Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

FIIT-5220-64336

Bc. Martin Geier

AUTOMATICKÁ ANOTÁCIA SÉMANTICKÉHO OPISU SCÉNY V OBRAZE

Diplomová práca

Vedúci práce: Ing. Vanda Benešová, PhD.

máj 2014

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
FIIT-5220-64336

Bc. Martin Geier

Automatická anotácia sémantického opisu scény v obraze

Diplomová práca

Študijný program: Softvérové inžinierstvo
Študijný odbor: Softvérové inžinierstvo
Miesto vypracovania: Ústav aplikovanej informatiky, FIIT STU, Bratislava
Vedúci práce: Ing. Vanda Benešová, PhD.
máj 2014

S T U
F I I T
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií

Zadanie diplomovej práce

Meno študenta: **Bc. Martin Geier**

Študijný program: Softvérové inžinierstvo

Študijný odbor: Softvérové inžinierstvo

Názov práce: **Automatická anotácia sémantického opisu scény v obraze s použitím normy MPEG 7**

Samostatnou výskumnou a vývojovou činnosťou v rámci predmetov Diplomový projekt I, II, III vypracujte diplomovú prácu na tému, vyjadrenú vyššie uvedeným názvom tak, aby ste dosiahli tieto ciele:

Všeobecný cieľ:

Vypracovaním diplomovej práce preukážte, ako ste si osvojili metódy a postupy riešenia relatívne rozsiahlych projektov, schopnosť samostatne a tvorivo riešiť zložité úlohy aj výskumného charakteru v súlade so súčasnými metódami a postupmi študovaného odboru využívanými v príslušnej oblasti a schopnosť samostatne, tvorivo a kriticky pristupovať k analýze možných riešení a k tvorbe modelov.

Špecifický cieľ:

Vytvorte riešenie, zodpovedúce návrhu textu zadania, ktorý je prílohou tohto zadania. Návrh bližšie opisuje tému vyjadrenú názvom. Tento opis je záväzný, má však rámcový charakter, aby vznikol dostatočný priestor pre Vašu tvorivosť.

Riaďte sa pokynmi Vášho vedúceho.

Pokiaľ v priebehu riešenia, opierajúc sa o hlbšie poznanie súčasného stavu v príslušnej oblasti alebo o priebežné výsledky Vášho riešenia alebo o iné závažné skutočnosti, dospejete spoločne s Vaším vedúcim k presvedčeniu, že niečo v texte zadania a/alebo v názve by sa malo zmeniť, navrhnite zmenu. Zmena je spravidla možná len pri dosiahnutí kontrolného bodu.

Miesto vypracovania: Ústav aplikovanej informatiky FIIT STU v Bratislave

Vedúci práce: **Ing. Vanda Benešová, PhD.**

Termíny odovzdania:

podľa harmonogramu štúdia platného pre semester, v ktorom máte príslušný predmet (Diplomový projekt I, II, III) absolvovať podľa Vášho študijného plánu

Predmety odovzdania:

V každom predmete dokument podľa pokynov na www.fiit.stuba.sk v časti:
home > Informácie o > štúdiu > organizácia štúdia > diplomový projekt

V Bratislave dňa 18. 2. 2013

prof. Ing. Pavol Návrat, PhD.
riaditeľ Ústavu informatiky a softvérového inžinierstva

STU FIIT
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce [1]

Študent:

Meno, priezvisko, tituly: Martin Geier, Bc.
Študijný program: Softvérové inžinierstvo
Kontakt: geier.xmartin@gmail.com

Výskumník:

Meno, priezvisko, tituly: Vanda Benešová, Ing. PhD.

Projekt:

Názov: Automatická anotácia sémantického opisu scény v obraze s použitím normy MPEG 7

Názov v angličtine: Automatic annotation of semantic description of the scene in the image using norm MPEG 7

Miesto vypracovania: Ústav aplikovanej informatiky, FIIT STU, Bratislava

Oblasť problematiky: Rozpoznávanie scény

Text návrhu zadania[2]

V dnešnej dobe stále rýchlejšie pribúda množstvo multimediálnych dát, v ktorých je veľmi náročné orientovať sa len na základe mena. Norma MPEG 7 je vytvorená práve pre opis obsahu multimediálnych dát, na základe ktorých je možné efektívne vyhľadávanie alebo porovnávanie obsahu. Základným problémom je však automatická anotácia týchto dát bez ľudskej spoluúčasti.

Analyzujte možnosti použitia deskriptorov navrhovaných normou MPEG 7 pre automatickú vizuálnu detekciu a rozpoznávanie objektov v nasnímanom obraze alebo videosekvenciách. Navrhnite a overte metódy z oblasti počítačového videnia pre automatickú anotáciu sémantického opisu scény, ktoré vychádzajú z normy MPEG 7. Navrhnuté metódy realizujte do systému generovania automatického opisu na vyššej sémantického úrovni, založeného predovšetkým na kombinácii vybraných príznakov MPEG 7 a vhodného klasifikátora.

Implementáciu realizujte s použitím knižnice OpenCV, ktorá poskytuje širokú škálu funkcionality pre potreby počítačového videnia ako sú transformácie, vyhľadávanie príznakov alebo porovnávanie obrazu. Táto knižnica je tiež optimalizovaná z pohľadu časovej výpočtovej náročnosti.

Overte vlastnosti navrhnutej metódy anotácie ako i časová náročnosť výpočtu na testovacom súbore dát, ktorý bude dostatočne reprezentatívny.

---

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- SHAO, Wenbin, G. NAGHDY a Son Lam PHUNG. Automatic Image Annotation for Semantic Image Retrieval. Lecture Notes in Computer Science. Springer-Verlag Berlin, 2007, vol. 4781, pp. 369-378.
- HUANG, Yin-Fu a Li-Wen CHEN. An event-based video retrieval system by combining broadcasting baseball video and web-casting text. ACM Symposium on Applied Computing. ACM Press, 2011, pp. 846-852. DOI: 10.1145/1982185.1982369.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Martin Geier, konzultoval(a) a osvojil(a) si ho Ing. Vanda Benešová, PhD. a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 14.1.2013

Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / ~~nie~~[4]

Dňa: 1.2.2013

Podpis garanta predmetov

[3] 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

STU FIIT
SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Návrh zadania diplomovej práce

Revízia č.: 1 [1]

Študent:

Meno, priezvisko, tituly: Martin Geier, Bc.
Študijný program: Softvérové inžinierstvo
Kontakt: geier.xmartin@gmail.com

Výskumník:

Meno, priezvisko, tituly: Vanda Benešová, Ing. PhD.

Projekt:

Názov: Automatická anotácia sémantického opisu scény v obraze
Názov v angličtine: Automatic annotation of semantic description of the scene in the image
Miesto vypracovania: Ústav aplikovanej informatiky, FIIT STU, Bratislava
Oblasť problematiky: Rozpoznávanie obrazu

Text návrhu zadania[2]

V dnešnej dobe stále rýchlejšie pribúda množstvo multimediálnych dát, v ktorých je veľmi náročné orientovať sa len na základe ich mena. Preto vznikajú rôzne metódy automatickej anotácie obrazových dát a to buď so spoluúčasťou používateľa, ale hlavne bez spoluúčasti používateľa, použitím vhodných metód rozpoznávania obsahu obrazu.

Analyzujte možnosti globálneho popisu a automatickej vizuálnej detekcie a rozpoznávania objektov v nasnímanom obraze alebo vo videosekvenciách s cieľom rozpoznať obsah scény u niektorých vybraných typov scén.

Navrhnite a overte metódy z oblasti počítačového videnia pre automatickú anotáciu sémantického opisu scény. Navrhnuté metódy realizujte ako systém rozpoznávania vybratých druhov scén ako sú napr. mesto alebo športová hra. Sústreďte sa predovšetkým na metódy založené na segmentačnom prístupe ako i rozpoznávanie kombináciou vybraných príznakov a vhodného klasifikátora. Navrhnite vlastnú metódu rozpoznávania scény, túto implementujte a overte.

Implementáciu realizujte s použitím knižnice OpenCV, ktorá poskytuje širokú škálu funkcionality pre potreby počítačového videnia ako sú transformácie, vyhľadávanie príznakov alebo porovnávanie obrazu.

Vyhodnoťte vlastnosti navrhnutej metódy anotácie ako i časovú náročnosť výpočtu na testovacom súbore dát, ktorý bude dostatočne reprezentatívny.

---

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- SHAO, Wenbin, G. NAGHDY a Son Lam PHUNG. Automatic Image Annotation for Semantic Image Retrieval. Lecture Notes in Computer Science. Springer-Verlag Berlin, 2007, vol. 4781, pp. 369-378.
- HUANG, Yin-Fu a Li-Wen CHEN. An event-based video retrieval system by combining broadcasting baseball video and web-casting text. ACM Symposium on Applied Computing. ACM Press, 2011, pp. 846-852. DOI: 10.1145/1982185.1982369.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Martin Geier, konzultoval(a) a osvojil(a) si ho Ing. Vanda Benešová, PhD. a súhlasí, že bude takýto projekt viesť.

V Bratislave dňa 20.2.2014

Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / nie[4]

Dňa: 7.4.2014

Podpis garanta predmetov

[3] 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

ANOTÁCIA

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: SOFTVÉROVÉ INŽINIERSTVO

Autor: Bc. Martin Geier
Diplomová práca: Automatická anotácia sémantického opisu scény v obraze
Vedúci diplomovej práce: Ing. Vanda Benešová, PhD.
máj, 2014

Veľké množstvo dostupných obrázkov aktívne vplýva na vývoj programov schopných rozpoznávať ich obsah bez spoluúčasti používateľa. Rozpoznávanie obsahu môže prebiehať na úrovni rozpoznávania jednotlivých objektov alebo rozpoznávania celej scény. Táto práca sa zameriava na rozpoznávania obrázkov na úrovni scény. Sú v nej analyzované rôzne prístupy rozpoznávania scény, od základných porovnávacích metód, cez hierarchické metódy až po metódy štatistické. Pre potreby vytvárania datasetu sú v práci opísané existujúce segmentačné nástroje a tiež vlastný nástroj na segmentáciu. V hlavnej časti je opísaný základný koncept projektu na rozpoznávanie scény v obraze. Koncept pozostáva z rozpoznávania miest pomocou nízkoúrovňových príznakov a futbalu pomocou kombinácie superpixelov a algoritmu Grabcut. Rozpoznávanie futbalu je rozpracované ako problém rozpoznania hráčov prostredníctvom navrhnutého binárneho elementu skombinovaného s obrazom rozdeleným na regióny. Tieto regióny sú vytvárané buď metódou superpixelov, alebo pravidelnej mriežky. Obe metódy vytvárajú vektory príznakov, ktoré sú klasifikované. Metódy rozpoznávania scén sú implementované do aplikácie.

ANNOTATION

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: SOFTWARE ENGENEERING

Author: Bc. Martin Geier
Master´s Thesis: Automatic annotation of semantic description of the scene in the image
Supervisor: Ing. Vanda Benešová, PhD.
2014, May

Large amount of the available pictures actively influence the development of the programs which can recognize their content without user's interaction. Content recognition can be applied on the recognizing of the single objects or the whole scene. This work is focused on the image recognition of the scene. There are analyzed different ways of the scene recognition like comparing, hierarchical and statistical methods. For the purposes of the dataset creation there are described several segmentation tools and also custom tool for segmentation. In the main part is described the base concept for the image recognition of the scene. This concept contains from the cities recognition by low-level descriptors and the football game recognition by custom algorithm, which combines the superpixel method and the Grabcut algorithm. The image recognition of the football game is elaborated as football player recognition problem combinating designed binary element and regions devided image. These regions are created by superpixel method or regular grid. Both these methods creates the vectors of the features. These vectors are classified. The scene recognition methods are implemented into single application.

Poďakovanie

Rád by som sa týmto poďakoval vedúcej práce Ing. Vande Benešovej, PhD. za odborné rady a vedenie pri vytváraní diplomovej práce.

Obsah

1 Úvod

Aktuálna doba nás už neobmedzuje v tvorbe akéhokoľvek audiovizuálneho obsahu, ponúka nespočetné množstvo zariadení od tých, ktoré máme stále pri sebe, ako sú telefóny s fotoaparátom a kamerou až po profesionálne zariadenia. Síce sa kvalita fotiek a videa pochádzajúcich z telefónov nedá porovnávať s profesionálnymi zariadeniami, ich kvalita je až mnohonásobne lepšia, ako tomu bolo pred niekoľkými rokmi, čo vedie k ich každodennému používaniu.

Prístup k internetu, pomocou ktorého je možné diela šíriť, tiež nie je obmedzujúci faktor, a preto veľké množstvo obsahu končí práve na serveroch, ktoré častokrát umožňujú obsah prehľadávať a kategorizovať podľa značiek zadaných používateľom.

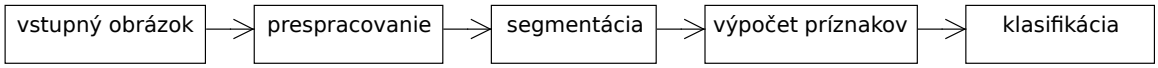
Obsah diela je zväčša popísaný len slovným popisom, ktorý zadal používateľ. Existujú však prípady, kedy používateľ nie je schopný slovne popísať celý obsah diela z dôvodu jeho rozsiahlosti a komplikovanosti. Takýmto príkladom sú videá, ktoré sa skladajú z mnohých scén a ak by k nim chcel používateľ pridať značky, musel by ručne opisovať každú scénu zvlášť. Takýto prístup by vyžadoval veľa času a prinášal by množstvo chýb spôsobených opakujúcou sa prácou. Výsledkom takýchto opisov je zhrnutie obsahu bez možnosti prístupu k jednotlivým scénam.

Takto vzniká široká medzera medzi vytváraním audiovizuálneho obsahu a jeho sprístupnením pre iných ľudí.

2 Analýza problémovej oblasti

Rozpoznávanie scény je úloha rozpoznania globálnych príznakov alebo objektov v obraze, pričom objektom rozpoznávame triedu, do ktorej patria. Takéto rozpoznávanie obvykle prebieha v krokoch zobrazených na obrázku 1 a to:

1. získanie vstupného obrazu,
2. predspracovanie obrazu,
3. segmentácia,
4. výpočet príznakov,
5. klasifikácia.



Obrázok 1: Rozpoznávanie scény

Tejto štruktúry sa pridržiava aj táto kapitola, ktorá postupne diskutuje o metódach rozpoznávania vzorov, segmentácii obrazu a s tým spojenou detekciou hrán, následne rozpoznávaním lokálnych príznakov a na tom založenom rozpoznávaní objektov a klasifikácii. V závere kapitoly sa nachádza analýza normy MPEG 7, ktorá môže poskytnúť niektoré definované príznaky a ich zápis.

2.1 Rozpoznávanie vzorov

Rozpoznávanie vzorov je zamerané na schopnosť pozorovať prostredie počítačom. Jedná sa o základné rozpoznávanie vzorov a triedenie ich do správnych tried. Rozpoznávanie vzorov je zaujímavé v mnohých oblastiach informatiky, tak ako aj v počítačovom videní. Rozpoznávanie vzorov môže byť založené na šablónach, štatistických metódach alebo syntaktickom prístupe [2].

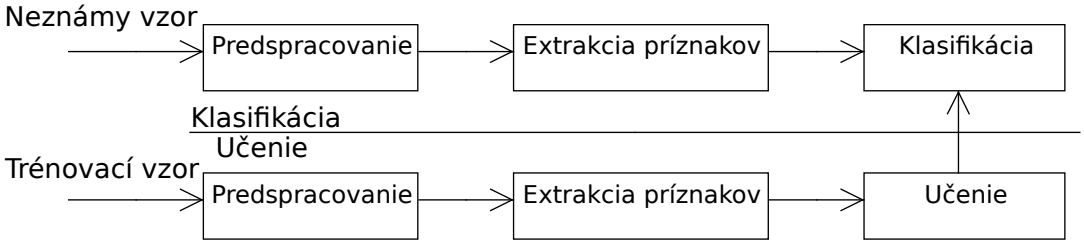
2.1.1 Rozpoznávanie vzorov založené na porovnávaní šablón

Rozpoznávanie na základe porovnávania šablón je najjednoduchšia metóda rozpoznávania vzorov. Metóda využíva trénovaciu množinu, ktorej prvky postupne porovnáva so vzorom. Pri porovnávaní je možné šablónu otáčať alebo inak transformovať. Šablóny sú v tomto prípade najčastejšie dvoj dimenzionálne tvary. Metóda nemusí fungovať v prípade, že vzor je vytvorený z iného pohľadu ako šablóna,

nie je správne predspracovaný alebo je medzi vzorom a šablónou veľký rozdiel, aj keď patria do rovnakej triedy.

2.1.2 Rozpoznávanie vzorov založené na štatistických metódach

Na štatistické rozpoznávanie vzorov je potrebné definovanie a extrakcia príznakov zo vzoru, ktoré sú usporiadané vo vektore. Hodnoty týchto príznakov sú numerické a vzor opísaný takýmto vektorom je možné reprezentovať ako bod v d-dimenzionálnom priestore, kde d je počet príznakov. Pri rozpoznávaní hrá najdôležitejšiu úlohu správny výber príznakov. Príznaky musia byť schopné rozdeliť vzory do príslušných tried, príznaky, ktoré pre všetky triedy dávajú rovnaké hodnoty sú nevhodné a výsledok môžu zhoršovať. Ako je zobrazené na obrázku 2, metóda tiež vyžaduje trénovacie dáta, z ktorých sa extrahujú príznaky vo fáze učenia a tie sú následne vstupom do klasifikačnej metódy. Pri triedení neznámeho vzoru – v klasifikácii je použitá táto natrénovaná klasifikačná metóda, ktorá vzor zaradí do príslušnej triedy.



Obrázok 2: Postup štatistického rozpoznania vzoru

Vhodným prístupom je vstup pred fázou extrakcie príznakov predspracovať tak, že pri získavaní príznakov dostaneme relevantnejšie údaje. Štatistické metódy sú popísané v kapitole 2.5 Klasifikácia.

2.1.3 Rozpoznávanie vzorov založené na syntaktickom prístupe

Mnoho vzorov je komplexných a je ich možné dekomponovať na jednoduchšie vzory a tie na jednoduché. Preto je niekedy vhodné pozerať na vzor ako na hierarchickú štruktúru. Najjednoduchšie vzory sú nazývané primitíva, tvoria teda akoby základné stavebné kamene, z ktorých je možné vybudovať väčšie stavby. Pri rozpoznávaní na syntaktickom prístupe je však nutné naďalej rozpoznávať primitíva, čo zahŕňa segmentáciu, a taktiež použitie štatistických metód.

2.2 Segmentácia

Cieľom segmentácie je rozdeliť obraz na disjunktné časti podľa určeného pravidla homogenity, resp. rozdelenie obrazu podľa objektov reálneho sveta [5]. Podľa tohto rozdeľujeme segmentáciu na čiastočnú a úplnú. Pri čiastočnej segmentácii sa stanoví pravidlo homogenity, podľa ktorého sa obrázok rozsegmentuje, takéto pravidlo môže byť úroveň jasu, farby, saturácie, textúry a pod. Pri úplnej segmentácii zodpovedajú oblasti objektom reálneho sveta.

Segmentáciu možno rozdeliť na 5 hlavných skupín – segmentácia založená na prahovaní, segmentácia založená na hranách, segmentácia založená na oblastiach, segmentácia založená na zhlukovaní a segmentácia založená na reze grafu.

2.2.1 Segmentácia založená na prahovaní

Je to najjednoduchšia metóda segmentácie, kedy je stanovený prah, ktorý určuje, ktorá časť je popredie a ktorá pozadie. Podľa oblasti, na ktorú sa prah použije rozdeľujeme prahovanie na:

- globálne – prah sa použije na celý obrázok,
- lokálne – prah sa použije iba na časť obrázku,
- adaptívne – prah sa prispôsobuje lokálnym podmienkam v obraze.

Len vo veľmi málo prípadoch je vhodné použiť globálny prah, tento problém znázorňuje obrázok 3.

Obrázok 3: Segmentácia na základe prahovania, a) zdrojový obrázok, b) globálne prahovanie, c) adaptívne prahovanie

Prahovanie podľa spôsobu priradenia výslednej farby môžeme doplniť o:

- intervalové – b(x,y) = 1 ak $f(x,y) \in D$, inak b(x,y) = 0,
- poloprahovanie b(x,y) = f(x,y) ak $f(x,y) \geq T$, inak b(x,y) = 0,

- s viacerými prahmi b(x,y) = 1, ak $f(x,y) \in D1$,
  b(x,y) = 2, ak $f(x,y) \in D2$, inak
  b(x,y) = 0.

