Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Bc. Michal Ševčík

[**Klasifikátor binárnych deskriptorov**](https://147.175.146.86/project/show/5560)

Diplomová práca

**Študijný program: Informačné systémy**

**Študijný odbor: 9.2.6 Informačné systémy**

**Miesto vypracovania: Bratislava, Slovensko**

**Vedúci diplomovej práce: Ing. Ján Kvak**

**December, 2014**

ANOTÁCIA

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: Informačné systémy

Autor: Bc. Michal Ševčík

Diplomový projekt: Klasifikátor binárnych deskriptorov

Vedenie diplomového projektu: Ing. Ján Kvak

December 2014

Cieľom tejto diplomovej práce je výskum existujúcich klasifikátorov použiteľných v oblasti rozpoznávania obrazu. Zameriava sa hlavne na tie klasifikátory, ktoré dokážu pracovať s binárnymi deskriptormi. Tieto klasifikátory sú v práci bližšie opísané. Výsledok práce je nájdenie a prispôsobenie klasifikátora, ktorý by dosahoval dobré výsledky pri použití v prostredí real-time. Spomenuté sú taktiež výsledky klasifikácie pri rôznych konfiguráciách. Hlavne sa opisuje ich úspešnosť a rýchlosť ktoré sa dosiahli.

ANNOTATION

Slovak University of Technology Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree Course: Informational systems

Author: Bc. Michal Ševčík

Diploma project: Classification of binary descriptors

Supervisor: Ing. Ján Kvak

2014, December

The focus of this diploma thesis is to research the existing image classificators. It focuses mainly on those classificators that work with binary descriptors. These classificators are furtherdescribed in the work. The output of the work is a classificator that will achieve good results while used in a real-time environment. It also contains results of the classificator with different configurations. It is describing mainly the success and the speed.

Contents

[1. Úvod 1](#_Toc419045706)

[2. Analýza 2](#_Toc419045707)

[2.1. Klasifikátor 2](#_Toc419045708)

[2.2. Trénovacia množina 3](#_Toc419045709)

[3. Deskriptory 4](#_Toc419045710)

[3.1. Binárne deskriptory 4](#_Toc419045711)

[3.3. Vyhodnotenie binárnych deskriptorov 8](#_Toc419045712)

[4. Algoritmy klasifikácie 11](#_Toc419045713)

[4.1. Nearest Neighbours (k-nearest neighbours) 11](#_Toc419045714)

[4.2. Naive Bayes Klasifikátor 16](#_Toc419045715)

[4.3. Semi-Naive Bayes 21](#_Toc419045716)

[4.4. Rozhodovacie stromy 24](#_Toc419045717)

[4.5. Náhodné lesy 27](#_Toc419045718)

[5. Návrh 30](#_Toc419045719)

[6. Implementácia 31](#_Toc419045720)

[6.1. Postup implementácie 32](#_Toc419045721)

[7. Testovanie 41](#_Toc419045722)

[8. Záver 47](#_Toc419045723)

[9. Literatúra 48](#_Toc419045724)

[10. Použivateľská príručka 50](#_Toc419045725)

[10.1. Nadstavenie vstupných parametrov pre triedu Main 50](#_Toc419045726)

[10.2. Špecifikácia parametrov klasifikátorov 51](#_Toc419045727)

Skratky

k-NN k-Nearest Neighbours

MLE Maximum Likehood estamination

FAST Features from Accelerated Segment Test

SURF Speeded Up Robust Feature

BRIEF Binary robust independent elementary features

ORB Oriented FAST and Rotated BRIEF

SIFT Scale Invariant Feature Transform

FREAK Fast Retina Keypoint

BRISK Binary Robust Invariant Scalable Keypoints

RANSAC Random Sample Consensus

OpenCV Opensource Computer Vision library

1. Úvod

Ako sa technológie vyvíjajú, programy sa stávajú viac sofistikované a majú tendenciu byť autonómnymi. Odvetvie umelej inteligencie sa zaoberá vytváraním inteligentného agenta, schopného pozorovať prostredie. Z tohto pozorovania sa program snaží vyvodiť najlepšie riešenie situácie. Aby získal údaje o vonkajších podnetoch, potrebuje senzory ako sú mikrofón, kamera atď, ktoré premenia vonkajšie podnety na dáta. Pri pozorovaní vonkajšieho prostredie kamerou, musí program spracovať dáta, ktoré vidí pred sebou. Typický príklad, kedy je identifikácia a rozpoznanie SPZ značiek pred vchodom automobilov do garáži nákupných stredísk alebo automobily, ktoré sú autonómne od spoločnosti Google. Ide o snahu zautomatizovať automobilovú dopravu, kedy človek len zadá cieľový bod a automobil sa tam automatický dostane. No na ceste sa môžu nachádzať rôzne nečakané prekážky. Na tieto prekážky musí automobil reagovať v čo najkratšom čase.

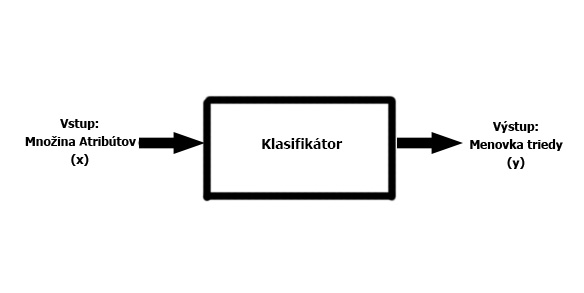
Známe existujúce deskriptory ako SIFT sa používajú v počítačoch, ktoré sú v dnešnej dobe veľmi výkonné a stále majú tendenciu sa stať ešte výkonnejšími. No nastala éra mobilných telefónov, ktoré disponujú takými hardware parametrami, ktoré mali počítače pred pár rokmi. Ľudia sa snažia nahradiť v teréne počítače za mobilné zariadenia.ť. Výkon týchto mobilných zariadení však nie je taký ako pri klasických počítačov a výpočty trvajú dlhší čas. Preto boli vytvorené binárne deskriptory, ktoré sú výpočtovo jednoduhšie a taktiež zaberajú v zariadeniach menej pamäťového miesta.

1. Analýza

Pri rozpoznávaní obrazu - klasifikácií, sa využíva strojové učenie. Strojové učenie je to,že program sa naučí sám klasifikovať čo na obrázku, ktorý práve sníma, vidí. Ide o transformáciu prijatých dát na informácie. Program sa z týchto informácií naučí rozpoznávať jednotlivé objekty z obrázkov. Rozpoznávanie pracuje na princípe vyhodnotenia nájdeného objektu, ktorý sa potom spracuje a porovná. Porovnáva sa s trénovacou množinou a ako výsledok sa berie ten objekt, na ktorý sa neznámy objekt najviac podobá. Aby sme našli objekt na ktorý sa objekt najviac podobá, používame klasifikátor, ktorý ho rozpozná [1].

* 1. Klasifikátor

Klasifikátor je funkcia f, ktorá mapuje vložený vektor príznakov na výstupnú triedu kde X je Trénovacia množina. Tento základný prístup klasifikátorov vysvetlí obrázok. [1]



Obrázok Proces klasifikácie [1]

* 1. Trénovacia množina

Pri rozpoznávaní sú získané dáta predspracované na príznaky. Aby sme získali správne výstupy (rozpoznanie objektu) používame algoritmy strojového učenia. Tieto algoritmy analyzujú zozbierané dáta. Jednou z metrík určujúce úspešnosť klasifikátora je veľkosť trénovacej množiny.[1] Rozpoznávanie prebieha tak, že sa nasníma neznámy objekt. Tento objekt sa rozpozná podľa klasifikácie čŕt a trénovacej množiny, ktorá ho naučila klasifikovať vlastnosti, podľa získaných čŕt. Môže nastať situácia že odhad nebude postačujúci. V tomto prípade sa môže zvýšiť počet objektov v trénovacej množine, alebo sa zvolí iný klasifikátor a proces klasifikácie sa zopakuje.

* + 1. Dáta pod dohľadom/ bez dohľadu

Dáta v trénovacej množine môžu byť označené menovkami (meno, vek,..) alebo neoznačené. Pri označených údajoch ide o dáta s dohľadom.[1] Tieto dáta sa posielajú klasifikátoru aj s nejakým údajom, ktorý ich opisuje. Učenie klasifikátora, ktorý sa učí s dátami s dohľadom, sa nazýva učenie s učiteľom. Môže ísť o kategorické učenie s učiteľom, kedy sa asociuje názov k objektu (meno k rozpoznávanej tvári), alebo dáta môžu mať číselné alebo usporiadané menovky (vek k rozpoznávanej tvári). Ak dáta majú menovky v podobe mien (kategórie), tak hovoríme že ide o klasifikáciu. Ak menovky sú číselné, hovoríme že ide o regresiu. Učenie môže byť jedno k jednému: párovanie menoviek s dátovými vektormi, alebo môže ísť o odložené učenie (zosilnené učenie).[1]

Pri zosilnenom učení, sa používa dátová menovka[1] (taktiež nazývaná odmena alebo trest). Rozpoznávanie prebieha tak, že systém získa oneskorený signál (odmenu alebo trest) a podľa toho sa snaží odvodiť ďalšie kroky pre rozpoznávanie. Učenie s učiteľom môže mať takisto čiastočne menovky, kde niektoré menovky chýbajú alebo sú zlé. Ide o tzv. semi supervised učenie. Väčšina algoritmov strojového učenia rieši jednu alebo dve situácie z opísaných. Napríklad algoritmy strojového učenia môžu riešiť klasifikáciu ale nie regresiu, algoritmy dokážu robiť semi supervised učenie, ale nie zosilnené učenie, algoritmus dokáže pracovať s číselnými dátami ale nie s kategorickými súčasne. [1]

Ďalšie z algoritmov rozpoznávania pracujú nad dátami, ktoré nie sú pod dohľadom, sú neopísané. Ide len o zistenie čí skúmane dáta spadajú do skupín. Skupina týchto algoritmov sa volá clusteringové algoritmy. Cieľom je zoskupiť neoznačené dátové vektory, ktoré sú si podobné.[1]

1. Deskriptory

Za deskriptory v počítačovom videní sú považované opisy význačných bodov tzv. keypointov v obrázkoch. Na nájdenie význačných bodov sa využívajú algoritmi detekcie význačných bodov, ktorého výstupom sú pixelové koordináty význačných bodov a tým aj význačných miest – plôch. Príkladom význačného bodu je hrana na obrázku kde je ostrý prechod. Tieto význačné body okrem pozície žiadnu inú relevantnú informáciu nedajú. Pre získanie informácií o týchto plochách je potrebné ich opísať pomocou deskriptorov. Existuje množstvo algoritmov, ktoré sa sústredia na opis týchto bodov.

* 1. Binárne deskriptory

Binárne deskriptory sa skladajú z troch základných častí:

* Zozbieranie bodov z okolia význačného bodu.
* Získanie informácií o orientácií – natočenia bodov a normalizácia tohoto natočenia
* Vytvorenie párov z bodov

Jedným z dôvodu vzniku binárnych deskriptorov sú stúpajúce počty obrázkov v databázach, ktoré sú spracované menej výkonnými mobilnými zariadeniami. Preto je snaha aby tieto zariadenia boli nie len presné ale dĺžka trvania výpočtov znesitelná. Na základe spomenutých faktorov prinášaju binárne deskriptory riešenie a alternatívu k známym deskriptorom ako je SIFT a SURF, ktoré pracujú s pohyblivou desatinnou čiarkou a tým je výpočet náročnejší. Podľa výskumov binárne deskriptory ponúkajú veľmi podobný rozpoznávací výkon pričom je zredukovaná veľkosť dát a jeho výpočtová cena.[18]

BRIEF

Binárny descriptor BRIEF pracuje na jednoduchom porovnávaní intenzít dvoch bodov pričom týchto porovnávaní je viac. Výsledkom je binárny reťazec, ktorý opisuje význačný bod. Pre porovnanie deskriptora BRIEF s inými deskriptormi:

* SIFT využíva 128 dimenzií pre vektor na deskriptor a spomínané čísla s pohyblivou desatinnou čiarkou, zaberá približne 512 bytov.[1]
* SURF podobne potrebuje minimálne 256 bytov pre vektor so 64 dimenziami.[1]

Vytváranie takýchto vektorov (deskriptormi SIFT a SURF) pre množstvo význačných bodov, zaberá veľa pamäte a preto aj následne rozpoznávanie – klasifikácia trvá dlhší čas. Všetky tieto dimenzie nie sú potrebné pre rozpoznávanie. Preto ich možno odľahčiť a skomprimovať pomocou viacerých metód ako sú PCA, LDA dokonca pomocou hashhovania – LSH. Tieto metódy sa teda využívajú na kompresiu z čísel s pohyblivou desatinnou čiarkou na binárne reťazce. Tieto binárne reťazce sa využívajú na rozpoznávanie – mapovanie nájdených význačných bodov (príznakov) za pomoci hamminových vzdialeností. Toto zabezpečuje zrýchlenie nájdenia zhôd keďže použitie hammingovej vzdialenosti je jednoduchá procesorová inštancia XOR a následný počet kladných bitov. A tak tento spôsob je veľmi rýchly v moderných procesoroch využívajúce sadu SSE inštrukcií. Avšak ako prvé treba tieto deskriptory nájsť a až potom je možne využiť jednu z metód konverzie do binarného reťazca.

Deskriptor BRIEF prichádza ako zjednodušenie zísaknia binárnych reťazcov bez potreby konverzie – získanie binárneho reťazca priamo. Postup generovania binárnych reťazcov je výber plôch obrázka, výber párov a ich vzájomné porovnávanie intenzít, pričom výsledok porovnávania je 0 alebo 1 a tak vznikne binárny reťazec.Porovnávanie intenzít dvoch bodov je založené na princípe náhodnosti. Nepočíta sa tu žiadna informácia o natočení a nemá vzor na získavanie vzoriek z okolia význačného bodu.

