{RGB}C_TC_D$, $C_{RGB}C_TD$

U struktury $C_{RGB}C_T$, která poskytuje relativně dobré výsledky na testovacích datech, bylo provedeno mírné zvětšení konvolučního jádra z hodnot 3×3 na hodnoty 5×5. Tím byl získán o něco lepší tvar Recall-Precision křivky na testovacích datech 3.16.



Obrázek 3.16: Testovací data: RGB, $C_{RGB}C_TD$ kernel=3, $C_{RGB}C_TD$ kernel=5

U reálných dat je situace daleko horší. Výsledky jednotlivých struktur jsou téměř shodné s RGB sítí 3.17. Na vině můžou být velké rozdíly mezi povahou generovaných dat a těch reálných.



Obrázek 3.17: Reálná data: RGB, $C_{RGB}C_TD$ kernel=3, $C_{RGB}C_TD$ kernel=5

Pomocí těchto výsledků byla vybrána pro segmentaci RGB dat přeučená síť FCN-32s. Pro segmentaci RGB-D dat bude sloužit síť $C_{RGB}D$ a pro trojici RGB-D-T struktura $C_{RGB}C_{T}D$ s velikostí jádra 5×5.

Kapitola 4

Implementace na robotovi

Na robotovi byla implementována aktivní detekce obětí s využitím dříve rozebraných poznatků a naučených modelů umělé neuronové sítě pro segmentaci. Implementace je rozdělena na dvě části: segmentační a na část starající se o řízení polohovatelné pan-tilt jednotky a tím tedy řízení pohledu termokamery. Robot běží na robotickém operačním systému ROS, který poskytuje framework pro jazyky C++, Python a Lisp [25]. Segmentační a řídicí část se spouští pomocí ROS launch souboru `thermo_view_seg_ctrl.launch`.

V systému robota běží několik podpůrných jednotek (nodes), které zajišťují potřebné kamerové, hloubkové, teplotní snímky a transformace mezi souřadnými soustavami robota, popřípadě mapy obsazenosti okolí, kterou tvoří node starající se o mapování a navigaci robota. Dále běží na robotovi potřebný ovládací node pan-tilt jednotky. Pro jednoduché spuštění těchto podpůrných jednotek byl vytvořen soubor `thermo_view_control_support.launch` integrující všechny potřebné launch soubory:

- `mapAndNav.launch` zajišťuje mapování a navigaci,
- `octomap_occupancy.launch` vytváří mapu obsazenosti,
- `ptu.launch` spouští node ovládající pan-tilt jednotku,
- `xthermo_drivers.launch` ovladač termokamery,
- `lb_crop_depth.launch` mapuje hloubkové data na RGB snímky,
- `thermo_to_lb_crop.launch` mapuje termosnímky na RGB,
- `rgb_voxels_save.launch` spouští node, který byl vytvořen za účelem uložit RGB obarvenou prostorovou mapu do textového souboru. Naslouchá příkazu s ukládací cestou na topicu *save_rgb*.

4.1 Segmentace

Segmentační část a tedy i práce s neuronovou sítí byla na robotovi implementována v jazyce Python, který podporuje jak ROS tak i *Caffe*. K tomuto účelu byl vytvořen ROS node s názvem *caffe_victim_segmentation*, který odebírá RGB-D-T data a publikuje segmentační výsledek.

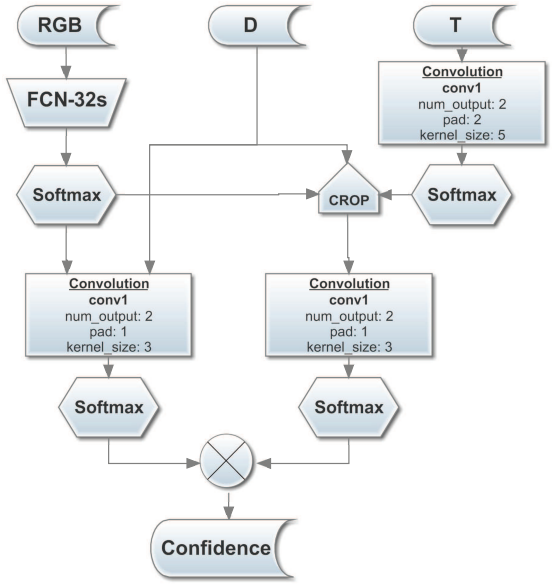
Jako zdroj RGB dat slouží přední dvě kamery z LadyBug III. Na tyto snímky je namapovaná hloubková informace z laserového skeneru a snímek z termokamery. K tomuto účelu byl využit již implementovaný node a díky tomu je k dispozici dvojice RGB-D-T snímků. Teplotní informace může být z části prázdná, jelikož termokamera se nemusí svým zorným polem překrývat s RGB-D složkami. Stejně tak i hloubková informace nemusí být konzistentní.