2.2.2 Segmentácia založená na hranách

Segmentácia založená na hranách využíva hranovú detekciu na rozdelenie obrazu na disjunktné časti. Základným problémom segmentácie založenej na hranách je prítomnosť lokálnych hrán na miestach, kde sú globálne hranice a prítomnosť globálnych hrán na miestach, kde sú lokálne hranice. Algoritmy na detekciu hrán sú opísané v časti 2.3 Detekcia hrán.

Na detekciu globálnych hrán môžeme použiť:

- Prahovanie – najjednoduchšia metóda, kedy je na hrany použitý prah, ktorý určí, či daná hrana patrí do globálnej, alebo nie. Metóda vychádza z predpokladu, že silnejšie hrany patria do globálnych hrán a slabšie do lokálnych.
- Optimalizácia hrán – aplikácia určitých pravidiel na hrany, ktoré tieto hrany zosilňujú alebo zoslabujú. Algoritmus postupne na každú hranu uplatňuje stanovené pravidlá, až pokiaľ výsledné hrany nekonvergujú k finálnej hrane alebo k odstráneniu hrany.
- Založené na heuristickom hľadaní – výsledný objekt musí mať stanovené kritérium optimálnosti. Algoritmus pokladá obrázok za graf, ktorého pixely reprezentujú vrcholy, hrany v obrázku reprezentujú hrany medzi vrcholmi. Nutnou súčasťou je určenie heuristiky, podľa ktorej sa v grafe nájde najkratšia cesta.
- Watershed segmentácia – algoritmus pokladá hrany za vrcholy kopcov a nehranové časti za údolia. Algoritmus simuluje tok riek v týchto mapách a miesta kadiaľ pretekajú rieky označuje za hrany.
- Houghova transformácia – je veľmi efektívna metóda pre vyhľadávanie známych tvarov v obraze, najčastejšie čiar a kruhov, avšak použiteľná aj na ľubovoľné útvary. Základný princíp metódy spočíva vo vytváraní všetkých možností umiestnenia hľadaného objektu cez bod, cez ktorý vedie hrana a následné umiestnenie objektu do miesta, kde prechádza najviac bodmi.

Houghova transformácia je odolná voči natočeniu, priblíženiu, avšak nie voči skoseniu.

2.2.3 Segmentácia založená na oblastiach

Segmentácia založená na oblastiach vyžaduje spĺňanie niekoľkých základných pravidiel:

- zjednotenie všetkých častí musí dať celý obrázok,
- prienik dvoch rôznych oblastí musí byť nulový,
- homogenita oblasti musí spĺňať určené kritérium.

Na segmentáciu založenú na oblastiach existujú tieto základné prístupy:

- spájanie oblastí – obrázok je rozdelený do viacerých oblastí, ako je potrebné, algoritmus prechádza susedné oblasti a ak spĺňajú zadané pravidlo homogenity, oblasti spojí,
- rozdeľovanie oblastí – obrázok nie je rozdelený na homogénne oblasti, algoritmus postupne obrázok delí na menšie časti, pokiaľ nie je splnené kritérium homogenity,
- rozdeľovanie a spájanie oblastí – kombinácia predchádzajúcich prístupov. Obrázok je postupne rozdelený na malé oblasti, následne sú susedné oblasti kontrolované či sa nedajú zlúčiť. Príkladom je algoritmus *Split & Merge*.

2.2.4 Segmentácia založená na zhlukovaní

Skupina týchto metód pokladá segmentáciu za zhlukovací problém, kde každý pixel obrázku patrí nejakému zhluku a obrázok je zložený z niekoľkých oblastí. Na zhlukovanie je možné využiť mnoho existujúcich algoritmov, napr.:

- K-Means – každý pixel je umiestnený do N rozmerného priestoru v závislosti od počtu atribútov, ktoré ku každému pixelu priradíme. Algoritmus predpokladá rozdelenie obrázku do *k* oblastí. Postup algoritmu:
  1. náhodná inicializácia *k* stredov,
  2. priradenie každého pixelu k najbližšiemu stredu podľa určeného kritéria
  3. výpočet nových súradníc stredov ako strednej hodnoty všetkých pixelov priradených k danému stredu,

4. opakovanie bodu 2 a 3 pokiaľ posuny stredov nespĺňajú zadané kritérium.

- Mean shift algortimus – každý pixel rozmiestni do priestoru podľa hodnôt atribútov. Obrázok je prechádzaný oknami o stanovenej veľkosti a pixely patriace do jedného okna, sú priradené do jednej oblasti. Základná verzia algoritmu vyžaduje konštantnú veľkosť okna, existujú však rozšírenia s dynamicky sa meniacou veľkosťou okna. Postup algoritmu:
  1. náhodné rozmiestnenie okien so stanovenou veľkosťou,
  2. vypočítanie posunutia okna vzhľadom na priemer hodnôt bodov v danom okne,
  3. opakovanie bodu 2 pokiaľ zmena pozície okna nespĺňa stanovené kritérium,
  4. spojenie okien na rovnakej pozícii, rozdelenie obrázku na oblasti podľa umiestnenia pixelov v oknách.

2.2.5 Segmentácia založená na reze grafu

Každý obrázok je možné pokladať za graf, ktorého vrcholy sú jednotlivé pixely a hrany medzi vrcholmi sú susednosti pixelov. Následne je možné pokladať segmentáciu ako problém binárneho označenia jednotlivých pixelov na pozadie a popredie. V teórii grafov nájdenie rezu grafu, ktorý nám vytvorený graf rozdelí na dve disjunktné oblasti.

Na nájdenie optimálneho rezu sa používa algoritmus hľadania maximálneho toku. Takýto algoritmus je napr. GrabCut [29].

GrabCut patrí medzi interaktívne algoritmy, ktorému je potrebné ako vstup určiť oblasť popredia a pozdia, resp. oblasti popredia, pozadia, pravdepodobného popredia a pravdepodobného pozadia. Výsledky segmentácie je tak možné iteratívne zlepšovať. Algoritmus pracuje iteratívne, v jednotlivých iteráciách spresňuje segmentované oblasti. V každej iterácii dochádza ku zhlukovaniu a následne k minimálnemu rezu grafu. Na zhlukovanie a výpočet energie medzi vrcholmi grafu sa využíva Gaussian Mixture Model. Nevýhodou algoritmu je časová náročnosť, ktorá výrazne rastie s rozlíšením obrázka.

2.3 Detekcia hrán

Základnou myšlienkou detekcie hrán je previesť obrázok na množinu čiar. Hrany vznikajú v obraze na miestach s veľkou zmenou intenzity farby. Tieto zmeny je možné získať v jednorozmernom priestore ako prvú deriváciu, kde hrany budú miesta s najväčšou zmenou, alebo ako druhú deriváciu s nulovou hodnotou na hranách [4].

V dvojrozmernom priestore je možné deriváciu uskutočniť použitím konvolučnej matice so správne nastavenými hodnotami. Podľa hodnôt konvolučnej matice preto rozpoznávame:

- Sobel filter – používa dve konvolučné matice s rozmerom 3x3, jednu pre deriváciu podľa hodnoty x a druhú podľa hodnoty y,
- Roberts filter – použitie dvoch konvolučných matíc s rozmerom 2x2,
- Laplace filter – použitie druhej derivácie pre detekciu hrán,
- Prewitt filter – použitie konvolučnej matice s kladnou a zápornou stranou.

Detekcia hrán je možná aj prostredníctvom odčítania rozmazaného obrázku od pôvodného, kedy v rozmazanom obrázku sú hrany minimalizované, ale zvyšok je relatívne zachovaný a po odčítaní sú hrany rozdielovým prvkom.

2.4 Lokálne príznaky

Rozpoznávanie objektov na základe lokálnych príznakov sa vykonáva v nasledujúcej postupnosti:

1. vyhľadanie množiny kľúčových bodov na obrázku,
2. definovanie okolia kľúčových bodov,
3. extrakcia a normalizácia obsahu okolia,
4. výpočet lokálnych príznakov,
5. porovnávanie.

Pre vyhľadávanie kľúčových bodov sa používa niekoľko prístupov, najjednoduchšie sú použitie pravidelnej mriežky, či rozmiestnenie bodov náhodne. Zložitejšie prístupy sa zameriavajú na body záujmu, ktoré by mali spĺňať nasledujúce vlastnosti:

- rovnaký obrázok má body záujmu na rovnakých miestach,

- body záujmu sú invariantné voči škále a natočeniu,
- body záujmu sú invariantné voči osvetleniu.

Na výpočet lokálnych príznakov najčastejšie používaný algoritmus SIFT a SURF.

2.4.1 Vizuálne slová

Metóda postavená na lokálnych príznakoch, zväčša SIFT, ktorého výstupom je bod v 128 rozmernom priestore. Spracovaním veľkého množstva obrázkov získavame mnoho bodov v takto veľkom priestore, ktorý následne môžeme klastrovať. Výsledkom sú klastre, ktorých centroidy nazveme vizuálne slová.

Obsah obrázku sa následne rozpoznáva na základe percentuálneho začlenenia lokálnych príznakov do vizuálnych slov.

2.5 Histogram orientovaných gradientov

Histogram orientovaných gradientov (HOG) je algoritmus používaný na detekciu ľudí. Na rozpoznanie osoby sa vyžaduje správna veľkosť obrázku. Prvá úspešná implementácia bola predstavená v [36]. Algoritmus sa skladá z nasledujúcich krokov:

- segmentácia obrázku na N blokov s 50% prekryvom,
- každý blok rozdelený na 2x2 bunky,
- pre každú bunku sa počíta orientácia a magnitúda gradientu,
- orientácia je zapísaná do histogramu obsahujúceho 9 košov.

Na rozhodnutie, či obrázok je, alebo nie je postava, sa používa klasifikátor SVM.

2.6 Klasifikácia

Na opis najčastejšie používaných klasifikátorov je potrebné definovať niekoľko základných označení:

- množina tried $C = \{c_1, c_2, \ldots, c_n\}$, kde c_j je konkrétna trieda, do ktorej chceme prvok klasifikovať, napr. mesto, krajinka, púšť, atď. počet týchto tried býva relatívne malý oproti počtu klasifikovaných objektov,
- množina objektov D, ktoré sú vstupom do klasifikácie a poznáme hodnoty príznakov týchto objektov,

- množina objektov O, ktorá je podmnožinou množiny D, je taká, kde poznáme nielen hodnoty príznakov, ale aj, do ktorej triedy tieto objekty patria.

Práca klasifikátora má dve základné fázy:

1. trénovaciu fázu, kedy učíme klasifikátor na známych objektoch z množiny O, táto fáza však nie je vždy potrebná a prebieha až počas klasifikácie,
2. klasifikácia, kedy klasifikátor radí prvky z množiny D do konkrétnych tried c_j

2.6.1 Naivný Bayesov klasifikátor

Naivný Bayesov klasifikátor [11] klasifikuje na základe podmienenej pravdepodobnosti P (c_j|d), čiže pravdepodobnosť, že objekt *d* patrí do triedy c_j. Z trénovacej množiny nie je možné určiť podmienenú pravdepodobnosť priamo, ale podľa Bayesovho teorému, podľa ktorého je podmienená pravdepodobnosť v nezávislej množine možnú vypočítať podľa vzťahu:

$$P(c_j|d) = P(d|c_j) * P(c_j) / P(d)$$

kde:

- P (d|c_j) - pravdepodobnosť, že ak je objekt z triedy c_j tak je typu d,
- P(c_j) - pravdepodobnosť, že objekt je z triedy c_j,
- P(d) - pravdepodobnosť, že objekt je typu d.

Zaradenie do triedy c_j je určené maximalizáciou hodnoty P (c_j|d) pre všetky j z intervalu 1 až n. Pravdepodobnosť P(c_j) je možné vypočítať ako podiel objektov patriacich do triedy c_j a všetkých objektov. Pravdepodobnosť P(d|c_j) sa počíta ako súčin pravdepodobností P (o_k|c_j) pre *k* z intervalu 1 až d, kde P (o_k|c_j) sa počíta ako podiel počtu objektov typu *k* patriacich do triedy c_j a počtu všetkých objektov patriacich do triedy c_j.

Výhodou algoritmu je jednoduchá implementácia, nevýhodou je nutnosť zabezpečiť nezávislosť atribútov.

2.6.2 K-najbližších susedov

Táto metóda je tiež označovaná ako k-NN [11]. Metóda predpokladá číselné hodnoty príznakov a reprezentuje vektor týchto príznakov ako bod v n-rozmernom priestore.

Vstupom metódy je množina objektov, na ktorých však neprebieha prvotná fáza trénovania. Pri klasifikovaní sa neznámy objekt *d*, ktorého príslušnosť k triede C je neznáma, umiestní tiež do tohto priestoru a pomocou matematickej metódy vypočíta vzdialenosť k okolitým známym objektom. Objekt *d* sa následne klasifikuje do tej triedy, ktorá má *k* najbližších prvkov k objektu *d*.

Matematické metódy na výpočet vzdialenosti/podobnosti dvoch vektorov, ak X, Y sú porovnávané vektory a $X = (x_1, x_2, \dots, x_n)$ a $Y = (y_1, y_2, \dots, y_n)$:

- Euklidova vzdialenosť:

$$d(X,Y) = \sqrt{\sum (x_i - y_i)^2}$$

kde:

$(x_i - y_i)$ - rozdiel hodnôt vektora x a y v bode i

- Manhattan vzdialenosť

$$d(X,Y) = \sum |x_i - y_i|$$

- Kosínusová podobnosť

$$podobnost = \cos(\Theta) = \frac{A.B}{\|A\|\|B\|} = \frac{\sum A_i \times B_i}{\sqrt{\sum (A_i)^2} \times \sqrt{\sum (B_i)^2}}$$

- Mahalanobisova vzdialenosť

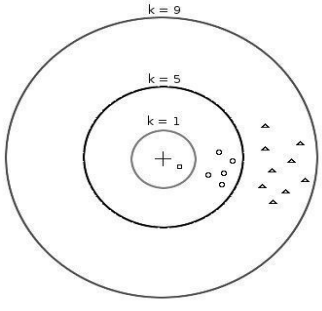
$$d(X,Y) = \sqrt{(x-y)^T S^{-1} (x-y)}$$

kde:

S^{-1} - inverzná kovariančná matica,

$(x-y)^T$ - transponovaná matica rozdielov vektorov x a y.

Najväčším problémom metódy je určenie parametra *k*, ktorý ovplyvní kvalitu klasifikácie. Výber *k* je znázornený na obrázku 4.



Obrázok 4: Vplyv výberu k na výber triedy

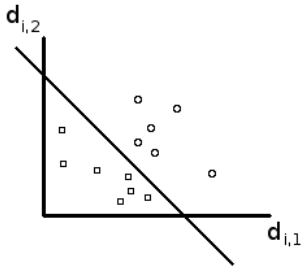
Pri výbere malého *k* reaguje algoritmus na šum, pri výbere veľkého *k* algoritmus prehliadne cieľovú triedu.

2.6.3 Support vector machine (SVM)

SVM klasifikátor patrí do skupiny lineárnych klasifikátorov, ktoré rovnako ako k-NN, predpokladajú, že hodnoty atribútov A_i sú numerické hodnoty a je možné reprezentovať vektor týchto hodnôt ako bod v *n* rozmernom priestore [10].

Lineárne klasifikátory sa snažia lineárne separovať dve triedy buď

- priamkou, ak sa jedná o dvojrozmerný priestor - obrázok 5, alebo
- hyperplochou vo viac rozmernom priestore.



Obrázok 5: Rozdelenie priestora priamkou

Priamka v dvojrozmernom priestore sa dá opísať rovnicou:

$$w_1 d_{i,1} + w_2 d_{i,2} + w = 0$$

kde:

w_j – vypočítaná konštanta,

$d_{i,k}$ – vzdialenosť prieniku priamky s osou.

V N-rozmernom priestore sa dá rovnica zovšeobecniť ako:

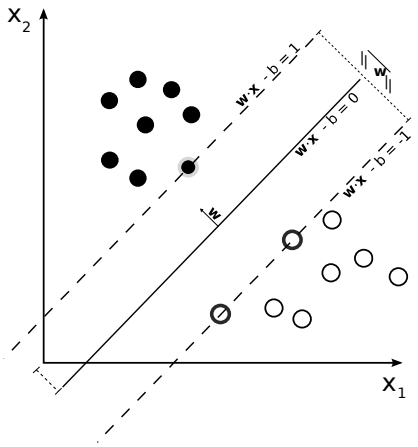
$$\sum w_j d_{i,j} + w = 0$$

Klasifikovaný objekt je zaradený do tej triedy, na ktorej strane hyperplochy sa nachádza. Rôzne lineárne metódy sa odlišujú výberom parametrov w_j.

Nevýhoda tohto klasifikátora je schopnosť klasifikovať iba do dvoch tried. Je preto nutné klasifikátory reťaziť.

Klasifikátor SVM

SVM klasifikátor rozdeľuje trénovacie vstupy hyperplochov optimálne, čiže tak, že body dvoch tried ležia vždy na opačnej strane hyperplochy a vzdialenosť hyperplochy od bodov je čo najväčšia – obrázok 6.



Obrázok 6: Optimálne rozdelenie vstupov hyperplochou

SVM pracuje na princípe prevedenia dimenzie vektora do vyššej dimenzie, kde je veľmi často možné problém lineárne separovať hyperplochou. Prevod vektora na inú dimenziu využíva SVM takzvanú kernel funkciu. Používané kernel funkcie sú:

- lineárny,
- polynomiálny,
- gaussian (RBF),
- sigmoid.

2.6.4 Rozhodovacie stromy

Rozhodovacie stromy [9] je metóda klasifikácie zhora nadol. Klasifikácia prebieha na základe stromu, kde každý nelistový vrchol je popísaný testom a každý listový vrchol reprezentuje triedu.

Na učenie rozhodovacieho stromu existuje viacero algoritmov, najčastejšie však algoritmus C4.5. Učenie stromu prebieha rekurzívnym spôsobom. Začína sa v koreni stromu, ak sú všetky objekty jednej triedy, uzol sa prehlási za listový, ak sú v uzle viaceré triedy, vyberie sa porovnanie, ktoré tieto objekty rozdelí na dve disjunktné množiny. Výber rozhodovacieho pravidla sa líši použitým učiacim algoritmom.

Výber pravidla sa snaží maximalizovať rozdelenie objektov na 2 triedy, ktorých prvky budú z jednej triedy. Pre výber sa môže použiť entropia, čiže miera chaosu.

Pri nerozdelených dátach je chaos väčší ako pri objektoch rozdelených do tried. Entropiu môžeme počítať podľa vzťahu pre Shannonovu entropiu:

$$H(S) = -\sum p(c_j)\log_2(p(c_j))$$

kde:

$p(c_j)$ - pravdepodobnosť, že do triedy c_j bude klasifikovaný nejaký príklad z uzlu S.

Efektívnejší spôsob získania pravidla je použitie pomerového informačného zisku tak, ako to robí aj algoritmus C4.5.

Algoritmus sa dokáže naučiť presne deliť trénované objekty do tried, avšak môže nastať prípad preučenia, kedy neznáme objekty klasifikuje s veľkou chybou. Preto je potrebné vytvorený strom orezávať tak, aby sa riešenie zovšeobecnilo a súčasne sa chyba nedostala cez určenú hranicu.

2.6.5 Boosting

Boosting je metóda zvyšovania kvality klasifikácie reťazením viacerých klasifikátorov. Tieto klasifikátory musia dávať aspoň o niečo lepšie výsledky ako náhodné zaradenie. Tieto klasifikátory sa tiež označujú pojmom slabý klasifikátor (*weak clasifier*).

Prvým klasifikátorom sa nechajú rozdeliť objekty do tried, nasledujúce klasifikátory sa učia na výstupných objektoch, ktoré sú nesprávne klasifikované a príznakom, ktoré sú schopné chybu odstrániť sa pridáva vyššia váha.

V súčastnosti existuje viacero algoritmov využívajúci boosting, najznámejším je algoritmus AdaBoost, ďalšie známe algoritmy sú napríklad LPBoost, BrownBoost alebo LogitBoost.

2.7 Superpixely

Obrázok ako vektor pixelov je často veľmi náročný na spracovanie. Obrázok však častokrát môžeme z pohľadu farby alebo inej vlastnosti rozdeliť na dostatočne malé homogénne oblasti, ktoré stále vytvárajú objekty, no dostatočne veľké na rapídne redukovanie zložitosti operácií nad obrázkom.

