|  |  |
| --- | --- |
| Žilinská univerzita v Žilinežilinská univerzita v žiline  Fakulta riadenia a informatikyfakulta riadenia a informatiky | |
| diplomová práca  Študijný program:  **Informačné systémy – Spracovanie dát**  Podnázov práce | |
| **Bc. František Kajánek**  **Paralelná implementácia extraktora príznakov vhodného pre detekciu objektov pomocou Adaboostu**  Vedúci práce: Ing. Peter Tarábek, PhD.  Registračné číslo: 282/2016  Žilina, 2017Bakalárska práca | |
|  |  |

|  |
| --- |
| Žilinská univerzita v Žilinežilinská univerzita v žiline  Fakulta riadenia a informatikyfakulta riadenia a informatiky |
| diplomová práca  Študijný program:  **Informačné systémy – Spracovanie dát**  Podnázov práce |
| **Bc. František Kajánek**  **Paralelná implementácia extraktora príznakov vhodného pre detekciu objektov pomocou Adaboostu**  Žilinská univerzita v Žiline  Fakulta riadenia a informatiky  Katedra matematických metód a operačnej analýzy  Žilina, 2017  Bakalárska práca |

Original zadania – scan

Čestné vyhlásenie

Poďakovanie

Abstrakt

Contents

[**Úvod** 7](#_Toc478487698)

Úvod

Teoretická časť

Počítačové videnie

Počítačové videnie je obor, ktorý sa zaoberá získavaním informácií z digitálneho obrazu a videa. Snaha je zanalyzovať, navrhnúť a naimplementovať činnosti, ktoré dokáže robiť ľudský zrakový systém. Pochopenie obrazu pre tento obor znamená transformáciu digitálneho obrazu na popisy sveta s ktorými dokážu pracovať iné myšlienkové procesy. Za účelom využitia vizuálnych dat možno použiť rôzne techniky z geometrie, fyziky, štatistiky a teorie učenia. Pod dátami, ktoré počítačové videnie používa, môžu byť ľubovoľné videá, rôzne pohľady z rôznych kamier, multi-dimenzionálne dáta zo zdravotných skenerov alebo aj len jednoduchá fotografia. Medzi podobory počítačového videnia patria napríklad rekonštrukcia scény, detekcia udalostí, sledovanie videa, detekcia objektov, učenie, indexovanie alebo aj predpovedanie pohybu.

Deskriptory/Klasifikátory

Deskriptory a klasifikátory nám umožňujú zobrať ľudoboľný obraz/video a z neho dostať nejaké dáta a úvahy. Väčšinou je potrebné testovať ich rôzne kobinácie a zistiť, ktorá funguje na danú úlohu najlepšie, v rámci výkonnosti a rýchlosti.

Obrazový deskriptor je popis nejakých vizuálnych vlastností obrazu alebo videa, alebo taktiež algoritmus alebo aplikácia, ktorá tento popis vytvorí. V dnešnej dobe komunikačných technológií a internetu je potrebné rýchlo a spoľahlivo spracovávať a analyzovať dáta nám dostupné. Presne preto sú potrebné deskriptory, aby sme mali dostupné data na riešenie rôznych úloh.

Rozdeľujú sa na dve skupiny:

1. Všeobecné informačné deskriptory – hlavné typy popisujú podľa farby, tvaru, regionov, textury a pohybu
2. Informačné deskriptory špecifickej domény – riešia už nejakú špecifickú úlohu napríklad detekcia chodcov alebo sledovanie pohybu áut na parkovisku

Príklady využití deskriptorov sú napríklad multimédiá, kde chceme aby nám systém odporúčal obsah, ktorý by sme chceli konzumovať, triedenie súborov, aby sme nemuseli pomenúvať súbory a aby to program urobil za nás, alebo aj ako medzistupeň pre komplexnejšie systémy, napríklad samoriadiace autá alebo rozšírená realita.

Klasifikácia je popis počas ktorého rozpoznáme, odlíšime a pochopíme rôzne typy objektov. Lineárny klasifikátor dokáže urobiť rozhodnutie pomocou hodnôt objektu lineárnou kombináciou charakteristík. Charakteristiky vstupného objektu na klasifikáciu sa volajú hodnoty čŕt a hodnoty objektu sa zväčša dodávajú klasfikátoru vo vektore čŕt.

