ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Detekce přítomnosti zboží v ruce zákazníka
Student:	Oliver Keruľ-Kmec
Vedoucí:	doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce zimního semestru 2017/18

Pokyny pro vypracování

1) Seznamte se s úlohou detekce ruky zákazníka při jeho manipulaci se zbožím v regálu a způsobem měření tohoto chování pomocí kamerového systému, které bylo použito pro měření v prodejně Albert.
2) Nastudujte vybrané metody pro zpracování obrazu využitelné pro danou úlohu.
3) Na základě znalosti procesu měření a analýzy naměřených dat navrhněte postup a dílčí metody pro detekci přítomnosti zboží v ruce zákazníka. Zohledněte specifika snímacího procesu.
4) Navržené metody implementujte v programovacím jazyku C++ s využitím volně dostupných knihoven.
5) Implementované metody ověřte na nezávislé sadě naměřených dat ze stejného měření, která jsou k dispozici, vyhodnoťte dosaženou přesnost a navrhněte možná zlepšení.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 26. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Bakalárska práca

Detekcia prítomnosti tovaru v ruke zákazníka

Oliver Keruľ-Kmec

Vedúci práce: doc. RNDr. Ing. Marcel Jiřina, Ph.D.

11. mája 2016

Poďakovanie

Týmto by som chcel poďakovať vedúcemu mojej práce Marcelovi Jiřinovi za možnosť pravidelných konzultácií a za všetky cenné rady a nápady počas tvorby tejto práce. Ďalej by som chcel poďakovať Jirkovi Ambrožovi za spoluprácu pri poskytovaní nameraných dát a tiež za konzultáciu niektorých problémov.

V neposlednom rade ďakujem aj mojej rodine a priateľke Tatiane za podporu a trpezlivosť nielen počas písania tejto práce, ale aj počas celého štúdia.

Vyhlásenie

Vyhlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov. V súlade s ustanovením § 46 odst. 6 tohoto zákona týmto udeľujem bezvýhradné oprávnenie (licenciu) k užívaniu tejto mojej práce, a to vrátane všetkých počítačových programov, ktoré sú jej súčasťou alebo prílohou, a tiež všetkej ich dokumentácie (ďalej len „Dielo“), a to všetkým osobám, ktoré si prajú Dielo užívať. Tieto osoby sú oprávnené Dielo používať akýmkoľvek spôsobom, ktorý neznižuje hodnotu Diela (vrátane komerčného využitia). Toto oprávnenie je časovo, územne a množstevne neobmedzené.

V Prahe 11. mája 2016 .....

Odkaz na túto prácu

Keruľ-Kmec, Oliver. *Detekcia prítomnosti tovaru v ruke zákazníka.* Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Táto bakalárska práca sa zaoberá detekciou tovaru v ruke zákazníka na kamerovom zázname. Záznam, ktorý sa skladá z RGB snímky a hĺbkovej mapy pochádza zo zariadenia Kinect. Cieľom práce je navrhnúť a implementovať algoritmus, ktorý rozhodne o tom, či bol z regálu vybraný, prípadne či doň bol vložený tovar. V teoretickej časti práce sú zanalyzované algoritmy na získanie príznakov z obrázkov a základné techniky spracovania obrazu. Práca obsahuje aj analýzu vhodných algoritmov na klasifikáciu. Navrhnutý algoritmus využíva Gradient boosted trees klasifikátor, ktorý hodnotí snímky podľa 60 príznakov. Použitím algoritmu SIFT je možné zistiť, v akom smere sa ruka pohybuje. Na základe týchto dvoch informácii je vytvorená postupnosť, ktorá opisuje pohyb ruky. Z nej je získaných deväť príznakov, ktorými sa naučí ďalší klasifikačný model a tak rozhodne, či bol tovar vložený alebo vybraný. Tento postup dosiahol presnosť 71%. Algoritmus je implementovaný v jazyku C++ s využitím knižnice OpenCV.

Kľúčové slová detekcia ruky, spracovanie obrazu, segmentácia obrazu, SIFT, strojové učenie, hĺbková analýza dát

Abstract

This Bachelor Thesis deals with the detection of goods in the customer's hand displayed on the camera system. The record composed of the RGB frame and the depth map was produced by the Kinect device. The aim of this Thesis is to design, then to implement the algorithm that would be able to recognize whether the goods are taken out or put on the shelf. The theoretical part encompasses the analyzed algorithms for purposes of the attributes extracting and the basic techniques of the image processing. The Thesis covers the analysis of the proper and applicable algorithms for the classification, as well. Designed algorithm works with Gradient boosted trees classifier which classifies the frames according to 60 chosen attributes. By means of SIFT algorithm it is possible to find out in what direction the hand moves. Providing these two pieces of information, we managed to generate the sequences describing the hand's movement. Out of the sequence nine attributes were extracted which were used for classifier training. Onward, the classifier was able to decide whether the goods had been taken or put on the shelf. The accuracy of this approach succeeded in reaching 71%. The algorithm was implemented in the C++ language and employed the OpenCV library, as well.

Keywords hand detection, image processing, image segmentation, SIFT, machine learning, data mining

Obsah

Úvod **1**

1 Analýza **3**
1.1 Súčasné riešenie . . . 3
1.2 Farebné modely . . . 4
1.3 Hĺbková mapa . . . 5
1.4 Nájdenie kontúry v hĺbkovej mape . . . 5
1.5 Konvexná obálka . . . 6
1.6 Transformácie a deskriptory . . . 6
1.7 OpenCV . . . 8

2 Klasifikácia **9**
2.1 Definícia . . . 9
2.2 Klasifikačné modely . . . 10
2.3 Vyhodnotenie výkonnosti klasifikátora . . . 16

3 Návrh **19**
3.1 Proces snímania . . . 19
3.2 Predspracovanie . . . 19
3.3 Príznaky . . . 22
3.4 Klasifikácia . . . 26
3.5 Detekcia pohybu . . . 27
3.6 Sledovanie ruky na zázname . . . 28
3.7 Postupnosť tried . . . 28
3.8 Vyhodnotenie postupností . . . 29

4 Implementácia **31**
4.1 Trieda `Evaluate` . . . 31
4.2 Trieda `CSequences` . . . 31
4.3 Trieda `CSeqeuence` . . . 31

4.4 Trieda `motionSIFT` 32
4.5 Trieda `CImage` 32
4.6 Trieda `CHand` 32

5 Výsledky **33**
5.1 Klasifikácia snímok podľa časti ruky 33
5.2 Detekcia pohybu 37
5.3 Sledovanie ruky 37
5.4 Klasifikácia postupností 37
5.5 Celková presnosť 39

Záver **41**

Literatúra **43**

A Algoritmus SIFT **47**
A.1 Detekcia lokálnych extrémov v scale-space 48
A.2 Lokalizácia kľúčových bodov 51
A.3 Priradenie orientácie 52
A.4 Vytvorenie deskriptora 53
A.5 Párovanie kľúčových bodov 53

B Zoznam použitých skratiek **55**

C Obsah priloženého CD **57**

Zoznam obrázkov

1.1 Výpočet Local Binary Patterns 8

2.1 Rozhodovací strom 11
2.2 Lineárny model 12
2.3 K-najbližších susedov 14

3.1 Schéma umiestnenia kamery 20
3.2 Fotografia umiestnenia kamery 21
3.3 Ukážkové snímky z kamery 22
3.4 Vyextrahované ruky 22
3.5 Priechod maticou metódou ZigZag 24
3.6 Ukážky jednotlivých tried 27

5.1 Dôležitosť príznakov snímok rúk podľa χ^2 testu 36
5.2 Dôležitosť príznakov postupností podľa χ^2 testu 38

A.1 Rekonštrukcia scény z čiastkových obrázkov, ktoré boli zhotovené z rôznych uhlov a vzdialeností 47
A.2 Vytvorenie scale space 49
A.3 Ukážka scale space s piatimi oktávami 49
A.4 Ukážka rozdielov gaussiánov 50
A.5 Hľadanie lokálnych extrémov 50
A.6 Vyradenie bodov s nízkym kontrastom a bodov na hranách 52
A.7 Ukážka tvorby deskriptora 54

Zoznam tabuliek

2.1 Kontigenčná tabuľka 16

3.1 Zoznam príznakov snímok rúk 24

5.1 Dôležitosť príznakov snímok rúk podľa χ^2 testu 33
5.2 Počet snímok v jednotlivých triedach pre klasifikáciu rúk 35
5.3 Celková klasifikačná presnosť a presnosti jednotlivých tried pri použití rôznych klasifikátorov pre klasifikáciu rúk. 36
5.4 Dôležitosť príznakov postupností podľa χ^2 testu 38
5.5 Celková klasifikačná presnosť a presnosti jednotlivých tried pri použití rôznych klasifikátorov pre klasifikáciu postupností. 39
5.6 Kontigenčná tabuľka klasifikátora postupností 39
5.7 Výkonnostné ukazovatele klasifikátora postupností 39
5.8 Kontigenčná tabuľka vyhodnotenia algoritmu 40

Úvod

Skúmanie správania sa zákazníka v predajni sa postupne stáva náplňou práce merchandisingových špecialistov viacerých obchodných reťazcov. Aj na základe týchto zistení sa určuje umiestnenie tovaru v predajni. "Správne miesto na vystavenie tovaru ovplyvňuje zákazníkov, zvyšuje dostupnosť a postavenie a v konečnom dôsledku zvyšuje tržby a zisk z predaja." [1] Dôležité je umiestnenie nielen na úrovni predajnej plochy, ale za čoraz viac dôležitejšie sa považuje aj umiestnenie v samotnom regáli. Či už hovoríme o výške umiestnenia – predpokladáme, že produkty umiestnené vo výške očí si zákazník všimne skôr ako niečo, čo sa nachádza pri nohách, prípadne veľmi vysoko – alebo o rozložení sortimentu v horizontálnom smere. Prínosom v tejto oblasti by mohla byť aj táto bakalárska práca. Myslíme si, že pomôže urýchliť proces skúmania vplyvu umiestnenia tovaru a tak rýchlejšie reagovať na požiadavky zákazníkov. To pomôže nielen predajcom, ale v konečnom dôsledku aj samotným zákazníkom.

V práci sa zaoberáme návrhom algoritmu na detekciu tovaru v ruke zákazníka na kamerovom zázname. Záznam je zhotovený zariadením Kinect, ktoré zhotovuje farebné snímky aj hĺbkovú mapu a je umiestnené nad regálom tak, že sníma pohľad na ruky zákazníka zhora. K dispozícií je záznam úzkeho pásu pozdĺž regálu. Úlohou je rozhodnúť, či zákazník vložil alebo vybral tovar z regálu. Práca je zameraná hlavne na návrh a implementáciu algoritmov, nevenuje sa používateľskému rozhraniu aplikácie.

K problému budeme pristupovať dvojfázovo. V prvej fáze sa budú klasifikovať jednotlivé snímky podľa toho, aká časť ruky sa na nich nachádza a či v nej je držaný tovar. Z týchto snímok sa potom vytvorí postupnosť, ktorá opisuje pohyb ruky z a do regálu. Použitím algoritmu SIFT zistíme rýchlosť pohybu ruky a tak sa postupnosť rozdelí na dve časti – pohyb smerom dnu a pohyb smerom von z regálu. V druhej fáze sa budú klasifikovať tieto postupnosti do dvoch tried – ruka s tovarom a prázdna ruka.

V prvej časti práce sa venujeme analýze algoritmov na spracovanie obrazu a získavanie príznakov a deskriptorov vhodných na klasifikáciu snímok. V dru-

hej časti je predstavený problém klasifikácie a jednotlivé algoritmy vhodné na riešenie tohto problému.

V ďalšej časti práce je navrhnuté moje vlastné riešenie. Neskôr je opísaná implementácia predstaveného návrhu a dosiahnuté výsledky po použití navrhnutého postupu.

Kapitola 1

Analýza

V tejto kapitole predstavíme a opíšeme algoritmy a metódy spracovania obrazu vhodné na získanie príznakov z farebnej snímky a hĺbkovej mapy. Táto kapitola je výsledkom literárnej rešerše.

1.1 Súčasné riešenie

Rovnakým problémom sa podľa verejne dostupných informácii nikto nezaoberal. Existuje však mnoho algoritmov, ktoré sa zaoberajú problémom detekcie ruky či už na obrázku alebo na kamerovom zázname. Väčšina algoritmov však očakáva, že ruka bude jasne viditeľná a následne sa venujú hlavne jej sledovaniu a získavania gest. Existujú však aj algoritmy, ktoré detekujú ruku aj v zložitom pozadí nie v úplne triviálnej podobe. Nerozlišujú však v tom, či ruka drží objekt, čo je pre náš problém kľúčové.

Článok *Hand detection using multiple proposals* [2] od Arpita Mittala sa zaoberá detekciou a lokalizáciou ruky na obrázku. Pomocou troch nezávislých metód sa navrhnú objekty, ktoré sú považované za ruku. Prvá metóda rozlišuje na základe tvaru, druha podľa okolia (napríklad nájde zápästie) a tretia hľadá podľa farby pokožky, ktorú získa detekovaním tváre. Každý z týchto návrhov je ohodnotený podľa všetkých troch kritérií a SVM klasifikátorom overený. Prekrývajúce sa oblasti sú odstránené upravenou NMS technikou, ktorá využíva aj informáciu z obrázku. Za prekrývajúce sa oblasti považuje tie, ktoré prekrývajú ten istý super-pixel.Tento algoritmus dokáže rozpoznať ruku v akejkoľvek polohe a nemusí ju byť vidno celú.

Ďalší algoritmus, ktorý je popísaný v práci *Hand Detection and Tracking Using Depth and Color Information* [3] je zameraný nielen na detekciu ruky, ale hlavne na jej sledovanie. Okrem RGB obrázku využíva na nájdenie ruky aj hĺbkovu mapu získanú z Kinectu. Na základe histogramu hĺbky sa navrhnú oblasti, ktoré by mohli byť rukou. Predpokladá sa, že ruka sa nachádza pred telom a je pomerne menšia. Následne sa podľa tvaru a farby ohodnotia jednotlivé oblasti a tá s najvyšším hodnotením sa označí za ruku. Na ohodnotenie farby

pokožky sa používa Bayesov klasifikátor. Algoritmus dokáže lokalizovať len ruku, ktorá je jasne viditeľná a má otvorenú dlaň.

V článku *A real-time hand detection system based on multi-feature* [4] je navrhnutý spôsob na detekciu ruky v reálnom čase, ktorý dosahuje vysokú rýchlosť a úspešnosť. Je založený na AdaBoost klasifikátore. Tento postupne zahadzuje vzorky a do ďalšej fázy vždy postupujú len tie, ktoré prešli predchádzajúcim kolom. Pre zlepšenie charakteristiky objektu bola ruka rozdelená do dvoch častí – dlaň a prsty. Na zlepšenie detekcie prstov sa využíva histogram orientácií gradientov. Na lepšiu detekciu dlane bola predstavená nová vlastnosť, a to rozptyl. Využíva sa zrýchlený výpočet, kde stačí každý obrázok prejsť iba raz. Posledným typom vlastností získaným z obrázku sú Haar vlastnosti.

Článok *Real time hand detection in a complex background* [5] sa zaoberá detekciou a segmentáciou ruky pred zložitým pozadím. Predstavuje nový klasifikátor založený na farbe pokožky. Pomocou rozdielov medzi snímkami detekuje pohyb a označí kandidáta na oblasť s rukou. Základom je známy robustný klasifikátor Skin Probability Map, čo je vlastne Bayesov klasifikátor založený na histograme farby pokožky. Následne sa prevedie morfologická analýza, v ktorej sa vypočítajú vzdialenosti čiernych pixelov od kontúry. Všetky tieto informácie sa skombinujú a využijú pri rozhodovaní.

1.2 Farebné modely

1.2.1 RGB farebný model

RGB (Red, Green, Blue) farebný model má svoj pôvod vo farebnej televízii [7]. Hodnota farby je vyjadrená vektorom troch zložiek – intenzity troch primárnych farieb. Tými sú, ako napovedá už názov, červená, zelená a modrá. Tento model je aditívny, to znamená, že jednotlivé zložky sa pridávajú a zmiešavajú. Ak sú všetky hodnoty 0, ide o čiernu farbu a naopak, ak sú všetky zložky najvyššej hodnoty, tak ide o bielu farbu.

