Kooperáció és gépi tanulás labor
Kernel módszerek - osztályozás
Mérési segédlet

Méréstechnika és Információs Rendszerek Tanszék
Orbán Gergely
orbanger@mit.bme.hu

2009. november 26.

Tartalomjegyzék

1. fejezet

A probléma

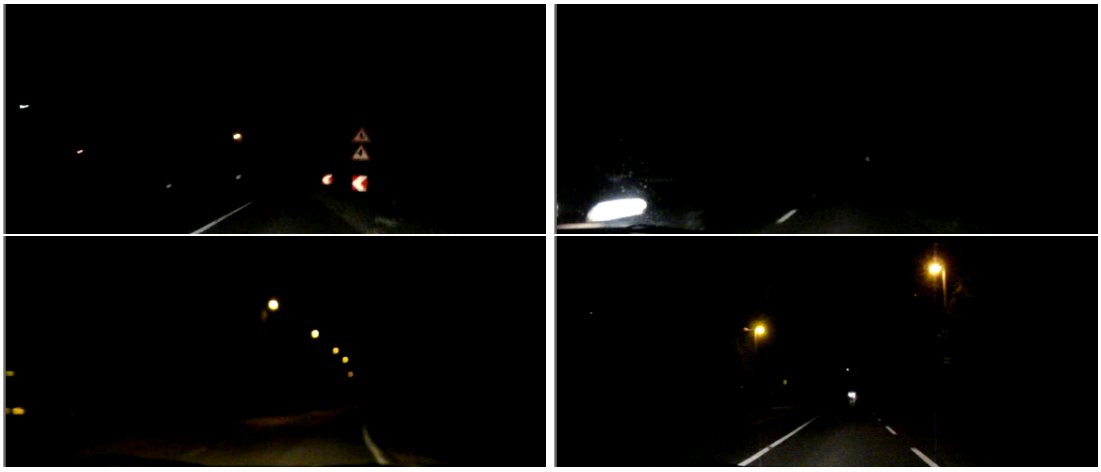
A labor során egy olyan rendszert igyekszünk kialakítani, mely képi információ alapján felismeri a szembejövő autót éjszakai környezetben. Egy ilyen eszközzel például lehetővé válhat a távolsági fényszóró automatikus vezérlése, mely tehermentesíti a vezetőt, miközben optimalizálja a látási viszonyokat és segít elkerülni a szembejövő autósok elvakítását.

Az ötlet nem új, egyszerű fényérzékelős megoldások már 1953-ban is léteztek[1]. Ezek azonban sem a közvilágítást, sem a táblákról érkező reflexiókat nem voltak képesek megkülönböztetni a szembejövő autóktól. 2005-ben jelent meg az első kamera alapú rendszer a Jeep Grand Cherokeeben, ahol egy képfeldolgozó számítógép már hatékonyan ellátta a feladatot.

A modern rendszerek alapja az autó orrára felszerelt digitális kamera és egy számítógép, mely ennek képét online dolgozza fel. A labor során egy ilyen számítógépre írandó algoritmust fogunk kialakítani gépi tanulási módszerek segítségével. A feladatot némileg leegyszerűsítve készítünk egy osztályozót, mely minden egyes beérkező kameraképről eldönti, hogy van-e rajta szembejövő autó (pozitív), vagy sem (negatív). Négy tipikus felvételt láthatunk az 1.1 ábrán.

Az idő rövidsége és a technikai korlátok miatt néhány egyszerűsítést teszünk. A legfontosabb, hogy nem foglalkozunk az előttünk haladó autóval. A velünk egy irányba haladó autó detektálása sem nehezebb feladat, de a tanításhoz több mintaképre lenne szükség, ami lelassítaná a folyamatot. Egy valós rendszerben amúgy is megfontolandó a két problémát (szembejövő, illetve azonos irányba haladó autó) különválasztani. A lakott terület detektálásával sem foglalkozunk, a képek nagyrészt ezen kívül készültek. Az „autópályán szembejövő teherautó” és hasonló extrém helyzetekre minták hiányában nem tudunk felkészülni. A tanításra kiválasztott képek tompított fény használata mellett készültek, mely nem teljesen tükrözi a majdani felhasználás körülményeit. Az algoritmusokat ez nem nagyon érinti, de az „éles” rendszer tanításához mindenképp reflektor használatával készült felvételeket kell beszerezni.

