|  |  |  |
| --- | --- | --- |
| Logo__SSPU_2016_Barva | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Využití OpenCV a Haar Cascades** | | |
| Dominik Endrych | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2016/2017 | |

Poděkování

*Chtěl bych poděkovat všem učitelům, kteří mi pomáhali při vývoji projektu, především pak panu učiteli Ing. Petru Grussmannovi za konzultaci postupu a vylepšení. Dále bych chtěl poděkovat všem spolužákům, kteří řešili podobné téma a podělili se se mnou o své poznatky.*

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité   
informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové   
a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2016

*podpis autora práce*

**ANOTACE**

Tato práce přibližuje problematiku sledování aut (popř. jiných objektů) s využitím knihovny OpenCV. Ukazuji zde postup při využití Haar Cascades a natrénování vlastního classifieru. Při tomto trénování však potřebujeme spoustu snímků a proto jsem se v Pythonu snažil vytvořit jednoduchý program, který by sběr snímků ulehčil. Program spustí námi určené video z kamery a umožní vybrat snímky přímo za běhu videa.

**Klíčová slova**

OpenCV, Python, Haar Cascades, classifier, sledování aut

OBSAH

[OBSAH 5](#_Toc502490352)

[Úvod 6](#_Toc502490353)

[1 Knihovna OpenCV 7](#_Toc502490354)

[1.1 Jednoduchý zkušební program v OpenCV 7](#_Toc502490355)

[1.2 Video v OpenCV 8](#_Toc502490356)

[1.2.1 Video soubor 8](#_Toc502490357)

[1.2.2 Webkamera 8](#_Toc502490358)

[2 Haar Cascade 9](#_Toc502490359)

[3 Postup trénování vlastního classifieru 10](#_Toc502490360)

[3.1 Úvod 10](#_Toc502490361)

[3.2 Pozitivní a negativní snímky 10](#_Toc502490362)

[3.3 Vytváření vzorků 11](#_Toc502490363)

[3.4 Trénování classifieru 11](#_Toc502490364)

[3.5 Průběh trénování 11](#_Toc502490365)

[4 využití grafické knihovny ke shromáždění snímků 13](#_Toc502490366)

[4.1 Úvod 13](#_Toc502490367)

[4.2 Knihovna GTK 13](#_Toc502490368)

[4.3 Princip fungování programu 13](#_Toc502490369)

[5 Výsledky a výstupy práce 14](#_Toc502490370)

[Závěr 15](#_Toc502490371)

[Seznam použitýCH INFORMAČNÍCH ZDROJů 16](#_Toc502490372)

[Seznam příloh 17](#_Toc502490373)

Úvod

Sledování objektů skrze kameru je v dnešní době běžná věc, ať už jde o měření rychlosti na silnicích nebo bezpečnostní kamery na náměstí. Každý si ale nemůže dovolit drahou kameru nebo senzor. Díky knihovně OpenCV nám ,ale stačí ke sledování našeho pozemku nebo příjezdové cesty pouhá webkamera a trochu programování.

Cílem tohoto projektu je přiblížit jednoduchou metodu natrénování vlastního classifieru s využitím Haar Cascades. Vysvětlím, jaké snímky na trénink pořebujeme, co se s nimi bude v procesu dít a jak bude vypadat výsledný soubor s příponou .xml.

Sbírání pozitivních a negativních snímků může být zdlouhavý proces, proto jsem se také snažil vytvořit program, který nám ušetří čas i námahu. V této dokumentaci tedy vysvětlím, na jakém principu můj program funguje.

# Knihovna OpenCV

OpenCV (Open Source Computer Vison) je knihovna funkcí, která se zaměřuje především na „počítačovou vizi.“ Tzn. Jak počítače dokážou rozumět digitálním obrázkům nebo videu. Chceme tedy, aby počítače byly schopny toho, co lidská vizuální soustava. Data, která pomocí OpenCV získáme, můžeme poté ukládat do různých databází, grafů atd. OpenCV bylo původně vyvíjeno společností Intel a nyní je poskytováno pod open-source BDS licencí.

OpenCV je napsáno v C++ a jeho primární interface je C++. Bez problému lze však použít také Python, Java C# nebo Ruby. Všechny nové algoritmy jsou vyvíjeny v C++. Knihovna je multiplatformní a tak ji lze využít na Windows, Linux, maxOS nebo FreeBSD.

