Detektor poznávacích značek aut

Tereza Černá
Klára Vaňková
Jakub Kvita

Příprava dat

Pro detekci značek jsem použila hned několik přístupů, od detekce pomocí hranového detektoru, kontur, MSERů, přes snahu o pochopení konvolučních neuronových sítí (conv-net, Caffe) až po trénování kaskádového klasifikátoru knihovny OpenCV.

Kaskádový detektor vyžaduje pro trénování seznamy pozitivních a negativních případů, tak byly vytvořeny aplikace pro přípravu dat. První aplikace **training---find-LP** je jednoduchý detektor založený na detekci MSER oblastí, které jsou uloženy do obalujícího boxu. Pro nalezené oblasti se kontroluje, zda jsou vodorovné se spodním okrajem vstupního obrázku, případně projdou oblasti s malým sklonem, a zda mají odpovídající poměr stran. Dále probíhá analýza pixelů v nalezené oblasti. Oblast je převedena na binární obraz, je spočítán počet pixelů v každém sloupci matice. Poté se prochází jednotlivé sloupce zleva doprava a kontroluje se, zda se v celé matici nacházejí místa, kde jde větší počet pixelů a kde menší. V nalezených místech se předpokládá znak poznávací značky. Pokud se v matici nenachází tato zajímavá místa, pak matice nepředstavuje SPZ. Tento skript se může spouštět se dvěma parametry. Je možné specifikovat maximální šířku SPZ ve vstupním obraze, zadává se poměrem ku šířce vstupního obrazu. Druhým parametr specifikuje velikost pruhu okolo SPZ, který se se bude uvažovat pro trénink společně s nalezenou značkou. Výstupem této aplikace je adresář dataset-LP obsahující nalezené SPZ a soubor LP-list.dat obsahující seznam souborů v adresáři dataset-LP. V této chvíli je potřeba projít adresář dataset-LP a smazat všechny obrázky, které nepředstavují SPZ auta.

Následně se použije aplikace **training---prepare-data**, která vezme na vstupu soubor LP-list.dat a prochází postupně všechny SPZ a kontroluje, zda byly uživatelem ponechány či smazány. Pro všechny pravdivě nalezené SPZ se vytvoří adresář positive-dataset, který obsahuje tyto značky, a soubor pos-list.dat se seznamem positivních vzorků v podobě:

```
cesta-k-obrázku počet-značek x y šířka výška x y šířka výška…
```

kde x a y představují pozici levého horního rohu SPZ v originálním obrázku. Dále se vytvoří adresář negative-dataset s originálními obrázky, ve kterých jsou začerněny SPZ, pro trénování negativních případů, a seznam těchto obrázků se uloží do souboru neg-list.dat. Následně musí uživatel projít adresář negative-dataset a začernit všechny zbývající SPZ, které nebyly jednoduchým detektorem nadetekovány.

Trénování klasifikátoru

Aplikace pro trénování klasifikátoru jsou v základu knihovny OpenCV. Bylo potřeba využít dvě funkce **opencv_createsamples** a **opencv_traincascade**. Pro trénování i testování byly zvoleny obrázky dopravy pořízené na území města Brna, v nichž je SPZ vodorovná se spodní hranou obrázku, případně s mírným sklonem.

Funkce **opencv_createsamples** vytváří pozitivní vzorky z jednoho obrázku či kolekce obrázků. Rozměry vzorků jsou dány ve stejném poměru, jako je rozměr SPZ.

```
./opencv_createsamples
        -info /$PATH/pos-list.dat
```

