

Debreceni Egyetem

Informatikai Kar

Automatikus Dokumentum-feldolgozás

Szakdolgozat

Készítette:

Nagy Lajos

programozó matematikus

Témavezető:

Dr. Fazekas Attila

egyetemi docens

Debrecen
2007

Tartalomjegyzék

1. Bevezetés	3
2. Képfeldolgozásról történeti áttekintése, területei	4
3. Alapismeretek	6
3.1. A papír alapú dokumentum és sajátosságai	6
3.2. Digitalizálási folyamat, szkennerek	6
3.3. A digitalizált kép és jellemzői	7
3.4. Színrendszerek	8
3.5. Grafikus formátumokról általában, a BMP formátum	10
4. A feldolgozó program megvalósítása	11
4.1. Fejlesztőkörnyezet kiválasztása	11
4.2. A program feladata, célkitűzése	11
4.3. A programmal szemben támasztott követelmények	16
4.4. A program működése és algoritmusai	17
4.4.1. Dokumentum betanítása	18
4.4.1.1. Alapadatok megadása	18
4.4.1.2. Adatmezők felvitele	19
4.4.1.3. Adatmezők grafikus bejelölése a sablonon	20
4.4.1.4. Az ADF fájl felépítése	22
4.4.2. Dokumentum feldolgozása	24
4.4.2.1. Betanított dokumentumtípus megnyitása	25
4.4.2.2. Adatmezők áttekintése	26
4.4.2.3. Grafikus fájlok létezésének ellenőrzése	27
4.4.2.4. Felismerés lépései	28
4.4.2.4.1. Két szintre vágás	29
4.4.2.4.2. Egymásra illesztés	29
4.4.2.4.3. Kerettörlés	30
4.4.2.4.4. Szegmentálás	31
4.4.2.4.5. A tényleges felismerés	31
4.4.2.4.5.1. Checkbox felismerése	32
4.4.2.4.5.2. Számmező felismerése Walsh-táblázat segítségével.....	32
4.4.2.5. Eredmény mentése	36
5. Összegzés, konklúzió, további tervek	37
6. Irodalomjegyzék	38
7. Köszönetnyilvánítás	39

1. Bevezetés

Felgyorsult világunkban folyamatosan keressük azokat a lehetőségeket, melyekkel mindennapi életünket egyszerűbbé, zökkenőmentesebbé tehetjük. A gépek, legfőképp a számítógépek megjelenésük óta folyamatosan vesznek át egyre több feladatot az embertől, kényelmesebbé, gyorsabbá téve munkánkat, kutatásainkat. A számítástechnika rohamos fejlődésével egyre nagyobb távlatok nyílnak meg, a gépek számítási teljesítménye évről évre exponenciálisan növekszik. Ezáltal lehetőség nyílt olyan területek automatizálására is, ahol ez a számítási teljesítmény alapkövetelmény, ilyen például a szakdolgozatomban is ismertetett terület, a képfeldolgozás.

A képfeldolgozást alkalmazó technikával ma már számos helyen találkozhatunk. A rendszámleolvasó rendőrségi rendszerektől kezdve a sakktáblán elhelyezkedő bábuk formáját és helyét felismerő robotokig át, az orvosi képfeldolgozó műszereken keresztül az ujjlenyomatot felismerő módszerekig. Vagy gondoljuk arra a talán nem is olyan szembetűnő alkalmazási területre, ahol a papír alapú dokumentumok elektronikus feldolgozása történik. Az automatikus dokumentum-feldolgozó rendszerekre. Első ránézésre talán nem is gondoljuk, nem is feltétlenül sejtjük, de ha kicsit utánajárunk itt is megfigyelhetjük a háttérben megbújó csúcstechnikai megoldások és a bonyolult matematikai számítások, összefüggések jelenlétét, szoros kapcsolatát. Egy vállalati kérdőív az alkalmazottak megelégedettségéről vagy egy adott tárgyból vizsgáló hallgatók tudásszintjének leméréséről és kiértékeléséről szóló dokumentumok feldolgozása emberi munkaerővel számolva igen időigényes feladat. Vagy gondoljunk egy hivatalos nyomtatványra, melyet nem csak egy 50-100 fős hallgatói vagy munkahelyi szintfelmérésre használnak, hanem amelyet akár több tízezer ember, a lakosság jelentős része is rendszeresen kitölthet és a megfelelő hatóságnak nyújt be. Ilyen lehet például egy adóbevallás is. Ekkor már mindenképpen számottevő tényező az idő, melyet a feldolgozásra kell fordítanunk. Ezen a ponton lépnek be azok a dokumentum-feldolgozó rendszerek, melyek átveszik az embertől ezt a borzasztóan időigényes feladatot, elvégzik helyettünk a feldolgozás és kiértékelés folyamatát, s a kész eredményt a megfelelő formában tárják elénk.

Szakdolgozatom alapjául szolgáló program is egy ilyen kutatás eredménye, mely azt a célt hivatott szolgálni, hogy a papír alapon elkészített és kitöltött dokumentumokat, formanyomtatványokat a lehető legrövidebb idő alatt lehessen elektronikusan, számítógép segítségével feldolgozni, majd abból a számunkra szükséges értékes információt kinyerni. Ezen kutatásom célkitűzése, hogy minél jobb, pontosabb és megbízhatóbb módszereket dolgozzak ki a lehető legtökéletesebb dokumentum-feldolgozás elérése érdekében.

Szakdolgozatommal a képfeldolgozás egy olyan területére szeretnék fókuszálni, amelynek megvalósítása ugyan már régebben felmerült az igények között, sőt alapvető funkcióját már meg is valósították, mégis egyedi elképzelés szerint, kutatás jelleggel kissé más szemszögből és specifikusan közelítem a problémát.

2. Képfeldolgozás történeti áttekintése, területei

A mai világban a fő gondot az okozza, hogy a számítógépes képfeldolgozásnak nincs egységes elmélete. A felmerülő problémákra az esetek túlnyomó részében eseti megoldásokat dolgoztak ki, és csak a legutóbbi időkben kezdtek az általánosításra törekedni. Emiatt egy-egy feladatra sokszor több megoldás is létezik, amelyek különböző szempontokból előnyösek. A nehézség gyakran abban van, hogy kiválasszuk a számunkra a legjobbat.

A korai képfeldolgozási megoldások után az 50-es évek második felében jelentek meg az irodalomban az első cikkek, amelyek a képi információk feldolgozásával foglalkoztak. Azóta a számítógépes képfeldolgozás témaköreinek skálája jelentősen kibővült és a 80-as évek elejére önálló alkalmazási területté fejlődött.

A fejlődési folyamat természetéből következően először az optikai és az analóg jelfeldolgozás (főleg az optikai képfeldolgozás) módszereit adaptálták. Tömegesen születtek az egyes konkrét feladatokra kidolgozott heurisztikus módszerek, amelyeknek érvényességi feltételeit és konvergenciáját sokszor nem is vizsgálták. A 60-as évek végére már elég nagy volt a káosz ahhoz, hogy megindulhasson a matematikai módszerek intenzív kutatása.

Az évtized második felétől hatalmas erővel megindult a speciális célú hardver eszközök fejlesztése (nagy felbontású képfelvevő, tároló- és megjelenítő eszközök, nagy

sebességű (párhuzamos) feldolgozó egységek, a képfeldolgozás alapfeladatait megvalósító LSI elemek, stb.). Ezzel egyidejűleg megjelentek az első univerzális célú rendszerek, amelyek az emberi látórendszert próbálták többé-kevésbé megközelíteni. Pillanatok alatt kiderült azonban, hogy ez olyan bonyolult feladat, amelyet próbálgatással nem lehet megoldani.

Ez a felismerés indította el az elméleti alapok kidolgozására irányuló nagyszabású kutatómunkát, amelyet az időközben társadalmi szükségletté lett számítógépes képfeldolgozás iránti egyre magasabb igények is ösztönöztek. A munka, amelynek az 5. generációs távlati terv további lendületet adott, napjainkban is folyik, ez idő szerint azonban az egységes elmélet még nem született meg.

A számítógépes képfeldolgozáson most kizárólag kétdimenziós (síkbeli), digitális képek gépi feldolgozását értjük. A digitális képfeldolgozás a természetes képeken foglalt képi információ megértésére irányul. A folyamatot a rész-célkitűzések szerint három szintre bonthatjuk.

a.) Előfeldolgozáson a digitális kép létrehozását, valamint a leképzési hibák kijavítását, a jellemző képtulajdonságok kiemelését és a képnek a további feldolgozásra kedvezőbb, adekvát átalakítását értjük.

b.) A képosztályozás során megkeressük a képen előforduló alakzatokat, illetve a képet jellemző különféle sajátosságokat. Az ezekhez mindig rendelhető numerikus értékekkel vagy a köztük lévő relációkkal leírjuk, majd osztályozzuk a képet, illetve egyes részeit. Fontos megjegyezni, hogy ez a leírás lényegesen eltérhet a grafikus rendszerekben említett leírásoktól, hiszen a sajátosságok sokszor nem is geometriai jellegűek.

c.) Végül a képfelismerés célja a képen rögzített valós világ leírása, a tárgyak felismerése és azonosítása, egy - tanulás útján bővülő - tudásbázisból már ismert alakzatokkal. Ennek alapján valamilyen feldolgozási célkitűzés teljesítése. A végső cél a látás automatizálása, a vizuális információt közvetlenül megértő, ezzel vezérelt robotok/automaták előállítása. A látási folyamatnak ezen egyszerűsített modell szerinti megközelítése sokféle és egymással bonyolult kölcsönhatásban lévő módszer és eljárás megvalósítását igényli, ezért az egyes szintek között nem lehet éles határvonalat vonni.

3. Alapismeretek

Ebben a részben szeretnék ismertetni olyan általános információkat, melyek később szükségesek a programban használt alkotóelemek, algoritmusok helyes értelmezéséhez.

3.1. A papír alapú dokumentum és sajátosságai

A legegyszerűbb, bárki számára könnyedén értelmezhető dokumentum a papír alapú dokumentum. Ez tartalmazhat valamilyen nyelven írt szöveget, grafikát, ábrákat. Lehet kézzel írt vagy számítógéppel szerkesztett, nyomtatott vagy rajzolt. Színeit tekintve lehet egyszerű fekete-fehér (az a valóságban mindig inkább szürkeskála) vagy épp színes. Méretét tekintve jellemezhetjük például a már széleskörben elterjedt A és B papírméretjelölésekkel (A4, B5, stb.) Ezek azok a lényeges tulajdonságok, amelyekre a későbbiekben kiemelt figyelmet kell hogy szenteljünk. Ez a fajta dokumentum lesz a feldolgozás folyamatának kiindulópontja.

3.2. Digitalizálási folyamat, szkennerek

A digitalizálási folyamat az a folyamat, amikor egy valós, fizikai alakot öltött, számunkra értékes logikai információt hordozó dokumentumot egy erre a célra rendelt képfeldolgozó eszközzel digitális formába „konvertálunk”.

