Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Deskriptor

Plzeň 2017 Kateřina Kratochvílová

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 16. dubna 2017

Kateřina Kratochvílová

Abstract

The text of the abstract (in English). It contains the English translation of the thesis title and a short description of the thesis.

Abstrakt

Text abstraktu (česky). Obsahuje krátkou anotaci (cca 10 řádek) v češtině. Budete ji potřebovat i při vyplňování údajů o bakalářské práci ve STAGu. Český i anglický abstrakt by měly být na stejné stránce a měly by si obsahem co možná nejvíce odpovídat (samozřejmě není možný doslovný překlad!).

Obsah

1 Úvod **6**

2 Úvod **7**
2.1 Výpočet gradientu 7
2.1.1 Vyýpočet u šedotónového obrázku 8
2.1.2 Výpočet 8
2.2 Textura 8
2.2.1 Barva 8
2.3 Kombinace vzdáleností 9
2.4 Přenesení klíčových slov 10

3 Návrh systému **11**

4 Použité programové prostředky **12**
4.1 OpenCV 12

5 Závěr **13**

6 Uživatelská dokumentace **14**

Literatura **15**

1 Úvod

V souboru `literatura.bib` jsou uvedeny příklady, jak citovat knihu [3], článek v časopisu [1], webovou stránku [2].

2 Úvod

V dnešní době, kdy je svět přesycen obrázky v digitální podobě, není vůbec snadné nalézt obrázek zobrazující požadovaný obsah. Naneštěstí počítače nedokáží vnímat obraz jako lidé, vnímají totiž obrazy jako sérii binárních informací. Přitom počítače a jejich práce s obrazy by se dala využít v mnoha oborech jako je lékařství nebo doprava. Na základě toho vyplouvá na povrch problém jak spravovat digitální obrázky a efektivně mezi nimi vyhledávat. Prostřednictvím klíčových slov přiřazených k obrázkům se dá problém vyhledávání zjednodušit. Přiřazení klíčových slov probíhá pomocí procesu automatické anotace obrázků. Klíčová slova přiřazená k obrázku by měla vyjadřovat jeho obsah (například les, strom). Při reálném použití můžeme ovšem narazit na problém při zadávání abstraktních slov, například šťastná rodina.
Pro automatickou anotaci obrázků se používá strojové učení. Můžeme ji rozdělit na dvě části. V první části získáme klíčové příznaky ve druhé už je samotná anotace, tedy přidělení klíčových slov. Abychom tento postup mohli provést v praxi, musíme nejdřív klasifikátor natrénovat pomocí trénovací množiny. Trénovací množina je množina obrázků, která již má ke každému obrázku přidána metadata s klíčovými slovy připravenými od lidí. Vybrané obrázky v trénovací množině musí být různorodé, aby anotace probíhala správně. Pojem automatická anotace obrázků je jednoduše řečeno proces, při kterém jsou k obrázku automaticky přiřazena metada, která obsahují klíčová slova.
Cílem práce je navrhnout a implementovat software umožňující automatickou anotaci obrázků. Konkrétně se bude zabývat metodou JEC. Práce bude využívat nízkoúrovňové příznaky, konkrétně barvu a texturu. Metodu budeme zkoušet na standardních datech, následně se budeme snažit výsledky vylepšít, a porovnat s další metodou a literaturou.

2.1 Výpočet gradientu

POEM deskritpro se používá jen na šedoonový obrázek, my by jsme ho chtěli na barevný obrázek

2.1.1 Vyýpočet u šedotónového obrázku

2.1.2 Výpočet

Nejprve je potřeba vypočítát gradient. Gradeint je směr růstu. Podle některých studii jsou nejlepší jednoduché masky jako je např. $[1, 0, -1]$ a $[1, 0, -1]^T$. Okraje se buď vypouštějí nebo se dají doplnit (existuje více způsobů doplňění). Masky použijeme na obrázek a vypadnou nám obrázky o rozměrech mxn každý bod původního obrázku je reprezentován jako 2D vektor. Pokud si vektor rozložíme na x a y složku dostali bychom dva obrázky jeden, který reprezentuje obrázek po použití x-ového filtru, a druhý který reprezentuje obrázek po použití y-filtru. Přičemž použití y filtru by nám mělo zvíraznit hrany v ylonovém směru (svislé) a x zvírazní hrany v x směru (vodorovné).