Segmentační node je možné spustit zvlášť souborem `victim_segmentation_2cam.launch`, ve kterém lze zadat parametrem, zda se bude při průchodu sítí využívat GPU nebo CPU. Na robotovi není grafický adaptér podporující CUDA výpočty a je nutné využít CPU. Díky tomu se segmentace značně zpomalí, přičemž největší časový interval zabere průchod velkou RGB FCN-32s sítí. Z tohoto důvodu byla zvolena i menší velikost vstupních RGB dat a tudíž i hloubkového snímku. Rozlišení dvojice RGB-D je $200 \times 308\,px$. Pro porovnání s touto velikostí snímku probíhá segmentace na grafické kartě s využitím CUDA a knihovnou cuDNN[1] v řádu desítek milisekund. Na mobilním procesoru Intel i7-3520M a kompilací *Caffe* s knihovnou Intel MKL[2] průchod strukturou 4.1 v průměru na jeden RGB-D-T snímek trvá $4\,s$. Na robotu s procesorem Intel i7-4860EQ a knihovnou OpenBLAS[3] je to kolem $18\,s$, jelikož na robotovi běží spoustu dalších procesů vytěžující procesor.

Node spouští Python skript `ros_2cam_seg_CrgbCTD.py`. Podpůrné funkce a konstanty jsou importovány z `support_ros_segmentation.py`.

Pro implementaci návrh z obrázku 3.14, který se skládá z tří nezávislých neuronových sítí, upravíme do podoby 4.1. Tj. provést segmentaci v plném zorném poli RGB a hloubkové informace. Dalším krokem v případě kdy termo-snímek je v překryvu s daným RGB-D provedeme segmentaci RGB-D-T, ale pouze z výřezu. Velikost výřezu je dána validní oblastí teplotní informace. U nezpracovaných dat se jedná o nulové hodnoty, a proto je jednoduché výřez termo-snímku najít. Chybějící data u ve vnitřní části hloubkového nebo teplotního snímku jsou interpolována podobně jako při učení nebo u pasivní detekce 3.2.0.1. Segmentovaný výstup je ve výsledku složený ze segmentace RGB-D a z výřezu RGB-D-T.

[1] https://developer.nvidia.com/cudnn
[2] https://software.intel.com/en-us/intel-mkl
[3] http://www.openblas.net/



Obrázek 4.1: Implementační struktura RGB-D-T segmentace

V každém segmentačním cyklu se segmentuje dvojice RGB-D-T dat a to až po příjmu všech tří RGB-D-T složek. Úprava dat je před vstupem do sítě se shoduje s přípravou trénovací množiny 3.2. Po průchodu sítí se publikuje segmentační obrázek ve formě konfidence na topicu *CrgbCTD_cam1* (resp. *CrgbCTD_cam2*) a dále obrázek na topicu *valid_T_cam1* (resp. *valid_T_cam2*), který poskytuje informaci o oblastech, kde chyběla validní teplotní složka, tzn. i včetně oblastí doplněných interpolací.

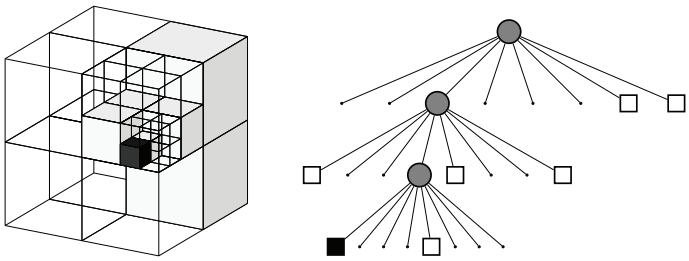
4.2 Řízení termokamery

Tato část se bude v konečném důsledku starat o zpracování segmentačních dat, detekci oběti ve formě konfidenční mapy a řízení termokamery. Je psána v jazyce C++ a postavená s využitím upravených částí kódu, tříd a funkcí od Ing. Tomáše Petříčka, které se týkají tvorby modelu prostorové mapy a procházení jednotlivých pohledů termokamery.

4.2.1 Mapování prostoru

Robotické aplikace často vyžadují prezentaci prostoru kolem sebe. Jednou z možností je využít OctoMap framework [3], vyvinutý právě pro efektivní mapování 3D prostoru. Je založený na oktalové stromové struktuře (Octree) 4.2 a umožňuje pravděpodobnostní reprezentaci pro modelování nezmapovaných oblastí. K dispozici je open-source knihovna pro jazyk C++. Knihovnu využívá třída `OctomapUserModel` od Ing. Tomáše Petříčka umožňující vytvářet vlastní modely

voxelových map a poskytující potřebné funkce. Voxel je jednotka v prostorové (x, y, z) mřížce. Podobně jako pixel u 2D obrázku nabývá voxel i určité hodnoty. Může se jednat například o pravděpodobnost obsazení daného místa v prostoru, kde se voxel nachází. Na rozdíl od výše zmíněné OctoMapy, založené na stromové struktuře, je voxel nedělitelný. Za voxelovou mapu pokládáme uskupení voxelů v prostoru. Rozlišení mapy je dáno objemovou velikostí jednotlivých voxelů. Dále se pracuje s point cloud reprezentací. Jedná se o množinu bodů v prostoru.



Obrázek 4.2: Octree [3]

4.2.2 Algoritmus řízení kamery

Fixní poloha kamery byla zvolena pro testovací a experimentální účely. Funkce udržuje pouze konstantní polohu kamery.

Kontinuální swing kamery prochází jednotlivé diskrétní pohledy, například ve vodorovném směru.

Greedy algoritmus Jedná se o navržené řízení polohovatelné pan-tilt jednotky na níž je umístěna termokamera využívající segmentační konfidenci. Cílem tohoto algoritmu je navrhnout pohled do míst, kde je to dle daného kritéria nejužitečnější.

