Vizuális kódok lokalizálásának javítása egyszerű jellemzők kombinációjával*

Bodnár Péter és Nyúl László

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem {bodnaar, nyul}@inf.u-szeged.hu

Kivonat A vizuális kódok automatikus felismerése számos alkalmazási területen szükséges. A képalkotás körülményei és a felismerés módszerei minden alkalmazásban különböző igényeket támasztanak a sebesség és pontosság felé. Korábbi munkánkban bemutattunk egy morfológiai műveletekre és uniform partícionálásra épülő módszert és tanulmányoztuk viselkedését egy tesztkép-halmazon. Ebben a munkánkban megvizsgáljuk egyszerű lokalizációs módszerek együttes hatékonyságát különböző egyesítési lehetőségeket használva, így növelve a lokalizáció hatékonyságát.

Keywords: vizuális kód, vonalkód, számítógépes látás, jellemzőkinyerés, morfológiai műveletek, távolság-transzformáció, Hough transzformáció

1. Bevezetés

A vonalkódok egydimenziós, jól definiált, párhuzamos vonalak halmazából álló vizuális kódok, melyek reprezentációjukban a hordozott adat könnyű gépi dekódolását célozzák például bevásárlóközponti terminálokkal vagy okostelefonokkal. A dekódolás gyors, automatikus, és a kód általában hibajavító információt is tartalmaz. Kétdimenziós vizuális kódokat is „barcode", azaz vonalkód elnevezéssel illet a szakirodalom. Jelen írásban mi az alábbi kódokat értjük vonalkód alatt (1. ábra). A módszerünk kiterjeszthető kétdimenziós vizuális kódok felismerésére is kisebb módosításokkal a képi jellemzők keresésében, az összehasonlításnál viszont csak egydimenziós esetekre szorítkozunk széles körben elterjedt 2D tesztadatbázisok hiánya miatt.

* A dolgozat angol nyelven a következő helyen jelent meg: Improving barcode detection with combination of simple detectors, Proceedings of the 8th International Conference on Signal Image Technology & Internet Systems (SITIS 2012) Sorrento, Italy, 25/11/2012-29/11/2012. IEEE, pp. 300-306. (ISBN:978-0-7695-4911-8)

(a) Code 128 (b) EAN-13 (c) UPC (d) UPC-A

1. ábra: Vizuális kód-szabványok

A vonalkódok lokalizálási módszereinek két fő törekvése van, ezek minél nagyobb sebesség és pontosság elérése. Okostelefonokon elvárt a gyors felismerés, viszont a pontosság nem követelmény, mivel a felhasználó szükség esetén többször is megismételheti a képalkotási eljárást, apró módosításokkal a kamera pozícióján és szögén. A pontosság kiemelten fontos ipari alkalmazásoknál (például postaszolgálatnál), ahol a kihagyott detektálások anyagi veszteségben nyilvánulnak meg. A sebesség itt másodlagos, bár szintén fontos szempont. A vonalkódok lokalizálásában két fő elvárást állítottunk. Egyrészt meg kell találni azokat a részeket a képtérben, amik nagy valószínűséggel vonalkódrészt tartalmaznak, és garantálni, hogy ne maradjon ki a detektálásnál vonalkód. Másrészről meg kell adnunk egy minél jobban illeszkedő befoglaló téglalapot ezekre a területekre, hogy a detektor minél kevesebb fölösleges munkát végezzen. A vonalkódok lokalizálása a képtérben a nehezebb feladat, hiszen a lokalizálás után az információ dekódolása már egyértelmű és relatíve egyszerű feladat.

Az egydimenziós vonalkódok esetében a lokalizálás alapvető módszere egy-egy vonal „szkennelése” a képtérben. Ez gyakori a kézi PoS lézerszkennereknél és az okostelefonra készült alkalmazásoknál. Ezek a vonalak egy egydimenziós intenzitásprofilt adnak, melyeket a detektáló algoritmusok [1–3] emberek számára értelmezhető információvá alakítanak. A módszerek lényege, hogy az intenzitásprofilból ideális bináris függvényt állítanak elő, majd keresőtáblából, esetleg valószínűségi becslés beiktatásával karakterekké konvertálják azt. Ez általában jól orientált kódokat igényel, melyek a képtér nagyobb részét kitöltik, ezért nem alkalmazható hatékonyan olyan esetekben, mikor a képtér tetszőleges részén keresünk kódot.

A völgydetektáláson alapuló módszer [1] erősen simított vagy alacsony felbontású képeken működik, melyek gyakran készülnek okostelefonok kameráival. Három lépésből áll. Először kezdőpontokat keres a képen, majd a gradiens segítségével követi a „völgyeket”, végül pedig felismeri azok végpontjait, ezzel definiálva egy szakaszt, ami a vonalkód része lehet.