Superpixely sa radia medzi segmentačné metódy a ich obľuba v oblasti spracovania obrazu stále rastie. Algoritmy na ich vytvorenie by mali spĺňať tieto základné kritéria:

1. mali by presne kopírovať hrany v obrázku,
2. ak sú použité pre ďalšie spracovanie, mali by byť rýchle na výpočet, pamäťovo nenáročné a jednoduché na použitie,
3. ak sú použité na segmentáciu, mali by redukovať zložitosť a zlepšovať, prípadne nezhoršovať výsledky.

Pre rozdelenie obrázku na superpixely existuje viacero rôznych prístupov, využívajúce napr. morfologické operácie či reprezentáciu obrázku ako graf a využívanie algoritmu rezu grafu.

2.7.1 Algoritmy založené na toku v grafe

Pri týchto algoritmoch je obrázok transformovaný na graf, ktorého vrcholy sú pixely obrázka a ceny hrán sú odvodené od podobnosti susedných pixelov. Takéto algoritmy sú napríklad:

- Normalizovaný algoritmus pre rez grafom, používa rekurzívne delenie obrázka podľa kontúr a textúry [19].
- Felzenszwalb and Huttenlocher, ktorých prístup využíva zhlukovanie vrcholov a hľadanie kostry grafu [18].
- Algoritmus SL08, ktorý hľadá optimálnu cestu v grafe tým, že delí obrázok na malé vertikálne a horizontálne regióny [17].

2.7.2 Algoritmy založené na zmene gradientu

Algoritmy založené na zhlukovaní pixelov, pokiaľ sa nedosiahne určitého kritéria konvergencie. Tieto algoritmy sú bohato spracované a patria medzi ne napr.:

- Algoritmus [16] využíva zhlukovací algoritmus mean-shift, jedná sa o jeden z najstarších algoritmov, výstupom algoritmu sú superpixely, ktorých množstvo a veľkosť nie je možné definovať.
- Prístup založený na morfologickej operácii watershed [15]. Využíva zmenu gradientu na vytvorenie hraníc. Algoritmus je relatívne rýchly, ale neposkytuje stabilné množstvo superpixelov, ani ich kompaktnosť.

- Algoritmus turbopixelov, ktorý využíva gometrický tok, ktorý kopíruje zmeny lokálnych gradientov. Narozdiel od predchádzajúcich algoritmov, algoritmus je nútený pre tvorbu superpixelov podobného tvaru a dodržiavanie kompaktnosti.
- Simple linear iterative clustering – SLIC je adaptácia algoritmu k-means s tým, že nespracúva celý obrázok, alebo iba okolie superpixela, a vstupom do váhovacej metódy je okrem farby aj priestorové rozloženie pixelov. Počas práce algoritmus neustále kontroluje veľkosť aktuálne vzniknutého superpixela, aby bola dosiahnutá podobná veľkosť jednotlivých superpixelov.
- Rôzne implementácie založené na algoritme SLIC.

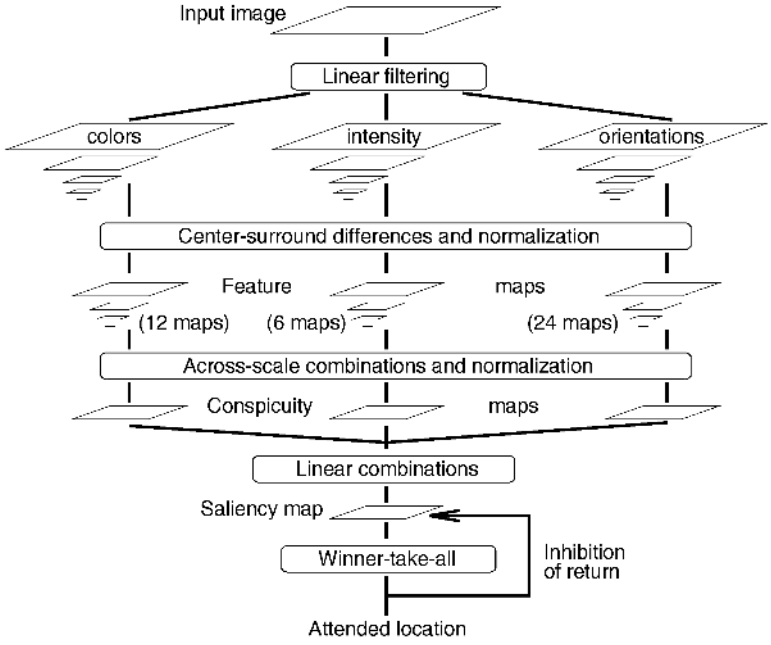
2.8 Mapa pozornosti (Saliency map)

Pozornosť na konkrétny objekt je biologicky daná pre nutnosť minimalizovať množstvo informácií spracovaných v čase. Rovnaký prístup je možné použiť aj pri spracovaní obrazu. Tejto problematike bolo venované veľké úsilie hlavne z dôvodu spracovania obrazu v reálnom čase. Mapa pozornosti je topografická mapa atraktívnosti objektov na obrázku.

Podľa spôsobu vytvárania poznáme:

- Prístup zdola nahor – analýza nízkoúrovňových príznakov obrázku, ako je intenzita, farba alebo orientácia. Základom tohto prístupu je publikácia od Itti a Koch, ktorý vytvorili model spracovania obrázka podľa spomínaných príznakov s využitím pyramídového prístupu zmenšovania analyzovaného obrázka – obrázok 7. Zlúčením týchto dát vytvorili mapu pozornosti [14].
- Prístup zhora nadol – ktorého cieľom je vyznačiť oblasti obsahujúce zaujímavé objekty, nie však na základe nízkoúrovňových príznakov, ale známych objektov ako ľudská tvár, či postava človeka.
- Mapa pozornosti bola prezentovaná aj pre oblasť teórie grafov, kde príznaky intenzity farby a orientácie tvorili vstup pre vytváranie vrcholov v grafe a následne hrany udávali podobnosť vrcholov. Na vytvorenie mapy sú aplikované Markovove modely [13].
- Ďalším zaujímavým smerom je možnosť využitia aplikácie superpixelov na zjednodušenia obrázka a následná analýza príznakov týchto oblastí.

Porovnávaním rozdielností príznakov superpixlov je možné rapídne redukovanie zložitosti problému.



Obrázok 7: L. Itti-ho model vytvárania mapy pozornosti

2.9 Štandard MPEG 7

MPEG 7 je štandard popisovania obsahu multimediálnych súborov, ktorý sa snaží normalizovať zápis príznakov. Štandard zavádza niektoré vlastné príznaky obrazu, ktoré by mohli byť použité v ďalšej práci. Zavádza taktiež jazyk zápisu pre jednoduché prehľadávanie a zapisovanie vypočítaných hodnôt a vzťahov medzi nimi.

MPEG7 bol štandardizovaný v ISO/IEC 15938 s oficiálnym názvom *Multimedia Content Description Interface*. Norma MPEG 7 neposkytuje spôsob, ako kódovať multimediálny obsah, ale ako ho opisovať. Týmto sa odlišuje od najznámejších štandardov rodiny MPEG, ako je MPEG 1, MPEG 2 alebo MPEG 4.

Hlavným cieľom štandardu je interoperabilné vyhľadávanie, indexácia a filtrovanie audiovizuálneho obsahu. Štandard definuje syntax a sémantiku deskriptorov pre možnosť vytvorenia nových deskriptorov, a tým rozširovať možnosti opisu obsahu.

Deskriptory definujú základné vlastnosti obsahu. Môžu to byť nízkoúrovňové príznaky obrazu, ako je farba, textúra alebo tvar, alebo tiež vysokoúrovňové vlastnosti, ako meno autora. Desktriptory sú definované prostredníctvom *Descrition Definition*

Language (DDL). Týmto jazykom sú tiež opísané schémy deskriptorov. Schémy desktriptorov umožňujú spájanie deskriptorov a vytváranie štruktúry medzi nimi. Schémy deskriptorov môžu obsahovať taktiež iné schémy deskriptorov, čím umožňujú vytváranie stromovej štruktúry.

Pre potreby aplikácie je možné zadefinovať nové deskriptory a tiež schémy deskriptorov jazykom DDL, ktorý je špecifikovaný v ISO/IEC 15938-2.

MPEG 7 poskytuje viaceré vrstvy opisu obsahu. V rámci tejto práce sú najzaujímavejšie časť 3 - vizuálna, ktorá definuje skupiny deskriptorov pre opis na nízkej úrovni, ako sú prevládajúca farba, hrany, tvary alebo textúry a časť 5, ktorá poskytuje možnosti popisu na sémantickej úrovni.

2.9.1 Description Definition Language

Description Definition Language (DDL) [3] poskytuje základnú sadu nástrojov potrebnú na vytvorenie nových deskriptorov a schém deskriptorov a ich úpravu alebo rozšírenie. DDL taktiež poskytuje spôsoby, ako kombinovať deskriptory a schémy deskriptorov.

DDL je postavený na jazyku XML, pričom mohutne využíva XML schémy. XML schéma sa uplatňuje na:

- štruktúry – poskytnutie spôsobu zápisu štruktúry XML dokumentu,
- dátové typy – prostredníctvom XML schémy sú definované primitívne dátové typy, odvodené dátové typy a taktiež poskytujú možnosti, ako si zadefinovať vlastné odvodené typy.

Pre potreby normy MPEG 7 boli do XML schém pridané vlastné rozšírenia, ktoré popis obrazu vyžadoval, a to napríklad polia a matice alebo vstavané základné dátové typy.

2.9.2 Visual

Táto časť normy definuje základné vizuálne prvky, ktoré je možné v obrázku rozpoznávať, a to farbu, tvar alebo textúru [1].

Príznaky farby

Norma MPEG 7 definuje tieto základné príznaky farieb:

- Color Space Descriptor CSD – špecifikuje, aký farebný priestor je použitý v iných príznakoch farieb.

- Dominant Color Descriptor DCD – dominantná farba v obraze môže byť reprezentovaná jedinou hodnotou alebo skupinou hodnôt. Príznak sa často používa na vyhľadávanie obrázkov
- Scalable Color Descriptor SCD – vytvára HSV farebný histogram z obrázku. Následne je prevedený na normalizovaný histogram tak, že jednotlivé koše (*bins*) predstavujú pravdepodobnosť výskytu farby v obraze. Následne je na histogram použitá diskrétna Haar transformácia.
- Color Structure Descriptor CSD – príznak sa stále najčastejšie využíva na vyhľadávanie obrázkov. Princíp práce je rozdelenie obrázku mriežkou 8x8 a následné vytvorenie farebného histogramu, popisujúceho dominantnú farbu v prvku mriežky.
- Color Layout Descriptor CLD – je príznak, ktorý je kompaktný a invariantný voči rozlíšeniu. Príznak popisuje priestorové rozloženie farieb. Je používaný prevažne na získavanie podobných obrázkov, filtrovanie obsahu a vizualizáciu. Príznak je možné tiež získať mriežkou v MPEG-7 a Dominant Color deskriptorom.

Príznaky tvaru

- Region-based Shape Descriptor RSD – príznaky založené na tvare objektu, popisujúce viacero disjunktných objektov. Používané na rozpoznávanie objektov.
- Conture-based Shape Descriptor CSD – príznaky založené na obryse objektu, schopné rozlíšiť objekty podobného tvaru, ale s rôznym obrysom. Využívané na vyhľadávanie objektov ľudsky príbuzným spôsobom. Invariantné voči natočeniu objektu.

Príznaky textúry

- Homogenenous Texture Descriptor HTD – popisuje štatistické rozloženie textúry v obraze. Príznak je reprezentovaný ako vektor, ktorý sa skladá zo 62 čísel typu integer, ktoré sú získané použitím Gabor filtra. Využíva sa v aplikáciách na získavanie obrázkov podľa obsahu.

- Edge Histogram Descritpro EHD – reprezentuje rozloženie hrán v obraze. Je tvorený histogramom skladajúcim sa z 80 kontajnerov. Príznak je väčšinou používaný na získavanie podobných obrázkov, ktorých textúra sa mierne líši, napr. príroda.
- Texture Browsing Histogram – poskytuje spôsob efektívneho popísania homogénnych plôch textúry obrázku a umožňuje efektívne prechádzanie a vyhľadávanie podobných obrázkov.

3 Existujúce riešenia

Súčasné riešenia zaoberajúce sa analýzou obsahu obrázkov sa venujú prevažne získavaniu obrázkov – *CBIR content based image retrieval* systémy. Príkladom týchto systémov sú:

- C-BIRD – poskytuje možnosti vyhľadávania obrázkov na základe jednoduchých príznakov, ako je farebný histogram, rozloženie farby a textúry a objektového modelu. Nástroj ponúka používateľovi možnosť interaktívne meniť jednotlivé nastavenia.
- MARS – poskytuje podobné možnosti nastavenia vlastností obrázku ako predchádzajúci systém, ponúka však tiež možnosti vytvárania podmienok v podobe spájania príznakov logickými spojkami.

Tieto systémy sa však nevenujú rozpoznávaniu sémantického obsahu, pracujú len na základe uložených príznakov, ktoré predtým získali.

Aplikácie rozpoznávajúce obsah môžeme z hľadiska oblasti rozpoznávania rozdeliť na:

- aplikácie venujúce sa úzkej oblasti obrázkov, napr. aplikácie slúžiace v lekárstve,
- aplikácie pokrývajúce široké spektrum obsahu.

Aplikácia z lekárskeho prostredia [6] je príkladom aplikácie s úzkou oblasťou spracovania obrázkov. Základom aplikácie je rozsegmentovanie obrázkov na základe hexagonálnej štruktúry a následné vytvorenie zhlukov s použitím algoritmu k-means. Na získanie opisu obrázka boli použité existujúce opísané farebné obrázky.

Aplikácia „Behold“, sa zameriava na širšie spektrum obrázkov. Ako databázu obrázkov využíva službu „flickr“ a umožňuje v nej vyhľadávanie na základe opisu avšak s možnosťou definovania rozpoznanej oblasti. Algoritmus anotácie obrázkov využíva databázu existujúcich anotovaných obrázkov.

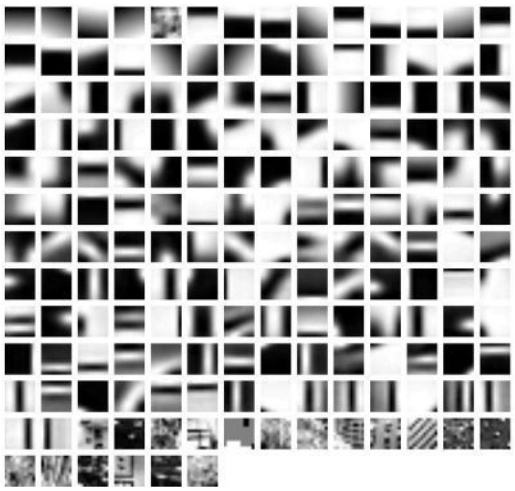
Viaceré práce napr. [7][8] sa zameriavajú hlavne na automatické rozpoznávanie ľudí, resp. tváre. Ich snahou je rozšíriť existujúce prístupy k detekcii tváre o nové príznaky, ako je farba alebo textúra.

Práca [12] sa zameriava na vytváranie príznakov, vhodných na opis rôznych scén. Obrázok reprezentujú ako histogram lokálnych transformácií, ktorý vychádza z *census transform* histogramu. Výsledky práce zaznamenávajú dobré rozlišovacie schopnosti prístupu a ukazujú rýchlosť tvorby navrhnutého príznaku.

Práca [30], sa snaží riešiť problém rozpoznávania vnútorných priestorov. Na popísanie celej scény používajú deskriptor Gist, ktorý vytvorí vektor príznakov o dĺžke 384. Na ďalšie rozpoznávanie používajú vizuálne slová popísané deskriptorom SIFT.

Rozpoznávaniu vonkajších scén sa venuje práca [27], v ktorej popisujú veľmi podobný prístup ako v [30]. Na popis scény použijú deskriptor Gist a následne vyhľadávajú pomocou deskriptora SIFT slová z *bag-of-features*. Na klasifikáciu vo vizuálnych slovách používajú algoritmus k-means. Pre klasifikáciu do rôznych scén následne algoritmus SVM.

Práca [28] sa, rovnako ako predchádzajúca práca, venuje rozpoznávaniu vonkajších scén. Pre rozpoznávanie scén zaviedli tzv. *codewords* – obrázok 8. Obrázok postupne prechádza pohyblivým okienkom, ktoré je náhodne škálované, a takto získané vzorky porovnávajú s kódovými slovami. Pre každú scénu majú vytvorených 50 rôznych kódových slov.



Obrázok 8: Kódové slová pre rozpoznávanie scény

Článok [35] popisuje spôsob rozpoznávania čísel hráčov, vďaka čomu je možné hráčov následne sledovať. Článok prináša novú metódu rozpoznávania založenú

na vyhľadávaní miest s nízkou a vysokou saturáciou rovnakej farby. Takto získajú okolia, kde vyhľadávajú text, resp. čísla.

Zaujímavý smer rozpoznávania scény a hráčov na scéne je možné dosiahnuť analýzou videa, tak ako sa o to snažia v práci [34]. Analýzou obrazu, pri ktorej sa pokúšajú nájsť odchýlky v pohybe tela od zvyšku plátna, dosiahli schopnosť úspešne odhaliť hráča na tenisovom zápase.

Na princípe detekcie pohybu pracujú aj viaceré systémy analyzujúce futbal. Takouto prácou je aj [33], ktorá najskôr nájde futbalové pole v krokoch:

- odstránenie bielej farby, ktorá sa prelína so zelenou farbou,
- použitie Hough transformácie na nájdenie postranných čiar,
- extrakcia týchto ohraničení,
- výpočet záchytných bodov ako rohy.

Následne vďaka statickej kamere a pohybu hráčov vie detegovať postavenie hráčov a hernú situáciu.

Prístup na báze externých zdrojov opisuje článok [32], ktorý sa snaží určovať dianie na ihrisku nie len z audiovizuálneho obsahu, ale súčasne podľa informácií dostupných z internetu. Takýmto prístupom je možné efektívne detegovať udalosti na ihrisku.

3.1 Existujúce riešenia pre segmentáciu a anotácia obrázkov

Pre možnosti vytvárania trénovacej množiny obrázkov je potrebné disponovať buď kvalitnou množinou obrázkov, ktoré už boli anotované, alebo nástrojom vhodným na anotovanie veľkého počtu obrázkov s možnosťami segmentácie častí obrazu. V nasledujúcom texte sú opísané niektoré vybrané anotačné a segmentačné nástroje.

- Iteractiv segmentation tool – Nástroj využívajúci viaceré segmentačné metódy, využíva algoritmus rastúcich regiónov, iteraktívny Grabcut algoritmus a algoritmus binárneho delenia stromov [20]. Aplikácia umožňuje z obrázku vybrať popredie, pozadie, vytvoriť binárnu masku, neumožňuje však orezávanie obrázku, a preto je nutné si obrázok vopred pripraviť.

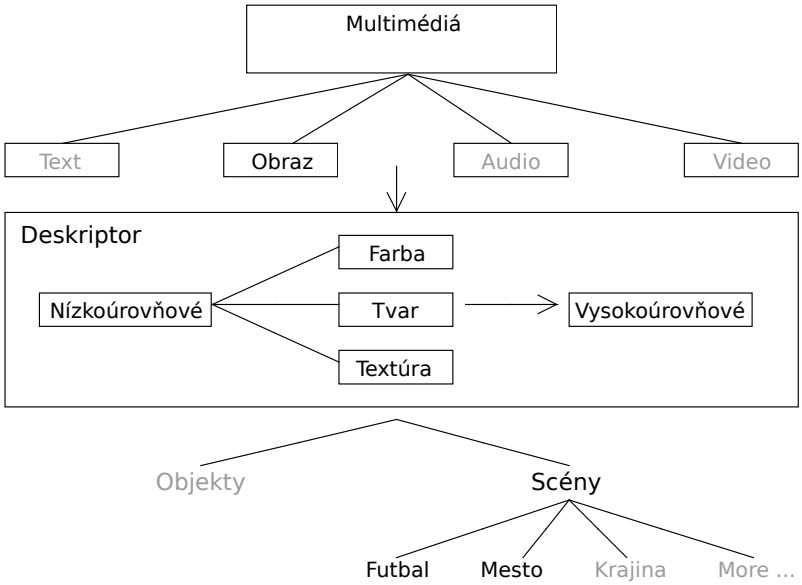
- LabelMe – webový nástroj na vyznačovanie objektov pomocou manuálneho vytvárania obálky. Následne nástroj umožňuje objekt pomenovať, pridať mu textové atribúty a rozmery obálky meniť, ako ju aj presúvať. Výstupom nástroja je vygenerovaný xml dokument obsahujúci body tvoriace obálku objektu [21].
- VideoAnnEx Annotation Tool – Komplexný nástroj na anotáciu obrázkov poskytovaný spoločnosťou IBM. Nástroj priamo podporuje prácu s videom. Umožňuje pre vybraný obrázok pripojiť informácie, ktoré sú priamo exportované do formátu MPEG-7 [22].

