

# **TDK-dolgozat**

Katona Melinda

**Szegedi Tudományegyetem  
Természettudományi és Informatikai Kar  
Képfeldolgozás és Számítógépes Grafika Tanszék**

## **Vonalkódok hatékony detektálása morfológiai módszerekkel**

*Készítette:*

**Katona Melinda**

IV. gazdaságinformatikus BSc  
szakos hallgató

*Témavezető:*

**Dr. Nyúl László**

egyetemi adjunktus

# Tartalomjegyzék

|                                                                       |           |
|-----------------------------------------------------------------------|-----------|
| <b>Bevezetés</b>                                                      | <b>4</b>  |
| <b>1. A vonalkódok áttekintése</b>                                    | <b>5</b>  |
| 1.1. A vonalkód kialakulásának története . . . . .                    | 5         |
| 1.2. Főbb 1D-s vonalkód típusok . . . . .                             | 6         |
| 1.3. Főbb 2D-s vonalkód típusok . . . . .                             | 9         |
| <b>2. Az alkalmazott alapvető képfeldolgozó műveletek áttekintése</b> | <b>12</b> |
| 2.1. Az előfeldolgozás során használt műveletek . . . . .             | 12        |
| 2.2. Morfológiai műveletek . . . . .                                  | 13        |
| <b>3. Vonalkód detektáló algoritmusok</b>                             | <b>15</b> |
| 3.1. Morfológiai alpműveleteken alapuló eljárás . . . . .             | 15        |
| 3.2. A kép többszöri pásztázásán alapuló eljárás . . . . .            | 17        |
| 3.3. Morfológiai völgydetektáláson alapuló eljárás . . . . .          | 20        |
| 3.4. A javasolt saját módszer . . . . .                               | 21        |
| <b>4. A tárgyalt algoritmusok összehasonlító elemzése</b>             | <b>24</b> |
| 4.1. Tesztsorozat, tesztkörnyezet és implementáció . . . . .          | 24        |
| 4.2. Eredmények vizsgálata . . . . .                                  | 25        |
| <b>Összefoglalás</b>                                                  | <b>31</b> |
| <b>Irodalomjegyzék</b>                                                | <b>32</b> |

# Bevezetés

A vonalkódtechnika az automatikus azonosítás egyik pillére. A különféle kódtípusok és az alkalmazási területek speciális problémákat vetnek fel, így folyamatosan szükség van újabb és hatékonyabb megoldásokra. Számos eljárás létezik a vonalkódok lokalizálására, melyek pontosság és sebesség szempontjából jól jellemezhetők. Kódok automatikus megtalálására elsősorban nagy sebességű feldolgozási helyeken, pl. futószalagokon, automatizált gyártósoroknál van szükség, ahol a hibás detektálásoknak akár komoly költségvonzatai is lehetnek.

A vonalkódoknak két fő csoportja létezik, az 1D-s és a 2D-s kódok. Jelen dolgozat keretein belül a klasszikus 1D-s vonalkódokat tartalmazó képek szegmentálásával foglalkozunk. A cél a képből kinyert jellemzők segítségével a vonalkód helyének automatikus meghatározása gyors, de ugyanakkor pontos módszer segítségével. Ismertetünk és összehasonlítunk néhány a szakirodalomban található módszert, melyek alapvetően a kódot reprezentáló alakzatok és a háttér közötti kontrasztra támaszkodnak, valamint bemutatunk egy új, saját fejlesztésű algoritmust is. A dolgozatban ismertetett eljárások mindegyike morfológiai műveleteken alapul. Az általunk fejlesztett algoritmus a morfológiai völgydetektáláson alapszik, mely hatékonyan készíti elő a képet a további feldolgozásra. Az algoritmus a detektálási fázis során is morfológiai műveleteket alkalmaz, melyek gyorsan, és mindemellett nagy pontossággal határozzák meg a képeken található vonalkódok elhelyezkedését.

Az első fejezetben tanulmányozzuk a manapság leggyakrabban használt 1D-s, illetve 2D-s vonalkód típusokat, azok vizuális jellemzőinek kiemelésével. A második fejezetben áttekintjük azon alapvető képfeldolgozó műveleteket, melyekből felépíthetők a bemutatott algoritmusok. Néhány irodalmi módszer és a saját algoritmus bemutatását a harmadik fejezet tartalmazza. A negyedik fejezetben végül összehasonlítjuk a tárgyalt algoritmusokat egy több ezer képből álló adatbázison, vizsgálva azok hatékonyságát. A tesztképek különböző típusú vonalkódokat, képenként egy vagy több vonalkódot tartalmaznak, illetve különféle torzításokkal terheltek.

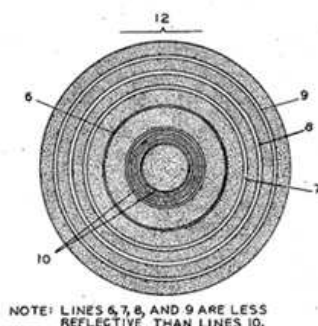
# 1. fejezet

## A vonalkódok áttekintése

Ebben a fejezetben ismertetjük a napjainkban leggyakrabban használt 1D-s és 2D-s vonalkódokat, hangsúlyozva azok vizuális jellemzőit, melyekre az automatikus kód-detektáló módszerek is támaszkodhatnak. A dolgozat későbbi részében ismertetésre kerülő algoritmusok 1D-s vonalkód típusokra lettek kifejlesztve, de léteznek olyan 2D-s kódtípusok is, melyekre működőképesek az eljárások. Mindez amiatt lehetséges, mert a 2D-s kódok közt megtalálhatóak olyanok (halmozott kódok), melyek szerkezetileg a hagyományos vonalkódhoz hasonlóan, egymással párhuzamosan elhelyezkedő vonalak alkotta alakzatok. A fejezetben az egyes vonalkód típusok ismertetését a [6] forrásra támaszkodva adtuk meg.

### 1.1. A vonalkód kialakulásának története

A vonalkód megjelenése viszonylag fiatalnak mondható, körülbelül az ötvenes évekre tehető, de tömeges elterjedése csak 1973-ban indult meg. Bernard Silver és Norman Joseph Woodland nevéhez fűződik az első vonalkód, aminek megjelenését az azt leolvasó rendszer kiépítése követett. A vonalkódok létrejötte a technika fejlődésével áll szoros összefüggésben, és nem a matematikai apparátus hiányától vagy rendelkezésre állásától tehető függővé.



1.1. ábra. Az első vonalkód

A vonalkód ősének sokan a Morse kódot tekintik, mivel Joseph Woodland és Bernard Silver a Morse kódok rajzolásából kapta az ötletet. Először arra gondoltak, hogy Morse

kódokat használjanak horizontálisan nyomtatva, létrehozva keskeny és széles vonalakat, de ezt később elvetve megalkották az első vonalkódot, mely koncentrikus körökből állt, mint ahogy az 1.1-es ábrán jól látható. Ez vékonyabb és vastagabb körökből álló kód, ahol a vonalak és a vonalközök voltak az információhordozók, ezeket bármilyen irányból, bármilyen fénnnyel le lehetett olvasni. 1835-ben dolgozta ki Samuel Finley Breese Morse a rendszerét, melynek alapjául a ritmus szolgált, vagyis az idő felosztása, tagolása. Morse az angol ábécé betűit feleltette meg a rövid (pont), illetve a hosszú (vonat) jeleknek, és az ezek kombinációjából kialakított jelcsoportoknak. Ez a megfeleltetés nem esetleges volt, hanem a betűk előfordulásának gyakorisága szerint történt: a leggyakoribb „E” betű kapta például a legegyszerűbb jelet, a pontot. A vonalkódban is a Morse rövid-hosszú jelsorozataihoz hasonlóan a különböző szélességű sávokat alkalmazták. Egy konkrét példát tekintve az „á” betű kódja

jel - szünet - szünet - jel - szünet, számjegyekkel írva: 10010

A vonalkódokat először vasúti kocsikon, majd az amerikai posta járművein próbálták ki. Az évek során a különböző fejlesztések következtében a vonalkód rengeteg változatban megjelent, olyanok is, amelyek már szöveget is tudnak tárolni, sőt a beléjük kódolt információ is maga egy kód, amelynek megfejtéséhez szükséges az adott kereskedelmi egység adatbázisához való hozzáférés. Magyarországon az 1990-es években terjedt el, de a Skála már 1984-ben hasznosította a vonalkódtechnika nyújtotta előnyöket.

## 1.2. Főbb 1D-s vonalkód típusok

A hagyományos vonalkódok felépítése azonos: egymással párhuzamos, különböző vastagságú világos és sötét közők, illetve vonalak meghatározott váltakozása fejezi ki az információt, mely gépek által optikailag leolvasható kód. Az egyes vonalkódok abban különböznek egymástól, hogy az adott karakternek milyen fekete és fehér vonalakkal álló sorozatot feleltethetünk meg.

A felépítése az alábbi struktúrát követi: a vonalkód elején egy nyugalmi zóna található, ezt követi a start karakter, egy vagy több adatkarakter, opcionálisan egy vagy több ellenőrző karakter, majd a stop karakter, végül a hátsó nyugalmi zóna.

Egy kód karakterkészlete azt adja meg, hogy az adott kódrendszer segítségével milyen karakterek kódolhatóak (numerikus, alfanumerikus, kis- és nagybetűk, vezérlő karakterek). A vonalkód modulmérete a vonalkódot alkotó legkeskenyebb vonal fizikai szélességét határozza meg, és általában a hüvelyk ezredrészében adják meg.

A vonalkódok leggyakoribb felhasználási területe a kereskedelem, például az áruk csomagolásán, amely lehetővé teszi az áru adatainak, úgy mint a gyártó országa, a gyári szám és a termék cikkszám gyors azonosítását. A vonalkódos azonosítás a GS1 rendszer legismertebb eleme, mely a kereskedelmi folyamatok elválaszthatatlan része. A vonalkód jelképekkel megjelenített azonosító számok teszik lehetővé a gépek számára az elektronikus leolvasást, ami nagyban segíti és gyorsítja az információ áramlását. A vonalkódok lehetnek fix vagy tetszőleges hosszúságúak. Fix hosszúságú kódok esetében a szabványban meghatározásra kerül, hogy milyen hosszú karaktersorozat kódolására képes. Ez a tetszőleges hosszúságú kódok esetén nem meghatározott.