TODO doplnit obrázky gradient

Magnituda

Magnituda je velikost směru růstu lze si ji představit jako velikost směru růstu pro každý pixel. Počítá se pro každý pixel. Takže se dá počítat jako velikost těch 2D vektrů které jsme dostaly při výpočtu gradientu.

Velikost vektoru v rovině:

$$|u| = \sqrt{u_1^2 + u_2^2} \tag{2.1}$$

2.2 Textura

LBP i Gabor pracují s informací o intenzitě obrazu. Detekce hran. Obyčejné LBP problém s rotací.

kombinace textur a barevne informace 1. Vytvoření společného příznaku, například rozšíření LBP na všechny barvené kanály informace o barvě a textuře se mohou oblivňovat protichůdně

2. Vyhodnotit a klasifikovat příznaky odděleně a pak výslednou klasifikaci nějak s pojit z několika částí to je například JEC

výhoda zachovává vlastnosti obou původních příznaků výpočetně náročnější a jeho úspěšnost je přímo závislá na způsobu kombinace obou informací

vutbrno 117319 10 stranka

Gabor filter je lineární filter používaný pro detekci hran. Frekvence a orientace reprezentující Gabor filter je podobná lidskému vnímání a jsou zvláště vhodné pro reprezentaci textury a rozlišování.

Gabor filtr reaguje na hrany a texturové změny.

Obrázky jsou filtrovány za použité reálné části z různých odlišních Gabor filtrů jader. Průměrný a rozptyl filtrovaných snímků jsou pak použity jako

funkce pro klasifikaci, která je založena na nejméně kvadratické chyby pro jednoduchost.

TODO Dohledat Haar a Gabor wavelety, přidat vzorečky a zase klidně i obrázky

2.2.1 Barva

U digitálního obrazu je barva reprezentovaná n-rozměrným vektorem. Jeho velikost a význam jednotlivých složek (tzv. barevných kanálů) zavisí na příslušném barevném prostoru. Počet bitů použitých k uložení buď celého vektoru nebo jeho jednotlivých složek se nazývá barevná hloubka (totožně bitová hloubka). Obvykle se můžeme setkat s hodnotami 8, 12, 14 a 16 bitů na kanál.
V použité metodě získáme vlastnosti z obrázků ve třech rozdílných barevných prostorech: RGB, HSV a LAB. RGB (Red, Green, Blue) je nejobvykleji používaný pro zachycení obrázu nebo jeho zobrazení. Oproti tomu HSV (Hue, Saturation and Value) se snaží zachytit barevný model tak jak ho vnímá lidské oko, ale zároveň se snaží zůstat jednoduchý na výpočet. Hue znamená odstín barvy, saturation systost barvy a value je hodnota jasu nebo také množství bílého světla. RGB je závislý na konkrétním zařízení, nemůže dosáhnout celého rozsahu barev, které vidí lidské oko, zatímco barevný model LAB je shopen obsáhnout celé viditelné spektrum a navíc je nezávislý na zařízení. L (ve zkratce LAB) značí Luminanci (jas dosahuje hodnot 0 - 100, kde 0 je černá a 100 je bílá). Zbylé A a B jsou dvě barvonosné složky, kdy A je ve směru červeno/zeleném a B se pohybuje ve směru modro/žlutém.

Pro RGB, HSV i LAB použijeme barevnou hloubku 16 bitů na kanál histogramu v jejich příslušném barevném prostoru.

Jako reprezentace textur budou použity Gabor a Haar wavelety. Každý obrázek bude filtrován s Gabor wavelet na třech škálách a čtyřech orientacích.