A morfológiai műveletekre alapozott algoritmusok [4–9] számos változata egy fő ötletre épül. Alapvető morfológiai műveleteket használ, mint az erózió, a dilatáció, és azok származtatott műveletei, mint például a *bottom-hat* művelet. Összefüggő területek fogják jelezni a lehetséges vonalkód-pozíciókat a jellemzőképeken. További feldolgozás, mint például a szegmentálás és a kis területek eldobása történik ezután. Ez a hozzáállás 1D és 2D vizuális kódokra egyaránt

használható. A mi módszerünk szintén használ morfológiai műveleteket az egyik jellemző előállítására.

Egyéb algoritmusok wavelet transzformáción alapulnak [10], és a képeken a vonalkód-szerű jellemzőket kaszkádolt gyenge osztályozókkal találják meg. Minden osztályozó a wavelet domainen dolgozik, és szűkíti a lehetséges kódok halmazát, ezáltal a falsnegatív értéket, miközben igyekszik a pontosságot megtartani.

A Hough-transzformáción és annak módosításain alapuló módszerek [11] a kép éltérképén dolgoznak. A két leggyakoribb módszer a standard és a valószínűségi Hough-transzformáció. Mindkettő az élpontokat először Hough-térbe transzformálja, majd ezután döntéseket hoz a lehetséges vonalkódra hasonlító képi területekről a talált vonalak alapján. Mi is végeztünk kísérleteket a valószínűségi Hough-transzformációval, és kiterjesztettük az alapötletet a jellemzők magasabb szintű vizsgálatával.

Egyéb módszerek is léteznek, melyek szűrőket, például Gabor-szűrőt használnak [12].

2. A javasolt lokalizálási módszer

Ebben a fejezetben bemutatunk több különböző lokalizációs megközelítést. A legtöbb esetben a képet lokálisan vizsgáljuk, a jellemzőket négyzetmozaikon nyerjük ki. Távolság-transzformációt [13] használunk és mérjük a kontraszt-varianciát a mérőszámok előállításához. A vonalkód-részek más textúrákhoz hasonlóan jól követhető jellemzőkkel rendelkeznek. Egyik a szomszédok hasonlósága, ami azt jelenti, hogy egy kódrész szomszédságában egy másik kódrész hasonló tulajdonságokkal rendelkezik egy megfelelően megválasztott csempeméret esetén. A minta ismétlődésének köszönhetően meg tudjuk állapítani, hogy hány érintett csempe tartalmaz kódrészt, és ezek egymással összefüggőek-e. Végül felmérjük a talált összefüggő komponensek kompaktságát, ami a végső döntésünket szintén befolyásolja. A kontraszt-információkat szintén figyelembe vesszük. Utolsó jellemzőként pixelszinten is keresünk összefüggő komponenseket.

A valószínűségi Hough-transzformációval a kódok vonalai egyenként detektálhatóak további feldolgozáshoz. Ezen vonalak klaszterekbe csoportosítása segít eldönteni, hogy egy vonal egy kódhoz tartozik-e.

A klaszterezés ötlete [4] szintén alkalmazható pixelszinten. Ezen jellemző mérőszámát a pixelklaszterek számosságára és alakjára alapozzuk.

Végül az egész képteret morfológiai műveletekkel is szűrjük, ami nagy hatékonysággal rendelkezik, és jó alapot nyújt egyéb, összetettebb lokalizáló algoritmusok számára is. A MIN–MAX [4] módszert használjuk egyik jellemzőnk előállítására.

Ezen jellemzők többféle módon aggregálhatóak, például többségi szavazással, a jellemzőképek halmazának maximum-értékével előállítva pixelszinten, továbbá súlyozott szavazással is. Mindegyik módszer különböző célt szolgál. A többségi szavazás akkor használható, ha sok egyszerű jellemzőn alapuló lokalizálónk van, melyek relatíve alacsony pontossággal rendelkeznek, de a kódok többségét megtalálják (magas recall érték). Nagyobb bizonyossággal osztályozhatjuk a pixe-

leket ezzel a módszerrel, miközben a falspozitívok arányát alacsonyan tudjuk tartani. A maximum alapú jellemzőkép előállítása jó, ha maximalizálni szeretnénk a recall értéket, hogy egyetlen kódrész se maradjon ki a detektálásból. Ez sajnos jelentősen csökkenti a pontosságot, főleg akkor, ha az egyes detektorok is gyengék voltak benne. Ipari alkalmazásokban mégis jól alkalmazható módszer, mivel itt fontos minden gyanús régió megtalálása. Súlyozott szavazással is előállíthatjuk a végső jellemzőképet. Mivel a recall értéket szeretnénk itt is maximalizálni, először külön megvizsgáltuk az egyszerű detektorok recall értékeit, majd ezt használtuk arra, hogy súlyokat rendeljünk a közös döntéshez. A detektoraink a Hough transzformáción, távolság-transzformáción, kontrasztmérésen, klaszterezésen és morfológiai műveleteken alapulnak. Az ajánlott módszert az 1. algoritmus foglalja össze.