Díky znalosti transformace mezi segmentační mapou a jednotlivými pohledy je možné určit u každého pohledu množinu viditelných voxelů. K sestavení této množiny byla využita již hotová funkce `visibleVoxels`. Budeme předpokládat, že konfidenční hodnota voxelu rovnající se $C(x, y, z) = 0$ představuje pozadí a $C(x, y, z) = 1$ oběť. Pro experimenty stanovíme práh nerozhodnosti $C_{th} = 0.5$. Muže být však být libovolný. Algoritmus vybere pohled s maximální hodnotou *score*, která je daná sumou vzdálenostní přes množinu viditelných voxelů od nadroviny dané tímto prahem. Pro každý pohled i se prochází

$$\begin{aligned} C_{diff}(x, y, z) &= \left| C(x, y, z) - C_{th} \right|, \\ score_i &= \sum_{x,y,z} \left[C_{th} - C_{diff}(x, y, z) \right]. \end{aligned} \tag{4.1}$$

Maximální přírůstek je právě při hodnotě $C(x, y, z) = C_{th} = 0.5$ a nulový pro $C(x, y, z) = 0$ a $C(x, y, z) = 1$. Suma neprobíhá přes voxely viditelné v daném pohledu, u kterých už známe hodnotu segmentace s využitím teploty. To by mělo zabránit uvíznutí ve sledování stále stejné oblasti v případě, že teplotní složka segmentaci nikterak neupřesní.

4.2.3 Implementace

Ke zpracování segmentace a řízení termokamery slouží ROS node *thermo_view_control* typu nodelet. Spuštění se provádí pomocí `termo_view_control.launch` a celý zdrojový kód se nachází v `thermo_view_control_nodelet.cpp`.

Nodelet odebírá následující ROS topics:

- *octomap* - jedná se o voxelovou mapu obsazenosti poskytující node mapování a navigace.
- *cloud_cam1*, *cloud_cam2* - segmentační informace ve formě prostorového point cloudu. Tuto informaci publikuje node s názvem *seg2points_cam1* (resp. *seg2points_cam2*). Node je typu `depth_image_proc/point_cloud_xyzi_radial`, který je součástí ROS. Odebírá hloubkové a segmentované obrázky publikované segmentační částí a vytváří z nich prostorové point cloud struktury. Díky tomu se přijatý segmentační point cloud dá zakomponovat do segmentační voxelové mapy.
- *depth_cam1_info*, *depth_cam2_info* - informace patřící jednotlivým hloubkovým snímkům.
- *thermo_cam_info* - informace o termokameře.
- *valid_T_cam1*, *valid_T_cam2* - obraz s informací o validitě teplotních dat. Díky tomu víme, pro které body segmentace byla použita teplotní informace.
- *save_signal* - po přijmu informace s cílovou souborovou cestou uloží voxelové mapy do textového souboru, který je možné načíst pro další zpracování například v programu Matlab.
- *tf* - obsahuje veškeré potřebné transformace mezi jednotlivými souřadnými systémy robota a kamer.

Nodelet publikuje:

- *control* - vyslání pan-tilt dvojice parametrů k polohovatelné jednotce kamery,
- *marker* - prostorové markery obsahující hodnotu segmentační konfidence,
- *thermo_marker* - prostorové markery obsahující informaci, které oblasti jsou pokryté segmentací s teplotními snímky.

Celý řídící node pracuje v cyklu. Po přijmu mapy obsazenosti se provádí "prořezávání" voxelových map. Jedná se o dvě instance třídy OctomapUserModel. Prostorově jsou shodné, jelikož jsou vytvářeny ze stejné mapy obsazenosti. První mapa obsahuje segmentační konfidenci

a druhá slouží k uložení informace, zda daná hodnota odpovídá segmentaci s využitím teplotní informace.

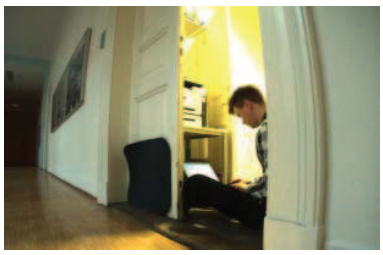
Po každé příchozí segmentačním point cloud struktury jsou výše zmíněné mapy aktualizovány. Hodnoty voxelů v segmentační mapě jsou dány aritmetickým průměrem staré hodnoty a nové příchozí. V případě aktualizace hodnoty tvořené segmentací s teplotou je možné využít vážený průměr. Váhu je možné nastavit parametrem $thermoUpdateWeight$ ve spouštěcím souboru. Samozřejmostí je možnost zabránění přepisu hodnoty tvořené segmentací s teplotou. Zároveň se s každou provedenou aktualizací publikují prostorové markery s barevnou informací ve formě segmentační konfidence oběti.

V případě, kdy jsou k dispozici výše zmíněné modely voxelových map, je spuštěna řídící smyčka kamery. Periodu, s jakou se řídicí smyčka volá, lze nastavit parametrem $control\_hz$. Dalším parametrem $control\_mode$ se volí druh algoritmu řízení, které jsou popsané v 4.2.2. Každý algoritmus vrací index, adresující jeden z diskrétních pohledu termokamery a jsou reprezentovány dvojicí kloubových souřadnic pan-tilt polohovatelné jednotky robota. Tyto jednotlivé souřadnice jsou nahrány formou parametrů ze spouštěcího souboru. Podle dvojice dané indexem pohledu, je vytvořena zpráva pro polohovatelnou jednotku a je publikována. Zpráva je přijata nodem ovládající pan-tilt jednotku a natočena do požadovaného směru.