K nejpoužívanějším algoritmům v OpenCV patří:

* Rozpoznávání obličeje
* Rozpoznávání gest
* Interakce člověk-počítač
* Mobilní robotika
* Zachycení pohybu
* Rozpoznání pohybu
* Identifikace objektů

## Jednoduchý zkušební program v OpenCV

Správnou funkčnost knihovny OpenCV lze ověřit velmi jednoduchým programem v C++ na otevření obrázku v novém okně.

Mat img; //objekt, ve kterém bude obrázek uložený

img = imread(argv[1], CV\_LOAD\_IMAGE\_COLOR); //funkce "iamread" načte soubor

namedWindow("Zobrazeny obrazek", WINDOW\_AUTOSIZE); //vytvoří okno pojmenované podle argumentu

imshow("Zobrazeny obrazek", img); //zobrazí obrázek uvnitř zvoleného okna

## Video v OpenCV

OpneCV nepracuje přímo s videem jako takovým. Video si naopak rozkouskuje na jednotlivé snímky a poté je přehrává velmi rychle za sebou s frekvencí definovanou samotným programátorem. Pokud je tato frekvence dostačující, vzinke jakási „iluze“ přehrávání videa.

V OpenCV lze spustit 2 druhy videa. Můžeme spustit jak video soubor přímo uložený v systému nebo video z webkamery. V obou případech je přístup prakticky stejný.

### Video soubor

Tato část kódu v C++ slouží ke spuštění uloženého video souboru.

VideoCapture cap("wildlife.wmv"); //otevře zvolené video

if(!cap.isOpened()) //jestli se video neotevře, program se ukončí

{

cout<<"Video se nepodarilo otevrit"<<endl;

return -1;

}

Zde si poté programátor pomocí funkce *waitKey()* určí refresh frekvenci.

Mat frame;

cap >> frame; //postupně zachycuje snímky

if(frame.empty()) //jsetli je snímek prázdný,hned break

break;

imshow("Prehravane video", frame); //zobrazí aktualní snímek

char c=(char)waitKey(25); //refresh frekvence

### Webkamera

Pokud je webkamera přímo připojena do počítače, lze k ní přistupovat velmi podobně jako k souboru. Místo názvu souboru stačí napsat pouze „0“

VideoCapture cap(0); //otevře webkameru

# Haar Cascade

Haar Cascade je classifier, který používáme pro detekci objektů, pro které byl vytrénován. Classifier musíme natrénovat použitím spousty pozitivních a negativních snímků. Pozitivní snímky obsahují objekt (auto, obličej, předmět), který chceme rozpoznávat. Negativní snímky jsou snímky, které daný objekt neobsahují. Nejlépe pozadí, na kterém by se objekt mohl vyskytovat. Doporučuji používat více negativních než pozitivních snímků, aby classifier uměl lépe a přesněji poznat daný objekt na různém pozadí. Také bych doporučil používat snímky, které mají stejné rozměry. Zkrátí nám to především samotný proces tréninku. Trénování probíhá na několik fází. Čím více snímků použijeme, tím déle to bude trvat, ale získáme lepší a „chytřejší“ classifier. V každé fázi se pozitivní snímky náhodně přiřazují na negativní snímky a tím classifier získává přehled o tom, jak sledované objekty vypadají.

Když classifier natrénujeme, můžeme jej použít na libovolná videa nebo obrázky (samozřejmě s podporou námi napsaného programu na samotnou detekci). Classifier nám vrátí „1“ pokud ve skenovaném regionu najde sledovaný objekt (v opačném případě vrátí „0“). Skenování a hledání objektu probíhá na celém obrázku.

Slovo „cascade“ používáme, protože výsledný classifier je složen z jednoduších fází. Každá fáze se poté aplikuje na skenované video nebo obrázke, dokud podezřelý objekt je buď potvrzen jako sledovaný objekt nebo vyvrácen.