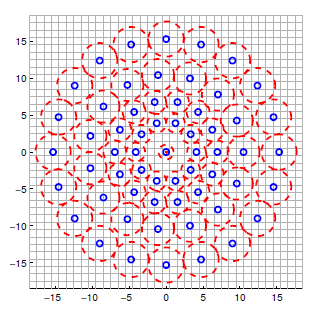
ORB

ORB deskriptor je veľmi podobný BRIEF deskriptoru, ale líši sa v dvoch základných bodoch.

* ORB rieši orientáciu jednotlivích bodov, vďaka čomu sa stáva invariantný voči natočeniu.[18]
* ORB používa učenie sa na výber optimálnych vzorkovacích párov – BRIEF funguje na úplne náhodnom princípe výberu týchto bodov.[18]

BRISK

Deskriptor brisk sa líši od deskriptorov ORB a BRIEF vo vzore zbierania bodov z okolia význačného bodu. Tento vzor bol ručne vytvorený a je konštantný. Ide o sústredné kruhy:

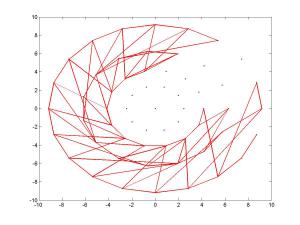
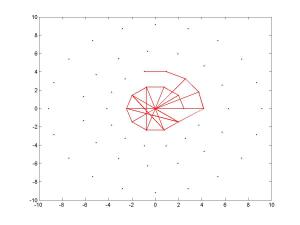


Obrázok 2 Vzor na vyhľadávanie význačných bodov [12]

Pri zbieraní bodov z každého miesta odberu vzoriek (Sampling point), sa zoberie malé okolie tohoto bodu, a aplikuje sa Gaussov vyhladzovací filter. Z obrázku možno vidieť že čevené okolie znázorňuje odchýlku Gaussového filtra pre každý bod vybranej vzorky.[18]

Pri používaní tohoto spôsobu výberu vzoriek, rozďelujú sa páry medzi krátke páry (short pairs) a dlhé páry (long pairs). Krátke páry su tie, ktorých vzdialenosť je menšia ako určitý threshold (d\_max) a naopak dlhé páry sú také, ktorých vzdialenosť je väčšia ako iný threshold (d\_min) a všeobecne platí: d\_min > d\_max takže žiaden pár nemôže byť klasifikovaný ako dlhý a krátky zároveň. Dlhé páry slúžia na získanie orientácie a krátke na porovnávanie intenzít, ktoré vytvárajú deskriptor ako aj pri deskriptore BRIEF a ORB.[18]

Príklad tvorby krátkych párov:

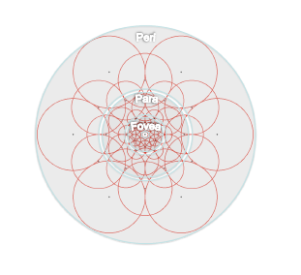


Obrázok 3 Tvorba párov [12]

FREAK

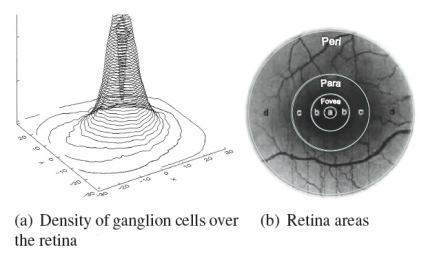
FREAK deskriptor je podobný BRISK deskriptoru z pohľadu vzoru na získavanie vzoriek. Ide taktiež o vzor ktorý je pevne stanovený. Podobnosť je aj prideskriptore ORB pretože používa strojové učenie pri získavanie množiny párov. FREAK disponuje aj mechanizmom na určenie rotácie, ktorý je podobný deskriptoru BRISK [12].

Vzor, ktorý sa používa na výber vzoriek sa nazýva retina mriežka (retina sampling grid), ktorá je taktiež kruhového tvaru (BRISK) pričom majú tieto kruhy hustejšie zastúpenie pri strede bodu. Hustota sa znižuje vzdialením sa od stredu:



Obrázok 4 Retina vzor [12]

Každá vzorka bodu je vyhladená Gaussovským filtrom, kde rádius kruhu znázorňuje veľkosť štandardnej odchýlky. Na obrázku možno vidieť rozloženie receptívneho pola cez retinu.



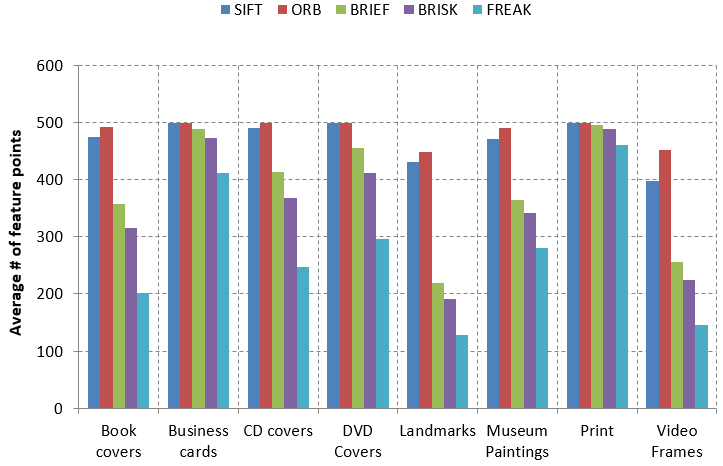
Obrázok 5 Hustota výberu bodov [12]

* 1. Vyhodnotenie binárnych deskriptorov

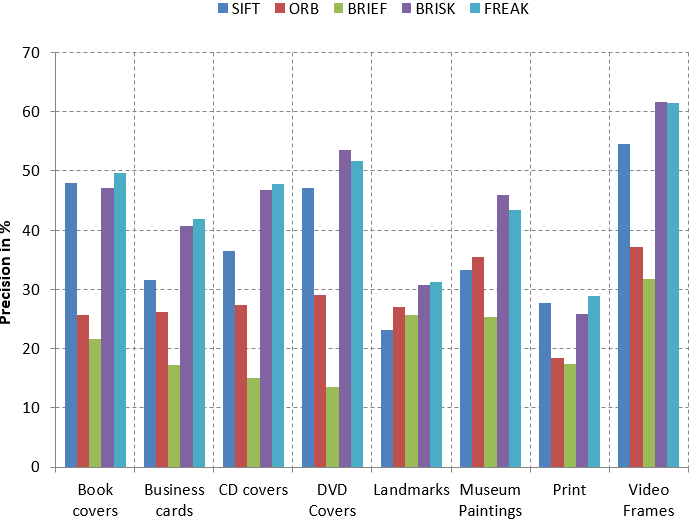
Výkony jednotlivích deskriptorov boli testované na dvoch datasetoch. Prvý je The Oxford data set [10] a druhý The Stanford Mobile Visual Search (SMVS) [11]. Prvý dataset obsahuje osem rôznych scén, kde každá obsahuje jednu zo zmien: jas, zmena bodu z ktorého bol obraz snímaný, rotácia, škála a rozmazanie. SMVS dataset obsahuje 3300 obrázkov pre 1200 tried v 8 rôznych kategóriach. Obrázky boli vytvorené rôznymi zariadeniami, od fotoaparátov v telefónoch až po klasické digitálne fotoaparáty. Scény boli fotené vnútri aj vonku.

Pri testovaní BRIEF, BRISK a FREAK deskriptoru, je použitý SURF detektor a pre ORB deskriptor sa používa ORB detektor.

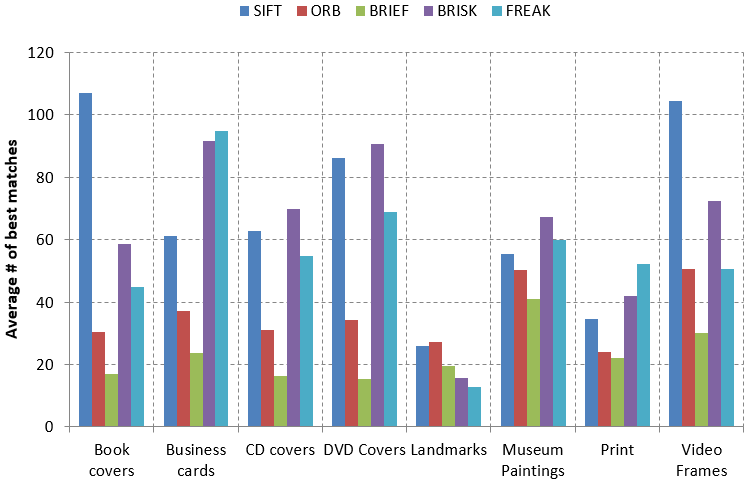
Počet význačných bodov je limitovaný na maximálnu hodnotu 500. Pri získaní viacerých deskriptorov sa počet zníži a zoberie sa len prvých 500. Na porovnávanie obrázkov sa využíva Brute Force matcher. Dva najlepšie body su vybraté podľa vzdialenosti medzi ich deskriptorimi. Ak táto vzdialenosť je moc nízka pre prvý nájdený bod a moc veľká pre druhý nájdený bod, tak sa vyberá prvý bod. Ak sú tieto dva body od seba blízko tak chybovosť môže byť veľká a v tom prípade oba body sú zamietnuté.Podľa výskumu článku [17] je detekcia význačných bodov následujúca:



Prvým testom bol test pomerov, ktorý pozostáva z výpočtu pomeru vzdialenosti medzi význačními bodmi. Ide o pomer vzdialenosti medzi najlepšiou zhodou a druhou najlepšou zhodou. Ak je výsledok nad 0.9 oba význačne body sú ignorované. Ak nie, najlepšia zhoda je zachovaná. Na základe tohoto testu je test RANSAC, ktorý sa snaží nájsť najlepšie vzťahy medzi dvoma obrázkami. Výsledkom sú najlepšie zhody respektíve správne zhody. Potom výsledkom je pomer najlepších zhôd k počtu zhôd, ktoré prešli pomerovým testom. A tak graf zobrazuje ako správne na základe týchto deskriptorov boli body rozpoznané. BRISK a FREAK majú najlepšie výsledky a BRIEF najhoršie. [17]:



Ďalším testom je test priemerov najlepších zhôd. Ide o test kde sa násobí priemerný počet význačných bodov s presnosťou počítanom v predchádzajúcom teste. Za presnosť sa pokladá pomer najlepších zhôd ku význačným bodom, ktoré sú pokladané za zhody, pričom sem patria aj tie body, ktoré boli odstránené pomerovým testom. V tomto prípade je vyhodnotený deskriptor BRISK ako najlepší okrem vzoriek zo sád pre Biznis kartičky, krajiny a tlač. Brief dáva najmenší počet dobrých zhôd. Hneď po BRISK deskriptore je FREAK, ktorý má najviac dobrých zhôd na Biznis kartičkách a pre tlač. ORB a BRIEF majú najlepšie výsledky pre rozpoznávanie krajín. [17]



Podľa dokumentu [17] je BRISK odporúčaný ako najlepší binárny deskriptor, ktorý má najviac dobrých zhôd. Výsledky deskriptora BRISK sa dajú porovnať s výsledkami deskriptora SIFT. Avšak na výpočet BRISK deskriptora treba viac výpočtovej sily ako na BRIEF, ORB a dokonca i FREAK deskriptor..

1. Algoritmy klasifikácie

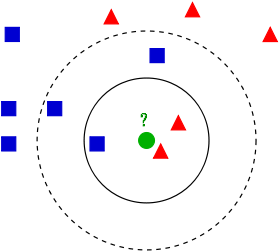
V nasledujúcej kapitole sa budeme zaoberať opisom jednotlivých algoritmov strojového učenia a klasifikácie. Väčšina týchto algoritmov, ktoré spomenieme, sú naimplementované v knižnici OpenCV.

* 1. Nearest Neighbours (k-nearest neighbours)

Klasifikácia NN je založená na porovnávaní neznámych objektov s identifikovanými objektmi z trénovacej množiny. Podľa toho, ku ktorému z jeho “susedov” sa najviac blíži vieme určiť že o čo s najväčšou pravdepodobnosťou pôjde. Je to často krát užitočné vziať viac ako jedného suseda pre získanie relevantnejších výsledkov. Kvôli tomuto sa často krát odkazuje tento klasifikátor na k-NN. Trénovacie vzorky sú potrebné počas spusteného programu (run-time). Celý čas musia byť v pamäti počítača, a preto sa táto klasifikácia označuje aj ako Memory-based classification (klasifikácia založená na pamäti). Pretože začiatok programu je spomalený z dôvodu predspracovania dát, je táto technika učenia považovaná za Lazy Learning technique (Technika lenivého učenia)[2]. A pretože klasifikácia je založená priamo na trénovacej množine, ide o Example-Based Classification (Klasifikácia založená na príkladoch) alebo Case-BasedClassification (Prípadovo založená klasifikácia).

Samotná klasifikácia k-NN má dve fázy: prvá je rozhodnutie, ktorý zo susedných prvkov je najbližší k hľadanému prvku. Druhá fáza – finálna je rozhodnutie triedy do ktorej spadá podľa najbližších susedov. Toto možno vidieť na obrázku. Okrúhly objekt je rozpoznávaný, a trojuholník spolu so štvorcom sú už rozpoznané. Klasifikátor sa podľa vzdialeností (sústredné kružnice) rozhoduje, ako ho klasifikovať. Berie do úvahy len tie prvky, ktoré sú po určenú vzdialenosť.

D je trénovacia množina, ktorá obsahuje všetky rozpoznané prvky, F je množina príznakov každého prvku a Y je množina menoviek.