Algorithm 1 A javasolt módszer vonalkódok lokalizálására

Funct ROI_Detect(I)

I = Normalize_levels(I)
I_{canny} = Canny_edge_detect(I)
I_{sharp} = Unsharp_masking(I)
$I_{distance}$ = Distance_Transform(I_{canny})
F_{minmax} = Min_Max(I_{sharp})
F_{hough} = Probabilistic_Hough(I_{canny})
for t in *tiles* **do**
  $V_{distance}(t)$ = Distance_Transformation_Measure($I_{distance}$) *Minden csempéhez értéket rendelünk*
  $V_{contrast}(t)$ = Contrast_Measure(I_{sharp})
  $V_{cluster}(t)$ = Local_Clustering_Measure(I_{sharp})
end for
Filter($V_{distance}$) *Connected component labelling*
Filter($V_{contrast}$) *a kicsinek aposztrofált területek eldobása*
Filter($V_{cluster}$)
for (x, y) in I **do**
  $F_{distance}(x, y)$ = GetTileValue($V_{distance}, x, y$) *a jellemzőképek felépítése a csempe-információ alapján*
  $F_{contrast}(x, y)$ = GetTileValue($V_{contrast}, x, y$)
  $F_{cluster}(x, y)$ = GetTileValue($V_{cluster}, x, y$)
end for
$F_{majorvote}(x, y) = Maj(F_{minmax}, F_{hough}, F_{distance}, F_{contrast}, F_{cluster})$
$F_{maxval}(x, y) = Max(F_{minmax}, F_{hough}, F_{distance}, F_{contrast}, F_{cluster})$
$F_{weightedvote}(x, y) = 1/(\sum_{F_i \text{in} F} \text{recall}(F_i)) \times \sum_{F_i \text{in} F} F_i(x, y) \times \text{recall}(F_i)$ *F: az összes jellemzőkép halmaza*
return

2.1. Előfeldolgozás

A kamerával rögzített képeknek gyakran előfeldolgozásra van szüksége az eszköz hibái és a környezeti nehézségek miatt. Az alacsony kontraszttal rendelkező képeknél normalizálás szükséges. Az előfeldolgozás során életlen maszkolást is használunk, ami az eredeti kép és a Gauss-simított negatív kép súlyozott összege. A Gauss szűrőt úgy állítjuk be, hogy ne semmisítse meg a legvékonyabb vonalakat a képen keresett vonalkódokból. Erre a paraméterre tudunk következtetni a végső felhasználási területből, és a várható hozzávetőleges kódméretből. Mivel néhány jellemzőt bináris képből nyerünk ki, szükséges küszöbölés is. Egyszerű küszöbölést használunk, ha a kép egyenletes megvilágítással rendelkezik, egyébként adaptív küszöbölés [14] szükséges.

A képfelbontásnak nem kell nagynak lennie, elég ha a vonalkód legvékonyabb vonala két pixel vastagságú. Ekkor már 3×3 pixeles mediánszűrőt alkalmazhatunk a só-bors zaj eltávolítására. Nagyobb felbontás nagyobb pontossághoz vezet, de növeli a feldolgozás idejét is. A legkevésbé időigényes megoldás a kép újramintavételezése a legközelebbi szomszéd interpolációval, ami jó választás az erős élek megőrzése miatt is. Elfogadható pontosság eléréséhez ajánlott olyan felbontást használni, ahol legalább 3 pixel a legvékonyabb keresett vonal.

2.2. Vonalak detektálása Hough-transzformációval

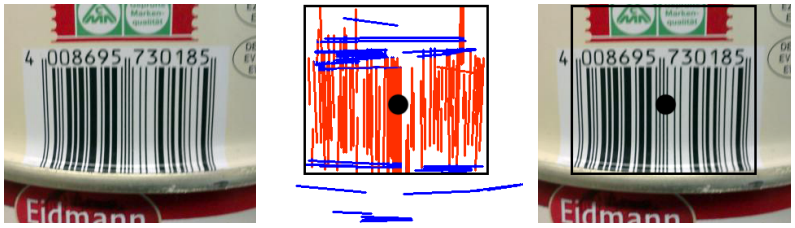
A módszer általános képfeldolgozó műveleteken alapszik, mint a Canny éldetektálás és a valószínűségi Hough-transzformáció [15]. A vonalkódok nagyjából egyforma hosszú, hasonlóan orientált, kis területen elhelyezkedő szakaszokból állnak, ezért az élpontok transzformációjával jó becslést kapunk a lehetséges vonalkódokat alkotó szakaszokra. Előfeldolgozásként ajánlott egy simító szűrő a Canny éldetektor számára. Mivel minden vonalkód a teszt-adatbázisunkban legalább 50 pixel magas, a minimum vonalhosszt 40 pixelnek vettük, és azokat a szakaszokat tartottuk meg a valószínűségi Hough-transzformáció után, amelyek legalább ilyen hosszúak.