# Postup trénování vlastního classifieru

## Úvod

Z Internetu si samozřejmě můžeme stáhnout již natrénované soubory. Není ale zaručeno, že budou správně detekovat objekty našeho zájmu. V této kapitole tedy ukážu, jak si tento „chytrý“ soubor můžeme vyrobit. Začneme tedy stažením složky z github.com, v níž najdeme již předpřipravené prostřední pro naše snímky. Nemusíme tedy všechny složky vytvářet sami.

git clone <https://github.com/mrnugget/opencv-haar-classifier-training>

## Pozitivní a negativní snímky

Abychom mohli natrénovat vlastní classifier, potřebujeme spoustu snímků, které ukazují objekt, který chceme detekovat (pozitivní) a ještě víc snímků, bez daného objektu (negativní). Není přesně známo, kolik je potřeba těchto snímků. Záleží to na spoustě faktorech – kvalita obrázků, výkon našeho CPU nebo charakter samotného objektu. Obecně ale platí, že čím víc snímků máme, tím líp. Více snímků ale znamená delší proces treninku, proto musíme brát v potaz sledovaný objekt samotný. Jestli sledujeme objekt, který se víceméně nemění v barvě, tvaru nebo úhlu pohledu, postačí nám méně snímků.



*příklad pozitivního snímku příklad negativního snímk*u

Všechny pozitivní snímky nyní musíme přesunou do složky ./*positive\_images* a všechny negativní snímky do *./negative\_images*. Po přesunutí musíme vytvořit seznam snímků a cest k nim.

find ./positive\_images -iname "\*.jpg" > positives.txt

find ./negative\_images -iname "\*.jpg" > negatives.txt

## Vytváření vzorků

Dalším krokem je vytvoření vzorků pro trenink. Všechny obrázky zapsané v *negatives.txt* nám poslouží jako negativní vzorky. K vytvoření poztivních vzorků můžeme použít spoustu metod. Zde budeme používat nástroj, který nám nabízí OpenCV: *opencv\_createsamples*. Tento nástroj nám na konci vytvoří *\*.vec* soubor, který můžeme použít na trénování classifieru. *Opencv\_createsamples* vygeneruje velké množství pozitivních vzorků tím, že na pozitivní obrázky aplikuje transftormace a deformace.

Nyní máme spoustu *\*.vec* souborů, které potřebujeme sloučit dohromady. Toto provedeme spuštěním nástroje, jehož výrobcem je Naotoshi Seo a který jsme si stáhli z github.com. Po spuštění nástroje *mergevec.cpp* se všechny soubor zkombinují do jednoho *\*.vec* souboru. Tento sloučený soubor použijeme při trénování.

## Trénování classifieru

OpenCV nabízí dvě metody trénování Haar classifieru: *opencv\_haartraining* a *opencv\_traincascade.* Já jsem použil *opencv\_traincascade,* protože podporuje použití více jader procesoru a je kompatibilní s novější verzí OpenCV. Nejprve musíme *opencv\_traincascade* přesměrovat na pozitivní vzorky a negativní obrázky, do jaké složky chceme, aby classifier vytvořil a velikost vzorků *(-w* a *-h*). *-numNeg* říká, kolik máme negativních vzorků a *-numPos* by mělo být měnší než počet vzorků, které jsme vygenerovali. *-precalcValBufSize* a -*precalcIdxBufSize* říká, kolik si má trenink vzít paměti.

opencv\_traincascade -data classifier -vec samples.vec -bg negatives.txt\

-numStages 20 -minHitRate 0.999 -maxFalseAlarmRate 0.5 -numPos 1000\

-numNeg 600 -w 80 -h 40 -mode ALL -precalcValBufSize 1024\

-precalcIdxBufSize 1024

## Průběh trénování

V průběhu trénování je možné vidět na obrazovce analízu jednotlivých fází:

===== TRAINING 0-stage =====

<BEGIN

POS count : consumed 1000 : 1000

NEG count : acceptanceRatio 600 : 1

Precalculation time: 11

+----+---------+---------+

| N | HR | FA |

+----+---------+---------+

| 1| 1| 1|

+----+---------+---------+

| 2| 1| 1|

+----+---------+---------+

| 3| 1| 1|

+----+---------+---------+

| 4| 1| 1|

+----+---------+---------+

| 5| 1| 1|

+----+---------+---------+

| 6| 1| 1|

+----+---------+---------+

| 7| 1| 0.711657|

+----+---------+---------+

| 8| 1| 0.53|

+----+---------+---------+

| 9| 1| 0.315|

+----+---------+---------+

END>

Training until now has taken 0 days 2 hours 46 minutes 19 seconds.

Každý řádek reprezentuje vlastnost, která je zrovna trénována a také obsahuje informaci o jejím HitRatio a FalseAlarm ratio. Pokud si jednotlivé fáze (*stage*) zvolí pouze velmi málo vlastností (např. *N = 2*), je možné, že s vašimi daty není něco v pořádku nebo jste zvolili málo snímků a classifier tak bude méně „chytrý.“

Na konci každé fáze se classifier uloží a aktualizuje. Proces trénování je tedy možné pozastavit a k procesu se vrátit později. To se hodí např. pokud potřebujete restartovat počítač.

