Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Kombinált képkeresés offline osztályozás segítségével

TDK dolgozat - 2013

Papp Dávid

Konzulens: Dr. Szűcs Gábor

Absztrakt

Manapság képi tartalmak között keresni nem újszerű dolog. Egy képkereső rendszer feladata, hogy a felhasználó által megadott keresési kulcs szerinti releváns képeket adjon vissza. A relevancia eldöntése viszont nem egyszerű feladat, bizonyos kereső kifejezések esetén pedig kifejezetten nehéz. Éppen ezért a mai napig fejlesztés alatt állnak még a legelterjedtebb képkeresők is (pl. Google, Bing, Flickr).

Azt az esetet, mikor nem állnak rendelkezésre metaadatok a kereséshez, és csupán a képek tartalmi információit használjuk fel, szemantikus képkeresésnek nevezzük. Amennyiben ezt ismeretlen képeken végezzük, szükség van egy tanulóállományra, amely alapján gépi tanulásos módszert alkalmazva elemezhetőek a keresési térbe tartozó, úgynevezett teszt képek. Az elemzésből származó adatok ezután már rendelkezésre állnak a kereséshez.

Dolgozatomban bemutatom azt az általam készített szemantikus képkereső rendszert, amelynek feladata, hogy olyan keresési kulcs alapján keressen, amelyet egyszerre több objektum vagy fogalom kombinálásával kapunk. A keresést megelőző szemantikai elemzéshez a state-of-the-art képfeldolgozási módszereket használom, mint például a Fisher-vektor, vagy a C-SVC osztályozó. Az offline módon rendelkezésre álló szemantikai információkat felhasználva több olyan módszert is implementáltam, amely megvalósítja a kombinált képkeresést.

A dolgozatban a kettő illetve három objektumból összeállított kereső kifejezésekre koncentrálok. Ezek eredményeit kiértékelem és összehasonlítom azokat két ismert internetes képkereső (Bing és a Flickr) eredményeivel. Tanulóállományként ehhez a Pascal VOC, képi klasszifikációs verseny weboldaláról letöltött tanító képeket használom, melyek eredetileg szintén a Flickr-ről lettek összegyűjtve.

1. BEVEZETŐ

A mai világban mindenki keres, előképzettségek nélkül. Ez azt jelenti, hogy az átlag felhasználó szinte biztos, hogy nem képes lekérdezését olyan pontosan, és abban a formában megadni, amelyet egy keresőrendszer megkövetel. A különböző médiumok közti keresés mindennapi tevékenységgé vált, így a képkeresés is hétköznapi.

1.1. SZEMANTIKUS KÉPKERESÉS

Képek közti keresést sokféle adat alapján végezhetünk. Attól függően, hogy egy adott kép milyen környezetben helyezkedik el, több-kevesebb, általában szöveges információ köthető hozzá, melyeket a kép meta-adatainak nevezünk. Ezek az adatok nagyon népszerűek a képkeresés futtatása során, viszont sokszor pontatlan eredményt adnak. Vegyük például a kép nevét, mint meta-adatot. Fényképezővel vagy kamerával készített fotóknak a neve általában az elkészítés dátumára utal, nem pedig a tartalomra. Könnyen megtéveszthet egy olyan képkereső algoritmust, amely a neveket is felhasználja. Egy ettől eltérő módszer az, mikor a képek vizuális tartalmát használjuk a kereséshez. A két módszer ötvözhető, hogy különböző kontextusból nyerjük az információkat. Amennyiben kizárólag a tartalmi információkra támaszkodunk, szemantikus képkeresésről beszélünk.

Ennek előnye, hogy egy képet semmi nem jellemez jobban, mint az, ami rajta látható. Emberi erőforrást használva így szinte tévesztés nélkül lehet végrehajtani a keresést, viszont ez nagyon időigényes és drága feladat. Célszerű lenne ezt automatizált módon megoldani.

1.1. ábra: Versenyautó

Erre az egyik legkézenfekvőbb megoldás, ha minden képet egy szöveges jellemzéssel látunk el a vizuális tartalma alapján. Amennyiben ez a jellemzés a keresési térben található összes képre adott, akkor a keresést tulajdonképpen ezek között a szöveges információk között tudjuk futtatni. Egy szöveges jellemzés több mindenre is vonatkozhat. Például tekintsük a következőt, az 1.1. ábra képére: *„Egy versenyautó látható a kép közepén, szürke színű aszfalton”*. A leírás megadja a képen szereplő egyetlen objektum helyzetét, azt, hogy mi az az objektum, illetve a hátteret és annak színét is.

Az objektum megnevezésén kívül tekintsünk el a többi annotációtól. Olyan szemantikus képkeresési feladatokban, ahol a keresési kulcs kizárólag objektumokból áll, a képenkénti annotációk leszűkíthetőek a képen jelen lévő objektumok halmazára. Egy képen szereplő objektum megnevezését leíró információt osztálycímkének, vagy kategóriának nevezünk. Abban az esetben, ha ezek minden, a keresési térbe tartozó képre adottak, akkor a keresés futtatható a kategóriák között. A legegyszerűbb keresőt például meghatározhatjuk úgy, hogy azokat a képeket adja vissza a találati listában, amelyekhez tartozó osztálycímkék között szerepel a keresési kulcsban megadott összes objektum. Ahhoz, hogy ez lehetővé váljon, szükségszerű osztálycímkékkel ellátni a képeket, méghozzá automatizált módon, mivel kézzel elkészíteni ezeket a jellemzéseket

nagyon időigényes és bizonyos esetekben szakértői munkát igényel. Ezt a folyamatot osztályozásnak, vagy más néven klasszifikációnak nevezzük.

1.2. KÉPOSZTÁLYOZÁS PROBLÉMÁJA

A számítógép kontextusában egy kép pixeleket, színeket, alakokat, textúrákat tartalmaz. Ezekből meghatározni azt, ami egy ember számára egyértelmű egy képre ránézve, közel sem triviális feladat. Nem ritka, hogy több száz kategória közül kell eldöntenie egy osztályozó algoritmusnak, hogy melyik az az egy vagy néhány, amelyikbe a tesztelt kép beleillik. Alapvető probléma az is, hogy hasonló szemantikus információval bíró képek is lehetnek nagyon különbözőek. Erre példaként tekintsük az 1.2. ábra képeit.

1.2. ábra: Madaras képek

A képosztályozási feladatok eredményei általában gyengék, még viszonylag bonyolult, nagy futási időt igénylő algoritmusok használata esetén is, mivel a fentebb említett okokból adódóan nagyon összetettek és rengeteg a hibalehetőség. Körülhatárolt területeken viszont jó eredménnyel alkalmazhatóak. Ilyen például az orvosi diagnosztika, vagy a karakterfelismerés.

A szemantikus képkereséshez szükséges képosztályozási problémát gépi tanulással oldottam meg. Ehhez szükség van egy úgynevezett tanulóállományra, amelyben minden képhez meg van adva a megfelelő osztálycímkék halmaza. Vagyis *c* darab kategória esetén, minden képhez tartozik egy *c* dimenziós ***x*** vektor, amelyre:

$$x_i \in \{0,1\}, \quad ahol\, i = 1 \dots c \tag{1.1}$$

Az ***x*** *i*-edik eleme akkor 1, ha az *i*-edik kategória szerepel az ***x***-hez tartozó képen. Általában a tanulóállomány két részből áll, egy tanító és egy megerősítő (validáló) halmazból. A tanítási folyamat alatt a megerősítő halmazt használom teszthalmazként, és visszacsatolásos módszerekkel javítom a tanítás folyamatát az elérhető legjobbig.

A tanító halmazon betanítom az osztályozót, majd minden megerősítő kép esetén becsülhető egy *c* dimenziós ***x'*** vektort. Az így számított vektor elemei lehetnek diszkrét értékek, vagy lehetnek konfidencia érékek, azaz, hogy milyen valószínűséggel tartozik bele az *i*-edik kategóriába. A konfidencia értékeket egy küszöb segítségével diszkrét értékekké alakítom. Ideális osztályozó esetén a becsült ***x'*** vektor megegyezik a tanulóállományban megadásra került ***x*** vektorral.

1.3. A DOLGOZAT FELÉPÍTÉSE

A dolgozat célja, hogy bemutassam az általam készített szemantikus képkereső rendszert egy konkrét eseten keresztül. A kísérletben kettő illetve három objektumból kombinálom össze a kereső kifejezést, és az erre kapott találati listát értékelem ki. Az eredményeimet összevetem a Bing és a Flickr eredményeivel. A felhasznált képhalmazról semmilyen tartalmi információ nem áll rendelkezésemre, ezért a kereső algoritmus mellé elkészítettem egy osztályozó algoritmust, melynek segítségével ellátom az ismeretlen képeket szemantikai információkkal.

A második fejezetben lépésről lépésre bemutatom, hogyan történik egy kép tartalmi információinak reprezentálása, illetve az alapján osztályba sorolása. Közben részletesen kifejtem a lépésekhez tartozó matematikai hátteret. A harmadik fejezetben rátérek a megvalósított, szemantikus képkereső módszerekre. Ez után a kísérlettel foglalkozok, a negyedik részben felvázolom a keresés körülményeit, utána pedig az utolsó fejezetben az eredményeket tárgyalom.

2. KÉPOSZTÁLYOZÁS

A fejezet célja, hogy ismertessem a szemantikus képkeresést megelőző tartalmi elemzést végző képi klasszifikációs algoritmus fontosabb lépéseit. Magára az implementációra nem térek ki, helyette a feladat mögött húzódó elméletet ismertetem.

2.1. EGYCÍMKÉS VAGY TÖBBCÍMKÉS OSZTÁLYOZÁS?

Egy képosztályozási feladat szempontjából nem mindegy, hogy milyen a képek és a kategóriák közti viszony. Abban az esetben, ha egy képet határozottan egyetlen osztályhoz rendelünk, egycímkés osztályozásról beszélünk. Ez azt jelenti, hogy a fentebb említett ***x'*** vektornak csak egyetlen eleme lehet 0-tól különböző. Amennyiben egy képet egyszerre több osztályhoz is rendelhetünk, többcímkés osztályozásról beszélünk. Formálisan, az ***x'*** vektornak tetszőleges számú 0-tól különböző eleme lehet.

Egycímkés osztályozás esetén a fenti megállapítás nem teljesen pontos, mivel konfidencia, azaz valószínűségi értékek meghatározása esetén egyszerre több kategóriához is rendelhetünk 0-tól különböző értéket. Ilyenkor viszont a legnagyobb valószínűségi értékhez tartozó kategóriát választjuk, az ahhoz tartozó értéket 1-re és a többi kategóriához tartozó értékeket pedig 0-ra állítjuk. Ezzel a kiegészítéssel megállja a helyét a fenti definíció is.

Többcímkés esetben ilyen probléma nincs. Az általános eljárás az, hogy egy küszöb értéket (*k*) határozunk meg, és minden, a küszöbnél nagyobb valószínűségi értéket 1-re állítunk, az összes többit pedig 0-ra:

$$x_i' = \begin{cases} 1, & ha\ p_i' \geq k \\ 0, & ha\ p_i' < k \end{cases} \qquad (2.1)$$

Tehát a végeredményül kapott ***x'*** vektornak azok az elemei lesznek 1 értékűek, amelyekhez tartozó kategóriákba jósoljuk az adott tesztképet. Az egycímkés és többcímkés osztályozás közötti különbség megértésének érdekében tekintsük a következő példát, amelyet a 2.1. ábra szemléltet.

2.1. ábra: Bal oldalon az egycímkés, jobb oldalon a többcímkés osztályozás

A bal oldali képén az egycímkés esetet szemléltetjük. A tesztkép mindkét kategóriából tartalmaz objektumokat, vagyis a hozzá tartozó kételemű ***x*** vektor mindkét eleme 1 értékű. A becsült ***x'*** vektor viszont, ahogy azt a képen megjelöltük, az *autó* kategóriát jelöli ki a tesztkép egyetlen kategóriájának. Ez egyértelmű hátránya az egycímkés esetnek, ugyanis ilyen, és ehhez hasonló többobjektumos tesztképekre biztos, hogy nem képes az ideális ***x'*** vektor előállítására. Egycímkés osztályozás hatékony lehet, ha olyan területen alkalmazzuk, ahol biztosan csak egy kategóriába tartozhat minden egyes kép. Például, ha az abc betűit kell felismerni, és minden kép egyetlen betűt tartalmaz.

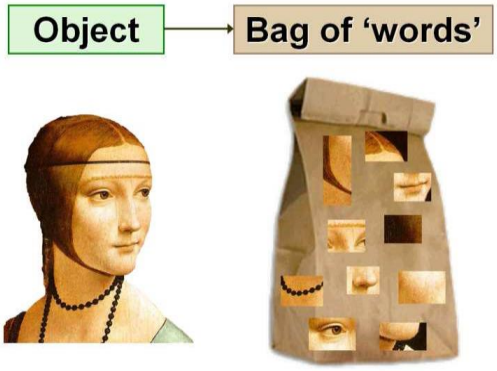
A fenti ábra jobb oldali képe a többcímkés osztályozás eredményét szemlélteti, ugyanerre a tesztképre. Amennyiben sikerül mindkét kategóriához a küszöb feletti konfidencia értéket rendelnie az osztályozónak, akkor a becsült ***x'*** vektor megegyezik az ideálissal. Ez persze nem mindig érhető el, hiszen általában több a lehetséges kategóriák száma, és nem biztos, hogy a megfelelőket sikerül a képhez jósolni.

Az osztályozó algoritmusban többcímkés osztályozást valósítottam meg, mivel természetes fényképekkel dolgozom, ahol előfordul több objektum egy képen.

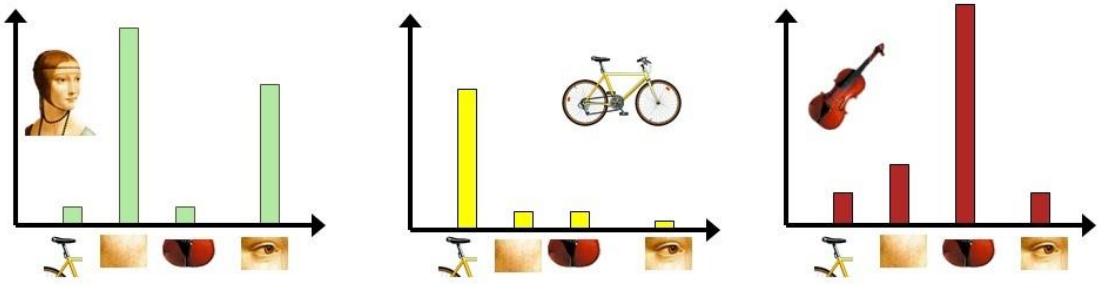
2.2. ALACSONY SZINTŰ LEÍRÓK SZÁMÍTÁSA

Képi tartalmak osztályozásához szükség van egy módszerre, amivel reprezentálhatók a képek szemantikus információi. Az általam használt módszer a szózsák modell képeken alkalmazott változata, [1], [2].

Az elképzelés az, hogy egy képet a rajta szereplő vizuális elemek, más néven vizuális kódszavak összességével reprezentálunk (ezt szemlélteti a 2.2. ábra), és ezek térbeli elhelyezkedésétől eltekintünk. Tehát egy képet csupán a vizuális kódszavak eloszlásával jellemzünk, és ebből próbálunk meg következtetni a szemantikájára, ahogy azt a 2.3. ábra mutatja.



2.2. ábra: A szózsák modell (forrás: [1])



2.3. ábra: Vizuális kódszavak eloszlása (forrás: [2])

A vizuális kódszavak meghatározásához úgynevezett alacsony szintű leírókra van szükségünk. Egy ilyen leíró a kép egy adott pontjának környezetében előforduló lokális tulajdonságokat foglalja magában. Ezek lehetnek színek, textúrák, stb.

Mielőtt rátérek az általam használt alacsony szintű leíróra, fontos megemlíteni, hogy azon pontok, amelyek környezetében ilyen leírókat számolok, többféleképpen is meghatározhatók. A következő pontban összefoglalom azt a két módszert, amelyet az algoritmusban használok.

2.2.1. ELEMZÉSRE KIVÁLASZTOTT PONTOK MEGHATÁROZÁSA

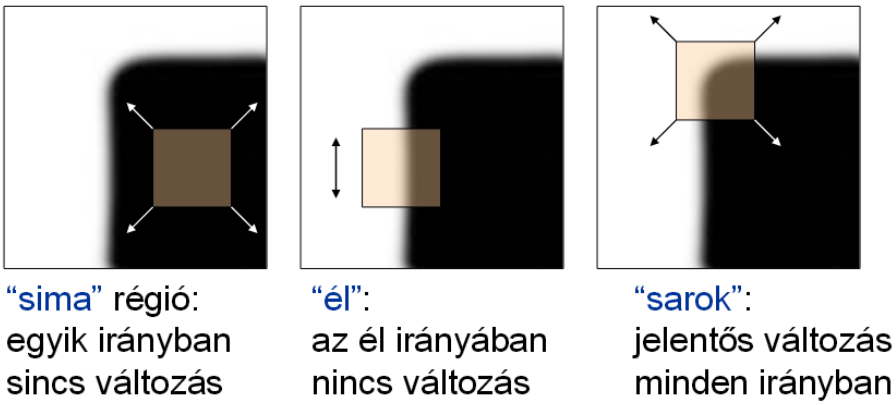
2.2.1.1 RÁCSVONALAK ILLESZTÉSE

A legegyszerűbb módszer az, ha egy rácsot illesztünk a képre, és minden rácspontban kijelölünk egy pontot, amelyet majd a megfelelő leíró vektorral reprezentálunk. Egyszerűsége ellenére ez a módszer rendkívül jól működik. A rendszerben ez az egyik módszer, amit alkalmazok. A rácsvonalak sűrűségének variálása jelenti az eltérő alkalmazási lehetőségeit.