Množstvo dostupného softvéru určeného na anotáciu a segmentáciu v obrázkoch je určených do medicínskej oblasti na spracovanie röntgenových snímkov a s tým spojená schopnosť pracovať s 3D obrázkami. Výber niekoľko dostupných nástrojov:

- Iteractiv tool for Image Segmentation – nástroj určený na 2D a 3D obrázky. Pracuje s obrázkami v úrovniach šedej, na segmentáciu využíva vyznačovanie oblasti s rovnakou úrovňou sivej. Pre vytvorené oblasti sú automaticky vytvorené okraje [31].
- MITK – komplexný nástroj na spracovanie medicínskych obrázkov. Spracovanie obrázkov v úrovni sivej, poskytuje rôzne metódy segmentácie [24].
- VITA – nástroj na anotácie 3D röntgenových snímkov pre následnú diagnostiku [23].

4 Koncept návrhu riešenia

Z multimediálneho obsahu sa bude riešenie venovať iba obrázkom, ktoré budú popísané vhodnými deskriptormi. Na popis môžu byť použité nízkoúrovňové deskriptory, ktoré tvoria vysokoúrovňové, ktoré sa snažíme navrhnúť. Pomocou týchto deskriptorov môžeme následne rozpoznávať buď obrázky, alebo, ako v našom prípade, scény. Základnú schému popisuje obrázok 9.



Obrázok 9: Vymedzenie návrhu riešenia

Deskriptory použité na rozpoznanie scény by mali túto scénu zovšeobecniť takým spôsobom, aby pri miernej zmene obrázku bola scéna stále rozpoznaná, no aby do tejto kategórie nebol zaradený obrázok, ktorý túto scénu neobsahuje.

Deskriptor však nemusí popisovať celý obrázok, často to môže vykonať iba nad malou časťou obrázku, vtedy je potrebná segmentácia, ktorá je popisovaná v kapitole 2.4 Segmentácia.

Výstup deskriptorov, ako vektora čísel je potrebné rozpoznávať voči obrázkom, ktoré boli rozpoznané človekom. Na rozpoznanie bude použitý niektorý z klasifikátorov popísaných v kapitole 2.5 Klasifikácia.

V nasledujúcich kapitolách bude popísané navrhnuté riešenie pre rozpoznávania:

- mesta, ako scény obsahujúcej bežné ľudské stavby,
- futbalu, ako scény, ktorá obsahuje trávnik s dostatočným množstvom futbalistov.

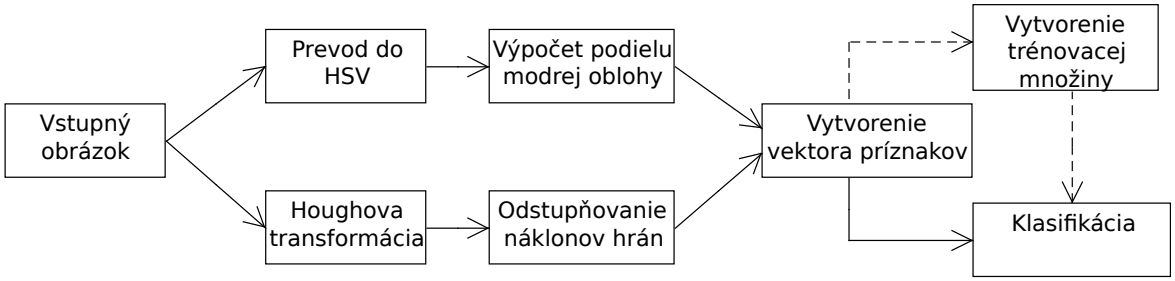
5 Návrh riešenia vizuálneho rozpoznávania miest

Návrh riešenia bol vytvorený ako semestrálny projekt predmetu Počítačové videnie. Zadanie projektu bolo vytvoriť program schopný rozpoznávať dve výstupné triedy – mesto, resp. budovy a obrázky, ktoré neobsahujú mestá, alebo nie sú ich ústredným motívom.

Na rozpoznávanie výstupných tried bola zvolená štatistická metóda rozpoznávania vzorov, ktorá zahŕňa vytvorenie vektora príznakov, učenie klasifikátora a následné klasifikovanie neznámych obrázkov.

5.1 Charakteristické príznaky

Rozpoznávanie budov, resp. miest ako možného typu scény bolo potrebné charakterizovať majoritnými prvkami stavieb. Takýmto charakteristickým prvkom je prítomnosť okien a hrán budov. Minoritným príznakom miest je prítomnosť oblohy. Na obrázku 10 je zobrazený postup rozpoznania scény mesto.



Obrázok 10: Postup rozpoznania miest

5.1.1 Príznaky hrán okien a budov

Okná budov majú väčšinou obdĺžnikový tvar, ktorý by mohol byť detegovaný Houghovou transformáciou s vyhľadávaním obdĺžnika. Tento prístup však vyžaduje splnené dve podmienky:

- kontúry okien sú dobre detegovateľné a spojité,
- okná majú obdĺžnikový tvar.

Prvá podmienka je splniteľná len pri veľkom rozlíšení a vysokej kvalite obrázka. Pri malých obrázkoch alebo vyššom šume je len ťažko možné získať použiteľné kontúry okna.

Obdĺžnikový tvar okien je veľmi častý, no vplyvom perspektívy dochádza k jeho skoseniu. Z týchto dôvodov bola detekcia okien zovšeobecnená na hľadanie hrán, rovnako ako detekcia hrán budov.

5.1.2 Príznaky oblohy

Obloha nie je jednoznačným identifikátorom mesta, ale prítomnosť oblohy zmenšuje oblasť, v ktorej sa vyskytujú hrany charakteristické pre mesto.

Pri detekcii oblohy je potrebné vysporiadať sa s niekoľkými problémami:

- obloha nie je vždy lineárne oddeliteľná,
- nemusí začínať na celej hornej hrane, hlavne z dôvodu natočenia,
- obloha má širokú škálu farieb, hlavne pri východe a západe slnka,
- obloha môže byť čierna, ak je napríklad noc.

5.2 Algoritmus detekcie hrán budov

Hrany budov budeme predpokladať za rovné a na ich detekciu je teda vhodná Houghova transformácia. Vstupom do tejto transformácie je binárny obrázok, kde biela farba určuje hrany a čierna pozadie. Na vytvorenie takéhoto obrázku je možné použiť niektorý z algoritmov opísaných v časti 2.3.3 Detekcia hrán.

5.2.1 Detekcia hrán a okien

Na detekciu hrán bol zvolený Sobelov filter, na ktorý bolo potrebné použiť prah určujúci, aká úroveň šedej je hrana a aká je pozadie. Na obrázku 11 je zobrazené mesto pred a po použití Sobelovho filtra.

Obrázok 11: Použitie Sobelovho filtra

Na hranový obrázok bol následne použitý prah na úrovni 80, ktorý bol zistený experimentálne.

Hranový obrázok bol použitý ako vstup Houghovej transformácie, ktorého výstupom je množina čiar. Tieto čiary sú zobrazené na obrázku 12, kde je naľavo hranový obrázok s použitím prahu a napravo obrázok s čiarami červenej farby.

Obrázok 12: Vizualizácia Houghovej transformácie

Príznaky pre vektor boli vytvorené na základe množstva hrán, ktoré dosahujú určitý náklon, a to podľa vzorca:

$$P_x = Pixels_x / Pixels_{org}$$

kde:

- P_x je hodnota príznaku pri naklonení x stupňov,
- $Pixels_x$ je počet nebielych pixelov po odstránení rovných hrán,
- $Pixels_{org}$ je počet všetkých pixelov v obrázku.

Náklon hrán bol odstupňovaný po 15° a hrany, ktoré prislúchajú tomuto náklonu sú všetky hrany, ktoré spĺňajú podmienku, že náklon hrany je väčší ako X – 15 a menší ako X + 15, kde X je náklon.

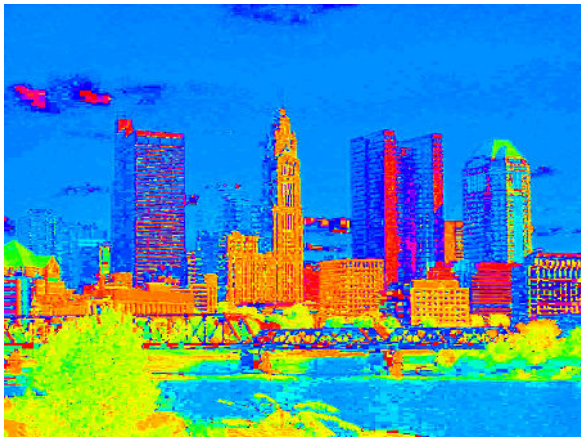
5.2.2 Detekcia oblohy

Algoritmus detekcie oblohy vychádza z predpokladu, že obloha sa bude nachádzať v hornej časti obrázku, úplne od vrchného okraja a bude v odtieňoch modrej.

Algoritmus prebieha v nasledovných krokoch:

1. transformácia RGB obrázku na HSV obrázok,

2. odstránenie zložky jasu a saturácie – obrázok 13,
3. prechádzanie obrázku zhora nadol, ak sa nájde pixel odlišnej farby ako modrej, všetky zvyšné pixely pod ním sa už nepovažujú za oblohu – obrázok 14.



Obrázok 13: Zobrazenie odtieňa farieb

Obrázok 14: Výber oblohy

Príznak je následne získaný ako podiel pixelov oblohy a všetkých pixelov.

5.3 Vyhodnotenie prístupu

Na klasifikáciu je použitý lineárny klasifikátor SVM. Výsledky presnosti klasifikácie sú zobrazené pre dva prípady príznakov:

1. ako príznaky sú použité počty čiar a čiar na ne kolmé – 8 hodnôt,
2. ako príznaky sú použité iba priame čiary – 14 hodnôt.

Vstupom do programu boli fotografie.

Typ detekcie \ Typ príznakov	1	2	Počet vzoriek
Detekcia miest	81,73	79,65	150
Detekcia nemiest	93,84	91,28	200

6 Návrh riešenia rozpoznávania scény so športovou hrou

Športy, ktoré hrajú viacerí ľudia súčasne, sa vyznačujú niekoľkými spoločnými vlastnosťami:

- hráči sú väčšinou vo vzpriamenej polohe,
- hráči majú na sebe dres, ktorý je rovnaký ako jeho tímoví spoluhráči a rozdielny od protihráčov, dres je kontrastnej farby oproti farbe pozadia pre lepší prehľad,
- hrajú na jednofarebnom povrchu – toto neplatí pri halových športoch, ktoré sú pokryté reklamami.

Problémom niektorých športov je hrúbka oblečenia, ktorá môže výrazne meniť základné kontúry človeka.

V nasledujúcej kapitole je opísaný spôsob rozpoznávaniu hráčov na trávniku. Algoritmus je zameraný na rozpoznávanie z televíznej kamery, ktorá sa odlišuje od fotografií hlavne v:

- hráčov je na ihrisku vidieť viacero s pomerne malými detailami,
- hráči sú v prirodzenom rozložení, nie na jednej kope,
- hráči sú často osamote a neprekrývajú sa,
- záber na hráčov je zhora, preto za ich chrbtom nie sú vidieť diváci, ale trávnik.

6.1 Algoritmus rozpoznania scény

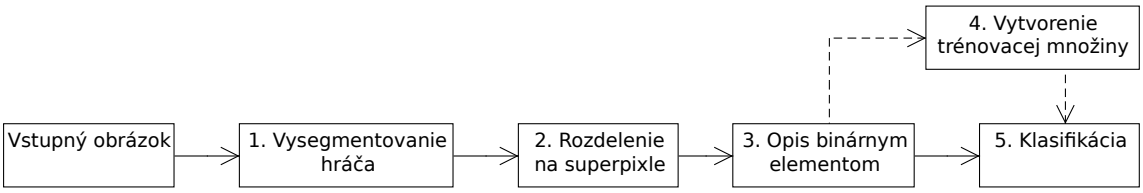
Pre rozpoznanie scény športovej hry bol zavedený predpoklad, že musí obsahovať hráčov, zväčša v menej bežnej polohe. Pre rozpoznávanie ľudí sa najčastejšie používa kombinácia HOG príznakov a klasifikátora SVM.

Na základe stanovených predpokladov bol navrhnutý algoritmus rozpoznávajúci hráčov, ktorý je zobrazený na obrázku 15.

Algoritmus spočíva v týchto krokoch:

1. výber hráča vhodnou segmentačnou metódou,
2. rozdelenie obrázku na regióny,
3. opis hráča deskriptorom odolným voči zmene škály a zmene farby, či osvetlenia,
4. vytvorenie vhodnej trénovacej množiny,

5. klasifikácia vstupných obrázkov na základe natrénovaného klasifikátora.



Obrázok 15: Algoritmus rozpoznania hráča

Pri rozpoznávaní sme sa nevenovali vyhľadaniu hráča na trávniku, ani detekciou trávnika ako plochy určitej farby a vlastnosti.

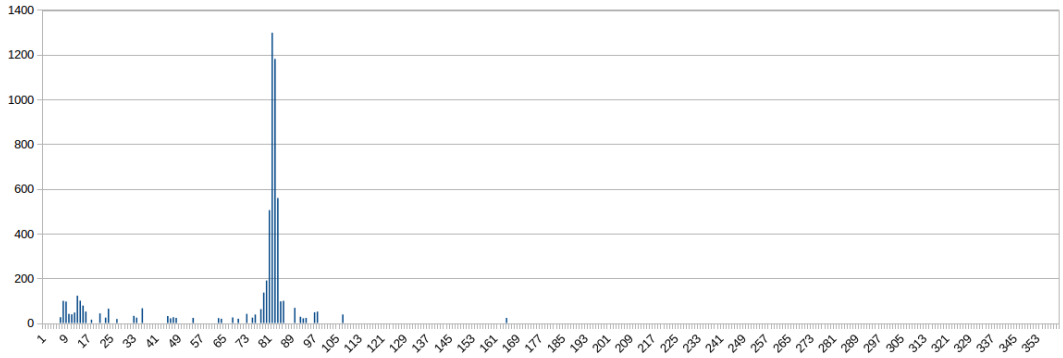
6.2 Automatické segmentovanie hráča

Na automatickú segmentáciu hráča zo zelenej je možné použiť niekoľko prístupov:

- odstránenie dominantnej zelenej farby z obrázku,
- využitie algoritmu Grabcut so vzorkou určenou obdĺžnikom okolo hráča,
- kombináciu superpixelov a algoritmu Grabcut,
- kombináciu algoritmu pre mapu pozornosti a algoritmu Grabcut.

6.2.1 Odstránenie zelenej farby pozadia

Na obrázku 16 je zobrazený histogram početností farieb vybraných častí trávnika, ktorá ukazuje prevládajúcu farbu na úrovni 80° na kruhu odtieňov (hue), čo zodpovedá farbe medzi žltou a zelenou.



Obrázok 16: Odtieň farby častí trávnikov

Na grafe je malá časť vzoriek pripisovaná začiatku kruhu, čo je spôsobené hlavne nízkym jasom vyhodnocovaných superpixelov, kde odtieň je vyhodnotený ako červený.

Metóda tak môže dosahovať dobré výsledky, predpokladá však pozadie zelenej farby, čo je možné len pri športoch na tráve, ale nie pri iných povrchoch. Čím by sme obmedzili ďalšie použitie algoritmu.

6.2.2 Segmentácia hráča algoritmom Grabcut

Algoritmus Grabcut ako vstup využíva masku tvorenú štyrmi rôznymi farbami. Sú to:

- popredia – algoritmus musí toto pole akceptovať ako popredie,
- pravdepodobné popredie – algoritmus pokladá časť za popredie, ale pri svojom behu môže pole zmeniť na pozadie,
- pravdepodobné pozadie – algoritmus pokladá časť za pozadie, no hodnotu môže zmeniť,
- pozadie – algoritmus musí túto časť akceptovať ako pozadie.

Pre jednoduchší vstup je však možné algoritmu odovzdať obdĺžnik, ktorý hovorí, že všetko mimo neho je pozadie, zvyšok je neznáme a bude segmentované. Ako vstup bol použitý vystrihnutý obrázok z videa - obrázok 18, ktorý mal všetky stanovené predpoklady.

Obrázok 18: Vstupný obrázok pre algoritmus Grabcut

Obrázok 17: Výstup z algoritmu Grabcut

Výstup však nezodpovedal očakávanému výsledku. Pri bližšom skúmaní obrázku bolo zistené, že vplyvom kompresie a pohybu vzniká okolo hráča oblasť – obrázok 19, ktorá nemusí byť správne vyhodnotená – obrázok 17.

Základným problémom sa stáva správny výber okolia hráča, kde sme presvedčení, že hráč nie je. Druhý problém vzniká z popísanej kompresie, čo môže byť vyriešené obrázkom s väčšou kvalitou, čo však pri bežnom videu je veľmi obtiažne.

Obrázok 19: Vplyv kompresie a pohybu na rozmazanie farieb

6.2.3 Kombinácia superpixelov a algoritmu Grabcut

Grabcut algoritmus je výpočtovo náročný, čo nie je problém pri obrázkoch veľkosti hráča vystrihnutého z videa. Problém nastáva pri analýze obrázku veľkých rozmerov, na ktorých sa nebude prejavovať vplyv kompresie. Pre tieto obrázky by bolo vhodné použiť vytvorené superpixely ako základná jednotka pre algoritmus Grabcut. Esenciálnym problémom je prispôsobenie algoritmu pre prácu s obrázkom, ktorého pixely nemajú práve 4, respektíve 8 susedov.

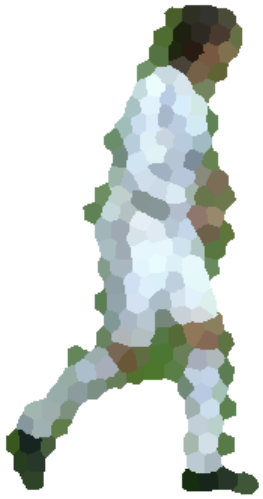
Obrázok 20: Obrázok tvorený 250 spriemerovanými superpixelmi

Obrázok 21: Obrázok tvorený 750 spriemerovanými superpixelmi

Zvolený prístup na overenie myšlienky spočíval vo výpočte superpixelov a následnou zmenou obrázka na priemernú farbu týchto superpixelov.

Vizuálnym porovnaním obrázkov 20 a 21 tvorených rôzne veľkými superpixelmi zistíme, že niektoré superpixely na rozmedzí hráča a trávnika pri nevhodnom segmentovaní vytvárajú farebný prechod. Tento farebný prechod následne zle spracuje aj algoritmus Grabcut, ako je zobrazené na obrázku 23, v porovnaní so segmentáciou pôvodného obrázku – obrázok 22.

Obrázok 22: Grabcut nad pôvodným obrázkom



Obrázok 23: Grabcut nad obrázkom zo superpixelov

6.2.4 Kombinácia mapy pozornosti a algoritmu Grabcut

Algoritmus Grabcut analyzuje pravdepodobné popredie a pravdepodobné pozadie na základe zadaného popredia a pozadia, a preto je nesmierne potrebné správne určiť, čo popredie je a čo popredie určite nie je. Takúto informáciu nám môže poskytnúť algoritmus pre tvorbu mapy pozornosti. Táto mapa je reprezentovaná, ako obrázok v odtieňoch šedej, kde pixely s vysokou intenzitou sú popredie a s nízkou sú pozadie.

Obrázok 24: Pôvodný obrázok a mapa pozornosti tvorená superpixelmi

Mapa pozornosti úspešne zachytila dres hráča, ktorý je ľahko rozpoznateľný od okolia, hlava a časti tela už nie sú tak dobre zachytené, obrázok 24. Prahovaním, podľa určitej minimálnej hodnoty intenzity, je tak možné určiť popredie, tmavá farba však už isté pozadie nie je. Pre určité pozadie by bolo potrebné mapu dilatovať dostatočne veľkým štrukturálnym elementom.

