

TDK-dolgozat

Szerző: Kovács Áron

Szótár és információkereső felület Android platformra, képfelismerési algoritmusok alkalmazásával

Szerző:

Kovács Áron, Babeş–Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, informatika szak, alapképzés, III. év

Témavezető:

dr. Csató Lehel egyetemi docens, Babeş–Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, Programozási Nyelvek és Módszerek Tanszék

Tartalomjegyzék

1. Bevezetés	3
2. A számítógépes írásfelismerés	4
2.1. Írásfelismerés általánosan	4
2.2. Felmerülő problémák	4
3. Az írásfelismerés lépései	5
3.1. Előfeldolgozás	5
3.1.1. Szürkeárnyaltos képpé való alakítás	5
3.1.2. Kontraszt állítása	5
3.1.3. Binarizálás (első kísérlet)	7
3.1.4. Binarizálás (második kísérlet)	8
3.1.5. Zajszűrés	9
3.2. Az írás felfedezése a képen	9
3.2.1. Konvolúció	10
3.2.2. Élfelismerés	11
3.2.3. Képtágítás	11
3.2.4. Binarizálás	12
3.2.5. Az írás irányának a meghatározása, kép forgatása	12
3.2.6. Sorok szegmentálása	14
3.2.7. Szavak, betűk szegmentálása	14
3.2.8. Normalizálás	15
3.2.9. A karakterek felismerése	15
4. A felismert szöveg felhasználása - előretekintés	16
4.1. Szótár és fordítóprogram	16
4.2. Kontakt információ mentés	16
4.3. Automatikus internetes keresés	16
4.4. Specializált internetes keresés	16
4.5. Extra információ szolgáltatása	16
4.6. Hangszintetizálás	17
5. Köszönetnyilvánítás	18
Hivatkozások	20

Kivonat

Munkám célja egy mobilalkalmazás fejlesztése, amely az okostelefonok beépített kameráját felhasználva OCR módszerek segítségével a fényképezett vagy filmezett írást felismeri és digitális formába hozza, majd ezt a szöveget felhasználva, az internet segítségével, további információkat szolgáltat a felhasználónak. A funkcionalitások közé tartozik egy online szótár, amely tetszőleges nyelvre lefordítja a felismert szöveget a Google Translate szolgáltatás révén, további fontosabb adatokat jeleníthet meg, illetve hanggá is alakíthatja ezt a felhasználó kérésére.

A mobil eszközök korlátozott kapacitásából adódóan rendkívül gyors és optimális algoritmusokra van szükségünk a valós idejű futás érdekében. A legnagyobb problémát a bemeneti kép előfeldolgozása jelenti, mivel a szöveg elkülönítése a háttértől rendkívül nehéz feladat és nem létezik egyértelmű, tökéletes megoldás. Kutatásom során az írásfelismerés fontosabb lépéseit tanulmányoztam, különböző lineáris szűrőket teszteltem és alkalmaztam a bemeneti képre, a szöveg felismerésére pedig egy megfelelően tanított mesterséges neurális háló bizonyult elfogadható és gyors megoldásnak.

1. Bevezetés

Az okostelefonok és főleg az Android [1] ingyenes operációs rendszer gyors terjedésével a hagyományos mobiltelefonokhoz és a notebookokhoz hasonlóan egy újabb eszköz vált luxuscikkből átlagember számára elérhetővé, státusszimbólumból munkaeszközzé. Az okostelefonok alapjaiban nem nyújtanak semmilyen újdonságot a régi eszközökhöz képest, csupán helyettesíti ezeket, egyetlen kompakt eszközbe integrál ezernyi funkcionális (többek között fényképezőgépet, GPS-t, jegyzetfüzetet, határidőnaplót, internethozzáférést stb), megkönnyítve és hatékonyabbá téve ezzel az ember életét.

Azáltal, hogy bárhol és bármikor a zsebünkben lehet az internet, egy új világ nyílik meg előttünk. Azonnal hozzáférhetünk bármilyen információhoz, így nem áll fenn a veszélye annak, hogy otthon elfelejtünk utánaolvasni fontos dolgoknak.

Az okostelefonok megjelenésével a számítógépes írás- és képfelismerés is új horizontra jutott, ezt bizonyítja a következő két alkalmazás: Google Goggles [2], Word Lens [7]. A beépített kamera segítségével névjegykártyák, telefonszámok azonnal digitalizálhatóak, a vonalkódok felismerése által a vásárlás és döntések is leegyszerűsödhetnek, mitöbb, a már létező szövegfelolvasó rendszerek és az írásfelismerés mobil alkalmazásának együttműködése a vakoknak is segédeszközt nyújthat.

Egy másik úttörő alkalmazása a mobil írásfelismerésnek egy szótár/fordító létrehozása lenne, ezáltal a nyelvkorlátok részben megszűnnének a turisták körében. Ugyanakkor az is rendkívül kényelmes lenne, ha egy egyszerű, helyben készített fényképből azonnal ki lehetne nyerni a szöveget és automatikusan rákeresni interneten, megtakarítva a kézzel való begépelést.

Kutatási munkám az írásfelismerésnek ezen alkalmazását célozza meg, fontosabb lépéseit és nehézségeit tanulmányozva. A továbbiakban a folyamat lesz részletesebben bemutatva.