1.2.2 HSV farebný model

HSV (Hue, Saturation, Value), známy aj ako HSB (Hue, Saturation, Brightness) je podobný ľudskému vnímaniu farieb.

- **Hue** – farebný tón alebo odtieň. Vyjadruje sa v stupňoch ako poloha na farebnom kruhu.
- **Saturation** – sýtosť farby, prímes inej farby. Predstavuje množstvo šedej v pomere k odtieňu.
- **Value** – hodnota jasu, vyjadruje množstvo bieleho svetla.

Funguje podobne ako miešanie farieb maliarmi, ktorí si vyberú odtieň a primiešaním inej farby zmenia sýtosť [7].

1.2.3 Prevod z RGB do HSV

Hodnoty R, G a B sú normalizované do intervalu [0,1]. Potom sa hodnoty v kanáloch H, S a V vypočítajú nasledujúcim spôsobom [8]:

$$V \leftarrow \max(R, G, B)$$

$$S \leftarrow \begin{cases} \frac{V - \min(R,G,B)}{V} & \text{if } V \neq 0 \\ 0 & \text{inak} \end{cases}$$

$$H \leftarrow \begin{cases} 60(G-B)/(V - \min(R,G,B)) & \text{if } V = R \\ 120 + 60(B-R)/(V - \min(R,G,B)) & \text{if } V = G \\ 240 + 60(R-G)/(V - \min(R,G,B)) & \text{if } V = B \end{cases}$$

Ak je $H < 360$, tak sa k nemu pripočíta 360.

1.3 HĺbkovÁ mapa

Je to obrázok v odtieňoch šedej. Obsahuje informáciu o vzdialenosti snímaného objektu od snímacieho zariadenia. Hodnota pixelu 0 znamená, že objekt je vzdialený najďalej a maximálna hodnota 255 znamená, že objekt je umiestnený najbližšie. Pixely hĺbkovej mapy vlastne definujú súradnicu na Z-osi [9].

1.4 Nájdenie kontúry v hĺbkovej mape

Algoritmus na nájdenie kontúry v binárnom obrázku opísaný v článku *Topological Structural Analysis of Digitized Binary Images by Border Following* [10] rozdeľuje obrázok na komponenty podľa hodnôt pixelov. V jednotlivých komponentoch rozlišuje vonkajšie a vnútorné hranice.

Jednotlivé pixely sú prehľadávané z ľavého horného rohu a algoritmus končí v pravom dolnom rohu obrázka. Ak sa nájde bod, ktorý spĺňa podmienku pre začiatočný hraničný bod, priradí mu unikátny identifikátor NBD a zmení podľa neho hodnotu pixelu. Pixely, ktoré sú pravým vonkajším okrajom sa nastavia na hodnotu -NBD. Takto získame hranice všetkých komponentov. Kontúru nájdeme ako prvú vonkajšiu hranicu.

Obsah kontúry Obsah kontúry sa vypočíta použitím Greenovej vety [11], pomocou ktorej vieme vypočítať obsah množiny v rovine. Zvolíme L a M, aby platilo

$$\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y} = 1.$$

Potom obsah množiny ohraničenej krivkou C je vyjadrený vzťahom

$$A = \oint_C (L\,dx + M\,dy).$$

1.5 Konvexná obálka

Konvexná obálka množiny bodov je najmenšia konvexná množina, ktorá tieto body obsahuje. Konvexná množina je taká množina, pre ktorú platí, že úsečka spájajúca ľubovoľné dva body tejto množiny, je celá obsiahnutá v tejto množine.

Na nájdenie konvexnej obálky slúži algoritmus predstavený v článku *Finding the Convex Hull of a Simple Polygon* [12] od Jacka Sklanskeho. V prvom kroku sú nájdené extrémne vrcholy v horizontálnom aj vertikálnom smere. Z týchto vrcholov vzniknú štyri priestory, pre ktoré je potrebné vytvoriť monotónnu krivku. Tá sa vytvára zahadzovaním bodov, ktoré nespĺňajú dané podmienky.

1.6 Transformácie a deskriptory

1.6.1 Diskrétna kosínova transformácia

Dopredná diskrétna kosínova transformácia je jednou z mnoha transformácií podobných diskrétnej Fourierovej transformácii. Transformácia matice X s veľkosťou M×N je definovaná ako [15]:

$$Y = C^{(\mathrm{N})} X (C^{\mathrm{N}})^{\mathrm{T}},$$

kde

$$C^{(\mathrm{N})}_{jk} = \sqrt{\alpha_j / \mathrm{N}} \cos\left(\frac{\pi\,(2k+1)\,j}{2\mathrm{N}}\right)$$
$$\alpha_0 = 1, \alpha_j = 2\mathrm{pre}j > 0.$$

1.6.2 Hu invarianty

Najprv sa vypočítajú priestorové momenty [13] z rastrového obrázka `img`:

$$\mathtt{m}_{ji} = \sum_{x,y} \left(\mathtt{img}(x,y) x^j y^i\right).$$

Potom sa vypočítajú centrálne momenty:

$$\mathtt{mu}_{ji} = \sum_{x,y} \left(\mathtt{img}(x,y) (x-\bar{x})^j (y-\bar{y})^i\right),$$

kde $(\bar{x}, \bar{y})$ je hmotný stred:

$$\bar{x} = \frac{\mathtt{m}_{10}}{\mathtt{m}_{00}}, \bar{y} = \frac{\mathtt{m}_{01}}{\mathtt{m}_{00}}.$$

Centrálne momenty sa následne normalizujú:

$$\mathtt{nu}_{ji} = \frac{\mathtt{mu}_{ji}}{\mathtt{m}_{00}^{(i+j)/2+1}}.$$

Je potrebné ešte dodefinovať niektoré hodnoty:

$$\mathtt{mu}_{00} = \mathtt{m}_{00}, \mathtt{nu}_{00} = 1$$

$$\mathtt{nu}_{10} = \mathtt{mu}_{10} = \mathtt{mu}_{01} = \mathtt{mu}_{10} = 0$$

Z centrálnych momentov sa vypočíta sedem Hu momentových invariantov, ktoré sú invariantné voči mierke a rotácii obrázka [14]:

$$\begin{aligned}
I_1 &= \mathtt{nu}_{20} + \mathtt{nu}_{02} \\
I_2 &= (\mathtt{nu}_{20} - \mathtt{nu}_{02})^2 + 4\mathtt{nu}_{11}^2 \\
I_3 &= (\mathtt{nu}_{30} - 3\mathtt{nu}_{12})^2 + (3\mathtt{nu}_{21} - \mathtt{nu}_{03})^2 \\
I_4 &= (\mathtt{nu}_{30} + \mathtt{nu}_{12})^2 + (\mathtt{nu}_{21} + \mathtt{nu}_{03})^2 \\
I_5 &= (\mathtt{nu}_{30} - 3\mathtt{nu}_{12})(\mathtt{nu}_{30} + \mathtt{nu}_{12})\left[(\mathtt{nu}_{30} + \mathtt{nu}_{12})^2 - 3(\mathtt{nu}_{21} + \mathtt{nu}_{03})^2\right] + \\
&\quad (3\mathtt{nu}_{21} - \mathtt{nu}_{03})(\mathtt{nu}_{21} + \mathtt{nu}_{03})\left[3(\mathtt{nu}_{30} + \mathtt{nu}_{12})^2 - (\mathtt{nu}_{21} + \mathtt{nu}_{03})^2\right] \\
I_6 &= (\mathtt{nu}_{20} - \mathtt{nu}_{02})\left[(\mathtt{nu}_{30} + \mathtt{nu}_{12})^2 - (\mathtt{nu}_{21} + \mathtt{nu}_{03})^2\right] + \\
&\quad 4\mathtt{nu}_{11}(\mathtt{nu}_{30} + \mathtt{nu}_{12})(\mathtt{nu}_{21} + \mathtt{nu}_{03}) \\
I_7 &= (3\mathtt{nu}_{21} - \mathtt{nu}_{03})(\mathtt{nu}_{30} + \mathtt{nu}_{12})\left[(\mathtt{nu}_{30} + \mathtt{nu}_{12})^2 - 3(\mathtt{nu}_{21} + \mathtt{nu}_{03})^2\right] - \\
&\quad (\mathtt{nu}_{30} - 3\mathtt{nu}_{12})(\mathtt{nu}_{21} + \mathtt{nu}_{03})\left[3(\mathtt{nu}_{30} + \mathtt{nu}_{12})^2 - (\mathtt{nu}_{21} + \mathtt{nu}_{03})^2\right].
\end{aligned}$$

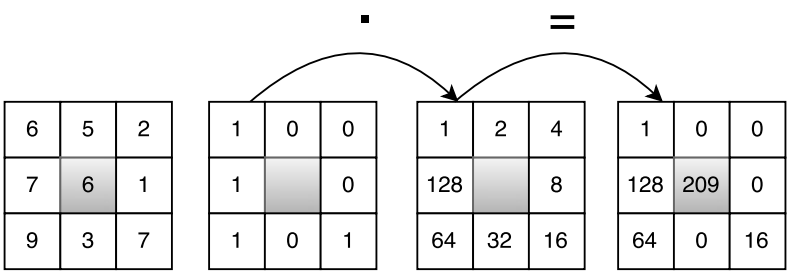
1.6.3 Local Binary Patterns

Táto metóda na získanie deskriptora obrázku bola prvýkrát opísaná v článku *Performance evaluation of texture measures with classification based on Kullback discrimination of distributions* [16]. Pre každý pixel je vypočítaná LBP hodnota, čoho výsledkom je obrázok rovnakej veľkosti.

Okolie skúmaného pixelu je prahované podľa jeho hodnoty (obrázok 1.1). Ak je hodnota centrálneho pixelu menšia alebo rovná ako hodnota pixelu z okolia, pixel z okolia sa nastaví na 1, inak na 0. Potom sú tieto hodnoty prenásobené mocninami dvojky. To sa sčíta a tak dostaneme LBP hodnotu centrálneho pixela.

1.6.4 FAST detektor

Metóda FAST (Features from accelerated segment test) slúži na nájdenie hrán na obrázku [17]. Pri skúmaní konkrétneho pixelu, či je alebo nie je hranou sa



Obr. 1.1: Výpočet Local Binary Patterns

vezme jeho 16 pixelové okolie. Ak sa v tomto okolí nachádza n susediacich pixelov, ktoré sú jasnejšie ako intenzita skúmaného pixelu zvýšená o prah t alebo tmavšie ako jeho intenzita znížená o prah t, potom je tento pixel považovaný za hranu.

1.6.5 SIFT

Algoritmus SIFT predstavený v práci *Distinctive Image Features from Scale-Invariant Keypoints* [18] slúži podobne ako metóda FAST na nájdenie tzv. kľúčových bodov v obrázku. Zo vstupného obrázku sú vytvorené tzv. Gaussiány, ktoré vzniknú konvolúciou Gaussovskej funkcie so vstupným obrázkom. Susedné Gaussiány sa od seba odčítajú, čím dostaneme rozdielové obrázky (Difference of Gaussians - DOG). V týchto obrázkoch nájdeme lokálne extrémy, ktoré sa stanú kandidátmi na kľúčové body. Následne sa odstránia body s nízkym kontrastom a body, ktoré sa nachádzajú na hrane. Bodom, ktoré neboli vyradené sa priradí orientácia vypočítaná podľa veľkostí a orientácií gradientov v okolí bodu. Podobne je získaný aj deskriptor, ktorý pozostáva zo 128 vlastností. Nájdené kľúčové body sú invariantné voči mierke, rotácii a čiastočne invariantné voči osvetleniu a polohe pozorovateľa. Algoritmus je podrobnejšie opísaný v prílohe A.

1.7 OpenCV

OpenCV je voľne dostupná knižnica vydaná pod licenciou BSD [6]. Je platformovo nezávislá a obsahuje rozhrania pre jazyky C, C++, Python a Java. Je navrhnutá pre výpočty v reálnom čase. Je napísaná v jazyku C/C++ a je paralelizovaná. Jej hlavným zameraním sú algoritmy počítačového videnia a spracovania obrazu. Tieto algoritmy a taktiež základné dátové štruktúry sú efektívne optimalizované. Obsahuje aj implementáciu algoritmov z oblasti strojového učenia. Celkovo ide až o viac ako 2500 optimalizovaných algoritmov. Je jednou z najpoužívanejších knižníc v tejto oblasti či už v komerčnom využití alebo vo výskume.

Kapitola 2

Klasifikácia

2.1 Definícia

Klasifikácia je najbežnejším problémom v odvetví strojového učenia [22]. Jeho cieľom je určiť triedu, do ktorej zaradíme nové pozorovanie na základe trénovacej množiny pozorovaní. Narozdiel od klastrovania, kde zaradenie vzoriek z trénovacej množiny nepoznáme a ich rozdelenie do tried je súčasťou tohto problému, pri klasifikácii je zaradenie vzoriek známe.

Samotné vzorky z množiny $\mathscr{X} = \mathscr{F}_1 \times \mathscr{F}_2 \times \cdots \times \mathscr{F}_d$ sú popísané množinou vlastností (features). Tieto vlastnosti môžu byť kategorické (napr. krvné skupiny "A", "B", "AB" alebo "0"), ordinálne (napr. "veľké", "stredné" alebo "malé"), celočíselne hodnoty alebo reálne čísla.

Matematicky môžeme klasifikáciu zapísať ako zobrazenie $\hat{c} : \mathscr{X} \to \mathscr{L}$, kde $\mathscr{L} = \{C_1, C_2, \ldots, C_k\}$ je konečná množina tried, ktorá je výstupom. Algoritmus, ktorý implementuje samotnú klasifikáciu sa nazýva klasifikátor.

2.1.1 Binárna klasifikácia

Najjednoduchší prípad klasifikácie je binárna klasifikácia, kde máme len dve triedy. Zvyčajne sa jedna trieda označuje ako pozitívna a druhá ako negatívna. Typickým príkladom binárnej klasifikácie je filtrovanie spamu. Spam berieme ako pozitívnu triedu (podobne ako pri chorobách), keďže sa ho snažíme identifikovať.

2.1.2 Viactriedna klasifikácia

V reálnom živote však je častejšia klasifikácia do viacerých tried. Existujú dva prístupy ku klasifikovaniu viacerých tried použitím binárnych klasifikátorov. Predstavme si, že máme k tried:

- *Jeden proti zvyšku (one-versus-rest)* – budeme mať k binárnych klasifikátorov, kde každý z nich oddelí jednu triedu od ostatných. Vzorky

triedy, ktorú oddeľujeme považujeme za pozitívne, zvyšné za negatívne.

- *Jeden proti jednému (one-versus-one)* – v tomto prípade natrénujeme $k(k-1)/2$ binárnych klasifikátorov. Každý z nich bude rozlišovať medzi dvojicou tried. Ak tento klasifikátor považuje triedy za asymetrické, použijeme pre každú dvojicu dva klasifikátory, čiže ich celkový počet bude $k(k-1)$.

2.2 Klasifikačné modely

2.2.1 Rozhodovacie stromy

Stromové modely sú najpopulárnejšími modelmi v odvetví strojového učenia. Stromy sú výrazné a jednoduché na pochopenie. Majú rekurzívnu povahu spôsobu *rozdeľuj a panuj (divide-and-conquer)*. Napríklad algoritmy v snímacom zariadení Kinect pre konzolu Xbox obsahujú rozhodovacie stromy, konkrétne náhodné lesy ako svoje jadro [22].