A vonalkódok tulajdonságaik alapján csoportosíthatók. Lehetnek bináris vagy delta, diszkrét vagy folytonos, illetve egy vagy több jelentésű kódok a különböző osztályozások

alapján. A bináris kódok esetében a széles vonal és vonalköz 1-est, míg a keskeny vonalak illetve vonalközök 0-t jelentenek. A delta kódok közös jellemzője, hogy a fekete vonalakhoz a szélességüknek megfelelő darabszámú 1-est rendelünk, a vonalközökhöz hasonló szempontok alapján 0-akat. Egy másik osztályozás szerint diszkrétnek nevezzük azokat a vonalkódokat, ahol a karaktereket elválasztó vonalközök nem hordoznak információt. Folytonos kódok esetén a karaktereket elválasztó vonalközök is értékes információt hordoznak. További osztályozási lehetőséget nyújt annak vizsgálata, hogy az adott vonal- és vonalközsorozat egyértelműen meghatároz-e egy adott karaktert vagy sem. A vonalkódok nagy része egyjelentésű, ami azt jelenti, hogy egyértelműen hozzárendelhető egy karakter egy vonalaktól álló mintázathoz. A többjelentésű kódok több karakterkészletet kódolnak, melyek közt választhatunk vezérlőkarakterek segítségével.

## Diszkrét kódok

Mint ahogyan már említettük, diszkrét kódok esetében a karaktereket elválasztó vonalközök nem hordoznak információt. Az alábbiakban ismertetünk egy manapság is ismert és használt szabványt, mely diszkrét kódokat definiál.

A Code 39 szabványkódok megjelenése az 1974-re tehető és David Allais és Ray Stevens nevéhez fűződik. A Code 39 egy tetszőleges hosszúságú típus.



1.2. ábra. Code 39 vonalkód

A kódolást tekintve egy karakter öt vonalból és négy vonalközből áll, melyből három széles. Alkalmazási területe igen kiterjedt, ami annak köszönhető, hogy egyszerű a felépítése. Leginkább ipari szervezetek, kórházak és könyvraktárok használják.

## Folytonos kódok

A folytonos kódok esetében a karaktereket elválasztó vonalközök is értékes információt hordoznak. Az alábbiakban az ilyen jellegű kódokat definiáló szabványok közül áttekintjük a Codabar, Code128, I2of5, illetve a UPC/EAN családot.

A Codabar típusú kódok 1972-es megjelenése Piney Bows nevéhez köthető. Minden karakter négy vonalból és három közből áll, amelyek szélessége határozza meg a kódolt jelet.

A Codabar egy tetszőleges hosszúságú vonalkód típus. A Codabar típusú kódokat leginkább a vérbankokban és a fénykép laborokban használják.

A Code 128 vonalkód típus az előzőhöz képest újabb keletű, de rohamosan terjed, ami annak köszönhető, hogy az eddig típusok összes előnyét magában hordozza.

A Code 128 szintén tetszőleges hosszúságú vonalkód típus. A karakterek 3 vonalból és 3 közből állnak, ezek 1-3 modul szélesek, karakterenként összesen 11 modul. Egy modul



1.3. ábra. Codabar vonalkód



1.4. ábra. Code 128 vonalkód

szélessége 1,27 milliméter és 1,9 milliméter között változhat. A legszélesebb felhasználási területen megtalálható.

A *UPC* egyidejűleg alkot egy vonalkód típust és egy termékazonosítási rendszert. A vonalkód rögzített hosszúságú.



1.5. ábra. UPC-A és UPC-E vonalkód

A UPC karakterek mindegyike két vonalból és két közből áll, amelyek 7 modul szélességben helyezkednek el. A kód olvashatóságát a középén, illetve a széleken elhelyezett speciális vonalak segítik, melyek három fekete és kettő fehér modulból állnak. A UPC-A vonalkód könnyen átalakítható EAN-13 szabványnak megfelelőre. Ez a kódtípus tipikusan kereskedelmi alkalmazásokra lett kifejlesztve körülbelül húsz évvel ezelőtt az USA-ban.



1.6. ábra. EAN-13 és EAN-8 vonalkód

Az európai gyártmánykód olyan vonalkódrendszer, ami az Észak-Amerikában kifejlesztett 12 számjegyű UPC kód kibővítése. Az *EAN* az első világméretű termékazonosító rendszer és kódtípus. Ezek a vonalkódok fix hosszúságú számok kódolására alkalmasak, hasonlóan, mint a UPC típusúak. Az EAN-13 felépítése alapján a kód első kettő vagy három karaktere az országazonosító, a rákövetkező négy-öt karakter a gyártó azonosítója. A megmaradt helyeken lévő karakterekből leolvasható a termékazonosító is.

Az EAN kódok a legismertebb felhasználási területe a bolti csomagolások rendszerezése.

Az *I2of5* (Interleaved 2 of 5) a „kettő az ötből” kódcsalád legismertebb tagja. Egyszerű felépítésű kódtípus. A kód tetszőleges hosszúságú, de csak páros számú számjegyet tartalmazhat. A vonalkód elnevezése abból ered, hogy egy karakter öt modulból áll, amelyből kettő széles, három pedig keskeny. Mindegyik karakter két vastag és három vékony vonalból áll.



1.7. ábra. Interleaved 2 of 5 vonalkód

Fő alkalmazási területe a gyűjtőcsomagolások vonalkódos jelölése, zárt ipari rendszerek, egészségügyi alkalmazások, okmány- és dokumentációs alkalmazások.

### 1.3. Főbb 2D-s vonalkód típusok

A hagyományos vonalkódok használata során felszínre kerültek az előnyeik és a felhasználók körében egyre több alkalommal igény mutatkozott olyan kódokra, melyek információ tárolására alkalmasak, és kisebb méretben képesek eltárolni több információt. Az 1D-s vonalkódok felépítésükben kötöttek, így ezen típusok esetében erre nincs lehetőség. Ezek következtében alakították ki a 2D-s kódokat. A 2D-s kódoknak két csoportját különböztetjük meg a felépítésük alapján, a halmozott szerkezetű, és a mátrix kódokat.

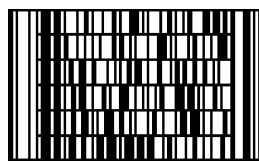
#### Halmozott kódok

A halmozott kódok a hagyományos vonalkódok szerkezetére jellemző vonalak és vonalközök váltakozó szélességű sokaságából állnak. A hagyományos vonalkódoktól való különbözőség abban mutatkozik meg, hogy több, vékony szeletre bontott vonalkód kerül egymás tetejére. Valamennyi ilyen típusú kód alkalmas nagyobb mennyiségű információ tárolására, viszont nem alkalmas az információk kis helyen való tárolására, mert minimális mérete megegyezik az 1D-s kódtípusokéval, illetve az olvasási irány is kötött. Az alábbiakban két ilyen kódtípust ismertetünk.

A *Codablock* vonalkódok az 1D-s kódoknál ismertetett Code 39 rendszerére épülnek. Kifejlesztője Heinrich Oehlmann. A szabványban a sorok száma 1 és 22 között mozoghat. Az egy sorban elhelyezkedő karakterek száma nem korlátozott. Minden kód tartalmaz start és stop karaktereket. A Codablock fizikai méretét tekintve szabadon alakítható, adott korlátok mellett.

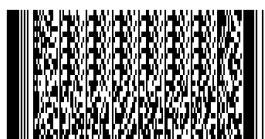
Felhasználási területe a kereskedelem és az ipar.

A következőekben bemutatásra kerülő halmozott kód a *PDF417*. A PDF417 szabványt 1991-ben Ynjiun Wang dolgozta ki. Ez az egyik legkorábban bevezetett 2D kódok egyike. A legkisebb értelmezhető információt tartalmazó elem a kódszó. Minden kódszó felépítését 4 fekete és 4 fehér elem jellemzi. Ezek szélessége 1-6 modul közt változhat. A modul egy elem minimális szélességét jelenti. A 4 fekete és 4 fehér modul teljes



1.8. ábra. Codablock vonalkód

modulszélessége 17, tehát a PDF417 elnevezés a kódrendszer felépítéséből származik. Maximálisan 30 sornyi és 34 oszlopnyi adat megjelenítésére képes.



1.9. ábra. PDF417 vonalkód

Széles körben elterjedt a PDF417 használata. Dokumentumazonosítás, gyártási és szállítási adatok rögzítésére alkalmazható.

## Mátrix kódok

A mátrix kódok, ellentétben a halmozott kódokkal, nem emlékeztetnek nagy mértékben a hagyományos kódokra. Szerkezetüket tekintve nem vonalakból állnak, hanem sötét és világos cellákból építkeznek, melyek elrendezése mátrixszerű alakzatot követ. Ezen kódolás előnye, hogy nagy mennyiségű információ tárolására képes kis helyen. A következőekben bemutatunk két fontosabb mátrix kódtípust.

A *Data Matrix* típusú kódokat a négyzetes elrendezés jellemzi. Helyzetét a négyzetet alkotó keretvonalak határozzák meg, mely egy része úgynevezett szilárd vonal, míg a másik része hiányos vonal. Mindez a kiolvasást segíti. A Data Matrix kódokat a kis méretük, illetve a gyors leolvasási lehetőség miatt használják.



1.10. ábra. Data Matrix vonalkód

Ezen típusú kódot olyan alkalmazásokban érdemes használni, ahol a terület gazdaságossága lényeges. Ezt ismerve a fő felhasználási területek a postai csomagok, nyomtatott termékek jelölése és leltározása.

A *QR kódok* manapság egyre szélesebb körben elterjedtek. Ezeket is, a Data Matrix kódokhoz hasonlóan, a mátrix kódok közé soroljuk. A QR kód nevét a Quick Response kifejezésből kapta, mely egyszerre utal a gyors visszafejtési sebességre, és a felhasználó által igényelt gyors reakcióra. Legfeljebb 177 modul lehet a szélessége és magassága a kódnak. Előnyös tulajdonsága, hogy a sarkokban elhelyezett négyzet alakú mezői miatt,

melyek önmagukban is információt hordoznak, bármely szögből tekintve könnyen értelmezhető, biztosan feldolgozható jelet ad. Felhasználási területe például a logisztika és a szórakoztatóipar.