2. A számítógépes írásfelismerés

2.1. Írásfelismerés általánosan

Az optikai karakterfelismerés (angolul: Optical Character Recognition - OCR) célja a gépelt, nyomtatott, vagy kézzel írt szöveg számítógép által értelmezhető és módosítható digitális szöveges állománnyá való alakítása. A feladat a gépi tanulás ([8]), mintafelismerés ([9], [13]), illetve számítógépes látás ([22]) kutatási körébe tartozik. Sok területen vannak hasznos alkalmazásai. A nyomtatott, latin abc-s dokumentumok felismerését 99%-ban megoldottnak tekinthetjük, viszont a hétköznapi fényképeken előforduló írás, illetve a kézirás felismerése jóval nehezebb feladat, és mai napig is az aktív kutatás részét képezi.

Az OCR-nek alapvetően két formája van: az on-line, illetve az off-line írásfelismerés. Az előbbi az írás közben történő kézmozgásra alapoz (ezt a technológiát alkalmazzák az okos táblák/telefonok esetében is), így nem alkalmazhatóak fényképekre. Az off-line felismerés ezzel szemben statikus formák feldolgozására alkalmas. A nagyobb pontosság érdekében felhasználhatjuk a kontextuális és nyelvtani többletinformációt is.

Jelen dolgozat a hétköznapi fényképeken található írásra összpontosít.

2.2. Felmerülő problémák

A hétköznapi helyszíneken készített fényképeken található írás felismerése sokkal bonyolultabb, mint a tiszta, szkennelt dokumentumok vizsgálata. A környezeti tényezők, fényviszonyok és a fényképezőgép/kamera technikai korlátai, beállításai rendkívül befolyásolják a kép minőségét, így a szöveget komplexebb, nem egyenletes háttértől és zavaró elemektől kell elkülöníteni. Egy másik gyakran előforduló probléma a képzaj, mely a további képfeldolgozás eredményességét csökkenti. Ugyanakkor a kézi fényképezés legfőbb hátránya a potenciális képelmosódás és homályosság, amelyet szintén korrigálni kell az eredményes írásfelismerés érdekében. Szintén előfordul, hogy a fényképen található írás ferde, vagy torzított, aszimmetrikus.

Láthatjuk, hogy nagyobb hangsúlyt kell fektetni a kép előfeldolgozására, mint a tiszta háttérrel rendelkező, kontrasztos, szkennelt dokumentumok esetében.

A rossz képminőségből adódóan egy-két karakter összemosódhat, előfordulhatnak nagyon hasonlóak is, melyek felismerésénél a neurális háló tévedhet. Ezeket a hibákat valamilyen kontextuális vagy nyelvtani többletinformáció felhasználásával javíthatjuk ki, egy megoldást a Markov lánc ([11]) jelentheti vagy online módon a Google AutoCorrect elemzőjét felhasználva.

3. Az írásfelismerés lépései

Alapvetően a számítógépes írásfelismerés folyamatát két nagyobb műveletre oszthatjuk: az előfeldolgozás és a konkrét felismerés.

Az előfeldolgozás [21] a bemeneti képre alkalmazott műveletek, transzformációk, szűrők sorozatát jelenti, amely az írás pontosabb felismeréséhez vezet. Ennek érdekében a karakterek a lehető legolvashatóbbak kell legyenek, optimális kontraszttal, élességgel, illetve színskálával kell rendelkezzenek. A végső cél, tehát, hogy a felismerő egység egy olyan bemeneti képet kapjon, amelyen az írás tökéletesen elkülönül a háttértől. Megjegyzendő, hogy nem létezik tökéletes, általánosan elfogadott megoldás, csupán javaslatok, melyek hatékonysága a különböző módszerek megfelelő kombinálásában rejlik.

A következőkben az allépésekről lesz szó.

3.1. Előfeldolgozás

3.1.1. Szürkeárnyaltos képpé való alakítás

A legelső lépés a folyamatban, hogy az eredeti, színes képet szürkeárnyaltossá alakítsuk, kiküszöbölve a színes képekben rejlő színárnyalatra és telítettségére vonatkozó többletinformációt, csupán a luminanciát őrizve meg, mely elégséges az írásfelismeréshez, harmadolva a memória- és processzorigényt.

A szürkeárnyaltos képek egyetlen mátrixból állnak, minden pixelnek megfelelően egy intenzitást a $0 - 255$, vagyis a fekete-fehér skálán. Színes képből az átalakítás a képpontok piros (R), zöld (G), illetve kék (B) színkomponensei súlyozott összegének kiszámítása révén történik, a következő képlet szerint:

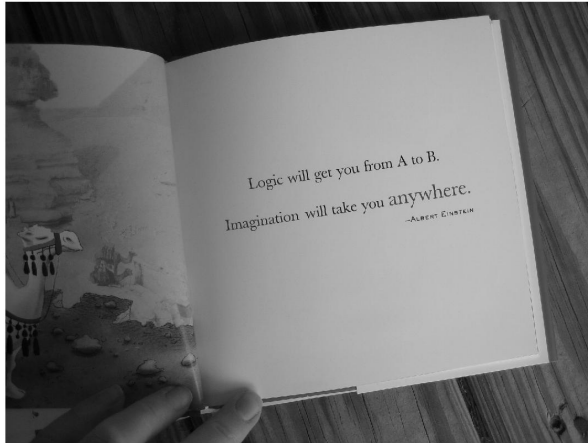
$$0.2989 * R + 0.5870 * G + 0.1140 * B \quad (1)$$

Ez az érték lesz az új, szürkeárnyaltos képpont intenzitása. A fenti skalárok értékeit statisztikailag állapították meg, általánosan elfogadottak, az emberi szem érzékenységevel arányosak a három színre nézve [23].