A most kidolgozandó rendszernél a bemenetet egy autóba szerelt 640x480 pixel felbontású kamera adja, de a képnek csak egy kisebb, körülbelül 580x250 pixel méretű tartománya hordoz hasznos információt. A képek pixelenként 3 színcsatornát tárolnak (RGB) színenként 8 bites felbontással.



1.1. ábra. Balra két negatív, jobbra pedig két pozitív felvétel látható.

2. fejezet

Felkészülési feladatok

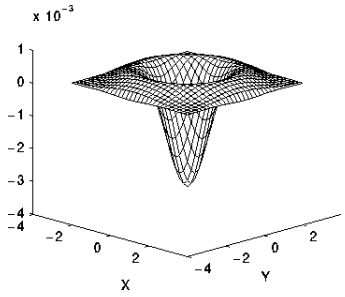
Miután felfrissítette ismereteit a tanuló algoritmusokról ([2] és [3]), tanulmányozza át a mellékelt felvételeket. Ezeket kell majd minél nagyobb arányban helyesen osztályoznia. A *true* könyvtárban olyan képeket talál, ahol van szembejövő autó, ezek lesznek továbbiakban a pozitív példák. A *false* könyvtárban a negatív példákat találja. Próbálja meg gondolatban megoldani a feladatot. A következő részben talál ötleteket a megoldáshoz, melyeket a mérésen részben önállóan meg kell valósítania.

A legegyszerűbb megoldás, ha a kép $580 \times 250 \times 3$ képpontját (RGB csatorna) mint 435000 dimenziós vektort átadja az osztályozónak. Ha visszaemlékszik, az alapképzés tanuló ágensek mérésére, ott ezt a módszert alkalmazták. Kétségtelen előny, hogy nem kell sokat gondolkodni, hiszen minden általánosítást így a tanuló rendszerre bíznak, nem szükséges a probléma mélyebb ismerete. Azonban ez a megoldás itt és ebben a formában nem működik, ugyanis az SVM és az MLP tanuló algoritmusa is nagyon hamar felemészti a PC-nk összes operatív memóriáját, szuperszámítógépeket pedig nem szeretnénk alkalmazni egy ekkora kaliberű problémára. Intuitív lépés a kép lekicsinyítése. Némi próbálkozás után adódhat például a $36 \times 15 \times 3$ pixeles felbontás, ahol még jól látszanak az autók fényei, de a pixelek száma megfelelően alacsony. Bemelegítő feladatként ezt a megoldást fogják megvizsgálni a mérésen. Nem titok, hogy az eredmény egész jó lesz, de azért több probléma is van vele. Egyrészt nem lesz túl gyors az algoritmus, másrészt az általánosítóképességben sem lehetünk biztosak, mivel így is túl sok adatból dolgozunk.

Hogyan oldjuk meg tehát, hogy még kevesebb adattal jellemezzük a képeinket? Számoljunk rá jellemzőket! Meg kell tehát alkotnunk olyan mérőszámokat, melyek jól jellemzik a képet az eredetinél lényegesen kisebb méretben. Felmerülhet az ötlet, hogy tömörítsük a képet, de itt kicsit másról van szó. Nem kell tudnunk a jellemzőkből az egész képet visszaállítani, az is elég, ha csak eldönthető belőlük, hogy jön-e szembe valaki, vagy sem. Emlékeink is hasonló módon működnek. Ha elolvasunk egy tankönyvet, (jó esetben) nem az egész szöveg marad meg a fejünkben, hanem a lényegi információk, amik érdekelnek minket.