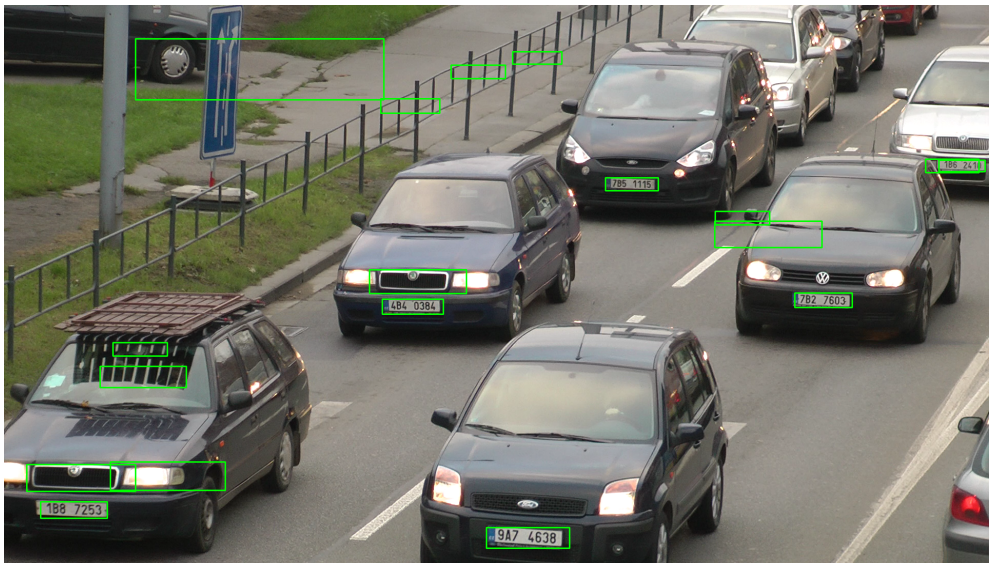
```
-vec  /$PATH/samples.vec
-w 45 -h 12
-num 3000
```

Samotné trénování probíhá ve funkci **opencv_traincascade**. Trénování probíhá v několika iteracích, je nutné dávat pozor, aby nebyl klasifikátor již přetrénovaný. Mě se osvědčilo 14 iterací. Dále bylo možné zvolit typ detektoru, pro tento účel byl zvolen detektor HAAR, který je sice pomalejší, ale poskytuje lepší výsledky, a algoritmus Gentle AdaBoost.

```
./opencv_traincascade
    -data /$PATH/classifier
    -vec  /$PATH/samples.vec
    -bg       /$PATH/neg-list.dat
    -numPox 976 -numNeg 976
    -numStages 14
    -w 45 -h 12
    -featureType HAAR
    -bt GAB
```

Výstup detektoru obsahuje spoustu šumu, ale převážnou většinu SPZ. Tento šum nevadí, protože se nad ním zkouší klasifikovat písmo a tyto chybně nadetekované plochy neprojdou zkouškou na znaky.

Natrénovaný detektor poskytuje až 95% úspěšnost. Závisí to na kvalitě vstupního datasetu. Při testování na horším setu obrázků byla úspěšnost kolem 75%. Detektor má problém se značkami, které jsou velmi malé vlivem vzdálenosti od kamery. Částečně umí najít SPZ, které jsou z části skryté za jiným autem. Úspěšnost byla získána na základě spočítání správně nadetekovaných a zapomenutých SPZ.



Rozpoznávání poznávacích značek aut

Nejprve jsme zvažovali dva možné přístupy. K rozpoznání textu je veřejně dostupná knihovna Tesseract, která pro vstupní obrázek vrátí přepis textu v něm. Její výsledky na dopravních značkách nebyly dost dobré, ale bylo možné vyzkoušet natrénovat Tesseract na pro nás vhodných fontech a lépe zvolit parametry pro klasifikátor. Druhou možností je vlastní detekce písmen v obrázku a následně jejich klasifikace. V tomto případě není jisté, zda se vůbec dokážeme rozpoznat alespoň nějaké písmena. Rozhodli jsme se vyzkoušet oba přístupy.

Rozpoznání s pomocí knihovny Tesseract

Při zpracování s Tesseractem bylo důležité přijít na to, jaký formát dat knihovna vyžaduje. Zjistili jsme, že šum, kterým byly okraje rozpoznané značky, Tesseractu velmi vadí a není poté schopen nic rozpoznat. Po filtraci obrázku a odmazání všeho co nebyly písmena, se úspěšnost rapidně navýšila, ale při hodnocení celých poznávacích značek, ve většině se alespoň jedno písmenko špatně klasifikovalo a v některých případech se detekovalo více symbolů. Například přebývající dvojtečka. Tehdy už jsme naráželi na jiné limity knihovny, a sice jsme mu dávali data v malém rozlišení. Tesseract je v základu rozpoznávač naskenovaných stránek textu ve vysokém rozlišení a pro písmena o velikosti blížící se 10 pixelům na výšku přestává fungovat. Naše data měla původně výšku i 7-8 pixelů, u malých značek. Také moc nepomáhalo, že máme pouze jeden řádek sedmi znaků a ne více textu, kde je na čem se zachytit.

