Bakalárska práca

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Model aktívnej kontúry s kľúčovými bodmi pre detekciu hranice objektu v obraze

Filip Dvořák

Školiteľ: doc. Ing. Radim Šára, Dr. Tech.
Odbor: Kybernetika a robotika
Zameranie: Robotika
August 2017

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Filip D v o ř á k

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Model aktivní kontury s klíčovými body pro detekci hranice objektu v obrazu

Pokyny pro vypracování:

Cílem projektu je navrhnout vhodnou metodu obohacení standardního model aktivní kontury (tzv. snake) o klíčové body rozpoznané detektorem na základě klasifikátoru klíčových bodů.

1. Seznamte se s modelem aktivní kontury používaným v počítačovém vidění a počítačové grafice.
2. Vytvořte detektor klíčových bodů na základě jednoduchého klasifikátoru okolí obrazových bodů.
3. Modifikujte algoritmus detekce aktivní kontury tak, aby pracoval s informací z detektoru klíčových bodů.
4. Metodu implementujte a otestujte na detekci siluety stojící postavy v přirozeném prostředí.

Seznam odborné literatury:

[1] Yu, T. - Luo, J. - Ahuja, N: Search Strategies for Shape Regularized Active Contour. Computer Vision and Image Understanding 113, pp. 1053-1063, 2009.
[2] Blake, A. and Isard, M: Active Contours. Springer 1998.
[3] Sonka M. - Hlavac, V. - Boyle, R: Image Processing, Analysis, and Machine Vision. Thomson Learning, 2007.
[4] Bishop, C: Pattern recognition and machine learning. Springer 2006.

Vedoucí bakalářské práce: doc. Ing. Radim Šára, Dr. Tech.

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 28. 1. 2016

Poďakovanie

Týmto ďakujem doc. Ing. Radimovi Šárovi, Dr. Tech. za jeho pomoc, trpezlivosť a cenné rady, ktoré som zúžitkoval nielen pri prenikaní do problematiky a princípov počítačového videnia, ale aj v živote mimo školu.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky dostupné informačné zdroje v súlade s Metodickým pokynom o dodržiavaní etických princípov pri príprave vysokoškolských záverečných prác.

V Prahe dňa 7. augusta, 2017

Abstrakt

Táto bakalárska práca pojednáva o problematike a návrhu vhodnej metódy pre obohatenie štandardného modelu aktívnej kontúry o kľúčové body. Kľúčové body sú rozpoznávané detektorom, ktorý funguje na základe klasifikácie vybraných objektov v obraze. To všetko za účelom detekcie ľudskej kontúry v obraze.

Kľúčové slová: aktívne kontúry, snakes, hľadanie kontúr, ľudské kontúry, rozpoznávanie v obraze, dynamické programovanie, Viola-Jones

Školiteľ: doc. Ing. Radim Šára, Dr. Tech.

Abstract

This thesis discusses a method for enchacement of the standard model of an active contour (ie. Snake) for key points. Key points are recognized by a detector which works on the basis of classification of selected objects in the image. The target application is detection of human body contours in images.

Keywords: active contours, snakes, contour detection, human contour detection, dynamic programming, Viola-Jones

Title translation: Model of active contour with key points used for edge detection in picture

Obsah

Obrázky

Tabuľky

Kapitola 1
Úvod

Táto práca sa zaoberá detekciou kontúry ľudskej postavy v dvojrozmernom obrázku. V rámci detekcie samotnej kontúry práca popisuje aj detekciu kľúčových bodov popisujúce konkrétne objekty v obraze. Informácia o polohe, prípadne vzájomných vzdialenostiach jednotlivých objektov v obraze, môže významne ovplyvniť kvalitu a presnosť detekcie ľudskej kontúry. Pretože bežná ľudská postava má niekoľko kĺbov, existuje veľké množstvo rôznych natočení jednotlivých končatín, a teda veľké množstvo odlišných, no pritom správnych riešení úlohy hľadania kontúry ľudskej postavy v obraze. Ďalším faktorom je oblečenie, ktoré môže zhoršovať presnosť celého algoritmu detekcie. Od toho sa odvíjajú pomerne vysoké nároky na robustnosť algoritmu v porovnaní s algoritmami pre detekciu jednoduchších objektov ako sú napríklad autá, značky alebo písmená.

Práca tvorí základy pre ďalšiu aplikáciu, ktorej cieľom by malo byť určovanie reálnych rozmerov postavy človeka na základe nájdenej kontúry v dvojrozmernom obraze. Využitím týchto údajov o výške postavy, šírke končatín alebo trupu by bolo možné dopočítať ďalšie telesné rozmery ako obvod pásu alebo končatín. Tieto miery získané z obrázku by následne bolo možné aplikovať napríklad pri jednoduchom odčítaní mier ľudského tela pre potreby šitia oblečenia na mieru, prípadne získavaní dodatočných biometrických údajov o osobách len za pomoci fotografie. V ďalšom kroku by bolo možné doplniť aplikáciu o modelovanie postáv v trojdimenzionálnom priestore, prípadne zo statického modelovania prejsť do modelovania postáv v trojdimenzionálnom priestore v reálnom čase na základe vstupu z kamery.

Ďalšie využitie práce alebo jej podobných algoritmov pre detekciu tvarov v obraze je široké [3][4]. Algoritmy a programy pre detekciu ľudských postáv, alebo obecne objektov, sú uplatňované hlavne v oblasti bezpečnostných a sledovacích zariadení, v medicínskych aplikáciách, priemyselných systémoch kontroly výrobkov, hernom priemysle, v autonómnej navigácii automobilov a vo veľa ďalších odvetviach. Rozpoznávanie ľudí v obraze je v prípade sledovacích aplikáciách[6] alebo hernom priemysle [7][8] hlavná úloha. Táto schopnosť zohráva dôležitú úlohu aj v systémoch autonómnej navigácie vozidiel a robotov v neznámom prostredí [5], kde do výpočtov pre riadenie vstupujú aj iné objekty a chodci ako účastníci premávky a vozidlo je vedené tak, aby sa im vyhlo. Preto vzrastá dopyt po algoritmoch, ktoré dokážu spoľahlivo,

presne a rýchlo identifikovať v obraze ľudské postavy alebo ich časti.

Pre hľadanie kontúr objektov alebo ich extrakciu z pozadia existuje už viacero funkčných a testovaných algoritmov ako napríklad Snakes [1][11][12], Geodetický model aktívnej kontúry [9] alebo aktívne kontúry bez použitia hrán [13]. Pri voľbe modelu aktívnej kontúry sme sledovali ich stávajúcu zložitosť a princíp na akom detekujú kontúry.

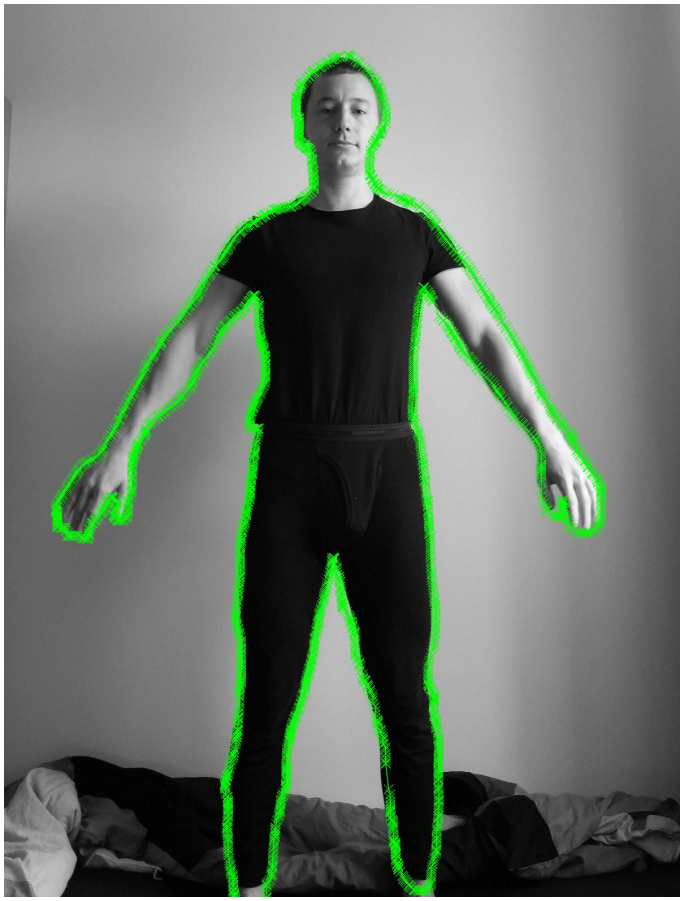
Zaujímavý je postup použitia regulovanej aktívnej kontúry [18] pri detekcii pľúc na obrázku z ultrazvuku, do ktorého je vkladaná dátová mriežka, v ktorej je následne pomocou dynamického programovania hľadá hneď niekoľko najlacnejších ciest pomocou minimalizovania energie obrázka. Zároveň je vytvorený klasifikátor, ktorý vie rozpoznávať tvar kontúry pľúc. To umožňuje kvantifikovať podobnosť s nájdenou kontúrou. Táto podobnosť je potom premietnutá ako cena do celkovej cenovej funkcie.

Nás zaujímalo, či je možné za pomoci modifikovania tohto konceptu dokázať detekovať kontúru ľudskej postavy, a či je možné dosiahnuť lepšie výsledky pri použití nie klasifikátora hodnotiaceho tvar celej krivky, ale pridaním sémantických značiek, v angličtine uvádzané ako *labels*, a to pre jednotlivé body krivky modelu aktívnej kontúry.

Krivka potom pozostáva z bodov, ktoré nadobúdajú určitú postupnosť identifikovaných tried značiek. Táto postupnosť má silnú štruktúru a v klasickom postoji človeka je relatívne jednoducho rozpoznateľná. Informáciou o tejto štruktúre je možné priamo ovplyvniť správanie aktívnej kontúry a jej geometrické vlastnosti ako tuhosť alebo elasticitu v rôznych miestach. Cieľom je identifikovať štruktúru, postupnosť jej bodov, umiestnenie kontúry v obrázku a správnosť riešenia.

Kapitola 2
Formulácia úlohy

Práca sa zaoberá hľadaním ľudskej kontúry v obraze a jej odlíšením od pozadia. To znamená, že ide o určenie presného obrysu ľudskej postavy. Ak je obraz reprezentovaný ako pole pixlov, hľadaná kontúra je krivka, ktorá separuje pixly popisujúce postavu človeka od okolia spôsobom, kedy v ploche opísanej krivkou sú pixly patriace ľudskej postave a mimo krivku sú pixly patriace okoliu. Názorná ukážka možného riešenia úlohy je na obrázku 2.1.



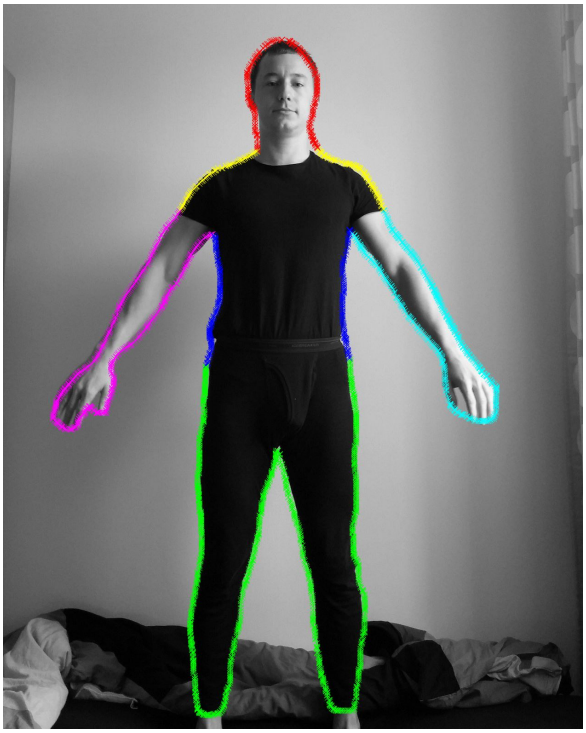
Obrázok 2.1: Ukážka hotovej detekcie kontúry.

Hľadanie takejto kontúry v obraze je obecne často riešené pomocou modelu aktívnej kontúry [1][2]. To znamená za pomoci určitého algoritmu, často iteračného alebo minimalizačného, ktorý prehľadáva obraz až pokiaľ nenájde žiadaný tvar a pozíciu kontúry, respektíve pokiaľ v kroku algoritmu nie je splnená podmienka ukončenia. Jednoduchšie modely aktívnych kontúr pracujú bez komplexnejšej informácie o obraze. Teda bez informácie o vyšších štruktúrach alebo objektoch, prípadne o vzťahoch medzi nimi. Presnosť takýchto modelov klesá s rastúcim šumom v obraze, respektíve s rastúcou zložitosťou obrazu. Krok k zvýšeniu presnosti a správnosti nájdenej kontúry je rozšírenie základného modelu aktívnej kontúry o ďalšie informácie, ako napríklad o pomenovanie konkrétnych objektov a ich polohe v obraze.

Obecne ľudia dokážu v obraze rozlišovať rôzne objekty a tvary. Schopnosti rozlišovať objekty predchádza učenie, kedy získavame s vizuálnou interakciou s objektom zároveň aj jeho popis. Následne pri ďalšej interakcii s objektom vieme tento objekt pomenovať aj bez popisu. Nami vnímaný obraz sa často skladá z viacerých objektov, ktoré takto vieme pomenovať. V rámci každého objektu vieme ísť do detailu a pomenovať aj jeho časti, z ktorých skúmaný objekt pozostáva.

V našom prípade ľudskej postavy vieme rozlíšiť končatiny a časti tela ako hlava alebo trup. Podobným spôsobom, ako sa to učia ľudia, vieme takéto rozlišovanie objektov v obraze naučiť aj počítačový algoritmus [2]. Vytvorením klasifikátora pomocou sady vstupov učíme algoritmus rozpoznávať objekty. Následne takto naučený klasifikátor využívame v detektore objektov v neznámom obraze. Predpoklad je, že na základe výstupu z detektora objektov získame informácie o polohe jednotlivých končatín a častí tela a na základe nich budeme vedieť spresniť inicializáciu kontúry a následné chovanie a správnosť nájdeného riešenia.

Na základe výstupu z detektora by sme mali vedieť určiť polohu, respektíve pravdepodobnosť výskytu konkrétnych častí tela v obrázku. To ideálne pre každý pixel v obrázku. Na základe týchto detekcií by sme mali vedieť priradiť každému pixlu obrázku značku alebo značky tak, že každá značka by sa v danom pixly vyskytovala s určitou pravdepodobnosťou. Tieto značky by mali kladne vplývať na správnosť chovania modelu aktívnej kontúry. Výsledná kontúra vykreslená farebne tak, aby odlíšila jednotlivé značky, a teda časti tela:



Obrázok 2.2: Ukážka hotovej detekcie kontúry s priradenými značkami.

Na obrázku 2.2 máme zobrazenú výslednú a žiadanú kontúru ľudskej postavy v obraze. Červená farba pre vrcholy kontúry, ktorým bola priradená značka *hlava*. Žltá farba pre vrcholy s priradenou značkou *ramená*, fialová pre vrcholy s priradenou značkou *pravá ruka*, modrá pre vrcholy so značkou *trup*, zelená pre vrcholy so značkami *nohy* a tyrkysová pre vrcholy, ktorým bola priradená značka *pravá ruka*.

2.1 Úlohy zadania

Hlavnou úlohou je nájsť kontúru ľudskej postavy v obraze. Tá môže byť tvorená uzatvoreným cyklom, respektíve postupnosťou vrcholov v obrázku. Táto postupnosť potom môže byť vyjadrená ako $\{\mathbf{v_i}\}_{i=0}^{n}$ s podmienkou, že $v_n = v_0$, čím je podmienené toto uzatvorenie. Takto zadefinovaná postupnosť vrcholov tvorí hľadanú kontúru, ktorá sleduje obrys postavy v obrázku.

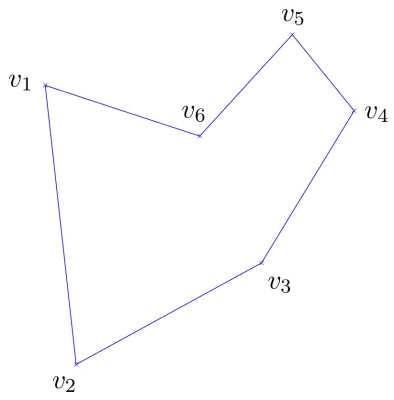
Môžeme povedať, že kontúra je popísaná uzatvorenou Jordanovou krivkou C, ktorá je súvislá a nepretína sama seba. Krivka C je v obrázku reprezentovaná ako polygón po sebe idúcich n vrcholov v_i, kde každý vrchol je určený polohou v obrázku, ktorý je reprezentovaný ako pole vrcholov, a to na súradniciach x a y. Pre polygón v obrázku potom platí:

$$C = \{v_i\}_{i=0}^{n} \tag{2.1}$$

Kde je uzatvorenie polygónu, respektíve cyklickosti postupnosti vrcholov dosiahnutá podmienkou:

$$v_0 = v_n \tag{2.2}$$

Príklad polygónu, pozostávajúceho zo šiestich vrcholov, náhodne umiestnených v obrázku 2.3.



Obrázok 2.3: Detail kontúry o počte vrcholov n=6.

S takýmto polygónom sa pracuje v rámci modelu aktívnej kontúry. Pre možnosť vytvorenia a úpravy chovania tohto polygónu v obrázku potrebujeme pracovať s viacerými informáciami, než je základná informácia o hodnotách jasu v jednotlivých pixloch. Potrebujeme pridať informáciu o hranách v obrázku. Každému vrcholu v_i tohto polygónu vieme podľa jeho súradníc priradiť hodnotu $\mathbf{x_i}$ z hranového obrázku I_{edge}. Hranový obrázok dostaneme úpravou pôvodného obrázku pomocou aplikácie filtra. Hodnoty z hranového obrázku nám o danom vrchole hovoria, ako výrazná hrana je v danom mieste obrázku. V obrázku nás zaujímajú miesta, kde je táto hrana najvýznamnejšia, keďže hrana ako taká popisuje mieru, ako veľmi sa mení detail obrázku v danom mieste. Hrany tvoria často hranicu medzi objektmi alebo štruktúrami v obraze [1]. Predpokladáme teda, že v súradniciach pixloch popisujúcich hľadanú kontúru bude veľkosť hrany veľká.

$$\mathbf{x_i} = (x_i, y_i) \tag{2.3}$$

Pridaním informácie o konkrétnych objektov v obrázku vieme ďalej vylepšiť model aktívnej kontúry. Jednotlivým vrcholom teda pridávame informáciu o tom, aký objekt vrchol na daných súradniciach popisuje. V našom prípade rozpoznávame a detekujeme v obrázku polohu a pravdepodobnosť výskytu objektov ako hlava, ramená, ľavá ruka, pravá ruka, trup a nohy. Na základe jednotlivých detekcií sú vrcholom priradené značky. Naviac pridávame značku *žiadne*, ktorú pridáme vrcholom na miestach v obrázku, na ktorých nedošlo k detekcii žiadnej z vyššie uvedených značiek. Teda každý z vrcholov bude mať vždy priradenú značku λ_i z množiny všetkých rozpoznávaných značiek Λ, ktoré v obrázku detekujeme.

$$\lambda_i \leftarrow \Lambda \tag{2.4}$$

$$\Lambda = \{hlava, ramená, pravá\ ruka, ľavá\ ruka, trup, noha, žiadne\} \tag{2.5}$$

Pre získanie informácie o týchto objektoch v obraze potrebujeme vytvoriť klasifikátor a následne detektor jednotlivých objektov tak, aby vedel vrcholom priradiť značku z množiny všetkých značiek Λ. Vytvorenie klasifikátora prebieha učením s učiteľom, teda predložením sady správne označkovaných obrázkov jednej časti tela. Detektor by mal byť schopný dať polohu výskytu jednotlivých objektov a pravdepodobnosť s akou je daný objekt na danom mieste detekovaný.

Priradením značiek jednotlivým vrcholom dokážeme následne segmentovať kontúru na menšie časti. Tieto časti sú definované množinou po sebe idúcich vrcholov $\mathbf{v_i}$, ktorým bola priradená rovnaká značka λ. Táto množina vrcholov potom reprezentuje konkrétnu časť tela alebo končatinu. Farebne je táto segmentácia znázornená na obrázku 2.2. Priradením značky vrcholu dostaneme:

$$\mathbf{v_i} = (\mathbf{x_i}, \lambda_i) \tag{2.6}$$

Úlohou modelu je zaručiť to, že takto definované segmenty kontúry budú závislé od značiek, ktoré sú im priradené. Na základe týchto značiek bude možné ovplyvniť geometrické parametre a celkovo chovanie aktívnej kontúry v celku, ale i v jej určitých segmentoch. Napríklad pre časť kontúry popisujúcej objekt *hlava* predpokladáme veľkú krivosť a určitú dĺžku v porovnaní s ostatnými segmentmi. Naopak pre časť kontúry pozostávajúcej z vrcholov so značkou λ v hodnote *trup* očakávame minimálnu krivosť.

Pre možnosť vyhodnocovania stavu kontúry potrebujeme zaviesť vyhodnocovacie kritérium, teda určitú cenovú funkciu, na základe ktorej definujeme chovanie aktívnej kontúry. Tá by mala byť závislá na hodnotách v jednotlivých vrcholoch krivky na súradniciach, kde sa krivka nachádza v obrázku. Veľkosť, ktorou vplývajú detekované značky alebo prítomnosť hrany na kontúru v určitých súradniciach by malo byť možné riadiť aj externe. Napríklad pomocou externých koeficientov, ktoré buď určíme fixne, alebo sa ich hodnoty naučíme. To všetko by malo vstupovať a popisovať náš model aktívnej kontúry.