3.1.2. Kontraszt állítása

A bemeneti kép kontrasztja sok tényezőtől függhet, kezdve a fényképezőgép minőségétől azok beállításain keresztül az adott körülményekig, illetve fényviszonyokig. Ez az információvesztés szerencsére kijavítható valamilyen mértékben bizonyos technikákkal.

A szürkeárnyalat-transzformáció nem függ az adott pixel helyzetétől. Egy T transzformációt, mely az eredeti kép p fényességét a $[p_0, p_k]$ skálából átviszi egy új, $[q_0, q_k]$ skálába, a következő-



1. ábra. Szürkeárnyaltos kép

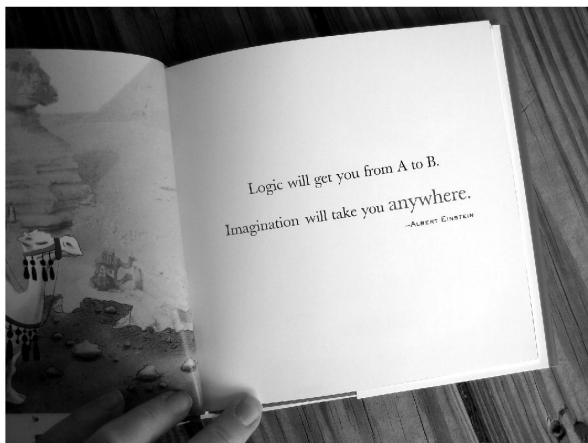
képpen definiálhatunk:

$$q = T(p) \quad (2)$$

Ezáltal változtatható a p_0 és p_k fényesség értékek közti kontrasztot.

Egyik megoldást a hisztogramkiegyenlítés [21] módszere jelentheti, mely segítségével egyenletes eloszlású fényességszinteket érhetünk el a teljes fényességskálán. Ez növeli ugyan a kontrasztot, azonban a képen található írást nem különíti el megfelelőképpen a háttértől, az egyenletesség miatt.

Egy másik, kevésbé bonyolult módszer ebben a fázisban megfelelőbb megoldást biztosít. Az úgynevezett kontrasztkiterjesztés [21] egy lineáris módszer, amely „kinyújtja” a képen található intenzitásértékeket a teljes intenzitásskálára, megnövelve a kontrasztot.



2. ábra. Kontrasztkiterjesztés

3.1.3. Binarizálás (első kísérlet)

A következő lépés a kép binarizálása [15] lenne, mely egy egyszerű pixelalapú osztályozás. A kimeneti képben minden pixel, amelynek a fényessége meghalad egy bizonyos megadott küszöböt 1-el (fehérrel) helyettesít, míg a többit 0-val (feketével). Ez a szigorú osztályozás a szöveg szegmentálását és a karakterek felismerését könnyíti meg, mivel csak a potenciálisan írást tartalmazó részeket kell ezáltal elemezniük. Rendkívül fontos ezt a küszöbértéket megfelelően meghatározni, ezen múlik a háttér elkülönítése a szövegtől.

A hisztogramkiegyenlítés vagy a kontrasztkiterjesztés módszere elégséges is lenne binarizáláshoz, feltéve ha a bemeneti kép nem zajos, illetve nincsenek a szövegen kívül más komponensek az egyenletes háttéren. Ez csupán jó minőségű, szkennelt dokumentumokra teljesül, komplexebb módszerekre lesz szükség.

Két nagy alfaja van a küszöbértéket használó algoritmusoknak. A globális küszöbérték a teljes kép intenzitásértékeit veszi figyelembe, a lokális (adaptív) pedig kisebb régiók lokális tulajdonságai alapján határozza meg az adott régióknak a legmegfelelőbb küszöbértéket.

- **Otsu algoritmus**

Otsu algoritmus [19] egy optimális globális küszöbértéket próbál meghatározni, felhasználva a képhisztogram formáját. Feltételezi, hogy két típusú pixelosztályt tartalmaz a kép (pl. háttér és előtér), ezután kiszámít egy küszöboptimumot, mely e két osztályt szétválasztja, úgy hogy a két osztály osztályon belüli szórásnégyzete minimális legyen. Az algoritmus *Nobuyuki Otsu*-ról kapta nevét. Az osztályon belüli szórásnégyzetet a két osztály szórásnégyzetének súlyozott összege adja meg:

$$\sigma_{\omega}(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t) \quad (3)$$

Az ω_i súlyok a t küszöbérték által elválasztott két osztály valószínűségei, a σ_i^2 pedig a hozzá tartozó szórásnégyzet.

Az osztályon belüli szórásnégyzet minimalizálása egyenértékű az osztályon kívüli szórásnégyzet maximalizálásával.

$$\sigma_b(t) = \sigma^2 - \sigma_{\omega}^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2 \quad (4)$$

Az ω_i szintén az osztályok valószínűségei, μ_i az adott osztály várható értéke, melyet iteratív módon lehet módosítani, így ez a módszer hatékonyabb algoritmushoz vezet.