Obrázok 6 Klasifikácia neznámeho objektu pomocou k-NN [2]

Máme trénovaciu množinu D, ktorá obsahuje trénovacích vzoriek. Vzorky sú opísané podľa príznakov F a všetky tieto numerické príznaky boli normalizované s rozsahom [0,1]. Každá trénovacia vzorka má menovku triedy Cieľom je klasifikovať neznámy prvok q. Pre každý môžeme pomocou výpočtov vzdialenosti medzi q a podľa vzorca:

Existuje veľa spôsobov vzdialenostných metrík. Základná verzia pre kontinuálne a diskrétne atribúty by bola:

* 0 – f je diskrétna a
* 1 - f je diskrétna a
* || - f je kontinuálna

V algoritme k-NN je selekcia založená na vzdialenostných metrikách. Existuje množstvo variácií spôsobov v ktorých k-NN môže byť použitá na rozpoznanie triedy q. Jednoduchým prístupom je priradiť majoritnú triedu hľadanému prvku. Pri vkladaní váh okolitým susedom, dáva väčší zmysel priradiť vyššiu váhu susedovi ktorý je bližšie, ako tomu, ktorý je ďalej. Technika, ktorá sa na takéto hodnotenie používa, sa nazýva vzdialenostné volenie váh.[2] Susedia „volia“ s hlasmi, ktoré majú váhu inverznú k ich vzdialenosti od neznámeho prvku.

Vzorec vzdialenostného volenia váh má tvar [2]:

Voľba prislúcha premennej podľa susedov podieľu jednotky podľa vzdialenosti . Táto vzdialenosť vráti 1 ak sa menovky triedy zhodujú a inak 0.

Ďalší z prístupov je voľba založená na Shepardovej práci a používa exponenciálnu funkciu radšej ako inverznú vzdialenosť[2]:

* + 1. Vzdialenostné metriky

Metrika podobnosti a metrika vzdialenosti sú často používané na zmeranie afinity medzi dvoma objektmi. Afinitou sa myslí príbuznosť. Metrika má formálne vysvetlenie v matematike. Vysvetľuje čo vypočítane hodnoty znamenajú. V prípade k-NN hovorí o vzdialenosti objektu:

* d(x,y) > 0 nezáporná
* d(x,y) = 0 iba ak x=y; identita
* d(x,y) = d(y,x) symetria
* d(x,z) > d(x,y) + d(y,z) trojuholníková nerovnosť

Podľa týchto prvých dvoch metrík vzniká špeciálny prípad Minkowského vzdialenostnej metriky[2] :

Minkowského vzdialenostná metrika je veľmi generický príklad metriky, ktorý možno použiť v k-NN klasifikátoroch. Táto vzdialenosť sa môže použiť pre rôzne dáta, ktoré reprezentujú vektor príznakov. Obrázok, ktorý sa väčšinou pri rozpoznávaní používa, je čiernobiely a histogram H s N levelmi odtieňov, alebo so zásobníkmi, kde je počet pixelov, ktoré spadajú do intervalu reprezentovaného zásobníkom i (vektor h je príznakový vektor). Minkowského vzdialenostný vzorec môže byť použitý na porovnanie dvoch obrázkov, opísaných histogramom. Pri práci s obrázkami existujú aj iné vzorce na porovnanie obrázkov. Ide napríklad o Kullback-Leiblerovu divergenciu a štatistiku [2]:

Pričom H a K sú dva histogramy, h a k sú vektory, ktoré korešpondujú vektorom zo zásobníkových hodnôt a . Tieto vzorce sa zameriavajú na informačnú a štatistickú teóriu a majú isté nevýhody. Prvá z nich je že nejde o metriky a tak nedokážu uspokojiť symetrické požiadavky. Avšak toto sa dá ľahko prekonať pri definovaní modifikovanej vzdialenosti medzi x a y, čo je spôsob vytvorenia priemerov a – Jeffreyho divergencia.[2] Jeffreyho divergencia je symetrická verzia Kullback-Leiblerovej divergencie. Ďalšia z nevýhod je, že tieto meranie sú náchylne na chyby kvôli ohraničeniam zásobníkov. Rozdiel medzi obrázkom a trošku tmavším tým istým obrázkom môže byť veľký ak pixely spadajú do rozdielnych zásobníkov.

* + 1. Výhody a nevýhody k-NN

k-NN je veľmi jednoduchý na pochopenie a implementáciu. Takže by sa dal použiť na každý klasifikačný problém. Výhody tohto prístupu sú:

* Keďže tento proces je transparentný, je ho jednoduché implementovať a ladiť.
* V situáciách, kde vysvetlenie výstupu klasifikátora je užitočné, k-NN môže byť veľmi efektívne ak analýza susedov je užitočná ako vysvetlenie.
* Existujú techniky redukcie hluku, ktoré pracujú len pre k-NN, ktoré môžu zlepšiť efektívnosti klasifikátora.

Na ďalšej strane tento prístup má aj svoje nevýhody:

* Pretože všetka práca sa vykonáva počas behu programu, k-NN môže byť dosť pomalý ak trénovacia množina je veľká.
* Klasifikátor k-NN je veľmi senzitívny na irelevanciu alebo redundanciu príznakov, pretože všetky príznaky prispievajú k podobe a ku klasifikácií. Toto môže byť zlepšené pri opatrnom výbere príznakov.
* Pri veľmi zložitých klasifikačných úlohách, k-NN môže byť prekonaný niektorými exotickými technikami ako napríklad Podpornými vektorovými strojmi alebo neurónovými sieťami.[2]
  1. Naive Bayes Klasifikátor

Naive Bayes klasifikátor je štatistický klasifikátor, ktorý sa zameriava na pravdepodobnosť. To znamená že využíva metódy, ktoré zobrazujú pravdepodobnosť že pri rozpoznávaní ide o daný objekt. Existujú dva spôsoby ako takýto výstup dosiahnuť. Prvý spôsob je priame naučenie funkcie, ktorá vypočíta triedu pravdepodobností. Tento model výpočtu sa nazýva diskriminatívny model, pretože diskriminuje spomedzi tried získaných pri vstupe. Alternatíva k tomuto modelu je class-conditional density (hustota tried-podmienok) pre každú hodnotu y a pre naučenie classpriors (priorít tried) . Potom sa môže aplikovať Bayesovo pravidlo na výpočet pravdepodobnosti. [3]

Toto sa nazýva generatívny model, keďže špecifikuje ako možno generovať príznakové vektory x, pre každú možnú triedu y. Alternatívou ku generatívnemu a diskriminativnému učeniu je rozdelené učenie so všetkými pravdepodobnosťami spolu. Táto metóda priamo mapuje vstupy na výstup:

* + 1. Výhody Naive Bayes Klasifikátora

Medzi výhody Naive Bayes klasifikátora patrí:

* Odmietnutie výberu. Podľa distribúcie pravdepodobnosti vieme, či výsledok klasifikácie je veľmi pravdepodobný alebo nepravdepodobný. Podľa toho sa môže rozhodnúť že neposkytne klasifikáciu a prenechá to na človeka aby rozhodol o čo sa jedná. Tento princíp má uplatnenie napríklad v medicíne, kedy doktor môže rozhodnúť ak pravdepodobnosť nie je dostatočne veľká. [3]
* Zmena užitočnej funkcie. Môže sa skombinovať pravdepodobnostná distribúcia s funkciou užitočnosti aby sa minimalizoval risk. Hoci môžeme diskriminant naučiť aby priamo minimalizoval riziko. Výhodou je, že naša funkcia užitočnosti sa mení a nemusíme znova učiť . [3]
* Kompenzácia pre nevyrovnanosť tried. Ak je jedna trieda omnoho viac pravdepodobná ako druhá (trieda2 = zriedkavá - 1 prípad z 1000). Ak učíme klasifikátor s nevyrovnanými dátami, môže získať triviálnu 99,9% presnosť keď jednoducho sa naučí pravidlo f(x) = trieda1, čiže vždy bude výstup rovnaký bez ohľadu na to, čo je vstupom. Aby sa vyhlo tomuto degradovanému riešeniu, môžeme ho natrénovať s trénovacou množinou, ktorej dáta majú rovnomerne pravdepodobnosti. Zavolá sa výsledný model . Na začiatku behu programu, môžeme tento model modifikovať, aby pracoval nad skutočnou distribúciou dát [3]:

Keďže

Táto konverzia pravdepodobnosti sa nazýva škálovaný pravdepodobný trik.

* Kombinácia modelov. Ak máme dva odlišné príznakové vektory x1 a x2 (jeden zobrazuje krvné testy a ďalší zlomeniny). Oplatí sa viac spraviť dva separátne klasfikátory p(y|x1) a p(y|x2) a potom ich skombinovať. Jeden monolitický klasifikátor sa neoplatí robiť. Základ pri kombinácií výstupov z rôznych zdrojov informácií (fúzny senzor) je vedieť ako spoľahlivý každý zdroj je. Toto presne je to, čo nám pravdepodobnosť hovorí. Jednoduchý spôsob ako skombinovať systémy je predpokladať že príznaky sú podmienkovo nezávisle poskytujúce menovky triedú [3]:

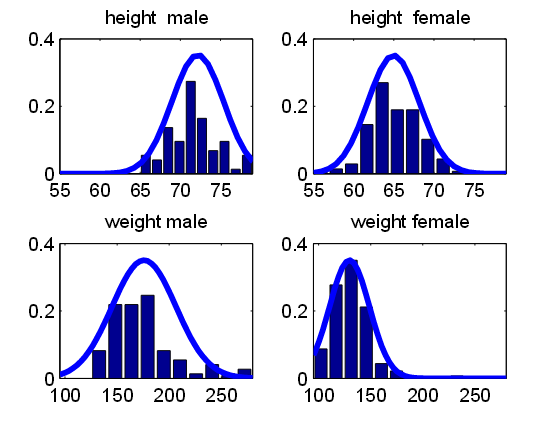
Ide o Naive Bayes predpoklad, ktorý možno vysvetliť využitím triku škálových podobností:

* + 1. Generatívny klasifikátor -Trieda priorít

Pri predpoklade že počet tried C je malý, môžeme jednoducho zistiť prioritnú triedu pri použití y ako multinominálnej náhodnej premennej [3]:

Kde je vektor tried pravdepodobnosti. MLE je [3]:

Pričom je veľkosť trénovacej množiny ktoré majú menovku c. Ak sa použije Dirichletova priorita potom stredná hodnota bude[3]:



Obrázok 7: Príklad gausovské rozloženie dvoch čŕt(výška a váha u žien a mužov)[3]

* + 1. Naive Bayes predpoklad

Naive Bayes predpoklad alebo idiot Bayes [3] predpoklad je že všetky príznaky sú podmienkovo nezávislé:

Hoci toto tvrdenie je zvyčajne nepravdivé (keďže príznaky sú väčšinou závislé na sebe), výsledný model v prípade Gaussovských dát má tvar [3]:

Takže treba len odhadnúť Gaussovské parametre. V prípade binárnych dát dostaneme [3]:

A v tomto prípade treba odhadnúť separátnyBernoulliho parameter . [3]

Celá Bayesovská pravdepodobnosť, je predpovednie hustoty na triedu s menovkou Y so vstupom X a trénovacou množinou D. Toto je dané nasledujúcim vzorcom[3]:

Čo je často približne:

V prípade kde všetky premenné majú multinominálnu distribúciu s Ditrechtovými prioritami, použitím stredného posteriora druhá rovnica bude úplne korektná.[3]

* + 1. Prístupy Naive Bayes klasifikátora

Pre výpočet pravdepodobnosti pri klasifikácií, existuje viac prístupov a modelov, ktoré možno použiť. Ide o matematické modely založené na rôznych matematických pravidlách. Známe typy sú [8]:

* Gaussian naive Bayes
* Multinominal naive Bayes
* Bernouli naive Bayes

**Gaussian naive Bayes**

Využíva sa pri práci s dátami, ktoré sú kontinuálne. Pri kontinuálnych dátach možno predpokladať že sú rozložené podľa gaussovského modelu. Najskôr sa vykoná segmentácia dát podľa tried a potom sa vypočíta priemer a odchýlka pre x v každej triede. Zo získaných hodnôt – priemeru a odchýlky, možno získať pravdepodobnosť po dosadení do vzorca normálového rozloženia, ktoré obsahuje parametre priemeru a odchýlky:

**Multinominal naive Bayes**

Vzorky v multinominalnom naive Bayes klasifikátore reprezentujú frekvenciu výskytu určitých udalosti, ktoré sú generované multinominálne(,, kde je pravdepodobnosť že udalosť i nastane. Typicky sa používa pre klasifikáciu dokumentov.

**Bernouli naive Bayes**

Hodnoty v tomto modeli, sú booleanove. Taktiež sa používa pre klasifikáciu dokumentov.

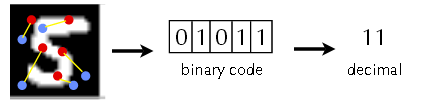
* 1. Semi-Naive Bayes

Semi-Naive Bayes oproti Naive Bayes má viac vyváženú komplexitu. Výmena komplexita/výkon je z dôvodu použitia fernov. Tieto ferny sú o veľkosti S a ich počet je L:

Ferny pracujú tak, že vytvoria sériu binárnych testov na vstupnom vektore. Vždy sa jedná o dvojice bodov, ktoré sa porovnávajú. Platí pravidlo:

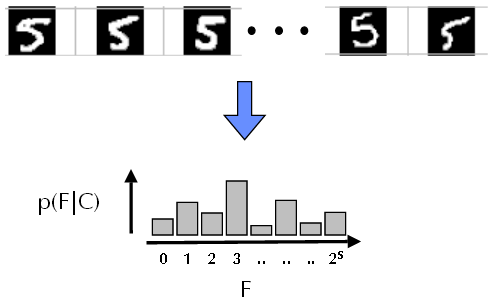
Čo vo všeobecnosti znamená, že ak intenzita prvého bodu je väčšia ako druhého bodu tak výsledok testu je 1 (True) a naopak.