2.2.1.2 HARRIS-LAPLACE DETEKTOR

A módszer teljes matematikai modelljét nem kívánom bemutatni, viszont a fontosabb elemeket kifejtem és értelmezem.

A módszer alapját a Harris detektor, más néven kombinált sarok és él detektálás adja, melyet 1988-ban fejlesztett ki C. Harris és M. Stephens, [3]. Egy képen ott vannak élek, ahol a kép pontjainak intenzitásértékeiben nagy változás következik be, két él találkozásánál (sarokpontnál) pedig az intenzitásváltozás, azaz a derivált két irányban is nagy abszolút értékű lesz. Ennek meghatározására egy csúszó ablakot vizsgálunk, ezt szemlélteti a 2.4. ábra. Fontos, hogy a sarokpontok jól ellenállnak a különböző geometriai és fotometriai deformációknak, ezért megbízhatóan detektálhatóak, valamint egy objektum garantáltan leírható a sarokpontjaival.

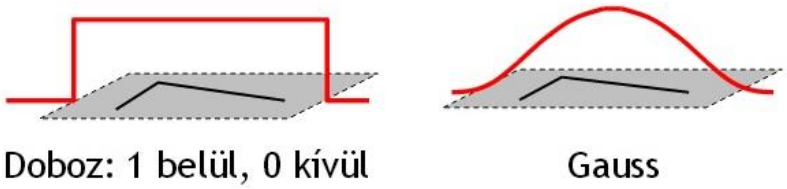


2.4. ábra: Sarokpont detektálás csúszó-ablakot vizsgálva (forrás: [4])

Egy csúszó ablak $w(x,y)$ tartalmának megváltozását egy $[u,v]$ eltolás hatására a következőképpen írhatjuk fel:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2 \tag{2.2}$$

A fenti egyenletben $w(x,y)$ jelöli az ablak függvényt, amely lehet egy „doboz” függvény, vagy egy Gauss-féle függvény (lásd 2.5. ábra) és $I(x,y)$ pedig az intenzitást. Az eltolt intenzitás és az intenzitás különbségének négyzetét tehát súlyozzuk az ablak függvénnyel. Ezt az ablak minden pontjára megtesszük és összegezzük azokat.



2.5. ábra: Doboz és Gauss függvény (forrás: [4])

Az $E(u,v)$ változásának jellemzéséhez fejtsük azt Taylor sorba, a $(0,0)$ pont körül a másod rendű (kvadratikus) tagig. Miután ezt megtettük, néhány egyszerűsítést alkalmazva közvetlenül a képfüggvény parciális deriváltjaival fejezhetjük ki a változást. A kvadratikus közelítés tehát a következő alakban írható fel:

$$E(u,v) \approx [u\ v] M \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.3}$$

ahol M a képfüggvény deriváltjának második momentum mátrixa,

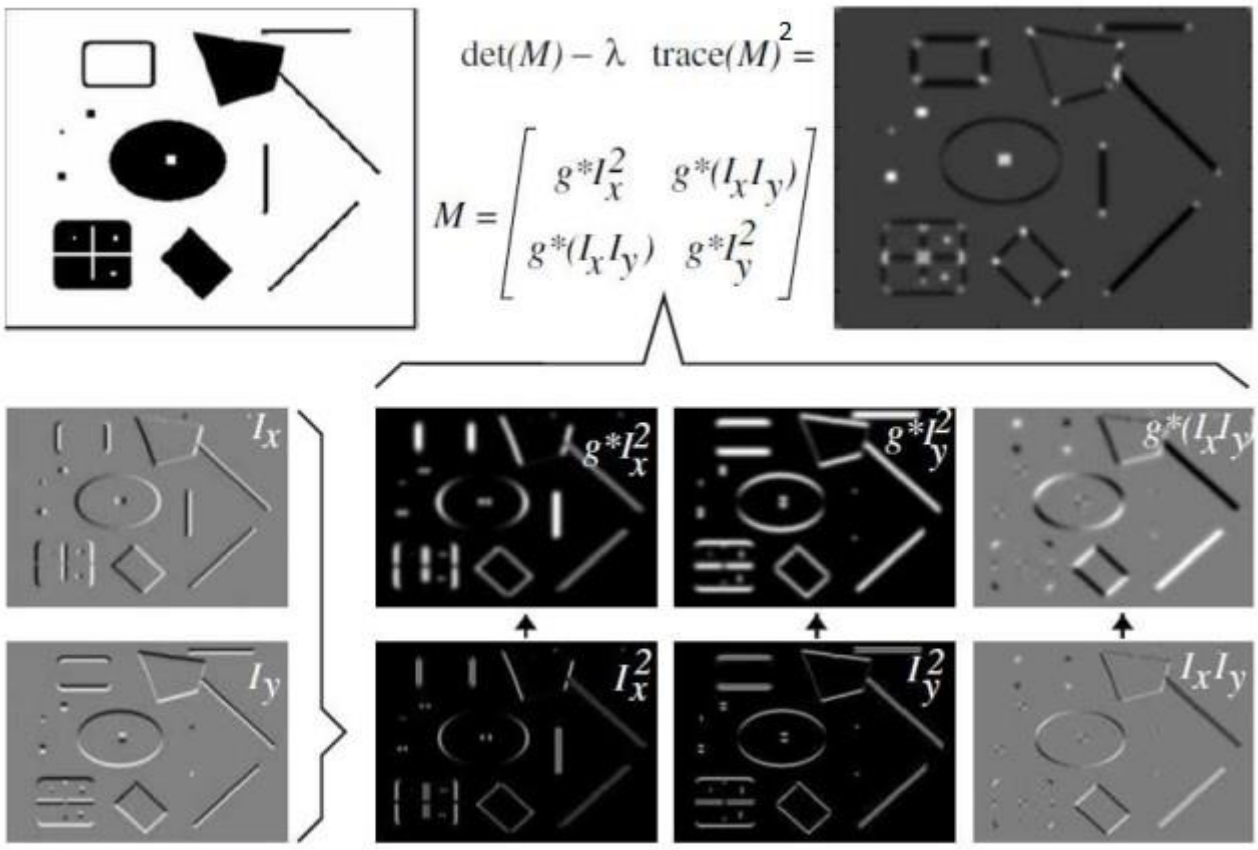
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{2.4}$$

az I_x, illetve I_y pedig a parciális deriváltakat jelölik.

Tengely állású sarokpont esetén M diagonális lesz, általános esetben pedig diagonizálható. Ez egyben azt is jelenti, hogy a képünk pontjait osztályozhatjuk M sajátértékei alapján. Az így előálló, sarkosságot jellemző függvény a következő:

$$R = \det(M) - \alpha * trace(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \quad (2.5)$$

Az α értéke általában a [0.04, 0.06] intervallumban mozog. A fontos pontok azok lesznek, ahol ennek a függvénynek lokális maximuma van, és az nagyobb egy adott küszöbértéknél. A 2.6. ábra egy példaképen keresztül mutatja be a sarokdetektálás menetét.

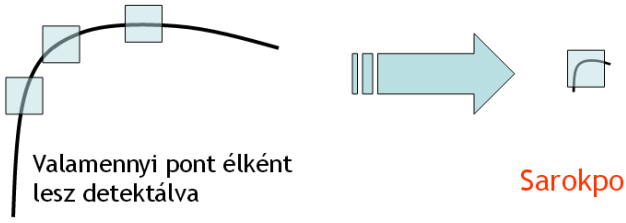


2.6. ábra: Sarokpont detektálásának menete (forrás: [5])

Tehát a Harris sarokdetektáló algoritmus lépései a következők:

1. Számítsuk ki az M mátrixot minden egyes képpont feletti ablakban és ebből megkapjuk az R sarkossági jellemzőt.
2. Keressük meg azokat a pontokat, amelyekre a sarkossági érték nagyobb, mint a megadott küszöbérték.
3. Tartsuk meg ezekből a lokális maximumokat.

Mivel csak deriváltakat használunk ezért, a fenti számítások invariánsak az eltolásra és az intenzitás változására, viszont csak részben invariánsak a skálázásra, mivel az függ a küszöbértéktől is. A 2.7. ábra bal oldalán látható, hogy éleket detektál az algoritmus, viszont egy másik skálázásban ugyanaz a vonal egy sarokként lesz detektálva, mivel a vizsgált ablakba belefér az egész.



2.7. ábra: Harris sarokdetektor nem invariáns a skálázásra (forrás: [4])

A Harris detektort K. Mikolajczyk és C. Schmid fejlesztette tovább, úgy, hogy az invariáns legyen a skálázásra is, ezt nevezik Harris-Laplace detektornak, [6]. Ehhez a képet egy σ_D szórású (differenciális skála) Gauss-kernellel simítjuk, majd ennek a parciális deriváltjaiból képzett második momentum mátrixot vizsgáljuk, amelyet egy σ_I szórású (integráló skála) Gauss-ablakon átlagolunk:

$$w(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.6}$$

$$L(x,y,\sigma) = w(x,y,\sigma) \otimes I(x,y) \tag{2.7}$$

$$L_x = \frac{\partial}{\partial x} L \tag{2.8}$$

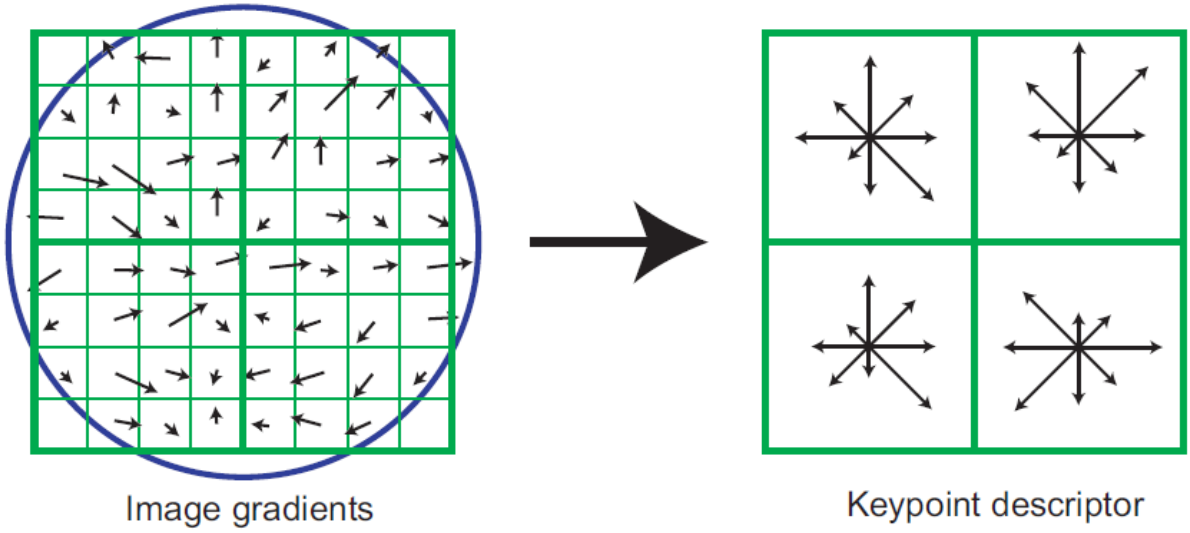
$$M(x,y,\sigma_I\sigma_D) = \sigma_D^2 * w(x,y,\sigma_I) \otimes \begin{bmatrix} L_x^2(x,y,\sigma_D) & L_x(x,y,\sigma_D)L_y(x,y,\sigma_D) \\ L_x(x,y,\sigma_D)L_y(x,y,\sigma_D) & L_y^2(x,y,\sigma_D) \end{bmatrix} \tag{2.9}$$

A sarkosságot jellemző függvény itt is a (2.5) egyenlet. A fenti mátrixot a kép többféle skálázásában is kiszámoljuk. Tehát minden skálán külön detektáljuk a fontos pontokat. Ez után az így kiválasztott fontos pontok halmazán egy Lindeberg által javasolt iteratív módszert futtatunk, [7]. Az algoritmus kilépési pontját az adja, ha két iterációt követően megegyeznek a pontok és skálaszintjeik. A futás után megkapjuk a végleges fontos pontokat.

2.2.2. A SIFT LEÍRÓ

A képosztályozó algoritmusban kizárólag SIFT (Scale Invariant Feature Transform) leírót használok, mint alacsony szintű leíró. A módszert David G. Lowe fejlesztette ki ([8]), amelyhez tartozik kulcspontdetektálás is, azonban azzal most nem foglalkozok. A módszer elsősorban egy konkrét objektum előfordulásainak meghatározására lett kifejlesztve, de emellett jól használható képosztályozási feladatokban is.

A SIFT leíró egy 128 dimenziós vektor, amely egy pont lokális környezetében lévő gradiensek orientációjából számolható. A módszer textúra alapú, és amennyire lehet ellenáll a fényviszonyok változásának. A 2.8. ábra segítségével részletesen bemutatom a SIFT működését.



2.8. ábra: SIFT alapelv (forrás: [8])

Elsőnek a gradiens irányát és nagyságát számoljuk ki minden egyes mintavételi pontban. A mintavételi pontok meghatározásához $n \times n$ cellát jelölünk ki egy fontos pont körül. Minden egyes cellát további $k \times k$ részre osztunk. Ezek lesznek a mintavételi pontok, ahogy az az ábra bal oldalán is látható. A számított gradiensek nagyságát egy Gauss-függvény szerint súlyozzuk, amelyet a bal oldali ábra befoglaló köre szemléltet. Ezt követően a súlyozott gradiensekből cellánként készítünk egy-egy orientáció szöghisztogramot, *d* irányra. Ennek eredménye látható az ábra jobb oldalán. Az ábrán $n=2$, $k=4$ és $d=8$. A SIFT standard paraméterezése a következő: *n=4, k=4, d=8*. Végül a

hisztogramok egymásutánjából kapunk egy n^2d dimenziós vektort. Ez az alapbeállítások mellett 128 dimenziót jelent.

Az említett fényviszony invarianciát a vektor normalizálásával érhetjük el, mivel ha egy kép pixeleit szorozzuk egy konstanssal, a gradiensek nagyságában fellépő változást a normalizálás semlegesíti. A fényerő változások esetén egy konstanst adunk a pixelekhez, viszont ez nem befolyásolja a gradienseket, mivel azt a pixelek különbségeiből számítjuk. A nem-lineáris megvilágítási változásokkal már nagyobb probléma van, mivel nem lehet egységesen kezelni a gradiensek változását. Ezek nagy különbségeket tudnak okozni a gradiensek nagyságában, viszont az irányukat nem változtatják. Tehát ennek a befolyását úgy tudjuk csökkenteni, ha meghatározunk egy küszöb értéket, amellyel korrigáljuk a gradiensek nagyságát. Ennek hatására a gradiensek nagysága nem kap akkora szerepet, mint az orientációja. Ehhez a leíró vektor minden 0,2-nél nagyobb elemét 0,2-re cseréljük, majd újra normalizálunk. Ezt a 0,2-es értéket Lowe javasolta, miután kikísérletezte ugyanazon képek megvilágításbeli változtatgatásával.

Az általam használt SIFT leíró teljesen megegyezik a Lowe által javasolt „standard” SIFT-el.

2.3. VIZUÁLIS KÓDSZAVAK ELŐÁLLÍTÁSA

Az egymáshoz hasonló alacsony szintű leírók jelentenek egy vizuális kódszót. Ezek meghatározásához klaszterezem a tanítóhalmazban található összes képből kinyert alacsony szintű leírót. Erre a feladatra a GMM (Gaussian Mixture Model) módszert használom. A klaszterezés előtt azonban lehetőség van az alacsony szintű leírók dimenziószámának csökkentésére. Erre azért van szükség, hogy a nagy számítási igényt minimalizáljam, illetve a legnagyobb információtartalommal bíró komponenseket emeljem ki. A 128 dimenziós SIFT leírókat PCA (Principal Component Analysis) használatával csökkentem 80 dimenziósra.

2.3.1. FŐKOMPONENS ANALÍZIS

A PCA az egyik leggyakrabban használt jellemzőkinyerő módszer, [9], [10]. Ebben a módszerben a meglévő koordináta tengelyek helyett újakat hozunk létre, báziscserét

alkalmazva. A dimenziószám csökkentését kisebb elemszámú bázisra való áttéréssel érjük el.

2.3.1.1 A FŐKOMPONENS JELENTÉSE

Legyen adott az X, M dimenziós vektor. Az a célunk, hogy olyan új koordináta-rendszert vezessünk be, amelyben az X vektorunk koordinátái korrelálatlanok. Az új koordináta-rendszer első tengelye irányában a legnagyobb az X szórása az összes lehetséges M dimenziós irány közül, és ez a maximum éppen az első kanonikus szórás. Ezt a tengelyt ezért első főkomponensnek is nevezzük. A második koordinátatengely iránya az elsőre merőleges irányok közül az, amelyikre nézve X szórása a legnagyobb, mégpedig éppen a második kanonikus szórással egyenlő. Ez az irány a második főkomponens, és így tovább. Tehát a főkomponensek az X vektor vetületei az új koordinátatengelyekre.

2.3.1.2 AZ ANALÍZIS MENETE

A módszer valójában a minták kovariancia mátrixának a sajátértékeit keresi meg. Legyenek adottak az $x_1 x_2 \dots x_N$ M dimenziós adatvektorok:

1. Annak érdekében, hogy arányosan fejtsék ki hatásukat a főkomponensekre, standardizáljuk azokat, hogy az elemzés kezdetén az átlaguk 0, a varianciájuk pedig 1 legyen.
2. Számítsuk ki a kovariancia mátrixot. Amennyiben az 1. lépést elvégeztük, ez egy korrelációs mátrix lesz.
3. Keressük meg a sajátértékeket, és a hozzájuk tartozó sajátvektorokat. Az i-edik főkomponens együtthatóját az i-edik sajátvektor, varianciáját pedig az i-edik sajátérték adja.
4. Ez után hagyjuk figyelmen kívül azokat a főkomponenseket, amelyek az adatoknak csak csekély arányú varianciáját magyarázzák. Ez által néhány főkomponenstől eltekintünk, és egy alacsonyabb dimenziószámú koordináta-rendszert kapunk.