Je vhodné určiť pravidlá a predpoklady pre vstupný obrázok tak, aby sme vedeli jednoduchšie overiť funkčnosti nami navrhovaných metód riešenia a modifikovania modelu aktívnej kontúry. Teda v praxi testovali model najprv v kontrolovanom prostredí, čo znamená na jednoduchších a konkrétnejších vstupoch, než riešili hneď komplexnú úlohu všetkých existujúcich natočení postavy. Pod týmito požiadavkami môžeme rozumieť napríklad predpoklad, že postava na vstupnom obrázku vypĺňa určitú časť obrázku, že je natočená tvárou kolmo k snímaciemu zariadeniu, že má ruky od seba a podobne.

2.2 Segmentácia a model aktívnej kontúry

Pri hľadaní ľudskej kontúry v obraze tým segmentujeme postavu v obrázku od pozadia. Metódy segmentácie v počítačovom videní sú algoritmy, ktoré slúžia na analýzu a zistenie vyššej informácie v obraze. Segmentujú v obraze jednotlivé objekty alebo oblasti, ktoré sa vyznačujú spoločnými vlastnosťami. V našom prípade pomocou segmentácie vyčleňujeme z obrazu ľudskú postavu od jej okolia. Na segmentáciu tak komplexného objektu nám nevystačia základné metódy. Medzi tieto základné metódy segmentácie patrí napríklad *thresholding* [1], ktorý segmentuje objekty na základe hodnôt jasu jednotlivých pixlov podľa toho, či hodnota v pixly prekročí určitú hranicu T. Ďalšou metódou je napríklad metóda sledujúca hranu regiónu [1], ktorá sa vyznačuje od okolia napríklad určitou hodnotou jasu v pixloch. Tieto metódy vychádzajú len zo základnej informácie v obrázku, teda najčastejšie len z hodnôt jasu jednotlivých pixlov, prípadne z masky získanej aplikovaním filtra na pôvodný obraz. Správnosť výstupu týchto základných metód segmentácie klesá s rastúcim šumom a množstvom informácie v obraze, teda samé o sebe ich pre riešenie nášho problému detekcie kontúry nie je možné využiť.

Pri skúmaní komplexnejších a pokročilejších algoritmov pre segmentáciu a detekciu hrán v obraze, ako napríklad level sets [1], geodetické modely aktívnych kontúr [9] alebo aktívne kontúry bez použitia hrán [13], sa pre naše potreby zdal byť ako najatraktívnejší model aktívnej kontúry, ktorý je v literatúre často nazývaný aj ako *Snake* [1] [11] [12]. Model aktívnej kontúry *Snake* je reprezentovaný deformovateľnou parametrickou krivkou umiestnenou v obraze, ktorá sa vyznačuje určitými vlastnosťami a energiou E_{snake}, ktorá je v tomto modeli minimalizovaná.

$$E_{snake} = \int_0^1 (\ E_{int}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s)) + E_{con}(\mathbf{v}(s))\ )\ \mathrm{d}s \tag{2.7}$$

Energia E_{snake} pozostáva z troch výrazov, ktoré vplývajú na chovanie aktívnej kontúry. E_{int} je interná energia kontúry, ktorá vplýva na jej krivosť a ako veľmi sa môže kontúra deformovať. E_{image} je energia obrázku, ktorá vyplýva na priťahovanie aktívnej kontúry k hranám alebo líniám v obrázku. E_{con} reprezentuje vonkajšie sily, ktoré zadá užívateľ, prípadne vstup z algoritmu skúmajúceho obraz na vyššej úrovni. Všetky tieto energie obsahujú konštanty, ktorými je možné ručne regulovať chovanie kontúry. $\mathbf{v}(s)$ je parametrická reprezentácia aktuálnej polohy aktívnej kontúry, kde $\mathbf{v}(s) = (x(s), y(s))$. Nevýhodou modelu *Snake* je, že vyžaduje prvotnú inicializáciu do blízkosti hľadanej kontúry užívateľom. V opačnom prípade by algoritmus pri minimalizovaní energie E_{snake} nemusel nájsť správne riešenie v rámci celého obrázku, ale zostať uväznený v lokálnom minime.

Pri skúmaní ďalších prác z praxe, ktoré sa zaoberajú detekciou a segmentáciou ľudskej postavy alebo iných objektov v obraze, prípadne zisťovaním pózy ľudského tela v obraze len z jedného statického obrázku [14] [15] [16]

[17], sme narazili na prácu zaoberajúcu sa hľadaním obrysu pľúcneho laloka v snímku z magnetickej rezonancie [18]. Hľadanie kontúry prebieha v rámci vyhľadávacieho priestoru pripomínajúceho mriežku, konštruovaného nad veľkou plochou obrazu. Z vrcholov tohto priestoru vzniká graf, v ktorom je pomocou dynamického programovania hľadaná najlacnejšia cesta. Jej energia $E_{image}(c)$ je počítaná z hodnôt v obrázku daných aktuálnym umiestnením vyhľadávacieho priestoru a teda polohou a tvarom kontúry c. Tá môže byť naviac regulovaná pomocou ω. Od konvenčného modelu *Snake* sa líši nahradením vyhladzovacieho člena nelineárnym členom $E_{shape}(c)$, ktorý hodnotí ako veľmi tvar kontúry c odpovedá hľadanému tvaru. Výsledná energetická funkcia, ktorá je minimalizovaná, vyzerá nasledovne:

$$E(c) = E_{shape}(c) + \omega E_{image}(c) \tag{2.8}$$

Výhodou tohto algoritmu je, že pomocou člena $E_{shape}(c)$ rieši problém minimalizácie energie aktívnej kontúry, kedy tá uviazne v lokálnom energetickom minime. V prípade, že kontúra v lokálnom minime síce má nízku energiu $E_{image}(c)$, jej tvar nie je podobný so žiadaným tvarom, a teda člen $E_{shape}(c)$ nadobúda vysoké hodnoty. Tým je výsledná $E(c)$ vysoká a algoritmus pokračuje v hľadaní.

V algoritme je zaujímavé využitie dynamického programovania [1] [19]. Dynamické programovanie pracuje na základe algoritmu hľadajúceho optimálnu cestu v grafe, ktorá je v tomto prípade vyjadrená cenovou funkciou. Výhodou modelu je, že dokáže nájsť globálne optimálny výsledok pri nízkej výpočtovej komplexnosti. Je rozšíriteľná o ďalšie vstupy, o ktoré je možné rozšíriť cenovú funkciu a zohľadniť ich vplyv pri minimalizácii energie kontúry $E(c)$.

Ako problém v uvedenom riešení vidíme nutnosť trénovania množiny správnych tvarov krivky pre možnosť testovania podobnosti v rámci člena $E_{shape}(c)$. To nám príde zbytočne komplikované, keďže pre náš prípad by sme museli ručne vyznačiť kontúry vo veľkom počte obrázkov s postavami, na ktorých je človek, čo by bolo časovo náročné.

Elegantnejšie riešenie je zapracovať tento element vplývajúci na tvar krivky c priamo do dynamického programovania. To tak, aby sme dokázali konkrétnym častiam grafu, respektíve krivky, povedať, aké geometrické vlastnosti má v daných miestach krivka nadobúdať. To pridaním výstupu z detektora, ktorý dokáže jednotlivým vrcholom priradiť značky. Trénovanie klasifikátora pre detektor je časovo menej náročné, keďže nejde o ručné obťahovanie kontúr objektov v obraze.

2.3 Detektor objektov

Detektor objektov v počítačovom videní je nástroj, pomocou ktorého vieme v obraze nájsť konkrétne objekty, to znamená rozpoznať objekt, pomenovať ho a určiť jeho polohu. Tým pridávame do obrázku informáciu, pomocou ktorej vieme pomôcť počítaču alebo algoritmu pochopiť kontext zachytený v obraze.

V praxi ide najčastejšie o detekciu ľudskej tváre [20], prípadne celej postavy [6]. Pokročilejšie algoritmy sa starajú o detekciu objektov v reálnom čase [21].

V našom prípade potrebujeme v obraze klasifikovať a detekovať časti ľudského tela. Základné uvažované algoritmy boli Support Vector Machine [12], inak aj SVM, v kombinácii s metódou *sliding window*, Viola-Jones detektor [25] alebo aj metódy segmentácie pozadia od objektov v obrázku [22] [21] kde po segmentovaní objektov nasleduje ich klasifikácia a detekcia.

Pre našu aplikáciu sme sa rozhodli využiť Viola-Jones detektor, ktorý je čo sa týka behu detekcie, časovo relatívne nenáročný, keďže využíva princíp kaskády klasifikátorov. Ďalšou výhodou by bola aj teoretická využiteľnosť pri ďalšom využití algoritmu pre detekciu ľudskej kontúry v záberoch snímaných v reálnom čase [26]. Zo začiatku sa zdala byť nevýhoda, že z detektora môžme dostať viacero odoziev na odlišných miestach pri detekcii jedného objektu. Teda o takzvané falošne pozitívne detekcie. Vyšší počet týchto detekcií ale nevadí, keďže vieme zohľadniť postupnosť týchto detekcií v rámci značiek priradených jednotlivým vrcholom v kontúre v celkovej cene v rámci modelu aktívnej kontúry. Teda pri postupnosti vrcholov s určitými značkami, ktorých poradie v reálnej ľudskej kontúre nie je možné, napríklad $ruka -> noha -> hlava$, zohľadňujeme túto postupnosť negatívne pridaním vyššej ceny v rámci cenovej funkcie. Teda takáto kontúra je v modeli potlačená.

Pomocou detektora rozoznávame množinu značiek $\Lambda = \{hlava, ramená, pravá\ ruka, ľavá\ ruka, trup, noha, žiadne\}$. Do modelu a jeho vrcholov tak pridávame značky, ktoré popisujú objekty nachádzajúce sa na určitých miestach v obrázku, čím ho dopĺňame o sémantiku. Pomocou informácie o týchto značkách vieme ovplyvniť geometrické vlastnosti ako krivosť, tuhosť alebo celkovú elasticitu kontúry.

2.4 Výstup

Voľbou tejto kombinácie modelu aktívnej kontúry riešeného pomocou dynamického programovania a doplneného detektorom na princípe Viola-Jones získame univerzálnejšiu štruktúru. Jej správanie je možné ovplyvniť napríklad zmenou koeficientov krivosti pre časti kontúry, ktoré majú určitú značku. Výstup z detektora tak má priamy vplyv na elasticitu aktívnej kontúry v určitých jej častiach.

Náš model je, oproti pôvodnému návrhu riešenia [18], možné jednoducho upraviť a aplikovať aj pre úlohy hľadania kontúr iných tvarov, než je kontúra ľudskej postavy. To naučením detektora na rozpoznávanie tvarov iných objektov a zmenou parametrov, ktoré popisujú chovanie časti kontúry v jednotlivých jej častiach.

Zároveň problém nájdenia krivky a jej tvaru nemusíme riešiť separátne, ale priamo v dynamickom programovaní, ktoré vie riešiť túto úlohu rýchlejšie, no hlavne je možné doň zakomponovať ďalšie vstupy, ktorými by bolo možné zlepšiť celkové chovanie modelu.

Jedným z ďalších vstupov do algoritmu, ktorý by sme vedeli pridať na základe prítomných sémantických značiek, a na základe ktorého by sme naviac vedeli modelovať úlohu, je výstup z takzvaného Skrytého Markovského modelu [12]. To by viedlo k ďalšiemu zvýšeniu presnosti detekcie kontúry. Toto rozšírenie je už nad mieru zadania tejto bakalárskej práce, no je to zaujímavá myšlienka k zapracovaniu.

Výstupom je detekovaná kontúra ľudskej postavy vyobrazená na obrázku 2.1, respektíve pri zvýraznených značkách na obrázku 2.2.

Kapitola 3

Algoritmus riešenia úlohy

3.1 Model a jeho prvky

Model aktívnej kontúry je tvorený spojitou uzatvorenou krivkou C, respektíve polygónom, popísaným ako postupnosť vrcholov $\{\mathbf{v_i}|i = 0...n; \mathbf{v_0} = \mathbf{v_n}\}$, a definovaným aj vzorcom 2.1 a 2.2. Aktívna kontúra sa vyznačuje energiou $E(C)$, ktorá je závislá od jej polohy v obraze a od jej geometrického tvaru. Minimálna hodnota tejto energie v obrázku reprezentuje polohu hľadanej kontúry ľudskej postavy. To predstavuje hľadané riešenie úlohy, znázornené aj na obrázku 2.1. Aktívna kontúra sa postupnou minimalizáciou energie $E(C)$ presúva z miesta jej inicializácie do miesta spĺňajúceho podmienky riešenia úlohy.

Chovanie aktívnej kontúry v obrázku je dynamické. Je závislé od informácií obsiahnutých v obrázku. Ide o informácie o hodnotách jasu v konkrétnych miestach a z nich odvodených hodnotách popisujúcich výskyt hrán. Zároveň pracuje s informáciou vyššej úrovne o konkrétnych objektoch. Túto informáciu dodáva detektor objektov. Keďže je model vytvorený pre detekciu ľudskej kontúry, detekujeme objekty odpovedajúce ľudským častiam a končatinám. Každému z nich priradíme značku podľa toho, ktorý detekovaný objekt popisuje. Tieto značky spolu tvoria množinu Λ definovanú v rovnici 2.5.

Tieto informácie sú reprezentované ako hodnoty a značky v jednotlivých vrcholoch $\mathbf{v_i}$, a zároveň v usporiadaných dvojiciach $(\mathbf{v_{i-1}}, \mathbf{v_i})$ alebo trojiciach $(\mathbf{v_{i-1}}, \mathbf{v_i}, \mathbf{v_{i+1}})$ po sebe idúcich vrcholoch polygónu. Vzájomné závislosti postupnosti vrcholov a teda dynamických vlastnosti polygónu C sú vyjadrené pomocou energetickej rovnice ako energia aktívnej kontúry $E(C)$. Tá je vyjadrená ako:

$$E(C, \lambda) = \sum_{i=1}^{n} [E_I(\mathbf{v_i}|\lambda_i) + E_E(\mathbf{v_{i-1}}, \mathbf{v_i}|\lambda_i) + E_G(\mathbf{v_{i-1}}, \mathbf{v_i}, \mathbf{v_{i+1}}|\lambda_i) + E_T(\lambda)] \tag{3.1}$$

Kde E_I je interná energia obrázku, ktorá reprezentuje silu, ktorou je aktívna kontúra priťahovaná k hranám objektov v obrázku. Energia E_E predstavuje internú energiu aktívnej kontúry, teda silu, ktorá pôsobí na jej celkovú dĺžku.

Energia E_G vyjadruje geometrické požiadavky na konkrétne segmenty kontúry. Energia E_T vyjadruje kvalitu postupnosti značiek. Všetky energie sú pritom podmienené a závislé na značkách λ priradených jednotlivým vrcholom.

Interná energia obrázku E_I je závislá na výskyte hrán v obrázku a ich sile. Vychádza z hodnôt daných hranovým obrázkom I_{edge}, ktorý získam detekciou hrán v pôvodnom obrázku I. Energia E_I je vyjadrená ako:

$$E_I(\mathbf{v_i}|\lambda_i) = E_{I_E}(\mathbf{v_i}) + E_\lambda(\mathbf{v_i}|\lambda_i) \tag{3.2}$$

Kde E_{I_E} hovorí, ako výrazná hrana sa nachádza v polohe vrcholu $\mathbf{v_i}$. Ak je vrchol $\mathbf{v_i}$ vyjadrený ako $\mathbf{v_i}(\mathbf{x_i}, \lambda_i)$, potom pre hodnotu hrany $\mathbf{x_i}$ platí $\mathbf{x_i} = I_{edge}(x_i, y_i)$. Hodnota hrany je nezávislá od značky λ. Hodnota značky je v rámci vrcholu vyjadrená pomocou E_λ.

Energia E_E je daná geometrickou vzdialenosťou medzi vrcholom $\mathbf{v_{i-1}}$, ktorý predchádza aktuálnemu $\mathbf{v_i}$. Teda reprezentuje dĺžku segmentu kontúry, ktorý je tvorený týmito dvoma vrcholmi. Energia je naviac ovplyvnená značkami λ priradených jednotlivým vrcholom. Energiu E_E potom môžme vyjadriť ako:

$$E_E(\mathbf{v_{i-1}}, \mathbf{v_i}|\lambda_i) = \alpha(\lambda_{i-1}, \lambda_i).||\mathbf{x_i} - \mathbf{x_{i-1}}||^2 \tag{3.3}$$

Kde $\alpha(\lambda_{i-1}, \lambda_i)$ je nami zvolený koeficient, ktorým riadime postupnosť značiek v rámci segmentov aktívnej kontúry, kedy je v minimalizačnom procese preferovaná časť kontúry, ktorá spája dva vrcholy s rovnakou značkou λ. Nami definované α je vyjadrené z tabuľky v závislosti od značiek vrcholov $\mathbf{v_i}$ a $\mathbf{v_{i-1}}$.

Energia E_G udáva krivosť kontúry v danom vrchole v_i. Teda vychádza z uhlu, ktorý zviera hrana daná vrcholmi $\mathbf{v_{i-1}}$ a $\mathbf{v_i}$ a hrana daná vrcholmi $\mathbf{v_i}$ a $\mathbf{v_{i+1}}$. Tento uhol je tiež rozšírený o vplyv postupnosti značiek, ktoré majú tieto tri vrcholy.

$$E_G(\mathbf{v_{i-1}}, \mathbf{v_i}, \mathbf{v_{i+1}}|\lambda_i) = \beta(\lambda_{i-1}, \lambda_i, \lambda_{i+1}).|\angle(\mathbf{v_{i-1}}, \mathbf{v_i}, \mathbf{v_{i+1}})| \tag{3.4}$$

Kde pomocou koeficientu $\beta(\lambda_{i-1}, \lambda_i, \lambda_{i+1})$ vieme riadiť krivosť v určitých segmentoch kontúry. Napríklad pre postupnosť vrcholov tvoriacich časť aktívnej kontúry, ktoré majú značku $\lambda = hlava$ vieme povoliť ostrejšie hodnoty uhlov. V prípade značky *trup* budeme uprednostňovať veľkosť uhla $\angle(\mathbf{v_{i-1}}, \mathbf{v_i}, \mathbf{v_{i+1}})$ v hodnotách blízkych 180 stupňov. Teda pre uprednostnené prípady bude mať segment aktívnej kontúry, ktorý to spĺňa, nižšiu energiu. Hodnoty koeficientov $\alpha(\lambda_{i-1}, \lambda_i)$ a $\beta(\lambda_{i-1}, \lambda_i, \lambda_{i+1})$ budú popísané podrobnejšie v sekcii 3.1.4.

Energia $E_T(\lambda)$ vyjadruje celkovú postupnosť značiek. Postupnosť môžeme vyjadriť ako:

$$E_T(\lambda) = \{\lambda_i\}_{i=0}^n \tag{3.5}$$

Kde λ sú značky z množiny všetkých značiek Λ. Energia E_T overuje správnosť krivky. Overuje postupnosť segmentov v rámci aktívnej kontúry, kde sú

jednotlivé segmenty tvorené vrcholmi označenými jedným druhom značiek, napríklad *hlava*. Postupnosť týchto segmentov musí odpovedať reálnym pozíciám ľudskej postavy. Zároveň táto energia zabezpečuje, že budú využité všetky značky z množiny Λ. To je ďalší faktor, ktorý bráni uviaznutiu aktívnej kontúry v prípadnom energetickom lokálnom minime. Naviac je tým vytvorený nárok na určitý tvar aktívnej kontúry C.

Postupnou minimalizáciou tejto energie $E(C, \lambda)$ je aktívna kontúra priťahovaná k objektom a ich hraniciam a tvarovaná podľa toho aké značky λ obsahujú jednotlivé jej segmenty. Minimálna energia tejto krivky predstavuje riešenie hľadania kontúry ľudskej postavy v obraze. Minimálna energia aktívnej kontúry je potom vyjadrená ako:

$$E(C^*) = arg \min_{\substack{C \in \Psi \\ \lambda \in \Lambda^n}} (E(C, \lambda)) \tag{3.6}$$

Kde Ψ je množina všetkých polygónov C s n vrcholmi v obrázku, ktoré spĺňajú nároky na postupnosť danú vzorcami 2.1 a 2.2, a zároveň ide o Jordanovské krivky, popísané v podkapitole 2.1. Polygón C^* potom predstavuje riešenie úlohy.

3.1.1 Kontúra a jej chovanie

Energia $E(C, \lambda)$ nie je minimalizovaná nad celým obrázkom naraz, teda nad celou množinou Ψ, ale postupne. Hlavným dôvodom je šetrenie výpočtového času a zníženie počtu krokov, za ktoré model aktívnej kontúry nájde riešenie. Úloha minimalizácie energie teda prebieha vždy v rámci menšieho priestoru ψ_j, ktorý tvorí okolie inicializačnej kontúry C_{I_j}. Prvá kontúra, respektíve polygón, ktorý vzniká v obrázku, je inicializovaný ručne užívateľom, ideálne do priestoru blízkeho hľadanej kontúry človeka, prípadne je inicializovaný automaticky pozdĺž okrajov obrázku. Pre prvý polygón inicializovaný užívateľom platí $j = 0$.