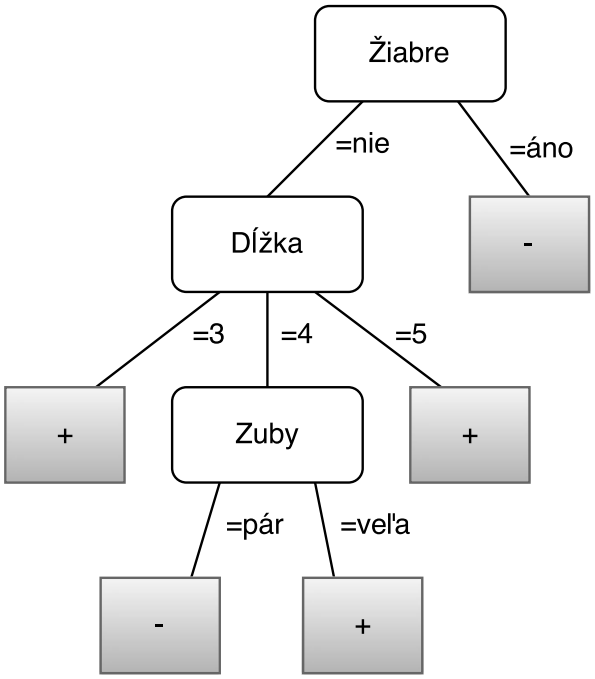
Štruktúra a tvorba Klasifikačný rozhodovací strom je prediktívny model, ktorého listy reprezentujú výstupné triedy (obrázok 2.1) a vetvy reprezentujú konjukciu vlastností, ktoré vedú k daným triedam. Každý vnútorný uzol odpovedá vstupnému atribútu, na základe ktorého sa vyberie hrana, po ktorej pokračuje vetva ku klasifikovanej triede.

Strom môže byť vytvorený procesom zhora-nadol. Učenie prebieha delením trénovacej množiny do podmnožín podľa atribútu, ktorý ju najlepšie rozdeľuje. Tento proces je rekurzívne opakovaný, pokiaľ nie sú v danej podmnožine inštancie iba jednej triedy. Tento spôsob je najrozšírenejším pri tvorbe modelu.

Metriky Výber najlepšieho atribútu je možný použitím rôznych metrík. Vypočíta sa hodnota pre každú podmnožinu a ich kombináciou sa určí miera kvality rozdelenia. Najčastejšie používané metriky sú:

- **Gini index** – vyjadruje očakávanú chybu, ak označíme triedy vzoriek v listoch náhodne: pozitívne s pravdepodobnosťou $\hat{p}$ a negatívne s pravdepodobnosťou $1-\hat{p}$. Pravdepodobnosť nesprávneho zaradenia pozitívnych je v tomto prípade $\hat{p}(1-\hat{p})$ a negatívnych $\hat{p}^3(1-\hat{p})$.

- **Informačný zisk** – porovnáva entropiu pred a po rozdelení. Vyjadruje, koľko informácii sme získali rozdelením podľa zvoleného atribútu. Entropia vyjadruje očakávanú informáciu v bitoch získanú tým, že nám niekto oznámi skutočnú triedu náhodnej vzorky. Čím je množina vzoriek jasnejšia, tým viac je správa predikovateľná a tým menšia je očakávaná získaná informácia.



Obr. 2.1: Rozhodovací strom

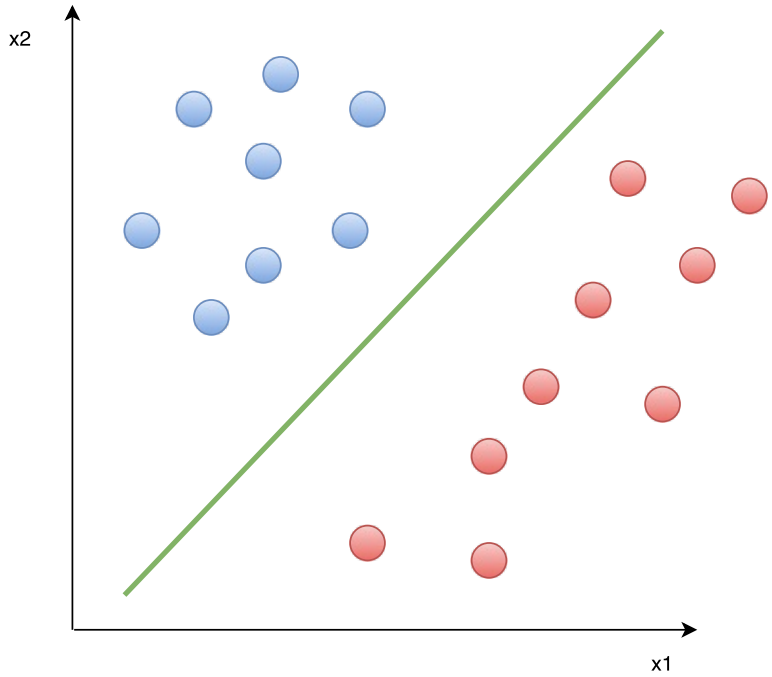
- **Redukcia rozptylu** – je definovaná ako celková redukcia rozptylu premennej x získaná rozdelením v uzle podľa tejto premennej.

- **Minoritná trieda** – niekedy je označovaná ako chybovosť, keďže meria podiel nesprávne klasifikovaných vzoriek, ak bol list označený podľa majoritnej triedy. Čím jasnejšia množina je, tým menšiu chybu vytvorí.

Regularizácia Rozhodovací strom nemôže popisovať trénovaciu množinu dokonale, pretože by nemal schopnosť generalizovať. Na zastavenie tvorby stromu sa používajú dve metódy [23]:

- **Zastavovacie pravidlo** – zastaví sa, ak neexistuje štatisticky významné rozdelenie. Existuje napríklad podmienka minimálneho počtu vzoriek v uzle alebo v liste, prípadne maximálna hĺbka stromu.

- **Prerezávanie** (pruning) – po vytvorení maximálneho stromu, ktorý vedie k preučeniu dôjde k prerezaniu. To spočíva v spätnom odstránení listov a vetiev, ktoré sa nepovažujú za významné. Väčšinou sa používa krížová validácia.



Obr. 2.2: Lineárny model

2.2.2 Lineárne modely

Lineárne modely sú definované v geometrickom priestore vzoriek. Predpokladá sa, že jednotlivé vzorky sú popísané d vektorom hodnôt, čiže vzorka $\mathscr{X} = \mathbb{R}^d$. Potom môžme každú vzorku zobraziť v d-dimenzionálnom priestore ako bod. Môžeme použiť geometrické koncepty ako sú priamka alebo rovina na predpísanie štruktúry tohto priestoru a vytvoriť z neho klasifikačný model.

Lineárne modely sú veľmi jednoduché. Ich jednoduchosť spočíva napríklad v týchto bodoch:

- Sú parametrické. To znamená, že majú fixnú formu, ktorá je závislá na malom počte parametrov, ktoré sú vypočítane naučením z trénovacích dát. Týmto sa líšia od stromových modelov, ktorých štruktúra je vždy rozdielna.

- Sú stabilné, čiže malé zmeny v dátach majú obmedzený dopad na naučený model. Stromové modely sú oveľa citlivejšie na takéto zmeny.

- Sú menej náchylné na preučenie (overfitting) v porovnaní s inými modelmi, pretože majú relatívne málo parametrov. Sú práveže viac náchylné na opačný jav (underfitting), čiže až príliš generalizujú.

Predpis lineárneho modelu, kde $\vec{x}$ je vektor atribútov a $\vec{w}$ vektor váh:

$$y = \vec{w}\vec{x} = \sum_j w_j x_j$$

Vektor $\vec{w}$ získame naučením z množiny trénovacích dát, čiže je to parameter lineárneho modelu. Výpočet výstupu je veľmi rýchly, takže sa využíva ak záleží na rýchlosti klasifikácie.

Support vector machine Support vector machine (SVM) je diskriminačný klasifikátor definovaný rozdeľujúcou nadrovinou. Pre každú množinu dát vieme nájsť nekonečne veľa rôznych nadrovín. Úlohou je nájsť tú správnu. Týmto sa zaoberá algoritmus SVM. Nájde nadrovinu, ktorá maximalizuje minimálnu vzdialenosť k vzorkám jednotlivých tried [24]. Dvojnásobku tejto vzdialenosti hovoríme *okraj*.

Nadrovina je definovaná ako

$$f(x) = \beta_0 + \beta^{\mathrm{T}} x$$

Problém nájdenia maximálneho okraju je ekvivalentný k problému minimalizovania funkcie $\mathrm{L}(\beta)$ s obmedzeniami. Formálne:

$$\min_{\beta,\beta_0} \mathrm{L}(\beta) = \frac{1}{2}\|\beta\|^2 \quad kde \quad y_i(\beta^{\mathrm{T}} x_i + \beta_0) \geq 1 \quad \forall i,$$

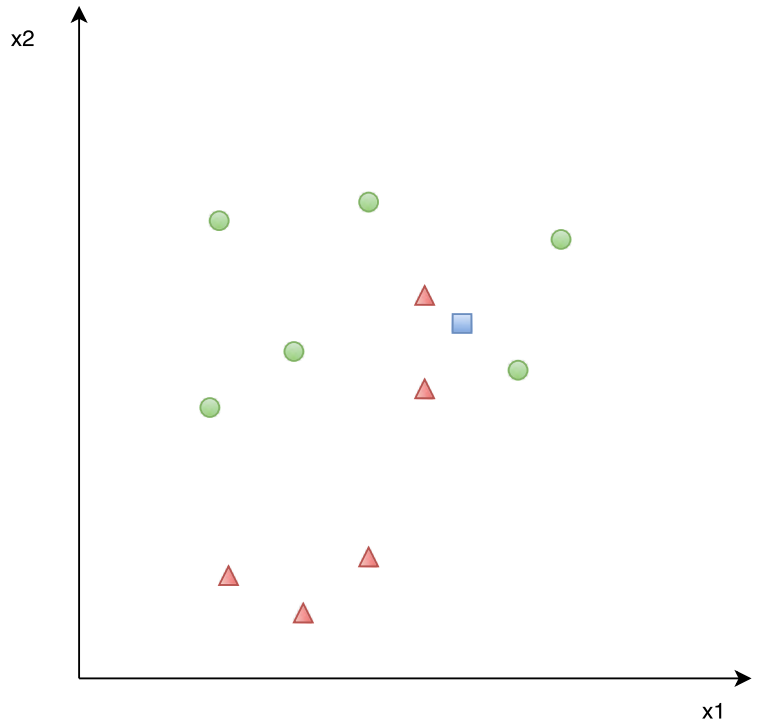
kde x_i predstavuje vektor atribútov a y_i predstavuje jeho triedu.

2.2.3 K-najbližších susedov

Algoritmus k-najbližších susedov (k-nearest neighbours) [25] je jedným z najjednoduchších algoritmov pre učenie s učiteľom. Myšlienkou algoritmu je nájsť najbližšie vzorky v priestore atribútov. Pre každú klasifikovanú vzorku nájdeme v priestore atribútov k-najbližších vzoriek. Novú vzorku zaradíme do triedy, ktorá medzi najbližšími susedmi prevažuje. V modifikovanej verzii môžu byť jednotlivým vzorkám priradené váhy na základe ich vzdialenosti. Pre tento model je vhodné dáta normalizovať na rovnakú škálu, aby neboli niektoré atribúty zvýhodňované.

Na určenie vzdialenosti medzi bodmi $\vec{a}$ a $\vec{b}$ môžu byť použité rôzne metriky [22]:

- **Euklidovská vzdialenosť:** $e\left(\vec{a}, \vec{b}\right) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$
- **Manhattanská vzdialenosť:** $\mathrm{M}\left(\vec{a}, \vec{b}\right) = \sum_{i=1}^{n} |a_i - b_i|$
- **Chebyshevova vzdialenosť:** $\mathrm{D}\left(\vec{a}, \vec{b}\right) = \max_i (a_i - b_i)$



Obr. 2.3: K-najbližších susedov. Zelene kruhy označujú jednu triedu a červené trojuholníky druhú. Modrý štvorec chceme klasifikovať. Pri nastavení parametra k ako 1 je najbližší sused červený trojuholník, takže ho zaradíme do tejto triedy. Rovnako aj pre k rovné 2. Od k väčšieho ako 3 bude nový bod zaradený do zelenej triedy.

- **Hammingova vzdialenosť:** $\mathrm{D}\left(\vec{a},\vec{b}\right) = \sum_{i=1}^{n}(a_i - b_i)^0$, pričom $0^0 = 1$

2.2.4 Bayesov klasifikátor

Na klasifikáciu využíva Bayesovu vetu. Bayesova veta udáva, ako súvisí podmienená pravdepodobnosť s opačnou podmienenou pravdepodobnosťou. Klasifikátor vypočíta podmienenú pravdepodobnosť, že sa daná vzorka vyskytuje v triede. Do triedy, pre ktorú je táto pravdepodobnosť najvyššia novú vzorku zaradíme. Predpokladá sa, že pre každú triedu sú pravdepodobnosti závislé na atribútoch rozdielne distribuované. Zjednodušená verzia – Naivný Bayesov klasifikátor – predpokladá, že jednotlivé atribúty sú nezávislé.

Rozhodovacie pravidlá [22], kde X je klasifikovaná vzorka a Y je množina tried:

- **Maximálna vierohodnosť:**

$$\arg\max_y P(X|Y=y)$$

- **Maximálna posteriorna pravdepodobnosť:**

$$\arg\max_y P(X|Y=y)/(Y=y)$$

Ak je rozdelenie tried rovnomerné, tak sú tieto pravidlá ekvivalentné.

2.2.5 Gradient boosted trees

Gradient boosted trees (GBT) [26] reprezentuje súbor regresných stromov zhotovených pažravým (greedy) spôsobom. Učenie je iteratívny proces podobný numerickej optimalizácii metódou klesajúceho gradientu. Celková strata na trénovacej množine závisí len na predikcii súčasného modelu, čiže

$$\sum_{i=1}^{N} \mathrm{L}\left(y_i, \mathrm{F}(x_i)\right) \equiv \mathcal{L}(\mathrm{F}(x_1), \mathrm{F}(x_2), \ldots, \mathrm{F}(x_N)) \equiv \mathcal{L}(\mathrm{F}).$$

Gradient $\mathcal{L}(\mathrm{F})$ sa vypočíta ako

$$\mathrm{grad}(\mathcal{L}(\mathrm{F})) = \left(\frac{\partial \mathrm{L}\left(y_1, \mathrm{F}(x_1)\right)}{\partial \mathrm{F}(x_1)}, \frac{\partial \mathrm{L}\left(y_2, \mathrm{F}(x_2)\right)}{\partial \mathrm{F}(x_2)}, \ldots, \frac{\partial \mathrm{L}\left(y_N, \mathrm{F}(x_N)\right)}{\partial \mathrm{F}(x_N)}\right)$$

V každom kroku učenia je vytvorený jeden strom. Schéma procesu učenia:

1. Nájde sa najlepší konštantný model.
2. Pre každé i z $[1, \mathcal{M}]$:
   a) Vypočíta sa antigradient.
   b) Rozšíri sa strom na predikovanie zložiek antigradientu.
   c) Upravia sa hodnoty v listoch stromu.
   d) Pridá sa strom do modelu.

Ako stratová funkcia L pre klasifikáciu sa používa strata odchýlky (deviance loss) alebo strata krížovej entropie (cross-entropy). Je vytvorených K funkcii pre každú výstupnú triedu.

$$\mathrm{L}\left(y, f_i(x), \ldots, f_K(x)\right) = -\sum_{k=0}^{K} \mathbb{1}(y=k) \ln \mathrm{p}_k(x),$$

kde $\mathrm{p}_k(x) = \frac{\exp f_k(x)}{\sum_{i=1}^{K} \exp f_i(x)}$ je odhad pravdepodobnosti, že $y = k$.

Tabuľka 2.1: Kontigenčná tabuľka slúžiaca na výpočet výkonnosti klasifikátora. Čísla na klesajúcej uhlopriečke zobrazujú správne predikcie, zatiaľ čo na rastúcej uhlopriečke znázorňujú klasifikačné chyby

Predikované \Skutočné	$\oplus$	$\ominus$	
$\oplus$	**30**	10	40
$\ominus$	20	**40**	60
	50	50	100

Výsledkom je nasledujúci model:

$$f(x) = f_0 + \nu \sum_{i=1}^{M} \mathrm{T}_i(x),$$

kde f_0 je počiatočný konštantný model a ν je regularizačný parameter *shrinkage* v intervale $(0, 1]$, ktorý určuje pomer učenia. Bolo empiricky zistené [27], že nastavenie $\nu < 1$ výrazne zvyšuje testovaciu úspešnosť. Ďalším regularizačným parametrom je *subsample_portion*, ktorý určuje podiel celkového množstva dát, ktorý sa použije na učenie modela. Množina je generovaná náhodne. Čím nižší podiel zvolíme, tým viac zabránime preučeniu. Výslednú triedu spočítame ako $\mathrm{argmax}_{i=1\ldots K}\,(f_i(x))$.