2.3.2. GAUSSIAN MIXTURE MODEL

A GMM egy generatív módszer az alacsony leírók klaszterezésére, tehát az egyes klaszterekhez egy-egy valószínűségi modellt rendel, [11]. Ebben az esetben egy pont

tartozhat több klaszterbe is, bizonyos valószínűséggel. A pontok valószínűségi sűrűségfüggvényét K darab Gauss-féle függvény súlyozott összegével írja le:

$$p(X|\lambda) = \sum_{j=1}^{K} \omega_j g(X|\mu_j,\sigma_j) \tag{2.10}$$

Itt az $X = \{x_1 x_2 \dots x_N\}$ jelöli az M dimenziós adatvektorokat, λ a keresendő paramétert, $g(X|\mu_j,\sigma_j)$ a Gauss-féle függvényeket (ahol μ_j a várható értéket, σ_j a szórást jelöli), valamint ω_j pedig a súlyozó együtthatókat, amelyekre a következő megkötésnek kell teljesülnie:

$$\sum_{j=1}^{K} \omega_j = 1 \tag{2.11}$$

2.3.2.1 MAXIMUM LIKELIHOOD

A GMM λ paraméterét ML (Maximum Likelihood) becsléssel határozzuk meg. Ennek célja egy olyan λ paraméter jóslása, amely maximalizálja a valószínűségét annak, hogy a megadott ponthalmaz keletkezett. Az x_i adatvektorok közt függetlenséget feltételezve (2.10) a következő módon alakul:

$$p(X|\lambda) = \prod_{i=1}^{N} p(x_i|\lambda) \tag{2.12}$$

Ez a kifejezés nem-lineáris a λ paraméterre nézve, tehát a direkt maximalizálása nem lehetséges. Erre egy speciális iteratív módszert szokás használni, az EM[1] (Expectation Maximization, [12], [13]) algoritmust. Az alapötlet az, hogy induljunk ki egy kezdeti λ paraméterből, majd ebből becsüljünk minden lépésben egy újat (λ'), amire $p(X|\lambda') \geq p(X|\lambda)$ teljesül. Ez után az új paraméter lesz a következő iteráció kezdeti paramétere, és így tovább. Egészen addig folytatjuk az iterációt, amíg konvergenciát nem tapasztalunk, vagy

[1] Az algoritmusnak csak a fontosabb lépéseit ismertetem, a részletes levezetés megtekinthető a megjelölt irodalomban.

elértük a maximálisan megengedett iterációk számát. A legelső kezdeti paraméter meghatározására többféle lehetőség is adódik (én a K-means módszert alkalmaztam).

2.3.2.2 K-MEANS ALGORITMUS

A K-means algoritmus egy klaszterező módszer, [14] melynek során K darab klaszterbe osztjuk a pontokat. Ennél a módszernél minden pont határozottan egy klaszterbe tartozik. A pontokat az elhelyezkedésük alapján csoportosítjuk, tehát egy klaszter tulajdonképpen az egymáshoz közel lévő pontok halmaza lesz, a klaszterközéppont pedig a csoportot alkotó pontok átlaga lesz. Ezeket a középpontokat egy iteratív módszerrel határozzuk meg:

1. Inicializáljuk a klaszterközéppontokat (pl. egyenletes eloszlással a térben)
2. A mintákat abba a klaszterbe soroljuk, melyhez tartozó középpont a legközelebb van a ponthoz
3. A középpontok újradefiniáljuk a klaszterek átlaga alapján
4. Ismételjük a 2. ponttól, amíg változik valamelyik klaszterközéppont

A fenti algoritmusok futtatásával megkapjuk a vizuális kódszavakból álló szótárt. Ez tekinthető a mintahalmazba tartozó képek tömör reprezentálásának. A szótár alapján lehetőségünk van egy olyan leíró megalkotására, amely az adott képen lévő vizuális kódszavak eloszlását jellemzi. Ezt magas szintű leírónak nevezzük. Ennek kiszámításához szükség van a kép alacsony szintű leíróira, melyeket a szótár segítségével egyetlen képi leíróvá alakítunk. Erre a célra én a Fisher-vektort használtam.

2.4. FISHER-VEKTOR ALAPELV ALKALMAZÁSA VIZUÁLIS SZÓTÁRRA

A GMM alapú Fisher-vektort ([15]) alapvetően a generatív modellezés és diszkriminatív osztályozás előnyeinek ötvözése érdekében fejlesztették ki. Képosztályozásban való használatát pedig F. Perronin és C. Dance javasolta 2007-ben, [16].

A GMM tárgyalásánál bevezetett jelöléseknek megfelelően legyen $p(X|\lambda)$ a valószínűségi sűrűségfüggvény, amelynek λ a paramétere, és legyenek $X = \{x_1 x_2 \dots x_N\}$ az

M dimenziós adatvektorok. A sűrűségfüggvény gradiense[2] megadja, hogyan írja le a legjobban a modell az adott képet:

$$\nabla_{\lambda} \log p(X|\lambda) \tag{2.13}$$

Ez a mennyiség lesz a Fisher-vektor. A kiszámítását a következő néhány lépésben mutatom be. A továbbiakban vezessük be az $L(X|\lambda) = \log p(X|\lambda)$ jelölést. Amennyiben feltesszük, hogy az adatvektorok egymástól kölcsönösen függetlenek, akkor $L(X|\lambda)$ felírható a következőképpen[3]:

$$L(X|\lambda) = \sum_{i=1}^{N} \log p(x_i|\lambda) \tag{2.14}$$

Annak a valószínűsége, hogy az x_i adatvektort a GMM generálta a következő:

$$p(x_i|\lambda) = \sum_{j=1}^{K} \omega_j g(x_i|\mu_j, \sigma_j) \tag{2.15}$$

Továbbá jelölje $\gamma_i(j)$ annak a valószínűségét, hogy az x_i adatvektort a j-edik Gauss-függvény generálta:

$$\gamma_i(j) = p(j|x_i, \lambda) = \frac{\omega_j p_j(x_i|\lambda)}{\sum_{jj=1}^{K} \omega_{jj} p_{jj}(x_i|\lambda)} \tag{2.16}$$

A súlyozási együtthatókra itt is érvényes a (2.11) által leírt megkötés. A fentiekből kiderül, hogy a GMM λ paramétere rendre az egyes Gauss-függvények súlya, várható értéke és szórása[4]:

$$\lambda = \{\omega_j, \mu_j, \sigma_j | j = 1 \dots K\} \tag{2.17}[5]$$

[2] A sűrűségfüggvény gradiense annak derivált logaritmusával írható fel.

[3] Logaritmusok összege a szorzatok logaritmusával egyenlő, ezért cserélhető a (2.12) produktuma summára.

[4] Feltesszük, hogy a Gauss-függvények szórása diagonális mátrix.

[5] Az egyenletben a várható értékek és szórások M dimenziósak.

A Fisher-vektor tagjainak meghatározásához, még deriválnunk kell az $L(X|\lambda)$ függvényt. Ezt pontosan a λ paraméter elemei által kijelölt mennyiségek szerint egyenként kell megtenni. Az így kialakuló vektor adja a kép magas szintű leíróját, Fisher-vektorát. Az alábbiakban megadom miként írhatók fel az egyes tagok deriváltjai:

$$\frac{\partial L(X|\lambda)}{\partial \omega_j} = \sum_{i=1}^{N} \left[\frac{\gamma_i(j)}{\omega_j} - \frac{\gamma_i(1)}{\omega_1} \right] \tag{2.18}$$

$$\frac{\partial L(X|\lambda)}{\partial \mu_j^m} = \sum_{i=1}^{N} \gamma_i(j) \left[\frac{x_i^m - \mu_j^m}{(\sigma_j^m)^2} \right] \tag{2.19}$$

$$\frac{\partial L(X|\lambda)}{\partial \sigma_j^m} = \sum_{i=1}^{N} \gamma_i(j) \left[\frac{(x_i^m - \mu_j^m)^2}{(\sigma_j^m)^3} - \frac{1}{\sigma_j^m} \right] \tag{2.20}$$

Itt $j = 1 \dots K$ jelöli a megfelelő Gauss-függvényt, és $m = 1 \dots M$ jelöli az adott dimenziót. Ezek után a gradiens vektorok dimenziónkénti eltérő nagysága miatt még egy további normalizálás is szükséges. Ennek a menetét jelen dokumentumban nem kívánom ismertetni, a pontos levezetés megtalálható a [16], [17] művekben. Annyi pontosítás még szükséges, hogy (2.11) miatt, például ω_1 kiszámítható a többi súlyozási együttható ismeretében, ezért (2.18) esetén csak K-1 szabad paraméter van. A Fisher-vektor tehát a következők szerint alakul ki:

- Az $L(X|\lambda)$-et deriváljuk egyenként a *K* darab Gauss-függvény súlyai szerint. Mivel a (2.11) miatt, például ω_1 kiszámítható a többi súlyozási együttható ismeretében, ezért csak K-1 szabad paraméter van. Tehát ez *K-1* elemet ad a FIsher-vektorba.
- Az $L(X|\lambda)$-et deriváljuk egyenként a *K* darab Gauss-függvény várható értékei szerint. Mivel ezek *M* dimenziósak, ezért ez *K*M* elemet fog adni a Fisher-vektorba.
- Az $L(X|\lambda)$-et deriváljuk egyenként a *K* darab Gauss-függvény szórásai szerint. Ez szintén *K*M* elem.

Ebből következik, hogy a Fisher-vektor összesen $K(2M+1)-1$ dimenziós. Azért volt szükség egy ilyen komplex leíró létrehozására, mert ez alapján tetszőleges képről meghatározható, hogy az milyen mértékben tartalmaz vizuális elemeket a betanított mintahalmazból.

2.5. TANÍTÁS ÉS OSZTÁLYOZÁS

Ahogy azt az 1.2 részben már említettem, gépi tanulásos módszerrel dolgozok. Ehhez adottak a mintahalmaz képein kiszámolt magas szintű leírók $\vartheta_1 \vartheta_2 \dots \vartheta_N$, valamint képenként a *c* dimenziós ***x*** vektor. Egy teszt képet a magas szintű leírója (ϑ) alapján osztályozok. Minden *c* kategóriára adok egy jóslatot $f(\vartheta)$ arra vonatkozóan, hogy szerepel-e az adott kategóriájú objektum a képen. Ezt úgy teszem meg, hogy létrehozok *c* darab egymástól független bináris osztályozót. Minden tanításnál a kiválasztott *c'* sorszámú osztályba tartozó képek a pozitív minták, az összes többi pedig negatív minta. Bináris osztályozónként meghatározok minden képhez egy y_i változót a következő szerint:

$$y_i = \begin{cases} 1, & ha\ x_{c'} = 1 \\ -1, & ha\ x_{c'} = 0 \end{cases} \tag{2.21}$$

A létrehozott y_i változó adja meg, hogy az adott kép pozitív vagy negatív minta a tanítóhalmazban. Ezt „one-against-all" módszernek nevezzük. Egyszerűsége ellenére rendkívül jól használható, ezért én is ezt használom.

Bináris osztályozóként az SVM (Support Vector Machine) egyik változatát használom amelyet C-SVC osztályozónak neveznek. Előreszámolt kernel-mátrix szükséges hozzá, illetve egy C paraméter, amelyet kereszt-validációs módszerrel határozok meg.

2.5.1. BINÁRIS LINEÁRIS OSZTÁLYOZÁS

Bináris osztályozásról akkor beszélünk, ha a választható kategóriák száma (*c* =) 2. Ezen felül az osztályozó lineáris, ha egy hipersíkkal el tudja különíteni a két osztály pontjait. Bináris lineáris esetben tehát a tanítópontokat két részre szeparálja egy hipersík.

Formálisan, az $n \times \vartheta + B = 0$ egyenletű[6] hipersík lineárisan szeparálja a tanítópontokat, ha:

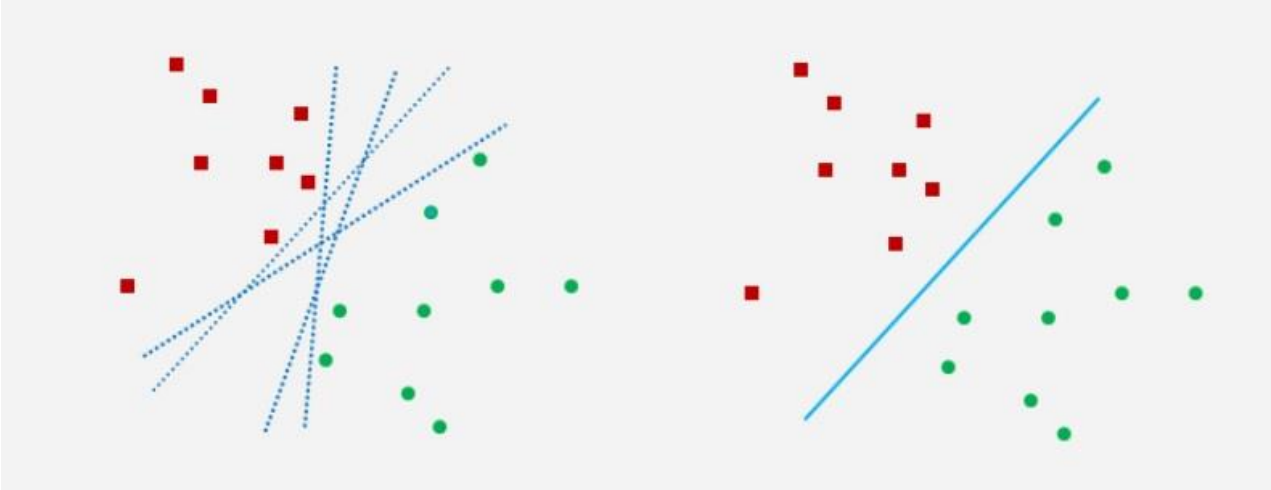
$$\begin{aligned} n \times \vartheta_i + B \geq a > 0, \quad & ha\ y_i = 1 \\ n \times \vartheta_i + B \leq -a < 0, \quad & ha\ y_i = -1 \end{aligned} \qquad i = 1 \ldots N \tag{2.22}$$

Egy új pont a térben (teszt kép) a hipersíktól vett előjeles távolsága alapján sorolható az egyik vagy a másik osztályba. Ezt a mennyiséget a pont és a hipersík normálvektorának skaláris szorzata adja. Tehát egy ϑ vektorra adott $f(\vartheta)$ jóslat a következőképpen írható fel:

$$f(\vartheta) = g(n * \vartheta) = g\left(\sum_{m=1}^{M} n_m \vartheta_m\right) \tag{2.23}$$

Itt *M* jelöli a dimenziók számát, a *g* függvény pedig a megfelelő értékkészletbe képezi az osztályozó kimenetét. Mivel az *n* vektorral tulajdonképpen súlyozzuk a ϑ vektort, ezért súlyvektornak is nevezhetjük.

2.5.2. SZUPPORT VEKTOR GÉPEK



2.9. ábra: Egy lineárisan szeparálható feladat különböző megoldásai (forrás: [20])

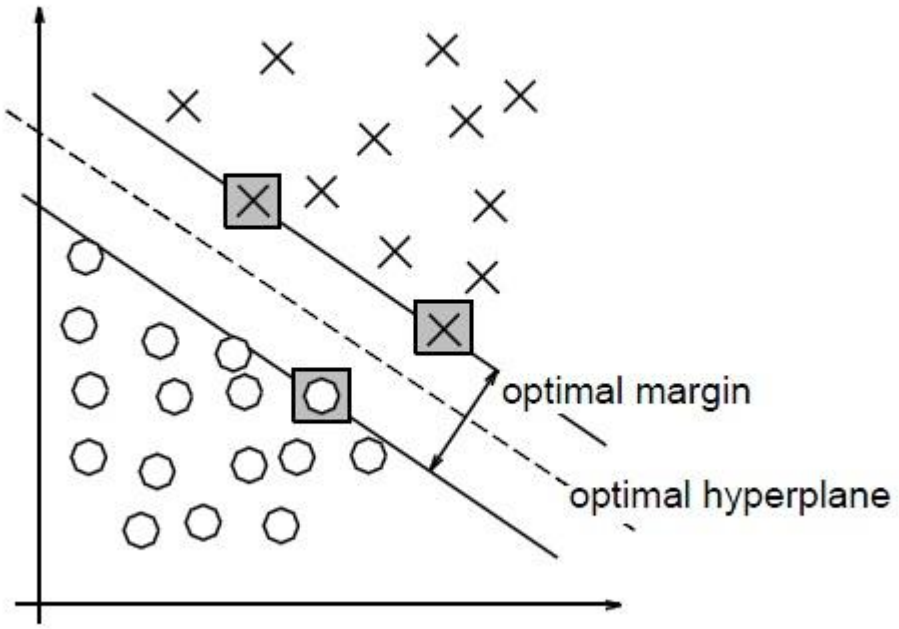
A B. Boser, I. Guyon és V. Vapnik által javasolt SVM ([18]) alapváltozata lineáris szeparálásra képes, de egy kiterjesztett változata nemlineáris szeparálásra is

[6] Itt a ϑ jelöli a magas szintű leírót.

lehetőséget nyújt. Lineárisan szeparálható esetben végtelen sok lineárisan szeparáló hipersík létezik, amelyek mindegyike hibátlanul szétválasztja a tanítópontokat. A szeparálás minősége azonban az egyes megoldások esetén jelentősen eltérő lehet (2.9. ábra). Az az elválasztás eredményez jobb általánosító képességet, amely a két osztályba tartozó tanítópontok között, a tanítópontoktól a lehető legnagyobb távolságra helyezkedik el. A lineáris bináris osztályozási feladat megoldását adó SVM ezt az „optimális" szeparátor síkot határozza meg. A pontok közt középre elhelyezett szeparáló felületet a tanítópontoktól egy margó, azaz egy biztonsági sáv választja el.