Priestor ψ_j tvorí priestor všetkých možných polôh, do ktorých sa tento polygón môže v ďalšom kroku minimalizácie energie posunúť. Keďže ide o polygón pozostávajúci z postupnosti n vrcholov definovanej rovnicami 2.1 a 2.2, tento priestor vzniká vygenerovaním určitého počtu možných polôh pre každý jednotlivý vrchol tohto polygónu. Pritom platí, že inicializačný polygón tvorí stred tohto priestoru ψ_j. Obecne pre ψ_j platí:

$$\psi_j \subset \Psi \tag{3.7}$$

Teda ψ_j je podmnožina množiny všetkých polygónov Ψ. V rámci tejto podmnožiny hľadáme minimálnu energiu ako:

$$E(C^*_{I_{j+1}}) = arg \min_{\substack{C \in \psi_j \\ \lambda \in \Lambda^n}} (E(C, \lambda)) \tag{3.8}$$

Kde $C_{I_{j+1}}$ je polygón s minimálnou energiou v rámci množiny ψ_j. Tento polygón slúži ako inicializačná kontúra v ďalšom kroku optimalizácie, kde v jeho okolí vzniká podmnožina ψ_{j+1}. Pre túto podmnožinu platí:

$$(\psi_{j+1} \subset \Psi) \wedge (\psi_{j+1} \neq \psi_j) \tag{3.9}$$

Týmto postupným minimalizovaním energie sa aktívna kontúra dynamicky pohybuje v obrázku. Proces pokračuje pokým nie je nájdená minimálna energia obrázku $E(C^*)$. Za túto minimálnu energiu je považovaná energia, ktorá spĺňa podmienku:

$$E(C^*_{I_k}) = E(C^*_{I_{k-1}}) \tag{3.10}$$

Teda ak aktívna kontúra dôjde do stavu, kedy v priestore ψ_{k-1} vytvorenom okolo inicializačného polygónu tohto kroku $C_{I_{k-1}}$ je nájdený nový polygón C_{I_k}, ktorého energia $E(C^*_{I_k})$ je zhodná s energiou $E(C^*_{I_{k-1}})$. Index k potom označuje celkový počet krokov, za ktoré model aktívnej kontúry dospeje k nájdeniu minimálnej energie.

Tento prípad ukončenia nie je jediný. Samostatne riešeným prípadom je vznik oscilácie, teda prípad, kedy by algoritmus iteroval medzi dvoma rovnakými energiami. Tento stav môžeme zapísať postupnosťou nájdených minimálnych energií polygónov:

$$\{E(C^*_{I_0}), E(C^*_{I_1}), ..., E(C^*_{I_{k-2}}), E(C^*_{I_{k-1}}), E(C^*_{I_k}) | E(C^*_{I_k}) = E(C^*_{I_{k-2}})\} \tag{3.11}$$

Prípad, kedy by model došiel do stavu, keď by sa minimálna energia polygónu vypočítaná v kroku k rovnala niektorej minimálnej energii vypočítanej pred krokom $k-2$, nastať nemôže. To je zaistené definíciou samotného polygónu, ktorý tvorí model aktívnej kontúry, a tým, že prehľadávame okolie, ktoré obsahuje predchádzajúci polygón. Teda každá novo-nájdená minimálna energia môže byť len nižšia alebo rovná tej, ktorá daný krok inicializovala.

Evolúcia a pohyb modelu aktívnej kontúry v obraze je vzhľadom na tieto podmienky ukončená v iterácii k nájdením minimálnej energie:

$$E(C^*) = E(C^*_{I_k}) \tag{3.12}$$

Vytvorenie podpriestoru

Priestor polygónov ψ i inicializačný polygón C_ψ, ktorý tvorí stred tohto priestoru, sú tvorené samostatnými vrcholmi. Teda je možné ich reprezentovať ako graf. Vrcholy tohto grafu sú tvorené vrcholmi polygónu C a ψ, do ktorých sa vrcholy polygónu môžu presunúť v ďalšom kroku minimalizácie. Hrany grafu sú tvorené spojením vrcholov, ktoré vzniknú ako alternatívne polohy v rámci jedného vrcholu polygónu s vrcholmi, ktoré vzniknú v rámci vrcholu polygónu, ktorý po ňom nasleduje. Keďže graf vychádza z polygónu, ktorý je daný postupnosťou vrcholov, hrany grafu sú orientované. Aj keď je incializačný

polygón uzatvorený, čo je podmienené rovnicami 2.1 a 2.2, tento graf nie je cyklický. Pri vytváraní priestoru pre polygón, na základe ktorého priestor vzniká, neplatí podmienka daná rovnicou 2.2. Dodržanie nároku na uzatvorenú krivku modelu je zabezpečené spôsobom, ktorým pracujeme pri vyhodnotení ceny minimálnej cesty týmto grafom. O tom píšeme v závere tejto sekcie. Definujeme graf G, pre ktorý platí:

$$G = (V, E) \tag{3.13}$$

Kde V sú vrcholy grafu a E sú hrany grafu spájajúce jednotlivé vrcholy.

Rozmery grafu sú závislé od dĺžky polygónu C, respektíve od počtu vrcholov **v**, ktoré ho tvoria, a od počtu alternatívnych vrcholov, ktoré vznikajú od každého vrcholu polygónu C. Tie vznikajú na obe strany rovnomerne a ich počet je volený proporcionálne k obrázku, prípadne je zadaný ručne užívateľom. Rozmer tohto grafu potom môžeme označiť ako M x N kde M je dĺžka a N je šírka grafu. Pre N zároveň platí, že ide vždy o nepárne číslo, keďže polygón C tvorí stred priestoru ψ. Pre vrcholy a hrany grafu potom platí:

$$V = \{v_{i,j} | i = 1 \dots M, j = 1 \dots N\} \tag{3.14}$$

$$E = E_{UL} \cup E_U \cup E_{UR} \tag{3.15}$$

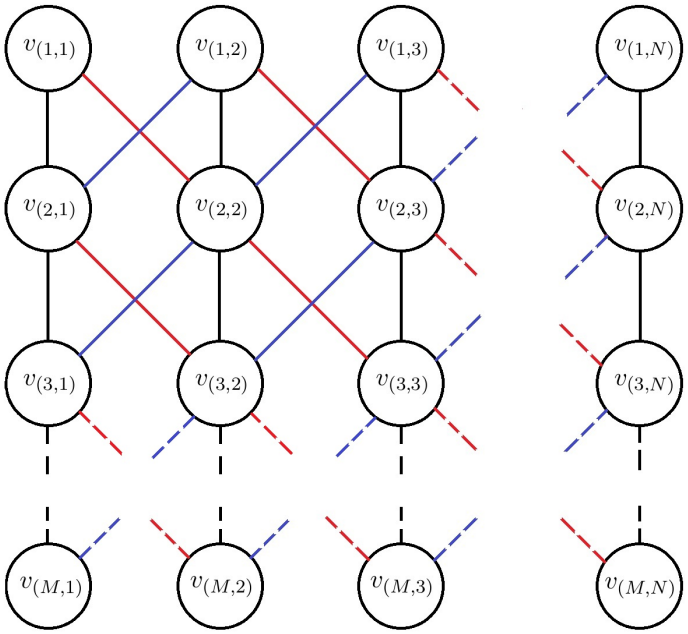
Kde E_{UL}, E_U a E_{UR} popisujú hrany a spájanie jednotlivých vrcholov. Platí pre ne:

$$E_{UL} = \{(v_{(i-1,j-1)}, v_{(i,j)}) | i = 2 \dots M, j = 2 \dots N\} \tag{3.16}$$

$$E_U = \{(v_{(i-1,j)}, v_{(i,j)}) | i = 2 \dots M, j = 1 \dots N\} \tag{3.17}$$

$$E_{UR} = \{(v_{(i-1,j+1)}, v_{(i,j)}) | i = 2 \dots M, j = 1 \dots (N-1)\} \tag{3.18}$$

Takto zadefinovaný graf G môžeme znázorniť graficky ako:



Obrázok 3.1: Ukážka grafu G.

Hrana E_{UL} spája vrchol zo stĺpca j a riadka i s vrcholom predchádzajúceho riadka $i-1$, ktorý je umiestnený v stĺpci $j-1$. V obrázku 3.1 je reprezentovaná červenou farbou. Hrana E_U spája vrcholy rovnakého stĺpca dvoch po sebe idúcich riadkov. Hrana E_{UR} spája vrchol zo stĺpca j a riadka i s vrcholom predchádzajúceho riadka $i-1$, ktorý je umiestnený v stĺpci $j+1$. V obrázku 3.1 je reprezentovaná modrou farbou. Ľubovolná cesta v grafe môže byť potom vyjadrená ako postupnosť vrcholov a dvojíc vrcholov, ktoré vyjadrujú hrany medzi nimi, následovne:

$$R = \{v_{(1,j_1)}, (v_{(1,j_1)}, v_{(2,j_2)}), v_{(2,j_2)}, \ldots, (v_{(M-1,j_{M-1})}, v_{(M,j_M)}), v_{(M,j_M)}\} \tag{3.19}$$

Táto definícia zaručuje, že cesta obsahuje vždy jeden vrchol z riadku na indexe i a zároveň pri prechode medzi stĺpcami označenými indexom j zmení tento index stĺpca maximálne o jedna. Takto definovaná cesta zároveň odpovedá jednému polygónu. Množina všetkých existujúcich ciest v grafe G potom odpovedá množine všetkých existujúcich polygónov ψ.

Pre inicializačný polygón tohto priestoru platí, že je v jeho strede, teda index vrcholov j sa nemení a jeho hodnota sa rovná polovici z celkovej šírky grafu vyjadrenej indexom N. Cestu v grafe, ktorá odpovedá tomuto polygónu, môžeme zapísať ako postupnosť vrcholov a hrán:

$$R_\psi = \{v_{(1,j)}, (v_{(1,j)}, v_{(2,j)}), v_{(2,j)}, \ldots, (v_{(M-1,j)}, v_{(M,j)}), v_{(M,j)} | j = N/2\} \tag{3.20}$$

Susedné vrcholy v rámci jedného riadku predstavujú polohy, do ktorých sa môže vrchol zo stredového indexu posunúť vplyvom minimalizácie energie.

Na základe polohy grafu G v obrázku, ktorá je zhodná s polohou priestoru ψ, vieme jednotlivým vrcholom grafu priradiť ceny. Tieto ceny sú do vrcholov grafu priradené na základe hodnôt počítaných z obrázku I_{edge} a na základe značiek λ detekovaných v polohách jednotlivých vrcholov. Teda energia $E_I(\mathbf{v_i}|\lambda_i)$, popísaná v rovnici 3.2 vychádza z ceny týchto vrcholov. Ceny hrán, ktoré sa nachádzajú medzi dvojicami vrcholov, sú dané vzájomnou vzdialenosťou týchto dvoch vrcholov a značkami, ktoré sú im priradené. Energia $E_E(\mathbf{v_{i-1}}, \mathbf{v_i}|\lambda_i)$, popísaná v 3.3 je vyjadrená týmito vzdialenosťami a závislosťami. Na ďalšej úrovni vieme určiť ceny hrán na základe postupnosti značiek priradených trojici po sebe idúcich vrcholov a uhlu, ktoré tieto hrany zvierajú. Táto závislosť je vyjadrená v rámci energie $E_G(\mathbf{v_{i-1}}, \mathbf{v_i}, \mathbf{v_{i+1}}|\lambda_i)$ popísanej v rovnici 3.4. Teda hodnota minimálnej cesty R^* v grafe G odpovedá hodnote minimálnej energie $E(C^*_\psi)$ priestoru ψ.

$$E(C^*_\psi) = R^* \tag{3.21}$$

Takto definovaná úloha hľadania minimálnej cesty v grafe R^*, a teda energie $E(C^*)$, môže byť efektívne počítaná a optimalizovaná použitím prehľadávacích kombinatorických metód, medzi ktoré patrí aj dynamické programovanie. To garantuje nájdenie globálneho minima v polynomickom čase [1]. Tým je zabezpečená robustnosť algoritmu vzhľadom na inicializačné nároky a možný problém uviaznutia aktívnej kontúry v lokálnom minime. Ďalšou výhodou je charakter energie a ich súm a celková diskrétnosť aktívnej kontúry, teda jej rozdelenie na samostatné vrcholy. V rámci dynamického programovania dokážeme zabezpečiť splnenie podmienky uzatvorenia krivky, o ktorom sme hovorili na začiatku tejto kapitoly.

3.1.2 Rola hrany

Hrana detekovaná v obraze má za úlohu priťahovať model aktívnej kontúry k hranám, ktoré môžu potencionálne predstavovať hľadanú kontúru ľudskej postavy. Hrany v obrázku nie sú reprezentované binárne, ale nadobúdajú rozsah hodnôt z množiny $(0, 1)$. To vďaka aplikácii filtra Gaussiánu. O tom viac v sekcii 3.2.

Dôležitý poznatok je, že týmto filtrom dosiahneme rozmazanie hrany spôsobom, že v obrázku zaberá väčšiu plochu, teda je jednoduchšie rozpoznateľná, respektíve rozpoznateľná z väčšej vzdialenosti od jej výskytu. Zároveň sa vyznačuje tým, že jej hodnoty klesajú postupne smerom od polohy tejto hrany.

Ak sa časť plochy ψ dostane do miesta v obrázku, kde je hodnota hrany vyššia, hodnoty energie E_I sú v týchto miestach nižšie, a teda je pravdepodobnejšie, že nájdený polygón s minimálnou energiou v rámci priestoru ψ bude prechádzať práve touto časťou plochy.

Hrany sú v obrázku reprezentované hranovým obrázkom I_{edge}. Hodnoty tohto obrázku potom predstavujú hodnoty energie E_I v jednotlivých vrcholoch.

3.1.3 Rola detektora

Pridaním informácie o značkách do jednotlivých vrcholov obrázku mu tým pridávame vyššiu informáciu o objektoch, ktoré obsahuje. Teda na základe tejto detektorom pridanej sémantiky vieme na model aktívnej kontúry vytvoriť konkrétnejšie nároky, než je len nutnosť nájdenia čo najsilnejšej hrany v obrázku.

Základný nárok daný značkami je nárok na ich postupnosť v rámci vrcholov, ktorým boli priradené. Definujeme si povolené a zakázané prechody medzi vrcholmi. Ideálny prechod je samozrejme prechod medzi dvoma vrcholmi s rovnakou značkou. Povolený je aj prechod medzi vrcholmi s dvoma odlišnými značkami, no len v prípade, že tieto značky môžu nasledovať v reálnej kontúre. Teda povolené dvojice značiek sú napríklad $(hlava, rameno)$ alebo $(ruka, trup)$. Ako zakázané prechody sú dvojice vrcholov, ktoré majú dvojicu značiek $(hlava, noha)$, teda dvojice, ktoré reálne nemôžu nastať. Tieto povolené a zakázané prechody sú vyjadrené ako ceny dané energiami E_E.

Preferovaním určitej postupnosti vrcholov polygónu dostávame segmenty, teda časti polygónu, ktorých postupnosť vrcholov sa vyznačuje rovnakou značkou λ. Táto značka je z množiny všetkých značiek Λ, teda $\lambda \leftarrow \Lambda$. Takáto postupnosť môže byť vyjadrená napríklad ako:

$$C_{S_\lambda} = \{(\mathbf{v_i}|\lambda)\}_i^n \tag{3.22}$$

V rámci tohto segmentu potom vieme vytvoriť nároky na tvar polygónu. Napríklad, že v segmente polygónu, ktorý pokrýva časť obrázku kde bola detekovaná $hlava$, má mať tento segment C_{S_hlava} vyššiu krivosť. Tieto nároky sú vyjadrené energiou E_G.

Výstup z detektora naviac zabezpečuje to, že sa aktívna kontúra dostane k postave človeka, aj keby bola inicializovaná na okrajoch obrázku a mala napríklad štvorcový tvar, teda tvar, ktorý neodpovedá tvaru kontúry ľudskej postavy. To zohľadňuje energia E_T, ktorá vyhodnocuje, či má polygón v rámci svojich vrcholov obsiahnuté všetky značky λ z množiny Λ, a zároveň, či sú obsiahnuté v odpovedajúcom poradí. Vďaka tomu model neuviazne v lokálnom minime vzdialenom od ľudskej kontúry. Prípad uviaznutia v lokálnom minime v blízkosti hľadaného riešenia zase ošetruje šírka priestoru ψ, ktorá prekrýva relatívne veľkú časť obrázku.

Pomocou týchto parametrov získavame kontrolu nad tvarom a elasticitou jednotlivých častí polygónu, a teda i modelu aktívnej kontúry ako celku. Toto riadenie je sprostredkované koeficientmi α a β, ktoré regulujú vplyv jednotlivých energií E_E a E_G na celkovú energiu modelu.

3.1.4 Vlastné koeficienty

Pomocou vlastných koeficientov α a β vieme obmedziť alebo posilniť vplyv energií E_E a E_G z rovnice 3.3, 3.4, ktoré tvoria základnú rovnicu modelu 3.1. Teda vyjadrujú vhodnosť postupnosti značiek λ v rámci dvojice alebo trojice po sebe idúcich vrcholov $\mathbf{v_i}$.

Tieto koeficienty definujeme ručne, a to na základe ladenia algoritmu. Postupnosti rovnakých značiek je pridelená hodnota 0, keďže je žiadaná. Postupnosti odlišných značiek, ktorá je ale dovolená, priradíme hodnotu 3. Postupnosti značiek, ktorá v reálnom popise ľudskej kontúry neexistuje, priradíme hodnotu 10. Ak by sme neexistujúcej postupnosti priradili hodnotu ∞, energia polygónu by potom len na základe dvoch vrcholov mohla nadobudnúť maximálnu hodnotu. Taký prípad by mohol nastať napríklad v prípade falošne pozitívnej detekcii objektu v obrázku, teda k detekcii objektu na nesprávnom mieste. Pritom polygón, ktorý by prechádzal cez takúto oblasť, by mohol vzhľadom na tvar a postupnosť značiek v jeho ostatných segmentov odpovedať minimálnej energii daného priestoru.

Zároveň chceme polygón, ktorý má čo najviac jednotných segmentov, ktoré nadväzujú v správnej postupnosti. Teda je uprednostnený polygón, ktorý obsahuje menej prechodov medzi rôznymi značkami. Od toho sa odvíja priradenie hodnoty 0 pre prechod medzi vrcholmi s rovnakou značkou a mierne penalizovaný povolený prechod medzi dvoma rôznymi značkami.

Koeficient $\alpha(\lambda_{i-1}, \lambda_i)$ má vplyv na energiu E_E, ktorá je daná vzdialenosťou dvoch po sebe idúcich vrcholov. Veľkosť tohto vplyvu závisí od značiek týchto dvoch vrcholov a môže byť odčítaná z tabuľky 3.1.

$\alpha(\lambda_{i-1}, \lambda_i)$	Hlava	Rameno	Pravá ruka	Ľavá ruka	Trup	Noha	Žiadne
Hlava	0	3	10	10	10	10	3
Rameno	3	0	3	3	10	10	3
Pravá ruka	10	3	0	10	3	10	3
Ľavá ruka	10	3	10	0	3	10	3
Trup	10	10	3	3	0	3	3
Noha	10	10	10	10	3	0	3
Žiadne	3	3	3	3	3	3	0

Tabuľka 3.1: Tabuľka hodnotení postupnosti značiek λ.

Koeficient $\beta(\lambda_{i-1}, \lambda_i, \lambda_{i+1})$ vyjadrujúci správnosť postupnosti troch po sebe idúcich vrcholov $\mathbf{v_i}$ má vplyv na energiu E_G, ktorá ohodnocuje uhol a teda krivosť polygónu v mieste daným týmito troma vrcholmi.

Keďže v obrázku detekujeme sedem rôznych objektov, množina Λ obsahuje sedem značiek λ. Pri tvorení trojíc získavame výpočtom variácii s opakovaním celkovo 343 možných kombinácií postupnosti troch značiek $\{\lambda_{i-1}, \lambda_i, \lambda_{i+1}\}$. Každopádne hodnota odpovedá násobku dvoch prechodov, ktoré sú tvorené

týmito vrcholmi. Potom môžeme koeficient $\beta(\lambda_{i-1}, \lambda_i, \lambda_{i+1})$ vyjadriť pomocou koeficienta α ako:

$$\beta(\lambda_{i-1}, \lambda_i, \lambda_{i+1}) = \sqrt{\alpha(\lambda_{i-1}, \lambda_i)^2 + \alpha(\lambda_i, \lambda_{i+1})^2}; \tag{3.23}$$

Kde jednotlivé koeficienty α vyčítame na základe ich hodnôt λ z tabuľky 3.1. Použitím odmocniny umocnených koeficientov normalizujeme hodnotu β.

Lepší spôsob by bol tieto koeficienty získať učením, no tento proces by bol časovo podstatne náročnejší, keďže by sme museli algoritmu predložiť obsiahlu sadu vyznačených kontúr, na základe ktorej by prebiehalo toto učenie.

3.2 Vstupný obraz a jeho spracovanie

Vstupný obraz v ktorom prebieha detekcia a hľadanie kontúry by mal spĺňať určité základné predpoklady. To z dôvodu zachovania presnosti klasifikácie a extrakcie jednotlivých časti tela v obraze. Základnými predpokladmi sú:

- Nutná vzpriamená póza s vystretými končatinami
- Postava by mal stáť na nohách s nohami mierne od seba
- Ruky v ideálnom prípade upažené vedľa seba tak, aby smerovali mierne k zemi
- Postava by mala tvoriť značnú časť obrazu
- Dostatočná svetelnosť vo fotografii
- V ideálnom prípade jednoduché pozadie, napríklad biela stena
- Kamera by mala snímať telo kolmo spredu približne vo výške ramien postavy, teda asi 150 centimetrov od zeme
- Vzdialenosť kamery od snímanej postavy by mala byť taká, aby postava s upaženými rukami tvorila viac než 50% celej fotografie, čo v praxi znamená pri použití fotoaparátu mobilného telefónu asi tri metre

Splnenie týchto predpokladov znižuje pravdepodobnosť chybných detekcií objektov natrénovaným detektorom a následným nesprávnym priradením detekovaných značiek jednotlivým vrcholom obrázku.

Naviac je vďaka tomu splnený predpoklad pre správne určenie rozmerov ľudského tela v neskoršej aplikácii.

Obrázok 3.2: Príklad správneho obrázku.

Obrázok 3.3: Príklad nesprávneho obrázku.

Model by mal byť schopný nájsť obrys ľudskej postavy aj v prípade, že obrázky boli vyfotografované v rôznom prostredí pri rôznom osvetlení a rôznom ošatení fotografovaného objektu. Vždy však ide o prirodzené prostredie. Táto podmienka síce zvýši obtiažnosť riešenej úlohy, no zároveň zaručuje komplexnosť výsledného riešenia. To je dôležité pre ďalšie použitie aplikácie.