Kapitola 5

Experiment

K ověření konceptu, byl zvolen experiment s robotem a průjezdem chodbou kolem otevřených dveří, ve kterých na zemi sedí figurant představující potenciální lidskou oběť. Pro demonstraci algoritmu byl proveden první průjezd s fixně namířenou termokamerou ve směru průjezdu robota a druhý průjezd s greedy algoritmem řízení kamery 4.2.2.



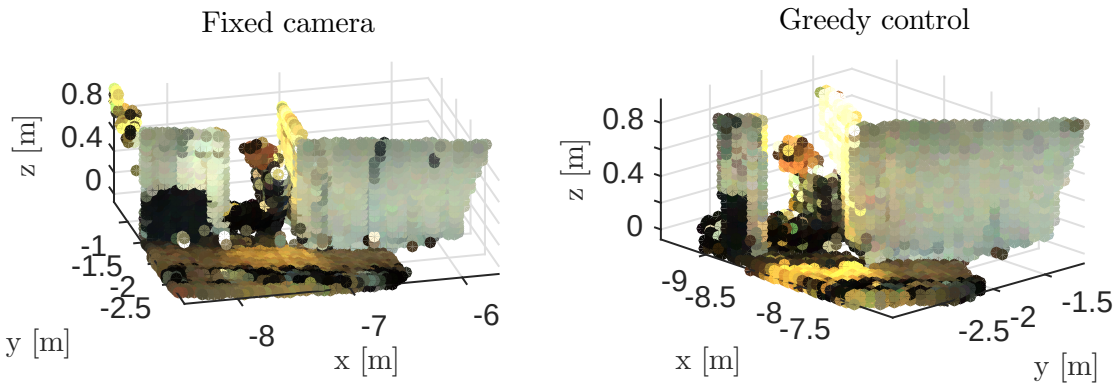
Obrázek 5.1: RGB snímek zachycený během experimentu robotem.

Lze namítat proč nebyly zvoleny i jiné možnosti řízení kamery, jako je například vodorovný swing zleva doprava, náhodný výběr pohledu nebo postupné snímání celého prostoru. Důvodem je, že tyto ostatní metody a jejich výsledky nejsou srovnatelné jelikož závisí na počáteční inicializaci experimentu a také na náhodě. A například postupné snímání celého prostoru musí být v konečném důsledku nakonec nejpřesnější neboť umožňuje pokrýt teplotní informací celý rozsah RGB-D snímků, ale je časově nejnáročnější z důvodu segmentace více snímků a také pohybu kamery.

Segmentace RGB-D-T snímku byla prováděna vždy se stojícím robotem. Pohyb kolem dveří s obětí byl tedy diskrétní z důvodu relativně dlouhé segmentační doby. Při každé zastávce proběhla segmentace čtyř RGB-D-T snímků (2×2 z levé a pravé kamery). Dva snímky při zastavení a po zpracování další dva po natočení termokamery dle řídícího algoritmu.

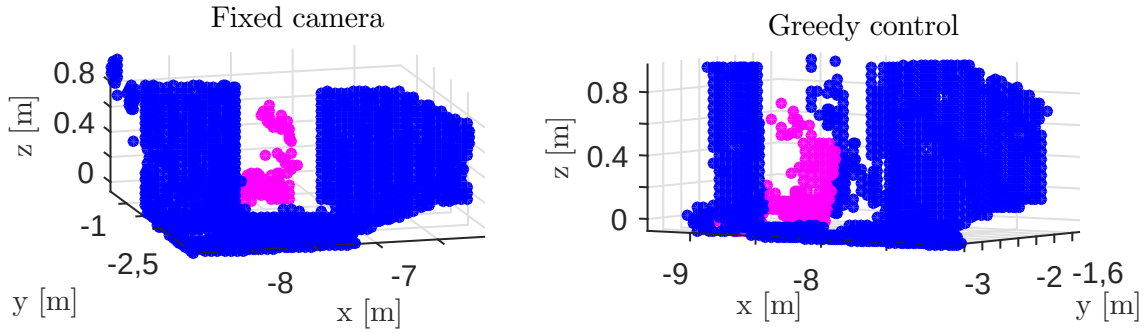
Na obrázku 5.2 lze vidět výřez voxelové RGB mapy. Ačkoli byla snaha o průjezdy s totožnou trajektorií a zastávkami, přesto vygenerovaná voxelová mapa se může trochu lišit. Ze zobrazených map lze vidět, že počátek mapování není u obou experimentů shodný, jelikož mapování

bylo spuštěno na rozdílných místech. To však nevadí neboť vyhodnocení probíhalo u každého experimentu zvlášť.