2.5.2.1 A MAXIMÁLIS MARGÓJÚ SZEPARÁLÁS

Az SVM alapelveit ez képezi. Ez a módszer a lineárisan szeparálható feladatok lineáris megoldását adja, amihez a szeparáló hipersíkok közül a maximális margójút választja (2.10. ábra), így biztosítja a fentebb leírt tulajdonságot.



2.10. ábra: Az optimális hipersík margója (forrás: [19])

A hipersík paramétereinek skálázhatósága lehetővé teszi, hogy felírjuk a margón elhelyezkedő tanítópontok és a hipersík közti távolságra a következőket:

$$n \times \vartheta_i + B \geq 1, \qquad ha\ y_i = 1$$
$$n \times \vartheta_i + B \leq -1, \qquad ha\ y_i = -1 \tag{2.24}$$

Ekkor a szeparáló hipersíkhoz legközelebb eső, különböző osztályba tartozó bemeneti vektorok közötti, a síkra merőlegesen mért távolság a következő:

$$\frac{1}{\|n\|} - \frac{-1}{\|n\|} = \frac{2}{\|n\|} \tag{2.25}$$[7]

Az optimális hipersík által biztosított margó tehát: $\frac{1}{\|n\|}$. Ebből az következik, hogy az optimális hipersík az, amelyikre $\|n\|$ minimális, az alábbi feltétel mellett:

$$y_i(n \times \vartheta_i + B) \geq 1 \qquad i = 1 \dots N \tag{2.26}$$

A fentiek alapján tehát egy (n^{opt}, B^{opt}) párost keresünk, amire teljesül minden kritérium. Ez egy feltételes szélsőérték-keresési probléma, amelynek megoldását egy Lagrange kritérium megoldásával kereshetjük. Az ebből következő duális probléma egy kvadratikus programozási feladathoz vezet. Az ide vágó levezetésektől eltekintek[8], egyből a probléma megoldása utáni részre ugrok.

Az optimumhoz tartozó Lagrange multiplikátorok (α^*-k) nagy része általában 0, így mind a súlyvektor kifejezésében, mind az SVM válaszában a tanítópontoknak csak egy része (N helyett N_S) vesz részt. Azokat a tanítópontokat, amelyek részt vesznek a megoldás kialakításában, szupport vektoroknak nevezzük (innen ered az elnevezés is). Ezért az SVM-ben a tér tényleges dimenziója nem a tanítópontok számával, hanem a szupport vektorok számával egyezik meg. Ez jelentős egyszerűsítést jelent a válasz számításában, különösen abban az esetben, ha $N_S \ll N$. Az SVM-nek ezt a tulajdonságát ritkasági tényezőnek nevezzük.

[7] Az || || operátor az euklideszi normázás műveletét jelöli.

[8] A probléma pontos levezetése megtalálható itt: [19], [20].

Továbbá az is igaz, hogy ahol $\alpha_i^* > 0$, ott $y_i(n^{opt} \times \vartheta_i + B^{opt}) = 1$, vagyis a szupport vektorok a hipersíkhoz legközelebbi tanítópontok, amelyek pontosan a margó határán helyezkednek el.

Az n^{opt} a szupport vektorok és a Lagrange együtthatók által meghatározható:

$$n^{opt} = \sum_{i \in N_S} \alpha_i^* y_i \vartheta_i \tag{2.27}$$

Az optimális eltolás értéke (bias) B^{opt} az $y_i(n^{opt} \times \vartheta_i + B^{opt}) = 1$ egyenletből számolható, ahol ϑ_i egy szupport vektor.

Végül, az SVM válasza a következő formában írható fel:

$$f(\vartheta) = \sum_{i \in N_S} \alpha_i^* y_i \vartheta_i \times \vartheta + B^{opt} \tag{2.28}$$

2.5.2.2 A SZOFT MARGÓJÚ SZEPARÁLÁS

A gyakorlatban ritkán adódik olyan feladat, amely lineárisan szeparálható úgy, hogy még osztályozási tartalék is biztosítható. Általában a hibamentes lineáris szeparálás nem lehetséges. Azonban néhány mintapont esetén megengedhető, hogy a margón belülre kerüljenek. Ezt lágy vagy szoft margójú megoldásnak nevezzük. A módszer lineárisan nem szeparálható feladatok lineáris megoldását teszi lehetővé.

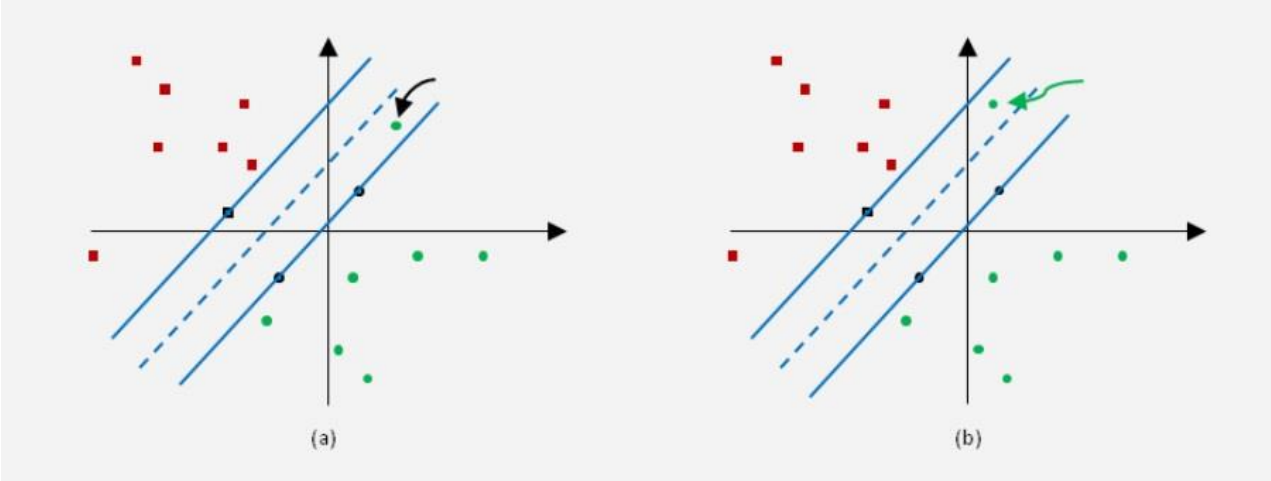
A (2.26) feltétel ebben az esetben nem állja meg a helyét minden i-re, így egy új feltétel szükséges, ahol úgynevezett gyengítő változókat használunk, hogy a (2.26) egyenlőtlenségnek nem megfelelő pontokat „büntessük”:

$$y_i(n \times \vartheta_i + B) \geq 1 - \xi_i \qquad i = 1 \dots N \tag{2.29}$$

A fenti egyenlőtlenségben ξ_i a gyengítő változó, amelyre igazak a következő megállapítások:

- $\xi_i = 0$ esetén visszakapjuk a (2.26) egyenlőtlenséget
- $0 < \xi_i \leq 1$ esetben tekintsük a 2.11. ábra(a) részét

- $\xi_i > 1$ esetre pedig a 2.11. ábra (b) részén láthatunk példát



2.11. ábra: (a): mintapont osztályozása helyes, de a mintapont a margóra, vagy a margó és a hipersík közé esik. (b): a mintapont osztályozása helytelen (forrás: [20])

Az optimális hipersíkot a maximális margójú esethez hasonlóan határozzuk meg, annyi módosítással, hogy a hibás osztályozások száma minimális legyen, miközben továbbra is törekszünk a lehető legnagyobb margó elérésére. Ennek megfelelően az $\|n\|$ helyett a következő kifejezést kell minimalizálni:

$$J(n) = \frac{1}{2}n^2 + C\left(\sum_{i=1}^{N}\xi_i\right) \tag{2.30}$$

Itt a *C* paraméter a két tag közti kompromisszumot állítja be. *C=0* esetén kapjuk a maximális margójú esetet. A *C*-t hiperparaméternek is szokás nevezni, és általában kereszt-validációs módszerrel határozhatjuk meg.

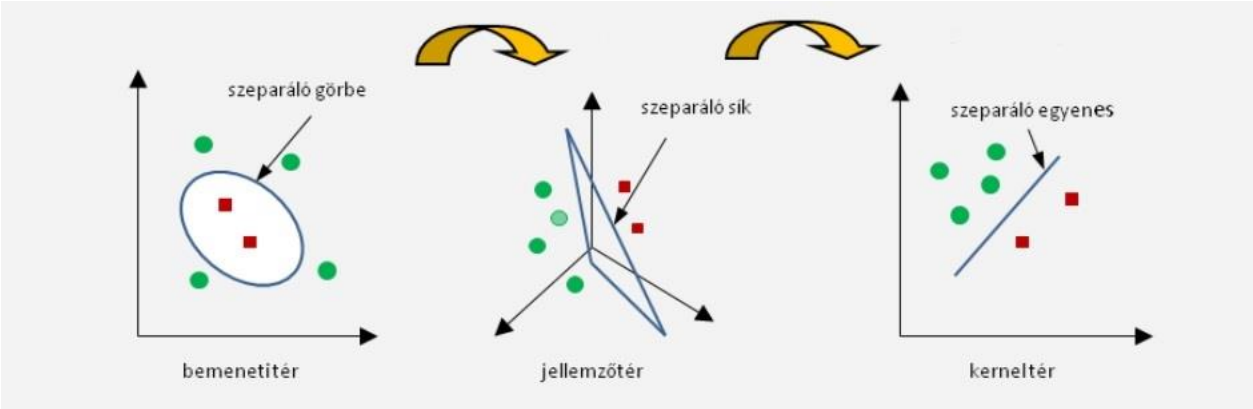
A megoldást most is a megfelelő Lagrange kritérium felírása adja. A gyengítő változók bevezetése az optimalizálási feladat duális alakját csak a Lagrange multiplikátorokra vonatkozó feltételekben módosítja. A α^*-k egy része most is 0, tehát ugyanúgy lesznek szupport vektorok, és egyéb, a megoldást nem befolyásoló mintapontok. Az osztályozó válasza megegyezik a (2.28)-ban felírtakkal. A B^{opt} viszont az $y_i(n^{opt} \times \vartheta_i + B^{opt}) - 1 + \xi_i = 0$ egyenletből számolható.

2.5.2.3 NEMLINEÁRIS SZEPARÁLÁS

Az osztályozási feladatok túlnyomó része lineárisan nem szeparálható. Ahhoz, hogy lineárisan szeparálhatóvá tegyük, egy dimenziónövelő, nemlineáris transzformációt hajtunk végre $x \rightarrow \varphi(x)$. Ezzel az úgynevezett jellemzőtérbe transzformáljuk a pontokat. A jellemzőtérben az SVM-nek megfelelően ismét a maximális margójú szeparáló hipersíkot keressük. Viszont nem a kapott jellemzőtérben oldjuk meg a feladatot, hanem az ebből származtatott, úgynevezett kernel térben.

A kernel trükk

Nemlineáris esetben a kernel reprezentációt két lépésben érhetjük el. Elsőként a bemeneti térre egy nemlineáris dimenziónövelő transzformációt hajtunk végre ($x \rightarrow \varphi(x)$). Így kapjuk meg a jellemzőtérbeli reprezentációt. Egy újabb transzformáció után a kernel térbe juthatunk. A jellemzőtér és a kernel tér közt a kapcsolatot a skaláris szorzás teremti meg ($\varphi(\vartheta) \rightarrow K(\vartheta, \vartheta_i) = \varphi(\vartheta) \times \varphi(\vartheta_i)$). Fontos, hogy egy nemlineáris feladatmegoldás során nem a 2.12. ábra által szemléltetett utat járjuk be, hanem pont a fordítottját. Ettől nevezik ezt „trükk”-nek, hiszen egyből a kernel térből indulunk ki, ami rögtön definiálja a jellemzőteret is. Ennek előnye, hogy nem kell explicit definiálnunk a jellemző teret, ami sok esetben nem triviális feladat. A megoldást a kernel térben nyerjük, viszont ehhez egy megfelelő kernel függvény definiálása szükséges.



2.12. ábra: Transzformáció a bemeneti tértől a kernel térig (forrás: [20])

Kernel függvények

A kernel függvények, más néven magfüggvények az adatok közti hasonlóságot mérik. Kernel függvényként csak olyan függvény használható, amely belső szorzatból

származtatható. A származtatás módjából adódóan a kernel függvényeknek bizonyos tulajdonságokat ki kell elégíteniük:

1. Egy kernel függvénynek mindig két argumentuma van, és ezekre nézve szimmetrikus
2. A függvény értéke nem lehet negatív és radiálisan szimmetrikus kell, hogy legyen
3. Azonos argumentumok esetén az értéke maximális
4. Két argumentum távolságának monoton csökkenő függvénye

Az alábbi táblázatban megadom a legelterjedtebb kernel függvényeket.

Lineáris	$K(\vartheta, \vartheta_i) = \vartheta \times \vartheta_i$
Polinomiális (d fokszámú)	$K(\vartheta, \vartheta_i) = (\vartheta \times \vartheta_i + 1)^d$
Gauss-féle (radiális bázisfüggvények, RBF)	$K(\vartheta, \vartheta_i) = e^{-\frac{\|\vartheta - \vartheta_i\|^2}{2\sigma^2}}$
Tangens hiperbolikusz (k és θ konstansok)	$K(\vartheta, \vartheta_i) = tanh(k(\vartheta \times \vartheta_i) + \theta)$

2.1. táblázat: Kernel-függvények

Az általam használt C-SVC osztályozóban RBF kernel függvényt használtam. Más kernel függvény is használható lett volna, viszont a teszt eredmények alapján ez teljesít a legjobban.

Szeparálás a kernel térben

Nemlineáris megoldás a lineáris eset megoldásából egyszerűen a $\vartheta \to \varphi(\vartheta)$ helyettesítéssel érhető el. Ilyenkor az optimális hipersíkot az $n \times \varphi(\vartheta) + B = 0$ alakban keressük. A megoldáshoz most is a Lagrange multiplikátorok meghatározása szükséges. A különbség annyi, hogy a duális feladatnál a $\vartheta \to \varphi(\vartheta)$ helyettesítést alkalmazzuk. Amennyiben a jellemzőtérben a maximális margójú megoldást kapjuk, akkor a multiplikátorokra szabott feltétel $0 \leq \alpha_i$, ha csak a szoft margójú változat megoldható, akkor a feltétel $0 \leq \alpha_i \leq C$.

A fentiek alapján az SVM válasza ebben az esetben a következő:

$$f(\vartheta) = \sum_{i \in N_S} \alpha_i{}^* y_i \varphi(\vartheta_i) \times \varphi(\vartheta) + B^{opt} \tag{2.31}$$

A kernel trükköt felhasználva, a $\varphi(\vartheta_i) \times \varphi(\vartheta)$ szorzatot felírhatjuk $K(\vartheta, \vartheta_i)$ formában, egy magfüggvényként. Ezt bevezetve a nemlineáris osztályozó válasza:

$$f(\vartheta) = \sum_{i \in N_S} \alpha_i{}^* y_i K(\vartheta, \vartheta_i) + B^{opt} \tag{2.32}$$

Külön nem említettem, viszont (2.31) és (2.32) alapján látszik, hogy nemlineáris esetben is lesznek szupport vektorok, tehát az SVM ilyenkor is tud „ritka” megoldást adni. Az SVM megoldás előnye a többi hasonló osztályozóval szemben abból származik, hogy a $\varphi(\vartheta)$ nemlineáris transzformációt nem közvetlenül, hanem a kernel függvényen keresztül közvetve definiáljuk. Így akár végtelen térbe dimenziójú is képezhetjük az adatokat, ha a kernel számolható (például RBF magfüggvényt használva).

2.5.3. ELŐRE SZÁMOLT KERNEL-MÁTRIX

Kernel térben megvalósított osztályozó esetén általános eljárás egy úgynevezett kernel-mátrix felépítése, amelyben megtalálhatók a magas szintű leírókból páronként alkotott kernel-függvények. Erre azért van szükség, hogy a (2.31) egyenletben látható magas szintű leírók jellemző térbeli reprezentációinak szorzata helyettesíthető legyen a megfelelő magfüggvénnyel (2.32 egyenlet). Jelölje $\vartheta_1 \vartheta_2 \dots \vartheta_N$ a magas szintű leírókat. Ekkor az ezekből alkotott kernel-mátrix a következő szerint alakul:

$$\begin{bmatrix} K(\vartheta_1, \vartheta_1) & \dots & K(\vartheta_1, \vartheta_N) \\ \vdots & \ddots & \vdots \\ K(\vartheta_N, \vartheta_1) & \dots & K(\vartheta_N, \vartheta_N) \end{bmatrix} \tag{2.34}$$

Itt K egy tetszőleges kernel-függvényt jelöl. Egy osztályozó tanítása és egy osztályozóval való tesztelés eltérő kernel-mátrixot igényel. Ez azért van így, mert tanítás esetén a tanító képek egymással alkotott magfüggvényei szükségesek, tesztelés esetén pedig egy teszt képet minden tanító képpel kell párba rendezni, viszont a tanítókat, illetve tesztelőket saját magukkal nem.