1.11. ábra. QR vonalkód

## 2. fejezet

# Az alkalmazott alapvető képfeldolgozó műveletek áttekintése

A fejezet keretein belül ismertetjük azokat a morfológiai operációkat, illetve az előfeldolgozás során használt képműveleteket, melyek a következő fejezetben bemutatott eljárások során felhasználásra kerültek.

### 2.1. Az előfeldolgozás során használt műveletek

Az előfeldolgozás során az ismertetésre kerülő algoritmusok három különféle képműveletet alkalmaznak, melyek a kontraszt széthúzás, Gauss simítás, illetve a gradiens kép képzése Sobel kernel használatával.

A kontraszt széthúzás az intenzitások egy megadott  $[old_{min}, old_{max}]$  intervallumát skálázza egy  $[new_{min}, new_{max}]$  intervallumba [1].

$$s = (r - old_{min}) \frac{new_{max} - new_{min}}{old_{max} - old_{min}} + new_{min},$$

ahol  $s$  kimeneti érték,  $r$  bemeneti érték. A kontraszt széthúzás helyett hisztogram kiegyenlítés is alkalmazható. A hisztogram kiegyenlítés olyan növekvő függvényű pontoperáció, mely „közel” konstans hisztogramú képet eredményez [1].

Gauss simítás alatt a képfüggvénynek egy Gauss-szűrő függvénnyel vett konvolúcióját értjük,

$$h = f * G,$$

ahol az  $f$  a kiindulási kép a  $G$  a Gauss-szűrő, a  $h$  a simított kép, illetve a  $*$  a konvolúció műveletét jelöli [1]. Ezenfelül meg kell adni a Gauss-függvény szórását is, ezt a  $\sigma$  jelöli, mely az elmosás mértékét befolyásolja.

A képeken az éleket az eltérő intenzitásérték változások jelzik, tehát a képfüggvény első parciális deriváltjait véve megkaphatjuk a gradiens képet. A Sobel operátor a parciális derivált közelítésére szolgál. Az adott pontbeli  $G_x, G_y$  értékek az alábbiaképpen állnak elő:

$$G_x = I * \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, G_y = I * \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$G_x \approx \frac{\partial I(x, y)}{\partial x}, G_y \approx \frac{\partial I(x, y)}{\partial y},$$

ahol  $I$  a kép.

A gradiens vektor hosszát a  $|G| = \sqrt{G_x^2 + G_y^2}$  euklideszi távolsággal lehet megadni.

## 2.2. Morfológiai műveletek

A morfológia szó a biológia szakterületén a növények és állatok alakjának leírását jelenti. Ennek analógiájára a képfeldolgozásban a matematikai morfológia eszköztárát használjuk a geometriai struktúrák kvantitatív leírásához. A matematikai morfológia halmazelméleti módszereken alapul. A módszer kifejlesztői G. Matheron és J. Serra.

A morfológiai műveletek bemenete egy ponthalmaz és egy elemi minta, a szerkesztőelem. A műveletek tárgyalását két olyan módszerrel kezdjük, amik a későbbi összetett eljárások alapelemei. Ez a két művelet duális tulajdonságú, így az egyik a másikból származtatható, de az alaphalmazra különböző hatással bírnak, ennek következtében külön definiálják őket.

Legyen  $X \subseteq \mathbb{Z}^2$  ponthalmaz és  $Y \subseteq \mathbb{Z}^2$  szerkesztőelem. Az  $X$  halmaz  $Y$ -nal való *dilatáltja* alatt a

$$X \oplus Y = \{a \mid (X \cap (\hat{Y})_a) \neq \emptyset\}$$

halmazt értjük. Ha  $Y \subseteq \mathbb{Z}^2$ , akkor  $\hat{Y}$  jelöli az  $Y$  elem origóra vett *tükröképét*,  $Y_a$  pedig az  $a \in \mathbb{Z}^2$  vektorral való *eltoltját*, vagyis  $\hat{Y} = \{-y : y \in Y\}$  és  $Y_a = \{y + a : y \in Y\}$ .

Dilatáció során először középpontosan tükrözzük az  $Y$  szerkesztőelemet, majd az  $X \oplus Y$  dilatáltban azon  $a$  eltolásvektorok szerepelnek, melyekkel a tükrözéssel kapott  $\hat{Y}$  halmazt eltolva, annak még marad közös pontja az  $X$  ponthalmazzal.

Az *erózió* definíciója a következőképpen írható fel:

$$X \ominus Y = \{a \mid (Y)_a \subseteq X\}$$

Az  $X$  halmaz  $Y$ -nal való *erodáltja* azon  $a$  vektorokat tartalmazza, amelyekkel  $Y$ -t eltolva az még az  $X$  belsejében marad.

A dilatáció és az erózió duális viszonyára az

$$X \ominus Y = (X^C \oplus \hat{Y})^C$$

$$X \oplus Y = (X^C \ominus \hat{Y})^C$$

összefüggések mutatnak rá. Mindezek a halmazelméleti De-Morgan azonosságokból ki-következtethetőek.

A dilatáció és az erózió duális műveletek, azonban a két művelet nem inverze egymásnak. Ez azt jelenti, hogy a dilatációt követő erózió ugyanazon szerkesztőelemmel, illetve a műveletek fordított sorrendje nem feltétlenül adja vissza az eredeti alaphalmazt.

Legyen  $X \subseteq \mathbb{Z}^2$  ponthalmaz és  $Y \subseteq \mathbb{Z}^2$  szerkesztőelem, ekkor

$$X \circ Y = (X \ominus Y) \oplus Y$$

módon definiáljuk az  $X$  halmaz  $Y$  szerkesztőelemmel való *nyitását*. A nyitás alkalmazható az objektumok határának simítására az azon lévő kisebb kinövések törlésével, kis objektumok törlésére, vékony nyúlványok eltávolítására, illetve alakdetektálásra is.

A  $Y$  szerkesztőelemmel való *zárás* az alábbiaképpen definiáljuk:

$$X \bullet Y = (X \oplus Y) \ominus Y$$

A zárás szintén simítja a határokat, eltünteti a kis üregeket, a határ kis hézagait feltölti, illetve a szűk kiszögelléseket és hosszú, vékony öblöket összeolvasztja. Még meg kell említenünk, hogy a nyitás

$$\begin{aligned} X \circ Y &\subseteq X \\ X_1 \subseteq X_2 &\Rightarrow X_1 \circ Y \subseteq X_2 \circ Y \\ (X \circ Y) \circ Y &= X \circ Y \end{aligned}$$

tulajdonságokkal, a zárás

$$\begin{aligned} X \bullet Y &\supseteq X \\ X_1 \subseteq X_2 &\Rightarrow X_1 \bullet Y \subseteq X_2 \bullet Y \\ (X \bullet Y) \bullet Y &= X \bullet Y \end{aligned}$$

tulajdonságokkal rendelkezik.

A nyitás és a zárás közt is fellép dualitás, annak köszönhetően, hogy az erózió és dilatáción alapul mindkét művelet. Jelen esetben dualitáson a következőket értjük:

$$X \circ Y = ((X^C \oplus Y) \ominus Y)^C, \text{ illetve } X \bullet Y = ((X^C \ominus Y) \oplus Y)^C,$$

azaz a halmaz nyitása megegyezik a komplementerhalmaz zárásának komplementerével, valamint a halmaz zárása megegyezik a komplementerhalmaz nyitásának komplementerével.

A következő morfológiai művelet, melyet tárgyalunk, a *hit-or-miss transzformáció*. Ezen transzformáció az alakfelismerés egy egyszerű eszköze. A hit-or-miss transzformáció segítségével adható meg a morfológiai vékonyítás, a vastagítás, a tisztítás és a konvex burok.

Legyen  $X \subseteq \mathbb{Z}^2$  halmaz és  $Y \subseteq \mathbb{Z}^2$  szerkesztőelem. Bontsuk fel az  $Y$  szerkesztőelemet két diszjunkt halmaz egyesítésére:  $Y = Y_1 \cup Y_2$  ( $Y_1 \cap Y_2 = \emptyset$ ). Az  $X$  halmaz  $Y$ -nal való hit-or-miss transzformációja az alábbi:

$$X \circledast Y = (X \ominus Y_1) - (X \oplus \hat{Y}_2),$$

ami azt jelenti, hogy a transzformált halmaznak az a pont lesz eleme, melyre az  $Y$ -t (a középpontjával ráhelyezve) az  $Y_1$ -beli elemek  $X$ -beli, az  $Y_2$ -höz tartozó elemek pedig az  $X^C$ -beliekkel kerülnek fedésbe.

A fejezet utolsó eljárása, melyet ismertetünk, a *völgydetektor*. Völgydetektálás során kivonjuk az eredeti képet, annak zárással kapott eredményképéből. Ez az alábbiaképpen írható fel:

$$W_s(A) = (A \bullet S) - A,$$

ahol  $A$  a kiindulási képet,  $S$  a záráshoz alkalmazott szerkesztőelemet jelöli.

## 3. fejezet

# Vonalkód detektáló algoritmusok

Ebben a fejezetben négy vonalkód detektáló algoritmust tárgyalunk. Mind a négy algoritmus más módszert vesz a helymeghatározás alapjául. A pseudokódokban a 2. fejezetben bevezetett, a [2] -ben szereplő morfológiai műveleti jeleket használtuk.

A digitális képek minősége manapság kevésbé kifogásolható, bár gyenge minőségű felvételek is készülnek. A minőségromlás oka pl. lehet maga a digitalizáló eszköz, illetve a környezet okozta nehézségek. Mindezek következményeként gyakran szükséges a kép minőségének javítása a konkrét feldolgozás előtt.

### 3.1. Morfológiai alpműveleteken alapuló eljárás

Ebben az alfejezetben Telkin és Coughlan [4] cikkében szereplő algoritmus kerül ismertetésre. Az eljárás morfológiai műveletekre támaszkodik. A legtöbb algoritmus az előfeldolgozás során egyszerű műveleteket alkalmaz, úgy mint a kvantálás, melynek során az inputban kapott képből bináris képet alkotunk. Jelen esetben arra alapozva, hogy a vonalkódban található vonalak és a köztük található pixelek intenzitásértékei közötti különbség nagy, gradiens képet képezve a vonalak erőteljesebben kiemelhetőek. A gradiens számítása  $x$  és  $y$  irányban történik, Sobel kernel használatával. A gradiens számítása egy képre lokális képművelet, mivel az adott pixel kernelben meghatározott környezete határozza meg az intenzitását. Ezt követően, a kapott eredményképet küszöbölve csak azok a régiók kerülnek megtartásra, amelyek gradiens értékei magasak, majd bináris képet képezünk.