3.2.1 Predspracovanie obrazu

Predspracovaním obrazu rozumieme prácu s obrazom na najnižšej úrovni, teda úrovni hodnôt intenzít jednotlivých pixelov, ktoré tvoria maticu reprezentujúcu celkový obraz [1]. Takýmto spracovaním obrazu sa nezvyšuje množstvo informácií, ktoré obsahuje. Naopak, snažíme sa potlačiť informácie, ktoré

nás v obraze nezaujímajú. Ide o rôzne šumy a rušenia, potlačením ktorých zvýrazníme časti obrazu, ktoré chceme spracovávať v ďalších krokoch.

Vytvorením masky, respektíve hranového obrázku I_{edge}, získavame predstavu o polohe a sile hrán v obraze v rámci jednotlivých pixlov.

Čiernobiely obrázok

Čiernobiely obraz je taká reprezentácia obrázku, kde je každý jeho pixel vyjadrený hodnotou intenzity, ktorú nesie. Teda obrázok je vyjadrený len bielou, čiernou a rôznymi stupňami šedej.

Čiernobiely obrázok získame vypočítaním váženej sumy jednotlivých RGB zložiek farebného obrázku [28], a to pomocou vzťahu:

$$Ibw(x, y) = 0.2989 * R + 0.5870 * G + 0.1140 * B \tag{3.24}$$

Kde Ibw je čiernobiely obrázok a R,G a B predstavujú hodnoty jednotlivých farebných zložiek v pixly danom súradnicami x a y.

Táto operácia znižuje množstvo spracovávaných informácií. V obrázku nás zaujímajú tvary a ich poloha v obraze. Kompletné farebné spektrum je pri našom modeli zbytočné. Touto operáciou naviac zvyšujeme kontrast obrázku, čo má za následok zvýraznenie kontúr v obrázku.

Filtrovanie Gaussiánom

Filtrovanie obrázku znamená úprava obrázku za účelom potlačenia šumu v obraze spôsobeného napríklad slabším svetlom pri fotení [1]. Tým sú vyhladené extrémne hodnoty, ktoré sa kvôli tomuto šumu môžu vyskytovať náhodne v pixloch. Filtrovanie zaistí aj odstránenie ďalších malých degradácií v obraze. Filtrovanie prebieha pomocou konvolúcie masky s obrázkom.

V našom prípade obraz filtrujeme za pomoci Gaussiánu, ktorý sa vyznačuje tým, že zachováva hrany. To za pomoci funkcie:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.25}$$

Kde σ označuje deviáciu Gaussian distribúcie. K tejto funkcii vytvárame jej diskrétnu aproximáciu vytvorením konvolučného jadra, ktoré môže vyzerať následovne:

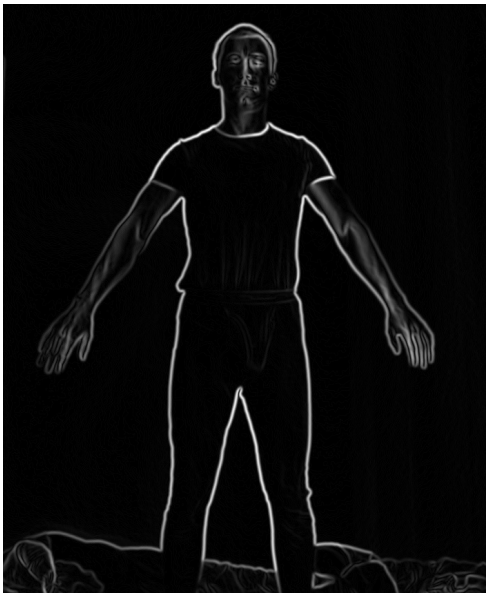
$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Obrázok 3.4: Príklad konvolučnej masky pri $\sigma = 1$.

Konvolúciou tejto masky s obrázkom Ibw dostávame nový filtrovaný obrázok $Ibwg$.

3.2.2 Hranový obraz

Hranový obraz je obraz pozostávajúcich z pixelov, ktoré nadobúdajú hodnoty na základe toho, ako prudko sa na daných súradniciach tohto pixela v pôvodnom obraze mení jas, respektíve intenzita. V praxi ide o prechod medzi pixelom s vysokou hodnotou intenzity do pixela s nízkou hodnotou intenzity alebo naopak.



Obrázok 3.5: Hranový obraz získaný z pôvodného obrazu.

Tieto hrany často označujú miesto, ktoré rozlišuje hranicu medzi objektom v popredí a pozadím.

Hranový obraz, respektíve veľkosť detekovanej hrany v určitom bode obrazu tvorí jednu zo základných premenných cenovej funkcie.

Smerové diferencie

Veľkosti hrany v jednotlivých bodoch odpovedajú veľkosti gradientov v týchto bodoch. Celková veľkosť gradientu v niektorom z bodov je závislá od veľkosti gradientov v smere osi x a v smere osi y vypočítaných pre ten konkrétny bod. Tieto veľkosti získame pomocou konvolúcie upraveného obrázka so Sobelovým operátorom[1].

Konvolúcia s operátorom v smere osi x:

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \star I$$

Konvolúcia s operátorom v smere osi y:

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \star I$$

Veľkosť gradientu

Zo smerových derivácii následne vieme určiť celkovú veľkosť gradientu, ktorý nám udáva ako silná je hrana. Teda ako prudko sa menia hodnoty intenzity.

$$G_{mag} = \sqrt{G_x^2 + G_y^2} \tag{3.26}$$

Výsledkom je hranový obrázok 3.5 , ktorý má vyššie hodnoty v miestach, kde je hrana v obraze silnejšia. Keďže v závere hľadáme minimálnu cenu a aktívna kontúra by mala byť priťahovaná k hranám a nie naopak, invertujeme hodnoty pixelov v obrázku. Teda hrany sú reprezentované číslami bližšími k nule.

Touto úpravou obrázku $Ibwg$ dostávame obrázok I_{edge}, ktorého hodnoty tvoria internú energiu E_I obrázku.

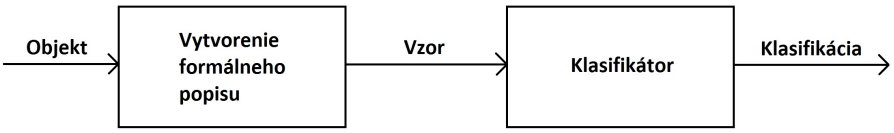
Dilatácia obrázku

Dilatácia je takzvaná izotropická expanzia [1], čo znamená, že rastie všetkými smermi rovnako. Morfologicky transformujeme hranový obrázok z dôvodu, aby sme odstránili možné výskyty oblastí tvorených niekoľko málo pixelmi, ktoré majú opačné hodnoty než ich okolie. Cieľom tejto operácie je zosilnenie hrán a ich vyhladenie.

3.3 Detekcia objektov v obrázku

Objekt v obraze, ktorý chceme detekovať, v našom prípade napríklad časť tela ako *hlava*, je v obraze popísaný určitou plochou a v nej obsiahnutými pixlami. Objekt je teda možné popísať určitou sadou unikátnych vlastností,

štruktúrou alebo vzorom. Súhrn takýchto vlastností, tvoriacich formálny popis, je definovaný ako trieda. V obraze vieme rozlišovať väčší počet objektov a teda aj unikátnych popisov, na základe ktorých vzniká viacero tried. Rozpoznávanie objektov je potom založené na priradení týchto tried jednotlivým objektom. O toto priradenie sa stará klasifikátor. Tento klasifikátor tvorí základ detektora objektov.



Obrázok 3.6: Základné kroky klasifikácie.

Aby klasifikátor dokázal správne priradiť triedy objektom, je nutné ho najprv naučiť vlastnosti jednotlivých tried. Učenie pozostáva z predloženia sady obrázkov objektov spolu s k nim priradeným značkám, respektíve pomenovaním triedy, ku ktorým jednotlivé obrázky prislúchajú. Množina všetkých týchto dvojíc $(objekt, značka)$ nazývame trénovací set alebo trénovacia množina [1] [2]. Od veľkosti a kvality trénovanej množiny závisí kvalita klasifikátora, teda schopnosť správne klasifikovať objekty. V závislosti od zvoleného algoritmu a aktuálneho využitia by takáto trénovaná množina mala obsahovať v rádoch stovák až tisícov obrázkov.

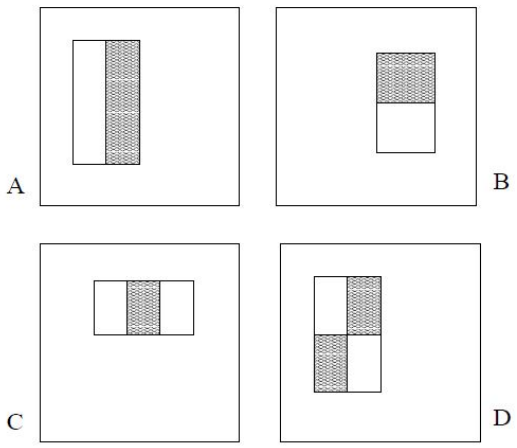
Cieľom učenia je minimalizovať optimalizačné kritérium, ktoré môže byť reprezentované priemernou chybovosťou klasifikácie. Učenie by malo byť zároveň induktívne, čo znamená, že naučené vlastnosti by mali byť generalizovateľné naprieč celou trénovanou množinou.

Samotná detekcia objektov v obraze môže prebiehať napríklad pomocou takzvaného *sliding window*, čo je metóda, kedy nad obrázkom iteruje menšie pod-okno a každý výsek obrázku určený týmto pod-oknom je testovaný klasifikátorom. Pod-okno rozmeru m x n je tvorené postupne nad každým pixlom. Obvyklý postup je od ľavého horného rohu po riadkoch až do pravého spodného rohu. Vynechané sa okraje obrázku, keďže by tak pod-okno zaberalo len časť pôvodného obrázku. Šírka tohto okraju závisí od veľkosti *sliding window*. Výsledok je postupne zapisovaný do samostatného poľa, respektíve masky obrázku, podľa aktuálnej polohy *sliding window*. Po kompletnom preskúmaní obrázku dostaneme pole hodnôt, ktoré obsahuje pravdepodobnosti, respektíve konkrétne miesta, kde bol výsledok klasifikácie polohy *sliding window* pre určitú triedu pozitívny. Rozmer okna *sliding window* by mal byť ideálne v rozmere hľadaného objektu. V prípade obecnejšieho prístupu prebieha okno skúmaným obrázkom viac krát, no vždy pri inom rozlíšením skúmaného obrázku. To zaistí detekciu výskytu objektu, tak sa tam ten vyskytuje v rôznych mierkach, respektíve relatívnych rozmeroch vzhľadom na perspektívu obrázku.

3.3.1 Detektor Viola-Jones

Princíp detektora objektov Viola-Jones [25] [26] je postavený na využití kaskády jednoduchých klasifikátorov, ktoré formujú silný klasifikátor. V literatúre sa tieto jednoduché klasifikátory označujú aj ako *weak learners* [25] alebo slabý žiaci. Ako každý klasifikátor, aj tieto jednoduché je nutné najprv natrénovať. Pri ich trénovaní poskytujeme algoritmu sadu pozitívnych a sadu negatívnych vzoriek. Pozitívne vzorky sú obrázky objektu, prípadne obrázky, na ktorých je vyznačený objekt, ktorý chceme detekovať. Negatívne vzorky sú obrázky, v ktorých sa objekt danej triedy nevyskytuje. Pre dosiahnutie dostatočne vysokej miery presnosti klasifikácie by mal byť počet týchto pozitívnych a negatívnych obrázkov čo najvyšší, ideálne v ráde stovák až tisícov. Počet negatívnych obrázkov by potom mal byť aspoň dvojnásobne vyšší, než počet pozitívnych.

Algoritmus naviac pracuje s množinou príznakov, ktoré využíva pri učení a následne pri klasifikácii a detekcií objektov. Využitie príznakov dáva niekoľko výhod. Napríklad práca s príznakmi je rýchlejšia než práca s jednotlivými pixlami vzhľadom na objem spracovaných dát. Teda obrázky sú reprezentované za pomoci vypočítaných hodnôt týchto príznakov. Tieto príznaky sú založené na princípe *Haar* príznakov [26].



Obrázok 3.7: Príklad základných typov príznakov.

Hodnoty jednotlivých príznakov dostaneme odčítaním sumy hodnôt pixlov, ktoré ležia v bielom obdĺžniku od sumy hodnôt pixlov, ktoré ležia v šedom obdĺžniku. Môžeme pracovať aj s väčším počtom plôch. Na obrázku 3.8 máme v prípade A a B plochy tvorené dvoma obdĺžnikmi. V prípade C ide o tri obdĺžnikové plochy a v prípade D o štyri plochy. Takýchto príznakov má detektor k dispozícii v ráde desaťtisícov až sto-tisícov. Ich počet závisí od rozmerov negatívnych a pozitívnych obrázkov. Vznikajú kombinovaním ich rôznych rozmerov a rôznych polôh alebo natočení.

Pre efektívne sčítanie a odčítanie týchto plôch, a teda vyjadreniu hodnôt

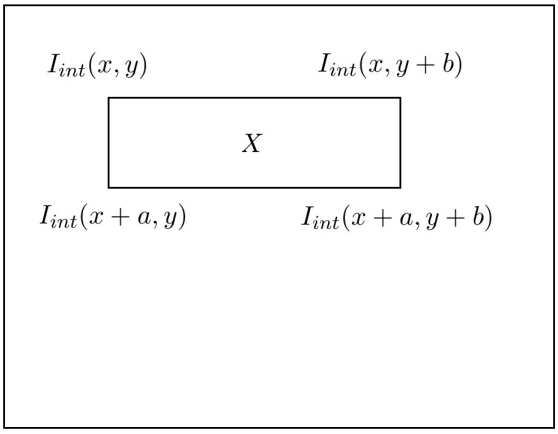
príznakov pre jednotlivé obrázky, sa využíva takzvaný *integrálny obraz*. Ide o reprezentáciu, ktorá je jednorázovo vyjadrená pre každý trénovaný obrázok. Hodnoty v pixloch integrálneho obrazu I_{int} vznikajú tak, že sú hodnoty každého pixlu tohto obrázku vyjadrené sumou hodnôt všetkých pixlov, ktoré ležia nad a vľavo od polohy daného pixlu. Teda hodnotu jedného pixlu integrálneho obrázku $I_{int}(x, y)$ vieme vypočítať ako:

$$I_{int}(x, y) = \sum_{i=1}^{x} \sum_{j=1}^{y} I_T(i, j) \tag{3.27}$$

Kde I_T je obrázok z trénovanej množiny. Vypočítaním hodnôt v celom tomto obrázku dostaneme integrálny obraz I_{int}. Získanie hodnoty plôch daných tvarom niektorého z príznakov sa týmto výrazne zjednoduší. Konkrétne na dve operácie sčítania a odčítania, ktoré sú zapísané:

$$X = I_{int}(x, y) + I_{int}(x + a, y + b) - (I_{int}(x + a, y) + I_{int}(x, y + b)) \tag{3.28}$$

Kde X predstavuje plochu obdĺžnika, ktorého hodnotu chceme vypočítať. Body $I_{int}(x, y)$,$I_{int}(x + a, y + b)$,($I_{int}(x + a, y)$ a $I_{int}(x, y + b)$ reprezentujú jeho vrcholy. Koeficienty a a b reprezentujú dĺžku hrán obdĺžnika. Graficky to môžme znázorniť ako obrázok:



Obrázok 3.8: Znázornenie výpočtu hodnoty X v integrálnom obrázku.

Po prvom vypočítaní hodnôt integrálnych obrazov sú hodnoty jednotlivých príznakov počítané vždy za konštantný čas, keďže ide pri výpočte plochy vždy o výpočet na základe štyroch hodnôt.

Aby bol proces učenia a klasifikácie rýchly, musíme vylúčiť veľké množstvo týchto príznakov a zaoberať sa len dôležitými. Pre výber príznakov, ktoré sa využijú pre tvorbu jednoduchých klasifikátorov a ich samotné učenie sa využíva takzvaný *boosting* [27]. Ten je využitý v klasifikačnom algoritme *AdaBoost* [26]. Výber príznakov pomocou tejto metódy prebieha postupným výberom jednotlivých príznakov a testovaním ich schopnosti správne klasifikovať vzorky. Teda ich schopnosťou klasifikovať objekt s lepším výsledkom,

než by bolo náhodné hádanie. Napríklad keď klasifikátor zo 100 objektov, aspoň v 51 prípadoch popíše objekt správne. Testujú sa všetky príznaky postupne na všetkých pozitívnych i negatívnych testovacích obrázkoch. Výber klasifikátorov prebieha pomocou postupnej úpravy váh u každého z nich. Tieto váhy sú na začiatku rozdelené rovnomerne. Úprava váhy je ovplyvnená schopnosťou klasifikátora rozlišovať pozitívne od negatívnych vzorkov. Teda pri každom klasifikátore je vyjadrený počet obrázkov, ktoré klasifikoval zle, čím je vyjadrená jeho chybovosť. Toto trénovanie prebieha v iteráciách, kde v rámci jednej iterácie je vždy vybraný klasifikátor s najmenšou chybovosťou, ktorý je aj so svojou váhou pridaný do lineárnej kombinácie, ktorá tvorí silný klasifikátor. Ten môžeme vyjadriť ako:

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + ... + \alpha_n f_n(x) \tag{3.29}$$

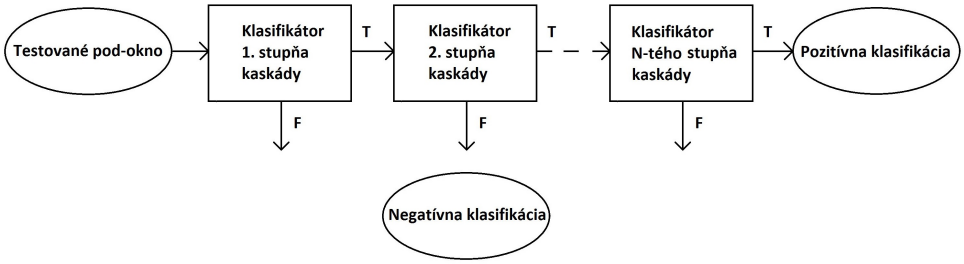
Kde $F(x)$ je silný klasifikátor a f_i sú slabé klasifikátory. Symboly α_i vyjadrujú váhy jednotlivých slabých klasifikátorov, ktoré sú im priradené počas trénovania. V rámci jednej iterácie trénovania dochádza vždy k úprave váh tak, že sa horším klasifikátor váha zvyšuje a lepším naopak znižuje. Iterácie pokračujú pokiaľ nedostaneme zvolený počet slabých klasifikátorov do lineárnej kombinácie, alebo ak vzhľadom na nedostatok trénovacích dát algoritmus nevie ďalej relevantne rozoznávať kvalitne hodnotiace príznaky. Algoritmus je podrobne popísaný v článku [26] v tabuľke *Table* 1.

V praxi žiadny jednoduchý klasifikátor nevie klasifikovať s výrazne nízkou chybovosťou. Klasifikátory volené v prvých kolách *boosting* procesu majú chybovosť klasifikácie rozsahu 10% až 30%. V neskorších kolách, keď je úloha klasifikácie náročnejšia, ide približne o 40% mieru chybovosti.

Vzhľadom na počet vstupných dát môžeme dostať silný klasifikátor, ktorý je tvorený lineárnou kombináciou niekoľkých stovák slabších klasifikátorov. Detekcia objektov v obraze pomocou tohto klasifikátora a pomocou *sliding window*, ktorým prechádzame skúmaný obraz, musí v skúmanom obrázku o veľkosti 300x300 pixlov klasifikovať okno *sliding window* o veľkosti 25x25 pixelov minimálne 76176 krát, čo predstavuje dlhú dobu výpočtu. Pri tomto výpočte vychádzame z predpokladu, že detekcia prebieha len v miestach, kde *sliding window* zaberá len plochu obrázku a jeho plocha nepresahuje za okraj skúmaného obrázku. Teda medzi stredovým pixlom tohto okna a okrajom skúmaného obrázku musí byť aspoň 12 pixlov, v ktorých detekcia neprebehne.

Namiesto využitia jedného silného klasifikátora sú slabé klasifikátory rozdelené po skupinách do kaskád. Potom pri klasifikácii niektorej polohy *sliding window* v rámci každej kaskády sú testované postupne všetky jej slabé klasifikátory. V prípade, že výsledok všetkých klasifikátorov je pozitívna klasifikácia, klasifikovanie pokračuje aj v rámci ďalšej kaskády. Pokiaľ všetky kaskády klasifikujú dané miesto pozitívne, výsledná klasifikácia je taktiež pozitívna. V prípade, že aspoň jedna kaskáda klasifikuje dané miesto ako negatívne, klasifikácia nepokračuje do ďalších kaskád, ale posunie testovacie okno v skúmanom obrázku na ďalšie miesto.

Takáto kombinácia klasifikátorov do kaskády, umožňuje rýchlo preskočiť miesta označujúce pozadie obrázku a umožní stráviť viac času na miestach, ktoré vyzerajú sľubne pre výskyt hľadaného objektu. Princíp klasifikátorov usporiadaných v kaskáde môžeme zakresliť:



Obrázok 3.9: Štruktúra kaskád klasifikátora.

Rýchlosť detekcie objektov potom klesá úmerne s počtom natrénovaných kaskád. Veľký počet kaskád avšak nezaručuje istotu detekcie, naopak rastie pravdepodobnosť, že detektor nedetekuje objekt v mieste obrázku, kde sa reálne nachádza. Takáto chyba sa označuje ako falošne negatívna detekcia. Menší počet kaskád na druhej strane môže detekovať objekt aj v mieste, kde sa nenachádza. V tomto prípade ide o falošne pozitívnu detekciu. Mieru jednotlivých detekcií vieme určiť na úrovni kaskády.

Aby klasifikácia fungovala správne, každá kaskáda musí dosahovať nízku úroveň falošne negatívnych klasifikácií. Pokiaľ daná kaskáda nesprávne klasifikuje objekt ako negatívny, klasifikácia sa zastaví a nie je možné opraviť chybu. Je lepšie, ak má každá kaskáda mierne vyššiu mieru falošne pozitívnych klasifikácií. Tieto chybné detekcie je možné opraviť v kaskádach ďalších úrovniach.