- **Niblack algoritmus**

Niblack algoritmus [14] lokálisan próbálja meghatározni a küszöbértéket a lokális várható érték és lokális standard szórás kiszámítása alapján. A küszöbértéket a következő képlet adja:

$$T(x, y) = m(x, y) + ks(x, y) \quad (5)$$

ahol az $m(x, y)$ a lokális átlag, $s(x, y)$ a szórás és k súly.

A lokális régió mérete megfelelően kicsi kell legyen, hogy a lokális tulajdonságokat megőrizzék, ugyanakkor megfelelően nagyok, hogy a zajt eltompítsák.

Zhang és Tan egy javított változatot is javasolt Niblack algoritmusára:

$$T(x, y) = m(x, y) \left[1 + k \left(1 - \frac{s(x, y)}{R} \right) \right] \quad (6)$$

ahol a k és R konstansok a zajra való érzékenységet szabályozzák.

3.1.4. Binarizálás (második kísérlet)

Otsu algoritmus jól meghatározza a küszöbértéket, feltéve, ha viszonylag kontrasztos a kép és nincsenek más, a szövegnek alacsonyabb intenzitású egyéb komponensek a képen. Ez esetben a szöveg háttérnek lesz osztályozva.



3. ábra. Az első képen látható az eredeti kép, a másodikon az Otsu módszer eredménye, a harmadikonon pedig ugyanarra a képre először más műveleteket is alkalmaztunk. Látható, hogy a táblán az írás teljesen elveszik az Otsu alkalmazása során.

Niblack algoritmus se a legmegfelelőbb számunkra, mivel a kimenet nagyon függ a k konstánstól, nem tesz különbséget a szöveg és nem szöveg alapú komponensek között, lokálisan a kisebb kontrasztkülönbségeket is kihangsúlyozza.

Emiatt a tényleges binarizálásra még nem kerülhet sor, előbb ki kell szűrni a lehető legtöbb nem szöveg alapú komponens. Valamilyen módszert kell találni, amely a szöveg alapú komponenseket különíti el.

3.1.5. Zajszűrés

A képzaj [21] véletlenszerű szín- vagy intenzitásingadozás egy adott képponton, ezt a képrögzítő eszköz szenzorának érzékenységhiánya okozza. A zajcsökkentés vagy zajszűrés az a módszer, amely ezt a képzajt próbálja csökkenteni, anélkül, hogy fontos információ veszne a képből.

A különböző algoritmusok nagyon hasonlóak, azonban több tényezőt kell mérlegelni: a rendelkezésre álló processzor kapacitását, időt, apró részletek fontosságát a képen stb.

Lineáris filterek alkalmazásával csökkenthető a képzaj [12], mint például az átlagoló filter vagy pedig a Gauss filter. Az átlagoló filter esetében minden egyes pixel értéke a környező pixelek átlagát veszi fel, ezáltal a lokális nagy variációk (zaj) kisimulnak. Ez a képélesség rovására mehet.

Az adaptív filterek, mint például a Wiener filter [17] (Matlab dokumentáció: [6]), a képeken az intenzitás lokális változásait veszik figyelembe, ahol nagy a különbség, ott kevésbé simít a képen, ahol pedig kicsi, ott erőteljesebb a filter hatása. A Wiener filter ezáltal jobban teljesít, mint a hagyományos lineáris filterek, jobban megőrzi az éleket és a magas frekvenciájú komponenseket, azonban magasabb a processzorigénye.

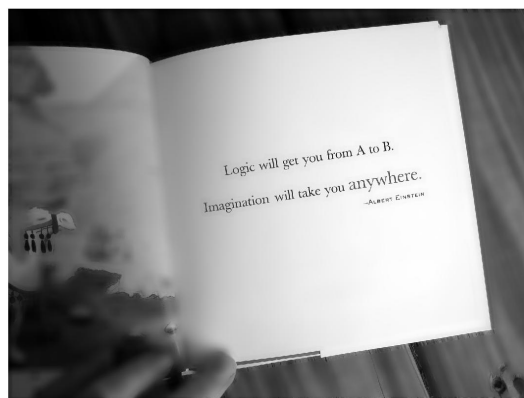
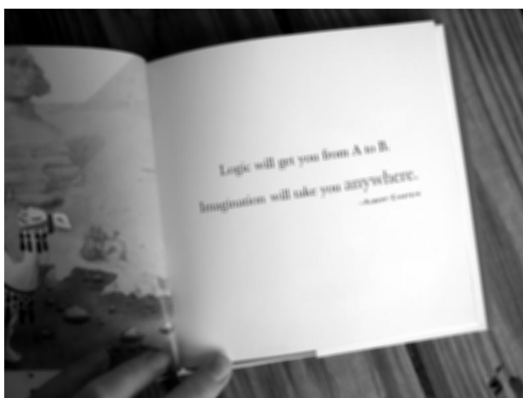
A medián filterezés [18] hasonlít az átlagoló lineáris filterhez, azonban nem a szomszédos pixelek értékeinek a várható értékére (átlagára) hagyatkozik, hanem a pixelek mediánjára, amely a statisztikában egy nevezetes középérték. A medián kevésbé érzékeny az extrém értékekre, a várható értékkel ellentétben. A medián filterezés, tehát, sokkal jobb munkát végez a zajszűrésben a képélesség rontása nélkül, ugyanakkor megőrzi a komponensek közti éleket. Ez a tulajdonsága nagyon előnyösnek bizonyul a további képfeldolgozásban az éldetektáláskor. Ez a típusú filter alkalmazása bizonyult a Wiener filter mellett hatékonyabbnak az előfeldolgozás során.