Úspěšnost by se dala zvýšit natrénováním knihovny na náš SPZ font, o což jsme se pokusili, ale jak jsme uvedli, Tesseract je hlavně pro zpracování stran tištěného textu v určitém jazyce a při trénování je potřeba dodat i slovníky, frekvenční analýzu nejčastějších slov atd. Toto všechno trénování velmi prodloužilo a nakonec jsme ho nedokončili, protože se ve stejné době naše vlastní detekce ukázala jako použitelné a dobré řešení.

Vlastní detekce a klasifikace písmen

Vlastní detekce je rozdělena do tří úkolů:
1. Detekovat všechny potenciální znaky ve značce.
2. Vybrat, které znaky nás zajímají, případně doplnit nenalezená písmena.
3. Rozpoznat čísla a písmena.

Pro detekci znaků jsme natrénovali klasifikátor SVM, který rozhoduje, zda vybraná oblast vykreslená do bitmapy je znak či nikoli.
Pro klasifikaci znaků jsme natrénovali KNN klasifikátor.

SVM klasifikátor JE/NENÍ znak

Klasifikátor jsme natrénovali na 901 znacích a 390 neznacích. Použili jsme 9 příznaků. Při jejich výběru jsme čerpali z práce Neumanna a Mattase [1]. Vybrané příznaky jsou uvedeny v tabulce 1. Oblast písmene je popsána vektorem bodů. Pro každý znak lze určit její plochu a, obvod P, plochu konvexní obálky a_c, počet děr H a obsah děr a_h. Dále je možné pro každou oblast znaku najít ohraničující obdélník B, u kterého známe souřadnice $[x, y]$ jeho levého horního rohu, jeho

výšku h a šířku w. Jednoduše je také možné spočítat horizontální počet průsečíků a písmenem v 1/6, 3/6 a 5/6 obrázku, ve kterém je znak vykreslený. Pro zjištění obvodu a počtu děr uvádí Pratt v knize [2] výpočet pomocí Eulerova čísla binárního obrazu.

Poměr šířky k výšce znaku	w/h
Počet děr	H
Kompaktnost	P^2/a
Poměr plochy děr k ploše oblasti	a_h/a
Poměr plochy konvexní obálky k ploše oblasti	a_c/a
Relativní výška	a/h^2
Počet horizontálních průsečíků v 1/6 písmene	$c_{1/6}$
Počet horizontálních průsečíků ve 3/6 písmene	$c_{3/6}$
Počet horizontálních průsečíků v 5/6 písmene	$c_{5/6}$

KNN klasifikátor na písmena

Tento klasifikátor jsme trénovali na cca 70 obrázcích každého písmene. Bohužel jsme nenasbírali dostatečné množství všech znaků, proto je klasifikátor natrénovat pouze na všechny číslice a písmena A a B. Při výběru příznaků jsme se inspirovali diplomovou prací Lukáše Neumanna [3]. Ten nejprve bitmapu s písmenem transformuje na velikost 35x35. My jsme zvolili velikost 16x28, tak v našem případě docházelo k nízké deformaci znaku. Poté je nad bitmapou měřena sada příznaků na základě 8směrového chain-code obvodových pixelů znaku. Pro každý směr je vytvořena vlastní bitmapa, ve které je zaznamenán daný směr tahu po obvodu znaku. Následně jsou tyto směrové bitmapy zmenšeny na čtvrtinovou velikost a převedeny na vektory, ty jsou dány za sebe a použity jako příznaky pro klasifikátor. Celkem tedy máme 224 příznaků pro jeden znak.

Příprava dopravní značky

Na rozdíl od běžného OCR nad reálnými obrázky je u SPZ výhoda, že písmena budou vždy černá, stejného fontu a na bílém pozadí. Proto si můžeme dovolit provést prahování obrázku s adaptivním prahem a spoléhat, že mezi černými spojitými oblastmi, které jsou nyní v obrázku, jsou i znaky SPZ. Po prahování nejprve odstraníme všechny oblasti, které jsou příliš malé, které nejsou s ostatními v jedné linii a ty, které mají od ostatních příliš rozdílnou výšku. Oblasti, které jsou oproti ostatním širší, zkusíme rozdělit na dvě a natrénovaným SVM klasifikátorem rozhodneme, zda jsme dostali dvě písmena nebo ne. Pokud ano, jsou přidána k seznamu oblastí.
Nyní předpokládáme, že všechny oblasti, které máme, jsou znaky SPZ. Zkontrolujeme, zda jich nemáme méně než 5, pokud ano, předpokládáme, že nalezená SPZ není SPZ. Pokud jich máme 5 nebo 6, zkusíme dohledat zbývající písmenka SPZ. Následně jsou všechny znaky v pořadí od nejlevějšího rozpoznány a vypsány na výstup.