A képbeviteli eszközök feladata, hogy valamely objektumról, térrészről azok képének megfelelő, számítógép által kezelhető formátumú adatokat állítsanak elő. Itt az adatok többnyire képpontok, melyek raszteres szervezésben tartalmazzák (sorról-sorra) a fényességi illetve színességi értékeket. Ezek az eszközök képérzékelők és jelátalakítók (illesztők, kódolók) együttesét jelentik. Ilyenek pl. a kamerák és képdigitalizáló kártyák. Vannak olyan készülékek, melyek egyben tartalmazzák mindkét funkciót. Ilyenek pl. a szabványos adatvonalra dolgozó szkennerek, illetve a digitális kamerák.

A mi esetünkben csak a hagyományos értelemben vett szkennerekkel foglalkozunk.

A szkennerek érzékelője CCD vonalkamera. A letapogatás során vagy a kamerát vagy a tárgyat kiegészítő mechanizmussal kell mozgatni, az érzékelő sorra merőleges irányban. A DTP alkalmazások és az irodai archiváló rendszerek elterjedésének köszönhetően a szkennerek igen széles választéka kapható, az egyszerű kézi szkennerektől a lapadagolóval ellátott, nagyfelbontású gyors lapolvasókig, természetesen monokróm és színes változatban egyaránt.

Minél több képpontérzékelőt integráltak egy CCD vonalkamerába, annál nagyobb felbontású digitális képet készíthetünk. Ezt az értéket általában DPI-ben szokták megadni. A mai általános célú szkennerek optikai felbontása 1200 DPI körül van, ami azt jelenti hogy átszámítva kb. 472 képpontérzékelő van centiméterenként. A fizikai felbontásnál nagyobb felbontást interpolációval lehet elérni, ami már gyakorlatilag szoftveres utófeldolgozás.

3.3. A digitalizált kép és jellemzői

Lényeges szempont a helyes digitalizálási felbontás és színmélység meghatározása. Ezt általában adott dokumentumtípusokhoz köthetjük, megvan a sajátossága egy fénykép szkennelésnek és egy újságcikk digitalizálásának is.

Ugyanis minél nagyobb felbontást választunk, annál nagyobb lesz a tárolandó képünk mérete, hiszen részletgazdagabban olvastuk be. A kapott méretre hatással van még a beállított színmélység is. A tárolandó kép mérete tehát a következő paraméterektől függ. A kép vízszintes és függőleges felbontásának szorzata adja a képpontok számát, valamint még attól hogy az egyes képpontok színinformációit milyen mélységben, azaz hány biten tároljuk, és végső soron hogy a tárolásnál alkalmazunk-e valamilyen tömörítő eljárást – mely lehet veszteséges vagy veszteség nélküli – a helytakarékoság érdekében.

Általánosságban elmondható, hogy egy jó minőségű fénykép digitalizálásához már szükséges a 600-800 DPI felbontás, míg egy újságcikk beolvasásához a 150-300 DPI is megfelelő. Ugyanilyen elvek alapján egy fénykép esetén indokolt a 24 vagy 32 bites szintárolás, újságcikk esetén pedig akár a 256 színű szürkeskála is megfelel.

Érdemes figyelembe venni azt a tényt, hogy míg egy nagyobb felbontásban beolvasott képet később bármikor konvertálhatunk, átalakíthatunk kisebb méretűre - mely nyilván információvesztéssel jár -, addig egy kisebb felbontással szkennelt kép nem igazán nagyítható anélkül, hogy a kép pixelesse válna, vagyis szétesne. Ezért célszerű mindig egy vagy két szinttel magasabb minőségű képet produkálni beolvasáskor.

3.4. Színrendszerek

Az emberi szemben lévő kb. 6-7 millió pálcikák – kísérletekkel bizonyítottan – megosztódnak három alapnak nevezett szín érzékelésén. Ezek a színek a vörös, zöld és a kék. Hozzávetőlegesen a pálcikák 65%-a a vörös fényre, 33%-a a zöldre és csak 2%-a a kék fényre érzékeny (de a kékre érzékeny pálcikák sokkal érzékenyebbek a többinél). Így alakul ki a az agyban a színes látás. Ezeknek a színeknek a különféle kombinációjában történő érzékelése adja a színek keveréke által elért szint.

Elsődleges színek, vagy tiszta színek (festészetben használatos): piros, sárga, kék

Alapszínek (világítástechnikában használatos): piros, zöld, kék

Additív színrendszer:

A fény-színek szerint keverednek, újabb szín bekeverése a színerőt növeli. Az alapszínek (RGB) egyforma keveréke a fehér színt eredményezi. A színek hiánya tökéletesen fekete színt eredményez. A színek keverését úgy lehet elképzelni, mintha újabb és újabb lámpákat kapcsolnánk be.

Szubtraktív színrendszer:

Az elnyelt színek szerint keverednek, újabb szín bekeverése a színerőt csökkenti. A fekete a színek összességét, a fehér a színek hiányát jelenti. A színek keverését úgy lehet elképzelni, mintha újabb és újabb színű áttetsző fóliákat helyeznénk egymásra.

A számítógépes tárolásnál a digitalizálás során a valósvilágbeli, végtelen színinformációkat valamilyen véges határok között definiált rendszerben kell rögzítenünk.

Míg a természetben a színek megszámlálhatatlan árnyalata és keveréke létezik, úgy ezek száma egy digitalizált kép esetén jól definiált szabály szerint maximálva van. Tehát már a digitalizálás során szükséges rögzítenünk azt az értéket, hogy a kép egy pontjához hányféle szín tartozhat. Ez maga a színösszetétel.

Black and White (fekete-fehér) színösszetétel: egy képpontnak két állapota van, fekete és fehér. Egy képpont állapotának rögzítése 1 bitet igényel.

16 color (16 szín): 16 megadott színe lehet egy képpontnak, 4 biten lehet tárolni egy képpont információját.

Grayscale (szürkeárnyalat): Egy képpont a szürke 256 árnyalatával rendelkezhet, 8 biten lehet tárolni a képpont információt.

256 Color (256 szín): 256 megadott színe lehet egy képpontnak, 8 biten tárolja az információt. (Weblapok képeihez ajánlott szintartomány)

RGB (vagy 24 Bit Color):

Egy képpont a piros, a kék és a zöld 256 féle árnyalatából áll össze, összesen 16 millió színárnyalattal. 24 biten tárolja az információt. Ez additív színrendszer, tehát a három alapszín egyforma keverése fehér, hiányuk fekete színt eredményez. Ezeket a színeket használja minden elektronikus kivetítő eszköz (monitor, projektor). Egy 24-bites RGB modellben tehát az egyes színek 256 árnyalatát tudjuk megkülönböztetni, tehát árnyalatonként 1 byte a tárolandó információ azaz egy képpontot 3 byte-al tudjuk megfeleltetni. Az így megkülönböztethető színek száma $256 \cdot 256 \cdot 256 = 16\,777\,216$ árnyalat. (Ezt hívjuk a számítástechnikában True color-nak, Példaként: Egy $1024 \cdot 768$ képpontból álló True color kép fizikai mérete $2\,359\,296$ byte = 2,36 Mbyte)

CMYK (vagy 32 Bit Color):

Egy képpont a türkiz (Cyan), a bíbor (Magenta) a sárga (Yellow) (másodlagos alapszínek) és a fekete (Black) $256 \cdot 4$ féle árnyalatából áll össze. 32 biten (4 byte) tárolja az információt. 4,3 milliárd árnyalata lehet egy képpontnak. A képszerkesztő programokban gyakran 0 és 100 közötti értékek adhatóak meg ezekből a színekből, ami 100 millió árnyalatot eredményez.

CMY:

Ugyanaz, mint a CMYK, csak fekete nélkül. A fekete nehezen keverhető ki (ezért veszik bele gyakran az alapszínek közé). A CYM alapszínei az RGB alapszíneinek komplementáris színei. Ez szubtraktív színrendszer. A színek hiánya fehéret eredményez. RGB - CMY átalakítás egy egyszerű matematikai művelet eredménye:

3.5. Grafikus formátumokról általában, a BMP formátum

A digitalizált kép számítógépes tárolására számos formátum létezik. Azt hogy melyiket érdemes választanunk az függ a felhasználás jellegétől. Egy egyszerű fényképet, melyet a későbbiekben már nem kívánunk tovább szerkeszteni, editálni érdemes a közkeletű JPEG vagy esetleg GIF formátumban elmenteni. Ezek veszteséges tömörítési eljárást használó tárolási formák, melyeknél a tömörítő algoritmusok az emberi szem számára kevésbé érzékeny képi információk csökkentésének lehetőségét kihasználva törekszenek a méret lényeges csökkentésére. Amennyiben további műveleteket szándékozunk végezni, abban az esetben más, információvesztéssel nem járó tárolási formát kell válasszunk, ilyen pl. a TIFF, BMP. Azaz itt minden képi információ teljes pontossággal van tárolva, ha tömörítési eljárást is alkalmazunk, az mindenképp veszteségmentes, reverzibilis, azaz az eredeti képet eredményezi kibontás után.

A BMP a Microsoft által a Windows rendszerhez kifejlesztett formátuma. Alkalmas 1, 4, 8 és 24 bit színmélységű képek tárolására, tömörítés nélkül vagy veszteség nélküli, RLE tömörítéssel. Ugyancsak BMP állományokat használ az IBM által kifejlesztett OS/2 operációs rendszer, de ezek szerkezetükben némileg eltérnek a Windowsos BMP-ktől. A BMP formátum hátránya, hogy nem támogatja a CMYK színábrázolást, ezért nyomdai felhasználásra nem alkalmas. Programomban ezt a formátumot használom.

4. A feldolgozó program megvalósítása

Kutatásom célja egy olyan program létrehozása volt, amely a lehető legjobban próbálja kiváltani a papír alapú dokumentumok ember általi feldolgozási fázisát. Mivel nem kifejezetten egy adott érvényességű területre kívánom használni, ezért mindenképpen szempont volt a rugalmasság, az újrafelhasználhatóság. Mindemellett egy nyomtatvány jellegű dokumentum mégis kötött szabályokkal rendelkezik, így nyilván az abban szereplő elemek felhasználásával a lehető legnagyobb felhasználói szabadsága törekedtem.

4.1. A fejlesztőkörnyezet kiválasztása

Programom fejlesztéséhez a Borland Software cég által biztosított Borland Delphi 7-es fejlesztőkörnyezetet használtam. Ez az objektumorientált fejlesztőkörnyezet rendelkezik minden olyan alapvető funkcióval, mely lehetővé teszi alkalmazások gyors és hatékony fejlesztését. Komponensei révén könnyedén bővíthető, s mivel igen széles körben elterjedt, ezért rengeteg kiegészítés, példa található hozzá. Kezelése felhasználóbarát, felülete könnyen átlátható, így a programozó jobban tud koncentrálni az adott feladat megvalósítására.