Jakmile se proces ukončí najdeme v námi zvoleném adresáři soubor s názvem *classifier.xml.* Tohle je náš finální „cyhtrý“ classifier a můžeme jej začít používat na sledování objektů.

# využití grafické knihovny ke shromáždění snímků

## Úvod

V této části dokumentace bych se chtěl zmínit o možném vylepšení a urychlení sběru snímků pro Haar Cascade trénink. Jde o program, naprogramovaný v jazyce Python, s využitím grafické knihovny GTK.

## Knihovna GTK

GTK je multi-platformní nástroj pro vytváření grafického uživatelského rozhraní. Nabízí obrovskou sadu widgetů, proto je vhodná pro malé programy i rozsáhlé aplikace. Knihovna podporuje velkou řadu jazyků jako C/C++, Perl nebo Python, který můžeme použít v kombinaci s programem Glade.

## Princip fungování programu

Program je složen poskládáním několika GTK widgetů nad sebe. Nejprve musíme do pozadí okna spustit video. Tato knihovna bohužel nepodporuje přehrávání videa, protože objekt *gtkImage* je určen pouze pro zobrazení obrázků. Nejlepším řešením tohoto problému je použití frameworku Gstreamer, který slouží ke streamování videa.

Na přehrávané video poté vložíme objekt *gtkDrawingArea.* V této kreslící oblasti myší (pomocí souřadnic dvou bodů) nakraslíme kolem objektu na videu geometrický útvar (čtverec nebo obdélník). Vnitřní obsah tohoto útvaru se uloží jako screenshot do námi definované složky s pozitivními snímky, které použijeme na trénink.

# Výsledky a výstupy práce

Výsledkem této práce je především plně funkčí a „chytrý“ soubor pro rozopznávání aut projíždějích po dálnici. Tento soubor má příponu .xml a jeho obsah vypadá takto:

<stages>

<\_>

<maxWeakCount>11</maxWeakCount>

<stageThreshold>-1.2678639888763428e+00</stageThreshold>

<weakClassifiers>

<\_>

<internalNodes>

0 -1 0 -4.8783610691316426e-04</internalNodes>

<leafValues>

5.9219348430633545e-01 -4.4163608551025391e-01</leafValues></\_>

<\_>

<internalNodes>

0 -1 1 -4.2209611274302006e-04</internalNodes>

<leafValues>

3.0318650603294373e-01 -3.2912918925285339e-01</leafValues></\_>

<\_>

<internalNodes>

0 -1 2 -4.9940118333324790e-04</internalNodes>

<leafValues>

4.8563310503959656e-01 -4.2923060059547424e-01</leafValues></\_>

Dále jsem se také věnoval programu na usnadnění sběru snímků. Program ještě není zcela funkční, protože se mi nepovedlo spojit jednotlivé funkce dohromady. Prozatím mám tedy funkční jednotlivé komponenty v samostatných aplikacích.

**Závěr**

Využití OpenCV je mnoho. Cílem práce bylo vysvětlit problematiku Haar Cascade trénování a vytvoření vlastního classifieru na sledování projíždějíchc aut. Máme tedy hotový soubor, který je přenositelný a můžeme jej použít v různých OpenCV programech, které sledují a rozpoznávají objekty.

Po spojení jednotlivých GTK aplikací by bylo možné také zkrátit proces shromažďování pozitivních a negativních snímků. To se mi zatím bohužel nepovedlo.

Ukazka co se Vam povedlo

Seznam použitýCH INFORMAČNÍCH ZDROJů

###### [0] OpenCV Dokumentace [online]. [cit. 2017-12-31]. Dostupné z: https://opencv.org/

[2] OpenCV 3.3 [online]. 2017 [cit. 2017-12-31]. Dostupné z: https://opencv.org/opencv-3-3.html

[3] Haar Cascades - Github [online]. 2017 [cit. 2017-12-31]. Dostupné z: https://github.com/mrnugget/opencv-haar-classifier-training

[4] GTK Dokumentace [online]. [cit. 2017-12-31]. Dostupné z: https://www.gtk.org/documentation.php

Seznam příloh

č. 1 Titulní list

č. 2 Čestné prohlášení

č. 3 Poděkování