3.2. Az írás felfedezése a képen

Alapvetően három módszercsoportot különíthetünk el [16] [12].

A színalapú szövegelfedezés színkvantálást használva megpróbálja megtalálni az összekötött komponenseket, és agresszív heurisztikus filterezéssel próbálja meghatározni a szöveget tartalmazó részeket.

Egy másik módszer a szöveg felfedezését textúra alapon közelíti meg. Matematikai transzformációkat és gépi tanulási módszereket alkalmazva íráshoz hasonló mintákat keres a képen. A



4. ábra. Az első képen látható az átlagoló filter eredménye a bemeneti képre, a másodikon pedig a Wiener filter. Látható, hogy a Wiener megőrzi a fontos éleket

legnagyobb akadályt a számítási komplexitás jelenti, illetve a nem egyértelműen meghatározható minta, mely a szöveg típusú komponenseket felismerné.

A harmadik módszer az élkeresés. Szomszédos pixelek közti magas kontrasztkülönbségeket keres, bináris képet eredményez, melyen az elkülönülő komponensek élei lesznek fehér színűek fekete alapon. Empirikusan ez a módszer tűnt a leghatékonyabbnak az írásfelismerés folyamatában.

3.2.1. Konvolúció

A lineáris módszereket, illetve vektortereket előszeretettel használják a képfeldolgozásban, a képi függvények definiálásában. A konvolúciónak, mint matematikai függvénynek leginkább a jelfeldolgozásban vannak alkalmazásai.

A konvolúció [22] rendkívül fontos művelet a lineáris képfeldolgozásban. Alapvetően két függvényre alkalmazható (f, h) , egy harmadikat hozva létre, mely az egyik előző függvény módosítottja. A 7 képlettel fejezhető ki, amely az $f(t)$ függvény átfedésének mértékét fejezi ki, ahogyan ezt egy $h(t)$ függvény fölött eltoljuk. Az egydimenziós konvolúció képlete:

$$(f * h)(t) = \int_{-\infty}^{\infty} f(t)h(t - \tau) d\tau. \quad (7)$$

A konvolúció fogalmát több dimenzióra is kiterjeszthetjük, képek feldolgozásában a két dimenziós konvolúció használatos:

$$(f * h)(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(X, Y)h(x - X, y - Y)dXdY. \quad (8)$$

A digitális képelemzésben a diszkrét konvolúció használatos, így az integrál helyett összegeket alkalmazunk, mivel a képnek korlátozott az értelmezési tartománya. A konvolúció ebben az esetben egy lineáris szűrőfolyamatot (filtert) jelent, ahol h a szűrő. A kimeneti képen egy adott

$g(i, j)$ pixel esetében az adott bemeneti kép $f(i, j)$ pixel közvetlen szomszédságában lévő pixel intenzitásának valamilyen lineáris kombinációját fejezi ki. A különböző pixel intenzitásának hozzájárulását a h tényező (konvolúciós maszk) súlyozza:

$$f(i, j) = \sum_{(m,n) \in O} h(i - m, j - n)g(m, n), \quad \text{ahol } O \text{ a szomszédos tartomány.} \quad (9)$$

Általában négyzetes szomszédságot határozunk meg, páratlan számú sorral és oszloppal, így egyértelműen meghatározható a középen található pixel, amelyre alkalmazzuk a konvolúciót.

3.2.2. Élfelismerés

A számítógépes látásban és képfeldolgozásban alapvető eszközt jelent az élfelismerés [22], különösen a komponensek elkülönítésében, határok és releváns tulajdonságok megállapításában. Ugyanakkor jelentősen lecsökkenti a feldolgozásra szánt adatmennyiséget, és kiszűri a nem releváns részteket is.

Több algoritmus létezik, a leghíresebbek a konvolúcióra alapuló Sobel, Prewitt, Roberts, Canny által bevezetett módszerek [21]. A Sobel és Canny algoritmusát vizsgáltam meg részletesebben.

A Sobel operátor minden pontban a képen kiszámolja az intenzitás gradiensét, visszaadva a legnagyobb intenzitásnövekedés irányát, így könnyen megállapítható, hogy az adott részen mekkora valószínűséggel található él és milyen irányban.

A Canny módszere lokális maximumokat keres a gradiensben, két küszöbértéket használ az erős és gyengébb élek felismerésére, a gyengébb élek csak akkor kerülnek a végeredménybe, ha erős élekkel vannak összekötve. Ezáltal kevésbé érzékeny a zajra és nagyobb valószínűséggel ismeri fel a halványabb éleket. Általános célokra a Canny módszere a legalkalmasabb, sajátos esetekben viszont a Sobel módszere jobban teljesíthet.

Az élfelismerés által a szövegrész jól elkülönül, mivel az előző kontraszt- és zajszűrő technikával a kép minősége és olvashatósága javult. Az íráson kívül megjelennek egyéb vonalak, élek is a háttérobjektumokból, ezeket viszont egyszerűbb és gyorsabb lesz ezen a bináris képen kiszűrni különböző heurisztikus módszerekkel, mint az eredetin a teljes objektumokat.