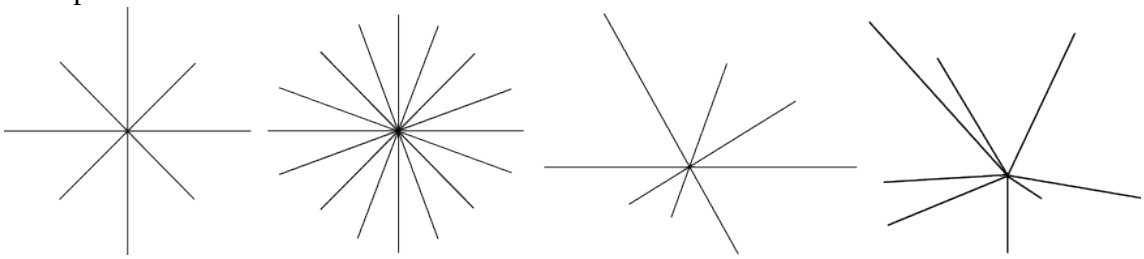
Metóda vytvára základné kontúry hráča, neumožňuje však na 100% určiť trávnik medzi nohami hráča za pozadie, a preto nijakým výrazným spôsobom nezlepšuje výsledok segmentácie oproti obrázku 17.

Metóda poskytuje základný odrazový mostík pre jednoduché určenie približných rozmerov hráča a jeho polohy, ktoré sa dajú použiť na vytvorenie obdĺžnika ako argumentu do metódy Grabcut.

6.3 Deskriptor na popis hráča

Vysegmentovaného hráča – obrázok 22, je potrebné popísať vhodným deskriptorom, ktorý dokáže vhodne zovšeobecniť tvar hráča tak, aby pokryl hráčov podobných tvarov a rozmerov. Náš navrhnutý deskriptor pracuje nad obrázkom tvoreným superpixelmi obsahujúci len informáciu či ide o popredie, alebo o pozadie.

Deskriptor je založený na elemente – obrázok 25, ktorý je prekladaný cez stredy všetkých superpixelov hráča. Stred superpixelu je počítaný ako podiel súčtov x-ových hodnôt pixelov v superpixely a ich počtu a súčet y-ových hodnôt pixelov v superpixely a ich počtu.



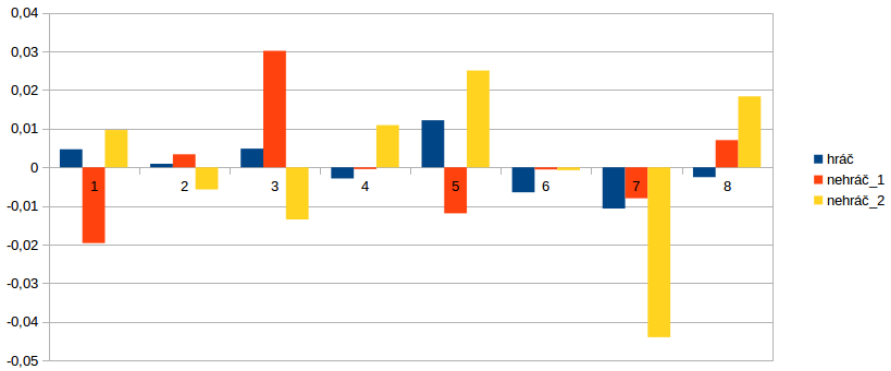
Obrázok 25: Príklady deskriptívnych elementov

Konce elementu následne určujú vyhodnocované superpixely. Pre každý superpixel vzniká binárny vektor, kde hodnota 1 znamená, že lúč elementu končí v superpixeli patriacemu hráčovi a hodnota 0, že patrí pozadiu. Dĺžka vektora je určená počtom lúčov elementu. Sčítaním všetkých binárnych vektorov získame histogram rozloženia hráča na obrázku.

Na správne rozpoznávanie hráčov následne potrebujeme určiť:

- veľkosť elementu, ktorá sa musí vhodne meniť pre rôzne veľké obrázky, výhoda takto jednoduchého elementu je, že jeho zmena je možná nezávisle v x-ovom aj y-ovom smere,
- počet lúčov elementu, ktoré určujú dĺžku histogramu, a teda mieru popisu obrázku,
- tvar elementu, ktorý môže byť súmerný ako prvé dva príklady na obrázku 25 alebo náhodný.

Na obrázku 26 je zobrazené porovnanie histogramov futbalistu s iným hráčom a dvoma obrázkami nehráčov. Na obrázku sú zobrazené diferencie histogramov, pričom histogramy boli normalizované podľa počtu prvkov. Z obrázka je patrné, že obrázky, ktoré neobsahujú hráčov, majú niektoré hodnoty výrazne vychýlené.



Obrázok 26: Diferencie histogramov ku histogramu hráča

6.4 Klasifikácia

Na klasifikáciu obrázkov bol použitý klasifikátor SVM, ktorý delí vyhodnocovanú množinu len do dvoch skupín. Klasifikácia prebiehala nad vektormi, ktoré boli získané normalizáciou vektora ako:

$$\vec{v} = \frac{\vec{u}}{\sum_{i=0}^{N} u_i}$$

Kde:

- v je výstupný vektor,
- u je vstupný histogram,
- N je dĺžka histogramu.

6.5 Experimentálne overenie

Vyhodnotenie výsledkov boli robené nad ručne segmentovanými obrázkami hráčov a náhodných obrázkov. Na segmentáciu bol použitý vlastný segmentačný nástroj opísaný v kapitole 4.2.

Na základný test funkčnosti bol použitý trénovací set 50 pozitívnych a 50 negatívnych vzoriek hráčov. Ako deskriptívny element bol použitý element s rozmermi {10, 0 ; 5, 5 ; 0, 7 ; -6, 6; -12, 0; -3, -3; 0, -8; 3, -2} zapísanými ako dvojice v karteziánskom priestore. Pre klasifikáciu SVM používalo lineárnu kernel funkciu s epsilon 10^-3.

Typ obrázka	Počet obrázkov	Počet kladne rozpoznaných
Futbalista	20	18
Nesfutbalista	20	15

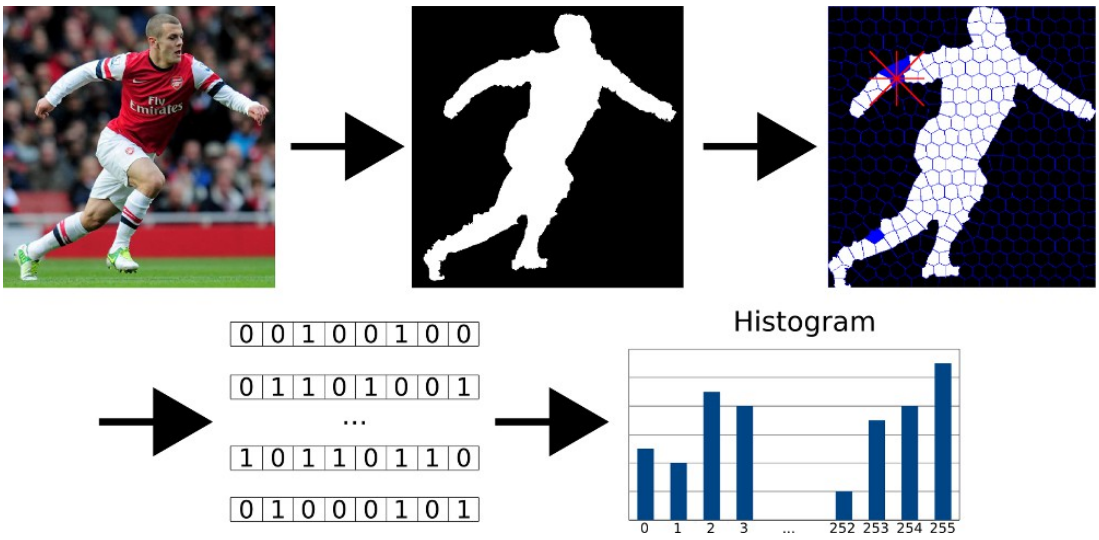
Celková správnosť sa pohybovala na úrovni 84%.

7 Návrh riešenia rozpoznávania hráča

Táto kapitola sa zameriava na rozpoznanie hráča v binárnom obrázku, kde:

- popredie obrázku obsahuje vysegmentovaného hráča,
- pozadie obsahuje zvyšok obrázku.

Možnosti segmentovania hráča boli popísané v predchádzajúcej kapitole, a preto v tejto kapitole bude predpokladaný úspešne vysegmentovaný hráč. Naďalej sa bude pridržiavať postupnosti zobrazenej na obrázku 27.



Obrázok 27: Postupnosť rozpoznania hráča

V kapitole budú tiež opísané možnosti vylepšenia rozpoznávania hráča, či už v možnosti vylepšenia úspešnosti, alebo rýchlosti vyhodnotenia. Preto budú opísané hlavne tieto elementy:

- tvar regiónov,
  - superpixely,
  - pravidelná mriežka,
- počet regiónov,
- tvar binárneho elementu,
- metóda analýzy obrázku,
- interpretácia binárnych vektorov.

7.1 Rozdelenie hráča na regióny

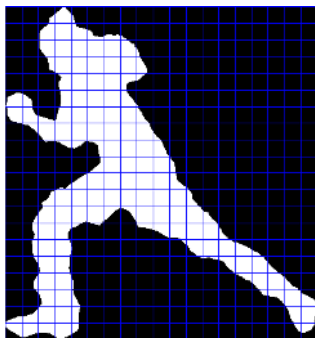
Prvotným krokom pri spracovaním vstupného binárneho obrázku je odstránenie nadbytočnej časti pozadia, obrázok 28. Operácia je nutná pre:

- zmenšenie obrázku pred rozdelením na regióny,
- normalizáciu obrázku.

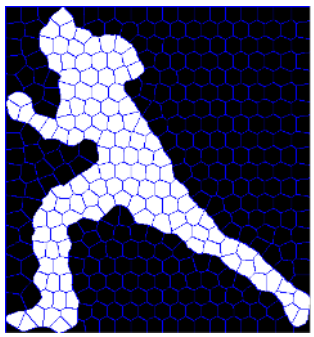
Obrázok 28: Orezanie pozadie hráča

Tvar regiónov, na ktoré hráča delíme, môže byť:

- pravidelná mriežka 29,
- nepravidelný tvar tvoriaci superpixely 30.



Obrázok 29: Rozdelenie hráča mriežkou



Obrázok 30: Rozdelenie hráča na superpixely

Rozdelenie obrázku pravidelnou mriežkou umožňuje zvýšiť rýchlosť spracovania obrázku, no znižuje jeho kvalitu. Pre zachovanie funkčnosti je potrebné rozdeliť obrázok tak, aby jednotlivé regióny boli štvorce.

$$fraction = \sqrt{\frac{c}{w * h}}$$

kde:

- c – požadovaný počet superpixelov,

- w – šírka obrázku,
- h – výška obrázku.

Následne počet superpixelov pre jednu os sa vypočíta ako

$$count = \lfloor w * fraction \rfloor$$

kde:

- count - počet superpixelov na os.

Presný počet regiónov na obrázok je potrebné prepočítať z dôvodu používania celých čísel.

$$num = \lfloor w * fraction \rfloor * \lfloor r * fraction \rfloor$$

kde:

- num – celkový počet regiónov.

Zmena počtu regiónov je možná pri oboch metódach, umožňuje zmenu vlastností okolitých regiónov a pri delení pravidelnou mriežkou znížiť odchýlky spôsobené regiónmi, ktoré obsahujú aj kus hráča aj pozadia.

7.2 Metódy analýzy obrázku

Obrázok rozdelený na regióny je potrebné opísať binárnym elementom. Tento element je prekladaný buď cez:

- regióny patriace hráčovi alebo,
- všetky regióny.

Pri analýze typu regiónu, boli uplatnené pravidlá:

- región patrí hráčovi, ak počet pixelov popredia je väčší ako počet pixelov pozadia vrámci regiónu,
- ak element zasahuje mimo obrázok, región nepatrí hráčovi.

Pri prechádzaní cez všetky regióny má zásadný význam vykonané odstránenie prebytočného pozadia obrázku.

Veľkosť prekladaného elementu je závislá od počtu regiónov a veľkosti samotného obrázka. Pri jeho výpočte sa musí brať do úvahy taktiež pomer strán obrázka, pretože od neho závisí počet stĺpcov a riadkov regiónov.

Element je definovaný pomocou dvojíc, ktoré udávajú vzdialenosť v počte regiónov. Na prepočet na absolútnu dĺžku pre konkrétny obrázok bol použitý vzťah:

$$fraction = \frac{1}{\sqrt{\frac{num}{w * h}}}$$

Kde:

- num – počet superpixelov
- w – šírka obrázku
- h – výška obrázku

$$x = w * fraction$$

$$y = h * fraction$$

Kde:

- x – vzdialenosť na x-ovej osi v pixeloch
- y – vzdialenosť na y-ovej osi v pixeloch

7.3 Interpretácia binárnych vektorov

Preložením elementu cez región získame binárny vektor, ktorý určuje, na ktorej pozícii sa nachádza región hráča a na ktorej pozícii región pozadia. Pri binárnom elemente skladajúcom sa z ôsmich lúčov sa vytvára vektor dĺžky osem. Prevod na histogram je možný ako:

- sčítaním binárnych hodnôt na rovnakých pozíciách vo vektore,
- interpretáciou vektora ako binárneho čísla, následne pre každý vektor inkrementovať histogram na pozícii binárneho čísla.

Interpretácia vektora ako binárne číslo môže mať zásadnú nevýhodu v tom, že zmena jedinej hodnoty vo vektore zásadne zmení celé číslo. Zmena čísla však nie je rovnomerná pre zmenu na ľubovolnej pozícii, ale zmena na vyššom mieste zmení číslo výraznejšie ako na nižšej pozícii.

Návrh úpravy metódy spočíva v inkrementácii pozície binárneho čísla, ale aj tých pozícií, ktoré majú hammingovu vzdialenosť zmenenú o jedna.

8 Vyhodnotenie rozpoznávania scén

V nasledujúcej kapitole sú zhrnuté výsledky rozpoznávania scén športová hra a mesto.

8.1 Vyhodnotenie rozpoznávania scény športová hra

Pri rozpoznávaní scény športovej hry je možné meniť viacero parametrov algoritmu a tým ovplyvňovať aj kvalitu výsledku. Pri testovaní budú vždy uvedené výsledky s klasifikátorom SVM a komparátorom využívajúcu kosínusovú mieru podobnosti, s regiónmi tvorenými superpixelmi alebo pravidelnou mriežkou.

Vstupné parametre pre testovania sú:

- 400 regiónov,
- rozšírený typ vektora,
- testovanie regiónov hráča,
- element {{2,0}, {4,1}, {0,6}, {-2,2}, {-3,1}, {-4,-2}, {0,-3}, {2,-1}}.

Priemerný čas výpočtu pre testovacie súbory trval spolu pre:

- mriežku – 0,783 sekundy,
- superpixely – 20,816 sekundy.

8.1.1 Vplyv zmeny počtu regiónov na presnosť rozpoznávania

Pri zmene počtu regiónov dochádza k zmene toho, čo obsahujú skúmané regióny a súčasne výsledný histogram obsahuje viacero vzoriek. V nasledujúcich tabuľkách je porovnanie 400, 800 a 1200 regiónov na obrázok.

400	SVM			Kosínusová miera podobnosti		
	TP	TN	ACC	TP	TN	ACC
superpixely	97 %	91%	94%	99%	70%	84,5%
mriežka	93%	91%	92%	97%	71%	84%

800	SVM			Kosínusová miera podobnosti		
	TP	TN	ACC	TP	TN	ACC
superpixely	99%	91%	95%	91%	73%	82%
mriežka	97%	93%	94%	86%	73%	79,5%

1200	SVM			Kosínusová miera podobnosti		
	TP	TN	ACC	TP	TN	ACC
superpixely	99%	90%	94,5%	87%	73%	80%
mriežka	96%	91%	93,5%	81%	73%	77%

8.1.2 Vplyv zmeny druhu vektora na presnosť rozpoznávania

Histogram pre obrázok môže vzniknúť:

- sčítaním rovnakých pozícií vektora – základný,
- vytváraním histogramu prevodom vektora na desiatkové číslo – rozšírený,
- vytváraním histogramu prevodom vektora na desiatkové číslo, inkrementovaním aj pozícií s hammingovou hodnotou zmenenou o 1 – modifikovaný.

základný	SVM			Kosínusová miera podobnosti		
	TP	TN	ACC	TP	TN	ACC
superpixely	94%	91%	92,5%	95%	96%	95,5%
mriežka	95%	91%	93%	95%	94%	94,5%

rozšírený	SVM			Kosínusová miera podobnosti		
	TP	TN	ACC	TP	TN	ACC
superpixely	97%	91%	94	99%	70%	84,5
mriežka	92%	91%	91,5	97%	71%	84

modifikovaný	SVM			Kosínusová miera podobnosti		
	TP	TN	ACC	TP	TN	ACC
superpixely	94%	91%	92,5%	91%	73%	82%
mriežka	96%	93%	94,5%	90%	73%	81,5%

8.1.3 Vplyv zmeny druhu regiónov na presnosť rozpoznávania

Prehľadávanie regiónov je možné cez:

- regióny hráča,
- všetky regióny na obrázku.

hráč	SVM			Kosínusová miera podobnosti		
	TP	TN	ACC	TP	TN	ACC
superpixely	97%	91%	94%	93%	70%	81,5%
mriežka	93%	91%	92%	97%	71%	84%

všetky	SVM			Kosínusová miera podobnosti		
	TP	TN	ACC	TP	TN	ACC
superpixely	96%	91%	93,5%	97%	72%	84,5%
mriežka	90%	96%	93%	97%	71%	84%

8.1.4 Vplyv zmeny tvaru elementu na presnosť rozpoznávania

Element {{2,0}, {4,1}, {0,6}, {-2,2}, {-3,1}, {-4,-2}, {0,-3}, {2,-1}}

	SVM			Kosínusová miera podobnosti		
	TP	TN	ACC	TP	TN	ACC
superpixely	98%	91%	94,5%	99%	70%	84,5%
mriežka	93%	91%	92%	97%	71%	84%

Element {{1,0}, {0,1}, {1,1}, {-1,1}, {1,-1}, {-1,0}, {0,-1}, {-1,-1}}

	SVM			Kosínusová miera podobnosti		
	TP	TN	ACC	TP	TN	ACC
superpixely	93%	91%	92%	74%	74%	74%
mriežka	83%	90%	86,5%	74%	74%	74%

Element {{2,0}, {4,1}, {0,6}, {-2,2}, {-3,1}, {-4,-2}, {0,-3}, {2,-1}, {1,0}, {0,1}, {-1,0}, {0,-1}}

	SVM			Kosínusová miera podobnosti		
	TP	TN	ACC	TP	TN	ACC
superpixely	94%	96%	95%	96%	70%	83%
mriežka	93%	96%	94,5%	97%	70%	83,5%

8.1.5 Rozpoznávanie hráčov algoritmom HOG

Algoritmus HOG ako vstupný parameter prijíma veľkosť okna, ktoré prehľadáva a deskriptor, ktorý je buď vytvorený vrámci Opencv, alebo natrénovaný.

Deskriptor	TP	TN	ACC	Čas
prednastavený	1%	100%	50,5%	0,157s
64x128	100%	21%	60,5%	0,157s
128x64	100%	19%	59,5%	0,157s
128x128	100%	67%	83,5%	0,220s
128x256	96%	87%	91,5%	0,349s
256x128	99%	74%	86,5%	0,349s
256x256	86%	96%	91%	0,597s

Z porovnania výsledkov je vidieť, že navrhnutá metóda dáva pri binárnych obrázkoch lepšie výsledky ako bežne používaná metóda HOG.

8.2 Vyhodnotenie rozpoznávania scény mesto

Pri rozpoznávaní scény mesto je možné meniť vlastnosti čiar:

- kolmé - príznaky sú použité počty čiar a čiar na ne kolmé,
- priame - príznaky sú použité iba priame čiary – 14 hodnôt.

	SVM		
	TP	TN	ACC
kolmé	81,73%	93,84%	88,57%
priame	79,65%	91,28%	68,57%

9 Návrh implementácie nástroja na rozpoznávanie scén

V kapitole je opísaný návrh používateľských scenárov, architektúra aplikácie, model dát a návrh používateľského rozhrania aplikácie. Tieto návrhy budú následne použité pri implementácii výslednej aplikácie, umožňujúcej rozpoznávanie scény mesto a hráč.