2.3 Kombinace vzdáleností

Nejrozumnějším přístupem ke zkombinování vzdáleností od různých desktiptorů je aby jednotlivé vzdálenosti přispívali rovnoceně. Z tohoto důvodu je potřeba vzdálenosti přeškálovat na jednotné měřítko. Označme si I_i jako i-tý obrázek a řekněme, že máme N jeho příznaků $f_i^1, ..., f_i^N$. Nedefinujme si $d^k_{(i,j)}$ jako vzdálenost mezi f_i^k a f_j^k. Chtěli bychom zkombinovat jednotlivé vzdálenosti $d^k_{(i,j)}$, $k = 1, ..., N$ to nám poskytne vzdálenosti mezi obrázky I_i a I_j. Vzdálenosti které nám vyjdou ale v praxi nám nevyjdou tak aby měli

stejný poměr na výsledku. Předtím než vzdálenosti zkombinujeme musíme je normalizovat do jednotné formy. Získáme maximální a minimální hodnotu pro každý příznak a na základě toho hodnotu přeškláujeme na interval od 0 do 1. Když mi označíme přeškálovánou vzdálenost jako $\tilde{d}^k_{(i,j)}$ následně můžeme označit vzdálenosti mezi obrázky I_i a I_j jako (2.2). Označíme ten to vzorec na vzdálenosti jako Joint Equal Contribution (JEC)

$$JEC = \sum_{k=1}^{N} \frac{\tilde{d}^k_{(i,j)}}{N} \tag{2.2}$$

2.4 Přenesení klíčových slov

Pro přenesení klíčových slov používáme metodu, kdy přeneseme n klíčových slov k dotazovanému obrázku $\tilde{I}$ od K nejbližších sousedů v trénovací sadě. Mějme $I_i, i = 1, ..., K$,tyto K nejbližší sousedy seřadíme podle vzrůstající vzdálenosti (tzn. že I_1 je nejvíce podobný obrázek). Počet klíčových slov k danému I_i je označen jako $|I_i|$. Dále jsou popsány jednotlivé kroky alogoritmu na přenesení klíčových slov.

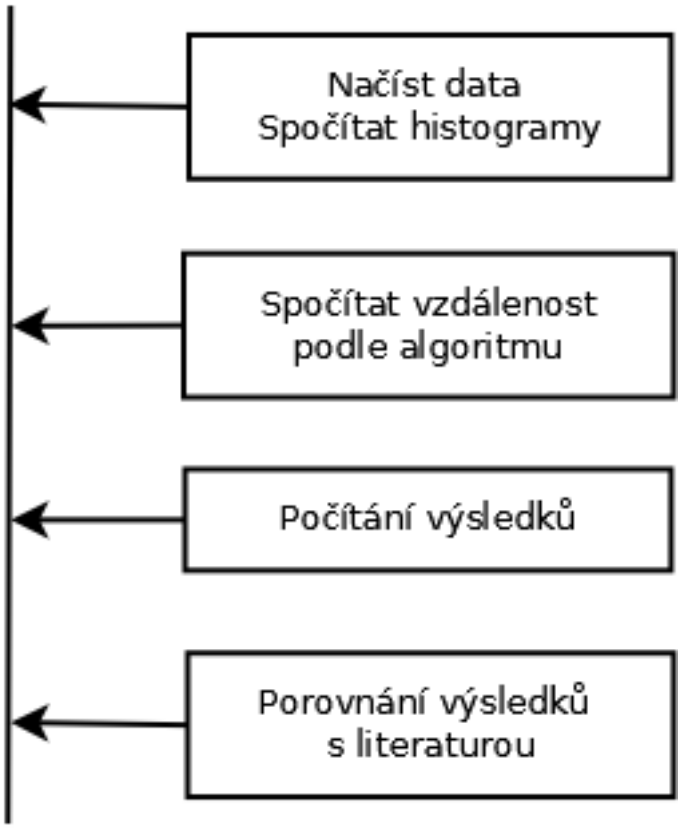
1. Seřadíme klíčová slova z I_1 podle jejich frekvence v trénovacích datech.
2. Z $|I_1|$ klíčových slov z I_1 přeneseme n nejvýše umístěná klíčová slova do dotazovaného $\tilde{I}$. Když $|I_1| < n$ pokračujte na krok 3.
3. Seřaď klíčová slova sousedů od I_2 do I_K podle dvou faktorů
   (a) Co výskyt v trénovacích datech s klíčovými slovy přenesených v kroku 2 a
   (b) místní frekvence (tj. jak často se vyskytují jako klíčová slova u obrázků I_2 až I_K). Vyber nejvyšší rankink $n - |I_1|$ klíčových slov převedených do $\tilde{I}$.