Klasifikátor vezme vektor čŕt a transformuje ho do bodu v N-dimenzionálnom priestore, a nejaká funkcia f, ktorá rozdelí tento priestor na dve časti. Podľa toho do ktorej časti patrí náš bod, je aj výsledná klasifikácia. Výhodou lineárnych klasifikátorov je ich jednoduchosť, čo následne umožňuje vysokú výpočetnú priepustnosť dát.

Klasifikátory nachádzajú využitie v oblasti strojového učenia, kde pomáhajú pri rozhodovaní v jednoduchých ale aj komplexných systémoch. Medzi príklady klasifikátorov patria napríklad Bayesovsky klasifikátor, Fisherov lineárny diskriminant alebo SVM (Support Vector Machine).

Klasifikátory väčšinou vyžadujú určitý tréning, pri ktorom sa zoberie optimalizačný algoritmus, ktorý sa snaží minimalizovať chybu. Do tohto algoritmu sa dodajú sa vstupy a výstupy pre danú trénovaciu sadu a ako výstup máme matematický model, ktorý dokáže rozhodovať o daných dátach. (TODO: obrázok nakresliť)

AdaBoost + slabé klasifikátory

AdaBoost je meta-algoritmus pre strojové účenie. Väčšina algoritmov využíva klasifikátory tak, že natrénuje jeden silný klasifikátor, ktorý rozhodne o nejakej úlohe. Takýmto algoritmom je napríklad SVM. AdaBoost ide o krok ďalej, a on nepracuje so samotnými dátami, ale pracuje s týmito klasifikátormi. Väčšinou sa používajú tzv. slabé klasifikátory, ktorých šanca urobiť chybu je menšia ako 50%, že rozhodnutie, ktoré urobia je správne. AdaBoost priradí týmto slabým klasifikátorom určitým spôsobom váhu, podľa toho ako veľmi daný klasifikátor ovplyvní finálny verdikt. Ako výsledok tohto procesuje je jeden silný klasifikátor, ktorý využijeme na danú úlohu. (obrázok)

Ako slabé klasifikátory sa zvyčajne používajú rozhodovacie stromy rôznych spôsobov. V práci (pridať prácu ktorá vylepšovala hog od dalal triggsa) používali ako slabé klasifikátory SVM klasifikátory, za účelom zjednodušenia viacdimenzionálnej úlohy na lineárnu.

Spôsob akým dostaneme váženie závisí od verzie AdaBoostu. Medzi hlavné odnože patria:

1. AdaBoost
2. Real AdaBoost
3. GentleBoost
4. LogitBoost

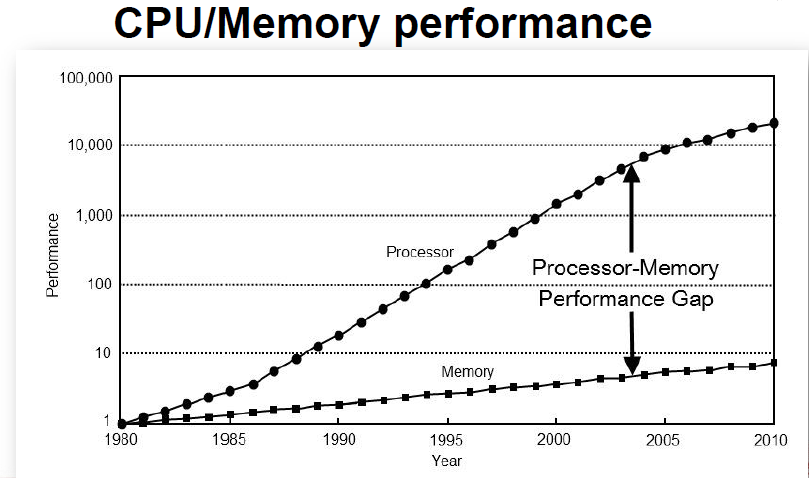
Adaboost a spojenie s deskriptormi

Viola-Jones kaskáda

Rýchlosť výpočtov

V počítačovom videní bolo vždy potrebné rýchle spracovanie dát. Vo väčšine prípadov uvažujeme, že najlepšie riešenie je nájsť taký popis alebo také vyhodnotenie, ktoré nám dodá čo najlepší výsledok. V tejto a taktiež v mnohých iných oblastiach je potrebné optimalizovať nielen dosiahnutý výsledok, ale aj rýchlosť algoritmu.