Az előfeldolgozást követően az algoritmus további részében bináris képpel kerülnek elvégzésre a képműveletek. A lokalizálás folyamata morfológiai műveletsorra épül. Első lépésben hit-or-miss transzformáció megvalósítása történik a megfelelő szerkesztőelem használatával. A cikkben nem meghatározott, hogy a transzformációhoz a szerkesztőelem nagysága miből adódik. Az implementálás során a képméret és a leghosszabb vonal függvényeként került számításra. Az eljárás folyamán eltávolításra kerülnek mindazon objektumok, melyek nagy valószínűséggel nem tartalmaznak vonalkódot.

Ezt követően, hogy az egymással nem szoros összefüggésben lévő objektumok egy régiót alkothassanak, dilatáció szükséges. A szerkesztőelem jelen esetben négyzet alakú, de a pontos nagysága ennek sem rögzített a cikkben, a gyakorlatban a képméret határozta meg. Természetesen a dilatáció következtében olyan régiók is összeolvasztásra kerülnek, melyek nem tartalmaznak vonalkódot, de majd eltávolításra kerülnek a későbbiekben.

**INICIALIZÁLÁS:**

|                                        |                                  |
|----------------------------------------|----------------------------------|
| $N$ = kép szélessége                   | $M$ = kép magassága              |
| $f$ = szürkeárnyaltos kép              | $i_t$ = minimális küszöbérték    |
| $\otimes$ = hit-or-miss transzformáció | $\oplus$ = morfológiai dilatació |
| $\ominus$ = morfológiai erózió         | $Se$ = szerkesztőelem            |
| $min_a$ = minimális terület            |                                  |

---

**ALGORITMUS:**

```

for  $i := 0$  to  $N$  do
  for  $j := 0$  to  $M$  do
    gradiens számítása  $x, y$  irányban
    if  $f_{i,j} > i_t$  then //  $f_{i,j}$  - kiszámított gradiens értékek
       $f_{i,j} \leftarrow 1$ 
    else
       $f_{i,j} \leftarrow 0$ 
    end if
  end for
end for

 $f := f \otimes Se$ 
 $f := f \oplus Se$ 
 $f := f \ominus Se$ 

if object area  $> min_a$  then
  Felvesszük a vonalkódot tartalmazó területek közé az objektumot
else
  Elvetjük az objektumot
end if

```

3.1. ábra. Tekin és Coughlan algoritmusának pszeudokódja [4]

A következő lépésben morfológiai erózió segítségével a vékony objektumokat töröljük a képről. Az erózióhoz felhasznált kernel megválasztása során ügyelni kell arra, hogy nagyobb legyen, mint a dilatació során alkalmazott, pontosan amiatt, hogy a nem kellő részek törlésre kerüljenek.

Az algoritmus zárásaként a vélhetően fals pozitív objektumok eltávolítására kerül sor, egy úgynevezett szilárdsági teszt keretében. Ennek során a kis területű objektumok törlésre kerülnek. A fentebb leírt műveletek végrehajtása egy inputként kapott képen meghatározza a vonalkód, vagy vonalkódok helyét. Az algoritmus pszeudokódja a 3.1 ábrán látható.

## 3.2. A kép többszöri pásztázásán alapuló eljárás

Az alábbiakban Tuinstra [3] cikkében szereplő eljárást tárgyaljuk, melyet legfőképp gyengén látó, illetve vak emberek mindennapjainak megkönnyítésére irányulóan terveztek. A cikk egy teljes leírást ad az input kép minőségi javításától egészen a vonalkód leolvasásáig. A dolgozat keretein belül csak a vonalkód lokalizálásáig tekintjük át a módszert.

Leginkább UPC-A típusú vonalkódok lokalizálására alkalmas eljárásról beszélünk, mely azért alkalmas más 1D-s kódok, úgy mint UPC-E, EAN szabványcsalád tagjainak helymeghatározására is. Az algoritmus a kép különböző irányokban való pásztázására épül.

Jelen esetben is két fázisra bontva mutatjuk be az eljárást. Az inputként kapott diszkrét képek minősége az esetek nagy részében nem ideális, így hasonlóan az előző fejezetben bemutatott módszerhez, szükséges egy előfeldolgozási fázis a képeknek a konkrét vonalkód lokalizálás megkezdése előtt. Az algoritmus előfeldolgozási fázisának első lépéseként az esetlegesen a képen lévő zajok csökkentésére irányulóan Gauss simítást alkalmaz az eljárás. Erre alapozva az eljárás következő lépéseként a kép egészére nézve kiszámításra kerülnek a pixelek gradiens értékei, ezáltal a képen egyfajta élkijelölést végezve.

A következő lépésben csak azon pixelektípusokkal dolgozik az algoritmus tovább, melyek egy magadott küszöbszám feletti gradiens értékkel rendelkeznek. A küszöbölést követően már nem szürkeárnyaltos képpel dolgozik a módszer, hanem bináris képpel, ami úgy adódik, hogy amely értékek a küszöbszám felett helyezkednek el, fehér pontok lesznek, míg az az alatt lévők fekete pontok.

Mindezek után a detektálás fázisa következik. Elsőként négy orientációban kerül pásztázásra a kép, horizontálisan, vertikálisan, és az átlók irányába ( $\pm 45^\circ$ ). Először a horizontális irány kerül megtekintésre, melynek során olyan él pixelek keresése történik, melyek irányultsága megegyezik a függőleges vonalakkal. Minden egyes él pixel szomszédságában vele ellentétes polaritású pixeleket keres az eljárás. Amennyiben kellő számú ilyen pixel található egy területen, mintha azok egy vonal szegmenst alkotnának, úgy ez a terület nem kerül eltávolításra. Ezt követően függőleges irányú pásztázás következik, melynek során azon szegmensek kerülnek megtartásra, melyek közel ugyanazon véggel és kezdettel rendelkeznek, tehát nagy valószínűséggel vonalkód területek. Ha több ilyen régió is felfedezhető, akkor a következő szakaszhoz adódik a műveletsor, melyben a fals pozitív régiók kerülnek kiszűrésre. Ezen algoritmusok pszeudokódjai láthatóak a 3.2 és 3.3 ábrákon.

A detektálás végső szakaszában előbb kiszámításra kerül az eredményképen az minden egyes pixel entrópia értéke. Ennek kiszámítása esetén megkapjuk, az adott pixel meghatározott szomszédságát véve az intenzitásértékeket mekkora rendezetlenség jellemzi. Mint ismeretes, egy vonalkódot alkotó vonalak rendezetten egymással párhuzamosan helyezkednek el, így az azon régiók entrópiája valószínűsíthetően alacsony lesz, míg a fals pozitív területeké magas. Eredményképen tehát azon régiók lesznek megtalálhatóak, melyek a legnagyobb valószínűséggel tartalmazhatnak vonalkódot.

**INICIALIZÁLÁS:**

$\tau_G$  = minimum gradiens küszöbérték

$n_E$  = # élek maximális száma

$d_E$  = két egymást követő vonal közti maximális távolság

---

**ALGORITMUS:**

```

for orientáció  $t = 0, 45, 90, 135$  do
  for vonal  $l = 1, \dots, utolsoVonalEbbenAzOrientacioban$  do
     $count \leftarrow 0$ 
    for pixel  $i = 1, \dots, utolsoPixelEbbenAzOrientacioban$  do
       $j$  az utolsó pixel a sorban, ami számításra kerül
      if  $|\nabla I_i| > \tau_G$  then
        // A gradiens érték a küszöbérték felett van
        függőleges vonal pásztázás
        if  $\angle \nabla I_i \approx \perp$  then
          if  $|\nabla I_i| \geq \max \{|\nabla I_{i-1}|, |\nabla I_{i+1}|\}$  then
            if  $\angle \nabla I_i \approx 180^\circ$ -ban nincs több terület  $\angle \nabla I_j$ -vel, and  $d_{i,j} < d_E$  then
               $count \leftarrow count + 1$ 
            end if
          end if
        else
           $count \leftarrow count - 1$  // rossz irányban lévő, nagy gradiensű pixel
        end if
      else if  $d_{i,j} > d_E$  then // nincs él pixel a közelben
         $count \leftarrow count - 1$ 
        if  $d_{i,j} > 2 * d_E$  then //nincs él a közelben
           $count \leftarrow 0$  // a jelölt régió vége
        end if
      end if
    if  $count = 0$  then // akkor tekintjük, ha elértük a jelölt régió végét
       $score \leftarrow \max_{i \in utolsRgi} count(i)$ 
      if  $score > n_E$  then // ha minimális # él látható
        Felvesszük ezt a régiót, mint vonalkód jelölt területet
      else
        Elvetjük ezt a régiót
      end if
    end if
  end for
end for
end for

```

3.2. ábra. Vonal pásztázó algoritmus [3]

---

**INICIALIZÁLÁS:**

$nL$  = minimálisan szükséges # vonalak száma

$d_L$  = maximális távolság két vonal közt

$\tau_S$  = szükséges minimális érték vonalkód régió deklarálásához

$\beta = \{\}$  // vonalkód régiók listája

---

**ALGORITMUS:**

```
for orientáció  $t = 0, 45, 90, 135$  do
  for line  $l = 1, \dots, utolsoVonalEbbenAzOrientacioban$  do
    for vonalkód régió jelölt  $c = 1, \dots, utolsoJeloltEbbenAzOrientacioban$  do
      if  $\exists b \in \beta : begin_b \approx begin_c, end_b \approx end_c$  and  $d_{lb} < d_L$  then
        //Az előző vonalkód régió
         $count_b \leftarrow count_b + 1$ 
      else
         $\beta \leftarrow \beta + c$  // felvesszük, mint új régió kezdetét
         $count_c \leftarrow 1$ 
      end if
      for  $b \in \beta$  do
        if  $d_{lb} > d_L$  then
           $count_b \leftarrow count_b - 1$ 
          if  $d_{lb} > 2 * d_L$  then
             $count_b \leftarrow 0$ 
          end if
        end if
        if  $count_b = 0$  then // régió vége
           $score_b \leftarrow \max_{l \in b} count_l$ 
          if  $score_b > \tau_S$  then // ha a minimális # számú vonalat látjuk
             $b$ -t felvesszük, mint vonalkód régiót
          else
             $\beta \leftarrow \beta - b$  // elvetjük a régiót
          end if
        end if
      end for
    end for
  end for
end for
```