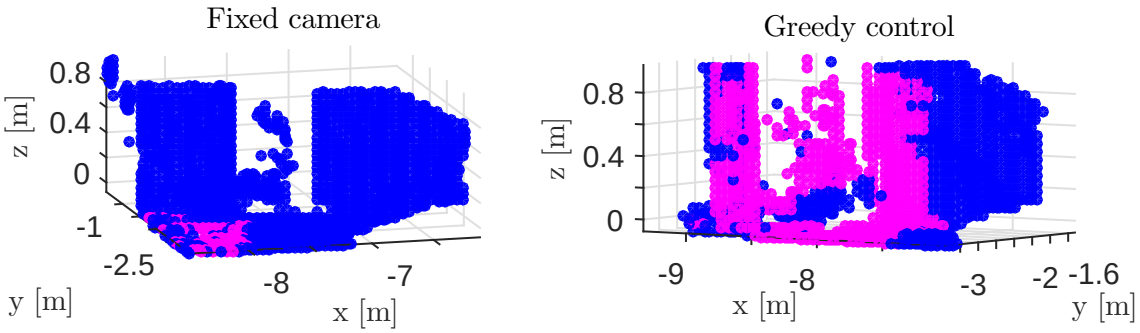
Obrázek 5.2: Výřez RGB mapy s figurantem

Další voxelové mapy jsou tvořeny voxely u kterých známe segmentační informaci, proto se liší od RGB mapy. Ta je tvořena obarvením mapy obsazenosti. Na obrázku 5.3 je možné vidět ručně označenou polohu oběti.



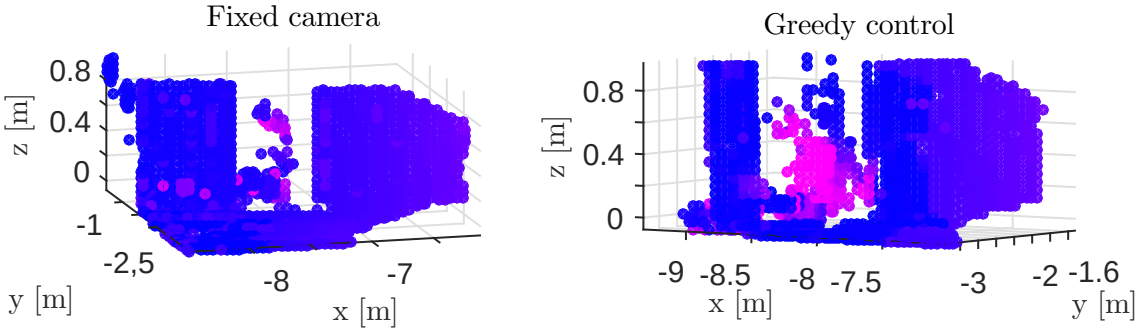
Obrázek 5.3: Výřez s označeným figurantem v mapách

Na pokrytí teplotní informací 5.4 je dobře viditelné, že v případě greedy algoritmu se kamera otočila směrem k oběti. U fixní kamery tato oblast nebyla při průjezdu termokamerou nasnímána.



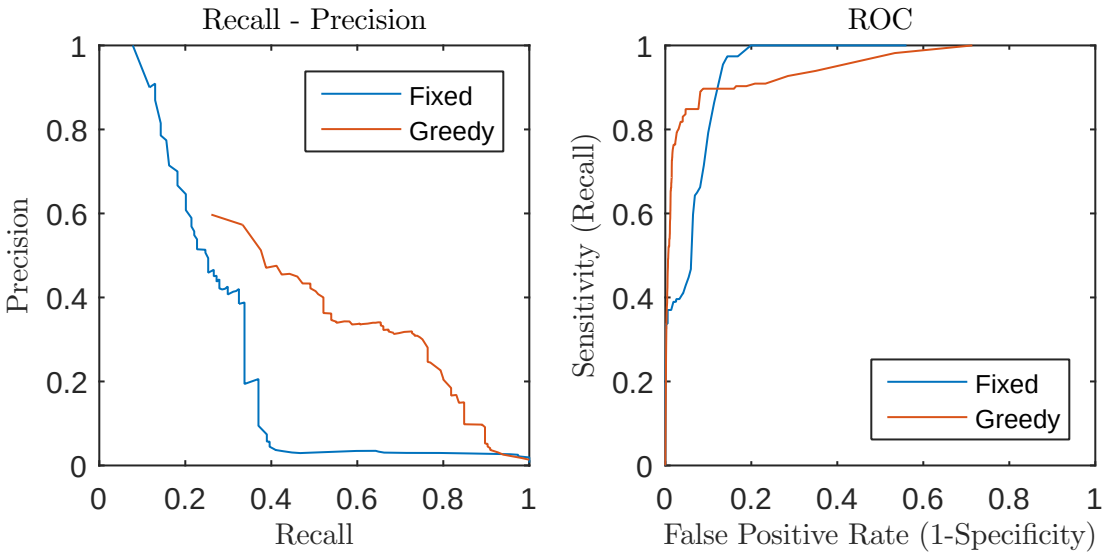
Obrázek 5.4: Výřez z mapy pokrytí termosnímky kolem figuranta

Výřez segmentační konfidence mapy lze vidět na obrázku 5.5. Už na první pohled je vidět výraznějšího zbarvení v oblasti, kde se nachází figurant.



Obrázek 5.5: Výřez segmentační mapy v okolí figuranta

Při vykreslení Recall-Precision charakteristiky a ROC křivky na obrázku 5.6 je vidět zlepšení u Greedy algoritmu řízení pohledu termokamery, který zajistil teplotní pokrytí oblasti s figurantem.



Obrázek 5.6: Recall-Precision a ROC křivky

Výsledky potvrzují funkčnost konceptu. V reálném experimentu se ukázal přínos teplotní složky při upřesnění detekce oběti. K získání binární mapy s předpokládaným umístěním oběti je nutné zvolit rozhodovací práh θ. Jeho volbu nám můžou usnadnit vykreslené křivky 5.6.