2.3 Vyhodnotenie výkonnosti klasifikátora

2.3.1 Kontigenčná tabuľka

Na ohodnotenie klasifikátora môžeme použiť kontigenčnú tabuľku. Riadky tejto tabuľky predstavujú skutočné triedy ako sú označené v testovacej množine a v stĺpcoch sa nachádzajú triedy predikované klasifikátorom. V tabuľke 2.1 je uvedených 50 pozitívnych vzoriek, z nich bolo 30 predikované správne a 20 bolo predikovaných nesprávne ako negatívne.

Správne klasifikované pozitívne, resp. negatívne vzorky nazývame *true positives (TP)*, resp. *true negatives (TN)*. Nesprávne klasifikované pozitívne sa nazývajú *false negatives (FN)* a podobne nesprávne klasifikované negatívne prípady *false positives (FP)*. Tieto štyri hodnoty obsahuje kontigenčná tabuľka. Túto terminológiu budeme používať pri výpočtoch jednotlivých výkonnostných ukazovateľov.

2.3.2 Výkonnostné ukazovatele

Z kontigenčnej tabuľky vieme vypočítať mnoho výkonnostných ukazovateľov.

Presnosť (accuracy) Je to najjednoduchší ukazovateľ. Vypočíta sa ako podiel správne klasifikovaných vzoriek k celkovému počtu:

$$\frac{TP + TN}{TP + FP + TN + FN}$$

Chybovosť (error rate) Je to doplnok k presnosti. Vypočíta sa ako podiel nesprávne klasifikovaných vzoriek k celkovému počtu:

$$\frac{FP + FN}{TP + FP + TN + FN}$$

Senzitivita (sensitivity, true positive rate) Vyjadruje presnosť pozitívnej triedy. Vypočíta sa ako podiel správne klasifikovaných pozitívnych vzoriek k celkovému počtu pozitívnych vzoriek:

$$\frac{TP}{TP + FN}$$

Špecifita (specificity, true negative rate) Vyjadruje presnosť negatívnej triedy. Vypočíta sa ako podiel správne klasifikovaných negatívnych vzoriek k celkovému počtu negatívnych vzoriek:

$$\frac{TN}{FP + TN}$$

Falošná negativita (false negative rate) Vyjadruje chybovosť pozitívnej triedy. Vypočíta sa ako podiel nesprávne klasifikovaných pozitívnych vzoriek k celkovému počtu pozitívnych vzoriek:

$$\frac{FN}{TP + FN}$$

Falošná pozitivita (false positive rate) Vyjadruje chybovosť negatívnej triedy. Vypočíta sa ako podiel nesprávne klasifikovaných negatívnych vzoriek k celkovému počtu negatívnych vzoriek:

$$\frac{FP}{FP + TN}$$

Vierohodnosť (precision, confidency) Vyjadruje pravdepodobnosť, že vzorka je naozaj pozitívna, ak ju tak klasifikátor zaradil. Vypočíta sa ako podiel správne klasifikovaných pozitívnych vzoriek k celkovému počtu pozitívne klasifikovaných vzoriek:

$$\frac{TP}{TP + FP}$$

Klasifikačná presnosť skresľuje úspešnosť klasifikátora pri nerovnomernom zastúpení tried. Napríklad, chceme rozlišovať neoprávnené platby platobnou kartou. 99% platieb je oprávnených a len 1% sú neoprávnené platby. Majoritný klasifikátor, ktorý by všetky platby označil ako oprávnené by mal resnosť 99%, ale v skutočnosti by nebol vôbec užitočný. Preto je v takýchto prípadoch lepšie počítať vierohodnosť namiesto presnosti.

F-miera (F-measure) Je to kompromis medzi senzitivitou a spoľahlivosťou. Môže byť interpretovaná ako ich vážený priemer. Vypočíta sa ako ich harmonický priemer:

$$\frac{2TP}{2TP + FP + FN}$$

Pri klasifikácii do viacerých tried využívame iba presnosť a jej doplnok, teda chybovosť. Tiež vieme spočítať presnosť a vierohodnosť konkrétnej triedy.

Kapitola 3

Návrh

3.1 Proces snímania

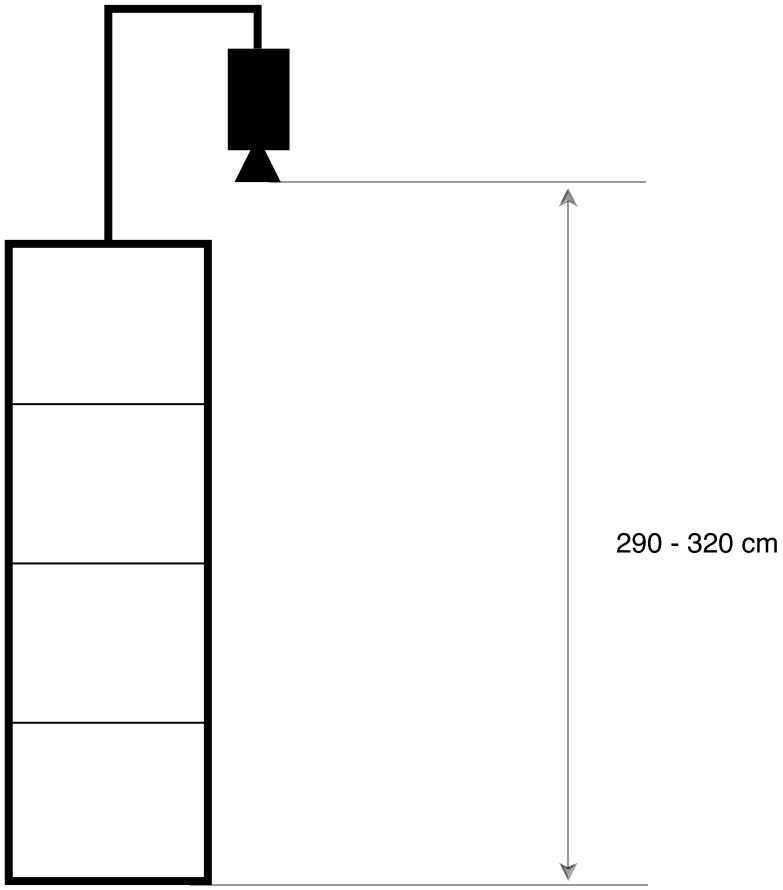
Dáta pochádzajú z hypermarketu Albert na Zličíne v Prahe. Meranie prebiehalo v období od 14. septembra do 11. októbra 2015. Ako kamera bolo použité zariadene Kinect, ktoré dokáže s použitím IR vysielača a senzora zhotoviť aj hĺbkovú mapu. Farebná snímka aj hĺbková mapa sa vyhotovujú v rozlíšení 640x20 pixelov. Senzor je umiestnený nad prednou stenou regálu vo výške 290 – 320 cm od zeme. Sníma úzky priestor od regálu smerom do uličky (obrázky 3.1 a 3.2). V ňom sa nachádzajú ruky zákazníkov, ktorí doň vkladajú alebo z neho vyberajú tovar.

Analyzujú sa iba dáta, ktoré obsahujú nenulový pixel v prvom rade od regála v hĺbkovej mape. To znamená, že je detekovaný objekt úplne pri regáli. Je to podmienka pre detekciu ruky zákazníka.

3.2 Predspracovanie

Procesom opísaným v predchádzajúcej sekcii získame série obrázkov, ktoré zodpovedajú jednotlivým situáciám vloženia a vybratia ruky z regálu. Ku každému farebnému obrázku zhotovenému RGB kamerou máme príslušnú hĺbkovú mapu. RGB obrázok aj hĺbková mapa sú v rozlíšení 640x20 pixelov. Následne sa v hĺbkovej mape vyhľadajú kontúry. Predtým je potrebné okolo hĺbkovej mapy vytvoriť čierny rám, pretože kontúry na kraji obrázku by neboli nájdené. Kontúry, ktoré neprechádzajú cez celú výšku obrázku, sú vyradené, pretože nemôžu byť rukou. Pre každú kontúru nájdeme najmenší ohraničujúci obdĺžnik, ktorého strany sú rovnobežné s osami snímky. Tento obdĺžnik umiestníme do stredu čierneho obrázku s veľkosťou 100x20 pixelov. Podľa týchto kontúr je orezaný aj RGB obrázok. Týmto od seba oddelíme jednotlivé objekty.

Na takto orezanom obrázku však ešte ostane v okolí objektu podlaha. Keďže tej zodpovedajú v hĺbkovej mape čierne pixely, tak ju môžeme odstrá-



Obr. 3.1: Schéma umiestnenia kamery

niť použitím bitovej masky. Vytvoríme RGB obrázok, ktorého pixely budú biele na tom mieste, kde sa nachádzajú nenulové pixely hĺbkovej mapy a ostatné budú čierne (vzťah 3.1). Tento obrázok použijeme ako bitovú masku na pôvodný farebný obrázok. Nakoniec dostaneme obrázok, na ktorom bude osamotený objekt a čierne okolie.

Na obrázkoch 3.3 sa nachádza RGB snímka s vykreslenými kontúrami a zodpovedajúca hĺbková mapa a na obrázkoch 3.4 vyextrahované jednotlivé ruky.

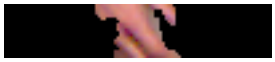
$$mask(x,y) = \begin{cases} (255,255,255) & \text{if } depthMap(x,y) > 0 \\ (0,0,0) & \text{inak} \end{cases} \tag{3.1}$$

Obr. 3.2: Fotografia umiestnenia kamery

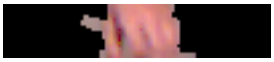
(a) Snímka z RGB kamery s vykreslenými kontúrami

(b) Hĺbková mapa

Obr. 3.3: Ukážkové snímky z kamery



(a) Ľavá ruka



(b) Pravá ruka

Obr. 3.4: Vyextrahované ruky z obrázku 3.3

3.3 Príznaky

Z takto orezaných snímkov následne vypočítame príznaky, ktoré budú použité na trénovanie klasifikátora. Triedy, do ktorých sa budú snímky klasifikovať budú spresnené v nasledujúcej časti. Kompletný zoznam príznakov je uvedený v tabuľke 3.1.

3.3.1 Farebná snímka

Kanály HSV farebného modelu Farebnú snímku skonvertujeme z farebného modelu RGB do modelu HSV algoritmom opísaným v sekcii 1.2.3. Tento algoritmus je implementovaný v knižnici OpenCV funkciou `cvtColor`. Vypočítame priemernú hodnotu, mediánovú hodnotu a smerodajnú odchýlku v každom kanáli. Berieme však do úvahy iba pixely s nenulovou hodnotou, pretože nechceme, aby boli tieto príznaky ovplyvnené veľkosťou ruky. Napríklad pri malej ruke by bola väčšia časť snímky čierna, čo by tieto hodnoty znížilo. Takto získame deväť príznakov.

Local Binary Patterns Vstupný obrázok prevedieme do odtieňov šedej a spočítame Local Binary Patterns. Dostaneme obrázok rovnakej veľkosti. Spravíme histogram hodnôt nového obrázku, ktoré rozdelíme do šestnásť rovnako veľkých intervalov. Tieto početnosti spriemerujeme a získame 16 príznakov.

Významné body Metódou FAST detekujeme rohy vo vstupnej snímke, ktorú sme si previedli do odtieňov šedej. Nájdene body zoradíme podľa dôležitosti a vyberiem prvých päť. Každý bod tvorí tri príznaky: x-ová súradnica, y-ová súradnica a dôležitosť bodu. Takto získame pätnásť príznakov. Ak nájdeme menej ako päť bodov, ďalšie príznaky budú mať hodnotu nula.

3.3.2 Kontúra

Kontúru nájdeme v hĺbkovej mape požitím algoritmu zo sekcie 1.4, ktorý je implementovaný funkciou `findContours`. Z nej následne vypočítame ďalších päť príznakov:

Obsah Vypočítame ho podľa Greenovej vety (sekcia 1.4) použitím funkcie `contourArea`.

Obvod Vypočítame ho ako súčet vzdialeností susedných bodov kontúry použitím funkcie `arcLength`.

Dĺžka konvexnej obálky Konvexnú obálku nájdeme algoritmom zo sekcie 1.5, ktorý je implementovaný vo funkcii `convexHull`. Jej dĺžku spočítame rovnako ako obvod kontúry.

Konvexnosť Vypočítame ju ako pomer dĺžky konvexnej obálky a obvodu kontúry.

Pomer strán opísaného obdĺžnika Nájdeme najmenší opísaný obdĺžnik. Pomer strán spočítame ako pomer kratšej strany obdĺžnika k dlhšej. Získame reálne číslo z intervalu (0,1).

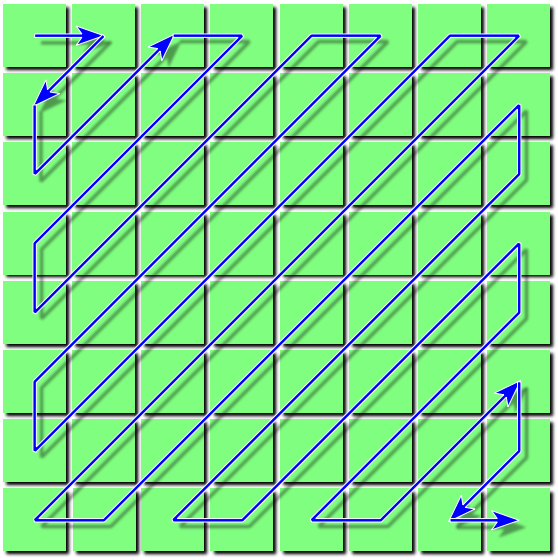
3.3.3 Hĺbková mapa

Hu invarianty V OpenCV je implementovaná funkcia `moments`, ktorá vypočíta momenty z hĺbkovej mapy ako sú uvedené v sekcii 1.6.2. Potom funkciou `HuMoments` vypočítame sedem Hu invariantov, ktoré použijeme ako ďalšie príznaky.

Diskrétna kosínova transformácia Prevedieme kosínovu transformáciu hĺbkovej mapy, po ktorej dostaneme maticu rovnakej veľkosti. Priechodom matice metódou ZigZag (obrázok 3.5) získame prvých šesť koeficientov. V tomto prípade to budú koeficienty na pozíciach: (0,0), (0,1), (1,0), (2,0), (1,1) a (0,2). Týmto získame koeficienty, ktoré obsahujú informáciu o nižších frekvenciách.

Smerodajná odchýlka výšok Smerodajná odchýlka nenulových hodnôt hĺbkovej mapy.

Rozpätie výšok Rovná sa rozdielu medzi najväčšou a najmenšou nenulovou hodnotou hĺbkovej mapy.