Miután kapunk egy listát a fontos szakaszokról, középpont, hossz és orientációs információkkal, csoportokba foglalhatjuk őket, hogy megvizsgáljuk, vonalkódot alkotnak-e. A minimum darabszámot és a szakaszok közelségét csoporton belül paraméterként állíthatjuk be, ahogyan az átlagtól vett toleranciát is a vonalak orientációjára. Mivel a vonalkódjaink legalább 25 párhuzamos vonalból állnak, 20-ra állítottuk a minimum értéket.

A végső lépésben a szakaszcsoportok középpontját keressük meg, és vágjuk a képeket, majd továbbítjuk a vágott területet a dekódernek, ami már ismert módszerekkel feldolgozza a kódot (2. ábra).

2.3. Uniform partícionálás távolság-transzformációval

A vonalkódok, mint általában a textúrák, könnyen azonosíthatóak egy részletük megfigyelésével. Ezek a részletek együtt alkotják a keresett vonalkód régiót a

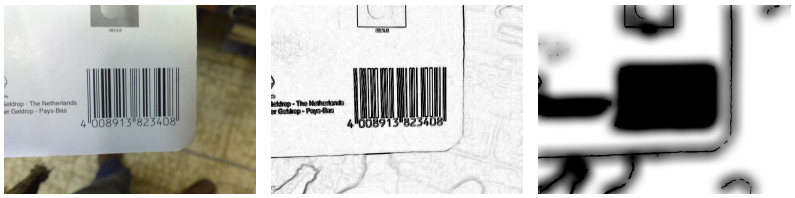


(a) eredeti kép (b) jellemzőkép (c) overlay

2. ábra: Canny éldetektor valószínűségi Hough transzformációval. A (b) ábrán pirossal jelöltük azokat a szakaszokat, melyek a vonalkódot alkotó szakaszcsoportokra jellemző kritériumoknak eleget tesznek, míg kékkel a többi megtalált szakaszt. A (c) ábra mutatja a megtalált befoglaló téglalapokat és a kód középpontját az eredeti képre vetítve.

képtérben, becsülhető szélességben és magasságban. Az első része a módszernek partícionálja a képet diszjunkt, négyzet alakú, a síkot teljesen lefedő blokkokra, és minden blokkhoz egy mérőszámot rendel ezután, amely azt jellemzi, mennyire tartalmazhat az adott blokk lehetséges vonalkódot. Minden blokkhoz rendelt érték ennek a jellemzőnek a mértékét adja meg. Globálisan egy mátrixot tudunk ebből formálni. A textúra-részek hasonló statisztikákkal bírnak egymás szomszédságában, ezért a mátrixban ilyen területek keresése megadja a kép számunkra fontos területeit, melyek nagy valószínűséggel vonalkódot tartalmaznak.

A társított érték az éltérkép távolság-transzformációja alapján mondja meg vonalkód-részlet előfordulási valószínűségét a blokkban (3. ábra). A távolságtranszformációhoz használt élpontokat Canny módszerével adjuk meg. Az előző fejezet módszeréhez előállított éltérkép itt újra felhasználható. Minden blokkban kiszámoljuk a távolság-transzformáció által kapott pixel-értékek átlagát. 1D kódokra ezek az értékek a minimum és maximum vonalszélesség fele közé esik.



(a) eredeti kép (b) éltérkép (c) távolságtérkép

3. ábra: Canny éltérkép (b) és távolság-térkép (c) egy valós példához (a). A vonalkódok sötét összefüggő területeken helyezkednek el. Megjegyzés: a vizualizáció érdekében az értékeket skáláztuk.

Miután minden blokkot lokálisan kiértékeltünk, összefüggő komponenseket keresünk a jellemzőmátrixban. A kísérletek azt mutatják, hogy az távolság-értékek a vonalvastagság fele körül szóródnak. Küszöböléssel eldöntjük minden blokkról, hogy vonalkód-szegmenst tartalmaz-e. 25 % toleranciát adunk ezeknek az ideális értékeknek, hogy a detektálás pontossága kielégítő legyen. A végfelhasználói alkalmazásoknál számolnunk kell zajjal, képhibákkal, reflekciókkal, karcolásokkal. Az erős zajt tartalmazó képeknél a távolság-értékek mérhetően esnek. A karcolások, szennyeződések szintén módosítják a távolságok átlagát attól függően, hogy azok milyen frekvenciájú komponenseket tartalmaznak. A toleranciát ennek figyelembe vételével kell beállítani, és a kielégítő paraméterezés csak teszteléssel érhető el. A tolerancia beállítása mindig megalkuvással jár a pontosság és a hamis pozitív detektálások között, tehát a végső alkalmazás dönti el annak mértékét.