Takéto testy nám potom dajú ako výsledok, S-miestny binárny kód pre príznak, ktorý môže byť interpretovaný ako prirodzené číslo z intervalu. Číslo S sa vyberá podľa počtu dvojíc príznakov resp. podľa počtu binárnych testov.



Obrázok 8: Binárne testy [5]

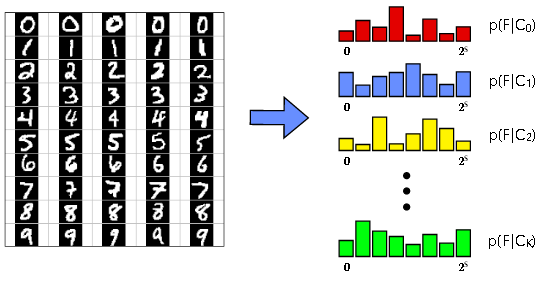
Potom ako sa uskutoční viacero testov na vstupných vektoroch (z trénovacej množiny) rovnakej triedy získame výstup, ktorý je multinominálne rozloženie.



Obrázok 9Multinominálne rozloženie triedy[5]

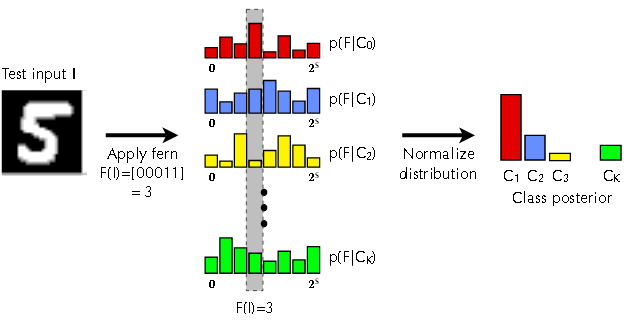
* + 1. Trénovanie fern-u

Ako bol fern opísaný v predchádzajúcej kapitole, na každý prvok z trénovacej množiny je vykonaný binárny test a vypočíta sa výstup . Po natrénovaní celej trénovacej množiny, týmto spôsobom sa naučí multinominálne hustoty v podobe histogramového výstupu pre každú triedu.



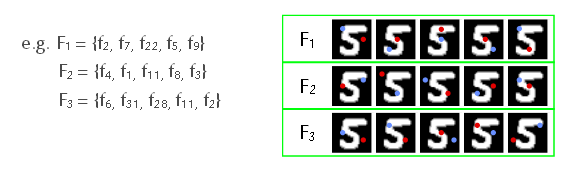
Obrázok 10 Vytvorenie histogramov pre celú trénovaciu množinu [5]

Pri klasifikácií aplikujeme fern na neznámy objekt, a následne vyhľadáme v histogramoch tried ku ktorému naučenému objektu sa tento neznámy objekt najviac podobá:



Obrázok 11 Klasifikácia [5]

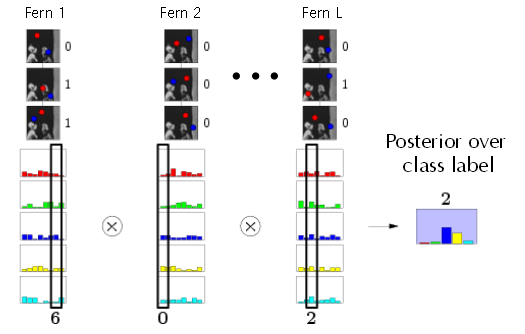
Tento spôsob klasifikácie sa dá vylepšiť pomocou pridania náhodnosti. Pri klasifikácií nebudeme používať iba jeden fern a stále tie isté príznačné body, ale vytvoríme množinu fernov. Každý fern bude obsahovať náhodne zvolené body z množiny príznakov. Tieto ferny budú nezávislé, čiže každý bude obsahovať podmnožinu príznakov, ktorá bude unikátna.



Obrázok 12 Množina fernov [5]

Ferny sú označené F1, F2,... a ich príznaky f1,f2,... Po tomto skombinujeme výstupy pomocou Semi-Naive Bayesovho prístupu:

Pri rozpoznávaní sa používaju viaceré fern-y, ktoré vznikli náhodným výberom príznakov. Každý fern rozpoznávaný objekt očísluje podľa spomínaných binárnych testov. Výstup každého fernu (stĺpec v histograme) sa vektorovo vynásobí. Podľa výsledku sa získa hodnota triedy z trénovacej množiny.[5]



Obrázok 13 Rozpoznávanie [5]

Na obrázku je možné vidieť že prvý binárny test má hodnotu 6 čiže vyberáme 6. hodnotu histogramu, pričom číslujeme od 0. Po vynásobení týchto histogramov, nám vznikne výsledný histogram, ktorý možno vidieť vpravo. V tomto vzniknutom histograme vidíme, že prevahu má 2. stĺpec a tak sa objekt klasifikuje ako trieda 3.

Keďže veľkosť fernov môže byť veľmi veľká (pre fern veľkosti 10 je výstupný histogram v intervale od [0…210] v desiatkovej sústave [0…1024]). Je tu vysoká šanca, že niektoré prvky histogramu zostanú prázdne. Toto bude problém pri Semi-NaiveBayesovi. Takže, ak budeme predpokladať Dirichletovu prioritu na distribúcií P(F|Ck), môžeme priradiť minimálnu pravdepodobnosť každému prvku histogramu. Táto pravdepodobnosť sa vypočíta nasledovne.[5]

kde určuje koľko krát sme pozorovali výstup Fern, ekvivalentný k z trénovacej množiny triedy . Výhoda fernov je, že sa porovnávajú vždy intenzity dvoch bodov. Ak je obraz pod inými svetelnými podmienkami (preniká naňho silnejšie svetlo), rozpoznávanie bude rovnaké lebo intenzita každého jedného bodu sa zvýši o rovnakú hodnotu.[4][5]

* 1. Rozhodovacie stromy

Ďalším  klasifikátorom sú tzv. rozhodovacie stromy. Ide ako už z názvu vyplýva o stromovú štruktúru. Skladá sa z týchto častí:

* Koreň – ktorý nemá žiadne vstupné hrany a má výstupné hrany, ktoré z neho vychádzajú.
* Uzol – má jedinú vstupnú hranu a presne dve výstupne hrany
* List alebo Terminál – má jednu vstupnú hranu a žiadnu výstupnú hranu

V každom rozhodovacom strome je listom pridelená menovka triedy. Uzly, obsahujú podmienky, podľa ktorých sa ďalej rozhoduje, ktorou vetvou sa ďalej vybrať pri klasifikácií. Toto separuje od seba rozdielne rozpoznávané objekty. Nasledujúci príklad rozhodovacieho stromu zobrazuje klasifikáciu živočícha medzi cicavce/necicavce:



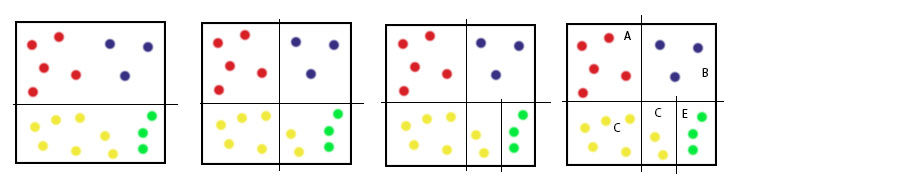
Obrázok 14 Príklad rozhodovacieho stromu pre klasifikáciu cicavcov [6]

Klasifikačný proces pri rozhodovacích stromoch je jednoduchý a priamy. Keď už máme vybudovaný rozhodovací strom začíname v koreni a postupujeme podľa podmienok z jedného uzla do ďalšieho, až kým neprídeme do Terminálu. V termináli  zistíme triedu, ku ktorej sa objekt klasifikuje.

* + 1. Trénovanie

Vytvoriť rozhodovací strom resp. natrénovať takýto strom, znamená rekurzívne delenie trénovacích dát podľa testu typu áno/nie na príznakové vektory. Toto delenie prebieha dovtedy, kým každá trieda obsahuje jej typické príznaky. Potom podľa týchto typických čŕt možno overiť, o ktorú triedu sa jedná.

* Prvé delenie je založené na osovom zarovnaní. Možno vidieť, že sa príznaky delia podľa čiar, ktoré sú rovnobežné s osami. Threshold sa robí na jediný príznak pre každý uzol. Tento prístup je veľmi rýchly na vyhodnotenie.



Obrázok 15 Delenie príznakov

* Druhé delenie sa nazýva všeobecné rovinné delenie. V tomto delení sa neopiera o osi, ale delí sa podľa toho, ako sa to hodí. Čiže deliť sa môže osami, ktoré nie sú rovnomerné s osou. Threshold sa robí s lineárnou kombináciou príznakov. Ide o náročnejší prístup, ale vygenerujú sa menšie stromy. [6]



Obrázok 16 Všeobecné rovinné delenie

* + 1. Výhody a nevýhody rozhodovacích stromov 

Výhody RS:

* Trénovanie môže byť veľmi jednoduché na implementáciu
* Jednoducho zvláda veľké množstvo vstupných premenných

Nevýhody RS:

* Je jasné, že vždy sa dá vytvoriť rozhodovací strom, ktorý má 100% úspešnosť na trénovacej množine, ale majú tendenciu byť veľmi veľké a negeneralizovať veľmi dobre.

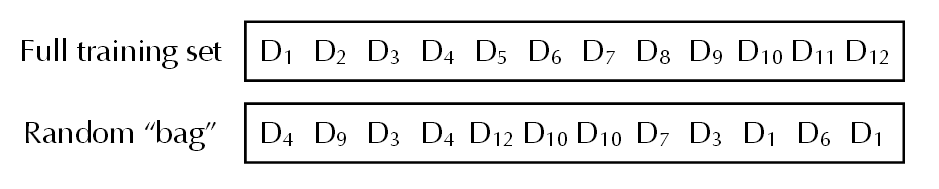
Aby sa predišlo týmto nevýhodám, boli vytvorené tzv. Náhodné lesy (Random Forests).[6]

* 1. Náhodné lesy

Základná myšlienka pri vytvorení tejto štruktúry, bola snaha zakomponovať náhodu do klasifikácie. [3] Táto náhoda bola pridaná do procesu stromového učenia. Keďže každý takýto strom obsahuje prvok náhody, vytvára sa ich viac a všetky sú založené na trénovacej množine. Po tom ako tento klasifikátor dostane vstup, každý strom dospeje k nejakému výstupu. Keďže tieto výstupy nie sú rovnaké, každý strom má hlas. S týmto hlasom prispeje ku konečnému výstupu rozpoznaného objektu. Ak sú stromy naozaj nezávisle, výkon sa zvýši s použitím viacerých stromov.[5]

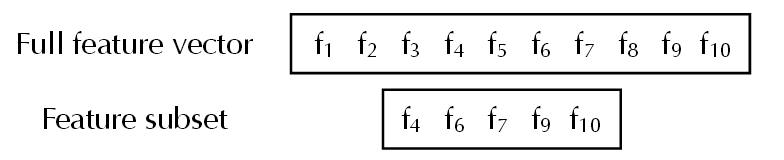
Pri vytváraní sa používajú tieto metódy:

* Bagging (balenie) – vytvorenie novej trénovacej množiny (bag), použitím pôvodnej trénovacej množiny. Z pôvodnej trénovacej množiny sa náhodné vyberú prvky(tieto prvky sa môžu aj opakovať – bootstrap sampling).



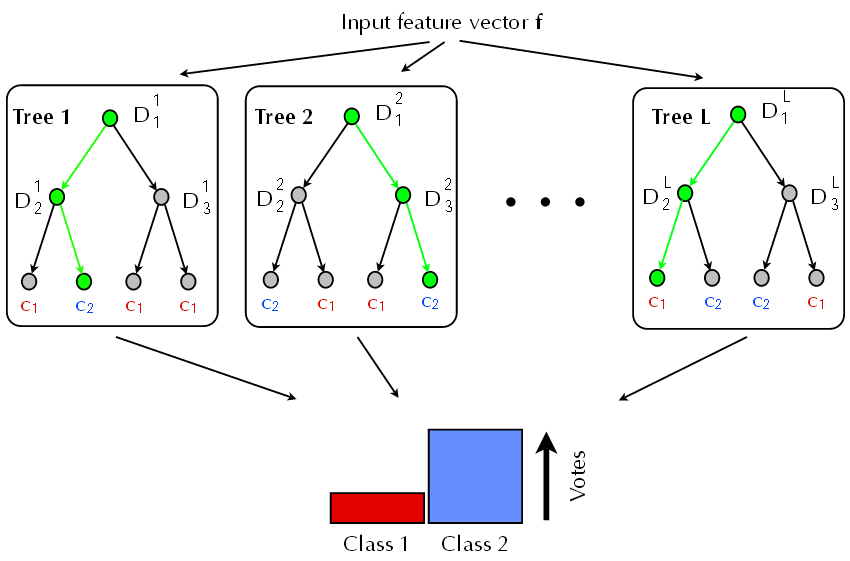
Obrázok 17 Výber prvkov pri bagging [6]

* Feature subset selection (Príznaková podmnožina výberu) – Z celkového príznakového vektora, sa vyberú náhodne podmnožiny príznakov na generovanie každej triedy. Keďže ide o podmnožiny, počet prvkov môže byť menší.