Obr. 3.5: Priechod maticou metódou ZigZag

Tabuľka 3.1: Zoznam príznakov snímok rúk

Č.	**Príznak**	**Typ**
1.	Priemerná hodnota kanálu H (farebný model HSV)	reálny
2.	Mediánová hodnota kanálu H (farebný model HSV)	reálny
3.	Smerodajná odchýlka kanálu H (farebný model HSV)	reálny
4.	Priemerná hodnota kanálu S (farebný model HSV)	reálny
5.	Mediánová hodnota kanálu S (farebný model HSV)	reálny
6.	Smerodajná odchýlka kanálu S (farebný model HSV)	reálny
7.	Priemerná hodnota kanálu V (farebný model HSV)	reálny
8.	Mediánová hodnota kanálu V (farebný model HSV)	reálny
9.	Smerodajná odchýlka kanálu V (farebný model HSV)	reálny
10.	Obsah kontúry	reálny
11.	Obvod kontúry	reálny
12.	Dĺžka konvexnej obálky	reálny
13.	Konvexnosť	reálny
14.	Pomer strán opísaného obdĺžnika	reálny
15.	Priemerná početnosť LBP v intervale [0,15]	reálny
16.	Priemerná početnosť LBP v intervale [16,31]	reálny
17.	Priemerná početnosť LBP v intervale [32,47]	reálny
18.	Priemerná početnosť LBP v intervale [48,63]	reálny
19.	Priemerná početnosť LBP v intervale [64,79]	reálny
20.	Priemerná početnosť LBP v intervale [80,95]	reálny

Pokračovanie na druhej strane

Tabuľka 3.1 – *Pokračovanie*

Č.	**Príznak**	**Typ**
21.	Priemerná početnosť LBP v intervale [96,111]	reálny
22.	Priemerná početnosť LBP v intervale [112,127]	reálny
23.	Priemerná početnosť LBP v intervale [128,143]	reálny
24.	Priemerná početnosť LBP v intervale [144,159]	reálny
25.	Priemerná početnosť LBP v intervale [160,175]	reálny
26.	Priemerná početnosť LBP v intervale [176,191]	reálny
27.	Priemerná početnosť LBP v intervale [192,207]	reálny
28.	Priemerná početnosť LBP v intervale [208,223]	reálny
29.	Priemerná početnosť LBP v intervale [224,239]	reálny
30.	Priemerná početnosť LBP v intervale [240,255]	reálny
31.	X-ová súradnica 1. najdôležitejšieho bodu podľa FAST	celočíselný
32.	Y-ová súradnica 1. najdôležitejšieho bodu podľa FAST	celočíselný
33.	Dôležitosť 1. najdôležitejšieho bodu podľa FAST	celočíselný
34.	X-ová súradnica 2. najdôležitejšieho bodu podľa FAST	celočíselný
35.	Y-ová súradnica 2. najdôležitejšieho bodu podľa FAST	celočíselný
36.	Dôležitosť 2. najdôležitejšieho bodu podľa FAST	celočíselný
37.	X-ová súradnica 3. najdôležitejšieho bodu podľa FAST	celočíselný
38.	Y-ová súradnica 3. najdôležitejšieho bodu podľa FAST	celočíselný
39.	Dôležitosť 3. najdôležitejšieho bodu podľa FAST	celočíselný
40.	X-ová súradnica 4. najdôležitejšieho bodu podľa FAST	celočíselný
41.	Y-ová súradnica 4. najdôležitejšieho bodu podľa FAST	celočíselný
42.	Dôležitosť 4. najdôležitejšieho bodu podľa FAST	celočíselný
43.	X-ová súradnica 5. najdôležitejšieho bodu podľa FAST	celočíselný
44.	Y-ová súradnica 5. najdôležitejšieho bodu podľa FAST	celočíselný
45.	Dôležitosť 5. najdôležitejšieho bodu podľa FAST	celočíselný
46.	Hu invariant 0	reálny
47.	Hu invariant 1	reálny
48.	Hu invariant 2	reálny
49.	Hu invariant 3	reálny
50.	Hu invariant 4	reálny
51.	Hu invariant 5	reálny
52.	Hu invariant 6	reálny
53.	1. koeficient DCT metódou ZigZag	reálny
54.	2. koeficient DCT metódou ZigZag	reálny
55.	3. koeficient DCT metódou ZigZag	reálny
56.	4. koeficient DCT metódou ZigZag	reálny
57.	5. koeficient DCT metódou ZigZag	reálny
58.	6. koeficient DCT metódou ZigZag	reálny
59.	Smerodajná odchýlka hĺbok v hĺbkovej mape	reálny
60.	Rozpätie hĺbok v hĺbkovej mape (max − min)	celočíselný

3.4 Klasifikácia

Samotnú detekciu tovaru v ruke vykonáme klasifikovaním snímky vopred natrénovaným klasifikátorom.

3.4.1 Klasifikačné triedy

Na natrénovanie klasifikátora si potrebujeme určiť triedy, do ktorých sa budu snímky radiť a jasne medzi nimi vymedziť hranice.

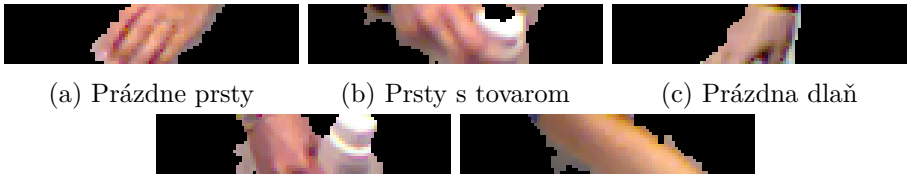
Na začiatku sme navrhli týchto osem tried:

1. Prázdne končeky prstov – menšia časť prstov, v ktorých sa nenachádza tovar
2. Končeky prstov s tovarom – menšia časť prstov, v ktorých sa nachádza tovar
3. Prázdne prsty – väčšia časť prstov, v ktorých sa nenachádza tovar
4. Prsty s tovarom – väčšia časť prstov, v ktorých sa nachádza tovar
5. Prázdna dlaň – dlaň (medzi prstami a zápästím), v ktorej sa nenachádza tovar
6. Dlaň s tovarom – dlaň, v ktorej sa nachádza tovar
7. Zápästie
8. Predlaktie – časť ruky od zápästia smerom k lakťu

Keďže na detekciu ruky na obrázku je potrebné, aby prechádzala celou jeho výškou, nie je možné použiť tieto triedy. Pre obrázky zaradené do prvej triedy by preto nebolo možné spočítať príznaky. Ďalším problémom je trieda *zápästie*. Zápästie je veľmi malá časť ruky a je náročné zachytiť ju na jednej snímke. Taktiež jej hranica medzi dlaňou a medzi predlaktím nie je výrazná. Preto sme sa rozhodli vyradiť prvé dve triedy a triedu *zápästie*.

Nové rozdelenie tried:

1. Prázdne prsty – prsty, v ktorých sa nenachádza tovar
2. Prsty s tovarom – prsty, v ktorých sa nachádza tovar
3. Prázdna dlaň – dlaň (medzi prstami a zápästím, vrátane zápästia), v ktorej sa nenachádza tovar
4. Dlaň s tovarom – dlaň, v ktorej sa nachádza tovar
5. Predlaktie – časť ruky od zápästia smerom k lakťu, vrátane zápästia



(a) Prázdne prsty (b) Prsty s tovarom (c) Prázdna dlaň

(d) Dlaň s tovarom (e) Predlaktie

Obr. 3.6: Ukážky jednotlivých tried

3.4.2 Učenie klasiifkátora

Na natrénovanie klasifikátora, ktorý použijeme na klasifikáciu jednotlivých častí ruky potrebujeme dostatočné množstvo označených obrázkov a k ním vypočítané príznaky z predchádzajúcej časti práce.

Ak snímka zachytáva viac častí ruky, anotujeme ju podľa prevažujúcej časti. Klasifikátor netrénujeme na veľmi podobných snímkach. Ak sa na viacerých po sebe idúcich obrázkoch veľmi nemení poloha ruky a nie je na nich zreteľná zmena, tak do trénovacej množiny zaradíme iba jeden z týchto snímkov.

3.4.3 Klasifikačné modely

Pre porovnanie skúsime viac typov klasifikátorov. Navrhujeme metódu k-najbližších susedov, kde zvolíme hodnotu parametra k v intervale [1,10], Bayesov klasifikátor, SVM, Rozhodovacie stromy, GBT a MLP neurónovú sieť. Pre každý vypočítame klasifikačnú úspešnosť a presnosti jednotlivých tried. Použijeme klasifikátor s najvyššou presnosťou.

3.5 Detekcia pohybu

Každú situáciu vybrania a vloženia ruky z regálu si potrebujeme rozdeliť na tri časti:

1. Ruka vchádza do regálu
2. Ruka sa nachádza v regáli
3. Ruka vychádza z regálu

V prvom a treťom prípade ruka vykonáva nejaký pohyb smerom od alebo k regálu, ale ak sa v ňom nachádza, tento pohyb je minimálny. Tento fakt vieme využiť na oddelenie týchto situácii.

3.5.1 SIFT

Použijeme algoritmus SIFT, ktorý nám v snímkach nájde významné body. Oblasť, v ktorej ich má hľadať obmedzíme vymaskovaním hĺbkovou mapu, čiže len na miesta, kde sa nachádza nejaký objekt. Tieto body nájdeme v dvoch po sebe nasledujúcich snímkach a potom ich spárujeme.

Na spárovanie sa používa algoritmus, ktorý pre každý bod z prvej snímky nájde bod z druhej snímky, ktorého deksriptor je mu najbližší [28]. Tento vzťah musí platiť aj opačne. Inak sa body nespárujú. Keďže ani po krížovej kontrole nemusia byť všetky body spárované správne, je potrebný ich výber. Nájdeme si pár, u ktorého bola vzdialenosť deskriptorov najmenšia (pravdepodobnosť zhody je najvyššia) a vyradíme všetky páry, ktorých vzdialenosť zhody je väčšia ako dvojnásobok najmenšej.

3.5.2 Vyhodnotenie pohybu

Z dvojíc bodov, ktoré nám ostali, vypočítame vzdialenosť posunu po y-ovej osi. Z týchto vzdialeností vypočítame medián, ktorý považujeme za výsledný posun ruky v smer od alebo k regálu. Ak na dvoch po sebe nasledujúcich snímkach vyhodnotíme pohyb oproti predchádzajúcej snímke menší ako dva, považujeme to za ruku, ktorá sa nehýbe a predpokladáme, že sa nachádza dnu v regáli.

Takýmto spôsobom si nájdeme čas, kedy sa ruka prestane pohybovať a takisto kedy sa opäť začne pohybovať. Tieto časové údaje neskôr využijeme pri rozdelení postupností snímok.

3.6 Sledovanie ruky na zázname

Aby sme ruku na zázname nestratili a aby sme vedeli oddeliť viac objektov na zázname, je potrebné sledovať ju. Predpokladáme, že ruka medzi jednotlivými snímkami spraví menší posun po x-ovej osy ako je polovica jej šírky. Preto nám stačí skontrolovať, či sa bod, ktorý bol stredom ruky z prechádzajúcej snímky nachádza v ohraničujúcom obdĺžniku ruky na aktuálnej snímke.

Často sa stáva, že v hĺbkovej mape je chyba a ruka neprechádza cez celú výšku obrázku, preto stále počkáme, či sa neobjaví na ďalšom. Ak na dvoch po sebe idúcich snímkach stratíme sledovanú ruku, vyhodnotíme to tak, že ruka sa na obrázku už nenachádza. Niekedy sa stane, že chyba v hĺbkovej mape je aj na viacerých snímkach, vtedy to už považujeme za ďalšiu ruku a teda za novú situáciu.

3.7 Postupnosť tried

Z orezaných snímok, na ktorých sa nachádzajú samotné objekty, teda ruky, vypočítame príznaky. Podľa týchto príznakov klasifikátor zaradí snímku do jed-

nej z klasifikačných tried.

Z klasifikovaných tried pre tú istú ruku vytvárame postupnosť. Táto postupnosť obsahuje triedy, do ktorých boli klasifikované snímky z celej situácie, odkedy ruka vošla do regálu, pokiaľ ho neopustila. Túto postupnosť rozdelíme na tri už spomínané časti podľa časových údajov, kedy ruka menila smer pohybu.

Dôležité pre ďalší postup sú však len podpostupnosti, ktoré popisujú časť, kedy ruka vchádza dnu a časť kedy vychádza. Z týchto podpostupnosti následne určíme, či v nej bol držaný tovar v ruke.

Postupnosti obsahujú prirodzené čísla z intervalu $[1, 5]$, ktoré predstavujú triedy podľa už spomenutého značenia.

3.8 Vyhodnotenie postupností

Keďže kvôli klasifikačnej chybe klasifikátora nie sú všetky jednotlivé snímky zaradené správne, treba navrhnúť algoritmus, ktorý túto postupnosť vyhodnotí.

Nastať môžu štyri kombinácie:

- Prázdna ruka dnu – ruka s tovarom von
- Prázdna ruka dnu – prázdna ruka von
- Ruka s tovarom dnu – ruka s tovarom von
- Ruka s tovarom dnu – prázdna ruka von

3.8.1 Väčšinový algoritmus

Rozhodnutie, či je v danej postupnosti držaný tovar alebo nie, môžeme urobiť aj podľa väčšinovej triedy. Všetkých päť tried vieme rozdeliť na tri skupiny:

1. S tovarom - trieda č. 2 a č. 4
2. Bez tovaru - trieda č. 1 a č. 3
3. Bez informácie o tovare - trieda č. 5

Trieda č. 3 síce obsahuje informáciu, že v ruke sa nenachádza tovar, to však ešte nemusí byť pravda. Ak zákazník drží tovar len v prstoch, dostaneme napríklad postupnosť $\{2, 2, 3, 3, 3, 3\}$. Väčšinovo v nej je zastúpená trieda č. 3, čiže vyhodnotili by sme ju, že v ruke sa tovar nenachádza. Toto rozhodnutie je však chybné. Preto je potrebné nastaviť triede 3 menšiu váhu.

Stále však budú existovať prípady, kedy bude táto váha príliš vysoká alebo naopak príliš nízka. Niekedy je trieda č. 3 rozhodujúca a práve ona rozhoduje o výsledku.

Podobný problém je s triedou č. 5, ktorá síce neobsahuje informáciu o tovare, ale niekedy do nej môžu byť zaradené snímky prázdnej dlane alebo prstov. Preto ju treba brať do úvahy.

Zvolené nastavenie váh:

- Trieda č. 1 – prázdna ruka s váhou 1
- Trieda č. 2 – ruka s tovarom s váhou 1
- Trieda č. 3 – prázdna ruka s váhou 0,5
- Trieda č. 4 – ruka s tovarom s váhou 1
- Trieda č. 5 – prázdna ruka s váhou 0,7

3.8.2 Klasifikácia

Ďalším spôsobom ako postupnosť vyhodnotiť je použitie klasifikačného modelu na klasifikáciu týchto postupností do dvoch tried. Zvolené sú tieto triedy:

- Ruka s tovarom
- Prázdna ruka

Keďže postupnosti môžu mať rôznu dĺžku, je potrebné v nich nájsť nejaké vzory a informácie a opísať ich vypočítanými príznakmi. Navrhli sme použitie týchto príznakov:

1. Relatívna početnosť triedy 1.
2. Relatívna početnosť triedy 2.
3. Relatívna početnosť triedy 3.
4. Relatívna početnosť triedy 4.
5. Relatívna početnosť triedy 5.
6. Relatívna početnosť triedy 2 a 4, čiže tých tried, ktoré zahŕňajú držanie tovaru.
7. Pomer 6. príznaku k 1. Ak sa 1. príznak rovná nule, tak tento príznak bude 1. Ak sa obe rovnajú 0, tak tento príznak bude 0.
8. Najpočetnejšia hodnota.
9. Medián.

Je potrebné vytvoriť čo najväčšiu množinu dát na učenie klasifikátora, aby sme zachytili všetky časté situácie.

Kapitola 4

Implementácia

Návrh predstavený v predchádzajúcej kapitole bol implementovaný v jazyku C++ s využitím knižnice OpenCV. Hlavou triedou, ktorá sa stará o celé vyhodnotenie je trieda `Evaluate`. Ďalšími dôležitými triedami sú `CSequences`, ktorá sa stará o správne zaraďovanie rúk do postupností a trieda `CImage`, ktorá predspracováva vstupné dáta.

V tejto kapitole predstavíme najdôležitejšie triedy a tým priblížime implementáciu algoritmov.

4.1 Trieda `Evaluate`

Táto trieda sa stará o celé vyhodnotenie záznamu. Trieda si uchováva naučený model na klasifikáciu rúk a objekt `CSequences`, ktorý vyhodnocuje jednotlivé postupnosti. Tejto triede dáme dve po sebe nasledujúce snímky (farebná aj hĺbkovú mapu) a dostaneme vektor ohodnotených situácií, teda rúk, ktoré boli vybrané z regálu.

4.2 Trieda `CSequences`

Táto trieda si uchováva naučený model, ktorý klasifikuje podpostupnosti klasifikovaných rúk a zoznam aktuálnych postupností. Stará sa o priraďovanie nových snímkov do správnej postupnosti a vyhodnocuje, či bola situácia ukončená.

4.3 Trieda `CSeqeuence`

Táto trieda reprezentuje konkrétnu postupnosť snímok vloženia a vybrania ruky z regálu. Zabezpečuje jej správne rozdelenie na jednotlivé podpostupnosti, výpočet príznakov a ich samotnú klasifikáciu. Uchováva si aj súradnice stredu ruky z poslednej snímky.