A jellemzők megalkotásához célszerű végiggondolni, hogy saját magunk mi alapján állapítanánk meg egy eset pozitív vagy negatív voltát. Ha megyünk a sötétségben és nincsenek zavaró objektumok a közelben, akkor nagyon egyszerű dolgunk van. Egyszerűen csak fényforrást kell keresni, ami jellemzően megnöveli az intenzitást valahol a képen. Ez például jól megfogható, ha kiszámítjuk a legfényesebb pont intenzitását. Ha egy fekete képen megnézzük a pixelértékek eloszlását, akkor egy 0 környékén csúcsosodó, kis szórású eloszlást kapunk. Változik ez, ha megjelenik egy autó fénye a képen?

Az élet azonban gyakran bonyolultabb, előfordulnak útszéli lámpák, illetve táblák, amik tükröződnek. Miben különböznek ezek a szembejövő autó lámpájának fényétől? Első ránézésre mintha az autó lámpája fényesebb lenne. Azonban ha jobban megvizsgálják a képeket, kiderül, hogy a kamera ezt máshogy gondolja, ugyanis az útszéli lámpák és tükröződések közepén is találunk maximális fényerejű (kiégett) pixeleket. Miért látjuk mégis fényesebbnek? Az autó lámpája környezetében



2.1. ábra. Egy LoG függvény.

gyakran több a kiégett pixel. A jelenlegi megoldásban azonban nem vizsgálunk részterületeket, így nem tudjuk megfogni a „környezetében" fogalmat, illetve némi trükkel mégis. Ha sikerülne az egyes pixelekhez egy olyan értéket rendelni, ami jellemzi a környezetének fényességét, akkor elég lenne megnézni ennek a képnek a maximumát. Másik oldalról megfogva a problémát, ha néhány fényforrás közelében több a kiégett pixel, akkor valószínűleg az egész képen is több lesz ilyen.

Másik dolog, ami szembe tűnhet, hogy az autók fénye általában magasabb színhőmérsékletű azaz kékesebb, mint az útszéli lámpák sárgás fénye. Ebből adódhat, hogy érdemes megvizsgálni az RGB kép kék és sárga csatornáját, illetve a HSV kép hue csatornáját (színrendszerekről bővebben ld. Számítógépes Grafika c. tárgy [4]). Mi történhet ezekkel, ha erős kékes fényforrás jelenik meg az eddigi sárgásak mellé? A kék csatorna gyakran telített, ezen hogyan segítene? Közelebbről vizsgálva a fényforrásokat az látszik, hogy szinte mindegyik közepén maximális fényességű fehér pixelek vannak (RGB: (1,1,1), HSV: (0,0,1)), de kicsit távolabb a középponttól már az autók esetében kékesebb, a lámpák esetében sárgásabb a pixel amellett, hogy a HSV képen az intenzitás maximális marad (azaz RGB képen a B maximális, a másik kettő csökken, HSV képen V maximális, és S nő). Érdemes jellemzőt készíteni az ilyen típusú pixelek megfogására.

A lámpák alakjával eddig még nem foglalkoztunk, pedig segíthet, ha egy más méretű vagy formájú alakzattól szeretnénk megkülönböztetni őket. A méretről sajnos nem tudunk sokat mondani, mert elég széles tartományban változhat, de az alak jellemzően körszerű. Jó ötlet lehet egy olyan szűrést végezni a képen, mely az ilyen körszerű alakzatokat emeli ki, mást viszont elnyom, esetleg pont fordítva. A „Laplacian of Gaussian" szűrő ilyen hatású. Egyik oldalról egy kétdimenziós Gauss függvény második deriváltjával való szűrésről (konvolúcióról) van szó, de szemléletesebb, ha a függvény alakját vizsgáljuk (2.1. ábra). Egy olyan radiális kétdimenziós függvényről van szó, mely a középpont körül nagyon nagy abszolút értékű és negatív, a peremén pozitív, az integrálja a teljes síkra pedig nulla. Érezhető, hogy a függvénnyel való konvolúció akkor adja a legnagyobb kimenetet, ha egy világos környezetben sötét foltra talál, melynek határvonala egybeesik a függvény zérushelyeivel. Igaz ez fordítva is, a legkisebb kimenetet pont a keresett sötét környezetből kiemelkedő világos foltokra adja. A szűrt képnek érdemes tehát kiszámolni a minimumát, illetve szórását, hogy jellemezzük egy ilyen minimumhely meglétét. A minimumhely pozíciója is annyiban érdekes, hogy ha nagyon eltér egy szembejövő autó szokásos helyétől, akkor valószínűleg nem egy autó lámpája adta a minimumhelyet a képen, ezért nem is érdemes figyelembe venni azt. Ilyen explicit szabályokat azonban nem akarunk megfogalmazni, inkább a pozíciót és a minimum értéket is átadjuk az osztályozónak, az pedig majd kezd vele valamit, ha tud.