3.3. ábra. Vonal ellenőrző algoritmus [3]

**INICIALIZÁLÁS:**

|                                |                             |
|--------------------------------|-----------------------------|
| • = morfológiai zárás          | $Se$ = szerkesztőelem       |
| $\ominus$ = morfológiai erózió | $min_a$ = minimális terület |
| $f$ = szürkeárnyaltos kép      |                             |

---

**ALGORITMUS:**

$$f := (f \bullet Se) - f$$

$$f := f - (f \ominus Se)$$

16 irányban morfológiai nyitás a képen  $0^\circ$  és  $180^\circ$  között

**if** object area  $> min_a$  **then**

Felvesszük a vonalkódot tartalmazó területek közé az objektumot

**else**

Elvetjük az objektumot

**end if**

3.4. ábra. Juett és Qi algoritmusának pszeudokódja [5]

### 3.3. Morfológiai völgydetektáláson alapuló eljárás

Az alábbiakban ismertetett módszerben Juett és Qi [5] fő célkitűzése egy olyan eljárás megalkotása volt, mely minden vonalkód gyanús területet detektál. Az eljárás a morfológiai völgydetektálást vette alapjául.

Az algoritmus az inputban kapott képet, amennyiben az nem szürkeárnyaltos, előbb azt kvantálással azzá alakítja. Ezek után a kép alkalmassá válik a további műveletek elvégzésére. Az előfeldolgozás során először a nem ideális képek javítására, a későbbi feldolgozási fázisokhoz, kontraszt széthúzás végrehajtása történik. Ekkor a kép intenzitás-tartománya egy megadott intervallumba skálázódik. Ezt követően történik a völgydetektálás, mely során a széthúzással kapott kép kivonásra kerül a széthúzott kép morfológiai zárással kapott eredményképéből. A cikk a 720x480-as felbontású képekhez viszonyítva mágikus konstansként adja meg a szerkesztőelem nagyságát. Kevésbé komplex képeken a völgydetektálás eredményképén a vonalkód területén kívüli fals területek száma kisebb, mint gradiens kép esetében a fent tárgyalt eljárásokban. A további műveletek elvégzéséhez a kapott képet bináris képpé alakítva a konkrét detektálás folyamata kezdődhet meg.

Első lépésként a képen található objektumok körvonalának meghatározása szükséges. Ehhez a képet egy ugyancsak a cikkben mágikus konstansként megadott 5x5-ös szerkesztőelemmel erodáljuk, majd az erodálni kívánt képből ezt kivonva megkapjuk a körvonalat.

Az előző lépésben kapott képen ezt követően 16 irányban történő morfológiai nyitás történik 11.25 fokenkénti léptékkel. A vonalkódok 0 és 180 fok közt akármilyen irányultságban elhelyezkedhetnek, egy megfelelő irány megválasztása nehéz. Az eljárás megalkotói ezért vizsgálnak több irányban nem nagy léptékkel. Ami viszont azt ered-

ményezi, hogy az algoritmus futási ideje lassabb lesz. A 16 irány által eredményezett kimeneteket összegezve csak a nagyon vonalkód gyanús területek maradnak a képen. Ezt követően a kép átlagolásra kerül, majd újból bináris képpé alakítva csak a nagyobb területű összefüggő komponensek maradnak a végső műveletekre.

Az utolsó fázisban előbb elimináljuk a képről azokat az objektumokat, melyek területe egy megadott küszöbnél kisebb. Mindezek után a völgydetektált képen az utolsó eredménykép objektumainak középpontját véve megvizsgálásra kerül, hogy léteznek-e egymást követő vonalak a képen, és amennyiben igen, úgy kiszámításra kerülnek az objektum lehetséges sarokpontjai. Amennyiben ez megtörtént, úgy a képen található lehetséges vonalkódok szegmentálásra kerültek. Az algoritmus pszeudokódja a 3.4 ábrán látható.

### 3.4. A javasolt saját módszer

Az eddigi alfejezetekben egy-egy szakirodalmi algoritmust ismertettünk. Most egy új, saját fejlesztésű eljárást mutatunk be vonalkód lokalizálásra. Az algoritmus alapjául a morfológiai képműveletek szolgálnak, melyek együttes alkalmazása esetén kellő hatékonysággal állapítható meg diszkrét képeken a vonalkódok helye. A fő cél egy olyan eljárás kidolgozása volt, mely időben és pontosságban is megfelelő.

Az algoritmus az inputban kapott képet előbb kvantálással szürkeárnyaltos képpé transzformálja, mivel az algoritmusnak az előfeldolgozási fázis során ilyen képtípusra van szüksége. Az előző három cikkhez [3, 4, 5] hasonlóképpen a módszert két fő fázisra bonthatjuk.

Miután kvantálással megkaptuk a szürkeárnyaltos képet, a kép esetleges minőségi romlásán semmiféle művelettel még nem javítottunk, hogy a későbbi feldolgozást segítsük. Az előfeldolgozás során csökkentve a képen lévő zajokat előbb Gauss simítást alkalmazunk, melynek  $\sigma$  értéke minimális. Ezek után a legtöbb vonalkód lokalizáló eljárás során az élkimelést gradiens számítással végzik, de ez esetben morfológiai völgydetektor segítségével tesszük ezt meg. A völgydetektor módszere esetén is az intenzitásértékek közti különbségeket használjuk fel, ezek változása mentén keletkeznek élek. Az erózióhoz alkalmazott szerkesztőelem nagysága bármely képméret esetében fix konstansként került meghatározásra. Noha a völgydetektor használata időben kevésbé optimális, mint az olyan élkimelő operációké, melyek nem irányítottak, a hatékonysága viszont jóval nagyobb. Az eljárás további szakaszaihoz bináris képpé kell alakítanunk a meglévő szürkeárnyaltosat. Ehhez definiáltunk egy küszöbértéket, mely fölötti intenzitásértékkel rendelkező pixeleket fehér pontoknak, míg az alatta elhelyezkedőket fekete pontoknak feleltetjük meg. A megfelelő érték kiválasztásához figyelembe vettük azt, hogy a vonalkód nulla feletti intenzitásértékkel rendelkezik, de nem feltétlen magas az érték.

Az előfeldolgozást követően a digitális kép alkalmassá vált érdemlegesen vonalkódot tartalmazó területek keresésére. Az eddigi képműveletekkel kapott eredménykép még nagyon sok fals régiót tartalmaz. Kihasnálva a vonalkódok azon tulajdonságát, hogy az azt alkotó vonalak egymással párhuzamosan és közel azonos távolságban helyezkednek el egymástól, megvizsgáljuk az egész képet horizontálisan pásztázva, hogy az adott irányban és a megadott távolságban helyezkednek-e el fehér pixelek. A kép felbontásától tettük függővé a maximális távolság értékét.

A pásztázás során több olyan összefüggő komponens is megfelelhetett a kritériumok-

nak, melyek nem valós vonalkód régiók. Mindezen fals területek további csökkentésére az algoritmus következő lépéseként azon objektumokat elimináljuk a képről, melyek területe egy megadott érték alatt van. Maga a küszöbérték (jelölje  $T$ ) a következőképpen adódik: Túl nagy érték választása egy nagyobb felbontású képen lévő kis vonalkód eltávolításához vezethet, ennek következtében az eljárás ezen lépésében a kisebb fals régiókat távolítjuk el.

A tapasztalatok alapján a vonalkód gyanús területeket mellett leginkább sűrű szöveget tartalmazó területek maradnak a detektálás későbbi szakaszaira. Mivel a kódot alkotó vonalak egymástól valamilyen távolságban helyezkednek el, így nagy valószínűséggel nem alkotnak egy összefüggő objektumot, de legalábbis nem az összes vonal. Emiatt a fázis ezen részében dilatációt alkalmazunk, mellyel az eddig nem összefüggő régiók is egybe tartozhatnak. Mindazonáltal ezen lépéssel más, nem vonalkód területeket is hizlalhatunk. A dilatációhoz használt téglalap alakú szerkesztőelem nagyságát (jelölje  $S$ ) a következőképpen definiáltuk:

$$S = \max(40, \text{vonalkód szélessége} * 2)$$

A morfológiai dilatáció olyan területeket is felhizlalt, melyek szükségtelenek voltak, illetve összeolvaszthatott nem kívánt területeket is. Az alább felmerült problémák kezelésére a dilatáció duális műveletét alkalmazzuk, vagyis erodálunk.

Az erodáláshoz alkalmazott szerkesztőelem ugyancsak téglalap alakú, következően abból, hogy a vonalkód is téglalap alakú, legalábbis az 1D-s kódok. A szerkesztőelem nagysága az dilatációhoz használthoz képest kisebb, az előző lépésben használt szerkesztőelem nagyságának  $1/3$  része lesz. Az eróziót követően azon területek lesznek megtalálhatóak a képen, melyek lehetséges, hogy vonalkódot tartalmazó területek. Természetesen előfordulnak még ezek után is fals pozitív objektumok, melyeket az algoritmus utolsó lépéseként eltávolítunk. Mindezt úgy tesszük, hogy megvizsgáljuk az eredményképpen maradt objektumok területeit, arányait, majd ezekből következtetve egy küszöbszámot meghatározva töröljük a képről azon objektumokat, melyek a megadott érték alatti területtel rendelkeznek.

Az algoritmus pszeudokódja a 3.5 ábrán látható.