Teda pri voľbe počtu kaskád musíme pracovať s určitým kompromisom medzi menším počtom kaskád s nižšou mierou falošne pozitívnych detekcií alebo vyšším počtom kaskád s vyššou mierou falošne pozitívnych detekcií na kaskádu. Kaskády s nižšou mierou falošne pozitívnych detekcií sú komplexnejšie, pretože kaskáda obsahuje vyšší počet slabých klasifikátorov. Naopak kaskády s vyššou mierou falošne pozitívnych detekcií obsahujú menej slabých klasifikátorov. Obecne je lepšie mať vyšší počet jednoduchých kaskád, keďže chybovosť vyjadrená mierou falošne pozitívnyh detekcií potom klesá exponenciálne s počtom týchto kaskád.

Čím viac kaskád chceme vytvoriť, tým viac dát pre trénovanie potrebujeme poskytnúť algoritmu na vstupe. Avšak počet pozitívnych vzoriek použiteľných pre trénovanie každej kaskády, a tým aj počet kaskád, vieme ovplyvniť pomocou povolenej miery skutočne pozitívnych detekcií. Táto miera špecifikuje koľko percent pozitívnych vzoriek môže klasifikátor klasifikovať chybne ako negatívne vzorky. Ak je pozitívna vzorka raz označená ako negatívna, jej hodnotenie sa už nezmení v žiadnej ďalšej kaskáde. Napríklad keď vyžadujeme mieru skutočne pozitívnych detekcií v miere 90% a všetky vzorky sú použité

pre trénovanie prvej kaskády. V tomto prípade môže byť až 10% vzoriek odmietnutých a označených ako negatívne a ostatných 90% je možné využiť pre trénovanie v ďalšej kaskáde. Pri dostatku trénovacích dát sa týmto spôsobom do trénovania každej ďalšej kaskády postupne dostane stále menej a menej vzoriek. Klasifikátory každej ďalšej kaskády musia riešiť stále náročnejší a náročnejší klasifikačný problém a kritérium s tým, že má k dispozícii menej kladných trénovacích vzoriek

.

Ideálne je použiť rovnaký počet vzoriek pre trénovanie každej kaskády. Počet pozitívnych vzoriek pre použitie pre každú kaskádu vieme určiť ako:

$$N_K = \frac{N_T}{1 + (K - 1) * (1 - TP)} \tag{3.30}$$

Kde N_K vyjadruje počet pozitívnych vzoriek použitých na kaskádu, N_T vyjadruje celkový počet pozitívnych vzoriek, K je počet kaskád a TP vyjadruje mieru skutočne pozitívnych detekcií na kaskádu.

Táto rovnica nám nezaručuje, že je pre každú kaskádu garantovaný rovnaký počet pozitívnych vzoriek. To preto, že je nemožné s určitosťou predpovedať koľko pozitívnych vzoriek bude odmietnutých, respektíve označených ako negatívne. Trénovanie pokračuje pokiaľ počet pozitívnych vzoriek N_K dostupných pre trénovanie kaskády je vyšší než 10% z všetkých vzoriek N_T.

Pokiaľ tu už nie je viac dostupných pozitívnych vzoriek, trénovanie sa zastaví. Výsledkom je klasifikátor s počtom kaskád, ktorý sa podarilo natrénovať až pokiaľ nedošlo k prerušeniu z dôvodu nedostatku dát.

3.3.2 Súbor pozitívnych a negatívnych obrázkov

Pozitívny obrázok je obrázok objektu, ktorého detekciu chceme neskôr vykonávať v obrázku neznámeho prostredia. Pozitívny obrázok by mal obsahovať ideálne len objekt, ktorý chceme detekovať. Objekt by mal byť umiestnený v rámci minima jeho pozadia, ktorého prítomnosť algoritmu pomôže určiť obecný tvar hľadaného objektu. Negatívny obrázok naopak obsahuje len pozadie obrázku. Nesmie sa v ňom vyskytovať hľadaný objekt v žiadnych rozmeroch. V opačnom prípade by dochádzalo k vytvoreniu klasifikátora s vysokou mierou falošne pozitívnych detekcií.

V praxi je vhodné využiť čo najväčší počet týchto pozitívnych a negatívnych obrázkov. Ide o počty v ráde vyšších stovák až tisícov obrázkov. To z dôvodu, aby natrénovaný klasifikátor dokázal správne klasifikovať objekt pri rôznych typoch jeho nasvietenia, respektíve pri rôznych hodnotách svetelnosti objektu a jeho pozadia. Pri rôznom osvetlení dochádza k zmene pomeru svetlosti farieb medzi samotným objektom a jeho pozadím. Teda v prípade svetelnej fotografie je objekt väčšinou tmavší než jeho pozadie. V prípade tmavej fotografie, ktorá bola vytvorená naviac za použitia blesku, je pozadie tmavšie a dominantný objekt v obraze naopak, svetlejší.

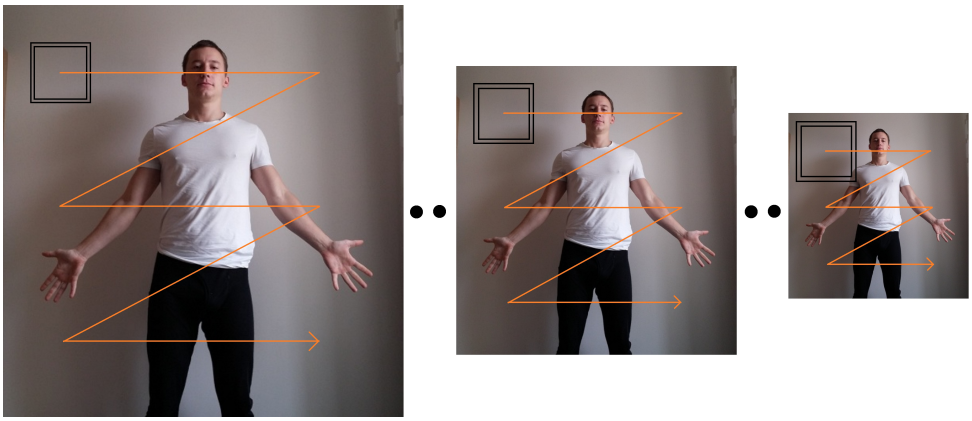
Pre našu aplikáciu predpokladáme, mimo nárokov na postavu definovanú v kapitole 3.2, že obrázky, v ktorých prebieha detekcia, budú z prostredia bytu alebo obecne izby. Preto pozitívne obrázky vznikali pri bežnom izbovom osvetlení, teda kde nedochádza k silnému slnečnému osvetleniu objektu, prípadne je ten osvetlený len žiarovkovým svetlom. Voľba obrázkov pre negatívne vzorky je potom taktiež z prostredia bytových i nebytových objektov.

Pozitívne i negatívne obrázky je možné získať z rôznych dostupných databáz, ako napríklad tie spomenuté v [35] [36] [37] [38], no pre naše využitie sme našli jedine súbory obrázkov označujúcich objekt *hlava*. Sadu pozitívnych obrázkov pre ďalšie objekty z množiny Λ, definovanej rovnicou 2.5, sme si vytvárali sami. Tak isto sadu negatívnych obrázkov.

3.3.3 Výstup z detektora

Detekcia objektov v obrázku prebieha pomocou menšieho okna *sliding window*, ktoré v iteráciách prechádza skúmaným obrázkom. V každej polohe tohto okna dochádza ku klasifikácii. V prípade, že je pozitívna, dôjde k detekcii objektu, ktorý daný klasifikátor rozpoznáva. Veľkosť okna *sliding window* by mala odpovedať veľkosti trénovacieho obrázku. V našom prípade veľkosti značky v pozitívnom obrázku, ktorou vyznačujeme hľadaný objekt.

Aby bolo možné detekovať objekty pri rôznych veľkostiach, detektor postupne zmenšuje obrázok, v ktorom hľadáme objekt. Teda mení pomer veľkosti tohto obrázku vzhľadom k veľkosti *sliding window*. Princíp iterácie tohto okna v obrázku a samotného zmenšovania obrázku môže byť znázornený obrázkom:



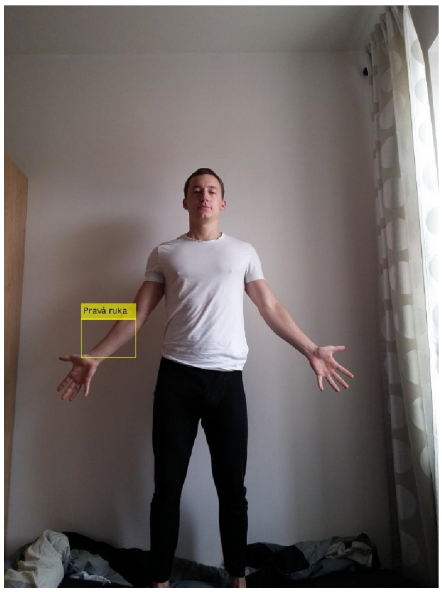
Obrázok 3.10: Postup pri detekovaní pomocou *sliding window*.

Počet krokov a mieru zmenšenia skúmaného obrázku v rámci jednej iterácie tohto menenia pomeru vieme určiť. Určujeme to na základe predpokladu, že ľudská postava vypĺňa určitú plochu obrazu, teda jednotlivé jej časti sú dostatočne veľké. Na základe toho obmedzujeme mieru, do akého zmenšovania algoritmus bude pokračovať. Teda zabraňujeme vzniku negatívne pozitívnych kasifikácií, ku ktorým dôjde v prípade, že je pomer okien taký, že okno

sliding widow vypĺňa väčšiu časť skúmaného obrázku. Ako je napríklad v treťom obrázku série vyobrazenej v obrázku 3.10. Teda zabraňujeme vzniku detekcií v rozmeroch, o ktorých vieme dopredu, že neexistujú.

Výstup z jedného detektora pozostáva z množiny súradníc a rozmerov detekovaných polôh objektov, ktoré sú dané polohou a veľkosťou *sliding widow* vzhľadom ku skúmanému obrázku v momente, kedy k niektorej z týchto detekcií došlo. Na základe týchto informácií vieme každú detekciu reprezentovať v obraze ako štvorhran umiestnený v polohe, v rámci ktorej došlo k tejto detekcii. Príklad môžeme vidieť na obrázku 4.3. Na základe tohto výstupu vytvárame masku o rozmeroch pôvodného obrázku, kde jej vrcholy, ktoré sa nachádzajú v mieste vyznačenom plochou detekcie, potom nadobúdajú hodnoty λ danej značky objektu. Všetky ostatné vrcholy v danej množine majú hodnotu $\lambda = \textit{žiadne}$.

Pre každý detektor každého objektu teda získame samostatné masky. Na základe ich hodnôt vieme pre niektorý z vrcholov povedať či sa v ňom vyskytuje jedna, prípadne viacero značiek. Ak tieto masky spojíme do jednej, potom môže polohy detekcií jednotlivých objektov, ktoré obsahuje znázorniť ako v obrázku 3.11, kde žltý box predstavuje detekciu pravej ruky.



Obrázok 3.11: Detekcia pravej ruky pomocou Viola-Jones.

3.4 Vytvorenie vyhľadávacieho priestoru

Hľadanie kontúry v obraze je redukované na hľadanie kontúry vo vymedzenom prostredí, ktoré nazývame vyhľadávací priestor. Tento priestor je dynamický a vzniká vždy v rámci jednej iterácie v okolí aktuálnej polohy aktívnej kontúry. Vyhľadávací priestor má konkrétne rozmery a jeho tvar závisí od aktuálneho tvaru polygónu aktívnej kontúry. Priestor je reprezentovaný ako

graf a jeho podrobnú definíciu sme zaviedli v podkapitole 3.1.1. V tejto kapitole prevedieme tento formálny popis na jeho konkrétnu reprezentáciu v obraze.

Pri vytváraní reprezentácie tohto priestoru ψ v obraze musíme brať v úvahu pravidlá, ktoré sa vzťahujú na model aktívnej kontúry, teda zabrániť prípadu, kedy by algoritmom vyhodnotená minimálna energia odpovedala polygónu C, ktorý tieto požiadavky nespĺňa. Podmienka, ktorú reprezentácia priestoru musí zabezpečiť je, aby nový v ňom obsiahnutý polygón reprezentujúci aktívnu kontúru nepretínal sám seba.

Graf z obrázku 3.1 vieme v obraze reprezentovať ako maticu vrcholov. Vieme, že v jej stredovom stĺpci sa nachádza inicializačný polygón, ktorý je v grafe vyjadrený cestou grafom R_ψ definovanou rovnicou 3.20. Stredový stĺpec matice je potom vyjadrený indexom j. Okolité vrcholy vznikajú ako alternatívne polohy každého vrcholu $\mathbf{v_i}$ inicializačného polygónu. Vyjadrujú miesta, kam by sa v rámci priestoru pri minimalizácii energie mohol daný vrchol, z ktorého vzišli, presunúť. Tieto vrcholy môžeme nazývať senzormi, keďže zisťujú popis obrázku v danom mieste, kedy tento popis slúži práve na výpočet minimálnej energie celého priestoru. Vrchol polygónu $\mathbf{v_i}$ spolu s jeho senzormi tvorí jeden riadok matice všetkých vrcholov, ktorá reprezentuje graf priestoru ψ.

Prvé vytvorenie vyhľadávacieho priestoru vzniká po inicializovaní počiatočnej aktívnej kontúry užívateľom. Poloha každého ďalšieho vyhľadávacieho priestoru ktorý vznikne je závislá od polohy minimálneho polygónu, ktorý bol vypočítaný z predchádzajúceho kroku. Toto postupné vytváranie a minimalizovanie pokračuje až kým nie je nájdená celková minimálna energia, prípadne nie je splnená niektorá ďalšia podmienka spomínaná v sekcii 3.1.1.

Vyhľadávací priestor vytvorený z inicializačnej kontúry môže vyzerať ako na obrázku 3.12. Detail tohto priestoru je vyobrazený na obrázku 3.13.

Obrázok 3.12: Príklad vytvoreného vyhľadávacieho priestoru

3.4.1 Inicializačná kontúra

Prvá inicializácia kontúry je zo strany užívateľa. Užívateľ zadáva vstup jedným ťahom kurzora priamo do obrázku. Tvar inicializačnej kontúry nemusí priamo sledovať existujúcu kontúru ani odpovedať jej tvaru. Čím je ale tvar podobnejší, tým je počet cyklov iterácií algoritmu menší. Pri inicializácii tejto kontúry musí užívateľ brať na vedomie, že inicializačná kontúra nesmie pretínať samu seba.

Takto inicializovaná krivka pozostáva zo samostatných vrcholov, ktoré sú v obrázku spájané jednotlivými čiarovými segmentmi. Počet vrcholov je závislý od dĺžky krivky, ktorú užívateľ inicializoval. Rozmiestnenie bodov je hustejšie v miestach, kde má krivka vyššiu krivosť. Body sú distribuované redšie v miestach, kde je priamka rovná.

Geometrické vzdialenosti medzi jednotlivými vrcholmi vrátane vzdialenosti medzi prvým a posledným Počet vrcholov inicializačného polygónu definuje dĺžku priestoru, ktorý vzniká v jeho okolí.

3.4.2 Rovnomerná distribúcia bodov

Problém, ktorý vzniká pri zadávaní vstupu rukou od užívateľa je, že môžu vzniknúť miesta, kde sú vrcholy polygónu husto distribuované, aj keď ide o rovný úsek kontúry. To z dôvodu, že vstup od užívateľa vzniká pohybom kurzora po obrázku, výsledkom čoho môže byť v niektorých miestach roztrasená krivka.

Inicializačnú kontúru preto pred vstupom do algoritmu upravíme tak, že jej body rovnomerne distribuujeme po celej kontúre. Tým sa krivka vyhladí a odstránia sa krátke úseky kontúry, ktoré sa vyznačujú takýmito zhlukmi vrcholov. Tento proces prebehne vždy len raz po zadaní kontúry užívateľom.

3.4.3 Generovanie senzorov a vyhľadávacieho priestoru

Po inicializovaní počiatočnej kontúry a jej úprave vzniká v každom vrchole polygónu, ktorý ju reprezentuje, sada ďalších vrcholov, ktoré slúžia ako jej senzory. Spolu s vrcholom, z ktorého vznikajú, tvoria senzorovú líniu. Počet vrcholov, ktoré takto vzniknú v rámci senzorovej línie, vrátane jej geometrickej dĺžky v obraze, definujeme výpočtom prípadne tieto hodnoty môže definovať užívateľ.

Výpočet odvodí hodnotu geometrickej dĺžky vzhľadom k rozmerom obrázku, v ktorom prebieha hľadanie kontúry. V tom je zohľadnený predpoklad o veľkosti postavy a hľadaného detailu v obraze z kapitoly 3.2. Dĺžka senzorovej línie $l_s =$ je potom určená ako:

$$l_s = \frac{\sqrt{h_I^2 + w_I^2}}{20} \tag{3.31}$$

kde h_I je šírka a w_I je výška skúmaného obrazu v pixloch. Vrcholy sú potom umiestnené rovnomerne každé 4 pixly od stredového vrcholu. Ich počet musí byť vždy nepárny. Voľba vzdialenosti medzi jednotlivými pixlami je v základnom nastavení relatívne malá z dôvodu, aby algoritmus dokázal zachytiť väčší detail, respektíve zmenu v obrázku. Toto nastavenie v praxi pre obrázok rozmerov 1000x1000 pixlov znamená senzorové línie s dĺžkou 50px a 11 bodmi.

Dĺžkou línie definujeme šírku vyhľadávacieho priestoru v rámci jeho grafickej reprezentácie v obrázku. Počtom vrcholov definujeme šírku priestoru v rámci jeho reprezentácie pomocou grafu. Voľbou dĺžky línie a počtu bodov, ktoré obsahuje, vieme ovplyvniť minimálnu a maximálnu vzdialenosť v obrázku, o ktorú sa môže model aktívnej kontúry v obrázku posunúť v priebehu jednej iterácie.

V rámci reprezentácie tohto priestoru v obrázku je senzorová línia orientovaná kolmo na inicializačný polygón. Z jej definície tejto línie vyplýva, že jej stredový vrchol patrí polygónu, teda tento vrchol je ich spoločný priesečník. Vrcholy polygónu, z charakteru hľadaného riešenia, nebudú nikdy usporiadané tak, aby spolu tvorili priamku. Teda orientáciu tejto senzorovej línie musíme počítať tak, aby smerovala do vnútra oblasti, ktorú vymedzuje polygón. Orientáciu senzorovej línie vytvorenej vo vrchole v_i počítame pomocou určenia smernice priamky k, ktorá je daná vrcholmi v_i a stredovým vrcholom polygónu danej vrcholmi v_{i-1} a v_{i+1}. Začneme vyjadrením stredového vrcholu ako:

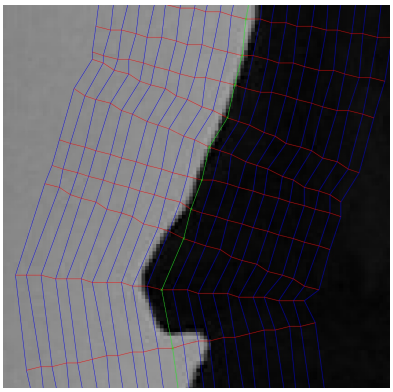
$$v_{s_i} = \frac{v_{i-1} + v_{i+1}}{2} \tag{3.32}$$

Kde pre priamku prechádzajúcu týmto vrcholom v_{s_i} a vrcholom v_i pre ktorý riešime túto úlohu natočenia, volíme smerový vektor $\mathbf{s} = v_{s_i} - v_i = (a_s, b_s)$. Zo smerového vektora vyjadríme normálový vektor $\mathbf{n} = (a_n, b_n)$ kde $a_n = -b_s$ a $b_n = a_s$. Potom pre hľadanú smernicu natočenia:

$$k_{v_i} = -(\frac{a_n}{b_n}) \tag{3.33}$$

To platí pre prípad, že táto priamka nie je rovnobežná s osou Y obrázku. Prípad rovnobežnosti nastane ak sa hodnota $b_n = 0$. V tom prípade je smer daný priesečníkom s osou X.

Takto definovaný vyhľadávací priestor a orientáciu jednotlivých senzorových línií môžeme v detaile obrázku vyobraziť následovne:

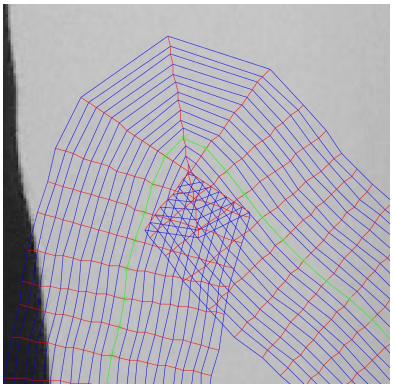


Obrázok 3.13: Ukážka detailu vytvoreného vyhľadávacieho priestoru.

Kde v obrázku zelená línia označuje polohu inicializačného polygónu, z ktorého vrcholov vznikajú senzorové línie. Tie sú v obrázku vyznačené červenou líniou. Červený segment čiary medzi dvoma vrcholmi senzorovej línie slúži v obrázku na zvýraznenie jej orientácie. V reprezentácii priestoru grafom v ňom nereprezentuje možnú cestu. Pre prehľadnosť obrázku sú modrou zakreslené len cesty v grafe, ktoré spájajú vrcholy v rámci jedného stĺpca grafu.