Az eddigieket elolvasva biztos számos ötlete van már a különböző képjellemzőkre. Írja le ezeket ötletek szintjén. Az utolsó bekezdésben ismertetett LoG szűrővel kapcsolatos méréseket már implementáltuk, azokat megtalálja a kiadott kódban, azonban a többi jellemzővel válhat csak igazán hatékonnyá a megoldása. Vizsgálja tovább a képeket, és alkosson meg egy jellemzőt, melyre nem talál utalást a fenti szövegben. Indokolja meg, hogy az miért lehet hasznos az osztályozás szempontjából. Vigyázzon, hogy ne legyen redundáns a már meglevő jellemzőkkel. Az algoritmust a mérés során kell majd implementálnia és futtatnia, tehát ügyeljen a megvalósíthatóságra és a nem túl hosszú futási időre.

3. fejezet

A kiadott kód leírása

3.1. createData.m

Ez a script felelős a tanításhoz szükséges adatok előállításáért. Amennyiben módosulnak a jellemzők vagy az adathalmaz, le kell futtatni. Egyenként megnyitja a "/true" illetve "/false" könyvtárakban levő képeket, kiszámoltatja rá a jellemzőket, majd elmenti az eredményeket az "inputs.mat" fájlba. A futási idő csökkentése végett minden képhez a HSV színtérben ábrázolt változatát is kimentettük PNG formátumba. Ha ezek nem állnak rendelkezésre, a konverzió az rgb2hsv függvénnyel lehetséges. Az "inputs.mat" így tartalmazni fog egy x mátrixot, melyben minden egyes sor egy képnek felel meg, az oszlopok pedig az egyes jellemzők. Az y oszlopvektor az x megfelelő soraihoz tartozó osztályozások, továbbá az *ids* az eredeti felvétel azonosítóit tartalmazza. A script viselkedése módosítható a $derivedFeatureMode$ és a $excludedFeatures$ paraméterek állításával. Előbbi dönti el, hogy a jellemző számítás milyen módon történjen, bővebben ld. "calcFeatures.m". Utóbbi listában megadható, hogy mely jellemzők ne kerüljenek kiszámításra egy futás során. Ennek akkor lehet értelme, ha egy úgy jellemző hozzáadásakor, vagy meglevő módosításakor nem akarják az összes jellemzőt újraszámítani, hogy időt spóroljanak meg. Működéséhez szükséges, hogy az inputs.mat tartalmazza a korábbi számítás eredményeit, és a jellemző indexek ne változzanak (Pl. ha nem utolsóként szúrnak be egy új jellemzőt, akkor az utánalevőket már újra kell számítani. Ha állítják a $derivefFeatureMode$-ot, akkor hagyják üresen a listát.).

3.2. calcFeatures.m

Ez a függvény felelős a jellemzők kiszámításáért. A "createData.m" hívja. Többféle funkciót megvalósít a $derivedFeatureMode$ paraméter függvényében. False érték mellett egy átméretezett kép pixeleit számítja ki. True érték mellett számítja a származtatott jellemzőket. Ezek közül az előre implementáltak az intenzitás kép minimuma és a Laplacian of Gaussian (LoG) jellemzők. Az LoG számításhoz különböző sugarú LoG függvényekkel szűri a képet:

```
kernel = fspecial('log',4*radius(rind)+1,radius(rind));
filteredImage = conv2(imageDouble,kernel,'same');
```

Ezek után a szűrt képre számolja ki a pixelértékek átlagát, szórását, minimumát és a minimum koordinátáit.

```
devs(rind)= std(filteredImage(:));
[colmin, rowind] = min(filteredImage);
[mins(rind), colind] = min(colmin);
mininds(2*rind-1:2*rind) = [rowind(colind), colind];
```

Az előre implementált jellemzők számítása után a program meghívja a „studentFeatures” függvényt, melybe a saját jellemzőiket implementálhatják.