---

**INICIALIZÁLÁS:**

|                                               |                                                     |
|-----------------------------------------------|-----------------------------------------------------|
| $N$ = kép szélessége                          | $M$ = kép magassága                                 |
| $f$ = szürkeárnyaltos kép                     | $i_t$ = minimális küszöbérték                       |
| $\bullet$ = morfológiai zárás                 | $Se_1$ = völgydetektáláshoz használt szerkesztőelem |
| $min_a$ = minimális terület                   | $max_d$ = maximális távolság                        |
| $\oplus$ = morfológiai dilatació              | $\ominus$ = morfológiai erózió                      |
| $Se_2$ = dilatacióhoz használt szerkesztőelem | $Se_3$ = erózióhoz használt szerkesztőelem          |

---

**ALGORITMUS:**

```

 $f := (f \bullet Se_1) - f$ 

for  $i := 0, \dots, N$  do
  for  $j := 0, \dots, M$  do
    if  $f_{i,j} > i_t$  then
      if  $f_{i,j} < max_d$  then
         $f_{i,j} \leftarrow 1$ 
      else if
        then  $f_{i,j} \leftarrow 0$  // nincs él a megadott maximális távolságon belül
      end if
    else if
      then  $f_{i,j} \leftarrow 0$ 
    end if
  end for
end for

 $f := f \oplus Se_2$ 
 $f := f \ominus Se_3$ 

if object area  $> min_a$  then
  Felvesszük a vonalkódot tartalmazó területek közé az objektumot
else
  Elvetjük az objektumot
end if

```

3.5. ábra. A javasolt saját algoritmus pszeudokódja

## 4. fejezet

# A tárgyalt algoritmusok összehasonlító elemzése

Az utolsó fejezetben a korábban ismertetett algoritmusok hatékonyságát vetjük össze adott jellemzők mellett. A dolgozat ezen részében átfogó képet adunk az egyes algoritmusok különböző torzításokkal terhelt képeken való viselkedéséről, illetve a tanulmányozzuk, hogy a hatékonyságukat egyes jellemzők hogyan módosítják. A későbbiekben az egyes algoritmusokra való hivatkozáshoz betűszavakat vezettünk be, melyek a következők: **ETJC** (Ender Telkin és James M. Coughlan eljárása), **TT** (Timothy R. Tuinstra eljárása), **JJXQ** (James Juett és Xiaojun eljárása), illetve a **Saját**, az általunk kidolgozott új algoritmus.

### 4.1. Tesztsorozat, tesztkörnyezet és implementáció

Digitálisan tesztképeket generáltunk az 1.2-es fejezetben bemutatott 1D-s vonalkódtípusokat alapul véve. Mindehhez elegendő volt az egyes típusú kódokból egyet kiválasztani. A tesztképek egy vagy több vonalkódot tartalmaznak az egyes típusokból, illetve különféle torzításokkal terheltek. Az általunk generált képek a következő tulajdonságokkal rendelkeznek: a képeket  $15^\circ$ -onként elforgattunk  $0^\circ$ -tól  $180^\circ$ -ig, Gauss elmosást alkalmaztunk  $3 \times 3$ -as szerkesztőelemmel és 6 különböző  $\sigma$ -val, illetve additív zajt is hozzáadtunk a képekhez 10% -os léptékkal 0% és 50% intervallumban. Összegezve tehát 8 különféle típusú vonalkódból 12-féle orientációban, 6 különböző Gauss szűrővel és 6 eltérő mértékű additív zajjal terhelve, összesen 20736 képet generáltunk.

További 100 vonalkódot tartalmazó valós életbeli képet is felhasználtunk. Ezen képeken semmiféle manipulációt nem hajtottunk végre, de torzításokkal terheltek, nem ideálisak.

Az eljárásokat MATLAB-ban implementáltuk, Image Processing Toolbox segítségével. A kiértékelést egy Pentium(R) Dual-Core 2.30Ghz CPU-val rendelkező számítógéppel hajtottuk végre.

## 4.2. Eredmények vizsgálata

Jelen alfejezetben azt tárgyaljuk, hogy az egyes algoritmusok egymáshoz viszonyítva milyen hatékonyak voltak különböző jellemzőkkel ellátott képeken, továbbá milyen futási időt produkáltak. A tesztelés során minden egyes tesztképet illetően, általunk előre meghatározott információk kerültek kinyerésre, melyek a jelen összehasonlítás alapjául szolgálnak. A pontosság kiszámításához a Jaccard-féle hasonlósági együtthatót használtuk, ahol a tényleges és a detektált kód régiók befoglaló téglalapjának átfedését vizsgáltuk. Az együttható az alábbiképpen írható fel:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

ahol A a valódi vonalkód, míg B a detektált kód befoglaló téglalapját jelöli.

Megállapítható, hogy mind a négy algoritmus hatékonyan képes vonalkód detektálásra. ETJC eljárása gyors, azonban, amennyiben egy képen több vonalkód található, és zajjal terhelt, úgy hatékonyságából veszít. Mint már említettük, a képgeneráláskor Gauss simítást is végeztünk, ezért az ilyen képeken nem módosul érdemben az algoritmus hatékonysága.

TT algoritmusának futási idejét tekintve mindhárom bemutatott algoritmusnál lassabb. A futás során több ízben 4 különböző orientációban ( $0^\circ, 45^\circ, 90^\circ, 135^\circ$ ) kerül pásztázásra a kép. Valószínűsíthető, hogy ekkor veszít a gyorsaságából, mely az előfeldolgozás fázisában volt.

A völgydetektálást alapul vevő JJXQ módszere elmarad futási időt tekintve a dolgozatban ismertetett gyors algoritmusokhoz képest. Az eljárás lassabb mivoltja abban rejlik, hogy a detektálás fázisában 16 különféle orientációban,  $11.25^\circ$ -onként kerül bejárásra a kép. Ez ugyan nagyobb eredményességet hoz az eljárásnak, de a futási idő rovására. A módszer teljes egészében morfológiai képműveleteket alkalmaz.

A saját algoritmusunkat tekintve is elmondhatjuk, hogy nagyon gyors. Az esetek nagy részében a gyorsabb eljárások kevésbé pontosan, vagy egyáltalán nem képesek meghatározni a vonalkódok helyét. Mondhatni a gyorsaság ára a pontosság csökkenése. A mi esetünkben a gyorsasággal nem párosul nagy mértékű kár. A tesztelési adatokat kiértékelve megállapítható volt, hogy ezen eljárás az összes tesztképre nézve a legjobb futási időt produkálta, bár pontosságban természetesen nem minden esetben volt a legjobb. A módszer gyengeségeként szerepel, hogy túl zajos képeken, illetve a képen vonalkódhoz hasonló területek megjelenése esetén fals detektálás történhet. Esetenként a vonalkód nem megtalálása is előfordul. Ideális, és nem extrém esetekben az algoritmus nagy hatékonyságú.

Korábban említettük, hogy a generált tesztképek különböző mértékű hozzáadott zajt, simítást, illetve képenként egy vagy több vonalkódot is tartalmaznak. A dolgozat további részében ezeket a jellemzőket vizsgáljuk az egyes eljárások esetében. A 4.1 táblázatban összesítve látható az adott jellemzők mellett az egyes eljárások pontossága. A módszerek hatékonyságának megítélésben fontos szempont a futási idő. A 4.2 táblázatban eljárásonként futtatva láthatóak a futási idők egy vonalkódot, illetve három vonalkódot tartalmazó képek esetében. Az egyes eljárások futási ideje közt láthatóan különbség mutatkozik.

A következőkben az algoritmusok viselkedését vizsgáljuk egy, illetve három vonalkódot tartalmazó képek esetében. Az egyes vonalkód típusokat változatos szerkezeti felépítés jellemzi, ennek következtében a lokalizálásuk sem ugyanolyan hatékonyságú. Lát-

4.1. táblázat. Az algoritmusok különféle torzítások melletti/nélküli pontossága (átlag  $\pm$  szórás %-ban kifejezve)

|                   | TT              | ETJC           | JJXQ            | Saját          |
|-------------------|-----------------|----------------|-----------------|----------------|
| simított          | 82.8 $\pm$ 3.5  | 88.4 $\pm$ 4.6 | 68.9 $\pm$ 13.2 | 95.4 $\pm$ 4.8 |
| simítás nélkül    | 91.2 $\pm$ 12.6 | 88.9 $\pm$ 4.2 | 92.4 $\pm$ 5.1  | 97.6 $\pm$ 3.0 |
| hozzáadott zajjal | 82.8 $\pm$ 1.8  | 88.3 $\pm$ 4.4 | 64.6 $\pm$ 5.8  | 95.6 $\pm$ 3.8 |
| zaj nélkül        | 90.4 $\pm$ 12.8 | 89.5 $\pm$ 4.6 | 89.1 $\pm$ 16.3 | 96.4 $\pm$ 4.4 |

4.2. táblázat. Az algoritmusok futási ideje (átlag  $\pm$  szórás)

|                  | TT              | ETJC            | JJXQ            | Saját           |
|------------------|-----------------|-----------------|-----------------|-----------------|
| egy kód esetén   | 0.23 $\pm$ 0.13 | 0.65 $\pm$ 0.22 | 0.80 $\pm$ 0.44 | 0.13 $\pm$ 0.10 |
| három kód esetén | 0.54 $\pm$ 0.36 | 1.72 $\pm$ 0.78 | 1.82 $\pm$ 0.77 | 0.18 $\pm$ 0.09 |

hatóan a 4.3 táblázatban, melyben egy kódot magában foglaló kép adatait vizualizáltuk, hogy az algoritmusok a Code128 típusú kódokat detektálták legkevesebb eredményesen.

Amennyiben megvizsgáljuk mindezt három kódot tartalmazó képek esetében is, úgy elmondható, hogy ez esetben már egyetlen típus sem emelhető ki, melyre lényegesen gyengébben működnének az algoritmusok, mint ahogyan ez látható a 4.4 táblázatban is.

A képek különféle torzításokkal terhelt eseteiben az algoritmusok veszítenek a hatékonyságukból. A következőkben azt vizsgáljuk, hogy az egyes torzításokat egyenként tekintve milyen mértékű ez a veszteség. Előbb a simított képeken figyeljük meg az egyes mértékek mellett hogyan viselkednek a módszerek. Ez esetben is külön tárgyaljuk az egy kódot, illetve a három kódot tartalmazó képeket. Látható, hogy a képeket simítva az eljárások nem vesztenek drasztikusan a hatékonyságukból. Mindez a 4.5 táblázatból leolvasható.