Mivel napjaink legelterjedtebb operációs rendszere a Microsoft Windows termékcsaládja és mint grafikus objektumokkal dolgozó program igényli a grafikus felhasználói felületet, ebből a szempontból is kitűnő választás. Delphiben a vizuális tervezés és a kódgenerálás szervesen összekapcsolódik.

4.2. A program feladata, célkitűzése

Automatikus dokumentum-feldolgozó rendszerről lévén szó, programom a következő célt kívánja megvalósítani.

Legyen adott egy olyan nyomtatvány, mely bizonyos formai kellékek tekintetében kötött ugyan, de a felhasználó által szabadon kialakítható, testreszabható. Legegyszerűbb

formája ennek például egy kérdőív. Ezen dokumentumot számítógép segítségével állítjuk elő, majd mivel nem elektronikus adatgyűjtésről van szó, papír alapra helyezzük, azaz kinyomtatjuk. Oldalszám tekintetében lehet akár többoldalas is. Ezeket a nyomtatványokat kézzel kitöltetjük.

Az adott nyomtatványon a következő tartalmi, információt bekérő részek találhatók:

- Feleletválasztás esetén ún. checkboxok, melyeknél az információkérés helyén felszólítjuk a kitöltő személyt, az általa helyesnek gondolt vagy nézeteivel egyező válasz egyértelmű bejelölésére a lapon.
- Numerikus adatok bekérése esetén pedig számmezőkről beszélhetünk, mely számmezőket a kitöltő személy saját kézírásával tölt ki.

Ezen típusú kérdések száma egy kérdőívben belül korlátlan mennyiségű lehet és a legkülönbözőbb jellegű információkról tájékozódhat.

A dokumentumok kitöltése és annak begyűjtése után a következő fázis a feldolgozás. Mint a dolgozatomban elején is említettem, ez nagy példányszámú dokumentumok esetén igen időigényes, ezért ennek elvégzését a számítógépre hárítjuk.

Ahhoz hogy számítógépes feldolgozást tudjunk rajta végrehajtani, szükséges a szintén fentebb említett eljárás, a digitalizálás. Ennek eredményeképpen olyan formátumot állítunk elő, melyet felhasználva programunk képes a kiértékelést elvégezni, azaz felismerni az egyes checkboxok állapotát (be lett-e jelölve vagy sem), illetve számmezők esetén képes felismerni hogy a kitöltő milyen számot írt az adott rubrikába.

Egy ilyen kérdőív lehetséges mintája a következő:

Szia!

Mint minden évben, most is szeretnénk tudni, melyek azok a feltételek, viszonyok, stb., amelyek javításával elérhetjük, hogy még jobban érezd magad vállalatunknál.

Ennek érdekében néhány olyan kérdést szeretnénk feltenni Neked, amely arra irányul, hogy milyennek látod a jelenlegi munkahelyi körülményeket és milyen konstruktív javaslataid vannak azok javítására.

Először is néhány alapadatot szeretnénk bekérni tőled, ezek csupán statisztikai célokat szolgálnak. Kérjük jól olvashatóan töltsd ki a megadott mezőket, a beírt értékek ne lógjanak ki a keretből és ne érintsék azt.

Születési dátumod: év hó nap

Nemed: Férfi Nő

Hány éve dolgozol vállalatunknál: év

Havi bruttó kereseted:

Felsőfokú végzettségeid száma: (amennyiben nem rendelkezel felsőfokú végzettséggel, kérlek írd 0-at)

(1. oldal MINTA)

Az alábbi kérdésekben – különböző munkahelyi feltételekkel kapcsolatban – az elégedettségedet kell kifejezni. Ehhez egy hétfokozatú skála áll a rendelkezésedre. A skálán 1-től 7-ig jelölheted meg azt az értéket, amely a leginkább kifejezi azt, hogy mennyire vagy elégedett vagy elégedetlen egy adott körülménnyel. Az értékelést az alábbi kategóriák segítik.

<i>Nagyon elégedetlen vagyok</i>	<i>Elégedetlen vagyok</i>	<i>Kissé elégedetlen vagyok</i>	<i>Közepesen</i>	<i>Kissé elégedett vagyok</i>	<i>Elégedett vagyok</i>	<i>Nagyon elégedett vagyok</i>
<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>

1. a munkahelyi elhelyezési körülményekkel (pl. irodák kialakítása):

1	2	3	4	5	6	7
☐	☐	☐	☐	☐	☐	☐

2. a munkavégzésedhez szükséges eszközökkel:

1	2	3	4	5	6	7
☐	☐	☐	☐	☐	☐	☐

3. a munkavégzésedhez feltétlenül szükséges információk áramlásával:

1	2	3	4	5	6	7
☐	☐	☐	☐	☐	☐	☐

4. kollegális kapcsolatokkal (együttműködés, jó hangulat):

1	2	3	4	5	6	7
☐	☐	☐	☐	☐	☐	☐

5. munkádhoz viszonyított fizetéseddel:

1	2	3	4	5	6	7
☐	☐	☐	☐	☐	☐	☐

Véleményed szerint a céghez való kötődés, elkötelezettség, lojalitás szempontjából melyek a számodra fontosak? Legalább kettőt jelölj meg!

Alapjövedelem

☐

Teljesítményarányos jutalék

☐

Kollegális kapcsolatok

☐

Cégkultúra

☐

A cég piaci megítélése

☐

Munkaköröd jellemzői

☐

Szakmai fejlődési lehetőség

☐

Munkahely fizikai környezete

☐

Köszönjük válaszaidat!

(3. oldal MINTA)

A mintán jól láthatóak az egyes információt bekérő, kitöltésre váró mezők. Ezek mindig téglalap alakúak, melybe értelemszerűen vagy X-et, mint jelölést teszünk vagy számjegyet írunk. Checkbox esetén a kérdés jellegéből adódóan megengedhető hogy akár többet is megjelöljünk, de szabhatunk ennek alsó és felső határt is. Számmező esetén úgynevezett összefüggő rubrikákról beszélhetünk. Ez a feldolgozás szempontjából lesz lényeges. Ugyanis pl. egy teljes születési dátumot bekérő mezőket érdemes a feldolgozás szempontjából logikailag különválasztani évre, hónapra és napra.

A kitöltött dokumentumokat beszkenelve grafikus formában a számítógépen tároljuk. Ezen kívül szükség van még egy úgynevezett sablonra is, mely egy üres, kitöltetlen példányt jelent. Ez is szintén digitális kép formájában van jelen a háttértáron.

Mivel programomat nem kifejezetten egy fajta dokumentum feldolgozására készítettem, ezért azt meg kell tanítani, azaz meg kell adni a program számára azokat a paramétereket, ami alapján tudni fogja hogy hol és milyen jellegű információkat keressen a feldolgozandó képen. Ezt nevezzük tanítási folyamatnak. A megismertett dokumentum adatait a program tárolja, így azt még egyszer nem kell betanítanunk, bármikor előhívhatjuk, ha ismételten ugyanolyan típusú dokumentumot kívánunk feldolgozni.

4.3. A programmal szembeni követelmények

A feldolgozóprogrammal szemben a következő követelményeket támasztottam.

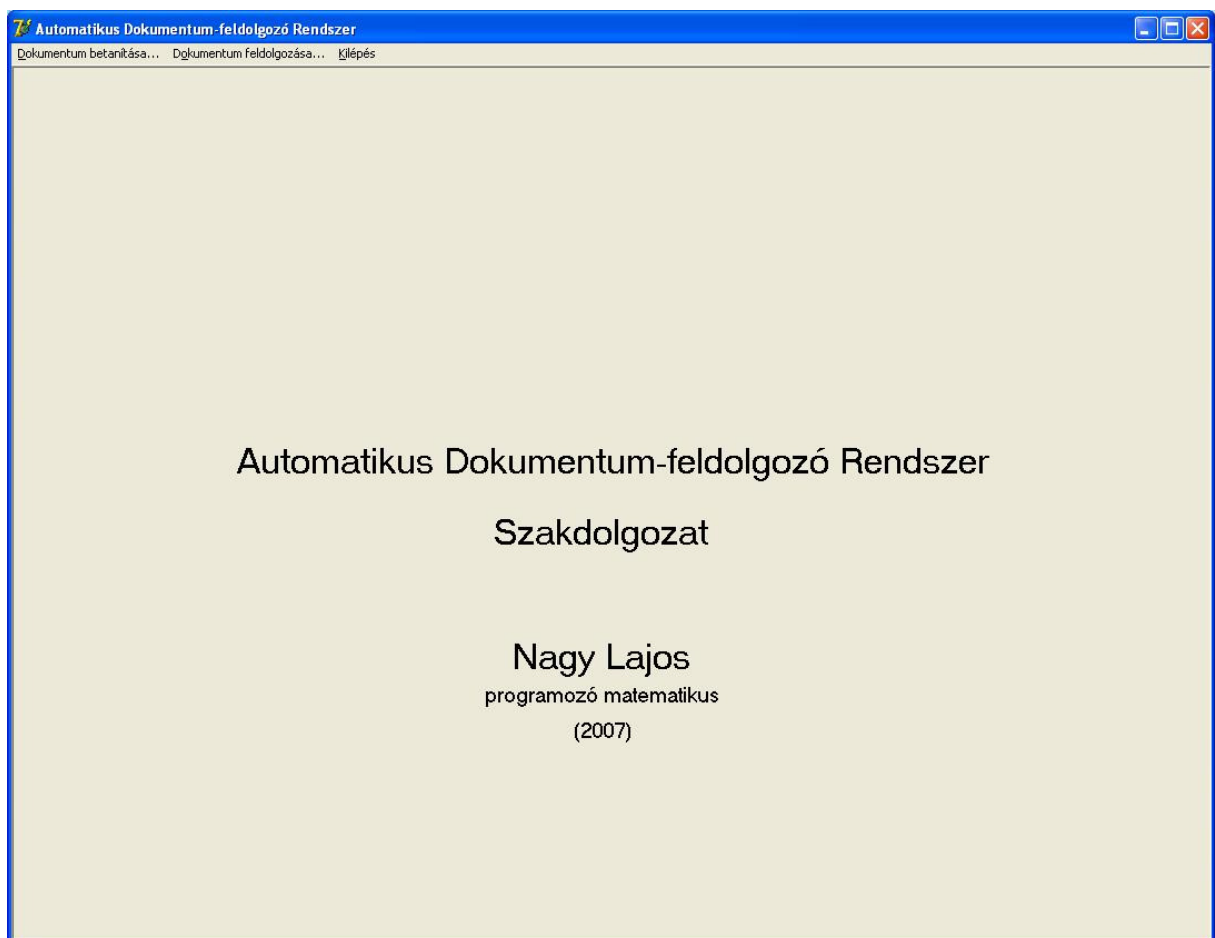
- Használható legyen mindenfajta olyan dokumentum feldolgozására, ahol a kézi kitöltés során feleletválasztást vagy számjegykitöltést kell alkalmazni valamilyen zárt alakzaton belül.
- Mivel az ilyen jellegű dokumentumokat nem a programmal készítjük, ezért azt feldolgozás előtt meg kell tudnunk ismertetni a programmal.
- Rugalmas legyen és érvényesüljön az újrafelhasználhatóság elve.
- Minél inkább automatizálható legyen a folyamat és a lehető legkevesebb interakciót igényelje. Azaz olyan eljárásokat próbálok alkalmazni, melyek a lehető legbiztosabban döntenek és a program csak szükség esetén kér segítséget a felhasználótól.