2.6. A MEGVALÓSÍTOTT ALGORITMUS

Most, hogy a felhasznált módszerek elméletét bemutattam, egy összefüggő leírást adok, miként látom el az ismeretlen képeket tartalmi információkkal.

A megadott mintahalmaz (tanulóállomány) összes képét felhasználva előállítok egy projekció mátrixot, amelynek segítségével a 128 dimenziós SIFT leírók 80 dimenziósra csökkenthetőek. Ehhez kiszámítom minden kép alacsony szintű leíróját, majd a leírók egymás után fűzéséből kapott mátrixra végrehajtom a főkomponens analízist. Ez után az alacsony leírókat redukálom 80 dimenziósra. Az így kapott SIFT leírók klaszterezéséből kapott klaszterközéppontokat adom át az EM algoritmusnak. A vizuális szótárt (GMM) tehát redukált dimenziószámú alacsony leírókból hozom létre. Ez azért van így, mert a szótár előállításánál fő szempont az, hogy tömör legyen. Ez alapján meghatározom minden képnek a Fisher-vektoros reprezentációját.

Ezen a ponton két részre osztom a mintahalmazt, egy tanító és egy megerősítő részre. Mivel a kernel térben való számítás ellenére is csak szoft margójú megoldás lehetséges, ezért meghatározok egy optimális C paramétert, amely mellett a lehető legjobban teljesít az osztályozó algoritmus. Az ilyen osztályozót C-SVC osztályozónak nevezik. A megfelelő C paraméter meghatározásához kereszt-validációs módszert használok. Ennek lényege, hogy minden iterációban végrehajtok egy tanítási és egy tesztelési folyamatot ugyanazokkal a beállításokkal, csak a C változik. Az eredményeket kiértékelem, és azt a C-t választom, amely mellett a legjobb eredményt kaptam. A tanítást a tanító halmazon a tesztelést a megerősítő halmazon végzem. Ehhez a tanító halmazba tartozó képek magas szintű leíróiból felépítek egy kernel mátrixot, valamint felépítek egy olyat is, ahol minden tanító halmazbeli kép Fisher-vektorát, minden megerősítő halmazbeli kép Fisher-vektorával párosítottam. Az előbbit tanításra, az utóbbit tesztelésre használom.

Ez után a teljes mintahalmazba tartozó képek Fisher-vektoraiból építek fel egy kernel-mátrixot, és azt használom fel a végső tanításra, valamint a kapott C paramétert. Ezzel az osztályozó megtanulja a tanulóállomány képeinek osztálycímkéit. A tanítás eredménye kategóriánként egy modell, amelyek a mintahalmaz képeiből előállított Fisher-vektorok szeparálásából jöttek létre az alapján, hogy melyik képhez milyen

előzetes annotáció párosul. Ezek alapján egy tetszőleges kép Fisher-vektorát felszanálva eldönthető, hogy az melyik kategóriába tartozik.

Következő lépésben a megadott ismeretlen képhalmazt (teszthalmazt) tesztelem a betanított osztályozóval. Ehhez először kiszámítom a tesztképek alacsony szintű leíróit. Egy teszt kép osztályozásához meg kell határoznom a rajta szereplő vizuális kódszavak eloszlását, tehát a magas szintű leíróját. Ezt a tanítási folyamat során már kiszámított szótár alapján teszem meg, ahogy a 2.4 alfejezetben kifejtettem. A mintahalmaz képeinek Fisher-vektorait a teszthalmaz képeinek Fisher-vektoraival párosítva létrehozok egy kernel-mátrixot. Ezt adom át az osztályozónak a teszteléshez. A tesztelés eredményeként minden tesztképhez kapok egy jóslatot, hogy az a betanított kategóriák közül, melyikbe milyen valószínűséggel tartozik bele. Ezeket az adatokat egy adatbázisba rendezve lemezre mentem, és a szemantikus képkeresést ezek alapján fogom futtatni. Az alábbi ábrán áttekinthető a teljes folyamata az algoritmusnak.

2.13. ábra: Az elemzést végző algoritmus lépései

A Matlab-ban készült implementációmhoz felhasználtam néhány előre elkészített könyvtárat, amelyek a bonyolult matematikai algoritmusok megértésében, illetve alkalmazásában segítettek. Ezek a következők:

- VLFeat library, [21]
- LIBSVM library, [22]
- enceval-toolkit[9], [23]

[9] Ebből a csomagból a gmm-fisher library-t használom, amelyet Jorge Sánchez készített, 2009-ben. Ez egy különálló C++ megvalósítás, amely a GMM klaszterezés EM algoritmussal kivitelezett változatát, és a Fisher-vektor készítését implementálja.

3. A KÉPKERESÉS MŰKÖDÉSE

Az előző fejezetben már bemutattam, hogyan állítom elő azt az adatbázist, amelyet a keresés során használok. A keresési térbe tartozó képek elemzései offline módon vesznek részt a keresésben, hiszen nem kell minden keresés indításánál újból kiszámítani, már rendelkezésre állnak. Folyamatosan bővülő, vagy változó adathalmaz esetén szükséges az adatbázis frissítése, hogy folyamatosan konzisztens legyen, az újonnan bekerülő képek, vagy esetleg az eltávolítottak miatt. Amennyiben feltételezzük, hogy az adatbázis tartalma mindig naprakész, a keresést az abban található adatok közt futtatjuk.

Az általam készített kereső rendszer a többobjektumos kereső kifejezésekre specializálódik. Ez azt jelenti, hogy keresési kulcsnak kizárólag objektum adható meg, vagy több objektum egymás után a megfelelő elválasztással, attól függően, hogy ’ÉS’, illetve ’VAGY’ kapcsolatban vannak-e egymással. Mivel a keresést egy fix adatbázison futtatom, ezért a kereshető objektumok halmaza is kötött.

A keresés tulajdonképpen egy rangsorolás, amelyet adott esetben egy szűrő feltétel előz meg. Mivel az adatbázisban valószínűségi értékeket tárolok, ezért a rangsorolásnál az a kép kerül legelőre, amelyikről a legvalószínűbb, hogy releváns az adott keresés szempontjából. A rendszerhez több különböző keresési módszert dolgoztam ki, majd ezeket implementáltam.

3.1. MÓDSZEREK

Az előre elkészített adatbázisban képenként egy valószínűségi jellegű vektor található, amelynek minden eleme egy betanított kategóriához tartozó konfidencia értékek. Ezek az értékek azzal arányosak, hogy az adott kép milyen valószínűséggel sorolható bele az adott kategóriába. A rangsor felállítása során ezeket a konfidenciákat használom. A visszaadott találatok kialakítására, pontosabban a találatok sorrendezésére több különböző módszert is megalkottam, melyek közös jellemzője, hogy az osztályozók eredményeit használják fel. A több saját módszert annak az érdekében hoztam létre, hogy össze tudjam őket hasonlítani, illetve a legjobbat ki tudjam közülük választani. A

módszerek abban térnek el egymástól, hogy hogyan kombinálják a különböző osztályozók által nyújtott konfidencia értékeket.

3.1.1. ÖSSZEG

Az összeg módszer alapja, hogy a keresési kulcsban megadott objektumoknak megfelelő betanított kategóriákhoz tartozó valószínűségi jellegű értékek összeadásával képezi a képhez tartozó relevancia értéket, amely alapján az megkapja a rangsorbeli helyét. Két változatát találtam ki, amelyek a következők:

1. (Ö1): A rangsorolás előtt nincs semmilyen szűrés, minden kép részt vesz a rangsorolásban. A visszaadott találati (rangsorolt) lista tehát a következő aggregált változó alapján történik csökkenő sorrendben: $p(k) = \sum_i p_i(k) \mid i \in query$, ahol $p_i(k)$ a k azonosítójú kép *i* kategóriához való tartozásának konfidencia értéke és az *i* kategória megjelenik a felhasználó „query” keresésében.
2. (Ö2): A rangsorolást szűrés előzi meg. A szűrési feltétel, hogy a megadott objektumokhoz tartozó valószínűségi értékek közül legalább az egyiknek meg kell haladnia egy megadott küszöbértéket. A visszaadott találati lista itt a következő aggregált változó alapján történik csökkenő sorrendben:
$$p(k) = \begin{cases} \sum_i p_i(k) & \mid i \in query,\ \exists p_i(k) > threshold \\ 0 & otherwise \end{cases}$$

Ez a módszer a ’VAGY’ típusú kereséseknél lesz a leghatékonyabb, ahol elég a felsorolt objektumok közül az egyiknek eleget tenni.

3.1.2. SZORZAT

A szorzat hasonlóan működik, mint az összeg, viszont a megfelelő valószínűségi értékeket nem összegezzük, hanem szorzatukat képezzük, és ez adja majd meg a rangsorbeli helyüket meghatározó mértéket. Ennek is két változata van:

1. (S1): Az Ö1 módszerhez hasonlóan semmilyen szűrést nem végez a keresési térbe tartozó képek között. A visszaadott találati (rangsorolt) lista tehát a következő aggregált változó alapján történik csökkenő sorrendben:

$p(k) = \prod_i p_i(k) \mid i \in query$, ahol $p_i(k)$ a k azonosítójú kép *i* kategóriához való tartozásának konfidencia értéke és az *i* kategória megjelenik a felhasználó „query” keresésében.

2. (S2): A szűrő feltétel, hogy a megadott objektumokhoz tartozó összes valószínűségi értéknek meg kell haladnia egy megadott küszöbértéket. A visszaadott találati lista itt a következő aggregált változó alapján történik csökkenő sorrendben: $p(k) = \begin{cases} \prod_i p_i(k) & \mid i \in query, \ \forall p_i(k) > threshold \\ 0 & otherwise \end{cases}$

Az ’ÉS’ típusú keresések esetén lesz igazán hatékony, ahol a felsorolt objektumok közül mindegyiknek eleget kell tenni.

3.1.3. MINIMUM FÜGGVÉNY

A minimum függvény a szorzathoz hasonlóan az ’ÉS’ kapcsolatot fejezi ki. A keresési kulcsban megadott objektumoknak megfelelő betanított kategóriákhoz tartozó valószínűségi értékek közül a legkisebbet választja, és az lesz az adott képhez tartozó relevancia érték. Az eljárás nevét röviden csak M1-nek fogom jelölni, és csak szűrő feltétel nélkül alkalmazható. A visszaadott találati (rangsorolt) lista tehát a következő aggregált változó alapján történik csökkenő sorrendben: $p(k) = \min\{p_i(k)\} \mid i \in query$, ahol $p_i(k)$ a k azonosítójú kép *i* kategóriához való tartozásának konfidencia értéke és az *i* kategória megjelenik a felhasználó „query” keresésében.

3.1.4. ÖSSZEG, TÖBBCÍMKÉS OSZTÁLYOZÁSSAL

Ez a megalkotott módszerem a legösszetettebb. Első lépésben egy többcímkés osztályozással határozom meg (pontosabban a módszerem határozza meg), hogy a betanított kategóriák közül melyekbe tartozik egy adott kép.

Azt az általános módszert használom, hogy kiválasztom az összes valószínűségi érték közül a legnagyobbat. Az ehhez tartozó osztályba a legvalószínűbb, hogy a kérdéses kép beletartozik. Ez után újravizsgálom a maradék kategóriákat, és ha találok olyat, amelyhez tartozó konfidencia érték meghaladja az előzőként kiválasztottnak a 75%-át, akkor azt hozzáveszem a képhez jósolt kategóriák halmazához. Ezt követően újra végigjárom a

megmaradtakat, és ugyanezt megteszem. Addig folytatom, amíg teljesül a 75%-os feltétel, vagy el nem fogynak a betanított kategóriák.

Ez után kiválasztom azokat a képeket, amelyek esetén a megbecsült kategóriák halmaza megegyezik a keresési kulcsban megadott objektumokkal. Ezeket teszem a rangsor elejére, további rendezés nélkül, ez lesz a rangsor első szintje. Következőnek azokat a képeket választom, amelyek csak egyetlen objektumnak nem felelnek meg a kereső kifejezésből. Ezek kerülnek a rangsor következő helyeire, további rendezés nélkül, ez lesz a második szint. Ezt addig folyatatom, amíg el nem fogynak a „query”-ben megadott objektumok. Azokat a képeket egyáltalán nem veszem be a rangsorba, amelyekhez becsült kategóriák közül egyik sem található meg a „query”-ben. Eredményként kapok egy félig rendezett rangsort, amelyen végrehajtom az Ö1 módszert, figyelve arra, hogy a szintek között nem mozoghatnak a képek. Ezt a módszert röviden T1-nek fogom jelölni.

3.1.5. KÜLÖNBSÉG

Ebben a módszerben a maximális valószínűséget felhasználva dupla komplementálással képzem a relevancia értéket. A visszaadott találati (rangsorolt) lista tehát a következő aggregált változó alapján történik csökkenő sorrendben: $p(k) = 1 - \prod_i (1 - p_i(k)) \mid i \in query$, ahol $p_i(k)$ a k azonosítójú kép *i* kategóriához való tartozásának konfidencia értéke és az *i* kategória megjelenik a felhasználó „query” keresésében. A módszerben (K1) nincs szűrőfeltétel, a rangsorolásban minden kép részt vesz.

4. A KÉPKERESÉSI KÍSÉRLET KÖRÜLMÉNYEI

4.1. A KIVÁLASZTOTT KATEGÓRIÁK, KERESÉSI KULCSOK

A dolgozatban bemutatott kísérlet legfontosabb jellemzője, hogy a saját eredményeimet a Bing és a Flickr keresők eredményeivel hasonlítom össze. Szükségszerű tehát, hogy olyan keresési kulccsal dolgozzak, amelyekre a Bing és a Flickr is értékelhető találati listát képes adni.

A kombinált képkeresés alapja, hogy a keresésre megadott kifejezés egynél több egységből tevődik össze. Jelen esetben egy egység nem más, mint egy objektum. A keresésekhez kettő illetve három objektumból összeállított keresési kulcsokat használok. A kiválasztott objektumok egyenként a következők: repülőgép, busz, autó, motorbicikli. A négy járműből összesen 7 különböző keresési kulcsot alkotok. Ezeket a 4.1. táblázat foglalja össze.

Kettő objektumos					Három objektumos	
1.	2.	3.	4.	5.	6.	7.
repülőgép	repülőgép	busz	busz	autó	repülőgép	busz
busz	autó	autó	motorbicikli	motorbicikli	busz	autó
					autó	motorbicikli

4.1. táblázat: A lefuttatott keresések kulcsai

A fenti kereső kifejezésekben szereplő objektumok viszonya minden esetben ’ÉS’ kapcsolat, tehát az együttes jelenlétüket keresem a képeken.

4.2. A FELHASZNÁLT KÉPHALMAZOK

4.2.1. TANÍTÁSHOZ

A tanításhoz használt mintahalmazt a Pascal VOC képi klasszifikációs verseny [24])weboldaláról töltöttem le, és a 2007-es évben közzétett képeket használtam[10]. Ez az állomány eredetileg a Flickr-ről származik, a teljes mintahalmaz 20 különböző

[10] A 2007-es verseny részletei itt találhatók: [25].

kategóriából tartalmaz képeket, amelyek között megtalálható az általam használt 4 jármű. Az egész halmazban összesen 5011 kép található, melyből 2501 a tanító és 2510 a validáló kép. A teljes halmazra nincs szükségem, ezért a 20 osztályt leszűkítettem 7 osztályra. A kiválasztott 4 kategórián kívül betanítottam még néhányat, hogy az osztályozónak legyen lehetősége olyan kategóriára döntenie, amelyik nem a 4 kiválasztott egyike. Az alábbi táblázatban összefoglalom, hogy melyik osztályból pontosan hány képet használtam, külön-külön a két halmazban, illetve az objektumok számosságát is feltüntetem a képeké mellett.

	Tanító		Validáló		Összes	
	Képek	Objektumok	Képek	Objektumok	Képek	Objektumok
Repülőgép	112	151	126	155	238	306
Busz	97	115	89	114	186	229
Autó	376	625	337	625	713	1250
Motorbicikli	120	167	125	172	245	339
Madár	180	243	150	243	330	486
Macska	163	186	174	190	337	376
Birka	48	130	48	127	96	257

4.2. táblázat: A mintahalmaz felépítése

A letöltött mintahalmaz képenként tartalmazza a megfelelő annotációkat. Ezek alapján betanítható a képosztályozó algoritmusom, illetve visszacsatolás segítségével ellenőrizhető, hogy a kereszt-validációban a validáló halmaz tesztelése milyen eredménnyel zárult.

4.2.2. KERESÉS TESZTELÉSE

A fejezet elején említettem, hogy fontos szempont volt a Bing és a Flickr által visszaadott találati lista, az egyes kereső kifejezésekre. Ez azért van így, mivel a kereséshez felhasznált képhalmazt ezekből a találati listákból állítottam elő. Amennyiben

értékelhetetlen lett volna az eredmény, az én kereső programomnak sem lett volna esélye a releváns képeket nem tartalmazó halmazból relevánsakat találni.

Mind a 7 keresésre visszakapott listából letöltöttem az első 50-et. Ezt megtettem mind a két internetes kereső esetén, így összesen 700 képből álló teszthalmazzal dolgoztam. Ezeket a képeket az osztályozó algoritmussal teszteltem, így láttam el azokkal a tartalmi információkkal, amelyek alapján képes voltam keresni köztük. Ahhoz, hogy ki tudjam értékelni az eredményeket, szükségszerű volt a képek manuális címkézése. Ez által eldönthető minden képről, hogy az releváns-e egy adott keresés szempontjából. Esetleges címkézési hibából nem alakulhatnak ki eredménybeli különbségek az én rendszerem és az összehasonlításra kiválasztott internetes keresők között, mivel ami náluk relevánsnak számít az nálam is, és fordítva. Egyenként végigjártam a képeket és néhány alapelvet felhasználva elkészítettem hozzájuk a megfelelő annotációkat, azaz az osztálycímkék halmazát. Az alapelvek a következők:

- Objektum felvétele, ha
    - nem túl kicsi a képen,
    - több mint 20%-a látszódik,
    - felismerhető.
- Egy adott objektum akkor szerepel relevánsan a képen, ha az kívülről látható, nem a belseje. A hibrid járművek nem tekinthetők ’jó’ objektumnak (például egy autó, amire szárnyakat illesztettek nem tekinthető repülnek és autónak sem).
- Minden kép, amin ezek alapján nem ismerhető fel a kiválasztott 4 osztály egyike sem, az egy vegyes osztályba kerül, további annotálás nélkül.