Vyhledání chybějících písmen

Pokud jsme nalezli pozici šesti písmen a jedno nám chybí, pokusíme se dopočítat jeho pozici, z pozic jednotlivých nalezených písmen a předpokládaného formátu SPZ. Při tom pracujeme s velikostmi děr mezi jednotlivými symboly, které, pokud jsou písmena u sebe, mají ‚malou' velikost, a pokud jsou od sebe dále, ‚velkou'. Tyto pojmy blíže specifikujeme tak, že do malé mezery se nevejde další písmeno, do velké mezery se vejde jedno další písmeno, atp. Správně rozpoznaná značka má formát, 2 malé, 1 velká, 3 malé mezery. V případě, že jedno písmeno chybí,

tento vektor se změní a je z něj možno vyčíst, která pozice není obsažena a kde se písmeno má vyskytovat. Protože mají symboly jednotnou velikost, není poté problém zjistit, kde je chybějící písmeno.
Následuje tabulka formátu mezer mezi 6 detekovanými symboly, kde 0 je malá mezera, 1 je velká mezera a 2 je mezera velikosti na dvě písmena.

Chybějící symbol	Kód velikosti mezer
1. pozice	0 1 0 0 0
2. pozice	1 1 0 0 0
3. pozice	0 2 0 0 0
4. pozice	0 0 2 0 0
5. pozice	0 0 1 1 0
6. pozice	0 0 1 0 1
7. pozice	0 0 1 0 0

Podobnou myšlenku „kódů mezer“ máme zpracovánu i pro dva chybějící symboly, kde nám počet možných kombinací naroste na 21 a velikost mezery na číslo 3.
Kolize je zde pouze jedna, kdy jsou dva různé případy se stejným kódem. Tento případ je nutno ošetřit dalšími kontrolami, např. jestli je v těchto oblastech písmeno nebo není atp. Následuje tabulka kódů:

Chybějící symbol	Kód velikosti mezer
1. a 2. pozice	1 0 0 0
1. a 3. pozice	2 0 0 0
1. a 4. pozice	!!! 0 2 0 0 !!!
1. a 5. pozice	0 1 1 0
1. a 6. pozice	0 1 0 1
1. a 7. pozice	0 1 0 0
2. a 3. pozice	3 0 0 0
2. a 4. pozice	1 2 0 0
2. a 5. pozice	1 1 1 0
2. a 6. pozice	1 1 0 1
2. a 7. pozice	1 1 0 0
3. a 4. pozice	0 3 0 0
3. a 5. pozice	0 2 1 0
3. a 6. pozice	0 2 0 1
3. a 7. pozice	!!! 0 2 0 0 !!!
4. a 5. pozice	0 0 3 0
4. a 6. pozice	0 0 2 1
4. a 7. pozice	0 0 2 0
5. a 6. pozice	0 0 1 2
5. a 7. pozice	0 0 1 1
6. a 7. pozice	0 0 1 0

Rozpoznání písmen

Z obrázku znaku vypočteme výše zmíněné příznaky, vytvoříme matici s příznaky pro zadané písmeno a tu předáme načtenému a natrénovanému KNN klasifikátoru. Pokud nám pro druhý znak určil, že se jedná o 8, provedeme korekci na znak B, na této pozici se totiž může vyskytovat jedině písmeno. Naopak pokud je v některých z posledních 4 znacích detekováno B, je potom vyměněno za znak 8, protože auta s B na konci se ještě nevyskytují.

Testování

Testovali jsme celkem na 47 obrázcích.

Rozpoznání	Počet značek	% z celku
V pořádku	51	45,5 %
1 chyba	21	18,7 %
2 chyby	9	8 %
Špatně klasifikovaná	18	16 %
Celá chybí	7	6,2 %
Navíc detekovaná	6	5,5 %

[1] Neumann, L.; Matas, J.: Real-Time Scene Text Localiation and Recognition. Conference on Computer Vision and Pattern Recognition, 2012.

[2] Pratt, W. K.: Digital image processing. Willey-interscience publication, John Wiley & Sons, INC., 2001, iSBN 0-471-22132-5.

[3] NEUMANN, Lukáš. *Vyhledání a rozpoznání textu v obrazech reálných scén*. Praha, 2010. Diplomová práce. České vysoké učení technické v Praze. Vedoucí práce Doc. Dr. Ing. Jiří Matas.