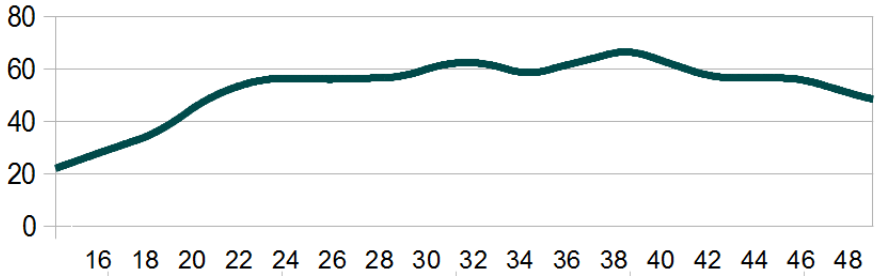
A kapott bináris mátrixot tovább analizáljuk az összefüggő komponensek címkézésével. Végül a kis méretű komponenseket kidobjuk, és a megmaradó komponensek súlypontját adjuk vissza. Egy komponens kicsinek tekinthető, ha N-nél kevesebb blokkot tartalmaz (1. egyenlet), ahol h és w a vonalkód magassága és szélessége, mely következtethető a végső alkalmazási helyből, s pedig a blokk területe.

$$N = \max\left(4, \frac{|h-s| \times |w-s|}{s^2}\right) \tag{1}$$

Mivel a legkisebb vonalkód a teszthalmazunkban 60 pixel magas, 30×30 pixelnél nagyobb blokkméret gyenge pontossághoz vezet. Ugyanakkor a túl kicsire választott blokkméret a kódok súlypontjának meghatározásánál okozhat problémát, mivel ebben az esetben a kód alatti karakterek hasonló távolságértékeik miatt a jellemzőképen az összefüggő komponens részévé válnak. A sima szöveg nincs ilyen hatással a detektálásra, és az általában megfelelő távolságra helyezkedik el a vonalkódtól. A legjobb blokkméret nagyjából a vonalkód magasságának harmadánál alakul (4. ábra). Mivel minden vizsgált kódunk ugyanúgy párhuzamos vonalakból áll, a teszteléshez azok hosszának átlagát választottuk blokkméretnek.

Lehetőségünk van ezt a módszert diszjunkt vagy átlapoló blokkokkal futtatni, de az átlapolás ebben az esetben nem növeli a hatékonyságot, csak a számítási időt. A blokkok kezdőpontjának eltolása nincs jelentős hatással a detektálásra, mivel ez csak a többé-kevésbé vonalkódszerű blokkokat tolja el a lehetséges vonalkód szélein.

A módszer magas falspozitív aránnyal rendelkezik (5. ábra), mivel a szöveg gyakran ugyanúgy kielégíti azt a feltételt a távolság-értékek átlagában, mint a vonalkód-szegmens. Ennek ellenére mégis használhatjuk gyenge osztályozóként, hogy kizárjunk olyan részeket a képtérből, amik biztosan nem tartalmaznak vonalkódot távolság-értékeik alapján, illetve olyan részeket, melyek nem elég nagyok és kompaktak ebben a tekintetben. A távolság-transzformációs eljárás mindenképp jó alapot szolgáltat bonyolultabb, pontosabb detektorok számára is a lokalizálásban. A paraméterek optimális értékeinek meghatározása erősen alkal-



4. ábra: Pontosság a blokkméret függvényében. X-tengely: blokkméret aránya a vonalkód magasságához képest; Y-tengely: pontosság (mindkettő százalékban kifejezve).

mazásfüggő, mégis becsülhetjük a várható vonalkód-dimenziókat, és a vonalak várható minimális és maximális vastagságát is. A megvilágítás mennyisége és egyenletessége szintén jól körülhatárolható ipari alkalmazásoknál. Itt a kódok mérete szintén becsülhető a digitális kép dimenziójából és a kamera által rögzített jelenetből.