9.1 Prípady použitia

Kľúčovým prvkom aplikácie je možnosť konfigurácie. Je potrebné umožniť používateľovi výber rozpoznávanej scény, podľa čoho bude umožnené nastaviť ďalšie parametre. Pre rozpoznávanie miest:

- klasifikátor,

pre rozpoznávanie hráča:

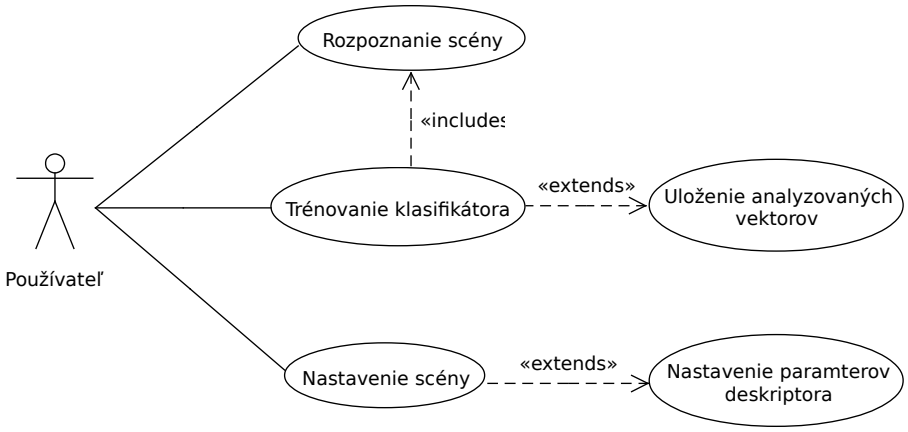
- klasifikátor,
- tvar binárneho elementu,
- tvar regiónov,
- počet regiónov,
- metóda interpretácie binárneho vektora.

Po nastavení cieľových parametrov, používateľ natrénuje klasifikátor:

- analýzou obrázkov,
- z existujúcich uložených vektorov.

Následne po natrénovaní je vhodné používateľovi poskytnúť možnosť uložiť analyzované obrázky v podobe výsledných vektorov. Pre možnosť vytvorenia trénovacích vektorov bez rozpoznávania je tento prípad nepodmienený rozpoznávaním, tak ako je zobrazené na obrázku 31.

Takto natrénovanú aplikáciu môže použiť na rozpoznávanie scény, na ktorú sa uplatní deskriptor vybraný v nastaveniach.

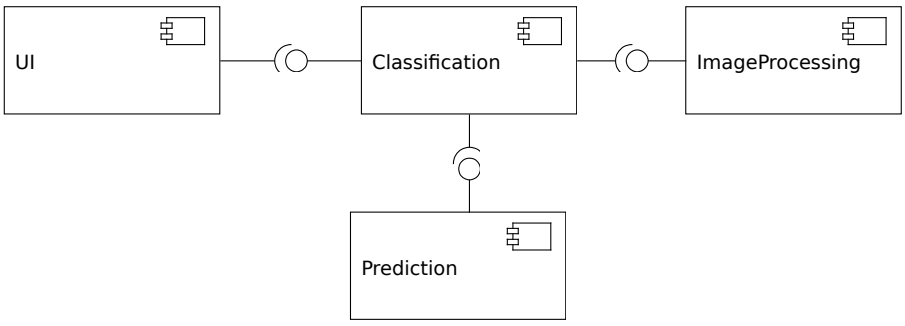


Obrázok 31: Prípady použitia rozpoznávania scén

9.2 Návrh architektúry

Aplikáciu je možné rozdeliť na štyri základné komponenty, obrázok 32. Navrhované komponenty:

- *UI* – obsahuje logiku pre interakciu s používateľom,
- *Classification* – obsahuje vysoko úrovňovú logiku pre výber rozpoznávanej scény, obrázok 33,
- *ImageProcessing* – nástroje na prácu s obrázkami,
- *Prediction* – triedy na klasifikáciu a porovnávanie vektorov.

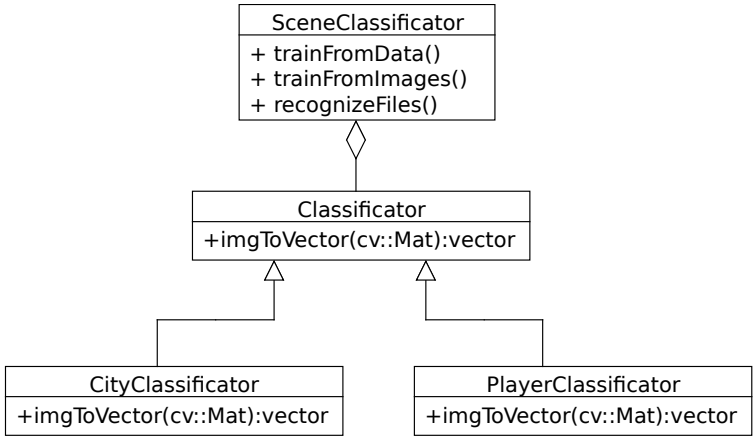


Obrázok 32: Komponentový diagram

Komponent *Classification* musí poskytovať univerzálny prístup k vytváraniu príznakov jednotlivých scén s nutnosťou implementovať čo najmenej povinných metód. V návrhu je preto vytvorená trieda *Classificator*, obrázok 33, ktorá poskytuje jedinú metódu *imgToVector*, ktorá konvertuje vstupný obrázok na vektor hodnôt.

Od tejto triedy dedia triedy:

- *CityClassificator* – vytváranie vektora príznakov pre mesto,
- *PlayerClassificator* – vytváranie vektora príznakov pre hráča.



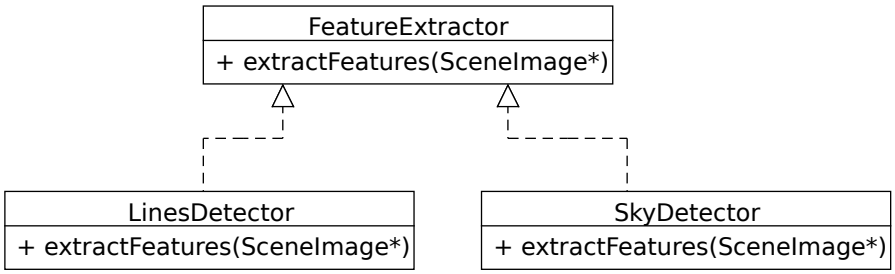
Obrázok 33: Diagram tried, klasifikácia scén

9.2.1 Návrh architektúry rozpoznávania miest

Rozpoznávanie scény mesta bolo navrhnuté na princípe vzoru rúry a filtre. Filtre predstavujú jednotlivé časti rozpoznávania scény a to:

- *LinesDetector* – rozpoznávanie náklonov hrán,
- *SkyDetector* – rozpoznávanie oblohy.

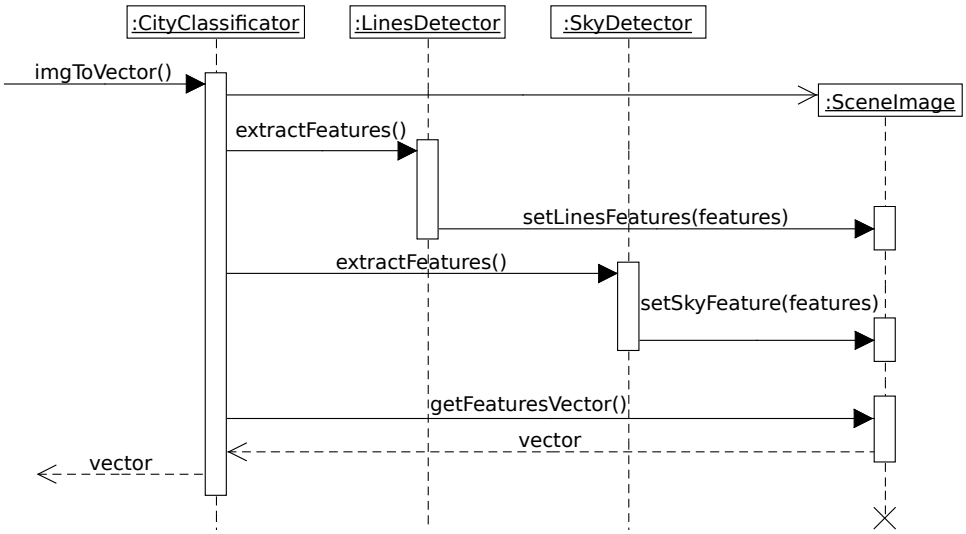
Tieto filtre dedia od spoločného predka *FeatureExtractor*, ktorý predpisuje metódu *extractFeatures*. Diagram tried je zobrazený na obrázku 34. Týmto je zaručené ľahké rozširovanie filtrov a teda pridávanie ďalších hodnôt scény.



Obrázok 34: Diagram tried, extrakcia príznakov mesta

Jednotlivé filtre pracujú nad triedou *SceneImage*, do ktorej ukladajú hodnoty svojich výstupov. Na obrázku 35 je zobrazené postupné volanie týchto filtrov. Filtre je možné prechádzať v cykle, pre zrozumiteľnosť boli rozkreslené jednotlivé filtre.

Po vykonaní všetkých filtrov obsahuje *SceneImage* všetky potrebné údaje na vygenerovanie vektora príznakov.

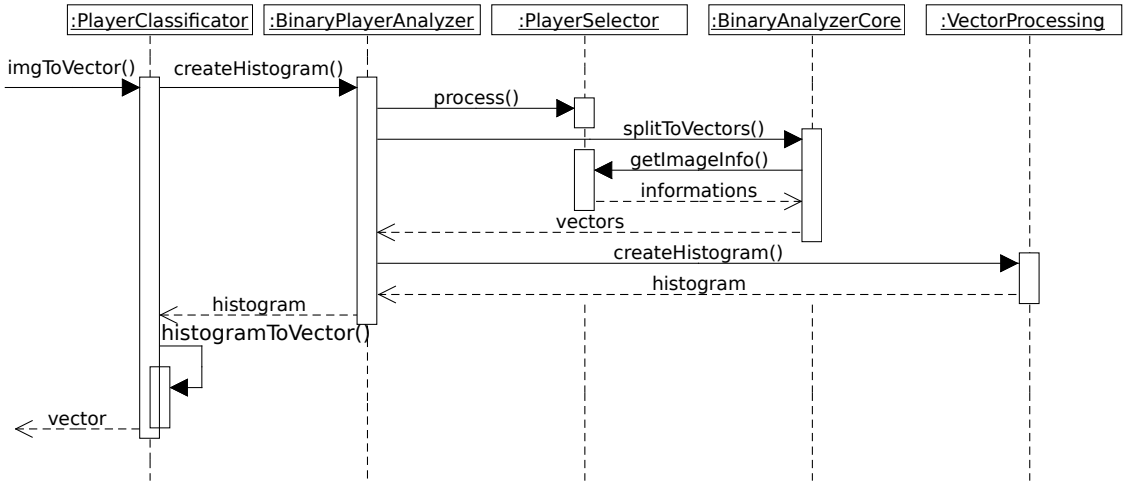


Obrázok 35: Sekvenčný diagram, vytvorenie vektora príznakov mesta

9.2.2 Návrh architektúry rozpoznávania hráča

Návrh rozpoznania hráča musí umožňovať širokú škálu zmeny parametrov. Aj z tohto dôvodu bol sekvenčný diagram navrhnutý pre použitie väčšieho počtu tried, obrázok 36. Sekvenčný diagram obsahuje triedu:

- *BinaryPlayerAnalyzer* - rozpoznávanie hráča z binárneho obrázku, v budúcnosti sa však táto trieda môže zmeniť a rozpoznávať farebné obrázky,
- *PlayerSelector* – výber hráča z binárneho obrázku, môže byť zmenená podľa rozdelenia obrázku na superpixely alebo pravidelnou mriežkou,
- *BinaryAnalyzerCore* – výber regiónov hráča alebo všetkých regiónov,
- *VectorProcessing* – možnosť určovania spôsobu vytvorenia výstupného histogramu, či sčítaním vektora, alebo prevodom do desiatkovej sústavy.



Obrázok 36: Sekvenčný diagram, vytvorenie vektora príznakov hráča

9.3 Návrh serializácie spracovaných dát

Spracovanie veľkého množstva obrázkov je časovo náročné. Ak tieto dáta použijeme na natrénovanie klasifikátora, stačí činnosť vykonať raz, ak však chceme klasifikátor zmeniť, či upraviť nastavenia klasifikátora, je potrebné spracovať obrázky znovu.

Serializácia spracovaných dát prebieha do súboru, kde sa uložia výstupné vektory príznakov. V tomto súbore je potrebné uviesť, ktoré vektory patria k pozitívnej a ktoré k negatívnej množine. Na zápis je možné použiť viacero formátov, napríklad:

- xml, ponúka dobrú čitateľnosť, no relatívne veľkú réžiu,
- json, poskytuje možnosť zápisu polí, jeho čitateľnosť je pomerne dobrá,
- ad-hoc riešenie.

Pre jednoduchý zápis a rýchle spracovanie je navrhnutý ad-hoc riešenie, kedy súbor na prvom riadku obsahuje magický text, na nasledujúcom riadku počet pozitívnych vektorov, ktorý určuje počet nasledovných riadkoch. V každom riadku sa nachádzajú bodkočiarkou oddelené hodnoty vektora. Na ďalších riadkoch sa nachádza počet negatívnych vektorov a tieto vektory.

Na konci tohto súboru sú uložené nastavenia, pri akých vektory vznikli. Tieto nastavenia sa zhodujú s exportom nastavení aplikácie, ktoré sú vo formáte xml pre jednoduché čítanie.

10 Implementácia nástroja

Implementácia nástroja sa pridržiava návrhu v kapitole 10. Pri jeho tvorbe sa prihliadalo na jednoduchú rozšíriteľnosť o ďalšie typy scén a možnosť ďalšieho vývoja.

10.1 Výber implementačného prostredia

Pri výbere implementačného prostredia bolo potrebné zohľadniť základné požiadavky aplikácie:

- použitie knižnice OpenCV na spracovanie obrazu,
- zaručiť rýchlosť algoritmov spracovania obrazu,
- možnosť integrovať existujúce algoritmy postavené na knižnici OpenCV zväčša v jazyku C++.

Pre tieto dôvody bol ako implementačný jazyk zvolený jazyk C++ v kombinácii s knižnicou OpenCV. Pre vytvorenie grafického používateľského rozhrania existujú pre tento jazyk viaceré knižnice:

- winforms – možnosť vytvárať aplikácie len pre systém Windows,
- Gtk, Gtkmm – multiplatformový toolkit, aktuálna verzia však nie je kompatibilná s OpenCV,
- Qt - multiplatformový framework.

Pre možnosť tvorby multiplatformovej aplikácie bol zvolený Qt framework, ktorý okrem tvorby grafického rozhrania ponúka množstvo vylepšení jazyka C++.

10.2 Implementácia aplikácie

Prvotným spôsobom vývoja aplikácie bolo prototypovanie. Na jednoduchých prototypoch bola overená základná funkčnosť algoritmov a následne boli prototypy rozširované. Po tejto úvodnej časti vývoj prebiehal iteratívne, pretože na začiatku vývoja neboli definované všetky požiadavky na projekt a projekt sa postupne menil a vyvíjal.

Funkčnosť aplikácie bolo vďaka prototypom možné sledovať počas celého vývoja, na vývoj preto nebol použitý model vývoja riadeného testami. Po implementácii bola aplikácia testovaná metódou čiernej skrinky z grafického používateľského rozhrania.

Časti aplikácie slúžiace na spracovanie obrazu boli navrhované pre možnosť znovupoužitia, pretože boli použité ako v aplikácii na rozpoznávanie scén, tak aj v aplikácii na anotáciu obrazu Antor.

Počas celého vývoja bol používaný verziovací systém Git.

11 Nástroj na anotáciu a segmentáciu obrázkov

Niektoré existujúce riešenia na opis obrazu boli opísané v kapitole 3.1 Existujúce riešenia pre segmentáciu a anotácia obrázkov. Všetky nástroje sa však zameriavajú na jednotlivé spracovanie obrázkov s nemožnosťou dávkového spracovania.

Pre možnosť segmentovať veľké množstvo obrázkov bolo potrebné navrhnúť vlastný nástroj, ktorý by slúžil práve na tento účel.

11.1 Návrh vlastného riešenia

Na vlastné riešenie problému boli definované tieto požiadavky:

- možnosť spracovať viacero súborov načítaných na začiatku práce,
- možnosť obrázok rozdeliť na menšie časti a tieto časti segmentovať jednotlivo,
- umožniť rýchlu segmentáciu, bez potreby pracného obťahovania kontúry objektu,
- poskytnúť možnosť viacerých druhov výstupu pre následné jednoduché spracovanie.

11.1.1 Implementácia nástroja

Na implementáciu bol zvolený jazyk C++ s využitím knižnice Opencv na spracovanie obrázkov a grafický framework Qt, ktorý umožňuje vytvárať multiplatformové aplikácie.

Pre rýchle segmentovanie bola zvolená metóda využívajúca superpixely. Rôzne nastavenie počtu superpixelov na obrázok dovoľuje rôzne kvality segmentácie. Relatívne vysoká zložitosť výpočtu superpixelov bola zlepšená rozdelením vstupného obrázka na malé časti, pre ktoré sú následne vytvárané superpixely. Algoritmus superpixelov bol použitý SLIC. Tento algoritmus poskytuje kvalitnú úroveň segmentácie a jeho implementácia je verejne dostupná.

Pri implementácii boli testované dve implementácie:

- implementácia od Pascal Mettes, ktorý využil časti algoritmu od Achanta et al. [26],

- adaptívna implementácia (SLICO) od Radhakrishna Achanta, predstaviteľa pôvodného algoritmu [25], ktorá nevyžaduje nastavenie parametra kompaktnosti, iba počtu resp. rozmerov superpixelov.

V nasledujúcej tabuľke je porovnaná rýchlosť vytvárania superpixelov nad obrázkami.

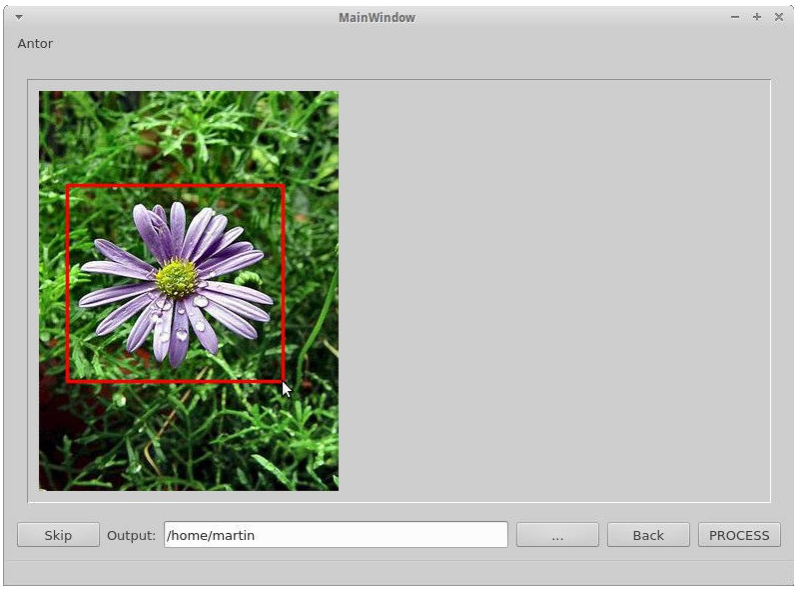
Rozlíšenie obrázka [px]	Čas výpočtu SLIC [ms]	Čas výpočtu SLICO [ms]	Počet superpixelov
276x378	1 130,302	661,250	250
575x579	3 440,643	1 949,478	250
1684x1191	20 302,455	11 113,433	250
2048x1448	31 759,910	16 286,411	250
276x378	1 264,323	672,652	750
575x579	3 810,723	2 143,373	750
1684x1191	22 593,574	11 650,421	750
2048x1448	34 034,372	17 289,791	750

Z dôvodu nižšej časovej náročnosti pri všetkých vstupných obrázkoch a rôznych nastaveniach počtu superpixelov implementácie a možnosti automatického nastavenia konektivity sme do ďalšej práce zvolil algoritmus SLICO. V budúcnosti by však tento algoritmus mohol byť upravený pre výpočty na grafických kartách.