Obrázok 18 Výber príznakov [6]

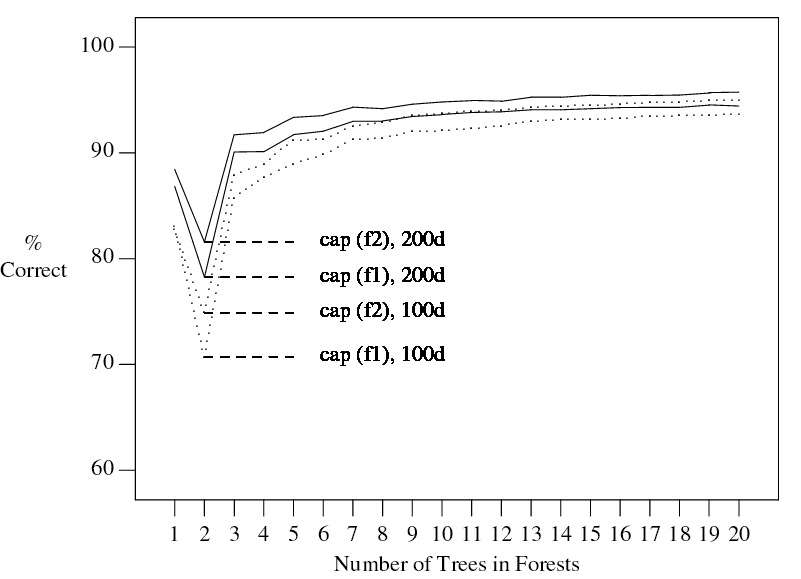
Na nasledujúcom obrázku je zobrazený príklad rozpoznávania objektu podľa vstupného vektora príznakov f. Každý strom ponúkne svoju odpoveď a na konci sa hlasy spočítajú. Podľa toho, ktorá trieda získa najväčší počet hlasov, za tú bude považovaný rozpoznávaný objekt. Na obrázku dostala najviac hlasov trieda Class 2 a tak neznámy objekt bude považovaný za Class 2:



Obrázok Hlasovanie stromov [6]

* + 1. Porovnanie výkonu pri použití viacerých stromov

V príklade z článku [6] boli ručne písané čísla takže išlo o 10 tried. Čísla boli o veľkosti 20x20 pixelov. Trénovacia množina obsahovala 60000 prvkov a testovacia množina 10000. Množina príznakov f1 obsahuje 400 príznakov a f2 obsahuje množinu f1 + gradient a to spolu tvorí 852 príznakov. Každý zo stromov obsahuje celú trénovaciu množinu (nepoužilo sa balenie). Porovnávajú sa rozdielne príznakové podmnožiny (100 a 200 príznakov):



Obrázok Porovnanie použitia viacerých stromov [6]

Z grafu možno vidieť, že skutočne počet stromov do istej miery zvýšil efektívnosť. Miestami je táto efektívnosť zvýšená až o 10%. [6]

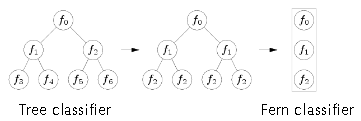
* + 1. Porovnanie náhodných lesov a fern-ov

Lesy

* Rozhodovacie stromy sa hneď naučia pravdepodobnostný posterior
* Aplikujú sa rozdielne sekvencie testov pre každý uzol
* Trénovací čas rastie exponenciálne s hĺbkou stromu

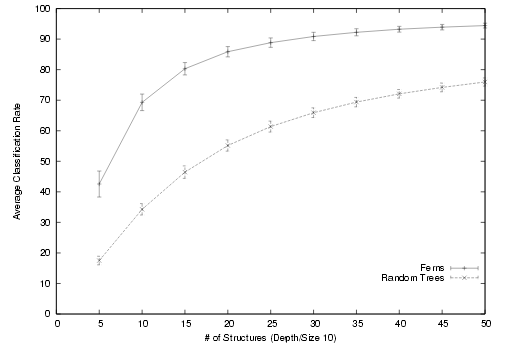
Ferny

* Učenie sa distribúciou trieda-podmienka
* Aplikujú sa rovnaké sekvencie testov, pre každý vstupný vektor
* Trénovací čas rastie lineárne s veľkosťou Fernu S
* Kombinuje hypotézy použitím Bayesovho pravidla (násobenie)[6]



Obrázok 21 Porovnanie stromov a fernov [6]

Pri používaní viacerých stromov sa zvýši miera správneho rozpoznávania a tak isto to je aj pri fern-och. Toto možno vidieť na nasledujúcom grafe.



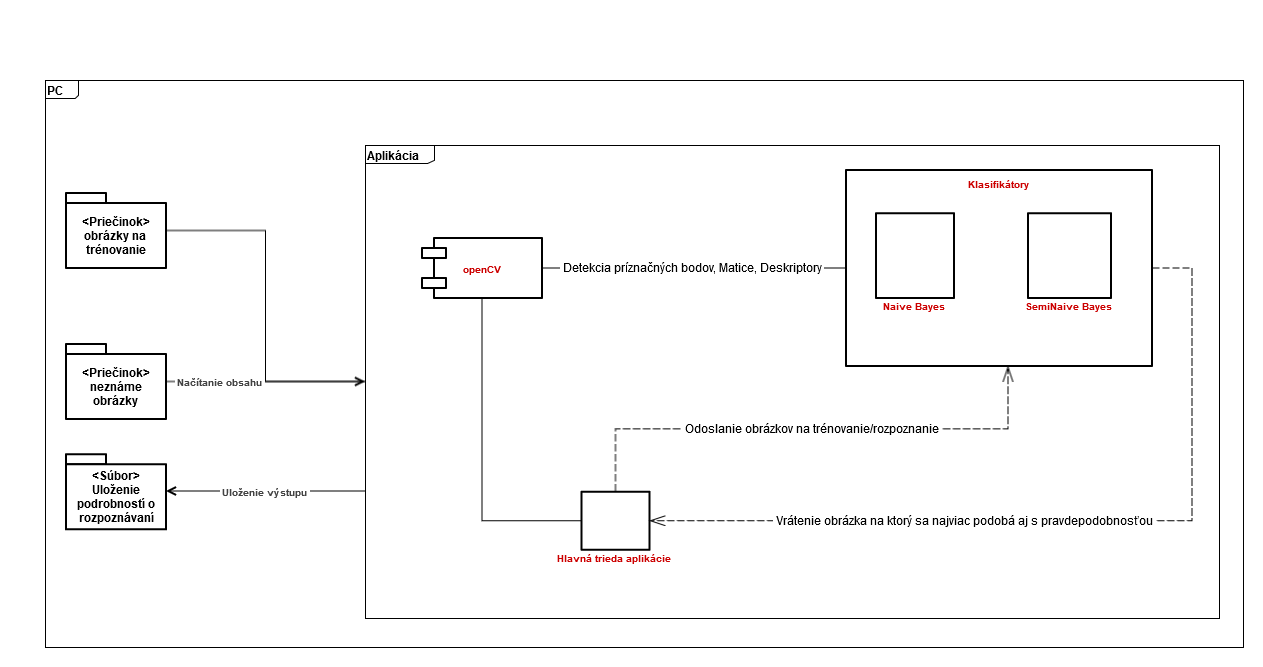
Obrázok 22 Porovnanie efektívnosti stromov a fernov[7]

1. Návrh

Vyvíjaná aplikácia bude desktopová. Na klasifikáciu sa použijú klasifikátory Naive Bayes a Semi Naive Bayes vynajdený Vincentom Lepetitom. Aplikácia bude používať knižnicu určenú na prácu s grafikou. Postup klasifikácie bude nasledujúci:

* Extrakcia význačných bodov obrázkov z trénovacej množiny
* Spustenie trénovania
* Extrakcia význačných bodov z neznámeho obrázka
* Spustenie rozpoznávania
* Získajú sa časy a úspešnosti rozpoznávania, ktoré sa uložia do súboru

Množina pre rozpoznávanie bude tvorená obrázkami, ktoré budú odfotené z rôznych pozícií a budú uložené v priečinku, tak isto budu obrázky na trénovanie uložené v súbore. Pre lepšie predstavenie je vytvorený následujúci obrázok:



Obrázok 23 Hrubý návrh aplikácie

1. Implementácia

Prvotná aplikácia sa začala implementovať v jazyku Java, no neskôr sa prešlo do jazyka C++ z následujúcich dôvodov:

* Jazyk C++ je rýchlejší
* Neúplná dokumentácia knižnice OpenCV pre jazyk Java

Po finálnom výbere vývojového jazyka sa prešlo k implementácii. Prostredie IDE pre vývoj je Microsoft Visual Studio 2012.

Pri implementácií sa použila knižnica OpenCV verzie 2.4.9. V samotnej implementácií sú vytvorené dva klasifikátory:

* Semi Naive Bayesov klasifikátor za použitia fernovej dátovej štruktúry
* Gaussian Naive Bayesov klasifikátor
  1. Postup implementácie

Ako prvé sme si vybrali jednoduché obrázky, ktoré sú súčasťou trénovacej množiny:



Obrázok 4 Obrázky z trénovacej množiny

Prvým procesom v aplikácií je načítanie všetkých obrázkov, ktoré spadajú do trénovacej množiny. Tie sú prevedené do čiernobieleho formátu, keďže farba nie je podstatná. Na detekciu význačných bodov sme použili detektor FAST. Ten zdetegoval všetky význačné body a následne sa extrahovali deskriptory. Na extrakciu deskriptorov sme použili metódu FREAK, ktorá podľa analýzy binárnych deskriptorov str.4 ma výborné výsledky. Presnosť tohto deskriptora podľa testov str.10 sa približuje k deskriptoru SIFT a má veľmi dobré výsledky pri práci s rozpoznávaním tlačených objektov.

Po detekcií a extrakcií význačných bodov sme zistili, že v knižnici OpenCV sa binárne reťazce rozpoznaných význačných bodov prepočítavajú do dekadickej sústavy. Tieto čísla sú 9 bitové. Vo svojej práci chceme pre Semi-Naivný Bayesov klasifikátor a Naivný Bayesov otestovať rôzne veľkosti fernov a jej dopad na výsledky rozpoznávania význačných bodov. Z tohto dôvodu sme prepočítali tieto dekadické reťazce na binárne a z nich sme vyberali počet bitov podľa veľkosti fernov, ktoré chceme testovať.

* + 1. Semi-Naive Bayes podľa článku [4]
    2. Inicializácia

Implementácia semi-naive bayes klasifikátora podľa autora V. Lepetita, pozostáva z inicializácie, trénovania a samotného rozpoznávania. Ako prvý je proces inicializácie, v ktorom sa vytvorí trojrozmerný vektor a inicializuje sa 40 náhodných hodnôt od 8 (0+veľkosť fernu) po 504 (512-veľkosť fernu). Horná hranica 504 respektívne 512 je určená maximálnou veľkosťou deskriptora. Vytvorené náhodné hodnoty budú ďalej slúžiť na výber fernov z binárnych reťazcov, o ktorých je písané v sekcii trénovania. Veľkosť prvého rozmeru trojrozmerného vektora je daný počtom obrázkov na ktorých sa bude klasifikátor trénovať. Druhý rozmer závisí od počtu význačných bodov na trénovacom obrázku. Tretí rozmer závisi od konfigurácie klasifikátora a je určený počtom fernov PF a veľkosťou fernov VF. Jeho hodnota je teda daná súčinom:

Hodnota pri veľkosti VF=8 a PF = 40 je 20480 čo je aj štandardná konfigurácia klasifikátora.

* + 1. Trénovanie klasifikátora

Ako ďalší krok je trénovanie klasifikátora. Na trénovanie sa prejdú obrázky z trénovacej množiny, ktorých absolútne cesty boli uložené do spájaného zoznamu štruktúry *ImageInformation* v poli Name. Táto štruktúra obsahuje v sebe aj význačne body, deskriptory, maticu a poradové číslo obrázka. Ako sa tento zoznam prechádza, posielajú sa jednotlivé obrázky na trénovanie. Metóda na trénovanie je train, ktorej argumenty sú matica obrázka a číslo obrázka. Na túto vstupnú maticu je vykonaných 50 deformácií – rotácií metódou rotate. Obrázok sa zrotuje o náhodnú hodnotu od 1 – 360 pričom sa význačné body zrotujú o rovnaký uhol. Takže pri rotácií sa spolu s obrázkom rotujú aj význačné body. Po rotácií sa vypočítajú nové deskriptory zrotovaných význačných bodov obrázka. Každý deskriptor je prevedený na 512 bitový binárny reťazec. Následuje cyklus, ktorý je ohraničený počtom fernov v našom prípade 40. Z poľa naplneného náhodnými číslami sa vyberajú hodnoty a podľa vybranej hodnoty sa z binárneho reťazca deskriptora vyberú bity s pozíciou hodnoty. Po výbere n-bitov (n je veľkosť fernu) sa tieto bity prevedú na dekádicke číslo, ktoré tvoria hodnotu fernu. Jednotlivé indexy trojrozmerného poľa sú: [číslo obrázka] [číslo deskriptora] [hodnota fernu] pričom posledná dimenzia vektora je hodnota fernu vynásobená indexom z vektora náhodných čísel a maximalnou veľkosťou fernu (pri veľkosti fernu 8 je to 2^8 – 256). Táto pozícia v trojrozmernom vektore sa inkrementuje. Po inkrementácií sa opakuje proces pre ďalši vybratý deskriptor a po prejdení všetkých deskriptorov sa vezme ďalší zrotovaný obrázok a celý cyklus sa zopakuje. Po vykonaní všetkých rotácií je klasifikátor natrénovaný na rozpoznávanie obrázka.