Tento algoritmus pro přenos klíčových slov je poněkud odlišný od algoritmů které se běžně používají. Jeden z běžně užívaných funguje na principu, že klíčová slova jsou vybrána od všech sousedů (se všemi sousedy je zacházeno stejně bez ohledu na to jak jsou danému obrázku podobní), jiný užívaný algoritmus k sousedům přistupuje váženě (každý soused má jinou váhu)a to na základě jejich vzdálenosti od testovaného obrázku. Při testování se ovšem ukázalo, že tyto přímé přístupy přináší horší výsledky v porovnání s použitým dvoufaktorovým algoritmem pro přenos klíčových slov.

V souhrnu použitá metoda je složenina ze svou složeniny obrázkové vzdálenosti měřítku (JEC nebo Lasso) pro nejbližší ranking, kombinuje se s výše popsaným algoritmem na přenášení klíčových slov.

3 Návrh systému

Systém byl navržen jako modulový a to z důvodu snadné obměny některé z částí, což je výhodné zejména pokud bychom potřebovali například spočítat vzdálenosti vektorů podle jiného algoritmu.



Obrázek 3.1: Návrh systému

4 Použité programové prostředky

Program byl navržen na operační systému Linux. Jako programovací jazyk byl zvolen Python a to z důvodu jeho jednoduchého použití, což je na prototyp, jako je tento velice výhodné na časovou náročnost. Program využívá knihovnu OpenCV 3.1.

4.1 OpenCV

OpenCV (Open source computer vision) je knihovna vydávána pod licencí BSD a je volně k dispozici jak pro akademické účely, tak pro komerční použití. Je vhodná pro použití v C++, C, Python a Javě. Podporuje operační systémy Windows, Linux, Mac OS, iOS a Android.
Knihovna byla navrhnuta pro výpočetní efektivitu v oblasti počítačového vidění a zpracování obrazu se zaměřením na zpracování obrazu v reálném čase. Z důvodu optimalizace byla napsána v C/C++.
Knihovnu OpenCV je možné stáhnout na adrese: http://opencv.org/

5 Závěr

V teoretické části byly popsány nízkoúrovňové příznaky barva a textura. Byla rozebrána metoda JEC, která bude v bakalářské práci implementována. Seznámili jsme se s knihovnou OpenCV, prostudovali obrázky a přiložená metadata od ČTK, ESP a iaprtc12.

6 Uživatelská dokumentace

popsani jak vypada zdrojovej soubor kterej to zere, nejdriv cesta k souboru a pak jeho klicovy slova

Literatura

[1] Hoare, C. A. R. Algorithm 64: Quicksort. *Commun. ACM*. July 1961, 4, 7, s. 321. ISSN 0001-0782. doi: 10.1145/366622.366644. Dostupné z: `http://doi.acm.org/10.1145/366622.366644`.

[2] *Class Graphics2D* [online]. Oracle, 2016. [cit. 2016/03/09]. Java SE Documentation. Dostupné z: `https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics2D.html`.

[3] Knuth, D. E. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., 1997. ISBN 0-201-89684-2.