Kapitola 6

Závěr

6.1 Shrnutí dosažených výsledků

Cílem práce bylo seznámení se s problematikou učení hlubokých konvolučních sítí a s jejich aplikačním využitím. Dále navržení a naučení sítě pro segmentaci RGB-D-T dat. Pomocí výsledku pak řídit pohled kamery provádějící snímání teplotní informace míst, kde selhává RGB-D segmentace. Posledním cílem byla integrace celého řešení do prostředí ROS na platformu robota. Všechny tyto dílčí cíle byly úspěšně splněny.

Po nastudování teorie hlubokých konvolučních sítí a seznámení s implementací frameworku *Caffe* pro učení a práci s neuronovými sítěmi, se podařilo navrhnout a naučit jednotlivé architektury sítí pro segmentaci z RGB, RGB-D a RGB-D-T. Jednotlivé modely dosahovaly velmi dobrých výsledků na testovacích datech. U reálných dat už nebyl přínos hloubkové a teplotní informace tak citelný. Na vině jsou rozdíly mezi generovanou trénovací, validační, testovací množinou dat a reálnou množinou. S využitím těchto a naučených modelů sítí byl následně úspěšně implementován segmentační algoritmus na robotovi. Pro řízení pohledu termokamery byl navrhnut Greedy algoritmus, který vybere z množiny diskrétních poloh kamery, umístěné na pohyblivé jednotce, optimální pohled dle zadaného kritéria. Důkaz funkčnosti tohoto algoritmu byl demonstrován experimentem s robotem a figurantem. Při experimentu se kamera natočila očekávaným směrem a výsledek segmentace byl dle Recall-Precision a ROC křivek lepší než v případě zafixovaného pohledu ve směru pohybu robota.

6.2 Návrh vylepšení

Rozdílný výsledek vyhodnocení na testovacích a reálných datech by se dal eliminovat při dostatku reálných dat promícháním s generovanými daty.

Další rozvoj práce by mohl být v postavení vlastní RGB-D-T hluboké konvoluční sítě a

trénování z významně větší množiny dat. K sestavení nové trénovací sady dat by bylo možné například využít ImageNet databázi. Výzvou pro další řešení je získat hloubková, teplotní data a vytřídit z obrovského množství RGB obrázků pro tento účel relevantní.

Nevýhodou zvoleného řešení je časová náročnost segmentace, která roste s velikostí sítě. Segmentace na procesoru robota trvala zhruba 20 s. To je velmi malá frekvence, se kterou můžeme v reálném čase segmentovat okolí. Tento nedostatek se však se zvyšováním výpočetního výkonu sníží. Už nyní na výkonných grafických kartách trvá segmentace v řádek desítek milisekund. Řešením by bylo přidání právě GPU akcelerátoru do výbavy robota nebo využít při dobrém spojení vzdálený výpočet na výpočetním serveru.

Ačkoli je zvolené řešení dnes omezeno dostupným výkonem, využití hlubokých konvolučních sítí bude mít určitě, při dynamickém růstu výpočetního výkonu, v budoucnu velký úspěch.

Literatura

[1] Materials, C. C.: CS231n: Convolutional Neural Networks for Visual Recognition.
URL `http://cs231n.github.io/`

[2] Long, J.; Shelhamer, E.; Darrell, T.: Fully Convolutional Networks for Semantic Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[3] Hornung, A.; Wurm, K. M.; Bennewitz, M.; aj.: OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 2013, doi:10.1007/s10514-012-9321-0, software available at `http://octomap.github.com`.
URL `http://octomap.github.com`

[4] ŠPANĚL, M.; BERAN, V.: Obrazové segmentační techniky: Přehled existujících metod. *Brno: VUT v Brně*, ročník 12, č. 2005, 2006: s. 19–1.

[5] Everingham, M.; Van Gool, L.; Williams, C. K. I.; aj.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[6] Everingham, M.; Eslami, S. M. A.; Van Gool, L.; aj.: The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, ročník 111, č. 1, Leden 2015: s. 98–136.

[7] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, editace F. Pereira; C. J. C. Burges; L. Bottou; K. Q. Weinberger, Curran Associates, Inc., 2012, s. 1097–1105.
URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`

[8] Russakovsky, O.; Deng, J.; Su, H.; aj.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, ročník 115, č. 3, 2015: s. 211–252, doi: 10.1007/s11263-015-0816-y.

[9] LeCun, Y.; Bottou, L.; Bengio, Y.; aj.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, ročník 86, č. 11, 1998: s. 2278–2324.

[10] Everingham, M.; Van Gool, L.; Williams, C. K. I.; aj.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.
URL `http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/ECVNET/CROWLEY1/node1.2.html`

[11] Ladybug3 1394b. `https://www.ptgrey.com/ladybug3-360-degree-firewire-spherical-camera-systems`, accessed: 2016-05-15.

[12] LMS1xx 2D laser scanners. `https://www.sick.com/de/en/product-portfolio/detection-and-ranging-solutions/2d-laser-scanners/lms1xx/c/g91901`, accessed: 2016-05-15.

[13] Datasheet: thermoIMAGER TIM 160. `http://www.micro-epsilon.cz/temperature-sensors/thermoIMAGER/thermoIMAGER_160/index.html`, accessed: 2016-05-15.