A felcímkézett állomány ezután készen áll a keresésre. Minden keresési kulcsra kapott találati lista első 50 elemét hagyom meg, a többi eldobom. Ezeket a rangsorokat elmentem, és kiértékelem. Emellett kiértékeltem a Bing és a Flickr találati listáinak első 50 eleméből álló listát is. A következő pontban bemutatom, hogy milyen mérőszámokat használok az összehasonlításra.

4.3. KIÉRTÉKELT MUTATÓK

A keresési térben található képek négyféle csoportba oszthatók az alapján, hogy szerepelnek-e egy adott keresés eredményeként visszaadott találati listában, illetve, hogy relevánsak-e az adott keresés szempontjából. Ennek a négy csoportnak a számosságát összefoglaló táblázatot konfúziós mátrixnak nevezzük (4.3. táblázat).

		Valóság	
		Releváns	Nem releváns
Jóslat	Releváns	TP	FP
	Nem releváns	FN	TN

4.3. táblázat: A konfúziós mátrix

A konfúziós mátrixban TP (True Positive) jelöli azoknak a képeknek a számát, amelyek bekerültek a találati listába, és valójában is relevánsak a keresés szempontjából. Az FP (False Positive) azoknak a képeknek a száma, amelyek bekerültek a találati listába, pedig valójában nem relevánsak. A TN (True Negative) azoknak a képeknek a száma, amelyeket nem adott vissza a kereső rendszer és nem is relevánsak. Az FN (False Negative) azoknak a képeknek a száma, amelyek nem kerültek be a találati listába, pedig relevánsak a keresés szempontjából. Ezek alapján többféle mutató is kiszámítható:

- Fedés (recall): $Rec = \frac{TP}{TP+FN}$
- Pontosság (precision): $Prec = \frac{TP}{TP+FP}$
- Average Precision (AP): $AP = \sum_{i=1}^{N} Prec(i) \times \Delta Rec(i)$
- Mean Average Precision (MAP): $MAP = \sum_{i=1}^{|C|} \frac{AP_i}{|C|}$

A fedés, pontosság és AP mutatók egy adott keresést minősítenek, mivel csak egy adott találati listából számítják a mérőszámot. A fedés azt adja meg, hogy a releváns képek mennyi százalékát adta vissza a rendszer. A pontosság pedig azt, hogy a rendszer válaszának mennyi százaléka releváns. Az AP mutató valamivel összetettebb, sokkal nagyobb hangsúlyt fektet arra, hogy a visszaadott releváns képek a találati listában hol helyezkednek el. Itt 'N' jelöli a találati lista méretét. Az elsőtől indulva mindegyik után

kiszámítjuk az aktuális pontosság értéket és a fedés megváltozását az előzőhöz képest. Ezek szorzatait összegezzük, ez adja az AP-t.

A MAP mutató nem egy konkrét keresést, hanem magát a kereső rendszert minősíti. Itt több keresés eredményéből számított AP mutatók átlagát képezzük. Megfelelő számú keresést vizsgálva jól jellemezhet egy adott rendszert. A fenti képletben a C jelöli a keresési kulcsok halmazát, amelyekre kapott válaszokat átlagoljuk.

A kísérletben mindegyik kereső kifejezésre külön kiszámítom az AP mutatót, majd ezek átlagából képezek egy MAP mutatót.

5. EREDMÉNYEK

5.1. EREDMÉNY, KERESÉSI KULCSONKÉNT

Mind a 7 keresési kulcs esetén lefuttattam az összes implementált módszert. A teljes eredményt táblázatban foglalom össze, ahol az összes módszer mellet a Bing és Flickr AP mutatóját is megadom, 5 különböző ponton a találati listán. Annak a módszeremnek az eredményét, amelyik a legjobb AP-t adta az 50. találat után, egy grafikonon ábrázolom a Bing és a Flickr mellett. Az ábrákon piros színnel jelölöm a Bing, kék színnel a Flickr és zöld színnel a saját eredményeimet.

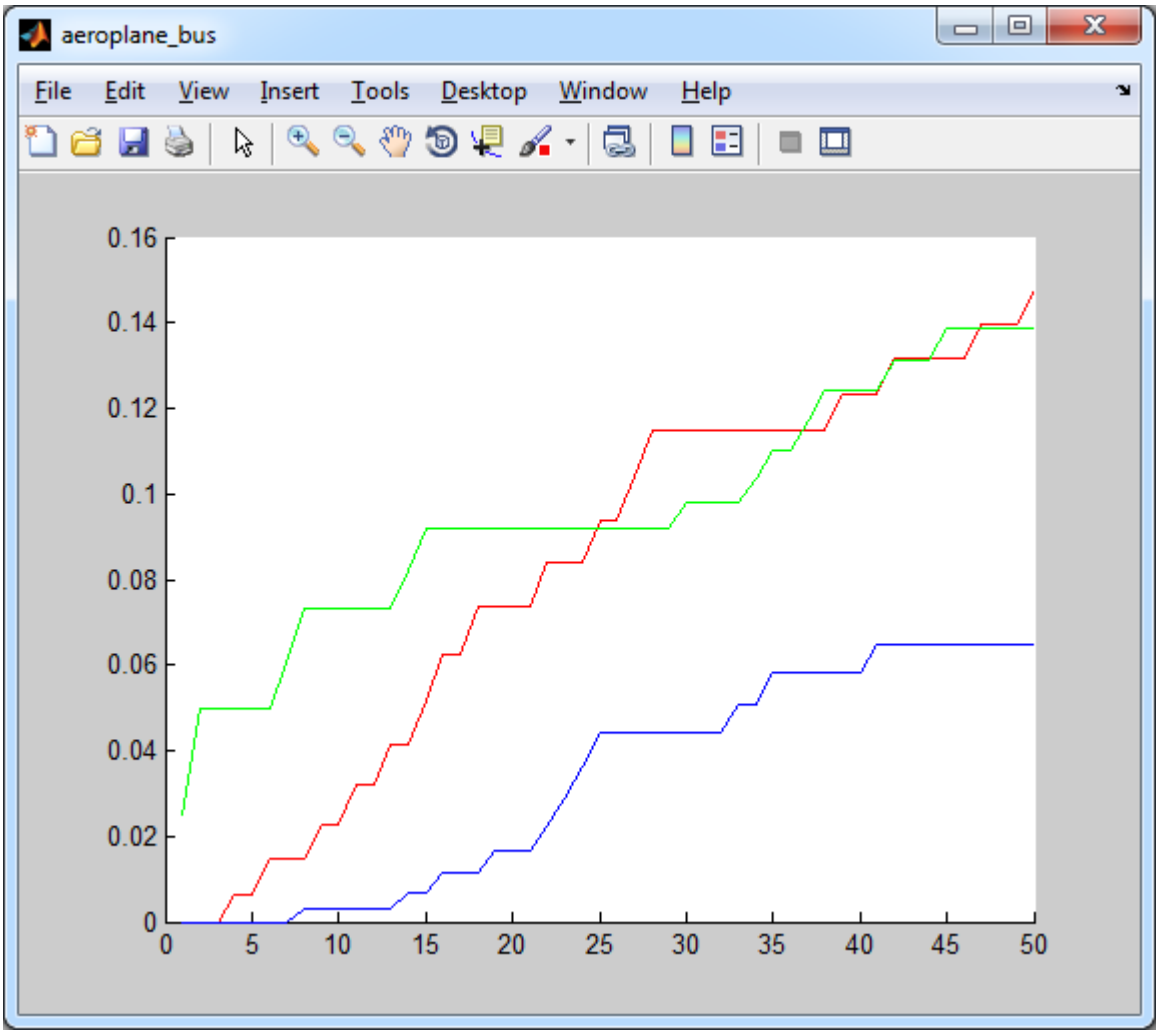
A módszerek közül az Ö2 esetén a küszöbértéket 0,5-re, az S2 esetén pedig 0,03-ra állítottam be. Ahogy később kitűnik az eredményekből: a szűréssel elért eredmények nem befolyásolják jelentős mértékben a pontosságot a szűrés nélkülihez képest, viszont a találati lista visszaadásának idejét jelentősen csökkenti.

5.1.1. REPÜLŐGÉP & BUSZ

	10	15	20	25	50
Bing	0,0229	0,0516	0,0737	0,0939	0,1474
Flickr	0,0031	0,0067	0,0166	0,0441	0,0648
Ö1	0	0	0	0	0,0085
Ö2	0	0	0	0	0,0085
S1	0,0367	0,0444	0,0506	0,0734	0,1106
S2	0,0367	0,0444	0,0506	0,0734	0,01106
M1	0,0732	0,0921	0,0921	0,0921	0,1386
T1	0,0500	0,0500	0,0500	0,0500	0,0571
K1	0	0	0	0	0,0058

5.1. táblázat: Repülőgép & busz eredményének AP értékei

A legjobb eredményt az M1 módszer, azaz a minimum függvény adta. Legrosszabbul az Ö1, Ö2, K1 módszerek teljesítettek. A 25. találatig egyetlen releváns képet sem sikerült visszaadniuk.



5.1. ábra: Repülőgép & busz esetén az AP értékek az M1, Bing és Flickr módszereknél

A fenti ábrán jól látszik, hogy az M1 az első 10-20 találatig magasan megelőzi a másik két keresőt. Fontos, hogy a visszaadott lista elején minél több releváns kép legyen.

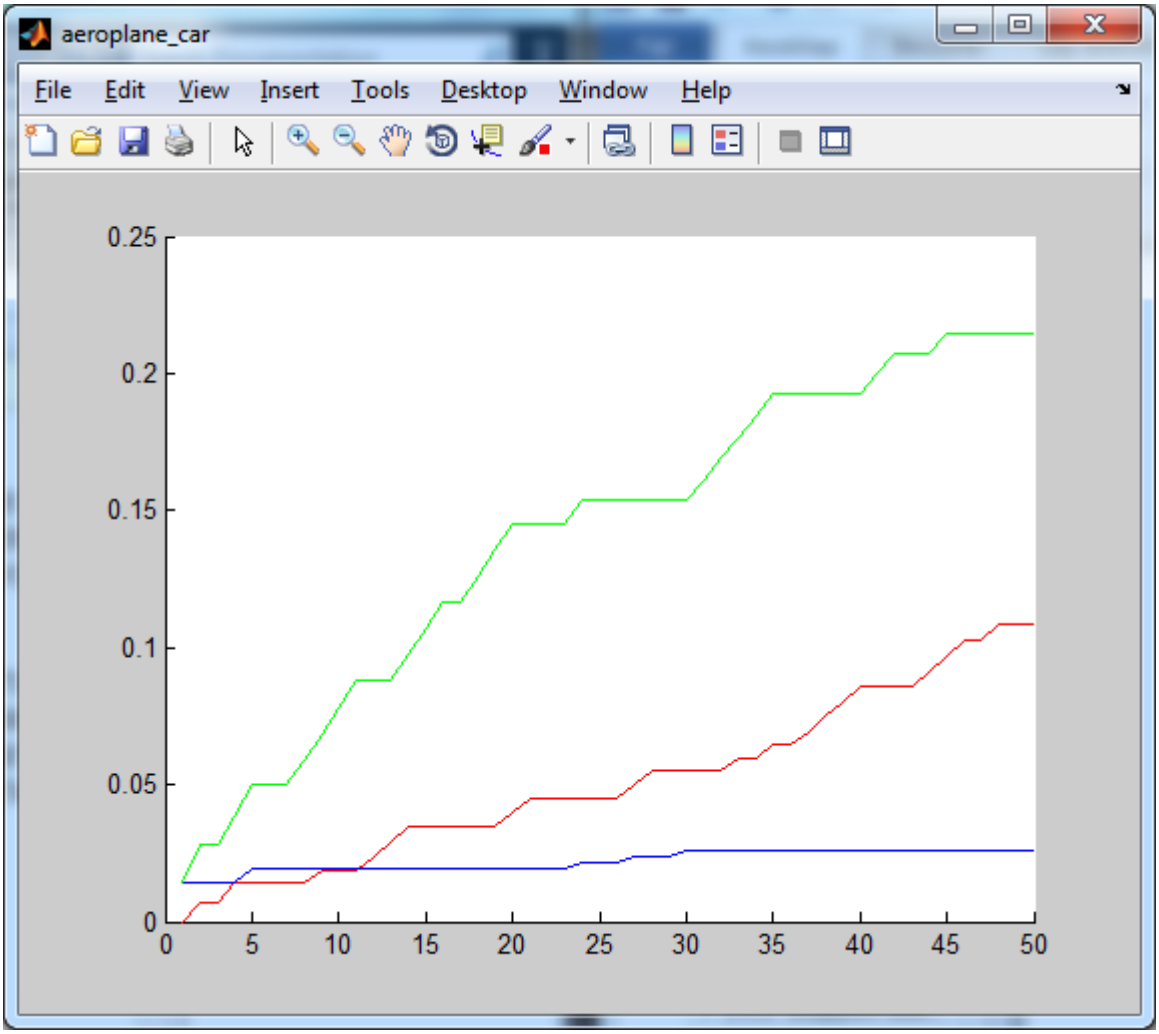
5.1.2. REPÜLŐGÉP & AUTÓ

	10	15	20	25	50
Bing	0,0188	0,0349	0,0399	0,0452	0,1084
Flickr	0,0197	0,0197	0,0197	0,0215	0,0259

Ö1	0,0571	0,0724	0,0790	0,0979	0,1363
Ö2	0,0571	0,0724	0,0790	0,0979	0,1363
S1	0,0810	0,0967	0,1206	0,1441	0,1969
S2	0,0810	0,0967	0,1206	0,1441	0,1969
M1	0,0781	0,1067	0,1453	0,1541	0,2143
T1	0,0419	0,0641	0,0704	0,0771	0,0985
K1	0	0,0031	0,0031	0,0031	0,0215

5.2. táblázat: Repülőgép & autó eredményének AP értékei

A legjobban teljesítő módszer ismét az M1, és ahogy azt az 5.2. ábra mutatja, magasan jobban teljesít, mint a Bing és a Flickr. Megjegyzem, az első 10 találatig az S2 jobb eredményt ad, viszont a 15. találat után már nem. A leggyengébb eredményt a K1 adta.



5.2. ábra: Repülőgép & autó esetén az AP értékek az M1, Bing és Flickr módszereknél

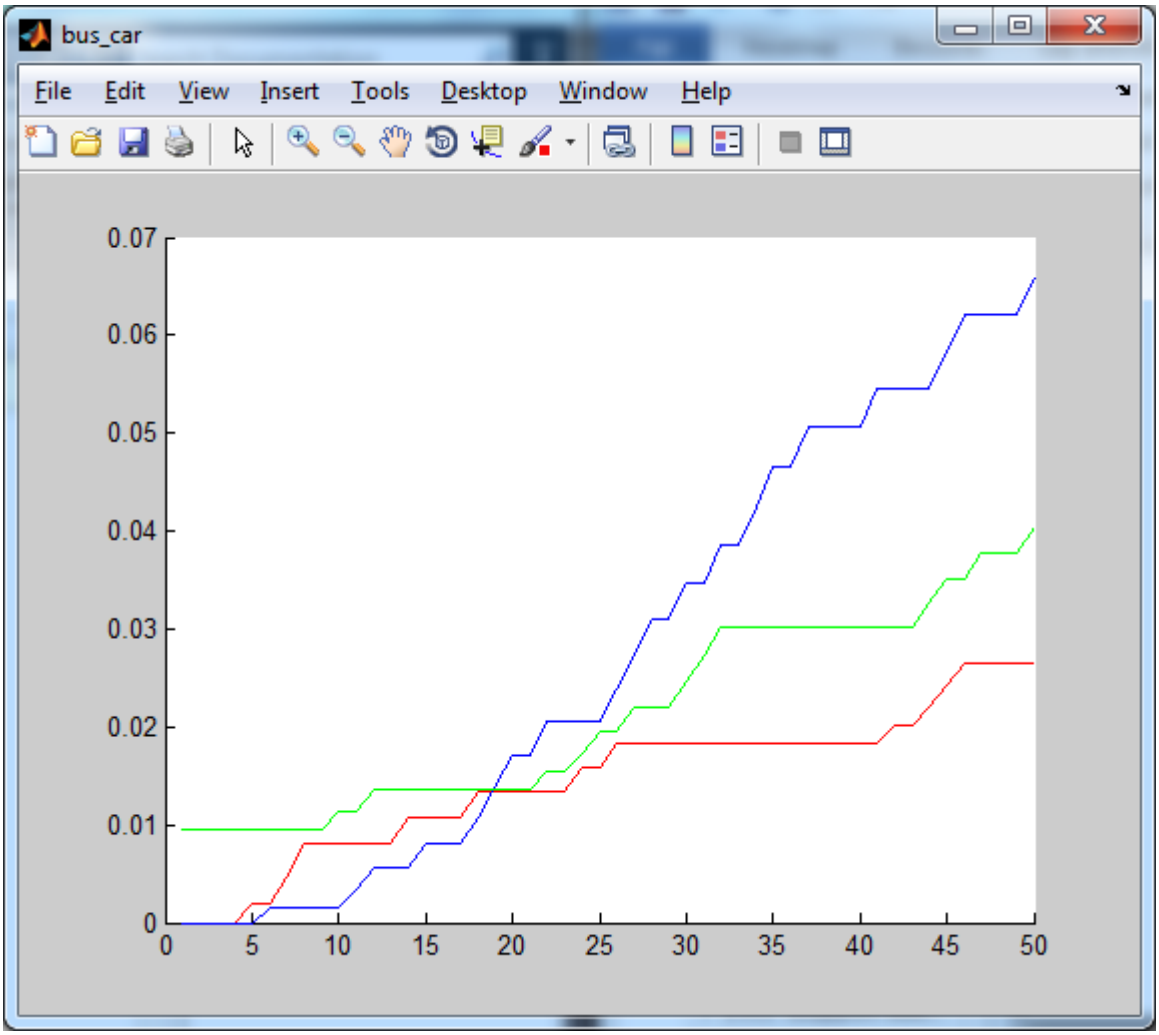
5.1.3. BUSZ & AUTÓ

	10	15	20	25	50
Bing	0,0081	0,0108	0,0134	0,0158	0,0264
Flickr	0,0016	0,0082	0,0171	0,205	0,0659
Ö1	0,0157	0,0186	0,0186	0,0186	0,0351
Ö2	0,0157	0,0186	0,0186	0,0186	0,0351
S1	0,0113	0,0137	0,0137	0,0196	0,0404
S2	0,0113	0,0137	0,0137	0,0196	0,0404

M1	0,0042	0,0088	0,0088	0,0110	0,0210
T1	0,0029	0,0029	0,0029	0,0056	0,0141
K1	0	0,0040	0,0040	0,0057	0,0176

5.3. táblázat: Busz & autó eredményének AP értékei

A legjobb eredményt az S1 és S2 módszerek adták, a leggyengébbet pedig a T1 és K1. Az 5.3. ábra csak egyetlen zöld görbét jelenít meg, mivel a két módszer eredménye egybe esik minden ponton. Az Ö1 és az Ö2 a 15. találatig több releváns képet ad vissza, mint bármelyik másik. Ez egy nem várt eredmény, hiszen az összeg módszer nem az 'ÉS, hanem a 'VAGY' kapcsolatot fejezi ki jobban.