V dnešnej dobe je jasným trendom pri zvyšovaní výkonu, zvyšovať počet jadier v procesore, či už ide o CPU alebo GPU. Kvôli tomu sú potrebné algoritmy, ktoré dokážu bežať paralelne a bez prílišnej synchronizácie medzi procesmi. Dostupný hardware umožňuje nielen zdieľané výpočty na jednom fyzickom procesori ale aj taktiež medzi rôznymi počítačmi/grafickými kartami.

Ďalším a narastajúcim faktorom pri rýchlosti algoritmu je využívanie pamäte. Zatiaľčo FLOPs (floating point operations per second) procesorov narastajú podľa Moorovho zákona, rýchlosť pamäte narastá omnoho pomalšie, čo spôsobuje, že veľa algoritmov má úzky profil v oblasti využitia pamäte. Tento problém je veľmi výrazný hlavne na GPU architektúrach, kde individuálne vlákna zdieľajú pamäťové zbernice a taktiež cache pamäť. Na CPU je tento problém čiastočne eliminovaný komplikovanou logikou predikcie a L1/L2/L3 cache pamäťami. Jedným z bežných problémov spôsobujúcich uzky profil pri práci s pamäťou sú časté izolované prístupy do pamäte.

Obrázok 1-Rozdiel medzi nárastom výkonu CPU a pamäte

Zdroj:Computer Architecture, a quantitative approach; Hennessy,Patterson,Arpaci-Dusseau

Vo výsledku tieto faktory spôsobujú, že niektoré algoritmy sú ťažko využiteľné v praxi. Často je možné vidieť algoritmy, ktoré sú len malou zmenou oproti pôvodnému konceptu a napriek tomu tento obmenený algoritmus už je možné využiť pri výpočtoch v reálnom čase. Ako príklad je úprava HOG deskriptoru na LiteHOG verziu (pridať referenciu) alebo snaha aproximovať niektoré medzistupne výpočtu HOG deskriptoru pomocou integrálneho obrazu (pridať referenciu na BC)

OpenCV

OpenCV je knižnica programovacích funkcií, ktorá slúži hlavne na prácu s obrazom v reálnom čase. Je dostupná pod BSD licenciou. Možno ju používať na viacerých platformách, napríklad Windows, Linux, OS X, Android, iOS. Obsahuje moduly na úpravu videa, rozpoznávanie objektov, sledovanie pohybu, segmentáciu obrazu, rozšírenú realitu, mobilnú robotiku a taktiež obsahuje podporu na paralelné výpočty na CPU aj GPU.

Prvotne bola knižnica vyvíjaná pobočkou Intelu v Nižnom Novgorode, odvtedy sa k podpore pridala Willow Garage a Itseez. V auguste 2012 prevzala vývoj knižnice nezisková organizácia OpenCV.org [7], ktorá udržuje stránku pre developerov a používateľov, vrátane online dokumentácie [8].

Väčšina knižnice je napísaná v jazyku C++. Taktiež je dostupná stará verzia v jazyku C. Jazyky Java, Python, MATLAB/OCTAVE majú dostupné úplné wrappery na C++ jadro a taktiež existujú verzie pre C#, Perl a Ruby, ktoré boli vyvinuté za účelom rozšírenia záujmu o túto knižnicu. Od roku 2010 sa taktiež vyvíjajú CUDA verzie dostupných algoritmov a od roku 2012 pre OpenCL.

V tejto práci sa hlavne čerpá z modulu na detekciu objektov, Viola-Jones kaskády, ktorá je dostupná ako separátna kompilovateľná aplikácia a z GPU modulov pre CUDA platformu.

Praktická časť

Porovnávanie implementácií AdaBoostu

Popis našich vlastných pokusov

Analýza Viola-Jones kaskády

Implementácia rozšírenia na CPU

Možnosti rozšírenia na GPU

Záver