4.4 Trieda `motionSIFT`

Trieda z dvoch po sebe nasledujúcich snímok (farebná aj hĺbková mapa) vypočíta medián pohybu objektu na zázname. Na nájdenie kľúčových bodov sa používa algoritmus SIFT.

4.5 Trieda `CImage`

Trieda reprezentuje jednu snímku zo záznamu. Vytvorí sa z farebnej snímky a k nej príslušnej hĺbkovej mapy. Danú snímku spracuje, vysegmentuje z nej jednotlivé objekty, ktoré sa ukladajú do triedy `CHand`. Trieda si uchováva vektor týchto objektov a obsahuje metódu, ktorá vráti vektor príznakov pre všetky objekty.

4.6 Trieda `CHand`

Trieda reprezentuje jeden objekt vysegmentovaný z obrázku triedou `CImage`. Postará sa o vypočítanie všetkých jeho príznakov.

Kapitola 5

Výsledky

Navrhnutý algoritmus môžeme rozdeliť a vyhodnotiť podľa týchto častí:

1. Klasifikácia snímok podľa časti ruky
2. Detekcia pohybu
3. Sledovanie ruky
4. Klasifikácia postupností

Rozhodujúcou je však presnosť algoritmu ako celku, to znamená rozhodnutie o tom, či zákazník vkladal a vyberal ruku s tovarom alebo prázdnu.

5.1 Klasifikácia snímok podľa časti ruky

5.1.1 Ohodnotenie príznakov

V tabuľke 5.1 a na obrázku 5.1 sú zobrazené vypočítané dôležitosti príznakov podľa χ^2 testu, ktoré určujú významnosť atribútu pri predikovaní triedy.

Tabuľka 5.1: Dôležitosť príznakov snímok rúk podľa χ^2 testu

Č.	**Príznak**	**Dôležitosť**
1.	Priemerná hodnota kanálu H (farebný model HSV)	662,2
2.	Mediánová hodnota kanálu H (farebný model HSV)	292,2
3.	Smerodajná odchýlka kanálu H (farebný model HSV)	550,4
4.	Priemerná hodnota kanálu S (farebný model HSV)	195,4
5.	Mediánová hodnota kanálu S (farebný model HSV)	246,8
6.	Smerodajná odchýlka kanálu S (farebný model HSV)	508,6
7.	Priemerná hodnota kanálu V (farebný model HSV)	282,3
8.	Mediánová hodnota kanálu V (farebný model HSV)	357,2
9.	Smerodajná odchýlka kanálu V (farebný model HSV)	324,8

Pokračovanie na druhej strane

Tabuľka 5.1 – *Pokračovanie*

Č.	**Príznak**	**Dôležitosť**
10.	Obsah kontúry	116,7
11.	Obvod kontúry	96,7
12.	Dĺžka konvexnej obálky	90
13.	Konvexnosť	101,5
14.	Pomer strán opísaného obdĺžnika	107,8
15.	Priemerná početnosť LBP v intervale [0,15]	118,2
16.	Priemerná početnosť LBP v intervale [16,31]	100,5
17.	Priemerná početnosť LBP v intervale [32,47]	49,6
18.	Priemerná početnosť LBP v intervale [48,63]	115,4
19.	Priemerná početnosť LBP v intervale [64,79]	103,6
20.	Priemerná početnosť LBP v intervale [80,95]	73,2
21.	Priemerná početnosť LBP v intervale [96,111]	54,5
22.	Priemerná početnosť LBP v intervale [112,127]	80,3
23.	Priemerná početnosť LBP v intervale [128,143]	105
24.	Priemerná početnosť LBP v intervale [144,159]	63,1
25.	Priemerná početnosť LBP v intervale [160,175]	59,8
26.	Priemerná početnosť LBP v intervale [176,191]	155
27.	Priemerná početnosť LBP v intervale [192,207]	56,8
28.	Priemerná početnosť LBP v intervale [208,223]	45,7
29.	Priemerná početnosť LBP v intervale [224,239]	113,9
30.	Priemerná početnosť LBP v intervale [240,255]	79,8
31.	X-ová súradnica 1. najdôležitejšieho bodu podľa FAST	74,7
32.	Y-ová súradnica 1. najdôležitejšieho bodu podľa FAST	48,1
33.	Dôležitosť 1. najdôležitejšieho bodu podľa FAST	156
34.	X-ová súradnica 2. najdôležitejšieho bodu podľa FAST	104,9
35.	Y-ová súradnica 2. najdôležitejšieho bodu podľa FAST	37,5
36.	Dôležitosť 2. najdôležitejšieho bodu podľa FAST	85,2
37.	X-ová súradnica 3. najdôležitejšieho bodu podľa FAST	88
38.	Y-ová súradnica 3. najdôležitejšieho bodu podľa FAST	32,9
39.	Dôležitosť 3. najdôležitejšieho bodu podľa FAST	60,2
40.	X-ová súradnica 4. najdôležitejšieho bodu podľa FAST	89,3
41.	Y-ová súradnica 4. najdôležitejšieho bodu podľa FAST	34
42.	Dôležitosť 4. najdôležitejšieho bodu podľa FAST	82,2
43.	X-ová súradnica 5. najdôležitejšieho bodu podľa FAST	109,4
44.	Y-ová súradnica 5. najdôležitejšieho bodu podľa FAST	39,3
45.	Dôležitosť 5. najdôležitejšieho bodu podľa FAST	121,4
46.	Hu invariant 0	106,5
47.	Hu invariant 1	0
48.	Hu invariant 2	0
49.	Hu invariant 3	0
50.	Hu invariant 4	0

Pokračovanie na druhej strane

Tabuľka 5.1 – *Pokračovanie*

Č.	**Príznak**	**Dôležitosť**
51.	Hu invariant 5	0
52.	Hu invariant 6	0
53.	1. koeficient DCT metódou ZigZag	109,7
54.	2. koeficient DCT metódou ZigZag	42,2
55.	3. koeficient DCT metódou ZigZag	256,3
56.	4. koeficient DCT metódou ZigZag	93,2
57.	5. koeficient DCT metódou ZigZag	110,8
58.	6. koeficient DCT metódou ZigZag	87,3
59.	Smerodajná odchýlka hĺbok v hĺbkovej mape	129,9
60.	Rozpätie hĺbok v hĺbkovej mape (max − min)	139,5

5.1.2 Ohodnotenie klasifikátorov

Na vyhodnotenie úspešnosti klasifikátorov sme použili trénovaciu množinu s veľkosťou 1 545 snímok. Ich podiel v jednotlivých triedach je uvedený v tabuľke 5.2.

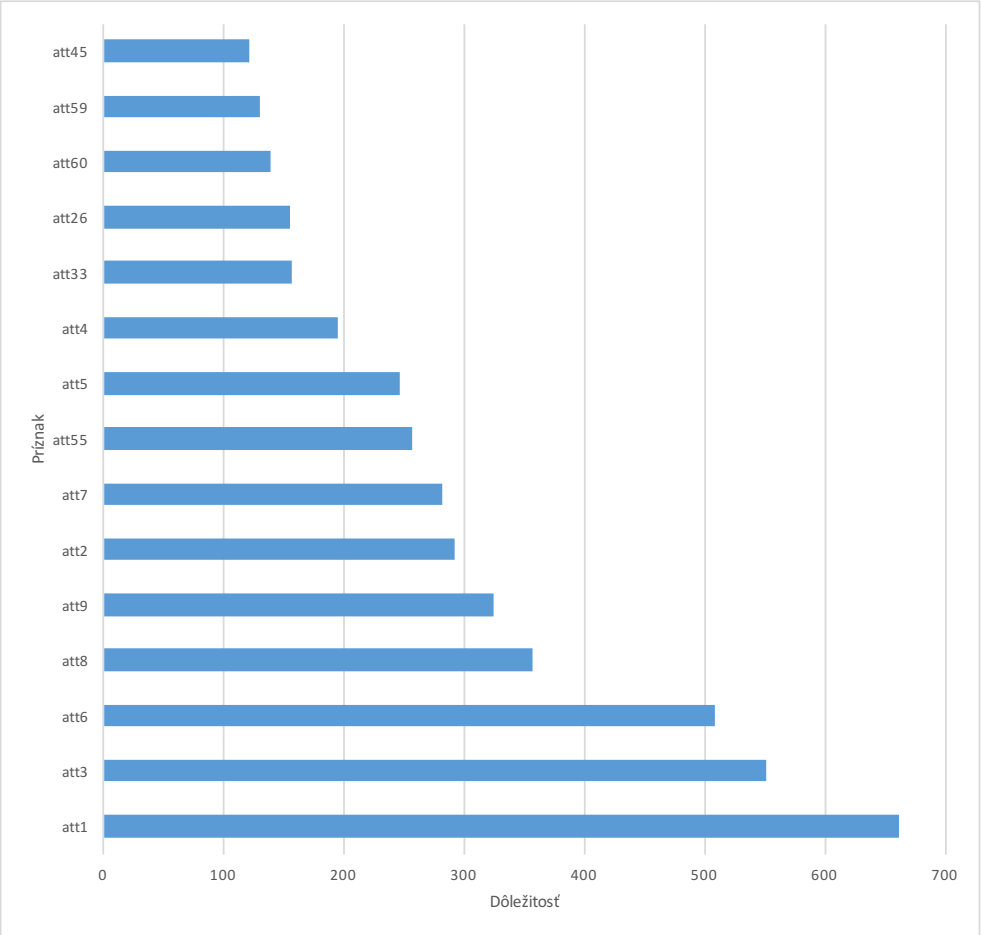
Pre každý klasifikátor sme vypočítali klasifikačnú presnosť a presnosti jednotlivých tried. Použili sme krížovú validáciu, čo znamená, že dáta sa rozdelili do desiatich rovnako veľkých skupín a z nich sa vždy deväť použilo na trénovanie klasifikátora a jedna na testovanie. Takto sa vystriedali všetky dáta a neprišli sme o časť dát na trénovanie. Celková úspešnosť sa počíta ako podiel správne klasifikovaných snímok k celkovému počtu inštancií. Rovnako sa určia aj presnosti v jednotlivých triedach. Namerané hodnoty sú uvedené v tabuľke 5.3.

Najvyššiu presnosť dosahuje klasifikátor Gradient boosted trees, preto ho budeme používať v ďalších častiach práce.

Úspešnosť tejto fázy algoritmu nám určuje úspešnosť použitého klasifikátora. Tá sa môže zmeniť a zvýšiť zväčšením trénovacej množiny. Predovšetkým

Tabuľka 5.2: Počet snímok v jednotlivých triedach pre klasifikáciu rúk

Trieda	Počet	Pomer
1	408	26,4%
2	403	26,1%
3	419	27,1%
4	108	7%
5	207	13,4%
Spolu:	1 545	100%



Obr. 5.1: Dôležitosť príznakov snímok rúk podľa χ^2 testu

Tabuľka 5.3: Celková klasifikačná presnosť a presnosti jednotlivých tried pri použití rôznych klasifikátorov pre klasifikáciu rúk.

Trieda	4-NN	6-NN	Bayes	SVM	Rozhod. stromy	MLP	GBT
1	61%	57%	60%	67%	54%	66%	73%
2	46%	48%	73%	34%	60%	69%	82%
3	52%	59%	80%	38%	51%	63%	81%
4	13%	9%	8%	20%	19%	29%	17%
5	27%	17%	49%	20%	35%	48%	52%
Úspešnosť:	47%	47%	64%	41%	50%	61%	70%

upravenie a vyrovnanie počtu vzoriek v jednotlivých triedach by určite viedlo k zvýšeniu presnosti.

5.2 Detekcia pohybu

Problémy v tejto fáze algoritmu spôsobujú hlavne netypické pohyby pri vkladaní a vyberaní rúk z regálu. Ak zákazník v tejto fáze ruku zastaví na dlhší čas skôr ako ju úplne vložil alebo vybral z regálu, postupnosť nebude rozdelená správne. Kvôli tomuto chybnému rozdeleniu budú podpostupnosti kratšie, čo znižuje úspešnosť algoritmu.

5.3 Sledovanie ruky

Sledovanie ruky na zázname prebieha na úrovni kontúr v hĺbkovej mape. Práve preto problémy spôsobujú hlavne chyby v jej zhotovení zariadením Kinect. Keďže za ruku je považovaný iba objekt, ktorý prechádza celou výškou snímky, tak akékoľvek chyby na hĺbkovej mape, ktoré narušia túto vlastnosť, znemožňujú trasovanie objektu. Väčšina týchto prípadov je eliminovaná nastavením istej tolerancie, ktorá predstavuje počet snímok, počas ktorých môže ruka "zmiznúť" z obrazu, ale budeme ju považovať, že sa na ňom nachádza. Táto tolerancia je nastavená na hodnotu 2. Pri nastavení na vyššiu hodnotu to spôsobovalo aj neželaný efekt, že ak v krátkom čase bola na tom istom mieste vybraná a opäť vložená ruka, algoritmus to vyhodnotil, že ruka sa celý čas na zázname nachádzala.

Ďalším problémom je, ak sa dve ruky priblížia tak blízko, že v hĺbkovej mape z nich vznikne iba jedna kontúra. Vtedy je tento objekt považovaný za jednu ruku, ktorej klasifikácia sa pridáva do postupností oboch rúk.

5.4 Klasifikácia postupností

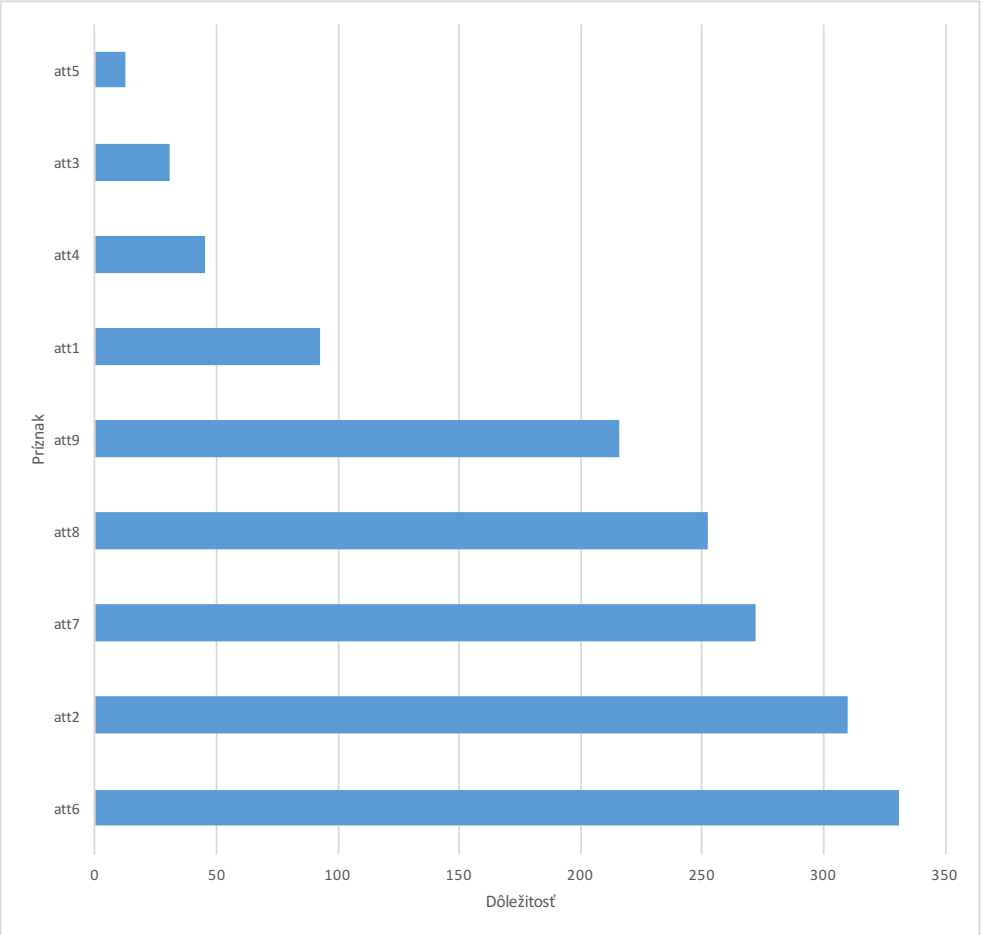
5.4.1 Ohodnotenie príznakov

V tabuľke 5.4 a na obrázku 5.2 sú zobrazené vypočítané dôležitosti príznakov podľa χ^2 testu, ktoré určujú významnosť atribútu pri predikovaní triedy.