4.3. táblázat. Az algoritmusok vonalkód típusonkénti pontossága egy vonalkódot tartalmazó képek esetén (átlag  $\pm$  szórás)

|          | TT              | ETJC            | JJXQ           | Saját           |
|----------|-----------------|-----------------|----------------|-----------------|
| Codabar  | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 | 99.2 $\pm$ 0.1 | 100.0 $\pm$ 0.0 |
| Code128  | 46.6 $\pm$ 0.2  | 85.6 $\pm$ 0.4  | 88.5 $\pm$ 0.3 | 95.4 $\pm$ 0.1  |
| Code39   | 100.0 $\pm$ 0.0 | 95.8 $\pm$ 0.1  | 94.3 $\pm$ 0.1 | 100.0 $\pm$ 0.0 |
| EAN-13   | 100.0 $\pm$ 0.0 | 96.0 $\pm$ 0.1  | 91.3 $\pm$ 0.2 | 100.0 $\pm$ 0.0 |
| EAN-8    | 100.0 $\pm$ 0.0 | 99.5 $\pm$ 0.1  | 85.5 $\pm$ 0.6 | 100.0 $\pm$ 0.0 |
| I2of5    | 99.9 $\pm$ 0.1  | 85.4 $\pm$ 0.2  | 92.2 $\pm$ 0.1 | 94.5 $\pm$ 0.1  |
| UPC-A    | 100.0 $\pm$ 0.0 | 98.6 $\pm$ 0.1  | 94.5 $\pm$ 0.3 | 86.5 $\pm$ 0.1  |
| UPC-E    | 85.8 $\pm$ 0.1  | 90.8 $\pm$ 0.1  | 74.6 $\pm$ 0.1 | 100.0 $\pm$ 0.0 |
| Összesen | 92.8 $\pm$ 0.1  | 86.4 $\pm$ 0.1  | 78.8 $\pm$ 0.2 | 96.5 $\pm$ 0.1  |

4.4. táblázat. Az algoritmusok vonalkód típusonkénti pontossága három vonalkódot tartalmazó képek esetén (átlag  $\pm$  szórás %-ban kifejezve)

|          | TT              | ETJC            | JJXQ            | Saját           |
|----------|-----------------|-----------------|-----------------|-----------------|
| Codabar  | 83.6 $\pm$ 0.1  | 56.0 $\pm$ 0.1  | 58.4 $\pm$ 0.8  | 92.2 $\pm$ 0.1  |
| Code128  | 83.7 $\pm$ 0.3  | 99.5 $\pm$ 0.1  | 79.3 $\pm$ 0.2  | 78.5 $\pm$ 0.1  |
| Code39   | 83.2 $\pm$ 0.2  | 91.2 $\pm$ 0.1  | 65.6 $\pm$ 0.1  | 94.9 $\pm$ 0.1  |
| EAN-13   | 83.3 $\pm$ 0.2  | 95.4 $\pm$ 0.2  | 76.0 $\pm$ 0.1  | 100.0 $\pm$ 0.0 |
| EAN-8    | 83.3 $\pm$ 0.1  | 100.0 $\pm$ 0.0 | 82.6 $\pm$ 0.4  | 100.0 $\pm$ 0.0 |
| I2of5    | 78.9 $\pm$ 0.1  | 39.8 $\pm$ 0.1  | 100.0 $\pm$ 0.0 | 99.8 $\pm$ 0.1  |
| UPC-A    | 73.5 $\pm$ 0.1  | 65.5 $\pm$ 0.1  | 97.0 $\pm$ 0.1  | 100.0 $\pm$ 0.0 |
| UPC-E    | 100.0 $\pm$ 0.0 | 79.6 $\pm$ 0.1  | 100.0 $\pm$ 0.0 | 99.9 $\pm$ 0.1  |
| Összesen | 74.1 $\pm$ 0.2  | 74.7 $\pm$ 0.1  | 76.5 $\pm$ 0.2  | 95.6 $\pm$ 0.1  |

4.5. táblázat. Az algoritmusok pontossága különböző simítások mellett egy vonalkódot tartalmazó képek esetén (átlag  $\pm$  szórás %-ban kifejezve)

|          | TT              | ETJC            | JJXQ            | Saját           |
|----------|-----------------|-----------------|-----------------|-----------------|
| 0        | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 |
| 0.5      | 90.6 $\pm$ 0.1  | 89.6 $\pm$ 0.1  | 90.6 $\pm$ 0.3  | 98.8 $\pm$ 0.1  |
| 1        | 90.6 $\pm$ 0.1  | 88.5 $\pm$ 0.1  | 90.6 $\pm$ 0.1  | 97.8 $\pm$ 0.1  |
| 1.5      | 90.5 $\pm$ 0.1  | 86.5 $\pm$ 0.1  | 90.5 $\pm$ 0.1  | 97.5 $\pm$ 0.1  |
| 2        | 88.9 $\pm$ 0.2  | 86.5 $\pm$ 0.1  | 89.5 $\pm$ 0.1  | 97.5 $\pm$ 0.1  |
| 2.5      | 86.2 $\pm$ 0.2  | 79.6 $\pm$ 0.1  | 87.5 $\pm$ 0.1  | 96.8 $\pm$ 0.1  |
| Összesen | 90.4 $\pm$ 0.1  | 88.5 $\pm$ 0.1  | 91.1 $\pm$ 0.2  | 96.4 $\pm$ 0.1  |

4.6. táblázat. Az algoritmusok pontossága különböző simítások mellett három vonalkódot tartalmazó képek esetén (átlag  $\pm$  szórás %-ban kifejezve)

|          | TT              | ETJC            | JJXQ            | Saját           |
|----------|-----------------|-----------------|-----------------|-----------------|
| 0        | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 |
| 0.5      | 93.3 $\pm$ 0.1  | 94.5 $\pm$ 0.2  | 92.8 $\pm$ 0.3  | 97.3 $\pm$ 0.1  |
| 1        | 91.3 $\pm$ 0.1  | 86.3 $\pm$ 0.2  | 91.7 $\pm$ 0.1  | 97.3 $\pm$ 0.1  |
| 1.5      | 91.2 $\pm$ 0.1  | 86.3 $\pm$ 0.1  | 91.7 $\pm$ 0.1  | 97.3 $\pm$ 0.1  |
| 2        | 86.3 $\pm$ 0.4  | 82.1 $\pm$ 0.1  | 89.3 $\pm$ 0.1  | 97.3 $\pm$ 0.0  |
| 2.5      | 86.3 $\pm$ 0.1  | 82.0 $\pm$ 0.1  | 89.2 $\pm$ 0.1  | 96.7 $\pm$ 0.1  |
| Összesen | 82.3 $\pm$ 0.2  | 77.4 $\pm$ 0.2  | 91.3 $\pm$ 0.1  | 97.3 $\pm$ 0.1  |

4.7. táblázat. Az algoritmusok pontossága különböző zajszintek mellett egy vonalkódot tartalmazó képek esetén (átlag  $\pm$  szórás %-ban kifejezve)

|          | ETJC            | TT              | JJXQ            | Saját           |
|----------|-----------------|-----------------|-----------------|-----------------|
| 0        | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 |
| 10       | 89.6 $\pm$ 0.1  | 89.6 $\pm$ 0.1  | 88.0 $\pm$ 0.2  | 96.9 $\pm$ 0.1  |
| 20       | 88.9 $\pm$ 0.1  | 87.8 $\pm$ 0.1  | 81.9 $\pm$ 0.2  | 96.6 $\pm$ 0.1  |
| 30       | 88.9 $\pm$ 0.1  | 84.4 $\pm$ 0.2  | 74.7 $\pm$ 0.1  | 96.8 $\pm$ 0.1  |
| 40       | 88.3 $\pm$ 0.1  | 83.7 $\pm$ 0.1  | 73.0 $\pm$ 0.3  | 96.7 $\pm$ 0.1  |
| 50       | 87.6 $\pm$ 0.1  | 83.7 $\pm$ 0.1  | 66.0 $\pm$ 0.3  | 95.4 $\pm$ 0.1  |
| Összesen | 89.4 $\pm$ 0.1  | 87.3 $\pm$ 0.1  | 82.5 $\pm$ 0.3  | 96.6 $\pm$ 0.1  |

Bár megfigyelhető fordított viszony a  $\sigma$  és a hatékonyság közt, de ez nem számottevő. Viszont szembevetendő, hogy a völgydetektáláson alapuló JJXQ eljárása és a saját algoritmusunk nagyobb hatékonyságra képes simított képek esetén a másik kettővel szemben. Mindez megfigyelhető nem csak egy kódot tartalmazó képeknél. Érdekes megfigyelni a 4.6 táblázatban, hogy némely esetekben nagyobb hatékonysággal kerülnek detektálásra a vonalkódok több kódot tartalmazó képeken.

Miután elemeztük az algoritmusokat a simítás függvényében, tanulmányozzuk azokat a különböző hozzáadott zajszintek mellett is, hasonlóan az előzőekhez, külön egy és három kód esetén. Megállapítható a 4.7 és 4.8 táblázatok alapján, hogy az algoritmusok mindegyike zajérzékeny. A hozzáadott zaj növekedésével erősen csökken a hatékonyságuk az ismertetett irodalmi eljárásoknak, míg a saját módszer esetében csekély mértékű a veszteség.