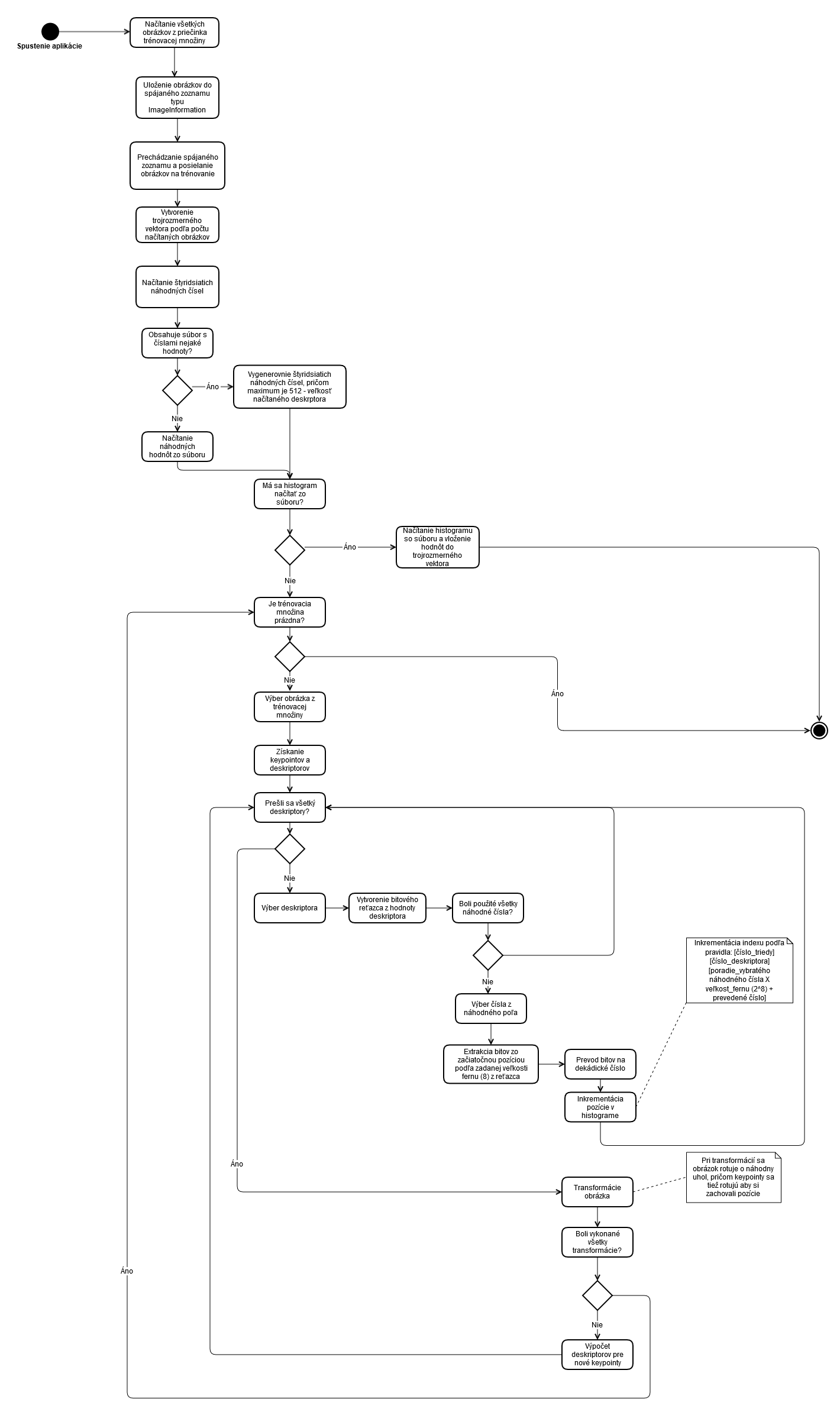
Výstupom trénovania klasifikátora je teda trojrozmerný vektor, ktorého hodnoty sú určené fernami. Hodnoty získane pri trénovaní sú uložené do textového súboru aby sa trénovanie nemuselo stále zapínať ak je vstupná trénovacia množina rovnaká. Následujúci diagram znázorňuje inicializáciu a tréning:                                       

Diagram Inicializácia a trénovanie Semi-Naive Bayes klasifikátora

* + 1. Rozpoznávanie

Po procese natrénovania klasifikátora je klasifikátor pripravený rozpoznávať neznáme obrázky. Rozpoznávanie je pomocou metódy recognize, ktorá je súčasťou triedy klasifikátora. Vstupom je obrázok, prázdny ukazovaťel na integer hodnotu, ktorá bude naplnená rozpoznaným číslom obrázka, príznak či sa má trénovacia množina načítať zo súboru a ukazovaťel na float hodnotu, ktorá bude určovať pravdepodobnosť správneho výsledku.

Následujúci cyklus závisi od počtu deskriptorov keďže nastáva rozpoznávanie neznámych význačných bodov . Vyberie sa deskriptor z matice a ten sa prekonvertuje na binárny reťazec. Následne sa prechádzajú všetky deskriptory, ktoré boli natrénovane počas učenia klasifikátora do trojrozmerného poľa. Z vybraného neznámeho deskriptora sa vyberú znova ferny pomocou vektora neznámych čísel. A podľa hodnôt fernov sa spočítajú hodnoty trojrozmerného poľa pričom index určí práve táto hodnota fernu. Čísla vytiahnuté z trojrozmerného vektora sa sčítajú a tento súčet je vydelený maximálnou hodnotu, ktorú môže súčet dosiahnúť(počet fernov\* počet rotácií ). Každý deskriptor je takto ohodnotený a ako výsledný je ten, ktorý dosiahne najvyššiu hodnotu. Všetky informácie o už rozpoznanom význačnom bode sú uložené vo vektore - resultVect, ktorého veľkosť je počet deskriptorov neznámeho význačného bodu a má 3 hodnoty :

* rozpoznaná trieda
* rozpoznaný deskriptor
* pravdepodobnosť

Pri rozpoznávaní o aký obrázok ide, sa prechádza premenná resultVect a do pomocného vektora, ktorého veľkosť je počet tried a robí sa súčet pravdepodobností pre každú triedu. Z týchto pravdepodobností sa vytvorí priemerná hodnota a najvyššia priemerná hodnota určí neznámy obrázok. Proces rozpoznávania je uvedený na následujúcom diagrame:

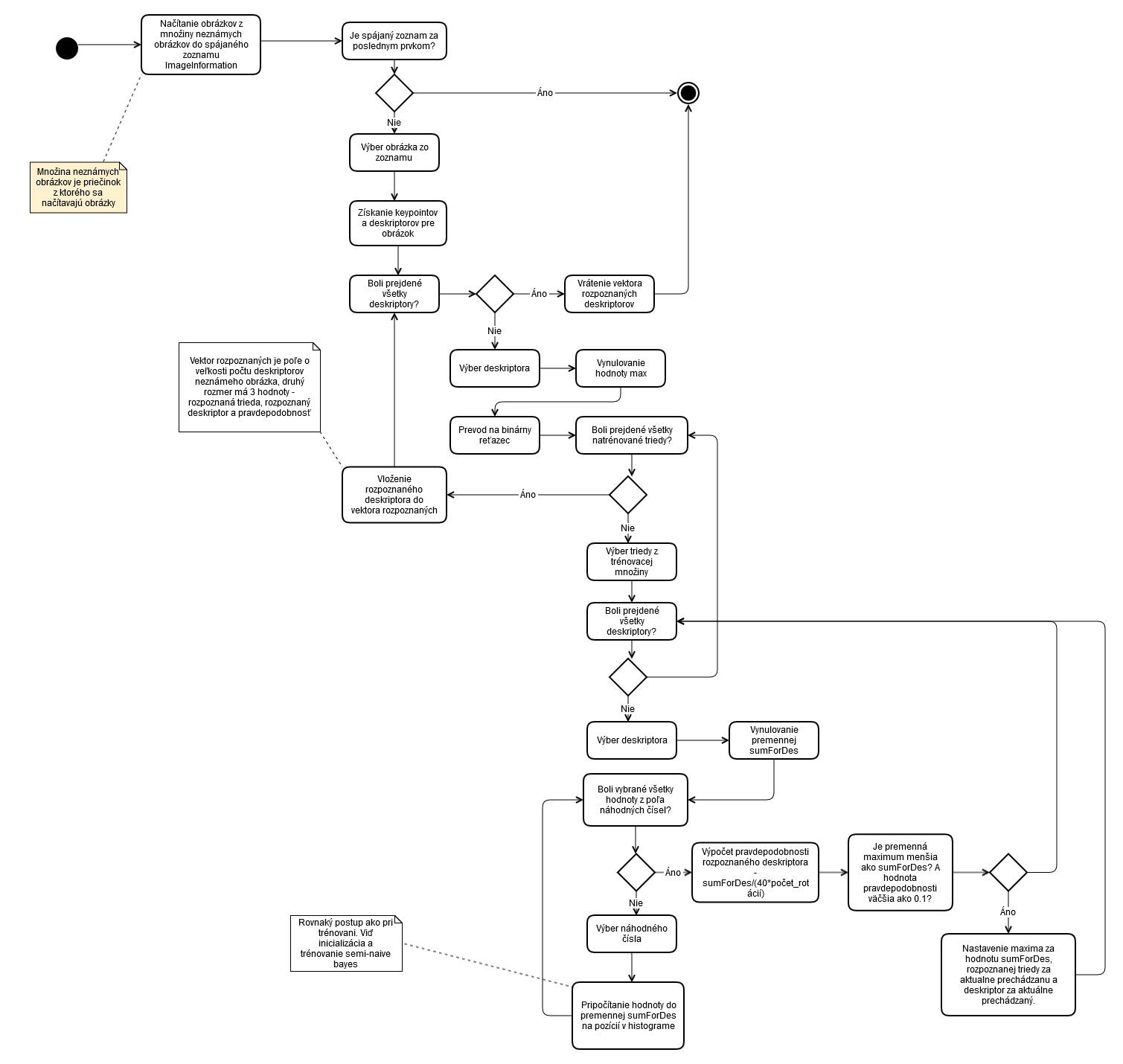


Diagram Rozpoznávanie pomocou Semi-Naive Bayes klasifikátora

* + 1. Naive-Bayes

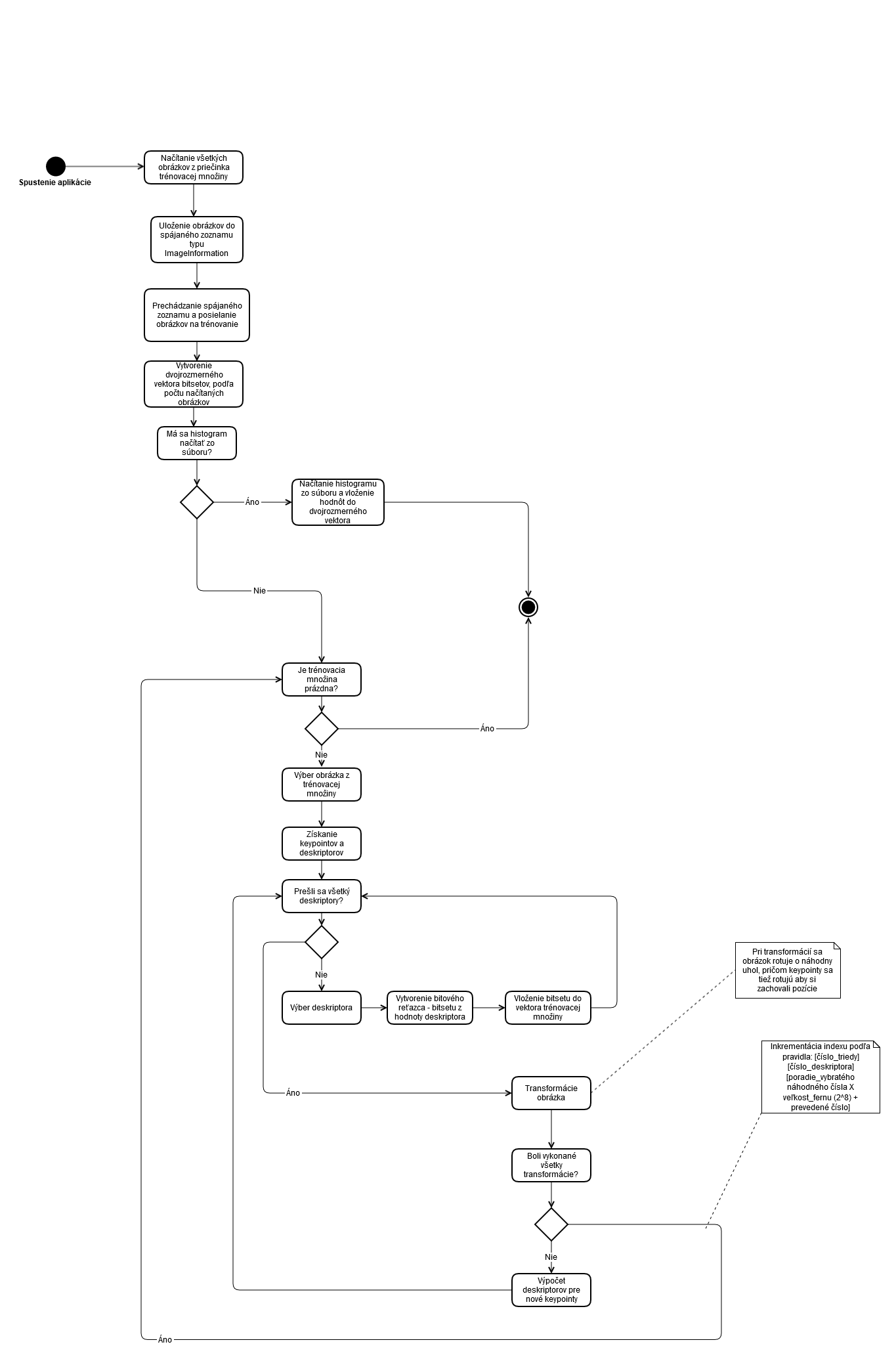
Inicializácia klasifikátora pracujúceho na princípe Naive-Bayes sa líši od pôvodného klasifikátora v tom, že pole do ktorého sa vkladajú natrénované hodnoty je dvojrozmerné a obsahuje bitové reťazce (bitsety). Pri trénovaní sa z obrázkov trénovacej množiny získavajú deskriptory reprezentované binárnymi reťazcami, ktoré sa vkladajú do spomínaného vektora. Prvý rozmer určuje triedu – obrázok a druhý deskriptor. Pri rotácií sa nové bitsety pridávajú do poľa a po modulácií poradového čísla vloženého deskriptora hodnotou počtu všetkých deskriptorov pre daný obrázok, sa získa skutočná hodnota deskriptora. Po ukončení všetkých rotácií a vkladaní do poľa je klasifikátor pripravený na rozpoznávanie.                               