Príznaky č. 2, 6, 7 a 8 majú dvoj- až trojnásobnú dôležitosť ako ostatné príznaky.

5.4.2 Ohodnotenie klasifikátorov

Úspešnosť klasifikácie týchto postupností klasifikačným modelom závisí na veľkosti trénovacej množiny. Postupným zberom nových dát dokážeme zachytiť väčšinu prípadov a tak naučiť klasifikátor vyhodnotiť viacero bežných, ale aj výnimočných fenoménov.



Obr. 5.2: Dôležitosť príznakov postupností podľa χ^2 testu

Tabuľka 5.4: Dôležitosť príznakov postupností podľa χ^2 testu

Č.	**Príznak**	**Dôležitosť**
1.	Relatívna početnosť triedy 1	92,7
2.	Relatívna početnosť triedy 2	309,9
3.	Relatívna početnosť triedy 3	30,9
4.	Relatívna početnosť triedy 4	45,4
5.	Relatívna početnosť triedy 5	12,6
6.	Relatívna početnosť triedy 2 a 4	331,3
7.	Pomer 6. príznaku k 1.	271,7
8.	Najpočetnejšia hodnota	252,2
9.	Medián	215,6

Tabuľka 5.5: Celková klasifikačná presnosť a presnosti jednotlivých tried pri použití rôznych klasifikátorov pre klasifikáciu postupností.

Trieda	3-NN	5-NN	Bayes	SVM	Rozhod. stromy	MLP	GBT
bez tovaru	89%	89%	51%	89%	89%	88%	87%
s tovarom	77%	78%	91%	78%	66%	78%	73%
Úspešnosť:	85%	85%	64%	86%	82%	84%	83%

Tabuľka 5.6: Kontigenčná tabuľka klasifikátora postupností

Predikované \Skutočné	*bez tovaru*	*s tovarom*	Vierohodnosť
bez tovaru	**394**	43	90%
s tovarom	49	**154**	76%
Presnosť	89%	78%	

Tabuľka 5.7: Výkonnostné ukazovatele klasifikátora postupností

Ukazovateľ	Hodnota
Presnosť	86%
Senzitivita	89%
Špecifita	78%
Vierohodnosť	90%
F-miera	90%

Ako klasifikačné modely sme skúsili tie isté ako na klasifikáciu častí ruky. Úspešnosť sme merali na trénovacej množine s veľkosťou 640 vzoriek, kde bolo 443 vzoriek z triedy *prázdna ruka* a 197 vzoriek z triedy *ruka s tovarom*. Použili sme opäť krížovú validáciu s desiatimi skupinami. Namerané výsledky sú zobrazené v tabuľke 5.5.

SVM a k-NN majú podobnú presnosť, rozhodli sme sa ďalej využiť klasifikátor SVM. V tabuľke 5.6 vidíme kontigenčnú tabuľku pri použití klasifikátora SVM a v tabuľke 5.7 sú zobrazené vypočítane výkonnostné ukazovatele pri zvolení triedy *bez tovaru* ako pozitívnu.

5.5 Celková presnosť

Celkovú presnosť algoritmu sme merali na dátach získaných počas štyroch dní – 27.9., 29.9., 8.10. a 9.10. Každú situáciu vloženia a vybrania ruky z regálu sme vyhodnotili algoritmom a zapísali do tabuľky 5.8. V tabuľke môžeme vidieť aj vypočítané presnosti a vierohodnosti jednotlivých skupín, do ktorých boli situácie zaraďované.

Tabuľka 5.8: Kontigenčná tabuľka vyhodnotenia algoritmu. OO – prázdna => prázdna, XX – s tovarom => s tovarom, OX – prázdna => s tovarom, XO – s tovarom => prázdna.

Predikované \Skutočné	OO	XX	OX	XO	Vierohodnosť
OO	**32**	0	13	7	62%
XX	1	**6**	0	2	67%
OX	1	0	**20**	1	91%
XO	1	0	2	**12**	80%
Presnosť	91%	100%	57%	55%	

Najčastejšou skupinou je prázdna ruka dnu – prázdna ruka von. Táto možnosť nastala 35-krát a v 32 prípadoch ju algoritmus rozpoznal správne, čo predstavuje 91% presnosť.

Druhou najčastejšou situáciou je prázdna ruka dnu – ruka s tovarom von. Tento prípad nastal 35-krát, ale iba 20-krát to bolo rozpoznané správne, preto sme dosiahli presnosť iba 57%. Pri tejto triede sme dosiahli vyššiu vierohodnosť – 91%. To znamená, že ak algoritmus rozpozná túto situáciu, má pravdu s pravdepodobnosťou 91%.

Situácia, kedy bola vložená aj vybraná ruka s tovarom nastala 6-krát a vždy bola klasifikovaná správne. Poslednou možnosťou je vloženie ruky s tovarom a vybranie prázdnej ruky. Táto varianta nastala 22-krát, z toho 12-krát bola klasifikovaná správne, čo sa rovná presnosti 55%. Dosiahli sme ale vyššiu vierohodnosť 80%.

Celkovo bolo zistených 98 situácií. Z nich bolo správne rozpoznaných 70, čiže celková presnosť algoritmu je 71%. Chyby boli najčastejšie spôsobené tým, že tovar bol z väčšej časti zakrytý rukou zákazníka, prípadne bol pohyb veľmi rýchly a snímky sa nestihli zhotoviť. Ďalším problémom boli výpadky na hĺbkovej mape, čo znamená, že sledovaná ruka "zmizla" zo záznamu a neskôr bola považovaná za ďalšiu vloženú ruku.

Záver

Práca sa zaoberá detekciou tovaru v ruke zákazníka na kamerovom zázname získaného z reálnej prevádzky obchodu. Záznam bol zhotovený zariadením Kinect, ktorý okrem farebnej snímky poskytuje aj hĺbkovú mapu. Kinect je umiestnený nad regálom a poskytuje pohľad na ruky zákazníka zhora.

Navrhli sme algoritmus, ktorý využíva dva klasifikátory. Na klasifikáciu jednotlivých snímok podľa častí ruky, ktorá sa na nich nachádza bol použitý model Gradient boosted trees. Na učenie bolo použitých 60 príznakov, ktoré boli získané z farebnej snímky aj hĺbkovej mapy. Z celej situácie vloženia a vybrania ruky z regálu sa vytvorí postupnosť tried, do ktorých boli jednotlivé snímky klasifikované. Táto postupnosť sa využitím algoritmu SIFT rozdelí na podpostupnosti, ktoré opisujú vloženie ruky a vybranie ruky z regálu. Z týchto podpostupností sa získa 9 príznakov, podľa ktorých sa naučí ďalší klasifikátor. Týmto spôsobom sa nám podarilo zvýšiť úspešnosť, ktorú sme dosiahli prvým klasifikátorom. Celková presnosť tohto algoritmu dosahuje 71%. Navrhnutý algoritmus sme implementovali v jazyku C++ s použitím knižnice OpenCV.

Algoritmus je stále možné v budúcnosti vylepšovať. Prvou možnosťou je navrhnutie ďalších príznakov pre oba klasifikátory, čo bude viesť k zvýšeniu ich presnosti a tým aj k zvýšeniu presnosti celého algoritmu. Ďalšou oblasťou, kde vidíme priestor na vylepšenie je postup pri rozdeľovaní postupnosti tried na podpostupnosti. Ak postupnosť bude rozdelená presnejšie, môžeme získať dlhšiu podpostupnosť, ktorá lepšie opisuje danú situáciu. Za úvahu stojí aj zmena spôsobu odstránenia pozadia, teda podlahy, od objektov na zázname. Týmto by sa eliminovali chyby na hĺbkovej mape.

Literatúra

[1] KITA, P.: Merchandising v obchodnej prevádzke predajne. In *Vedecké state Obchodnej fakulty 2011*, Bratislava: Ekonóm, 2011, ISBN 978-80-225-3326-3.

[2] MITTAL, A.; Zisserman, A.; Torr, P. H.: Hand detection using multiple proposals. In *BMVC*, Citeseer, 2011, s. 1–11.

[3] PARK, M.; Hasan, M. M.; Kim, J.; aj.: Hand detection and tracking using depth and color information. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV'12)*, ročník 2, 2012, s. 779–785.

[4] MEI, K.; Xu, L.; Li, B.; aj.: A real-time hand detection system based on multi-feature. *Neurocomputing*, ročník 158, 2015: s. 184–193.

[5] STERGIOPOULOU, E.; Sgouropoulos, K.; Nikolaou, N.; aj.: Real time hand detection in a complex background. *Engineering Applications of Artificial Intelligence*, ročník 35, 2014: s. 54–70.

[6] Itseez: About. *OpenCV* [online]. ©2016 [cit. 2016-04-03]. Dostupné z: `http://opencv.org/about.html`

[7] SONKA, V. H. a. R. B., Milan: *Image processing, analysis, and machine vision*. Toronto: Thompson Learning, třetí vydání, 2008, ISBN 9780495082521.

[8] OpenCV: Miscellaneous Image Transformations. *OpenCV 2.4.12.0 documentation* [online]. ©2011-2014 [cit. 2016-04-03]. Dostupné z: `http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html`

[9] i Art Corporation: About Depth. *i-Art 3D* [online]. ©2012-2015 [cit. 2016-04-03]. Dostupné z: `http://www.i-art3d.com/Eng/About_Depth.htm`

[10] SUZUKI, S.; aj.: Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, ročník 30, č. 1, 1985: s. 32–46.

[11] WEISSTEIN, E. W.: Green's theorem. 2003. Dostupné z: `http://mathworld.wolfram.com/GreensTheorem.html`

[12] SKLANSKY, J.: Finding the convex hull of a simple polygon. *Pattern Recognition Letters*, ročník 1, č. 2, 1982: s. 79–83.

[13] OpenCV: Structural Analysis and Shape Descriptors. *OpenCV 2.4.12.0 documentation* [online]. ©2011-2014 [cit. 2016-04-03]. Dostupné z: `http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html`

[14] HU, M.-K.: Visual pattern recognition by moment invariants. *information Theory, IRE Transactions on*, ročník 8, č. 2, 1962: s. 179–187.

[15] OpenCV: Operations on Arrays. *OpenCV 2.4.12.0 documentation* [online]. ©2011-2014 [cit. 2016-04-03]. Dostupné z: `http://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html`

[16] OJALA, T.; Pietikainen, M.; Harwood, D.: Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. In *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, ročník 1, IEEE, 1994, s. 582–585.

[17] OpenCV: FAST Algorithm for Corner Detection OpenCV. *3.0.0-dev documentation* [online]. ©2011-2014 [cit. 2016-04-03]. Dostupné z: `http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html`

[18] LOWE, D. G.: Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, ročník 60, č. 2, 2004: s. 91–110.

[19] Salomonsson, F.: Automatic Panorama Stitching. *Fredrik "PlaTFooT"Salomonsson* [online]. ©2011-2014 [cit. 2016-04-03]. Dostupné z: `https://fredriksalomonsson.wordpress.com/category/projects/automatic-panorama-stitching/`

[20] SUROVEC, T.: *Extrakce obrazových lokálních deskriptorů pomocí GPU*. Bakalárska práca, Masarykova univerzita, Fakulta informatiky, Brno, 2011, [cit. 2016-04-03]. Dostupné z: `http://is.muni.cz/th/255880/fi_b/bc.pdf`

[21] OpenCV: Introduction to SIFT (Scale-Invariant Feature Transform). *OpenCV tutorials* [online]. ©2011-2014 [cit. 2016-04-03]. Dostupné z: http://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html

[22] FLACH, P. A.: *Machine learning: the art and science of algorithms that make sense of data*. New York: Cambridge University Press, 2012, ISBN 1107422221.

[23] KORDÍK, P.: *Rozhodovací stromy*. Prednáška, BI-VZD, ČVUT, Fakulta informačních technologií, Praha, 2016.

[24] OpenCV: Introduction to Support Vector Machines. *OpenCV 2.4.12.0 documentation* [online]. ©2011-2014 [cit. 2016-04-03]. Dostupné z: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[25] OpenCV: Understanding k-Nearest Neighbour. *OpenCV tutorials* [online]. ©2011-2014 [cit. 2016-04-03]. Dostupné z: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_ml/py_knn/py_knn_understanding/py_knn_understanding.html

[26] OpenCV: Gradient Boosted Trees. *OpenCV 2.4.12.0 documentation* [online]. ©2011-2014 [cit. 2016-04-03]. Dostupné z: http://docs.opencv.org/2.4/modules/ml/doc/gradient_boosted_trees.html

[27] FRIEDMAN, J. H.: Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 2001: s. 1189–1232.

[28] OpenCV: Common Interfaces of Descriptor Matchers. *OpenCV 2.4.12.0 documentation* [online]. ©2011-2014 [cit. 2016-04-03]. Dostupné z: http://docs.opencv.org/2.4/modules/features2d/doc/common_interfaces_of_descriptor_matchers.html

Dodatok **A**

Algoritmus SIFT

Algoritmus Scale Invariant Feature Transform (SIFT) predstavený v práci *Distinctive Image Features from Scale-Invariant Keypoints* [18] slúži na nájdenie tzv. kľúčových bodov (features) v obrázku. Tieto body je možné následne popísať deskriptorom, čo nám umožní ich párovanie s týmito bodmi na inom obrázku. Tento postup sa využíva pri sledovaní objektu vo videu, nájdenie objektu na obrázku alebo napríklad pri rekonštrukcii scény (obrázok A.1) z viacerých čiastkových obrázkov, ktoré môžu byť rôzne veľké, prípadne pootočené.

Kľúčové body sú invariantné voči mierke obrázku, rotácii a čiastočne invariantné aj voči zmene osvetlenia a polohe fotoaparátu. Extrahovanie týchto bodov je efektívne a je možné nájsť ich dostatočne veľké množstvo. Sú vysoko

Obr. A.1: Rekonštrukcia scény z čiastkových obrázkov, ktoré boli zhotovené z rôznych uhlov a vzdialeností [19]

rozlíšiteľné, čo zvyšuje pravdepodobnosť ich správneho spárovania voči mohutnej množine bodov. Táto vlastnosť nám poskytuje základ pre rozpoznanie scény a objektov.

Algoritmus pozostáva zo štyroch hlavných častí:

1. Detekcia lokálnych extrémov v scale-space
2. Lokalizácia kľúčových bodov
3. Priradenie orientácie
4. Vytvorenie deskriptoru

A.1 Detekcia lokálnych extrémov v scale-space

V tejto časti sa lokalizujú vhodní kandidáti na kľúčové body, ktoré budú invariantné k zmenám mierky.

A.1.1 Vytvorenie scale-space

Využiva sa tu tzv. scale space obrázku, ktorý je definovaný ako funkcia $L(x, y, \sigma)$, ktorá je tvorená konvolúciou Gaussovskej funkcie $G(x, y, \sigma)$ so vstupným obrázkom $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{A.1}$$

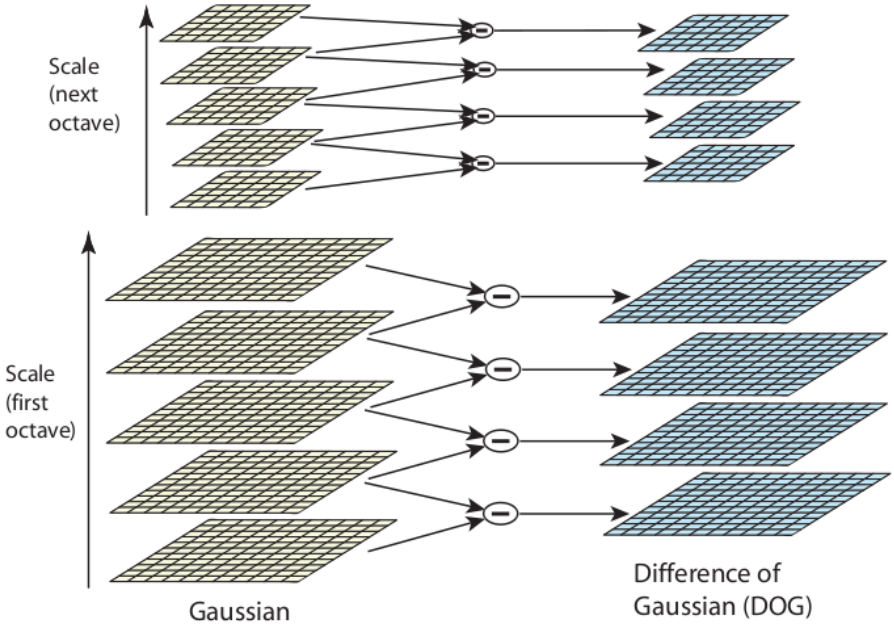
kde $*$ označuje konvolúciu x a y a

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

Najprv je vytvorená tzv. pyramída (obrázok A.2), ktorá pozostáva z oktáv. Vstupný obrázok je v prvej oktáve viacnásobne konvoluvaný podľa vzťahu A.1. Po každej oktáve je posledný rozostrený obrázok dvojnásobne downsamplovaný a vytvorí sa ďalšia oktáva (obrázok A.3), v ktorej sa postupuje rovnako. Po vygenerovaní pyramídy každé dva susedné gaussiány od seba odčítame a dostaneme tzv. rozdielové obrázky (Difference of Gaussians – DOG, obrázok A.4).