A függvény kap egy $excludedFeaturesIndexes$ paramétert, melynek alapján bizonyos jellemzők számítását elhagyja, és azok helyén 0-t ad vissza.

3.3. studentFeatures.m

Itt célszerű implementálnia a saját jellemzőit. Bemenetként kap egy RGB és egy HSV képet, kimenetként ki kell adnia a jellemzők tömbjét. Az osztályozás során ezek a 6. dimenziótól kezdve fognak megjelenni (tehát ha itt egy jellemző a tömb második eleme, akkor arra a classify.m-ben hetedikként lehet hivatkozni. Ötleteket talál a kódoláshoz a „calcFeatures.m”-ben. Ha szeretné gyorsítani a kódját, használja a $excludedFeaturesIndexes$ tömböt a „calcFeatures.m”-ben látott módon. Az 5-nél nagyobb indexek vonatkoznak a saját jellemzőire.

3.4. classify.m

Ez a script felelős az osztályozó tanításáért és teljesítményének méréséért. Az "inputs.mat" fájlból nyeri a bemenő adatokat. Az MLP-s osztályozáshoz a Matlab beépített Neural Networks toolboxát, az SVM-hez az SVM KM ingyenes toolboxot használja[5]. A script megfelelő paraméterezéssel és commentezéssel képes a bemenő adatvektorokból komponenseket kiválasztani, ezeket normalizálni, majd MLP-vel vagy SVM-mel osztályozni. Az osztályozók paramétereinek hangolásához legfeljebb háromdimenziós numerikus paramétertér végigpróbálható logaritmikus skálán és az eredmény kimenthető.

A beállítandó paraméterek:

- mode: SVM vagy MLP osztályozza-e a mintapontokat.
- numreptest: Azonos paraméterekkel történő újrafuttatások száma. Az eredmények átlagolódnak. Pontos mérésekhez érdemes legalább 20-ra választani.
- selectedFeatures: A bemeneti jellemzőkből kiválaszthatjuk a ténylegesen felhasználtakat sorszám alapján. Az első 5 helyen találja az előre implementált jellemzőket, a részletekhez tanulmányozza a kódot.
- kernel: Az SVM által használt kernelfüggvény.
- MSE: MLP leállási feltétele.
- Epochs: maximális iterációk száma az MLP tanításánál.
- hypparamn: Az osztályozók hiperparaméterei. Értékük a logspace függvény megfelelő paraméterezésével állítható be. MLP-nél hypparam1 az első rejtett réteg neuronjainak száma. SVM-nél hypparam1 a kernelfüggvény paramétere, hypparam2 az általánosítási paraméter (C vagy γ).

Irodalomjegyzék

[1] Wikipedia.org, "http://en.wikipedia.org/wiki/Automatic_headlight_dimmer" *Wikimedia Foundation, Inc.* 2009

[2] Altrichter M., Horváth G., Pataki B., Strausz Gy., Takács G., Valyon J., *Neurális Hálózatok*, Panem, 2006

[3] BME-MIT, "Intelligens Rendszerek I. laboratórium, Tanuló ágensek tervezése c. mérés segédlete"
http://portal.mit.bme.hu/oktatas/targyak/vimia360/jegyzet/lab05/lab05_melleklet.pdf 2007

[4] dr. Szirmay-Kalos László, "Számítógépes Grafika c. tárgy" *http://www.iit.bme.hu/ szirmay/szg.htm* 2009

[5] S. Canu and Y. Grandvalet and V. Guigue and A. Rakotomamonjy, "SVM and Kernel Methods Matlab Toolbox" *Perception Syste'mes et Information, INSA de Rouen, Rouen, France* 2005