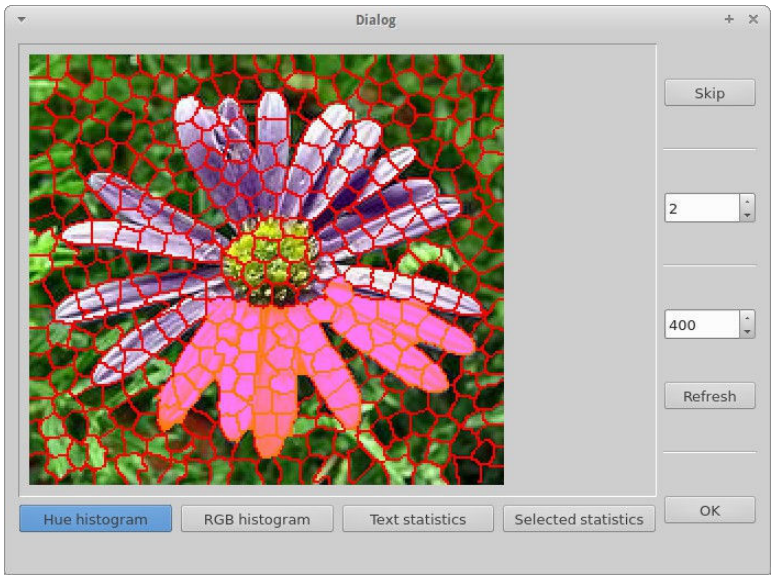
11.1.2 Rozhranie aplikácie

Aplikácia je rozdelená na dve grafické okná:

- obrázok 37 – výber oblasti záujmu z väčšieho obrázku, súčasťou tohto okna je nastavenie výstupného priečinka a možnosti obrázok preskočiť, alebo vrátiť späť krok výberu oblasti záujmu,
- obrázok 38 – výber superpixelov tvoriacich objekt, kde je možné okrem označovania a odznačovania superpixelov ťahaním myši so stlačeným ľavým alebo pravým tlačidlom približovať obrázok, meniť počet superpixelov na obrázok alebo vytvárať pomocné štatistiky.



Obrázok 37: Výber oblasti záujmu



Obrázok 38: Vysegmentovanie objektu

11.1.3 Formáty výstupu

Výstup aplikácie je potrebný optimalizovať na triedu úloh, pre ktoré sú segmenty vytvárané. Aplikácia preto umožňuje tieto výstupy:

- farebný obrázok – obrázok vytvorený výberom oblasti záujmu, ktorá následne bude rozdelená na superpixely (Obrázok 39),
- priemerný obrázok – obrázok tvorený priemernými farbami superpixelov (Obrázok 40),

Obrázok 39: Orezaný vstupný obrázok

Obrázok 40: Spriemerované farby superpixelov

- binárny obrázok – obrázok zobrazujúci vysegmentované časti obrázku,

Obrázok 41: Binárny obrázok

- dátový súbor – súbor obsahujúci informácie o výstupe segmentácie, obsahuje údaje o veľkosti obrázku, polohe a veľkosti superpixelov, ich priemernú farbu a informáciu, či boli používateľom označené. Pre optimalizáciu veľkosti súbora je súbor komprimovaný

12 Záver

Práca si kládla za cieľ rozpoznávania obrázkov na úrovni scény. Sú v nej analyzované rôzne prístupy rozpoznávania scény, od základných porovnávacích metód, cez hierarchické metódy až po metódy štatistické. V analýze je rozpracovaná segmentácia obrazu, niektoré existujúce prístupy rozpoznávania obrazu a klasifikácia.

Pre potreby tvorby datasetu sú v práci opísané existujúce segmentačné nástroje a z toho plynúca nutnosť tvorby vlastného nástroja. Tento nástroj bol použitý pre tvorbu datasetu pre vyhodnotenie úspešnosti navrhnutých metód.

V hlavnej časti dokumentu je opísaný koncept projektu na rozpoznávanie scény v obraze. Koncept pozostáva z rozpoznávania miest pomocou nízkoúrovňových príznakov, ako sú hrany a obloha a futbalu pomocou kombinácie superpixelov a algoritmu Grabcut. Rozpoznávanie futbalu je rozpracované ako problém rozpoznania hráčov prostredníctvom navrhnutého binárneho elementu skombinovaného s obrazom rozdeleným na regióny. Obe tieto metódy vytvárajú vektory príznakov, ktoré sú klasifikované prostredníctvom lineárneho klasifikátora SVM alebo komparátora používajúceho kosínusovú mieru podobnosti.

Obe metódy boli implementované do jednotného nástroja na rozpoznávanie scén, ktorý bol v práci navrhnutý a následne implementovaný. Použitie nástroja je uvedené v prílohe a zdrojové kódy programu na priloženom CD. Výsledky aplikácie sú zhodnotené v časti vyhodnotenie.

Prostredníctvom navrhnutých metód sa podarilo dosiahnuť mieru presnosti rozpoznania miest na úrovni 88% a mieru presnosti rozpoznania hráčov až do 95,5%. Metóda HOG používaná na rozpoznávanie ľudí dosiahla pri rovnakej trénovacej a testovacej množine presnosť len 91,5%, čím sa podarilo ukázať, že prezentovaná metóda je na takýto dataset vhodnejšia. Trvanie výpočtu superpixelov bolo však výrazne vyššie – 20,8 sekundy, ako pri HOG – 0,35 sekundy, pri mriežke bol však čas výpočtu asi len dvojnásobný – 0,78 sekundy.

Ďalším smerovaním práce môže byť zameranie sa na rozpoznávanie hráčov z videa. Tento spôsob umožňuje detekciu pohyblivých častí a tým aj tvaru hráčov. Takto nájdená silueta hráča by bola pripravená na rozpoznávanie navrhnutou metódou.

Zoznam použitej literatúry

[1] ISO/IEC 15938-3 Multimedia Content Description Interface - Part 3: Visual. Final Committee Draft, ISO/IEC/JTC/SC29/WG11, Doc. N4062, Mar. 2001

[2] Jain, A. K., Fellow, Duin, R. P.W.,Mao, J. Statistical Pattern Recognition: A Review. In: IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, vol. 22, no. 1, pp. 4-37

[3] Hunter, J, An Overview of the MPEG-7 Description Definition Language (DDL). In: IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, vol 11, no. 6, pp. 765-722

[4] Maini, R., Aggarwal, H. Study and Comparison of Various Image Edge Detection Techniques. In: International Journal of Image Processing (IJIP), vol 3, pp. 1-12

[5] Salembier, P. An overview of MPEG-7 Multimedia Description Schemes and of future visual information analysis challenges for content-based indexing. "International Workshop on Content-Based Multimedia Indexing". 2001.

[5] Maintz, T., Digital and Medical Image Processing Part II, Chapters 10 (partially) [Medische Beeldverwerking] Periode IV, 2000–2001

[6] Burdescu, D.D., Mihai, C.G., Stanescu, L., Brezovan, M., Automatic image annotation and semantic based image retrieval for medical domain. In: Neurocomputing, pp. 33-48

[7] Cooray, H.S., O'Connor. N.E., Content-Based Image Descriptors for Enhanced Person Annotation in Personal Digital Photo Archives. in: IEEE International Conference on Signal and Image Processing Applications , pp. 449-458

[8] Jiang M.R., Sadka A.H., Zhou H., Automatic Human Face Detection for Content-based Image Annotation. In: Centre for Sensor Web Technol, pp. 449 - 454

[9] Paralič, J. et. al.: Dolovanie znalostí z textov. Vol 1. Košice: Equilibria, 2010. ISBN 978-80-89284-62-7.

[10] Jakkula, R., V., Tutorial on Support Vector Machine (SVM), in: School of EECS

[11] Manning D., Ch., Raghavan, P., Schütze, H., *Introduction to Information Retrieval, In:* Cambridge University Press, *2008,* ISBN: 0521865719

[12] Meng, X., Wang, Z., Wu, L., Building global image features for scene recognition. In: Pattern Recognition, Vol 45, pp. 373-380

[13] - Harel, J., Koch, C., Perona, P., Graph-based visual saliency. In: Advances in Neural Information Processing Systems 19, USA:MIT Press, 2007, pp. 545-552.

[14] - Itti, L., Koch, C., Niebur, E., A model of saliency-based visual attention for rapid scene analysis, In: IEEE Trans. PAMI, 1998, pp. 1254-1259.

[15] - Vincent L., Soille P., Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations, In: IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 13, no. 6, 1991, pp. 583-598.

[16] - Comaniciu, D., Meer, P., Mean Shift: A Robust Approach toward Feature Space Analysis, In: IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 5, May 2002, pp. 603-619.

[17] - Moore, A., Prince, S., Warrell, J., Mohammed, U., Jones, G., Superpixel Lattices, In: IEEE Conf. Computer Vision and Pattern Recognition, 2008.

[18] - Felzenszwalb, P., Huttenlocher, D., Efficient Graph based Segmentation, In: Int'l J. Computer Vision, vol. 59, no. 2, 2004, pp. 167-181.

[19] - Shi, J., Malik, J., Normalized Cuts and Image Segmentatio In: IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22, no. 8, 2000, pp. 888-905.

[20] - McGuinness, K., O'Connor, N., The K-Space segmentation tool set. In: SAMT 2008 - 3rd International Conference on Semantic and Multimedia Technologies, 3-5 December 2008, Koblenz, Germany.

[21] - Russell, Torralba, A., Murphy, K., Freeman, W., LabelMe: A Database and Web-Based Tool for Image Annotation, In: International Journal of Computer Vision 77(1-3), 2008, pp: 157-173.

[22] - Ching-Yung L., Tseng B., Smith, J., VideoAnnEx: IBM MPEG-7 Annotation Tool for Multimedia Indexing and Concept Learning, In: IEEE Intl. Conf. on Multimedia & Expo, Baltimore, 2003.

[23] - Roy, S., Brown, M., Shih, G., Visual Interpretation with Three-Dimensional Annotations (VITA): Three-Dimensional Image Interpretation Tool for Radiological Reporting, In: Society for Imaging Informatics in Medicine 2013

[24] MITK, 2013, The Medical Imaging Interaction Toolkit (MITK), [online], [2013.4.12], Dostupné na internete: http://www.mitk.org/

[25] Image and visual representation group IVRG, 2013, SLIC Superpixels [online], [2013.4.12], Dostupné na internete: http://ivrg.epfl.ch/research/superpixels

[26] PSMM / SLIC-Superpixels, 2013, SLIC-Superpixels [online], [2013.4.12], Dostupné na internete: https://github.com/PSMM/SLIC-Superpixels

[27] Lazebnik, S., Schmid, C., Beyond J., bags of features: Spatial pyramid matching for recognizing natural scene categories. In: cvpr, 2006 , pp: 2169–2178,.

[28] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In: cvpr, 2005 , pp: 524–531.

[29] Rother, C., Kolmogorov, V., Blake, A., “GrabCut” — Interactive Foreground Extraction using Iterated Graph Cuts. In: Microsoft Research, 2004

[30] Quattoni, A., Torralba, A., Recognizing indoor scenes, In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2009

[31] Jackowski, M., Goshtasby, A., Satter, M., Interactive tools for image segmentation. In: Image processing, 1999

[32] Huaxin, X., Tat-Seng, C., The fusion of audio-visual features and external knowledge for event detection in team sports video. In: ACM SIGMM international workshop on Multimedia information retrieval, 2004 pp: 127-134 .

[33] Naidoo, W., Ch., Tapamo, J., R., Soccer video analysis by ball, player and referee tracking. In: SAICSIT '06 Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries, 2006 pp: 51-60

[34] Guangyu, Z., Changsheng, X., Qingming, H., Wen, G., Liyuan, X., Player action recognition in broadcast tennis video with applications to semantic analysis of sports game. In: MULTIMEDIA '06 Proceedings of the 14th annual ACM international conference on Multimedia, 2006, pp: 431-440

[35] Šaric, M., Dujmic H., Papic, V., Rožic, N., Radic, J., Player number recognition in soccer video using internal contours and temporal redundancy. In: ICAI'09 Proceedings of the 10th WSEAS international conference on Automation & information, 2009, pp:175-180

[36] Dalal, N. Triggs, B., Histograms of Oriented Gradients for Human Detection, In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005

Príloha A – Obsah CD

- DiplomovyProjekt.pdf - text diplomového projektu

+ zdrojove_kody

    + Antor - *anotačný nástroj*

        + ImageProcessing - zdrojové kódy pre spracovanie obrazu

        + UserInterface - zdrojové kódy používateľského rozhrania

    + SceneRecognition - *nástroj na rozpoznávanie scén*

        + Clasification - zdrojové kódy pre klasifikátorov scén

        + ImageProcessing - zdrojové kódy pre spracovanie obrazu

        + Prediction - zdrojové kódy klasifikátorov a komparátorov

    + Ui - zdrojové kódy používateľského rozhrania

+ binarne_subory

    + Antor - binárne súbory pre spustenie aplikácie pre Windows

    + SceneRecognition - binárne súbory pre spustenie aplikácie pre Windows

- README.txt

Príloha B – Inštalačná príručka

Anotačný nástroj Antor

Spustenie aplikácie pod systémom Windows:

- prekopírovanie adresára binarne_subory/Antor na lokálny disk,
- spustiť binárny súbor Antor.exe

Kompilácia aplikácie pod systémom GNU/Linux:

- prekopírovanie adresára zdrojove_kody/Antor na lokálny disk,
- spustenie aplikácie qmake v priečinku Antor
- spustenie aplikácie make

Nástroj na rozpoznávanie scén

Spustenie aplikácie pod systémom Windows:

- prekopírovanie adresára binarne_subory/ SceneRecognition na lokálny disk,
- spustiť binárny súbor SceneRecognition.exe

Kompilácia aplikácie pod systémom GNU/Linux:

- prekopírovanie adresára zdrojove_kody/ SceneRecognition na lokálny disk,
- spustenie aplikácie qmake v priečinku SceneRecognition
- spustenie aplikácie make

Knižnice potrebné na kompiláciu

Na kompiláciu je potrebné:

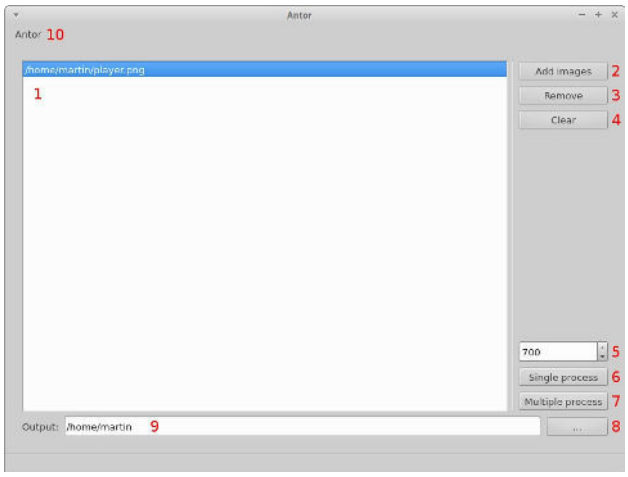
- autotools nástroje
- qt developer library
- opencv developer library

Príloha C - Používateľská príručka

Anotačný nástroj Antor

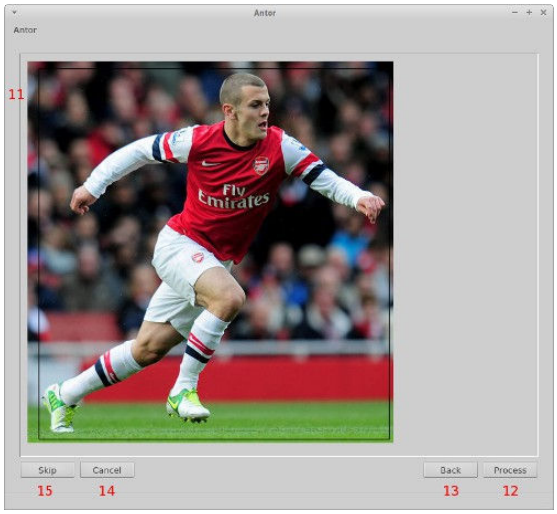
Anotačný nástroj slúži na dávkové i jednotlivé spracovanie obrázkov a to hlavne:

- výber oblasti záujmu z obrázku,
- výber popredia a pozadia obrázku.

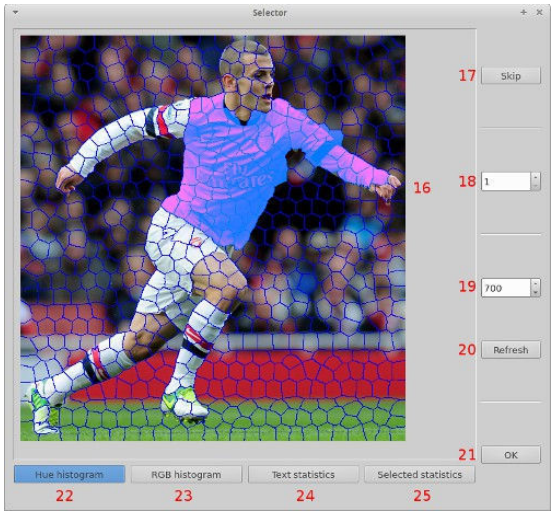


Obrázok 42: Antor, hlavné okno

1. Oblasť zobrazujúca pridané obrázky
2. Tlačidlo pridávania obrázkov
3. Odstránenie vybraného obrázku
4. Odstránenie všetkých obrázkov
5. Nastavenie počtu superpixelov, na ktoré bude obrázok rozdelený
6. Spustenie módu jednotlivého spracovania, obrázok 44
7. Spustenie módu dávkového spracovania, obrázok 45
8. Výber umiestnenia pre ukladanie obrázkov
9. Zobrazené vybraté umiestnenie ukladania obrázkov
10. Tlačidlo pre zobrazenie menu, obrázok 46
11. Výber oblasti záujmu ťahom myši
12. Prechod do módu pre výber popredia – obrázok 43
13. Vrátenie vytvorenia oblasti záujmu
14. Zrušenie spracovania všetkých obrázkov
15. Preskočenie spracovania aktuálneho obrázku
16. Klikom myši vybraté superpixely popredia, klikom pravého tlačidla odznačenie
17. Zrušenie spracovania aktuálneho obrázku
18. Nastavenie priblíženia obrázku

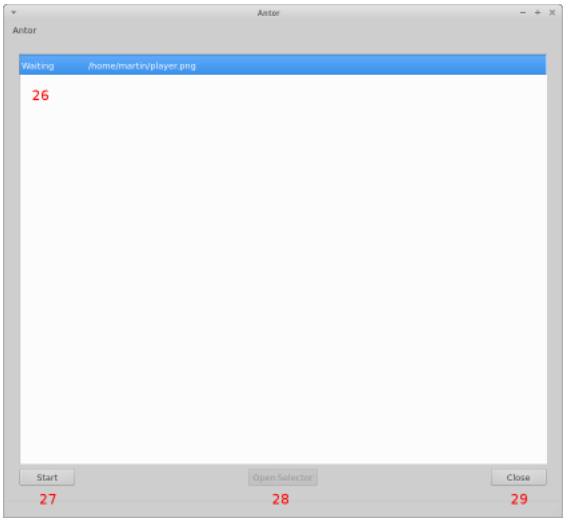


Obrázok 44: Antor, výber oblasti záujmu

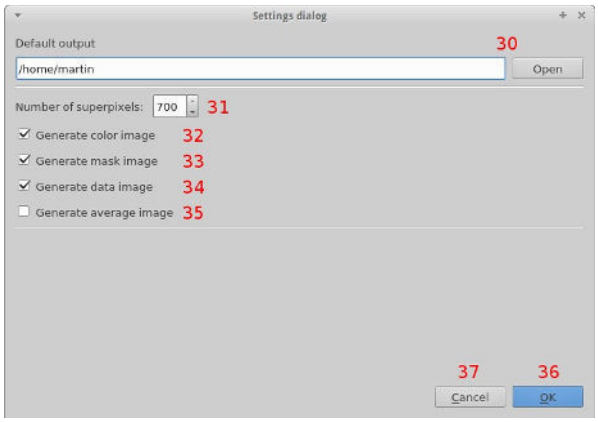


Obrázok 43: Antor, výber popredia

19. Nastavenie počtu superpixelov pre aktuálny obrázok
20. Opätovné prepočítanie superpixelov
21. Uloženie výberu a prechod na ďalší obrázok
22. Zobrazenie histogramu v HUE farebnom priestore
23. Zobrazenie histogramu v RGB farebnom priestore
24. Vygenerovanie textových štatistík obrázka
25. Vygenerovanie textových štatistík pre vybranú oblasť



Obrázok 45: Antor, dávkové spracovanie



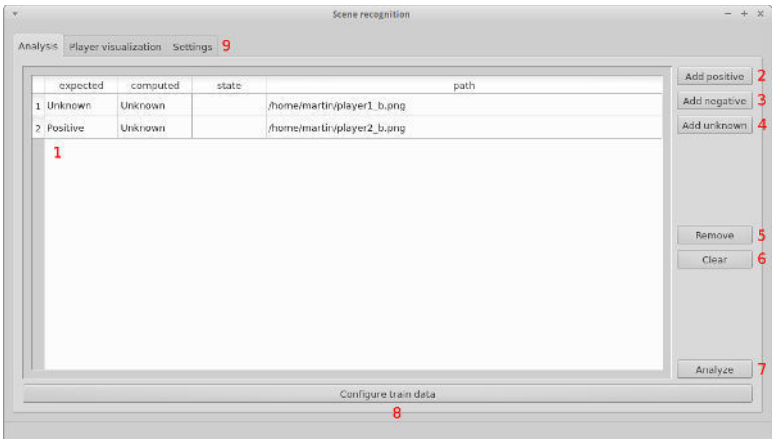
Obrázok 46: Antor, dialóg nastavení

26. Oblasť zobrazujúca spracované súbory a ich stav
27. Spustenie spracovania
28. Možnosť znovuotvorenia dialógu pre výber popredia pre spracované obrázky
29. Zrušenie spracovania
30. Prednastavené miesto ukladania súborov
31. Prednastavený počet superpixelov
32. Povolenie generovania farebného obrázku na výstupe

33. Povolenie generovania binárneho obrázku na výstupe
34. Povolenie generovania dátového výstupe
35. Povolenie generovania obrázka s priemernou farbou superpixelov
36. Uloženie nastavení
37. Zrušenie dialógu

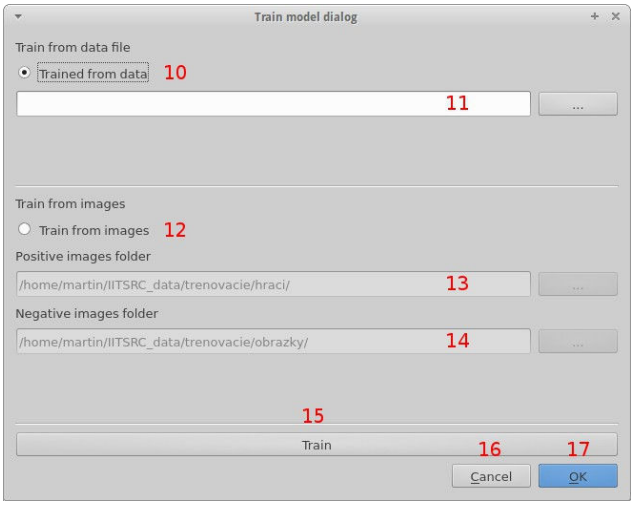
Nástroj na rozpoznávanie scén

Nástroj na rozpoznávanie scény športovej hry a mesta. Algoritmy na rozpoznávanie scén vychádzajú z návrhu v kapitolách 6, 7, 8.