- A feldolgozás eredménye elmenthető legyen, a felhasználó számára könnyen értelmezhető formában rögzítse azt.

4.4. A program működése és algoritmusai

Ebben a részben ismertetném a program működését, alapvető funkcióit, annak matematikai és programozástechnikai megvalósítását, hivatkozva a korábban ismertetett információkra. Lényeges szempont, lévén hogy grafikai szoftverről van szó, melynek használata során – a grafikus bejelölésnél – közvetlenül is dolgozunk képfájlokkal, szükséges a megfelelő felbontású megjelenítő (monitor) használata.

Indítás után a következő lehetőségek állnak rendelkezésre.



4.4.1. Dokumentum betanítása

A dokumentum betanítása az a folyamat, amikor egy eddig még nem ismert, nem használt dokumentumot megismertetünk a rendszerrel. Megismertetés alatt azt értjük, hogy definiáljuk az egyes adatmezőket nevükkel, típusukkal és egyéb paramétereikkel, majd a kitöltetlen sablon dokumentumon grafikusán megjelöljük, rögzítjük azok a koordinátáit, azokat a területeket, ahol a kitöltést jelentő hasznos képi információt fogjuk keresni és feldolgozni. Ezt mintegy varázsló segítségével tehetjük meg.

4.4.1.1. Alapadatok megadása

Első lépésként meg kell adnunk hogy hány oldalas dokumentumot szeretnénk betanítani, ezek fogják később az aloldalak számát jelölni.

Automatikus Dokumentum-feldolgozó Rendszer - [Betanítás]

Dokumentum betanítása... Dokumentum feldolgozása... Kilépés

Dokumentum alapadatainak megadása

Mégse Tovább >>

Kérem adja meg a betanítandó dokumentum oldalainak számát (max. 20):

1

A jelenlegi fejlesztés állapotában ez maximálisan 20 oldalra van limitálva, később ez a korlát természetesen feloldódik. Az egész tanítási műveletet bármikor félbeszakíthatjuk a „Mégse” gomb használatával.

4.4.1.2. Adatmezők felvitele

A következő fázisban az adatmezőkkel kell megismertetni programunkat.

Automatikus Dokumentum-feldolgozó Rendszer - [Betanítás]

Dokumentum betanítása... Dokumentum feldolgozása... Kilépés

Adatmezők felvitele az adatbázisba

Mégse Tovább >> Vigye fel az adatmezőket (számmező, checkbox) és adja meg paramétereiket!

Adatmező törlése
Sorszám: 1 Törölés

Ssz.	Oldalszám	Típus	Megnevezés	Összefüggő rubrikák száma	Kötelezően megadandó (számmező)	Min. bejelölendő (checkbox)	Max. bejelölhető (checkbox)
1.	1	számmező	Születési év	4	4	-	-
2.	1	számmező	Születési hónap	2	2	-	-
3.	1	számmező	Születési nap	2	2	-	-
4.	1	checkbox	Neme	2	-	1	1
5.	1	számmező	Eltöltött évek száma	2	1	-	-
6.	1	számmező	Havi bruttó kereset	7	5	-	-
7.	1	számmező	Felsőfokú végzettség	1	1	-	-

Számmező hozzáadása (pl. születési év)

A dokumentum hanyadik oldalán található (max. az oldalszám):
1

Adatmező megnevezése:

Összefüggő számrubrikák száma (max. 20):
1

Hány rubrika kötelező kitölteni kötelező jobbról (max. az ö. függő rubrikák száma):
1

Felvitel

Checkbox mező hozzáadása (pl. jelölje be az Őnt érdeklő témákat)

A dokumentum hanyadik oldalán található (max. az oldalszám):
1

Adatmező megnevezése:

Összefüggő számrubrikák száma (max. 20):
1

Maximálisan hány rubrika lehet bejelölve (max. az ö. függő rubrikák száma):
1

Minimálisan hány rubrika lehet bejelölve (max. az ö. függő rubrikák száma):
1

Felvitel

Itt egy listába gyűjti a rendszer a felvitt objektumokat és a hozzá kapcsolódó paramétereiket.

Számmező esetén meg kell adnunk hogy adatok a dokumentum hanyadik oldalán találhatóak. Itt alapértelmezés szerint 1-el kezdődik, hiszen általában az első oldallal kezdünk. Szükség van még a későbbi kényelmes értelmezés érdekében egy névre, mellyel saját

magunknak azonosítjuk majd hogy a megadott adatmező milyen jellegű értéket képvisel. S végül definiálhatjuk, hogy logikailag mennyi számjegyet kapcsoljunk össze, azaz az összefüggő rubrikák számát. Ez például évszám esetén 4 darabot, hónap esetén 2 darabot jelent. Valamint megadhatjuk, hogy jobbról minimálisan hány rubrikának kell kitöltve lennie. Évszám esetén ez itt szintén 4 darab, de ha például már fizetés iránt érdeklődünk, akkor az lehet 4, 5 vagy akár 6 számjegyű is, tehát beállítható egy alsó határ. Természetesen figyelmet kapott az a logikai biztonság, hogy nagyobbra nem állíthatjuk ezt az értéket, mint az összefüggő rubrikák száma. Végül a „Felvitel” gombra kattintva adjuk hozzá a listához az adatmezőt.

Checkbox esetén szintén az oldalszám és a név megadásával kell kezdenünk. Majd az összefüggő rubrikák számának megadása után be kell állítanunk hogy minimálisan és maximálisan mennyi lehet megjelölve belőlük. A helyes adatok megadása végeztével a rekordot rögzítjük.

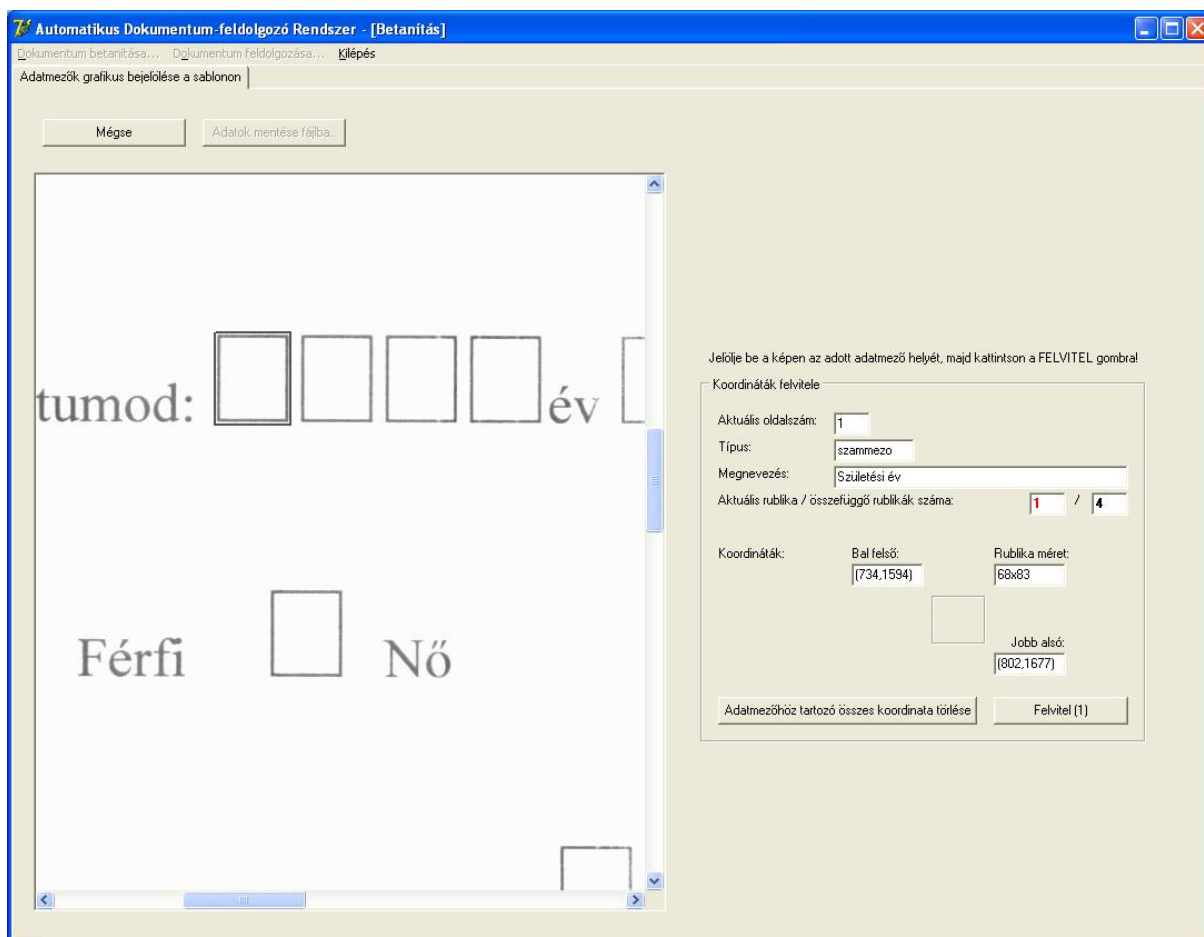
A program mindkét típus felvitelekor ellenőrzi az értékeket és csak akkor rögzíti, ha azok helyesek. A megjelenített listát minden egyes felvitel után aktualizálja, oldalszám szerint növekvő sorrendbe rendezi. A lista egy elemére állva az adott tétel törölhető. Csak akkor léphetünk a következő oldalra, ha legalább egy adatmezőt megadtunk. A felvitt adatokat a rendszer a háttérben ideiglenes fájlba gyűjti.

4.4.1.3. Adatmezők grafikus bejelölése a sablonon.

Ezen a ponton kell az előzőleg beolvasott üres sablonon grafikusán bejelölni az adatmezők pontos elhelyezkedését.

Megjegyezni kívánom, hogy kísérletezésem során arra a megállapításra jutottam, hogy mind az üres sablont, mind pedig a feldolgozandó dokumentumokat 300 DPI-vel és 256 szürkeárnyalatban érdemes digitalizálni. Ugyanis mint az később kiderül, ezzel a felbontással lehet kényelmesen és viszonylag pontosan dolgozni.

A felvitt adatok alapján a program a megadott sablon aloldalak grafikus fájljainak megnyitására szólít fel, majd végighalad az adott oldalra definiált adatmezőkön. Ezek értékeit és egyéb más segédinformációkat megjeleníti.