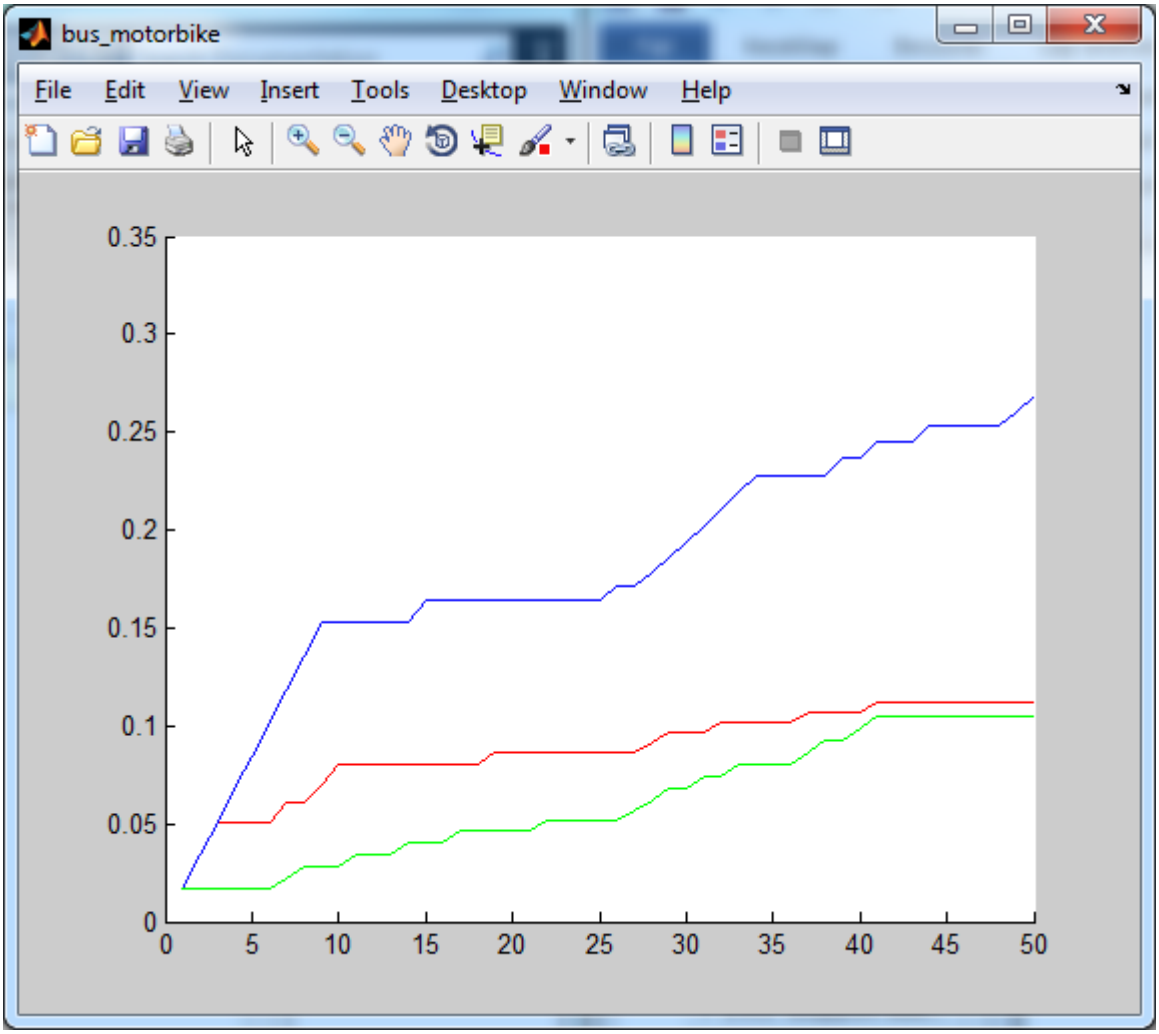
5.3. ábra: Busz & autó esetén az AP értékek az S1, S2, Bing és Flickr módszereknél

5.1.4. BUSZ & MOTORBICIKLI

Az egyetlen kereső kifejezés, amely esetén mind a Bing, mind pedig a Flickr jobban teljesít, mint az én kereső rendszerem. A találati lista minden pontján elmaradnak a módszerek által adott eredmények. Azt az eredményt, amely a legjobban megközelíti a Bing által adottat az M1 módszer adta.

	10	15	20	25	50
Bing	0,0801	0,0801	0,0864	0,0864	0,1118
Flickr	0,1525	0,1638	0,1638	0,1638	0,2682
Ö1	0,0019	0,0019	0,0019	0,0033	0,0033
Ö2	0,0019	0,0019	0,0019	0,0033	0,0033
S1	0,0282	0,0282	0,0282	0,0282	0,0296
S2	0,0282	0,0282	0,0282	0,0282	0,0296
M1	0,0281	0,0404	0,0463	0,0517	0,1054
T1	0	0,0011	0,0011	0,0011	0,0022
K1	0	0,0013	0,0013	0,0013	0,0026

5.4. táblázat: Busz & motorbicikli eredményének AP értékei



5.4. ábra: Busz & motorbicikli esetén az AP értékek az M1, Bing és Flickr módszereknél

Ahogy az 5.4. ábra mutatja, a Flickr az első 10 találatig folyamatosan releváns képet adott. Fontos megjegyezni, hogy az első 7 visszaadott kép ugyanaz volt, tehát egy olyan szigorítás bevezetésével, amely azt mondja ki, hogy csak különböző képeket adhat vissza a kereső, jelentősen visszaesne a Flickr által visszaadott releváns képek száma.

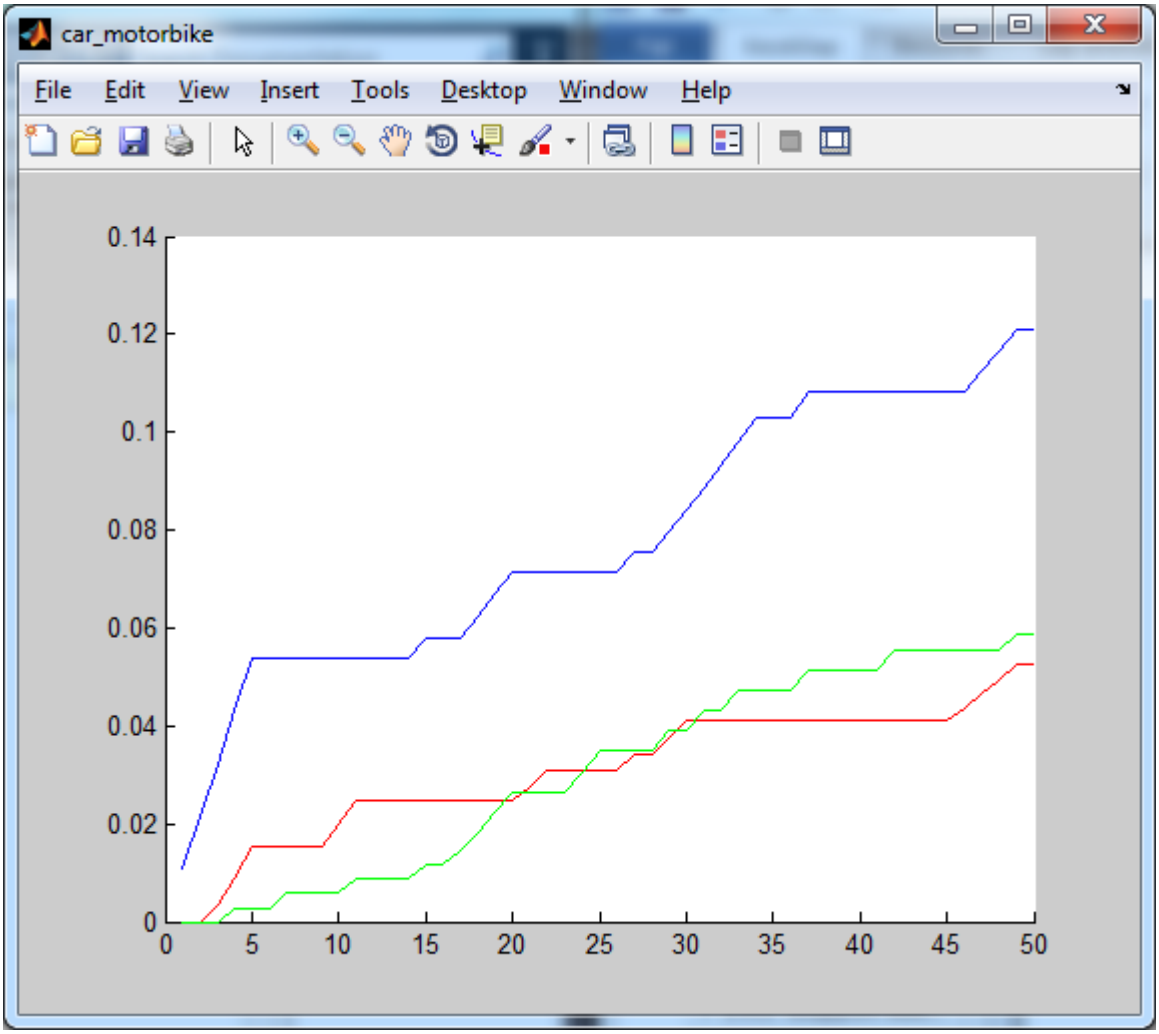
5.1.5. AUTÓ & MOTORBICIKLI

	10	15	20	25	50
Bing	0,0197	0,0246	0,0246	0,0311	0,0525
Flickr	0,0538	0,0581	0,0716	0,0716	0,1208

Ö1	0,0039	0,0039	0,0080	0,0106	0,0236
Ö2	0,0039	0,0039	0,0080	0,0106	0,0236
S1	0,0013	0,0124	0,0204	0,0325	0,0514
S2	0,0013	0,0124	0,0204	0,0325	0,0514
M1	0,0058	0,0116	0,0266	0,0339	0,0588
T1	0,0146	0,0191	0,0227	0,0325	0,0362
K1	0	0	0,0007	0,0007	0,0030

5.5. táblázat: Autó & motorbicikli eredményének AP értékei

Az autó & motorbicikli keresési kulcs esetén az 50. találat után sikerült a Bing teljesítményét felülmúlni. A Flickr viszont sokkal jobb eredményt adott, mint bármelyik módszerem. A legjobban a minimum függvény teljesített, a leggyengébben pedig a K1.



5.5. ábra: Autó & motorbicikli esetén az AP értékek az M1, Bing és Flickr módszereknél

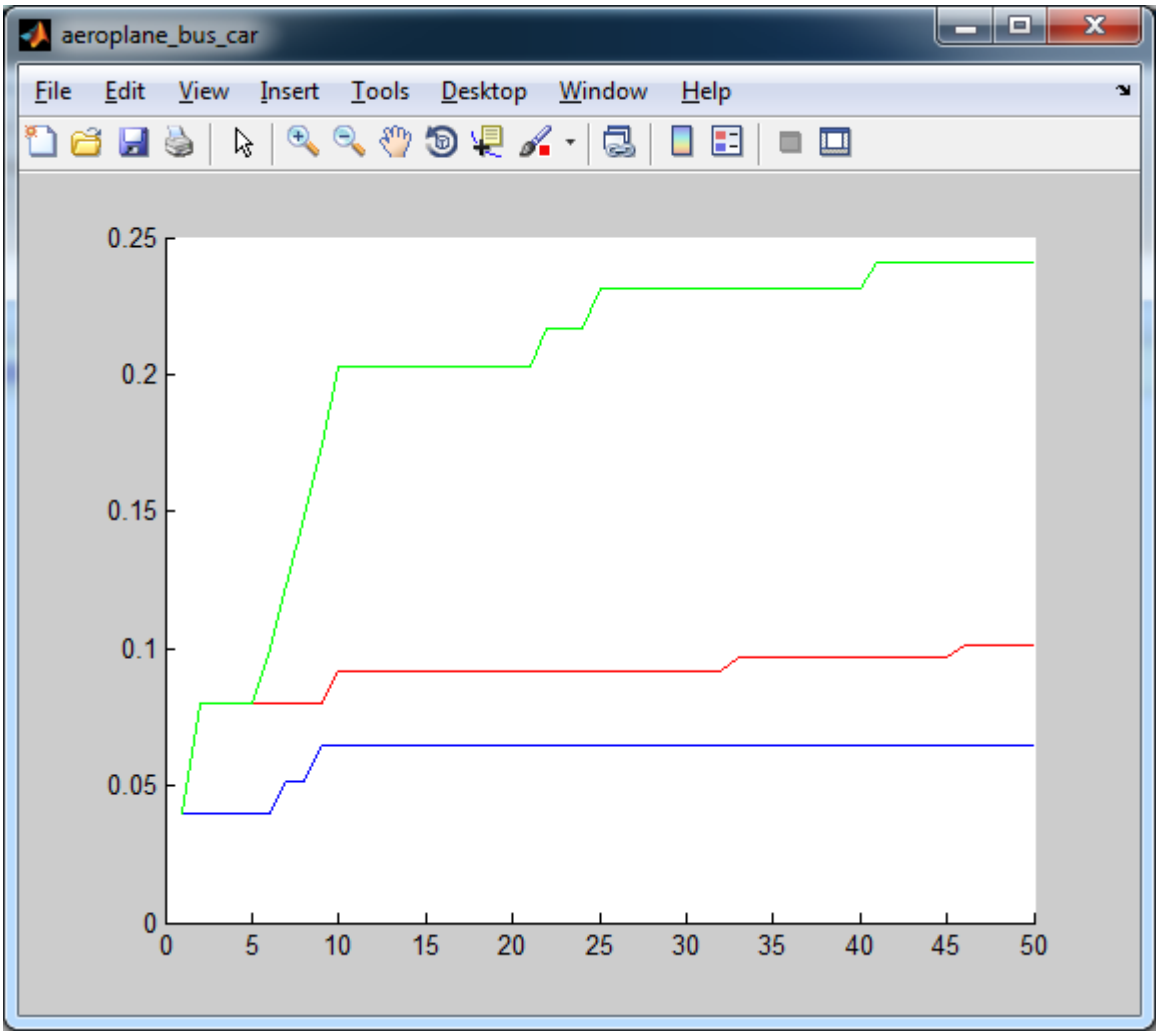
5.1.6. REPÜLŐGÉP & BUSZ & AUTÓ

	10	15	20	25	50
Bing	0,0920	0,0920	0,0920	0,0920	0,1012
Flickr	0,0648	0,0648	0,0648	0,0648	0,0648
Ö1	0,0800	0,1023	0,1274	0,1274	0,1472
Ö2	0,0800	0,1023	0,1274	0,1274	0,1472
S1	0,2025	0,2025	0,2025	0,2315	0,2412
S2	0,0100	0,0100	0,0144	0,0194	0,0335

M1	0,1229	0,1554	0,1554	0,1554	0,1745
T1	0,1100	0,1223	0,1223	0,1233	0,1346
K1	0	0	0	0	0,0107

5.6. táblázat: Repülőgép & busz & autó eredményének AP értékei

A három objektumból kombinált kereső kifejezésekre a Bing és a Flickr is nagyon gyenge eredményt ad. A vizsgált 50 találatból alig pár releváns, viszont az általam készített rendszer ennél jóval több releváns kép visszaadására képes. A legjobb eredményt az S1 módszer adta. Az 5.6. ábra mutatja, hogy mekkora a különbség a keresők közt.