3.4.4 Orezávanie senzorových línií

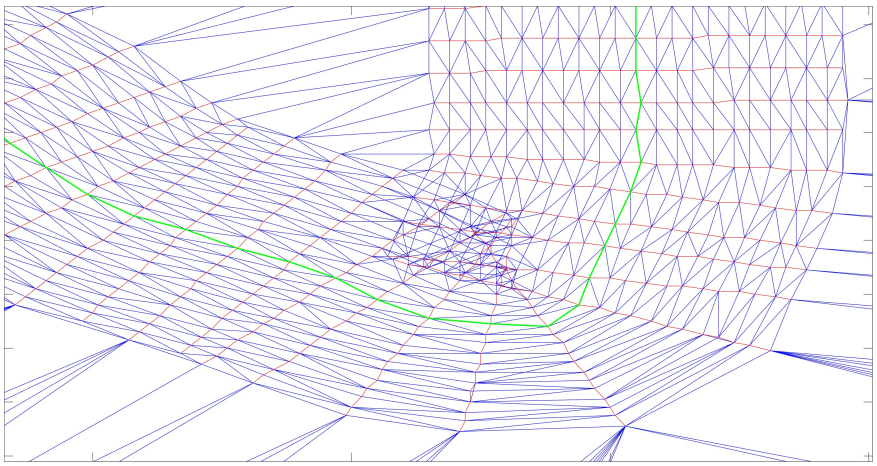
Zakrivenie takto vytvoreného vyhľadávacieho priestoru spôsobuje, že v niektorých miestach prekrýva sám seba. Vzniká v ňom možnosť, kedy by novo vypočítaný prechod s najnižšou cenou obsahoval slučku, teda by pretínal sám seba. Tým by došlo k vytvoreniu krivky, ktorá nie je Jordanovská. Preto potrebujeme ošetriť prípady, v ktorých sa deje toto prekrývanie.



Obrázok 3.14: Ukážka prekrývania vyhľadávacieho priestoru.

Jedným riešením je počítanie priesečníkov línií každej existujúcej dvojice vrcholov z jednej úrovne so všetkými ostatnými dvojicami vrcholov. Toto riešenie by bolo výpočtovo veľmi náročné, keďže pri ňom ide o kvadratickú náročnosť.

Vyhľadávací priestor vieme pomocou 2D Delaunay triangulácie [32] transformovať do obrazu trojuholníkov. Touto operáciou dostaneme štruktúru, ktorá pozostáva z trojuholníkov vytvorených tak, že v kružnici opísanej trojuholníku tvoreného konkrétnymi vrcholmi sa nenachádza žiadny iný vrchol. Ďalšou vhodnou vlastnosťou je, že v štruktúre nevznikajú žiadne vrcholy naviac, teda vrcholy všetkých trojuholníkov z výslednej triangulácie odpovedajú niektorému z vrcholov vyhľadávacieho priestoru. Ukážku vytvorenej triangulácie môžeme vidieť na obrázku 3.15.



Obrázok 3.15: Detail Delaunay triangulácie.

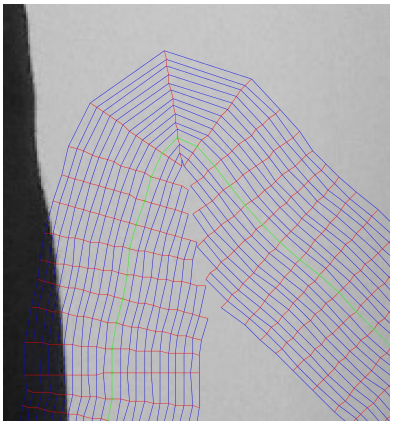
Výsledná štruktúra obsahuje zoznam všetkých trojuholníkov a vrcholov, z ktorých sú vytvorené. Zo štruktúry vieme zistiť informáciu o tom, ktorý vrchol sa nachádza v akom trojuholníku, prípadne s ktorými ďalšími vrcholmi tvorí hranu niektorého z trojuholníkov. Jednotlivým vrcholov vieme podľa ich súradníc priradiť index senzorovej línie, ktorej pripadajú, a taktiež úroveň, v

ktorej sa na danej línii nachádzajú. Úroveň znamená vzdialenosť od stredového vrcholu pripadajúceho aktívnej kontúre, teda o ktorý vrchol v poradí od stredu práve ide. Prácou s týmito informáciami vieme definovať podmienky, za ktorých sa niektoré senzorové línie začnú z niektorého ich konca skracovať.

Rozhodnutie o vylúčení niektorého z koncových vrcholov závisí na najbližšom okolí týchto vrcholov. Orezávací algoritmus sleduje okolie hrany tvorenej dvoma po sebe idúcich vrcholov rovnakej senzorovej línie. Okoliu takejto hrany vo väčšine prípadov odpovedajú presne dva vrcholy. Porovnaním indexov senzorových línií a indexov úrovne, v ktorom sa nachádzajú dané senzory funkcia rozhoduje o orezaní alebo ponechaní aktuálne skúmanej hrany.

V niektorých prípadoch vznikajú v obrázku malé defekty, ktoré sa vymykajú vyššie uvedeným základným podmienkam. Ide asi o 5% všetkých dvojíc po sebe idúcich vrcholov v obrázku. Príkladom je chýbajúca hrana línie tvorenej dvoma po sebe idúcimi vrcholmi $p(i, j)$ a $p(i, j + 1)$ v zozname hrán trojuholníkov vytvorenej triangulácie.

Tento spôsob skracovania má konštantnú zložitosť. Výsledok tohto skracovania je vyobrazený na obrázku 3.16.



Obrázok 3.16: Ukážka úpravy vyhľadávacieho priestoru.

3.4.5 Skracovanie a predlžovanie aktívnej kontúry

Počas behu algoritmu nastávajú situácie, kedy sa kontúra v niektorých miestach zmršťuje a v ďalších sa snaží rozširovať. Čím je určitá časť kontúry definovaná dvoma vrcholmi dlhšia, tým menší detail dokáže zachytiť. Veľká hustota vrcholov kontúry na jednom mieste zase zvyšuje šancu chyby algoritmu a vytvorení slučky v krivke.

Týmto problémom predchádzame uberaním a pridávaním vrcholov do polygónu. To prebieha na základe toho, akú dĺžku majú jednotlivé jej segmenty. Vytvorením rozmedzia, ktoré udáva priemernú dĺžku segmentu. To na základe počtu vrcholov inicializovanej kontúry a na základe celkového rozmeru obrázku. Potom ak niektorý segment prekročí násobok tejto dĺžky, tak je rozdelený na 2 segmenty pridaním vrcholu do stredu tohto dlhého segmentu. V prípade,

že je niektorý segment polovičný alebo kratší, než je dĺžka daná rozhraním, dôjde k spojeniu týchto segmentov tým, že prostredný vrchol je odstránený.

To dovoľuje kontúre sa dostať do rôznych záhybov, teda miest ako je napríklad podpažie alebo priestor rozkroku. Naopak v prípade zhlukovania vrcholov napríklad v okolí ostrých vrcholov ako sú napríklad ruky, princíp znižuje hustotu vrcholov a tým predchádza vzniku chýb v rámci priestoru.

Určujeme maximálny a minimálny počet vrcholov na kontúre. Aby nedošlo k prehnanému doplneniu vrcholov alebo zase odobratiu vrcholov polygónu. Zároveň počet vrcholov v kontúre zvyšuje výpočtový čas celého algoritmu a času trvania jednej iterácie.

Zároveň je to rýchlejší spôsob udržiavania relatívne konzistentnej dĺžky segmentov medzi dvoma vrcholmi, než keby sme zakaždým rovnomerne distribuovali vrcholy v polygóne. Táto operácia je obzvlášť pri polygónoch s vyšším počtom bodov z pohľadu výpočtového času nesmierne náročná.

3.4.6 Úprava pre dynamické programovanie

Aby sme cenovej funkcii počítanej v dynamickom programovaní zabránili vyhodnoteniu najlacnejšej cesty vo vrchole, ktorý sme úpravou vyhľadávacieho priestoru orezali, pridávame na súradnice týchto orezaných vrcholov hodnotu ∞. Tým zaistíme to, že minimálna cesta nikdy neprejde týmto vrcholom.

Akonáhle zistíme, že je senzorová línia orezaná v niektorom vrchole $v(i,j)$, všetky ostatné vrcholy $v(i,j+1)$, $v(i,j+2)$ až $v(i,j+end)$ smerom od stredu, ktoré nasledujú po tomto bode sú tiež automaticky vyhodnotené ako orezané.

3.5 Dynamické programovanie

Dynamické programovanie je algoritmus používaný pre riešenie optimalizačných úloh [18][1]. Riešenie týchto úloh prebieha rozdelením veľkej úlohy na menšie, kedy sú tieto menšie úlohy postupne riešené a ich výsledky sú použité pre riešenie komplexnejšieho problému. Každé riešenie podproblému je zapamätané pre prípad, že ho bude treba riešiť aj v budúcnosti. Je to prístup, kedy sú testované všetky metódy alebo kombinácie na ceste k riešeniu úlohy. Tým, že sa vyskúšajú všetky možnosti, je zaručené, že výsledok je optimálny. Výhodou dynamického programovania je jeho rýchlosť a komplexnosť. Algoritmus sa dá jednoducho rozširovať o ďalšie vstupy.

Naša aplikácia dynamického programovania je využívaná pri hľadaní najnižšej ceny cesty v grafe definovanom v sekcii 3.1.1 a znázornenom na obrázku 3.1. V tomto grafe určujeme najnižšiu cenu cesty, ktorá je závislá na polohe grafu, a od hodnôt daných v jednotlivých vrcholoch a hranách. Cesta je vyhľadávaná postupným priechodom grafu, počas ktorého je zhodnocovaná cena vrcholov a hrán.

3.5.1 Graf a cena cesty

Každý vyhľadávací priestor definovaný v 3.4 je reprezentovaný ako matica MxN, kde M je dĺžka a N je šírka matice. Každá takáto matica obsahuje cesty. Cesta začína v najvrchnejšom riadku riadku matice a pokračuje až na jej posledný riadok tak, že musí prejsť všetkými riadkami matice. Na každom riadku je cesta reprezentovaná iba jedným bodom. Pri prechode medzi jednotlivými riadkami môže cesta pokračovať do bodu v stĺpci s rovnakým indexom ako má bod, z ktorého vychádza, alebo môže pokračovať do jeho susedných bodov. Pre každú cestu platí, že pozostáva z bodov v matici MxN, pre ktorých postupnosť platia rovnice:

$$((1, j_1), (2, j_2), \ldots, (M, j_M)) \text{ kde } 1 \leq j_i \leq N \text{ pre } i = 1 \ldots M$$
$$|j_{i+1} - j_i| \leq 1 \text{ pre } i = 1 \ldots M - 1$$

V sekcii 3.1.1 sme popísali cestu grafom, ako sú tvorené vrcholy a hrany grafu. Každý vrchol a hraná má svoju cenu, teda hodnotu, ktorá ju definuje na základe polohy a značky, ktorú má vrchol, prípadne dvojica vrcholov ktoré spája hrana. Jednotlivé hrany grafu sú potom tvorené vrcholmi:

$$C_{UL}(i,j) = (\mathbf{v_{(i-1,j-1)}}, \mathbf{v_{(i,j)}}) \; \epsilon \; E_{UL} \tag{3.34}$$

$$C_{U}(i,j) = (\mathbf{v_{(i-1,j}}, \mathbf{v_{(i,j)}}) \; \epsilon \; E_{U} \tag{3.35}$$

$$C_{UR}(i,j) = (\mathbf{v_{(i-1,j+1)}}, \mathbf{v_{(i,j)}}) \; \epsilon \; E_{UR} \tag{3.36}$$

Kde (i, j) sú súradnice vrcholu, do ktorého vedú tieto hrany. Pre každú z hrán počítame energie E_E a E_G v závislosti od značiek, ktoré majú vrcholy spojené s aktuálnym vrcholom s indexom (i, j) prostredníctvom hrán $C_{UL}(i, j)$, $C_U(i, j)$ alebo $C_{UR}(i, j)$. Teda nová cena vrchola $C_V(i, j)$ závisí od ceny predchádzajúcich vrcholov a od ceny hrán, ktoré ich navzájom spájajú. Pre tieto ceny v závislosti na energiách môžeme napísať:

$$C_{UL}(i,j) = E_E(\mathbf{v_{(i-1,j-1)}}, \mathbf{v_{(i,j)}} | \lambda_{(i-1,j-1)}, \lambda_{(i,j)}) \tag{3.37}$$

$$C_{U}(i,j) = E_E(\mathbf{v_{(i-1,j}}, \mathbf{v_{(i,j)}} | \lambda_{(i-1,j)}, \lambda_{(i,j)}) \tag{3.38}$$

$$C_{UR}(i,j) = E_E(\mathbf{v_{(i-1,j+1)}}, \mathbf{v_{(i,j)}} | \lambda_{(i-1,j+1)}, \lambda_{(i,j)}) \tag{3.39}$$

Kde najnižšiu cestu do vrcholu $\mathbf{v_{(i,j)}}$ z predchádzajúceho riadku $i - 1$ vypočítame ako:

$$R_{\mathbf{v_{(i,j)}}} = min\{(C_{UL}(i,j) + \mathbf{v_{(i-1,j-1)}}), (C_U(i,j) + \mathbf{v_{(i-1,j)}}), (C_{UR}(i,j) + \mathbf{v_{(i-1,j+1)}})\} \tag{3.40}$$

Potom pre cenu vrchola danom súradnicami (i, j) môžeme napísať:

$$C_V(i,j) = \mathbf{v_{(i,j)}} + R_{\mathbf{v_{(i,j)}}} \tag{3.41}$$

Týmto postupným pričítaním hodnôt predchádzajúcich ciest a vrcholov s hodnotami hodnotami hrán, ktoré vedú k aktuálnemu vrcholu sa dostaneme až na posledný riadok grafu. Odtiaľ vrchol, ktorý má najnižšiu hodnotu reprezentuje najnižšiu cenu cesty grafom a teda minimálnu energiu polygóna $E(C)$.

3.5.2 Rekonštrukcia najlacnejšej cesty

Najkratšia cesta $R_{i,j}$ a jej cena, ktorá predstavuje sumu všetkých vrcholov a hrán tejto cesty, je nájdená použitím dynamického programovania, pomocou ktorého v poslednom kroku, teda na poslednom riadku grafu vypočítame najkratšie cesty do všetkých vrcholov. To za pomoci vypočítaných hodnôt najkratších ciest do vrcholov v predchádzajúcom riadku.

Ako bolo spomenuté v predchádzajúcej časti, v poslednom riadku grafu nájdeme vrchol, ktorý obsahuje najnižšiu cenu cesty R. Zostáva nám len sledovať graf zdola nahor po vrcholoch reprezentujúcich najnižšiu cestu na danom riadku.

K tomu nám pomôže pamäť M do ktorej si zapisujeme pri prvotnom počítaní ciest v grafe v jednotlivých vrcholoch indexy, ktoré predchádzajú tej najlacnejšej ceste. Zároveň počítame, že predchádzajúca najlacnejšia cesta leží nad aktuálnym vrcholom alebo maximálne o jeden stĺpec vľavo alebo vpravo. To potom môžeme zapísať:

$$M(i,j)\epsilon\{j-1, j, j+1\} \tag{3.42}$$

Teda po vypočítaní ciest vo všetkých vrcholoch grafu nám pre rekonštrukciu polôh vrcholov najlacnejšej cesty stačí použiť pamäť M a indexy najlacnejších ciest, ktoré sú v nej uložené.

Kapitola 4
Experiment

Hlavnou úlohou experimentu je overenie funkčnosti navrhnutého riešenia obohatenia modelu aktívnej kontúry o rozpoznané kľúčové body z obrazu. Pre overenie tejto funkčnosti teda vždy porovnávame chovanie modelu aktívnej kontúry len pri využití informácie z hranového obrázka oproti chovaniu aktívnej kontúry pri jej obohatení o výstup z detektora objektov. Predpoklad je, že obohatený model aktívnej kontúry dokáže kontúru ľudskej postavy detekovať kvalitnejšie než model, ktorý detekuje postavu len na základe interných energií obrázku.

Toto porovnanie prebieha na rôznych typoch obrázkov, ktoré sa líšia rôznou zložitosťou pozadia, pred ktorým sa nachádza postava, ktorej kontúru hľadáme. Tieto rôzne podmienky by mali testovať jednak schopnosť správnej detekcie objektov vyobrazených v prirodzenom prostredí, ktoré sa môže často líšiť, tak aj následnú schopnosť algoritmu rozpoznať správne hrany a teda správnu kontúru.

Testovanie rôznych polôh inicializačnej kontúry overí schopnosť algoritmu nájsť optimálne riešenie nad celým obrázkom. To znamená, že algoritmus pri hľadaní minimálnej energie neuviazne v lokálnom minime vytvorenom napríklad hranou, ktorá je tvorená iným objektom, než ľudskou postavou. Takúto hranu ale preskočí, a dôjde k optimálnemu riešeniu, a to za určitý počet iterácií.

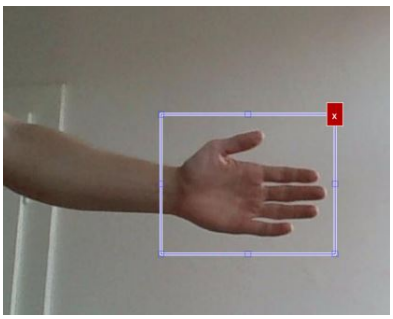
Opakovateľný je experiment, ktorý sa vyznačuje rovnakými podmienkami, čo v našom prípade znamená experiment ktorý začína inicializáciou kontúry v presne danej polohe. Teda napríklad od spomínaného okraja obrázka, prípadne z inak pevne definovanej polohy.

4.1 Detekcia objektov v obraze

Pred samotnou detekciou objektov pomocou detektora Viola-Jones popísanou v kapitole 3.3.1 potrebujeme najprv vytvoriť súbor pozitívnych a negatívnych obrázkov.

Pre rýchly zber obrázkov sme využili kontinuálne snímanie z kamery notebooku. Za krátky čas pohybu v miestnosti sme tak dokázali vytvoriť veľké

množstvo snímkov. Rozlíšenie snímkov z kamery je 1280 x 720 pixlov, teda ide o HD. Tým je zaistená dostatočná kvalita snímkov. Pre vytvorenie sady negatívnych snímkov stačí vytvoriť sadu snímkov, kde sa nevyskytujú hľadané objekty z množiny Λ. V praxi sa stačí prejsť po miestnosti tak, aby kamera mierila od ľudskej postavy. Pre vytvorenie pozitívnych obrázkov musíme ich detail vystrihnúť z obrázkov, kde sa hľadané objekty nachádzajú, respektíve obrázky ručne anotovať. Toto priradenie anotácie vyzerá ako:



Obrázok 4.1: Označovanie pozitívnych vzoriek.

Pre jeden pozitívny obrázok potom získavame popis tvorený dvojicou informácií, kde jedna je vektor popisujúci polohu a veľkosť značky v obrázku, a druhá je samotné pomenovanie značky. Potom pre jednotlivé objekty, ktoré chceme v obraze detekovať, vytvárame sadu 400 pozitívnych obrázkov, ku ktorým je priradená obecná sada 2000 negatívnych obrázkov. Jednotlivé sady pozitívnych a negatívnych obrázkov sú potom predané ako vstup do algoritmu, ktorý učí jednotlivé klasifikátory, z ktorých každý slúži na detekciu jedného objektu v obraze.

Výsledky Viola-Jones

Výstupom z detektora Viola-Jones je mapa polôh, v ktorých došlo k pozitívnej klasifikácii naprieč všetkými kaskádami daného detektora. Teda výstup v danom mieste je 1 pre detekciu a 0 pre miesta v obrázku, kde k detekcii nedošlo.

Po načítaní obrázku nad ním spustíme klasifikátor. Pritom určujeme parameter *MergeThreshold*, ktorý potláča falošné detekcie. Teda detekcie, ktoré neboli rozpoznané v každej kaskáde klasifikácie. Čím vyššia hodnota, tým viac falošných detekcií potlačím. Efekt je znázornený na obrázkoch, kde ide o detekciu pravej ruky pomocou Viola-Jones pri siedmych kaskádach, *FalseRate=0.2* a odlišnom *MergeThreshold*:

(a) : *MergeThreshold = 5* **(b) :** *MergeThreshold = 10*

(c) : *MergeThreshold = 20* **(d) :** *MergeThreshold = 25*

Obrázok 4.2: V-J pri 7 kaskádach, *FalseRate=0.2*.

Pri takto nastavenom detektore sme dokázali spracovávať aj ďalšie vstupy, pričom sme dosahovali prakticky rovnakú úspešnosť pri detekcii.

Výsledky sa v tomto prípade dajú interpretovať pomocou pravdepodobností len obtiažne, pretože algoritmus používa boostované rozhodovacie stromy, ktoré samé o sebe nevedia počítať pravdepodobnostné skóre. Počas testovania sa však prakticky nestalo, že by sa úpravou koeficientov pri trénovaní detektora, zmenou MergeThreshold pri klasifikovaní alebo pridaním väčšieho množstva trénovacích dát nepodarilo eliminovať niektorý z extrémov.

Časová náročnosť programu rástla len pri volení vyšších kaskád v spojení s nízkou hodnotou FalseAlarmRate. Trénovacie data v množstve 300 pozitívnych a 600 negatívnych vzorkov algoritmus natrénoval pri 7 kaskádach a hodnote FalseAlarmRate rovnej 0.2 v priemere za 4 minúty.

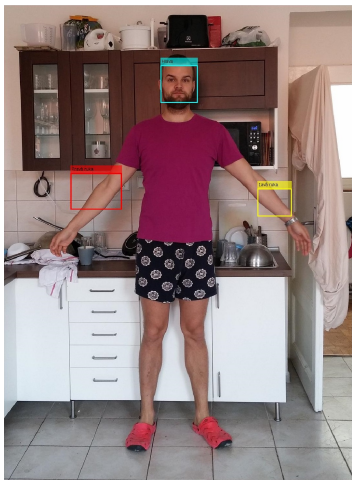
To znamená, že je tento algoritmus robustný a zvládne správne vyhodnotiť veľké množstvo dát za relatívne krátku dobu.

Sa podarilo aplikovať aj na zložitejšie obrázky s ruchom na pozadí ako:

Finálne obrázky s detekovanými oboma rukami a tvárou. V prípade obrázku *(a)* je detekcia správna. V prípade obrázku *(b)* vidíme mierny posun detekovaných rúk. Dôvodom je rušné prostredie. Ďalším dôvodom tohto posunutia môže byť, že obrázky v datasete pozitívnych vzorkov pre ruky majú vo väčšine svetlé pozadie. To by sa dalo napraviť rozšírením trénovacej množiny a množiny negatívnych vzorkov.