Lépésről lépésre haladhatunk az egyes adatmezők felvitelével. Először a görgetősávok használatával kell a sablon képen az adott helyre pozicionálni, majd az egér segítségével egy bennfoglaló téglalapot kell kijelölni. Ennek mindenképpen tartalmaznia kell azt a keret, melyben a számjegy vagy megjelölés található. Ha hibáztunk, akkor a kijelölés újra elvégezhető. Amennyiben helyesen jártunk el, akkor a „Felvitel (x)” gombbal rögzíthetjük a koordinátákat. Az „x” az aktuális rubrika számát jelenti.

Összefüggő rubrikák esetén csak az első rubrika kijelölésénél van lehetőség szabad méretet választani, a hozzá tartozó többi esetén feltételezzük a dokumentum egységességét és ott már csak a balfelső sarkot kell a képen igazítani. Ez azt jelenti, ha például az évszám

számjegyeinek helyét kívánom rögzíteni, akkor csak az első számjegynél tudok egyéni méretet kijelölni, a maradék három esetén ugyanolyan méretet feltételezünk. Amennyiben elrontottuk, az „Adatmezőkhöz tartozó összes koordináta törlése” gombbal előről kezdhethetjük az aktuális adatmező rubrikáinak bejelölését.

Ha az összes adatmezőt bejelöltük, a betanított dokumentumot egyéni, ADF kiterjesztésű fájlba mentjük el.

4.4.1.4. Az ADF fájl felépítése

Egyszerű szöveges fájl, mely a Windows által is használt INI fájlok formátumával egyezik meg. Szekció-kulcs-érték összerendeléseket tartalmaz. Példaként az elmentett anyagunk a következő. A fájl tartalmát a könnyebb olvashatóság miatt két hasábra tördeltem.

[basic]	ofrsz=2
osszoldalszam=1	szam_kotelezo=2
adatmezok_szama=7	checkbox_min=null
sablon01=sablon_szakdolgozathoz.bmp	checkbox_max=null
[1]	completed=1
oldalszam=1	koordinata001=1315,1596,1383,1679
tipus=szammezo	koordinata002=1391,1596,1459,1679
nev=Születési év	[4]
ofrsz=4	oldalszam=1
szam_kotelezo=4	tipus=checkbox
checkbox_min=null	nev=Neme
checkbox_max=null	ofrsz=2
completed=1	szam_kotelezo=null
koordinata001=734,1593,802,1677	checkbox_min=1
koordinata002=810,1594,878,1678	checkbox_max=1
koordinata003=886,1594,954,1678	completed=1
koordinata004=962,1595,1030,1679	koordinata001=504,1823,573,1905
[2]	koordinata002=782,1824,851,1906
oldalszam=1	[5]
tipus=szammezo	oldalszam=1
nev=Születési hónap	tipus=szammezo
ofrsz=2	nev=Vállalatnál eltöltött évek
szam_kotelezo=2	ofrsz=2
checkbox_min=null	szam_kotelezo=1
checkbox_max=null	checkbox_min=null
completed=1	checkbox_max=null
koordinata001=1098,1596,1166,1679	completed=1
koordinata002=1175,1596,1243,1679	koordinata001=1043,2054,1112,2136
[3]	koordinata002=1118,2054,1187,2136
oldalszam=1	[6]
tipus=szammezo	oldalszam=1
nev=Születési nap	tipus=szammezo

```

nev=Havi bruttó kereset
ofrsz=7
szam_kotelezo=5
checkbox_min=null
checkbox_max=null
completed=1
koordinata001=797,2284,865,2365
koordinata002=873,2284,941,2365
koordinata003=949,2284,1017,2365
koordinata004=1024,2284,1092,2365
koordinata005=1101,2285,1169,2366
koordinata006=1177,2285,1245,2366
koordinata007=1254,2285,1322,2366

```

```

[7]
oldalszam=1
tipus=szammezo
nev=Felsőfokú végzettségek száma
ofrsz=1
szam_kotelezo=1
checkbox_min=null
checkbox_max=null
completed=1
koordinata001=1010,2514,1078,2596

```

A [BASIC] szekció alatt találhatóak meg az alapvető adatok, az **osszoldalszam**, mely megadja hogy hány aloldalból áll a dokumentum. Az **adatmezokszama** kulcshoz tartozó érték azt mutatja, hogy összesen mennyi adatmező rekordot vittünk fel (évszám, hónap, nap, neme, vállalatnál eltöltött idő, havi bruttó kereset és felsőfokú végzettségek száma). Valamint itt kerül tárolásra a sablonként megnyitott fájl neve is, **sablonXX** kulcs formájában, ahol XX az adott aloldalt jelenti.

A következő számozott szekciók [1] , [2] , ... stb. az egyes adatmezők adatait tartalmazza. Az **oldalszam** kulcshoz rendelt érték jelöli, hogy az melyik aloldalon található. A **tipus** azonosítja a hogy checkboxot kell majd felismerni vagy számmezőt. A **nev** kulccsal az adatmező nevét adjuk meg, az **ofrsz** az összefüggő rubrikák számát jelöli. Ez checkbox esetén megmondja, hogy hány kis téglalap tartozik logikailag egységbe. Számmező esetén hány számjegyet kell majd „összeolvasni” egy számmá. A **szam_kotelezo** kulccsal azt definiálom, hogy több összefüggő számmező rubrika esetén, abból mennyinek kell kitöltve lennie minimálisan. Ha a típus checkbox, akkor ennek értéke „null”. A **checkbox_min** és **checkbox_max** kulcsok checkboxok esetén jelölik a minimálisan és maximálisan megjelölhető rubrikák számát. Számmező típus esetén ezek értéke szintén „null”.

A **completed** kulcs egy logikai értéket képvisel (0 vagy 1), mely azt hivatott jelölni, hogy a rubrikákhoz tartozó grafikus koordináták már feltöltésre kerültek-e. Amennyiben igen, akkor ennek értéke 1, egyébként 0. Ez a programot vezérlő algoritmust segíti, grafikus bejelölés fázisban ezt az értéket figyeli a program és szükség esetén megjeleníti a koordináta felvitelhez tartozó adatokat, információkat.

A **koordinataXXX**, ahol az XXX egy 001-től induló háromjegyű szám az egyes összefüggő rubrikák koordinátáit adja meg a következő sorrendben: bal-felső X, bal-felső Y, jobb-alsó X, jobb-alsó Y. Ezek vesszővel vannak elválasztva. A „completed” kulcs értéke csak akkor vált 1-re ha minden koordináta felvitelre került.

Ezzel a módszerrel minden olyan lényeges információt eltárolunk az adott dokumentumtípusról, amelyekre szükségünk lesz annak feldolgozásakor.

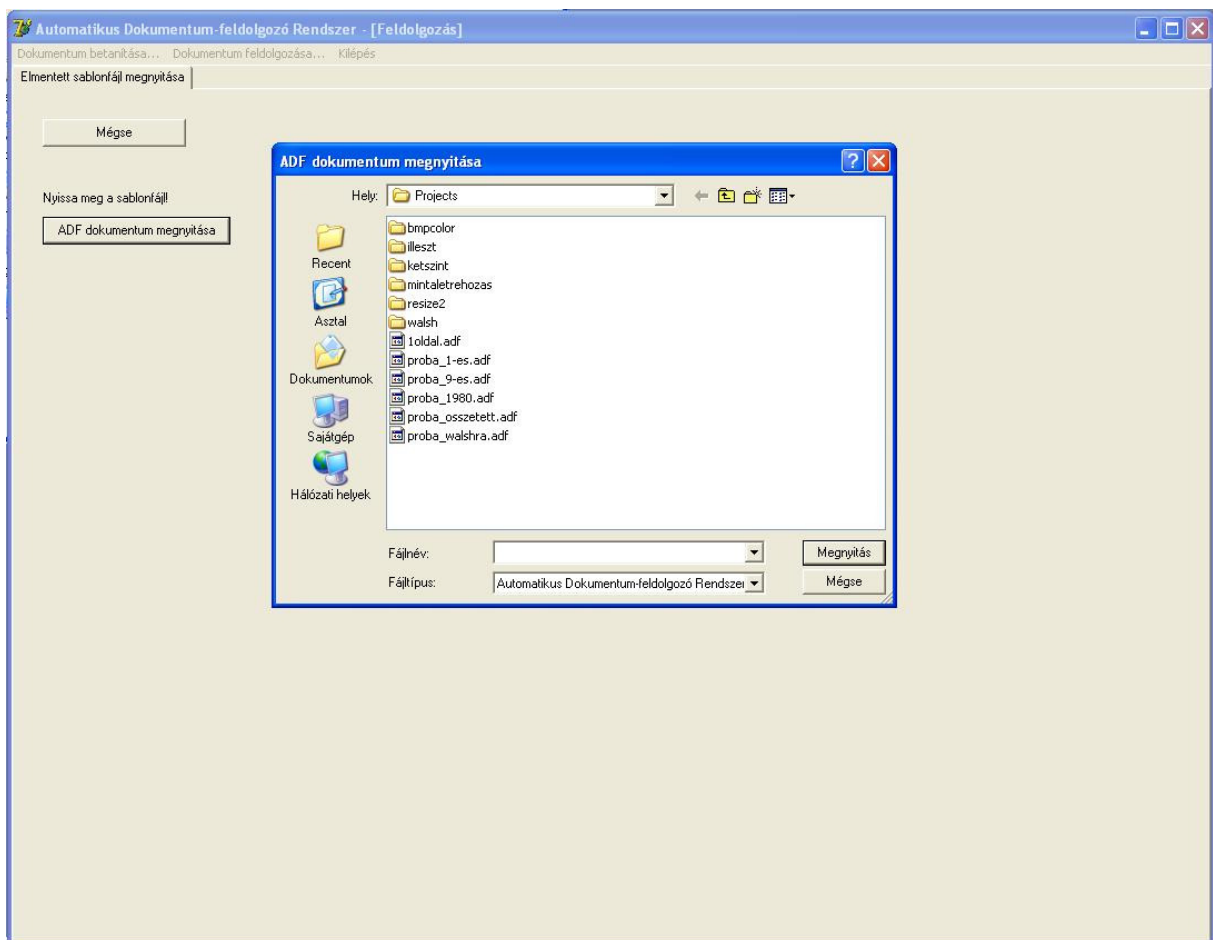
4.4.2. Dokumentum feldolgozása

Amennyiben már van kész, betanított dokumentumunk és a kitöltött, beszkenelt példányok is rendelkezésre állnak, hozzáfoghatunk azok tényleges feldolgozásához.

Ebben a részben szeretném végigvezetni a program használatán, illetve az egyes részeknél - különösen a feldolgozásnál - részletezném a használt algoritmusokat, módszereket, megvalósított ötleteket.