Diagram Inicializácia a trénovanie Naive Bayes klasifikátora

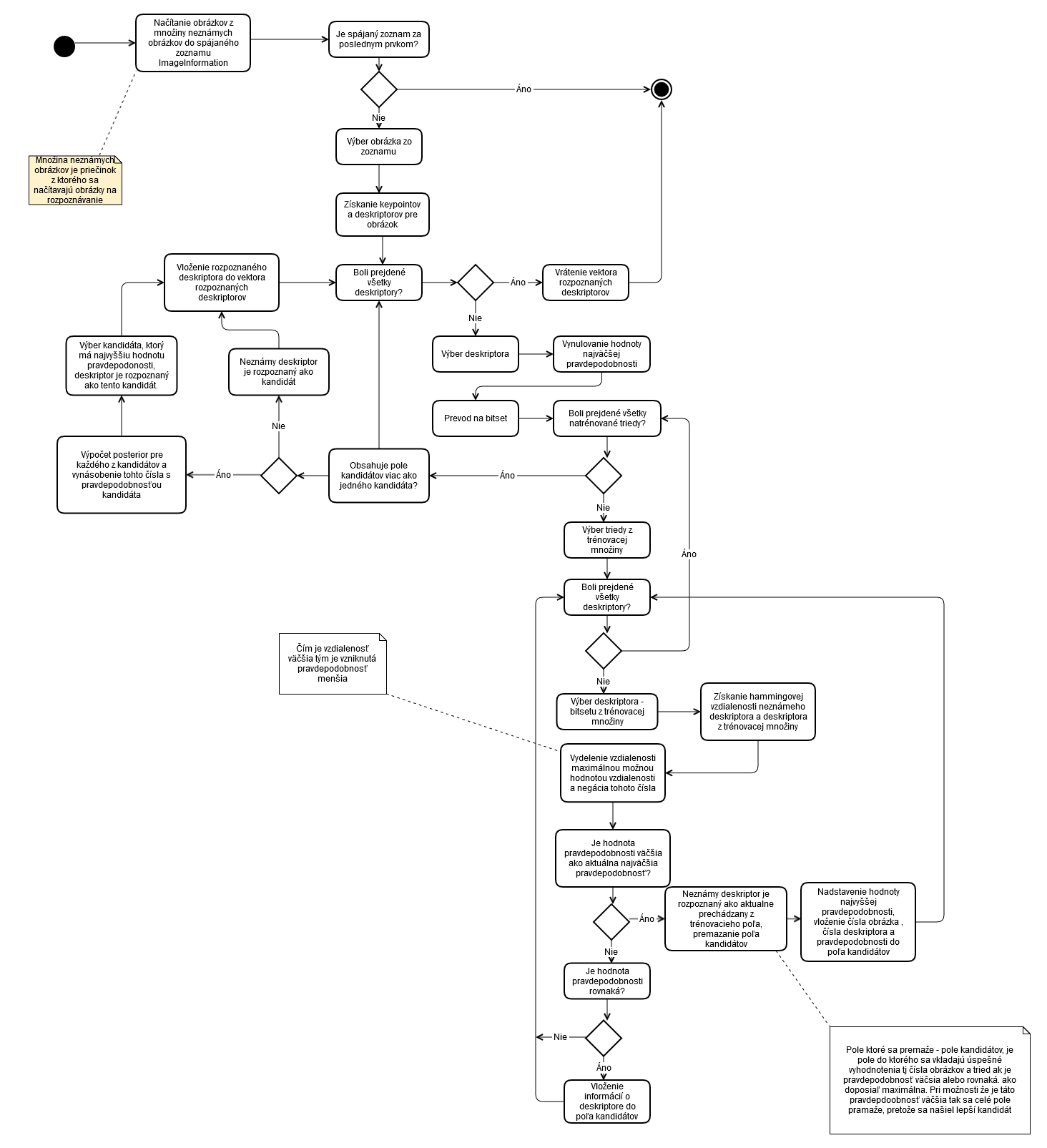
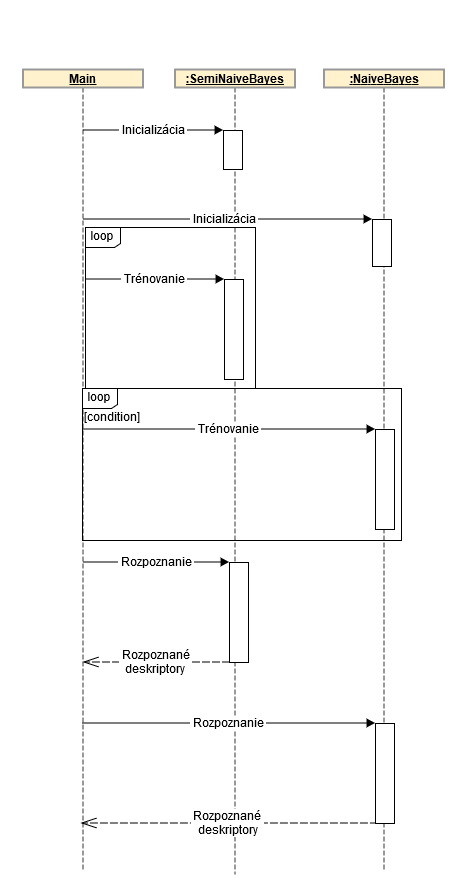
Pri rozpoznávaní obrázka sa prechádzajú všetky deskriptory neznámeho obrázka. Tieto deskriptory sa rozpoznávajú na základe vzdialenosti od jednotlivých deskriptorov v trénovacej množine. Vzdialenosť sa určuje podľa hammingovej vzdialenosti keďže ide o porovnávanie vzdialenosti binárnych čísel. Pri prechádzaní poľa trénovacej množiny môže byť vzdialenosť medzi deskriptormi menšia ako doposiaľ najmenšia vzdialenosť a to znamená že je nový lepší kandidát tj. deskriptor, ku ktorému je neznámy deskriptor bližšie. Veľkosti vzdialenosti sú normalizované na percentá. Tieto pravdepodobnosti sú vypočítané ako podiel aktuálnej vzdialenosti a maximálnej vzdialenosti. A keďže čím je vzdialenosť menšia tým má byť pravdepodobnosť väčšia, celá táto vypočítaná pravdepodobnosť je znegovaná. Ak je pravdepodobnosť rovnaká ako doposiaľ najväčšia pravdepodobnosť, je rozpoznaný kandidát vložený do poľa možných kandidátov. Po ukončení rozpoznávania neznámeho deskriptora a zistení že pole obsahuje viac než jedného kandidáta sa na základe výpočtu posteriora zistí, na ktorého z kandidátov sa viac podobá .                                     


Diagram Rozpoznanie pomocou Naive Bayes klasifikátora



Aplikácia sa skladá z troch základných tried. Sú to triedy Main, SemiNaiveBayes a NaiveBayes. Podľa názvov je už zrejmé, že SemiNaiveBayes je SemiNaiveBayes klasifikátor a NaiveBayes je NaiveBayes klasifikátor. Main je trieda kde sa nachádza základná metóda main ktorá stopuje čas pri výpočtoch každého  klasifikátora (trénovanie a rozpoznávanie). Taktiež táto trieda obsahuje metódy na načítanie všetkých obrázkov ako do trénovacej množiny tak aj tých, ktoré sú neznáme. Ako prvé sa vykonáva inicializácia a následne natrénovanie oboch klasifikátorov. V tomto prípade sa spúšta stopovanie času. Po úspešnom natrénovaní oboch klasifikátorov sa časy uložia do súboru. Pred samotným rozpoznávaním sa načítajú všetky neznáme obrázky a postupne sa posielajú na rozpoznávanie. Najskôr do SemiNave Bayes klasifikátora a potom do NaiveBayes klasifikátora. Aby bol klasifikátor invariantný aj voči škálovaniu, pri rozpoznávaní sa používa pyramídová technika. Na rozpoznávanie sa posiela najskôr 1/5 pôvodného obrázka, následne 1/4 až po 1/1 čiže pôvodný obrázok. Neznámy obrázok je potom vyhodnotený ako ten, ktorého pravdepodobnosť dosiahla najvyššiu hodnotu. Po rozpoznaní obrázka sa tieto hodnoty uložia do textového súboru, spolu s časmi trvania rozpoznávania. Základná trieda main obsahuje aj metódu showHomography, ktorá zobrazí zhodné význačné body medzi neznámym a rozpoznaným obrázkom. Aby bolo lepšie vidieť kde aplikácia rozpoznala obrázok, na fotke je vytvorené orámovanie okolo rozpoznaného obrázka. Na toto orámovanie sa používa openCV metóda findHomography s príznakom CV\_RANSAC.

V ďalšej kapitole sa budeme venovať testovaniu úspešnosti samotných klasifikátorov.

1. Testovanie

Neznáme obrázky, na ktorých sa bude overovať úspešnosť jednotlivých klasifikátorov, budú vytvorené pomocou mobilného zariadenia Nokia Lumia 920, ktorý disponuje 9 Mpx fotoaparátom. Obrázky budú vytlačené na papier formátu A4 a následne odfotené z rôznych uhlov a pri rôznych natočeniach v teréne.

Na testovanie boli použité obrázky s následujúcimi parametrami:

* Philadelphia 806x541px [32 vzoriek] – 70 význačných bodov
* Topánka 500x500px [53 vzoriek] – 232 význačných bodov
* Erb Tvrdošína 225x225px [27 vzoriek] – 156 význačných bodov

Obrázkov na ktorých prebehali testy bolo 122, veľkosť fernov 8.

Celková správnosť rozpoznávania bola pre Semi-Naive Bayes klasifikátor 41,34% a Naive Bayes 52.98%.

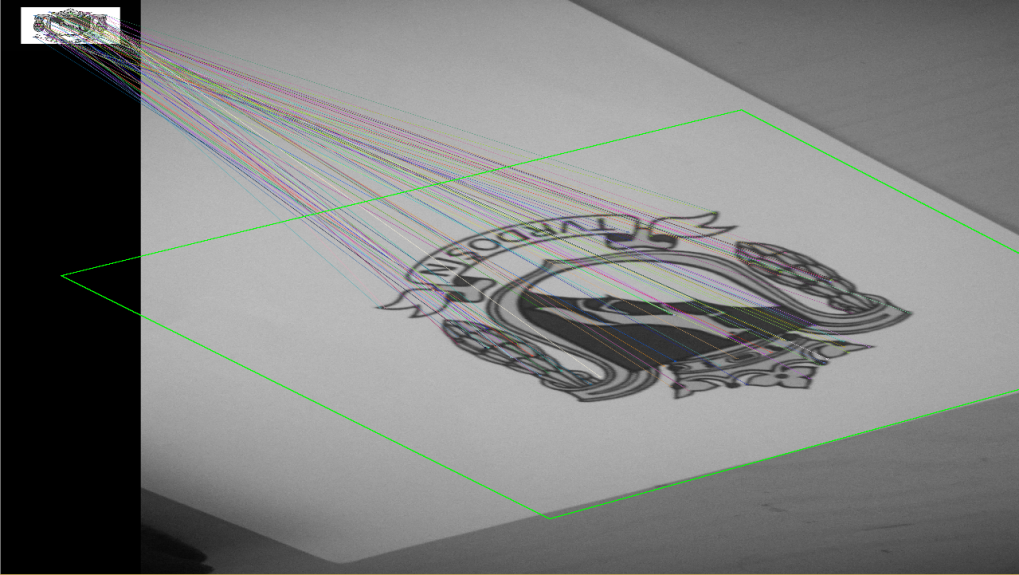
Úspešnosti pre jednotlivé obrázky pre Semi-Naive Bayes a NaiveBayes klasifikátor:

Hodnoty úspešnosti sa hýbali okolo 50%,okrem erbu mesta tvrdošín, ktorý získal veĺmi nízke hodnoty. Semi Naive Bayes mal výsledky rozpoznávania topánky lepšie ako Naive Bayes. Rozpoznávanie znaku philadelphie bolo lepšie s klasifikátorom NaiveBayes, ktorý získaval veľmi vysoké hodnoty.