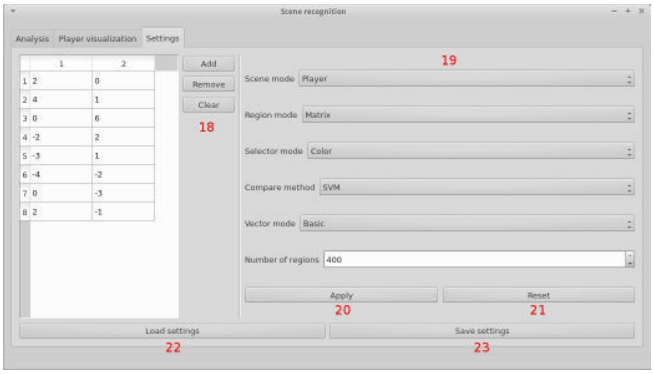


Obrázok 47: SceneRecognition, hlavné okno

1. Zoznam vybraných obrázkov zobrazujúci očakávaný stav a vypočítaný stav
2. Pridanie obrázku s pozitívnym očakávaným stavom
3. Pridanie obrázku s negatívnym očakávaným stavom
4. Pridanie neznámeho obrázku
5. Odstránenie aktuálne vybraného obrázka
6. Odstránenie všetkých súborov
7. Spustenie analýzy obrázkov
8. Konfigurácia trénovacej množiny, okno 48
9. Zobrazenie nastavení, obrázok 49 alebo vizualizácie, obrázok 50
10. Výber módu trénovania zo súboru
11. Súbor, z ktorého prebehne trénovanie
12. Výber módu pre trénovanie z obrázkov
13. Priečinok s pozitívnymi obrázkami
14. Priečinok s negatívnymi obrázkami
15. Spustenie trénovania
16. Zrušenie dialógu

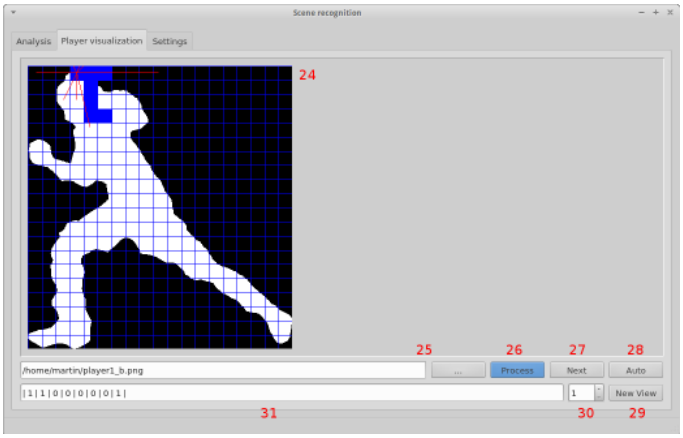


Obrázok 48: SceneRecognition, trénovacie dáta



Obrázok 49: SceneRecognition, konfigurácia

17. Ukončenie dialógu
18. Konfigurácia binárneho elementu
19. Konfigurácia módu rozpoznávania scén
20. Aplikovanie zmien
21. Vrátenie zmien
22. Import nastavení zo súboru
23. Export nastavení do súboru



Obrázok 50: SceneRecognition, vizualizácia

24. Plocha zobrazujúca obrázok rozdelený na regióny s binárnym elementom
25. Cesta k vizualizovanému obrázku
26. Vygenerovanie vizualizovaného obrázku
27. Prechod binárnym elementom na ďalší región
28. Automatický prechod cez všetky regióny
29. Vytvorenie nového okna s vizualizáciou
30. Nastavenie priblíženia
31. Vektor získaný preložením binárneho elementu

Príloha D – Technická dokumentácia

Nasledujúca kapitola obsahuje dôležité časti kódu a technický popis produktu.

Ukážky zdrojového kódu

Výpočet histogramu naklonení hrán pri rozpoznávaní miest. V metóde dôjde najskôr k úprave vstupného obrázku a následne sa vykoná houghova transformácia. Podľa nastavení sa vytvorí histogram náklonov.

```
void LinesDetector::process(SceneImage* image){
    cv::Mat src = image->getImage()->clone();
    cv::cvtColor(src, src, CV_RGB2GRAY);

    cv::Mat preprocess = this->preProcessing(src);
    cv::Mat edges = this->edgeDetection(preprocess);
    cv::Mat edgeImage = this->edgeProcessing(edges);

    std::vector<cv::Vec4i> edgeLines;
    double len = calcLinesLength(edgeImage);
    cv::HoughLinesP(edgeImage, edgeLines, 1, CV_PI/180.0, 1, len, 0);

    std::list<double> imageFeatures;
    int size = (edgeImage.cols* edgeImage.rows);

    if(lineTypes == LINES180){
        imageFeatures = this->get180Degrees(edgeLines, size);
    }
    if(lineTypes == LINES90){
        imageFeatures = this->get90Degrees(edgeLines, size);
    }
    image->setLinesFeatures(imageFeatures);
}

std::list<double>
LinesDetector::get180Degrees(const  std::vector<cv::Vec4i>&  edgeLines,
int size) {
    std::list<double> features;
    for (double i = 0; i < 180; i += this->deviation) {
        int count = numOfLinePixels180(i, this->deviation, edgeLines);
        features.push_back(count / sqrt((double) (size)));
    }
    return features;
}
```

Metóda *scaleCore* slúži na preškálovanie veľkosti elementu podľa aktuálneho obrázku, ktorý je základom pre určovanie cieľových superpixelov.

```
std::vector<std::pair<int, int> >
scaleCore(const std::vector<std::pair<int, int> > &core, int height,
int width, const int& numPixels) {
    std::vector<std::pair<int, int> > resizeCore(core.size());
    double fraction = 1.0 / sqrt(numPixels /((double)height * width));

    for(unsigned int i = 0; i < core.size(); i++){
        resizeCore[i].first = core[i].first * fraction;
        resizeCore[i].second = core[i].second * fraction;
    }

    return resizeCore;
}
```

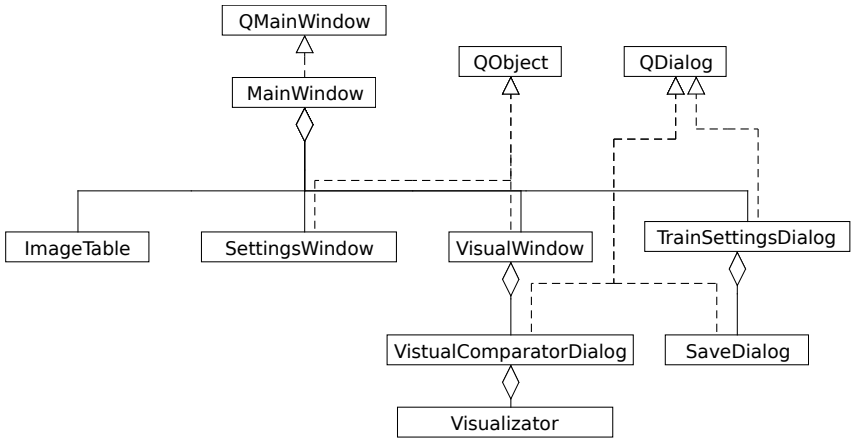
Následný výpočet vektora pre superpixel je vykonávaný metódou *analyzePPixel.*

```
std::vector<int>
BinaryAnalyzerBinaryCore::analyzePPixel(PlayerSelector &image, const
BigPixel& pixel, const std::vector<std::pair<int, int> >& core)
{
    std::vector<int> vector(core.size());

    if(!pixel.getIsPlayer()){
        for(unsigned int i = 0; i < core.size(); i++){
            vector[i] = 0;
        }
        return vector;
    }

    for(unsigned int i = 0; i < core.size(); i++){
        int x = pixel.getX() + core[i].first;
        int y = pixel.getY() + core[i].second;

    vector[i] = image.getBigPixel(x,y).getIsPlayer();
    }

    return vector;
}
```

Architektúra aplikácie

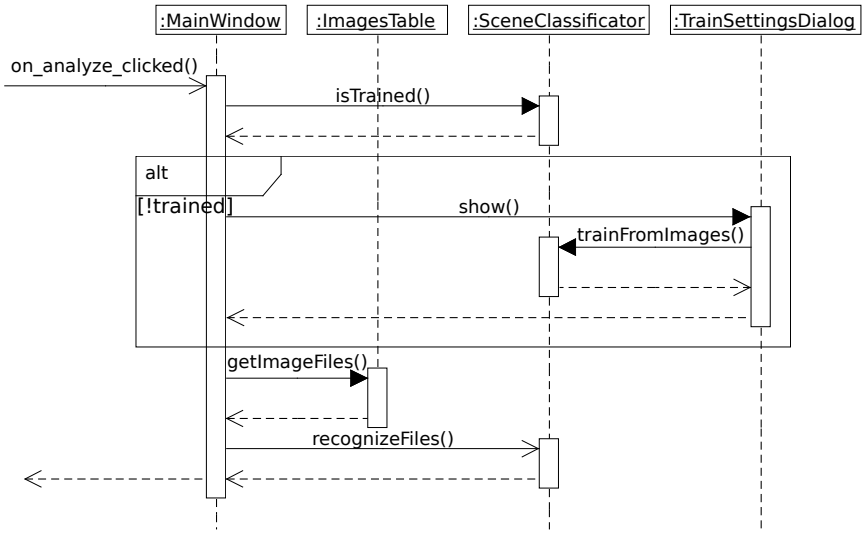
Používateľské rozhranie je založené na Qt frameworku, obrázok 51. Používateľ interaguje s triedou *MainWindow*, ktorá poskytuje základné ovládacie prvky. Komplexnejšie ovládacie prvky sú implementované v samostatných triedach:

- *ImageTable* – zobrazovanie spracovaných súborov,
- *SettingsWindow* – nastavenia príznakov,
- *VisualWindow* – vizualizácia rozpoznávania hráča.



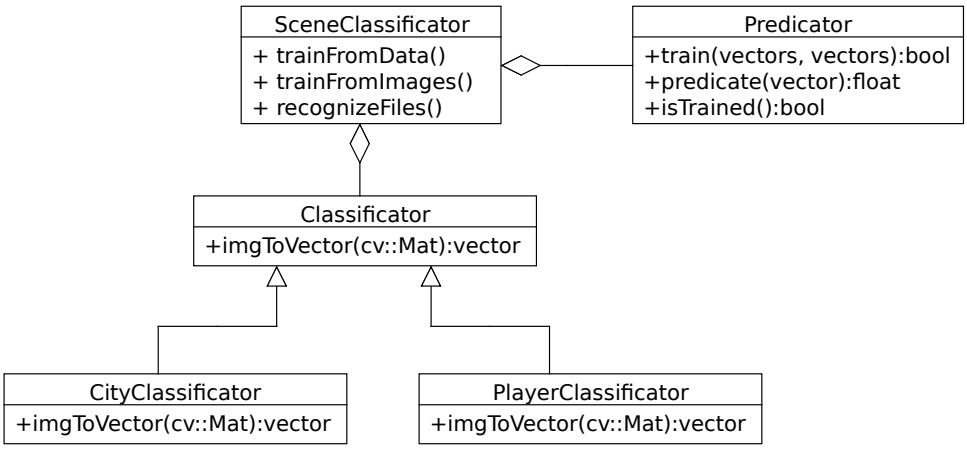
Obrázok 51: Diagram tried, používateľské rozhranie

Používateľské rozhranie je prepojené na logiku rozpoznávania scén cez *MainWindow* a *TrainSettingsDialog* na triedu *SceneClassificator*, obrázok 52.



Obrázok 52: Sekvenčný diagram, používateľské rozhranie

SceneClassificator agreguje typ *Classificator*, ktoré môže byť *CityClassificator* alebo *PlayerClassificator*. Výsledky z vytvorenia vektorov týchto tried sa následne rozpoznáva v triede typu *Predicator*, obrázok 53.



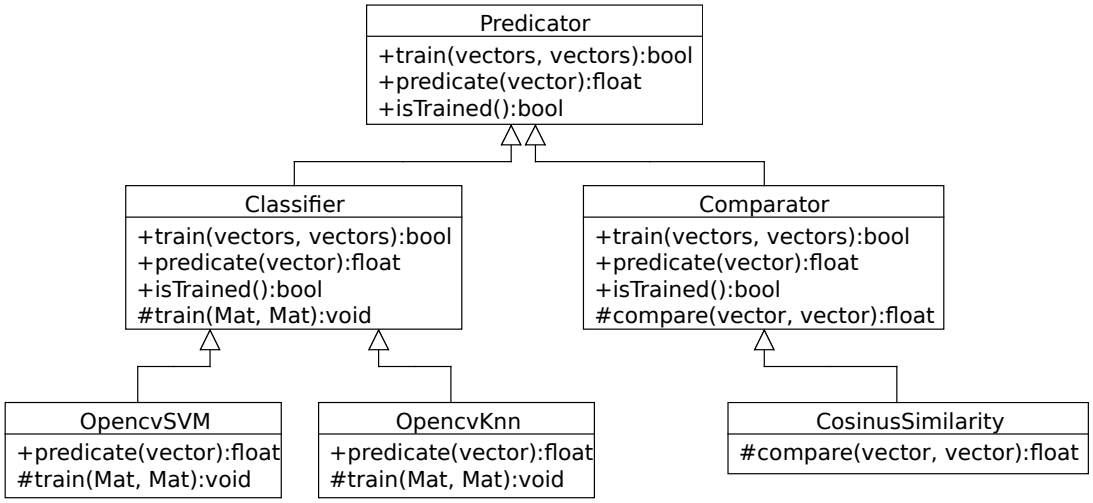
Obrázok 53: Diagram tried, rozpoznanie scén

Rozpoznávanie miest a hráčov je implementovaných podľa návrhu v kapitole 9 Návrh implementácie nástroja na rozpoznávanie scén.

Predikácia výsledkov z vektorov môže byť na základe klasifikátora ako:

- OpencvSVM klasifikátor,
- OpencvKnn klasifikátor

alebo komparátora, obrázok 54.



Obrázok 54: Diagram tried, predikcia

Object Recognition based on a Description of the Superpixels Neighborhood.

Martin GEIER*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
geier.xmartin@gmail.com

Extended abstract

Nowadays there are no limitations in audiovisual content creation. There are many devices around us, which can create such content like mobile phones, cameras etc.

This content is mostly described only by wordy description entered by its creator. This process leads to wide gap between creation of audiovisual content and it's accessing for other people. To overcome this gap it is necessary to create and combine appropriate methods of the automatic image recognition.

One of the most used approaches of the image recognition is image description by descriptors. There descriptors are classified later. In this way there were created some common used algorithms, for example algorithm for face or people recognition [2]. We use this concept in this algorithm too.

The base steps of the algorithm are:

- object segmentation, not a key part of this work, but can be done in the future,
- superpixel segmentation,
- overlaying element through the superpixels.

Input of the algorithm is already segmented image into foreground and background. The foreground represents the analyzed object and the background is an image fill. In segmented image there is necessary to cluster into superpixels which will outline the image [1]. Created superpixels are clearly divided into superpixels belonging to recognized object and superpixels belonging to background, see figure 1.

The recognition algorithm uses a structured element. The element is set over the center of superpixels belonging to recognized object. The beams of the overlayed element clearly identify the neighbors we are testing, figure 1. The algorithm decides whether the target superpixel belongs or does not belong to the recognized object. These logical values create a binary vector.

By overlaying the element through all superpixels of recognized object the algorithm creates a set of vectors. The composition of the same vectors positions creates a histogram of superpixels occurrence frequencies belonging to recognized object.

---

* Master degree study programme in field: Software Engineering
Supervisor: Dr. Vanda Benešová, Institute of Applied Informatics, Faculty of Informatics and Information Technologies STU in Bratislava

IIT.SRC 2014, Bratislava, April 29, 2014, pp. 1–2.

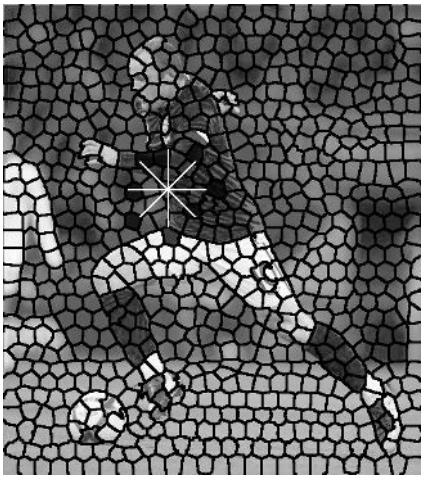


Figure 1. Element overlayed through a superpixel.

By overlaying the element through all superpixels of recognized object the algorithm creates a set of vectors. The composition of the same vectors positions creates a histogram of superpixels occurrence frequencies belonging to recognized object.

Adding more beams to the element is a simple way to change the length of descriptor and increase the ability of image recognition. Thus the element is only a composition of tested superpixels, one beam may include various test points and may test multiple superpixels in one direction.

An important factor in recognizing images is scalability invariance. Therefore it is necessary to change the structured element to always show the various dimensions of the image as the same superpixels. Element is not defined as an absolute distance from the center of the superpixel to the adjacent superpixel, but as a decimal number indicating the ratio of the length and width of the image.

Descriptor accuracy was evaluated in combination with SVM classifier. SLIC algorithm was set to make 400 superpixels. Structured element has eight beams and element shape was asymmetrical.

Input for evaluation was binary pictures, where pixel values determine foreground and background. As positive trained data were used set of forty animated football players and as negative trained data were used furty random pictures. Test data contained segmented pictures of football players from video record and horse pictures from different views.

Table 1. Method accuracy

Image type	Number of images	Correctly recognized	Accuracy
Footbal player	71	61	86%
No footbal player	71	63	89%

Global accuracy of this approach is 87.5%. We plan to evaluate our method on larger dataset.

References

[1] Achanta R., Shaji A., Smith K., Lucchi A., Fua, P., Süsstrunk, S.: SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012, pp: 2274-2282

[2] Jiang M. R., Sadka A. H., Zhou H.: Automatic Human Face Detection for Content-based Image Annotation. In: *Centre for Sensor Web Technol*, pp. 449 – 454