Súradnice predaných bodov následne použijem pre modifikáciu funkcie aktívnych kontúr.

(a) : Detekcia v obraze so svetlým pozadím.



(b) : Detekcia v obraze s tmavým pozadím.

Obrázok 4.3: Detekcia hlavy a oboch rúk v obraze.

Po extrakcii hľadaných objektov si uložíme štruktúru prvkov, kde je na každom riadku odkaz na obrázok a vektor reprezentujúci polohu a rozmer označeného vzorku v trénovacom obrázku. Takto vygenerovaný objekt sa spolu so súborom negatívnych vzorkov pošle ako parameter pre funkciu trénovania kaskády. Pre nami volené rozmery trénovacích množín postačovalo takmer vždy 6 kaskád klasifikátorov. To je dané množstvom dostupných trénovacích dát. Čím viac dát, tým viac úrovní klasifikátorov vieme využiť. Hodnotu parametra *FalseAlarmRate* sme držali pod 0.2. Výstupom z funkcie je detektor v *xml* formáte. Ten obsahuje všetky natrénované dáta.

4.2 Hľadanie kontúry obraze

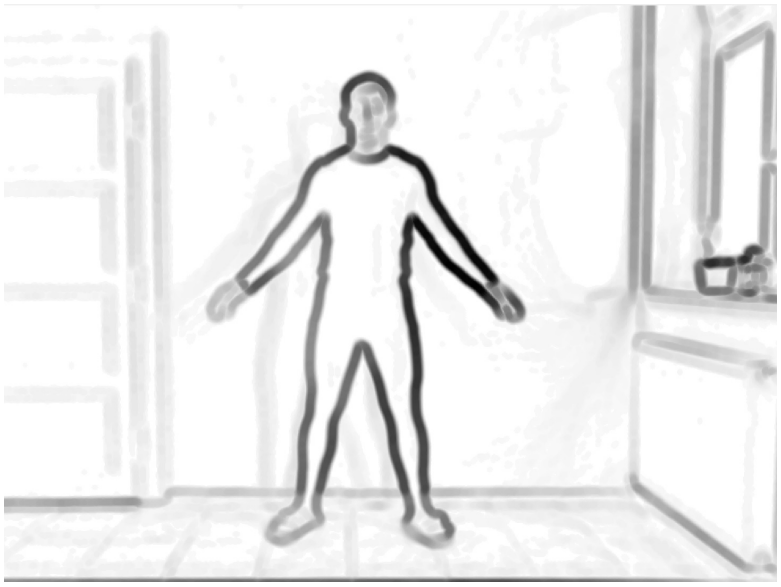
Po natrénovaní detektorov priradíme ich výstupy algoritmu pre hľadanie kontúry v obraze. Ten tieto masky spracuje pomocou energetických rovníc a koeficientov, teda prepočíta ich vplyv vzhľadom na kombinácie ich výskytov v rámci jednotlivých vrcholov. V tejto časti zároveň porovnávame funkčnosť modelu, ktorý minimalizuje energiu len na základe hranového obrázku oproti modelu, ktorý pracuje aj s informáciami z detektora objektov.

4.2.1 Obraz s jednoduchým pozadím

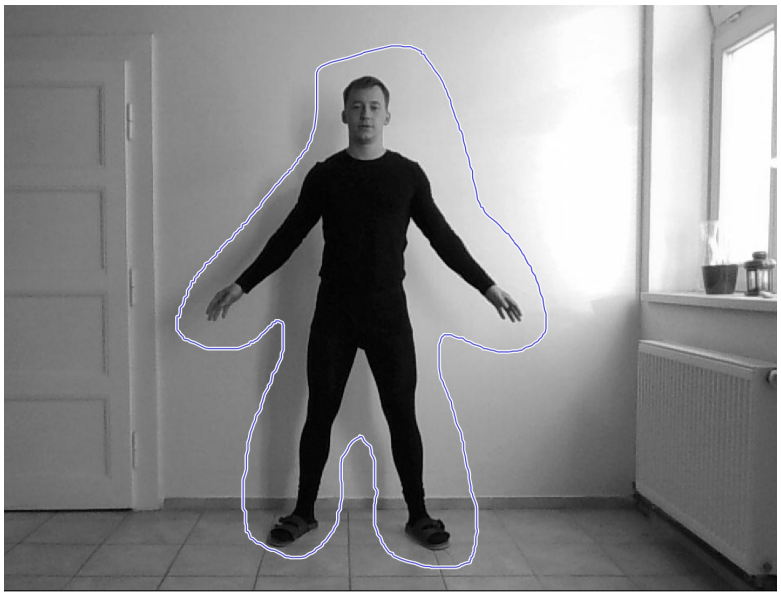
V prvom prípade sme implementáciu testovali na obrázku, v ktorom sa snažíme napodobniť ideálne podmienky. Teda splnenie požiadavkov na polohu a postavenie postavy spomenutých v časti 3.2, no naviac sa snažíme vytvoriť čo najvýraznejší kontrast postavy s jej pozadím. Postava je oblečená celá v čiernom a je postavená pred bielu stenu. Po prevedení do šedého spektra sa hrany v obrázku javia voľným okom dosť výrazné, viz obrázok 4.4.

Obrázok 4.4: Obrázok s jednoduchým pozadím, v ktorom prebieha detekcia.

Avšak vzhľadom na existujúce nasvietenie, ktoré vstupuje do miestnosti skrz okno, a ktoré nebolo možné kompenzovať interným osvetlením vznikajú na stene za stojacou osobou tiene. Tieto tiene nie sú výrazné pri pohľade na čiernobiely obrázok, no stanú sa výraznejšími v hranovom obrázku, kde sa prejavujú ako šedé línie, viz obrázok 4.5 bod a). Pred vytvorením hranového obrázku bol pôvodný obraz filtrovaný Gaussiánom s hodnotou $\sigma = 3$, no stále sú v obraze viditeľné hrany, ktoré tvorí svetlo na drsnejšej štruktúre steny. Tento šum by sa dal ďalej filtrovať použitím vyššej hodnoty σ pri aplikácii filtra Gaussiánu. Výraznejší tieň od postavy by však táto úprava nepotlačila.



(a) : Hranový obrázok.



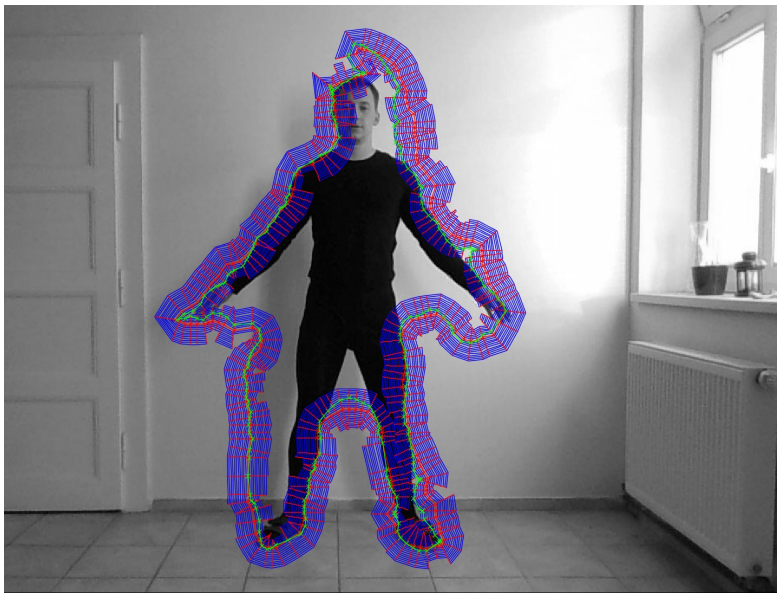
(b) : Inicializačná kontúra.

Obrázok 4.5: Obrázky, ktoré predchádzajú hľadaniu kontúry.

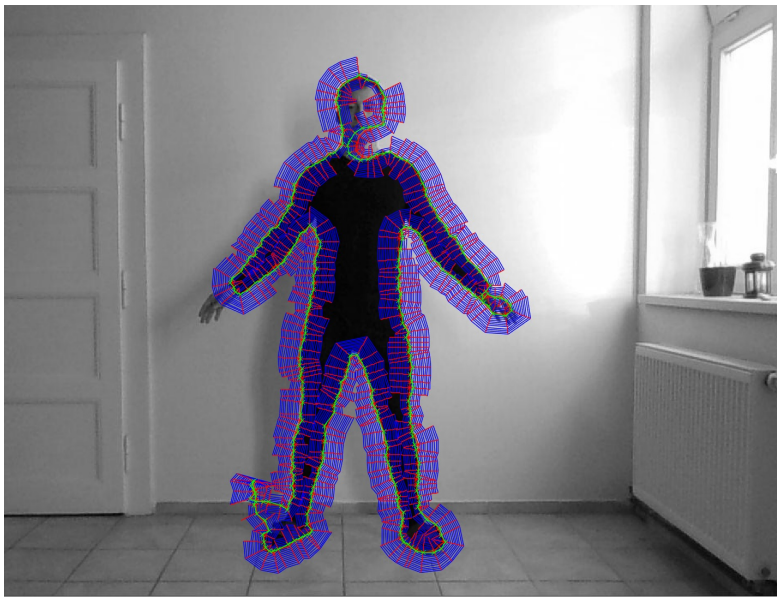
Po filtrovaní obrázku, vytvorení hranového obrázku a načítaní detektorov popísaných v predchádzajúcej časti 4.1 a detekcii objektov v obrázku pokračujeme inicializáciou kontúry v blízkosti hľadaného objektu, teda v blízkosti ľudskej postavy. Táto inicializácia je vyobrazená na obrázku 4.5 v bode *b*).

Táto inicializačná kontúra tvorí počiatočnú polohu, od ktorej sa odvíja model aktívnej kontúry. To postupným rozvojom a prehľadávaním vyhľadávacieho priestoru. Ten je v nasledujúcich obrázkoch 4.6, 4.7 a ďalších vyznačený modrou farbou. Zelenou farbou je poloha aktívnej kontúry, ktorá pokračuje

do ďalšieho kroku iterácie, respektíve ktorá odpovedá riešeniu úlohy.



(a) : Iterácia číslo 3.

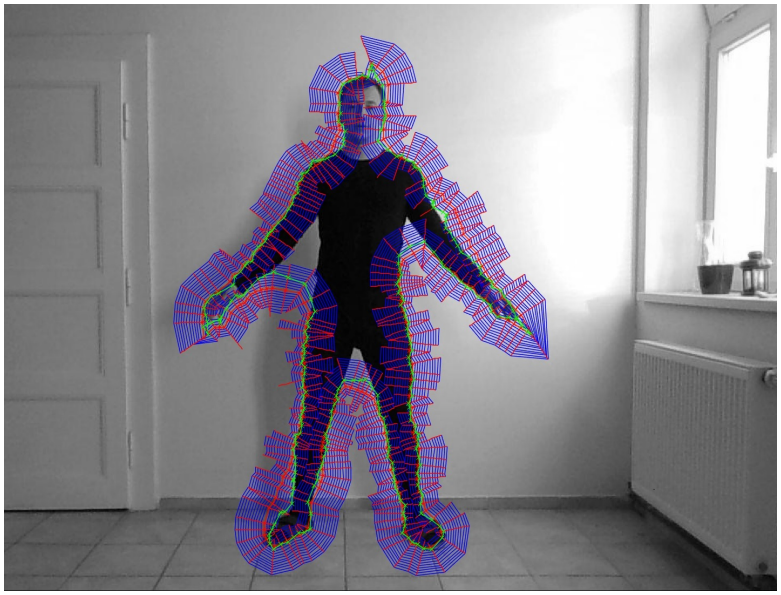


(b) : Posledná iterácia číslo 14.

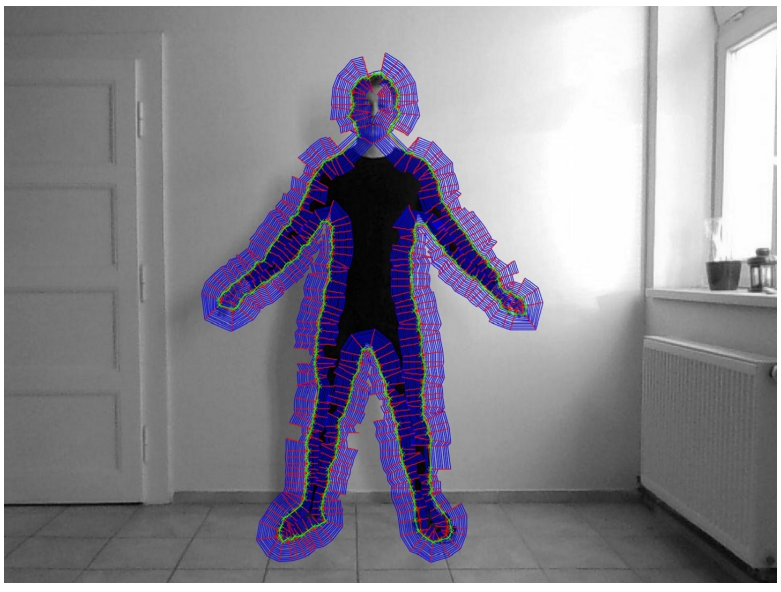
Obrázok 4.6: Hľadanie kontúry bez využitia informácie z detektora.

Na obrázku 4.6 je znázornený priebeh vývinu aktívnej kontúry, ktorá vychádza len z internej energie obrázku. V obrázku v bode *a*) je viditeľné, ako kontúra v pravej časti obrázku sleduje hranu tieňa, ktorý vrhá postava na stenu za sebou. Táto poloha je po niekoľkých iteráciách potlačená, respektíve aktívna kontúra zachytí silnejšiu hranu, ktorú tvorí kontúra nohy. Avšak algoritmus nadobúda extrémnu krivosť v oblasti krku, kam je priťahovaný hranou, ktorá je tvorená hranicou medzi tričkom a krkom postavy. Zároveň

v oblasti dlaní dochádza k strate s kontaktom s touto časťou tela a aktívna kontúra je znovu priťahovaná k hranici tvorenej rukávom a rukou osoby. Teda je tu viditeľný silný vplyv hrán na výsledné riešenie a na počet krokov, za ktoré model aktívnej kontúry dospeje k riešeniu.



(a) : Iterácia číslo 4.



(b) : Posledná iterácia číslo 7.

Obrázok 4.7: Hľadanie kontúry s využitím informácie z detektora.

Na obrázku 4.7 je viditeľný odlišný vývin modelu aktívnej kontúry. V miestach, kde sa nachádza tieň spôsobený postavou, detektor nerozpoznal žiadny z objektov. Teda aktívna kontúra pokračuje v hľadaní miesta, ktoré sa vyznačuje silnou hranou a zároveň určitou značkou. Teda model dospel

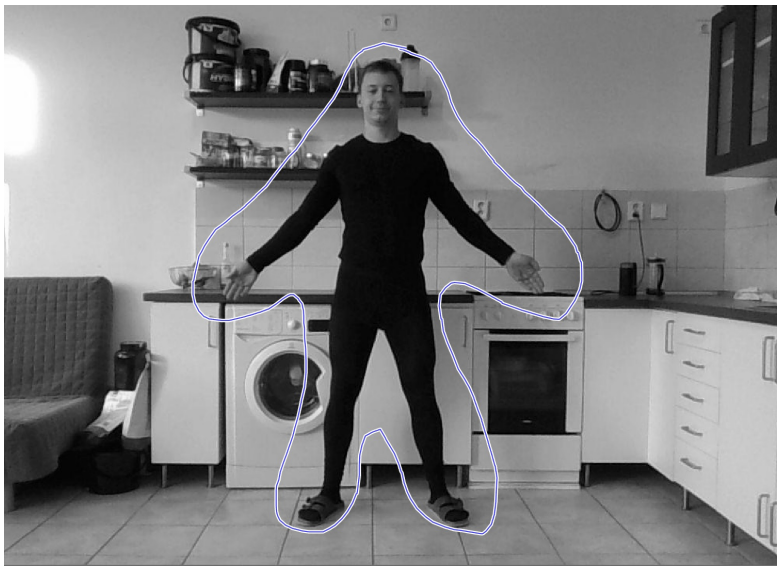
k hľadanej hrane nohy podstatne skôr. Naviac vďaka daným parametrom krivosti v oblasti hlavy nedochádza k extrémnemu sledovaniu hrany, ktorá by viedla model aktívnej kontúry do oblasti krku.

Beh iterácie bol ukončený z dôvodu dosiahnutia energetického optima. Kontúra obsahuje všetky detekované objekty a popisuje takmer ideálnu kontúru ľudskej postavy. V prípade, že by model nebol kontrolovaný energiou dodanou z detektora objektov, minimalizácia by pokračovala a pravdepodobne by došlo k zásahu aktívnej kontúry do postavy človeka v miestach rúk alebo krku, podobne ako tomu bolo v prvom prípade popísanom na obrázku 4.6.

4.2.2 Obraz so zložitejším pozadím

Obraz so zložitejším pozadím znamená, že pozadie pozostáva z viacero odlišných štruktúr a tvarov, než je čistá biela stena. Detekcia objektov a hľadanie kontúry prebiehalo s pozadím, v ktorom nehrali tiene tak výraznú rolu, zato sa tu vyskytovalo podstatne viac iných a výraznejších hrán. Podľa obrázku 4.8 vidíme, že hrany dané rohmi kuchynskej linky, prípadne hrana tvorená prechodom z chladničky na skrinku, sú silnejšie, než je hrana, ktorá separuje osobu od pozadia.

(a) : Hranový obrázok.



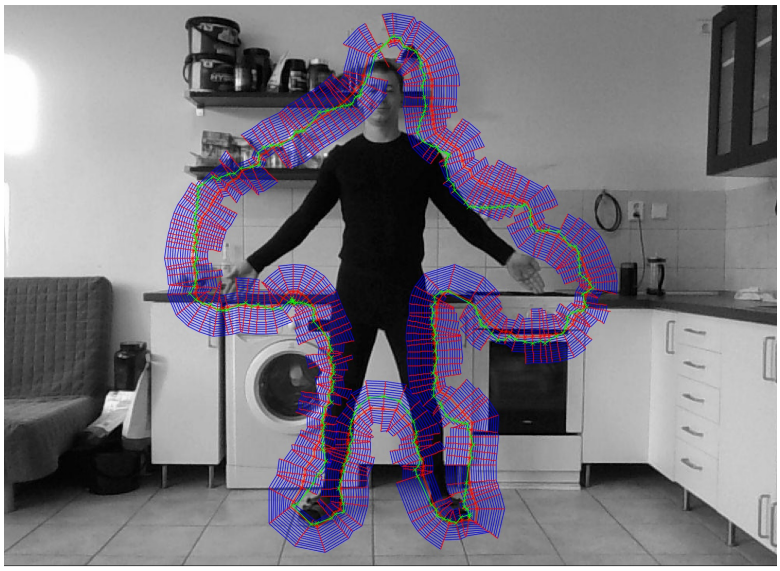
(b) : Inicializačná kontúra.

Obrázok 4.8: Obrázky, ktoré predchádzajú hľadaniu kontúry.

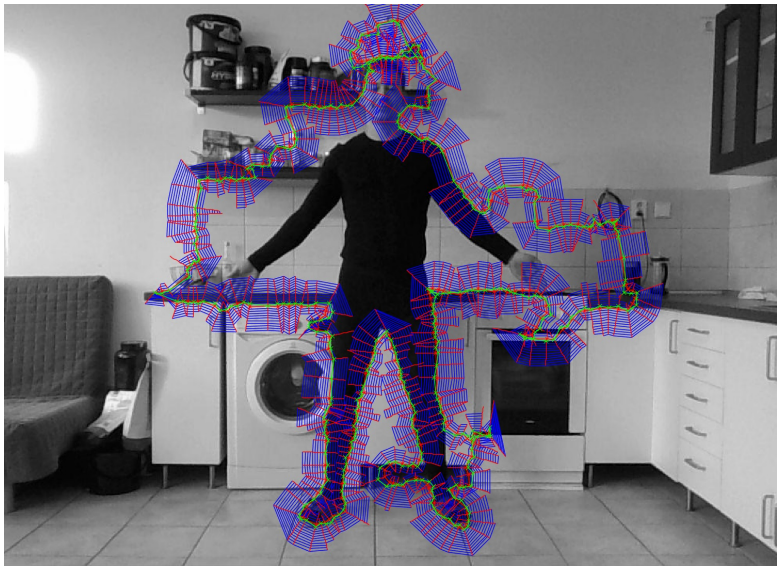
Po filtrovaní obrázku, vytvorení hranového obrázku a načítaní detektorov popísaných v predchádzajúcej časti 4.1 pokračujeme inicializáciou kontúry v blízkosti hľadaného objektu, teda v blízkosti ľudskej postavy. Táto inicializácia je vyobrazená na obrázku 4.8 v bode *b*).

V prípade obrázku 4.9, v ktorom hľadá model aktívnej kontúry riešenie bez pomoci informácií z detektora je od začiatku viditeľné, ako je kontúra priťahovaná silnými hranami tvorenými kuchynskou linkou, prípadne prechodom medzi kachličkami a stenou. V tomto prípade by sa model rozvíjal ďalej, až by narazil na horný okraj obrázku a na hranu tvorenú kuchynskou linkou. Výsledná kontúra teda z daného miesta bez ďalších informácií ľudskú kontúru

nenájde.



(a) : Iterácia číslo 4.