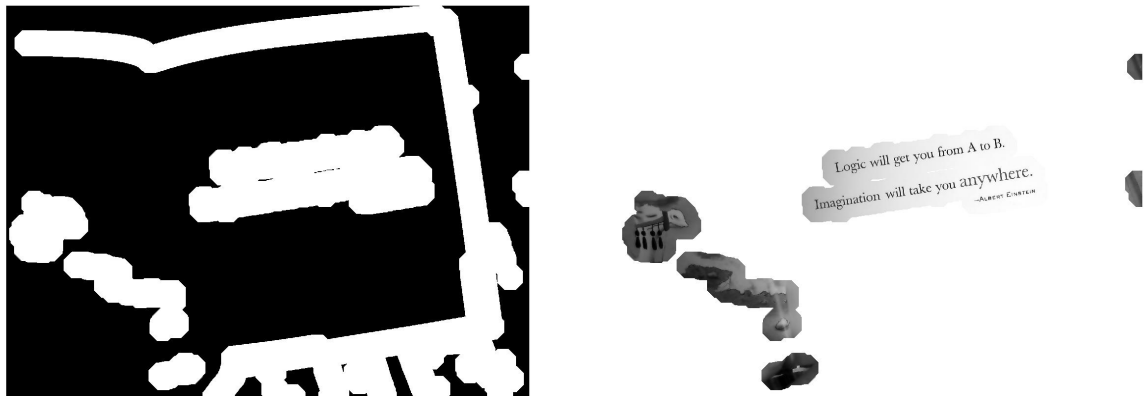
3.2.3. Képtágítás

Az éldetektálás után ki szeretnénk szűrni a nem szöveghez tartozó zavaró komponenseket, hogy majd Otsu algoritmusát alkalmazva, a lehető legjobb küszöbértéket találjuk meg, melyet más, nem szöveg alapú komponens nem befolyásol.

Ennek érdekében az éleket kitágítjuk, kissé összemoszuk, hogy összefüggő komponenseket kapjunk [21]. Ezáltal a közeli karakterek, esetleg sorok egyetlen komponenset fognak alkotni. Ez a bináris kép egy szűrőként fog viselkedni, az eredeti, kontrasztos képből kiválasztjuk azokat a pixeleket, amelyeknek a bináris képen való megfelelője egyes (fehér).



5. ábra. A Sobel operátor eredménye.



6. ábra. A kitágított élek eredménye, majd a szűrő alkalmazása. A potenciális írást tartalmazó részek láthatók a második képen

3.2.4. Binarizálás

A fenti módszer alkalmazása után egy olyan képet kapunk, amelynek a legtöbb komponense írást kódol, tehát alkalmazni lehet az Otsu algoritmusát.

Mivel a zavaró komponensek nagy részben eltűntek, meghatározható egy globális küszöbérték az algoritmushoz. Ebben a fázisban már jól elkülöníthetőek a szöveg alapú komponensek a háttértől.

3.2.5. Az írás irányának a meghatározása, kép forgatása

Ismét a fentebb említett képtágítást alkalmazzuk, hogy a betűk kissé egybemosódjanak. Ezeknek a komponensek az irányítását szeretnénk megkapni. Erre az adott komponens másodrendű momentait számoljuk ki (kovariancia mátrix). Az irány szöge meg fog egyezni annak az ellipszisnek a



7. ábra. Komponens irányának meghatározása.

főtengelyének az x tengellyel alkotott szögével, amelynek másodrendű momentumai egyenlőek a komponens másodrendű momentumaival [10]. (Matlab dokumentáció: [5]). Ez a módszer empirikusan szinte tökéletes megoldást eredményez.

2 dimenziós folytonos $f(x, y)$ függvényre a $(p + q)$ -adrendű momentumot a következő képlet adja:

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy \quad (10)$$

Képek esetében, ahol az intenzitást $I(x, y)$ -nal jelöljük, a fenti függvényt diszkrétizáljuk:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y) \quad (11)$$

Kovariancia mátrix:

$$\text{cov}[I(x, y)] = \begin{bmatrix} \mu'_{20} & \mu'_{11} \\ \mu'_{11} & \mu'_{02} \end{bmatrix}, \quad \begin{cases} \mu'_{20} = \frac{M_{20}}{M_{00}} - \frac{M_{10}^2}{M_{00}^2} \\ \mu'_{02} = \frac{M_{02}}{M_{00}} - \frac{M_{01}^2}{M_{00}^2} \\ \mu'_{11} = \frac{M_{11}}{M_{00}} - \frac{M_{10} M_{01}}{M_{00}^2} \end{cases} \quad (12)$$

ahonnan ki lehet számolni a Θ irányítási szöget:

$$\Theta = \frac{1}{2} \arctan \left(\frac{2\mu'_{11}}{\mu'_{20} - \mu'_{02}} \right) \quad (13)$$

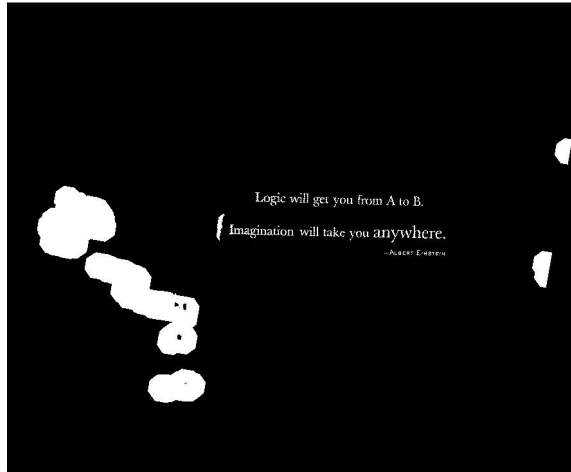
Az összes komponensnek kiszámoljuk ezzel a módszerrel az irányát, és átlagolás helyett a szögek hisztogramjának maximumát vesszük és csupán az itt található szögeket átlagoljuk. A hisztogramot előre meghatározott értékkel definiáljuk.