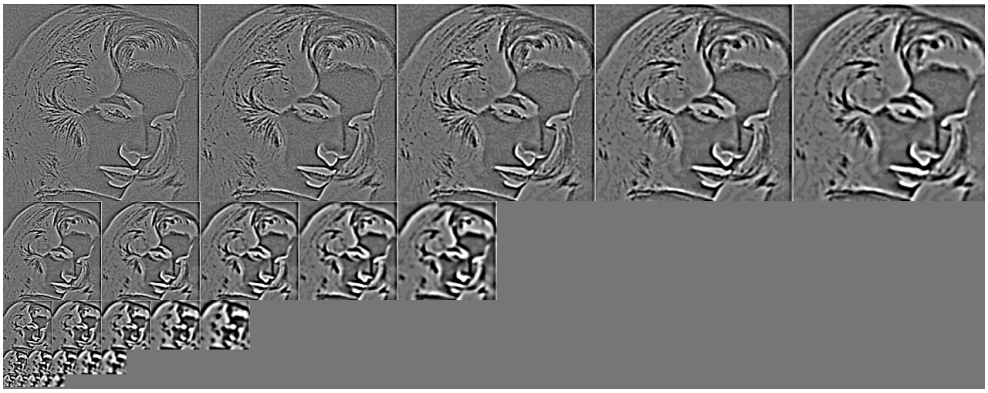
A.1.2 Nájdenie extrémov

Ak je vytvorená pyramída rozdielových obrázkov, pokračuje sa nájdením lokálnych extrémov. Každý pixel je porovnaný s okolitými 8 pixelmi a s 9 pixelmi v oboch susedných obrázkoch. Na obrázku A.5 je porovnávaný pixel označený čiernym krížikom. Ak je pixel lokálnym extrémom, stane sa kandidátom na kľúčový bod. Znamená to, že tento bod je najlepšie reprezentovaný v danej škále.

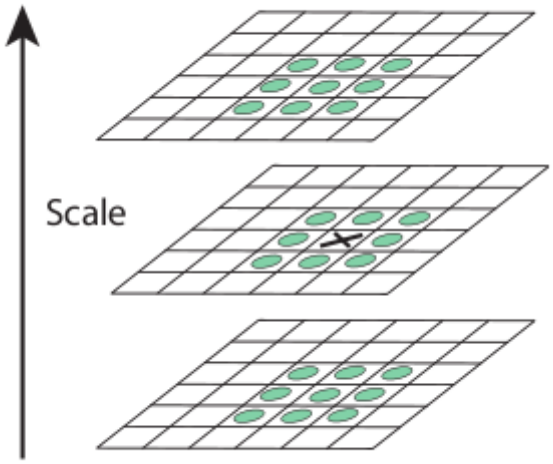


Obr. A.2: Vytvorenie scale space [18]

Obr. A.3: Ukážka scale space s piatimi oktávami [20]



Obr. A.4: Ukážka rozdielov gaussiánov [20]



Obr. A.5: Hľadanie lokálnych extrémov [18]

A.2 Lokalizácia kľúčových bodov

Po nájdení potenciálnych kľúčových bodov je potrebné upresniť ich polohu a škálu. To nám umožní odstrániť body, ktoré majú nízky kontrast a sú citlivé na šum a body, ktoré sa nachádzajú na hrane.

A.2.1 Odmietnutie bodov s nízkym kontrastom

Využíva sa Taylorov rad funkcie na rozdiel gaussiánov $D(x, y, \sigma)$ posunutej tak, aby vychádzala zo skúmaného bodu:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{A.3}$$

kde D a jeho derivácia je vypočítaná v skúmanom bode a $\mathbf{x} = (x, y, \sigma)^T$ je posun z tohto bodu. Poloha extrému $\hat{\mathbf{x}}$ je určená deriváciou tejto funkcie podľa x, ktorej určíme nulový bod:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}} \tag{A.4}$$

Dosadením rovnice A.4 do rovnice A.3 dostaneme

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}} \tag{A.5}$$

čo nám určuje funkčnú hodnotu upresneného extrému. Ak je $|D(\hat{\mathbf{x}})|$ menej než určitá hranica, potenciálny kľúčový bod sa zahodí. Táto hranica je parametrom algoritmu SIFT a nazýva sa *contrastThreshold* [21].

A.2.2 Odmietnutie bodov na hranách

Algoritmus nájde veľké množstvo bodov na hranách, ktoré sú citlivé aj na malý šum, takže nie sú vhodné na ďalšie spracovanie a je potrebné ich odstrániť. Vypočítame hlavné zakrivenie funkcie rozdielov gaussiánov pomocou 2×2 Hessianovej matice $\mathbf{H}$:

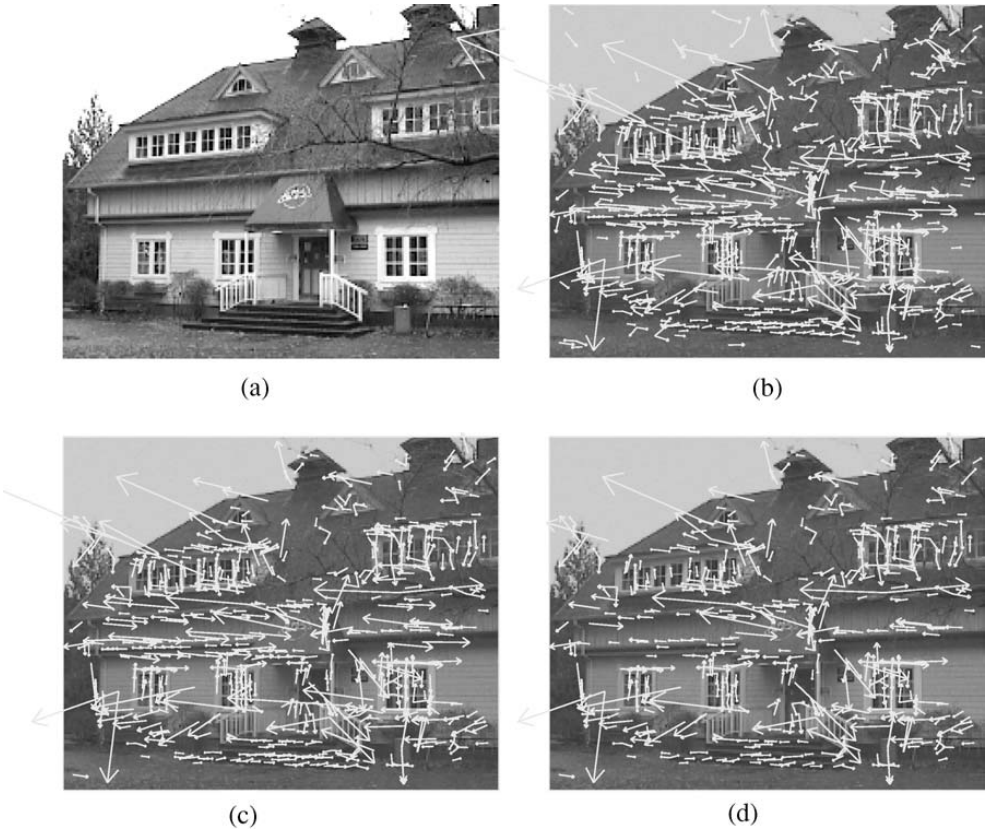
$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{A.6}$$

Nepotrebujeme vypočítať vlastné čísla, keďže nás zaujíma iba ich pomer. Nech α je väčšie a β menšie vlastné číslo, potom vieme vypočítať ich súčet a ich súčin:

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$
$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

Nech r je pomer väčšieho vlastného čísla k menšiemu, tak $\alpha = r\beta$. Potom

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$



Obr. A.6: Na tomto obrázku môžme vidieť ako prebieha vyradenie nájdených bodov. (a) Originálny 233x189 pixelový obrázok. (b) Zobrazených 832 bodov, ktoré boli nájdené ako extrémy na rozdielových obrázkoch. Body sú znázornené ako vektory. (c) Odstránené body s nízkym kontrastom, zostalo 729 bodov. (d) Konečných 536 bodov po odstránení bodov na hranách [18].

čo závisí len na pomere vlastných čísel, nie na ich hodnotách. Tento výpočet je veľmi efektívny a pri určenej hranici r, ktorá sa nazýva *edgeThreshold* [21] nám stačí overiť či

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}.$$

V opačnom prípade tento bod vyradíme a ďalej s ním už nepracujeme.

A.3 Priradenie orientácie

Každému kľúčovemu bodu je priradená orientácia na dosiahnutie invariancie voči rotácii obrázku. Vypočíta sa veľkosť $m(x, y)$ a orientácia $\theta(x, y)$ gradientov v okolí bodu, pričom sa nepoužíva rozdielový obrázok, ale gaussovsky

rozostrená snímka $L(x, y)$:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

Z vypočítaných veľkostí a smerov gradientov v okolí bodu sa vytvorí sa histogram orientácií v intervaloch po 10 stupňov. Pri výpočte výslednej orientácie sa zahrnie globálne maximum a ďalšie extrémy, ktorých hodnota je väčšia než 80% maxima. Ak sa tieto orientácie líšia, vytvorí sa viac kľúčových bodov, ktoré budu mať rovnakú lokáciu, ale odlišnú orientáciu.

A.4 Vytvorenie deskriptora

Posledným krokom je vytvorenie deskriptora, ktorý je vysoko rozlíšiteľný. Tento deskriptor rozšíri invarianciu aj voči ďalším zmenám ako je napríklad zmena osvetlenia alebo zmena polohy fotoaparátu.

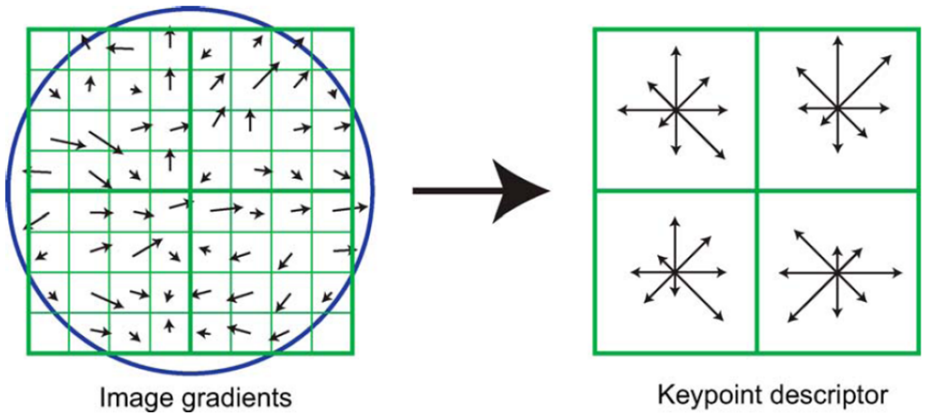
Pre vytvorenie deskriptora potrebujeme najprv vypočítať veľkosti a orientácie gradientov v okolí kľúčového bodu s veľkosťou 16x16 pixelov. Orientácie gradientov sú rotované relatívne k orientácii bodu, aby bola zachovaná invariancia voči smeru. Týmto hodnotám sú priradené váhy z Gaussovej funkcie so σ rovnajúcej sa polovici šírky deskriptora. Okolie sa rozdelí do 16 blokov s veľkosťou 4x4 pixely. Pre každý tento bod je zhotovený histogram orientácii gradientov s ôsmymi intervalmi hodnôt. Takýmto spôsobom získame pre každý blok osem hodnôt, čiže celkovo 128 vlastností pre každý kľúčový bod.

Posledným krokom pri tvorbe deskriptora je jeho úprava, ktorá zredukuje efekt zmeny osvetlenia. Ako prvé, vektor vlastností normalizujeme do intervalu $[0, 1]$. Zmena kontrastu, pri ktorej sa hodnoty pixelov vynásobia konštantou, vynásobí tou istou konštantou aj gradienty, takže táto zmena kontrastu bude odstránená normalizáciou. Zmena jasu, pri ktorej sa hodnoty pixelov zvýšia o konštantu, nezmení gradienty, pretože tie sa počítajú z rozdielov pixelov. Tým pádom je deskriptor invariantný voči zmenám v osvetlení.

Okrem lineárnych zmien však môžu nastať aj nelineárne. Ich príčinou je saturácia zdroja obrazu alebo rozdielne nasvietenie rôzne naklonených hrán objektov. Tieto zmeny redukujeme prahovaním hodnôt väčších než 0,2 na túto hodnotu a potom opäť normalizujeme do intervalu $[0, 1]$. Hodnota tohto prahu bol určená experimentálne použitím obrázkov s rôznym nasvietením tých istých objektov.

A.5 Párovanie kľúčových bodov

Na využitie získaných kľúčových bodov a ich deskriptorov v praxi je dôležité vedieť ich spárovať s bodmi na inom obrázku. Napríklad pri hľadaní objektu z jedného obrázku na druhom.



Obr. A.7: Deskriptor sa vytvára vypočítaním veľkostí a orientácii gradientov v okolí kľúčoveho bodu ako je to zobrazené na ľavom obrázku. Kružnica znázorňuje Gaussovu funkciu, ktorou sú tieto hodnoty prenásobené. Na pravom obrázku je histogram, do ktorého sú tieto hodnoty nakumulované. V tejto ukážke sa vypočítava deskriptor s veľkosťou 2x2 získaný z 8x8 pixelov veľkého okolia. V algoritme sa využíva deskriptor veľký 4x4 získaný z 16x16 pixelového okolia [18].

Keďže deskriptor je vlastne bod v 128-dimenzionálnom priestore, párovať ich budeme vypočítaním Euklidovskej vzdialenosti. Spárujeme ho s bodom, s ktorým má skúmaný bod najmenšiu vzdialenosť.

Nie všetky body však majú správny pár, keďže môžu pochádzať napríklad z pozadia. Tieto body je potrebné vyradiť. Najefektívnejším spôsobom je porovnať vzdialenosť najbližšieho a druhého najbližšieho bodu. Ak je ich pomer väčší ako 0,8, body vyradíme. Týmto postupom eliminujeme 90% nesprávnych spárovaní.

Dodatok B

Zoznam použitých skratiek

BSD Berkeley Software Distribution

DOG Difference of Gaussian

FAST Features from accelerated segment test

GBT Gradient boosted trees

HSV Hue, saturation, value model

IR Infrared

k-NN k-nearest neighbours

LBP Local binary patterns

MLP Multilayer perceptron

RGB Red, green, blue model

SIFT Scale-invariant feature transform

SVM Support vector machine

Dodatok C

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `attributes` ..... vypočítané príznaky
  - `hand.csv` ..... príznaky z obrázkov ruky
  - `seq.csv` ..... príznaky z postupností
- `models` ..... klasifikáčné modely vo formáte YAML
  - `hand.yml` ..... klasifikácia rúk
  - `seq.yml` ..... klasifikácia postupností
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `BP_Kerul-Kmec_Oliver_2016.pdf` ..... text práce