4.4.2.1. Betanított dokumentumtípus megnyitása

Először is a meg kell nyitni a már elmentett dokumentumtípus sajátosságait rögzítő fájlt, melybe a betanítási folyamat során rögzítve lettek a szükséges adatok. Ezt a rendszer a már fentebb említett ADF kiterjesztésű fájlokban tárolja. A szokásos Windowsos fájlmege nyitási dialógust használva kereshetjük meg a számunka szükséges dokumentumot.



4.4.2.2. Adatmezők áttekintése

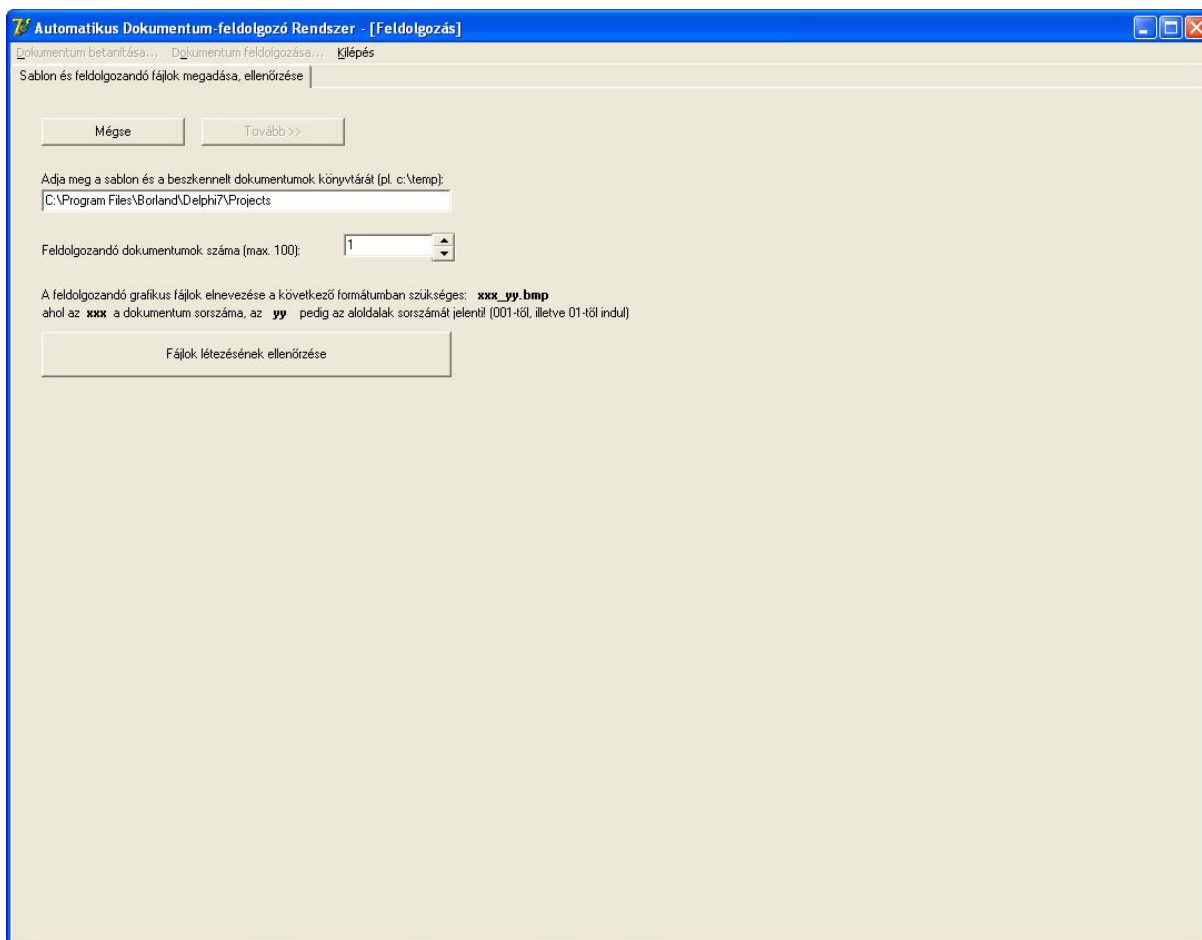
Az adott leíró fájlt a program beolvassa, majd megjeleníti számunkra az abban definiált adatmezőket és azok paramétereit. Ez a programrész azt a célt szolgálja, hogy még a feldolgozás indítása előtt ellenőrizhessük, hogy jó dokumentumtípust nyitottunk-e meg és az tartalmaz-e minden fontos információt.

Ssz.	Oldalszám	Típus	Megnevezés	Összefüggő rubrikák száma	Kötelezően megadandó (számmező)	Min. bejelölendő (checkbox)	Max. bejelölhető (checkbox)
1.	1	számmező	Születési év	4	4	-	-
2.	1	számmező	Születési hónap	2	2	-	-
3.	1	számmező	Születési nap	2	2	-	-
4.	1	checkbox	Neme	2	-	1	1
5.	1	számmező	Vállalatnál eltöltött	2	1	-	-
6.	1	számmező	Havi bruttó kereset	7	5	-	-
7.	1	számmező	Felsőfokú végzettség	1	1	-	-

Összoldalszám: 1

4.4.2.3. Grafikus fájlok létezésének ellenőrzése

Amennyiben meggyőződünk, hogy minden rendben van, a „Tovább” gombra kattintva a feldolgozandó fájlok létezésének ellenőrzéséhez jutunk.



Itt elsődlegesen meg kell adni azt a könyvtárat, ahol a feldolgozandó BMP fájlok találhatóak, valamint hogy hány darab kitöltött dokumentumot kell feldolgozni. Ez jelenleg 100-ig limitált. A fájloknak a következő névkonvenció szerint kell elnevezve lennie az adott könyvtárban. Minden egyes feldolgozandó fajlnak a **xxx_yy.bmp** névformátumot kell követnie, ahol az „xxx” a dokumentumok sorszáma szerint 001-től 099-ig követik egymást, az „yy” pedig az adott nyomtatványon belül az aloldal számát jelöli 01-től 99-ig.

Példaként egy három oldalból álló, négy példányban kitöltött dokumentum esetén összesen 12 darab beolvasott BMP fájlnak és három darab ugyanabban a könyvtárban lévő sablon fájlnak kell léteznie.

001_01.bmp	002_01.bmp	003_01.bmp	004_01.bmp	sablon_01.bmp
001_02.bmp	002_02.bmp	003_02.bmp	004_02.bmp	sablon_02.bmp
001_03.bmp	002_03.bmp	003_03.bmp	004_03.bmp	sablon_03.bmp

A „Fájlok létezésének ellenőrzése” gombra lefutó algoritmus ezen szabály szerint ellenőrzi az adott könyvtárban lévő fájlok létezését és hiányosság esetén figyelmeztet a sablon vagy a feldolgozandó képek nem létezésére. Amennyiben minden fájl megtalálható, engedélyezi a továbblépést.

Ezt a részt azért építettem bele programomba, hogy az automatizálási folyamat gördülékenyebb legyen, ne a feldolgozás közben kelljen megállni egy esetleg feldolgozandó fájl hiánya miatt.

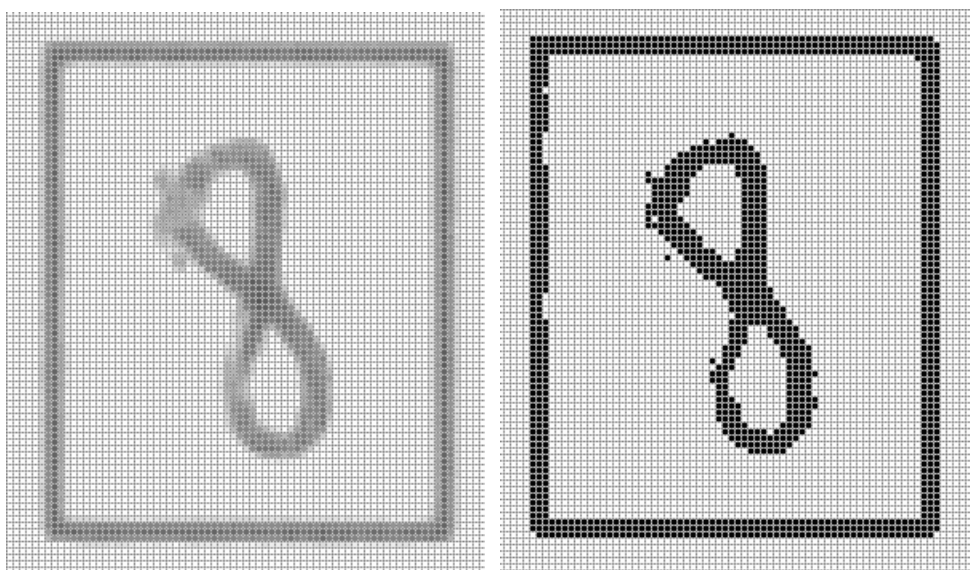
4.4.2.4. A felismerés lépései

A feldolgozási folyamat a következő lépéseket alkalmazza. Adott egy szürkeskálás sablonunk és egy vagy több kitöltött mintánk. A mindkettő digitalizált képet először egy jól meghatározott világossági érték alapján két szintre vágjuk, azaz bináris képet állítunk elő belőle. Ezzel a bináris képpel dolgozunk tovább. A megadott koordináták alapján kivágjuk a feldolgozandó képrészleteket. Ezt a sablon esetén pontosan megtehetjük, hiszen azon végeztük a bejelölést, a feldolgozandó minták esetén azonban érdemes néhány pixelrel nagyobb területet figyelembe venni a szkennelési pontatlanság lehetősége miatt. Következő lépésként a két képet egymásra illesztjük, meghatározzuk a legjobb lefedést és kivonjuk egymásból azokat. Innentől már csak egy képpel dolgozunk tovább. Levágjuk a vélhetően még létező keretet, majd az így kapott képet szegmentáljuk. Checkbox esetén ez már készen áll a felismerésre, itt mindössze a fekete pixelek arányát határozzuk meg és egy küszöbszintet vizsgálunk. Számmező esetén a képet tovább konvertáljuk 64x64 képpont méretűre, majd Walsh-transzformáció segítségével előállítjuk a sajátságvektort. Ezen sajátságvektort

hasonlítjuk a már előre definiált, eltárolt sajátosságvektorokhoz. Ahol a legkisebb a két vektor eltérése, minden bizonnyal az lesz az adott számjegy. Az összefüggő rubrikákba írt számjegyeket összefűzzük és elkészítjük belőle a végleges numerikus adatot.