Ki kell viszont szűrni itt is a maradék, nem szövegalapú komponens irányításának befolyását. Definiálunk egy szögintervallumot a $[-90, 90]$ tartományból (pl $[-25, 25]$) amelyen kívül eső értékeket nem veszünk figyelembe.

Az eddigi előfeldolgozási lépések nagy valószínűséggel biztosítják, hogy a képen található komponensek nagy része szövegalapú, ebből kifolyólag a megállapított szög pontos lesz. A ké-

pet elforgatjuk ezzel a szöggel, minden képpontra a alkalmazzuk a geometriai forgatást:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \quad (14)$$



8. ábra. A kiegyenesített kép.

3.2.6. Sorok szegmentálása

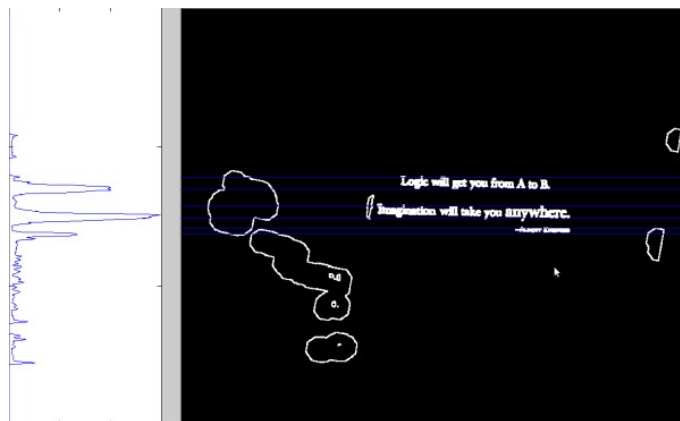
A sorok szegmentálására az úgynevezett Y -hisztogram projekció módszerét alkalmazzuk [20]. Ez egy hatékony megoldás, mivel feltehetően a sorok vízszintesek és kevés zavaró elem található.

A módszer abból áll, hogy soronként vesszük a képet és a képpontok intenzitásértékeit összegezzük, ez lesz a hisztogram értéke egy pontban. A hisztogram mérete megegyezik a kép magasságával. Mivel a karakterek helyén tömörülnek a fehér pixelek, ezért a hisztogramon a sorok helyén tüskék fognak szerepelni, ezáltal a sorokat el lehet különíteni. A nagyon keskeny és nagyon széles sorokat, mint hamis sorokat kizárjuk.

3.2.7. Szavak, betűk szegmentálása

A fenti hisztogramos módszert is alkalmazhatnánk a megkapott sorok X tengelye szerint a betűk és szavak detektálására, azonban ez egy elég naiv módszer, nagy az esélye annak, hogy egyes betűket is többre szegmentáljon, mint például a kis m betűt (3 i betűre vagy egy r és n betűre).

Ehelyett újra megcímkézzük a komponenseket, ez esetben remélhetőleg a külön betűket és minimális számú zavaró foltot, és egyenként meghatározzuk a legkisebb határolókeretét. Ezeknek az egymáshoz viszonyított távolságának átlagából következtethetünk a szavak határára.



9. ábra. A sorok detektálása, Y-hisztogram által.

3.2.8. Normalizálás

A megkapott karaktereket (és nem karakter komponenseket) egységes méretre kell hozni, mivel a felismerő rendszer bemenetként ilyen kis képeket vár el. Ez a standard méret tetszőlegesen 18×9 pixel felbontású.

Egy kép átméretezésére több algoritmus létezik, köztük a bilineáris, bikubikus, legközelebbi szomszédos interpoláció stb.

3.2.9. A karakterek felismerése

A megkapott, normalizált karaktereket már át lehet adni a felismerő rendszernek, a mi esetünkben egy előre betanított neurális hálónak [11], amely 1 – 1 valószínűségi értéket rendel az *abc* összes betűjéhez a bemeneti karakter függvényében. A legnagyobb valószínűséggel rendelkező karakter lesz kiválasztva, mint a felismert betű. A nem karaktert tartalmazó bemeneti kép vélhetően egyetlen karakterhez se fog hasonlítani, így egy valószínűségi küszöbértéket meghatározva ezeket a zavaró elemeket is kiszűrhetjük.

A neurális háló tanítására egy előre elkészített karakterekből álló adatbázist használunk.

Ezáltal a teljes képből kinyert karaktereket felismeri a rendszer és egy digitális szöveget térít vissza, melyet különböző funkcióknak adhatunk tovább.

4. A felismert szöveg felhasználása - előrettekintés

A munka neheze a szövegfelismerésben és ezen belül az előfeldolgozásban rejlik, a felismert szöveg felhasználását érintő rész még implementálásra vár. Külső, online meg Androidos API-kat és szolgáltatásokat használva számtalan lehetőség létezik. Ezek közül pár javaslat következik.