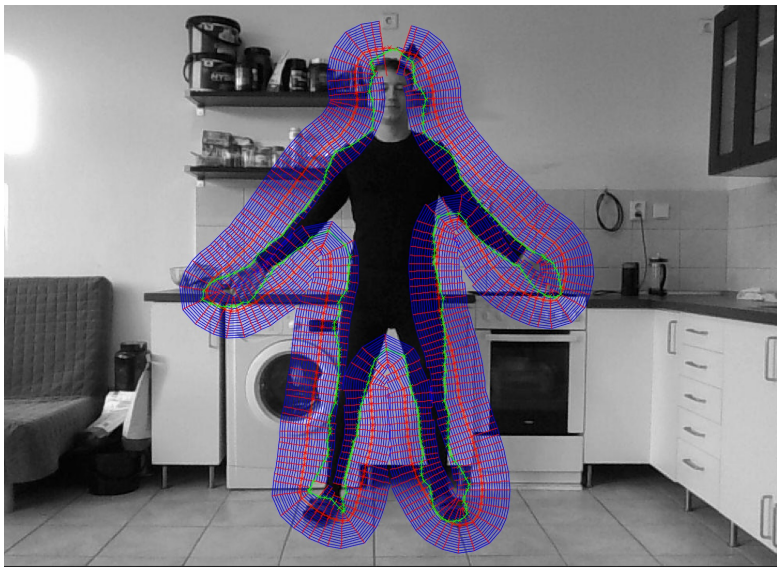


(b) : Posledná iterácia číslo 11.

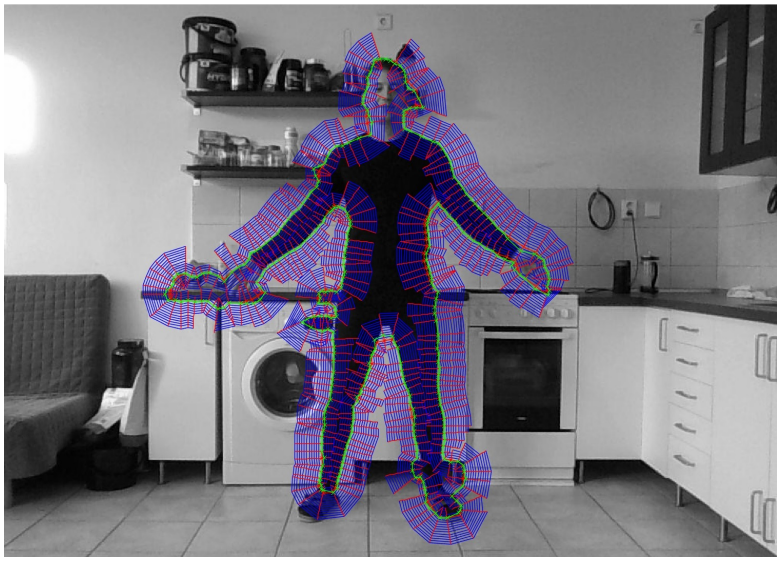
Obrázok 4.9: Hľadanie kontúry bez využitia informácie z detektora.

Na obrázku 4.10 je znázornený priebeh vývinu aktívnej kontúry, ktorej miesto inicializácie z predchádzajúcej iterácie je vyznačené červenou líniou, ktorá prechádza stredom vyhľadávacieho priestoru. Zelená línia popisujúca polohu kontúry, ktorá bola rozpoznaná v aktuálnej iterácii, sa pohybuje smerom ku kontúre a nie naopak. Problém s nájdeným riešením je v oblasti hlavy, kde kontúra čiastočne popisuje obrys police, ktorá sa nachádza za postavou. Podobne v prípade pravej ruky, kde dochádza k rozšíreniu oblasti aj na objekt, ktorý ruka v danom mieste prekrýva. Podobne je to i v mieste

chodidla, kde v prípade pravého chodidla kontúra vynechala samotné chodidlo. Koncové body pritom zaznamenali detekcie, teda z pohľadu značiek vrcholov kontúry je všetko v poriadku. Chýbajúca detekcia trupu tela nedokázala zabrániť zakriveniu kontúry v mieste, kde by správne mala byť rovná. Preto kontúra opísala časť hrany patriacej objektom za stojacou postavou.



(a) : Iterácia číslo 2.



(b) : Posledná iterácia číslo 6.

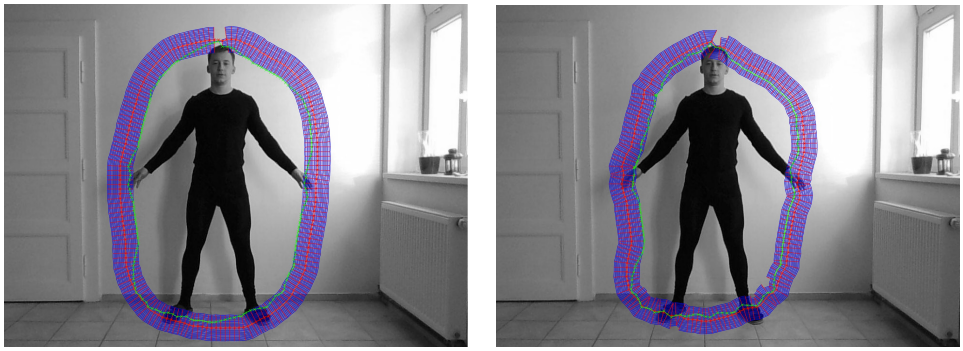
Obrázok 4.10: Hľadanie kontúry s využitím informácie z detektora.

4.2.3 Inicializácia kontúrou oválneho tvaru

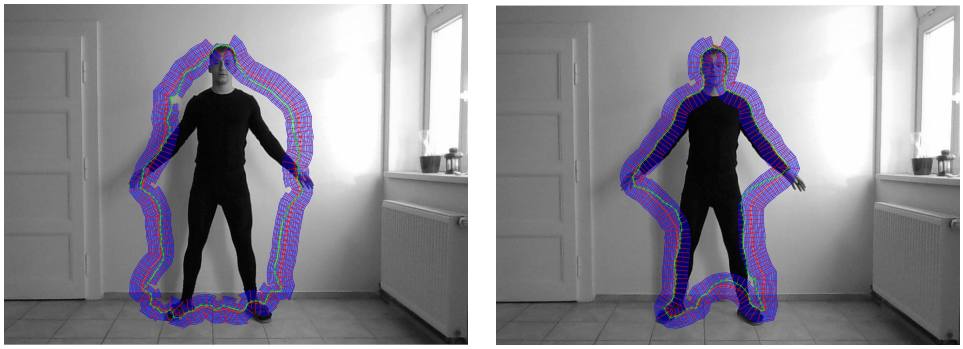
Testujeme chovanie kontúry pri tvare inicializačnej kontúry, ktorá neodpovedá tvaru hľadanej kontúry. Teda pre prípad, ak inicializačná kontúra nemá

na začiatku tvar hľadaného objektu a naviac je od neho mierne vzdialená. Testujeme verziu rozšírenú o detektor na jednoduchom obrázku viz 4.4.

Algoritmus detekoval kontúru po 32 iteráciách, no vzhľadom na výraznejšie hrany, ktoré reprezentujú rukáv trička, nezahrnul do kontúry dlane. Podobne vynechal nohy.



(a) : *Iterácia 2.* **(b) :** *Iterácia 4.*



(c) : *Iterácia 8.* **(d) :** *Iterácia 12.*

(e) : *Iterácia 18.* **(f) :** *Iterácia 32.*

Obrázok 4.11: Hľadanie kontúry pri inicializácii tvarom odlišným od kontúry.

4.3 Vyhodnotenie experimentu

V experimentoch popísaných v bodoch 4.2.1 a 4.2.2 sme porovnávali výkonnosť modelu aktívnej kontúry bez a s informáciou z detektora kľúčových bodov. Algoritmus v prípade jednoduchšieho pozadia rozpoznal hľadanú kontúru v oboch prípadoch. V prípade s použitím detektora bola detekcia o niečo hladšia a hlavne rýchlejšia. Na tvar kontúry sa podieľa priamo výstup z detektora. V prípade obtiažnejšieho pozadia mal algoritmus bez detektora značne horší výsledok, kedy k detekcii kontúry nedošlo. Dôvodom boli veľmi výrazné hrany v okolí, ktorým sa bez vyššej informácie model aktívnej kontúry nedokázal vyhnúť. V prípadne náročnejšieho obrázku s použitím detektora sa model aktívnej kontúry nevzdialil od hľadanej kontúry a detekoval ju relatívne presne.

Algoritmus iteroval k riešeniu, prípadne predčasnému ukončeniu algoritmu z dôvodu nájdenia riešenia, respektíve z iných dôvodov, ako je zasiahnutie mimo obrázok, prípadne detekovaná extrémna dĺžka kontúry. Počet iterácií sa pohyboval v prípade základnej implementácie bez rozšírenia o výstup z detektora v počte 11 až 14 iterácií. V prípade rozšírenej implementácie išlo o 6 až 7 iterácií. V prípade inicializovanej kontúry, ktorá vôbec neodpovedala tvaru hľadanej kontúry proces trval až 32 iterácií.

Trvanie jednej iterácie je približne 2 až 5 sekúnd, teda algoritmus je relatívne rýchly. Spomalenie nastáva ak pracuje s rovnomernou distribúciou bodov po kontúre. V rámci jednej iterácie sa kontúra pohne maximálne o 70 pixlov, čo predstavuje približne 15% celej výšky obrázku.

Pokiaľ bola aktívna kontúra inicializovaná v blízkosti hľadanej kontúry, prípadne ešte spôsobom, kedy jej senzorové línie zasiahli túto kontúru, algoritmus prakticky vždy konverguje k riešeniu, ktoré z veľkej časti odpovedá hľadanej kontúre ľudskej postavy. V prípade, že ide o inicializáciu z okraja obrázku, ako je uvedené v obrázku 4.11, model aktívnej kontúry má problém usporiadať svoje body tak, aby dokázal bez prerušenia detekovať kontúru ľudskej postavy. Riešením by malo byť lepšie vyladenie koeficientov, ktoré určujú vplyv detekovaných značiek na tvar kontúry v určitých jej miestach. Ďalším elementom, ktorý na to má vplyv je slabšia schopnosť aktívnej kontúry meniť svoju dĺžku, respektíve meniť dĺžku svojich segmentov. Zvolený postup výpočtu z obrázku, respektíve aktuálnych rozmerov aktívnej kontúry nespĺňa požiadavky na chovanie.

Problémom ktorý spôsobuje čiastočne detektor, a ktorý model vzhľadom na jeho nastavené parametre nevie úplne riešiť je neschopnosť rozlišovať tvary. Tento jav je viditeľný na obrázku 4.10, kde došlo k orezaniu chodidla, pritom sa tam nevyskytuje hrana. To je spôsobené tým, že v rámci tejto kratšej cesty, ktorú algoritmus určil ako najlacnejšiu, je suma nehranových pixlov nižšia, než je suma hranových pixlov po obvode kontúry. To by bolo možné riešiť použitím napríklad *beta rozdelenia* pri úprave hodnôt z hranového obrázka. To by zvýšilo preferenciu hrany tým, že by pridala nepomerne vyššiu cenu miestam, kde sa hrana určite nenachádza.

Niektoré tieto zistené nedostatky je možné upraviť vyladením koeficientov, ktoré upravujú vplyv detekovaných značiek na jednotlivé energie v obrázku. Ďalším priestorom je doplnenie detektora, respektíve jeho pretrénovanie za účelom zníženia falošne pozitívnych detekcií.

Kapitola 5

Záver

V práci sme popísali prakticky vlastný model aktívnej kontúry, v rámci ktorého sme definovali jeho energiu a pravidlá jej minimalizácie pomocou dynamického programovania. Taktiež sme definovali vyhľadávací priestor, ktorého reprezentáciu dostaneme pomocou Delaunay triangulácie, nad ktorou dokážeme vykonávať matematické operácie a ktorej priestor vieme prehľadávať za konštantný čas. Energiu, ktorá popisuje model aktívnej kontúry sme obohatili o informácie z detektora objektov a kľúčových bodov. To tak, že s týmito informáciami dokážeme pracovať v rámci algoritmu dynamického programovania.

Implementovaný vplyv Viola-Jones detektora na model aktívnej kontúry sme demonštrovali experimentom, kde sme ukázali na jednoduchšom i zložitejšom obrázku rozdiel, ak model pracuje len na základe informácií z nižšej úrovne, a prípad, kedy model pracuje aj na základe informácií o konkrétnych objektoch a ich polohách v obrázku. Pridaním tohto vplyvu sme dokázali ovplyvniť krivosť aktívnej kontúry v niektorých jej vybraných miestach. Napríklad pri páse a nohách. Funkčnosť modelu aktívnej kontúry sme demonštrovali i pri použití inicializačnej kontúry, ktorej tvar bol odlišný od tvaru kontúry, ktorý hľadáme. Model aktívnej kontúry dokázal samostatne detekovať kontúru ľudskej postavy v obrázku.

Ďalšie zlepšenie presnosti modelu a zlepšenie jeho schopnosti zahrnúť do kontúry i chodidlá a dlane, ktoré teraz boli kvôli prítomnosti rukávu zanedbávané, je možné dosiahnuť použitím regresnej funkcie namiesto detektora, ktorý aktuálne určí oblasť bez ohľadu na jej tvar. Definovaním regresnej funkcie, ktorá dokáže zobraziť lokálny obrázok a previesť ho na hodnotu krivosti, by sme dokázali lepšie rozpoznávať konkrétne tvary. Tým by sme dokázali lepšie detekovať konkrétne hranice objektov. V praxi by to ale znamenalo označkovať veľa obrázkov.

Ďalšou variantou k použitiu Viola-Jones detektora je použitie hlbokých neurónových sietí, ktoré dokážu riešiť takéto problémy podstatne rýchlejšie a efektívnejšie, prípadne využitie Skrytých Markovských reťazcov, ktoré môžeme aplikovať vďaka nami definovanej štruktúre modelu. Tá pozostáva z postupnosti značiek. Táto postupnosť sa dá pomocou tejto metódy vhodne optimalizovať.

Literatúra

[1] Milan Sonka, Václav Hlaváč a Roger Boyle. *Image Processing, Analysis, and Machine Vision*. 3rd edition. Toronto: Thomson Learning, 2007. ISBN-10: 0-495-08252-X.

[2] Christopher M. Bishop. *Pattern recognition and machine learning*. New York: Springer Science+Business Media, 2006. ISBN-10: 0-387-31073-8.

[3] Elena Šikudová, Zuzana Černeková, Wanda Benešová, Zuzana Haladová a Júlia Kučerová. *Počítačové videnie. Detekcia a rozpoznávanie objektov*. Praha: Vydavateľstvo Wikina, 2011. ISBN: 978-80-87925-06-5.

[4] Richard Szeliski. *Computer Vision: Algorithms and Applications*. London: Springer, 2011. ISBN: 978-1-84882-934-3.

[5] Heiko Hirschmüller, Korbinian Schmid, Michael Suppa. *Computer Vision for Mobile Robot Navigation*. Wichmann/VDE Verlag, Belin and Offenbach, pp. 143-152, 2015.

[6] David Gerónimo, Antonio López, and Angel D. Sappa. *Computer Vision Approaches to Pedestrian Detection: Visible Spectrum Survey*. Computer Vision Center, Universitat Autonoma de Barcelona, 2007.

[7] Jonathan Rihan. *Computer Vision Based Interfaces for Computer Games*. Oxford Brookes University, 2010.

[8] Freeman, W., Tanaka, K., Ohta, J. and Kyuma, K. *Computer vision for com- puter games*, pp. 100–105. 1996.

[9] Caeselles Vincet, Kimmel Ron and Sapiro Guillermo *Geodesic active contours*. International Journal of Computer Vision, pp. 61-79, 1997.

[10] Václav Hlaváč a Miloš Sedláček. *Zpracování signálu a obrazu*. Fakulta elektrotechnická, ČVUT, 2005. ISBN: 80–01–03110–1.

[11] Demetri Terzopoulos, Michael Kass and Andrew Witkin. *Snakes: Active Contour Models*. International Journal of Computer Vision, pp. 321-331, 1998. Dostupné z `http://web.cs.ucla.edu/~dt/papers/ijcv88/ijcv88.pdf`

[12] Blake, A. and Isard, M. *Active Contours.* [online], Springer 1998. Dostupné z: `http://www.vavlab.ee.boun.edu.tr/courses/574/material/books/blake_Active_Contours.pdf`

[13] T. F. Chan and L. A. Vese. *Active contours without edges.* IEEE Trans. Image Processing, pp. 266–277, 2001.

[14] C. Sminchisescu and A. Telea. *Human pose estimation from silhouettes. A consistent approach using distance level sets.* Department of Mathematics and Computer Science, Eindhoven University of Technology, 2002.

[15] Mun Wai Lee and Isaac Cohen. *Human upper body pose estimation in static images.* Institute for Robotics and Intelligent Systems, University of Southern California, 2004.

[16] Carlos Barŕon and Ioannis A. Kakadiaris. *Estimating Anthropometry and Pose from a Single Uncalibrated Image.* Department of Computer Science, University of Houston, 2000.

[17] Camillo J. Taylor. *Reconstruction of Articulated Objects from Point Correspondences in a Single Uncalibrated Image.* GRASP Laboratory, CIS Department, University of Pennsylvania, 2000.

[18] Yu, T. – Luo, J. – Ahuja, N. *Search Strategies for Shape Regularized Active Contour.* Computer Vision and Image Understanding 113, pp. 1053-1063, 2009.

[19] Pedro F. Felzenszwalb and Ramin Zabih. *Dynamic Programming and Graph Algorithms in Computer Vision.* Computer Science Department, University of Chicago, 2011.

[20] Kwok-Wai Wong, Kin-Man Lam, Wan-Chi Siu. *An efficient algorithm for human face detection and facial feature extraction under different conditions.* Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, 2000.

[21] I. Haritaoglu, D. Harwood, L. Davis,*Real Time Detection and Tracking of People and their Parts.* Technical Report, University of Maryland, 1997.

[22] N. Loewke *Depth-Based Image Segmentation* [online], Stanford University. 2015. Dostupné z `https://stanford.edu/class/ee367/Winter2015/report_loewke.pdf`

[23] *Basic Concepts in Digital Image Processing* [online], Florida State University. 2002. Dostupné z `https://micro.magnet.fsu.edu/primer/digitalimaging/imageprocessingintro.html`

[24] Bahadir K. Gunturk. *EE 7730 - Image Analysis* [online], Division of electrical and computer engineering, Lousiana State University. 2009. Dostupné z `http://slideplayer.com/slide/5199185/`

[25] Paul Viola, Michael J. Jones. *Robust Real-Time Face Detection.* International Journal of Computer Vision 57(2), pp. 137–154, 2004. Dostupné z http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf

[26] Paul Viola, Michael J. Jones. *Rapid Object Detection using a Boosted Cascade of Simple Features.* IEEE CVPR, 2001.

[27] Yoav Freund, Robert E. Schapire. *A decision-theoretic generalization of on-line learning and an application to boosting.* In Computational Learning Theory: Eurocolt 1995, 23–37, 1995.

[28] *Convert RGB image or colormap to grayscale* [online]. The MathWorks, Inc. http://www.mathworks.com/help/matlab/ref/rgb2gray.html

[29] Miroslav Čepek. *Lineární klasifikátor, rozšíření báze, LDA, logistická regrese* [prednáška], FEL ČVUT 2014. Dostupné z https://cw.fel.cvut.cz/wiki/_media/courses/a7b36vyd/prednasky/09_-_linear.pdf

[30] Vojtěch Franc a Václav Hlaváč. *Statistical Pattern Recognition Toolbox for Matlab - User's guide* [online], CTU-CMP, 2014. Dostupné z http://cmp.felk.cvut.cz/cmp/software/stprtool/stprtool.pdf

[31] Liang Zhao. *Dressed Human Modeling, Detection and Parts Localization* [online], The Robotics institute, Carnegie Mellon University Pittsburgh, 2001. Dostupné z https://www.ri.cmu.edu/pub_files/pub4/zhao_liang_2001_1/zhao_liang_2001_1.pdf

[32] D. T. Lee and B. J. Schachter *Two Algorithms for Constructing a Delaunay Triangulation* [online], International Journal of Computer and Information Sciences, Vol. 9, No. 3, 1980. Dostupné z http://www.personal.psu.edu/cxc11/AERSP560/DELAUNEY/13_Two_algorithms_Delauney.pdf

[33] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester a Deva Ramana. *Object Detection with Discriminatively Trained Part Based Models* [online], Computer Science Division, Berkeley, 2010. Dostupné z https://www.cs.berkeley.edu/~rbg/papers/Object-Detection.pdf

[34] Constantine Papageorgiou, Theodoros Evgeniou a Tomaso Poggio. *A Trainable Pedestrian Detection System* [online], Center for Biological and Computational Learning and Artificial Intelligence L aboratory, MIT, Cambridge, 1998. Dostupné z http://www.ai.mit.edu/courses/6.899/papers/icip99.published.pdf

[35] *Caltech 256 images dataset.* [online dataset]. Computational Vision at Califorina Institute of Technology. 2011. Dostupné z http://www.vision.caltech.edu/Image_Datasets/Caltech256/images/

[36] *The Database of Faces.* [online dataset]. AT&T Laboratories Cambridge University. Dostupné z http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[37] *UCI Machine Learning Repository: Data Sets.* [online dataset]. Center for Machine Learning at University of Carolina, Irvine. 2016. Dostupné z `http://archive.ics.uci.edu/ml/datasets.html`

[38] *MPII Human Pose Dataset.* [online dataset]. Max Planck Institut Informatik. 2014. Dostupné z `http://human-pose.mpi-inf.mpg.de/`

[39] *Train support vector machine classifier* [online]. Statistics and Machine Learning Toolbox, The MathWorks, Inc. Dostupné z: `http://www.mathworks.com/help/stats/svmtrain.html`

[40] *Classify using support vector machine* [online]. Statistics and Machine Learning Toolbox, The MathWorks, Inc. Dostupné z: `http://www.mathworks.com/help/stats/svmclassify.html`

[41] *Rearrange image blocks into columns* [online]. Image Processing Toolbox, The MathWorks, Inc. Dostupné z: `http://www.mathworks.com/help/images/ref/im2col.html`

[42] *Cascade Trainer: Specify Ground Truth, Train a Detector* [online]. Brett Shoelson, 2013. Dostupné z: `http://www.mathworks.com/matlabcentral/fileexchange/39627-cascade-trainer--specify-ground-truth--train-a-detector`

[43] *Specify polygonal region of interest - ROI* [online]. Image Processing Toolbox, The MathWorks, Inc. Dostupné z: `http://www.mathworks.com/help/images/ref/roipoly.html`