4.4.2.4.1. Két szintre vágás

Jelen esetben 8 bites szürkeskálás képpel dolgozom, amely 256 különböző szürkeségi szintet tartalmazhat a fekete színtől a fehér színig. A fekete színhez a 0-ás, a fehér színhez a 255-ös értéket rendeljük, így a köztes szürkeségi szintek a (0..255) intervallumból vesznek fel egész értéket. A két szintre vágás érdekében meg kell határozni egy szürkeségi szintet, amelynél sötétebb színeket feketévé, a világosabbakat fehérre minősíthetjük át. Az ezzel folytatott kísérletek alapján arra a megállapításra jutottam, hogy ez az érték kb. 180-as világossági értéknél a legmegfelelőbb. Az ettől az értéktől kisebb világossági szintek esetén a képpont színértékét 0-ra, ettől nagyobb esetén 1-re állítjuk. Ezzel előállt a bináris képünk.



Két szintre vágás előtt

Két szintre vágás után

4.4.2.4.2. Egymásra illesztés

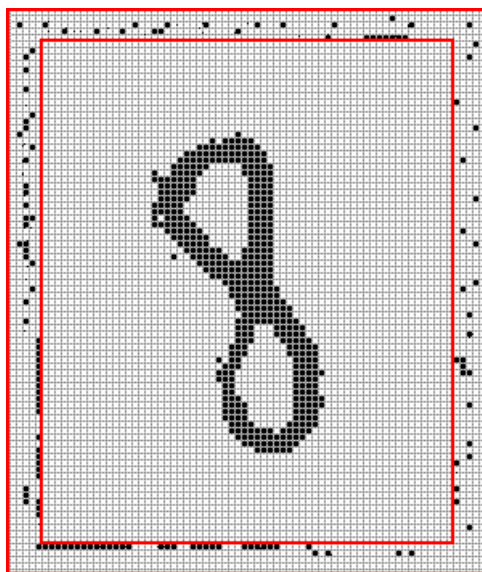
Adott tehát két darab kétszintre vágott képünk. Az egyik az üres sablon pontosan, a másik a kitöltött minta néhány pixellel szélesebben kivágva. A két képet a lehető

legpontosabban kell egymásra illeszteniük. Ezt úgy tesszük meg, hogy a kisebb képet pixelenként X és Y irányban végigtologatva „egymásra helyezzük” és az adott (X,Y) eltoláshoz egy értéket rendelünk. Ezt az értéket úgy határozzuk meg, hogy a nagyobbik képre vonatkozóan megszámoljuk azoknak a pixeleknak számát, ahol a két kép eltér egymástól. A sablon képét tovább csúsztatjuk és újra meghatározzuk ezen értéket. A legjobb illesztés annál az (X,Y) eltolásnál lesz, ahol ez a szám a legkisebb. Ez lesz az a pont ahová eltolva a sablont a mintán a legjobban fedí egymást a két képrészlet. Így végül a mintát a sablonnal megegyező méretűre vághatjuk.

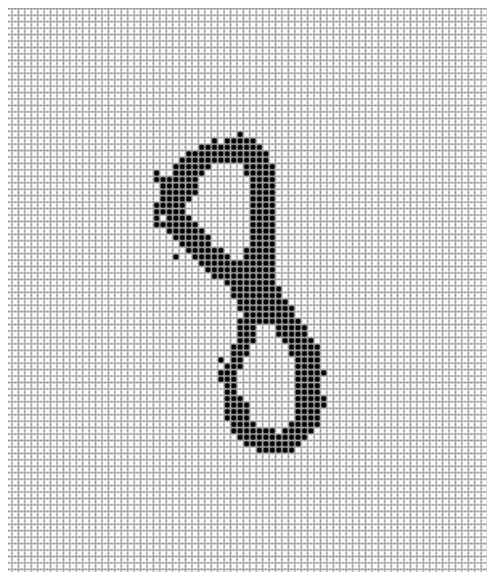
Az hogy mennyivel nagyobb képet vágjunk ki a mintán, a program számára megadható. Tapasztalat szerint ez a felismerés legidőigényesebb része, ugyanis minél nagyobb területre kell illeszteniük – tehát minél nagyobb az esetleges beolvasásból eredő elcsúszás – annál több időt vesz igénybe ennek pontos meghatározása.

4.4.2.4.3. Kerettörlés

Az egymás illesztés során a bennfoglaló keretnek egy része (80-90%) eltűnik, ám a szkennelési sajátossága miatt keletkezett zaj a sablon és a minta kerete között eltérést eredményez. Ezek eltüntetéséhez az illesztett képre kerettörlést alkalmazunk, azaz megadott pixel vastagságban a kép szélétől minden képpontot fehérre állítunk. Ezt a vastagságot a programnak a feldolgozás előtt megadhatjuk.



Kerettörlés előtt

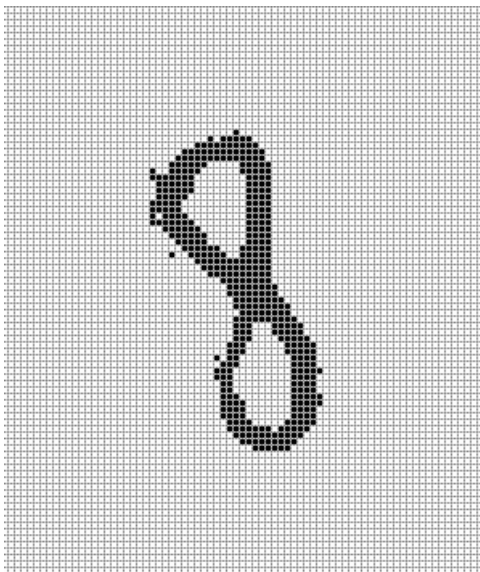


Kerettörlés után

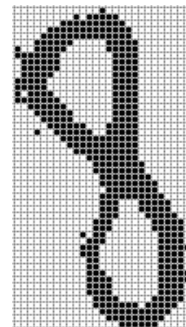
4.4.2.4.4. Szegmentálás

Az így kapott kép már keret nélküli és számunkra már csak a hasznos információt tartalmazza, de ennek ellenére még tovább kell pontosítanunk. Le kell vágnunk a képen mind a négy oldalról (alsó és felső vízszintes, valamint bal és jobb függőleges) azokat a sorokat, melyeken végighaladva megállapítjuk, hogy nem tartalmaznak fekete pontokat. Ennek eredményeképpen már csak a tényleges hasznos tartalomra fókuszáltunk.

Megjegyezném, hogy itt be kellett építenem egy küszöböt, ami egy minimális képrészletet mindenképpen meghagy a szegmentálandó képből abban az esetben, ha az minden pontjában fehér színű. Ez akkor fordulhat elő, ha az adott adatmező checkbox típusú és épp nincs kitöltve. Ellenkező esetben a szegmentálás hatására a képünk „elfogyna”.



Szegmentálás előtt



Szegmentálás után

4.4.2.4.5. A tényleges felismerés

Ezen a ponton a kép feldolgozása két külön úton halad tovább, attól függően hogy csak bejelölést vagy épp számjegyet kell felismernünk. Az egyszerűbb a bejelölés felismerése.

4.4.2.4.5.1. Checkbox felismerése

Adott a szegmentált kép, amely vagy minimális méretű teljesen fehér kép vagy egy szegmentált ábra, mely a téglalapba tett jelölést tartalmazza. A kép pontjain végighaladva megszámláljuk a fekete képpontok számát, majd ebből meghatározzuk annak arányát a képen. Amennyiben ez egy meghatározott értéken felül van, az adott mezőt bejelöltnek minősítjük, alatta pedig bejelöletlennek.

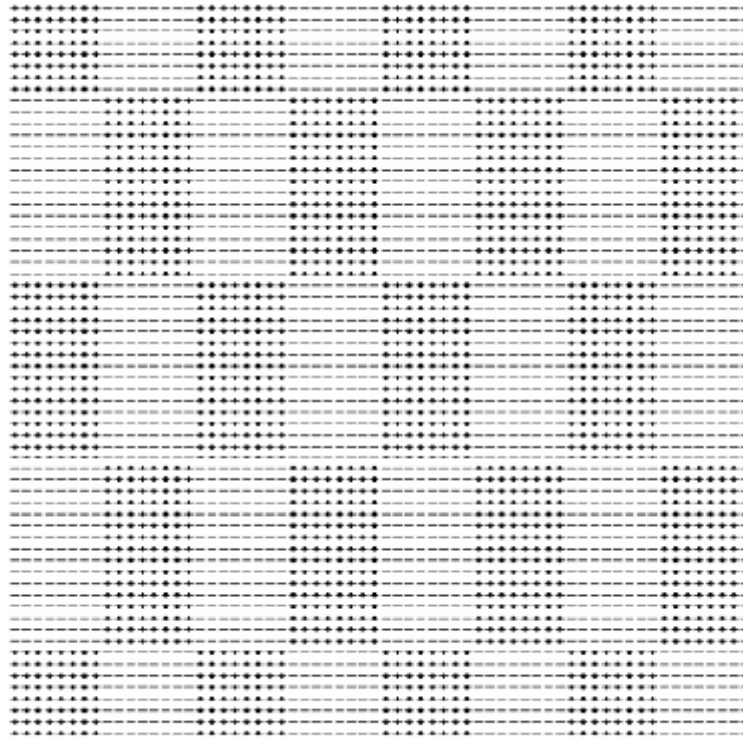
Megjegyezni kívánom, hogy a checkbox felismerésénél a program nem vizsgálja a téglalapba tett jel alakját, csupán a bejelöléssel járó sötét képpontok arányából következtet.

4.4.2.4.5.2. Számmező felismerése Walsh-táblázat segítségével

Számmezők felismerése már sokkal bonyolultabb és egy jóval kifinomultabb eljárást igényel. Ebben az esetben a felismeréshez egy Walsh-táblázatot hívjuk segítségül. A kétdimenziós diszkrét Walsh-transzformáció $N = 2^n$ esetén a következő formában határozható meg.

$$W(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u) + b_i(y)b_{n-1-i}(v)}$$

ahol $b_k(z)$ a z bináris reprezentációjának k -adik bitje.



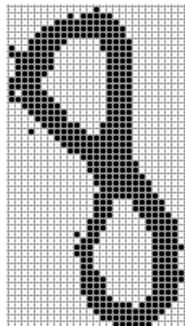
A mátrix $u = 6$, $v = 4$ esetén, ahol $a +$ jelöli $a + 1$ -et és $-$ jelöli $a - 1$ -et

Jelen algoritmus esetén $n = 6$, ennek következtében készítettünk 8×8 -as, azaz 64 darab 64×64 -es mátrixot a következő módon. Minden egyes (u, v) értékpárhoz egy 64×64 -es mátrix tartozik, amelynek elemei vagy -1 -et vagy $+1$ -et tartalmaznak, így letárolásához egy bit is elég. Ezen mátrixok kiszámítása időigényes, viszont egyszeri kiszámítás után a memóriában tárolásra kerül, így elkerüljük azt, hogy újra meg kelljen határozni őket.