4.1. Szótár és fordítóprogram

A Google Translate [4] szolgáltatását használva szinte bármilyen nyelvről bármilyen nyelvre le tudunk fordítani teljes szöveget, viszonylag pontos szemantikai eredménnyel. Természetes nyelvi feldolgozást (angolul Natural Language Processing - NLP), használ a rendszer. Turisták számára rendkívül hasznos és hatékony felhasználása lenne az alkalmazásnak.

4.2. Kontakt információ mentés

Névjegykártyákon található telefonszám és személyes adatok azonnali mentése is egy hasznos felhasználás lenne. Megjegyzendő, hogy léteznek már erre specializálódott mobilalkalmazások.

4.3. Automatikus internetes keresés

A felismert szövegre automatikusan megnyit egy böngészőt és pl. a Google keresőmotrát felhasználva rákeres az adott szövegre. Nagy előnyt jelent a Google AutoCorrect funkciója, ugyanis a potenciálisan félreismert karaktereket nagy pontossággal ki tudja javítani, ezzel betetőzve az írásfelismerés és javítás részt.

4.4. Specializált internetes keresés

Az internetezők nagy része sokszor csak pár mondatot szeretne gyorsan megtudni egy adott témáról, színésről, ételről stb. Például egy folyóiraton lefotózott cím vagy fontosabb kulcsszó, esetleg film/színész/város neve, felhasználható lenne egy specializált online, gyors keresésre. Több ilyen portál illetve enciklopédia létezik, többek között a Wikipedia, Imdb, Wolfram Alpha stb., melyeket felhasználva 1-1 mondatot kiragadva az adott cikkből, meg lehetne jeleníteni a készülék kijelzőjén további linkekkel együtt.

Ez a pár másodperc időspórlás jelentheti a lusta ember számára a halogatás/felejtés és tájékozottság közti különbséget.

4.5. Extra információ szolgáltatása

A tudatos vásárlás érdekében a csomagolt ételeken szereplő összetevőket felmérő és elemző rendszer különböző online szakoldalakról szerzett információk alapján kategorizálják az adott ételt és

figyelmeztetik a fogyasztó-felhasználót a potenciálisan rákkeltő, veszélyes összetevőkről. Ez a rendszer már komolyabb munkát igényel, de hasznos következményei lennének.

4.6. Hangszintetizálás

Szintén a Google-nek létezik egy szöveget hanggá alakító szolgáltatása (Google Text to Speech [3]), mely főleg a vakoknak és diszlexiásoknak jelenthetne előnyt. Ez a képen felismert szöveget vagy az internetről visszakapott választ is felolvashatná.

5. Köszönetnyilvánítás

A munka során rengeteg segítséget nyújtott a témavezetőm, dr. Csató Lehel, aki ötletekkel és tanácsokkal látott el, ezért nagy köszönet illeti. Szintén köszönöm a közeli ismerőseimnek a támogatást.

Hivatkozások

- [1] Android honlapja. <http://www.android.com/>.
- [2] Google goggles. <http://www.google.com/mobile/goggles/#text>.
- [3] Google text to speech.
<http://techcrunch.com/2009/12/14/the-unofficial-google-text-to-speech-api/>.
- [4] Google translate.
<http://translate.google.com/>.
- [5] Komponens irányításának matlab implementációja.
<http://www.mathworks.com/help/toolbox/images/ref/regionprops.html>.
- [6] Wiener filter matlab implementációja.
<http://www.mathworks.com/help/toolbox/images/ref/wiener2.html>.
- [7] Word lens. <http://questvisual.com/>.
- [8] E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2004.
- [9] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [10] François Chaumette. Image moments: A general and useful set of features for visual servoing. *IEEE TRANSACTIONS ON ROBOTICS, VOL. 20, NO. 4,* 2004.
- [11] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2001.
- [12] Megan Elmore and Margaret Martonosi. A morphological image preprocessing suite for ocr on natural scene images.
- [13] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.
- [14] Kalyan Takru Joie Hadi Nata Tan Graham Leedham, Chen Yan and Li Mian. Comparison of some thresholding algorithms for text/background segmentation in difficult document images. *IEEE Computer Society*, 2003.
- [15] Prof. William Hoff. Binary image processing (eggn 512 computer vision - colorado school of mines, engineering division).
- [16] Anil K. Jain Keechul Jung, Kwang In Kim. Text information extraction in images and video: A survey.
- [17] Jae S. Lim. *Two-Dimensional Signal and Image Processing*. Prentice Hall, 1990.

- [18] A. Marion. *Introduction to Image Processing*. Van Nostrand Reinhold Company, 1990.
- [19] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. Sys.*, 2:62–66, 1979.
- [20] Tsang Ing Ren Rodolfo P. dos Santos, Gabriela S. Clemente and George D.C. Calvalcanti. Text line segmentation based on morphology and histogram projection. *2009 10th International Conference on Document Analysis and Recognition*, 2009.
- [21] John C. Russ. *The Image Processing Handbook, Third Edition*. CRC Press, 1998.
- [22] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis and Machine Vision*. CL-Engineering, 2008.
- [23] Colin Ware. *Information Visualization*. Morgan Kaufmann, 2004.