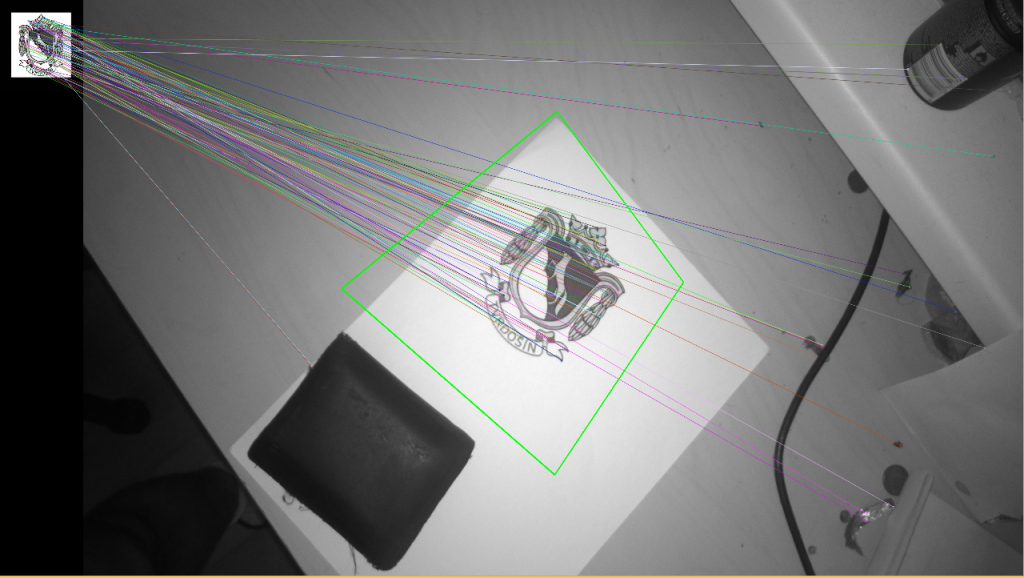
Výsledky rozpoznávania neznámych obrázkov pre SemiNaive Bayes klasifikátor:

Výsledky rozpoznávania neznámych obrázkov pre Naive Bayes klasifikátor:

Hoci bola úspešnosť erbu mesta Tvrdošín zlá, existujú aj prípady kedy bol tento obrázok úspešne klasifikovaný:



Obrázok 25 Rozpoznanie erbu tvrdošín prostredníctvom Semi-Naive Bayes klasifikátora



Obrázok 26 Rozpoznanie erbu tvrdošín prostredníctvom Naive Bayes klasifikátora

Úspešne rozpoznanie loga philadelphia:

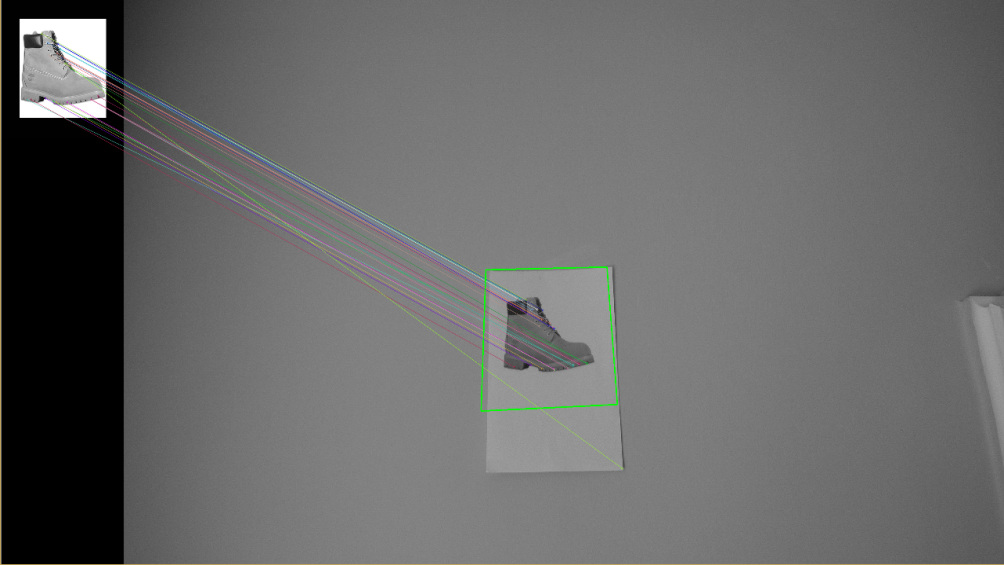


Obrázok 27 Rozpoznanie loga philadelphie prostredníctvom Semi-Naive Bayes klasifikátora



Obrázok 28 Rozpoznanie loga philadelphie prostredníctvom Naive Bayes klasifikátora

Úspešne rozpoznanie obrázka topánky:



Obrázok 29 Rozpoznanie fotky topánky prostredníctvom Semi-Naive Bayes klasifikátora



Obrázok 30 Rozpoznanie fotky topánky prostredníctvom Naive Bayes klasifikátora

Rozpoznávanie každého klasifikátora bolo odlišné. Ako v grafoch možno vidieť Semi-Naive Bayes dosahoval lepšie výsledky na obrázku topánky, ktorá mala viac význačných bodov ako ostatné obrázky. Na druhej strane Naive-Bayes mal lepšie výsledky pri obrázku Philadelphie, ktorá mala najmenej význačných bodov. No žiaden klasifikátor nemal úspešnosť rozpoznávania erbu mesta Tvrdošín nad 15%. Oba klasifikátory dosahovali výsledky pomerne rovnaké avšak Naivný Bayes o zopár percent viac. Táto neschopnosť klasifikácie môže byť spôsobená tým, že na vstupe nebolo rozpoznaných dosť význačných bodov, ktoré by boli súčasťou obrázka. Alebo keďže obrázok Tvrdošína je pomerne komplikovaného tvaru, a má rôzne drobné časti, nemuseli byť nájdené relevantné príznačné body, na ktoré boli klasifikátory trénované. Význačné body mohli vzniknúť na iných miestach. Taktiež kvalita záberov nebola až na tak vysokej úrovni a tvoril sa šum.

Tvarovo jednoduché obrázky nepodliehajú takýmto deformáciam a preto ich úspešnosť pri rozpoznávaní je oveľa vyššia. . Po týchto vyhodnoteniach možno povedať že Naive Bayes klasifikátor pre zvolené trénovacie obrázky a vytvorené testovacie obrázky dosahuje úspešnosť, ktorá je vyššia a tým pádom prináša lepšie výsledky.

1. Záver

Cieľom práce bolo preskúmanie témy binárnych deskriptorov a ich klasifikácia s následnou aplikáciou týchto získaných vedomostí, na vytvorenie klasifikačnej aplikácie resp aplikácie, ktorá používa dva klasifikátory.

Na začiatku práce jej vytvorená analýza samotných deskriptorov od všeobecného pojmu binárne deskriptory, opísanie ich typov a zhodnotenie ich úspešnosti podľa vedeckého článku. Táto analýza nám napomohla pri výbere deskriptora, s ktorým sa bude pracovať a testovať samotná klasifikácia. Po analýze tých deskriptorov sa analýza sústredila na samotné klasifikátory. Bolo uvedených viacero klasifikátorov spolu s ich výhodami/nevýhodami, využitím a  princípom na ktorých sú založené. Táto analýza nas priviedla k tomu, že boli vybraté dva klasifikátory, na ktorých implementácií sa počas semestra pracovalo.

Postup tejto implementacie, spolu s jej predimplementačným návrhom je obsiahnutý v kapitolách návrh a implementácia. V implementácií sme sa sústredili na opis konkrétneho riešenia s použitím vybraných klasifikátorov. Riešenie je zobrazené aj pomocou diagramov, ktoré jasne zobrazujú logickú šturktúru implementovaných klasifikátorov. Po tejto implementácií následuje kapitola, ktorá sa sústredí na testovanie implementovaných klasifikátorov, ktoré boli natrénované pomocou binárnych deskriptorov.

Ako samotný výstup tejto práce je teda kapitola testovanie, kde sú opísané výsledky testovania týchto klasifikátorov.

1. Literatúra

[1] G. Bradski and A. Kaehler, *Learning OpenCV*. .

[2] S. J. Delany, “k -Nearest Neighbour Classifiers,” pp. 1–17, 2007.

[3] K. P. Murphy, “Naive Bayes classifiers Generative classifiers,” pp. 1–8, 2006.

[4] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, “Fast keypoint recognition using random ferns.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 448–461, 2010.

[5] A. Bosch, A. Zisserman, and X. Muoz, “Image Classification using Random Forests and Ferns,” *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. pp. 1–8, 2007.

[6] D. Capel, “Random Forests and Ferns The Multi-class Classification Problem.”

[7] M. Ozuysal, P. Fua, and V. Lepetit, “Fast Keypoint Recognition in Ten Lines of Code,” *2007 IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007.

[8] John, George H.; Langley, Pat (1995). "*Estimating Continuous Distributions in Bayesian Classifiers*". Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence. Morgan Kaufmann. pp. 338–345.

[9] Metsis, Vangelis; Androutsopoulos, Ion; Paliouras, Georgios (2006). "*Spam filtering with Naive Bayes—which Naive Bayes*?". Third conference on email and anti-spam (CEAS) 17.

[10] Krystian Mikolajczyk and Cordelia Schmid, “A performance evaluation of local descriptors,”IEEE Transac-tions on Pattern Analysis and Machine Intelligence, vol.27, no. 10, pp. 1615–1630, 2005.

[11] Vijay R. Chandrasekhar, David M. Chen, Sam S.Tsai, Ngai-Man Cheung, Huizhong Chen, Gabriel Takacs, Yuriy Reznik, Ramakrishna Vedantham, Radek Grzeszczuk, Jeff Bach, and Bernd Girod, “The stanford mobile visual search data set,” in Proceedings of the second annual ACM conference on Multimedia systems, New York, NY, USA, 2011, MMSys ’11, pp. 117–122, ACM.

[12] Tola, Engin, Vincent Lepetit, and Pascal Fua. “A fast local descriptor for dense matching.” Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008.‏

[13] Leutenegger, Stefan, Margarita Chli, and Roland Y. Siegwart. “BRISK: Binary robust invariant scalable keypoints.” Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011.

[14] Rublee, Ethan, et al. “ORB: an efficient alternative to SIFT or SURF.” Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011.‏‏

[15] Alahi, Alexandre, Raphael Ortiz, and Pierre Vandergheynst. “Freak: Fast retina keypoint.” Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012.‏

[16] Calonder, Michael, et al. “Brief: Binary robust independent elementary features.” Computer Vision–ECCV 2010. Springer Berlin Heidelberg, 2010. 778-792.‏

[17] Bekele, D., Teutsch, M., & Schuchert, T. (2013). Evaluation of binary keypoint descriptors. *2013 IEEE International Conference on Image Processing*, 3652–3656. doi:10.1109/ICIP.2013.6738753

[18] Webová lokalita

<http://cvlab.epfl.ch/research/detect/binarydescriptors> (pristupované 13.3.2015)

1. Použivateľská príručka

Vytvorený program slúži na klasifikáciu obrázkov pomocou dvoch odlišných klasifikátorov ako boli v predošlích kapitolách spomenuté. Celá aplikácia je desktopová bez grafickej nadstavby, keďže jej cieľ je otestovanie výkonu rozpoznávania.

* 1. Nadstavenie vstupných parametrov pre triedu Main

V základnej triede Main.cpp sa nachádzajú v metóde main() dve premenné pathPrefix a trainingPath. Premenná **pathPrefix** je absolútna cesta k priečinku v ktorom sa nachádzajú neznáme obrázky tj. obrázky, ktoré sa budu klasifikovať. Druhá premenná **trainingPath** je absolútna cesta k priečinku kde sa nachádzajú obrázky na trénovanie klasifikátorov. V tomto priečinku musia byť jednotlivé obrázky v separátnych priečinkoch. Nad týmito lokalizačnými premennými sa nachádza globálna premenná **fout**, ktorá obsahuje názov súboru, do ktorého sa budú zapisovať všetky výsledky. Ak názov zadaného súboru nebude existovať, vytvorí sa. No ak už existovať bude, prepíše sa jeho obsah.

Po nadstavení všetkých ciest si aplikácia uloží všetky obrázky do spájaných zoznamov a automaticky začne spúštať metódy na trénovanie a následnu klasifikáciu. Ak klasifikátor už vytvoril trénovací model, je možné jeho hodnoty z jeho súborov načítať pomocou nadstavenia príznaku **FileLoader** na hodnotu 1.

Po spustení aplikácie si môže používateľ overiť v súbore, ktorého názov zadal výsledky. Tento súbor obsahuje časi trénovania oboch klasifikátorov v sekundách. Rozpoznávaný obrázok resp Absolútnu cestu k nemu, počet význačných bodov ktoré extraktor extrahoval, veľkosť obrázka vo formáte výška\*šírka, čas trvania rozpoznávania a výsledok rozpoznávania.

* 1. Špecifikácia parametrov klasifikátorov

SemiNaiveBayes

V prvej triede SemiNaiveBayes je možno tento klasifikátor upraviť tak, že sa niektore jeho parametre zmenia. Ide hlavne o typ deskriptora význačných bodov v premennej **DES\_MODE**. Tieto typy sú očíslované od 0 po 1, pričom platí že tieto čísla predstavujú následujúce deskriptory:

* BRIEF
* BRISK
* ORB
* FREAK

Ďalším z parametrom je umiestnenie súborov natrénovaného modelu a náhodných čísel. Premenná **TRAIN\_FILE** obsahuje názov resp cestu k súboru natrénovanému modelu pre klasifikátor SemiNaive Bayes a premenná **RANDOM\_FILE** obsahuje názov resp cestu k súboru s náhodnými číslami, ktoré sa využívajú pri trénovani a klasifikácií. Bližšie sa používajú na výber pozícií z binárnych reťazcov. Na učenie klasifikátora sa používajú rotácie originálneho obrázka. Na určenie počtu týchto rotácií treba zmeniť číslo v premennej **ROTATION\_COUNT**. Následujúce dva parametre sú tie, ktoré sa najviac týkajú fernov. Ide o veľkosť fernu v premennej **FERN\_SIZE** vyjadrujúca veľkosť fernu – počet bitov jednotlivých fernov. **FERN\_COUNT**  slúži na určenie počtu týchto fernov a podľa tohoto parametra sa vytvorí aj počet náhodných čísel v súbore s náhodnými číslami. Posledný parameter **MIN\_PROBABILITY** sa týka rozpoznávania a určuje aká musí byť minimálna pravdepodobnosť rozpoznaného význačneho bodu, aby bol tento bod validný.

NaiveBayes

V druhej triede Naive Bayes klasifikátora týchto parametrov nie je až v takom veľkom počte. Tú sú len parametre pre nadstavenie deskriptora, pričom typ nastavenia je ten istý ako v predchádzajúcom klasifikátore, v premennej **BDES\_MODE** a nastavenie počtu rotácií premmenou **BROTATION\_COUNT**.

Ako predvolené nastavenia premenných sú následujúce:

Pre klasifikátory:

* DES\_MODE = 4
* TRAIN\_FILE = „training\_semiNaive.txt“
* RANDOM\_FILE = „random\_numbers.txt“
* ROTATION\_COUNT = 40
* FERN\_SIZE = 10
* FERN\_COUNT = 40yg
* BROTATION\_COUNT = 40
* BDES\_MODE = 4
* MIN\_PROBABILITY = 80

Pre základnú triedu Main:

* pathPrefix = “C:\\Pictures\\Nezname\\”
* trainingPath = “C:\\Pictures\\PocitacoveVidenie\\”
* FileLoader = 0
* Fout = “out.txt”