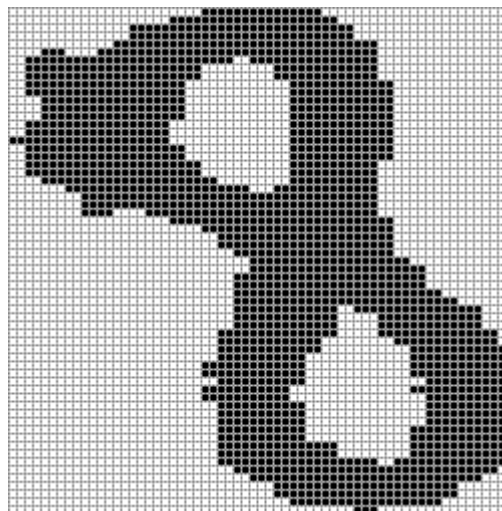
Sajátságvektor

A szegmentált képhez a Walsh-táblázatok segítségével egy 64 számból álló vektort rendelünk, ez lesz a kép sajátosságvektora. Ennek előállítása a következő.

Először a szegmentált képet 64×64 méretűre konvertáljuk.



Konvertálás előtt



Konvertálás után

Adott (u, v) párhoz rendelt Walsh-mátrixot pontonként összeszorozzuk az átméretezett kép mátrixos alakjával úgy, hogy a kép előtérpontjait 1-nek, a háttérpontjait 0-nak tekintjük. Így egy olyan mátrixot kapunk, ahol az egyes elemek -1-et, 0-át vagy 1-et vehetnek fel értékül. Ezen mátrix elemeit összeadva egy egész számot kapunk -4096 és +4096 között. Ezen (u, v) párokhoz tartozó mátrixokhoz tartozó összegeket minden lehetséges u -ra és v -re oszlopfolytonos felsorolással egy 64 számból álló vektorba tároljuk. Ez lesz a kép sajátságvektora.

Számjegysablonok sajátságvektorai

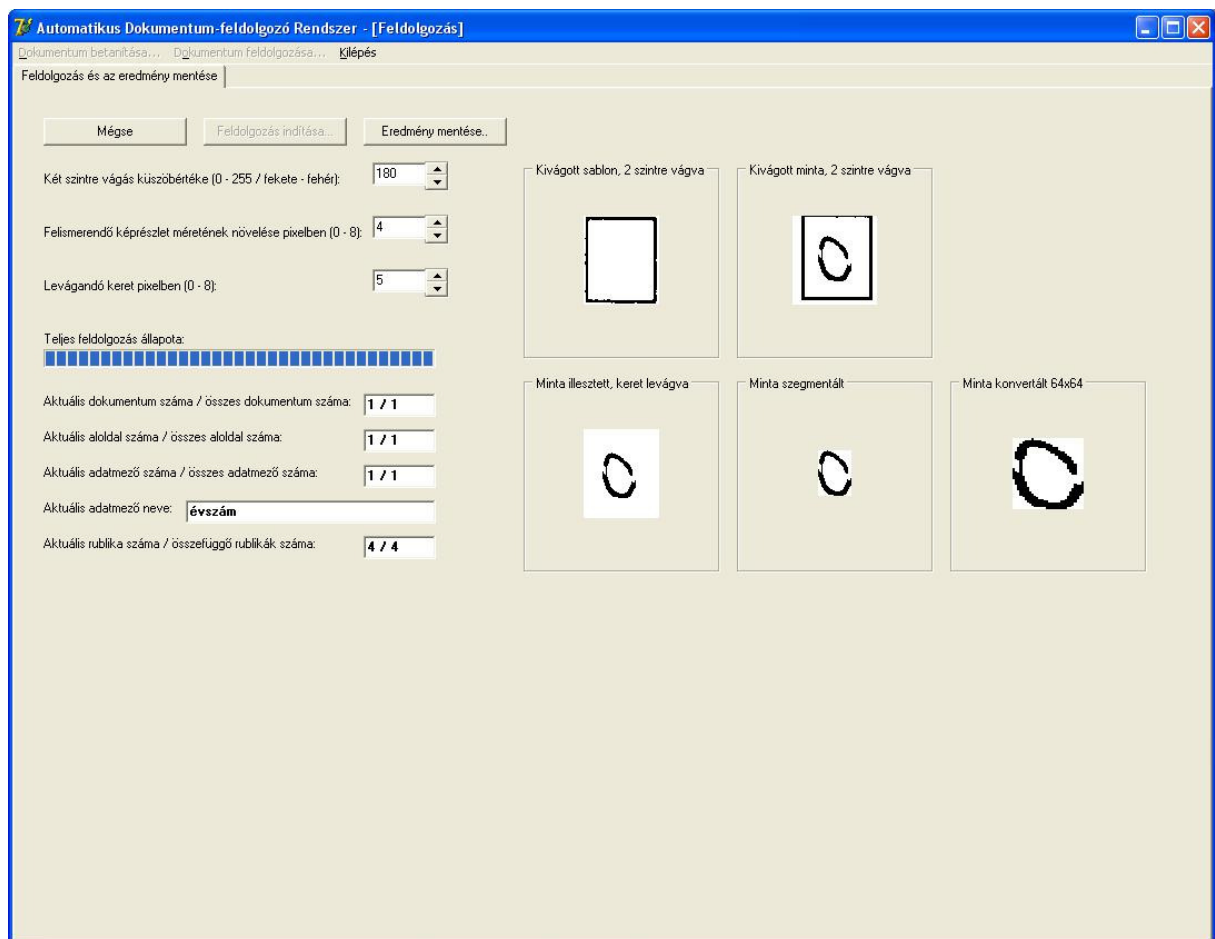
Ugyanilyen felépítésű sajátságvektorokkal már rendelkezünk a programban. Ezeket a program részeként meglévő „szamok.wls” fájlban tároljuk. Itt vannak összegyűjtve a számjegyek (0-tól 9-ig) sablonjai, melyet különböző emberektől bekért kézírásminták alapján készítettem. Minél több sablonnal dolgozunk, annál valószínűbb, hogy a legjobban megfelelőt tudjuk majd kiválasztani. Egy ilyen külső fájl használata abból a szempontból célszerű, hogy a program ismerete a későbbiekben könnyen bővíthető.

Tehát létezik egy, az adott képrészlet alapján előállított sajátságvektorunk, valamint az egyes számjegyekhez tartozó, kézírásminták alapján előállt sajátságvektoraink. Felismerés során megnézzük, hogy a képhez számított vektornak melyik etalonvektortól való távolsága a legkisebb. Ehhez a vektor elemein páronként végighaladunk, az egyes párok különbségének

abszolút értékét összeadjuk. Amelyik etalonvektornál ez a szám a legkisebb, azon etalonvektorhoz rendelt számjegyünk lesz a felismert számjegy.

A felismert számjegyeket az ADF fájlban rögzített összefüggő rubrikák száma alapján összefűzzük és előállítjuk belőle a tényleges, több számjegyből álló számot.

A program a feldolgozás során végig mutatja hogy éppen melyik a feldolgozás alatt álló adatmező, illetve a két szintre vágott, illesztett, szegmentált és konvertált képeket is megjeleníti.



4.4.2.5. Az eredmény mentése

A feldolgozás végeztével annak eredményét szöveges fájlba menthetjük. A szöveges állományban összesítésre kerül, hogy mely adatmezőknél mely rubrikák lettek bepipálva és hányszor, valamint számmező esetén milyen értékek fordultak elő és mennyiszer. Checkbox esetén a rubrikákat a rendszer a felvitel sorrendjében sorszámozza.

Példánknál maradva ez a következő lehet. A fájl tartalmát két hasábra törtem.

```
[1]
Mező neve=Születési évszám
1980=2
1978=1
1965=1
```

```
[2]
Mező neve=Születési hónap
01=1
04=1
05=1
06=1
```

```
[3]
Mező neve=Születési nap
12=1
19=1
21=2
```

```
[4]
Mező neve=Neme
rublika01=2
rublika02=2
```

```
[5]
Mező neve=Eltöltött évek száma
3=2
1=1
2=1
```

```
[6]
Mező neve=Havi bruttó kereset
80000=1
125000=1
139000=1
180000=1
```

```
[7]
Mező neve=Felsőfokú végzettségek
száma
0=2
1=1
2=1
```

5. Összegzés, konklúzió, további tervek

Mint azt említettem, dolgozatomban egyben egy kisebb kutatás része, mely közel 4 hónapos munka eredményét tükrözi. Bár az empirikus kísérletek még hátravannak, a fejlesztés során, s eddig lefolytatott tesztek elvégzése, valamint a beállítások finomhangolása után a tapasztalat azt mutatja, hogy programom a checkboxok jelölésének állapotát közel 90%-os, míg a számjegyeket körülbelül 70%-os biztonsággal ismerte fel. Természetesen a munka ezzel nem fejeződött be, folyamatban vannak még olyan algoritmusok és megoldások implementálása, melyek ezen hatékonyságot tovább növelik. Számmező felismerés esetén bővíthető a felvitt sablonok száma, ezáltal még több kézírásfajtát ismerhetünk fel, illetve dolgozom ezen sablonok sajátosságvektoraiból egy általános, átlagolt minta létrehozásán is.

További tervek között szerepel még a gyorsaság növelése, ez az adott matematikai algoritmusok optimalizálásával érhető el. Az eredmény megjelenítése szempontjából beépítésre kerül egy HTML alapú eredményexportálás, mellyel akár az interneten egyszerűen publikálható a feldolgozás végeredménye. Beépíteni kívánom még a betanított dokumentumtípusok adatainak betöltésének, szerkesztésének lehetőségét vagy akár a kitöltött dokumentumok szkennerről történő beolvasását.

A programban végig törekedtem a felhasználói figyelmetlenségből származó hibák kiküszöbölésére, a program biztonságosabbá tételére, ezért már most is számos olyan elem került beépítésre, mely nem engedi a programot helytelen működésre bírni.

Mindezekből látszik, hogy egy olyan területről van szó, mely folyamatos fejlesztést igényel, mindig és mindig lehet új ötletekkel előállni, azokat tesztelni, sikeresség esetén a programba beépíteni. A lehetőségek tárháza gyakorlatilag végtelen.

6. Irodalomjegyzék

Papír alapú források:

Marco Cantù, *Delphi 7 mesteri szinten*, Kiskapu Kiadó (2003)

Internetes források:

<http://www.prog.hu/tudastar/?op=view> – *fórum és tudásbázis*, (2007)

<http://www.torry.net/pages.php?id=344> – *Torry's Delphi Pages*, Code Samples, (2007)

Elektronikus jegyzetek:

Dr. Fazekas Attila, *Képfeldolgozás matematikai alapjai* (2005)

Iszály György Barna, *Multimédiás adatformátumok* (2005)

7. Köszönetnyilvánítás

Ezúton mondok köszönetet mindazoknak, akik elősegítették diplomamunkám elkészülését.

Dr. Fazekas Attilának, aki személyes segítségével, ötleteivel, konzulensi tevékenységével nagyban hozzájárult diplomamunkám elkészültéhez.

Nagy András hallgatónak, aki hasonló területen végzett kutatásainak eredményével segítette munkámat.