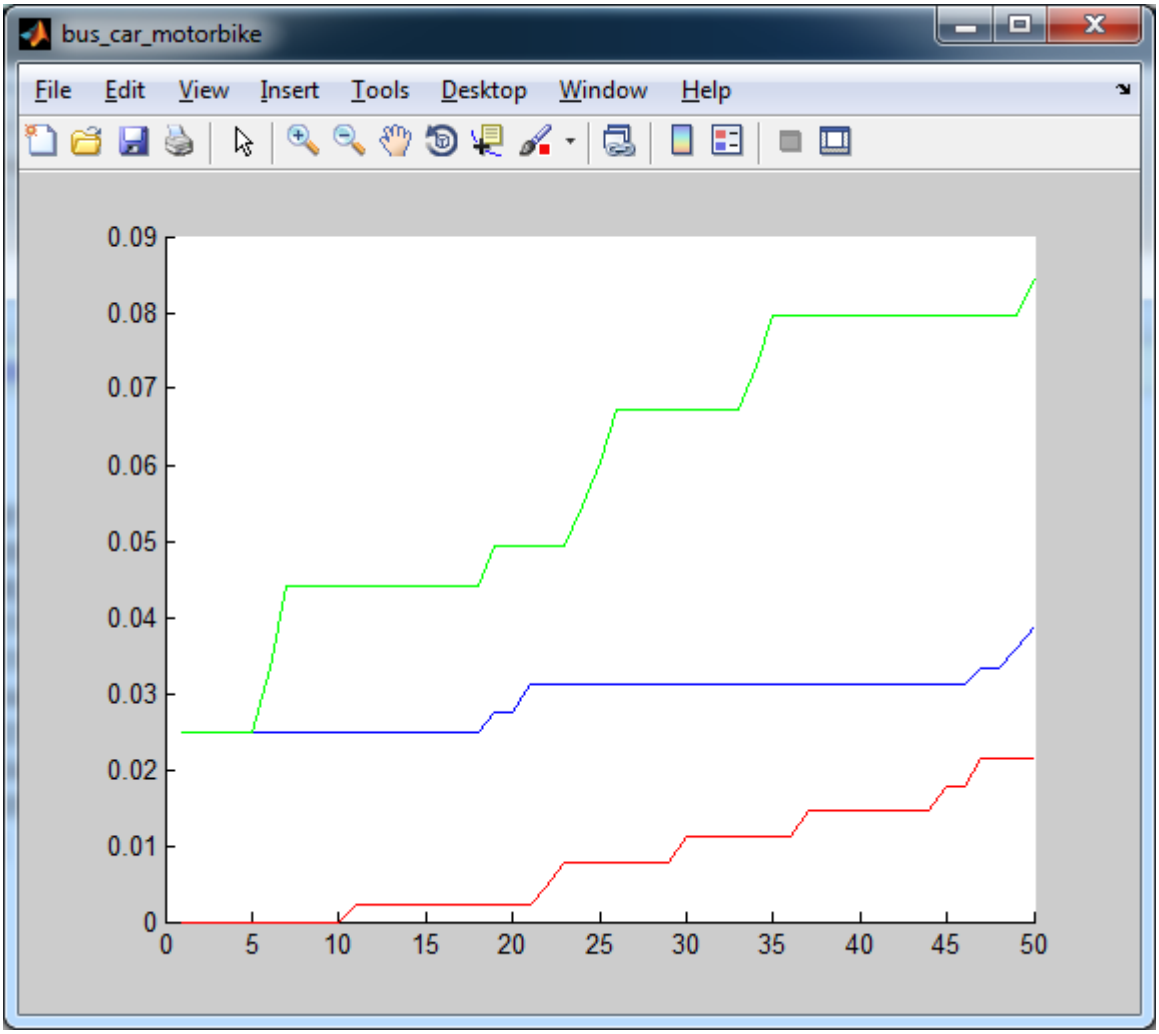
5.6. ábra: Repülőgép & busz & autó esetén az AP értékek az S1, Bing és Flickr módszereknél

5.1.7. BUSZ & AUTÓ & MOTORBICIKLI

	10	15	20	25	50
Bing	0	0,0023	0,0023	0,0078	0,0216
Flickr	0,0250	0,0250	0,0276	0,0312	0,0389
Ö1	0	0	0	0,0012	0,0054
Ö2	0	0	0	0,0012	0,0054
S1	0,0113	0,0113	0,0151	0,0151	0,0259
S2	0	0	0	0	0,0006
M1	0,0440	0,0440	0,0493	0,0605	0,0846
T1	0	0	0	0	0,0054
K1	0	0	0	0	0,0018

5.7. táblázat: Busz & autó & motorbicikli eredményének AP értékei

Ugyanaz igaz erre a keresési kulcsra is mint az előzőre. A Bing és a Flickr kevés releváns találatot ad, ennél viszont az én módszereim is gyengén teljesítenek. Több módszer is csak a 20. találat után adja vissza az első releváns képet. A legjobb eredményt az M1 adta, a többi módszerem AP értékei viszont alacsonyak.



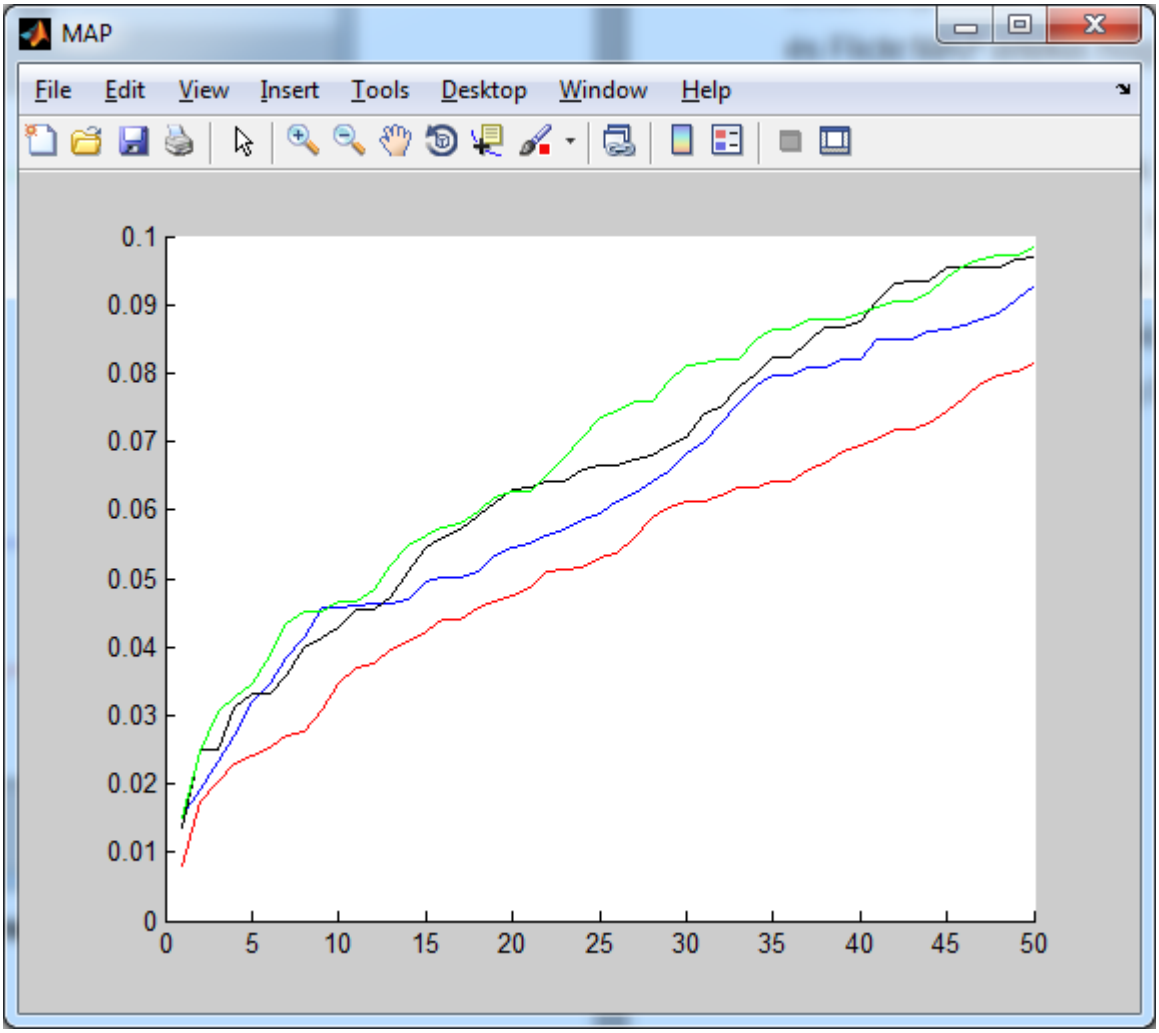
5.7. ábra: Busz & autó & motorbicikli esetén az AP értékek az M1, Bing és Flickr módszereknél

5.2. ÖSSZESÍTETT EREDMÉNY

A módszerek közül a legjobban az M1, azaz a minimum függvény teljesített. A 7 keresési kulcsból összesen 5-nél volt ez a legeredményesebb. Az S1 kétszer, az S2 pedig egyszer volt a leghatékonyabb. A legeredménytelenebbnek a K1 módszer bizonyult.

A fenti eredményekből kiszámítható egy MAP érték, amely megadja a keresők teljesítményét ezeknek az objektumoknak a keresésére nézve. Mindegyik módszer 50. találat után kapott AP mutatóiból kiszámítottam a MAP mutatókat. Ezek közül a legmagasabb az S1 módszeré, tehát átlagban az S1 teljesít a legjobban, annak ellenére,

hogy az M1 sokkal több esetben volt eredményesebb. Ennek oka, hogy az S1 mindig nagyon közel volt a minimum függvényhez, viszont mikor a szorzat volt az eredményesebb, akkor az M1 jóval elmaradt tőle. Az 5.8. ábra mutatja, hogy az S1, M1, Bing és Flickr MAP értékei hogyan alakulnak az 50 elemű találati listára nézve. Az ábrán pirossal jelöltem a Bing, kékkel a Flickr, feketével az M1, zölddel az S1 eredményeit.



5.8. ábra: A 7 kereső kifejezés átlagolásával kapott MAP mutatók

A fenti ábrán látszik, hogy az általam készített rendszer minden egyes találat után mérve tud jobban teljesítő megoldást nyújtani, mint a Bing vagy a Flickr keresők, ezekre a kereső kifejezésekre nézve. Az 50. találat után a Bing-nek 0,0813, a Flickr-mek 0,0928, az M1-nek 0,0969, az S1 módszernek pedig 0,0983 a MAP mutatója.

6. ÖSSZEGZÉS

Elkészítettem egy szemantikus képkereső rendszert, amely objektumok kombinálásával kapott keresési kulcsok esetén használható. Mivel a keresés tartalmi információk alapján történik, ezért létrehoztam egy osztályozó algoritmust, amely a keresési térbe tartozó képeket ellátja a kereséshez szükséges szemantikai információkkal. A kapott elemzésekből egy állandó adatbázist alkotok, amely új képek bekerülése esetén frissíthető. Kidolgoztam többféle módszert, amelyek mindegyike megvalósítja a keresést. Az ezek közti eltérés az, hogy milyen típusú keresési kulcs esetén használhatók eredményesen.

A dolgozatban megmutattam egy konkrét képkeresési kísérleten keresztül, hogy az általam készített rendszer eredményesen használható. Sikerült jobb eredményt elérnem, mint két ismert internetes kereső, a Bing és a Flickr. Abban az esetben, ha olyan tanítás lenne lehetséges, amely lefedi az interneten található objektumok halmazát, az eredmény még tovább javulna. Ezzel együtt a keresésre megadható objektumok száma is növekedne.

7. IRODALOMJEGYZÉK

[1] Slides from: S. Lazebnik, A. Torralba, L. Fei-Fei, D. Lowe, C. Szurka, Bag-of-Words models – Lecture 9, http://cs.nyu.edu/~fergus/teaching/vision_2012/9_BoW.pdf, pp. 1-32. (letöltés dátuma: 2013.10.04.)

[2] L. Fei-Fei, R. Fergus, and A. Torralba. "Recognizing and Learning Object Categories, ICCV 2009", *Part1 - Single object classes: Bag of Words models, Part-based models, and Discriminative models*, pp. 2-16.

[3] C. Harris, M. Stephens: *„A combined corner and edge detector”*, in Alvey Vision Conference, pp. 147–151, 1988, http://csce.uark.edu/~jgauch/library/Features/Harris.1988.pdf (letöltés dátuma: 2013.10.04.)

[4] Kató Zoltán, Képfeldolgozás és Számítógépes Grafika tanszék SZTE, *Sarokpontok detektálása*, http://www.inf.u-szeged.hu/~kato/teaching/DigitalisKepfeldolgozasTG/07-CornerDetection.pdf (letöltés dátuma: 2013.10.04.)

[5] T. Tuytelaars, K. Mikolajczyk: *„Local Invariant Feature Detectors: A Survey”*, Foundations and Trends in Computer Graphics and Vision, Vol. 3, No. 3 (2007) pp. 177–280, http://epubs.surrey.ac.uk/726872/1/Tuytelaars-FGV-2008.pdf (letöltés dátuma: 2013.10.04.)

[6] Mikolajczyk, K., Schmid, C.: *„Scale & affine invariant interest point detectors”*, International Journal on Computer Vision 60(1): pp. 63-86, 2004, http://www.cse.unr.edu/~bebis/CS773C/ObjectRecognition/Papers/Mikolajczyk04a.pdf (letöltés dátuma: 2013.10.04.)

[7] T. Lindeberg: *„Feature detection with automatic scale selection”*, International Journal of Computer Vision 30 (2): pp. 77-116, 1998, http://www.itu.dk/people/stud3154/Assignment2_1/Lindeberg98a.pdf (letöltés dátuma: 2013.10.04.)

[8] Lowe, D. G.: *„Distinctive Image Features from Scale-Invariant Keypoints”*, International Journal of Computer Vision, 60, 2, pp. 91-110, 2004, http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf (letöltés dátuma: 2013.10.06.)

[9] Móri Tamás, ELTE Valószínűségelméleti és Statisztikai Tanszék, 1999, Főkomponens- és faktoranalízis, http://www.cs.elte.hu/~mori/faktor.pdf (letöltés dátuma: 2013.10.06.)

[10] Münnich Á., Nagy Á., Abari K., Többváltozós statisztika pszichológushallgatók számára, Vol. 2, No. 3 (2006), http://psycho.unideb.hu/statisztika/pages/p_2_3.xml (letöltés dátuma: 2013.10.06.)

[11] Douglas Reynolds, MIT Lincoln Laboratory, Gaussian Mixture Models, http://www.ll.mit.edu/mission/communications/ist/publications/0802_Reynolds_Biometrics-GMM.pdf (letöltés dátuma: 2013.10.06.)

[12] Carlo Tomasi: *„Estimating Gaussian Mixture Densities with EM – A Tutorial”* http://www.cs.duke.edu/courses/spring04/cps196.1/handouts/EM/tomasiEM.pdf (letöltés dátuma: 2013.10.06.)

[13] Dempster, A., Laird, N., Rubin, D.: Maximum Likelihood from Incomplete Data via the EMAlgorithm. Journal of the Royal Statistical Society 39(1) (1977) 1–38, http://www.cnbc.cmu.edu/~tai/readings/classics/Dempster.pdf (letöltés dátuma: 2013.10.06.)

[14] Szegedi Tudományegyetem - Informatikai Tanszékcsoport, Mesterséges Intelligencia II – Felügyelet nélküli tanulás, http://www.inf.u-szeged.hu/~ormandi/ai2/03-k-means.pdf (letöltés dátuma: 2013.10.06.)

[15] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, pages 487.493, 1999.

[16] Perronnin, F., Dance, C.: „*Fisher Kernels on Visual Vocabularies for Image Categorization”* Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference

[17] Jorge Sánchez, Florent Perronnin, Thomas Mensink: *„Improved Fisher Vector for Large Scale Image Classification”*, COMPUTER VISION – ECCV 2010. Lecture Notes in Computer Science, 2010, Volume 6314/2010, pp. 143-156.

[18] Boser, B. - Guyon, I. - Vapnik, V.”A Training Algorithm for Optimal Margin Classifier” *Proc. of the 5th Annual ACM Workshop on Computational Learning Theory,* pp. 144-152. 1992.

[19] Corinna Cortes, Vladimir Vapnik: *„Support-vector networks”,* Machine Learning, Volume 20 (1995), Number 3, pp. 273-297, http://homepages.rpi.edu/~bennek/class/mmld/papers/svn.pdf (letöltés dátuma: 2013.10.11.)

[20] Altrichter Márta, Horváth Gábor, Pataki Béla, Strausz György, Takács Gábor, Valyon József: *„Neurális hálózatok”,* PANEM, 2006, http://mialmanach.mit.bme.hu/neuralis/ch06s03 (letöltés dátuma: 2013.10.11.)

[21] Vedaldi and B. Fulkerson, VLFeat: An Open and Portable Library of Computer Vision Algorithms, 2008. http://www.vlfeat.org/ (letöltés dátuma: 2013.06.22.)

[22] Chang, Chi-Chung and Lin, Chin-Jen; LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology, Vol(2), pp. 27:1-27:27, 2011. http://www.csie.ntu.edu.tw/~cjlin/libsvm/ (letöltés dátuma: 2013.06.22.)

[23] Ken Chatfield, Encoding Methods Evaluation Toolkit (Version 1.1), Visula Geometry Group, University of Oxford http://www.robots.ox.ac.uk/~vgg/software/enceval_toolkit/

[24] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A. : *“The PASCAL Visual Object Classes (VOC) Challenge”* International Journal of Computer Vision, 88(2), 303 338, 2010

[25] Everingham, M. and Van Gool, L. and Williams, C. K. I. and Winn, J. and Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results; http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html