[14] Yangqing Jia, E. S.: Caffe: Deep learning framework by the BVLC. 2015.
URL `http://caffe.berkeleyvision.org/`

[15] Jia, Y.; Shelhamer, E.; Donahue, J.; aj.: Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[16] Springenberg, J. T.; Dosovitskiy, A.; Brox, T.; aj.: Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[17] Srivastava, N.; Hinton, G.; Krizhevsky, A.; aj.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, ročník 15, 2014: s. 1929–1958.
URL `http://jmlr.org/papers/v15/srivastava14a.html`

[18] Ng, A.; Ngiam, J.; Foo, C. Y.; aj.: UFLDL Tutorial: Deep Learning Tutorial. Březen 2013.
URL `http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/`

[19] Bottou, L.: Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, Springer, 2012, s. 421–436.

[20] RNDr. PaedDr. Eva Volná, P.: *Studijní materiály pro distanční kurz: Neuronové sítě 1.* Ostravská univerzita v Ostravě, 2008.

[21] D'Errico, J.: inpaint-nans. 2012.
URL `http://www.mathworks.com/matlabcentral/fileexchange/4551-inpaint-nans`

[22] Chu, H.: Lightning Memory-Mapped Database Manager (LMDB). Symas Corporation, 2015.
URL `http://symas.com/mdb/doc/`

[23] Yamaguchi, K.: Matlab LMDB wrapper for UNIX environment. 2015.
URL `https://github.com/kyamagu/matlab-lmdb`

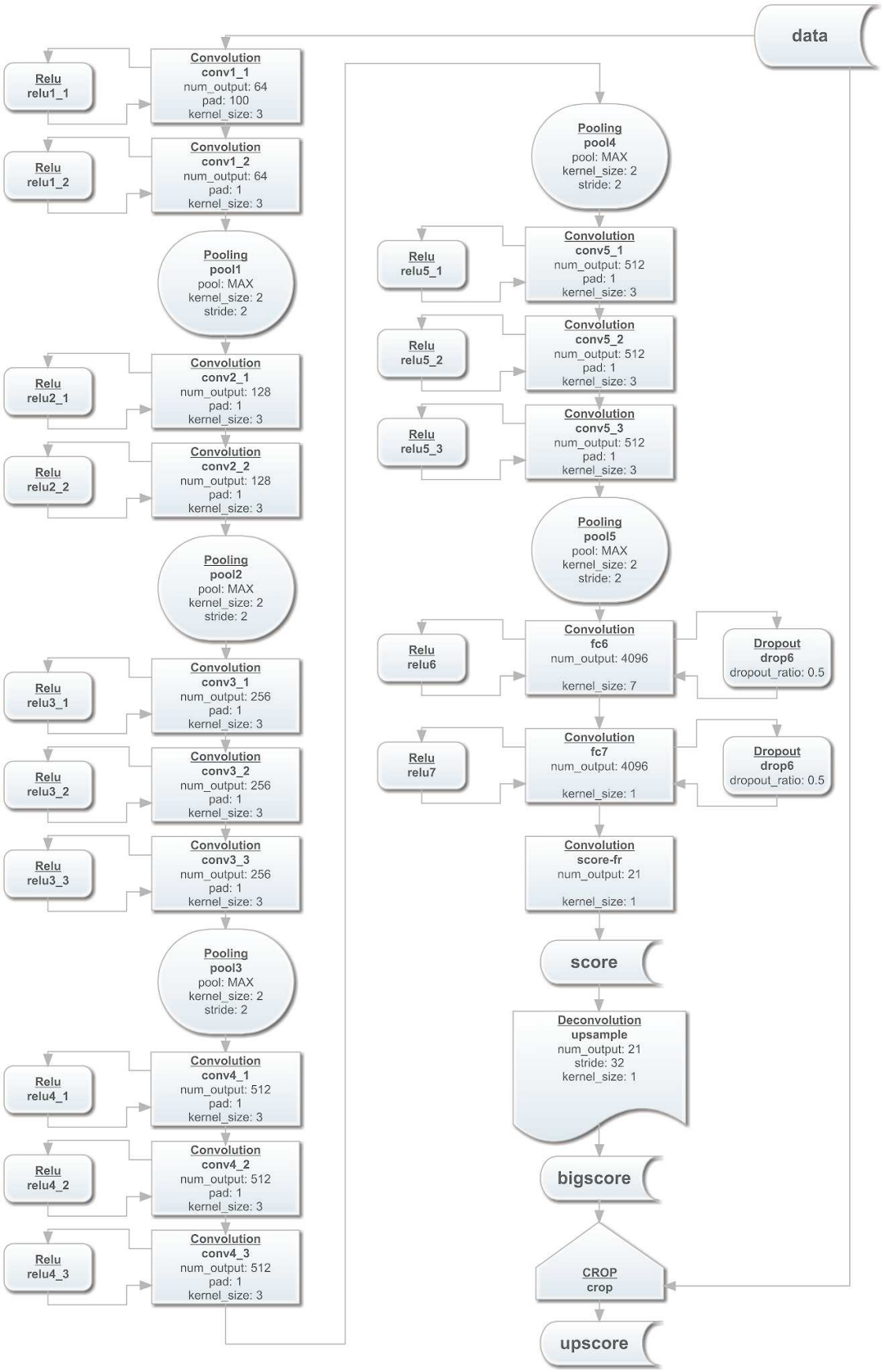
[24] Davis, J.; Goadrich, M.: The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, s. 233–240.