(a) eredeti kép (b) távolságtérkép (c) detektált fontos téglalapok

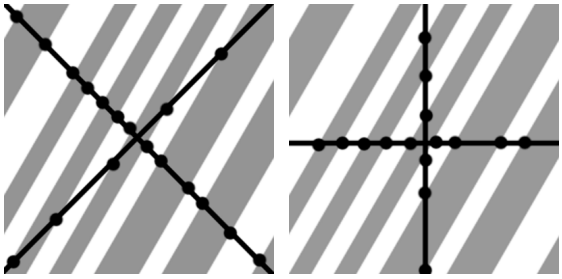
5. ábra: Valódi példa nem egyenletes megvilágítással. Eredeti kép (a), távolságtranszformáció (b), és a végső fontos területek (c)

2.4. Kontrasztmérés uniform partícionálással

A partícionálás ötletével számos jellemző kinyerését kombinálhatjuk, például a kontraszt mérését is. Az egydimenziós vonalkódok általában nagy varianciát mutatnak a kontrasztban egyik irányban, míg arra merőlegesen kevesebbet. A detektálásról szóló szakirodalom szintén használja ezt az információt az orientáció megtalálására, viszont gyakran ezek a módszerek már közel függőlegesen vagy vízszintesen álló kódokból indulnak ki. Az előző fejezetben tárgyalt szabályok a partícionálásról itt is érvényesek, a különbség mindössze annyi, hogy itt

a kontraszt-értékek alapján rendeljük hozzá a blokkhoz a mérőszámot, amit a végső döntésben használunk.

Minden blokkot a vonalszkennelés módosított változatával vizsgálunk. Két pár vonalat használunk, egy 0 és 90 fokos, egymásra merőleges párt, és egy 45 és 135 fokost. Egy vonalkód orientációjától függetlenül reagálni fog az egyik párra, például egy függőlegesen állított vonalkód az elsőre. Az első pár 0 fokos vonala sok eltérést fog találni, míg a második tagja keveset, így a mérőszám magas lesz (2. egyenlet). A módszer lényege a 6. ábrán látható. A két vonalpárra adott reakció maximumát vesszük a blokk mérőszámának, és ez 1 közeli lesz vonalkód esetén, 0 közeli zaj vagy homogén intenzitás esetén.

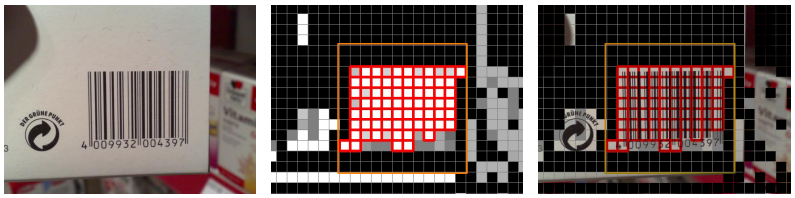


6. ábra: Két vonalpár „szkennel" végig a képen. Az első párban nagy eltérést fogunk kapni a kontraszt varianciájában a két vonal mentén. Ebben a példában az első vonalpár jól felismeri a vonalkódot

$$C_i = \frac{|V_{i1} - V_{i2}|}{\max(V_{i1}, V_{i2})} \tag{2}$$

ahol V_{ij} a kontraszt-variancia j vonal mentén az i vonalpárban.

A módszer többi része ugyanaz, mint a 2.3 fejezetben.



(a) eredeti kép (b) jellemzőkép befoglaló téglalappal (c) overlay-kép

7. ábra: Kontrasztmérés valódi példán

3. Kiértékelés

Az említett módszereket egy saját képadatbázison teszteltük, és a pontosságát összehasonlítottuk ismert vonalkód-lokalizálási technikákkal, egyik a bottom-hat morfológiai műveletre épülő [8], másik pedig gradiens-alapú [1].

A paramétereket mindig az alkalmazáshoz finomhangoljuk, mégis adhatunk olyan kezdeti értékeket, amik megfelelőek lehetnek általánosan. Becsülhető a várható vonalkód-méret, vonalhossz és szélesség. Egyéb paraméterek szintén, melyek a kamera vagy a jelenet tökéletlenségeiből adódnak. Ilyen például a zaj vagy a kép simítottsága. Ezek a körülmények döntik el az életlen maszkolásunk maximális túllövését is és a küszöböléshez használt értéket. Egy tipikus beállítás a következő paraméterekkel rendelkezhet: 80 % a minimum várható vonalhossz a Hough-traszformációnak, és ugyanennyi a vonalkódot alkotó vonalak darabszámára is adható küszöbértékként, mikor közelség és orientáció alapján csoportosítunk. Az orientációnál a toleranciát a várható kódok sokféle előfordulása döntheti el (például, hogy mennyire szenvedhetnek affin transzformációtól a kamera által rögzített képen). Az optimális blokkméret a kód kisebbik dimenziójának harmada (4. ábra). A küszöbérték kontrasztméréskor 0.5–0.75 közé eshet a mátrixban. A MIN–MAX módszer kernelméretét úgy kell megválasztani, hogy legalább a legvastagabb várható vonal szélessége legyen [4]. A paramétereket finomhangolhatjuk tapasztalat útján vagy ismert optimalizációs eljárásokkal.