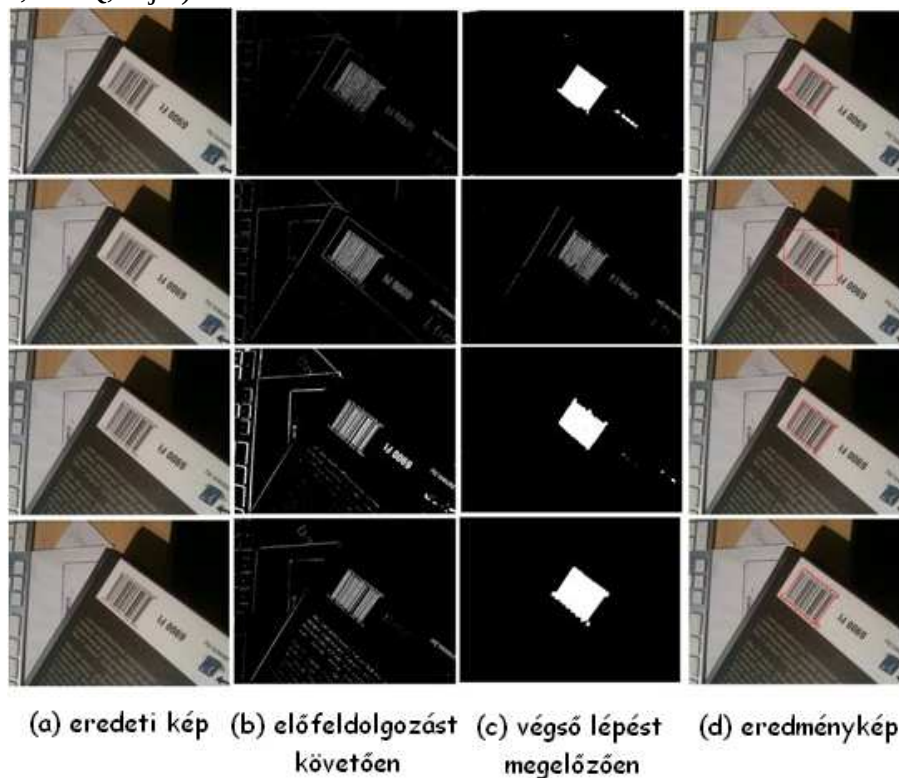
A 4.1 ábrán ugyanazon képre futtatva az algoritmusok mindegyikét, azok főbb fázisaiban készült képeit prezentáltuk. A választott kép (a) nem ideális, zajjal terhelt. A

4.8. táblázat. Az algoritmusok pontossága különböző zajszintek mellett három vonalkódot tartalmazó képek esetén (átlag  $\pm$  szórás %-ban kifejezve)

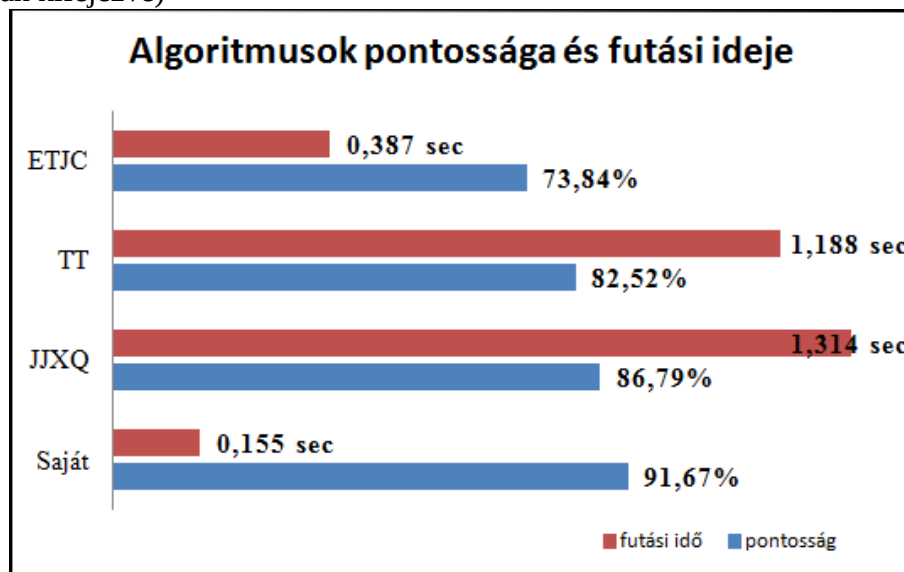
|          | ETJC            | TT              | JJXQ            | Saját           |
|----------|-----------------|-----------------|-----------------|-----------------|
| 0        | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 | 100.0 $\pm$ 0.0 |
| 10       | 93.3 $\pm$ 0.1  | 97.5 $\pm$ 0.2  | 91.0 $\pm$ 0.2  | 96.9 $\pm$ 0.1  |
| 20       | 87.2 $\pm$ 0.2  | 87.9 $\pm$ 0.1  | 87.6 $\pm$ 0.1  | 97.1 $\pm$ 0.1  |
| 30       | 84.7 $\pm$ 0.2  | 83.9 $\pm$ 0.2  | 81.0 $\pm$ 0.1  | 96.8 $\pm$ 0.1  |
| 40       | 84.3 $\pm$ 0.2  | 83.5 $\pm$ 0.2  | 76.3 $\pm$ 0.2  | 96.8 $\pm$ 0.1  |
| 50       | 74.3 $\pm$ 0.1  | 81.7 $\pm$ 0.1  | 63.4 $\pm$ 0.1  | 91.3 $\pm$ 0.1  |
| Összesen | 89.4 $\pm$ 0.1  | 87.3 $\pm$ 0.1  | 82.5 $\pm$ 0.3  | 96.6 $\pm$ 0.1  |

(b) képen az előfeldolgozást követően az eljárások eredményképei találhatóak. Ezeket megvizsgálva nem vonható le fontos konklúzió az algoritmusokat illetően, mivel az előfeldolgozás során még csak előkészítésre kerültek a képek a későbbi képműveletekre. A következő (c) képen a végső lépést megelőző szegmentálás eredményeit jelenítettük meg. Jól látható, hogy ETJC, TT eljárások esetében szinte már csak a vonalkód, míg a saját eljárást tekintve csak a vonalkód található a képen. A (d) képet tanulmányozva megfigyelhető, hogy az egyes eljárások közelítőleg milyen pontossággal képesek meghatározni a vonalkódok helyét.

4.1. ábra. Az algoritmusok egyes főbb fázisaiban készült képek (fentről lefele haladva: ETJC, TT, JJXQ, saját)



4.2. ábra. Összefoglaló diagram az összes tesztképre illetően (futási idő sec-ben, pontosság %-ban kifejezve)



Az általunk generált képek maximális mérete 800x800-as volt. Ezen felbontás mellett is megállapítható, hogy TT és JJXQ algoritmusai időben nem veheti fel a versenyt a másik két algoritmusmal szemben. A saját eljárásunk futási időben nem sokkal gyorsabb 800x800-as képek esetén ETJC algoritmusához képest, de futtatva őket egy 2500x1722-es képre az ETJC algoritmusai 3.2 mp alatt futott le, míg a saját algoritmus esetében 1.3 mp elég volt a képen található vonalkód detektálásához.

Tehát megállapítható, hogy nagyobb felbontású képek esetében már számottevő az algoritmusok közti futási idő különbség. A legfontosabb jellemző, mely ugyancsak leolvasható a diagramról, hogy a saját algoritmusunk hatékonysága jóval nagyobb, mint a többi ismert algoritmusé. Beleértve a pontosságot és a futási időt is.

A 4.2 ábrán szemléltetjük az egyes algoritmusok átlagos futási idejét és pontosságát az összes tesztképre. Látható, hogy az előfeldolgozás során morfológiai völgydetektálást alkalmazó algoritmusok pontossága nagyobb, mint a másik kettőé, viszont futási időt tekintve eltérőek. A 4.1 táblázatot tekintve az is egyértelmű, hogy JJXQ eljárása a kép minőségi romlásával egyenesen arányosan pontosságából is vesz, míg a saját algoritmusunk sokkal nagyobb pontossággal képes még ekkor is a vonalkódot lokalizálni. A diagramból ugyan nem olvasható le, de a tesztek kimutatták, minden esetben a kép méretével együtt szignifikánsan nőtt a futási idejük az algoritmusoknak.

# Összefoglalás

A dolgozatban ismertettük a vonalkód detektálás három szakirodalomban található módszerét, valamint egy új, saját fejlesztésű algoritmust. Megállapítottuk azt is, hogy az egyes algoritmusok nem specializálódtak egyes kódtípusokra, hanem hatékonyan detektálnak különféle típusú 1D-s vonalkódokat. Az eljárások pontosságának és futási idejének vizsgálatára létrehoztunk egy több mint 21000 tesztképet magában foglaló adatbázist.

Az algoritmusok pontosságát vizsgáltuk különféle szempontokat tekintve. A digitálisan generált képekhez különböző mértékű simítást, illetve zajt rendeltünk. A tesztképeken található kódok elforgatásra is kerültek, de a későbbiekben a tesztelés során világossá volt, hogy ezen algoritmusok egyike sem orientációfüggő, így ez az összehasonlítás során nem került elemzésre. A tesztelés során megmutatkozott az is, hogy az irodalmi algoritmusok nagyon zajérzékenyek, a saját algoritmusunk is veszít hatékonyságából, de nem akkora mértékben, mint a többi tárgyalt eljárás. Tanulmányozva a futási időket és az algoritmusok pontosságát, beláttuk, hogy a saját algoritmusunk hatékonysága nagyobb mind ideális képek esetében, mind a képek különböző torzításokkal terhelt eseteiben, mint a többi ismertett irodalmi eljárásé. Ezen túlmenően az futási idők eredményeit tekintve, valós idejű implementáció is kivitelezhetővé vált, mely megvalósításra is került.

A jövőbeli tervek közt szerepel a saját módszer kiterjesztése 2D-s vonalkódok esetére is, noha ez esetben a probléma sokkal komplikáltabb, így várhatóan módosításokra lesz szükség hasonló pontosság és/vagy sebesség eléréséhez.

# Irodalomjegyzék

- [1] Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing*, Prentice Hall, 3rd Edition, 2007.
- [2] Frank Y. Shih *Image Processing and Mathematical Morphology: Fundamentals and Applications*, CRC Press,
- [3] Ender Telkin, James M. Coughlan, „An Algorithm Enabling Blind Users to Find and Read Barcodes” *In Applications of Computer Vision (WACV), 2009 Workshop on*, pages 1-8, dec. 2009.
- [4] Timothy R. Tuinstra, „Reading Barcodes from Digital Imagery”, *PhD thesis, Cedarville University*, 2006.
- [5] James Juett, Xiaojun Qi, „Barcode localization using bottom hat filter” *Bemutatva a RUA09 nyári iskolában, elérhető a <http://digital.cs.usu.edu/~xqi/Teaching/REU09/Website/James/jjuett.html> webcímen*
- [6] Roger C. Palmer *The Bar Code Book: Reading, Printing, and Specification of Bar Code Symbols*, Helmers Pub, 3rd edition (November 1995)

# Köszönetnyilvánítás

Ezúton szeretném megköszönni témavezetőmnek, Dr. Nyúl Lászlónak, hogy tanácsaival és türelmével segítette dolgozatom elkészülését.