[25] ROS Wiki Documentation. `http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm`, accessed: 2016-05-12.

Příloha A

Struktura sítě FCN-32s

Podrobná architektura modelu FCN-32s.

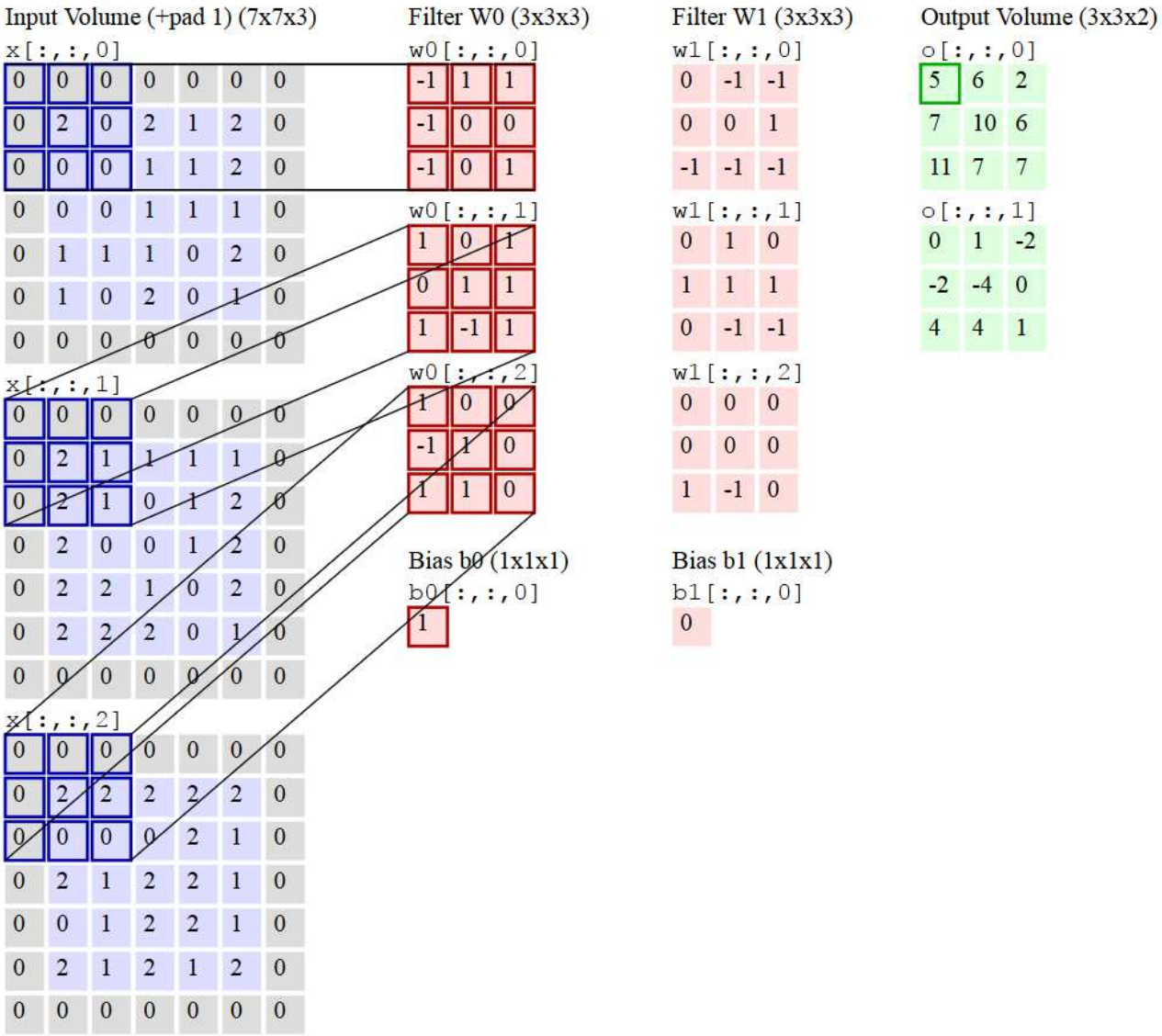


Obrázek A.1: FCN-32s PASCAL [2]

Příloha B

Princip konvoluční vrstvy

Princip konvoluční vrstvy s hloubkou $num\_output = 2$, konvolučním jádrem $kernel\_size = 3$ a vstupem (obrázkem se 3 kanály) o rozměrech $7 \times 7 \times 3$ doplněný o $pad = 1$.



Obrázek B.1: Princip konvoluční vrstvy [1]

Příloha C

Obsah přiloženého DVD

K této práci je přiloženo DVD, na kterém jsou uloženy zdrojové kódy a elektronická verze práce ve formátu pdf.

Adresářová struktura přiloženého DVD:

- .\dip_prace_tezkyjir.pdf
- .\zdrojove_kody:
  - .\learning: Adresář obsahuje skripty a funkce sloužící k úpravě dat a k učení jednotlivých sítí.
  - .\robot: V této složce se nacházejí zdrojové kódy náležící implementaci na robotovi.
  - .\experiments: Adresář obsahuje skripty, který byly použity pro vyhodnocení experimentů. Obsahuje také zaznamenaná data.
  - .\models: Zde jsou uloženy modely jednotlivých sítí a jejich definiční soubory.