3.1. Teszt adatbázis

Mivel nem találtunk hivatalos vonalkód-adatbázist, hogy elvégezzük a tesztet, készítettünk 103 képet áruházi termékekről egy Nokia N95 okostelefonnal, melyeket 640×480 pixel méretűre skáláztunk bilineáris interpolációval. Kisebb tükröződések, simítottság, képhibák és torzulások fordulnak elő ezeken a képeken. Találtunk továbbá egy vonalkód-adatbázist összehasonlítás céljára, amit Tekin és Coughlan[1] készített. A *ground truth* képek manuálisan készültek el, anélkül hogy a vonalkódok szabvány szerinti „csendes” zónáit vagy a hozzájuk tartozó karaktersorozatot a képen figyelembe vettük volna.

Számos detektáló szoftver és keretrendszer van kereskedelmi forgalomban, mint például a DTK Barcode Reader SDK[2], BC Tester[3], és Barcode Recognition SDK a DataSymbol[4] cégtől, sajnos ezek nem publikálták a detektáló mechanizmust.

3.2. Megvalósítás és tesztkörnyezet

A módszert `C++`-ban implementáltuk `OpenCV` függvénykönyvtár segítségével. A C++ modern, objektum-orientált megközelítést biztosított gyors végrehajtással,

[1] http://www.ski.org/Rehab/Coughlan_lab/Barcode
[2] http://www.dtksoft.com/
[3] http://www.bctester.de/
[4] http://www.datasymbol.com/

míg az OpenCV minden fontos segédfüggvényt tartalmaz, amely a képfeldolgozáshoz szükséges. A kiértékelés és a futásidőket Intel Core 2. Duo 3.00 Ghz CPU-val végeztük, GPU gyorsítás nélkül.

3.3. Pontosság és sebesség

Ahhoz, hogy összehasonlítsuk a módszer hatékonyságát, a leggyakoribb mérőszámokat vettük elő. Ilyenek a precision, recall, accuracy és F-measure. Ezeket az értékeket a Jaccard-indexre alapozva számítottuk ki

$$J(G,D) = \frac{\sum_{x,y}(G(x,y) \wedge (D(x,y))}{\sum_{x,y}(G(x,y) \vee (D(x,y))} \tag{3}$$

ahol G és D bináris függvények a ground truth és a saját binaritált kimeneti jellemzőképeink alapján.

Az átlagos hatékonysági indikátorainkat a 1. táblázat mutatja.

1. táblázat: A javasolt módszer hatékonysági indikátorai

Módszer	Precision	Recall	Accuracy	F-measure	Futásidő
Tuinstra	57.08 %	85.29 %	84.19 %	48.39 %	160 ms
Juett et al.	34.26 %	94.08 %	72.76 %	36.13 %	230 ms
Hough transzf.	64.83 %	85.07 %	84.22 %	58.76 %	230 ms
Távolság-transzf.	20.00 %	95.85 %	54.52 %	23.54 %	190 ms
Lokális klaszterezés	81.68 %	72.34 %	89.22 %	62.12 %	120 ms
MIN–MAX	26.39%	97.59 %	54.45 %	29.62 %	460 ms
Kontrasztmérés	51.17 %	88.02 %	82.58 %	49.07 %	140 ms
Többségi szavazás	53.15%	85.70%	85.25%	48.44%	680 ms
Jellemzők maximuma	21.25%	96.45%	61.84%	24.51%	680 ms
Súlyozott szavazás	37.45%	91.97%	78.11%	37.35%	680 ms

A távolság-transzformáció inkább gyenge osztályozóként használható önmagában. Nagy arányú falspozitívval rendelkezik, mégis, jó recallt mutat. Inkább kivételeket ad meg, mint valós detekciókat.

A valószínűségi Hough-transzformáció robusztus választás a vonalkódok lokalizálására, mivel paraméterezhető minimum vonalhosszal és vonalszegmenseket összekötő lyukmérettel rendelkezik. Jól kezeli a zajt is bizonyos fokig, míg érzékeny az affin torzulásokra.

A távolság-transzformáció és a lokális klaszterezés természetének köszönhetően megbízható marad kisebb torzulásoktól szenvedő képek esetén is, ellentétben a Hough transzformációval, ami sokkal érzékenyebb a szakaszok orientációjára.

A legmegbízhatóbb módszerek morfológián alapulnak, mivel ez a megközelítés jól kezeli a zajt, a simított képeket és realtíve nagy torzulásokat is. Sajnos a konvolúciók miatt ez lassú futást is jelent.

A képtér partícionálása és a jellemzőképen azonos jellemzőkkel rendelkező blokkok keresése általános megközelítése a minták keresésének. Különböző jellemzőkkel, mint a kontraszt-variancia, a hisztogram-információ vagy a távolság-információ, jól használható vonalkód-lokalizáló eljárásként.

Az egyszerű detektorokat együttesen használva jelentősen javul a teljesítményük. A többségi szavazással növelhető a pontosság, bár ez járhat fontos területek elvesztésével. A jellemzőképek maximumának használata jó recall értéket biztosít, viszont csökkenti a pontosságot. A súlyozott szavazás az egyéni recall értékek alapján jó együttes recall értéket biztosít, míg realtíve magasan tartja a pontosságot. Mivel a jellemzőképek ugyanazok, csak az aggregációjuk történik más módszerrel, itt a futásidők közel azonosak.

4. Következtetések

Bemutattunk több megközelítést vonalkódok lokalizálására digitális képeken, egyszerű jellemzőket használva. Tanulmányoztuk a viselkedésüket egy teszt-adatbázison, melyek különböző típusú vonalkódokat tartalmaztak. Kísérleteztünk az együttes teljesítményükkel és megmutattuk a hatékonyságát.

Ipari környezetben párhuzamos végrehajtással tovább növelhető a detektálás sebessége, továbbá a javasolt módszer köztes adatai, például az éltérkép több helyen is felhasználható.

Jövőbeli munkánkhoz tartozik bonyolultabb jellemzők vizsgálata a recall maximalizálása érdekében ipari alkalmazások számára, ahol az érvényes kódok elvesztése elfogadhatatlan. Gyors lokalizálási megoldásokon is dolgozunk, melyek kamerák szoftverébe ágyazhatóak.

5. Köszönetnyilvánítás

Jelen kutatási eredmények megjelenését „Az SZTE Kutatóegyetemi Kiválósági Központ tudásbázisának kiszélesítése és hosszú távú szakmai fenntarthatóságának megalapozása a kiváló tudományos utánpótlás biztosításával” című, TÁMOP-4.2.2/B-10/1-2010-0012 azonosítószámú projekt támogatja. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Nyúl László kutatásait a Magyar Tudományos Akadémia Bolyai János Kutatási Ösztöndíja támogatja.

Hivatkozások

1. Timothy R. Tuinstra. *Reading Barcodes from Digital Imagery.* PhD thesis, Cedarville University, 2006.
2. Eugene Joseph and Theo Pavlidis. Bar code waveform recognition using peak locations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):630–640, 1994.

3. Robert Adelmann. Toolkit for bar code recognition and resolving on camera. In *Phones – Jump Starting the Internet of Things. In: Informatik 2006 workshop on Mobile and Embedded Interactive Systems*, 2006.
4. Péter Bodnár and László G. Nyúl. Barcode detection with morphological operations and clustering. In *Signal Processing, Pattern Recognition, and Applications, Proceedings of the Ninth IASTED International Conference on*, pages 51–57, 2012.
5. Péter Bodnár and László G. Nyúl. Barcode detection with uniform partitioning and morphological operations. In *Conference of PhD Students in Computer Science, Proceedings of Conference*, pages 4–5, 2012.
6. Daw-Tung Lin, Min-Chueh Lin, and Kai-Yung Huang. Real-time automatic recognition of omnidirectional multiple barcodes and dsp implementation. *Machine Vision and Applications*, 22:409–419, 2011. 10.1007/s00138-010-0299-3.
7. Daw-Tung Lin and Chin-Lin Lin. Multi-symbology and multiple 1d/2d barcodes extraction framework. In Kuo-Tien Lee, Wen-Hsiang Tsai, Hong-Yuan Liao, Tsuhan Chen, Jun-Wei Hsieh, and Chien-Cheng Tseng, editors, *Advances in Multimedia Modeling*, volume 6524 of *Lecture Notes in Computer Science*, pages 401–410. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-17829-0-38.
8. Xiaojun Qi James Juett. Barcode localization using bottom-hat filter. *NSF Research Experience for Undergraduates*, 2005.
9. Melinda Katona and László G. Nyúl. A novel method for accurate and efficient barcode detection with morphological operations. In *The 8th International Conference on Signal Image Technology (SITIS 2012)*, 2012. Accepted for publication.
10. R. Oktem. Bar code localization in wavelet domain by using binary morphology. In *Signal Processing and Communications Applications Conference, 2004. Proceedings of the IEEE 12th*, pages 499–501, april 2004.
11. D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.
12. A.K. Jain and Y. Chen. Bar code localization using texture analysis. In *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, pages 41–44, 1993.
13. Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell Computing and Information Science, 2004.
14. Sue Wu and Adnan Amin. Automatic thresholding of gray-level using multistage approach. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 493–497 vol.1, 2003.
15. Nahum Kiryati, Yuval Eldar, and Alfred M. Bruckstein. A probabilistic hough transform. *Pattern Recognition*, 24(4):303–316, 1991.